Fiche 1 : Faire ses premiers pas avec R

Ouverture, sauvegarde, chargement, et fermeture d'une session R

On peut ouvrir une session de R classique ou dans l'IDE RStudio en cliquant sur l'icone correspondant.
Pour quitter la session et en sauvegarder une image : `q()` ou `save.image()`.
Pour charger les éléments d'une session sauvegardée : `load()`, `loadhistory()`.

Gestion de l'espace de travail depuis la console

Pour connaître, choisir le répertoire courant : `getwd()`, `setwd()`.

Gestion des packages depuis la console

Pour installer, mettre à jour un package : `install.packages()`, `update.packages()`.
Pour appeler un package installé : `library()`.

Gestion des instructions et des sorties

Pour exécuter les instructions d'un script : `source()`.
Pour rediriger les sorties R : `sink()`.
Pour rappeler les instructions précédentes : `history()`.
Pour séparer, commenter des instructions dans un script : ;, #.

Gestion de l'aide

Pour obtenir de l'aide : `help.start()`, `help()` ou `?`, `help(package=)`, `help.search()`, `RSiteSearch()`.
Pour exécuter les exemples donnés dans une aide : `example()`.

Exercice 1

1. Ouvrir une session de R dans l'IDE RStudio. Regarder quel est le répertoire courant. Créer un répertoire `StatDonnees`, puis un sous-répertoire `Fiche1`. Choisir ce sous-répertoire comme répertoire courant.
2. Exécuter les commandes suivantes directement dans la console :

```
> 2+3
> 2+
+ 3
> x <- 2; y <- 3
> x
> y
> x+y
```

3. Quitter R en sauvegardant l'espace de travail.
4. Ouvrir une nouvelle session de R en récupérant les objets et l'historique de la session précédente. Vérifier le chargement correct (s'aider de `history()`).
5. Exécuter `> u <- x^2`, puis sauvegarder tous les objets de cette session dans un fichier `Exercice1.RData`.
6. Exécuter `> v <- y^2`, puis ajouter cet objet dans le fichier `Exercice1.RData`.
7. Quitter R sans sauvegarde automatique et ouvrir une nouvelle session. Charger les objets sauvegardés dans `Exercice1.RData`. Vérifier que le chargement est correct. Quitter R sans sauvegarde.

Exercice 2

1. Ouvrir une nouvelle session de R dans RStudio et écrire les instructions de l'exercice 1 ci-dessus sous forme de script, en ajoutant des commentaires.
2. Exécuter ces instructions à l'aide de `Run` puis à l'aide de la fonction `source`.

Exercice 3

1. Consulter l'aide du package `MASS`, puis celle du jeu de données `UScrime`.
2. Appeler l'aide de la fonction `sample`, exécuter les instructions données comme exemples dans cette aide.

Fiche 2 : Créer et manipuler des objets de base

Généralités : création, affichage, sauvegarde des objets

Pour créer un objet nommé `x`, l'afficher (ou en partie) : `x<-`, `->x` ou `x=`, `print(x)`, `View(x)`, `summary(x)`, `head(x)`, `tail(x)`.
Pour afficher la liste des objets : `objects()`, `ls()`, `ls.str()`. Pour effacer / sauvegarder un objet : `rm()`, `save()`.

Affichage, manipulation des classes, attributs, valeurs manquantes des objets

Pour connaître/convertir la classe d'un objet : `class()`, `is.vector()`..., `as.vector()`...
Pour connaître/convertir les attributs d'un objet : `mode()`, `length()`, `attributes()`, `is.numeric()`..., `as.numeric()`...
Pour savoir si l'objet contient des valeurs manquantes, les enlever : `is.na()`, `is.nan()`, `na.fail()` `na.omit()`.

Création de vecteurs, matrices, facteurs, listes, data frames

Pour créer un vecteur numérique : `x=` , `c()`, `scan()`, `numeric()`, `rep()`, `seq()`, `1:10`, `sequence()`.
Pour créer un vecteur de caractères : `x=" "`, `x=' '`, `c()`, `character()`, `rep()`, `format()`, `paste()`...
Pour créer un vecteur logique : `x=TRUE`, `x=FALSE`, `c()`, `logical()`. À l'aide d'opérateurs : (`!=,==,<,>,<=,>=,`%in%).
Pour créer une matrice : `matrix()`, `rbind()`, `cbind()`, `diag()`. Pour créer un facteur : `factor()`, `gl()`, `cut()`.
Pour créer une liste, un data frame : `list()`, `data.frame()`.

