UNIVERSITÉ DE LORRAINE | BIBLIOTHÈQUES UNIVERSITAIRES

AVERTISSEMENT
DROIT D'AUTEUR

Ce document est le fruit d'un long travail approuvé par le jury de soutenance et mis à disposition de l'ensemble de la communauté universitaire élargie.

Il est soumis à la propriété intellectuelle de l'auteur. Ceci implique une obligation de citation et de référencement lors de l'utilisation de ce document. Toute contrefaçon, plagiat ou reproduction illicite encourt une poursuite pénale.

Contact bibliothèque : ddoc-theses-contact@univ-lorraine.fr
(Cette adresse ne permet pas de contacter les auteurs)

Code de la Propriété Intellectuelle. articles L 122. 4
Code de la Propriété Intellectuelle. articles L 335.2- L 335.10

UNIVERSITÉ DE LORRAINE
34 COURS LÉOPOLD — CS 25233
54052 NANCY CEDEX
WWW.UNIV-LORRAINE.FR

Ecole doctorale BioSE

Thèse

Pour l'obtention du titre de

Docteur de l'Université de Lorraine

Mention : « Sciences de la Vie et de la Santé»

Titre : Intelligence artificielle explicable pour la fiabilité des algorithmes d'apprentissage automatique, l'analyse et la découverte dans les données omiques.

Par Ghislain FIEVET

Sous la direction du Pr. Julien BROSÉUS et du Dr. Sébastien HERGALANT

Présentée et soutenue publiquement le 2025/11/20

Membres du Jury :

Président de jury :	Daniel Gautheret	Pr HDR, I2BC, Université de Paris Saclay
Rapporteurs :	Anaïs Bardet	Dr HDR, igbmc, Université de Strasbourg
	Mohamed Elati	Pr HDR, Canther, Université de Lille
Examinateurs :	Sophie Mézières	Dr MdC, IECL , Université de Lorraine
	Julien Broséus	Pr HDR, NGERE, Université de Lorraine (Directeur de thèse)
	Sébastien Hergalant	Dr, NGERE, Université de Lorraine (Co–directeur de thèse)

UMRS INSERM 1256 NGERE : Faculté de Médecine, Maïeutique et métiers de la Santé
9 Av. de la Forêt de Haye, 54500 Vandœuvre-lès-Nancy

Résumé en français

Cette thèse explore l'apport de l'intelligence artificielle explicable (XAI, *eXplainable Artificial Intelligence*) dans l'analyse et la découverte à partir des données omiques, domaine clé de la bioinformatique moderne. Les données omiques regroupent la génomique, la transcriptomique (y compris le *single-cell* RNA-seq), la protéomique, l'épigénomique et la métabolomique. Leur intégration représente un défi méthodologique, notamment face à la complexité et au volume croissant des données produites.

Le premier axe de ce travail porte sur la robustesse des classifieurs de types cellulaires issus de données de séquençage d'ARN en cellule unique (scRNA-seq) face aux attaques adverses. Ces attaques consistent à modifier légèrement les données d'entrée pour tromper les modèles, révélant ainsi leurs faiblesses structurelles. Nous avons développé *adverSCarial*, un package *R* disponible sur *Bioconductor*, qui propose plusieurs algorithmes d'attaque adaptés au scRNA-seq. L'évaluation menée sur différents jeux de données et modèles de classification a mis en évidence des vulnérabilités notables, ouvrant des pistes pour renforcer la fiabilité des analyses.

Le second axe se concentre sur la capacité des méthodes XAI à dépasser les limites des tests statistiques classiques dans la détection de signaux pertinents. Nous présentons un cadre comparatif de méthodes de sélection de variables fondées sur l'importance des caractéristiques (*SHAP*, *LIME*, Gini, permutations, etc.), que nous avons converties en p-values afin de les rendre comparables aux approches statistiques traditionnelles. Ce travail a abouti au développement du package *XAItest*, qui permet de tester et comparer ces méthodes sur différents scénarios simulés et réels.

Nous avons ensuite appliqué ce framework à grande échelle sur des données scRNA-seq issues de la plateforme *CZ CELLxGENE*, démontrant la pertinence des approches XAI pour découvrir des gènes signatures dans des contextes où les méthodes statistiques échouent. Enfin, nous illustrons l'utilisation de ces outils dans une étude sur les lymphomes, où XAI complète ou remplace avantageusement les approches classiques pour identifier des biomarqueurs.

En conclusion, cette thèse propose des outils concrets et validés pour fiabiliser l'analyse des données omiques par apprentissage automatique, tout en montrant que l'explicabilité n'est pas seulement un moyen de comprendre les modèles, mais aussi un levier de découverte scientifique.

English abstract

This dissertation investigates the contribution of explainable artificial intelligence (XAI) to the reliability, analysis, and discovery processes in omics data, a central field in modern bioinformatics. Omics encompass genomics, transcriptomics (including single-cell RNA-seq), proteomics, epigenomics, and metabolomics. The integration of these layers remains a methodological challenge due to their increasing complexity and volume.

The first research axis focuses on the robustness of cell type classifiers derived from single-cell RNA sequencing (scRNA-seq) data against adversarial attacks. Such attacks introduce subtle perturbations to the input, deceiving models and exposing their structural weaknesses. We developed *adverSCarial*, an R package available on *Bioconductor*, implementing several attack algorithms tailored to scRNA-seq. Evaluation across multiple datasets and classifiers revealed significant vulnerabilities, highlighting the need to improve reliability in single-cell analysis pipelines.

The second axis addresses the ability of XAI methods to overcome the limitations of conventional statistical tests (t-test, Wilcoxon) in detecting relevant biological signals. We designed a comparative framework for feature selection methods (SHAP, LIME, Gini, permutation tests, etc.), adapting them into p-values to ensure compatibility with statistical approaches. This work led to the development of the *XAItest* R package, enabling systematic testing and benchmarking of XAI-based feature importance methods across simulated and real scenarios.

We further applied this framework at scale on scRNA-seq datasets from the *CZ CELLxGENE* platform, demonstrating the added value of XAI in identifying gene signatures where traditional statistics fall short. Finally, we showcased the relevance of these methods in a lymphoma study, where XAI complemented or outperformed conventional strategies in biomarker discovery.

In conclusion, this thesis provides concrete and validated tools to enhance the robustness of machine learning applied to omics data. Beyond interpretability, XAI emerges as a powerful driver of scientific discovery, bridging the gap between algorithmic performance and biological insight.

Table des matières

Table des matières

Liste des abréviations....15

Introduction....17

Intelligence artificielle en médecine....17

Recherche biomédicale....17

Utilisation clinique....18

Transformers et *Large Language Models* (LLM)....19

Apprentissage automatique ou *machine learning*....20

Intelligence artificielle explicable (XAI)....25

Bases de biologie moléculaire....27

Dogme central....28

Régulation génique....29

Cancer....29

Données omiques....30

Génomique....30

Transcriptomique : des puces au *single-cell*....31

Bulk....31

Single-cell, scRNA-seq....32

Protéomique....34

Épigénomique....35

Méthylome....35
Organisation et accessibilité de la chromatine....36
Métabolomique....37
Vers une biologie multi-omique intégrée....38
Technologies émergentes et futur des omiques....38
Axe 1 : attaques adverses sur scRNA-seq....40
Attaques adverses....40
Définition et exemples....40
FGSM (*Fast Gradient Sign Method*) et PGD (*Projected Gradient Descent*)....44
Adversarial training....45
Peu d'attaques adverses adaptées au scRNA-seq....46
Attaques adverses sur données scRNA-seq....46
Single-gene attack....47
Max-change....49
Cluster Gradient Based (CGD)....52
Modifications pour *single-gene* et *max-change*....53
Package *R Bioconductor adverSCarial*....54
Étude générale sur plusieurs classifieurs et plusieurs jeux de données....57
Méthode....57
Classifieurs scRNA-seq....57
Jeux de données scRNA-seq....59
Étude préliminaire sur *PBMC3k*....61
Étude sur 5 classifieurs et 4 jeux de données....63
Axe 2 : Découverte grâce au XAI....73
Limites des tests statistiques....73
Problèmes de classification....73
t-test et test de Wilcoxon....73
Cas invisibles pour les tests statistiques....75
Le F-test pour les distributions bimodales et les différences de variances....78

Problèmes de régression....78
Régressions linéaires et corrélations....78
Cas invisibles pour les tests statistiques....79
Machine learning et *feature importance*....82
SHAP (*SHapley Additive exPlanations*)....83
TreeSHAP....84
Importance de Gini....84
TreeSHAP vs Importance de Gini....84
LIME (*Local Interpretable Model-agnostic Explanations*)....85
Permutations....86
Score d'Olden....87
Conversion entre *feature importances* et *p-values*....88
PIMP (*Permutation Importance p-values*)....88
1Probe/mProbes....88
simThresh (seuil simulé)....89
Scénarios à détecter....95
Package *R Bioconductor XAItest*....103
Framework de comparaison de méthodes de découverte....112
Temps de calcul....115
Efficacité des différentes combinaisons....117
Application sur données scRNA-seq de *CZ CELLxGENE*....126
CZ CELLxGENE....126
Étude sur périmètre réduit....126
Combinaison de méthodes choisie....126
Résultat sur périmètre réduit....127
Étude sur périmètre étendu....130
Résultats sur *low_diff*....132
Discussion et perspectives....144
Conclusion....155

Références..... 156
Annexes..... 167

Liste des Figures

Figure 1: Exemple d'arbre de décision. Réf. : https://commons.wikimedia.org/wiki/File:CART_tree_titanic_survivors.png.....21

Figure 2: Illustration de SVM à noyau linéaire. Réf. : https://en.wikipedia.org/wiki/Support_vector_machine#/media/File:SVM_margin.png. 23

Figure 3: *LIME* permet de savoir quelle partie de l'image est utilisée par le classifieur, extrait de Tulio Ribeiro et al. 2016 [51].....27

Figure 4: Dogme central. Réf. : https://commons.wikimedia.org/wiki/File:Dogme-central-BioMol.svg..... 29

Figure 5: Séquenceur Illumina MiSeq. Réf. : https://commons.wikimedia.org/wiki/File:Illumina_MiSeq_sequencer_2.jpg.....31

Figure 6: Illustration de la préparation d'échantillon FFPE pour scRNA-seq spatial issue d'un document de protocole du site STAR Protocols. [149].....33

Figure 7: Exemple de résultat de Western Blot pour mesurer le niveau d'expression des protéines Gemin3, SMN et Tubuline. Réf. : https://commons.wikimedia.org/wiki/File:Western_Blot_2022-10-13_06_41_26.png.....35

Figure 8: Vue d'ensemble des différentes couches de données omiques et des outils associés..... 38

Figure 9: Exemple d'attaque adverse sur image de panda sur le classifieur GoogLeNet, extrait de Goodfellow et al. (2015) [100].....40

Figure 10: Des images générées à partir d'un bruit gaussien puis modifiées en direction de la classe "avion", parviennent à tromper un classifieur entraîné sur le jeu de données CIFAR-10, extrait de Goodfellow et al. 2015 [100].....41

Figure 11: a) Radiographie thoracique d'origine, non modifiée. b) Bruit adversarial ajouté c) Radiographie modifiée (image originale + bruit), extrait de Han et al. 2021 [105]..... 41

Figure 12: Adversarial attacks sur lames histologiques perturbant la classification des échantillons. Le degré de confiance du modèle pour chaque prédiction est indiqué sur chaque image, extrait de Ghaffari Laleh et al 2022 [106].....42

Figure 13: Exemple d'attaque adverse sur contenu textuel, le remplacement de mots par un synonyme fait passer le texte d'une prédiction « negative » à « positive », extrait de Jin et al. 2020 [108].....43

Figure 14: Exemple d'attaque adverse sur ECG avec PGD, extrait de Han et al. 2021 [109] 43

Figure 15: Présentation des fonctionnalités d'adverSCarial. Exécution des algorithmes d'attaque single-gene, max-change et CGD (descente de gradient basée sur les clusters). Représentations UMAP du jeu de données *PBMC3k* annoté avec scType, avant et après les attaques single-gene, max-change ou CGD ciblant le cluster de cellules B naïves. Extrait de Fievet et al. 2025 [120]..... 54

Figure 16: Vulnérabilité de cinq classifieurs scRNA-seq face aux attaques adverses single-gene et max-change. Cinq types de modifications sont appliqués, un par un, sur les neuf types cellulaires du jeu de test. (A) Heatmaps du nombre d'attaques single-gene réussies (B) Heatmap montrant la longueur de la signature des attaques max-change. Pour les attaques single-gene, un nombre supérieur à 0 indique une vulnérabilité du classifieur. Pour les attaques max-change, un classifieur est considéré comme robuste si le nombre est élevé..... 62

Figure 17: Boxplots résumant l'impact de chaque type de modification, en cumulant les résultats pour chaque type cellulaire, et ce pour les deux types d'attaques, classé par classifieur. Les données sont représentées en échelle logarithmique (log10)..... 63

Figure 18: Visibilité des changements après attaque adverse. Les classifieurs sont entraînés et testés avec les training et test set respectivement. Au-dessus de chaque heatmap, les modifications appliquées sont indiquées par des traits rouges. (A) heatmaps avec clustering hiérarchique et représentations UMAP après classification et annotation avec scType, avant attaque (en haut), après une attaque single-gene (en bas à gauche), après une attaque max-change (en bas à droite), ciblant les lymphocytes B du jeu de données *PBMC3k*..... 65

Figure 19: Heatmaps montrant les gènes les plus discriminants, avant et après différentes attaques single-gene, appliquées à plusieurs classifieurs scRNA-seq sur *LIVER10k*..... 66

Figure 20: Heatmaps montrant les gènes les plus discriminants, avant et après différentes attaques max-change, appliquées à plusieurs classifieurs scRNA-seq sur *AXILLA10k*..... 66

Figure 21: (A) Heatmaps montrant les gènes les plus discriminants, avant et après des attaques CGD de puissance croissante, paramètres α et ε, sur scAnnotatR, sur *KIDNEY10k* (B) Idem, après CGD de niveaux comparables, appliquées à différents classifieurs, sur *KIDNEY10k*..... 67

Figure 22: Performance et évaluation de CGD. L'algorithme CGD a été appliqué aux quatre classifieurs compatibles : scType, scAnnotatR, scRF et scMLP (A) Nombre de gènes à tester et/ou modifier pour modifier la classification d'un cluster cellulaire, en fonction de paramètres α et ε égaux (B) Même analyse, mais cette fois : à gauche => variation de ε quand α = 0, à droite : variation de α quand ε = 0..... 68

Figure 23: Précision des prédictions en fonction du nombre de gènes testés, avec différents jeux de valeurs pour ε et α, sur différents jeux de données..... 68

Figure 24: (A) Courbes ROC, et (B) Scores F1 correspondants, selon différentes valeurs de ε, α et du nombre de gènes testés, sur différents jeux de données. FPR : taux de faux positifs, TPR : taux de vrais positifs.....69

Figure 25: Cas valides et non valides pour l'utilisation du *t-test* (Annexe CONTENT4PHD : 0001_*R*_ttest_*vs*_wc.ipynb, im/figs/0001_ttest_3picInRow_hypothesesVerifies.png).....74

Figure 26: Différence des moyennes, *t-test vs* test de Wilcoxon (Annexe CONTENT4PHD : 0001_*R*_ttest_*vs*_wc.ipynb, im/figs/0001_ttest_4picInRow_comparaison_ttest_*vs*_wilcox.png).....75

Figure 27: Distribution gaussienne bimodale (Annexe CONTENT4PHD : 0002_*R*_invisible2statTests.ipynb, im/figs/0002_distribution_bimodale.png).....76

Figure 28: Distributions gaussiennes de variances différentes (Annexe CONTENT4PHD : 0002_*R*_invisible2statTests.ipynb, im/figs/0002_diff_variance.png).....76

Figure 29: Illustration des interactions entre variables (Annexe CONTENT4PHD : 0002_*R*_invisible2statTests.ipynb, im/figs/0002_interaction_geneX_geneY.png).....77

Figure 30: Illustration des limites du F-test (Annexe CONTENT4PHD : 0002_*R*_invisible2statTests.ipynb, im/figs/0002_ftest_2x2.png).....78

Figure 31: Distribution gaussienne pour norm_noise01 et parabole construite à partir de norm_noise01 en ordonnée, voir détails de construction dans le paragraphe « scénarios à détecter ». (Annexe CONTENT4PHD 0008_*R*_invisible_regression.ipynb, im/figs/0008_parabole.png).....80

Figure 32: Calcul d'une régression linéaire et d'une corrélation de Pearson sur le jeu de données "parabole" (Annexe CONTENT4PHD 0008_*R*_invisible_regression.ipynb, im/figs/0008_parabole_regr_correl.png).....80

Figure 33: Distribution uniforme pour unif_noise01 et sinusoïde à amplitude croissante construite à partir de unif_noise01 en ordonnée, voir détails de construction dans le paragraphe « scénarios à détecter ». (Annexe CONTENT4PHD 0008_*R*_invisible_regression.ipynb, im/figs/0008_sinus.png).....81

Figure 34: Calcul d'une régression linéaire et d'une corrélation de Pearson sur le jeu de données "sinuosïde à amplitude croissante" (Annexe CONTENT4PHD 0008_*R*_invisible_regression.ipynb, im/figs/0008_sinus_regr_correl.png).....82

Figure 35: Workflows basés sur les *p-values* ou sur les *feature importance*s (Annexe CONTENT4PHD im/figs/dr_0001_pval *vs feature* imp.png).....83

Figure 36: Importance de Gini *vs SHAP*/TreeSHAP: (A) nuage de points 3D des échantillons A et B (B) règles de l'arbre de décision (C, D) importance de Gini et *feature importance* de *SHAP*/TreeSHAP (Annexe CONTENT4PHD im/figs/0003_2x2_gini_*vs*_shap.png).....85

Figure 37: Jeu de données composé de 50 points de catégorie A, 50 points de catégorie B, 100 variables gaussiennes X1, X2, X3, etc allant du plus différenciant au moins différenciant. (A) *scatter plots* illustrant la progressivité de la différenciation au fil des variables (B) classification par *random forest* et calcul des *feature importance*s selon plusieurs méthodes (C) corrélations entre les différentes méthodes. (Annexe CONTENT4PHD im/figs/0004_fullPanel_shap_gini_lime_permutation.png).....87

Figure 38: Exemples des valeurs des *p-values* à chaque itération de *simThresh*. Le jeu de données utilisé pour la classification correspond au scénario n°1, et celui pour la régression au scénario n°5 (voir paragraphe « Scénarios à détecter »). (Annexe CONTENT4PHD 0006_*R*_create_jeu de données_scenariosX10.ipynb, im/figs/0006_pvalues_iter_distrib.png).....94

Figure 39: Distribution empirique des *p-values* (–log10) obtenues sur 100 *t-test*s entre deux groupes gaussiens de taille 50, avec un décalage de moyenne de 0.3. (Annexe CONTENT4PHD im/figs/0010_*R*_100_gaussian_tirage.png).....94

Figure 40: Exemples où la variable cible a une même moyenne dans les deux classes, mais une variance différente. On observe la diversité des résultats de tirages différents d'une même distribution aléatoire. (Annexe CONTENT4PHD 0006_*R*_create_jeu de données_scenariosX10.ipynb, im/figs/0006_scenario1_plotsX10.png).....95

Figure 41: Diversité des résultats de tirages différents d'une même distribution aléatoire pour le cas des distributions bimodales. (Annexe CONTENT4PHD 0006_*R*_create_jeu de données_scenariosX10.ipynb, im/figs/0006_scenario2_plotsX10.png).....96

Figure 42: Diversité des résultats de tirages différents d'une même distribution aléatoire dans le cas d’une relation simple de type XOR entre deux variables. (Annexe CONTENT4PHD 0006_*R*_create_jeu de données_scenariosX10.ipynb, im/figs/0006_scenario3_plotsX10.png).....98

Figure 43: Diversité des résultats de tirages différents de mêmes distributions aléatoires pour le cas des cercles concentriques. (Annexe CONTENT4PHD 0006_*R*_create_jeu de données_scenariosX10.ipynb, im/figs/0006_scenario4_plotsX10.png).....99

Figure 44: Fonction *R* de génération de la parabole. (Annexe CONTENT4PHD 0006_*R*_create_jeu de données_scenariosX10.ipynb).....100

Figure 45: Diversité des résultats de tirages différents de mêmes distributions aléatoires pour le cas des paraboles. (Annexe CONTENT4PHD 0006_*R*_create_jeu de données_scenariosX10.ipynb, im/figs/0006_scenario5_plotsX10.png).....100

Figure 46: Fonction *R* de génération de la sinusoïde à amplitude croissante. (Annexe CONTENT4PHD 0006_*R*_create_jeu de données_scenariosX10.ipynb).....101

Figure 47: Diversité des résultats de tirages différents de mêmes distributions aléatoires pour le cas des sinusoïdes à amplitude croissante. (Annexe CONTENT4PHD 0006_*R*_create_jeu de données_scenariosX10.ipynb, im/figs/0006_scenario6_plotsX10.png)..... 101

Figure 48: Récapitulatif des 6 scénarios simulés : variables bruitées, différences de distributions entre les classes A et B, et configurations spécifiques illustrant des cas de classification ou de régression complexes. (Annexe CONTENT4PHD 0006_*R*_create_jeu de données_scenariosX10.ipynb, im/figs/0006_panel_scenarios.png)..... 102

Figure 49: Scatter *plots* du fichier d'exemple "extdata/dfclassifieur.txt". Réf. : https://www.bioconductor.org/packages/release/bioc/vignettes/XAItest/inst/doc/XAItest.html..... 104

Figure 50: Résultat de la fonction mapPvalImportance(). Réf. : https://www.bioconductor.org/packages/release/bioc/vignettes/XAItest/inst/doc/XAItest.html..... 106

Figure 51: Illustration de l'utilisation de la fonction plotModel. Réf. : https://www.bioconductor.org/packages/release/bioc/vignettes/XAItest/inst/doc/XAItest.html..... 107

Figure 52: Scatter *plot* représentant la variable cible et la variable d'intérêt du jeu de données dataset_regress. Réf. : https://www.bioconductor.org/packages/release/bioc/vignettes/XAItest/inst/doc/XAItest.html..... 108

Figure 53: Résultat de la fonction mapPvalImportance(). Réf. : https://www.bioconductor.org/packages/release/bioc/vignettes/XAItest/inst/doc/XAItest.html..... 109

Figure 54: Résultats de plotModel pour la régression linéaire et RF. Réf. : https://www.bioconductor.org/packages/release/bioc/vignettes/XAItest/inst/doc/XAItest.html..... 110

Figure 55: Workflow de comparaison de méthodes XAI de découverte. (Annexe CONTENT4PHD im/figs/gdrive_XAI scenarios *workflow*.png)..... 115

Figure 56: (gauche) trois exemples de combinaison de méthodes et leurs temps de calcul pour chacun des 10 réplicats pour chaque scénario, (droite) écart-type/moyenne pour chaque combinaison testée. (Annexe XAI_SIMU : 0008_timesx3.ipynb, im/phd/0007_time_4boxplots.png)..... 115

Figure 57: Boxplots des temps de calcul pour différentes catégories de combinaisons de méthodes (Annexe XAI_SIMU : 0008_timesx3.ipynb, im/phd/0008_time_8boxplots.png) 116

Figure 58: Temps moyens, médian, Q1, Q3, min et max des temps de calcul pour différentes catégories de combinaisons de méthodes (Annexe XAI_SIMU : 0008_timesx3.ipynb)..... 116

Figure 59: Tableau de résultats pour *simThresh* (Annexe XAI_SIMU 0009_*R*_results_tableX10.ipynb, data/0009_*simThresh*_table.csv, im/phd/0009_*simThresh*_table_exc_cropped.png)..... 118

Figure 60: Tableau de résultats pour *PIMP* (Annexe XAI_SIMU 0009_*R*_results_tableX10.ipynb, data/0009_pimp_table.csv, im/phd/0009_pimp_table_exc_cropped.png)..... 119

Figure 61: Tableau de résultats pour mProbes (Annexe XAI_SIMU 0009_*R*_results_tableX10.ipynb, data/0009_probes_table.csv, im/phd/0009_probes_table_exc_cropped.png)..... 120

Figure 62: Tableau de la meilleure méthode entre *simThresh*, *PIMP* et mProbes (Annexe XAI_SIMU 0010_*R*_analysis_results_tableX10.ipynb)..... 122

Figure 63: Nombre de victoires par méthode (Annexe XAI_SIMU 0010_*R*_analysis_results_tableX10.ipynb)..... 122

Figure 64: Nombre moyen de détections sur les 10 réplicats de chaque scénario pour chaque combinaison (Annexe XAI_SIMU 0010_*R*_analysis_results_tableX10.ipynb)..... 123

Figure 65: Nombre de succès par combinaison pour les 10 réplicats du scénario 3 (Annexe XAI_SIMU 0010_*R*_analysis_results_tableX10.ipynb)..... 124

Figure 66: Nombre de détections des variables d'intérêt pour les combinaisons de méthodes DT/Gini/*simThresh* (Annexe XAI_SIMU 0010_*R*_analysis_results_tableX10.ipynb)..... 125

Figure 67: (gauche) Nombre de gènes restant après le filtre sur la *p-value* à 0.2 pour le *t-test* et le test de Wilcoxon sur les 91 jeux de données (droite) Accuracy des DT de profondeur max=2 entraînés sur les gènes ayant passé le filtre. (Annexe *XAI_CZ_CELLxGENE* 0020_*R_plot*_kept_genes_*accuracies*.ipynb)..... 127

Figure 68: (gauche) Exemple d'arbre de décision d'*accuracy* faible de 0.57 (droite) Exemple d'arbre de décision avec une forte *accuracy* de 0.96..... 128

Figure 69: Scatter *plots* des deux gènes avec la plus grande importance de Gini des 9 arbres de décision avec la plus forte *accuracy* parmi les 91 arbres (Annexe *XAI_CZ_CELLxGENE* 0020_*R_plot*_kept_genes_*accuracies*.ipynb, im/phd/0020_top9_gene_expression_*scatter*.png)..... 129

Figure 70: Création des données pour l'étude sur périmètre étendu..... 131

Figure 71: Illustration du biais de normalisation sur le sous-ensemble high_diff..... 132

Figure 72: Accuracy des DT de profondeur max=2 entraînés sur les gènes ayant passé les filtres sur l'ensemble low_diff. (0055_XAIframework/0023_*R_plot*_lowDiff_*accuracies*.ipynb).....133

Figure 73: Scatter *plots* 1 à 50 des DT avec la plus grande *accuracy*, sur les deux gènes avec le plus grand score de Gini, couleur par type cellulaire. (0055_XAIframework/0027_*R*_low_diff_stats_100grid_bigDots.ipynb, 0027_ids_top100_grid1_10x5_bigDots.png).....134

Figure 74: Scatter *plots* 51 à 100 des DT avec plus la grande *accuracy*, sur les deux gènes avec le plus grand score de Gini, couleur par type cellulaire. (0055_XAIframework/0027_*R*_low_diff_stats_100grid_bigDots.ipynb, 0027_ids_top100_grid1_10x5_bigDots.png).....135

Figure 75: Scatter *plots* 1 à 50 des DT avec la plus grande *accuracy*, sur les deux gènes avec le plus grand score de Gini, couleur par nombre de *counts*. (0055_XAIframework/0027_*R*_low_diff_stats_100grid_bigDots.ipynb, 0024_ids_top100_grid1_10x5_cntCol_bigDots.png).....136

Figure 76: Scatter *plots* 51 à 100 des DT avec la plus grande *accuracy*, sur les deux gènes avec le plus grand score de Gini, couleur par nombre de *counts*. (0055_XAIframework/0027_*R*_low_diff_stats_100grid_bigDots.ipynb, 0024_ids_top100_grid2_10x5_cntCol_bigDots.png).....137

Figure 77: Exemples du motif du biais visible (Annexe *XAI_CZ_CELLxGENE*, im/figs/dr_top100_effects_visible_pattern.png).....138

Figure 78: Exemples du motif de la diagonale (Annexe *XAI_CZ_CELLxGENE*, im/figs/dr_top100_effects_diag.png).....138

Figure 79: Exemples du motif des axes X et Y (Annexe *XAI_CZ_CELLxGENE*, im/figs/dr_top100_effects_axis.png).....139

Figure 80: Exemples du motif du manque de variabilité (Annexe *XAI_CZ_CELLxGENE*, im/figs/dr_top100_effects_all_red.png).....139

Figure 81: Panel de l'ID 1, *accuracy* de l'arbre de décision 0.96, illustre le scénario de différence de variance (Annexe *XAI_CZ_CELLxGENE*: im/phd/dr_app_panels_1.png).....140

Figure 82: Panel de l'ID 40, *accuracy* de l'arbre de décision 0.82, illustre le scénario de différence de variance (Annexe *XAI_CZ_CELLxGENE*: im/phd/dr_app_panels_2.png).....141

Figure 83: Panel de l'ID 5, *accuracy* de l'arbre de décision 0.89, illustre le scénario de distribution bimodale (Annexe *XAI_CZ_CELLxGENE*: im/phd/dr_app_panels_3.png).....141

Figure 84: Panel de l'ID 11, *accuracy* de l'arbre de décision 0.86, illustre le scénario de distribution bimodale (Annexe *XAI_CZ_CELLxGENE*: im/phd/dr_app_panels_4.png).....142

Figure 85: Résumé des scénarios d'appartenance du gène ayant le plus haut score de Gini pour les 41 arbres (Annexe *XAI_CZ_CELLxGENE*: im/phd/dr_var_bidistrib_other.png) 143

Figure 86: Heatmap des 20 protéines dont la *p-value* du test de Fisher sur chimiosensibles *vs* chimioréfractaires est la plus faible. (Annexe CONTENT4PHD : im/figs/0341_hm_missings_fisher_top20.png, 0341_*R*_FOR_PHD_mnars_decision_trees_RF_pretty_top5.ipynb)..... 153

Figure 87: Heatmap des 20 protéines dont la *feature importance* est la plus élevée après entraînement d'un modèle RF sur chimiosensibles *vs* chimioréfractaires. (Annexe CONTENT4PHD : im/figs/0341_hm_missings_RF_featImp_top20.png, 0341_*R*_FOR_PHD_mnars_decision_trees_RF_pretty_top5.ipynb)..... 154

Liste des abréviations

AD	*Alzheimer's Disease*
AI	*Artificial Intelligence*
CGD	*Cluster Gradient Based*
CLL	*Chronic Lymphocytic Leukemia*
DL	*Deep Learning*
DLBCL	*Diffuse Large B-Cell Lymphoma*
DMR	*Differentially Methylated Region*
DNA	*Deoxyribonucleic Acid*
DT	*Decision Tree*
ECG	ÉlectroCardioGramme
FDA	*Food and Drug Administration*
FDR	*False Discovery Rate*
FFPE	*Formalin-Fixed Paraffin-Embedded*
FGSM	*Fast Gradient Sign Method*
GC	*Germinal center-like*
GWAS	*Genome-Wide Association Studies*
LIME	*Local Interpretable Model-agnostic Explanations*
LLM	*Large Language Model*
LSTM	*Long Short-Term Memory*
MDA	*Mean Decrease in Accuracy*
MHC	*Major Histocompatibility Complex*
MLP	*Multi Layer Perceptron*
MRI	*Magnetic Resonance Imaging*
MRS	*Magnetic Resonance Spectroscopy*
MS	*Mass Spectrometry*
PGD	*Projected Gradient Descent*
PCR	*Polymerase Chain Reaction*
QC	*Quality Control*
RBF	*Radial Basis Function*
RF	*Random Forest*
RNA	*Ribonucleic Acid*

scRNA-seq	*Single Cell RNA Sequencing*
SNP	*Single Nucleotide Polymorphisms*
SVM	*Support Vector Machine*
SHAP	*SHapley Additive exPlanations*
TCR	*T-Cell Receptor*
ViT	*Vision Transformer*
XAI	*eXplainable Artificial Intelligence*

Introduction

Cette recherche se propose d'explorer le potentiel de l'intelligence artificielle explicable (*eXplainable Artificial Intelligence, XAI*) dans l'analyse de données omiques en bioinformatique. Je décrirai tout d'abord brièvement comment l'intelligence artificielle (IA) est utilisée dans la recherche biomédicale et les applications cliniques. Je détaillerai ensuite le concept d'XAI et discuterai de son rôle dans l'amélioration des algorithmes et dans la contribution à la découverte scientifique. Le premier axe de la thèse se concentre sur l'évaluation de la vulnérabilité des classifieurs de types cellulaires en séquençage ARN en cellule unique (scRNA-seq) aux attaques adverses, avec le développement d'un outil spécifique : *adverSCarial*. Le second axe explore comment les méthodes XAI peuvent accompagner les méthodes statistiques conventionnelles pour l'analyse des données omiques. Nous verrons le cadre théorique qui sous-tend l'amélioration des capacités prédictives offertes par l'XAI par rapport aux méthodes statistiques conventionnelles. Puis nous présenterons un *framework* de comparaison de méthodes de sélection des biomarqueurs, ainsi qu'une application à grande échelle sur données de scRNA-seq, et enfin un exemple d'utilisation de ces méthodes XAI en alternative aux méthodes statistiques dans le cas d'une étude sur les lymphomes.

Intelligence artificielle en médecine

Recherche biomédicale

L'intelligence artificielle est devenue un outil essentiel dans divers champs d'application, y compris en recherche biomédicale. Voici quelques exemples d'utilisation.

Structure des molécules. Le prix Nobel de chimie 2024 a été décerné à Demis Hassabis et John M. Jumper pour leurs travaux sur la prédiction de la structure des protéines avec AlphaFold [1], [2], ainsi qu'à David Baker pour ses recherches sur le design de protéines avec RFdiffusion [3] et ProteinMPNN [4]. Ces outils sont basés sur des architectures de réseaux de neurones variées. Ils permettent d'élucider *in silico* le rôle de protéines, réduisant le temps et le coût habituellement requis par les expérimentations en laboratoire. Le domaine du *protein design* a un grand potentiel pour la conception thérapeutique, mais également en biotechnologies avec notamment l'amélioration de l'efficacité des enzymes.

Études génétiques de population. Face à l'augmentation de l'incidence de la maladie d'Alzheimer (AD), la communauté scientifique intensifie les études épidémiologiques et l'utilisation des outils d'IA. La revue de Mishra & Li présente comment ces technologies facilitent les analyses génétiques de la maladie [5]. Par exemple, Wei et al. ont mis au point un algorithme de Bayes naïf pour modéliser le risque de développer AD à partir de polymorphismes nucléotidiques simples (SNPs), permettant de créer un test de dépistage de risque [6]. De leur côté, Xu et al. ont employé un modèle à vecteurs de support (SVM) qui utilise les séquences de protéines codées par les gènes associés à l'AD pour identifier les marqueurs prédictifs pertinents

[7]. Utiliser les protéines plutôt que les gènes permet d'augmenter la puissance statistique à partir des mêmes données.

Découverte de médicaments. Parmi les applications fréquemment mentionnées de l'IA on trouve la conception de nouveaux médicaments. Ekins proposait en 2016 de tirer profit de l'augmentation des bases de données de molécules, et de l'amélioration alors récente des algorithmes de *deep learning* (DL), pour évaluer rapidement l'affinité de composés pour leurs cibles thérapeutiques [8]. Quatre ans plus tard, Stokes et al. (2020) ont démontré une application pratique en entraînant un algorithme de DL à partir de bases de données de médicaments existants, révélant des propriétés antibactériennes jusqu'alors inconnues de certaines molécules [9]. Des applications concrètes émergent dans l'industrie, comme la start-up *Owkin*, fondée en 2016, qui se base sur l'analyse multi-omique par IA et commercialise des tests diagnostiques à partir de biomarqueurs qu'ils ont identifiés, et propose des repositionnements de molécules [10]. L'analyse de données à grande échelle, l'utilisation de la simulation et du criblage numérique promettent d'accélérer la conception de médicaments en réduisant le temps et les ressources nécessaires.

Utilisation clinique

Aide à la décision clinique

La classification des cancers gagne en complexité. Alors qu'il y a quelques années, on définissait principalement les tumeurs par leur organe d'origine (cancer du poumon, cancer du sein, etc.), on les caractérise désormais de manière plus fine en s'appuyant sur leurs caractéristiques omiques : mutations somatiques, profils épigénétiques ou encore aberrations chromosomiques. Ceci permet une meilleure compréhension et un traitement plus ciblé des différentes tumeurs. On trouve par exemple l'outil *moBRCA-net* qui fait appel aux méthodes de DL pour classifier les tumeurs du cancer du sein à partir de l'expression des gènes, des microARN et de la méthylation de l'ADN [11].

Imagerie médicale

L'IA s'implante progressivement dans le domaine de l'analyse d'images médicales avec l'analyse des radiographies pour détecter des anomalies, le diagnostic de maladies cutanées en dermatologie, et l'examen automatisé des lames histologiques en pathologie. Certains outils sont désormais aussi disponibles sur nos smartphones, avec des applications telles que *SkinVision* [12] ou *MoleScope* [13], qui permettent d'analyser des nævus à partir de photos. On constate concrètement cette démocratisation des outils d'IA dans la région de Nancy avec la mise en place de scanners utilisant l'IA dès 2019 au CHRU pour améliorer la qualité des images et aider à la détection des anomalies [14], ou encore l'installation en 2023 d'un outil de radiothérapie d'aide au à la délimitation des zones à irradier à l'Institut de Cancérologie de Lorraine [15].

Objets connectés

L'IA et ses capacités d'analyses automatiques ont démocratisé le port de dispositifs de santé. Par exemple, l'*Apple Watch* est approuvée par la *Food and Drug Administration* (FDA) pour son électrocardiogramme intégré, permettant la détection

précoce de la fibrillation auriculaire. Les capteurs de glucose en continu aident les personnes diabétiques [16] ou les sportifs [17] à surveiller leur glycémie en temps réel. Ou encore les bagues connectées qui intègrent des capteurs de fréquence cardiaque, de température, d'oxygénation du sang, un accéléromètre et un capteur de conductivité de la peau. Des études en proposent des utilisations pour détecter les apnées du sommeil [18] ou pour prédire la date d'accouchement [19].

Transformers et *Large Language Models* (LLM)

L'un des progrès les plus significatifs dans le domaine de l'IA ces dernières années fut l'avènement des modèles de langage de grande taille ou LLMs, fondés sur une nouvelle architecture de réseaux de neurones appelée *transformer*. Cette approche a profondément modifié notre capacité à traiter, générer et comprendre le langage naturel, ouvrant la voie à des avancées dans de nombreux domaines scientifiques, notamment en recherche biomédicale [20].

Les LLMs s'appuient sur l'architecture des « *transformers* », ou « transformeurs », initialement développée pour le traitement du langage naturel [21]. Ce type de réseau neuronal repose sur un mécanisme dit « d'attention » qui permet de relier chaque élément à son entourage proche ou lointain dans une séquence, ce qui est particulièrement utile pour capter les dépendances longues dans les séquences d'ADN ou de protéines. Les modèles BERT [22] analysent une séquence de manière bidirectionnel, ce qui est utile pour des tâches comme l'annotation fonctionnelle. Les modèles GPT, eux, sont unidirectionnels et plus adaptés à la génération de texte ou de séquence.

Les *transformers* permettent d'extraire une représentation riche et contextuelle de chaque nucléotide ou acide aminé, ce qui facilite des tâches telles que la prédiction d'épissage, la détection de mutations fonctionnelles ou l'identification de structures secondaires dans les protéines. Par exemple, DNABERT est pré-entraîné de manière auto-supervisée sur des séquences d'ADN non annotées, où il apprend à prédire des *k-mers* masqués à partir du reste de la séquence, ce qui lui permet ensuite de reconnaître efficacement les motifs fonctionnels [23].

Certains modèles récents intègrent des données multimodales, par exemple, la séquence d'ADN, l'ARN, et la structure tridimensionnelle des protéines. Cela permet d'anticiper plus finement des effets régulateurs complexes, comme l'impact d'une mutation sur l'expression d'un gène ou la stabilité d'une protéine. Le modèle *IsoFormer* [24] illustre cette approche : il utilise des encodeurs séparés pour l'ADN, l'ARN et les protéines, puis agrège ces représentations via un mécanisme de *cross-attention* afin de prédire le niveau d'expression des isoformes d'ARN mesurées par RNA-seq. Ce modèle surpasse les approches mono-modales sur cette tâche transcriptomique spécifique.

Ces modèles donnent lieu à des méthodes d'explicabilité spécifiques, comme les *sparse autoencoder* dont on reparle dans la partie discussion.

Bien que cette thèse ne se consacre pas spécifiquement aux LLMs, de nombreux travaux récents, notamment la revue de Ruan et al. [20], mettent en évidence l'intérêt croissant de ces approches IA en bioinformatique. Au-delà de la représentation des séquences, les LLMs sont aujourd'hui explorés pour des applications variées : la prédiction de structures secondaires ou tertiaires de protéines, l'annotation de

séquences fonctionnelles comme les promoteurs ou les *enhancers*, la classification de types cellulaires à partir de données *single-cell*, ou encore la génération assistée de bases de données biologiques synthétiques. Ces exemples illustrent la capacité des LLMs à généraliser sur des tâches complexes à partir de larges corpus biologiques, et pourraient être les outils de demain pour la découverte dans les données omiques via des méthodes XAI.

Apprentissage automatique ou *machine learning*

Le terme « *artificial intelligence* » a été introduit en 1956 par John McCarthy à l'occasion de la conférence de *Dartmouth Summer Research Project*, considérée comme le point de départ officiel du domaine [25]. Cette rencontre visait à explorer comment des machines pouvaient simuler l'intelligence humaine à travers des symboles, des règles logiques et des calculs formels.

Le premier algorithme reconnu comme relevant de l'apprentissage automatique ou du *machine learning* (ML) est souvent attribué à Arthur Samuel, qui développa dès 1959 un programme capable d'apprendre à jouer au jeu de dames par renforcement de ses performances au fil des parties [26]. Il en conclut que son algorithme prouve que « *a computer can be programmed so that it will learn to play a better game of checkers than can be played by the person who wrote the program.* ». Dans ce manuscrit, les termes « apprentissage automatique », « *machine learning* » et ML seront employés de manière équivalente.

Dans cette section, nous décrivons brièvement les algorithmes d'apprentissage automatique mentionnés dans le manuscrit. Un algorithme de ML est une méthode qui apprend automatiquement à partir des données, lors d'une phase appelée entraînement du modèle. Selon la nature de la variable à prédire, ces algorithmes permettent soit de prédire une catégorie, dans le cas d'un problème de classification, soit une valeur numérique, dans le cas d'un problème de régression.

Decision tree **(DT)**

Les *decision trees*, ou arbres de décision, prédisent une catégorie ou une valeur numérique en suivant une série de règles simples basées sur les variables d'entrée. Leur structure arborescente les rend faciles à comprendre et adaptés à l'exploration de données.

L'algorithme de construction repose sur un principe récursif : à chaque étape, il choisit la variable et le seuil qui séparent le mieux les données, de manière à former des sous-groupes les plus homogènes possibles par rapport à la variable cible. Pour mesurer cette homogénéité, on utilise des critères comme l'entropie, ou l'indice de Gini en classification, et la variance résiduelle en régression. Ce processus se répète jusqu'à atteindre une condition d'arrêt, comme une profondeur maximale de l'arbre, un nombre minimum d'observations par nœud, ou un gain d'information trop faible.

Nous utilisons pour cela le package *R rpart*, qui permet notamment de contrôler la complexité de l'arbre via le paramètre *cp* (*complexity parameter*), ou de fixer la profondeur maximale avec *maxdepth*.

Les arbres de décision sont rapides à entraîner et à interpréter, mais ils ont tendance à surapprendre, on parle d'*overfitting*, si aucune régularisation n'est appliquée. C'est pourquoi ils sont souvent utilisés comme briques de base dans des méthodes plus robustes comme les *random forests* ou le *boosting*.

En biologie, les arbres de décision ont par exemple été utilisés pour identifier des biomarqueurs protéiques du cancer de l'ovaire à partir de données de spectrométrie de masse [27], pour classer des profils transcriptomiques liés à la COVID-19 [28], ou encore pour prédire la réponse au traitement dans des cancers hématologiques à partir de données cliniques et omiques [29].

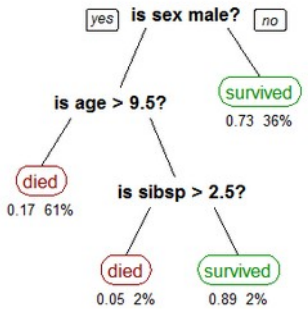

Figure 1: Exemple d'arbre de décision. Réf. : https://commons.wikimedia.org/wiki/File:CART_tree_titanic_survivors.png

Random Forest **(RF)**

Les *random forests*, ou « forêts aléatoires », combinent plusieurs arbres de décision construits sur des sous-ensembles aléatoires de données et de variables pour améliorer la précision des prédictions et limiter le surapprentissage. Elles peuvent être utilisées aussi bien pour prédire une catégorie que pour estimer une valeur numérique.

Le principe de base repose sur le "*bagging*", ou *bootstrap aggregating* : chaque arbre est entraîné sur un échantillon aléatoire des données, obtenu par tirage avec remise, et à chaque nœud seul un sous-ensemble de variables est testé pour le *split*. Ce double aléa introduit de la diversité entre les arbres, ce qui réduit la variance du modèle global. La prédiction finale est obtenue par vote majoritaire en classification ou par moyenne en régression.

Nous utilisons pour cela le package *R randomForest*, qui permet notamment de régler le nombre d'arbres avec le paramètre *ntree*, et le nombre de variables sélectionnées aléatoirement à chaque nœud avec *mtry*.

Cette méthode est robuste et offre de bonnes performances dans de nombreux contextes, même avec des données bruitées ou des variables corrélées. Elle fournit aussi une estimation de l'importance des variables, utile pour l'interprétation. En

revanche, le modèle résultant est souvent considéré comme une “boîte noire” moins interprétable que les arbres de décision individuels.

Un autre avantage pratique est qu’un *random forest* peut inclure une validation interne via les données non sélectionnées dans chaque *bootstrap*, ce sont les échantillons « *out-of-bag* », permettant d’obtenir une estimation directe de l’erreur globale sans passer par le processus de train/test split.

Les RF sont largement utilisées en bioinformatique, par exemple pour sélectionner et classer des gènes discriminants dans des données *microarrays* [30], pour analyser l’hétérogénéité tumorale à partir de données protéomiques [31], ou encore pour prédire la mortalité chez les patients insuffisants cardiaques [32].

Support Vector Machine (SVM)

Les machines à vecteurs de support, ou SVM, sont des algorithmes qui apprennent à séparer les données à l’aide d’une frontière optimale, appelée « hyperplan ». Elles permettent principalement de prédire une catégorie dans le cas de problèmes de classification, mais peuvent aussi être adaptées à la régression (*Support Vector Regression*).

Le principe de base consiste à trouver l’hyperplan qui maximise la marge, c’est-à-dire la distance entre cette frontière et les points les plus proches de chaque classe, appelés vecteurs de support. Cette maximisation de la marge permet d’obtenir un modèle plus robuste et mieux généralisable. Lorsque les données ne sont pas linéairement séparables, une pénalisation des erreurs de classification est introduite via un paramètre de régularisation C, qui contrôle le compromis entre marge large et erreurs tolérées.

Nous utilisons le package *R e1071*, qui permet notamment de choisir le type de noyau utilisé avec le paramètre *kernel*. Le noyau, ou « *kernel »*, détermine la forme de la frontière de séparation : un noyau linéaire trace une frontière droite, tandis qu’un noyau radial, ou *Radial Basis Function* (RBF) permet de tracer des frontières plus flexibles et non linéaires.

L’introduction de fonctions noyaux permet en effet de projeter les données dans un espace de dimension supérieure, où elles deviennent plus facilement séparables. Cette capacité rend les SVM très efficaces sur des jeux de données complexes ou de petite à moyenne taille. Toutefois, ils deviennent plus coûteux en calcul et en mémoire sur des ensembles très volumineux, et leur interprétation reste moins intuitive que celle d’un arbre de décision.

Les SVM ont été utilisés pour analyser les profils d’expression génique et prédire la fonction de gènes de levure [33], pour identifier des biomarqueurs transcriptomiques associés aux disparités ethniques dans le cancer du sein triple-négatif [34], ou encore pour la classification de mélanomes [35].

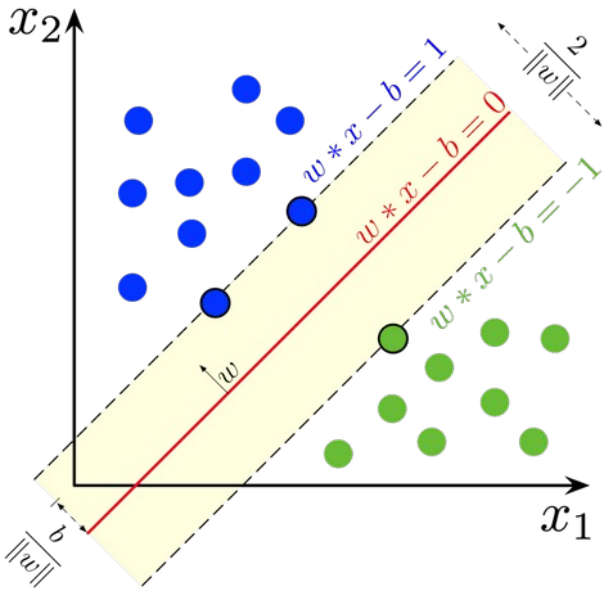

Figure 2: Illustration de SVM à noyau linéaire.
Réf. :
https://en.wikipedia.org/wiki/Support_vector_machine#/media/File:SVM_margin.png

Réseaux de neurones/ *Deep Learning*

Le *deep learning*, ou réseaux de neurones, est une approche inspirée du fonctionnement du cerveau humain. Ces modèles sont constitués de couches de « neurones » artificiels, qui transforment progressivement les données en combinant des pondérations ajustées lors de l'entraînement. Grâce à leur architecture flexible, ils peuvent modéliser des relations complexes entre les variables et sont utilisés pour des tâches variées comme la classification, la régression ou l'analyse de séquences.

L'entraînement repose sur la rétropropagation du gradient (*backpropagation*), qui consiste à calculer l'erreur du modèle puis à ajuster progressivement les poids de chaque connexion à l'aide d'un algorithme d'optimisation. Les plus utilisés sont la descente de gradient stochastique (SGD) et l'algorithme Adam, ce dernier étant devenu le standard dans de nombreux cas pratiques. L'empilement de plusieurs couches permet d'apprendre des représentations hiérarchiques : les couches initiales captent des motifs simples, par exemple des bords dans une image, tandis que les couches profondes combinent ces motifs pour reconnaître des structures plus complexes, par exemple un domaine protéique ou un type cellulaire.

Ce type de modèle est particulièrement performant sur des données massives et complexes, mais il nécessite un volume de données important, une puissance de calcul élevée, souvent via GPU, et reste difficile à interpréter. C'est pourquoi des architectures spécialisées ont été développées, comme les réseaux convolutifs (CNN) pour les images, les perceptrons multicouches (MLP), les réseaux récurrents (RNN), ou à mémoire longue (LSTM) pour les données séquentielles, et plus récemment les transformeurs, très utilisés en traitement du langage naturel et de plus en plus en biologie.

- **CNN**

Les réseaux de neurones convolutifs, ou CNN, sont un type de réseau de neurones spécialement adaptés à l'analyse de données structurées comme des images ou des matrices. Ils utilisent des filtres appelés convolutions pour repérer automatiquement des motifs dans les données, comme des formes, des zones d'activation ou des profils locaux. Ce fonctionnement les rend particulièrement efficaces pour extraire des caractéristiques pertinentes dans des données complexes, sans avoir besoin de les sélectionner manuellement.

Les couches convolutives sont souvent combinées à des opérations de regroupement, ou « *pooling* », qui réduisent la dimension des données tout en conservant l'essentiel de l'information, ce qui rend les modèles plus efficaces et moins sensibles au bruit. À mesure que l'on empile les couches, les CNN apprennent des représentations hiérarchiques : les premières couches captent des motifs simples, tandis que les couches plus profondes identifient des structures plus complexes et abstraites.

Ces architectures sont devenues incontournables pour l'analyse d'images médicales [36], mais trouvent aussi des applications en biologie computationnelle, par exemple pour détecter des motifs de séquences d'ADN ou d'ARN [37], prédire des domaines protéiques [38], ou encore analyser des matrices d'expression issues de données omiques [39].

- **MLP**

Les perceptrons multicouches, ou MLP, sont des réseaux de neurones composés de plusieurs couches entièrement connectées entre elles. Chaque neurone d'une couche est relié à tous ceux de la couche précédente. Ce type d'architecture est utilisé pour modéliser des relations complexes dans des données tabulaires ou vectorielles, et permet de réaliser des tâches de classification ou de régression.

L'apprentissage se fait également par rétropropagation. Grâce à la présence de plusieurs couches cachées et de fonctions d'activation non linéaires, comme *ReLU*, *tanh* ou *sigmoid*, les MLP peuvent approximer pratiquement n'importe quelle fonction, ce qui les rend très flexibles.

En revanche, ils peuvent nécessiter un grand nombre de paramètres et donc un entraînement coûteux en données et en calcul, ce qui les rend parfois moins adaptés que d'autres architectures spécialisées, comme les CNN ou les RNN, pour certaines tâches.

Malgré cela, les MLP restent un outil de référence en apprentissage supervisé, notamment pour l'analyse de données tabulaires en bioinformatique, où ils peuvent capturer des relations complexes entre variables biologiques. On trouve par exemple des cas de classification de sous-types cellulaires immunitaires [40], ou de classification de lymphomes [41]. C'est également l'architecture la plus courante dans les cours et tutoriels pour débuter en *deep learning*.

- **RNN / LSTM**

Les réseaux de neurones récurrents, ou RNN, sont conçus pour traiter des données séquentielles, comme des séries temporelles ou des séquences biologiques. Contrairement aux architectures classiques qui considèrent les données comme indépendantes, les RNN maintiennent un état interne qui leur permet de prendre en compte les informations des étapes précédentes lors de la prédiction. Cela les rend adaptés à des tâches comme la reconnaissance vocale, ou l'analyse de signaux physiologiques tels que les électrocardiogrammes. Toutefois, les RNN standards souffrent du problème du gradient qui disparaît ou explose, ce qui limite leur capacité à modéliser des dépendances sur de longues séquences.

Pour pallier cette limite, les LSTM, pour *Long Short-Term Memory*, ont été développés. Grâce à une architecture de cellules récurrentes enrichies de portes *input/output/forget*, elles permettent de mieux retenir l'information utile sur de longues séquences, tout en oubliant les éléments non pertinents. Les LSTM sont devenus un standard dans l'analyse de séquences longues et complexes.

En biologie, ces architectures trouvent des applications variées : par exemple, un réseau récurrent profond a permis de découvrir des règles complexes pour distinguer les ARN codants des ARN non-codants à partir de séquences nucléotidiques [42]. De même, des modèles LSTM sont utilisés pour l'analyse de signaux physiologiques, comme la classification des rythmes cardiaques à partir d'ECG [43]. Ces exemples illustrent la capacité des RNN et LSTM à capturer la structure temporelle ou séquentielle des données biologiques, offrant un outil pour mieux comprendre et prédire ces phénomènes.

On mentionne également dans ce manuscrit des notions courantes en ML telles que les *embeddings*, les espaces latents et le gradient.

- Un ***embedding*** est une manière de transformer des données complexes, par exemple des mots, des gènes ou des séquences, en vecteurs numériques dans un espace continu. Par exemple, dans l'analyse transcriptomique, on peut représenter chaque gène comme un vecteur dont les dimensions résument son profil d'expression dans différents types cellulaires. Cette représentation permet de quantifier les similarités entre gènes et d'enrichir l'analyse fonctionnelle. Ces vecteurs sont projetés dans ce qu'on appelle un **espace latent**, un espace de plus faible dimension dans lequel les relations structurelles ou fonctionnelles entre les éléments sont préservées : deux gènes impliqués dans la même voie biologique y seront souvent proches, même s'ils n'ont pas une forte corrélation brute dans les données d'origine.
- Le **gradient** représente la pente de la fonction d'erreur du modèle. Il indique dans quelle direction il faut ajuster les paramètres pour que le modèle fasse moins d'erreurs. La **descente de gradient** consiste à suivre cette pente pas à pas pour améliorer progressivement les prédictions du modèle.

Intelligence artificielle explicable (XAI)

L'intelligence artificielle explicable, ou XAI pour *eXplainable Artificial Intelligence*, regroupe un ensemble de méthodes qui visent à mieux comprendre comment les

modèles d'apprentissage automatique prennent leurs décisions. Ces méthodes peuvent soit expliquer une décision particulière, ce sont les explications locales, soit donner une vue d'ensemble sur le fonctionnement du modèle, ce sont les explications globales.

Parmi les outils les plus connus, on trouve *SHAP* pour *SHapley Additive exPlanations* et *LIME* pour *Local Interpretable Model-agnostic Explanations*. *SHAP* utilise des concepts issus de la théorie des jeux pour attribuer à chaque variable une contribution claire à la décision. *LIME*, de son côté, construit un modèle local plus simple, autour d'un exemple précis, pour en expliquer la prédiction. Nous reviendrons plus en détail sur ces méthodes dans la suite de la thèse.

Il existe aussi des méthodes d'explicabilité propres à certains types de modèles. Par exemple, on utilise *Mean Decrease in Accuracy* (MDA) ou les scores de Gini pour les DT et RF, ou le score d'Olden pour les réseaux de neurones. Ces méthodes spécifiques sont souvent plus rapides, car elles sont soit calculées pendant l'entraînement du modèle, soit directement déduites des paramètres finaux du modèle. Nous les présenterons dans l'axe 2 de la thèse.

Les approches XAI trouvent aujourd'hui de nombreuses applications en bioinformatique. Par exemple, *SHAP* a permis de vérifier que des CNN entraînés sur des données transcriptomiques issues de RNA-seq identifiaient bien les gènes discriminants associés à chaque tissu [44]. Dans une autre étude, un modèle de *Greedy Decision Forest* (GDF), une variante de RF qui construit les arbres en privilégiant les séparations les plus informatives à chaque étape en diminuant la part d'aléatoire, combiné à *SHAP* a mis en évidence des modules de gènes multi-omiques liés à des pathologies et identifié de nouveaux candidats biologiques pertinents [45]. De même, *LIME* et *SHAP* ont été utilisés pour expliquer des modèles de survie chez des patients atteints de cancer du nasopharynx, permettant d'identifier les facteurs pronostiques majeurs comme l'âge ou le stade tumoral [46]. D'autres travaux ont montré que l'application de diverses méthodes XAI sur des classifieurs transcriptomiques pouvait dégager des listes robustes de gènes contributeurs spécifiques aux tissus, facilitant l'interprétation biologique [47]. Enfin, une étude de classification pan-cancer a révélé grâce à *SHAP* un ensemble de 976 gènes signatures spécifiques de l'origine tissulaire des cancers, fournissant ainsi une explication transparente des prédictions du modèle et des candidats diagnostiques potentiels [48]

Un autre outil intéressant pour mieux comprendre les modèles est celui des attaques adverses. Ces techniques consistent à modifier très légèrement les données d'entrée pour tromper le modèle. En observant ce qui suffit à le perturber, on peut en déduire quels sont les éléments vraiment déterminants dans la prise de décision. Cela permet de révéler les faiblesses du modèle, ou les raccourcis qu'il a appris à tort. Nous verrons dans cette thèse des exemples sur des classifieurs sur données de scRNA-seq.

Les approches XAI permettent aussi de repérer des biais dans les modèles. Un exemple célèbre est celui d'*Amazon*, qui avait développé un outil de sélection de CV, mais qui favorisait les candidatures masculines. L'algorithme avait appris ce biais en analysant un historique de recrutement principalement masculin [49], [50]. Un autre exemple de référence est celui d'un modèle de classification d'images différenciant les loups des huskies. Bien que performant, il se basait sur la présence de neige en arrière-plan pour décider, car la plupart des images de loups avaient été prises dans la

neige [51]. C’est en utilisant *LIME* qu’on a pu visualiser que le modèle se focalisait surtout sur la neige, et non sur l’animal lui-même.

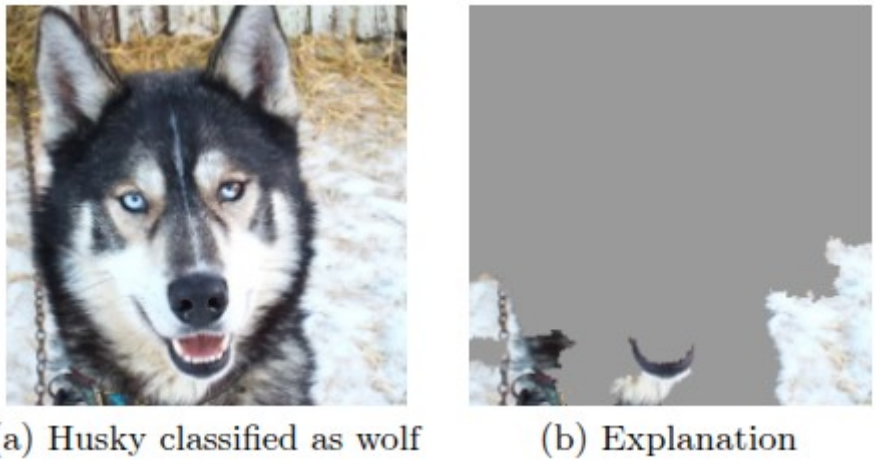

Figure 3: LIME permet de savoir quelle partie de l'image est utilisée par le classifieur, extrait de Tulio Ribeiro et al. 2016 [51]

En résumé, l’explicabilité est essentielle pour comprendre, corriger et fiabiliser les modèles de ML. Elle permet de savoir sur quoi le modèle s’appuie vraiment, d’identifier les erreurs et les biais, et de renforcer la confiance dans ses prédictions. Nous étudions dans l’axe 2 de la thèse comment ces outils permettent également de faire de la découverte en analyse de données omiques.

Bases de biologie moléculaire

Pour bien comprendre les données omiques décrites dans la section suivante, il est nécessaire de rappeler quelques mécanismes fondamentaux de biologie moléculaire. La biologie moléculaire étudie les mécanismes fondamentaux par lesquels l'information génétique contenue dans l'ADN est convertie en molécules fonctionnelles. Il existe deux grands types d'organismes : les eucaryotes et les procaryotes. Les eucaryotes, comme les cellules humaines, possèdent un noyau délimité par une membrane ainsi que des organites intracellulaires spécialisés (mitochondries, réticulum endoplasmique, etc.), tandis que les procaryotes, comme les bactéries, n'ont pas de noyau : leur ADN flotte librement dans le cytoplasme. Par exemple, *Escherichia coli* est un procaryote modèle très utilisé en biologie moléculaire. Chez les eucaryotes, par exemple les mammifères, l'organisation complexe permet une régulation fine des processus comme la transcription ou l'épissage. Cette thèse se concentre sur les eucaryotes. Nous décrivons ci-dessous les mécanismes propres à ce domaine du vivant.

Dogme central

L’ADN, extrêmement long, est empaqueté dans le noyau sous forme de chromatine, une structure composée d’ADN et de protéines, notamment les histones. Cette organisation permet de compacter l'information génétique tout en régulant son accessibilité à la machinerie cellulaire, notamment aux enzymes impliquées dans la transcription. Cette dernière constitue la première étape du dogme central de la biologie moléculaire : l'ADN est transcrit en ARN, lui-même traduit en protéines. Un grand groupe des protéines produites sont des enzymes, qui catalysent les réactions biochimiques et augmentent considérablement leur vitesse, permettant à la cellule de fonctionner efficacement même à température physiologique. Toutes ces réactions nécessitent de l'énergie, fournie en grande partie par un organite clé de la cellule : la mitochondrie. Cette structure joue un rôle central dans la production d’ATP, la principale source d’énergie de la cellule, grâce à la respiration cellulaire. Elle contient aussi des enzymes qui accélèrent fortement les réactions nécessaires, ce qui permet à la cellule de bien fonctionner à température normale. La traduction a lieu dans le cytoplasme, où le ribosome lit la séquence de l'ARNm pour assembler les acides aminés dans l'ordre correct, formant ainsi une chaîne polypeptidique. Le ribosome est un complexe ribonucléoprotéique formé en grande partie d’ARN ribosomique (ARNr), qui constitue à lui seul la majorité de la masse totale d’ARN dans une cellule. Il joue un rôle central dans la biologie cellulaire en catalysant la formation des liaisons peptidiques entre les acides aminés. La transcription est réalisée par l'ARN polymérase, qui synthétise un ARN messager (ARNm) à partir de la séquence d'ADN. L'ADN et l'ARN sont des molécules orientées, avec une extrémité dite 5' (cinq prime) et une autre 3' (trois prime), ce qui reflète l'organisation chimique de leurs nucléotides. La synthèse de l'ARN se fait toujours dans le sens 5' vers 3', c’est-à-dire que l’ARN polymérase ajoute les nucléotides à l’extrémité 3'. Il est important de noter qu’il existe plusieurs types d’ARN, et que tous ne codent pas pour des protéines. Parmi les ARN non codants, on retrouve les ARN de transfert (ARNt), les ARN ribosomiques (ARNr), mais aussi des ARN régulateurs tels que les microARN (miARN) ou les ARN longs non codants (lncARN), qui participent à la régulation de l’expression génique et à d’autres fonctions cellulaires essentielles. Chez les eucaryotes, cet ARNm précurseur subit ensuite plusieurs étapes de maturation, dont le clivage, l'ajout d'une coiffe en 5', une queue poly-A en 3', et surtout l'épissage, ou *splicing*, processus qui consiste à retirer les introns, c’est-à-dire les séquences non codantes, pour ne conserver que les exons, c’est-à-dire les séquences codantes. Ces protéines peuvent subir des modifications post-traductionnelles comme la phosphorylation, l’acétylation ou l'ubiquitination, qui modifient leurs fonctions, leur localisation ou leur stabilité, jouant un rôle clé dans les voies de signalisation cellulaire. Dans de nombreux cas, la signalisation repose sur le transfert de groupes phosphates d'une protéine à une autre, un mécanisme catalysé par des kinases et réversible par des phosphatases. Ce transfert en cascade permet d'amplifier le signal et de contrôler rapidement diverses fonctions cellulaires comme la croissance, la différenciation ou l'apoptose.

