TP n°1 – Tutoriel pour la Régression

L'objectif de ce document est d'apprendre à utiliser R pour effectuer une **régression linéaire simple ou multiple**. L'enjeu d'une telle méthode est de taille puisqu'il s'agit de traduire la relation fonctionnelle qu'il peut y avoir entre une **variable d'intérêt *y*** et certaines variables $\boldsymbol{x^1}$**, …,** $\boldsymbol{x^p}$ qui pourraient avoir un *pouvoir prédictif* ou explicatif fort sur *y*. Le cas p = 1 d'un seul prédicteur correspond à la **régression linéaire simple** et le cas général $p \geq 2$ à la **régression linéaire multiple**. La distinction en pratique vient au moins de deux raisons :

- deux variables prédictives (ou plus) peuvent être très liées ou corrélées, ce qui peut poser des problèmes sérieux dans l'ajustement du modèle. On parle de **multi-colinéarité**. Noter que ce problème ne se pose pas si p = 1.

- il peut y avoir des **interactions** plus ou moins fortes entre les prédicteurs dans leur relation avec la variable à expliquer *y*, ce qui pose le problème de traduire correctement ces interactions pour avoir un bon modèle prédictif.

Toutes les variables considérées sont supposées **quantitatives**. On commence par la régression linéaire simple.

☞ **On ne cherchera pas nécessairement à comprendre ou interpréter tous les résultats fournis par les fonctions R, ce sera l'objectif du cours de comprendre les méthodes ou algorithmes qui sont implémentés et d'apprendre à analyser les résultats fournis.**

PARTIE 1 – Un exemple simple

Commençons par un petit rappel (du cours 1A ou équivalent) sur le modèle de régression linéaire simple. On dispose de n observations (x_1, y_1), …, (x_n, y_n) d'un couple de variables (x, y) où y est la variable d'intérêt à expliquer et x la variable explicative (prédicteur). On fait l'hypothèse forte que les valeurs $y_1, \ldots, y_n$ sont les réalisations de variables aléatoires $\mathbf{Y}_1, \ldots, \mathbf{Y}_n$ telles que

$$\mathbf{1 \leq i \leq n,} \qquad \mathbf{Y_i = \beta_0 + \beta_1 x_i + \varepsilon_i}$$

où β_0, β_1 sont deux scalaires et $\boldsymbol{\varepsilon_1, \ldots, \varepsilon_n}$ **des variables aléatoires gaussiennes indépendantes centrées et de même variance** $\boldsymbol{\sigma^2}$. Remarquer que cette modélisation ne dit rien sur la variable x (qui n'est d'ailleurs par forcément de nature aléatoire, cas de **prédicteur contrôlé**). Par contre, **connaissant les valeurs** $\boldsymbol{x_1, \ldots, x_n}$, les valeurs $y_1, \ldots, y_n$ correspondent à des réalisations de variables **indépendantes** de lois normales respectives $\mathbf{N(\beta_0 + \beta_1 x_i, \sigma^2)}$ de moyenne $\beta_0 + \beta_1 x_i$ et de variance σ^2. Il s'agit d'un **modèle statistique** qui traduit simplement que la relation statistique entre y et x est linéaire (affine) et, de manière précise, qu'une augmentation de 1 unité de x conduit **en moyenne** à un changement de β_1 unités pour y. Ce modèle s'oppose à un *modèle déterministe* où la valeur de x déterminerait entièrement la valeur de y. L'écart $\boldsymbol{\varepsilon_i}$ entre la variable Y_i et sa moyenne $\beta_0 + \beta_1 x_i$ est dû le plus souvent à d'autres variables ou facteurs influençant l'observation n° i de y (associée à la valeur x_i) ainsi qu'à des erreurs de mesure sur y. Mais cet écart correspond aussi en pratique à une *erreur de modèle* tant il est illusoire de vouloir réduire la complexité d'une relation entre deux variables par une simple droite, dite de régression pour des raisons historiques. Néanmoins, en l'absence évidemment d'un gros volume de données (cf. le Big Data), il faudra apprendre à se contenter d'une telle relation, aussi simple soit-elle !