Extraction

Pour extraire des éléments d'un vecteur `vect` : `vect[ ]`, `vect[- ]`.
Pour extraire des éléments d'une matrice `mat` : `mat[ , ]`, `diag(mat)`.
Pour extraire des éléments d'un facteur `fact` : `fact[ ]`, `levels(fact)`.
Pour extraire des éléments d'une liste, d'un data frame `liste` : `liste$` , `liste[[ ]]`, `liste[c()]`.

Manipulation générale de vecteurs

Pour créer un vecteur à partir d'autres : `c()`, `sample()`, `rev()`, `match()`...
Pour repérer des éléments : `which()`. Pour connaître la dimension d'un vecteur : `length()`, `NROW()`, `NCOL()`.

Manipulation de vecteurs numériques

Opérateurs : +, -, *, / , ^, %%, %/%. Fonctions : `sqrt`, `abs`, `log`, `exp`, `log10`, `sin`, `cos`, `tan`, `asin`, ...
Sommes, produits, différences : `sum()`, `prod()`, `diff()`, `cumsum()`, `cumprod()`. Rangs : `sort()`, `rank()`, `order()`.
Minimum, maximum : `min()`, `max()`, `range()`, `which.min()`, `which.max()`, `pmin()`, `pmax()`, `cummin()`, `cummax()`.
Signe, parties entières, arrondis : `sign()`, `floor()`, `ceiling()`, `round()`, `signif()`.
Fonctions statistiques de base : `table()`, `mean()`, `var()`, `cov()`, `corr()`, `sd()`, `scale()`, `median()`, `quantile()`.

Manipulation de vecteurs de caractères

Pour extraire ou remplacer des éléments : `substr()`, `sub()`, `gsub()`.

Manipulation de vecteurs logiques

Opérateurs logiques : !x (non), x&y (et), x&&y, x|y (ou), x||y, `xor(x,y)`. Tests logiques : `all()`, `any()`.
Opérateurs numériques : `FALSE` est transformé en 0, `TRUE` en 1.

Manipulation de matrices

Pour concaténer des matrices : `rbind()`, `cbind()`. Pour appliquer une fonction à une marge : `apply()`.
Opérateurs et fonctions vectoriels agissent élément par élément.
Opérateurs matriciels : `nrow()`, `ncol()`, `NROW()`, `NCOL()`,`t()`, A%*%B, `det()`, `solve()`.
Diagonalisation, réduction : `eigen()`, `svd()`, `chol()`, `qr()`.

Utilisation de facteurs

Pour appliquer une fonction à un vecteur selon les niveaux d'un ou plusieurs facteurs : `tapply()`, `by()`.

Manipulation de listes

Pour concaténer des listes : `c()`. Pour appliquer une fonction aux composantes d'une liste : `lapply(), sapply()`.

Manipulation de data frames

Pour agréger des data frames : `cbind(), rbind(), merge()`.
Pour appliquer une fonction à une marge : `apply(), lapply(), sapply(), sweep()`.
Pour appliquer une fonction à certains éléments liés à un facteur : `tapply(), aggregate(), by()`.