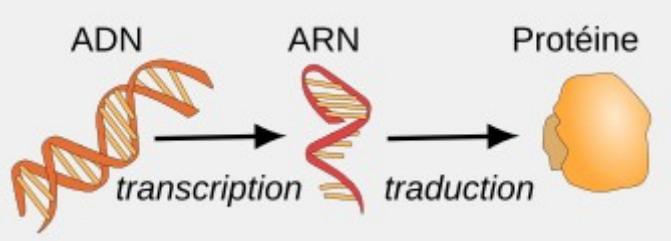

Figure 4: Dogme central. Réf. : https://commons.wikimedia.org/wiki/File:Dogme-central-BioMol.svg

Régulation génique

Parmi ces voies, on retrouve notamment celles contrôlées par des facteurs de transcription, des protéines capables de réguler l'expression des gènes en se fixant directement à l'ADN, ou encore par des récepteurs membranaires, qui transmettent des signaux extérieurs à la cellule. L'expression des gènes est également régulée par des mécanismes épigénétiques, tels que la méthylation de l'ADN, qui contrôle l'accessibilité des gènes aux facteurs de transcription. Un autre mécanisme important repose sur l'état de compaction de la chromatine : lorsque celle-ci est ouverte (euchromatine), les gènes sont accessibles et potentiellement exprimés ; à l'inverse, une chromatine condensée (hétérochromatine) rend l'ADN moins accessible et réprime l'expression génique. Ce contrôle repose en grande partie sur des modifications post-traductionnelles des histones, notamment l'acétylation évoquée précédemment, qui diminue l'interaction entre l'ADN et les histones, facilitant ainsi l'ouverture de la chromatine et l'accès de la machinerie transcriptionnelle. Cette régulation s'exerce souvent au niveau des promoteurs, des régions spécifiques de l'ADN situées en amont des gènes, où les facteurs de transcription se fixent pour initier ou inhiber la transcription. Ces promoteurs peuvent être proximaux (juste en amont du site de transcription) ou distaux (plus éloignés, mais capables d'agir via un repliement de l'ADN). La méthylation excessive de ces régions, en particulier des promoteurs, peut bloquer l'accès aux facteurs de transcription et ainsi réprimer l'expression des gènes cibles. Ensemble, ces mécanismes aboutissent à la production de métabolites, petites molécules impliquées dans l'énergie, la croissance et la survie cellulaire, reflétant ainsi l'état fonctionnel de la cellule.

Cancer

Comprendre ces mécanismes moléculaires est essentiel pour élucider le fonctionnement des cancers, qui en dérivent souvent par altération ou détournement. Cette maladie résulte souvent de perturbations multiples : mutations génétiques, dérégulation des mécanismes épigénétiques, ou altération des voies de signalisation cellulaire. Par exemple, le blocage d'un récepteur mitochondrial tel que BCL-2 peut induire l'apoptose, un mécanisme de mort cellulaire programmé. La surexpression de molécules comme PD-L1 peut permettre aux cellules cancéreuses d'échapper au système immunitaire en inhibant l'activation des lymphocytes T cytotoxiques. En temps normal, les lymphocytes T cytotoxiques reconnaissent les cellules anormales

via leur récepteur TCR (*T-cell receptor*), qui détecte les antigènes présentés par les molécules du MHC (*Major histocompatibility complex*). Le récepteur PD-1, quant à lui, agit comme un frein immunologique : lorsqu'il est activé par PD-L1, il inhibe la fonction effectrice des lymphocytes T pour éviter des réponses auto-immunes excessives. L'interaction entre PD-L1 (exprimé par la cellule tumorale) et PD-1 (porté par les lymphocytes T) envoie un signal inhibiteur qui bloque leur activité cytotoxique, permettant ainsi à la tumeur de persister. Ou encore, des facteurs tels que VEGF qui stimulent la croissance tumorale par la formation de nouveaux vaisseaux sanguins, appelé angiogenèse. Le micro-environnement tumoral joue également un rôle crucial : il peut influencer le comportement des cellules cancéreuses en modifiant leur métabolisme, leur capacité à proliférer ou à résister aux traitements. Comprendre ces mécanismes interconnectés grâce aux approches omiques est essentiel pour développer de nouvelles stratégies thérapeutiques ciblées et efficaces. On peut retrouver un récapitulatif des mécanismes du cancer dans les publications « *hallmarks* du cancer », publiées en 2000, 2011 et 2022, [52], [53], [54].

Données omiques

Les données omiques désignent l'ensemble des approches permettant de caractériser à grande échelle les différentes couches moléculaires au sein d'un système vivant. Elles permettent de mesurer de façon exhaustive, les gènes (génomique), les ARN (transcriptomique), les protéines (protéomique), les métabolites (métabolomique), ou encore les modifications épigénétiques comme la méthylation de l'ADN (épigénomique). Ces approches ont connu une forte croissance grâce aux avancées technologiques en biologie moléculaire et en instrumentation, associés à de nombreux développements logiciels pour leur analyse.

Dans cette section, nous présentons les principales technologies omiques, leurs fondements méthodologiques, ainsi que leurs évolutions récentes.

Génomique

La génomique vise à caractériser la séquence et les variations de l'ADN. Elle constitue la couche initiale de l'information biologique, avant passage à l'ARN et aux protéines. Grâce au séquençage à haut débit (*Next Generation Sequencing*, NGS), il est désormais possible de séquencer des génomes entiers ou ciblés à des coûts devenus compatibles avec une application à large échelle. Ces données permettent d'identifier les mutations, les délétions, les duplications, les aberrations chromosomiques ou les polymorphismes nucléotidiques (SNPs).

Un domaine central de la génomique moderne est celui des GWAS (*Genome-Wide Association Studies*), qui visent à détecter des associations entre des variants génétiques et des traits phénotypiques ou des maladies complexes. Ces études s'appuient sur des centaines de milliers d'échantillons génotypés, et nécessitent des modèles statistiques spécifiques pour corriger le taux de fausses découvertes (*False Discovery Rate*, FDR) lors de multiples comparaisons. Depuis l'étude considérée comme étant le premier GWAS par Klein et al. en 2005, qui a identifié une association entre des polymorphismes du gène *CFH* et la dégénérescence maculaire liée à l'âge (AMD) à l'aide d'une puce *Affymetrix 100K* Klein et al. [55], en 2005, les GWAS ont

permis l'identification de milliers de loci associés à des maladies courantes comme le diabète de type 2, les maladies cardiovasculaires ou certains cancers.

L'analyse des données de séquençage commence par l'alignement des lectures sur le génome de référence, souvent réalisé avec des outils comme *Bowtie* [56] ou *BWA* [57]. Ensuite, des programmes comme *samtools*, *bcftools* [58] ou *GATK* [59] sont utilisés pour manipuler les fichiers et identifier les variations génétiques. Pour les études GWAS, les fichiers de génotypes sont ensuite analysés avec des outils comme *PLINK*, qui permet de filtrer les données, réaliser les tests statistiques et corriger les biais.

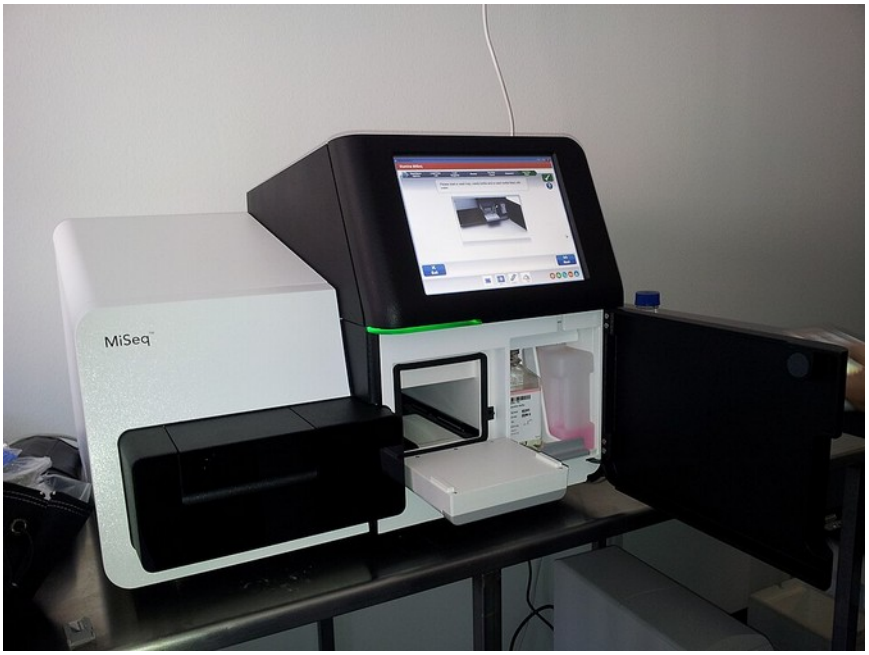

Figure 5: Séquenceur Illumina MiSeq. Réf. : https://commons.wikimedia.org/wiki/File:Illumina_MiSeq_sequencer_2.jpg

Transcriptomique : des puces au *single-cell*

La transcriptomique vise à quantifier l'expression des gènes. Historiquement, les premières analyses transcriptomiques à grande échelle se sont appuyées sur des puces à ADN (*microarrays*), où des sondes correspondant à des gènes connus sont hybridées avec des ARN marqués. Cette technologie a été massivement utilisée entre 2000 et 2010, avant d'être peu à peu supplantée par le séquençage d'ARN (RNA-seq), plus sensible et capable de détecter des isoformes ou des transcrits rares.

Bulk

Le RNA-seq en mode *bulk* permet de quantifier l'expression des gènes à partir d'un mélange de cellules. L'ARN total est extrait, converti en ADNc, puis séquencé à haut débit. Le résultat est une estimation de l'abondance des transcrits pour chaque gène,

sous forme de comptes bruts ou de valeurs normalisées (TPM, FPKM). Cette approche a supplanté les puces à ADN au début des années 2010, car elle permet une détection non-biaisée de tous les transcrits présents, y compris les isoformes alternatifs ou les gènes faiblement exprimés. Elle ne nécessite pas de sondes prédéfinies, contrairement aux *microarrays*.

Le RNA-seq *bulk* reste aujourd'hui une méthode de référence pour explorer les effets globaux d'une condition (traitement, mutation, maladie…) sur l'expression génique, en comparant différents groupes d'échantillons. Ses avantages incluent une bonne robustesse statistique, une profondeur de séquençage élevée, et un coût maîtrisé. En revanche, il masque l'hétérogénéité cellulaire, en produisant une mesure moyenne d'expression, ce qui peut masquer l'existence de sous-populations ou de profils rares.

Pour des tissus complexes comme les tumeurs, les organes immunologiques ou les mélanges cellulaires du sang, l'interprétation des données *bulk* nécessite parfois une étape dite de « déconvolution », qui vise à estimer la proportion relative des différents types cellulaires présents dans l'échantillon à partir du profil d'expression global.

Pour analyser ces données, on commence par aligner les séquences sur un génome de référence avec un outil comme *STAR* [60]. On quantifie ensuite combien de séquences correspondent à chaque gène, par exemple avec *featureCounts* [61]. Enfin, pour identifier les gènes dont l'expression change entre les groupes, on utilise des packages *R* comme *DESeq2* [62], *edgeR* [63] ou *limma* [64] qui repose sur une approche *empirical Bayes* ou *eBayes* , qui utilise le niveau d'expression de tous les gènes pour estimer une variance plus probable de chacun, ce qui est particulièrement utile lorsque le nombre d'échantillons est limité. Par rapport à un *t-test* classique, qui estime la variance séparément pour chaque transcrit, *eBayes* [65] réduit le risque de faux positifs et améliore la puissance statistique. Pour l'étape de déconvolution, des algorithmes comme CIBERSORT [66] sont fréquemment utilisés pour estimer les proportions des différents types cellulaires.

Single-cell, scRNA-seq

La transcriptomique en *single-cell*, ou scRNA-seq, consiste à capturer, amplifier puis séquencer l'ARN de chaque cellule d'un échantillon, on obtient leur profil transcriptomique complet. La première preuve de concept a été publiée en 2009, le transcriptome d'un unique blastocyte de souris a pu être lu, ouvrant la voie à l'analyse d'hétérogénéité intra-tissulaire [67]. L'automatisation permet aujourd'hui de traiter des centaines de milliers de cellules par expérience.

Deux grandes avancées technologiques ont eu lieu ces dernières années :

- Le **CITE-seq** (*Cellular Indexing of Transcriptomes and Epitopes*) [68] ou la solution commerciale *AbSeq* [69] ajoutent des anticorps codés par codes-barres ADN pour quantifier simultanément ARN et protéines de surface au sein de chaque cellule.
- Le ***spatial transcriptomics*** qui fixe les cellules sur une lame recouverte d'amorces portant un code-barre, de sorte que chaque molécule d'ARN conserve ses coordonnées, permettant une représentation spatiales des cellules [70].

Avec ces technologies on peut répondre à des questions biologiques diverses. Par exemple déterminer des sous catégories de tumeurs, ou comparer le tissu tumoral au tissu sain. On peut également identifier les cellules à l'origine de la tumeur [71], ou des sous-clones responsables de rechute ou de chimio-résistance [72]. Avec le *spatial transcriptomics* il devient également possible d'étudier les relations entre tumeur et micro-environnement [73]. Un autre cas d'usage classique est l'analyse des trajectoires de différenciation, par exemple quels sont les différents stades de la cellule entre une cellule souche hématopoïétique et une cellule immunitaire mature.

Pour analyser ces données, plusieurs outils bioinformatiques sont largement utilisés, notamment *Seurat* [74] en *R*, *Scanpy* [75] en python et *Monocle* [76] pour l'analyse de trajectoires. Ces packages permettent de réaliser l'ensemble du pipeline, depuis la normalisation jusqu'à la visualisation, en passant par le clustering et l'annotation cellulaire. Pour l'annotation automatisée des types cellulaires, il existe également des plateformes en ligne comme *Azimuth* [77], développée par les auteurs de *Seurat*, qui permet d'annoter un jeu de données scRNA-seq en le projetant sur un atlas de référence via une interface web ou une API *R*/*Seurat*.

Cela ouvre une perspective en pratique clinique. Les coûts de séquençage diminuent et les pipelines analytiques se standardisent, plusieurs études récentes montrent la faisabilité d'analyses spatiales sur coupes FFPE pour la stratification de patients [78], [79], [80]. Le terme FFPE (pour *Formalin-Fixed, Paraffin-Embedded*) désigne une méthode standard de conservation des échantillons biologiques en anatomopathologie. Les tissus sont fixés au formol puis inclus dans la paraffine. Le fait de pouvoir réaliser des analyses transcriptomiques spatialisées directement sur ces coupes nous rapproche de l'utilisation en diagnostic de routine.

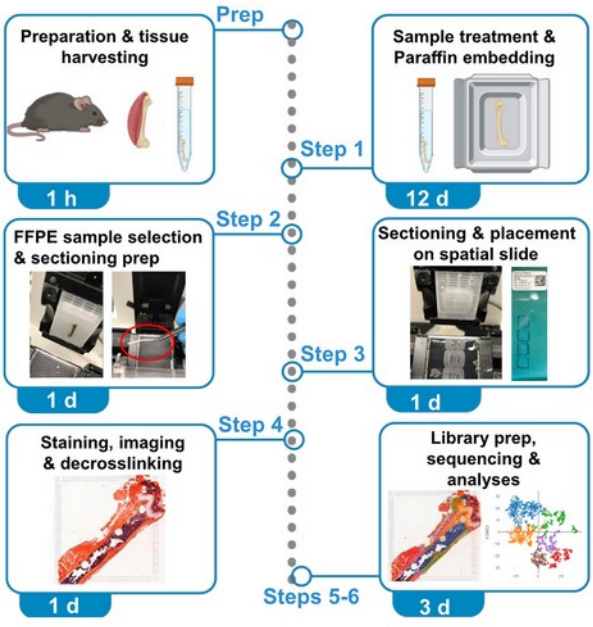

Figure 6: Illustration de la préparation d'échantillon FFPE pour scRNA-seq spatial issue d'un document de protocole du site STAR Protocols. [149]

Protéomique

La protéomique permet d'analyser le niveau d'expression des protéines et peptides contenus dans un échantillon biologique. Alors que les techniques classiques comme le *Western blot* ou l'ELISA ne permettent de mesurer qu'une seule protéine à la fois, les technologies modernes basées sur la spectrométrie de masse (MS) ont ouvert la voie à une lecture globale du protéome.

La protéomique quantitative repose principalement sur la chromatographie liquide couplée à la spectrométrie de masse (LC-MS/MS), souvent après digestion enzymatique des protéines en peptides. Cette technologie permet d'identifier et de quantifier simultanément plusieurs milliers de protéines dans un même échantillon.

Les données brutes issues de cette méthode peuvent être traitées, par exemple, avec le logiciel *MaxQuant* [81], qui réalise l'identification des peptides et la quantification relative ou absolue des protéines. Les résultats peuvent ensuite être analysés statistiquement avec le package *R DEP* [82] ou *limma* [64].

Des stratégies de marquage isotopique, comme iTRAQ ou TMT, permettent en outre de comparer quantitativement plusieurs conditions biologiques en parallèle.

En complément de cette analyse globale, des protocoles spécialisés permettent d'étudier des modifications post-traductionnelles (PTM), telles que la phosphorylation, la glycosylation, ou l'ubiquitination, qui régulent la fonction, la stabilité, la localisation ou les interactions des protéines. Ces modifications sont souvent impliquées dans les mécanismes de signalisation cellulaire. En particulier, la phosphorylation, où l'ajout réversible d'un groupe phosphate sur des résidus sérine, thréonine ou tyrosine, joue un rôle essentiel dans l'activation ou l'inactivation des protéines, et donc dans la régulation de voies de signalisation, par exemple la prolifération, l'apoptose ou la réponse aux stress.

L'analyse des protéines phosphorylées, la phosphoprotéomique, nécessite généralement une étape d'enrichissement en amont de l'analyse LC-MS/MS, par exemple via des billes d'affinité pour les phosphopeptides.

À Strasbourg, la plateforme LSMBO [83] (Laboratoire de Spectrométrie de Masse Bio-Organique, IPHC-CNRS) propose une expertise avancée pour ce type d'analyses. Elle travaille également à la miniaturisation des protocoles pour des analyses à partir d'échantillons de petite taille, un critère déterminant pour la faisabilité de certaines analyses sur des biopsies qui doivent parfois servir pour plusieurs mesures (séquençage, transcriptomique, recherche de marqueurs membranaires, etc). Ye et al. [84] ont récemment mis en œuvre des approches de protéomique unicellulaire (*single-cell proteomics*) permettant de détecter plus de 5000 protéines par cellule. Ces avancées ouvrent la voie à une caractérisation fine de l'hétérogénéité cellulaire au niveau protéique, jusque-là difficilement accessible.

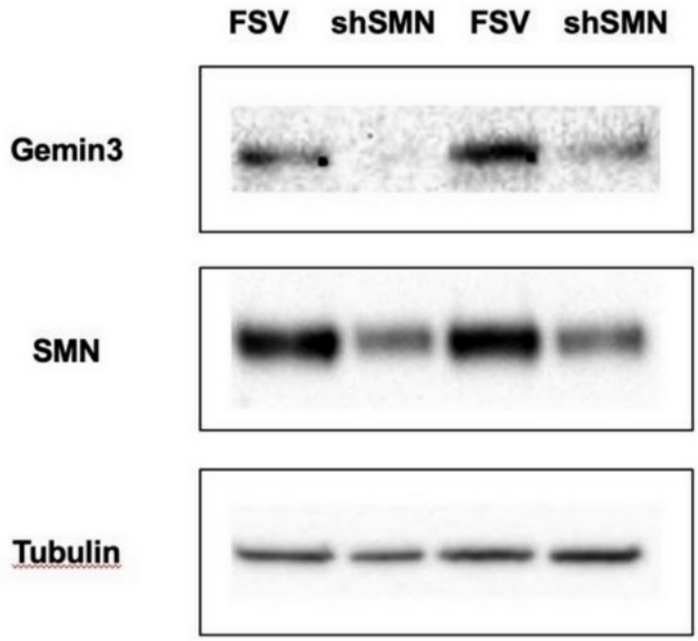

Figure 7: Exemple de résultat de Western Blot pour mesurer le niveau d'expression des protéines Gemin3, SMN et Tubuline. Réf. : https://commons.wikimedia.org/wiki/File:Western_Blot_2022-10-13_06_41_26.png

Épigénomique

Méthylome

La méthylation de l'ADN, principalement au niveau des cytosines CpG, est un mécanisme fondamental de régulation génique. Le méthylome reflète à la fois l'identité cellulaire, les états de différenciation, et les dérégulations pathologiques, notamment en cancérologie. Différentes approches permettent sa mesure :

- Les puces d'*Illumina* (ex. *Infinium 450K* ou *EPIC*) sondent plusieurs centaines de milliers de sites CpG avec un bon rapport coût/précision.
- Les méthodes de séquençage basées sur le bisulfite exploitent une conversion chimique spécifique : après traitement de l'ADN par une solution de bisulfite de sodium, les cytosines non méthylées sont transformées en uraciles (puis lues comme des thymines lors du séquençage), tandis que les cytosines méthylées restent inchangées. Cela permet d'inférer le statut de méthylation de l'ADN avec une résolution fine :
    - Le *Whole Genome Bisulfite Sequencing* (WGBS) couvre l'ensemble de l'ADN, offre une résolution nucléotidique mais reste coûteux.
    - Le *Reduced Representation Bisulfite Sequencing* (RRBS) permet une analyse ciblée des régions riches en CpG, telles que les promoteurs et les îlots CpG, en utilisant des enzymes de restriction spécifiques pour

fragmenter préférentiellement ces régions. Cette approche constitue un bon compromis entre coût et couverture du génome.

La première carte complète du méthylome humain a été publiée par Lister et al., 2009 [85]. L'intégration du méthylome à d'autres couches omiques améliore la prédiction des phénotypes, comme l'a montré Luo et al., 2017 [86], en combinant des données de méthylation de l'ADN à l'échelle cellule unique avec des profils transcriptomiques pour identifier des sous-types neuronaux distincts et leurs éléments régulateurs spécifiques dans le cortex de mammifère.

L'analyse des données brutes nécessite des outils bioinformatiques spécifiques. Pour les données de séquençage traité au bisulfite, WGBS ou RRBS), on utilise des aligneurs spécialisés comme *Bismark* [87] pour cartographier correctement les lectures converties sur le génome de référence. Une fois l'alignement effectué, le package *R methylKit* [88] est souvent utilisé pour quantifier le niveau de méthylation pour chaque site CpG et identifier les régions différentiellement méthylées comme les DMRs (*Differentially Methylated Region*). Pour l'analyse des puces Illumina, des packages *R* comme *minfi* [89] et *ChAMP* [90] sont utilisés pour le contrôle qualité, la normalisation des données et l'identification des positions différentiellement méthylées.

Organisation et accessibilité de la chromatine

Pour décrire l'état de la chromatine et ses mécanismes de régulation, trois grandes méthodes sont aujourd'hui utilisées : ATAC-seq [91], ChIP-seq [92] et Hi-C [93].

ATAC-seq

Pour étudier l'ouverture de la chromatine, plusieurs techniques ont été développées. La plus utilisée actuellement est ATAC-seq, qui utilise une enzyme site « transposase » pour insérer des adaptateurs dans les régions accessibles de l'ADN. Ces fragments sont ensuite séquencés, ce qui permet de cartographier rapidement les zones ouvertes du génome. ATAC-seq est devenue une méthode de référence, y compris en *single-cell* [94], où elle permet de mesurer l'accessibilité chromatinienne cellule par cellule. Ces données sont utiles pour repérer les éléments régulateurs actifs, comprendre les dynamiques épigénétiques, et relier certaines régions non codantes à des fonctions biologiques précises.

ChIP-seq

Pour savoir où une protéine se fixe sur l'ADN, comme des facteurs de transcription ou des histones, on utilise ChIP-seq. L'ADN et les protéines sont fixés de manière covalente, puis suit une étape de fragmentation et enfin les fragments sont sélectionnés grâce à un anticorps spécifique. On séquence ces fragments et on obtient une carte des sites de liaison sur le génome. Pour les facteurs de transcription, on voit des pics étroits, et pour les marques d'histones, des domaines plus larges. On a par exemple la protéine H3K27ac (version acétylée de H3K27) pour les régions actives, et H3K27me3 (version méthylée) pour les régions réprimées. Concrètement, cela sert à relier des régions régulatrices à leurs facteurs et comparer les états épigénétiques entre conditions.

Hi-C

Pour voir comment l'ADN est plié dans le noyau, on utilise surtout Hi-C. On fixe l'ADN, on le coupe, puis on relie entre eux les morceaux qui étaient proches ; on séquence ces paires et on obtient une carte de contacts. Cette carte montre des domaines, les TADs, et des boucles où des régions se touchent souvent. En *single-cell*, Hi-C révèle l'hétérogénéité d'une cellule à l'autre. Concrètement, cela sert à relier des *enhancers* à leurs gènes cibles, ou comparer l'architecture entre conditions, comme différents types cellulaires, tumoral *vs* bénin, etc.

En pratique, ATAC-seq et ChIP-seq suivent un schéma proche : alignement, par exemple avec *Bowtie2*, et appel de pics avec *MACS2*, et *DESeq2*/*edgeR*/*limma* pour les statistiques différentielles. Pour Hi-C il existe des pipelines dédiés comme *HIC-Pro* [95].

Métabolomique

La métabolomique consiste à quantifier les petites molécules (<1500 Da) produites par le métabolisme cellulaire : acides aminés, lipides, sucres, nucléotides… Elle fournit une image fonctionnelle de l'état physiologique de la cellule ou de l'organisme, sensible aux perturbations génétiques, environnementales ou thérapeutiques.

Les plateformes utilisent la spectrométrie de masse (MS) ou la *magnetic resonance spectroscopy* (MRS), couplées à des méthodes de séparation, chromatographie gazeuse ou liquide. Les applications sont nombreuses : biomarqueurs diagnostiques, suivi de traitement, nutrition, microbiote ou pharmacocinétique.

L'analyse des données brutes de spectrométrie de masse débute par le traitement des spectres pour détecter les pics, les aligner et les annoter. Des logiciels comme *XCMS* [96] en *R* ou des plateformes intégrées comme *MetaboAnalyst* [97] sont des standards pour ces premières étapes. Une fois les métabolites quantifiés, l'analyse statistique différentielle et l'enrichissement fonctionnel des voies métaboliques peuvent également être faites avec *MetaboAnalyst*.

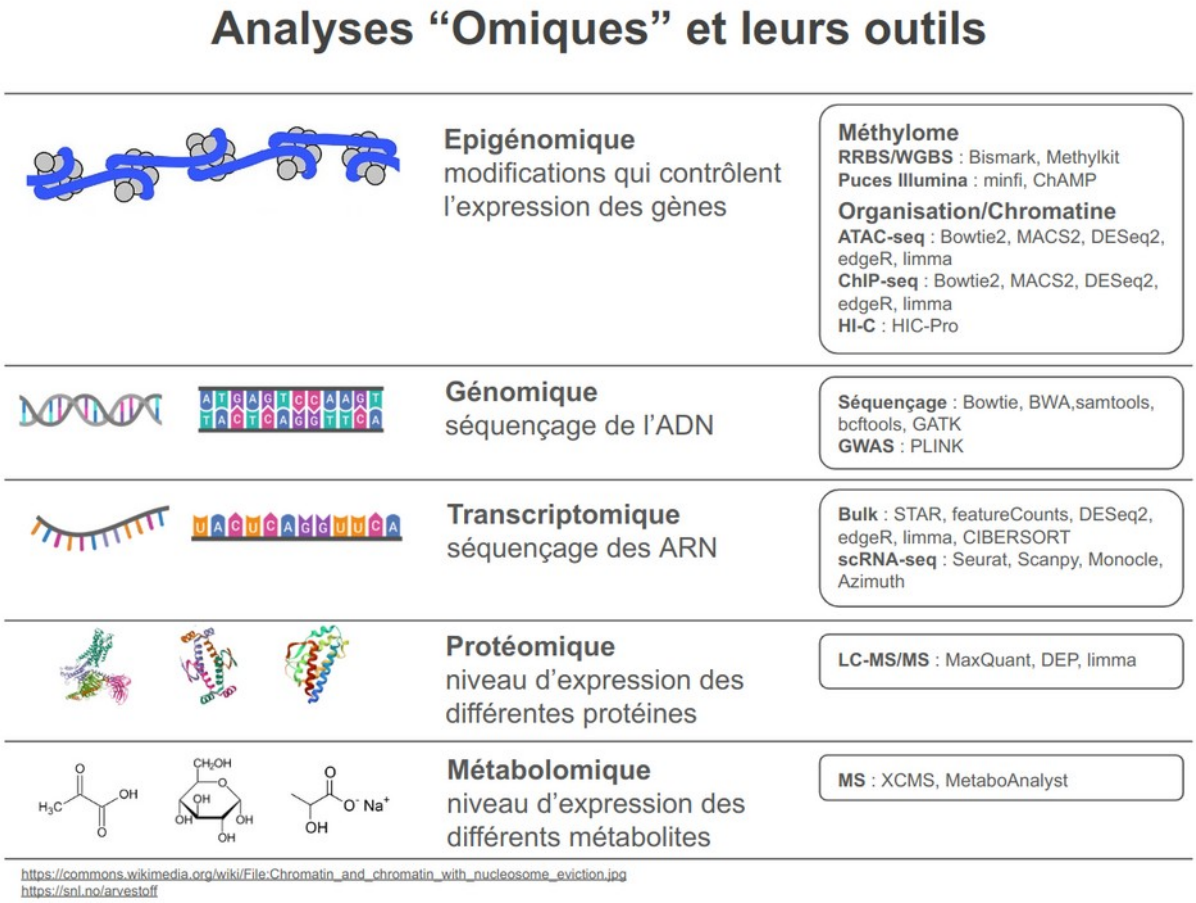

Figure 8: Vue d’ensemble des différentes couches de données omiques et des outils associés.

Vers une biologie multi-omique intégrée

Les données omiques ne prennent tout leur sens que dans une approche intégrée. La combinaison transcriptomique/protéomique permet d’expliquer des divergences d’expression post-transcriptionnelles. L’ajout du méthylome renseigne sur les mécanismes de régulation, tandis que le métabolome reflète la physiologie finale. L’analyse conjointe de plusieurs couches omiques (*multi-omics*) est devenue une stratégie répandue pour modéliser les systèmes biologiques complexes.

Technologies émergentes et futur des omiques

De nouvelles technologies font encore évoluer les approches omiques traditionnelles :

- *Nanopore sequencing*, par exemple *Oxford Nanopore* [98], permet de séquencer des molécules d’ADN ou d’ARN sans amplification, avec une lecture directe, longue, avec un dispositif portatif. Elle ouvre la voie à la lecture simultanée de séquences, méthylation, et isoformes ARN.

- *Multi-omics single-cell* : des approches combinées (ex. CITE-seq, scNMT-seq) permettent de mesurer simultanément plusieurs couches (ARN, protéines, méthylation) sur la même cellule.
- Séquençage direct des protéines : des prototypes de nanopores pour la protéomique (*Oxford Nanopore, Quantum-Si* [99]) visent à révolutionner la protéomique comme cela a été le cas pour l'ADN.

Axe 1 : attaques adverses sur scRNA-seq

Dans cet axe, nous développons un type d'algorithme XAI relevant de la catégorie des « attaques adverses », appliqué à des classifieurs entraînés sur des jeux de données transcriptomiques en single cell (scRNA-seq). Nous commencerons par expliciter la notion d'attaque adverse.

Attaques adverses

Définition et exemples

Les attaques adverses, ou *adversarial attacks*, sont une classe d'algorithmes qui explorent la façon dont réagissent les modèles de ML lors de perturbations de leur input [100]. On en trouve deux grands types, soit on cherche à perturber au maximum l'output du modèle en modifiant le moins possible les données entrantes (ou *input*), soit au contraire on cherche à modifier au maximum les données entrantes sans que les données sortantes (ou *output*) ne changent.

Les premiers exemples concernaient des attaques adverses sur classifieurs d'images.

L'exemple ci-dessous provient du jeu de données *ImageNet* [101]. Elle est analysée par un modèle d'apprentissage automatique, en l'occurrence le classifieur *GoogLeNet* [102]. La première image représente un panda, et le modèle la classe correctement. On y ajoute ensuite un bruit spécialement conçu pour tromper le modèle. À l'œil humain, l'image modifiée paraît identique à la précédente, mais cette fois *GoogLeNet* la prédit à tort comme étant un gibbon.

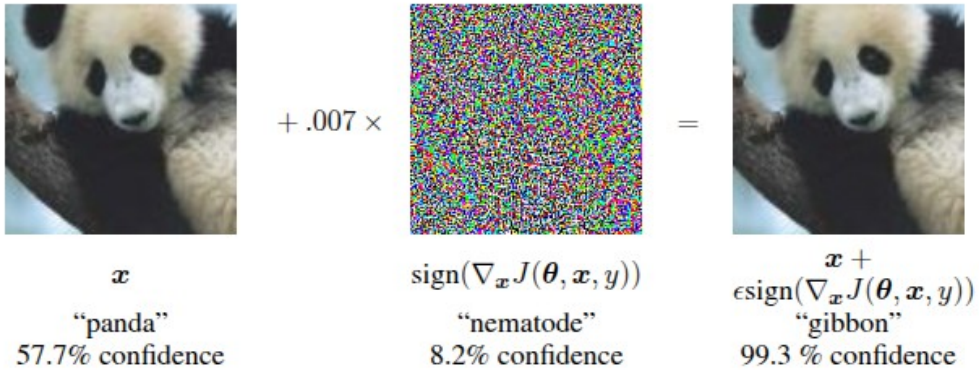

Figure 9: Exemple d'attaque adverse sur image de panda sur le classifieur GoogLeNet, extrait de Goodfellow et al. (2015) [100]

L'article de *Goodfellow* donne également un exemple correspondant au deuxième type d'attaque adverse, avec des images de bruit aléatoire modifiées pour augmenter le score de prédiction pour la catégorie « avion », les images encadrées en jaune dans l'exemple ci-dessous montrent les cas où l'image est effectivement classée comme « avion » après la modification.

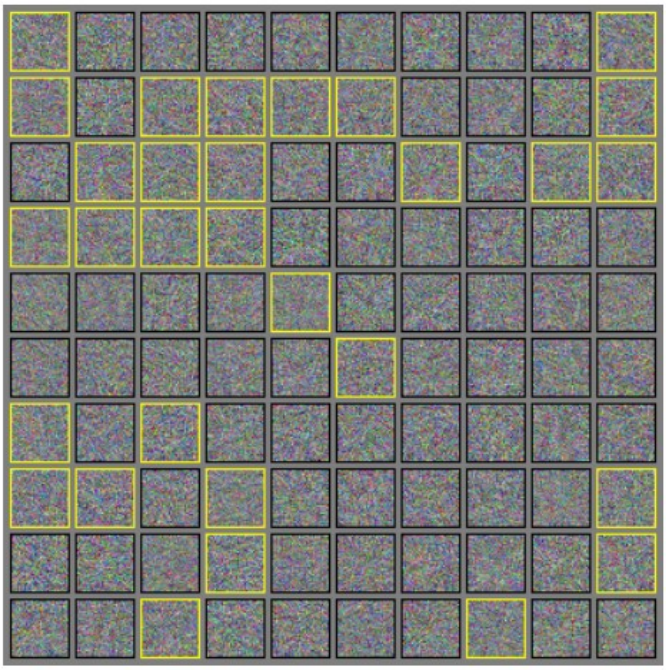

Figure 10: Des images générées à partir d'un bruit gaussien puis modifiées en direction de la classe "avion", parviennent à tromper un classifieur entraîné sur le jeu de données CIFAR-10, extrait de Goodfellow et al. 2015 [100]

Depuis 2015, la vulnérabilité des réseaux neuronaux à des attaques adverses a été démontrée dans de nombreux contextes biomédicaux. Sur les **radiographies thoraciques**, Han et al. ont montré qu'un classifieur CNN basé sur une architecture *ResNet* [103] entraîné sur le jeu de données de radiographies labellisées *ChestX-ray8* [104] pouvait être détourné par les algorithmes d'attaques adverses FGSM ou PGD (algorithmes FGSM et PGD décrits dans le paragraphe suivant), induisant un faux diagnostic de pneumonie alors que l'image reste bénigne pour un radiologue [105] .

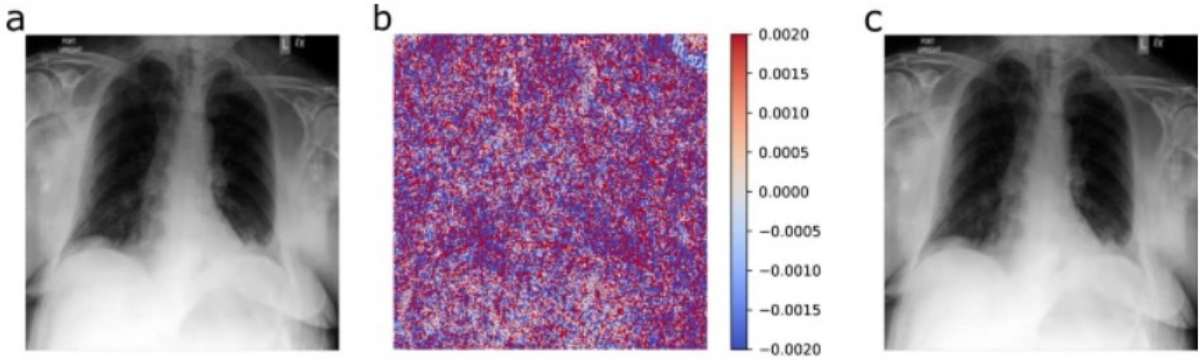

Figure 11: a) Radiographie thoracique d'origine, non modifiée. b) Bruit adversarial ajouté c) Radiographie modifiée (image originale + bruit), extrait de Han et al. 2021 [105]

En analyse d'**images histopathologiques**, Ghaffari Laleh et al. ont appliqué différents algorithmes d'attaque adverse dont PGD et FGSM à la classification de lames histologiques colorectal par deux types de classifieurs, les CNN et les *Vision Transformers* (ViTs), un type de classifieurs tirant parti du mécanisme « d'attention » à l'origine des progrès récents des LLMs comme chatGPT [21]. Ils arrivent à la conclusion que les ViTs résistent mieux que les CNN classiques, tout en soulignant la fragilité générale des modèles lorsque les perturbations sont invisibles à l'œil humain [106].

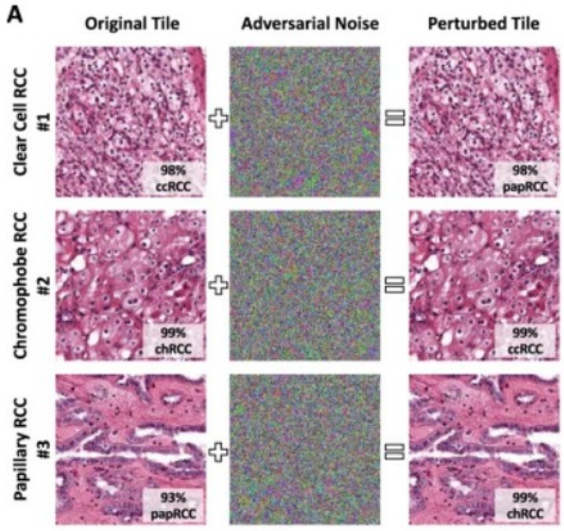

Figure 12: Adversarial attacks sur lames histologiques perturbant la classification des échantillons. Le degré de confiance du modèle pour chaque prédiction est indiqué sur chaque image, extrait de Ghaffari Laleh et al 2022 [106]

Dans le **traitement du langage naturel**, l'algorithme *TextFooler* de Jin et al. remplace itérativement des mots importants par des synonymes tout en préservant la signification et la grammaire du texte. Les auteurs démontrent que *TextFooler* peut tromper des modèles tels que BERT [107], LSTM et CNN sur des tâches de classification de texte [108] .

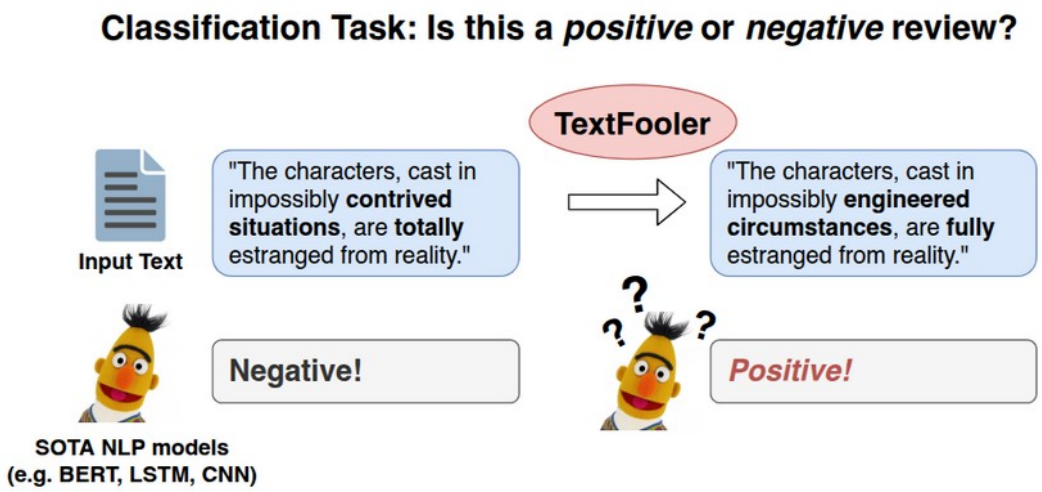

Figure 13: Exemple d'attaque adverse sur contenu textuel, le remplacement de mots par un synonyme fait passer le texte d'une prédiction « negative » à « positive », extrait de Jin et al. 2020 [108]

Enfin, pour les **séries temporelles, sur les électrocardiogrammes (ECG)**, Han et al. ont montré que des modèles CNN entraînés sur des enregistrements ECG peuvent être trompés par des attaques adverses imperceptibles à l'œil nu. Ces altérations générées par une version de PGD, invisibles pour des cardiologues expérimentés, induisent pourtant des erreurs de classification sur des tracés cliniquement interprétables, illustrant la vulnérabilité potentielle de ces systèmes dans des contextes médicaux réels. [109] .

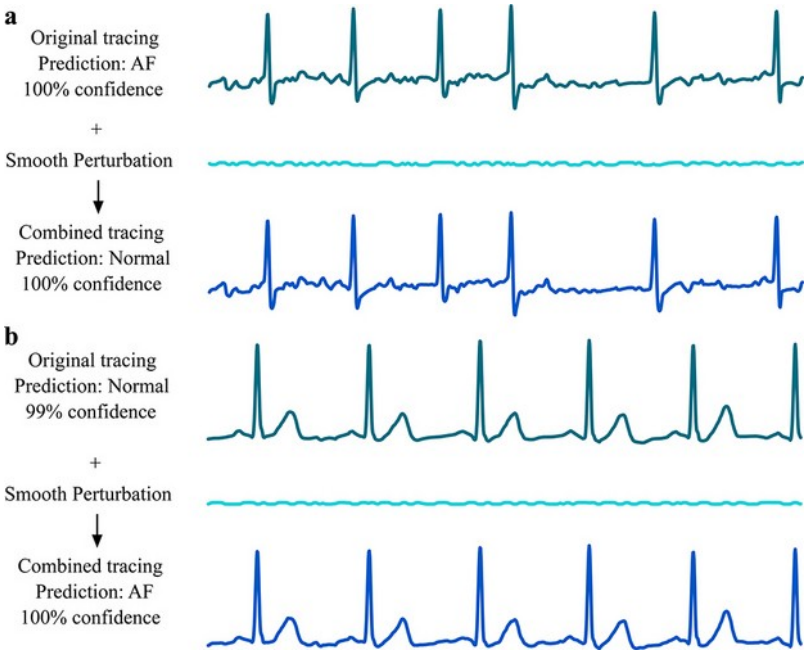

Figure 14: Exemple d'attaque adverse sur ECG avec PGD, extrait de Han et al. 2021 [109]

FGSM (*Fast Gradient Sign Method*) et PGD (*Projected Gradient Descent*)

Un réseau de neurones est une fonction composée de couches successives de neurones artificiels, où chaque neurone est relié à ceux des couches précédentes et suivantes par des poids. Lorsqu'un input, comme une image, un texte ou un vecteur, est présenté au réseau, il est transformé progressivement couche après couche, jusqu'à produire une prédiction en sortie.

On peut calculer, pour chaque input, comment la sortie du réseau varie si l'on modifie légèrement cet input. Plus précisément, pour chaque composante de l'input, par exemple chaque pixel d'une image ou chaque valeur d'un vecteur, on peut calculer la dérivée de la sortie du réseau par rapport à cette composante. Ces dérivées indiquent à la fois dans quelle direction la prédiction va évoluer, positivement ou négativement, et dans quelle proportion. L'ensemble de ces dérivées constitue ce qu'on appelle le **gradient du réseau par rapport aux données entrantes**.

Ce gradient est utilisé à l'entraînement du réseau pour ajuster les poids du modèle, mais il peut aussi être utilisé pour modifier directement l'input de manière à influencer la sortie du réseau. C'est le principe fondamental des attaques adverses FGSM [100] et PGD [110], en exploitant le gradient, on peut calculer comment perturber légèrement l'input pour modifier fortement la prédiction.

FGSM (*Fast Gradient Sign Method)* il s'agit d'une méthode simple qui consiste à faire un seul pas dans la direction du signe du gradient, en ajoutant une petite perturbation ε à chaque composante de l'input. Le but est de maximiser l'erreur du modèle avec un minimum de modification visible. Cela donne :

$$\mathbf{x}_{\text{adv}} = \mathbf{x} + \varepsilon \cdot \text{sign}\left(\nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}, y)\right)$$

- $\mathbf{x}$: input d'origine
- $\mathbf{x}_{\text{adv}}$: input après attaque adverse
- ε : amplitude de la perturbation
- $\mathcal{L}(\mathbf{x}, y)$: fonction de perte
- $\nabla_{\mathbf{x}} \mathcal{L}$: gradient de la perte par rapport à l'input
- $\text{sign}(.)$: direction du gradient

FGSM est ainsi l'une des premières méthodes proposées pour générer des attaques adverses, et reste largement utilisée comme approche de référence en raison de sa simplicité et de sa rapidité. Toutefois, son efficacité est facilement limitée face à des défenses telles que l'*adversarial training* (voir ci-dessous), car une seule mise à jour du gradient ne permet pas toujours d'explorer suffisamment l'espace des perturbations possibles.

PGD (*Projected Gradient Descent*) est une généralisation plus puissante de FGSM. Elle applique plusieurs itérations successives de petites perturbations calculées à partir du gradient, tout en s'assurant que l'input modifié reste dans une zone autorisée autour de l'input d'origine. Cela permet de produire des attaques adverses plus efficaces, tout en restant imperceptibles à l'œil humain.

$$\mathbf{x}_{t+1} = \Pi\left(\mathbf{x}_t + \varepsilon \cdot \operatorname{sign}\left(\nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}_t, y)\right)\right)$$

- $\mathbf{x_t}$: input perturbé à l'itération t
- $\mathbf{x_{t+1}}$: input mis à jour après une étape d'attaque
- ε : taille du pas, c'est l'amplitude de la perturbation à chaque itération
- $\mathcal{L}(\mathbf{x_t}, y)$: fonction de perte entre la prédiction sur $\mathbf{x_t}$ et la vraie classe y
- $\nabla_{\mathbf{x}} \mathcal{L}$: gradient de la perte par rapport à l'input
- $\operatorname{sign}(.)$: direction du gradient
- $\Pi(\cdot)$: projection pour que l'input reste dans une zone proche de l'input d'origine

PGD est considéré comme l'une des attaques adverses les plus fortes, car il explore l'espace des perturbations de manière itérative et plus exhaustive que FGSM. C'est pourquoi il est souvent utilisé comme référence dans les travaux de robustesse, au point d'être qualifié de *universal first-order adversary [111]*. Toutefois, cette puissance accrue a un coût : l'attaque est plus coûteuse en calcul, car elle nécessite plusieurs passes de gradient au lieu d'une seule.

Adversarial training

L'*adversarial training* est la méthodes la plus étudiée pour améliorer la robustesse des modèles de ML face aux attaques adverses. Le principe repose sur l'idée d'entraîner le modèle non seulement avec les données d'origine, mais également avec des exemples perturbés générés par des attaques adverses, comme FGSM ou PGD. L'objectif est que le modèle apprenne à reconnaître correctement même des inputs modifiés, en intégrant dans son processus d'entraînement la résistance aux perturbations.

Concrètement, à chaque itération d'entraînement, on génère des exemples adversariaux à partir du batch courant de données, puis on entraîne le modèle sur un mélange d'exemples propres et perturbés. Cela peut se formaliser comme un problème min-max :

$$\min_{\theta} \ \mathbb{E}_{(x,y)} \left[\max_{\delta \in S} \ \mathcal{L}\big(f_\theta(x+\delta), y\big) \right]$$

où δ est une perturbation contrainte à rester dans un espace admissible S, par exemple une boule L_p, $\mathcal{L}$ est la fonction de perte, et θ les paramètres du modèle.

L'*adversarial training* a montré une efficacité pour renforcer les modèles, même contre les attaques de type PGD, mais il souffre de certaines limites :

- augmentation du temps de calcul, car il faut générer des attaques pendant l'entraînement
- risque de sur-adaptation à un type d'attaque particulier
- parfois baisse de performance sur les données propres.

En bioinformatique, plusieurs études ont déjà montré l'intérêt pratique de l'*adversarial training*. Par exemple, il a été utilisé pour renforcer la robustesse et améliorer l'interprétabilité de modèles appliqués à des données de *single-cell* RNA-seq, permettant une meilleure identification des mécanismes cellulaires sous-jacents [112]. Dans le domaine de l'imagerie oncologique, l'*adversarial training* a permis de stabiliser les performances de réseaux convolutifs vulnérables aux attaques, augmentant la précision de classification de lésions cancéreuses qui chutait drastiquement sous attaques adverses [113]. Enfin, en neuro-imagerie, où il a permis de maintenir une bonne précision même face à des attaques capables de faire effondrer la performance des modèles non protégés [114].

Peu d'attaques adverses adaptées au scRNA-seq

Les attaques adverses ont été développées dans divers domaines tels que la reconnaissance d'images, le traitement du langage naturel, l'analyse d'enregistrements ECG, et d'autres. Elles ont démontré leur efficacité pour explorer les limites de robustesse des modèles et pour mieux comprendre leur fonctionnement interne. Toutefois, leur application dans le domaine du scRNA-seq reste très limitée.

Une étude a proposé une méthode d'attaque pour les données scRNA-seq [112]. Leur approche repose sur les méthodes FGSM et PGD. Ces méthodes nécessitent l'accès au gradient du modèle, ce qui suppose deux contraintes : que le classifieur soit basé sur des réseaux de neurones, et que le modèle soit ouvert pour qu'on ait accès soit aux poids soit au gradient, dit « *white box* ». Or, dans la pratique, les classifieurs utilisés sur des données transcriptomiques sont très variés [115]. Ils incluent des modèles de type *random forest* [116], SVM [117], *XGBoost* [118], des modèles statistiques [119] et autres. Nous avons donc développé de nouvelles méthodes d'attaques adverses adaptées aux classifieurs sur données scRNA-seq, qu'ils soient *white box* ou non, qu'ils soient à base de réseaux de neurones ou non. Ces méthodes sont conçues pour être applicables à tout type de classifieur, et ont été implémentées dans un package *R*, « *adverSCarial* » disponible sur *Bioconductor* : https://bioconductor.org/packages/adverSCarial/ .

Attaques adverses sur données scRNA-seq

Nous avons développé 3 types d'attaques que nous décrivons ci-dessous [120].

Single-gene attack

Le *single-gene attack* est une modification d'un seul gène des cellules d'un cluster qui provoque la modification de sa classification. La fonction *advSingleGene* est conçue pour identifier toutes les attaques de ce type sur un cluster cellulaire, en fonction d'un type de modification donné. Sur un ordinateur aux performances moyennes, la plupart des classifieurs nécessitent plus de 10 secondes pour classifier un cluster de cellules, c'est à dire prédire son type cellulaire. Considérons un jeu de données scRNA-seq contenant 20 000 gènes, une approche par force brute testant chaque gène individuellement prendrait environ 55 heures. À la place, la fonction utilise une stratégie en deux étapes : elle divise d'abord la liste des gènes en 100 groupes, c'est le paramètre par défaut, et applique ensuite une recherche dichotomique dans chaque groupe pour identifier les gènes "attaquables", et isole finalement un gène unique responsable du changement de classification. La recherche d'un élément dans une liste de n éléments peut être réalisée en un temps logarithmique $\log_2(n)$: si le jeu de données contenait un gène par attaque *single-gene*, $100 + \log_2(20\,000) = 114$ essais seraient nécessaires, soit moins de 20 minutes de calcul.

```
Algorithme du single-gene attack


Input:
 matrix: rna_expr
 list: cells_type
 string: cell_type_target
 function: classifieur
 object: modification_type
 int: init_split_size

Output:
 list: results_attack_genes

Begin
    results_attack_genes <- empty list
    gene_groups <- groups of genes based on init_split_size

    For each group in list_gene_groups do
        results_attack_genes <- dichotSingleGenes(
                results_attack_genes,
                rna_expr,
```

```
                    cells_type,
                    cell_type_target,
                    function_classifieur,
                    modification_type,
                    genes)

        Return results_attack_genes
    End


    Function dichotSingleGenes
    Input:
     list: results_attack_genes
     matrix: rna_expr
     list: cells_type
     string: cell_type_target
     function: classifieur
     object: modification_type
     list: genes

    Output:
     list: results_attack_genes

    Begin
            matrix: modified_rna_expr <- modified rna_expr matrix on genes with
the modification modification_type

            string: new_classification <- classification made by classifieur on
modified_rna_expr

            If new_classification equals cell_type_target then
                    return results_attack_genes

            If length of genes is 1 then
                    results_attack_genes[genes[0]] <- new_classification
                    return results_attack_genes

            list: genes_half1 <- one half of genes

            list: genes_half2 <- other half of genes
```

```
        results_attack_genes <- dichotSingleGenes(
                results_attack_genes,
                rna_expr,
                cells_type,
                cell_type_target,
                function_classifieur,
                modification_type,
                genes_half1)

        results_attack_genes <- dichotSingleGenes(
                results_attack_genes,
                rna_expr,
                cells_type,
                cell_type_target,
                function_classifieur,
                modification_type,
                genes_half2)
 return results_attack_genes
End
```

Max-change

Cette attaque, implémentée dans la fonction *advMaxChange*, cherche à modifier le plus grand nombre possible de gènes des cellules d'un cluster, sans modifier la classification de ce cluster. Il renvoie l'ensemble des gènes, donc l'ensemble complémentaire de ces gènes correspond aux gènes qu'on ne peut alors plus modifier sans modifier la classification. Là encore, une approche par force brute atteindrait rapidement une barrière de complexité (pour un jeu de données contenant n gènes : complexité en $O(2^n)$). À la place, la méthode modifie tous les gènes d'un cluster cellulaire, puis divise les gènes en deux sous-ensembles si la classification change. Si la modification d'un sous-ensemble ne modifie pas la classification, celui-ci est combiné à un ensemble de gènes précédemment identifiés comme "sans effet sur la classification", sinon, la subdivision continue. Ce processus se répète récursivement jusqu'à identification du plus grand ensemble de gènes modifiables sans impact. Cette liste complémentaire constitue alors la signature du cluster cellulaire.

```
Algorithme du max-change attack

Input:
```

```
matrix: rna_expr
list: cells_type
string: cell_type_target
function: classifieur
object: modification_type

Output:
list: results_attack_genes

Begin
results_attack_genes <- empty list
        list: genes <- list of all gene names from matrix_rna_expr

        results_attack_genes <- dichotMaxChange(
        results_attack_genes,
        rna_expr,
        cells_type,
        cell_type_target,
        classifieur,
        modification_type,
        genes)

return results_attack_genes
End

Function dichotMaxChange
Input:
list: results_attack_genes
matrix: rna_expr
list: cells_type
string: cell_type_target
function: classifieur
object: modification_type
list: genes

Output:
list: results_attack_genes

Begin
```

```
    matrix: modified_rna_expr <- rna_expr matrix with genes values modified with
the modification_type on the cell_type_target cells

    string: new_classification <- classification of the cell_type_target cells of
modified_rna_expr with classifieur

        If new_classification equals cell_type_target then
        list: concatenated_genes <- concatenate genes and results_attack_genes

        matrix: modified_rna_expr <- rna_expr matrix with concatenated_genes
values modified with the modification_type on the cell_type_target cells

        string: new_classification <- classification of the cell_type_target cells of
modified_rna_expr with classifieur

    If new_classification equals cell_type_target then
        return concatenated_genes

    list: genes_half1 <- one half of genes

    list: genes_half2 <- other half of genes

    results_attack_genes <- dichotMaxChange(
        results_attack_genes,
        rna_expr,
        cells_type,
        cell_type_target,
        classifieur,
        modification_type,
        genes)

    results_attack_genes <- dichotMaxChange(
        results_attack_genes,
        rna_expr,
        cells_type,
        cell_type_target,
        classifieur,
        modification_type,
        genes)
```

```
    return results_attack_genes
End
```

Cluster Gradient Based (CGD)

La méthode CGD est implémentée dans la fonction *advCGD.* Il s'agit d'une fonction basée sur un gradient estimé, fonctionnant donc sur tout type de classifieurs. Le processus est itératif, il modifie les gènes individuellement, et il est pensé pour être appliqué sur un cluster entier, bien qu'il puisse être appliqué sur une cellule seule. La matrice de l'ensemble des cellules par l'ensemble des gènes est définie par X, le sous-ensemble de la matrice correspondant au cluster m est X_{clust_m} et le vecteur d'expression des gènes de la cellule i est x_i .

$$X = (x_i)_{i \leq \|Cells\|}$$
$$x_i = (x_{ij})_{j \leq \|Genes\|}$$

$\hat{y}_{x_i} = (score_k)_{k \in CellTypes}$ est le vecteur de prédiction pour la cellule i , un score pour chaque type cellulaire.

$\hat{y}_{X_{\text{clust}_m}} = \sum_{i \in \text{clust}_m} e_k$ est la prédiction pour le cluster m avec $k = \arg\max_k (\hat{y}_{x_i})_k$

En premier la méthode infère le gradient de $\hat{y}$ pour chaque cellule de X_{clust_m} par rapport à l'expression d'un gène j', par différence finie. L'amplitude de la perturbation apportée est ajustée par deux paramètres α et ε :

$$\nabla^{fd}_{j'} \hat{y}_{x_i} = \hat{y}_{f_{j'}(x_i,\alpha,\varepsilon)} - \hat{y}_{x_i} \text{ , avec}$$
$$f_{j'}(x_i, \alpha, \varepsilon) = x_{ij}(1 - \delta_{jj'}) + ((1 + \alpha \cdot \text{sign}(x_{ij}))x_{ij} + \varepsilon)\, \delta_{jj'}$$

A partir du résultat d'un classifieur pour une cellule nous définissons $ctype1_x$ et $ctype2_x$ le plus probable et le second plus probable type cellulaire de la cellule x.

A partir du résultat d'un classifieur pour un cluster nous définissons $ctype1_{X_{clust_m}}$ et $ctype2_{X_{clust_m}}$ le plus probable et le second plus probable type cellulaire du cluster X_{clust_m}.

CGD suit le gradient estimé pour diminuer l'écart de score entre $ctype1_{X_{clust_m}}$ et $ctype2_{X_{clust_m}}$ en modifiant un à un les gènes de chaque cellule de X_{clust_m} jusqu'à ce que $ctype2_{X_{clust_m}}$ devienne le nouveau type cellulaire le plus probable selon le classifieur.

L'algorithme retourne alors la matrice modifiée $X^{adv}_{clust_m}$, avec l'amplitude de modification de chaque étape définie par les même paramètres multiplicatifs, α, et additifs, ε :

$$\text{slope} = \left(\nabla^{fd}_{j'} \hat{y}_{x_i}\right)_{\text{ctype2}_{X^n_{\text{clust}_m}}} - \left(\nabla^{fd}_{j'} \hat{y}_{x_i}\right)_{\text{ctype1}_{X^n_{\text{clust}_m}}}$$

$$\left(x_i^{n+1}\right)_j = x_{ij}(1 - \delta_{jj'}) + ((1 + \alpha \cdot \text{sign}(x_{ij}) \cdot \text{sign}(\text{slope}))x_{ij} + \varepsilon \cdot \text{sign}(\text{slope}))\, \delta_{jj'}$$

$$X^{\text{adv}}_{\text{clust}_m} = \left(x_i^{n+1}\right)_{i \in \text{clust}_m}$$

Pour réduire le nombre d'itérations nécessaires pour l'attaque, et parce que chaque étape cible un gène différent, nous privilégions les gènes les plus susceptibles de modifier la prédiction du classifieur. La fonction *getSignGenes* peut être utilisée pour ordonner les gènes selon leur *p-values* après *t-test*, sur chaque paire de clusters possibles, tout en s'assurant que chacune des paires de clusters soit équitablement représentée. Pour éviter les calculs inutiles nous gardons $x_i^{n+1} = x_i^n$ quand $ctype1_{x_i^n} = ctype2_{X^n_{clust_m}}$ ou quand $slope = 0$, c'est à dire quand la modification du gène ne modifie pas la prédiction du classifieur.

Modifications pour *single-gene* et *max-change*

Les différentes fonctions d'attaque proposées dans le package nécessitent un paramètre supplémentaire : la méthode de modification de l'expression génique à appliquer. Plusieurs types de modifications sont disponibles, pour refléter différentes situations biologiques, des aléas techniques ou des actions manuelles.

Deux modifications, appelées *perc1* et *perc99*, simulent respectivement l'inactivation et l'activation d'un gène. Elles remplacent toutes les valeurs d'expression d'un gène dans toutes les cellules d'un cluster par le 1er ou le 99e percentile.

Il est aussi possible de faire varier l'expression de gènes en la diminuant, *decile-N*, ou en l'augmentant, *decile+N*, de façon progressive selon les déciles, avec un niveau N allant de 1 à 10.

La modification *random* ajoute un bruit aléatoire uniforme entre la valeur minimale et la valeur maximale d'expression. Cela simule des variations naturelles ou des erreurs de mesure.

Avec la fonction custom, l'utilisateur peut aussi définir ses propres types de modifications.

Enfin, certaines erreurs techniques ou de traitement des données sont simulées en ajoutant des valeurs aberrantes. Les paramètres *positive-aberrant* et *negative-aberrant* créent des valeurs extrêmes en ajoutant ou en retirant 10 000 fois la valeur maximale du jeu de données à l'expression génique.

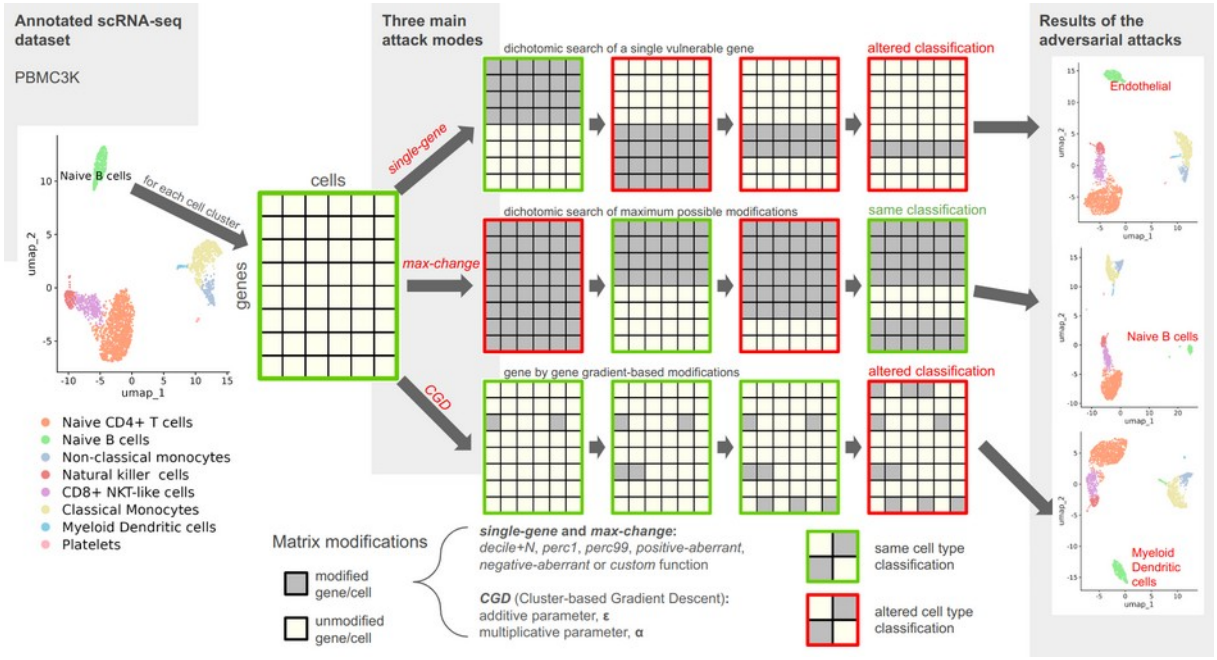

Figure 15: ***Présentation des fonctionnalités d'adverSCarial.*** *Exécution des algorithmes d'attaque single-gene, max-change et CGD (descente de gradient basée sur les clusters). Représentations UMAP du jeu de données PBMC3k annoté avec scType, avant et après les attaques single-gene, max-change ou CGD ciblant le cluster de cellules B naïves. Extrait de Fievet et al. 2025 [120]*

Package *R Bioconductor adverSCarial*

Le package *adverSCarial* a été développé pour aider à l'analyse de la robustesse des classifieurs *single-cell* face à des attaques adverses, des perturbations contrôlées de l'expression génique. Il implémente plusieurs types d'attaques adaptées aux données scRNA-seq, et permet de les appliquer sur n'importe quel classifieur, notamment ceux à base de marqueurs, SVM, *decision trees*, *random forest* ou réseaux de neurones. Le code source est disponible sur *GitHub* https://github.com/GhislainFievet/adverSCarial, et le package est référencé sur *Bioconductor* (version ≥ 3.18).

Les arguments principaux communs aux fonctions clés du package sont les suivants :

- *exprs* : une matrice d'expression (*data.frame*, matrix ou DelayedMatrix) ou un objet *SingleCellExperiment* ou *Seurat*.
- *clusters* : un vecteur indiquant le type cellulaire de chaque cellule
- *target* : le nom du cluster ou du type cellulaire à attaquer
- *classifier* : une fonction retournant, pour un cluster donné, le type prédit et un score

- *advMethod* ou *modifications* : le type de perturbation, par exemple "*perc1*", "*perc99*", "*decile+5*", ou une fonction personnalisée.
- *exclGenes* : liste de gènes à ne pas prendre en compte
- *argForClassif* : type de l'objet à fournir à la fonction *classifier*, défaut : "*data.frame*"

Fonctions principales

advSingleGene()

Implémente la méthode *single-gene*, identifie les gènes dont la modification individuelle suffit à changer la classification du cluster cible. Cette fonction effectue une recherche dichotomique pour détecter les attaques.

Arguments spécifiques :

- *firstDichot* : nombre initial de groupes pour la recherche, défaut : 100
- *maxSplitSize* : taille maximale du batch après dichotomie, défaut : 1
- *returnFirstFound* : TRUE si on veut que la fonction s'arrête après avoir trouvé une attaque, FALSE si on veut que la fonction trouve toutes les attaques possibles.

Exemple d'utilisation :

```
advSingleGene(      exprs = myMat,
                    clusters = myClusters,
                target = "B_naive",
                    classifieur = myclassifieur,
                    advMethod = "perc99")
```

advMaxChange()

Implémente la méthode *max-change*, cherche le plus grand ensemble de gènes pouvant être modifiés sans changer la classification. Elle utilise également une recherche dichotomique.