Désormais, on confondra la v.a. Y_i et sa réalisation y_i, ce qui permet d'écrire le modèle sous la forme suivante (dite empirique)

$$\mathbf{1 \leq i \leq n, \qquad y_i = \beta_0 + \beta_1 x_i + \varepsilon_i}$$

et, de manière plus condensée, avec le langage matriciel, sous la forme :

$$\mathbf{Y = X\beta + \varepsilon}$$

Dans cette dernière écriture,

- Y est le vecteur colonne de taille n formé des n observations y_i (ou des Y_i),
- X est la matrice de taille n×2 formé d'un vecteur colonne noté $X^{(1)}$ ne contenant que des 1 et d'un vecteur colonne noté $X^{(2)}$ associé aux x_i
- β vecteur colonne de taille 2 = vecteur des coefficients de régression β_0 et β_1
- ε est le vecteur colonne de taille n formé des **écarts** ε_i au modèle, écarts appelés encore **résidus théoriques** ou, le plus souvent, **erreurs** de régression (**bruit**).

Dans ce modèle, β_0 correspond à l'ordonnée à l'origine (ou **intercept** dans les logiciels) puisque c'est l'ordonnée du point de la droite de régression qui intercepte l'axe des *y*. β_1 est simplement la **pente** de cette droite. Noter qu'un troisième paramètre du modèle, à savoir la **variance σ^2 des erreurs**, se cache dans l'écriture de ε.

Nous allons maintenant voir comment estimer les paramètres inconnus β_0 et β_1 ainsi que la variance d'erreur ou bruit σ^2. Pour cela, on considère l'exemple 1 du support de cours (disponible sur Campus). On dispose de n = 54 données de rendement de blé (variable *y* nommé **rendement** sans unité) sur plusieurs parcelles et plusieurs années ainsi que des précipitations printanières cumulées (variable **pluie** *x* en mètre), voir tableau de données page 10. Ces observations sont contenues dans le fichier texte nommé **data1.txt**.

Chargement des données sous R et premières manipulations (fichiers sur Campus)

Ouvrir le fichier **data1.txt** (avec le Bloc-notes) et remarquer la présence d'un en-tête (**header**) indiquant le nom des deux variables. Les observations sont disposées en deux colonnes séparées par une tabulation.

Créer un répertoire de travail **Regression** dans lequel vous déposerez les deux fichiers **data1.txt** et **ScriptTP1**. Se placer ensuite dans ce répertoire à l'aide (rappel) du menu **Session → Set Working Directory → Choose Directory**… (ou directement en ligne de commande avec la fonction **setwd**()). Ouvrir le fichier **ScriptTP1** de format texte qui contient une liste d'instructions R que vous allez exécuter pas à pas (avec le menu **Run** ou, au clavier, avec la combinaison de touches **Ctrl + Enter**).

Commencer par la première instruction ci-dessous (noter l'argument en option **header=TRUE** puisque par défaut **header=FALSE**) :

```
data1 <- read.table("data1.txt", header=TRUE)
```

La fonction R **read.table** crée automatiquement un objet « **tableau de données** » ou « **data frame** » avec les deux variables nommées **pluie** et **rendement** en colonnes, les valeurs observées étant donc en

ligne, ce qui est l'usage en Statistique. Pour vérifier que l'on a bien un **data.frame** et en savoir plus sur son contenu, faire successivement :

```
is.data.frame(data1)
names(data1)
print(data1)
summary(data1)
```

Ajustement du modèle linéaire (droite aux moindres carrés) et premier graphique.