Exercice

1. Créer de plusieurs façons différentes le vecteur $(1, 2, 3, 4, 5, 6)$.
2. Créer le vecteur `x` égal à $(1, 1, 2, 2, 2, 3, 3, 3, 3, 4, 4, 4, 4, 4)$.
3. Extraire les 1er, 3ème, 6ème et 10ème éléments du vecteur `x`, puis extraire tous les éléments de `x` sauf les précédents.
4. Extraire les éléments d'indices pairs de `x`.
5. Créer un tableau contenant les effectifs de chaque valeur dans `x`.
6. Créer un vecteur `y` de 40 réels tirés au hasard entre 45 et 100 à l'aide de la fonction `runif`, et arrondir les éléments de `y` à 1 décimale.
7. Extraire les éléments de `y` :
   - compris entre 50 et 70, et retrouver les indices de ces éléments dans le vecteur `y`,
   - compris entre 50 et 60 ou entre 70 et 80, et retrouver les indices de ces éléments dans le vecteur `y`,
   - égaux à 50, 60 ou 70, et retrouver les indices de ces éléments dans le vecteur `y`,
   - différents de 50, 60 et 70, et retrouver les indices de ces éléments dans le vecteur `y`.
8. Créer un vecteur `z` de 40 entiers tirés au hasard avec remise entre 1 et 10.
9. Créer un vecteur `Patients` composé des chaînes de caractères `(Patient1,...,Patient40)`.
10. Créer une matrice `A` de taille 2×5 contenant colonne par colonne les 10 premiers éléments du vecteur `z` puis une matrice `B` de taille 5×2 contenant ligne par ligne les 10 éléments suivants du vecteur `z`.
11. Exécuter `>A*t(B)` et commenter. Calculer le produit matriciel `AB` de `A` par `B`. Calculer le déterminant de la matrice `AB`, ainsi que son inverse si `AB` est inversible. Extraire la diagonale de `AB`, la première ligne de `AB`, puis les éléments supérieurs à 100 de `AB`.
12. Créer un facteur `Sexe` à 2 niveaux `F` et `M` et composé de 40 éléments pris au hasard dans {`F,M`}.
13. Créer un facteur `Service` à 6 niveaux codés de 1 à 6, et composé de 40 éléments pris au hasard dans {`1,2,...,5`}. Remplacer deux valeurs du facteur par `NA`. Vérifier que le facteur `Service` contient des données manquantes.
14. Créer un vecteur de dates `Date.entree` composé de 40 dates prises au hasard entre le 1er septembre 2015 et le 30 septembre 2015.
15. Créer un data.frame `Urgences1` contenant les objets `Sexe`, `Service`, `Date.entree`, `y` renommé en `Poids`, et `z` renommé en `NbreExamens`, et dont les lignes sont dénommées par `Patients`. Afficher un résumé de `Urgences1`.
16. Extraire de `Urgences1` :
   - ses 10 premières lignes,
   - la variable `Poids`,
   - les 10 premiers éléments de la variable `Poids`,
   - les patients pour lesquels la variable `Poids` est comprise entre 80 et 100,
   - les patients pour lesquels la variable `Service` est égale à 2 ou 3,
   - les patients pour lesquels la variable `Poids` est comprise entre à 80 et 100, et la variable `NbreExamens` est supérieure à 7.
17. Retirer de `Urgences1` les lignes contenant des données manquantes.
18. Extraire les données correspondant à un sexe féminin et les stocker dans un data.frame `Femmes`. Extraire les données correspondant à un sexe masculin et les stocker dans un data.frame `Hommes`.
19. Calculer la moyenne du nombre d'examens pour chaque niveau du facteur `Service`, pour les femmes puis pour les hommes, à partir des data.frames `Femmes` et `Hommes`. Faire la même chose à partir du data.frame d'origine `Urgences1`.
20. Créer un data.frame `Urgences2` contenant un facteur `Service` composé des éléments $1, \ldots, 6$, puis un vecteur `Médecin` composé des chaînes de caractères "Cameron", "Chase", "Foreman", "Masters", "Taub" et "Numéro13". Agréger dans un data frame `Urgences` les data frames `Urgences1` et `Urgences2` suivant la clef `Service`. Afficher le résultat obtenu et le sauvegarder dans un fichier `UrgencesHouse.txt`.

Fiche 3 : Manipuler des données

Création, affichage de tableaux ou tables de données

Pour créer un data frame : `data.frame()`.
Pour créer une data table, ou transformer un data frame en data table : `library(data.table)`, `data.table()`.
Pour afficher un résumé statistique : `summary()`.
Pour afficher des data tables : `tables()`.

Extraction d'éléments d'un tableau ou d'une table de données

Pour extraire des éléments d'un data frame `donnees` : `donnees[,]`, `donnees$` , `donnees[[ ]]`, `donnees[c()]`, `donnees[[" "]]`.
Pour "attacher" les composantes d'un data frame : `attach()`.
Pour extraire des éléments d'une data table `donnees.t` : `donnees.t[]`, `donnees[! ]`, `donnees[,list()]`.
Pour retirer une composante `v` d'une data table `donnees.t` : `donnees.t[,v:=NULL]`.

Manipulation de tableaux ou de tables de données

Pour afficher/attribuer des noms : `names()`.
Pour agréger des data frames : `cbind(), rbind(), merge()`.
Pour appliquer une fonction aux composantes d'un data frame : `apply(), lapply(), sapply(), sweep()`.
Pour diviser un data frame en data frames définis par les valeurs d'un facteur : `split()`.
Pour appliquer une fonction à des éléments d'un data frame : `tapply(), aggregate(), by()`.
Pour afficher les classes des objets d'une data table : `sapply(,class)`
Pour créer, manipuler la clef d'une data table : `setkey()`, `haskey()`, `key()`.
Pour appliquer une fonction aux composantes d'une data table : `donnees.t[,fun(),by=]`, `donnees.t[,fun()]`.
Pour agréger des data tables : `donnees.t[], donnees.t[,fun(),by=]`.