Arguments spécifiques :

- *maxSplitSize* : taille maximale du batch après dichotomie, défaut : 1

Exemple d'utilisation :

```
advMaxChange(     exprs = myMat,
                  clusters = myClusters,
                  target = "CD4_T",
                        classifieur = myclassifieur,
                        advMethod = "perc1")
```

Les fonctions de synthèse *singleGeneOverview()* et *maxChangeOverview()* permettent d'appliquer les fonctions précédentes sur l'ensemble des clusters du jeu de données, avec un *maxSplit* = 100, afin d'obtenir un tableau récapitulatif des vulnérabilités rapidement.

advCGD()

Implémente la méthode CGD, effectue une attaque par descente de gradient estimée sur les cellules d'un cluster. Les gènes sont modifiés un par un en suivant une estimation du gradient entre la prédiction d'origine et la prédiction après modification.

Arguments spécifiques :

- *alpha* : facteur multiplicatif du gradient, défaut : 0.1
- *epsilon* : terme additif, défaut : 0
- *stopAtSwitch* : arrête dès que la classe change, défaut : TRUE. Avec l'option FALSE, la fonction devient alors utile pour faire de l'*adversarial training*.

Exemple d'utilisation :

```
advCGD(     exprs = myMat,
        clusters = myClusters,
        target = "Monocytes",
        classifieur = myclassifieur,
        alpha = 0.05,
        epsilon = 0.05)
```

On trouve dans le package d'autres fonctions plus utilitaires :

- ***advRandWalkMinChange.R* :** Implémente une marche aléatoire pour identifier une combinaison minimale de gènes et de perturbations provoquant un changement de classification. Peu scalable, à utiliser pour les petits nombres de gènes (< 10).

- ***getSignGenes.R* :** Permet d'ordonner les gènes selon leur pouvoir discriminant entre les clusters. Utilisé notamment pour sélectionner les gènes à tester dans les fonctions *advRandWalkMinChange()* ou *advCGD()*. La fonction fait en sorte que chaque duo de clusters soient représentés de façon équitable.
- ***advModifications.R*** : Fonction centrale *advModifications()* qui applique les modifications (type *perc1*, *random*, *custom*…) à une matrice d'expression.
- ***advTraining.R*** : Fonctions pour entraîner les modèles intégrés (*Random Forest*, MLP...) à utiliser comme classifieurs de test.
- ***matrixFromSCE.R*** : Transforme un objet *SingleCellExperiment* en matrice d'expression avec noms de gènes HGNC.
- ***Mclassifieur.R*** : Fournit un classifieur simple basé sur des moyennes de gènes marqueurs — utilisé comme modèle de démonstration.
- ***predictWithNewValue.R*** : Applique une ou plusieurs modifications définies à un cluster cible et retourne directement la nouvelle prédiction du modèle.
- ***sceConvertToHGNC.R*** : Modifie un objet *SingleCellExperiment* en renommant les gènes avec la nomenclature HGNC.

Étude générale sur plusieurs classifieurs et plusieurs jeux de données

Ici nous illustrons l'utilisation d'*adverSCarial* en évaluant la robustesse de cinq classifieurs scRNA-seq, chacun représentant une catégorie différente d'algorithme d'apprentissage automatique : *scType* (*marker*-based) [121], *CHETAH* (*hierarchical clustering*) [122], *scAnnotatR* (SVM) [117], *scRF* (RF) et *scMLP* (MLP). Nous avons testé ces classifieurs sur 5 jeux de données scRNA-seq, *PBMC3k*, *LIVER10k*, *AXILLA10k* et *KIDNEY10k*, voir description plus bas.

Méthode

Classifieurs scRNA-seq

Pour *scType*, *CHETAH* et *scAnnotatR*, nous avons préparé et annoté les jeux de données en suivant les procédures décrites dans la documentation de chaque outil. Manuel, tutoriel ou fichier *readme*, disponibles sur *Bioconductor* ou dépôts *GitHub*.

scType

La méthode standard nécessite de charger une base de données contenant, pour chaque type cellulaire, des marqueurs positifs et négatifs prédéfinis. Pour chaque cellule, *scType* calcule un vecteur de scores et attribue à chaque cluster le type cellulaire le plus fréquent.

CHETAH

Le *workflow* par défaut utilise des objets *SingleCellExperiment*. Dans *adverSCarial*, la fonction *CHETAHclassifieur* prend en entrée deux objets *SingleCellExperiment* : l'un pour annoter l'ensemble d'entraînement, l'autre pour le jeu de test.

Le paramètre de seuil de confiance de *CHETAH*, *tresh*, a été fixé à 0.01. Pour chaque cellule, cet outil renvoie le type cellulaire prédit et un score.

scAnnotatR

Cet outil construit un classifieur binaire intermédiaire basé sur un algorithme SVM, pour chaque type cellulaire. Chaque modèle prédit si une cellule appartient ou non à un type donné. Ensuite, ces prédictions sont combinées en un seul modèle, qui donne un vecteur de scores représentant la probabilité d'appartenance à chaque type cellulaire, pour chaque cellule.

Le SVM nécessite une liste de gènes discriminants entre les clusters. Dans *adverSCarial*, la fonction *getSignGenes* permet de classer les gènes les plus importants pour chaque comparaison de 2 clusters. Nous avons utilisé les 20 premiers gènes de ce classement comme entrée pour *scAnnotatR*.

scRF et *scMLP* ont été spécifiquement développés pour cette étude. Nous avons construit les modèles selon les architectures *random forest* et m*ulti-layer perceptron*, et nous les avons entraînés avec des jeux d'entraînement à classes équilibrées, c'est à dire avec le même nombre de cellules pour chaque type cellulaire dans les jeux d'entraînement.

scRF

Ce classifieur basé sur *random forest* a été implémenté avec le package *randomForest* (version 4.7.1.1). Le modèle a été construit avec les paramètres par défaut :

- 500 arbres (*ntree* = 500)
- Le nombre de variables sélectionnées aléatoirement à chaque division (*mtry*) a été calculé comme la racine carrée du nombre de variables d'entrée :
  - *PBMC3k* : 117
  - *AXILLA10k* : 159
  - *KIDNEY10k* : 181
  - *LIVER10k* : 160
- La variable de réponse a été encodée en *one-hot* pour permettre une classification multiclasse.

scMLP

Ce modèle MLP a été construit avec la bibliothèque *Keras* pour *R* (version 2.15.0).

Architecture du modèle :

- 5 couches entièrement connectées
- 1ère couche : 128 neurones, activation *ReLU*
- Suivie d'une couche *dropout* (taux 0,5) pour éviter le surapprentissage
- Puis une couche de 64 neurones, activation *ReLU*
- Suivie d'une autre *dropout* (taux 0,5)
- Couche de sortie : nombre de neurones égal au nombre de types cellulaires, avec activation *softmax* pour une classification multiclasse
- Fonction de perte : *categorical cross-entropy*.
- Optimisation avec *Adam optimizer*
- Entraînement avec :
  - 20 *epochs*
  - Taille de *batch* : 32
  - 20 % des données utilisées pour la validation.

Jeux de données scRNA-seq

Quatre jeux de données scRNA-seq ont été utilisés dans cette étude :

PBMC3k

Jeu de données souvent utilisé dans les *benchmarks* en scRNA-seq. Il contient 2 700 cellules mononucléées du sang périphérique (PBMC), ainsi que quelques plaquettes et cellules dendritiques résiduelles, toutes issues d'un donneur sain. Les cellules de mauvaise qualité ont déjà été filtrées (jeu de données scRNA-seq produit avec *Cell Ranger* v1.1.0, *Illumina NextSeq 500*, *10x Genomics*).

PBMC3k contient des types cellulaires bien distincts, comme les cellules B et les cellules T CD4, mais aussi d'autres ayant des profils transcriptomiques proches, comme les monocytes classiques (CD14+) et non classiques (FCGR3A+).

AXILLA10k

Ce jeu contient 10 623 cellules annotées provenant de biopsies de métastases d'un cancer du sein prélevées dans un ganglion lymphatique axillaire chez une patiente [123]. On y trouve des cellules T, des cellules endothéliales des vaisseaux sanguins, des cellules musculaires lisses, des fibroblastes, des macrophages et des cellules cancéreuses.

LIVER10k

Il contient 10 016 cellules annotées issues de biopsies de métastases du même type de cancer, mais cette fois prélevées dans le foie [123]. On y trouve des cellules musculaires lisses des vaisseaux, des cellules endothéliales, des macrophages, des cellules cancéreuses, des cellules NK T matures et des monocytes. Les types cellulaires très peu représentés, 7 cellules endothéliales et 9 fibroblastes, ont été retirés.

KIDNEY10k

Ce jeu comprend 10 790 cellules annotées provenant de néphrons fœtaux issus de 6 donneurs différents [124]. Il contient des cellules épithéliales du tubule proximal, des cellules rénales diverses, des cellules de l'anse de Henle, des cellules mésenchymateuses et des podocytes.

Selon le modèle de classification utilisé, plusieurs jeux de données ont été préparés à partir de ces sources, en suivant la procédure standard de *Seurat* (v5.0.3). Cette procédure inclut des étapes pour filtrer, normaliser, clusteriser et annoter les cellules [125].

Les étapes de contrôle qualité et de pré-traitement étaient spécifiques à chaque jeu de données et respectaient les seuils d'inclusion/exclusion utilisés par les auteurs d'origine. Par exemple :

Les cellules avec trop peu ou trop de gènes exprimés ont été retirées :

- <200 ou >2500 gènes pour *PBMC3k*
- <500 pour *KIDNEY10k*
- <8000 pour *LIVER10k* et *AXILLA10k*

Les cellules avec une forte proportion d'ARN mitochondrial ont été exclues :

- 5 % pour *PBMC3k*
- 20 % pour *KIDNEY10k*
- 50 % pour *LIVER10k* et *AXILLA10k*

Les données brutes ont ensuite été log-normalisées et *scaled*. Les gènes variables ont été sélectionnés avec la fonction *FindVariableGenes* de *Seurat* pour *PBMC3k* et *KIDNEY10k*, ou avec le *workflow Scanpy* (v1.7.2) pour *AXILLA10k* et *LIVER10k*, selon les publications d'origine.Les algorithmes de réduction de dimension PCA et UMAP ont ensuite été appliqués.

Nous avons procédé à ces étapes de pré-traitement pour *PBMC3k*, les autres jeux de données ont été téléchargés déjà traités.

Pour chaque jeu de données nous avons défini plusieurs sous-jeux de données :

- ***Undivided set*** : contient toutes les cellules annotées.
- ***Training set*** : contient la moitié des cellules, en gardant les proportions de chaque type cellulaire.

- ***Test/validation set*** : contient l'autre moitié (non utilisée pour l'entraînement), en respectant aussi les proportions.
- ***Balanced training set*** : Utilisés avec les classifieurs *scRF* et *scMLP*. Ils contiennent 100 cellules par type cellulaire, choisies au hasard. Si un type a moins de 100 cellules, des cellules supplémentaires ont été générées par *bootstrap* sur les valeurs d'expression des gènes.

Annotation

PBMC3k a été annoté en regroupant les cellules avec la fonction *FindClusters* puis chaque cluster a été identifié grâce à des gènes marqueurs. *AXILLA10k*, *LIVER10k* et *KIDNEY10k* étaient déjà annotés dans les jeux de données d'origine.

Étude préliminaire sur *PBMC3k*

Nous avons réalisé une analyse du classifieur *marker*-based scType en l'appliquant au test set de *PBMC3k*. En utilisant *adverSCarial* et l'attaque *single-gene*, nous avons étudié si la modification d'un seul marqueur pouvait changer le résultat du classifieur, et dans quelle mesure.

Les modifications *perc1* et *decile+5* se sont révélées les moins perturbatrices, avec très peu d'attaques possibles selon les clusters, entre 0 à 8. À l'inverse, les modifications *negative-aberrant* et *positive-aberrant* ont provoqué de nombreuses erreurs de classification, de 29 à 37 pour le *negative-aberrant* et de 135 à 149 pour les *positive-aberrant*. La modification *perc99* a eu un impact modéré de 0 à 64.

La vulnérabilité globale aux attaques *single-gene* dépendait aussi du type cellulaire :

Les plaquettes étaient les plus résistantes, sans aucune erreur causée par les modifications *perc1* ou *perc99*.

La figure ci-dessus de, présentation des fonctionnalités d'*adverSCarial*, illustre un changement de classification : le cluster des cellules B naïves est alors classé comme cellules endothéliales après activation du gène *ICAM1* avec la modification *perc99*.

Avec l'attaque *max-change*, nous avons cherché à quel point on peut modifier les gènes sans que la classification ne soit modifiée. On voit par exemple que le cluster des cellules B naïves est resté correctement classé même après avoir activé 99,5 % de tous les gènes avec la modification perc99.

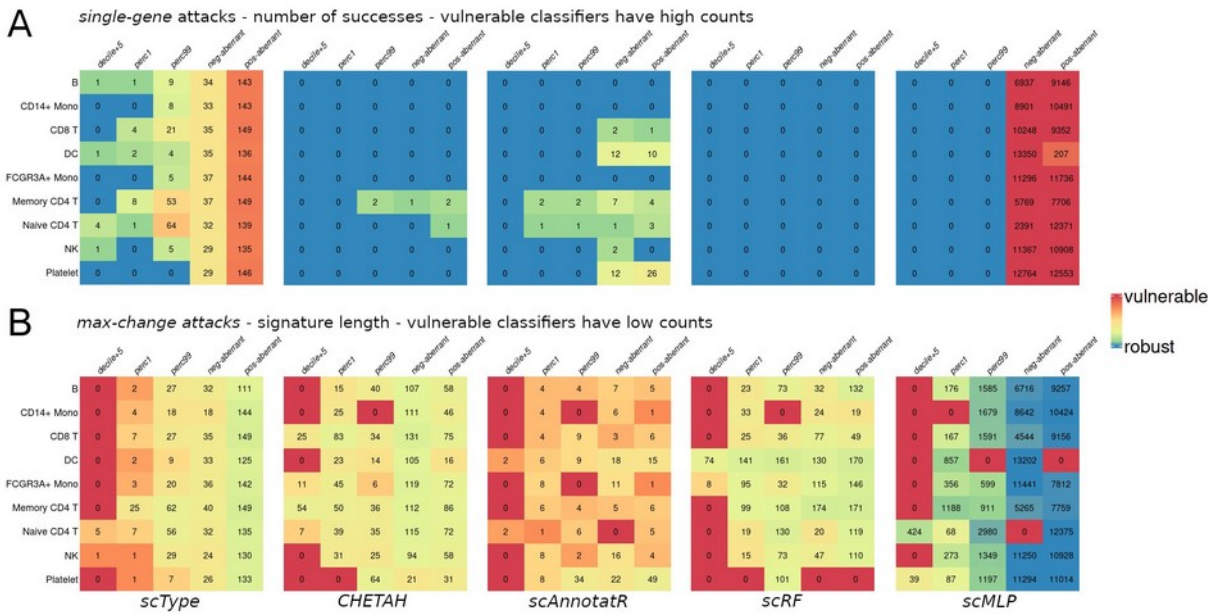

Figure 16: Vulnérabilité de cinq classifieurs scRNA-seq face aux attaques adverses single-gene et max-change. Cinq types de modifications sont appliqués, un par un, sur les neuf types cellulaires du jeu de test. (A) Heatmaps du nombre d'attaques single-gene réussies (B) Heatmap montrant la longueur de la signature des attaques max-change. Pour les attaques single-gene, un nombre supérieur à 0 indique une vulnérabilité du classifieur. Pour les attaques max-change, un classifieur est considéré comme robuste si le nombre est élevé.

Enfin, nous avons observé une corrélation globale entre la longueur des signatures détectées (la longueur de l'ensemble complémentaire des gènes du *max-change*) et la vulnérabilité aux attaques *single-gene* avec un coefficient de Spearman de ρ = 0,66.

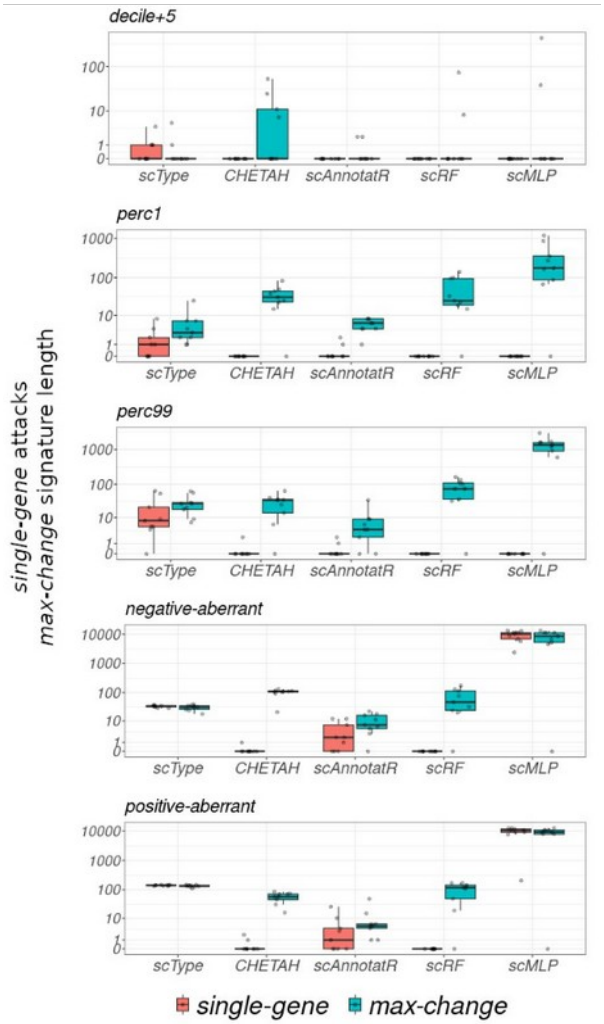

Figure 17: Boxplots résumant l'impact de chaque type de modification, en cumulant les résultats pour chaque type cellulaire, et ce pour les deux types d'attaques, classé par classifieur. Les données sont représentées en échelle logarithmique (log10).

Étude sur 5 classifieurs et 4 jeux de données

Nous avons ensuite analysé la robustesse des cinq classifieurs sur les quatre jeux de données de scRNA-seq.

Trois de ces classifieurs proviennent de la littérature existante : *scType*, *CHETAH* et *scAnnotatR*. Et nous avons entraîné deux nouveaux classifieurs : *scRF* basé sur *random forest*, et *scMLP* basé sur un *MLP*.

Nous avons testé ces modèles sur quatre jeux de données distincts : *PBMC3k*, *LIVER10k*, *AXILLA10k* et *KIDNEY10k*, chacun ayant été divisé en jeux d'entraînement et de test.

Vulnérabilité à *single-gene* et *max-change*

Les attaques *single-gene* appliquées aux cinq classifieurs ont révélé des niveaux de sensibilité variables selon la nature de l'algorithme. *scType* s'est souvent montré le plus vulnérable, tandis que *scRF* s'est révélé complètement insensible à tous les types de modifications. Dans ce mode d'attaque, la modification *decile+5* a été la moins efficace, et n'a affecté que *scType*, avec seulement sept attaques possibles (Annexe ADVERSCARIAL, docs/supplementary_data_v7.pdf, S7).

Les modifications *perc1* et *perc99* ont été plus efficaces, avec respectivement 185, 2 et 6 attaques réussies sur *scType*, *CHETAH* et *scAnnotatR*.

Les modifications *positive-aberrant* et *negative-aberrant* ont entraîné un grand nombre d'erreurs : plus de 1 500 pour *scType*, 4 pour *CHETAH*, 80 pour *scAnnotatR*, et environ 200 000 pour *scMLP*.

Concernant les attaques max-change, des tendances similaires ont été observées pour tous les classifieurs, la modification *decile+5* produisait les signatures les plus courtes, tandis que les modifications *positive-aberrant* et *negative-aberrant* généraient les signatures les plus longues. Selon les combinaisons classifieur/modification, il arrivait que tous les gènes de toutes les cellules d'un cluster soient modifiés sans altérer la prédiction (signature = 0), y compris pour les modifications aberrantes extrêmes.

Dans l'ensemble, en comparant les classifieurs entre eux et en tenant compte de l'intensité des modifications :

- *decile+5* était une modification à faible impact
- *perc1* et *perc99* étaient de niveau moyen
- Les modifications *positive-aberrant* et *negative-aberrant* étaient à fort impact, pour les deux types d'attaques.

Les classifieurs *CHETAH* et *scRF* ont montré peu de réactions aux modifications de niveau moyen ou élevé, ce qui indique une forme de saturation de la sensibilité, propre aux algorithmes à base d'arbres : une fois qu'une valeur d'entrée dépasse un seuil défini dans un nœud de l'arbre, l'augmentation supplémentaire n'a plus d'effet.

À l'inverse, *scMLP* était peu réactif aux modifications faibles, mais extrêmement vulnérable aux fortes modifications, avec des milliers d'attaques possibles. Cet effet s'explique par l'utilisation de coefficients multiplicatifs dans les réseaux de neurones, qui amplifient la sensibilité aux valeurs extrêmes.

Détectabilité des modifications

Un aspect essentiel des attaques adverses est la perceptibilité humaine des modifications introduites.

On peut observer dans la figure ci-dessous le résultat d'attaques *single-gene* et *max-change* ciblant les cellules B du jeu de test de *PBMC3k*. L'attaque *single-gene*/*perc99* est sur le gène *ICAM1* et l'attaque *max-change*/*perc99* a une signature de

27 gènes. On observe que le résultat de ces attaques conserve une topologie de clusters similaire sur les UMAP.

Nous avons choisi une représentation par *heatmap* pour pouvoir apprécier visuellement l'ampleur des modifications apportées par les attaques. Dans ce but de mettre en lumière ces changements nous avons choisi de n'afficher que les 100 gènes les plus variables permettant de discriminer les clusters cellulaires, on peut ainsi voir les changements même sur un seul gène.

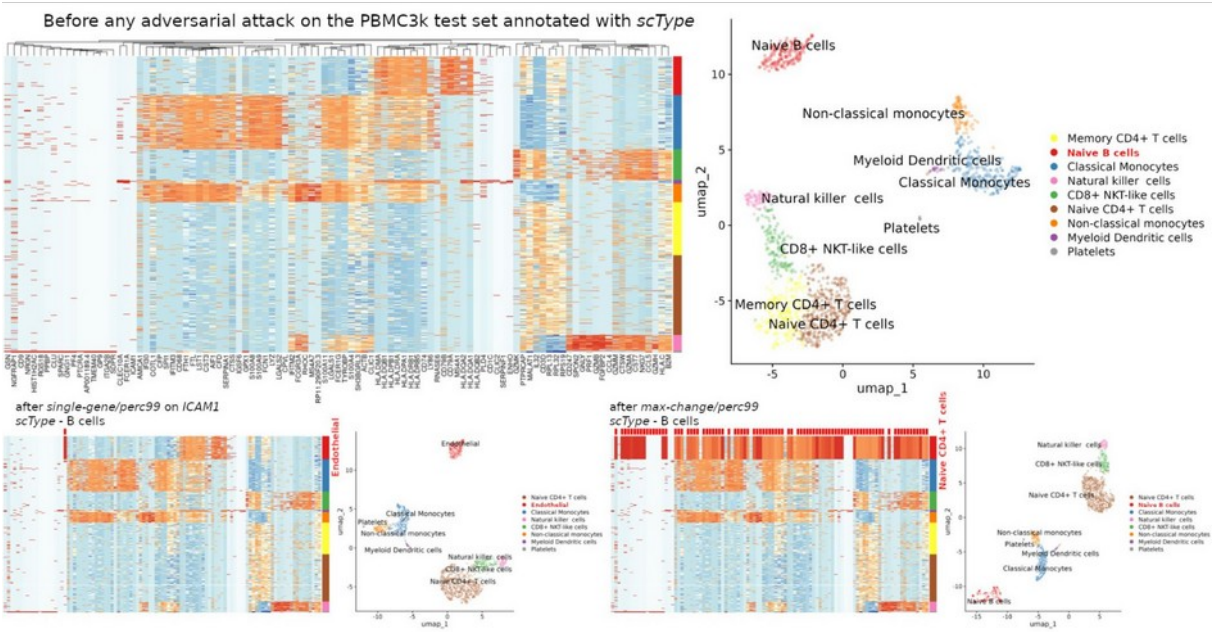

Figure 18: Visibilité des changements après attaque adverse. Les classifieurs sont entraînés et testés avec les training et test set respectivement. Au-dessus de chaque heatmap, les modifications appliquées sont indiquées par des traits rouges. (A) heatmaps avec clustering hiérarchique et représentations UMAP après classification et annotation avec scType, avant attaque (en haut), après une attaque single-gene (en bas à gauche), après une attaque max-change (en bas à droite), ciblant les lymphocytes B du jeu de données PBMC3k.

Sur d'autres jeux de données, les attaques *single-gene* ont provoqué des altérations variables de l'expression du gène ciblé, mais ont conservé les motifs visuels caractéristiques du type cellulaire (figure ci-dessous).

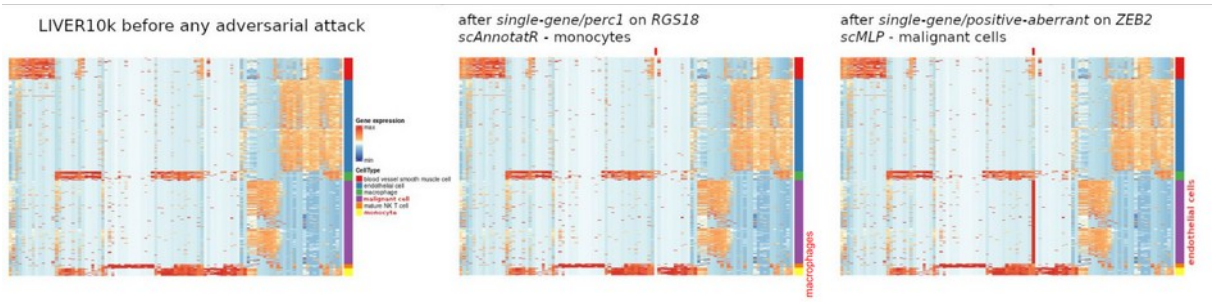

Figure 19: Heatmaps montrant les gènes les plus discriminants, avant et après différentes attaques single-gene, appliquées à plusieurs classifieurs scRNA-seq sur LIVER10k

Comme attendu, les attaques max-change avec *perc99* et *perc1* ont fortement brouillé les motifs visuels des clusters ciblés, sans pour autant changer la classification (figure ci-dessous). Pour *CHETAH* et *scRF*, les signatures des attaques *max-change* comprenaient respectivement 793 et 4506 gènes.

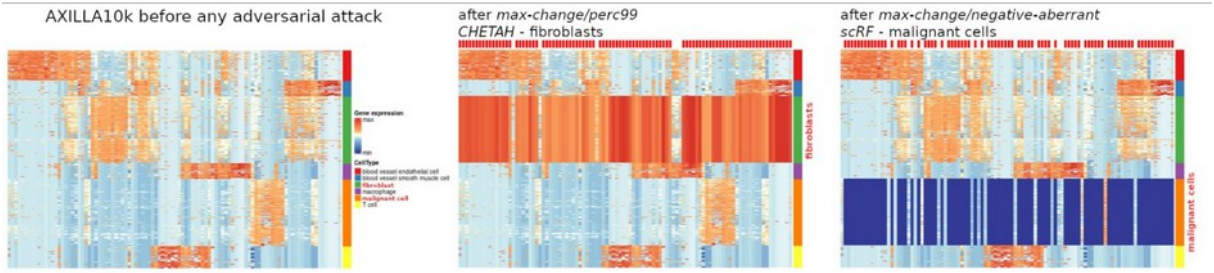

Figure 20: Heatmaps montrant les gènes les plus discriminants, avant et après différentes attaques max-change, appliquées à plusieurs classifieurs scRNA-seq sur AXILLA10k

Grâce aux paramètres α et ε, il est possible de contrôler l'intensité des modifications des attaques CGD. Des valeurs faibles comme α = ε = 0.5 produisent des modifications subtiles. Des valeurs plus élevées comme α = ε = 1 entraînent des modifications plus visibles. Cependant, avec des valeurs faibles, il faut modifier un grand nombre de gènes pour modifier la prédiction du classifieur, tandis qu'avec des valeurs plus fortes, très peu de gènes suffisent. Par exemple, l'attaque par CGD a modifié légèrement 15 gènes pour que *scAnnotatR* modifie sa classification sur le cluster des cellules épithéliales du rein, alors que 2 gènes fortement modifiés suffisaient pour le même résultat (figure ci-dessous, panel A).

Selon le classifieur, les mêmes valeurs de paramètres peuvent produire des résultats très différents :

- 1 seul gène modifié pour *scType*
- Jusqu'à 577 gènes pour *scMLP*, avec α = ε = 1 (figure ci-dessous, panels A et B).

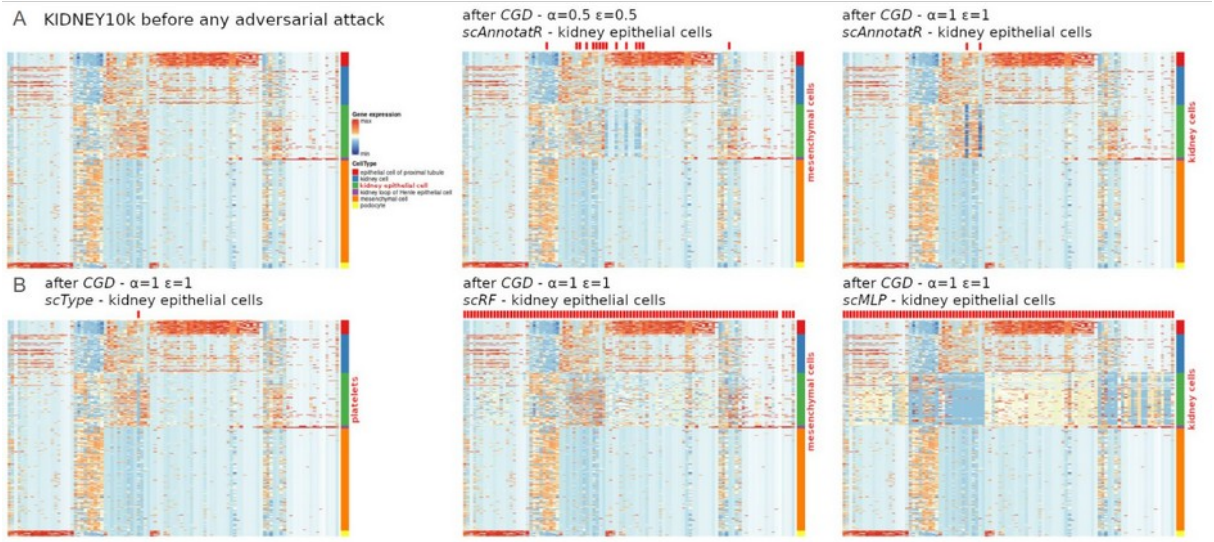

Figure 21: (A) Heatmaps montrant les gènes les plus discriminants, avant et après des attaques CGD de puissance croissante, paramètres α et ε, sur scAnnotatR, sur KIDNEY10k (B) Idem, après CGD de niveaux comparables, appliquées à différents classifieurs, sur KIDNEY10k.

Performance, exactitude et dynamique de prédiction

Une caractéristique notable de CGD est qu'il ne modifie que certains gènes et s'arrête dès que la classification du cluster cellulaire est modifiée. Ainsi, seuls les gènes influençant directement la sortie sont altérés, ce qui peut entraîner un écart entre le nombre de gènes testés et celui des gènes effectivement modifiés. Le ratio gènes testés/gènes modifiés varie selon les classifieurs (figure ci-dessous, panel A).

Par exemple, pour le cluster cellulaire FCGR3A+ Mono dans le jeu de données *PBMC3k*, une attaque de *scType* avec α = ε = 1 a nécessité le test de 18 gènes, mais n'en a modifié que 4, soit un ratio de 0,22. Pour *scAnnotatR*, avec les mêmes paramètres, 23 gènes ont été testés et 9 modifiés, soit un ratio de 0,39.

Cette différence s'explique par l'architecture des algorithmes : l'un repose sur des marqueurs prédéfinis, l'autre sur un SVM. Tous deux sélectionnent un sous-ensemble de variables d'entrée pour la classification. À l'inverse, *scRF* a testé 374 gènes et en a modifié 290, soit un ratio de 0,78, tandis que *scMLP* a testé et modifié 296 gènes, soit la totalité. Ces deux modèles utilisent presque toutes les variables d'entrée, même si leur contribution individuelle à la prédiction est faible.

Entre ces deux groupes de classifieurs, sélectifs *vs* non sélectifs sur leurs entrées, des réponses différentes ont aussi été observées lors de tests de l'influence indépendante des paramètres α et ε (figure ci-dessous, panel B).

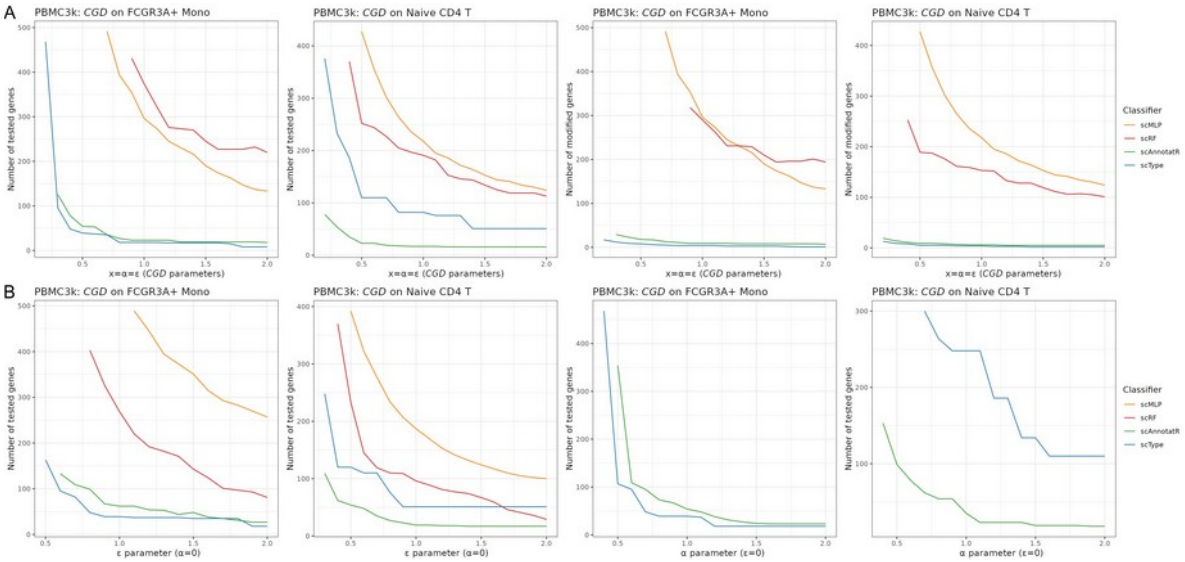

Figure 22: Performance et évaluation de CGD. L'algorithme CGD a été appliqué aux quatre classifieurs compatibles : scType, scAnnotatR, scRF et scMLP (A) Nombre de gènes à tester et/ou modifier pour modifier la classification d'un cluster cellulaire, en fonction de paramètres α et ε égaux (B) Même analyse, mais cette fois : à gauche => variation de ε quand α = 0, à droite : variation de α quand ε = 0

Chaque classifieur pouvait être attaqué uniquement avec ε. Cependant, si on se limite à 500 gènes testés lors des attaques, le coefficient multiplicatif α ne suffit pas pour faire modifier les prédictions de *scMLP* et *scRF*.

Afin d'analyser l'*accuracy,* ou exactitude, et la dynamique de prédiction en fonction du nombre de gènes testés, nous avons implémenté une version de CGD poursuivant le processus itératif sur l'ensemble des gènes fournis, même après que la prédiction du cluster cellulaire ait été modifiée.

Comparativement à *scRF* et *scMLP*, *scType* et *scAnnotatR* ont montré des perturbations plus précoces et ont modifié leur prédiction avec un plus petit nombre de gènes. Pour tous les classifieurs, des valeurs plus élevées de α et ε entraînaient des perturbations plus importantes (figure ci-dessous). Les courbes ROC variaient selon le classifieur et le cluster ciblé.

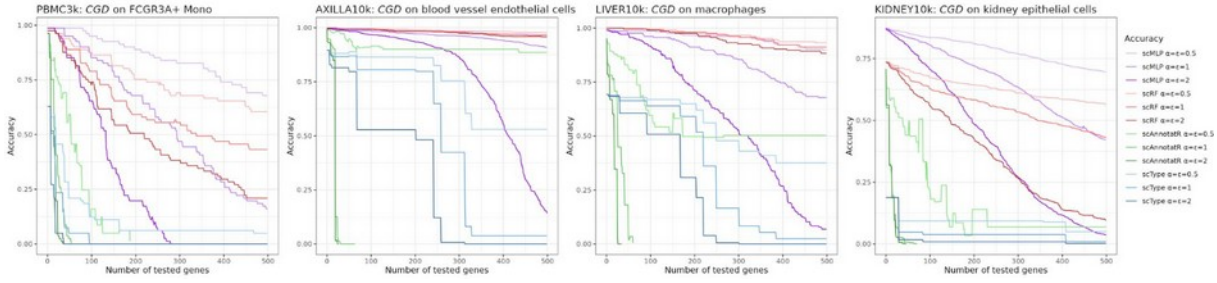

Figure 23: Précision des prédictions en fonction du nombre de gènes testés, avec différents jeux de valeurs pour ε et α, sur différents jeux de données

Pour *scAnnotatR*, la performance a été significativement affectée par les attaques avec les paramètres standards, α = ε = 1 et 500 gènes testés. En revanche, *scRF* n'a

montré que des modifications mineures de performance, même avec des paramètres élevés, α = ε = 10 et 4000 gènes testés (figure ci-dessous, panel A).

Dans les mêmes conditions, l'analyse du *F1-score* a révélé que les classifieurs étaient principalement affectés lorsqu'un seuil de probabilité élevé était requis pour prédire un type cellulaire (figure ci-dessous, panel B).

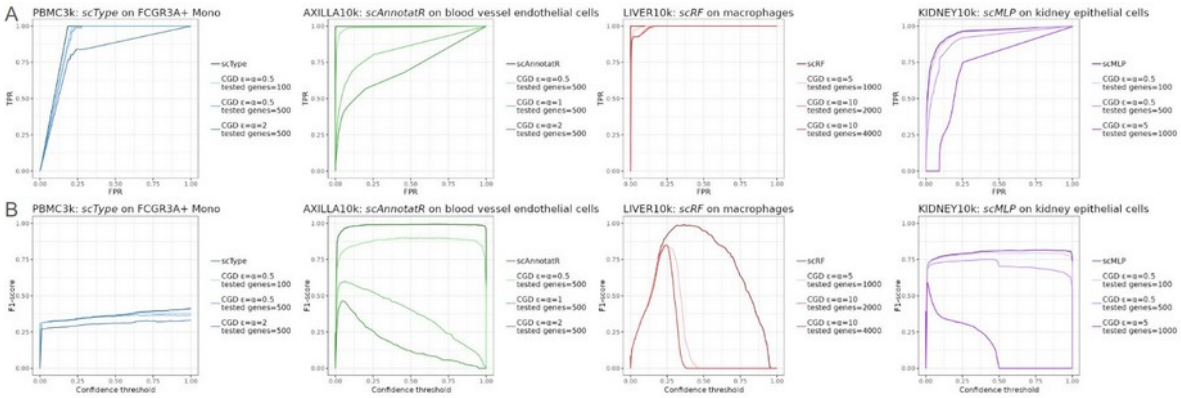

Figure 24: (A) Courbes ROC, et (B) Scores F1 correspondants, selon différentes valeurs de ε, α et du nombre de gènes testés, sur différents jeux de données. FPR : taux de faux positifs, TPR : taux de vrais positifs

Quelle attaque choisir, avec quels paramètres?

Lors de la sélection d'une méthode d'attaque, plusieurs facteurs doivent être pris en compte pour évaluer le taux de réussite et le temps de calcul requis (tableau ci-dessous) :

Targeted classifier	Targeted jeu de données				
	PBMC3k	***AXILLA10k***	***KIDNEY10k***	***LIVER10k***	
attaques adverses: *single-gene* with *perc1* modifications					
scType	NK	malignant cells	kidney epithelial cells	blood vessel smooth muscle cells	**Targeted cluster**
	72	1627	1325	505	**# cells**
	0	0	2	1	**# successful attacks**
	4 minutes	2 hours	4 hours	3 hours	**Runtime**
	Intel i7, RAM 8Gb				**Specifications**
attaques adverses: *max-change* with *decile+5* modifications, *MaxSplitSize*=1000					
scRF	memory CD4 T	blood vessel endothelial cells	podocytes	malignant cells	**Targeted cluster**
	172	744	139	1917	**# cells**
	0	0	7183	0	**# signature length**
	3 minutes	20 minutes	9 hours	40 minutes	**Runtime**
	Intel i7, RAM 8Gb				**Specifications**
attaques adverses: *max-change* with *perc99* modifications, *MaxSplitSize*=50					
CHETAH	memory CD4 T	blood vessel endothelial cells	podocytes	malignant cells	**Targeted cluster**
	172	744	139	1917	**# cells**
	941	546	448	0	**# signature length**
	2 hours	2 hours	16 hours	3 min	**Runtime**
	Intel i5, RAM 64Gb				**Specifications**
attaques adverses: *CGD* with $\alpha = \varepsilon = 0.5$					
scAnnotatR	CD8 T	blood vessel smooth muscle cells	mesenchymal cells	macrophages	**Targeted cluster**
	139	389	2559	205	**# cells**
	memory CD4 T	fibroblast	kidney cell	malignant cell	**New cell type**
	10	17	20	28	**# modified genes**
	2 minutes	30 minutes	1 hour	56 minutes	**Runtime**
	Intel i5, RAM 64Gb				**Specifications**
scMLP	CD8 T	blood vessel smooth muscle cells	mesenchymal cells	macrophages	**Targeted cluster**
	139	389	2559	205	**# cells**

	NK	fibroblast	kidney epithelial cell	monocyte	**New cell type**
	151	2121	1109	1806	**# modified genes**
	4 minutes	5 hours	4 hours	4 hours	**Runtime**
	Intel i5, RAM 64Gb				**Specifications**
attaques adverses: *CGD* with α = ε = 10					
scMLP	CD8 T	blood vessel smooth muscle cells	mesenchymal cells	macrophages	**Targeted cluster**
	139	389	2559	205	**# cells**
	NK	fibroblast	kidney epithelial cell	monocyte	**New cell type**
	3	43	55	65	**# modified genes**
	6 seconds	6 minutes	11 minutes	9 minutes	**Runtime**
	Intel i5, RAM 64Gb				**Specifications**

Tableau 1. Exemples d'utilisation d'adverSCarial sur différents jeux de données et classifieurs, avec les paramètres, résultats du nombre de succès, du temps de calcul et ressources utilisées.

Ressources computationnelles : chaque étape des trois types d'attaques implique la modification d'une matrice, suivie de la classification du résultat afin d'évaluer leur impact. Les modifications appliquées à un gène prennent en moyenne 9 sec avec un *Intel(R) Core(TM) i7-8550U* (fréquence de base : 1,80 GHz) avec 8 Go de RAM DDR4 et 1 sec avec un *Intel(R) Core(TM) i5-7400* (fréquence de base : 3,00 GHz) avec 64 Go de RAM DDR4 (Annexe ADVERSCARIAL, docs/supplementary_data_v7.pdf, S9). Le temps de calcul dépend également du classifieur, avec une moyenne de 2 min pour *scMLP* à 9 min pour scAnnotatR avec 8 Go de RAM, et 1 min pour *scMLP* à 5 min pour *scAnnotatR* avec 64 Go de RAM (Annexe ADVERSCARIAL, docs/supplementary_data_v7.pdf, S10).

***Single-gene* :** Dans le cas où aucune attaque n'est possible, l'algorithme évalue chacun des 100 splits de gènes initiaux une fois et retourne un résultat vide. Lorsque plusieurs attaques sont possibles, une recherche binaire est initiée dans chaque compartiment potentiel, augmentant le temps de calcul (Annexe ADVERSCARIAL, docs/supplementary_data_v7.pdf, S11). Par exemple, une attaque infructueuse sur un cluster *AXILLA10k* avec *scType* a consommé 2 h de temps d'exécution comparé à 4 h pour une attaque réussie sur un cluster *KIDNEY10k* contenant moins de cellules (voir tableau ci-dessus).

***Max-change* :** Quand le processus récursif implique des modifications à faible impact, qui sont moins susceptibles de modifier la classification, un grand nombre de gènes peut être intégré dans les *splits* du début, englobant parfois l'ensemble complet (lorsque la longueur de la signature est 0). En revanche, les modifications à fort impact nécessitent des recherches récursives plus profondes pour identifier des ensembles de gènes n'entraînant aucun changement de classification. Pour accélérer le calcul, l'option *maxSplitSize* permet d'arrêter la récursion dès qu'un certain nombre de gènes est atteint dans un groupe. Fixer une valeur plus élevée pour *maxSplitSize* réduit les

ressources de calcul nécessaires, mais augmente la taille de l'ensemble de gènes identifié, ce qui donne une signature moins spécifique.

CDG : Avec CGD, le temps de calcul dépend principalement des paramètres α et ε : des valeurs plus élevées de ces paramètres réduisent le nombre de gènes devant être modifiés pour mener une attaque réussie. Par exemple, avec le classifieur *scMLP* et α = ε = 0.5, 3 heures de calcul et 1297 gènes modifiés sont nécessaires, contre seulement 7 minutes de calcul et 42 gènes modifiés lorsque α = ε = 10.

***Overview mode* :** Enfin, *adverSCarial* propose des outils permettant d'évaluer la vulnérabilité globale d'un jeu de données face aux attaques pour un classifieur donné (Annexe ADVERSCARIAL, docs/supplementary_data_v7.pdf, S12). L'utilisateur peut rapidement examiner les attaques *single-gene* ou *max-change*, avec différents types de modifications, afin d'identifier ce qui fonctionne le mieux pour un couple classifieur/jeu de données. Étant donné que tester chaque cluster cellulaire avec toutes les modifications possibles est très chronophage, une configuration moins précise mais plus rapide de ces attaques est utilisée. Premièrement, les subdivisions récursives sont arrêtées dès que les sous-ensembles atteignent 100 gènes. Deuxièmement, le mode *single-gene* démarre avec moins de groupes initiaux, 20 sous-ensembles par défaut. Ces approximations permettent d'avoir rapidement un aperçu des types de modifications potentiellement efficaces et d'identifier les clusters cellulaires vulnérables. Si besoin, des attaques plus précises peuvent ensuite être lancées à partir de ces informations.

Axe 2 : Découverte grâce au XAI

L’analyse multi-omique se concentre sur l'identification d’entités d’intérêt : gènes, transcrits, protéines, sites de méthylation, et autres. Les méthodes les plus couramment employées sont des tests statistiques comme le *t-test*, le test de Wilcoxon pour la classification, et des modèles linéaires pour la régression. Cependant, ces techniques reposent sur des hypothèses comme l'unicité de distribution pour chaque catégorie ou la linéarité pour les régressions, qui ne sont pas toujours valides en pratique. Nous commencerons par illustrer ces limites à l’aide d’exemples théoriques, puis nous proposerons un cadre de comparaison des différentes méthodes. Nous appliquerons ensuite la méthode retenue sur les jeux de données de scRNA-seq de *CZ CELLxGENE*. Enfin, nous présenterons un cas pratique d’élaboration d’une signature de chimiorésistance pour la transformation de Richter.

Limites des tests statistiques

Problèmes de classification

t-test et test de Wilcoxon

Les tests statistiques univariés sont souvent utilisés pour comparer deux groupes à partir d’une variable. Ces groupes peuvent être, par exemple, des conditions cliniques (tumeurs *vs*. échantillons sains) ou des sous-types de patients (répondeurs *vs*. non-répondeurs à un traitement). En contexte omique la variable sera le niveau d’expression d’un gène, d’une protéine, un niveau de méthylation, etc. Deux des approches les plus courantes sont le *t-test* et le test de Wilcoxon.

***t-test* (ou test de Student)**

Il s’agit d’un test qui compare la moyenne d’une variable entre deux groupes et vérifie si la différence observée est significative.

Hypothèse :

- distribution gaussienne des deux groupes
- même variance dans les deux groupes

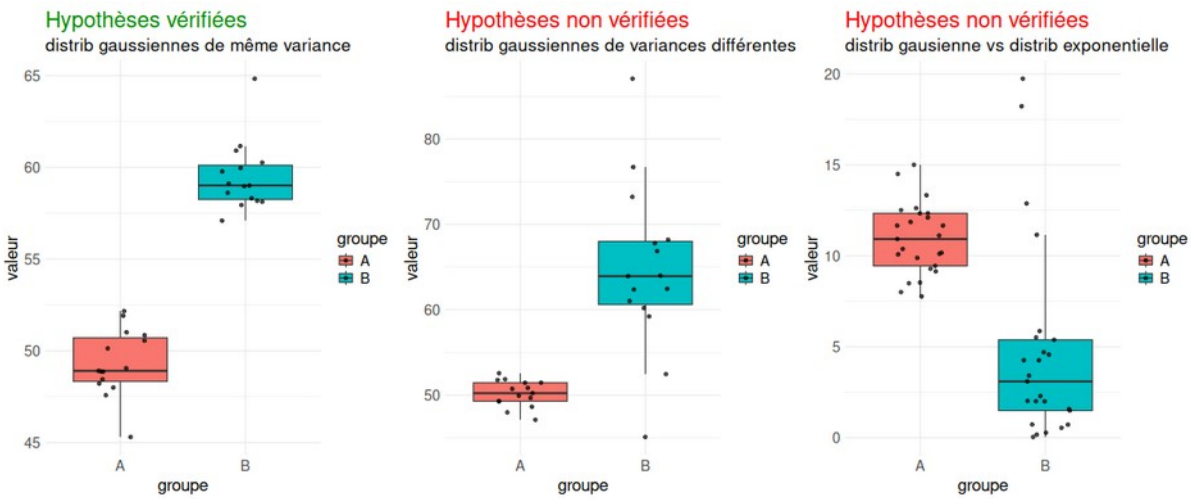

Figure 25: Cas valides et non valides pour l'utilisation du t-test (Annexe CONTENT4PHD : 0001_R_ttest_vs_wc.ipynb, im/figs/0001_ttest_3picInRow_hypothesesVerifies.png)

Test de Wilcoxon

Il s'agit d'un test non paramétrique, c'est à dire qui ne pose pas d'hypothèse sur la distribution des données, qui évalue les différences de rang entre deux groupes. Il est très permissif mais ne permet pas de prendre en compte les variations de moyennes, raison pour laquelle le *t-test* est plus courant.

Dans la figure ci-dessous on peut voir deux exemples montrant les mêmes *p-values* du test de Wilcoxon malgré des écarts de moyennes différents. Le premier exemple illustre les cas de faibles effectifs, où le *t-test* retourne des résultats significatifs avec des *p-values* = 0.0213 et 9.1e-05 (< 0.05), et le test de Wilcoxon des résultats non significatifs avec une *p-value* identique de 0.1 . Dans le second exemple, bien que toutes les *p-values* soient significatives avec deux *t-test* à 3e-08 et 1.22e-14 et le test de Wilcoxon à 1.08e-05, le choix du test peut avoir un effet en cas de correction pour la *False Discovery Rate* (FDR). Supposons que les tests soient effectués pour analyser l'expression d'un gène parmi un un jeu de données de transcriptomique comportant 20 000 gènes, les *p-values* ajustées pour la FDR seraient alors respectivement 20000*3e-08=0.0006, 20000*1.22e-14=2.44e-10 et 20000*1.08e-05=0.216. Le gène analysé serait considéré comme significativement différent entre les deux groupes avec le *t-test*, mais pas avec le test de Wilcoxon.

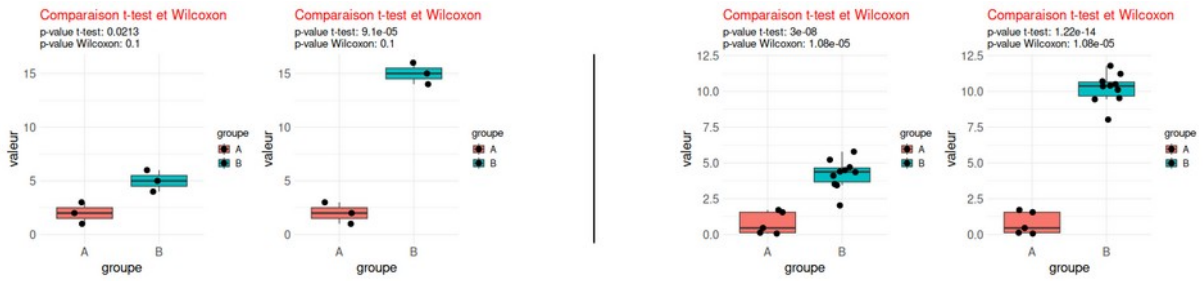

Figure 26: Différence des moyennes, t-test vs test de Wilcoxon (Annexe CONTENT4PHD : 0001_R_ttest_vs_wc.ipynb, im/figs/0001_ttest_4picInRow_comparaison_ttest_vs_wilcox.png)

Cas invisibles pour les tests statistiques

On a vu des cas où les hypothèses pour l'utilisation du *t-test* ne sont pas vérifiées, et des cas où le test de Wilcoxon classe comme non significatifs des variables qui devraient l'être. Cette section explore différents scénarios où les tests statistiques standards, comme le *t-test* et le test de Wilcoxon, échouent à identifier des variables biologiquement pertinentes : les distributions bimodales, les différences de variances et les interactions entre variables. L'objectif de cette thèse est d'examiner ces scénarios et de développer des approches pour les détecter.

Distribution bimodale

Imaginons l'étude de l'expression d'un gène chez deux groupes de patients. Dans le groupe A, composé de patients sains, la distribution des niveaux d'expression est unimodale, avec la majorité des valeurs concentrées autour d'une moyenne donnée. Dans le groupe B, qui regroupe des patients présentant une pathologie, la distribution de l'expression du même gène est bimodale. Autrement dit, ce groupe se divise en deux sous-populations distinctes : l'une présentant une expression relativement basse, et l'autre une expression plus élevée. La moyenne globale du groupe B pourrait s'avérer similaire à celle du groupe A, rendant le phénomène invisible au *t-test* et au test de Wilcoxon. C'est ce qu'on voit dans la figure ci-dessous avec le groupe A qui contient 100 valeurs d'une distribution gaussienne de moyenne $\mu=0$ et d'écart type $\sigma=1$, et le groupe B qui est le résultat d'une distribution bimodale, 50 valeurs d'une distribution gaussienne de moyenne $\mu=4$ et d'écart type $\sigma=1$ et 50 valeurs de paramètres $\mu=-4$ et $\sigma=1$. Pour ce gène, le *t-test* et le test de Wilcoxon ne montrent pas de différence significative entre les groupes, avec des *p-values* respectives de 0.652 et 0.993.

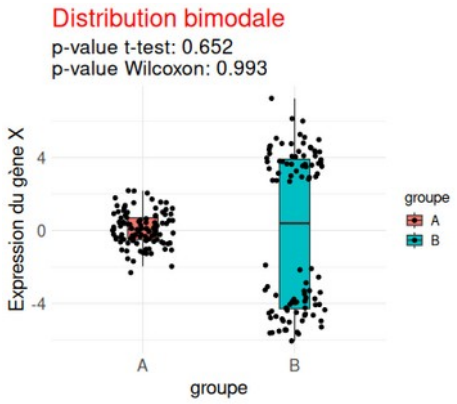

Figure 27: Distribution gaussienne bimodale (Annexe CONTENT4PHD : 0002_R_invisible2statTests.ipynb, im/figs/0002_distribution_bimodale.png)

Différence de variance

Imaginons un scénario où nous comparons les niveaux d'expression d'un gène entre deux groupes, chacun avec une distribution gaussienne unimodale mais des dispersions différentes. Le groupe A garde les même paramètres $\mu=0$ et $\sigma=1$. Nous construisons le groupe C à partir d'une distribution gaussienne unimodale de moyenne $\mu=0$ et de variance égale à celle du groupe B. Nous choisissons de conserver la même variance pour les groupes B et C en prévision des étapes suivantes. Avec des *p-values* de 0.405 pour le *t-test* et de 0.438 pour le test de Wilcoxon, ce gène ne serait pas considéré comme significativement différent entre les groupes.

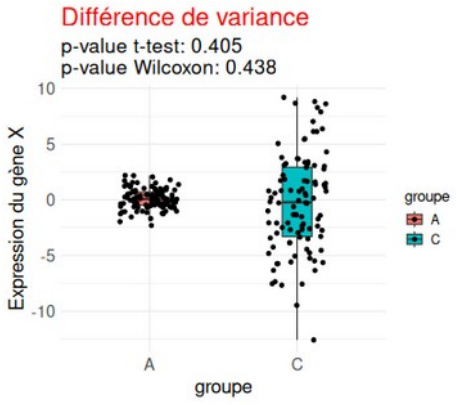

Figure 28: Distributions gaussiennes de variances différentes (Annexe CONTENT4PHD : 0002_R_invisible2statTests.ipynb, im/figs/0002_diff_variance.png)

Interactions entre variables

Jusqu'à présent, nous avons illustré des situations dans lesquelles un test statistique est appliqué à une variable isolée pour différencier les groupes A et B. Il peut arriver qu'une variable ne révèle son lien avec les groupes qu'en considérant une autre variable. La figure ci-dessous illustre ce phénomène avec deux gènes, X et Y. Le premier panneau montre un *boxplot* de l'expression du gène X seul en fonction des deux groupes. Les *p-values* du *t-test* et du test de Wilcoxon de respectivement 0.667 et 0.778 ne mettent en évidence aucune différence significative entre A et B. Le panneau en haut à droite montre qu'il existe une relation entre groupe, gène X et gène Y. Lorsque l'expression des deux gènes X et Y est faible alors c'est le groupe A. Quand l'expression est forte pour les deux gènes c'est également le groupe A. Mais dans les autres cas c'est le groupe B. Ceci indique que le gène X est pertinent pour distinguer les deux groupes, malgré les *p-values* élevées, après classification des échantillons selon Y. C'est ce que révèlent les deux panneaux du bas. Pour la sous-population gène Y > 0, on observe une différence d'expression de X entre A et B avec des *p-values* pour le *t-test* et le test de Wilcoxon de respectivement 1.09e-8 et 1.87e-8. De la même manière on trouve pour la sous-population 'gène Y < 0' des *p-values* de 5.55e-8 et 2.48e-9 pour le *t-test* et Wilcoxon respectivement.

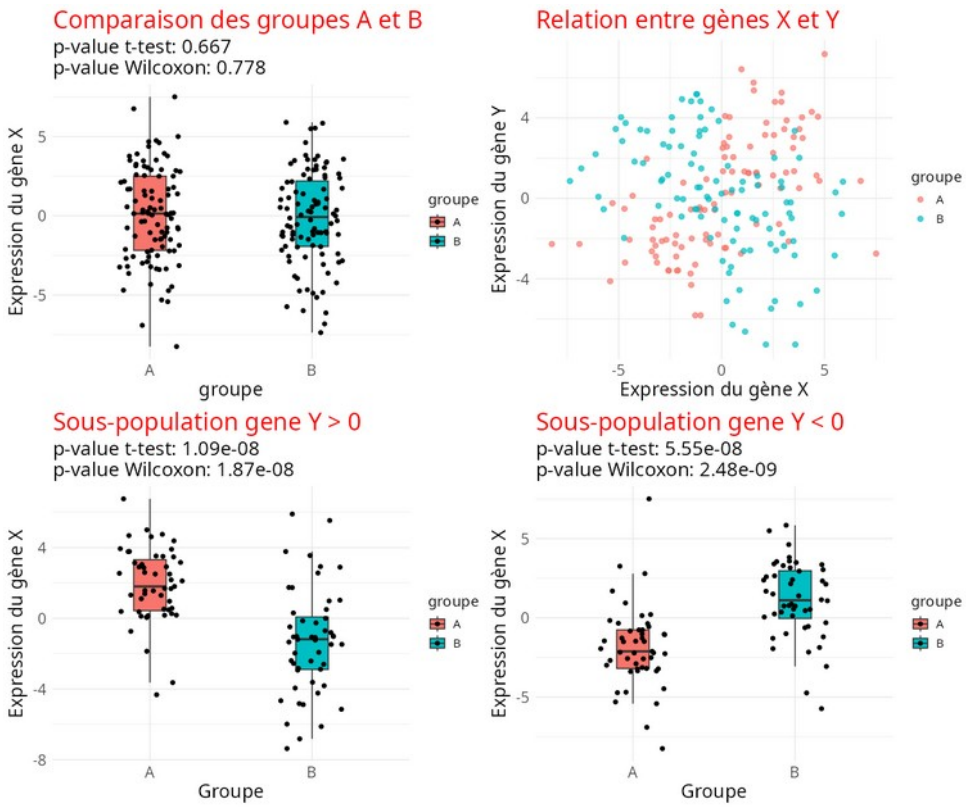

Figure 29: Illustration des interactions entre variables (Annexe CONTENT4PHD : 0002_R_invisible2statTests.ipynb, im/figs/0002_interaction_geneX_geneY.png)

Le F-test pour les distributions bimodales et les différences de variances

Le *F-test* est un test statistique conçu pour évaluer les différences de variances entre deux distributions. La figure ci-dessous montre qu'il repère ces différences aussi bien dans les distributions bimodales que dans celles présentant des variances distinctes, avec des *p-values* proches de 1e-40. Cependant, dans le cas de la distribution bimodale, la séparation entre les groupes A et B apparaît plus nette qu'avec la distribution unimodale pour les groupes A et C, et le *F-test* ne reflète pas cette différence.

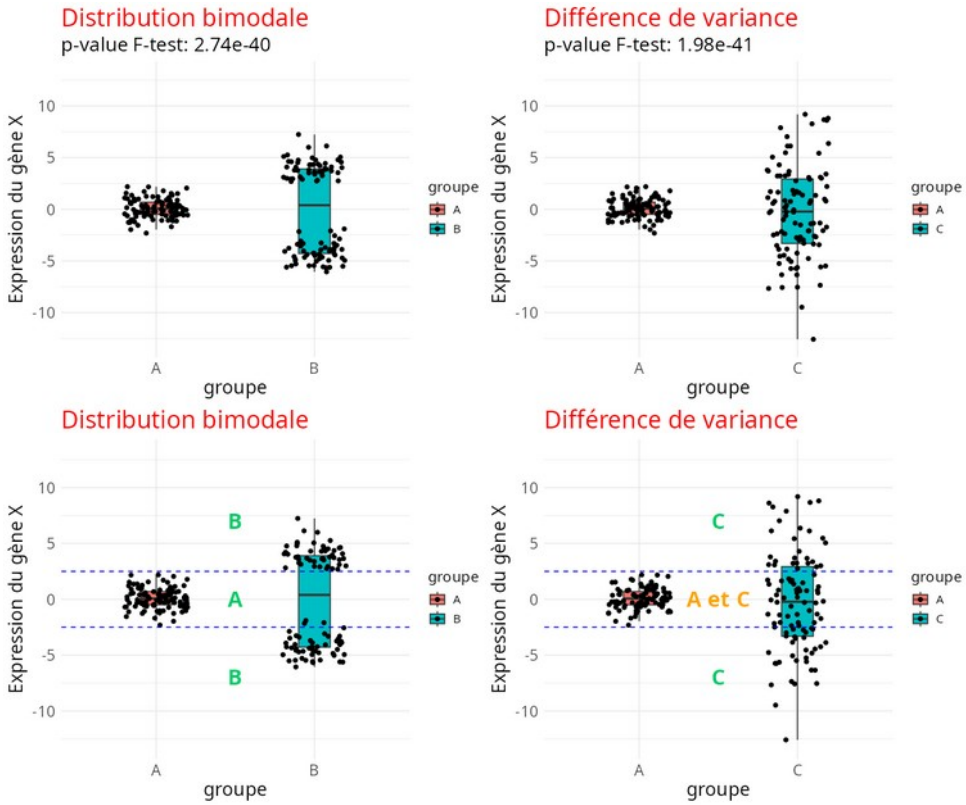

Figure 30: Illustration des limites du F-test (Annexe CONTENT4PHD : 0002_R_invisible2statTests.ipynb, im/figs/0002_ftest_2x2.png)

Problèmes de régression

Régressions linéaires et corrélations

Les modèles de régression linéaire et les tests de corrélation sont fréquemment utilisés pour évaluer la relation entre des variables explicatives et une variable continue. Ces approches reposent sur l'hypothèse d'une relation linéaire entre les deux variables.

Régressions linéaires

Les modèles de régression linéaire sont des outils statistiques utilisés pour estimer la relation linéaire entre une ou plusieurs variables explicatives et une variable cible continue. On cherche à estimer les coefficients β_i tels que :

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p + \varepsilon$$

où y est la variable cible, $x_1, x_2, ..., x_p$ sont les variables explicatives et ε l'erreur aléatoire supposée normalement distribuée.

Chaque coefficient β_i est associé à une ***p-value*** permettant de tester l'hypothèse nulle $H_0 : \beta_i = 0$ c'est-à-dire l'absence d'effet significatif de la variable x_i

sur la variable cible. Une ***p-value*** faible signifie que la variable explicative x_i

est statistiquement liée à la variable cible.

Corrélations

Les coefficients de corrélation les plus couramment utilisés sont ceux de Pearson et de Spearman, notés respectivement r et ρ. Le coefficient de Pearson mesure la force et la direction d'une relation linéaire entre deux variables quantitatives, tandis que le coefficient de Spearman évalue une relation monotone, qu'elle soit linéaire ou non, en se basant sur les rangs des données. Leur valeur est comprise entre -1 et 1 : une valeur proche de 1 indique une corrélation linéaire positive forte, une valeur proche de -1 une corrélation négative forte, et une valeur proche de 0 l'absence de relation linéaire.

Le test de Pearson permet d'évaluer la significativité de cette corrélation à l'aide d'une ***p-value***, en testant l'hypothèse nulle $H_0 : r = 0$, c'est-à-dire l'absence de lien linéaire entre les deux variables.

Le test de Spearman, basé sur les rangs, est utilisé pour détecter des relations non linéaires, mais quand même monotones. Il évalue l'hypothèse $H_0 : \rho = 0$ et est plus robuste aux valeurs aberrantes et aux distributions non normales.

Cas invisibles pour les tests statistiques

On a vu que les régressions linéaires et les corrélations permettent de détecter quelles variables ont une relation linéaire ou monotone avec une variable cible. Ceci permet de détecter des biomarqueurs dans le cadre d'une variable cible continue. Voici deux exemples illustrant une relation forte entre deux variables continues, mais qui n'est révélée ni par la régression linéaire ni par la corrélation. Les détails de construction de ces exemples sont disponibles dans la partie « scénarios à détecter ».

