TP 3 : analyses statistiques

Exercice I : Le test de student

1. Construire un vecteur (nommé X) de 100 valeurs dont chaque élément est issu d'une loi normale de moyenne nulle et de variance unitaire. Construire un vecteur (nommé Y) de 50 valeurs dont chaque élément est issu d'une loi normale centrée en 1 et de variance unitaire.
2. Réaliser un test de student pour tester l'égalité des moyennes
3. Quel type d'objet est renvoyé par la fonction utilisée en question 2.
4. Récupérer la p-valeur du test. Qu'en concluez-vous ?

Exercice II : La régression linéaire simple

Dans le package datasets est disponible le jeu de données cars. Ce jeu de données est composé de deux variables la première indiquant la vitesse de voitures et la seconde le temps de freinage.

1. Charger le jeu de donnée 'cars'
2. Tracer le graphe bivarié (abscisses=vitesse, ordonées= distance). Que constate t'on ?
3. Construire un modèle linéaire qui analyse la distance de freinage comme fonction de la vitesse. Stocker le résultat du modèle dans un objet nommé result.lm.
4. Ajouter au graphe bivarié construit à la question 1, la droite de régression résultant du modèle de la question 2.
5. Donner les valeurs prédites par le modèle pour chacune des observations.
6. Calculer la différence entre les valeurs prédites par le modèle et les valeurs observées. Comparer ce résultat aux résidus disponibles dans l'objet residuals de la liste des outputs
7. Construire et visualiser des intervalles de prédiction et de confiance.

Exercice III : l'analyse en composantes principales

Pour illustrer l'Analyse en Composantes Principales, nous allons utiliser le jeu de données bordeaux_R.txt.

1. Charger le jeu de données bordeaux_R.
2. Réaliser une analyse en composantes principales (à partir des 4 variables température, soleil, chaleur, pluie) sur les variables centrées réduites.

3. Afficher sur un même graphique les variables et les individus (biplot) sur les deux premiers axes principaux.
4. Afficher les qualités des vins comme labels des individus.
5. Interpréter le résultat de l'analyse.
6. Donner le pourcentage de variance expliquée par les deux premières composantes

Exercice IV : la régression logistique

1. Récupérer les deux premières composantes principales de l'exercice précédent et les stocker dans un objet nommé X .
2. Construire une variable binaire (nommée y) prenant la valeur 1 si le vin est « bon » et prenant la valeur 0 si le vin est « moyen » ou « médiocre »
3. En utilisant la fonction glm() construire un modèle logistique de Y sur X.
4. Récupérer les probabilités prédites par le modèle
5. En déduire les décisions du modèle pour chacun des individus (On seuillera les probabilités d'appartenance à un seuil fixé à 0.5).
6. Renvoyer le tableau de contingence. En déduire le taux de bonne classification en apprentissage.
7. On s'intéresse maintenant au taux de bonne classification en test. Pour ce faire sélectionner aléatoirement 22 individus (avec la fonction 'sample') pour construire votre modèle et le restant pour le tester.
8. Comparer les tableaux de contingence d'apprentissage et de test. En déduire les taux d'erreur en apprentissage et en test.

Exercice V : La methode des k-means

1. Récupérer l'objet nommé X avec les scores de l'analyse en composantes principales et tracer le graphe bivarié du premier score sur le deuxième ; colorer les points selon leur qualité (« bon », « moyen », « médiocre »).
2. Construire une partition de 3 groupes de ce nuage de points via l'algorithme des kmeans.
3. Lier chacune des observations à son centroïde par un segment et compter visuellement le taux de mal classés.

Correction de l'exercice I : Le test de student

```
X = rnorm(100) ; Y = rnorm(50, mean = 1)
res = t.test(X, Y)
res est un objet de type list()
summary(res)
names(res)
```

On constate que la p-valeur est stockée dans le troisième élément de la liste. On récupère la p-valeur par la commande res[[3]]. La p-valeur est inférieur à 0.05 ce qui nous conduit, comme attendu, à rejeter l'hypothèse nulle d'égalité des moyennes.