Importation et exportation de données

Pour importer des données sous forme de vecteur : `scan()`.
Pour importer des données sous forme de data frame : `read.table(), read.csv(), read.csv2()`.
Pour importer des données de façon robuste sous forme de data table : `fread()`.
Pour exporter des données : `write.table()`, `write.csv()`.

Exercice 1

1. Importer les données stockées dans le fichier `UrgencesHouse.txt` créé dans la Fiche 2 à l'aide de la fonction `read.table`, en les stockant dans un data frame `UrgencesHouse`. Afficher les premières lignes de `UrgencesHouse`.
2. Faire la même chose à l'aide de `read.csv` ou `read.csv2`. Quelles sont les différences entre les 3 fonctions d'importation utilisées ?

Exercice 2

On dispose du jeu de données utilisé dans l'article "Prediction for the 2012 United States presidential election using multiple regression model", écrit par trois chercheurs de l'Université de Delhi et publié en septembre 2012 dans le journal académique *The Journal of Prediction Markets*.

Ce jeu de données, recueilli en vue d'élaborer un modèle statistique de prédiction des résultats des élections présidentielles américaines, contient pour les différentes élections présidentielles américaines entre 1948 et 2008 les variables économiques et politiques suivantes :

- `Year`, l'année de l'élection,
- `Vote`, la proportion des voix obtenues par le parti déjà au pouvoir par rapport au total des voix obtenues par les deux partis ultra-majoritaires, républicain et démocrate,

— `UnemplRate`, le taux de chômage annuel moyen lors du mandat écoulé (en %),
— `Infl`, l'inflation annuelle moyenne lors des trois premières années et demie du mandat écoulé (en %),
— `HealthBudg`, la part du budget fédéral, en proportion du PIB, consacrée aux prestations sociales liées à la santé (en %),
— `GrowthRate`, la croissance mesurée lors des trois premiers trimestres de l'année électorale (en %),
— `BudgSurDef`, l'excès ou le déficit de recettes par rapport aux dépenses du budget prévisionnel, rapportés à ce dernier (en %),
— `PublDebt`, la dette fédérale, mesurée en proportion du PIB (en %),
— `GoldPrices`, le prix moyen de l'or pendant le mandat écoulé (en dollars),
— `OilPrices`, le prix moyen du baril de pétrole brut pendant le mandat écoulé (en dollars),
— `Scandal`, une variable prenant trois valeurs 0,1 ou 2, selon qu'au cours du mandat écoulé, il n'y a eu aucun scandale politique, au moins un scandale n'ayant pas soulevé la question d'un *impeachment*, ou un scandale retentissant ayant mené ou ayant failli mener à un *impeachment*,
— `RatingJune`, une mesure de popularité de l'action présidentielle effectuée au mois de juin de l'année électorale (précédant donc l'élection de quelques mois).

1. Importer le jeu de données disponible dans Cursus sous le nom `Presidential-elections.csv`, dans un tableau de données que l'on intitulera `ywc` (pour *Yes we can !*).
2. Afficher les 5 premières lignes de `ywc`, puis un résumé statistique de `ywc`.
3. Quelles sont les classes des différentes composantes de `ywc` ? Quelles sont les composantes correspondant à des variables quantitatives ? À des variables qualitatives ? À des dates ?
4. Transformer la variable `Scandal` en un facteur à deux niveaux : 0 si le mandat présidentiel n'a connu aucun scandale, 1 sinon. Que se passe-t-il si l'on transforme `Scandal` en un vecteur d'entiers prenant les valeurs 0 et 1 ?
5. Transformer la variable `Scandal` en un facteur prenant les valeurs "non" et "oui".
6. Créer un facteur `classRJ` à 5 niveaux : $[20, 30)$, $[30, 40)$, $[40, 50)$, $[50, 60)$, $[60, 80)$ issu du découpage de la variable `RatingJune` en classes, à l'aide de la fonction `cut`.
7. Regrouper des deux premiers niveaux de `classRJ`. Ajouter le facteur `classRJ` à `ywc`.
8. Transformer la variable `Year` en date. Vérifier.
9. Exporter le data frame `ywc` ainsi transformé sous la forme d'un fichier texte `newywc.txt`, avec un point comme décimale et une tabulation comme séparateur.
10. Retrouver les années des élections présidentielles pour lesquelles la variable `Vote` est minimale, puis maximale. Y a-t-il eu un scandale lors du mandat écoulé ces années là ?
11. Calculer la médiane empirique de `Vote` pour chaque niveau du facteur `Scandal`.
12. Calculer la médiane empirique de `Vote` pour chaque niveau des facteurs `Scandal` et `classRJ` (de façon croisée).
13. Calculer la médiane empirique de `Vote` pour les années au cours desquelles la variable `RatingJune` prend des valeurs inférieures à 50, puis pour celles au cours desquelles `RatingJune` prend des valeurs supérieures à 50.
14. Regrouper les données correspondant aux mandats présidentiels ayant connu un scandale dans un data.frame `scandal`, et celles correspondant aux mandats présidentiels n'ayant pas connu de scandale dans un data.frame `noscandal`, à l'aide de la fonction `split`.