Exemple de la parabole

Le jeu de données illustré ci-dessous contient 100 observations. Il comporte 20 variables générées aléatoirement, ainsi qu'une variable cible y , une parabole construite à partir d'une des variables aléatoires.

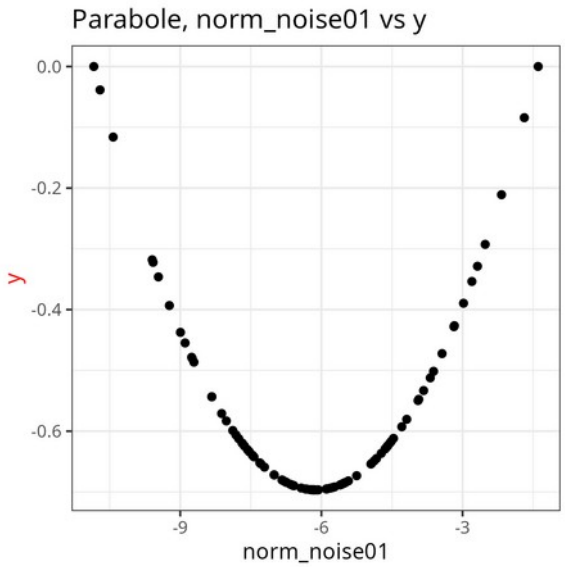

Figure 31: Distribution gaussienne pour norm_noise01 et parabole construite à partir de norm_noise01 en ordonnée, voir détails de construction dans le paragraphe « scénarios à détecter ». (Annexe CONTENT4PHD 0008_R_invisible_regression.ipynb, im/figs/0008_parabole.png)

Une régression linéaire multiple a été réalisée sur l'ensemble des variables explicatives, et le coefficient de corrélation de Pearson a été calculé spécifiquement entre la variable cible et la variable aléatoire utilisée pour sa construction.

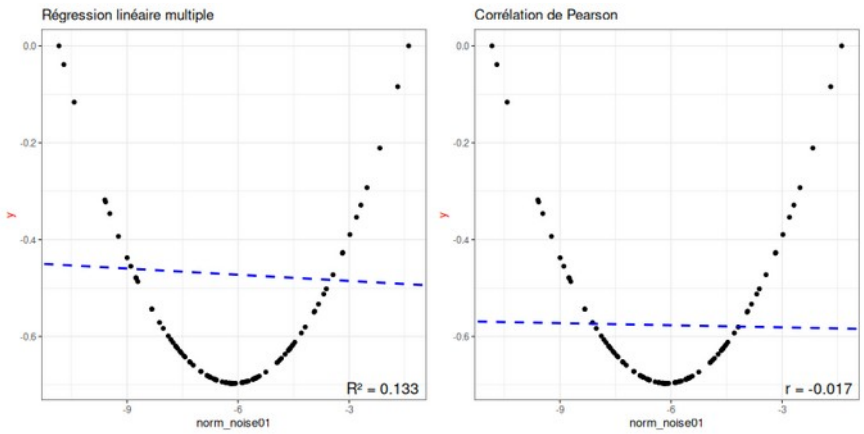

Figure 32: Calcul d'une régression linéaire et d'une corrélation de Pearson sur le jeu de données "parabole" (Annexe CONTENT4PHD 0008_R_invisible_regression.ipynb, im/figs/0008_parabole_regr_correl.png)

La régression linéaire multiple retourne un coefficient R^2 de 0.133 , tandis que la corrélation de Pearson entre la variable cible y et la variable utilisée pour sa construction donne un coefficient $r = -0.017$. Ces résultats suggèrent un lien linéaire faible entre les variables explicatives et y alors qu'elle a directement été construite à partir de l'une d'entre elles, illustrant la limite de l'utilisation des tests classiques pour détecter les relations non linéaires entre variables.

Exemple de la sinusoïde à amplitude croissante

Le jeu de données illustré ci-dessous contient 75 observations. Il comporte également 20 variables générées aléatoirement, ainsi qu'une variable cible y , une sinusoïde à amplitude croissante construite à partir d'une des variables aléatoires.

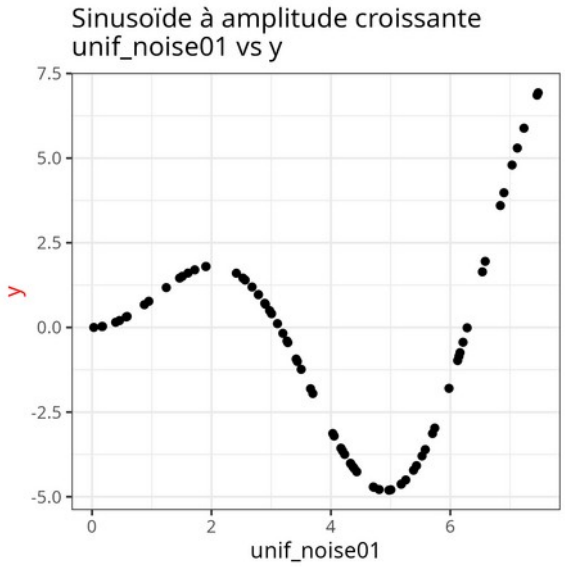

Figure 33: Distribution uniforme pour unif_noise01 et sinusoïde à amplitude croissante construite à partir de unif_noise01 en ordonnée, voir détails de construction dans le paragraphe « scénarios à détecter ». (Annexe CONTENT4PHD 0008_R_invisible_regression.ipynb, im/figs/0008_sinus.png)

Tout comme dans l'exemple précédent, une régression linéaire multiple a été réalisée sur l'ensemble des variables explicatives, et le coefficient de corrélation de Pearson a été calculé spécifiquement entre la variable cible et la variable aléatoire utilisée pour sa construction.

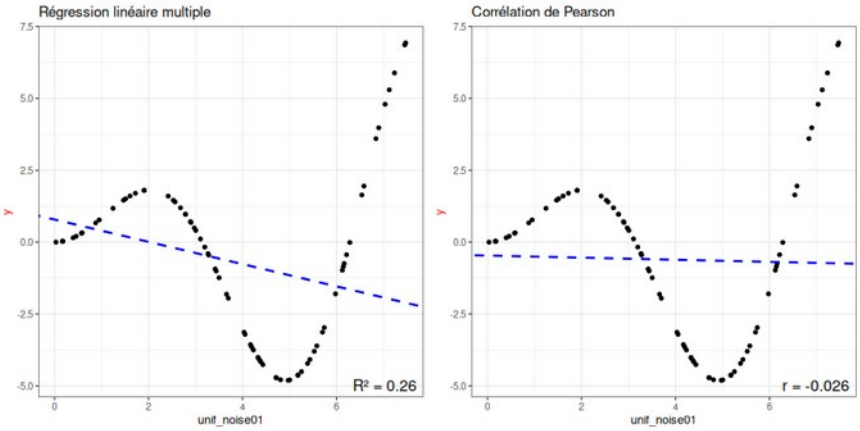

Figure 34: Calcul d'une régression linéaire et d'une corrélation de Pearson sur le jeu de données "sinuosïde à amplitude croissante" (Annexe CONTENT4PHD 0008_R_invisible_regression.ipynb, im/figs/0008_sinus_regr_correl.png)

La régression linéaire multiple retourne un coefficient R^2 de 0.26, suggérant une relation linéaire modérée entre les variables explicatives et la variable cible y, ce qui est incompatible avec la nature oscillante de cette dernière. Par ailleurs, la corrélation de Pearson entre y et la variable ayant servi à sa construction donne un coefficient $r = -0.026$ traduisant un lien linéaire faible, alors même que y a été directement générée à partir de cette variable, illustrant à nouveau qu'une relation non linéaire peut échapper aux tests classiques.

Machine learning et *feature importance*

Nous proposons de pallier les limites des tests statistiques en les accompagnant d'autres méthodes qui révèlent également le niveau d'importance de variables pour expliquer des catégories ou des variables continues, regroupées sous le terme de *feature importance*. Le principe est le suivant. Nous avons un tableau dont les colonnes sont des variables numériques qui ont été mesurées (niveau d'expression des gènes, de protéines, niveaux de méthylation) et une colonne correspondant à la catégorie (tumeurs *vs*. échantillons sains) ou une variable d'intérêt continue (température, espérance de vie). Chaque ligne correspond à un échantillon. Les tests statistiques évaluent chaque colonne séparément, générant une *p-value* pour chaque variable, permettant de sélectionner les plus pertinentes. Les méthodes de *feature importance* débutent par l'entraînement d'un algorithme de *machine learning* sur les données du tableau, où l'objectif est de prédire la catégorie ou la variable continue de chaque échantillon (représenté par les lignes) à partir des variables (les colonnes). Une fois que l'algorithme est capable de prédire la catégorie ou la variable continue, les méthodes de *feature importance* sont appliquées pour identifier les variables que

l'algorithme utilise dans ses prédictions et leur attribuer un score d'importance. Nous analyserons divers algorithmes de *machine learning*, notamment les *decision trees*, les *random forests*, les perceptrons multicouches (MLP) et les SVM. Pour chaque modèle, nous explorerons différentes techniques de *feature importance*, comme *SHAP*, *treeSHAP*, *LIME*, l'importance de Gini et le calcul de l'importance par permutation. Cette approche nous permettra de comparer l'efficacité et la pertinence de chaque méthode.



Figure 35: Workflows basés sur les p-values ou sur les feature importances (Annexe CONTENT4PHD im/figs/dr_0001_pval vs feature imp.png)

SHAP (*SHapley Additive exPlanations*)

SHAP est la méthode de XAI la plus répandue [126]. Il s'agit d'une méthode issue de la théorie des jeux, la méthode des valeurs de Shapley [127] qui permet de quantifier la contribution de chaque joueur dans une équipe. L'idée est la suivante : supposons une équipe de 6 joueurs $N=\{1,2,3,4,5,i\}$ qui jouent à un jeu coopératif et on souhaite partager les gains équitablement entre les joueurs en fonction de leur contribution. On calcule le gain équitable du joueur i. Pour ce faire on prend des sous-ensembles de ces 6 joueurs, par exemple l'équipe $S=\{1,2,3\}$, et on note leur gain $v(\{1,2,3\})$. Puis on ajoute le joueur i à l'équipe et on note à nouveau leur gain $v(\{1,2,3,i\})$. La contribution de i pour cette équipe est donc de $v(\{1,2,3,i\})-v(\{1,2,3\})$. On répète ce calcul pour toutes les combinaisons possibles de joueurs et on en fait la moyenne :

$$\phi_i(v) = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!\,(|N| - |S| - 1)!}{|N|!} \left(v(S \cup \{i\}) - v(S)\right)$$

SHAP transpose ce mécanisme pour calculer un score de *feature importance* à chaque variable d'un modèle prédictif. Supposons l'exemple suivant, un modèle qui prend en entrée un ensemble de 6 gènes $G=\{gene_1, gene_2, gene_3, gene_4, gene_5, gene_i\}$ et prédit une condition A ou B, c'est à dire qu'il retourne une probabilité p_A que ce soit A et une probabilité $p_B=1-p_A$ que ce soit B. Pour calculer la contribution de $gene_i$ dans

le modèle, *SHAP* examine l'impact de l'ajout de $gene_i$ aux différents sous-ensembles de gènes comme $S=\{gene_1, gene_2, gene_3\}$. On mesure d'abord la prédiction p_A^S avec le sous-ensemble S, puis on ajoute $gene_i$ à S pour former $S'=\{gene_1, gene_2, gene_3, gene_i\}$ et on mesure à nouveau la prédiction $p_A^{S'}$. La différence $\left|p_A^S - p_A^{S'}\right|$ indique l'impact de l'ajout de $gene_i$ sur la prédiction. Ce processus est répété pour toutes les combinaisons de sous-ensembles, et la contribution moyenne de $gene_i$ est calculée. Cette moyenne représente la *feature importance* de $gene_i$ dans le modèle prédictif pour distinguer entre les conditions A et B.

TreeSHAP

TreeSHAP est une variante de *SHAP* conçue pour les modèles basés sur des arbres de décision, comme les arbres de décision simples ou les forêts aléatoires. Il parcourt les arbres de décision pour évaluer la contribution de chaque division du ou des arbres à la prédiction finale. Cette approche donne les mêmes résultats que *SHAP* mais évite la simulation de toutes les combinaisons possibles, rendant le processus plus rapide.

Importance de Gini

Le calcul de l'importance de Gini [128] s'appuie sur l'indice de Gini [129]. L'indice de Gini est une mesure statistique qui évalue l'impureté au sein d'un nœud dans les arbres de décision, 1 signifie que tous les échantillons dans le nœud appartiennent à une seule classe, et 0 indique une répartition égale des échantillons entre deux classes. L'importance de Gini mesure l'efficacité avec laquelle chaque variable pourrait contribuer à la pureté des nœuds dans les arbres, en calculant la diminution de l'indice de Gini, donc la réduction de l'impureté. Cette réduction d'impureté est cumulée pour chaque variable à travers l'ensemble des nœuds des arbres, fournissant les scores d'importance de Gini. Plus le score est élevé, plus la variable a le potentiel de séparer les échantillons de classes différentes. À noter des différences d'implémentation de l'importance de Gini :

- ***scikit-learn*** (*Python*), la réduction de l'indice de Gini est calculée uniquement sur la variable qui a été gardée pour chaque nœud.

- ***rpart***, ***randomForest*** (*R*), la réduction de l'indice de Gini est calculée sur toutes les variables à chaque nœud, et pas seulement pour la variable qui a finalement été choisie dans l'arbre pour diviser ce nœud.

TreeSHAP vs Importance de Gini

On trouve des écarts de résultats entre *SHAP/TreeSHAP* et l'importance de Gini (celle calculée dans les packages *rpart* et *randomForest*), notamment en présence de variables corrélées. En effet *SHAP/TreeSHAP* calcule la contribution marginale de chaque variable dans un modèle donné, donc si le modèle n'utilise qu'une des deux variables corrélées, le score d'importance de la seconde sera nul. Inversement l'importance de Gini des packages *rpart* et *randomForest* calcule la contribution générale potentielle de chaque variable pour chaque nœud de chaque arbre, deux

variables corrélées auront des scores d'importance proches. Pour illustrer cette différence, nous avons choisi un exemple avec trois variables, dont deux sont fortement corrélées. La variable X_1 consiste en 1000 valeurs générées aléatoirement suivant une distribution gaussienne avec une moyenne de 0 et un écart type de 1. La variable X_2 (celle corrélée à X_1) est la somme de X_1 et d'une distribution gaussienne de moyenne 0 et d'écart type 0.2. La variable X_3 est générée par 1000 tirages au sort d'une distribution uniforme entre 0 et 1. La variable cible y est définie comme $y = A$ si $X_1 > 0$ et $y = B$ si $X_1 < 0$. On entraîne un arbre de décision avec les paramètres par défaut du package *rpart*, et on entraîne un *random forest* avec 300 arbres avec le package *randomForest*. L'importance de Gini montre des scores proches pour X_1 et X_2 dans l'arbre de décision, avec 500 pour X_1 et 448 pourX_2. Pour le *random forest*, les scores sont de 301 pour X_1 et 191 pour X_2. Par contre les valeurs *SHAP/TreeSHAP* révèlent des écarts significatifs : 0.49 pour X_1 et 0 pour X_2 dans l'arbre de décision, et 0.43 pour X_1 et 0.06 pour X_2 dans la forêt aléatoire.

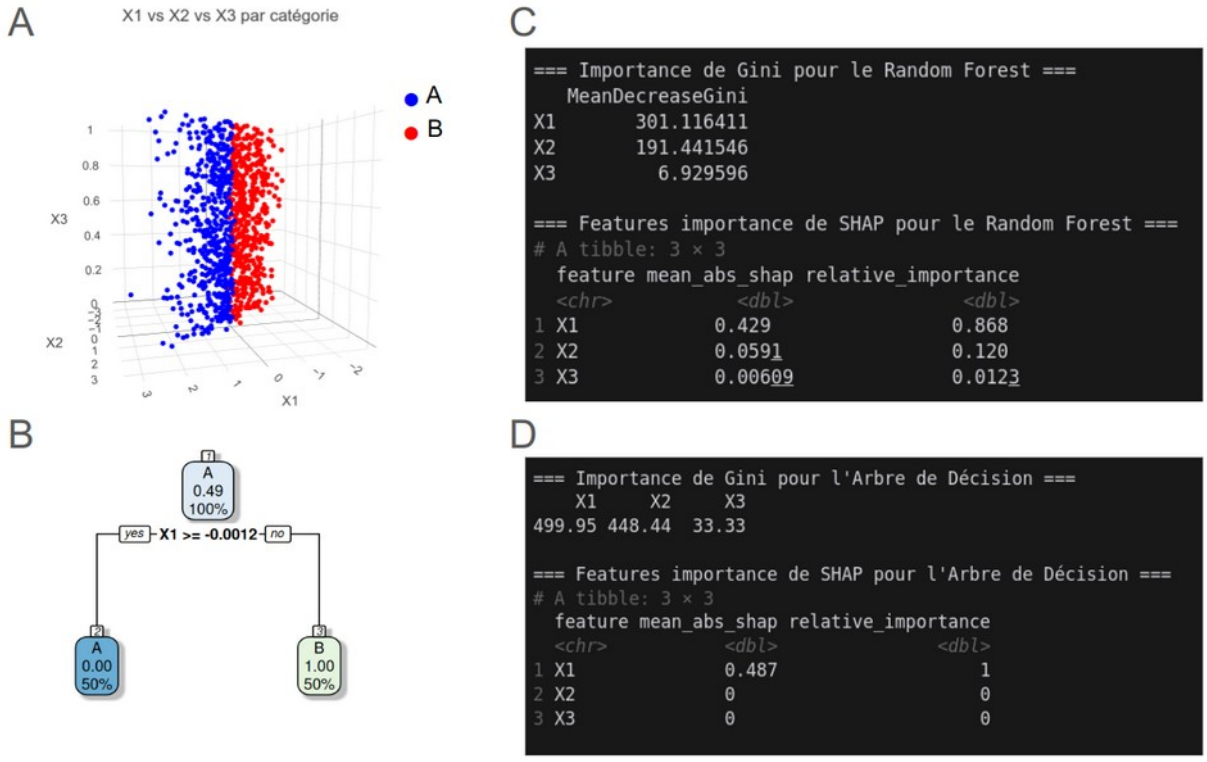

Figure 36: Importance de Gini vs SHAP/TreeSHAP: (A) nuage de points 3D des échantillons A et B (B) règles de l'arbre de décision (C, D) importance de Gini et feature importance de SHAP/TreeSHAP (Annexe CONTENT4PHD im/figs/0003_2x2_gini_vs_shap.png)

LIME (*Local Interpretable Model-agnostic Explanations*)

Le *Local Interpretable Model-agnostic Explanations*, ou *LIME* [51], vise à mesurer l'importance de chaque variable pour la prédiction d'un modèle prédictif. À la différence de l'importance de Gini qui décrit l'importance globale des variables sur les prédictions du modèle, *LIME* se concentre sur l'importance locale, décrivant le comportement du

modèle autour d'un point spécifique. Cela permet à *LIME* de détecter des relations spécifiques à des situations particulières, alors que l'importance de Gini met en évidence des influences plus monotones et uniformes à travers l'ensemble du jeu de données.

$$\xi(x) = \underset{g \in G}{\operatorname{argmin}} \left[L(f, g, \pi_x) + \Omega(g) \right]$$

Supposons un modèle d'IA f qui prédit la probabilité d'une catégorie A ou B à partir d'une liste d'expression de gènes $x' = \{gene_1, gene_2, gene_3, gene_4\}$, $f(x') = P(A|x') = 1 - P(B|x')$. On cherche $\xi(x)$, une fonction proche de f en un point x parmi toutes les fonctions possibles G. Pour obtenir une approximation de f on cherche à minimiser une fonction de perte L correspondant à une distance entre $f(x)$ et $g(x)$, et on y ajoute des termes de régularisation, π_x et $\Omega(g)$, pour simplifier le résultat.

- π_x donne un score plus faible aux points qui sont proches de x et plus élevé aux points éloignés. Comme $argmin$ cherche à minimiser ce score cela permet de concentrer l'explication sur le voisinage local.
- $\Omega(g)$ reflète le niveau de complexité de g. Plus g est complexe, plus $\Omega(x)$ sera élevé. Les fonctions classiques utilisées sont les termes de régularisation $L1$ (lasso) et $L2$ (ridge).

Le plus courant est de choisir $\xi(x)$ parmi l'ensemble des fonctions linéaires, ses coefficients correspondent alors aux *feature importances* de *LIME* qu'on cherche à calculer.

Permutations

D'abord introduite dans la publication de référence des *random forests* [128], le calcule de la *feature importance* des variables par permutation est en fait applicable à l'ensemble des modèles prédictifs. Il s'agit de mesurer l'influence de la permutation des données d'une variable dans un ensemble d'inputs, par exemple les données d'entraînement, sur l'output du modèle. Il en est souvent fait mention dans les fonctions par *mean decrease in accuracy*.

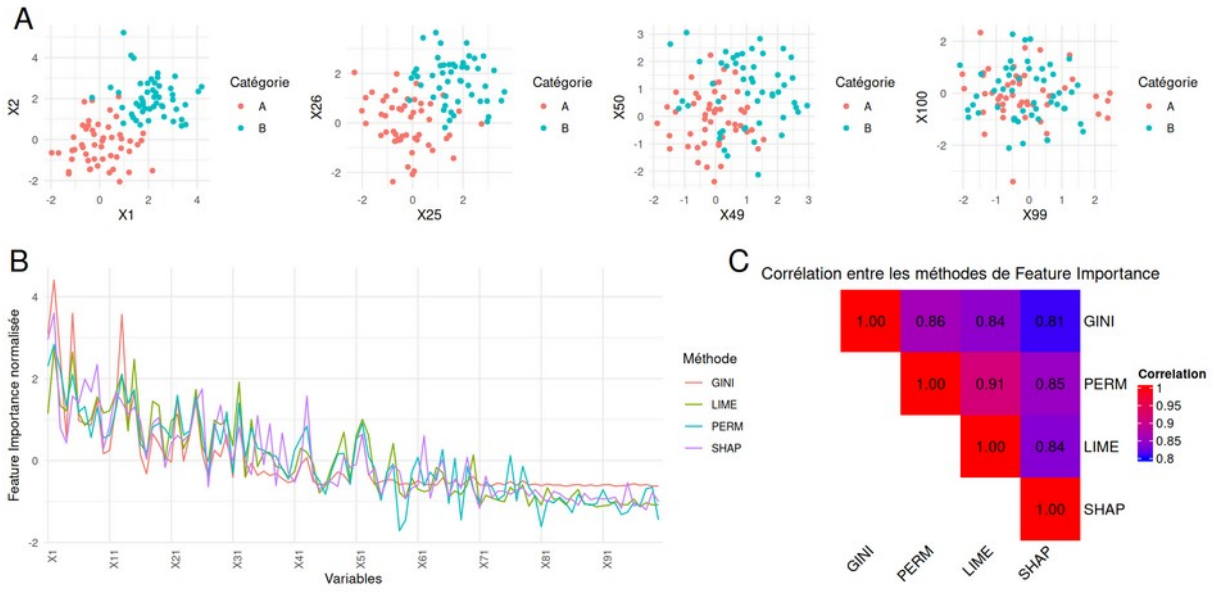

Figure 37: Jeu de données composé de 50 points de catégorie A, 50 points de catégorie B, 100 variables gaussiennes X1, X2, X3, etc allant du plus différenciant au moins différenciant. (A) scatter plots illustrant la progressivité de la différenciation au fil des variables (B) classification par random forest et calcul des feature importances selon plusieurs méthodes (C) corrélations entre les différentes méthodes. (Annexe CONTENT4PHD im/figs/0004_fullPanel_shap_gini_lime_permutation.png)

Score d'Olden

Le score d'Olden est une méthode de calcul de l'importance des variables pour les réseaux de neurones, par exemple les MLP. Contrairement aux techniques basées sur des permutations ou des méthodes agnostiques comme *SHAP* ou *LIME*, il utilise directement la structure du modèle. Pour chaque variable en entrée, la *feature importance* est estimée en multipliant les poids de chaque chemin entre l'entrée et la sortie, et en sommant ces contributions. L'idée est que les chemins avec des poids élevés, qu'ils soient positifs ou négatifs, traduisent une forte influence de la variable sur la sortie du modèle.

Prenons le cas d'un MLP avec une couche de neurones cachée :

- x_i une variable d'entrée
- w_{ij} le poids entre l'entrée x_i et le neurone caché h_j
- w_{jo} le poids entre le neurone caché h_j et la sortie o

alors l'importance $I(x_i)$ de la variable x_i selon la méthode d'Olden est :

$$I(x_i) = \sum_{j=1}^{H} w_{ij} \cdot w_{jo}$$

Conversion entre *feature importances* et *p-values*

On a vu que les méthodes de *feature importance* permettent d'identifier quelles variables sont les plus utiles pour prédire une condition ou une variable continue. Mais contrairement aux tests statistiques, elles ne fournissent pas de *p-value*, ce qui complique l'identification des variables dépassant un seuil de significativité, et rend difficile l'application de corrections pour le taux de fausses découvertes (FDR).

Des méthodes visant à convertir les scores de *feature importance* en *p-values*, telles que *PIMP* [130] ou *1Probe/mProbe* [131], ont été proposées dans la littérature. Nous présenterons également *simThresh*, la méthode que nous avons développée, qui permet d'estimer des seuils de significativité pour les *feature importance*, sans nécessiter une conversion explicite de chaque score en *p-value*.

PIMP (*Permutation Importance p-values*)

La méthode *PIMP* permet d'associer une valeur de *p-value* aux scores de *feature importance* issus d'un modèle d'apprentissage automatique. Pour chaque variable, le score de *feature importance* est d'abord calculé sur les données d'origine, puis recalculé après avoir aléatoirement permuté la variable cible un grand nombre de fois, appelons ce paramètre *n_perm*. Cette procédure génère une distribution empirique nulle des scores d'importance sous l'hypothèse d'absence de lien entre les variables explicatives et la variable à prédire. En comparant le score observé à cette distribution nulle on obtient une *p-value* indiquant la probabilité d'observer ce score par hasard.

PIMP permet de fournir un cadre statistique rigoureux pour l'interprétation des scores d'importance, permettant une sélection de variables fondée sur la significativité, avec une correction possible pour la FDR. Toutefois, cette méthode peut être coûteuse en temps de calcul car elle nécessite d'entraîner un nouveau modèle pour chacune des *n_perm* versions permutées des données.

1Probe/mProbes

La méthode *1Probe/mProbes* propose également d'associer une *p-value* à chaque score de *feature importance* produits par des modèles de *machine learning*. Le principe repose sur l'ajout de variables factices, les probes, aléatoires et non informatives, au jeu de données d'origine. Ce jeu de données comporte alors les variables d'origines et les probes. L'idée est de générer une distribution nulle empirique des scores d'importance, à partir de variables censées ne pas avoir d'effet prédictif.

Dans la version *1Probe*, une seule variable aléatoire est ajoutée pour chacune des *n_iter* itérations, tandis que dans *mProbes*, *m* variables sont ajoutées. Chaque itération consiste au réentraînement du modèle avec les probes, et au calcul des nouvelles *feature importances*. La *p-value* associée à une variable d'origine correspond alors à la proportion de probes dont le score d'importance est supérieur ou égal au sien. Ce processus est coûteux en temps de calcul, car il nécessite également de réentraîner le modèle *n_iter* fois

Le paramètre déterminant est le nombre total de scores issus de la distribution nulle, *n_iter* × *m* : on peut donc choisir un *m* élevé pour réduire *n_iter* et gagner du temps, mais un *m* trop grand risque de déstabiliser l'apprentissage en ajoutant trop de

bruit dans l'espace des variables. En effet, les hyperparamètres d'entraînement comme la profondeur des arbres, le taux d'apprentissage ou les régularisations, sont calibrés sur le jeu de données initial pour obtenir un modèle avec une bonne précision mais en évitant le surapprentissage. L'ajout d'un trop grand nombre de probes peut rendre les paramètres d'entraînement inadaptés, dégrader les performances du modèle et la fiabilité des scores d'importance.

simThresh (seuil simulé)

Nous avons développé une méthode appelée *simThresh*. Contrairement à *PIMP* et à *1Probe/mProbes*, *simThresh* ne convertit pas les *feature importance*s en *p-value*, mais définit un seuil de *feature importance* au-delà duquel le seuil de significativité est dépassé.

Pour la classification, l'approche repose sur la création d'une variable simulée, dont les valeurs suivent des distributions spécifiques à chaque catégorie, avec un écart modulable entre les classes. On cherche l'écart δ entre deux distributions $\mathcal{D}(\mu, \sigma^2)$ telles que :

$$\tilde{v}_i \sim \begin{cases} \mathcal{D}(\mu + \delta, \sigma^2), & \text{si } y_i = A \\ \mathcal{D}(\mu - \delta, \sigma^2), & \text{si } y_i = B \end{cases} \quad \text{tel que} \quad |\text{pval}\left(t\text{-test}(\tilde{v} \mid y)\right) - p_{\text{cible}}| \leq \varepsilon$$

où :

- $\tilde{v}$ est la variable simulée
- y est la variable catégorielle (ex. classes A et B)
- $\mathcal{D}(\mu, \sigma^2)$ est une distribution de moyenne μ et de variance σ^2
- p_{cible} est la *p-value* à atteindre
- ε est l'écart de *p-value* permis

Un processus de recherche dichotomique est utilisé. Si la *p-value* obtenue est trop élevée, l'écart δ entre les distributions est augmenté et si elle est trop faible, il est réduit.

Pour la régression, la variable simulée est construite en ajoutant un bruit à la variable cible, avec un écart-type ajusté pour contrôler la *p-value* de corrélation entre la variable simulée et la variable cible. On cherche l'écart type σ d'une distribution $\mathcal{D}(0, \sigma^2)$ tel que :

$$\tilde{y} = y + \varepsilon, \quad \varepsilon \sim \mathcal{D}(0, \sigma^2), \quad \text{tel que} \quad |\text{pval}(\text{corr}(\tilde{y}, y)) - p_{\text{cible}}| \leq \delta$$

où

- $\tilde{y}$ est la variable simulée
- y est la variable cible
- $\mathcal{D}(\mu, \sigma^2)$ est une distribution de moyenne μ et de variance σ^2
- p_{cible} est la *p-value* à atteindre

- δ est l'écart de *p-value* permis

Un processus de recherche dichotomique est également utilisé. Si la *p-value* de corrélation est trop élevée, l'écart-type du bruit est diminué ; si elle est trop faible, il est augmenté.

Ce processus itératif se poursuit jusqu'à ce que la *p-value* atteigne une valeur cible prédéfinie, avec une marge de tolérance prédéfinie. La *feature importance* associée à cette variable simulée calibrée est ensuite utilisée comme seuil de significativité, auquel sont comparés les scores des variables réelles. Cette approche permet un contrôle du niveau de significativité sans conversion individuelle des scores. Elle offre également une charge computationnelle réduite : la phase de recherche dichotomique demande un calcul rapide de *t-test*, de test de Wilcoxon ou de corrélation par itération, et la seconde phase demande un entraînement unique du modèle de ML avec l'ajout de la variable simulée dans le jeu de données, contrairement à *PIMP* et *1Probe/mProbe* qui demandent de multiples réentraînements.

Algorithme *simThresh* pour la classification

```
Entrées
------------------
X              : matrice des variables réelles
y              : catégorie (A ou B)
D              : distribution de la variable simulée
p_cible        : p-value visée (ex. 0,05)
tolérance      : marge d'erreur autorisée sur la p-value (ex. 1×10^-3)
modèle         : algorithme de machine learning
FI             : méthode de feature importance

Étape 1 – Générer une variable de p-value proche de p_cible
------------------
delta_min  ← 0                    # écart initial minimum
delta_max  ← max(abs(X))          # écart initial maximum
mean_distrib ← mean(X)
sd_distrib ← sd(X)

Répéter
   delta ← (delta_min + delta_max) / 2        # point milieu
```

v_sim ← générer une variable suivant D(mean_distrib + delta, sd_distrib) pour A et suivant D(mean_distrib - delta, sd_distrib) pour B # écart contrôlé entre classes

p_actuelle ← *p-value* du test t entre v_sim|classe A et v_sim|classe B

Si |p_actuelle – p_cible| ≤ tolérance
convergence atteinte → sortir de la boucle

Si p_actuelle > p_cible # variable trop neutre
delta_min ← delta # on augmente l'écart la prochaine fois
Sinon # p_actuelle < p_cible, variable trop discriminante
delta_max ← delta # on réduit l'écart la prochaine fois
Jusqu'à convergence

Étape 2 – Calcul du seuil de *feature importance*

X_plus ← concaténer v_sim à X
entraîner modèle sur (X_plus, y)

importance_simulée ← *feature importance* de v_sim calculée avec FI

importances_réelles ← *feature importance*s des autres variables calculées avec FI

Étape 3 – Significativité des variables

Variable réelle i est *significative* si
importance_réelle[i] > importance_simulée

Sortie

- importance_simulée (seuil)
- liste des variables réelles significatives

Algorithme *simThresh* pour la régression

Entrées

X : matrice des variables réelles
y : variable cible (continue)
D : distribution de la variable simulée
p_cible : *p-value* visée pour le test de corrélation (ex. 0,05)
tolérance : marge d'erreur autorisée sur la *p-value* (ex. 1×10^{-3})
modèle: algorithme de *machine learning*
FI : méthode de *feature importance*
test_corr : type de corrélation (Pearson ou Spearman)

Étape 1 – Générer une variable de *p-value* proche de p_cible
---
sigma_min ← 0 # bruit minimal initial
sigma_max ← 5 × sd(y) # bruit maximal initial

Répéter
  sigma ← (sigma_min + sigma_max) / 2 # point milieu

  ε ← tirage de vecteur suivant D(0, sigma²)

  y_sim ← y + ε # variable simulée bruitée

  p_actuelle ← *p-value* du test_corr entre y_sim et y

  Si |p_actuelle – p_cible| ≤ tolérance
    convergence atteinte → sortir de la boucle

  Si p_actuelle > p_cible # corrélation trop faible
    sigma_min ← sigma # on réduit le bruit la prochaine fois
  Sinon # corrélation trop forte
    sigma_max ← sigma # on augmente le bruit la prochaine fois
Jusqu'à convergence

Étape 2 – Calcul du seuil de *feature importance*

```
    ------------------
    X_plus ← concaténer y_sim à X
    entraîner modèle sur (X_plus, y)

    importance_simulée ← feature importance de y_sim calculée avec FI
    importances_réelles ← feature importances des autres variables calculées avec
FI

    Étape 3 – Significativité des variables
    ------------------
    Variable réelle i est *significative* si
        importance_réelle[i] > importance_simulée

    Sortie
    ------
    - importance_simulée (seuil)
    - liste des variables réelles significatives
```

L'utilisateur peut choisir la forme de la distribution simulée. Par défaut, la variable est générée à partir d'une distribution centrée sur la moyenne globale du jeu de données, avec une variance identique à celle des variables réelles. Dans nos premiers essais, nous avons utilisé des distributions gaussiennes aléatoires, afin de simuler un comportement « naturel » de variable. Cependant, la variabilité entre deux tirages se révélait trop importante, rendant la recherche dichotomique instable et peu efficace : la convergence vers le seuil de *p-value* dépendait alors davantage du hasard que du processus de recherche.

Pour pallier cette limite, nous recommandons l'utilisation d'une distribution déterministe, correspondant aux quantiles théoriques de la loi normale, ce qui permet une répartition régulière et stable des valeurs simulées. Cette approche améliore fortement la vitesse et la robustesse de la convergence de l'algorithme.

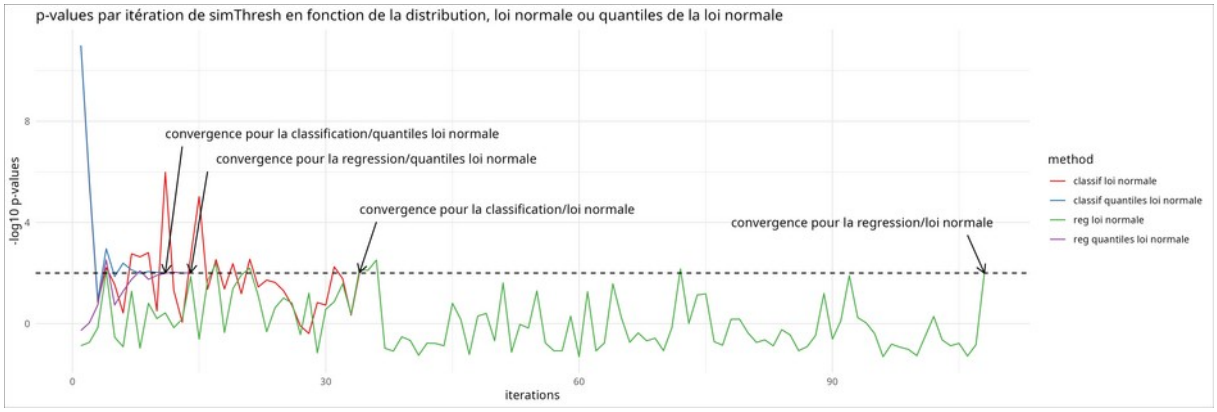

Figure 38: Exemples des valeurs des p-values à chaque itération de simThresh. Le jeu de données utilisé pour la classification correspond au scénario n°1, et celui pour la régression au scénario n°5 (voir paragraphe « Scénarios à détecter »). (Annexe CONTENT4PHD 0006_R_create_jeu de données_scenariosX10.ipynb, im/figs/0006_pvalues_iter_distrib.png)

On peut voir dans la figure ci-dessous la distribution des *p-values* obtenues en répétant 100 fois le tirage de deux échantillons : 50 observations issues de $\mathcal{N}(0,1)$ pour le groupe A, et 50 issues de $\mathcal{N}(0.3,1)$ pour le groupe B. À chaque répétition, un *t-test* est appliqué pour comparer les deux groupes.

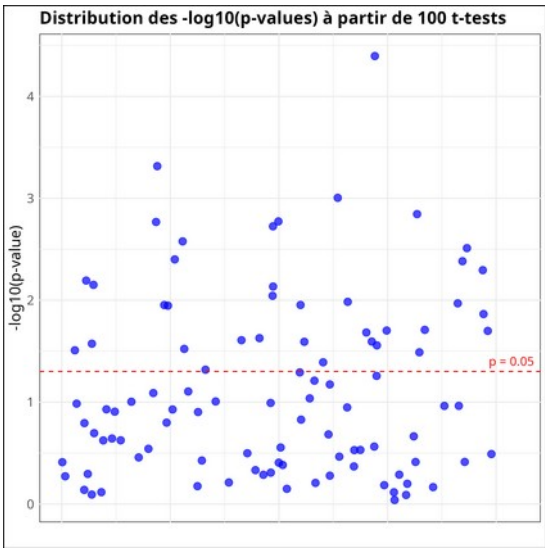

Figure 39: Distribution empirique des p-values (–log10) obtenues sur 100 t-tests entre deux groupes gaussiens de taille 50, avec un décalage de moyenne de 0.3. (Annexe CONTENT4PHD im/figs/0010_R_100_gaussian_tirage.png)

La médiane des *p-values* obtenues est de 0.11, mais on observe une forte variabilité, allant de 4,02e-5 à 0,91. Pourtant, tous les tirages proviennent des mêmes lois de probabilité. Ce phénomène illustre la difficulté de convergence en prenant des distributions gaussiennes.

Ces méthodes ont été intégrées dans le package *XAItest*, voir description dans section dédiée.

Scénarios à détecter

Pour illustrer le fonctionnement de *XAItest* nous avons créé des jeux de données synthétiques reflétant plusieurs scénarios de variables à détecter.

Scénario1 : Classification - différence de variance

Dix jeux de données ont été générés contenant 100 observations réparties équitablement entre deux classes A et B. Chaque jeu de données comprend :

- **10 variables bruitées** de `norm_noise01` à `norm_noise10`, générées indépendamment selon des lois normales $\mathcal{N}(\mu, \sigma)$, où μ et σ sont tirés aléatoirement pour chaque variable selon une distribution uniforme sur les intervalles respectifs $[-10, 10]$ et $[1, 5]$.
- **2 variables discriminantes** `diff_distrib01` et `diff_distrib02` différant entre les classes par leur moyenne, avec des distributions de la forme $\mathcal{N}(+\mu, \sigma)$ pour la classe A et $\mathcal{N}(-\mu, \sigma)$ pour la classe B, où $\mu \in 2, 4$ et $\sigma \in 2, 4$ selon la variable considérée.
- **1 variable à variance différente suivante la classe** `diffVar01` ayant une moyenne nulle dans les deux classes $(\mu = 0)$, mais des écarts-types différents : $\sigma = 1$ pour la classe A et $\sigma = 5$ pour la classe B.

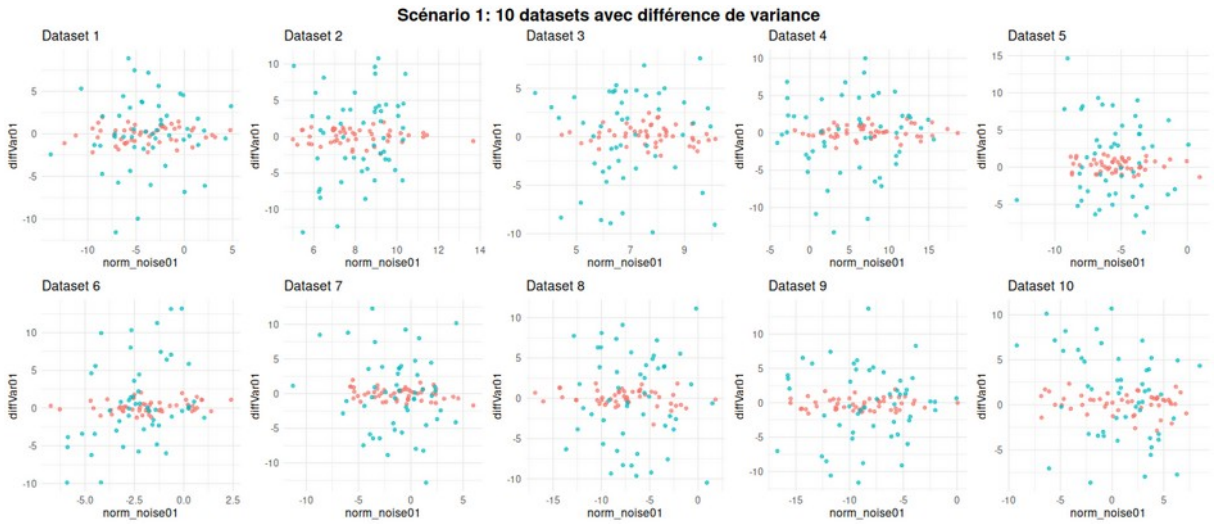

Figure 40: Exemples où la variable cible a une même moyenne dans les deux classes, mais une variance différente. On observe la diversité des résultats de tirages différents d'une même distribution aléatoire. (Annexe CONTENT4PHD 0006_R_create_jeu de données_scenariosX10.ipynb, im/figs/0006_scenario1_plotsX10.png)

Scénario 2 : Classification - distribution bimodale

Dix jeux de données ont été générés contenant 100 observations réparties équitablement entre deux classes A et B. Chaque jeu de données comprend :

- **10 variables bruitées** de `norm_noise01` à `norm_noise10`, générées indépendamment selon des lois normales $\mathcal{N}(\mu, \sigma)$, où μ et σ sont tirés aléatoirement pour chaque variable selon une distribution uniforme dans les intervalles respectifs $[-10, 10]$ et $[1, 5]$.
- **2 variables discriminantes** `diff_distrib01` et `diff_distrib02` différant entre les classes par leur moyenne, avec des distributions de la forme $\mathcal{N}(+3, 2)$ pour la classe A et $\mathcal{N}(-3, 2)$ pour la classe B dans le premier cas, puis $\mathcal{N}(+5, 2)$ pour la classe A et $\mathcal{N}(-4, 4)$ pour la classe B dans le second cas.
- **1 variable bimodale** `biDistrib` ayant, pour la classe A, une distribution bimodale résultant de deux lois normales $\mathcal{N}(1, 1)$ et $\mathcal{N}(11, 2)$, et pour la classe B une distribution unimodale $\mathcal{N}(5, 1)$.

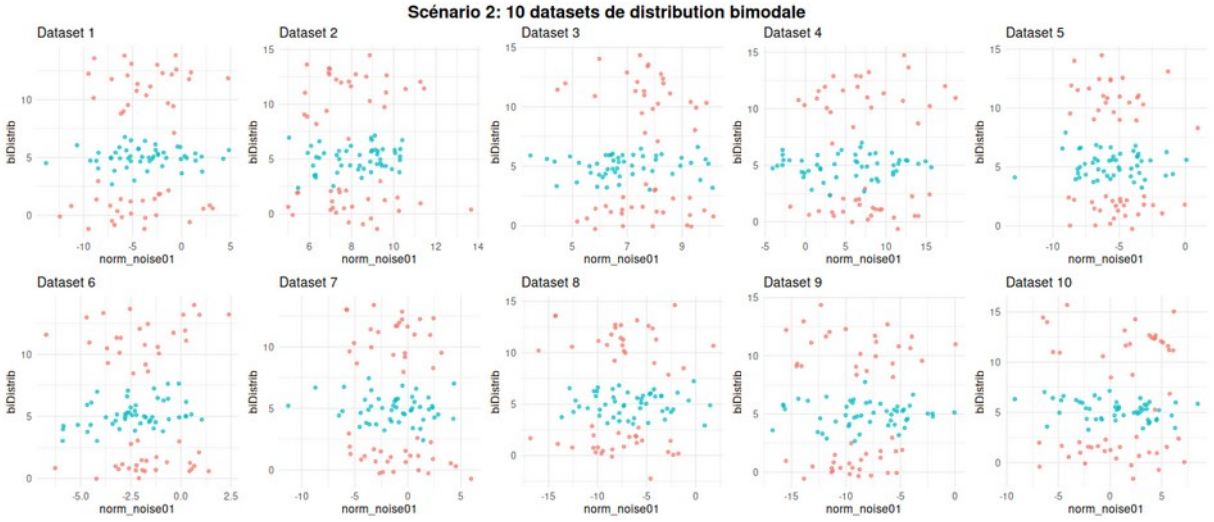

Figure 41: Diversité des résultats de tirages différents d'une même distribution aléatoire pour le cas des distributions bimodales. (Annexe CONTENT4PHD 0006_R_create_jeu de données_scenariosX10.ipynb, im/figs/0006_scenario2_plotsX10.png)

Scénario 3 : Classification - relation simple entre variables

Dix jeux de données ont été générés, chacun contenant 100 observations réparties équitablement entre deux classes A et B.

Chaque jeu de données contient :

- **10 variables bruitées** de `norm_noise01` à `norm_noise10`, générées indépendamment selon des lois normales $\mathcal{N}(\mu, \sigma)$, où μ et σ sont tirés aléatoirement pour chaque variable selon une distribution uniforme dans les intervalles respectifs $[-10, 10]$ et $[1, 5]$.

- **2 variables discriminantes** `diff_distrib01` et `diff_distrib02` différant entre les classes par leur moyenne, avec des distributions de la forme $\mathcal{N}(+2, 2)$ pour la classe A et $\mathcal{N}(-2, 2)$ pour la classe B dans le premier cas, puis $\mathcal{N}(+4, 4)$ pour la classe A et $\mathcal{N}(-4, 4)$ pour la classe B dans le second.
- **2 variables avec relation simple**, `relVar01` et `relVar02`, définies sur quatre sous-groupes de 25 individus chacun :
  - **Groupe A 1** (observations 1 à 25) : $\mathcal{N}(+2, 1)$ pour les deux variables
  - **Groupe A 2** (26 à 50) : $\mathcal{N}(-2, 1)$ pour les deux
  - **Groupe B 1** (51 à 75) : $\mathcal{N}(-2, 1)$ pour `relVar01` et $\mathcal{N}(+2, 1)$ pour `relVar02`
  - **Groupe B 2** (76 à 100) : $\mathcal{N}(+2, 1)$ pour `relVar01` et $\mathcal{N}(-2, 1)$ pour `relVar02`.

Les variables `relVar01` et `relVar02` suivent une logique de type XOR : les observations pour lesquelles les deux variables sont soit hautes, soit basses, appartiennent à la classe A et celles pour lesquels une seule des deux est haute appartiennent à la classe B. Ce schéma aboutit à une séparation non linéaire des classes, car les espérances conditionnelles marginales ne discriminent pas les classes :

$$\mathbb{E}[\text{relVar01} \mid Y = A] = \mathbb{E}[\text{relVar01} \mid Y = B], \text{ et}$$
$$\mathbb{E}[\text{relVar02} \mid Y = A] = \mathbb{E}[\text{relVar02} \mid Y = B]$$

avec Y la classe A ou B de l'observation

En revanche, une différence apparaît lorsqu'on conditionne sur la valeur de l'autre variable. Par exemple :

$$\mathbb{E}[\text{relVar01} \mid Y = A, \text{relVar02} > 0] > \mathbb{E}[\text{relVar01} \mid Y = B, \text{relVar02} > 0], \text{ et}$$
$$\mathbb{E}[\text{relVar01} \mid Y = A, \text{relVar02} < 0] < \mathbb{E}[\text{relVar01} \mid Y = B, \text{relVar02} < 0]$$

L'information discriminante réside donc dans l'interaction entre les deux variables, et non dans une différence de moyenne entre les deux classes A et B.

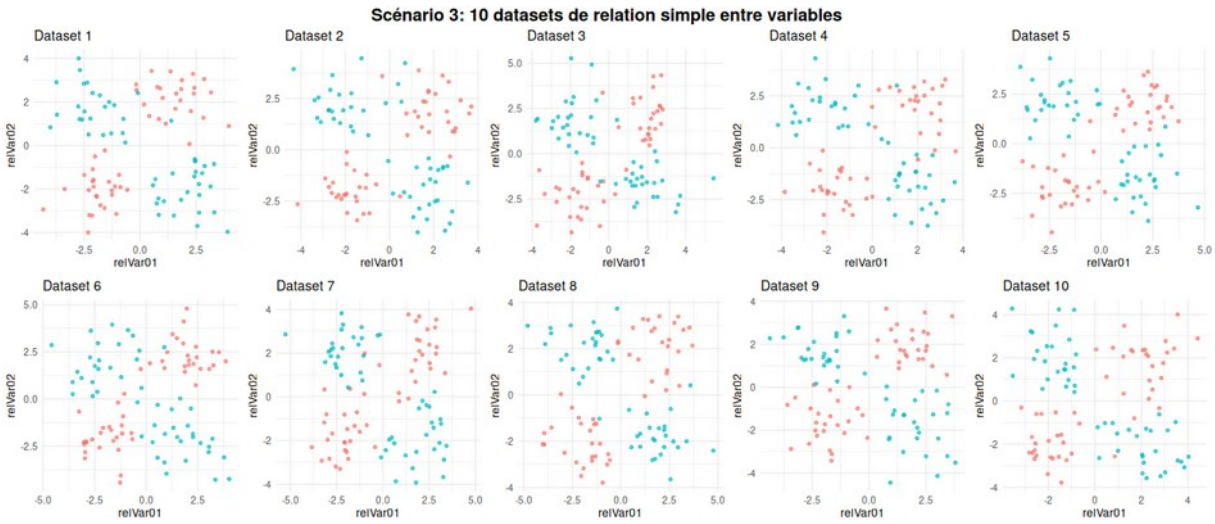

Figure 42: Diversité des résultats de tirages différents d'une même distribution aléatoire dans le cas d'une relation simple de type XOR entre deux variables. (Annexe CONTENT4PHD 0006_R_create_jeu de données_scenariosX10.ipynb, im/figs/0006_scenario3_plotsX10.png)

Scénario 4 : Classification - relation entre variables, cercles concentriques

Dix jeux de données ont été générés, chacun contenant 100 observations réparties équitablement entre deux classes A et B.

Chaque jeu de données contient :

- **10 variables bruitées** de `norm_noise01` à `norm_noise10`, générées indépendamment selon des lois normales $\mathcal{N}(\mu, \sigma)$, où μ et σ sont tirés aléatoirement pour chaque variable selon une distribution uniforme dans les intervalles respectifs $[-10, 10]$ et $[1, 5]$.
- **2 variables discriminantes** `diff_distrib01` et `diff_distrib02` différant entre les classes par leur moyenne, avec des distributions de la forme $\mathcal{N}(+2, 2)$ pour la classe A et $\mathcal{N}(-2, 2)$ pour la classe B dans le premier cas, puis $\mathcal{N}(+4, 4)$ pour la classe A et $\mathcal{N}(-4, 4)$ pour la classe B dans le second.
- **2 variables à structure géométrique**, `relVar01` et `relVar02` décrivant les coordonnées de trois cercles concentriques. Les observations de la classe A sont réparties en deux cercles concentriques : un cercle externe de rayon 4 contenant 35 individus, et un cercle interne de rayon 0.5 contenant 15 individus. Un bruit de distribution $\mathcal{N}(0, 0.2)$ est ajouté aux coordonnées pour simuler un effet de dispersion naturelle. Les observations de la classe B sont distribuées sur un cercle unique de rayon 2.5, centré au même endroit, avec un bruit de même distribution que pour la classe A.

Les variables `relVar01` et `relVar02` définissent une séparation non linéaire entre les classes, qui ne peut être détectée par une différence de moyenne ou de variance. En effet, les espérances conditionnelles marginales restent similaires entre les classes :

$$\mathbb{E}[\text{relVar01} \mid Y = A] \approx \mathbb{E}[\text{relVar01} \mid Y = B], \text{ et}$$

$$\mathbb{E}[\text{relVar02} \mid Y = A] \approx \mathbb{E}[\text{relVar02} \mid Y = B]$$

La séparation entre les classes apparaît uniquement à travers la structure géométrique globale des données, et nécessite donc des modèles capables de capturer des interactions non linéaires complexes dans le plan formé par ces deux variables.

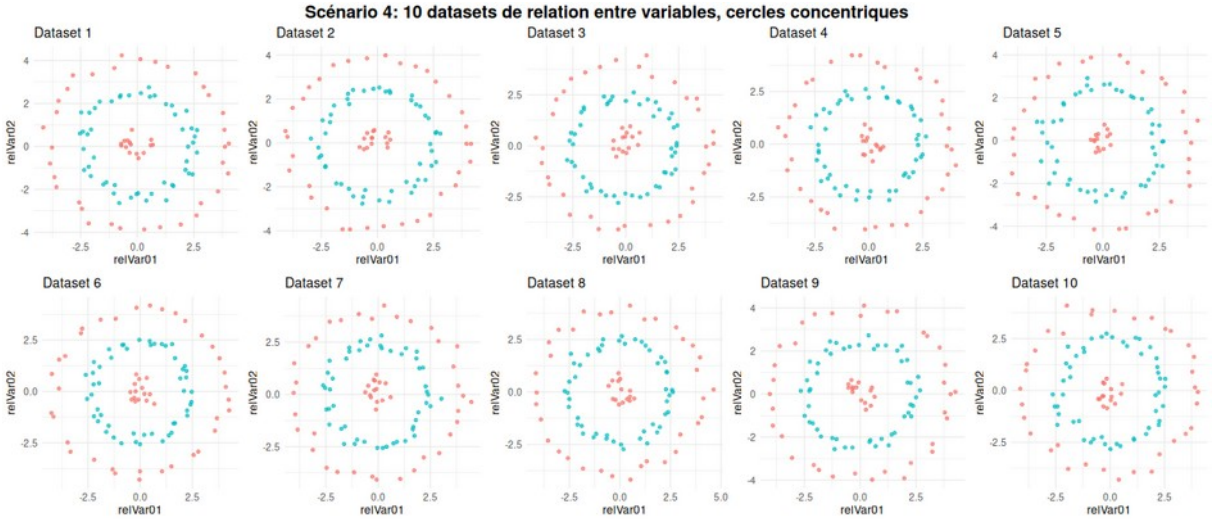

Figure 43: Diversité des résultats de tirages différents de mêmes distributions aléatoires pour le cas des cercles concentriques. (Annexe CONTENT4PHD 0006_R_create_jeu de données_scenariosX10.ipynb, im/figs/0006_scenario4_plotsX10.png)

Scénario 5 : Régression - hyperbole/parabole

Dix jeux de données ont été générés, chacun contenant 100 observations avec une variable cible continue simulant un problème de régression.

Chaque jeu de données contient :

- **10 variables bruitées de lois normales**, de `norm_noise01` à `norm_noise10`, générées indépendamment selon des lois normales $\mathcal{N}(\mu, \sigma)$, où μ et σ sont tirés aléatoirement pour chaque variable selon une distribution uniforme dans les intervalles respectifs $[-10, 10]$ et $[1, 5]$.
- **10 variables bruitées issues de lois uniformes**, de `unif_noise01` à `unif_noise10`, générées selon des lois $\mathcal{U}(a, b)$, où la borne a est tirée aléatoirement d'une distribution $\mathcal{U}(-10, 0)$ et b d'une distribution $\mathcal{U}(0, 10)$.
- **Une variable cible continue**, y, une parabole générée à partir de `norm_noise01`. Plus précisément, y est définie comme :

$$y_i = a \cdot (x_i - x_{\min}) \cdot (x_i - x_{\max})$$

$$a = \frac{k}{(h - x_{\min})(h - x_{\max})}$$

$$h = \frac{x_{\min} + x_{\max}}{2}$$

$$k = \frac{x_{\max}}{2}$$

```r
transfo_parab <- function(xs){
    x1 <- min(xs)
    x2 <- max(xs)
    h <- (x1 + x2) / 2
    k <- max(xs)/2
    a <- k / ((h - x1) * (h - x2))
    y <- a * (xs - x1) * (xs - x2)

    return (y)
}
```

Figure 44: Fonction R de génération de la parabole. (Annexe CONTENT4PHD 0006_R_create_jeu de données_scenariosX10.ipynb)

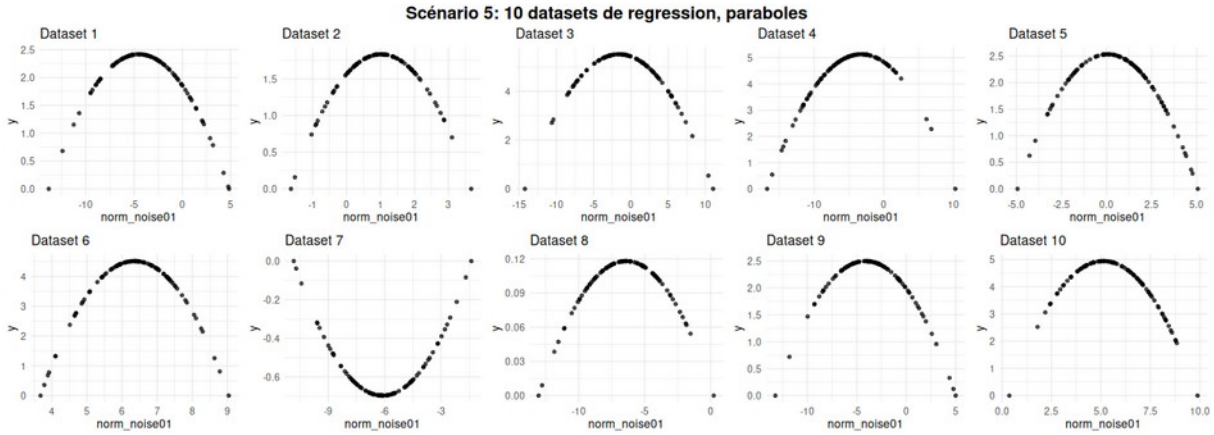

Figure 45: Diversité des résultats de tirages différents de mêmes distributions aléatoires pour le cas des paraboles. (Annexe CONTENT4PHD 0006_R_create_jeu de données_scenariosX10.ipynb, im/figs/0006_scenario5_plotsX10.png)

Scénario 6 : Régression - sinusoïde à amplitude croissante

Dix jeux de données ont été générés, chacun contenant 100 observations avec une variable cible continue simulant un problème de régression.

Chaque jeu de données contient :

- **10 variables bruitées de lois normales**, de `norm_noise01` à `norm_noise10`, générées indépendamment selon des lois normales $\mathcal{N}(\mu, \sigma)$, où μ et σ sont tirés aléatoirement pour chaque variable selon une distribution uniforme dans les intervalles respectifs $[-10, 10]$ et $[1, 5]$.

- **10 variables bruitées issues de lois uniformes**, de `unif_noise01` à `unif_noise10`, générées selon des lois $\mathcal{U}(a, b)$, où la borne a est tirée aléatoirement d'une distribution $\mathcal{U}(-10, 0)$ et b d'une distribution $\mathcal{U}(0, 10)$.
- **Une variable cible continue**, y, une sinusoïde dont l'amplitude augmente avec x, générée à partir de `unif_noise01`. Plus précisément, y est définie comme :

$$y_i = \sin(x_i) \cdot x_i \,,$$

où $x_i = \texttt{unif_noise01}_i$ c'est-à-dire une variable uniforme sur $[0, 10]$.

```r
croissSinusoid <- function(x){
    return (sin(x) * x)
}
```

Figure 46: Fonction R de génération de la sinusoïde à amplitude croissante. (Annexe CONTENT4PHD 0006_R_create_jeu de données_scenariosX10.ipynb)

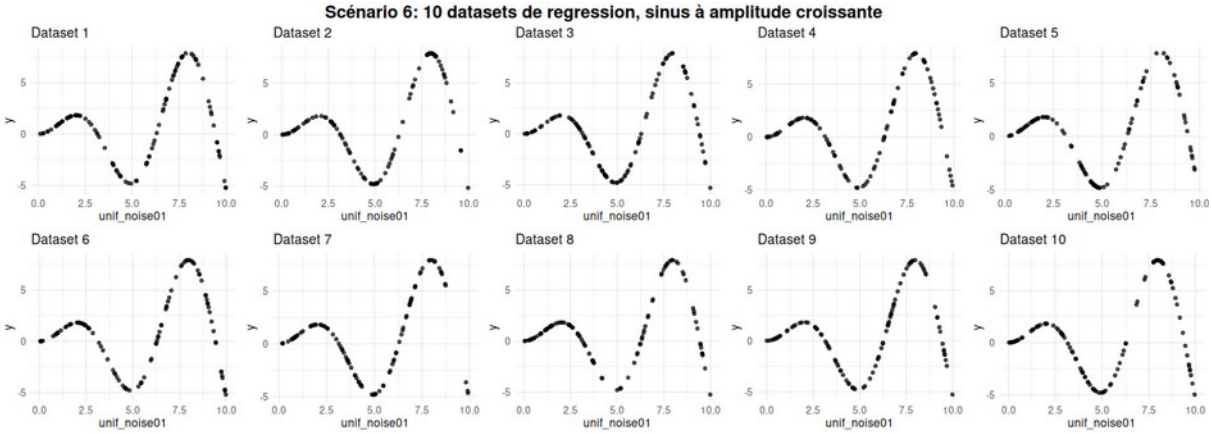

Figure 47: Diversité des résultats de tirages différents de mêmes distributions aléatoires pour le cas des sinusoïdes à amplitude croissante. (Annexe CONTENT4PHD 0006_R_create_jeu de données_scenariosX10.ipynb, im/figs/0006_scenario6_plotsX10.png)

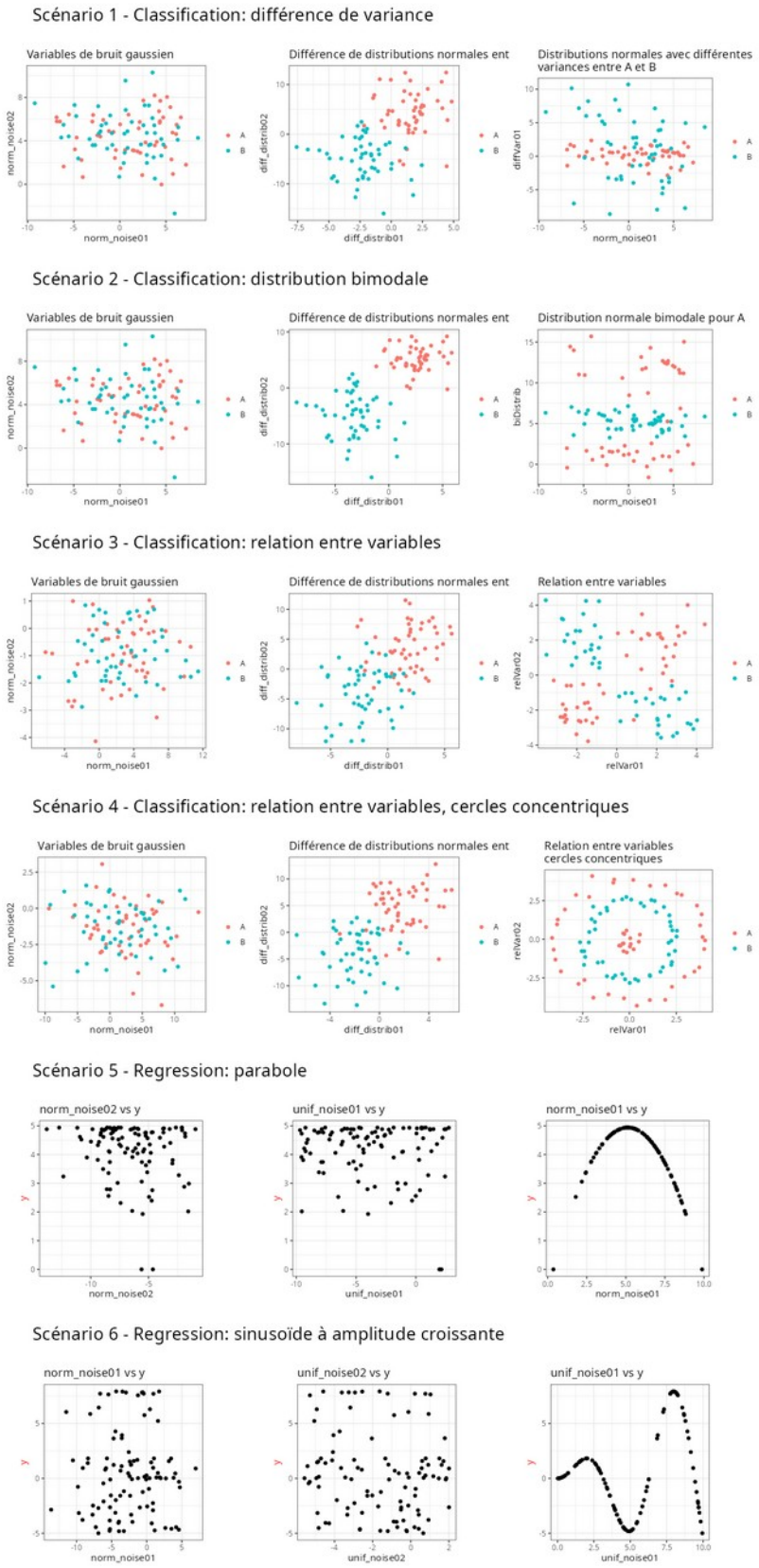

Figure 48: Récapitulatif des 6 scénarios simulés : variables bruitées, différences de distributions entre les classes A et B, et configurations spécifiques illustrant des cas de classification ou de régression complexes. (Annexe CONTENT4PHD 0006_R_create_jeu de données_scenariosX10.ipynb, im/figs/0006_panel_scenarios.png)

Package *R Bioconductor XAItest*

Nous avons développé un package *R* disponible sur *Bioconductor* à partir de la version 3.21, nécessitant *R* version ≥ 4.5 dont le but est d'aider à l'évaluation et à la comparaison des méthodes XAI pour la découverte de variables d'intérêt, basé sur le *workflow* présenté au début de la section sur l'axe 2. Concrètement, *XAItest* compare des mesures de significativité classiques comme les *p-values* de *t-test*s, ou de corrélations avec des estimations d'importance de variables issues de modèles de *machine learning*. Il prend par défaut *random forest* comme modèle de ML, et les méthodes de *feature importance SHAP* et *LIME*. Il intègre en outre notre algorithme *simThresh*, un système de données simulées calibrées pour faire le lien entre *p-values* et *feature importance*. Les options de fonctions customs permettent d'utiliser facilement d'autres algorithmes de ML (DT, SVM et MLP) et de *feature importance* (*decrease d'accuracy*, score de Gini, score d'Olden) à l'aide d'un script disponible en annexe (*Annexe XAI_SIMU 0006_R_run10X_simData=F.ipynb*). Enfin, le package propose des visualisations intuitives, comme des tableaux avec des codes couleur et des graphes, afin de faciliter l'interprétation des résultats.

