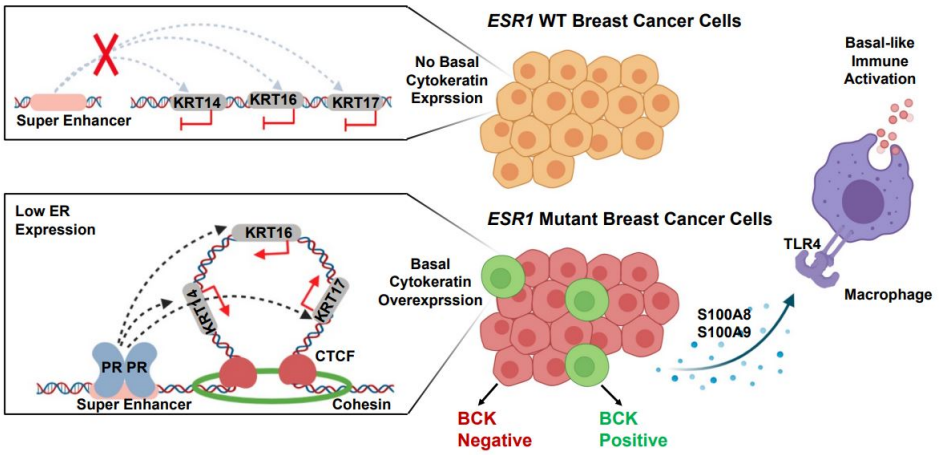
Cancer du sein, expression des gènes S100A8 et ESR1
Régression linéaire et optimisation
- L2 Math, IMIA -

Lhormonorésistance constitue l'un des défis majeurs dans le traitement du cancer du sein avancé exprimant le récepteur aux estrogènes (RE) et sans surexpression de HER2. Des mutations dans le gène ESR1, qui altèrent la région où le ligand se lie (c'est-à-dire la partie du récepteur aux strogènes où les hormones ou autres molécules se fixent), ont été découvertes récemment comme étant l'un des principaux moyens par lesquels certaines cellules cancéreuses résistent aux inhibiteurs de l'aromatase (IA). Les IA sont des médicaments utilisés dans le traitement du cancer du sein hormono-dépendant en bloquant la production d'strogènes. Ces mutations induisent une activation constitutionnelle du RE conduisant à une résistance acquise aux IA.
ESR1 est actif dans environ 75 % des tumeurs du cancer du sein.

Les protéines de la famille S100 sont souvent dérégulées dans le cancer. La protéine S100A8, mésurée dans le sang des patients, a été identifié comme étant l'une deux protéines dont l'expression augmente le plus dans le cancer ESR1 mutant.



Source : Li, Z., McGinn, O., Wu, Y., Bahreini, A., Priedigkeit, N. M., Ding, K., ... , Oesterreich, S. (2022). ESR1 mutant breast cancers show elevated basal cytokeratins and immune activation. Nature Communications, 13(1), 2011.

Le jeu de données est issu d'une étude impliquant 32 patientes atteintes d'un cancer du sein avec une tumeur à récepteurs d'øestrogènes positifs et ayant reçu une chimiothérapie au tamoxifène.
Les variables enregistrées sont :

- `grade` : grade histologique de la tumeur (grade 1 vs 3),
- `ganglion` : état des ganglions lymphatiques (0 : non affecté, 1 : ganglions lymphatiques affectés et enlevés),

— `taille` : taille de la tumeur en cm,

— Expression des gènes `ESR1` et `S100A8` dans les biopsies de tumeurs (technologie des puces à ADN).

Ici on cherche à savoir si l'expression du gène ESR1 est liée à celle de le protéine S100A8. L'intérêt clinique est que l'expression de la protéine S100A8 est plus facile à mesurer que l'expression du gène ESR1. La connaissance précise du type de cancer permet de mieux cibler les traitements.