Exercice 3

On dispose de données d'un institut de sondage spécialisé en audiences TV et radio, stockées dans les fichiers `SocioDemo.xlsx` et `Audience.xlsx` disponibles dans Cursus.

1. Ouvrir le fichier `SocioDemo.xlsx`. Préparer ce fichier pour une importation et l'enregistrer au format `.csv`. Importer les données du fichier `.csv` à l'aide de la fonction `read.table` en les stockant dans un data frame `media1`.
2. Importer les données du fichier `SocioDemo.xlsx` directement à l'aide de la fonction `read_excel` du package `readxl`, en les stockant dans un data frame `media2`.
3. Afficher les 5 premières lignes et un résumé statistique de `media1` et `media2`. Quelles sont les classes des composantes de ces data frames ?
4. Importer les données du fichier `Audience.xlsx` à l'aide de la fonction `read_excel` du package `readxl`, en les stockant dans un data frame `audience`.
5. Agréger les data frames `media2` et `audience` dans un nouveau data frame `audmedia`.

Fiche 4 : Réaliser des représentations graphiques

Gestion des fenêtres graphiques

Pour gérer les fenêtres graphiques : `dev.list()`, `dev.cur()`, `dev.set()`, `dev.off()`.
Pour partitionner les fenêtres graphiques : `par(mfrow=c(,))`, `split.screen(c(,))`, `screen()`, `erase.screen()`, `close.screen()`, `layout(matrix(),widths=,heights=)`, `layout.show()` (affichage).

Représentation de variables quantitatives continues

Pour représenter un nuage de points : `plot(x,y)`, `plot(y~x)` ou `plot(y~x, data=)`.
Pour représenter une function `fun` : `plot(x,fun(x),type="l")`, `plot(fun,min,max)`, `curve(fun,min,max)`.
Pour tracer un histogramme : `hist()`, un diagramme cumulatif : `plot(ecdf())`.
Pour tracer un diagramme en boîte : `boxplot()`.
Pour tracer un graphe quantiles-quantiles gaussiens : `qqnorm()`.

Représentation de variables quantitatives discrètes

Pour tracer un diagramme en bâtons : `plot(x,type="h")`, un diagramme cumulatif : `plot(ecdf())`.

Représentation de variables qualitatives

Pour tracer un diagramme en tuyaux d'orgue : `plot(x)`, un diagramme en bande : `barplot(matrix(table(x)))`, un diagramme circulaire : `pie(x)`.

Représentation de données multivariées

Pour comparer deux variables quantitatives continues : `boxplot()`, `qqplot()`.
Pour tracer des graphes multivariés : `scatterplot()`, `pairs()`.

Personnalisation de graphe : paramètres des commandes primaires

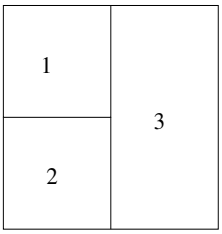
Pour trouver un paramètre graphique : `help(par)`.
Pour changer un paramètre graphique de façon durable : `par()`.
Pour changer le type de tracé d'un graphe : `type`, le symbole utilisé pour les points : `pch`, le type de lignes tracées : `lty`, la couleur : `col`...
Pour ajouter un titre : `main`, un sous-titre : `sub`.
Pour changer les bornes des axes : `xlim, ylim`, annoter les axes : `xlab, ylab`.

