Université Joseph Fourier **Frédérique Leblanc**
L2/STA230 **leblanc@imag.fr**

TP1 : Prise en main de R - Description d'une variable discrète ou qualitative

Objectifs : *Commencer à utiliser le logiciel comme une calculatrice et savoir faire quelques manipulations élémentaires sur une série de données rangées dans un vecteur. Savoir faire une première étude descriptive de données discrètes à partir d'un tableau en effectifs ou d'une série de données brutes et réaliser des graphiques représentatifs d'une variable qualitative ou quantitative discrète.*

R est un logiciel libre développé par une très large communauté scientifique et basé sur "des objets". Il ne s'agit pas du tout ici de présenter académiquement les fonctionnalités et la structure de ce logiciel mais de les appréhender pas à pas à l'occasion des besoins pour traiter les problèmes statistiques qui nous intéressent dans ce cours. Vous trouverez sur la "toile" un grand nombre de cours ou "tutorials" et bien sur vous avez toujours à votre disposition l'aide en ligne de R que l'on pourra appeler avec la fonction `help(·)`. Avant toute chose créez dans votre environnement (votre "home" ou dossier "documents") un répertoire STA230 et un sous-répertoire TPSTA230 dans lequel vous rangerez tous vos documents de Tps : fiches de Tps .pdf, scripts .R, graphiques .pdf, ...

1 Premiers pas avec R et l'environnement R Studio :

Si R (R.base) et Rstudio sont installés sur votre machine une icône R est disponible sur le bureau (sinon lancez R à partir de la listes des programmes et au pire installez vous même R et Rstudio en allant par exemple sur les nombreux sites proposés par un moteur de recherche à qui vous demandez de chercher R). Cliquer sur l'icône R et l'environnement de R studio s'ouvre et se présente en quatre fenêtres. La fenêtre située en bas à gauche est celle où l'on tape une ligne de commande que l'on peut exécuter avec le retour chariot du clavier (Entrée $\hookleftarrow$). Dans la fenêtre en haut à gauche on charge où on écrit un script R qui est l'ensemble des commandes que l'on souhaite sauvegarder, dans la fenêtre en bas à droite s'afficheront les graphiques et dans celle du haut à droite par défaut s'affiche l'historique de la dernière session. Nous allons dans un premier temps tester une petite ligne de commande, faire un script, un graphique élémentaire, sauvegarder ce que l'on veut garder, sortir puis relancer le logiciel et retrouver ce qui a été fait dans la session que vous venez de fermer. Pour se simplifier la vie nous allons effectuer cette session dans le répertoire de travail TPSTA230. Il suffit par exemple d'aller dans l'onglet session de la barre supérieure du menu et choisir ensuite Set Working Directory (Répertoire de Travail). Dans la section suivante sont décrites `dans ce format de caractères` les commandes R à taper (fenêtre en bas à gauche) après le symbole >. Noter également que le logiciel différencie majuscules et minuscules. Ainsi l'objet `y` n'est pas le même que l'objet `Y`.

1. *Créer* : la suite de données $(1, 2, 3, 4, 5)$ en tapant la commande puis en l'exécutant avec Entrée $\hookleftarrow$:
   `c(1,2,3,4,5)`
   `c(.)` est la fonction de concaténation qui "range" les unes à la suite des autres les valeurs ou caractères tapés les uns à la suite des autres et séparés par des virgules. L'objet ainsi produit est un vecteur colonne. C'est un objet qui contient une colonne et dans cette colonne se trouvent les valeurs que vous y avez saisies. La réponse de R est [1] 1 2 3 4 5 qui vous indique que l'objet ainsi créé est constitué d'une colonne et que la colonne numéro 1 contient la suite des valeurs $(1, 2, 3, 4, 5)$ (par souci d'ergonomie ces valeurs sont listées sur une seule ligne).

2. *Affecter* : la suite précédemment créée dans un vecteur nommé `x` en exécutant la commande : `c(1,2,3,4,5)`→ `x` (ou bien `x`←`c(1,2,3,4,5)`)
   R ne retourne rien sinon le traditionnel début de ligne > pour vous redonner la main. C'est normal et si vous voulez vérifier que le vecteur `x` existe bien quelque part et contient les informations saisies il vous suffit de l'appeler en tapant seulement `x` suivi de Entrée ↩. A présent que vous avez appris à fabriquer un vecteur refaites l'opération pour créer le vecteur `y` contenant la suite des valeurs $(2, 4, 6, 8, 10)$.