1 Statistiques descriptives

1. Importer les données

```
import pandas as pd
data = pd.read_csv("brca.csv",sep=",")
print(data) # pour voir ce que contient la table de donnees
ESR1 = data["ESR1"]
S100A8 = data["S100A8"]
```

2. Tracer un nuage de points de de l'expression du gène ESR1 en fonction l'expression de la protéine S100A8 en fonction. Voyez vous un lien entre les deux quantités ?
3. Le lien linéaire n'est pas évident. On propose d'appliquer une transformation en log aux deux expressions. Tracer le nuage de points pour les quantités transformées. Que peut-on dire de la relation entre ces deux variables ?
4. Quel est l'intérêt selon vous de tracer ces expressions de gènes en échelle log-log avec la fonction `loglog` du module `matplotlib` ? Tracer la figure avant de répondre.

2 Régression linéaire simple

Le premier objectif ici est d'écrire une fonction qui prend en entrée deux séries de données de longueur n (le prédicteur x_i et la réponse y_i) et renvoie les paramètres inconnus de la droite de régression.

On rappelle qu'on a obtenu les formules des estimateurs en TD.

$$\begin{aligned}\widehat{\beta}_1 &= \frac{\frac{1}{n}\sum_{i=1}^n y_i x_i - \frac{1}{n}\sum_{i=1}^n y_i \frac{1}{n}\sum_{i=1}^n x_i}{\frac{1}{n}\sum_{i=1}^n x_i^2 - (\frac{1}{n}\sum_i x_i)^2} \\ \widehat{\beta}_0 &= \frac{1}{n}\sum_{i=1}^n y_i - \widehat{\beta}_1 \frac{1}{n}\sum_{i=1}^n x_i\end{aligned}$$

Interprétation statistique des éléments de la formule

On rappelle que la moyenne empirique d'une série $x_i, i = 1 \cdots, n$ est définie par

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$$

et que la variance empirique est définie par

$$var(x) = \frac{1}{n} \sum_{i=1}^{n} x_i^2 - \bar{x}^2$$

En outre, on définit la covariance entre deux séries $x_i, i = 1 \cdots, n$ et d'une série $y_i, i = 1 \cdots, n$ par

$$cov(x, y) = \frac{1}{n} \sum_{i=1}^{n} x_i y_i - \bar{x}\bar{y}$$

5. Quelles fonctions du module `numpy` de python permettent de calculer la moyenne et la variance d'une série $x_i, i = 1 \cdots, n$ réprésentée par un vecteur `x` ?
6. Quelle fonction du module `numpy` permet de calculer la covariance deux séries représentées par des vecteurs `x` et `y` de même longueur ?
7. Comment peut-on reformuler l'estimateur de $\widehat{\beta}_1$ de β_1 en utilisant les statistiques empiriques ?
8. Comment peut-on reformuler l'estimateur de $\widehat{\beta}_0$ de β_0 en utilisant les statistiques empiriques ? On ne remplacera pas $\widehat{\beta}_1$ par son expression.

Fonction python

9. Ecrire une fonction python nommée `SimpleRegLin`. La fonction ne doit pas faire appel à des boucles `for` mais utiliser les fonctions moyenne, variance et covariance du module `numpy`.
   Entête de la fonction

```
def SimpleRegLin(x,y):
    # Estimation des paramètres inconnus de la régresison linéaire simple
    # Entrées
    # x: prédicteurs (tableau numpy nx1)
    # y: réponses (tableau numpy nx1)
    # Sorties
    # par : vecteur de paramètres contenant l'ordonnée à l'origine beta0
    #       et la pente beta1
```

Mise en oeuvre

10. Créer des variables `lESR1 = np.log(ESR1)` et `lS100A8 = np.log(S100A8)` puis estimer les paramètres β_0 et β_1 du modèle de régression linéaire suivant :

$$\text{lESR1} = \beta_0 + \beta_1 \text{lS100A8} + \epsilon$$

où ϵ est une erreur.