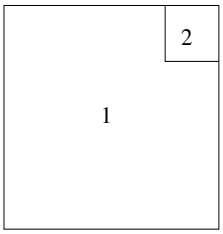
Personnalisation de graphe : commandes graphiques secondaires

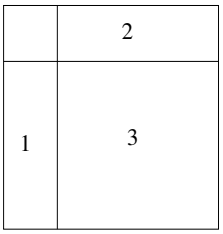
Pour ajouter un nuage de points : `points()`, un graphe en ligne continue : `lines()`, une droite : `abline()`, un segment : `segments()`, une flèche : `arrows()`, un rectangle : `rect()`, un polygône : `polygon()`.
Pour ajouter un titre et un sous-titre : `title(main=,sub=)`, une légende : `legend()`.
Pour ajouter du texte : `text(x,y," ")`, `text(locator(1)," ")`, du texte dans la marge : `mtext()`, une expression mathématique : `text(x,y,expression())`.
Pour ajouter un axe, des graduations sur les axes : `axis()`, `rug()`.
Pour repérer des points et leur accoler du texte : `identify(x,y," ")`.

Exercice 1 : Partitionner une fenêtre graphique

Partitionner, de différentes façons, la fenêtre graphique active en trois, deux, trois parties de la forme :







Exercice 2 : Personnaliser un graphique

Les graphes tracés dans cet exercice seront enregistrés dans un deuxième temps dans un fichier graphique.

Partitionner la fenêtre graphique active en deux sous-fenêtres. Dans les deux sous-fenêtres, tracer le graphe de la fonction $f(x) = \sin(2x) + 2e^{-0.4x} + 1$ sur $[0, 4\pi]$ représentant un signal (théorique) reçu par un radar en fonction du temps avec les options suivantes : le graphe tracé en noir en ligne continue, l'axe des abscisses gradué de 0 à 5 de 1 en 1 puis de 2 en 2, l'axe des ordonnées gradué de 0 à 5 de 1 en 1.

1. Dans la première sous-fenêtre :
a) Ajouter l'annotation de l'axe des abscisses, le titre "Signal radar déterministe théorique" dans une font différente et, près de la courbe (à l'aide de la fonction `locator`), l'expression mathématique :

$$y = \sin(2x) + \frac{2}{e^{0.4x}} + 1.$$

b) Tracer en pointillés rouges les graphes des fonctions $g(x) = 2\exp(-0.4x) + 2$ et $h(x) = 2\exp(-0.4x)$.
c) Repérer le minimum global de la fonction, le maximum global. Annoter ces points "Min global", "Max global".
d) Tracer en rouge la tangente à la courbe représentative de f en un point de son choix. Préciser son équation sur le graphe (en rouge).
2. Dans la deuxième sous-fenêtre :
a) Ajouter l'annotation de l'axe des abscisses, le titre "Signal radar observé" dans la même font que celle du titre de la première partie.
b) Tracer un nuage de points aléatoires $(y_1, \ldots, y_{10})$ représentés par des carrés et dont la loi est donnée par $y_i = \sin(2x_i) + 2e^{-0.4x_i} + 1 + \varepsilon_i$, où les x_i sont des points de $[0, 4\pi]$ régulièrement espacés, les ε_i des observations d'une variable aléatoire de loi $\mathcal{N}(0, 0.5)$.
c) Répérer graphiquement les points les plus éloignés de la courbe à l'aide de la fonction `identify`.

Exercice 3 : Représenter des données

1. Importer les données du fichier `ronfle.txt`. Ces données sont issues d'une étude de la population angevine par le CHU d'Angers, qui s'est intéressé à la propension à ronfler d'hommes et de femmes. Elles correspondent à un échantillon de 100 patients, pour lesquels on a considéré les variables suivantes :
   - `AGE` : âge du patient en années,
   - `POIDS` : poids en kg,
   - `TAILLE` : taille en cm,
   - `ALCOOL` : consommation d'alcool du patient en nombre de verres bus par jour (en équivalent verre de vin rouge),
   - `SEXE` : sexe du patient (F=femme, H=homme),
   - `RONFLE` : diagnostic de ronflement (O=ronfle, N=ne ronfle pas),
   - `TABA` : comportement tabagique (O=fumeur, N=non fumeur).
2. Transformer les variables qualitatives en facteurs si besoin.
3. On s'intéresse aux observations de la variable `AGE`.
a) Représenter ces observations.
b) Tracer un histogramme de fréquences, un histogramme de fréquences relatives des observations, en personnalisant les graphiques de façon à les rendre le plus explicite possible, et en faisant varier les classes sur lesquelles sont calculées les fréquences. Commenter.
c) Tracer un diagramme cumulatif des observations.
d) Tracer un diagramme en boîte des observations.
e) Tracer un graphe quantiles-quantiles gaussiens des observations. Ajouter la droite de Henry. Commenter.
4. On s'intéresse aux observations de la variable `ALCOOL`. Tracer un diagramme en bâtons, puis un diagramme cumulatif de ces observations.
5. On s'intéresse maintenant aux observations de la variable `RONFLE`. Tracer un diagramme en tuyaux d'orgue, un diagramme en bande, puis un diagramme circulaire de ces observations.
6. Comparer graphiquement l'âge, puis le poids, des ronfleurs avec celui des non-ronfleurs.
7. Trouver des représentations graphiques permettant de comparer le nombre de ronfleurs chez les femmes à celui chez les hommes.
8. Représenter le nuage de points des observations de la variable `POIDS` en fonction de la variable `TAILLE`. Repérer des points atypiques et reporter sur le graphe l'identifiant des patients correspondants, puis leur statut de ronfleur.