La principale fonction du package est ***XAI.test()***. Elle procède à l'ensemble des analyses statistiques selon les arguments en paramètre, calcule les *feature importance*s et les *p-values* associées définies dans les arguments. Voici la liste des arguments principaux de la fonction. La documentation exhaustive est disponible dans les vignettes *Bioconductor* et le repertoire github. (https://www.bioconductor.org/packages/devel/bioc/html/XAItest.html, https://github.com/GhislainFievet/XAItest) :

- ***data :*** Jeu de données d'entrée, soit un *data.frame* (observations en lignes, variables en colonnes), soit un objet SummarizedExperiment contenant les données dans le slot 'assay' et la variable cible dans les metadata.
- ***y*** : Nom de la variable cible, colonne du *data frame* ou metadata de SummarizedExperiment à prédire.
- ***featImpAgr :*** Méthode d'agrégation des *feature importance*s lorsque plusieurs valeurs sont obtenues, par défaut "mean".
- ***simData :*** Booléen indiquant si la méthode *simThresh* doit être exécutée, une *feature* simulée est alors ajoutée aux données. Par défaut FALSE. En activant TRUE, *XAI.test* génère une variable artificielle "*simFeat*" servant de référence pour relier *feature importance*s et *p-values*.
- ***simMethod* :** Méthode de simulation utilisée pour *simFeat*, "*regrnorm*" pour des points simulés suivant les quantiles théoriques d'une distribution gaussienne, par défaut, ou "*rnorm*" pour une distribution gaussienne classique.
- ***simPvalTarget :*** *p-value* cible pour la *feature* simulée si simData=TRUE. Par exemple 0.01 produira une variable simulée suffisamment différenciée pour obtenir une *p-value* ≈ 0.01 (ajustée) au *t-test*. Cette *feature* "*simFeat*" permet de déterminer le niveau de *feature importance* considéré comme significatif.
- ***adjMethod :*** Méthode de correction des *p-values* pour la FDR, par défaut *bonferroni*, option *bh* possible.

- ***defaultMethods :*** Liste des méthodes à exécuter. Par défaut inclut *t-test*, *ebayes* (modération Bayesienne de *limma*), cor (corrélation), lm (modèle linéaire), *rf* (*Random Forest*), *shap* et *lime*. Cette liste peut être modifiée pour n'exécuter qu'un sous-ensemble de méthodes.
- ***customPVals, customFeatImps, customFIPV :*** Listes de fonctions personnalisées pour calculer des *p-values*, *feature importance* ou une combinaison des deux, respectivement. Chaque fonction doit retourner un *data.frame* de *p-values*, de *p-values* ajustées et/ou de *feature importance*, respectivement. Ces arguments permettent à *XAItest* d'assurer sa mission d'analyse de tout type de méthodes XAI.
- ***caretMethod :*** Méthode d'apprentissage supervisé sous-jacente, utilisée via le package caret, pour entraîner le modèle de *feature importance* par défaut. Par défaut *rf*, pour *random forest*. On peut par exemple utiliser *xgbTree* pour utiliser *XGBoost*. Une installation supplémentaire de packages peut être nécessaire.
- ***verbose :*** Booléen, par défaut FALSE. Si TRUE, la fonction imprime des informations détaillées pendant l'exécution.

Exemple d'utilisation pour un cas de classification

Le package contient un fichier d'exemple correspondant au scénario 2 décrit section précédente, *extdata/dfClassif.txt*. Il s'agit du scénario de la double distribution contenant 50 échantillons A , 50 échantillons B, 10 variables de bruit gaussien, deux variables de distribution gaussienne avec différence de moyenne entre A et B, et une variable avec distribution bimodale pour A. On peut voir ci-dessous différents *scatters plots* issus de ce jeu de données.

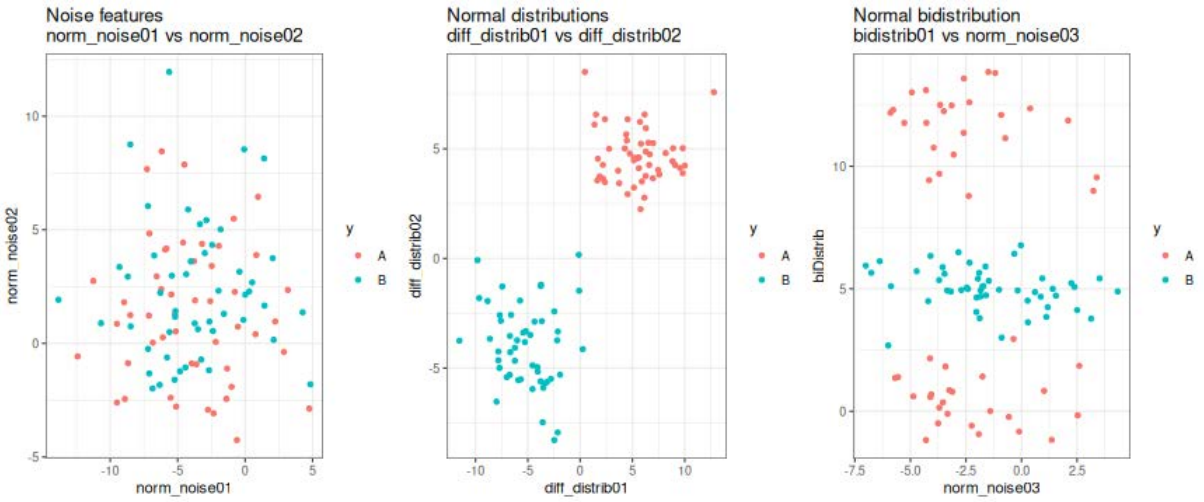

Figure 49: Scatter plots du fichier d'exemple "extdata/dfclassifieur.txt". Réf. : https://www.bioconductor.org/packages/release/bioc/vignettes/XAItest/inst/doc/XAItest.html

Exemple d'application :

```
library("XAItest")
df_path <- system.file("extdata", "dfClassif.txt", package="XAItest")
jeu de données_classif <- read.table(df_path)
objXAI <- XAI.test(        jeu de données_classif,
                           "y",
                           simData = TRUE,
                           simPvalTarget = 0.01)
```

L'objet *objXAI* contient alors plusieurs informations, le temps de calcul, les différentes *p-values*, les *feature importance*s et les prédictions des différents modèles de ML utilisés. L'objet peut alors être utilisé par d'autres fonctions du package.

mapPvalImportance()

Cette fonction permet d'afficher les *p-values* et *feature importance*s dans un tableau avec un code couleur pour mettre en valeur les éléments significatifs. Les cases vertes correspondent aux *feature importance*s au dessus du niveau le plus bas des *feature importance*s dont la *p-value* est au dessus du seuil déterminé. Ci-dessous un exemple avec les *feature importance*s dont le *ttest_adjPval* est en dessous de 0.01 (le paramètre *simPvalTarget*) surlignés en vert.

```
mpi <- mapPvalImportance(objXAI)
mpi$dt
```

Show 10 entries Search:

	ttest_pval	ttest_adjPval	ebayes_pval	ebayes_adjPval	lm_coef	lm_pval	lm_adjPval	RF_feat_imp	SHAP_feat_imp	LIME_feat_imp
diff_distrib02	1.02e-43	1.42e-42	8.04e-48	1.13e-46	-0.0668	2.21e-21	3.09e-20	16.4	0.251	0.212
diff_distrib01	7.93e-37	1.11e-35	1.07e-37	1.49e-36	-0.0312	1.24e-11	1.73e-10	19.5	0.244	0.234
simFeat	0.000702	0.00983	0.000627	0.00878	-0.00767	0.125	1	1.19	0.000386	0.0148
norm_noise03	0.0797	1	0.0792	1	0.00275	0.605	1	1.16	0.000232	0.0081
norm_noise08	0.111	1	0.106	1	0.000455	0.899	1	0.378	0.00023	0.0083
norm_noise09	0.144	1	0.144	1	0.0035	0.521	1	0.746	0.000192	0.0135
norm_noise02	0.146	1	0.143	1	0.00461	0.305	1	0.565	0.000118	0.017
biDistrib	0.208	1	0.199	1	0.00321	0.411	1	8.25	0.000669	0.0645
norm_noise06	0.22	1	0.222	1	-0.00817	0.196	1	0.243	0.000176	0.00663
norm_noise10	0.521	1	0.554	1	0.019	0.143	1	0.245	0.000109	0.00676

Showing 1 to 10 of 14 entries Previous 1 2 Next

Figure 50: Résultat de la fonction mapPvalImportance(). Réf. : https://www.bioconductor.org/packages/release/bioc/vignettes/XAItest/inst/doc/XAItest.html

On voit sur la table ci-dessus que la variable *biDistrib* correspondant au scénario de la distribution gaussienne bimodale a bien été détectée par la méthode *simThresh*.

plotModel()

Lors de l'exécution de la fonction *XAI.test()* plusieurs modèles sont entraînés sur le jeu de données d'entrée. L'objet *objXAI* contient alors les prédictions de la classe cible, dans notre exemple 'A' ou 'B'. La fonction *plotModel()* permet de récupérer et d'afficher facilement ces prédictions sous forme de *scatter plot*.

```
p1 <- plotModel(objXAI, "lm_pval", "biDistrib", "simFeat")
p2 <- plotModel(objXAI, "RF_feat_imp", "biDistrib", "simFeat")
p3 <- plotModel(objXAI, "SHAP_feat_imp", "biDistrib", "simFeat")
p4 <- plotModel(objXAI, "LIME_feat_imp", "biDistrib", "simFeat")

# Affichage en grille
grid.arrange(p1, p2, p3, p4, ncol = 2, nrow = 2)
```

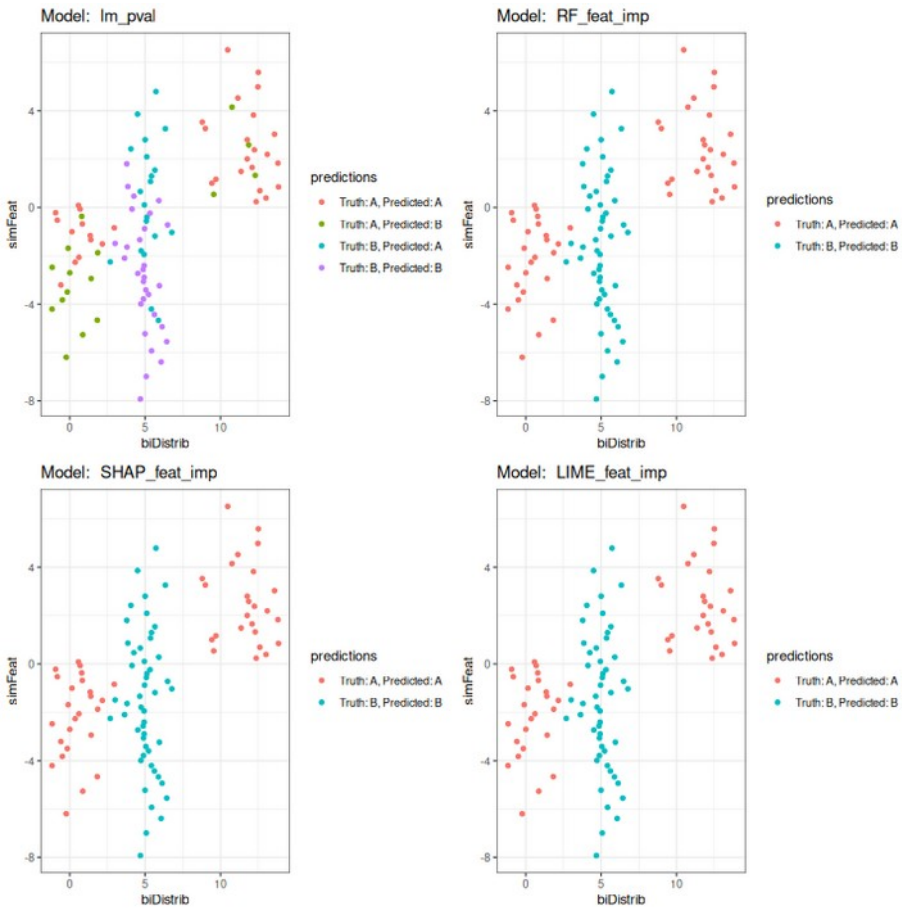

Figure 51: Illustration de l'utilisation de la fonction plotModel. Réf. : https://www.bioconductor.org/packages/release/bioc/vignettes/XAItest/inst/doc/XAItest.html

On voit sur les *plots* ci-dessus que les modèles RF entraînés pour *SHAP*, *LIME*, et pour la *feature importance* native de RF, catégorisent correctement les classes A et B. Au contraire, le modèle de régression linéaire présente un grand nombre de mauvaises classifications.

Exemple d'utilisation pour un cas de régression

Dans le même esprit voici un exemple d'utilisation de régression correspondant au scénario n°5 de parabole. Le jeu de données est disponible dans le package à *extdata/dfRegress.txt*.

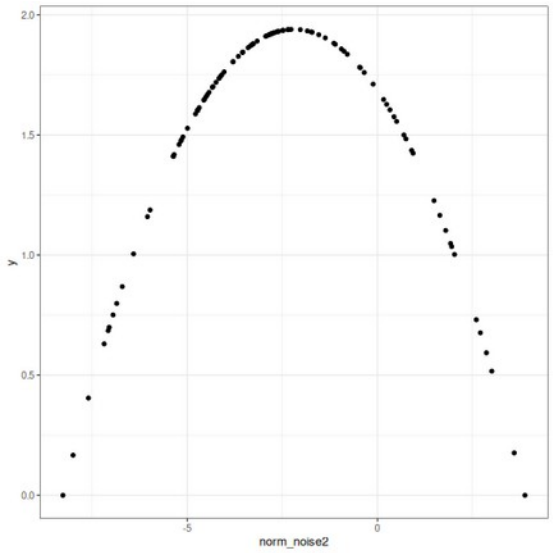

Figure 52: Scatter plot représentant la variable cible et la variable d'intérêt du jeu de données dataset_regress. Réf. : https://www.bioconductor.org/packages/release/bioc/vignettes/XAItest/inst/doc/XAItest.html

```
df_path <- system.file("extdata", "dfRegress.txt", package="XAItest")
jeu de données_regress <- read.table(df_path)
regr_results <- XAI.test(    jeu de données_regress,
                             "y",
                             simData=TRUE,
                             simPvalTarget = 0.01)

# Affichage du tableau avec code couleur pour les variables significatives
mpi <- mapPvalImportance(       regr_results,
                             refPvalColumn = "cor_adjPval",
                             refPval = 0.01)
mpi$dt
```

Show 10 entries Search:

	cor	cor_pval	cor_adjPval	lm_coef	lm_pval	lm_adjPval	RF_feat_imp	SHAP_feat_imp	LIME_feat_imp
simFeat	0.311	0.00162	0.00974	0.0921	0.00113	0.00681	2.48	0.00264	0.0122
unif_noise1	0.208	0.038	0.228	0.35	0.0317	0.19	1.95	0.00287	0.0145
unif_noise2	0.147	0.143	0.859	0.127	0.121	0.729	1.54	0.00313	0.0163
unif_noise3	0.125	0.215	1	0.0168	0.538	1	1.42	0.00505	0.0124
norm_noise1	0.0612	0.546	1	0.00252	0.918	1	1.65	0.00169	0.0123
norm_noise2	-0.0112	0.912	1	-0.00851	0.608	1	12.9	0.381	0.557

Showing 1 to 6 of 6 entries Previous 1 Next

Figure 53: Résultat de la fonction mapPvalImportance(). Réf. : https://www.bioconductor.org/packages/release/bioc/vignettes/XAItest/inst/doc/XAItest.html

On voit sur la table ci-dessus que la variable *norm_noise2* à partir de laquelle la parabole de la variable cible '*y*' a bien été détectée par la méthode *simThresh*.

On peut également utiliser la fonction *plotModel* pour voir comment les modèles ont réussi à modéliser le comportement de la variable cible '*y*'. On voit que la régression linéaire '*lm*' n'arrive pas à coller à la parabole, car cette dernière n'est pas linéaire, alors que le modèle RF utilisé pour calculer les valeurs de *SHAP* s'en sort mieux.

```
p1 <- plotModel(regr_results, "lm_pval", "norm_noise2")
p2 <- plotModel(regr_results, "SHAP_feat_imp", "norm_noise2")

# Affichage en grille
grid.arrange(p1, p2, ncol = 2, nrow = 1)
```

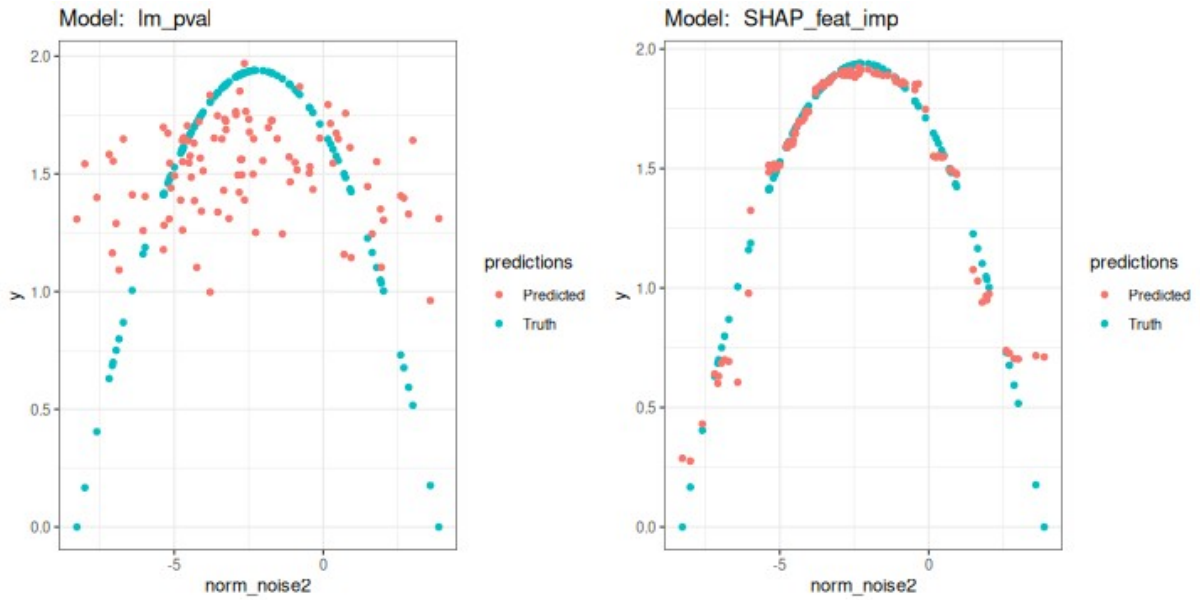

Figure 54: Résultats de plotModel pour la régression linéaire et RF. Réf. : https://www.bioconductor.org/packages/release/bioc/vignettes/XAItest/inst/doc/XAItest.html

Script pour utilisation avancée du package

L'argument *customFIPV* de la fonction *XAI.test()* permet de définir des fonctions custom générant à la fois *feature importance*s et *p-values*. C'est cet argument qui est utilisé pour étudier les méthodes de conversion *feature importance => p-value* comme *PIMP* ou mProbes. La fonction utilisée, ***fctParamFIPV()*,** se trouve dans le script disponible dans l'annexe *XAI_SIMU : 0006_R_run10X_simData=F.ipynb*.

fctParamFIPV()

- ***data :*** Jeu de données de type *data.frame* contenant les variables explicatives et la variable cible.
- ***categ*** (par défaut = "*y*") : Nom de la colonne correspondant à la variable cible.
- ***modelType*** (valeurs possibles : "*classification*", "*regression*", "*default*") : Type d'apprentissage. Si "*default*", le type est déduit automatiquement du type de la variable *categ*.
- ***ml_algo*** (par défaut = "*rf*") : Algorithme de ML utilisé pour entraîner le modèle :
  - "*rf*" pour *Random Forest*
  - "*mlp*" pour perceptron multicouche (Keras)
  - "*dt*" pour arbres de décision
  - "*svm-rbfdot*", "*svm-polydot*" pour SVM avec différents noyaux.
- ***fi_algo*** (par défaut = "*accuracy*") : Méthode d'estimation de l'importance des variables :
  - "*accuracy*", "*gini*" (*Random Forest natif*)

- "*shap*" (*SHAP avec fastshap*)
- "*Olden*" (pour mlp)

- ***fi_transfo*** (par défaut = "*none*") : Transformation appliquée aux valeurs d'importance, *jitter* utile pour les cas d'égalité :
  - "*none*" : aucune transformation
  - "*rank*" : classement des *feature importance*s
  - "*jitter*" ou "*rank+jitter*" : ajout d'un bruit aléatoire pour traiter les cas d'égalités
- ***pv_algo*** (par défaut = "*pimp*") : Méthode de calcul des *p-values* :
  - "*pimp*" : permutation de la variable cible (méthode *PIMP*)
  - "*probes*" : ajout de colonnes aléatoires (méthode *1Probe/mProbe*s)
- ***adjMethod*** (par défaut = "*bonferroni*") : Méthode d'ajustement des *p-values* multiple, compatible avec la fonction *p.adjust()* de *R* (ex. "*BH*", "*fdr*", "*holm*"...).
- ***n_perm*** (par défaut = 100) : Nombre de permutations utilisées dans les méthodes *PIMP* ou *1Probe/mProbe*s.
- ***n_probes*** (par défaut = 10) : Nombre de colonnes "*probe*" aléatoires ajoutées à chaque itération si pv_algo = "*probes*".
- ***shap_nsim*** (par défaut = 100) : Nombre de simulations pour estimer les valeurs *SHAP* via *fastshap*.

Exemple d'utilisation

```
objXAI ← XAI.test(data,
                  y="y",
                  simData=F,
                  customFIPV=list("try1"=function(data, ...) {
                        fctParamFIPV(     data = data,
                                          fi_algo = 'SHAP',
                                          pv_algo = 'pimp',
                                          ml_algo = 'mlp',
                                          fi_transfo = 'none',
                                          n_perm = 100,
                                          shap_nsim = 100,
                                          ...)
                  }))
```

Framework de comparaison de méthodes de découverte

Dans cette étude, nous avons développé un cadre d'évaluation des méthodes XAI pour la sélection de variables à l'aide du package *XAItest*. L'objectif est de comparer différentes combinaisons d'algorithmes d'apprentissage automatique, de méthodes de calcul des *feature importance*s, et de techniques de conversion statistique de ces *feature importance*s en *p-values* ou en seuil de significativité. Connaissant les variables d'intérêt par design, nous avons évalué chaque combinaison sur des jeux de données synthétiques simulant des scénarios que les tests statistiques classiques peinent à détecter.

Scénarios simulés, 6 configurations décrites dans une section précédente, chacune déclinée en 10 jeux de données :

1. **Classification, différences de variances** : deux classes avec variances différentes mais moyennes identiques.
2. **Classification, distributions bimodales** : classe A unimodale, classe B bimodale.
3. **Classification, relations XOR entre variables** : interactions non linéaires de type logique XOR.
4. **Classification, cercles concentriques** : séparation géométrique non linéaire entre les classes.
5. **Régression, parabole** : variable cible construite comme une parabole à partir d'une des variables explicatives.
6. **Régression, sinusoïde à amplitude croissante** : cible issue d'une fonction sinusoïdale non linéaire construite à partir d'une des variables explicatives.

Algorithmes de *machine learning* testés :

- *Decision Tree*
- *Random Forest*
- SVM, noyaux linéaire et radial
- *Multi-Layer Perceptrons*

Méthodes de *feature importance* utilisées :

- score de Gini pour DT et RF
- *Accuracy decrease* pour RF
- *SHAP* pour tous les algorithmes de ML
- Olden pour MLP

Méthodes de conversion en *p-value* ou seuils de significativité :

- ***PIMP***: permutations de la variable cible pour générer une distribution nulle.
- **mProbe** : ajout de variables aléatoires, les probes, pour simuler une distribution nulle.
- ***simThresh*** (méthode développée pour la thèse) : génération d'une variable simulée calibrée pour atteindre une *p-value* cible, et utilisation de son importance comme seuil.

Enlever les variables triviales

Dans les jeux de données, certaines variables ont été conçues pour être très facilement détectables, notamment celles qui présentent une simple différence de moyenne entre les classes. Leur présence peut compliquer la tâche des algorithmes d'apprentissage automatique qui pourraient se contenter de ces variables triviales pour réaliser leurs prédictions, au détriment des réelles variables d'intérêt des différents scénarios. Or elles sont aisément identifiables par des méthodes statistiques classiques comme le *t-test*. Donc pour mieux tester la capacité des méthodes à révéler les variables d'intérêt complexes nous avons adopté une approche en deux temps : une première analyse est menée sur le jeu de données complet, puis une seconde est réalisée après avoir retiré les variables détectables par *t-test*. Comparer les jeux complets et ceux simplifiés (ou réduits) permet d'analyser dans quelle mesure les méthodes s'appuient sur l'ensemble des variables d'intérêt ou se restreignent aux variables les plus facilement détectables, influençant leur capacité de découverte.

Définition du succès

L'évaluation des méthodes repose sur quatre critères définis à partir de deux types de jeux de données, complets ou simplifiés, et de deux stratégies de sélection, par seuil de *p-value* ou par classement des *p-values*. Une méthode est considérée comme réussie si elle identifie exactement les variables d'intérêt attendues selon l'un des critères suivants :

1. Les variables à détecter correspondent exactement à celles ayant une *p-value* inférieure à 0.05 dans le jeu de données complet.
2. Les variables à détecter correspondent exactement à celles ayant une *p-value* inférieure à 0.05 dans le jeu simplifié (sans les variables triviales).
3. Les variables à détecter correspondent exactement à celles présentes dans le top du classement des *feature importance*s sur le jeu complet.
4. Les variables à détecter correspondent exactement à celles présentes dans le top du classement des *feature importance*s sur le jeu simplifié.

Ces quatre critères sont hiérarchisés du plus exigeant au plus permissif : réussir le critère (1) suppose que la méthode est capable de détecter les bonnes variables avec une significativité statistique, même en présence des variables triviales. Le critère (2) allège cette contrainte en supprimant ces variables triviales, identifiables par *t-test*, ce qui peut aider certaines méthodes à se concentrer sur les variables complexes. Les

critères (3) et (4), basés sur le rang plutôt que sur une *p-value*, sont plus souples mais restent informatifs quant à la capacité des méthodes à hiérarchiser correctement les variables pertinentes. Ces 4 critères permettent une évaluation progressive et nuancée des performances des méthodes testées.

Importance de l'évaluation du temps de calcul

Un point important de notre étude concerne le temps de calcul associé à chaque combinaison testée. Selon les méthodes utilisées, ce temps peut aller d'un simple entraînement rapide suivi d'une seule inférence, jusqu'à des procédures plus lourdes impliquant 100 entraînements, potentiellement lents, et 100 inférences par entraînement. Prenons deux exemples concrets pour illustrer ces différences.

Prenons la combinaison *Decision Tree* + importance de Gini + *simThresh*. L'arbre de décision s'entraîne très rapidement, l'importance de Gini des variables est disponible immédiatement après l'entraînement, et la méthode *simThresh* permet d'évaluer la significativité sans nécessiter ni ré-entraînement ni inférences supplémentaires.

À l'inverse, imaginons que l'on choisisse la combinaison MLP + *SHAP* + *PIMP* (ou mProbes). L'entraînement du MLP est beaucoup plus long, et nous avons choisi 100 inférences pour le calcul de de la *feature importance* avec *SHAP*. En ajoutant une méthode comme *PIMP* ou mProbes, il faut répéter cet entraînement, et toutes les inférences *SHAP* associées, 100 fois pour construire une distribution nulle. Ce qui illustre comment le couplage de méthodes peut entraîner un effet combinatoire du temps de traitement.

Dans nos jeux simulés, avec 100 échantillons et une vingtaine de variables, ces calculs restent réalisables. Mais dans la pratique, les jeux de données atteignent des tailles plus élevées. En scRNA-seq il est courant d'atteindre 30 000 gènes, 10 types cellulaires, et 1 000 cellules par type, soit 10 000 échantillons. Les puces de méthylation montent jusqu'à 850 000 variables, et les jeux de protéomique ou RNA-seq bulk comptent souvent entre 2 000 et 20 000 variables. Sur ces volumes, les méthodes reposant sur des ré-entraînements répétés ou des inférences multiples deviennent très coûteuses.

Étant donné les différences de complexité entre les combinaisons de méthodes, et l'écart de volume entre nos jeux simulés et les données réelles, il devient essentiel d'évaluer le temps de calcul associé à chaque approche.

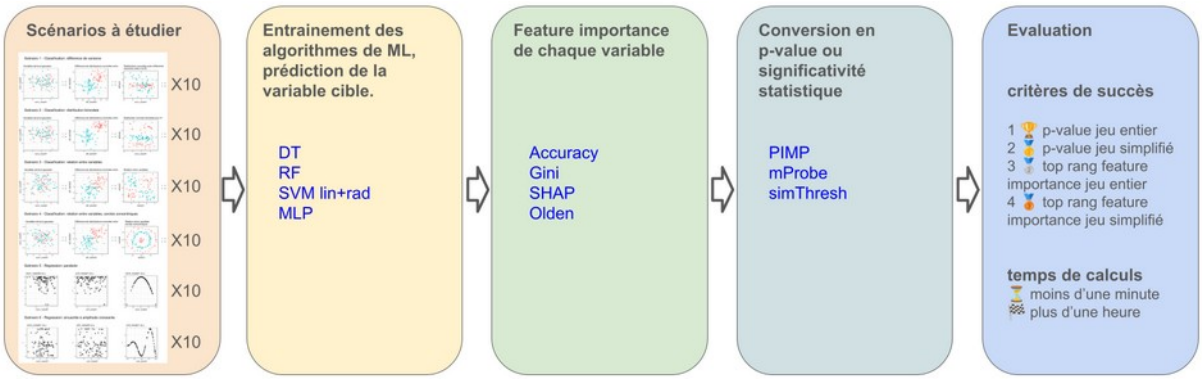

Figure 55: Workflow de comparaison de méthodes XAI de découverte. (Annexe CONTENT4PHD im/figs/gdrive_XAI scenarios workflow.png)

Temps de calcul

On commence par regarder si les temps de calcul sont comparables à combinaisons équivalentes.

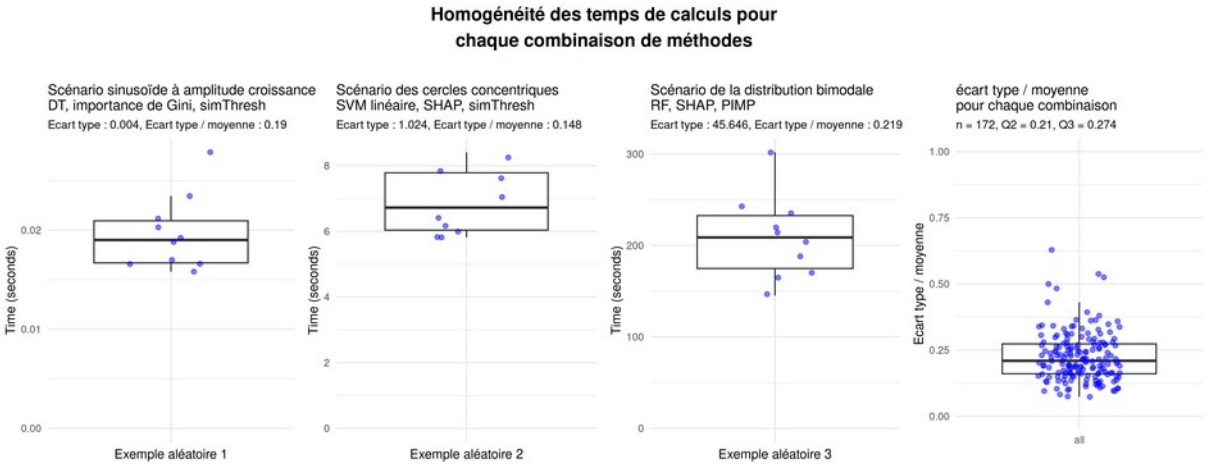

Figure 56: (gauche) trois exemples de combinaison de méthodes et leurs temps de calcul pour chacun des 10 réplicats pour chaque scénario, (droite) écart-type/moyenne pour chaque combinaison testée. (Annexe XAI_SIMU : 0008_timesx3.ipynb, im/phd/0007_time_4boxplots.png)

On peut voir dans la figure ci-dessus que les temps de calcul d'une combinaison de méthode pour les 10 réplicats d'un scénario gardent les mêmes ordres de grandeur. Prenons 3 situations représentant 3 ordres de grandeurs de temps de calcul. Pour les 10 réplicats du scénario 6, c'est à dire le cas de la sinusoïde à amplitude croissante, avec la combinaison DT + importance de Gini + *simThresh*, les temps de calcul vont de 0.015 à 0.03 secondes. Pour les 10 réplicats du scénario 3, c'est à dire le cas des cercles concentriques, avec la combinaison SVM linéaire + *SHAP* + *simThresh*, les temps de calcul vont de 5,8 à 8,5 secondes. Et enfin, pour les 10 réplicats du scénario 2, les distributions bimodales, avec la combinaison RF + *SHAP* + *PIMP*, les temps de calcul vont de 150 à 310 secondes. Chaque combinaison présente un temps de calcul relativement homogène entre les réplicats, ce qui permet de raisonner en termes de temps de calcul moyen par combinaison de méthodes.

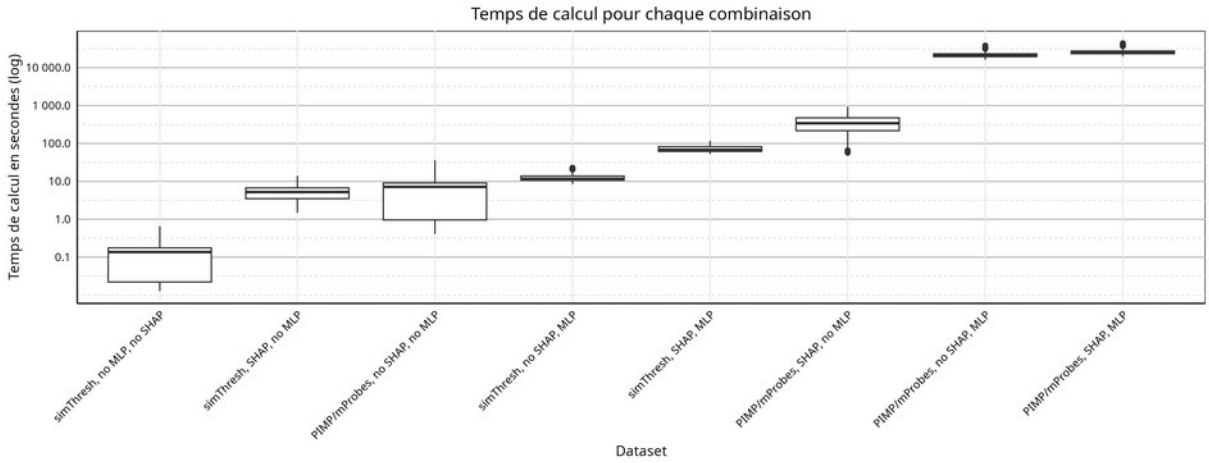

Figure 57: Boxplots des temps de calcul pour différentes catégories de combinaisons de méthodes (Annexe XAI_SIMU : 0008_timesx3.ipynb, im/phd/0008_time_8boxplots.png)

Dataset	Temps_moyen	Temps_médian	Q1	Q3	Min	Max
<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
simThresh, no MLP, no SHAP	0.15	0.14	0.02	0.18	0.01	0.65
simThresh, SHAP, no MLP	5.31	5.23	3.50	6.71	1.48	14.01
simThresh, SHAP, MLP	7.53	7.16	0.96	9.05	0.41	36.35
simThresh, no SHAP, MLP	12.70	11.63	10.75	13.84	8.49	22.97
PIMP/mProbes, no SHAP, no MLP	73.74	68.19	61.67	82.56	52.75	118.69
PIMP/mProbes, no SHAP, MLP	358.31	341.39	217.35	474.01	56.67	938.97
PIMP/mProbes, SHAP, no MLP	21822.30	21129.00	19444.09	23105.48	16192.67	38196.63
PIMP/mProbes, SHAP, MLP	25953.83	24797.21	23323.31	27423.86	19979.71	44081.53

Figure 58: Temps moyens, médian, Q1, Q3, min et max des temps de calcul pour différentes catégories de combinaisons de méthodes (Annexe XAI_SIMU : 0008_timesx3.ipynb)

Nous avons regroupé les expériences de manière à ce que chaque groupe présente un temps de calcul comparable. La mention « *no MLP* » signifie que les combinaisons du groupe ont été entraînées avec DT, RF ou SVM. La mention « no *SHAP* » signifie que les combinaisons du groupe ont déterminé leurs *feature importance*s avec *decrease accuracy*, le score de Gini ou le score d'Olden. La mention « *PIMP*/mProbes » signifie que la combinaison a utilisé *PIMP* ou mProbes pour la conversion des *feature importance*s en *p-values*. Chaque groupe créé comporte les 10 réplicats des 6 scénarios, car avec des dimensions et des temps de calcul comparables.

Le groupe dont le calcul est le plus rapide est « *simThresh* + *no MLP* + no *SHAP* » avec un temps moyen de 0.15 secondes. Le groupe nécessitant le plus de temps de calcul est « *PIMP*/mProbes + *SHAP* + MLP » avec 25 953 secondes, soit plus de 7 heures, donc un ratio de 25953/0.15 = 173 020 par rapport au groupe le plus rapide.

On remarque que les trois groupes nécessitant le moins de calculs sont « *no MLP* ». En effet, l'entraînement d'un modèle MLP avec les paramètres choisis, epoch=5000, dure environ 2min30, tandis que l'entraînement des DT, RF et SVM dure chacun moins d'une seconde.

Inversement, les 3 groupes nécessitant le plus de temps sont « *PIMP*/mProbes » + « *SHAP* ». *PIMP* et mProbes sont chronophages car ils doivent ré-entraîner le modèle de nombreuses fois pour estimer une distribution nulle des *feature importance*s. *SHAP* est également coûteux car il doit tester de nombreux sous-groupes de variables pour calculer la contribution marginale de chaque *feature*.

Pour résumer, on observe les règles suivantes :

DT, RF, SVM < MLP

simTresh < *PIMP*, mProbes

decrease accuracy, score de Gini, score de Olden < *SHAP*

Efficacité des différentes combinaisons

On trouve ci-dessous les tables répertoriant les résultats du *workflow* d'évaluation sur les 10 réplicats des 6 scénarios, sur les différentes combinaisons de méthodes déterminées précédemment. On a utilisé un code couleur pour déterminer le niveau d'efficacité de la méthode, chaque point correspond au résultat d'une combinaison sur un des réplicats d'un des scénarios :

- 🟢 indique que les *markers* trouvés par seuil sur la *p-value* correspondent exactement aux *markers* du scénario à partir du jeu de données complet. Par exemple *diff_distrib01*, *diff_distrib02*, *relVar01* et *relVar02* pour le scénario 4.
- 🟡 indique que les *markers* trouvés par seuil sur la *p-value* correspondent exactement aux *markers* du scénario à partir du jeu de données réduit. Par exemple *relVar01* et *relVar02* pour le scénario 4.
- 🟠 indique que les *markers* trouvés par rang sur *p-value* correspondent exactement aux *markers* du scénario à partir du jeu de données complet. Par exemple *diff_distrib01*, *diff_distrib02*, *relVar01* et *relVar02* sont les 4 premiers *markers* les plus significatifs, dans l'ordre ou dans le désordre, pour le scénario 4.
- 🔴 indique que les *markers* trouvés par rang sur *p-value* correspondent exactement aux *markers* du scénario à partir du jeu de données réduit. Par exemple *relVar01* et *relVar02* sont les 2 premiers *markers* les plus significatifs, dans l'ordre ou dans le désordre, pour le scénario 4.

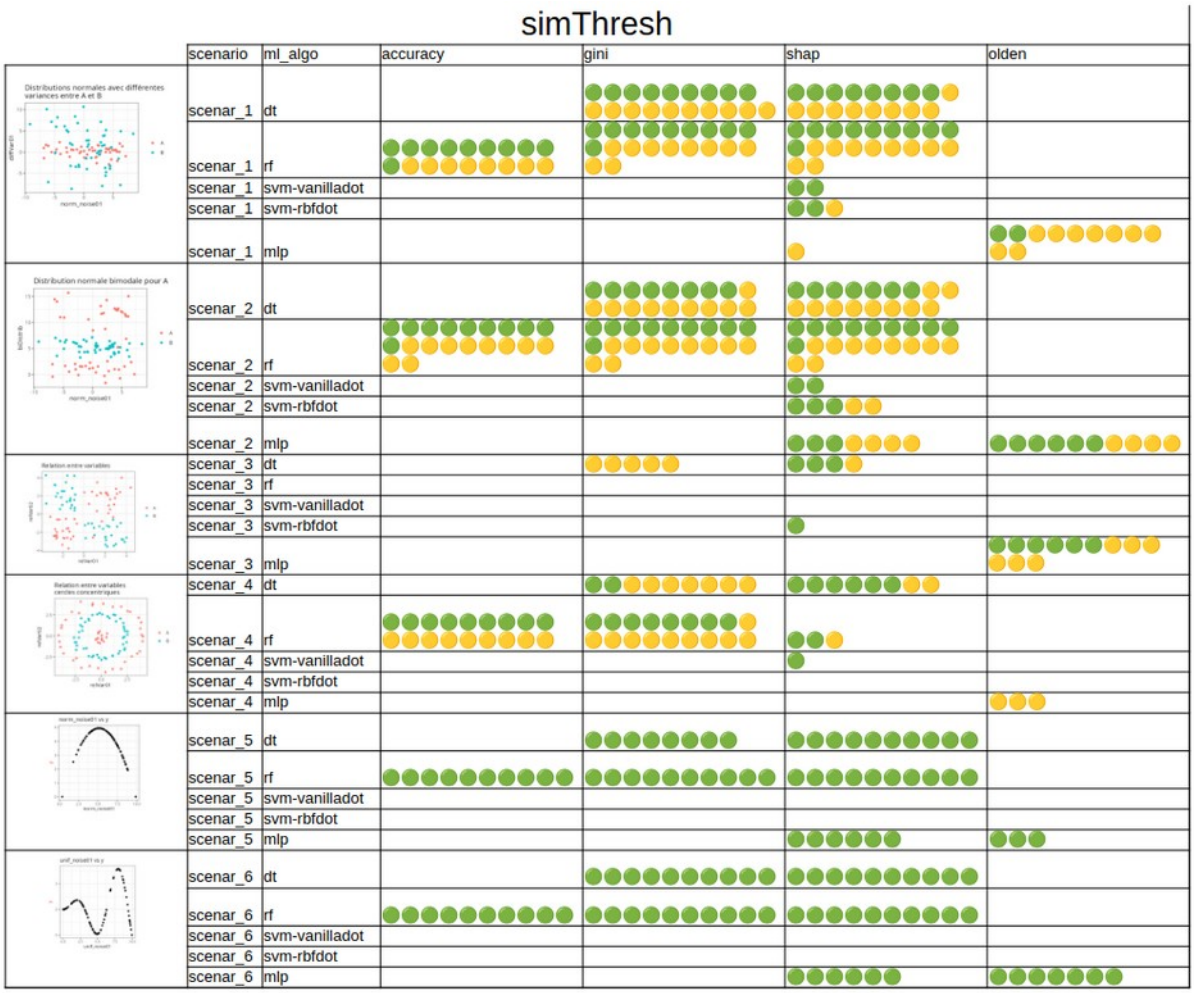

Figure 59: Tableau de résultats pour simThresh (Annexe XAI_SIMU 0009_R_results_tableX10.ipynb, data/0009_simThresh_table.csv, im/phd/0009_simThresh_table_exc_cropped.png)

On trouve ci-dessus les résultats du *workflow* utilisant notre méthode *simThresh*, qui détermine si les *feature importance*s dépassent un seuil de significativité déterminé. A noter l'absence de 🟠 et de 🔴 qui sont liés à un classement des *p-values*. Contrairement à *PIMP* et mProbes, simTresh n'associe pas directement une *p-value* à chaque *feature importance*, mais indique uniquement si le seuil est dépassé ou non, les évaluations à partir du rang des *p-values* n'est donc pas pertinent.



Figure 60: Tableau de résultats pour PIMP (Annexe XAI_SIMU 0009_R_results_tableX10.ipynb, data/0009_pimp_table.csv, im/phd/0009_pimp_table_exc_cropped.png)

On notera l'absence des cas 🟡 et 🔴 dans les scénarios de régression, scénarios 5 et 6, que ce soit avec *simThresh*, *PIMP* ou mProbes, ces jeux de données étant déjà réduits à une unique variable d'intérêt.

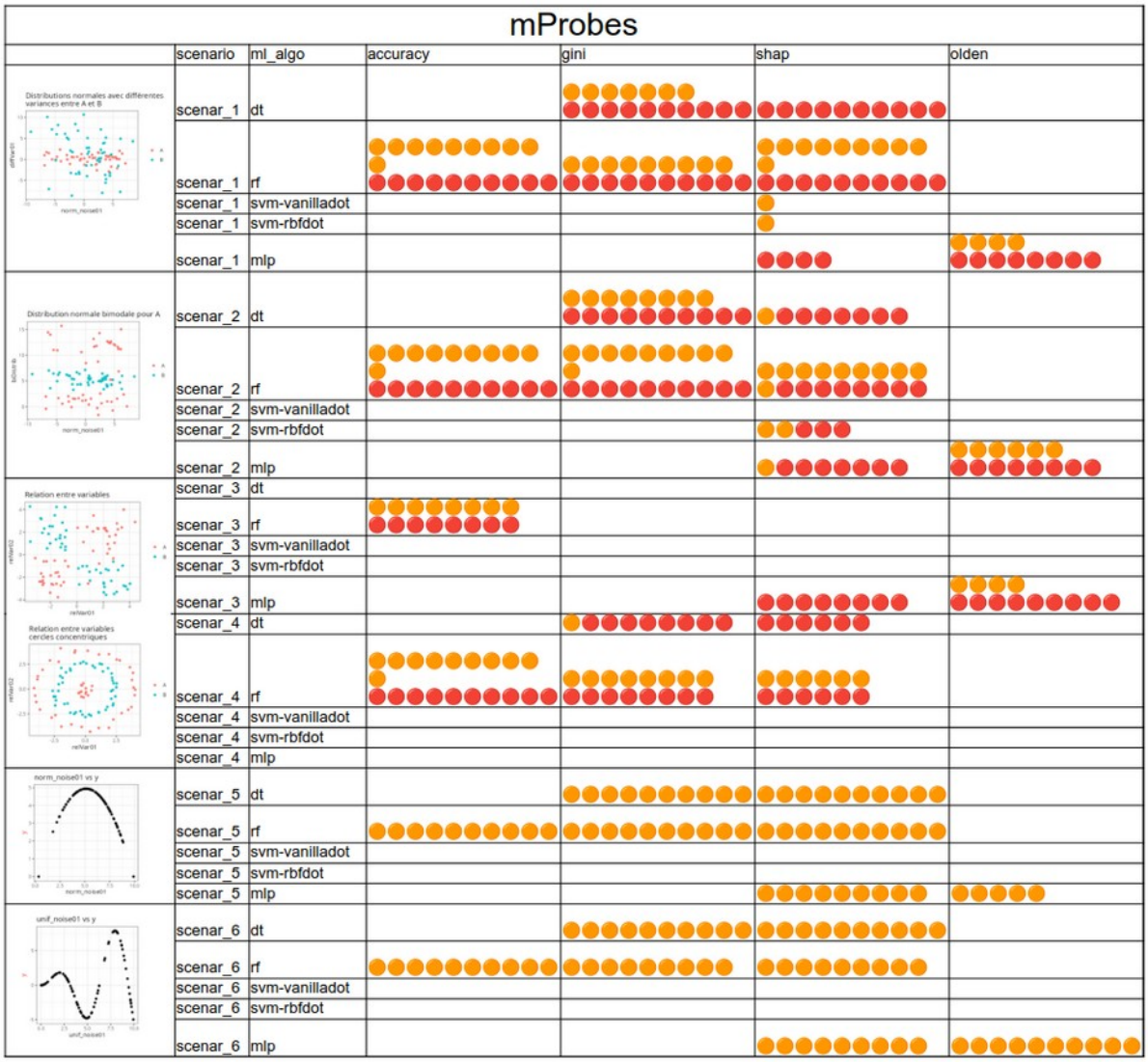

Figure 61: Tableau de résultats pour mProbes (Annexe XAI_SIMU 0009_R_results_tableX10.ipynb, data/0009_probes_table.csv, im/phd/0009_probes_table_exc_cropped.png)

Il est rare que *SHAP* performe mieux que les autres méthodes de *feature importance*s testées. Pour les DT, on peut comparer les *feature importance*s déterminées par *SHAP* aux *feature importance*s déterminées par le score de Gini. Pour les RF, *SHAP* est comparable au *decrease d'accuracy* et au score de Gini. Pour les SVMs, *SHAP* est la seule méthode dans notre test à déterminer les *feature importance*s. Et enfin, pour les MLPs, *SHAP* est comparable au score de Olden. Pour DT, RF et MLP on va donc pouvoir évaluer dans quelle mesure *SHAP* est meilleur que les autres méthodes. Prenons comme exemple *simThresh* avec DT et *SHAP vs* Gini. Le scénario 1 a 9 🟢 et 10 🟡 pour Gini, soit 9+10=19 ⚫ , et pour *SHAP* : 8 🟢 et 9 🟡 , soit 8+9=17 ⚫. On est donc à 19 succès pour Gini contre 17 succès pour *SHAP*. On peut comptabiliser ainsi les succès pour une combinaison donnée, ainsi que le nombre de victoires de Gini sur *SHAP*, et inversement :

	Gini	*SHAP*
Scénario 1	19	17
Scénario 2	18	17
Scénario 3	5	4
Scénario 4	9	8
Scénario 5	8	10
Scénario 6	10	10
Nombre de victoires	4	1

Dans la combinaison *simThresh*/DT, Gini *vs SHAP* on a un ratio de victoire de 4/1. On peut alors déterminer ce ratio de victoire pour chaque condition.

- *simThresh*
  - DT : *SHAP vs* Gini 1 *vs* 4 victoire pour Gini
  - RF : *SHAP vs accuracy* 0 *vs* 1 victoire pour *accuracy*
  - RF : *SHAP vs* Gini 0 *vs* 1 victoire pour Gini
  - MLP : *SHAP vs* Olden 1 *vs* 5 victoire pour Olden
- *PIMP*
  - DT : *SHAP vs* Gini 0 *vs* 3 victoire pour Gini
  - RF : *SHAP vs accuracy* 0 *vs* 4 victoire pour *accuracy*
  - RF : *SHAP vs* Gini 1 *vs* 2 victoire pour Gini
  - MLP : *SHAP vs* Olden 1 *vs* 4 victoire pour Olden
- mProbes
  - DT : *SHAP vs* Gini 0 *vs* 5 victoire pour Gini
  - RF : *SHAP vs accuracy* 0 *vs* 6 victoire pour *accuracy*
  - RF : *SHAP vs* Gini 0 *vs* 5 victoire pour Gini
  - MLP : *SHAP vs* Olden 1 *vs* 3 victoire pour Olden

Sur les 48 cas de non égalité, *SHAP* fait mieux que l'autre méthode de *feature importance* seulement 5 fois, pour un coup computationnel beaucoup plus élevé. Dans une étude omique à dimension élevée nous privilégierons ces méthodes alternatives.

***simThresh* fait souvent mieux que *PIMP* et/ou mProbes avec les paramètres utilisés.** Pour évaluer cette assertion nous considérons uniquement les 🟢 et 🟡, communs aux trois méthodes. La combinaison *simThresh*/DT/gini/scénario 1 comporte 9 🟢 + 10 🟡 = 19, la combinaison *PIMP*/DT/gini/scénario 1 comporte 7 🟢 + 10 🟡 = 17, et la combinaison mProbes/DT/gini/scénario 1 comporte 0 🟢 + 0 🟡 = 0. Ici c'est *simThresh* qui permet de détecter les variables d'intérêt dans le plus grand nombre de cas. Sur ce principe nous pouvons construire le tableau répertoriant la méthode détectant le plus grand nombre de variables d'intérêt pour chaque configuration.

	simThresh	PIMP	mProbes	winner
scenar_1_dt_gini	19	17	0	simThresh
scenar_2_dt_gini	18	20	0	PIMP
scenar_3_dt_gini	5	0	0	simThresh
scenar_4_dt_gini	9	9	0	simThresh
scenar_1_dt_shap	17	0	0	simThresh
scenar_2_dt_shap	17	1	0	simThresh
scenar_3_dt_shap	4	0	0	simThresh
scenar_4_dt_shap	8	0	0	simThresh
scenar_1_rf_accuracy	18	20	0	PIMP
scenar_2_rf_accuracy	20	20	0	simThresh
scenar_3_rf_accuracy	0	11	0	PIMP
scenar_4_rf_accuracy	18	20	0	PIMP
scenar_1_rf_gini	20	17	0	simThresh
scenar_2_rf_gini	20	20	0	simThresh
scenar_3_rf_gini	0	0	0	simThresh
scenar_4_rf_gini	18	9	0	simThresh
scenar_1_rf_shap	20	0	0	simThresh
scenar_2_rf_shap	20	0	0	simThresh
scenar_3_rf_shap	0	0	0	simThresh
scenar_4_rf_shap	3	0	0	simThresh
scenar_1_svm-vanilladot_shap	2	0	0	simThresh
scenar_2_svm-vanilladot_shap	2	0	0	simThresh
scenar_3_svm-vanilladot_shap	0	0	0	simThresh
scenar_4_svm-vanilladot_shap	1	0	0	simThresh
...				
scenar_5_mlp_shap	6	7	0	PIMP
scenar_6_mlp_shap	6	2	0	simThresh
scenar_5_mlp_olden	3	1	0	simThresh
scenar_6_mlp_olden	7	5	0	simThresh

Figure 62: Tableau de la meilleure méthode entre simThresh, PIMP et mProbes (Annexe XAI_SIMU 0010_R_analysis_results_tableX10.ipynb)

```
table(result_df$winner)
✓ 0.0s

PIMP simThresh
   7        43
```

Figure 63: Nombre de victoires par méthode (Annexe XAI_SIMU 0010_R_analysis_results_tableX10.ipynb)

Avec les paramètres choisis pour mProbes, nombre de simulation = 100, nombre de probes = 10 et transformation des données en rang + *jitter*, cette méthode n'aboutit pas à des *p-values* suffisamment basses pour être significatives. Pour ces paramètres, la méthode mProbes n'est jamais la meilleure, quelle que soit la condition. En ce qui concerne les deux méthodes restantes, sur les 50 conditions étudiées, *PIMP* est la meilleure méthode 7 fois et *simThresh* 43 fois, soit un ratio d'environ 1 pour 6.

Des scénarios mieux détectés que d'autres. Le scénario 2, correspondant à la distribution bimodale, est par exemple détecté avec une moyenne de 4.94 🟢 sur 10 maximum (on ne compte que *simThresh* et *PIMP*), et 5.17 🟡 sur 10 réplicats par combinaison. Le scénario 3, correspondant au XOR, est lui détecté plus rarement, avec une moyenne de 0.67 🟢 sur 10 et 1.17 🟡 sur 10. On peut ainsi établir une table reflétant la facilité à détecter chaque scénario.

	simThresh_green_avg	simThresh_yellow_avg	PIMP_green_avg	PIMP_yellow_avg	green_total_avg	yellow_total_avg
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
scenar_1	5.89	6.44	2.67	4.11	4.28	5.28
scenar_2	6.56	6.67	3.33	3.67	4.94	5.17
scenar_3	1.11	1.33	0.22	1.00	0.67	1.17
scenar_4	3.11	3.56	1.11	3.11	2.11	3.33
scenar_5	6.33	0.00	5.22	0.00	5.78	0.00
scenar_6	7.00	0.00	5.00	0.00	6.00	0.00

Figure 64: Nombre moyen de détections sur les 10 réplicats de chaque scénario pour chaque combinaison (Annexe XAI_SIMU 0010_R_analysis_results_tableX10.ipynb)

Parmi nos scénarios, les mieux détectés sont les scénarios de régression, c'est à dire les 5 et 6, avec des taux de détection respectifs de 5.78 et 6 sur 10 pour les 🟢. Les deux scénarios les plus difficiles à détecter sont les scénarios avec relations entre plusieurs variables, le XOR (scénario 3) et les cercles concentriques (scénario 4), avec respectivement des scores de 0.67 et 2.11 sur 10 pour les 🟢. Les scénarios de différence de variance (scénario 1) et de distribution bimodale (scénario 2) ont un score moyen de 4.28 et 4.94 sur 10.

Chaque scénario est plus ou moins facile à détecter en moyenne, mais on trouve une disparité entre les méthodes de détection. Le scénario 3 correspondant au XOR possède par exemple 6 🟢 sur 10 et 6 🟡 sur 10 pour la combinaison MLP/Olden/*simThresh*. On a ainsi comptabilisé le nombre de succès pour chaque combinaison dans la table ci-dessous :

model_method	conv_method	🟢	🟡	🟢+🟡
<chr>	<chr>	<dbl>	<dbl>	<dbl>
scenar_3_mlp_olden	simThresh	6	6	12
scenar_3_rf_accuracy	PIMP	2	9	11
scenar_3_dt_gini	simThresh	0	5	5
scenar_3_dt_shap	simThresh	3	1	4
scenar_3_svm-rbfdot_shap	simThresh	1	0	1
scenar_3_rf_accuracy	simThresh	0	0	0
scenar_3_rf_gini	simThresh	0	0	0
scenar_3_rf_shap	simThresh	0	0	0
scenar_3_svm-vanilladot_shap	simThresh	0	0	0
scenar_3_mlp_shap	simThresh	0	0	0
scenar_3_dt_gini	PIMP	0	0	0
scenar_3_dt_shap	PIMP	0	0	0
scenar_3_rf_gini	PIMP	0	0	0
scenar_3_rf_shap	PIMP	0	0	0
scenar_3_svm-vanilladot_shap	PIMP	0	0	0
scenar_3_svm-rbfdot_shap	PIMP	0	0	0
scenar_3_mlp_shap	PIMP	0	0	0
scenar_3_mlp_olden	PIMP	0	0	0

Figure 65: Nombre de succès par combinaison pour les 10 réplicats du scénario 3 (Annexe XAI_SIMU 0010_R_analysis_results_tableX10.ipynb)

On voit sur la table ci-dessus que sur 18 combinaisons de méthodes (on ne considère que *PIMP* et *simThresh* car mProbes n'a aucun 🟢 ni 🟡) seules 5 combinaisons arrivent à détecter, parfois, les bonnes variables. Les deux meilleures combinaisons sont MLP/Olden/*simThresh* et RF/*accuracy*/*PIMP*. Cependant, comme nous l'avons vu précédemment, MLP est coûteux en temps de calcul, *PIMP* également, ce qui peut poser problème pour une application à grande échelle sur données omiques réelles. Nous nous intéressons donc à la troisième combinaison la plus performante, DT/gini/*simThresh*, qui est bien plus rapide. Dans la prochaine section de l'axe 2 nous appliquons la méthode à des cas réels de classification, sur ensembles réduits (après avoir enlevé les variables détectables par les tests statistiques classiques), on peut donc tirer des enseignements des scénarios 1 à 4. On voit dans la table ci-dessous que la méthode détecte parfaitement, à 10/10, les scénarios 1 et 2, correspondant à la différence de variance et aux distributions bimodales. La méthode détecte correctement dans la moitié des cas des scénarios de relation entre plusieurs variables, avec 7/10 pour le scénario 4 des cercles concentriques et 5/10 pour le scénario 3 du XOR.

model_method	conv_method	🟢	🟡	🟢+🟡
<chr>	<chr>	<dbl>	<dbl>	<dbl>
scenar_1_dt_gini	simThresh	9	10	19
scenar_2_dt_gini	simThresh	8	10	18
scenar_6_dt_gini	simThresh	10	0	10
scenar_4_dt_gini	simThresh	2	7	9
scenar_5_dt_gini	simThresh	8	0	8
scenar_3_dt_gini	simThresh	0	5	5

Figure 66: Nombre de détections des variables d'intérêt pour les combinaisons de méthodes DT/Gini/simThresh (Annexe XAI_SIMU 0010_R_analysis_results_tableX10.ipynb)

Application sur données scRNA-seq de *CZ CELLxGENE*

CZ CELLxGENE

CELLxGENE est une plateforme développée par la Chan Zuckerberg Initiative, dédiée à l'exploration de données de transcriptomique en *single-cell* [132]. Elle regroupe actuellement 1787 jeux de données, représentant plus de 110 millions de cellules humaines et murines, et 994 types cellulaires (13/05/2025).

Les cellules sont annotées avec des métadonnées telles que le type cellulaire, le tissu d'origine, ou encore des informations sur le donneur. Les données proviennent de projets académiques et de consortiums publics, et sont accessibles notamment en format *Seurat* via une interface web ou par des outils programmatiques comme l'API *CELLxGENE Census* disponible en *Python*. Le fait que les données soient déjà pré-traitées, normalisées et annotées par type cellulaire permet de sélectionner facilement des sous-ensembles pertinents pour des analyses comparatives.

Étude sur périmètre réduit

Pour un premier test, nous avons sélectionné manuellement 116 jeux de données parmi ceux contenant le moins de cellules, mais incluant toujours au moins quelques milliers chacune. Nous les avons téléchargé au format *Seurat* et normalisés avec la fonction *NormalizeData* du package *Seurat* quand cela n'avait pas été fait. Nous avons extrait la matrice d'expression log normalisée, c'est à dire le *slot* « *data* » de l'objet *Seurat*. Nous avons ensuite sélectionné les 5000 gènes les plus variants par jeu de données, et sélectionné au hasard 100 cellules réparties équitablement en deux types cellulaires différents quand cela était possible, c'est à dire dans 91 des 116 jeux de données. Enfin nous avons exporté ces matrices d'expression d'ARN de 5000 gènes x 100 cellules pour la suite de l'étude.

Disponibles dans l'annexe *XAI_CZ_CELLxGENE* :

- Rds téléchargés à la main dans : *data/genexcells_jeux de données/*
- Code de construction des jeux de données : *0009_R_hdl_build50by50.ipynb*
- Matrices exportées dans : *data/scRNAseq_hdl_genexcells_rand2CellTypes_ds_ns50/*

Combinaison de méthodes choisie

On a tiré plusieurs enseignements de l'étude sur les données simulées.

1. On gagne facilement en efficacité en enlevant les variables détectables par *t-test* car on donne plus de chance aux algorithmes de ML d'utiliser les variables plus complexes.

2. Les DT, RF et MLP donnent de bons résultats, mais MLP demande beaucoup plus de temps de calcul.
3. L'algorithme *SHAP* demande beaucoup de temps de calcul en inférence, et n'améliore pas le pouvoir de détection des variables d'intérêt comparé aux autres algorithmes de *feature importance*s.

Nous avons choisi pour notre étude sur cas réels d'enlever les gènes détectables par *t-test*. Nous utilisons DT comme algorithme de ML car rapide à entraîner, donc pertinent pour une étude à large échelle. Les arbres de décisions offrent également l'avantage d'être représentable et interprétables facilement. D'après les enseignements de l'étude sur données simulées nous espérons détecter efficacement les différences de variances et les distributions bimodales, et détecter une partie des cas de relations entre variables.

Résultat sur périmètre réduit

Pour chacun des 91 jeux de données du périmètre réduit nous avons calculé un *t-test* et un test de Wilcoxon entre les deux types cellulaires et avons gardé les gènes dont les deux *p-values* étaient au dessus de 0.2. Avec les gènes restant nous avons entraîné un arbre de décision avec la fonction *rpart* du package *R rpart* avec pour paramètre *maxdepth=2* et nous avons noté l'*accuracy* de chaque arbre.

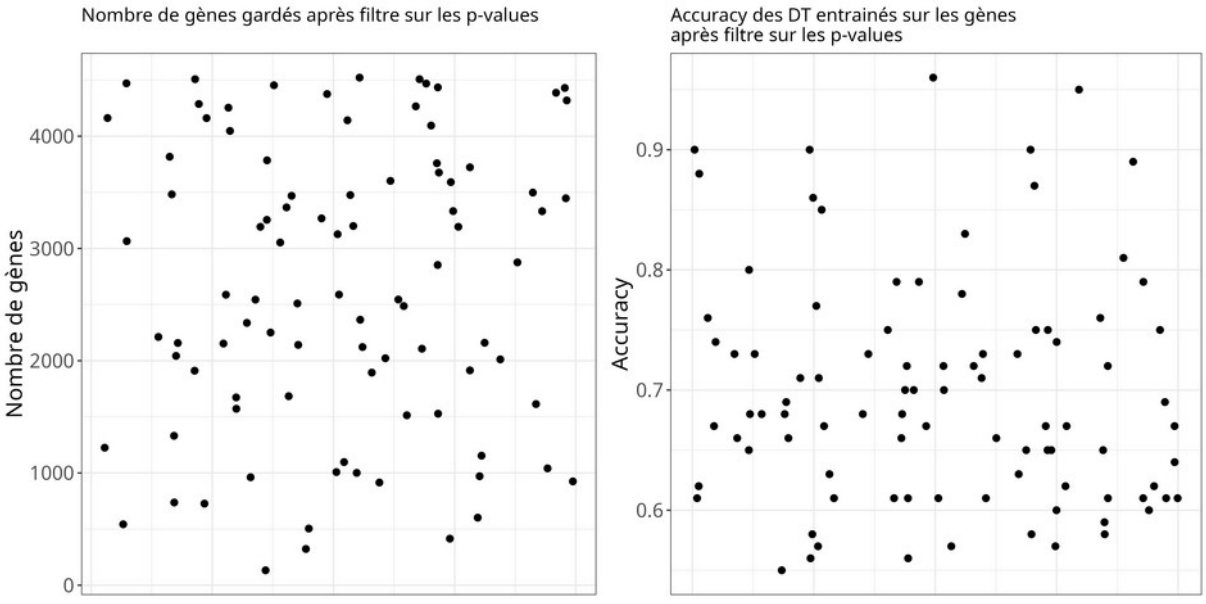

Figure 67: (gauche) Nombre de gènes restant après le filtre sur la p-value à 0.2 pour le t-test et le test de Wilcoxon sur les 91 jeux de données (droite) Accuracy des DT de profondeur max=2 entraînés sur les gènes ayant passé le filtre. (Annexe XAI_CZ_CELLxGENE 0020_R_plot_kept_genes_accuracies.ipynb)

On peut voir sur la figure ci-dessous deux exemples d'arbres de décision résultant de cette étude sur périmètre réduit.