11. Tracer, sur une même figure, le nuage de points de lESR1 en fonction de lS100A8 et la doirte de régression. Commenter le graphique obtenu.
12. Ecrire une fonction `predict_SimpleRegLin` qui prend en entrée le résultat de la régression linéaire simple et une (ou plusieurs) valeur(s) du prédicteur et qui renvoie la prédiction de la régression linéaire.
Entête de la fonction

```
def predict_SimpleRegLin(par,x):
   # prédiction pour régresison linéaire simple
   # Entrées
   # par: paramètres estimés par la fonction SimpleRegLin
   # x: prédicteurs (tableau numpy mx1)
   # Sorties
   # y : prédiction
```

13. Utiliser la fonction `predict_SimpleRegLin` pour donner la valeur attendue de lESR1 pour une valeur de lS100A8 de 7 ?
14. Déduire du modèle de régression linéaire, un modèle pour la relation entre l'expression du gène ESR1 et l'expression du de la protéine S100A8. Quelle est la valeur attendue pour l'expression du gène ESR1 pour une valeur de l'expression de la protéine de 1100 ?

3 Optimisation numérique

Dans la suite, nous allons essayer de retrouver les résultats obtenu par la méthode directe (régression linéaire simple) en appliquant un algorithme d'optimisation numérique.

3.1 Estimation par la méthode de la descente du gradient

La solution obtenue ci-dessus est bonne mais l'algorithme utilisé n'est pas général dans le sens où il ne peut être utilisé que pour la régression linéaire simple.
On propose alors d'implémenter l'algorithme de la descente de gradient pour approcher l'estimateur des moindres carrés.

3.2 Exercice préparatoire en dimension 1

On choisit la fonction de perte de l'erreur aux moindres carrés

$$L(w) = \frac{1}{2}\sum_{i=1}^{n}(y_i - w_0 - w_1 x_i)^2$$

3.3 Implémentation du gradient

15. Calculer le gradient de L
16. Implémenter une fonction qui calcule le gradient de L pour la régression linéaire.

```
def grad_reglin(w,X,y) :
# Gradient du coût aux moindres carrés pour la régression linéaire
# Entrées
# - w : vecteur des poids (tableau numpy (d+1)x1)
# - x : prédicteurs (tableau numpy nxd)
# - y : matrice des réponses (tableau numpy nx1)
#Sortie :
# - grad : gradient en w
```

QUESTION OPTIONNELLE : Si vous avez traité les questions 11 et 12 du TD, vous pouvez utiliser les expressions matricielles (voir questions (a) et (b) ci dessous, optionnel).

(a) Pour préparer les données, pour la formulation matricielle, suivre les indications ci-dessous.
Créer un tableau numpy `X`, tel que

$$X = \begin{bmatrix} 1 & x_{11} \\ \vdots & \vdots \\ 1 & x_{n1} \end{bmatrix}$$

Vous pourrez utiliser la fonction repeat du module numpy pour créer un vecteur de 1 et concatenate pour concaténer les deux vecteurs.

(b) On rappelle que (voir TD), si L est la fonction du coût aux moindres carrés,

$$\nabla L(w) = -\mathbf{X}^T(\mathbf{y} - \mathbf{X}\mathbf{w})$$

Attention, `w` et `y` doivent être des tableaux numpy. La fonction dot du module numpy permet de multiplier des vecteurs et/ou des matrices entre eux. La fonction transpose permet de transposer.

17. En choisissant, $w = \begin{pmatrix} 8 \\ -1 \end{pmatrix}$, vérifier que la fonction renvoie $gL = \begin{pmatrix} -118 \\ -551 \end{pmatrix}$

3.4 Algorithme de descente du gradient

18. Implémenter l'algorithme de descente du gradient

```
def grad_desc(w0,grad,x,y,alpha=.001,tol=1e-3,maxiter=1e4) :
# Algorithme de descente du gradient
# Entrées
# - w0 : point initial (tableau numpy (d+1)x1)
```

```
# - grad : gradient de la fonction de coût des moindres carrés
# - x : prédicteurs (tableau numpy nxd)
# - y : matrice des réponses (tableau numpy nx1)
# - alpha : taux d'apprentissage (defaut 0.1)
# - tol :  tolerance (defaut 1e-3)
# - maxiter :  nombre maximum d'itérations (defaut 1e4)
# Sortie :liste contenant
# - sol : solution
# - grad_w: valeur du gradient de la fonction de perte à la solution
# - iter : nombre d'itérations effectuées
# - cvge : booleen (True si l'algorithme a convergé)
```