Fiche 5 : Programmer avec R

Fonctions

Pour définir une fonction : `myfun=function(x,y){instructions; return(nom=)}`, et l'utiliser : `myfun(x,y)`.
Pour définir un nouvel opérateur binaire : `"%!%"=function(x,y){instructions}`, et l'utiliser : `x%!%y`.

Boucles et structures de contrôle

Pour faire une boucle for : `for (i in vecteur) {instructions}`.
Pour faire une boucle while : `while (condition) {instructions}`.
Pour faire une boucle repeat : `repeat {instructions; if (condition) break}`.
Pour utiliser une structure de contrôle : `if (condition) {instructions} else {alternatives}`.

Exercice 1 : Définir des fonctions

1. Programmer une fonction `mymedian` qui calcule la médiane empirique d'un n échantillon. Si $X = (X_1, \dots, X_n)$ est n un échantillon, $(X_{(1)}, \dots, X_{(n)})$ sa statistique d'ordre, la médiane empirique de X est définie par : $\text{med}(X) = X_{(k+1)}$ si $n = 2k + 1$, $\text{med}(X) = (X_{(k)} + X_{(k+1)})/2$ si $n = 2k$ ($k \in \mathbb{N}$). Comparer avec la fonction `median` pré-programmée.
2. Programmer une fonction `estvar` qui prend en entrée un vecteur numérique x et un argument `biased` (valeur par défaut `FALSE`), et qui donne en sortie l'estimateur empirique de la variance calculé sur x, sans biais si `biased` est `FALSE`, biaisé si `biased` est `TRUE` :
   — en utilisant une structure de contrôle,
   — en utilisant les opérateurs sur les vecteurs logiques.

Exercice 2 : Utiliser des boucles et des structures de contrôle

1. Définir des fonctions `mymean` et `myvar` qui font la même chose que `mean` et `var`, sans utiliser les fonctions mathématiques pré-programmées.
2. Définir une fonction `mywhich` qui fait la même chose que `which`, sans utiliser les fonctions mathématiques pré-programmées.
3. Définir des fonctions `mymin` et `mywhich.min` qui font la même chose que `min` et `which.min` pour un vecteur numérique, sans utiliser les fonctions mathématiques pré-programmées.
4. Définir une fonction `myapply` qui fait la même chose que `apply` pour une matrice réelle et une fonction numérique donnée.
5. Définir une fonction `mytapply` qui fait la même chose que `tapply` pour un vecteur numérique, un facteur et une fonction numérique donnés.

Fiche 6 : Illustrer les théorèmes de convergence en probabilités

---

Fonctions relatives aux lois usuelles

Densité (ou fonction de masse) : `dlaw()`, fonction de répartition : `plaw()`, quantile : `qlaw()`, où `law` désigne une loi de probabilité usuelle :

Loi	`law`	Paramètres
Binomiale	`binom`	`size,prob`
Poisson	`pois`	`size,lambda`
Uniforme continue	`unif`	`min,max`
Gaussienne	`norm`	`mean,sd`
Exponentielle	`exp`	`rate`
Student	`t`	`df`
Chi Deux	`chisq`	`df`
Fisher	`f`	`df1,df2`

Simulation de variables aléatoires

Pour simuler l'observation d'un n-échantillon de la loi `law` : `rlaw(n=,)`.

Exercice 1 : Loi discrète

On lance dix fois de suite une pièce équilibrée et on note X la variable aléatoire correspondant au nombre de fois où l'on a obtenu un pile sur les dix lancers.

1. Quelle est la loi de X ?
2. Calculer $P(X = 7)$ à l'aide de la fonction `dbinom` ou de la fonction `pbinom`.
3. Représenter graphiquement la fonction de masse de X à l'aide de la fonction `plot`.
4. Représenter graphiquement la fonction de répartition de la loi de X.
5. Lire sur le graphique précédent la médiane de la loi de X, ainsi que son quantile de niveau 0.95. Retrouver ces résultats numériquement.