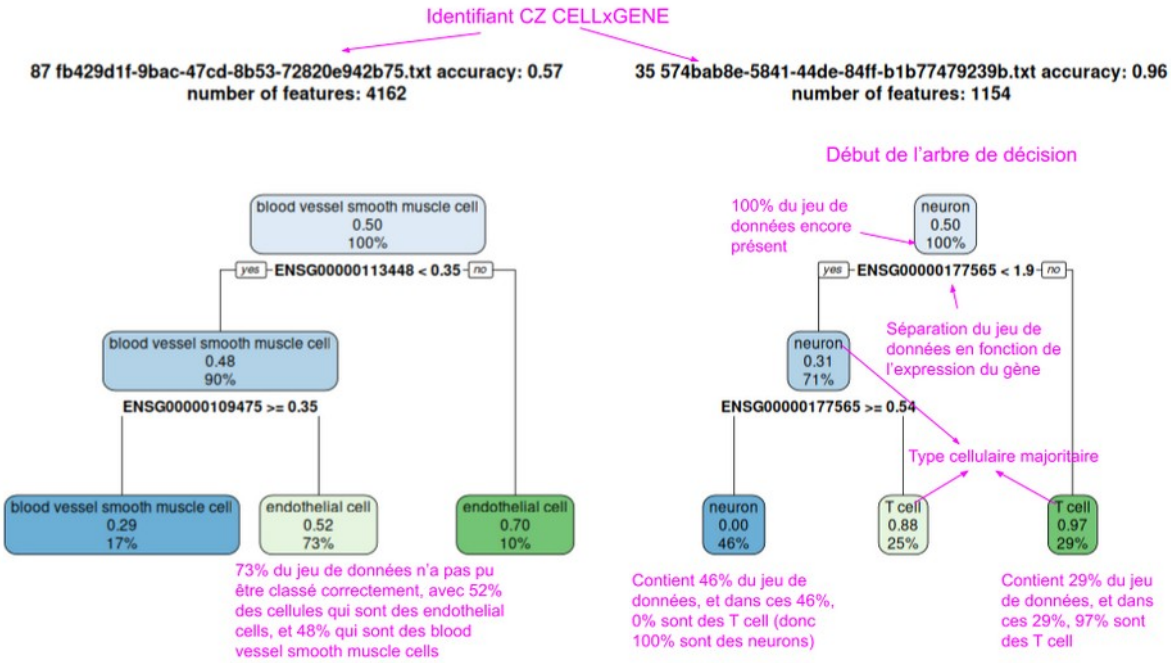

Figure 68: (gauche) Exemple d'arbre de décision d'accuracy faible de 0.57 (droite) Exemple d'arbre de décision avec une forte accuracy de 0.96

Afin d'explorer la nature des gènes retenus par les arbres de décision, entraînés sans les gènes détectables par *t-test* ou Wilcoxon, nous avons utilisé des *scatter plots*. Ces visualisations nous permettent de situer ces gènes dans les scénarios théoriques de biomarqueurs complexes que nous avons définis.

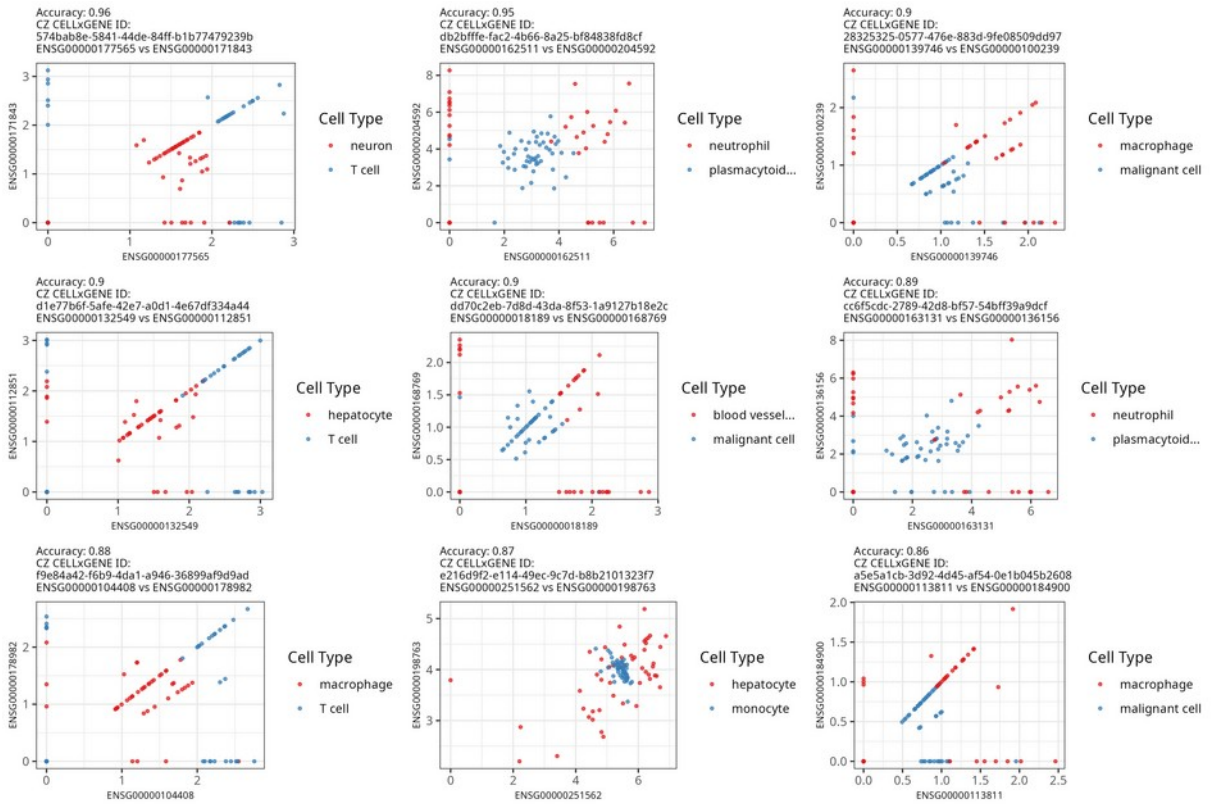

Figure 69: Scatter plots des deux gènes avec la plus grande importance de Gini des 9 arbres de décision avec la plus forte accuracy parmi les 91 arbres (Annexe XAI_CZ_CELLxGENE 0020_R_plot_kept_genes_accuracies.ipynb, im/phd/0020_top9_gene_expression_scatter.png)

On observe dans les graphiques 1, 3, 4, 5, 7 et 9, un motif commun composé de points alignés sur la diagonale ainsi que sur les axes des abscisses et des ordonnées. Ce motif s'explique par les effets induits par la normalisation appliquée aux données brutes issues du séquençage. En effet, après alignement des séquences et traitement des fichiers *.fastq*, les matrices d'expression contiennent des nombres de *reads* par gène et par cellule. Une première étape consiste à ramener le total de *reads* par cellule à 10 000, suivie d'une transformation logarithmique. Les structures en diagonale observées dans les *scatter plots* correspondent typiquement à des cellules pour lesquelles les deux gènes, gène 1 et gène 2, avaient initialement comme nombre de *reads* brut (1,1), mais dans des cellules dont le nombre de *reads* était différent.

Prenons le cas du premier *plot*, imaginons une cellule « *neuron* » possédant initialement 20 000 *reads*, alors une expression de (1,1) serait ramenée à (0.5, 0.5) après normalisation à 10 000. À l'inverse, une cellule « *T cell* » possédant seulement 5 000 *reads* verrait la même expression brute (1,1) normalisée à (2,2). Ce phénomène crée des alignements diagonaux qui ne reflètent pas une variation biologique d'expression, mais un artefact de la normalisation. Les gènes identifiés comme importants dans ces cas sont probablement liés à ces effets, et non à une différence biologique entre les types cellulaires.

Afin de contrôler ces artefacts dans l'analyse sur périmètre étendu, nous avons distingué les paires de types cellulaires présentant un faible écart du nombre de *reads* moyen de celles présentant un écart fort, pour éviter que l'importance attribuée à

certains gènes ne soit dominée par des effets de taille plutôt que par une réelle différence d'expression.

Étude sur périmètre étendu

Les premiers résultats obtenus sur un échantillon de 116 jeux de données, chacun ne contenant qu'une seule paire de types cellulaires, illustrent l'importance d'une analyse à large échelle. En effet, parmi ces 116 jeux de données, seuls cinq atteignaient une exactitude supérieure à 0.9, dont quatre semblaient présenter des effets de normalisation plutôt qu'une réelle différence biologique. Ce constat nous a conduit à étendre notre analyse à une plus large collection d'échantillons, en tirant parti de l'ensemble de la base de données *CELLxGENE*.

Pour des raisons de stockage, nous n'avons pas conservé l'intégralité de la base *CELLxGENE* en local. Le protocole mis en place consiste à télécharger les objets *Seurat* un par un, à en extraire des sous-matrices pertinentes de 5 000 gènes par 50 cellules pour chaque type cellulaire, puis à supprimer l'objet.

Afin d'étudier les effets de la normalisation, nous avons limité notre analyse aux objets *Seurat* contenant les matrices de *counts* de *reads* bruts. Pour chaque échantillon, nous avons généré l'ensemble des paires possibles de types cellulaires, que nous avons classées en trois catégories selon leur écart du nombre de *reads* par cellule :

- *low_diff* : écart moyen du nombre de *reads* par cellule entre les deux types cellulaires ≤ 20 %
- *high_diff* : écart ≥ 80 %
- *medium_diff* : cas intermédiaires

Seules les paires *low_diff* et *high_diff* sont conservées pour l'analyse. Pour chacune, on extrait une sous-matrice de 5 000 gènes par 100 cellules.

Les données *high_diff* nous permettrons d'étudier les effets de la normalisation, et les données *low_diff* nous permettrons d'étudier les différences biologiques détectées en minimisant les risques liés à la normalisation.

Sur les 1787 jeux de données de *CZ GENExCELL*, 467 ont à la fois les *counts* bruts et une paire de types cellulaire d'écart de moyenne ≤ 20 %. L'ensemble *low_diff* comporte 2041 jeux de données car plusieurs paires de types cellulaire sont souvent possibles pour chacun des 467 objets Seurats.

Sur les 1787 jeux de données de *CZ GENExCELL*, 359 ont à la fois les *counts* bruts et une paire de types cellulaire d'écart de moyenne ≥ 80 %. L'ensemble *high_diff* comporte 1260 jeux de données car plusieurs paires de types cellulaire sont possibles pour chacun des 359 objets *Seurat*.

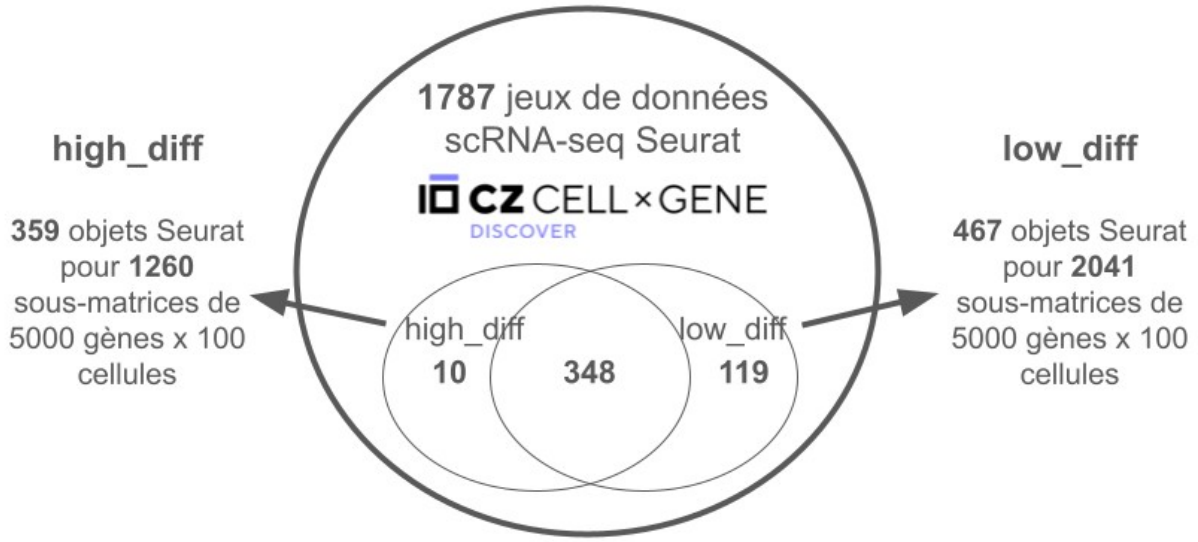

Figure 70: Création des données pour l'étude sur périmètre étendu.

Disponibles dans l'annexe *XAI_CZ_CELLxGENE* :

- Code de génération des jeux de données *low_diff* et *high_diff* : pipe_low_high/0003_*R*_mono_function4octopus.ipynb
- Répertoire des jeux de données *low_diff* : data/low_diff_res
- Répertoire des jeux de données *high_diff* : data/high_diff_res

Biais de normalisation

Pour chaque sous-matrice générées nous avons entraîné un arbre de décision selon le protocole précédemment défini, puis nous avons sélectionné les gènes présents dans ces arbres pour les représenter sous forme de *scatter plots*. Cette représentation permet de sélectionner les cellules correspondant au *pattern* de diagonal observé précédemment. Étant donné que nous avons préalablement restreint l'analyse aux jeux de données *CZ CELLxGENE* fournissant les comptes de *reads* bruts, il est possible de colorer les points en fonction de cette information. Cela nous permet de valider l'hypothèse d'un biais induit par la normalisation. En effet, les points situés sur la diagonale proche de l'origine correspondent à des cellules présentant un nombre élevé de *counts* bruts, tandis que ceux éloignés de l'origine correspondent à des cellules ayant un faible nombre de *counts*. Pour éviter d'interpréter ces artefacts comme des signaux biologiques, nous avons choisi de restreindre l'analyse au sous-ensemble de données présentant des différences de profondeur plus faibles, c'est à dire à l'ensemble *low_diff*.

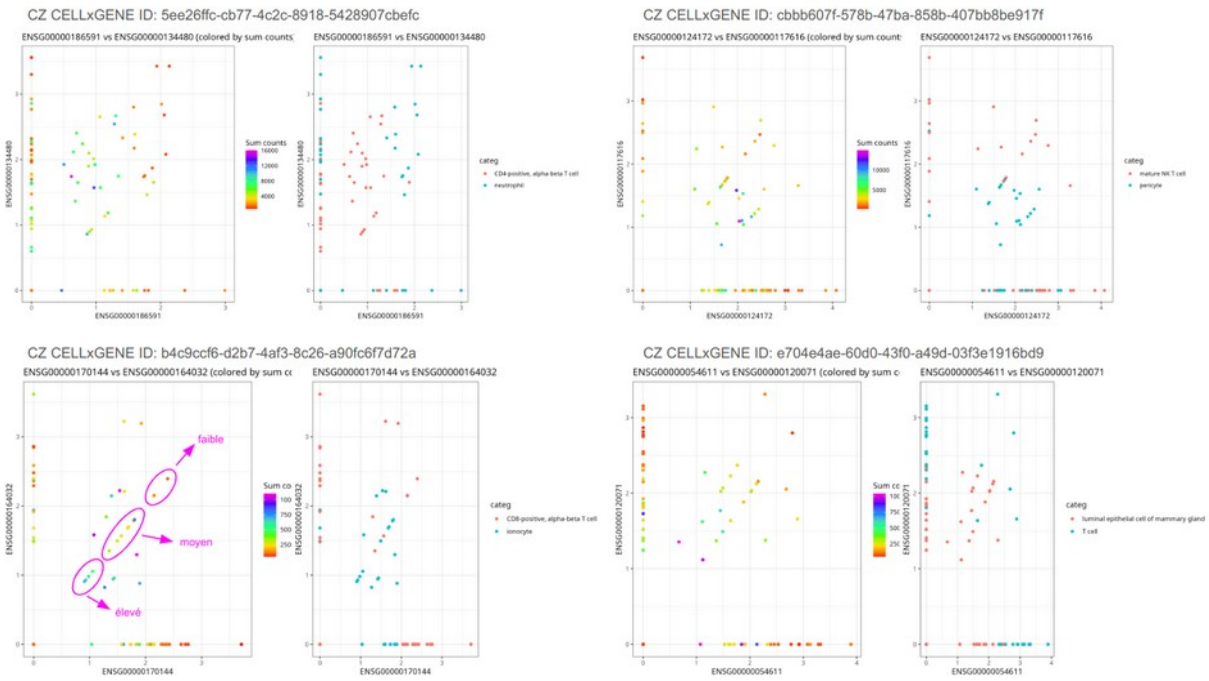

Figure 71: Illustration du biais de normalisation sur le sous-ensemble high_diff

Résultats sur *low_diff*

Comme pour *high_diff*, nous avons entraîné un arbre de décision pour chacune des 2041 sous-matrices de *low_diff*, en utilisant les mêmes paramètres que précédemment. La figure ci-dessous montre la distribution des *accuracies* obtenues pour ces 2041 arbres. Nous allons étudier les 100 arbres ayant les meilleures *accuracies*, dont la plus basse est de 0,79.

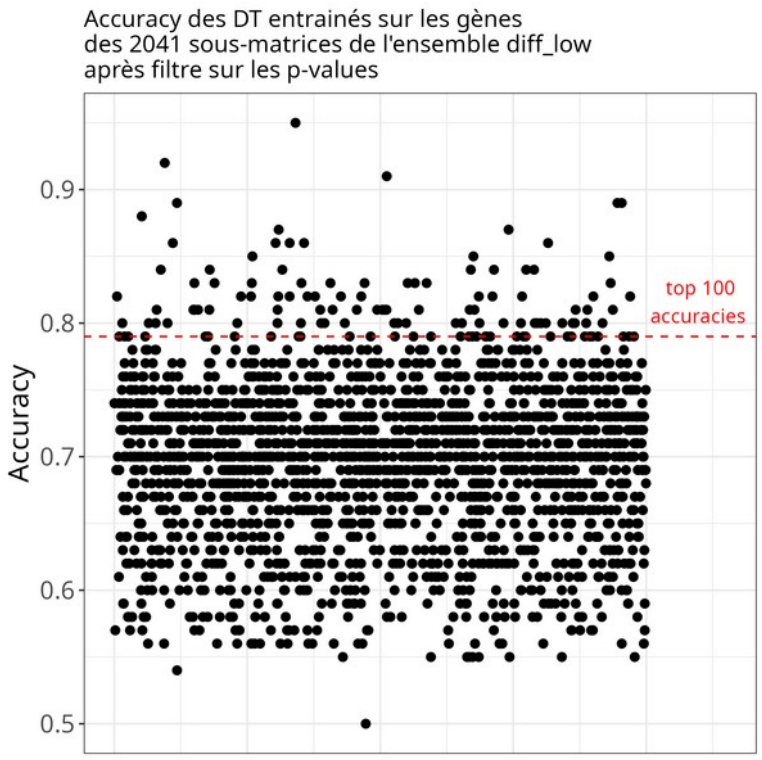

Figure 72: Accuracy des DT de profondeur max=2 entraînés sur les gènes ayant passé les filtres sur l'ensemble low_diff. (0055_XAIframework/0023_R_plot_lowDiff_accuracies.ipynb)

Pour chaque arbre, nous avons inclus un *scatter plot* des deux gènes ayant les scores de Gini les plus élevés. Ces gènes correspondent en général à ceux utilisés dans l'arbre. Il arrive que, même avec une profondeur de 2, un arbre ne réalise ses séparations qu'à partir d'un seul gène, ou parfois trois.

Les *scatter plots* présentent les deux types cellulaires en couleurs distinctes, ce qui permet de visualiser les motifs expliquant pourquoi certains gènes, bien qu'écartés par les *t-test*s ou tests de Wilcoxon, ont tout de même été utilisés par l'arbre pour classifier les cellules.

D'autres *scatter plots* montrent, via la couleur des points, le nombre brut de *reads* par cellule. Cela permet de repérer d'éventuels biais de normalisation, bien que *low_diff* ait été construit pour les minimiser.

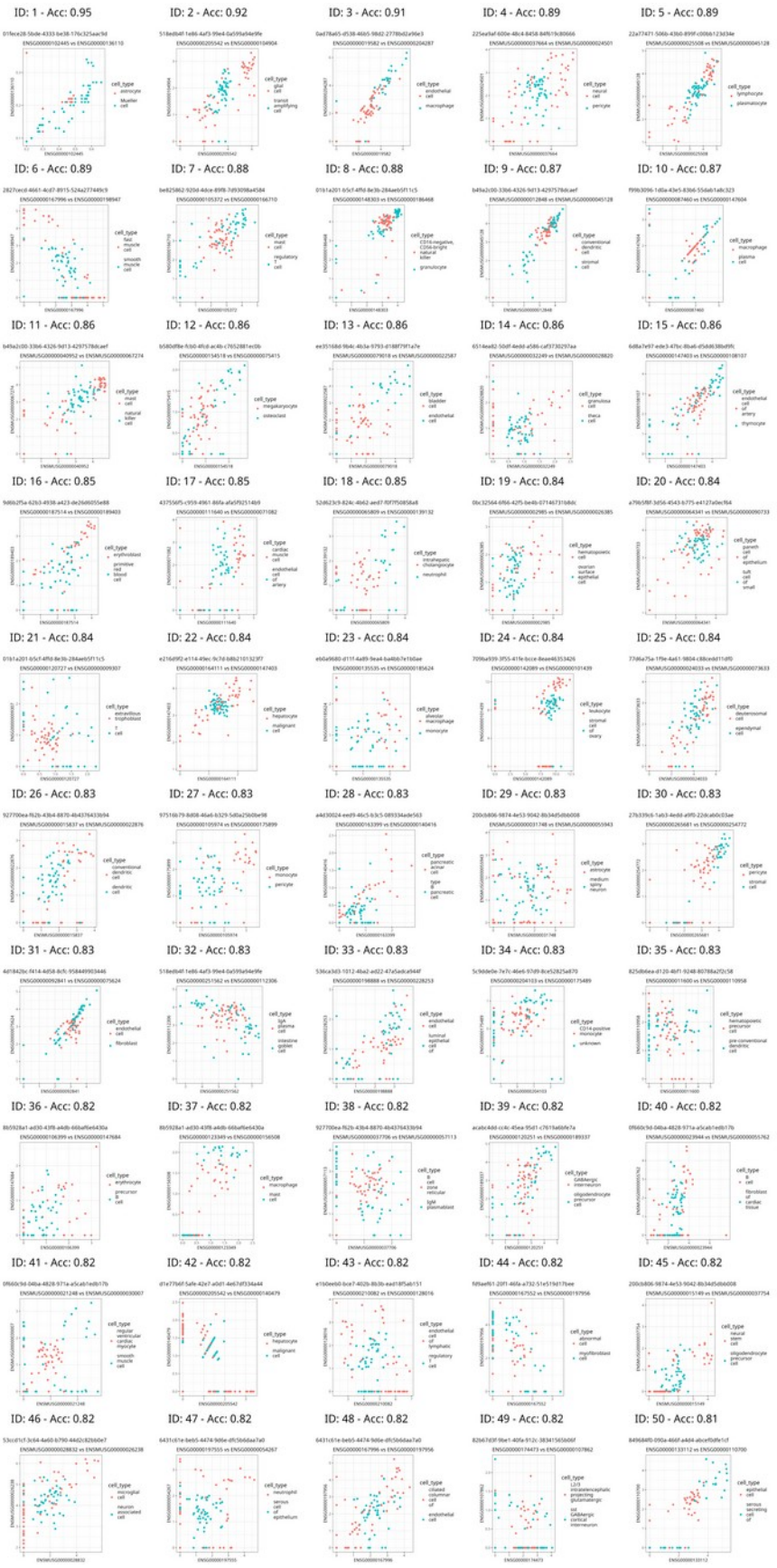

Figure 73: Scatter plots 1 à 50 des DT avec la plus grande accuracy, sur les deux gènes avec le plus grand score de Gini, couleur par type cellulaire. (0055_XAIframework/0027_R_low_diff_stats_100grid_bigDots.ipynb, 0027_ids_top100_grid1_10x5_bigDots.png)

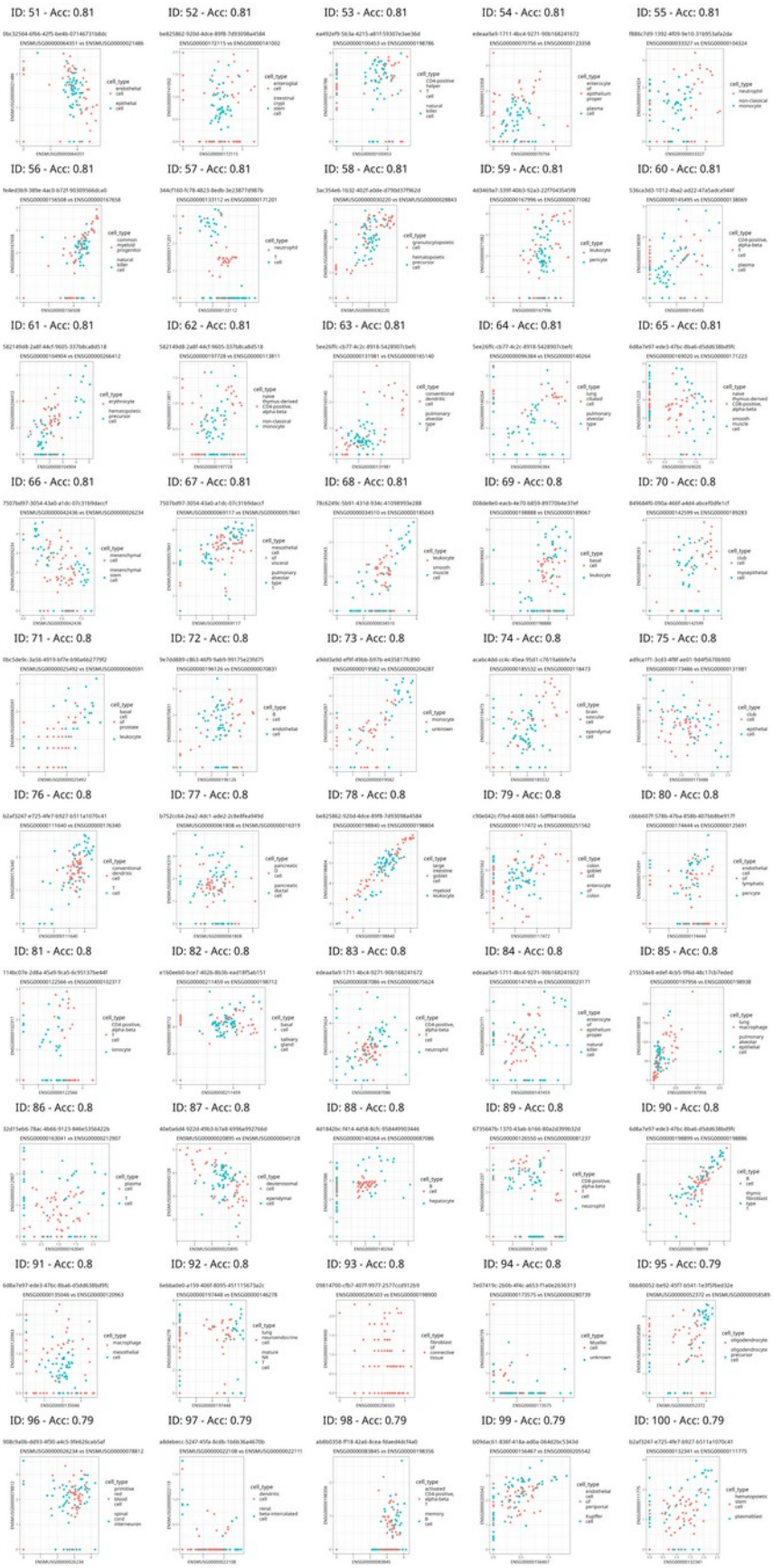

Figure 74: Scatter plots 51 à 100 des DT avec plus la grande accuracy, sur les deux gènes avec le plus grand score de Gini, couleur par type cellulaire. (0055_XAIframework/0027_R_low_diff_stats_100grid_bigDots.ipynb, 0027_ids_top100_grid1_10x5_bigDots.png)

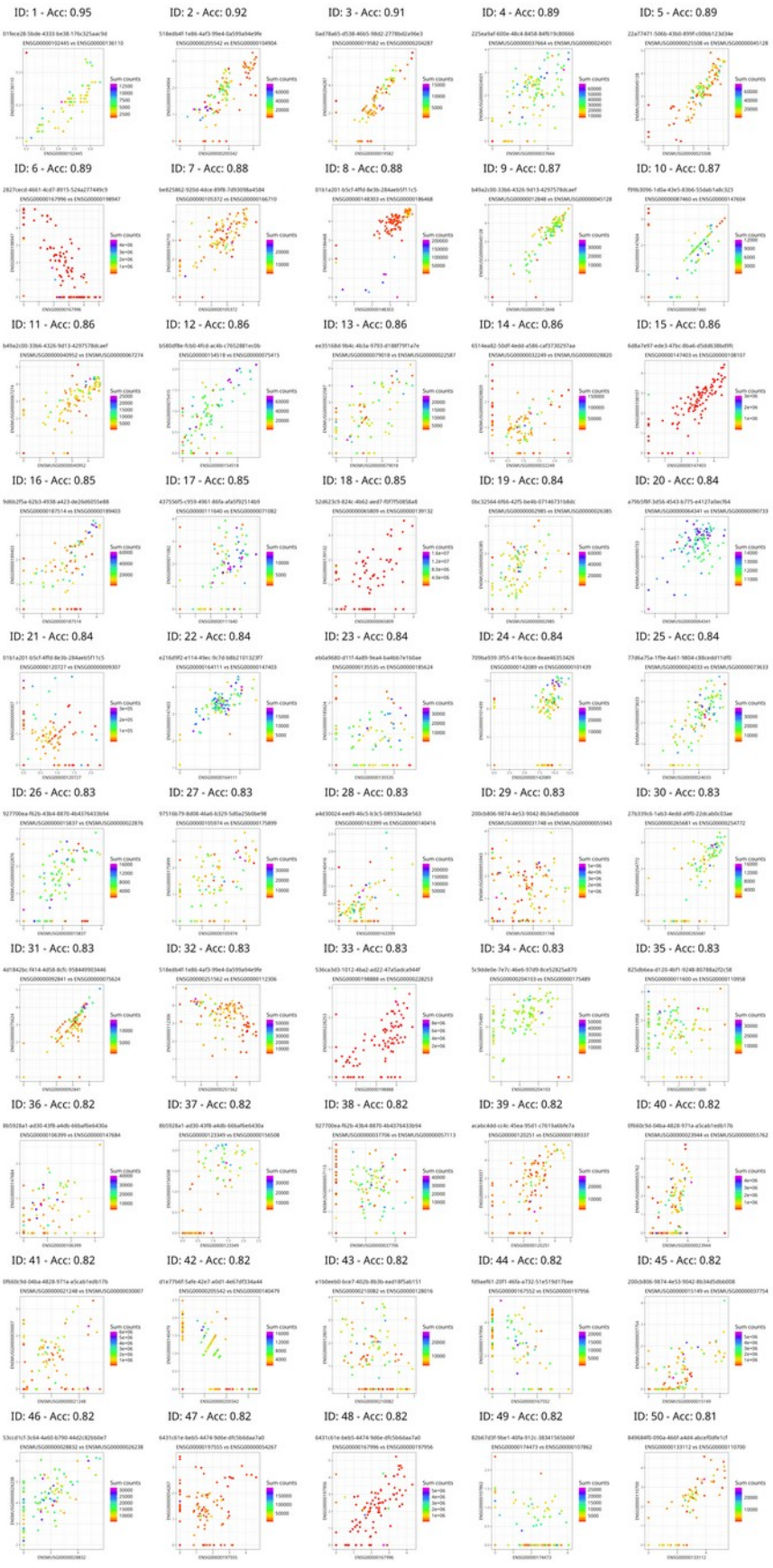

Figure 75: Scatter plots 1 à 50 des DT avec la plus grande accuracy, sur les deux gènes avec le plus grand score de Gini, couleur par nombre de counts. (0055_XAIframework/0027_R_low_diff_stats_100grid_bigDots.ipynb, 0024_ids_top100_grid1_10x5_cntCol_bigDots.png)

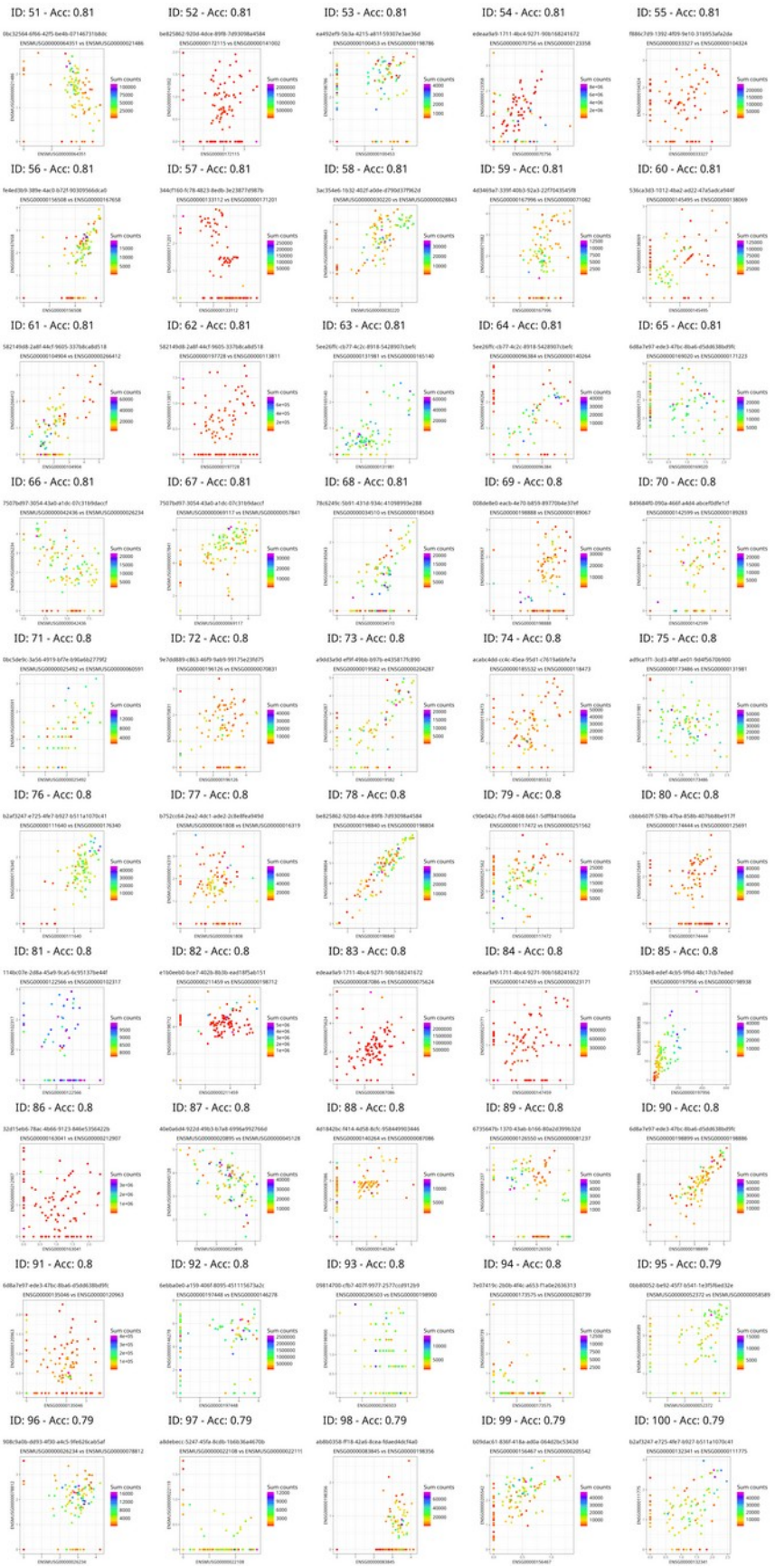

Figure 76: Scatter plots 51 à 100 des DT avec la plus grande accuracy, sur les deux gènes avec le plus grand score de Gini, couleur par nombre de counts. (0055_XAIframework/0027_R_low_diff_stats_100grid_bigDots.ipynb, 0024_ids_top100_grid2_10x5_cntCol_bigDots.png)

Parmi ces 100 premiers résultats, nous observons plusieurs motifs qui suggèrent que certains gènes sélectionnés par notre méthode pourraient être liés à des biais de normalisation, plutôt qu'à un réel processus biologique (Annexe *XAI_CZ_CELLxGENE*, *data/low_diff_top100_description.xlsx*).

Pattern **de biais visible**

Sur certains *scatter plots*, on distingue des groupes de cellules ayant un nombre de *counts* plus élevé ou plus faible, ou encore un gradient progressif de *counts* à travers la figure. Ces motifs suggèrent la présence de biais liés à la normalisation.

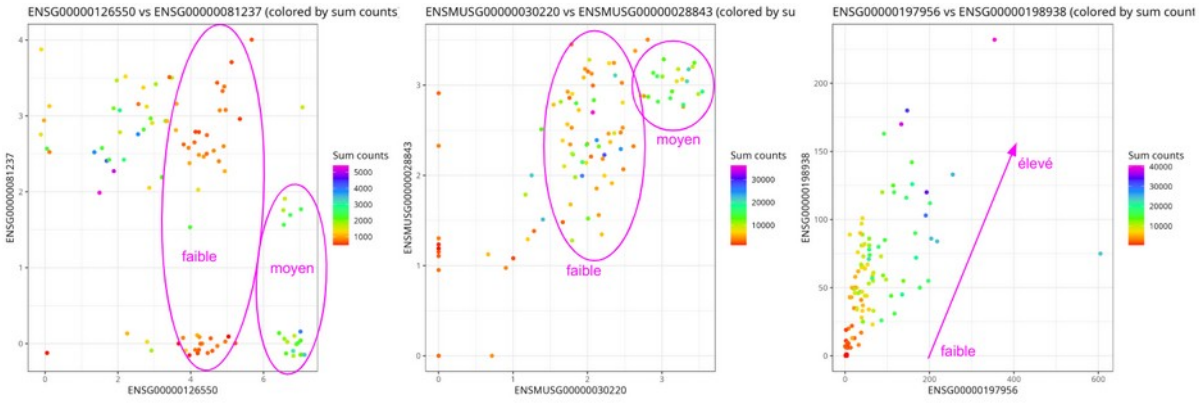

Figure 77: Exemples du motif du biais visible (Annexe XAI_CZ_CELLxGENE, im/figs/dr_top100_effects_visible_pattern.png)

Pattern **de la diagonale**

Comme vu précédemment, la présence de points alignés sur la diagonale suggère un nombre de *counts* identique sur les deux gènes représentés, dont la log normalisation a aboutit à des valeurs différentes dues au nombre différent de *counts* dans chaque cellule.

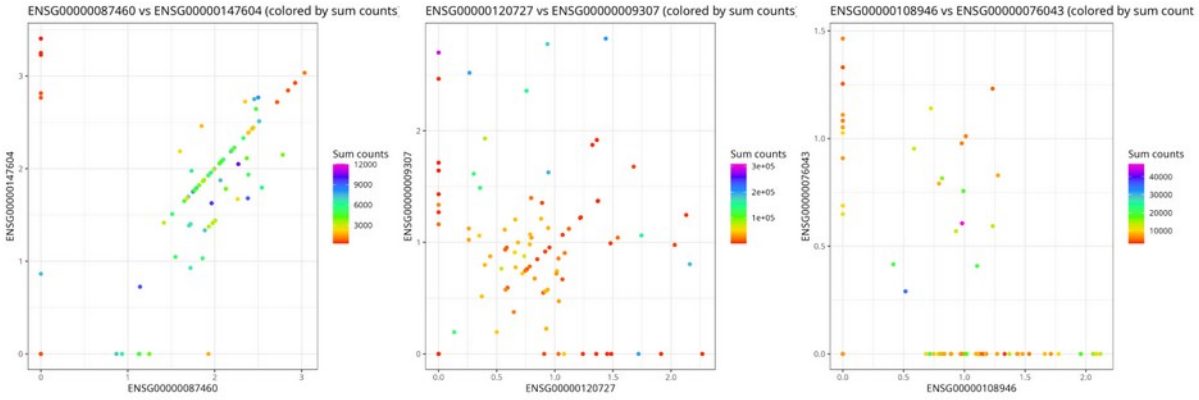

Figure 78: Exemples du motif de la diagonale (Annexe XAI_CZ_CELLxGENE, im/figs/dr_top100_effects_diag.png)

Pattern **des axes X et Y**

Certains *scatter plots* montrent une accumulation de points le long de l'axe X ou Y, un motif probablement lié aux effets de *dropout* (effet courant des technologies scRNA-seq attribuant un score de 0 *read* à des gènes pourtant exprimés). Il est difficile de savoir si les gènes sélectionnés dans ces cas reflètent un réel mécanisme biologique ou s'ils ont été retenus en raison de cet artefact. Nous considérons donc également ce motif comme suspect.

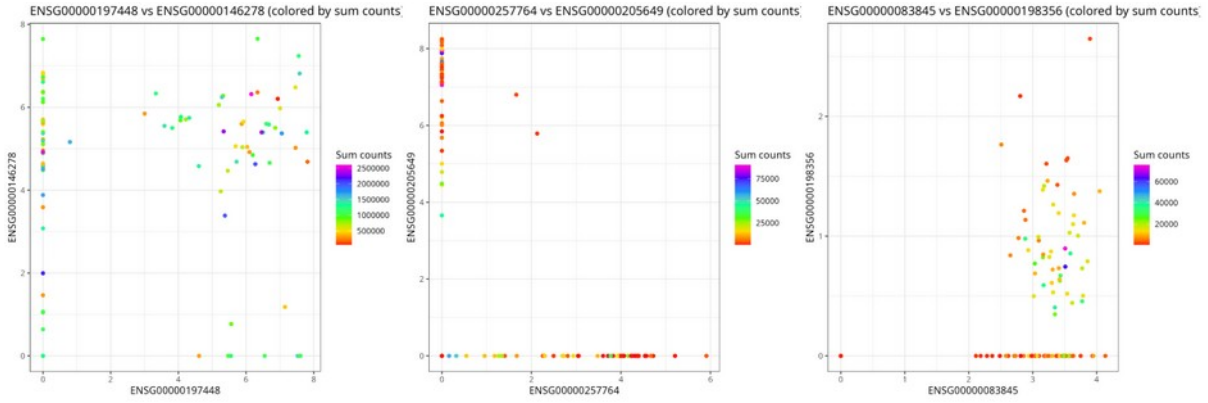

Figure 79: Exemples du motif des axes X et Y (Annexe XAI_CZ_CELLxGENE, im/figs/dr_top100_effects_axis.png)

Manque de variabilité

Certains *scatter plots* ne permettent pas d'évaluer correctement la distribution des nombres de *reads*, soit à cause d'un manque réel de variabilité, soit parce que l'échelle de couleur a été compressée par la présence d'*outlier*s. Dans les deux cas, cela donne des graphiques où presque tous les points apparaissent dans une couleur uniforme, rouge dans notre cas.

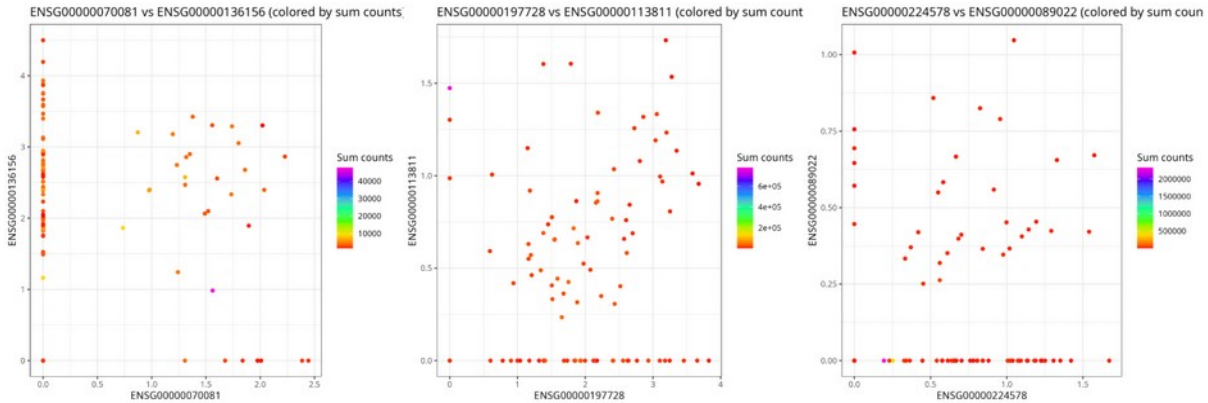

Figure 80: Exemples du motif du manque de variabilité (Annexe XAI_CZ_CELLxGENE, im/figs/dr_top100_effects_all_red.png)

Une comptabilité de ces motifs sur ce top 100 donne :

- ***Pattern* de biais visible** - 26/100
- ***Pattern* de la diagonale** - 13/100

- ***Pattern* des axes X et Y** - 17/100
- **Manque de variabilité –** 16/100

Certains *scatter plots* peuvent comporter plusieurs patterns, en tout 59/100 sont considérés comme suspects. Les 41 jeux de données restants sont utilisés pour la suite de l'analyse.

A titre d'illustration on trouve ci-dessous 4 panels de résultats, avec des informations sur le jeu de données d'origine, les deux types cellulaires étudiés et les gènes impliqués. Les deux premiers illustrent le phénomène de différence de variance, et les deux suivants le phénomène de distribution bimodale. Le fichier contenant les 41 panels est disponible en annexe *XAI_CZ_CELLxGENE, im/phd/axis2_application_summary_panels/success_panels_cz_cellxgene.pdf*.

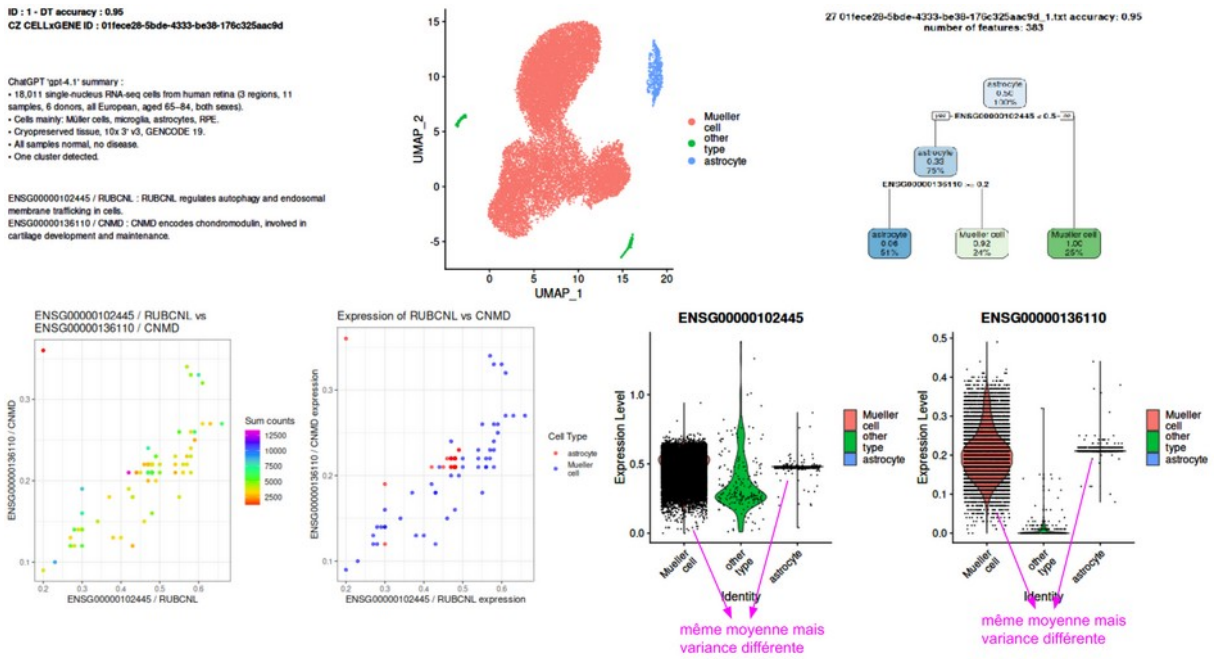

Figure 81: Panel de l'ID 1, accuracy de l'arbre de décision 0.96, illustre le scénario de différence de variance (Annexe XAI_CZ_CELLxGENE: im/phd/dr_app_panels_1.png)

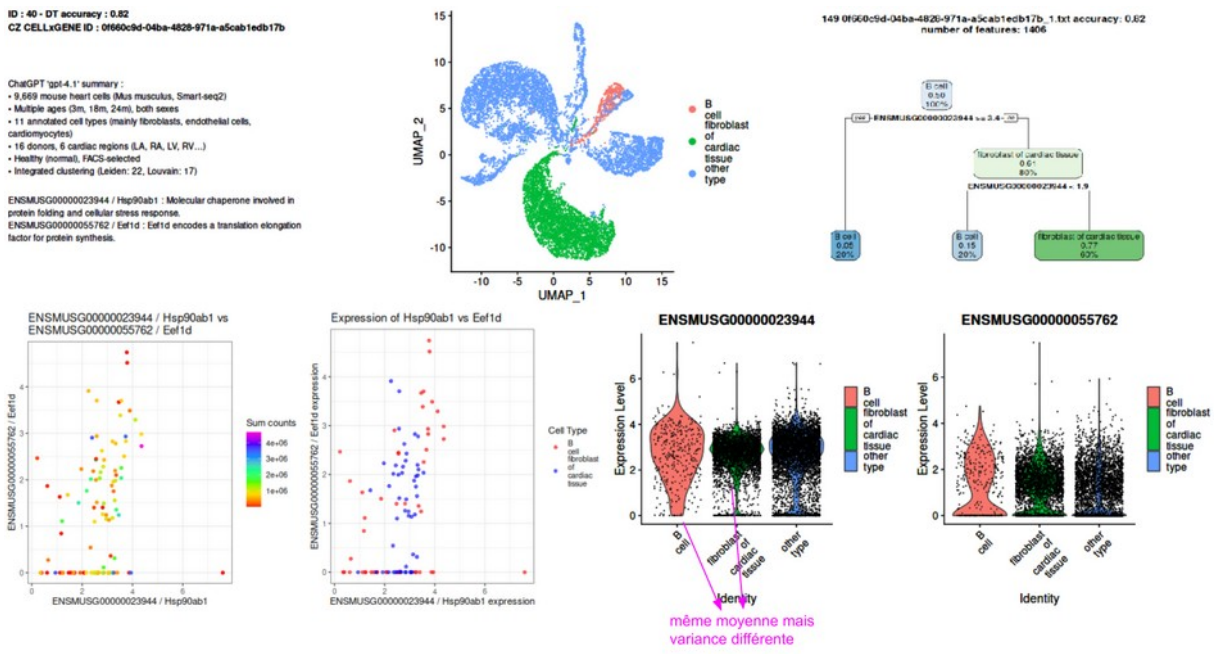

Figure 82: Panel de l'ID 40, accuracy de l'arbre de décision 0.82, illustre le scénario de différence de variance (Annexe XAI_CZ_CELLxGENE: im/phd/dr_app_panels_2.png)

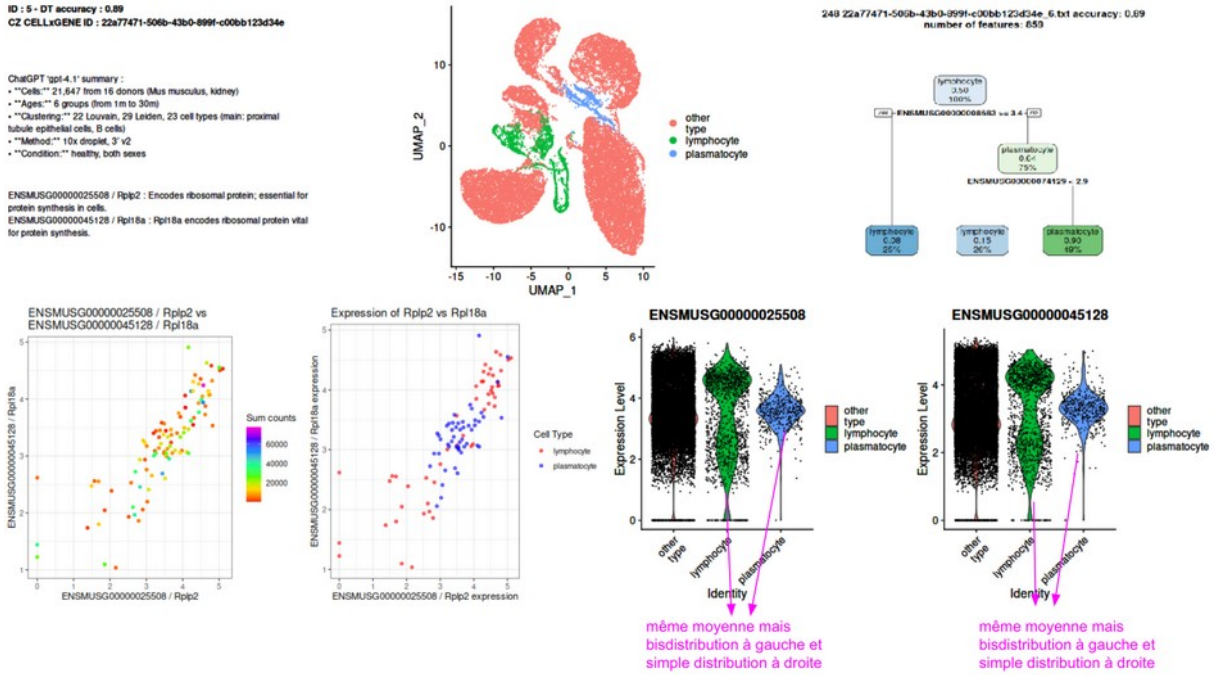

Figure 83: Panel de l'ID 5, accuracy de l'arbre de décision 0.89, illustre le scénario de distribution bimodale (Annexe XAI_CZ_CELLxGENE: im/phd/dr_app_panels_3.png)

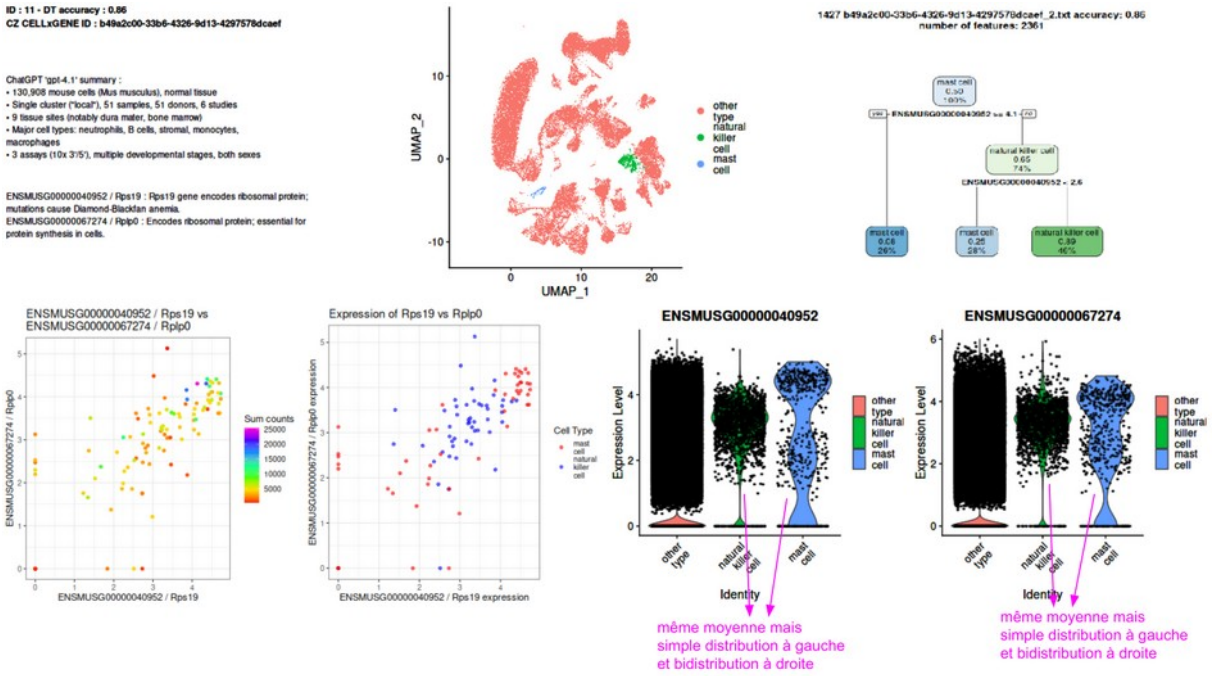

Figure 84: Panel de l'ID 11, accuracy de l'arbre de décision 0.86, illustre le scénario de distribution bimodale (Annexe XAI_CZ_CELLxGENE: im/phd/dr_app_panels_4.png)

On a classifié les 41 jeux de données en fonction du scénario du gène avec le plus grand score de Gini. Nous n'avons pas trouvé de situation claire de relation entre les gènes comme ceux définis en scénario 3 (XOR like motifs) ou en scénario 4 (cercles concentriques). Par contre, sur les 41, 9 correspondent aux différences de variances, 13 correspondent à des distributions bimodales, et 19 correspondent à des situations moins bien établies. Une poursuite des analyses serait nécessaire pour déterminer quelle proportion de ces cas reflète des différences biologiques réelles, ce qui constitue notre objectif principal, et quelle part relève de biais d'analyse, pouvant amener à des algorithmes de détection d'anomalies. Nous explorons ces pistes plus en détail dans la section discussion du manuscrit.

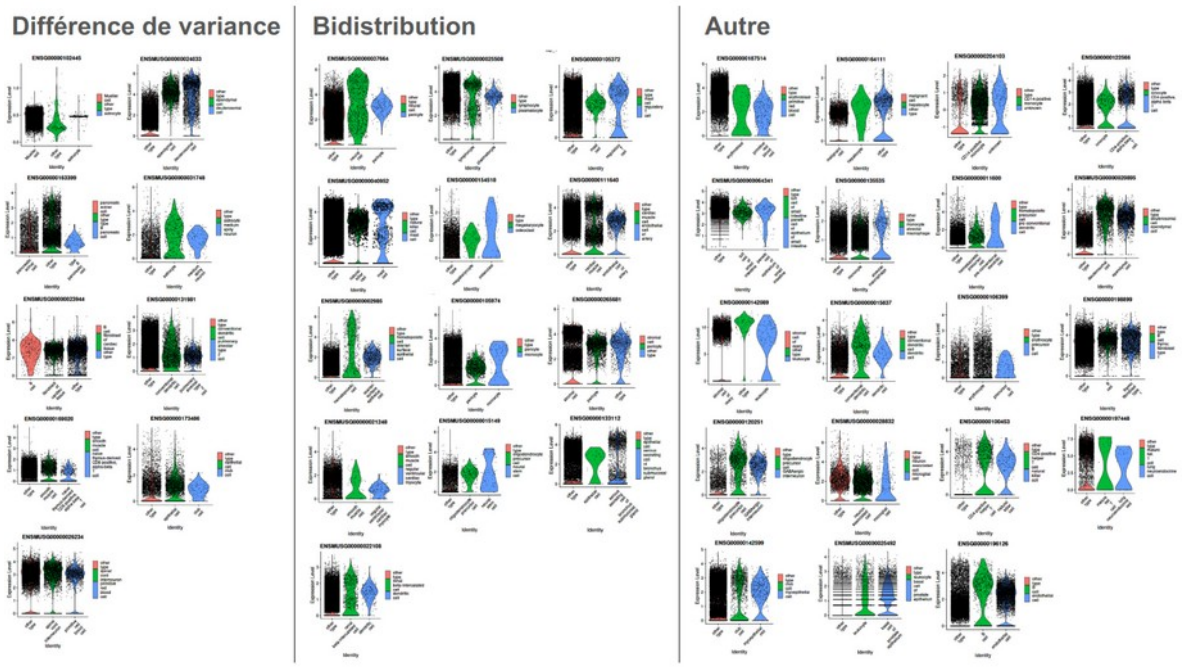

Figure 85: Résumé des scénarios d'appartenance du gène ayant le plus haut score de Gini pour les 41 arbres (Annexe XAI_CZ_CELLxGENE: im/phd/dr_var_bidistrib_other.png)

Discussion et perspectives

Ce travail de thèse a exploré le potentiel de l'intelligence artificielle explicable pour l'analyse des données omiques à travers deux approches complémentaires.

Le premier axe s'est concentré sur la fiabilité des modèles. En développant l'outil *adverSCarial*, nous avons évalué la robustesse des classifieurs scRNA-seq face à des perturbations des données. Nos résultats ont montré que différents types de modèles présentent des vulnérabilités spécifiques, ce qui confirme la nécessité de tester leur fiabilité avant utilisation en conditions réelles.

Le second axe a porté sur l'utilisation de la XAI pour la découverte scientifique. Nous avons mis au point des fonctions disponibles dans le package *XAItest*, incluant la méthode *simThresh*, pour montrer comment l'explicabilité des modèles peut révéler des signatures biologiques complexes que les tests statistiques classiques ne détectent pas. L'analyse à grande échelle de données réelles de *CZ CELLxGENE* a confirmé que notre approche est scalable, qu'elle peut servir à identifier des artefacts techniques, agissant comme un outil de découverte et de contrôle qualité.

Ensemble, ces deux axes montrent comment la XAI transforme les modèles de *machine learning* de "boîtes noires" en outils d'analyse plus transparents et fiables. Le premier axe questionne leur robustesse, tandis que le second exploite leur logique interne pour la découverte. La suite de cette discussion va maintenant synthétiser les apports et les limites de ce travail, et aborder les perspectives de recherche qu'il ouvre.

Dans l’axe 1 et à travers l’évaluation de cinq classifieurs scRNA-seq sur quatre jeux de données différents, nous avons mis en évidence un éventail de faiblesses face aux attaques adverses, dépendant à la fois du type d’algorithme d’apprentissage automatique utilisé et de l’ampleur des modifications appliquées. Par exemple, nous démontrons clairement que les valeurs aberrantes détournent facilement les prédictions des outils, et qu’ils doivent impérativement être filtrés lors des étapes de prétraitement pour garantir un fonctionnement correct. Toutefois, les classifieurs restent vulnérables aux biais techniques ou biologiques, même légers.

Logiquement, tous les classifieurs finissent par modifier leur prédiction au-delà d’un certain seuil de perturbation. Notre étude permet de quantifier ce seuil et d’évaluer s’il reste compatible avec la variabilité technique ou biologique attendue dans les données scRNA-seq. Que les attaques reposent sur des modifications substantielles de quelques gènes ou sur des altérations minimes réparties sur un grand nombre de gènes, un basculement de la prédiction intervient dès qu’un niveau cumulé de perturbation est atteint.

Ce constat met en évidence l’existence d’un seuil de sensibilité propre à chaque modèle. Les analyses de précision, ainsi que les courbes ROC et les scores F1, ont mis en lumière des différences de comportements entre classifieurs face à diverses attaques, révélant à la fois des vulnérabilités générales et les façons spécifiques dont chaque algorithme y répond.

Comparé aux outils d’attaque adverse existants, *adverSCarial* présente l’avantage d’être compatible avec plusieurs algorithmes de classification, potentiellement tous

types de modèles exploitant des données scRNA-seq. Il propose différents modes d'attaque adverse, dont deux ne nécessitent pas d'accès au gradient, et un autre qui l'approche par approximation, ce qui permet de tester presque tous les modèles *white box* ou *black box*.

En prenant directement en charge les objets *Seurat* ou *SingleCellExperiment*, et en encapsulant le fonctionnement et les paramètres du classifieur, *adverSCarial* limite les manipulations de données dès le départ, simplifiant ainsi l'expérience utilisateur. Les versions futures pourraient intégrer un traitement parallèle pour réduire le temps de calcul, et inclure d'autres algorithmes d'attaque comme FGSM et PGD afin de permettre des analyses plus larges.

Pour la suite, il serait pertinent d'envisager l'intégration d'outils adversariaux directement dans les pipelines bioinformatiques standards, comme pour les étapes de contrôle qualité (QC). Alors que le QC permet de vérifier la cohérence des données avant l'analyse, un module de robustesse basé sur des attaques adverses pourrait systématiquement évaluer la stabilité des prédictions à la fin du pipeline. Une telle étape deviendrait un gage de fiabilité, en particulier pour des applications sensibles comme la recherche biomédicale ou des applications cliniques. Elle offrirait aux utilisateurs une assurance supplémentaire que les résultats obtenus ne dépendent pas d'une fragilité du modèle face à de légères variations techniques ou biologiques, et compléterait les pratiques actuelles d'analyse scRNA-seq.

Un autre point important concerne la généralisation et la reproductibilité des attaques adverses. Nos résultats montrent des faiblesses claires dans différents classifieurs, mais il reste à savoir si ces observations se retrouvent dans d'autres jeux de données, sur d'autres types cellulaires ou avec d'autres technologies de séquençage. Les modèles entraînés sur un jeu de données peuvent en effet présenter des fragilités spécifiques, qui ne s'appliquent pas forcément ailleurs. Pour évaluer plus solidement la robustesse, il serait donc utile de tester les attaques sur plusieurs cohortes et conditions expérimentales. Cela permettrait de voir si elles mettent en évidence des limites générales des algorithmes ou seulement des effets liés aux données initiales. Une telle approche renforcerait la confiance dans la portée des résultats et leur reproductibilité dans différents contextes.

Plusieurs variantes et améliorations de l'algorithme CGD pourraient être explorées. Un algorithme plus simple ignorant la probabilité de la seconde classe cellulaire la plus probable serait utile pour l'*adversarial training*, par exemple pour l'amélioration de la robustesse ou de la précision d'un classifieur. De plus, CGD modifie un gène dès qu'il influence, même marginalement, la prédiction du classifieur. Avec les modèles de réseaux de neurones, ce processus peut ne pas être optimal car les prédictions reposent souvent sur presque toutes les variables d'entrée. Introduire un seuil de gradient pour la modification des gènes pourrait améliorer l'efficacité, en garantissant que seuls des changements pertinents sont effectués.