La fonction **lm** (pour **Linear Model**) est la fonction de base pour effectuer une régression linéaire sous R. Retrouver les valeurs de β_0 et β_1 estimées à partir des données par la *méthode des moindres carrés*, et dont les valeurs sont données page 12 du support.

```
pluie <- data1$pluie
rend <- data1$rendement
data1.reg <- lm(rend ~ pluie)
print(data1.reg)
```

Obtenir le graphique de la page 12 permettant de visualiser à la fois les données et la droite de régression en exécutant les commandes correspondantes et reproduites ci-dessous :

```
plot(pluie, rend, xlab="hauteur précipitations cumulées (m)", ylab="rendement (sans unité)", lwd=2)
abline(data1.reg, lwd=2, lty=1, col="red")
title(main=list("Régression linéaire : rendement de blé contre quantité de pluie", cex=1))
```

Vous pouvez ensuite facilement exporter ce graphique pour la production d'un rapport ou étude statistique à l'aide du menu de la fenêtre graphique. On aimerait maintenant en savoir un peu plus sur ce que retourne la fonction **lm** dans l'objet **data1.reg**. Pour cela, on utilise la fonction générique **summary** :

```
data1.reg.s <- summary(data1.reg)
print(data1.reg.s)
```

On retrouve certains résultats du support page 30, lesquels ?

```
Call:
lm(formula = rend ~ pluie, data = data1)

Residuals:
     Min       1Q   Median       3Q      Max
-0.119861 -0.034987  0.003603  0.040208  0.108037

Coefficients:
             Estimate   Std. Error  t value   Pr(>|t|)
(Intercept)  0.06620    0.02405     2.752     0.00813 **
pluie        1.63673    0.07526    21.747    < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.05201 on 52 degrees of freedom
Multiple R-squared:  0.9009,       Adjusted R-squared:  0.899
F-statistic: 472.9 on 1 and 52 DF,  p-value: < 2.2e-16
```

Dans l'ordre d'affichage, R renvoie

- les valeurs des résidus estimés si la série est courte, sinon des résumés numériques (comme ci-dessus)

- les estimations des coefficients β_0 et β_1 avec l'erreur d'estimation associée, valeur de la statistique t de Student pour tester leur nullité et la p-valeur associée
- estimation de l'écart-type σ de l'erreur ε
- le fameux coefficient de détermination R^2 ainsi que le R^2 ajusté
- enfin la statistique du test de Fisher et sa p-valeur

Pour créer la table d'analyse de la variance associée (**AN**alysis **O**f **V**ariance ou **ANOVA table**), on peut utiliser la fonction **anova**

```
anova(data1.reg)
```

ce qui fournit (comparer avec la table page 19 du support) :

```
Analysis of Variance Table

Response: rend
           Df  Sum Sq  Mean Sq F value    Pr(>F)
pluie       1 1.27917 1.2792    472.92 < 2.2e-16 ***
Residuals 52 0.14065 0.0027
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

On verra en cours l'interprétation complète de ces résultats. Pour aller plus loin, on peut construire un intervalle de confiance sur les paramètres β_0 et β_1 ainsi qu'une région de confiance :

```
confint(data1.reg)
confint(data1.reg, level=0.9)
# ellipse de confiance
library(car) # pour tracer des ellipses!
confidenceEllipse(data1.reg)
```

Expliquer la forme de l'ellipse en considérant la matrice de variance-covariance des estimateurs de β_0 et β_1 donnée par la fonction **vcov** :

```
VarBeta.hat <- vcov(data1.reg)
print(VarBeta.hat)
```

Passons maintenant à la prédiction en supposant provisoirement que le modèle proposé soit valide. Pour une nouvelle valeur x_{new} de la variable pluie, on peut se demander quelle serait la valeur de la variable rendement. Si le modèle est juste, le rendement associé est une variable aléatoire Y de la forme

$$Y = \beta_0 + \beta_1 x_{new} + \varepsilon$$

de loi $N(\beta_0 + \beta_1 x_{new}, \sigma^2)$. La moyenne $\boldsymbol{\beta_0 + \beta_1 x_{new}}$ de cette loi est **l'espérance du rendement** en réponse à la valeur x_{new} de la variable pluie, appelée encore **réponse espérée**. Elle ne doit pas être confondue avec la **prédiction** ponctuelle de Y. On indique maintenant comment construire un **intervalle de confiance sur cette réponse espérée** ainsi qu'un **intervalle de prédiction pour Y**. On comparera le résultat avec celui donné page 34 du support.