Exercice 2 : Loi continue

1. Représenter graphiquement la densité et la fonction de répartition de la loi gaussienne centrée réduite $\mathcal{N}(0, 1)$ sur l'intervalle $[-4, 4]$.
2. Ajouter l'axe des abscisses, et une droite d'équation $y = 0.95$. En déduire graphiquement une valeur approchée du quantile de niveau 0.95 de la loi $\mathcal{N}(0, 1)$.
3. Calculer numériquement ce quantile noté q, et ajouter sur le graphe précédent une droite d'équation $x = q$.
4. Combien valent les aires sous la courbe représentative de la densité à gauche et à droite de la droite d'équation $x = q$?

Exercice 3 : Simuler des variables aléatoires

1. Simuler l'observation `obs` d'un échantillon $(X_1, \ldots, X_{100})$ de la loi uniforme sur $[0, 1]$.
2. Tracer un histogramme de `obs` et superposer le graphe de la densité de la loi uniforme sur $[0, 1]$.
3. Reprendre les questions 1 et 2. Que remarque-t-on ? Reprendre les mêmes questions, mais en exécutant au préalable à chaque fois : `set.seed(12345)`. Que remarque-t-on ?
4. Tracer un diagramme cumulatif de `obs` et superposer le graphe de la fonction de répartition de la loi uniforme sur $[0, 1]$.

5. Reprendre les questions précédentes mais pour la loi gaussienne centrée réduite.
6. Comment peut-on vérifier de façon empirique la propriété suivante : si U et V sont des variables aléatoires indépendantes de loi uniforme sur $[0,1]$, alors $\sqrt{-2\ln U}\cos(2\pi V)$ suit une loi gaussienne centrée réduite ?

Exercice 4 : Illustrer les lois des grands nombres

1. Rappeler la loi forte des grands nombres, et l'exprimer pour une loi exponentielle de paramètre $1/2$.
2. Simuler l'observation $(x_1, \ldots, x_{1000})$ d'un échantillon de taille 1000 de la loi exponentielle de paramètre $1/2$.
3. Représenter graphiquement la suite $\left(\sum_{i=1}^{n} x_i/n\right)_{n=1\ldots1000}$.
4. Ajouter au graphe précédent une droite d'équation $y = 2$.
5. Reprendre plusieurs fois les questions précédentes. Que constate-t-on ?
6. Soit $(X_i)_{i\in\mathbb{N}}$ une suite de variables aléatoires i.i.d. de loi exponentielle de paramètre $1/2$, et $Y_n = \overline{X_n}$, avec $\overline{X_n} = \sum_{i=1}^{n} X_i/n$. Rappeler la loi faible des grands nombres, et l'exprimer pour une loi exponentielle de paramètre $1/2$ à l'aide de Y_n.
7. Écrire une fonction `simu` qui prend en entrée n et qui donne en sortie la simulation d'une observation d'un 2000 échantillon de la loi de Y_n. À partir de cette fonction, écrire une nouvelle fonction `proba` qui prend en entrée un entier n, un réel ε, et qui donne en sortie une estimation de $P\left(|Y_n - 2| > \varepsilon\right)$.
8. Pour différentes valeurs de $\varepsilon > 0$, représenter graphiquement `proba`(n, ε) en fonction de n. Que constate-t-on ?

Exercice 5 : Illustrer le théorème central limite

Soit $(X_i)_{i\in\mathbb{N}}$ une suite de variables aléatoires i.i.d. de loi exponentielle de paramètre $1/2$, et $Z_n = \sqrt{n}\left(0.5\overline{X_n} - 1\right)$, avec $\overline{X_n} = \sum_{i=1}^{n} X_i/n$.

1. Rappeler le théorème central limite et l'exprimer pour la loi exponentielle de paramètre $1/2$ à l'aide de Z_n.
2. Simuler l'observation d'un 2000 échantillon de la loi de Z_1. Tracer l'histogramme de cette observation et superposer le graphe de la densité de la loi $\mathcal{N}(0,1)$.
3. Dans des fenêtres graphiques différentes, reprendre la question précédente pour Z_2, Z_3, Z_{10} et Z_{100}. Que constate-t-on ?
4. Dans une nouvelle fenêtre graphique, tracer en rouge la fonction de répartition de la loi gaussienne centrée réduite. Dans la même fenêtre graphique, tracer les diagrammes cumulés des simulations de Z_1, Z_2, Z_3, Z_{10} et Z_{100}. Que constate-t-on ?