19. Appliquer l'algorithme de descente du gradient avec les paramètres par défaut, aux données du cancer du sein pour estimer la droite de régression entre $\log(S100A8)$ et $\log(ESR1)$.
    On choisira comme point de départ $w = \begin{pmatrix} 8 \\ -1 \end{pmatrix}$.
    Quel résultat obtient-on ? L'algorithme a t'il convergé ?
20. Essayer d'autres valeurs du taux d'apprentissage.Vous pouvez aussi faire varier la tolérance et le nombre d'iterations maximum.
    Quel résultat obtient-on ? Pour quelle valeur du taux d'apprentissage ? En combien d'itérations ?

3.5 Algorithme de descente du gradient avec back-tracking

21. Implémenter l'algorithme du gradient avec back-tracking, dans une fonction dont l'entête sera la suivante

```
def grad_desc_bt(start,L,grad,x,y,alpha=.1,a=.5,b=.5,tol=1e-3,maxiter=1e4) :
# Algorithme de descente du gradient
# Entrées
# - start : point initial (tableau numpy (d+1)x1)
# - L : fonction de perte
# - grad : gradient de la fonction de perte
# - x : prédicteurs (tableau numpy nxd)
# - y : matrice des réponses (tableau numpy nx1)
# - alpha : taux d'apprentissage (defaut 0.1)
# - a, b : paramètres du back-tracking
# - tol :  tolerance (defaut 1e-4)
# - maxiter :  nombre maximum d'itérations (defaut 10)
# Sortie :liste contenant
# - sol : solution
# - grad_w: valeur du gradient de la fonction de perte à la solution
# - iter : nombre d'itérations effectuées
# - cvge : booleen (True si l'algorithme a convergé)
```

22. Appliquer l'algorithme de descente du gradient avec back-tracking avec les paramètres par défaut, aux données du cancer du sein pour estimer la droite de régression entre

$\log(S100A8)$ et $\log(ESR1)$. On choisira comme point de départ $w = \begin{pmatrix} 8 \\ -1 \end{pmatrix}$. Quel résultat obtient-on ? L'algorithme a t'il convergé ? En combien d'itérations ?

4 Validation croisée

En intelligence artificielle et plus spécifiquement en machine learning, on utilise une partie des données pour mesurer la qualité du modèle (ici la régression linéaire simple). En pratique, quand on dispose de peu d'observations, on utilise la technique de la validation "leave-one-out".

23. Exécuter et commenter les commandes suivantes ainsi que les résultats obtenus. En particulier, vous justifierez que les commandes codent la validation croisée.

```
n = len(lS100A8)
X = lS100A8
y = lESR1
# Listes pour stocker les résultats
y_true, y_pred = [], []

# Effectuer la validation croisée Leave-One-Out
for i in range(len(X)):
    # Créer les ensembles d'entraînement et de test
    X_train = np.delete(X, i, axis=0)
    y_train = np.delete(y, i, axis=0)
    X_test = X[i].reshape(1, -1)
    y_test = y[i].reshape(1, )

    # Entraîner le modèle
    par = SimpleRegLin(X_train, y_train)

    # Prédire l'échantillon de test
    y_pred_sample = predict_SimpleRegLin(par,X_test)

    # Stocker les résultats
    y_true.append(y_test[0])
    y_pred.append(y_pred_sample[0])

y_true = np.array(y_true)
y_pred = np.array(y_pred)
mse = np.mean((y_true-y_pred)**2)
rmse = np.sqrt(np.mean((y_true-y_pred)**2))
mae_rel = np.mean((np.abs(y_true-y_pred)/y_true))

print("Erreur aux moindres carres : ", np.round(mse,2))
print("Racine de l'erreur aux moindres carres : ", np.round(rmse,2))
print("Erreur relative en valeur absolue : ", np.round(mae_rel,2))
```

24. Tracer le nuage de points des prédictions `y_pred` en fonction des observations `y_true` (en log-log puis dans les unités de mesure des expressions). Critiquer le modèle ajusté.

5 Conclusion

L'expression du gène S100A8 est-elle liée à celle de la protéine ESR1 ? Quel est le lien (écrire l'équation) ?