```
# intervalle de confiance pour la réponse espérée

plot(pluie, rend, xlab="hauteur précipitations cumulées (m)", ylab="rendement (sans unité)", lwd=2, pch=3)
abline(data1.reg, lwd=2, lty=1, col="blue")

newdata <- data.frame(pluie=seq(min(pluie), max(pluie), by=0.01)) # attention à bien spécifier pluie = ...
int.conf <- predict(data1.reg, newdata, interval="confidence")
lines(newdata[,1], int.conf[,2], col="red", lty=2, lwd=2)
lines(newdata[,1], int.conf[,3], col="red", lty=2, lwd=2)

# Intervalle de prédiction pour la réponse

int.pred <- predict(data1.reg,newdata,interval="prediction")
lines(newdata[,1], int.pred[,2], col="black", lty=2)
lines(newdata[,1], int.pred[,3], col="black", lty=2)
title(main=list("Intervalle de confiance pour la réponse espérée et intervalle de prédiction", cex=1))
```

Après avoir vu l'intérêt d'un modèle de régression linéaire pour faire de la prédiction, se pose maintenant la délicate question de la validité du modèle et donc aussi la question de la fiabilité des prédictions obtenues. Par exemple, est-on bien sûr que les intervalles de prédiction calculés sont de niveau de confiance 95% ?

La réponse à ces questions passe par l'analyse des résidus. Rappelons que les données $y_1, \ldots, y_n$ sont supposées être des réalisations de v.a. de la forme $\mathbf{Y_i = \beta_0 + \beta_1 x_i + \varepsilon_i}$ où les résidus (théoriques) $\boldsymbol{\varepsilon_i}$ sont i.i.d. $N(0, \sigma^2)$ (soit un bruit blanc gaussien). Comme β_0 et β_1 sont inconnus, on n'a pas accès directement aux valeurs des ε_i, il faut les estimer.

Résidus estimés. Notons $\hat{\beta}_0$ et $\hat{\beta}_1$ les valeurs des paramètres estimés (coefficients de la droite aux moindres carrés). On appelle **résidus estimés** les quantités $\hat{\varepsilon}_i = y_i - (\hat{\beta}_0 + \hat{\beta}_1 x_i)$ **(1 ≤ i ≤ n)**. On considère dans un premier temps que ces résidus estimés approchent bien les résidus ε_i du modèle, lesquels ne sont pas directement observables. La première chose à faire est de les tracer pour vérifier s'ils correspondent à peu près à un bruit blanc.

La fonction **residuals** permet de calculer ces résidus estimés et donc de les représenter (on peut aussi les récupérer directement avec **$residuals**) :

```
res <- residuals(data1.reg)
n <- length(res)
plot(1:n, res, ylim=c(-0.2,0.15), xlab="numéro d'observation i = 1, ..., n ", ylab="résidus estimés")
title(main=list("Tracé séquentiel des résidus estimés", cex=1))
```

Qu'en pensez-vous ?

Un autre moyen de les visualiser est de considérer les commandes suivantes :

```
plot(data1.reg$fitted.values, res, ylim=c(-0.2,0.15), xlab="réponse estimée", ylab="résidus estimés")
title(main=list("Tracé des résidus estimés en fonction de la réponse estimée",cex=1))
```

Retrouver le graphique page 14 du support. **Qu'observe-t-on ?** On s'intéresse donc naturellement au modèle linéaire

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \varepsilon$$

qui consiste à rajouter le prédicteur x^2. C'est encore la fonction **lm** qui permet de mettre en œuvre une telle régression multiple (cf. p = 2) :

```
# vers le modèle linéaire y ~ x + x^2
pluie2 <- pluie*pluie
data1.reg2 <- lm(rend ~ pluie + pluie2)
print(data1.reg2)
```

Remarquer la création de la deuxième variable pluie2 puis la formule « rend ~ pluie + pluie2 » pour régresser la variable **rend** contre les deux variables **pluie** et **pluie2**.

On peut ensuite visualiser l'ajustement du modèle (comparer avec la figure bas de page 14 du support) :