Une autre piste d'amélioration réside dans la présélection des gènes à modifier. Dans cette étude, nous avons surtout cherché à évaluer la diversité des types de classifieurs, et à obtenir des informations sur les performances, la précision et la dynamique de prédiction lors des attaques avec *adverSCarial*. C'est pourquoi nous avons priorisé les gènes à tester à l'aide de classements préliminaires basés sur les statistiques différentielles entre tous les types cellulaires possibles. Ce n'est pas idéal dans le cadre d'attaques adverses, où il peut être préférable de cibler des marqueurs moins évidents. Pour surmonter cette limitation, et parce que tester chaque gène

individuellement avec un algorithme basé sur le gradient est coûteux en ressources, d'autres stratégies de présélection des gènes pourraient être envisagées, qu'elles soient aléatoires, basées sur la variance observée, ou en établissant des scores reflétant des caractéristiques phénotypiques.

Dans l'axe 2 nous avons vu un exemple d'étude sur données simulées donc nous avons pu tirer des enseignements.

Essais sur jeux de données simulés

Les tests statistiques classiques n'attrapent qu'une partie des signaux : dans nos six scénarios simulés (variance, bimodalité, XOR, cercles concentriques, régressions parabole et sinusoïde à amplitude croissante), les *t-test*s ou tests de Wilcoxon passent à côté de plusieurs configurations, en particulier les relations non linéaires et multivariées.

Les méthodes XAI récupèrent ces "angles morts" : la combinaison *Decision Tree* + Gini + *simThresh* détecte 10 / 10 fois les scénarios « différence de variance » et « bimodal », et atteint encore 7 / 10 (cercles) et 5 / 10 (XOR) pour les scénarios les plus difficiles.

L'algorithme *simThresh* domine les approches *PIMP* et mProbes : sur 50 configurations, *simThresh* sort vainqueur 43 fois, contre 7 pour *PIMP* et 0 pour mProbes (paramètres courants), soit ≈ 6 × plus de « victoires ».

SHAP est rarement meilleur que des métriques plus simples et coûte cher : sur 48 duels, *SHAP* ne gagne que 5 fois face au score de Gini, à la *decrease-accuracy* ou au score d'Olden. De plus il est coûteux en temps de calculs, surtout lorsqu'on le combine à *PIMP* ou mProbes.

Hiérarchie des temps de calcul observée :

- DT / RF / SVM < MLP
- *simThresh* < *PIMP*, mProbes
- Gini / *accuracy* / Olden < *SHAP*
- Le contraste extrême est de 0,15 s (*simThresh* + *no MLP* + no *SHAP*) *vs* 25 953 s (MLP + *SHAP* + *PIMP*/mProbes), un facteur ≈ 1.7×10^5

.

Essai large scale sur données réelles de *CZ CELLxGENE*

Les méthodes XAI détectent souvent des "biomarqueurs" qui sont en réalité des artefacts liés à la normalisation, comme la profondeur de séquençage.

Sur les données bien normalisées, les méthodes XAI ont retrouvé des motifs d'intérêt complexes comme la variance ou la bimodalité, souvent invisibles aux tests standards.

Les méthodes XAI peuvent donc servir à la fois à trouver des marqueurs et à surveiller la qualité des jeux de données.

L'ensemble du *workflow* a prouvé sa scalabilité avec la combinaison DT + Gini + *simThresh* : analyse automatique de milliers de cas sur CPU en quelques heures.

Création de *benchmarks* spécifiques

Une réponse aux limitations des analyses sur données simulées pourrait être la création de *benchmarks* biologiques dédiés à l'évaluation des méthodes XAI ou de découverte de marqueurs. Plusieurs *benchmarks* biologiques sont déjà utilisés pour évaluer les LLMs, c'est-à-dire des modèles de langage entraînés sur de vastes corpus textuels et capables de comprendre, générer ou structurer du langage naturel comme chatGPT, y compris dans des contextes bioinformatiques [133], comme :

- ***BioASQ*** : pour l'évaluation de systèmes de question/réponse biomédicaux à partir de publications scientifiques [134]
- ***BC5CDR*** : pour l'extraction d'entités nommées, maladies, gènes, composés chimiques, dans les textes biomédicaux [135]
- ***HoC, Hallmarks of Cancer*** : pour la classification multi-label de textes biomédicaux en fonction des mécanismes du cancer [136].

Cependant, ces *benchmarks* sont principalement conçus pour évaluer des tâches textuelles ou de classification. Ils ne sont pas adaptés à l'évaluation des méthodes visant à découvrir de nouveaux marqueurs biologiques, qui nécessiteraient plutôt des données labellisées simulées, le but étant d'être certain que les marqueurs à identifier relèvent bien d'un lien logique avec le label d'intérêt. Il serait donc pertinent de concevoir des *benchmarks* spécifiques à cette tâche. Concernant les données simulées utilisées dans ce travail de thèse, plusieurs améliorations pourraient être envisagées pour renforcer leur capacité d'évaluation.

Étendre la complexité du jeu de données simulé : dans ce travail, seules une vingtaine de variables ont été générées sur 100 échantillons répartis en deux classes A et B. Il serait pertinent d'explorer des jeux de données de plus grandes dimensions, en nombre de variables et/ou d'échantillons, pour mieux refléter les données bioinformatiques réelles et la scalabilité des méthodes évaluées.

Introduire des artefacts : l'ajout de valeurs manquantes, de valeurs aberrantes ou d'effets batchs permettrait de simuler des conditions expérimentales plus proches de celles rencontrées dans les données biologiques réelles.

Données multimodales : des jeux de données simulés intégrant d'autres modalités comme des images ou des séries temporelles permettraient d'évaluer les méthodes dans des contextes d'intégration de données hétérogènes. On peut rencontrer par exemple des données scRNA-seq couplé à l'imagerie, ou des données transcriptomiques couplées à des données cliniques.

D'autres méthodes de *feature importance*

Le travail pourrait également être étendu en considérant un plus grand nombre de méthodes XAI.

De nombreuses approches ont été développées pour expliquer les prédictions des modèles de *machine learning*, autres que celles que nous avons mentionné.

Linardatos et al propose une review sur ces algorithmes permettant d'obtenir des score de *feature importance* [137], en voici quelques exemples :

- ***Integrated Gradients*** : Approche basée sur les dérivées qui estime la contribution d'une *feature* en intégrant le gradient du modèle le long d'un chemin entre une entrée de référence (*baseline*) et l'entrée réelle. Elle satisfait des propriétés théoriques importantes comme la sensibilité et l'additivité. Utilisée principalement pour les réseaux profonds, notamment en vision ou génomique.
- ***Anchors* :** Méthode d'explication locale qui génère des règles de type if-then (ancres) qui garantissent une prédiction stable dans une zone de l'espace des données. Contrairement à *LIME*, qui produit des modèles linéaires locaux, Anchors vise à fournir des règles compréhensibles, exprimées avec peu de variables, tout en garantissant une fidélité élevée à la prédiction du modèle
- **TCAV (*Testing with Concept Activation Vectors*)** : Méthode conceptuelle qui permet d'interpréter un réseau de neurones en fonction de concepts humains définis à l'avance (ex. « inflammation », « structure nucléaire »). Elle quantifie dans quelle mesure un concept influence la sortie du modèle, en s'appuyant sur des directions dans l'espace latent. Très utile pour relier des prédictions à des notions biomédicales interprétables

D'autres méthodes de conversion en *p-values*

Huynh-Thu et al analyse par exemple plusieurs algorithmes de conversion de *feature importance* en *p-value* autre que *PIMP* et mProbes [138], en voici quelques exemples:

- ***CER* pour *Conditional Error Rate*** : Méthode qui utilise des tests par permutation. Elle calcule, pour chaque *feature*, la probabilité d'avoir au moins un faux positif dans le sous-ensemble considéré. Fournit un taux d'erreur conditionnel, interprétable statistiquement [139].
- ***nFDR* pour *naïve False Discovery Rate*** : Estime le FDR via permutation, en comparant la distribution obtenue permutée à la distribution réelle. Donne un seuil pour contrôler le taux de fausses découvertes [140] [138].
- ***eFDR* pour *estimated False Discovery Rate*** : Variante améliorée du *nFDR* en fournissant une estimation plus précise du FDR, plus proche du réel. Permet de convertir la *feature importance* en un seuil FDR, donc en une mesure statistique interprétable [141].
- ***mr-test*** : Test par permutation comparant le rang moyen d'une *feature* à celui attendu sous permutation produisant une p-value pour distinguer les variables réellement importantes [142].

Evaluation du temps de calcul

Concernant l'évaluation de la scalabilité des méthodes aux données de grande dimension, retenons que l'ensemble des expériences a été réalisé sur CPU. Cette configuration ne reflète pas les architectures matérielles actuellement utilisées pour

l'entraînement des modèles à base de réseaux de neurones, où les GPU sont privilégiés. Il est par exemple documenté que l'entraînement de réseaux de neurones multicouches, les MLP, peut être accéléré d'un facteur 4 à 5 sur GPU, voir d'un facteur 10 pour l'entraînement avec un paramètre de batch élevé, grâce à la parallélisation massive permise par plusieurs milliers de cœurs CUDA pour une carte GPU NVIDIAs classique, contre 8 à 32 cœurs sur CPU standard [143].

Détection d'anomalies avec les méthodes XAI

Dans ce travail, nous avons d'abord utilisé les méthodes XAI pour identifier des biomarqueurs dans les données omiques. L'objectif initial était de trouver les variables les plus importantes pour expliquer un phénomène biologique. Au fil de nos analyses, nous avons remarqué que ces méthodes ne détectent pas uniquement des biomarqueurs, mais plus généralement tout ce qui est atypique dans les données.

Autrement dit, les méthodes XAI ont tendance à mettre en évidence des éléments qui sortent de l'ordinaire, qu'il s'agisse d'un signal biologique réel comme un biomarqueur, ou d'un artefact technique, potentiellement un effet batch, une mauvaise normalisation, une mauvaise gestion des valeurs manquantes, etc.

Cela ouvre une nouvelle perspective, ces méthodes pourrait également servir à détecter des anomalies dans les données, en plus de leur rôle dans la découverte. Dans certains cas, on ne peut pas savoir directement si la variable détectée est un vrai biomarqueur ou une anomalie technique. Il est alors nécessaire de faire des analyses complémentaires, soit bioinformatiques par exemple en croisant plusieurs sources de données, soit par validation expérimentale en laboratoire.

Une idée intéressante serait de construire des jeux de données simulés dans lesquels on introduit volontairement des effets batch ou des erreurs de normalisation, pas pour aider la découverte de biomarqueurs en situation difficile comme proposé précédemment, mais pour évaluer directement la capacité des méthodes XAI à détecter ces anomalies. Par exemple, on pourrait :

- créer des lots artificiels avec des décalages de moyennes ou de variances
- introduire des valeurs manquantes, puis mal les imputer
- appliquer des normalisations incorrectes sur tout ou partie des données

On pourrait aussi fabriquer des jeux de données "réalistes mais biaisés" en partant de jeux publics bien normalisés et en y ajoutant des perturbations de manière contrôlée. Cela permettrait de créer une ressource de test spécifique pour mesurer la sensibilité des méthodes XAI à différents types d'anomalies.

Enfin, il serait utile d'intégrer cette capacité de détection d'anomalies dans les pipelines analytiques, comme étape exploratoire systématique, avant d'analyser les résultats.

XAI sur *embedding* dans les réseaux de neurones, *autoencoders* et *sparse autoencoders*

Dans les sections précédentes, nous avons décrit des méthodes d'explicabilité relativement simples, basées sur des modèles supervisés classiques appliqués à des

données mono-omiques. Cependant, de nouvelles approches permettent d'intégrer des données complexes, multi-omiques et/ou multimodales, tout en fournissant des explications interprétables. Par exemple, *CustOmics* propose des architectures *deep learning* qui permet d'intégrer plusieurs sources de données omiques et des images, au sein d'un modèle unifié, en passant par un *embedding* commun des jeux de données [144].

Contrairement à notre approche mono-omique, *CustOmics* opère en deux phases : il entraîne d'abord un réseau dédié à chaque type de données, puis apprend une représentation commune, l'*embedding*, qui capture les interactions entre ces différentes modalités. En tenant compte des spécificités de chaque source, cette méthode exploite plus efficacement la complémentarité des données multi-omiques que des fusions simples, après *embedding* séparé de chaque modalité.

Avec *SHAP*, *CustOmics* peut fournir des résultats interprétables même avec des données hétérogènes. Il existe cependant d'autres méthodes XAI qui arrivent à donner du sens à cet *embedding*. Une méthode prometteuses consiste à employer des *sparse autoencoders*, afin de forcer le réseau à décomposer ses activations internes en composantes plus lisibles. A l'origine les *autoencoders* fonctionnent dans un mode opposé, sous forme de goulot d'étranglement visant à forcer le réseau de neurones à réduire les dimensions de l'input tout en gardant l'information essentielle. En imposant au contraire une forte sparsité, une couche de neurones plus grande mais avec de nombreuses valeurs nulles, ces autoencodeurs obligent le modèle à isoler des caractéristiques discriminantes, en théorie plus facilement interprétables. Cette approche a été testée avec succès dans différents contextes, par exemple sur des modèles de protéines et des LLMs.

Modèles de protéines

La méthode *Matryoshka SAEs* [145] utilise un *sparse autoencoder* pour « lire » les neurones d'un modèle de langage protéique, *ESMFold* [146], et en a extrait des milliers de *features* interprétables correspondant à des concepts biologiques connus, comme des binding sites, des motifs structuraux, domaines fonctionnels, etc. Leur travail montre que ces concepts sont superposés dans les neurones du modèle, et que l'*autoencoder* a pu les séparer en dimensions distinctes et compréhensibles.

LLMs

L'entreprise *Anthropic*, créatrice du LLM Claude, a démontré qu'en entraînant un *sparse autoencoder* sur les activations internes d'un grand modèle de langage, on peut isoler des *features* sémantiques utiles [147]. Dit autrement, cette technique résout en partie la polysémie des neurones, le fait qu'un neurone encode plusieurs concepts à la fois, en produisant des variables qui ne s'activent chacune que pour une seule notion bien précise.

Ces méthodes utilisant les *embeddings* et les *sparses autoencoders*, et ouvrent la voie à l'utilisation du *framework* XAI proposé aux intégrations multimodales en permettant d'interpréter ce qu'apprennent les neurones internes des réseaux de neurones.

Corrélation *vs* causalité

Les méthodes XAI, tout comme les approches statistiques traditionnelles, permettent d'identifier des relations associatives entre variables, par exemple entre l'expression d'un gène et un phénotype, ou entre un score de *SHAP* et une classe prédite. Toutefois, ces relations restent le plus souvent de l'ordre de la corrélation, sans que l'on puisse conclure à une relation causale. Autrement dit, même si un modèle met en évidence qu'un gène est prédictif d'un état cellulaire, cela ne signifie pas nécessairement qu'il en est la cause.

Pourtant, déterminer si un lien est causal a un grand intérêt, notamment en recherche biomédicale pour identifier des cibles thérapeutiques fiables.

Plusieurs approches statistiques permettent de s'attaquer à cette question. Par exemple, les modèles de graphes causaux (tels que les *Bayesian Networks* ou les *Directed Acyclic Graphs*) qui permettent de modéliser des dépendances entre variables et d'inférer des liens causaux potentiels à partir de données observationnelles, en intégrant des hypothèses structurelles sur le système.

Dans le domaine des données génétiques, la méthode de *Mendelian Randomization* (MR) offre une approche pour tester la causalité à partir de variantes génétiques. Elle part du principe que certains SNPs influencent un facteur biologique (comme l'expression d'un gène), qui lui-même influence un phénotype. Si cette chaîne est confirmée, cela suggère un lien de cause à effet. Si le SNP influence le facteur, et que ce facteur influence à son tour le phénotype, alors un lien causal peut être suggéré. Cette approche est particulièrement utile dans les études à grande échelle comme les GWAS.

Apprentissage frugal sur données rares

L'un des défis mis en évidence dans ce travail est la question du passage à l'échelle : à mesure que les bases de données biologiques croissent, entraîner des modèles sur des données déjà vues ou redondantes apporte un gain marginal. Il devient alors pertinent de repenser les stratégies d'entraînement pour les rendre plus frugales, en sélectionnant les sous-ensembles de données rares, atypiques ou informatives. Apprendre à identifier et à prioriser ce type de jeux de données, plutôt que de traiter uniformément des corpus massifs, permettrait de maximiser le taux de découverte pour un coût calculatoire réaliste. Si cette problématique ne se pose pas encore de manière critique aujourd'hui, elle deviendra incontournable avec l'explosion du volume de données multi-omiques. Cela pourrait ouvrir la voie au développement de nouvelles méthodes pour l'optimisation du tri et de la sélection des données d'apprentissage à grande échelle.

Des méthodes pour la confiance dans les tests cliniques utilisant les marqueurs identifiés par les méthodes XAI

Traditionnellement, les biomarqueurs utilisés en clinique reposent sur des seuils simples : un taux sanguin ou un niveau d'expression dépasse un certain seuil, le test est considéré comme positif. Ces approches sont efficaces lorsque la distribution des valeurs montre une différence claire entre les groupes, ce qui n'est justement pas forcément le cas pour les marqueurs identifiés avec les méthodes XAI proposées dans

ce travail de thèse. Ils ne présentent pas forcément de différence de moyenne entre groupes, mais peuvent être informatifs dans leur combinaison. Pour pouvoir les exploiter il devient alors pertinent d'utiliser des algorithmes de type *machine learning*, des classifieurs, capables de prendre en compte les motifs nécessaires. Ces modèles, une fois entraînés, peuvent être utilisés comme outils décisionnels cliniques : ils donnent une prédiction binaire (positif/négatif), mais fondée sur des interactions complexes entre variables. Or, qui dit algorithme de ML dans un contexte médical dit nécessité de comprendre les critères de décision du modèle, nécessitant le développement de méthodes XAI spécifiques.

Modèles ML spécifiquement conçus pour l'explicabilité

Actuellement, la plupart des modèles d'apprentissage automatique sont conçus pour optimiser leurs performances prédictives, sans chercher l'interprétabilité. C'est dans un second temps qu'on applique des méthodes d'XAI pour tenter de comprendre leur fonctionnement. Il pourrait être intéressant d'inverser cette logique, en développant des architectures pensées dès le départ pour être transparentes et interprétables, sans sacrifier leur performance.

Plusieurs travaux explorent déjà cette idée. Par exemple, les modèles dits "*self-explaining neural networks*" [148] intègrent dans leur structure même des modules dédiés à l'interprétation. Comme on l'a vu, d'autres intègrent des "*sparse encoders*" dans leurs réseaux de neurones, pour obtenir des représentations compressées dans lesquelles seule une minorité de neurones sont activés pour chaque prédiction. Cela favorise une forme de lisibilité interne, car chaque neurone activé peut alors être interprété comme portant un concept distinct.

Ces approches restent encore minoritaires, mais elles ouvrent la voie à des modèles plus fiables et directement exploitables dans des contextes sensibles comme la médecine ou la recherche biomédicale.

Étude des lymphomes, Transformation de Richter *vs* DLBCL *de novo*

Le projet CGPSR (Caractérisation Génomique et Proéomique du Syndrome de Richter ; *ClinicalTrials.gov ID : NCT03619512*) vise à mieux comprendre les mécanismes de résistance aux traitements dans les lymphomes, en particulier en comparant les DLBCL (*Diffuse Large B-cell Lymhoma*) *de novo* et les transformations de Richter (TR), qui correspondent à l'évolution d'une leucémie lymphoïde chronique (LLC) vers une forme secondaire de DLBCL. La LLC est une hémopathie maligne fréquente chez l'adulte, caractérisée par l'accumulation de lymphocytes B matures mais dysfonctionnels dans les ganglions, le sang, et la moelle osseuse. La TR survient dans environ 2 à 10 % des cas de LLC, et son pronostic est sombre, avec une médiane de survie après diagnostic de quelques mois.

Dans ce projet, nous analysons des données protéomiques issues de 55 biopsies ganglionnaires de patients atteints de DLBCL *de novo* ou de TR, générées par LC-MS/MS. Cette technique produit un grand nombre de valeurs manquantes, présentant un challenge pour les analyses. Ces données manquantes sont en grande partie

MNAR (*Missing Not At Random*), c'est-à-dire qu'une protéine est absente non pas par manque de sensibilité analytique, mais parce qu'elle est réellement absente dans certains échantillons. Ces absences peuvent donc être informatives biologiquement, par exemple si une protéine est détectée uniquement chez les patients chimiosensibles ou au contraire uniquement chez les chimiorésistants.

La méthode classique consiste à appliquer un test statistique, comme le **test de Fisher**, individuellement pour chaque protéine, afin d'identifier celles dont la *p-value* est inférieure à un seuil défini. Une autre approche consiste à sélectionner directement les protéines les plus différentielles, par exemple en prenant les meilleures selon un classement, sans utiliser de seuil de *p-value*. La sélection se fait sur une valeur statistique. On peut voir le résultat de la sélection du top 20 du test de Fisher dans la figure ci-dessous.

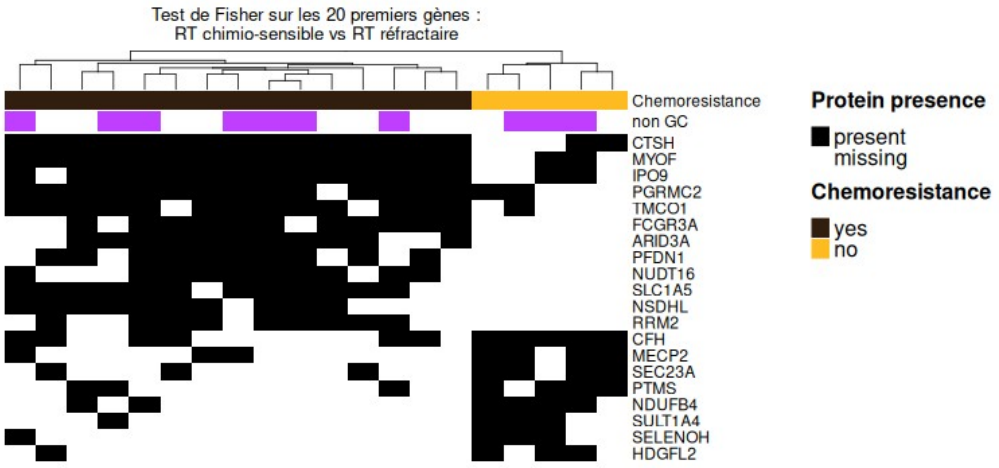

Figure 86: Heatmap des 20 protéines dont la p-value du test de Fisher sur chimiosensibles vs chimioréfractaires est la plus faible. (Annexe CONTENT4PHD : im/figs/0341_hm_missings_fisher_top20.png, 0341_R_FOR_PHD_mnars_decision_trees_RF_pretty_top5.ipynb)

Une autre méthode consiste à utiliser les outils XAI développés précédemment. Nous avons entraîné un modèle de ***random forest*** avec le package *R* randomForest, avec nombre d'arbres=500 et les valeurs par défaut pour les autres paramètres, pour prédire la chimio résistance à partir de la valeur présence/absence de chacune des 4390 protéines. Dans un second temps nous avons attribué un score de *feature importance* pour chaque protéine. Nous avons sélectionné le top 20 de ces protéines. Cette sélection n'est donc pas sur une base statistique de différence entre deux groupes, mais sur le potentiel de séparation des deux groupes à partir d'arbres de décision. Il est donc possible que certaines protéines présentent une *p-value* élevée au test de Fisher, suggérant une faible différence globale entre les groupes, tout en étant néanmoins utiles pour classer correctement un sous-groupe spécifique. À l'inverse, des protéines avec une *p-value* faible peuvent essentiellement refléter les caractéristiques du sous-groupe majoritaire, sans contribuer à une bonne

discrimination des autres sous-groupes.On peut voir le résultat de la sélection du top 20 par *feature importance* sur RF sur la figure ci-dessous.

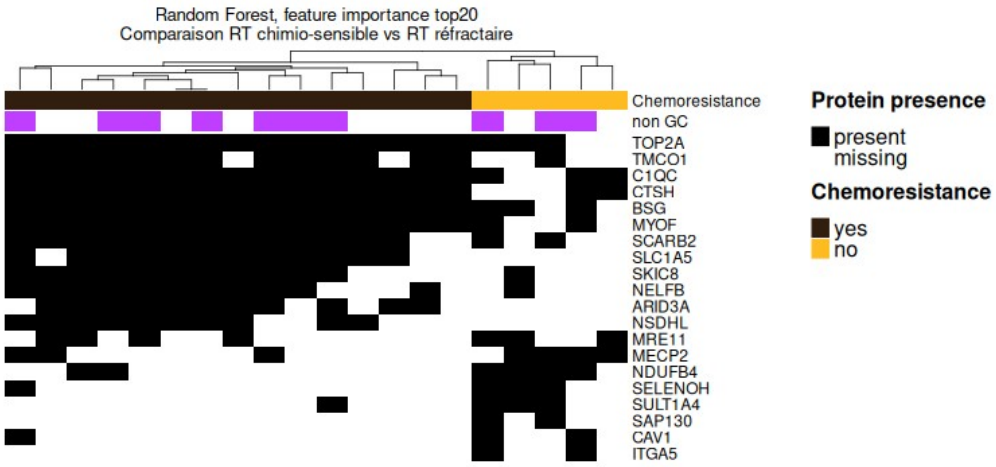

Figure 87: Heatmap des 20 protéines dont la feature importance est la plus élevée après entraînement d'un modèle RF sur chimiosensibles vs chimioréfractaires. (Annexe CONTENT4PHD : im/figs/0341_hm_missings_RF_featImp_top20.png, 0341_R_FOR_PHD_mnars_decision_trees_RF_pretty_top5.ipynb)

Cette comparaison montre que les deux méthodes mettent en évidence des protéines différentes, suggérant qu'une approche explicable comme celle-ci pourrait aider à identifier des scénarios comme ceux évoqués dans cette thèse. Ces résultats serviront à un article en préparation, dans lequel nous visons une intégration multi-omique pour améliorer la stratification des patients et la compréhension des mécanismes de résistance dans les TR et les DLBCL *de novo*.

Conclusion

En conclusion, *adverSCarial* introduit de nouvelles approches pour identifier les faiblesses face aux attaques adverses et évaluer la robustesse des classifieurs de transcriptomique en single cell. Ses fonctionnalités variées permettent des modulations flexibles dans les modes d'attaque et les types de modifications, allant de changements subtils et pratiquement indétectables à des altérations plus visibles. Sur différents jeux de données scRNA-seq et des outils de ML tels que *scType*, *CHETAH*, *scAnnotatR*, et des modèles basés sur RF ou MLP, il a mis en évidence de nombreuses vulnérabilités et profils de sensibilité aux modifications d'expression génique, soulignant les défis pour garantir des classifications fiables en présence d'évènements imprévus. À mesure que les méthodes de scRNA-seq se développent dans de nombreux domaines, y compris la médecine de précision, nous pensons qu'*adverSCarial* pourra constituer un outil précieux pour les chercheurs, développeurs et utilisateurs finaux afin d'examiner les vulnérabilités, répondre aux préoccupations de sécurité et de confiance dans les algorithmes, et d'identifier les mécanismes décisionnels internes des algorithmes de ML exploitant les données scRNA-seq.

Le second axe de la thèse s'inscrit dans un contexte d'expansion massive des données biologiques, de limite des outils statistiques classiques et de démocratisation des outils de *machine learning*. Face à cette complexité croissante, le besoin de modèles à la fois performants et scalables devient central. En explorant différentes approches d'intelligence artificielle explicable appliquées aux données omiques, ce travail met en lumière à la fois les possibilités offertes par ces méthodes, et les limites à ne pas ignorer.

Nous avons montré qu'au-delà de l'interprétabilité des modèles, certaines méthodes permettent d'identifier des signatures biologiques difficiles à détecter par des approches classiques.

À l'avenir, un des enjeux majeurs sera d'unifier ces avancées dans des outils opérationnels, fiables et généralisables, capables d'être utilisés par les bioinformaticiens et biologistes comme des assistants d'analyse et d'exploration. Cela nécessitera des efforts conjoints en modélisation, en ingénierie logicielle et en expérimentation, mais ouvre la perspective d'une bioinformatique qui tirera le plein potentiel de la masse des données biologiques disponibles pour l'analyse des cas particuliers, dans la recherche biomédicale ou en utilisation clinique.

Références

[1] J. Jumper *et al.*, “Highly accurate protein structure prediction with AlphaFold,” *Nature*, vol. 596, no. 7873, pp. 583–589, Aug. 2021, doi: 10.1038/s41586-021-03819-2.

[2] J. Abramson *et al.*, “Accurate structure prediction of biomolecular interactions with AlphaFold 3,” *Nature*, vol. 630, no. 8016, pp. 493–500, Jun. 2024, doi: 10.1038/s41586-024-07487-w.

[3] J. L. Watson *et al.*, “De novo design of protein structure and function with RFdiffusion,” *Nature*, vol. 620, no. 7976, pp. 1089–1100, Aug. 2023, doi: 10.1038/s41586-023-06415-8.

[4] J. Dauparas *et al.*, “Robust deep learning–based protein sequence design using ProteinMPNN,” *Science*, vol. 378, no. 6615, pp. 49–56, Oct. 2022, doi: 10.1126/science.add2187.

[5] *R*. Mishra and B. Li, “The Application of Artificial Intelligence in the Genetic Study of Alzheimer’s Disease,” *Aging Dis.*, vol. 11, no. 6, pp. 1567–1584, Dec. 2020, doi: 10.14336/AD.2020.0312.

[6] W. Wei, S. Visweswaran, and G. F. Cooper, “The application of naive Bayes model averaging to predict Alzheimer’s disease from genome-wide data,” *J. Am. Med. Inform. Assoc.*, vol. 18, no. 4, pp. 370–375, Jul. 2011, doi: 10.1136/amiajnl-2011-000101.

[7] L. Xu, G. Liang, C. Liao, G.-D. Chen, and C.-C. Chang, “An Efficient Classifier for Alzheimer’s Disease Genes Identification,” *Molecules*, vol. 23, no. 12, Art. no. 12, Dec. 2018, doi: 10.3390/molecules23123140.

[8] S. Ekins, “The Next Era: Deep Learning in Pharmaceutical Research,” *Pharm. Res.*, vol. 33, no. 11, pp. 2594–2603, Nov. 2016, doi: 10.1007/s11095-016-2029-7.

[9] J. M. Stokes *et al.*, “A deep learning approach to antibiotic discovery,” *Cell*, vol. 180, no. 4, pp. 688-702.e13, Feb. 2020, doi: 10.1016/j.cell.2020.01.021.

[10] Lhou, “Comment casser Big Pharma grâce à l’IA,” Génération Do It Yourself. Accessed: Jan. 26, 2025. [Online]. Available: https://www.gdiy.fr/business/thomas-clozel/

[11] J. M. Choi and H. Chae, “moBRCA-net: a breast cancer subtype classification *framework* based on multi-omics attention neural networks,” *BMC Bioinformatics*, vol. 24, no. 1, p. 169, Apr. 2023, doi: 10.1186/s12859-023-05273-5.

[12] G. Akshay, M. Irfan, S. KG, and A. Singh, “Skin-Vision: An Innovative Mobile-Based Automated Skin Disease Detection Application,” in *2023 OITS International Conference on Information Technology (OCIT)*, Dec. 2023, pp. 835–840. doi: 10.1109/OCIT59427.2023.10430941.

[13] A. T. Young *et al.*, “The role of technology in melanoma screening and diagnosis,” *Pigment Cell Melanoma Res.*, vol. 34, no. 2, pp. 288–300, 2021, doi: 10.1111/pcmr.12907.

[14] “Intelligence artificielle : la révolution à l’hôpital, pour améliorer la prise en charge des patients,” France 3 Grand Est. Accessed: Jan. 27, 2025. [Online]. Available: https://france3-regions.francetvinfo.fr/grand-est/meurthe-et-moselle/

nancy/intelligence-artificielle-la-revolution-a-l-hopital-pour-ameliorer-la-prise-en-charge-des-patients-2950982.html
[15] ICL_Team, “L’Intelligence Artificielle au service des patients en radiothérapie à l’ICL,” Institut de Cancérologie de Lorraine. Accessed: Jan. 27, 2025. [Online]. Available: https://www.icl-lorraine.fr/intelligence-artificielle-patients-radiotherapie-icl-hopital-nancy/
[16] H. Teymourian, A. Barfidokht, and J. Wang, “Electrochemical glucose sensors in diabetes management: an updated review (2010–2020),” *Chem. Soc. Rev.*, vol. 49, no. 21, pp. 7671–7709, Nov. 2020, doi: 10.1039/D0CS00304B.
[17] A.-L. M. Bowler, J. Whitfield, L. Marshall, V. G. Coffey, L. M. Burke, and G. *R*. Cox, “The Use of Continuous Glucose Monitors in Sport: Possible Applications and Considerations,” Dec. 2022, doi: 10.1123/ijsnem.2022-0139.
[18] Z. Wu, H. Wu, K. Fang, K. S.-F. Sze, and Q. Feng, “A Transformer-Based Deep Learning Model for Sleep Apnea Detection and Application on RingConn Smart Ring,” in *2024 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2024, pp. 1–5. doi: 10.1109/ISCAS58744.2024.10558425.
[19] E. N. Erickson, N. Gotlieb, L. M. Pereira, L. Myatt, C. Mosquera-Lopez, and P. G. Jacobs, “Predicting labor onset relative to the estimated date of delivery using smart ring physiological data,” *Npj Digit. Med.*, vol. 6, no. 1, pp. 1–13, Aug. 2023, doi: 10.1038/s41746-023-00902-y.
[20] W. Ruan *et al.*, “Large Language Models for Bioinformatics,” Jan. 10, 2025, *arXiv*: arXiv:2501.06271. doi: 10.48550/arXiv.2501.06271.
[21] A. Vaswani *et al.*, “Attention is All you Need,” in *Advances in Neural Information Processing Systems*, Curran Associates, Inc., 2017. Accessed: Jun. 06, 2025. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html
[22] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” May 24, 2019, *arXiv*: arXiv:1810.04805. doi: 10.48550/arXiv.1810.04805.
[23] Y. Ji, Z. Zhou, H. Liu, and *R*. V. Davuluri, “DNABERT: pre-trained Bidirectional Encoder Representations from Transformers model for DNA-language in genome,” *Bioinformatics*, vol. 37, no. 15, pp. 2112–2120, Aug. 2021, doi: 10.1093/bioinformatics/btab083.
[24] J. J. Garau-Luis *et al.*, “Multi-modal Transfer Learning between Biological Foundation Models,” Jun. 20, 2024, *arXiv*: arXiv:2406.14150. doi: 10.48550/arXiv.2406.14150.
[25] J. McCarthy, M. L. Minsky, N. Rochester, and C. E. Shannon, “A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence, August 31, 1955,” *AI Mag.*, vol. 27, no. 4, Art. no. 4, Dec. 2006, doi: 10.1609/aimag.v27i4.1904.
[26] A. L. Samuel, “Some Studies in Machine Learning Using the Game of Checkers,” *IBM J. Res. Dev.*, vol. 3, no. 3, pp. 210–229, Jul. 1959, doi: 10.1147/rd.33.0210.
[27] A. Vlahou, J. O. Schorge, B. W. Gregory, and *R*. L. Coleman, “Diagnosis of Ovarian Cancer Using *Decision Tree* Classification of Mass Spectral Data,” *BioMed Res. Int.*, vol. 2003, no. 5, p. 231929, 2003, doi: 10.1155/S1110724303210032.

[28] "*Decision Tree*s for the Analysis of Gene Expression Levels of COVID-19: An Association with Alzheimer's Disease." Accessed: Aug. 17, 2025. [Online]. Available: https://www.mdpi.com/2673-7426/5/2/26
[29] F. J. H. Brims *et al.*, "A Novel Clinical Prediction Model for Prognosis in Malignant Pleural Mesothelioma Using *Decision Tree* Analysis," *J. Thorac. Oncol.*, vol. 11, no. 4, pp. 573–582, Apr. 2016, doi: 10.1016/j.jtho.2015.12.108.
[30] "Gene selection and classification of microarray data using *random forest* | BMC Bioinformatics." Accessed: Aug. 17, 2025. [Online]. Available: https://link.springer.com/article/10.1186/1471-2105-7-3
[31] C. Contini *et al.*, "Combined High—Throughput Proteomics and *Random Forest* Machine-Learning Approach Differentiates and Classifies Metabolic, Immune, Signaling and ECM Intra-Tumor Heterogeneity of Colorectal Cancer," *Cells*, vol. 13, no. 16, p. 1311, Jan. 2024, doi: 10.3390/cells13161311.
[32] F. Miao, Y.-P. Cai, Y.-X. Zhang, X.-M. Fan, and Y. Li, "Predictive Modeling of Hospital Mortality for Patients With Heart Failure by Using an Improved Random Survival Forest," *IEEE Access*, vol. 6, pp. 7244–7253, 2018, doi: 10.1109/ACCESS.2018.2789898.
[33] M. P. S. Brown *et al.*, "Knowledge-based analysis of microarray gene expression data by using support vector machines," *Proc. Natl. Acad. Sci.*, vol. 97, no. 1, pp. 262–267, Jan. 2000, doi: 10.1073/pnas.97.1.262.
[34] B. Sahoo, Z. Pinnix, S. Sims, and A. Zelikovsky, "Identifying Biomarkers Using Support Vector Machine to Understand the Racial Disparity in Triple-Negative Breast Cancer," *J. Comput. Biol.*, vol. 30, no. 4, pp. 502–517, Apr. 2023, doi: 10.1089/cmb.2022.0422.
[35] X. Yuan, Z. Yang, G. Zouridakis, and N. Mullani, "SVM-based Texture Classification and Application to Early Melanoma Detection," in *2006 International Conference of the IEEE Engineering in Medicine and Biology Society*, Aug. 2006, pp. 4775–4778. doi: 10.1109/IEMBS.2006.260056.
[36] I. D. Mienye, T. G. Swart, G. Obaido, M. Jordan, and P. Ilono, "Deep Convolutional Neural Networks in Medical Image Analysis: A Review," *Information*, vol. 16, no. 3, p. 195, Mar. 2025, doi: 10.3390/info16030195.
[37] B. Alipanahi, A. Delong, M. T. Weirauch, and B. J. Frey, "Predicting the sequence specificities of DNA- and RNA-binding proteins by deep learning," *Nat. Biotechnol.*, vol. 33, no. 8, pp. 831–838, Aug. 2015, doi: 10.1038/nbt.3300.
[38] Q. Shi *et al.*, "DNN-Dom: predicting protein domain boundary from sequence alone by deep neural network," *Bioinformatics*, vol. 35, no. 24, pp. 5128–5136, Dec. 2019, doi: 10.1093/bioinformatics/btz464.
[39] M. Mostavi, Y.-C. Chiu, Y. Huang, and Y. Chen, "Convolutional neural network models for cancer type prediction based on gene expression," *BMC Med. Genomics*, vol. 13, no. 5, p. 44, Apr. 2020, doi: 10.1186/s12920-020-0677-2.
[40] X. Yang, Y. Zheng, X. Xing, X. Sui, W. Jia, and H. Pan, "Immune subtype identification and multi-layer perceptron classifier construction for breast cancer," *Front. Oncol.*, vol. 12, p. 943874, Dec. 2022, doi: 10.3389/fonc.2022.943874.
[41] J. Carreras and *R*. Hamoudi, "Artificial Neural Network Analysis of Gene Expression Data Predicted Non-Hodgkin Lymphoma Subtypes with High Accuracy,"

Mach. Learn. Knowl. Extr., vol. 3, no. 3, pp. 720–739, Sep. 2021, doi: 10.3390/make3030036.
[42] S. T. Hill, *R*. Kuintzle, A. Teegarden, E. Merrill III, P. Danaee, and D. A. Hendrix, "A deep recurrent neural network discovers complex biological rules to decipher RNA protein-coding potential," *Nucleic Acids Res.*, vol. 46, no. 16, pp. 8105–8113, Sep. 2018, doi: 10.1093/nar/gky567.
[43] S. Saadatnejad, M. Oveisi, and M. Hashemi, "LSTM-Based ECG Classification for Continuous Monitoring on Personal Wearable Devices," *IEEE J. Biomed. Health Inform.*, vol. 24, no. 2, pp. 515–523, Feb. 2020, doi: 10.1109/JBHI.2019.2911367.
[44] M. Yap *et al.*, "Verifying explainability of a deep learning tissue classifier trained on RNA-seq data," *Sci. Rep.*, vol. 11, no. 1, p. 2641, Jan. 2021, doi: 10.1038/s41598-021-81773-9.
[45] B. Pfeifer, H. Baniecki, A. Saranti, P. Biecek, and A. Holzinger, "Multi-omics disease module detection with an explainable Greedy Decision Forest," *Sci. Rep.*, vol. 12, no. 1, p. 16857, Oct. 2022, doi: 10.1038/s41598-022-21417-8.
[46] *R*. O. Alabi, M. Elmusrati, I. Leivo, A. Almangush, and A. A. Mäkitie, "*Machine learning* explainability in nasopharyngeal cancer survival using *LIME* and *SHAP*," *Sci. Rep.*, vol. 13, no. 1, p. 8984, Jun. 2023, doi: 10.1038/s41598-023-35795-0.
[47] Y. Zhao, J. Shao, and Y. W. Asmann, "Assessment and Optimization of Explainable Machine Learning Models Applied to Transcriptomic Data," *Genomics Proteomics Bioinformatics*, vol. 20, no. 5, pp. 899–911, Oct. 2022, doi: 10.1016/j.gpb.2022.07.003.
[48] M. Divate, A. Tyagi, D. J. Richard, P. A. Prasad, H. Gowda, and S. H. Nagaraj, "Deep Learning-Based Pan-Cancer Classification Model Reveals Tissue-of-Origin Specific Gene Expression Signatures," *Cancers*, vol. 14, no. 5, p. 1185, Feb. 2022, doi: 10.3390/cancers14051185.
[49] M. De-Arteaga *et al.*, "Bias in Bios: A Case Study of Semantic Representation Bias in a High-Stakes Setting," arXiv.org. Accessed: Jun. 08, 2025. [Online]. Available: https://arxiv.org/abs/1901.09451v1
[50] J. Dastin, "Insight - Amazon scraps secret AI recruiting tool that showed bias against women," *Reuters*, Oct. 11, 2018. Accessed: Jun. 08, 2025. [Online]. Available: https://www.reuters.com/article/world/insight-amazon-scraps-secret-ai-recruiting-tool-that-showed-bias-against-women-idUSKCN1MK0AG/
[51] M. T. Ribeiro, S. Singh, and C. Guestrin, "'Why Should I Trust You?': Explaining the Predictions of Any Classifier," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, in KDD '16. New York, NY, USA: Association for Computing Machinery, Aug. 2016, pp. 1135–1144. doi: 10.1145/2939672.2939778.
[52] D. Hanahan and *R*. A. Weinberg, "The hallmarks of cancer," *Cell*, vol. 100, no. 1, pp. 57–70, Jan. 2000, doi: 10.1016/s0092-8674(00)81683-9.
[53] D. Hanahan and *R*. A. Weinberg, "Hallmarks of cancer: the next generation," *Cell*, vol. 144, no. 5, pp. 646–674, Mar. 2011, doi: 10.1016/j.cell.2011.02.013.
[54] D. Hanahan, "Hallmarks of Cancer: New Dimensions," *Cancer Discov.*, vol. 12, no. 1, pp. 31–46, Jan. 2022, doi: 10.1158/2159-8290.CD-21-1059.
[55] *R*. J. Klein *et al.*, "Complement factor H polymorphism in age-related macular degeneration," *Science*, vol. 308, no. 5720, pp. 385–389, Apr. 2005, doi: 10.1126/science.1109557.

[56] B. Langmead, C. Trapnell, M. Pop, and S. L. Salzberg, “Ultrafast and memory-efficient alignment of short DNA sequences to the human genome,” *Genome Biol.*, vol. 10, no. 3, p. R25, 2009, doi: 10.1186/gb-2009-10-3-r25.
[57] H. Li and *R.* Durbin, “Fast and accurate short *read* alignment with Burrows-Wheeler transform,” *Bioinforma. Oxf. Engl.*, vol. 25, no. 14, pp. 1754–1760, Jul. 2009, doi: 10.1093/bioinformatics/btp324.
[58] P. Danecek *et al.*, “Twelve years of SAMtools and BCFtools,” *GigaScience*, vol. 10, no. 2, p. giab008, Feb. 2021, doi: 10.1093/gigascience/giab008.
[59] G. van der Auwera and B. D. O’Connor, *Genomics in the Cloud: Using Docker, GATK, and WDL in Terra*. O’Reilly Media, Incorporated, 2020.
[60] A. Dobin *et al.*, “STAR: ultrafast universal RNA-seq aligner,” *Bioinforma. Oxf. Engl.*, vol. 29, no. 1, pp. 15–21, Jan. 2013, doi: 10.1093/bioinformatics/bts635.
[61] Y. Liao, G. K. Smyth, and W. Shi, “featureCounts: an efficient general purpose program for assigning sequence *reads* to genomic *features*,” *Bioinformatics*, vol. 30, no. 7, pp. 923–930, Apr. 2014, doi: 10.1093/bioinformatics/btt656.
[62] M. I. Love, W. Huber, and S. Anders, “Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2,” *Genome Biol.*, vol. 15, no. 12, p. 550, Dec. 2014, doi: 10.1186/s13059-014-0550-8.
[63] Y. Chen, L. Chen, A. T. L. Lun, P. L. Baldoni, and G. K. Smyth, “edgeR v4: powerful differential analysis of sequencing data with expanded functionality and improved support for small *counts* and larger jeux de données,” *Nucleic Acids Res.*, vol. 53, no. 2, p. gkaf018, Jan. 2025, doi: 10.1093/nar/gkaf018.
[64] M. E. Ritchie *et al.*, “limma powers differential expression analyses for RNA-sequencing and microarray studies,” *Nucleic Acids Res.*, vol. 43, no. 7, p. e47, Apr. 2015, doi: 10.1093/nar/gkv007.
[65] G. K. Smyth, “Linear models and empirical bayes methods for assessing differential expression in microarray experiments,” *Stat. Appl. Genet. Mol. Biol.*, vol. 3, p. Article3, 2004, doi: 10.2202/1544-6115.1027.
[66] A. M. Newman *et al.*, “Robust enumeration of cell subsets from tissue expression profiles,” *Nat. Methods*, vol. 12, no. 5, pp. 453–457, May 2015, doi: 10.1038/nmeth.3337.
[67] F. Tang *et al.*, “mRNA-Seq whole-transcriptome analysis of a single cell,” *Nat. Methods*, vol. 6, no. 5, pp. 377–382, May 2009, doi: 10.1038/nmeth.1315.
[68] M. Stoeckius *et al.*, “Simultaneous epitope and transcriptome measurement in single cells,” *Nat. Methods*, vol. 14, no. 9, pp. 865–868, Sep. 2017, doi: 10.1038/nmeth.4380.
[69] “BD AbSeq | Antibody Conjugates | BD Biosciences.” Accessed: Jun. 29, 2025. [Online]. Available: https://www.bdbiosciences.com/en-us/products/reagents/*single-cell*-multiomics/abseq
[70] P. L. Ståhl *et al.*, “Visualization and analysis of gene expression in tissue sections by spatial transcriptomics,” *Science*, vol. 353, no. 6294, pp. 78–82, Jul. 2016, doi: 10.1126/science.aaf2403.
[71] F. Tatyana *et al.*, “Case study of a neuroendocrine tumor of uncertain origin: *single-cell* transcriptomics unravels potential primary location,” *J. Cancer Res. Clin. Oncol.*, vol. 151, no. 1, p. 28, Dec. 2024, doi: 10.1007/s00432-024-06071-z.

[72] K. Li *et al.*, “Single-cell analysis reveals the chemotherapy-induced cellular reprogramming and novel therapeutic targets in relapsed/refractory acute myeloid leukemia,” *Leukemia*, vol. 37, no. 2, pp. 308–325, Feb. 2023, doi: 10.1038/s41375-022-01789-6.
[73] *R.* Arora *et al.*, “Spatial transcriptomics reveals distinct and conserved tumor core and edge architectures that predict survival and targeted therapy response,” *Nat. Commun.*, vol. 14, no. 1, p. 5029, Aug. 2023, doi: 10.1038/s41467-023-40271-4.
[74] Y. Hao *et al.*, “Dictionary learning for integrative, multimodal and scalable *single-cell* analysis,” *Nat. Biotechnol.*, vol. 42, no. 2, pp. 293–304, Feb. 2024, doi: 10.1038/s41587-023-01767-y.
[75] “SCANPY: large-scale *single-cell* gene expression data analysis | Genome Biology.” Accessed: Aug. 14, 2025. [Online]. Available: https://link.springer.com/article/10.1186/s13059-017-1382-0
[76] “Trajectory-based differential expression analysis for *single-cell* sequencing data | Nature Communications.” Accessed: Aug. 14, 2025. [Online]. Available: https://www.nature.com/articles/s41467-020-14766-3
[77] Y. Hao *et al.*, “Integrated analysis of multimodal *single-cell* data,” *Cell*, vol. 184, no. 13, pp. 3573-3587.e29, Jun. 2021, doi: 10.1016/j.cell.2021.04.048.
[78] Y. Takano *et al.*, “Spatially resolved gene expression profiling of tumor microenvironment reveals key steps of lung adenocarcinoma development,” *Nat. Commun.*, vol. 15, no. 1, p. 10637, Dec. 2024, doi: 10.1038/s41467-024-54671-7.
[79] M. F. de Oliveira *et al.*, “High-definition spatial transcriptomic profiling of immune cell populations in colorectal cancer,” *Nat. Genet.*, pp. 1–12, Jun. 2025, doi: 10.1038/s41588-025-02193-3.
[80] Y. Yang *et al.*, “Spatial transcriptomics analysis identifies therapeutic targets in diffuse high-grade gliomas,” *Front. Mol. Neurosci.*, vol. 17, Oct. 2024, doi: 10.3389/fnmol.2024.1466302.
[81] S. Tyanova, T. Temu, and J. Cox, “The MaxQuant computational platform for mass spectrometry-based shotgun proteomics,” *Nat. Protoc.*, vol. 11, no. 12, pp. 2301–2319, Dec. 2016, doi: 10.1038/nprot.2016.136.
[82] X. Zhang, A. H. Smits, G. B. van Tilburg, H. Ovaa, W. Huber, and M. Vermeulen, “Proteome-wide identification of ubiquitin interactions using UbIA-MS,” *Nat. Protoc.*, vol. 13, no. 3, pp. 530–550, Mar. 2018, doi: 10.1038/nprot.2017.147.
[83] “Presentation of the platform Proteomics Strasbourg.” Accessed: Jun. 28, 2025. [Online]. Available: https://plateforme-psge.u-strasbg.fr/PLATEFORME_LSMBO_WEB/UK/PAGE_Presentation.awp?utm_source=chatgpt.com
[84] Z. Ye *et al.*, “High-throughput and scalable single cell proteomics identifies over 5000 proteins per cell,” Nov. 28, 2023, *bioRxiv*. doi: 10.1101/2023.11.27.568953.
[85] *R.* Lister *et al.*, “Human DNA methylomes at base resolution show widespread epigenomic differences,” *Nature*, vol. 462, no. 7271, pp. 315–322, Nov. 2009, doi: 10.1038/nature08514.
[86] C. Luo *et al.*, “Single-cell methylomes identify neuronal subtypes and regulatory elements in mammalian cortex,” *Science*, vol. 357, no. 6351, pp. 600–604, Aug. 2017, doi: 10.1126/science.aan3351.

[87] F. Krueger and S. *R*. Andrews, “Bismark: a flexible aligner and methylation caller for Bisulfite-Seq applications,” *Bioinformatics*, vol. 27, no. 11, pp. 1571–1572, Jun. 2011, doi: 10.1093/bioinformatics/btr167.
[88] A. Akalin *et al.*, “methylKit: a comprehensive *R* package for the analysis of genome-wide DNA methylation profiles,” *Genome Biol.*, vol. 13, no. 10, p. R87, Oct. 2012, doi: 10.1186/gb-2012-13-10-r87.
[89] M. J. Aryee *et al.*, “Minfi: a flexible and comprehensive Bioconductor package for the analysis of Infinium DNA methylation microarrays,” *Bioinformatics*, vol. 30, no. 10, pp. 1363–1369, May 2014, doi: 10.1093/bioinformatics/btu049.
[90] Y. Tian *et al.*, “ChAMP: updated methylation analysis pipeline for Illumina BeadChips,” *Bioinformatics*, vol. 33, no. 24, pp. 3982–3984, Dec. 2017, doi: 10.1093/bioinformatics/btx513.
[91] J. D. Buenrostro, P. G. Giresi, L. C. Zaba, H. Y. Chang, and W. J. Greenleaf, “Transposition of native chromatin for fast and sensitive epigenomic profiling of open chromatin, DNA-binding proteins and nucleosome position,” *Nat. Methods*, vol. 10, no. 12, pp. 1213–1218, Dec. 2013, doi: 10.1038/nmeth.2688.
[92] “Genome-Wide Mapping of in Vivo Protein-DNA Interactions | Science.” Accessed: Aug. 15, 2025. [Online]. Available: https://www.science.org/doi/10.1126/science.1141319
[93] E. Lieberman-Aiden *et al.*, “Comprehensive Mapping of Long-Range Interactions Reveals Folding Principles of the Human Genome,” *Science*, vol. 326, no. 5950, pp. 289–293, Oct. 2009, doi: 10.1126/science.1181369.
[94] J. D. Buenrostro *et al.*, “Single-cell chromatin accessibility reveals principles of regulatory variation,” *Nature*, vol. 523, no. 7561, pp. 486–490, Jul. 2015, doi: 10.1038/nature14590.
[95] “HiC-Pro: an optimized and flexible pipeline for Hi-C data processing | Genome Biology | Full Text.” Accessed: Aug. 15, 2025. [Online]. Available: https://genomebiology.biomedcentral.com/articles/10.1186/s13059-015-0831-x
[96] C. A. Smith, E. J. Want, G. O’Maille, *R*. Abagyan, and G. Siuzdak, “XCMS: Processing Mass Spectrometry Data for Metabolite Profiling Using Nonlinear Peak Alignment, Matching, and Identification,” *Anal. Chem.*, vol. 78, no. 3, pp. 779–787, Feb. 2006, doi: 10.1021/ac051437y.
[97] Z. Pang *et al.*, “Using MetaboAnalyst 5.0 for LC–HRMS spectra processing, multi-omics integration and covariate adjustment of global metabolomics data,” *Nat. Protoc.*, vol. 17, no. 8, pp. 1735–1761, Aug. 2022, doi: 10.1038/s41596-022-00710-w.
[98] “Welcome to Oxford Nanopore Technologies,” Oxford Nanopore Technologies. Accessed: Jun. 28, 2025. [Online]. Available: https://nanoporetech.com/
[99] “Single-Molecule Protein Sequencing Solutions,” Quantum-Si. Accessed: Jun. 28, 2025. [Online]. Available: https://www.quantum-si.com/
[100] I. J. Goodfellow, J. Shlens, and C. Szegedy, “Explaining and Harnessing Adversarial Examples,” Mar. 20, 2015, *arXiv*: arXiv:1412.6572. doi: 10.48550/arXiv.1412.6572.
[101] J. Deng, W. Dong, *R*. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “ImageNet: A large-scale hierarchical image database,” in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2009, pp. 248–255. doi: 10.1109/CVPR.2009.5206848.

[102] C. Szegedy *et al.*, “Going Deeper with Convolutions,” Sep. 17, 2014, *arXiv*: arXiv:1409.4842. doi: 10.48550/arXiv.1409.4842.
[103] S. Kornblith, M. Norouzi, H. Lee, and G. Hinton, “Similarity of Neural Network Representations Revisited,” in *Proceedings of the 36th International Conference on Machine Learning*, PMLR, May 2019, pp. 3519–3529. Accessed: Jun. 06, 2025. [Online]. Available: https://proceedings.mlr.press/v97/kornblith19a.html
[104] X. Wang, Y. Peng, L. Lu, Z. Lu, M. Bagheri, and *R*. M. Summers, “ChestX-ray8: Hospital-scale Chest X-ray Database and Benchmarks on Weakly-Supervised Classification and Localization of Common Thorax Diseases,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017, pp. 3462–3471. doi: 10.1109/CVPR.2017.369.
[105] T. Han *et al.*, “Advancing diagnostic performance and clinical usability of neural networks via *adversarial training* and dual batch normalization,” *Nat. Commun.*, vol. 12, no. 1, p. 4315, Jul. 2021, doi: 10.1038/s41467-021-24464-3.
[106] N. Ghaffari Laleh *et al.*, “attaques adverses and adversarial robustness in computational pathology,” *Nat. Commun.*, vol. 13, no. 1, p. 5711, Sep. 2022, doi: 10.1038/s41467-022-33266-0.
[107] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” May 24, 2019, *arXiv*: arXiv:1810.04805. doi: 10.48550/arXiv.1810.04805.
[108] D. Jin, Z. Jin, J. T. Zhou, and P. Szolovits, “Is BERT Really Robust? A Strong Baseline for Natural Language Attack on Text Classification and Entailment,” Apr. 08, 2020, *arXiv*: arXiv:1907.11932. doi: 10.48550/arXiv.1907.11932.
[109] X. Han, Y. Hu, L. Foschini, L. Chinitz, L. Jankelson, and *R*. Ranganath, “Deep learning models for electrocardiograms are susceptible to attaque adverse,” *Nat. Med.*, vol. 26, no. 3, pp. 360–363, Mar. 2020, doi: 10.1038/s41591-020-0791-x.
[110] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, “Towards Deep Learning Models Resistant to attaques adverses,” Sep. 04, 2019, *arXiv*: arXiv:1706.06083. doi: 10.48550/arXiv.1706.06083.
[111] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, “Towards Deep Learning Models Resistant to attaques adverses,” Sep. 04, 2019, *arXiv*: arXiv:1706.06083. doi: 10.48550/arXiv.1706.06083.
[112] M. Sadria, A. Layton, and G. D. Bader, “Adversarial training improves model interpretability in *single-cell* RNA-seq analysis,” *Bioinforma. Adv.*, vol. 3, no. 1, p. vbad166, Jan. 2023, doi: 10.1093/bioadv/vbad166.
[113] M. Z. Joel *et al.*, “Using Adversarial Images to Assess the Robustness of Deep Learning Models Trained on Diagnostic Images in Oncology,” *JCO Clin. Cancer Inform.*, no. 6, p. e2100170, Mar. 2022, doi: 10.1200/CCI.21.00170.
[114] A. Yinusa and M. Faezipour, “A multi-layered defense against attaques adverses in brain tumor classification using ensemble *adversarial training* and *feature* squeezing,” *Sci. Rep.*, vol. 15, no. 1, p. 16804, May 2025, doi: 10.1038/s41598-025-00890-x.
[115] X. Zhao, S. Wu, N. Fang, X. Sun, and J. Fan, “Evaluation of *single-cell* classifiers for *single-cell* RNA sequencing data sets,” *Brief. Bioinform.*, vol. 21, no. 5, pp. 1581–1595, Sep. 2020, doi: 10.1093/bib/bbz096.

[116] Y. Tan and P. Cahan, “SingleCellNet: A Computational Tool to Classify Single Cell RNA-Seq Data Across Platforms and Across Species,” *Cell Syst.*, vol. 9, no. 2, pp. 207-213.e2, Aug. 2019, doi: 10.1016/j.cels.2019.06.004.
[117] V. Nguyen and J. Griss, “scAnnotatR: *framework* to accurately classify cell types in *single-cell* RNA-sequencing data,” *BMC Bioinformatics*, vol. 23, no. 1, p. 44, Jan. 2022, doi: 10.1186/s12859-022-04574-5.
[118] Y. Lieberman, L. Rokach, and T. Shay, “CaSTLe – Classification of single cells by transfer learning: Harnessing the power of publicly available single cell RNA sequencing experiments to annotate new experiments,” *PLOS ONE*, vol. 13, no. 10, p. e0205499, Oct. 2018, doi: 10.1371/journal.pone.0205499.
[119] A. W. Zhang *et al.*, “Probabilistic cell-type assignment of *single-cell* RNA-seq for tumor microenvironment profiling,” *Nat. Methods*, vol. 16, no. 10, pp. 1007–1015, Oct. 2019, doi: 10.1038/s41592-019-0529-1.
[120] G. Fievet, J. Broséus, D. Meyre, and S. Hergalant, “adverSCarial: assessing the vulnerability of *single-cell* RNA-sequencing classifiers to attaques adverses,” *Bioinformatics*, vol. 41, no. 4, p. btaf168, Apr. 2025, doi: 10.1093/bioinformatics/btaf168.
[121] A. Ianevski, A. K. Giri, and T. Aittokallio, “Fully-automated and ultra-fast cell-type identification using specific *marker* combinations from *single-cell* transcriptomic data,” *Nat. Commun.*, vol. 13, no. 1, p. 1246, Mar. 2022, doi: 10.1038/s41467-022-28803-w.
[122] J. K. de Kanter, P. Lijnzaad, T. Candelli, T. Margaritis, and F. C. P. Holstege, “CHETAH: a selective, hierarchical cell type identification method for *single-cell* RNA sequencing,” *Nucleic Acids Res.*, vol. 47, no. 16, p. e95, Sep. 2019, doi: 10.1093/nar/gkz543.
[123] J. Klughammer *et al.*, “A multi-modal *single-cell* and spatial expression map of metastatic breast cancer biopsies across clinicopathological *features*,” *Nat. Med.*, vol. 30, no. 11, pp. 3236–3249, Nov. 2024, doi: 10.1038/s41591-024-03215-z.
[124] B. J. Stewart *et al.*, “Spatiotemporal immune zonation of the human kidney,” *Science*, vol. 365, no. 6460, pp. 1461–1466, Sep. 2019, doi: 10.1126/science.aat5031.
[125] Y. Hao *et al.*, “Dictionary learning for integrative, multimodal and scalable *single-cell* analysis,” *Nat. Biotechnol.*, vol. 42, no. 2, pp. 293–304, Feb. 2024, doi: 10.1038/s41587-023-01767-y.
[126] S. M. Lundberg and S.-I. Lee, “A Unified Approach to Interpreting Model Predictions,” in *Advances in Neural Information Processing Systems*, Curran Associates, Inc., 2017. Accessed: Feb. 13, 2025. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/hash/8a20a8621978632d76c43dfd28b67767-Abstract.html
[127] L. Shapley, “7. A Value for n-Person Games. Contributions to the Theory of Games II (1953) 307-317.,” in *Classics in Game Theory*, H. W. Kuhn, Ed., Princeton University Press, 2020, pp. 69–79. doi: 10.1515/9781400829156-012.
[128] L. Breiman, “*Random Forest*s,” *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, Oct. 2001, doi: 10.1023/A:1010933404324.
[129] C. Gini, *Variabilità e mutabilità: contributo allo studio delle distribuzioni e delle relazioni statistiche. [Fasc. I.]*. Tipogr. di P. Cuppini, 1912.