Correction de l'exercice II : La régression linéaire simple

```
1.
library(datasets) # déjà chargé
2.
plot(cars$speed, cars$dist)
```

on constate une relation linéaire entre la distance de freinage et la vitesse du véhicule. Comme on pouvait s'y attendre, plus le véhicule est rapide, plus le temps d'arrêt est important.

```
3.
result.lm = lm(cars$dist ~ cars$speed)
4.
abline(result.lm)
5.
Yhat = predict(result.lm)
6.
cbind(cars$dist - Yhat, result.lm$residuals) # à comparer à
result.lm$residuals
7.
#intervalle de prediction!:
predict(result.lm, interval="prediction")
#intervalle de confiance!:
predict(result.lm, interval="confidence")
# visulalisation
pred1<-predict(result.lm, interval="prediction")
pred2<-predict(result.lm, interval="confidence")
l.min = min(pred1[,2],pred2[, 2])
l.max = max(pred1[,3],pred2[, 3])
plot(cars$speed, cars$dist, ylim = c(l.min, l.max))
ord<-order(cars$speed)
lines(cars$speed[ord], pred1[ord, 2], lwd = 1, col = "green")
lines(cars$speed[ord], pred1[ord, 3], lwd = 1, col = "green")
lines(cars$speed[ord], pred2[ord, 2], lwd = 1, col = "red")
lines(cars$speed[ord], pred2[ord, 3], lwd = 1, col = "red")
```

Correction de l'exercice III : l'analyse en composantes principales

```
1.
A = read.table("Datasets/bordeaux_R.txt", row.names=1, header =
TRUE, sep = "\t")
2.
result.pca = princomp(A[ ,1:4], cor = TRUE)
3.
biplot(result.pca)
4.
nom = factor(A$QUALITE, labels=c("bon", "moyen", "mediocre"))
#ou nom = c("bon", "moyen", "mediocre")[as.numeric(A$QUALITE)]
biplot(result.pca, xlabs = nom)
5.
On remarque qu'une année peu pluvieuse jumelée à de forte chaleur
fournira des vins de bonne qualité
6.
summary(result.pca) # = (result.pca$sdev^2)/4
plot(result.pca)
```

Correction de l'exercice IV : la régression logistique

```
1.
X <- result.pca[[6]][ ,1:2] # ou X <- result.pca$scores[,1:2]
2.
y <- A$QUALITE ;  y[which(y>1)] = 0 ; y[y>1] <- 0
3.
res.log <- glm(y~X, family = "binomial")
4.
proba <- res.log$fitted.values
5.
ypredit = rep(1, nrow(A)) ; ypredit[proba<0.5] = 0
6.
tab = table(y, ypredit) ; sum(diag(tab))/nrow(A)
7.
set.seed(11)
ind = sample(nrow(A),22)
glm.app = glm(y~X, family = "binomial", subset=ind)
glm.app$fitted.values
glm.proba <- predict(glm.app,type ="response",
newdata=as.data.frame(X))
proba.app <- glm.proba[ind]
proba.test <- glm.pred[-ind]
pred.app <- rep(1,length(proba.app))
pred.app[proba.app<0.5] = 0
pred.test <- rep(1,length(proba.test))
pred.test[proba.test<0.5] = 0
tab.app <- table(pred.app, y[ind])
sum(diag(tab.app))/sum(tab.app)
```

```
tab.test <- table(pred.test, y[-ind])
sum(diag(tab.test))/sum(tab.test)
```

Remarque importante : Si l'on souhaite évaluer la qualité d'un modèle, le taux d'erreur important est celui évaluer sur la base de test. Un taux d'erreur faible sur la base d'apprentissage et fort sur la base de test indique que le modèle n'a pas de capacité de généralisation.

Correction de l'exercice V : La methode des k-means

```
1.
result.pca$scores[,1:2] <- X
plot(X, col = A$QUALITE)
2.
result.kmeans = kmeans(X, 3)

3.
plot( X, col = A$QUALITE,
xlab = "premiere composante principale",
ylab = "deuxieme composante principale ",
main = "Les vins de bordeaux")

points(result.kmeans$centers, col = c(2, 1,3) , pch = 7, lwd = 3)

segments(X[result.kmeans$cluster == 1, ][ , 1],
     X[result.kmeans$cluster == 1, ][ , 2],
     result.kmeans$centers[1, 1],
     result.kmeans$centers[1, 2],
     col = 2
)

segments(X[result.kmeans$cluster == 2, ][ , 1],
     X[result.kmeans$cluster == 2, ][ , 2],
     result.kmeans$centers[2, 1],
     result.kmeans$centers[2, 2],
     col = 1
)

segments(X[result.kmeans$cluster == 3, ][ , 1],
     X[result.kmeans$cluster == 3, ][ , 2],
     result.kmeans$centers[3, 1],
     result.kmeans$centers[3, 2],
     col = 3
)
```