```
plot(pluie, rend, xlab="hauteur précipitations cumulées (m)", ylab="rendement (sans unité)", lwd=2, xlim=c(0,0.5), ylim=c(0,0.9))
lines(pluie, data1.reg2$fitted.values, col="red", lwd=2)
title(main=list("Régression linéaire avec terme quadratique : rendement de blé contre pluie", cex=1))
```

ainsi que les « nouveaux » résidus (comparer avec la figure page 15) :

```
res2 <- residuals(data2.reg)
plot(data1.reg2$fitted.values, res2, xlim=c(0,1), ylim=c(-0.15,0.1), xlab="réponse estimée", ylab="résidus estimés")
title(main=list("Tracé des résidus estimés en fonction de la réponse estimée", cex=1))
```

A ce stade, on se dit que l'on rend bien compte de la réponse espérée par une fonction de x, la fonction de régression n'étant pas affine (droite) mais un polynôme de degré 2 (parabole). Par contre, il faut s'intéresser à d'autres hypothèses du modèle comme l'indépendance et la normalité des résidus.

Pour l'indépendance, les résidus estimés (du dernier modèle avec terme quadratique) sont liés par les p + 1 = 3 relations algébriques suivantes

$$\sum_{i=1}^{n} \hat{\varepsilon}_i = 0 \; ; \; \sum_{i=1}^{n} x_i \times \hat{\varepsilon}_i = 0 \; ; \; \sum_{i=1}^{n} x_i^2 \times \hat{\varepsilon}_i = 0$$

Le vérifier sous R. Pour aller plus loin dans l'étude des résidus, on utilise le résultat suivant (voir support page 37) : $\hat{\boldsymbol{\varepsilon}} = (\hat{\varepsilon}_1, \ldots, \hat{\varepsilon}_n)'$ **est un vecteur gaussien** $\mathbf{N(0, \sigma^2(I_n - H))}$

où I_n est la matrice identité de taille n et H la matrice explicite $X(X'X)^{-1}X'$ dite matrice chapeau. On introduit alors les **résidus standardisés** (ou résidus studentisés internes)

$$\mathbf{T_i} = \frac{\hat{\boldsymbol{\varepsilon}}_i}{\hat{\sigma}\sqrt{1 - h_{ii}}}$$

où h_{ii} est le i-ème terme diagonal de la matrice H (associé donc à l'observation n° i). Sous R, il sont calculés avec la fonction **rstandard** :

```
plot(rstandard(data1.reg2), xlab="numéro d'observation i", ylab="Résidu standardisé", ylim=c(-3,3))
title('Résidus standardisés')
```

Pour être plus précis dans l'analyse, il faut en réalité considérer les **résidus studentisés** (de manière externe ou **résidus par validation croisée**) $\mathbf{T_i^* = \frac{\hat{\varepsilon}_i}{\hat{\sigma}_{(i)}\sqrt{1-h_{ii}}}}$

où $\hat{\sigma}_{(i)}$ est l'estimation de σ lorsqu'on fait une régression sans la i-ème donnée. On montrera en cours que les T_i^* sont de loi de Student à n – (p+1) degrés de liberté. Du coup, on peut préciser le graphique précédent :

```
plot(rstandard(data1.reg2), xlab="numéro d'observation i", ylab="Résidus", ylim=c(-3,3), lwd=2)
points(rstudent(data1.reg2), col="red", pch=3)
abline(h=qt(p=0.025, df=data1.reg2$df.residual), col="red", lwd=2)
abline(h=qt(p=0.975, df=data1.reg2$df.residual), col="red", lwd=2)
title('Résidus standardisés et studentisés')
```

Expliquer ce que représente la bande en rouge. Conclusion.

Partie II – Régression linéaire multiple

Ecrire un script R pour étudier l'exemple 2 du support page 37. Les données sont disponibles dans le fichier texte « data2.txt ». On souhaite expliquer le temps en minutes (variable **temps**) pour approvisionner un réseau de distributeurs de boissons à partir du nombre de caisses de bouteilles placées (variable **nb**) et de la distance parcourue en mètres (variable **dist**).

On utilisera les fonctions suivantes :

- **read.table** pour la lecture des données
- **plot, pairs, boxplot, summary** pour examiner des données
- **lm** pour construire un modèle de Régression