[130] A. Altmann, L. Toloşi, O. Sander, and T. Lengauer, “Permutation importance: a corrected *feature importance* measure,” *Bioinformatics*, vol. 26, no. 10, pp. 1340–1347, May 2010, doi: 10.1093/bioinformatics/btq134.
[131] V. A. Huynh-Thu, Y. Saeys, L. Wehenkel, and P. Geurts, “Statistical interpretation of *machine learning*-based *feature importance* scores for biomarker discovery,” *Bioinformatics*, vol. 28, no. 13, pp. 1766–1774, Jul. 2012, doi: 10.1093/bioinformatics/bts238.
[132] CZI Cell Science Program *et al.*, “*CZ CELLxGENE* Discover: a *single-cell* data platform for scalable exploration, analysis and modeling of aggregated data,” *Nucleic Acids Res.*, vol. 53, no. D1, pp. D886–D900, Jan. 2025, doi: 10.1093/nar/gkae1142.
[133] W. Ruan *et al.*, “Large Language Models for Bioinformatics,” Jan. 10, 2025, *arXiv*: arXiv:2501.06271. doi: 10.48550/arXiv.2501.06271.
[134] G. Tsatsaronis *et al.*, “An overview of the BIOASQ large-scale biomedical semantic indexing and question answering competition,” *BMC Bioinformatics*, vol. 16, no. 1, p. 138, Apr. 2015, doi: 10.1186/s12859-015-0564-6.
[135] C.-H. Wei *et al.*, “Assessing the state of the art in biomedical relation extraction: overview of the BioCreative V chemical-disease relation (CDR) task,” *Database J. Biol. Databases Curation*, vol. 2016, p. baw032, 2016, doi: 10.1093/database/baw032.
[136] S. Baker *et al.*, “Automatic semantic classification of scientific literature according to the hallmarks of cancer,” *Bioinforma. Oxf. Engl.*, vol. 32, no. 3, pp. 432–440, Feb. 2016, doi: 10.1093/bioinformatics/btv585.
[137] P. Linardatos, V. Papastefanopoulos, and S. Kotsiantis, “Explainable AI: A Review of Machine Learning Interpretability Methods,” *Entropy*, vol. 23, no. 1, p. 18, Dec. 2020, doi: 10.3390/e23010018.
[138] V. A. Huynh-Thu, Y. Saeys, L. Wehenkel, and P. Geurts, “Statistical interpretation of *machine learning*-based *feature importance* scores for biomarker discovery,” *Bioinformatics*, vol. 28, no. 13, pp. 1766–1774, Jul. 2012, doi: 10.1093/bioinformatics/bts238.
[139] V. A. Huynh-Thu, L. Wehenkel, and P. Geurts, “Exploiting tree-based variable importances to selectively identify relevant variables,” in *Proceedings of the Workshop on New Challenges for Feature Selection in Data Mining and Knowledge Discovery at ECML/PKDD 2008*, PMLR, Sep. 2008, pp. 60–73. Accessed: Jun. 14, 2025. [Online]. Available: https://proceedings.mlr.press/v4/huynhthu08a.html
[140] Y. Ge, S. Dudoit, and T. P. Speed, “Resampling-based multiple testing for microarray data analysis,” *Test*, vol. 12, no. 1, pp. 1–77, Jun. 2003, doi: 10.1007/BF02595811.
[141] Y. Ge, S. C. Sealfon, and T. P. Speed, “SOME STEP-DOWN PROCEDURES CONTROLLING THE FALSE DISCOVERY RATE UNDER DEPENDENCE,” *Stat. Sin.*, vol. 18, no. 3, pp. 881–904, 2008.
[142] C. Zhang, X. Lu, and X. Zhang, “Significance of Gene Ranking for Classification of Microarray Samples,” *IEEE/ACM Trans. Comput. Biol. Bioinform.*, vol. 3, no. 3, pp. 312–320, Jul. 2006, doi: 10.1109/TCBB.2006.42.
[143] E. Buber and B. Diri, “Performance Analysis and CPU *vs* GPU Comparison for Deep Learning,” in *2018 6th International Conference on Control Engineering &*

Information Technology (CEIT), Istanbul, Turkey: IEEE, Oct. 2018, pp. 1–6. doi: 10.1109/CEIT.2018.8751930.
[144] H. Benkirane *et al.*, “Multimodal CustOmics: A unified and interpretable multi-task deep learning *framework* for multimodal integrative data analysis in oncology,” *PLOS Comput. Biol.*, vol. 21, no. 6, p. e1013012, Jun. 2025, doi: 10.1371/journal.pcbi.1013012.
[145] N. Parsan, D. J. Yang, and J. J. Yang, “Towards Interpretable Protein Structure Prediction with Sparse Autoencoders,” Mar. 11, 2025, *arXiv*: arXiv:2503.08764. doi: 10.48550/arXiv.2503.08764.
[146] Z. Lin *et al.*, “Evolutionary-scale prediction of atomic-level protein structure with a language model,” *Science*, vol. 379, no. 6637, pp. 1123–1130, Mar. 2023, doi: 10.1126/science.ade2574.
[147] “Towards Monosemanticity: Decomposing Language Models With Dictionary Learning.” Accessed: Jun. 26, 2025. [Online]. Available: https://transformer-circuits.pub/2023/monosemantic-*features*
[148] D. Alvarez Melis and T. Jaakkola, “Towards Robust Interpretability with Self-Explaining Neural Networks,” in *Advances in Neural Information Processing Systems*, Curran Associates, Inc., 2018. Accessed: Jun. 29, 2025. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2018/hash/3e9f0fc9b2f89e043bc6233994dfcf76-Abstract.html
[149] “Protocol for preparing formalin-fixed paraffin-embedded musculoskeletal tissue samples from mice for spatial transcriptomics.” Accessed: Jun. 06, 2025. [Online]. Available: https://star-protocols.cell.com/protocols/3390

Annexes

Annexe *XAI_CZ_CELLxGENE*

Étude XAI sur les données de *CZ CELLxGENE* :
https://www.dropbox.com/scl/fo/cukny04bp16u8w93seunl/AD-RZ_8Lyq0dlqMmZmx6asE?rlkey=3fh9qjzajjjdoz76b8alz89un&st=fh3yx1ok&dl=0

Annexe *CONTENT4PHD*

Codes et images pour l'XAI sur données simulées et l'application sur la transformation de Richter :
https://www.dropbox.com/scl/fo/m2xnah19gcy6vd9qyhkn5/ALOM88WIwnrNh_GDse1Amx4?rlkey=cpz507gyf65dy7bzdemwdfp8v&st=pt8cewo0&dl=0

Annexe *ADVERSCARIAL*

Jupyter notebooks *R* et images pour l'axe 1, package *adverSCarial* et article Bioinformatics :
https://www.dropbox.com/scl/fo/s9gcpge678sxauqlj9974/AENFjXB3UnDGMuFahu4hAJM?rlkey=19qfa13wws4e9kmf2i79xqom2&st=z9ufm8x0&dl=0

Annexe *XAI_SIMU*

Codes et images permettant la réplication de l'étude XAI sur données simulées :
https://www.dropbox.com/scl/fo/fspgwbmdqhthoa3f35ndg/ALBpEQiFRLmOCYL0fJ1vn_E?rlkey=vlbh1qy65p62b0nyxi1nnq58x&st=ayqzqdfe&dl=0

Répertoires *Github* et *Bioconductor*

Code source de *adverSCarial : https://github.com/GhislainFievet/adverSCarial*

Lien *Bioconductor d'adverSCarial :*
https://www.bioconductor.org/packages/devel/bioc/html/adverSCarial.html

Code source de XAItest : https://github.com/GhislainFievet/XAItest

Lien *Bioconductor de XAItest :*
https://www.bioconductor.org/packages/devel/bioc/html/XAItest.html

Bioinformatics, 2025, **41(4)**, btaf168
https://doi.org/10.1093/bioinformatics/btaf168
Advance Access Publication Date: 15 April 2025
Original Paper

OXFORD

Gene expression

adverSCarial: assessing the vulnerability of single-cell RNA-sequencing classifiers to adversarial attacks

Ghislain Fievet[1], Julien Broséus[1,2], David Meyre[1,3], Sébastien Hergalant[1,*]

[1]INSERM U1256, Nutrition, Genetics, and Environmental Risk Exposure (NGERE), University of Lorraine, Nancy, 54500, France
[2]Department of Biological Hematology, Laboratory Center, University Hospital of Nancy, Nancy, 54500, France
[3]Department of Molecular Medicine, Division of Biochemistry, Molecular Biology, and Nutrition, University Hospital of Nancy, Nancy, 54500, France

*Corresponding author. INSERM U1256, NGERE lab, Biology-Medicine-Health Campus, 9 avenue de la forêt de Haye, 54500, Vandoeuvre-lès-Nancy, France. E-mail: sebastien.hergalant@inserm.fr.

Associate Editor: Anthony Mathelier

Abstract

Motivation: Several machine learning (ML) algorithms dedicated to the detection of healthy and diseased cell types from single-cell RNA sequencing (scRNA-seq) data have been proposed for biomedical purposes. This raises concerns about their vulnerability to adversarial attacks, exploiting threats causing malicious alterations of the classifiers' output with defective and well-crafted input.

Results: With adverSCarial, adversarial attacks of single-cell transcriptomic data can easily be simulated in a range of ways, from expanded but undetectable modifications to aggressive and targeted ones, enabling vulnerability assessment of scRNA-seq classifiers to variations of gene expression, whether technical, biological, or intentional. We exemplify the usefulness and performance with a panel of attack modes proposed in adverSCarial by assessing the robustness of five scRNA-seq classifiers, each belonging to a distinct class of ML algorithm, and explore the potential unlocked by exposing their inner workings and sensitivities on four different datasets. These analyses can guide the development of more reliable models, with improved interpretability, usable in biomedical research and future clinical applications.

Availability and implementation: *adverSCarial* is a freely available R package accessible from *Bioconductor*: https://bioconductor.org/packages/adverSCarial/ or https://doi.org/10.18129/B9.bioc.adverSCarial. A development version is available at https://github.com/GhislainFievet/adverSCarial.

1 Introduction

Adversarial attacks represent a significant threat to machine learning (ML) tools (Goodfellow *et al.* 2015). These attacks have been investigated in various medical data fields, such as X-rays (Zhou *et al.* 2021), electrocardiograms (Han *et al.* 2020), and tissue slide analyses (Ghaffari Laleh *et al.* 2022). On single-cell-transcriptomic-data, their potential impact has been studied on white-box gradient-based ML algorithms (Sadria *et al.* 2023) but remains under-evaluated for other methods. As scRNA-seq cell type classifiers hold high potential for routine clinical applications, understanding their vulnerabilities to such attacks is increasingly popular (González-Silva *et al.* 2020, Su *et al.* 2022). Fast Gradient Sign Method (FGSM) and Projected Gradient Descent (PGD) are the two adversarial white-box methods that have been applied on scRNA-seq data (Sadria *et al.* 2023). FGSM is a single-step loss-minimizing function which uses the sign of the existing gradient with respect to the input data to create perturbations that push predictions away from the originally predicted class (Goodfellow *et al.* 2015). The size of the step, or attack level, is set by the parameter ε:

$$x_{adv} = x + \varepsilon\, sign\left(\nabla_x L\left(f(x), y\right)\right) \quad (1)$$

PGD iteratively applies small perturbations in the gradient of the loss' direction, with respect to the input data. It is followed by a projection step to ensure the perturbation remains in the same range (Madry *et al.* 2019). Here the step size is set with each iteration:

$$x_{N+1}^{adv} = Proj_{x,\varepsilon}\left\{x_N^{adv} + \varepsilon\, sign\left(\nabla_x J\left(x_N^{adv}, y_{true}\right)\right)\right\} \quad (2)$$

While these approaches target white-box gradient-based methods such as deep learning models, most scRNA-seq classifiers rely on different algorithms (Zhao *et al.* 2020), such as random forest (Tan and Cahan 2019), support vector machines (Nguyen and Griss 2022), XGBoost (Lieberman *et al.* 2018), and statistical models (Zhang *et al.* 2019): this singularity urges for the development of new, specialized strategies. Given the time-intensive nature of cell classification processes, these novel attack modes should be computationally cheap. Beyond the immediate security concerns, they should also be able to provide help in interpreting the generated models and offer insights into their decision-making layers.

adverSCarial features specific scRNA-seq adversarial attack algorithms: two of these attacks cause cell misclassifications

Received: 8 February 2024; **Revised:** 1 April 2025; **Editorial Decision:** 4 April 2025; **Accepted:** 13 April 2025
© The Author(s) 2025. Published by Oxford University Press.
This is an Open Access article distributed under the terms of the Creative Commons Attribution License (https://creativecommons.org/licenses/by/4.0/), which permits unrestricted reuse, distribution, and reproduction in any medium, provided the original work is properly cited.

Downloaded from https://academic.oup.com/bioinformatics/article/41/4/btaf168/8114001 by guest on 17 September 2025

with switching on/off unique genes or with imperceptibly modifying several genes, another one modifies large groups of genes without altering cell labeling, and another one implements a novel gradient descent algorithm adapted to cell clusters and black-box models. The full set of functions enables the investigation of single-cell transcriptomic classifiers under a dual security/transparency scrutiny, within a development frame that is both time- and resource-efficient. Here we exemplify the use, usefulness, and performance of *adverSCarial* by assessing the robustness of five scRNA-seq classifiers, each belonging to a distinct class of ML algorithm: *scType* (marker-based) (Ianevski *et al.* 2022), *CHETAH* (hierarchical clustering) (de Kanter *et al.* 2019), *scAnnotatR* [support vector machine (SVM)] (Nguyen and Griss 2022), *scRF* [random forests (RF)] (Breiman 2001), and *scMLP* [multilayer perceptron (MLP)] neural networks (Almeida 2020).

2 Materials and methods

2.1 Datasets

Four distinct scRNA-seq datasets were used in this study:

- PBMC3k is commonly used in benchmarks and is composed of 2700 peripheral blood mononuclear cells plus a few residual platelets and dendritic cells from a healthy human donor, already filtered for poor quality cells (scRNA expression dataset by Cell Ranger v1.1.0, Illumina NextSeq 500, 10x Genomics, https://cf10xgenomics.com/samples/cell/pbmc3k/pbmc3k_filtered_gene_bc_matrices.tar.gz). PBMC3k contains cells with distinct or with similar transcriptomic profiles, such as B cells and CD4 T cells, or classical (CD14+) and non-classical (FCGR3A+) monocytes, respectively.
- AXILLA10k includes 10 623 annotated cells from breast cancer metastatic core biopsies sampled within the axillary lymph node of a single patient (Klughammer *et al.* 2024) (https://cellxgene.cziscience.com/e/12c868c6-94df-48b5-acf2-b82f6aa14074.cxg/). AXILLA10k contains T cells, blood vessel endothelial cells, blood vessel smooth muscle cells, fibroblasts, macrophages, and malignant cells.
- LIVER10k includes 10 016 annotated cells from breast cancer metastatic core biopsies sampled within the liver of a single patient (Klughammer *et al.* 2024) (https://cellxgene.cziscience.com/e/cd6398a9-c0af-4467-9091-c536866535bd.cxg/). LIVER10k contains blood vessel smooth muscle cells, endothelial cells, macrophages, malignant cells, mature NK T cells, and monocytes. Underrepresented cell types (7 blood vessel endothelial cells and 9 fibroblasts) were removed.
- KIDNEY10k includes 10 790 annotated cells from fetal nephrons of 6 different donors (Stewart *et al.* 2019) (https://cellxgene.cziscience.com/e/08073b32-d389-41f4-a4fd-616de76915ab.cxg/). KIDNEY10k contains proximal tubule epithelial cells, kidney cells, kidney epithelial cells, loop of Henle cells, mesenchymal cells and podocytes.

Depending on the scRNA-seq model used, multiple datasets were prepared from the aforementioned sources following the procedure provided with *Seurat* (v5.0.3), which includes a standardized pipeline for further filtering, scaling, clustering and annotating the data (Hao *et al.* 2024). These quality control and preprocessing steps were distinct for each scRNA-seq dataset and strictly depended on the thresholds applied by the original authors in their inclusion/exclusion protocols. Specifically, cells with abnormally low or abnormally high numbers of genes were excluded (<200 and >2500 for PBMC3k, <500 for KIDNEY10k, <8000 for LIVER10k, and AXILLA10k), as well as cells with a high proportion of mitochondrial gene expression (>5% for PBMC3k, >20% for KIDNEY10k, >50% for LIVER10k and AXILLA10k). Raw data were log-normalized and scaled. Variable genes were selected with the *FindVariableGenes* function in *Seurat* for PBMC3k and KIDNEY10k (Stewart *et al.* 2019, Hao *et al.* 2024), or following the *Scanpy* (v1.7.2) workflow for AXILLA10k and LIVER10k (Klughammer *et al.* 2024), as detailed in the original publications. PCA and UMAP were then applied as dimensional reduction algorithms.

- *Undivided sets* contain all annotated cells.
- *Training sets* contain half of the cells, proportionally representing each of the cell types.
- *Test/validation sets* contain about half of the cells not overlapping with the training set cells and proportionally representing each of the cell types.
- *Balanced training sets* are alternatives for training sets used with RF and MLP algorithms and comprise 100 randomly selected cells per cell type. When the initial number of cells was <100, a bootstrap sampling on gene expression values within each relevant cell type generated additional ones.

PBMC3k was annotated by clustering the cells with the *FindClusters* function (a KNN graph-based approach), then for each cluster, each cell type was identified using a set of marker genes. For AXILLA10k, LIVER10k, and KIDNEY10k, cell annotations were provided directly with the datasets.

2.2 scRNA-seq classifier preparation

For *scType*, *CHETAH* and *scAnnotatR*, we prepared and annotated the datasets according to the workflows provided with each software documentation (manual, tutorial and readme available in Bioconductor or github repositories).

- *scType*: The standard approach requires the load of a database with predetermined positive and negative marker genes for each cell type. For each cell, *scType* returns a vector of scores that assigns the most frequently occurring cell type to the computed cell clusters. We first used *scType* for an overview of *adverSCarial* functioning with the PBMC3k *undivided set*. For further evaluation and comparison with the other classifiers, *training* and *test sets* were used.
- *CHETAH*: The default workflow implies manipulations of *SingleCellExperiment* (Amezquita *et al.* 2020) objects. Within *adverSCarial*, the *CHETAHClassifier* function inputs two *SingleCellExperiment* objects, one for annotating the *training set* as reference, and the other one for the *test set*. *CHETAH* confidence threshold parameter (*tresh*) was set to 0.01. For each cell, this classifier returns a predicted cell type and a score.
- *scAnnotatR*: The tool constructs an intermediate binary classifier based on an SVM algorithm for every cell type. Each of these intermediate models predicts the likelihood of a cell to belong to its respective type. The predictions are then integrated into a single model that outputs a vector of scores representing the probability of each cell type for each individual cell. The SVM classification

Downloaded from https://academic.oup.com/bioinformatics/article/41/4/btaf168/8114001 by guest on 17 September 2025

necessitates a predefined list of genes to discriminate efficiently between cell clusters. Within *adverSCarial*, the *getSignGenes* function ranks the most significant genes between every 1 versus 1 cluster combination. Here we returned the top 20 of this ranked list as input genes for *scAnnotatR*.

scRF and *scMLP* were specifically implemented in the present work for the purpose of *adverSCarial* analysis. We built the models according to RF and MLP architectures, respectively, and trained them first with *balanced training sets*:

- *scRF*: This RF-based classifier was implemented with the *randomForest* package (version 4.7.1.1). The model was built with default parameters: the number of trees (*ntree*) was set to 500, and the number of variables randomly selected at each subdivision (*mtry*) was calculated as the square root of the number of predictor variables, which in PBMC3k was 117, in AXILLA10k was 159, in KIDNEY10k was 181, and in LIVER10k was 160. The response variable y was one-hot encoded to represent the cell types as a binary matrix, suitable for multinomial classification.
- *scMLP*: This MLP-based classifier was built with the R *Keras* library (version 2.15.0). The model architecture consisted of five fully connected layers. The first layer included 128 units with ReLU activation, followed by a dropout layer at a rate of 0.5 to prevent overfitting, then a layer of 64 units with ReLU activation, followed by another dropout layer at a rate of 0.5. The output layer consisted of units equal to the number of unique cell types in the dataset, using *softmax* activation function to normalize the output for multinomial classification over the cell types. The categorical *cross-entropy* loss function was used to measure the performance of the model. The adaptive moment estimation (*Adam*) optimizer (Kingma *et al.* 2017) was used for gradient descent computation during model updates. Training was performed with 20 epochs, a batch size of 32, and 20% of the data as validation.

2.3 Visualization

UMAP (Unified Manifold Approximation and Projection) plots were computed and rendered with *Seurat* (v5.0.3) by using the 2000 most variable genes of the dataset along with the genes of interest selected by the adversarial attacks (e.g. with *scType*, 2000 + 27 genes for *max-change/perc99*). Heatmaps were generated with *ComplexHeatmap* (v2.17) (Gu 2022) after hierarchical clustering on median-centered data with Euclidean distance and complete linkage. Each time, the clustering was performed on the 100 most discriminant genes (obtained with the *getSignGenes* function) along with the genes of interest selected by the adversarial attacks (e.g. with *scType*, 100 + the 4 genes *ICAM1*, *RPS27*, *ACTB* and *CAMKK2* for *single-gene/perc99*). Otherwise, graphics were generated with ggplot2 (v3.4.4).

3 Results

3.1 Implementation and algorithm

To be compatible with *adverSCarial*, classifiers should be set up as a function that inputs a gene × cell expression matrix, a categorical vector/list labeling each cell with the examined classifier, and the target label for the group of cells (cluster) under investigation (Fig. 1). The function should return the predicted cluster classification with its likelihood score, and a cell likelihood score × cell type matrix along with the predicted classification for each individual cell. This step ensures input/output encapsulation of any analysis mode, whether cell-wise or cluster-wise, into *adverSCarial*. Examples of preparation with *scType* (Ianevski *et al.* 2022) and *CHETAH*

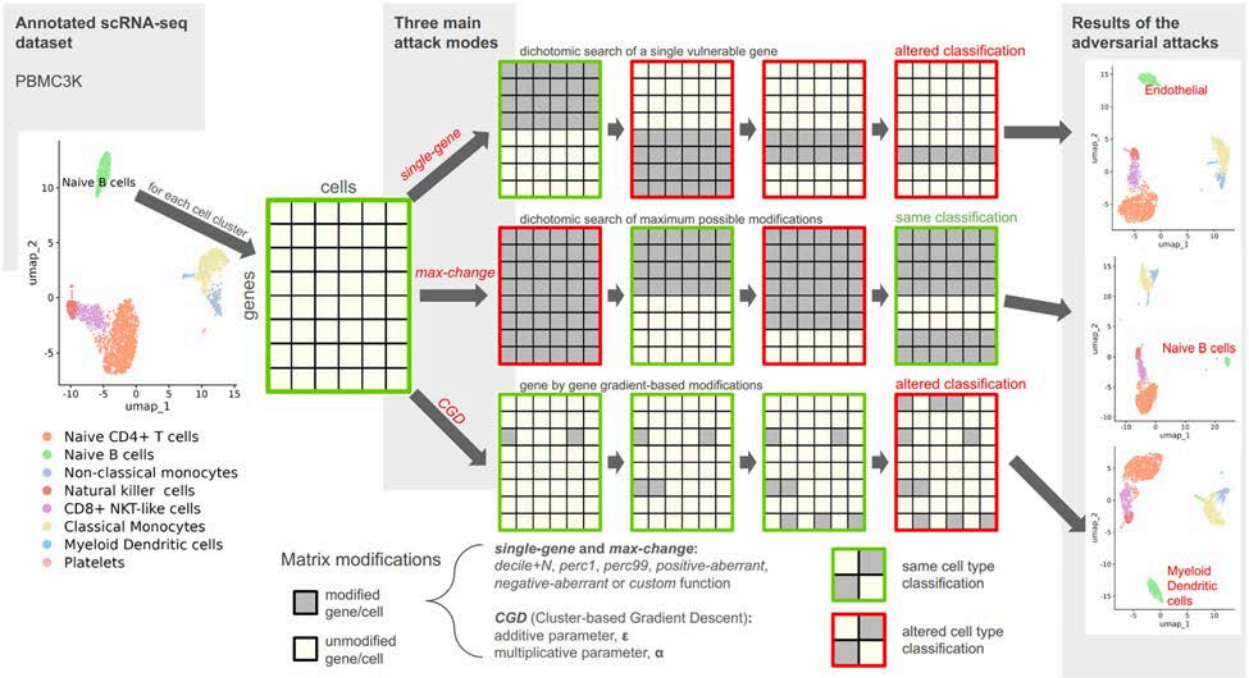

Figure 1. adverSCarial feature overview. Running of the *single-gene, max-change* and *CGD* (Cluster-based Gradient Descent) attack algorithms. UMAP representations of the *scType*-annotated PBMC3k dataset before and after *single-gene, max-change* or *CGD* attacks on the Naive B-cell cluster.

Downloaded from https://academic.oup.com/bioinformatics/article/41/4/btaf168/8114001 by guest on 17 September 2025

(de Kanter *et al.* 2019) as respective classifiers are provided in the Supplementary Material (Supplementary Data S1).

3.1.1 Types of modifications

The different attack functions provided by the package require another critical parameter defining the gene modification method to be applied. Several gene expression modulations are available, reflecting a variety of biological, technical, or malicious events. Two modifications, ***perc1*** and ***perc99***, simulate an inactivation and activation by replacing all cell expression values for a given gene by the 1st and the 99th percentile, respectively. Genes or groups of genes can also be down- or upregulated in a synchronized manner using the ***decile-N*** or the ***decile+N*** modifications, which simulate a gradual decrease or increase by deciles of expression, on a scale N, with $N = 1$ to 10. The ***random*** modification introduces uniform random noise between the minimum and the maximum expression values and simulates either real-world variations or errors in gene expression. With the ***custom*** function, users can also implement and apply any custom modification. For example, one can simulate the dropout effect frequently observed in single-cell transcriptomics and use the function to generate an attack (Supplementary Data S2). Last, we simulate potential technical errors or data processing miscalculations by introducing aberrant values. ***Positive-*** and ***negative-aberrant*** parameters create extreme outlier values by modifying gene expression levels with plus and minus 105 times the dataset maximum, respectively.

3.1.2 scRNA-seq attack modes

- *Single-gene*: The *single-gene* attack produces cell misclassification and mislabeling after altering the expression of one gene only. The *advSingleGene* function is designed to identify every *single-gene* attack on a cell cluster given a modification type. On a computer with average technical specifications, most classifiers require >10 s to cluster a group of cells. As standard scRNA-seq datasets contain approximately 20 000 genes, a brute-force approach testing each gene individually would run for 55 h and be computationally intensive. Instead, the function uses a two-step strategy that first partitions the gene list into 100 bins (default settings), then applies a recursive dichotomous search into each bin to identify attackable genes, and finally isolates a single gene responsible for the misclassification (Fig. 1, Supplementary Data S3). The search for an item within an n-item list can be executed in *log2 (n)* time: if the dataset contained a *single-gene* attack, it would necessitate 100 + *log2(20 000)* = 114 runs and <20 min of computational time.
- *Max-change*: This attack (*advMaxChange* function) tries to modify as many genes as possible without affecting the cell clustering output and returns the complementary gene set: the genes that cannot be touched lest a misclassification occurs. Again, a brute force approach would quickly reach a complexity barrier [for an n-gene dataset: *O(2n)*]. Instead, the method alters the cell cluster on all genes and then splits the genes in two subsets when the classification changes. If further subset modification leaves the classification unchanged, it is combined with a previous classification-unchanging gene-set picked out in earlier stages, otherwise the subdivision continues. This process runs recursively until the largest unchanging set is found. The final complementary gene list consisting of the untouchable genes is referred to as the cell cluster signature (Fig. 1, Supplementary Data S4).
- *Cluster-based Gradient Descent (CGD): CGD* is implemented in the *advCGD* function and is a new gradient-derived adversarial perturbation method devised for the purpose of *adverSCarial* in-depth testing of black-box scRNA-seq classifiers. This iterative, cluster-based modification process (one gene by step, whole cluster perturbation instead of cells) is guided by an approximated gradient of the cell type likelihood with respect to a gene j expression. With the entire cell × gene matrix defined as X, the subset matrix corresponding to a cluster of cells m as $X_{clust\ m}$, and the gene expression vector of a cell i as x_i:

$$X = (x_i)_{i \le \|Cells\|}, \text{ and}$$
$$x_i = (x_{ij})_{j \le \|Genes\|},$$
$$\hat{y}_{x_i} = (score_k)_{k \in CellTypes} \text{ classifies a cell i, and}$$
$$\hat{y}_{X_{clust\,m}} = \sum_{i \in clust\,m} e_{k'} \text{ classifies a cluster of cells } m \text{ with}$$
$$k' := \underset{k}{\operatorname{argmax}} \left(\hat{y}_{x_i}\right)_k.$$

Initially, the method infers the gradient of $\hat{y}$ for each cell of $X_{clust\ m}$ with respect to a gene j' expression, by finite differences. The magnitude of this difference is adjusted by the two parameters α and ε (Fig. 1):

$$\nabla^{fd}_{j'} \hat{y}_{x_i} = \hat{y}_{f_{j'}(x_i,\alpha,\varepsilon)} - \hat{y}_{x_i}, \text{ with}$$
$$f_{j'}(x_i,\alpha,\varepsilon) = x_{ij}\left(1-\delta_{jj'}\right) + \left(\left(1+\alpha.sign(x_{ij})\right)x_{ij} + \varepsilon\right)\delta_{jj'} \quad (3)$$

From both classifier results (by cell and by cluster), we define $ctype1_{x_i}$ and $ctype2_{x_i}$, the most and second most likely cell types of xi, and $ctype1_{X_{clust\,m}}$ and $ctype2_{X_{clust\,m}}$, the most and second most likely cell types of $X_{clust\ m}$, respectively. *CGD* follows the inferred gradient to lower the gap between $ctype1_{X_{clust\,m}}$ and $ctype2_{X_{clust\,m}}$ scores by modifying the genes on each cell of m one by one, until $ctype2_{X_{clust\,m}}$ becomes the new most likely cell type predicted by the classifier. The adversarial matrix $X^{adv}_{clust\,m}$ is returned, with the step size, and thus the attack level, set by the same multiplicative and additive parameters α and ε, respectively:

$$slope = \left(\nabla^{fd}_{j'} \hat{y}_{x_i}\right)_{ctype2_{X^n_{clust\,m}}} - \left(\nabla^{fd}_{j'} \hat{y}_{x_i}\right)_{ctype1_{X^n_{clust\,m}}},$$
$$\left(x_i^{n+1}\right)_j = x_{ij}\left(1-\delta_{jj'}\right) + \left(\left(1+\alpha.sign(x_{ij})sign(slope)\right)x_{ij} + \varepsilon.sign(slope)\right)\delta_{jj'},$$
$$X^{adv}_{clust\,m} = \left(x_i^{n+1}\right)_{i \in clust\,m} \quad (4)$$

To reduce the number of iterations required for the adversarial attack, and because each step targets a different gene, we prioritize genes based on their likelihood of altering the classifier's output. The function *getSignGenes* can be used to rank the genes according to their inter-cluster differential t-statistics while ensuring their differentiating between all possible pairs of clusters (Supplementary Data S5). To avoid unnecessary computations, $x_i^{n+1} = x_i^n$ is set when $ctype1_{x_i^n} = ctype2_{X^n_{clust\,m}}$ or $slope = 0$ (the applied modification did not change the output of the classifier).

Downloaded from https://academic.oup.com/bioinformatics/article/41/4/btaf168/8114001 by guest on 17 September 2025

- *Random-walk*: This attack (*advRandWalkMinChange* function) applies various random modifications on a subset of genes and then progressively reduces this initial combination of genes × modifications while reclassification occurs. With this function, classifiers can be mined arbitrarily for fortuitous vulnerabilities and working attacks can be quickly identified.
- *Brute-force*: Last, the *brute/grid* attack (*advGridMinChange* function) exhausts every possible attack mode × modification type for a reduced number of starting genes. The combinatorial complexity of such an approach limits the number of genes to be tested to <10.

3.2 Showcase example of *adverSCarial* use

We performed an overall survey of the scRNA-seq marker-based classifier *scType* (Ianevski *et al.* 2022), applied on the PBMC3k *undivided set* (Supplementary Data S6). Using *adverSCarial* and the *single-gene* attack, we investigated whether modifying a single marker could alter the classifier output and to what extent. The *perc1* modification emerged as the least impactful with only a few potential attacks by cell cluster (mean 1, range 0–5). In contrast, *positive* and *negative-aberrant* modifications produced numerous misclassification events (mean 151 and 30.8, range 144–158 and 26–34, respectively), while *perc99* induced a moderate number of attacks (mean 11.9, range 0–40). The overall vulnerability to *single-gene* attacks was equally variable depending on the cell type, with platelet cells demonstrating the highest resistance and no weaknesses to the *perc1* or *perc99* modifications. Figure 1 (on the top-right corner) showcases the misclassification of the naive B cell cluster into endothelial cells after switching on *ICAM1* with *perc99* (more details in Supplementary Data S6). With the *max-change* attack, we further explored the minimum signatures of unalterable genes defining a correct classification with *scType*. Here the naive B cell cluster remained unaltered even after switching on 99.5% of all genes. The remaining 9 candidates thus constituted a critical gene set defining naive B cells for this classifier. Interestingly, there was an overall correlation between the reported signature lengths and the vulnerability to *single-gene* attacks with *scType*, after exclusion of the *random* modification method which by nature cannot be reproduced between analyses (Spearman's rho = 0.66).

3.3 Evaluation of different types of scRNA-seq classifiers

We next conducted a comprehensive examination of the robustness of five distinct ML models dedicated to classifying scRNA-seq data. Three of these classifiers were sourced from existing literature: *scType* (marker-based model) (Ianevski *et al.* 2022), *CHETAH* (hierarchical classification) (de Kanter *et al.* 2019), and *scAnnotatR* (SVM) (Nguyen and Griss 2022). For the purpose of this evaluation, we implemented and trained two new classifiers, one based on random forests, hereby called *scRF*, and one deep learning neural network model based on a multilayer perceptron, called *scMLP*. Additionally, we tested four different datasets, PBMC3k, LIVER10k, AXILLA10k and KIDNEY10k, each divided into *training* and *test sets* (see Section 2).

3.3.1 Survey of vulnerability to adversarial attacks

Single-gene attacks on the five classifiers revealed varying susceptibilities depending on the class of algorithm tested, ranging from *scType* often being the most vulnerable, to *scRF* being entirely impervious to all modification types (Fig. 2A). With this attack mode, *decile + 5* was the least effective modification and solely affected *scType*, with seven possible attacks (Supplementary Data S7). *Perc1* and *perc99* modifications were effective on *scType*, *CHETAH*, and *scAnnotatR* with 185, 2, and 6 successful attacks, respectively (Supplementary Data S8), while *positive-* and *negative-aberrant* modifications yielded >1500 results for *scType*, 4 for *CHETAH*, 80 for *scAnnotatR*, and about 200 000 for *scMLP*. Concerning *max-change* attacks, the same trends were observed for all classifiers, from *decile + 5* producing the shortest signature lengths to *positive-* and *negative-aberrant* modifications yielding the longest ones (Fig. 2B). Dependent of a classifier/modification combination, situations arose for every classifier where all genes of all cells from a cluster could be modified without changing the prediction (signature length = 0), including the extreme *positive-* or *negative-aberrant* modifications. Overall, relatively to each other and given the extent of the presented modifications with all classifiers, *decile + 5* was a low-impacting modification while *perc1* and *perc99* were medium-impacting, and *positive-* and *negative-aberrant* were high-impacting for both attack modes (Fig. 2C). Both classifiers *CHETAH* and *scRF* showed few responses to medium or high modifications, indicating a plateau in sensitivity due to a natural characteristic of tree-based algorithms: once an input reaches the defined threshold of a tree node, further increases have no additional impact on the outcome (Tan *et al.* 2018). In contrast, *scMLP* was unresponsive below medium modifications and very vulnerable beyond with thousands of possible attacks: this effect stems from the neural network's use of multiplicative coefficients, which amplify sensitivity to outliers (Tan *et al.* 2018).

3.3.2 Detectability of the modifications

An essential aspect of adversarial attacks is the human perceptibility of the introduced changes (Goodfellow *et al.* 2015). Confirming the results obtained with *scType* on the PBMC3k *undivided set*, a *single-gene* and *max-change/perc99* attacks on B cells from the *test set* revealed critical features for the classification, *ICAM1* and a 27-gene signature, respectively, while maintaining a comparable cell cluster topology on UMAP (Fig. 3A). Of note, clustering heatmaps enhance visual detectability of these features along with the applied modifications by focusing on the 100 most variable genes discriminating between each cell cluster instead of displaying all genes (see Section 2). On other datasets, *single-gene* attacks produced variably discernible alterations in the targeted gene's expression but preserved the visual patterns of the cell type (Fig. 3B). Expectedly, *max-change* attacks with *perc99* and *perc1* obliterated the visual patterns of the targeted cell clusters but did not alter the classification (Fig. 3C). For *CHETAH* and *scRF*, the respective residual signatures contained 793 and 4506 genes. Allowing for control over modification intensity using α and ε parameters, the *CGD* algorithm yielded subtle modifications with low values such as $\alpha = \varepsilon = 0.5$ to more visible ones with $\alpha = \varepsilon = 1$. However, with low values the attacks had to impact many genes to fool the classifier, whereas far less genes were needed with higher values. For example, *CGD* slightly adjusted 15 genes to alter *scAnnotatR* classification of the kidney epithelial cell cluster, but only 2 genes had to be ostensibly modified for the same result (Fig. 3D). Depending on the classifier,

Downloaded from https://academic.oup.com/bioinformatics/article/41/4/btaf168/8114001 by guest on 17 September 2025

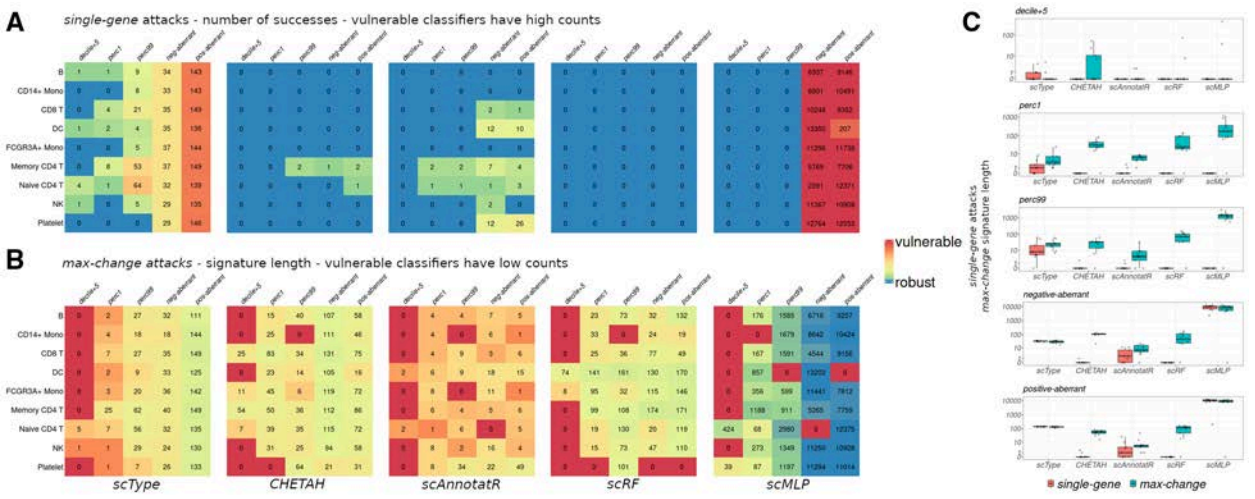

Figure 2. Vulnerabilities of five scRNA-seq classifiers to *single-gene* and *max-change* adversarial attacks. After a training step on the PBMC3k training set, five modification types are sequentially applied on the nine cell types of the test set. Heatmaps reporting (A) the number of successful *single-gene* attacks (lower is better), and (B) the signature length of *max-change* attacks (higher is better) by classifier, cell type and modification type. For *single-gene*, counts > 0 point out vulnerabilities, while classifiers are considered robust to *max-change* attacks with high counts. (C) Boxplots summarizing the impact of each modification (sum over each cell type) for both attack modes, by classifier. Data are represented in log10 scale.

equal parameter values produced different results, from 1 gene for *scType* to 577 genes for *scMLP* with $\alpha = \varepsilon = 1$ (Fig. 3D and E).

3.3.3 Performance, accuracy, and prediction dynamics

A notable feature of the *CGD* algorithm is that it modifies only some of the genes and stops after the cell cluster's classification has switched. Thus, only the genes influencing the output are modified, which can lead to a discrepancy between the number of genes tested and those modified. The ratio of tested genes/modified genes differed among the eligible classifiers for *CGD* adversarial attacks (Fig. 4A). For instance, on the PBMC3k FCGR3A+ Mono cell cluster, an attack of *scType* with $\alpha = \varepsilon = 1$ required testing 18 genes but modified only 4 genes (0.22-fold). For *scAnnotatR*, there was 23 tested/9 modified genes (0.39-fold) with the same parameters. With both classifiers, this gap can be explained by the underlying algorithms' architectures, one being marker-based and the other SVM-based: there is a selection of input variables for classification. Conversely, *scRF* tested 374 genes and modified 290 of them (0.78-fold) while *scMLP* tested 296 genes and modified them all. These two models inherently consider nearly all input variables, even if their contribution to the prediction is marginal. Between these two groups of classifiers (selective versus unselective in their inputs), responses were also different when testing the influence of the parameters α and ε independently (Fig. 4B). Each classifier could be attacked solely with the ε additive part of the algorithm. However, with a maximum of 500 tested genes, *scMLP* and *scRF* could not be attacked by exclusively using the α multiplicative part. To enable accuracy and prediction dynamics' analyses as functions of the number of tested genes, we implemented an augmented version of *CGD* that continues the iterative process until all provided genes are tested, even after the cell cluster predictions are deviated from the original cell type. As compared to *scRF* and *scMLP*, *scType* and *scAnnotatR* exhibited disturbances more rapidly and switched predictions with fewer genes. For each classifier, higher values of α and ε were associated with greater disturbances (Fig. 4C). ROC curves varied depending on the classifier and the targeted cluster. For *scAnnotatR*, performance was significantly impacted by attacks with standard parameters $\alpha = \varepsilon = 1$ and 500 tested genes. In contrast, *scRF* showed only minor performance changes even with high parameter values such as $\alpha = \varepsilon = 10$ and 4000 tested genes (Fig. 4D). In the same conditions, the F1-score analysis revealed that the classifiers were primarily affected when a high probability threshold was set for predicting a positive (Fig. 4E).

3.4 Which attack/modification for my classifier/data?

Upon selecting an attack method, several factors need to be considered to evaluate the success rate and computation time required with *adverSCarial* (Table 1):

- *Computing resources*: Each step of the three main attack modes involves modifying a matrix and classifying the result to evaluate their impact. The modifications applied to one gene take an average of 9 s with an Intel(R) Core (TM) i7-8550U (base clock speed: 1.80 GHz) with 8 Gb of DDR4 RAM and 1 s with an Intel(R) Core(TM) i5-7400 (base clock speed: 3.00GHz) with 64 Gb of DDR4 RAM (Supplementary Data S9). The computation time also depends on the classifier, averaging on 2 min for *scMLP* to 9 min for *scAnnotatR* with 8 Gb RAM, and 1 min for *scMLP* to 5 min for *scAnnotatR* with 64 Gb RAM (Supplementary Data S10).
- *Single-gene*: With no attack possible, the algorithm evaluates each of the initial 100 gene bin once and returns an empty result. However, with multiple possible attacks, a binary search is initiated within each potential bin, increasing the computational time (Supplementary Data S11). For example, an unsuccessful attack on an AXILLA10k cluster with *scType* consumed 2 h of runtime as compared to 4 h for a successful one on a KIDNEY10k cluster containing less cells (Table 1).

Downloaded from https://academic.oup.com/bioinformatics/article/41/4/btaf168/8114001 by guest on 17 September 2025

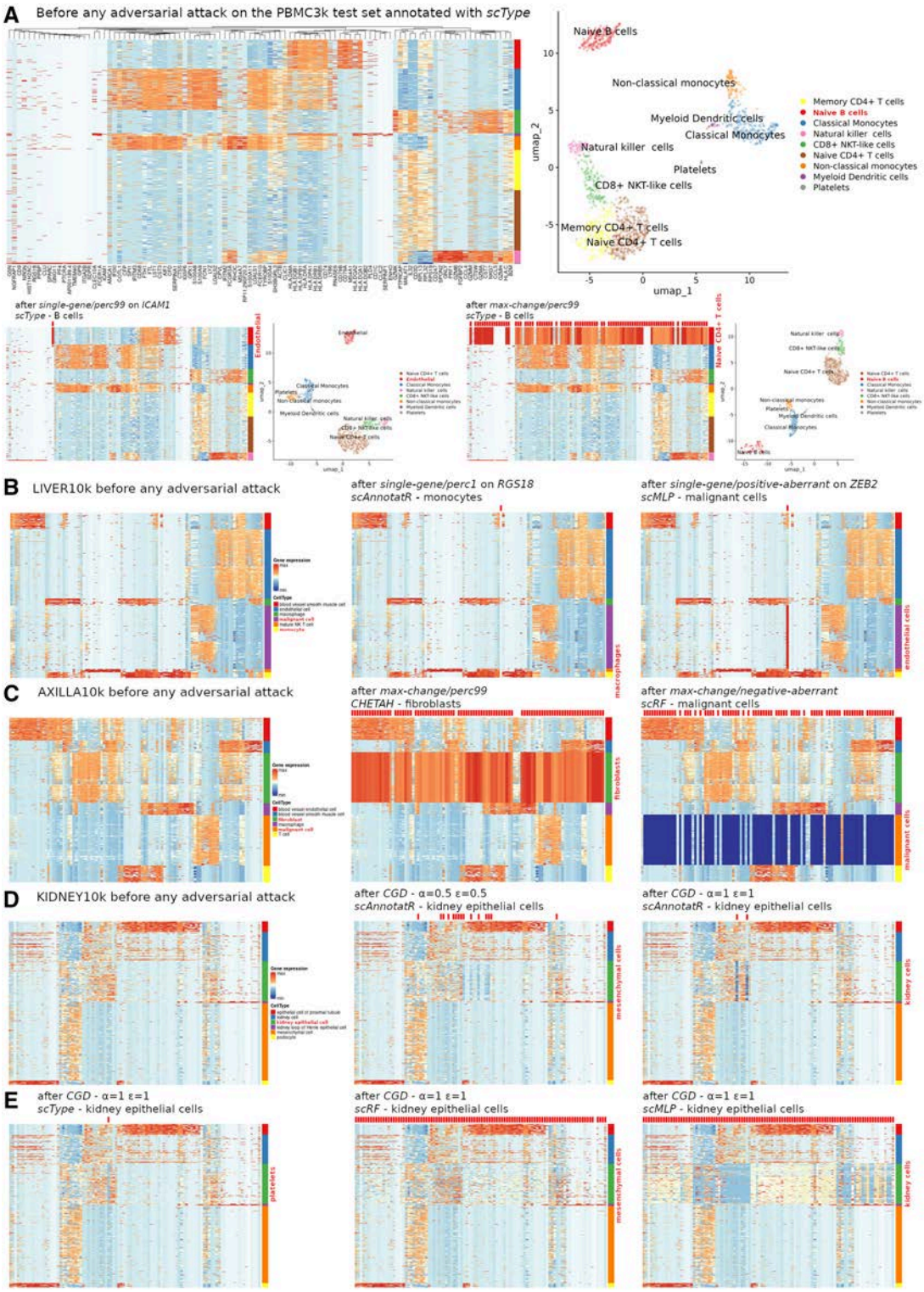

Figure 3. Perceptibility of the changes induced by various adversarial attacks. Classifiers are trained and tested with training and test sets, respectively. On top of each heatmap, applied modifications are marked with discrete ticks, whether visible or not. (A) Hierarchical clustering heatmaps and UMAP representations after classification and annotation with *scType*, before (on top) and after a *single-gene* attack (bottom-left) or a *max-change* attack (bottom-right) on PBMC3k B cells. (B) Hierarchical clustering heatmaps of the most discriminating genes before and after various *single-gene* attacks of different scRNA-seq classifiers (dataset: LIVER10k). (C) Idem after various max-change attacks (dataset: AXILLA10k). (D) Idem after *CGD* attacks of incrementing magnitudes (parameters α and ε) on scAnnotatR (KIDNEY10k). (E) Idem after *CGD* attacks of comparable levels on different classifiers (KIDNEY10k).

- *Max-change*: When the recursive process involves low-impacting modifications that are less likely to alter the classification, a substantial number of genes can be integrated into the search bin early on, sometimes encompassing the entire set (as indicated by a signature length of 0). In contrast, high-impacting modifications require

Downloaded from https://academic.oup.com/bioinformatics/article/41/4/btaf168/8114001 by guest on 17 September 2025

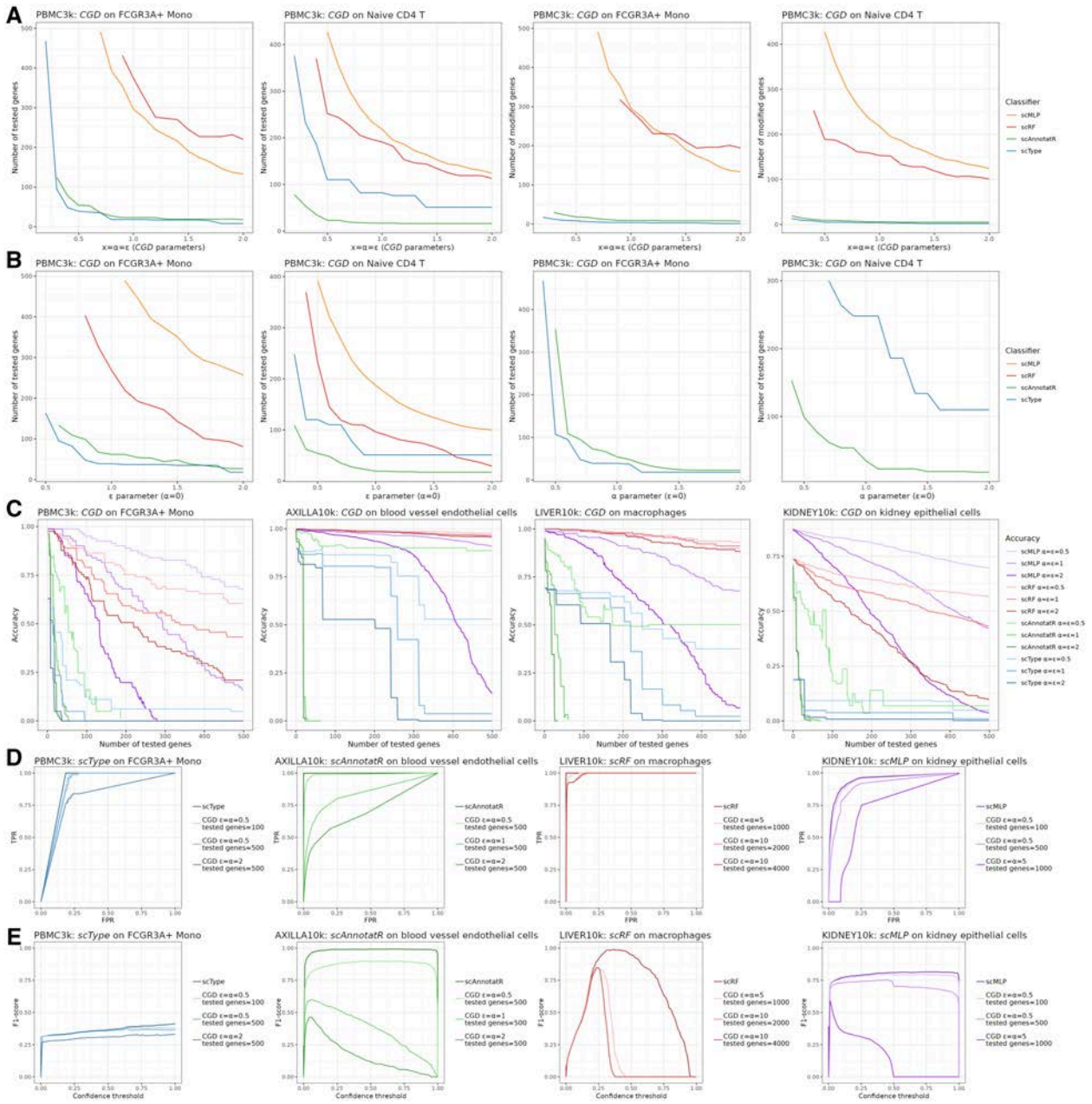

Figure 4. Performance and evaluation of the cluster-based gradient descent (*CGD*) algorithm. *CGD* has been applied to the four eligible classifiers *scType*, *scAnnotatR*, *scRF*, and *scMLP*. (A) Number of genes to be tested and/or modified for deviating the classification of the cell cluster, according to equal parameters α and ε. (B) Idem according to ε when $\alpha = 0$ (left), and to α when $\varepsilon = 0$ (right). (C) Accuracy of the prediction according to the number of tested genes, with different sets of values for ε and α, on different datasets. (D) ROC curves and (E) corresponding F1-scores for different values of ε, α and the number of tested genes, on different datasets. FPR: false positive rate, TPR: true positive rate.

deeper recursive searches to identify gene sets associated with no classification change and require >2 h of runtime if successful. For faster computation, the *maxSplitSize* option can be used to stop the recursion once a specified gene bin size bas been reached. Setting higher values for *maxSplitSize* reduces the computational resources but increases the size of the identified gene set, resulting in a less specific signature.

- *CDG*: With CGD, computation time primarily depends on α and ε as higher values for these parameters reduce the number of genes that need to be modified to achieve a successful attack. For instance, with *scMLP* and $\alpha = \varepsilon = 0.5$, 3 h of runtime and 1297 modified genes are required, which is contrasted by 7 min of runtime and 42 modified genes with $\alpha = \varepsilon = 10$.

Overview mode: Finally, *adverSCarial* implements tools for the assessment of overall attack susceptibilities in a complete dataset given a classifier (Supplementary Data S12). The user can quickly review both *single-gene* and *max-change* attacks in combination with various modification modes to gain insights on what works best for a specific classifier/dataset pair. Given the time intensive nature of testing every cell cluster across multiple modifications, a less precise yet faster configuration of these attack modes is used. First, recursive subdivisions are halted once the subsets reach 100 genes; second, the *single-gene*

Downloaded from https://academic.oup.com/bioinformatics/article/41/4/btaf168/8114001 by guest on 17 September 2025

Table 1. Examples of *adverSCarial* use on different datasets and classifiers, along with parameters, outcomes, computational resources, and specifications.

Targeted classifier	Targeted dataset				
	PBMC3k	**AXILLA10k**	**KIDNEY10k**	**LIVER10k**	
Adversarial attacks: *single-gene* with *perc1* modifications					
scType	NK	Malignant cells	Kidney epithelial cells	Blood vessel smooth muscle cells	Targeted cluster
	72	1627	1325	505	No. of cells
	0	0	2	1	No. of successful attacks
	4 min	2 h	4 h	3 h	Runtime
			Intel i7, RAM 8 GB		Specifications
Adversarial attacks: *max-change* with *decile* + *5* modifications, *MaxSplitSize* = 1000					
scRF	memory CD4 T	Blood vessel endothelial cells	Podocytes	Malignant cells	Targeted cluster
	172	744	139	1917	No. of cells
	0	0	7183	0	No. of signature length
	3 min	20 min	9 h	40 min	Runtime
			Intel i7, RAM 8 GB		Specifications
Adversarial attacks: *max-change* with *perc99* modifications, *MaxSplitSize* = 50					
CHETAH	memory CD4 T	Blood vessel endothelial cells	Podocytes	Malignant cells	Targeted cluster
	172	744	139	1917	No. of cells
	941	546	448	0	No. of signature length
	2 h	2 h	16 h	3 min	Runtime
			Intel i5, RAM 64 Gb		Specifications
Adversarial attacks: *CGD* with $\alpha = \varepsilon = 0.5$					
scAnnotatR	CD8 T	Blood vessel smooth muscle cells	Mesenchymal cells	Macrophages	Targeted cluster
	139	389	2559	205	No. of cells
	memory CD4 T	Fibroblast	Kidney cell	Malignant cell	New cell type
	10	17	20	28	No. of modified genes
	2 min	30 min	1 h	56 min	Runtime
			Intel i5, RAM 64 Gb		Specifications
scMLP	CD8 T	Blood vessel smooth muscle cells	Mesenchymal cells	Macrophages	Targeted cluster
	139	389	2559	205	No. of cells
	NK	Fibroblast	Kidney epithelial cell	Monocyte	New cell type
	151	2121	1109	1806	No. of modified genes
	4 min	5 h	4 h	4 h	Runtime
			Intel i5, RAM 64 Gb		Specifications
Adversarial attacks: *CGD* with $\alpha = \varepsilon = 10$					
scMLP	CD8 T	Blood vessel smooth muscle cells	Mesenchymal cells	Macrophages	Targeted cluster
	139	389	2559	205	No. of cells
	NK	Fibroblast	Kidney epithelial cell	Monocyte	New cell type
	3	43	55	65	No. of modified genes
	6 s	6 min	11 min	9 min	Runtime
			Intel i5, RAM 64 GB		Specifications

mode begins with fewer initial bins by default (20 subsets). These approximations allow for a quick overview of potentially hazardous modification types and point out cell clusters for any kind of vulnerability. If needed, precise attack runs can next be set up with this information.

4 Discussion

Through this examination of five scRNA-seq classifiers on four different datasets, we unveiled a spectrum of weaknesses to adversarial attacks dependent of the underlying class of ML algorithm and the extent of the applied modulations. For instance, we provide a clear demonstration that outliers and aberrant values easily deviate the tools' predictions and need to be filtered in preprocessing steps as a prerequisite for correct operation. This, however, might not be sufficient for flawless reports in cases of slight technical or even biological biases. Eventually, every classifier failed after a threshold of accumulated modifications was reached: this threshold was one of the most constant characteristics observed during the course of this survey. From substantial changes applied on a few genes only to invisible ones applied on hundreds of genes, susceptibilities arose preferentially with each class of models, but the sum of modifications did not differ drastically. Accuracy analyses, as well as ROC and F1-score curves further revealed differences in classifier behavior following diverse attacks, highlighting general vulnerabilities and the specific ways with which each algorithm respond. Similar to

Downloaded from https://academic.oup.com/bioinformatics/article/41/4/btaf168/8114001 by guest on 17 September 2025

what has been observed in other fields, notably in image recognition, none of the ML models examined here were impervious to discrete and imperceptible changes throughout the information matrix (Goodfellow *et al.* 2015, Han *et al.* 2020, Zhou *et al.* 2021, Ghaffari Laleh *et al.* 2022).

As compared to existing adversarial tools, *adverSCarial* presents the advantage of being compatible with various classifier algorithms, potentially all types of models working with any kind of scRNA-seq data. It offers distinct adversarial attack modes, two that do not require access to the gradient and another one that approximates it, enabling testing on nearly any white-box and black-box ML model. By handling *Seurat* or *SingleCellExperiment* objects directly, by encapsulating the classifier's functioning and parameters, *adverSCarial* ensures minimal data manipulation from the beginning, simplifying the user's experience. Future versions could include parallel processing to reduce computational time and incorporate other adversarial algorithms, such as FGSM and PGD, to enable broader analyses. For example, several versions and improvements of the *CGD* algorithm could be explored. A simpler algorithm disregarding the likelihood of the second most likely cell type would be useful for classic adversarial training tasks, such as improving a classifier's robustness or accuracy. Moreover, *CGD* modifies a gene whenever it exerts an influence, however marginal, on the classifier's outcome. With deep learning models this process might not always be efficient because the predicted outputs are based on nearly every input variable. Introducing a gradient threshold for modifying a gene could enhance efficiency by ensuring that only relevant changes are implemented from the start. Another avenue for improvement lies in the preselection of genes to be modified. In this work, we first and foremost aimed to evaluate a diversity of classifier types, and to gain insights on performance, accuracy and prediction dynamics during attacks with *adverSCarial*. Therefore, we prioritized the genes to be tested by preliminary rankings with differential statistics between all possible cell types. This is not ideal in adversarial attacks where targeting less conspicuous markers might be preferable, e.g. to leverage *CGD*'s potential for marker discovery. To address this limitation, and because testing every single gene with a gradient-based algorithm is resource-consuming, alternative gene preselection strategies could be considered, whether at random, according to the variance observed, or by establishing scores reflecting phenotypic features.

Considering that classic adversarial attack analyses primarily aim to evaluate the robustness of systems against various forms of input variability, whether malicious or not, certain observations still suggested a potential for cell marker differentiation using these methods (Supplementary Data S13). The present survey highlighted several genes during *single-gene* or *max-change* attacks: on B cells, switching *ICAM1* on changed *scType*'s prediction to endothelial cells; on naive CD4 T cells, switching *RPS27* off changed *scAnnotatR*'s prediction to memory CD4 T cells; *CHETAH* and *scMLP* were also fooled by modifications on *ACTB* (from memory to naive CD4 T cells) and *CAMKK2* (from CD8 T to FCGR3A+ Mono cells), respectively. If *RPS27* and *ACTB* were effectively differentially expressed in naive versus memory CD4 T cells (adjusted P-values $= 1.69 \times 10^{-5}$ and 3.06×10^{-9}, respectively), hence discoverable with usual approaches, in the case of *CAMKK2* differential statistics in CD8 T versus FCGR3A+ Mono cells were not significant (adjusted P-value $= 0.21$). As another example, the CD14+ monocyte signature included several statistically significant ribosomal genes, such as RPL13A, RPL21, RPS15A, RPS23, RPS27A, RPS4X, RPS4Y1, and RPS5, and others that were not: RPL29 and RPS11. Thus, among the genes identified through adversarial attacks that do not overlap with those reported by standard explorations discriminating between cell types, such as statistical analyses, dimension reduction and clustering techniques, some candidates could be informative to expose the inner workings of the ML models. When for instance, the modification of a single gene typically should not suffice to change the prediction of a marker-based classifier. Nevertheless, whenever an original signature is uncovered, cross-validations by redundancy should be endeavored and gene overlaps estimated (Cantini *et al.* 2018, Chi *et al.* 2020).

In conclusion, *adverSCarial* introduces novel approaches to pinpoint weaknesses against adversarial attacks and evaluate the robustness of single-cell-transcriptomics classifiers. The diverse functionalities allow flexible modulations in attack modes and modification types, ranging from subtle and virtually undetectable changes to clearly visible and aggressive alterations. On hallmark scRNA-seq datasets and ML tools such as *scType, CHETAH*, *scAnnotatR*, and specifically implemented RF and MLP-based models, it highlighted many vulnerabilities and susceptibility profiles to gene expression modifications, underscoring the remaining challenges left to ensure accurate classification following technical, experimental design, and malicious events. As single-cell transcriptomics continuously expands in many domains, including clinical use and precision medicine, we believe *adverSCarial* can be a useful toolkit for researchers, developers and end-users to survey the vulnerability, address security concerns, and identify the inner decision-making mechanisms of the ML algorithms exploiting scRNA-seq data.

Acknowledgements

The authors would like to thank Prs. Pierre Feugier and Marie-Thérèse Rubio as well as the foundation ID+ Lorraine for their financial and executive involvement into the employment of GF.

Author contributions

Ghislain Fievet (Conceptualization [equal], Formal analysis [lead], Investigation [equal], Methodology [equal], Resources [lead], Software [lead], Writing—original draft [equal], Writing—review & editing [equal]), Julien Broséus (Project administration [supporting], Supervision [supporting], Writing—review & editing [supporting]), David Meyre (Funding acquisition [supporting], Project administration [supporting], Writing—review & editing [supporting]), and Sébastien Hergalant (Conceptualization [equal], Formal analysis [supporting], Funding acquisition [lead], Investigation [equal], Methodology [equal], Project administration [lead], Resources [supporting], Software [supporting], Supervision [lead], Writing—original draft [equal], Writing—review & editing [equal])

Supplementary data

Supplementary data are available at *Bioinformatics* online.

Conflict of interest: None declared.

Downloaded from https://academic.oup.com/bioinformatics/article/41/4/btaf168/8114001 by guest on 17 September 2025

Funding

This work was supported by the Contrat de Plan Etat-Régions—Biologie Santé du site Lorrain; and Inserm U1256 NGERE endowment subsidy.

Data availability

The data and code underlying this article are available in GitHub at https://github.com/GhislainFievet/adverSCarial (main development branch) and https://github.com/GhislainFievet/adverSCarial_article (supplementary code and data) and can be accessed freely.

References

Almeida LB. Multilayer perceptrons. In: *Handbook of Neural Computation*. England: CRC Press, Taylor & Francis group, 2020. pp. C1–2.

Amezquita RA, Lun ATL, Becht E *et al.* Orchestrating single-cell analysis with Bioconductor. *Nat Methods* 2020;**17**:137–45.

Breiman L. Random forests. *Mach Learn* 2001;**45**:5–32.

Cantini L, Calzone L, Martignetti L *et al.* Classification of gene signatures for their information value and functional redundancy. *NPJ Syst Biol Appl* 2018;**4**:2.

Chi X, Sartor MA, Lee S *et al.* Universal concept signature analysis: genome-wide quantification of new biological and pathological functions of genes and pathways. *Brief Bioinf* 2020;**21**:1717–32.

de Kanter JK, Lijnzaad P, Candelli T *et al.* CHETAH: a selective, hierarchical cell type identification method for single-cell RNA sequencing. *Nucleic Acids Res* 2019;**47**:e95.

Goodfellow IJ, Shlens J, Szegedy C. Explaining and harnessing adversarial examples. arXiv:1412.6572v3, 2015, preprint: not peer reviewed.

Ghaffari Laleh N, Truhn D, Veldhuizen GP *et al.* Adversarial attacks and adversarial robustness in computational pathology. *Nat Commun* 2022;**13**:5711.

González-Silva L, Quevedo L, Varela I. Tumor functional heterogeneity unraveled by scRNA-seq technologies. *Trends Cancer* 2020;**6**:13–9.

Gu Z. Complex heatmap visualization. *iMeta* 2022;**1**:e43.

Han X, Hu Y, Foschini L *et al.* Deep learning models for electrocardiograms are susceptible to adversarial attack. *Nat Med* 2020;**26**:360–3.

Hao Y, Stuart T, Kowalski MH *et al.* Dictionary learning for integrative, multimodal and scalable single-cell analysis. *Nat Biotechnol* 2024;**42**:293–304.

Ianevski A, Giri AK, Aittokallio T. Fully-automated and ultra-fast cell-type identification using specific marker combinations from single-cell transcriptomic data. *Nat Commun* 2022;**13**:1246.

Kingma DP, Ba J, Adam A. Method for stochastic optimization. arXiv:1412.6980v9, 2017, preprint: not peer reviewed.

Klughammer J, Abravanel DL, Segerstolpe A *et al.* A multi-modal single-cell and spatial expression map of metastatic breast cancer biopsies across clinicopathological features. *Nat Med* 2024;**30**:3236–49.

Lieberman Y, Rokach L, Shay T. CaSTLe—classification of single cells by transfer learning: harnessing the power of publicly available single cell RNA sequencing experiments to annotate new experiments. *PLoS One* 2018;**13**:e0205499.

Madry A, Makelov A, Schmidt L *et al.* Towards deep learning models resistant to adversarial attacks. arXiv:1706.06083v4, 2019, preprint: not peer reviewed.

Nguyen V, Griss J. scAnnotatR: framework to accurately classify cell types in single-cell RNA-sequencing data. *BMC Bioinformatics* 2022;**23**:44.

Sadria M, Layton A, Bader GD. Adversarial training improves model interpretability in single-cell RNA-seq analysis. *Bioinf Adv* 2023;**3**:vbad166.

Stewart B, Ferdinand J, Young M *et al.* Spatiotemporal immune zonation of the human kidney. *Science* 2019;**365**:1461–6.

Su M, Pan T, Chen QZ *et al.* Data analysis guidelines for single-cell RNA-seq in biomedical studies and clinical applications. *Military Med Res* 2022;**9**:68.

Tan PN, Steinbach M, Karpatne A *et al. Introduction to Data Mining*, 2nd edn. 330 Hudson Street, NY, US: Pearson, 2018.

Tan Y, Cahan P. SingleCellNet: a computational tool to classify single cell RNA-Seq data across platforms and across species. *Cell Syst* 2019;**9**:207–13.e2.

Zhang AW, O'Flanagan C, Chavez EA *et al.* Probabilistic cell-type assignment of single-cell RNA-seq for tumor microenvironment profiling. *Nat Methods* 2019;**16**:1007–15.

Zhao X, Wu S, Fang N *et al.* Evaluation of single-cell classifiers for single-cell RNA sequencing data sets. *Brief Bioinf* 2020;**21**:1581–95.

Zhou Q, Zuley M, Guo Y *et al.* A machine and human reader study on AI diagnosis model safety under attacks of adversarial images. *Nat Commun* 2021;**12**:7281.

Downloaded from https://academic.oup.com/bioinformatics/article/41/4/btaf168/8114001 by guest on 17 September 2025

© The Author(s) 2025. Published by Oxford University Press.
This is an Open Access article distributed under the terms of the Creative Commons Attribution License (https://creativecommons.org/licenses/by/4.0/), which permits unrestricted reuse, distribution, and reproduction in any medium, provided the original work is properly cited.
Bioinformatics, 2025, 41, 1–11
https://doi.org/10.1093/bioinformatics/btaf168
Original Paper

Intelligence artificielle explicable pour la fiabilité des algorithmes d'apprentissage automatique, l'analyse et la découverte dans les données omiques.

Cette thèse montre comment l'intelligence artificielle explicable (XAI) peut améliorer l'analyse des données omiques et la découverte de biomarqueurs. Nous avons d'abord étudié la vulnérabilité des classifieurs de types cellulaires en scRNA-seq face aux attaques adverses, et développé *adverSCarial*, un package R révélant ces faiblesses. Ensuite, nous avons créé *XAItest*, un cadre comparant des méthodes XAI d'importance de variables à des approches statistiques classiques, permettant parfois de détecter des gènes que les statistiques ne révèlent pas. Au final, l'XAI apparaît non seulement comme un outil d'interprétation des modèles, mais aussi comme un moyen d'améliorer la robustesse et la découverte scientifique en bioinformatique.

Mots clés : bioinformatique, intelligence artificielle, XAI, données omiques

Explainable artificial intelligence for the reliability of machine learning algorithms, analysis and discovery in omics data.

This thesis shows how explainable artificial intelligence (XAI) can improve omics data analysis and biomarker discovery. We first investigated the vulnerability of single-cell RNA-seq cell type classifiers to adversarial attacks and developed adverSCarial, an R package that reveals these weaknesses. Then, we created XAItest, a framework comparing XAI-based feature importance methods with classical statistical approaches, sometimes identifying genes overlooked by traditional tests. Overall, XAI proves to be not only a tool for model interpretation but also a means to strengthen robustness and scientific discovery in bioinformatics.

Keywords : bioinformatics, artificial intelligence, XAI, omics data