3. *Ecrire son script* : Au fur et à mesure de votre travail, vous le conserverez dans un script R qui sera sauvegardé comme fichier : "NomFichier.R"; par exemple ScriptEssai.R. Pour cela aller dans la barre de menu supérieure choisir File puis New puis Rscript. Ensuite copier et coller les lignes de commande déjà testées à savoir :
   `c(1,2,3,4,5)`→ `x`
   `x`
   `c(2,4,6,8,10)`→ `y`
   `y`
   Il est vivement recommandé de commenter un script en ajoutant en première position préfixée par le signe # Mon commentaire : blabla...
   Si vous souhaitez interrompre votre session, avant de quitter R-studio il faut :

4. *Sauvegarder votre script R* : En activant la fenêtre qui le contient (celle du haut à gauche : le curseur doit clignoter en noir) puis en allant de nouveau dans la barre de menu supérieure dans File puis Save as et en sélectionnant le répertoire de travail TPSTA230 que vous avez créé précédemment. Ou encore vous pouvez quitter R avec la commande `q()` et on vous propose alors de sauver votre script ou votre espace de travail. Prenez l'habitude de sauvegarder votre script et vos graphes à la fin de chaque séance.

5. *Un premier graphique :* La fonction la plus standard pour créer un graphique est la fonction `plot()` que nous allons tester avec la commande suivante :
   `plot(x,y)`
   Dans la fenêtre en bas à droite s'affiche le graphique. Il représente les points (il y en a 5) de coordonnées données dans `x` et `y` (abscisse x et ordonnée y) et comme aucune option n'a été paramétrée il choisit par défaut la couleur noire, le symbole "circle" et un nuage de points non reliés entre eux. On peut bien sur enrichir ce graphique à l'aide des options disponibles pour la fonction `plot()`. En essayant par exemple :
   `plot(x,y,main="y selon x",pch=1,col=1,type="p",xlab="abscisse",ylab="ordonnée")`
   Les options de la ligne précédentes prennent de nombreux arguments, par exemple essayer de remplacer l'argument `"p"` dans `type="p"` par l'argument `"l"` ou encore l'argument `"b"`, de changer le symbole représentant les points avec `pch=...` ou la couleur avec `col=...` Il serait fastidieux ici de décrire toutes les options possibles pour faire un plot, d'autant que cela est très bien documenté avec l'aide en ligne sur la fonction `plot()` accessible en tapant la commande :
   `help(plot)`
   Puis, pour aller ensuite un peu plus loin, chercher ce que fait, par exemple, l'option `pch` documentée dans graphical parameters. Pour finir amusez-vous un peu à modifier les arguments des options `col`, `main`, `type`,... et observez ensuite votre graphique se modifier. N'oubliez pas d'utiliser si besoin l'aide graphique en ligne.

6. *Exercice* : Réaliser un graphique "en escalier" (constant par morceaux et toujours avec les vecteurs `x` et `y`) de couleur rouge intitulé "Mon premier escalier" avec la légende "numéro de marche" en x et "hauteur à partir du niveau 0 en dm" en y).

7. *Sauvegarder un graphique* : Activer la fenêtre graphique (en cliquant dedans : aucun curseur ne doit clignoter dans les trois autres fenêtres) puis sélectionner Export puis le format choisi pour le fichier

(pdf, png,...) et le nom de fichier, (par exemple PremierGraphique.pdf). Attention, en format pdf à l'option landscape (paysage) ou portrait choisie.

2 Calculer avec R : manipulations de base sur les vecteurs

Il s'agit là d'expérimenter les commandes élémentaires que l'on doit savoir faire avec un ou plusieurs vecteurs numériques de même longueur. On travaillera toujours avec nos deux vecteurs x et y.

Avec un seul vecteur :

1. *Multiplier ou diviser par un scalaire* : Par exemple pour produire le résultat du calcul $\frac{1}{5}(1,2,3,4,5)$ (resp. du calcul $5*(1,2,3,4,5)$) on exécutera :
   `x/5` (resp. `x*5`)

2. *Ajouter un scalaire* : par exemple pour calculer $(1+2, 2+2, 3+2, 4+2, 5+2)$ on exécute :
   `x+2`

3. *Sommer* : on obtient la somme des éléments de x avec la fonction `sum()` :
   `sum(x)`

4. *Sommer en cumulant* : on obtient les sommes cumulées des éléments de x avec la fonction `cumsum()`:
   `cumsum(x)`

5. *Calculer la racine* : avec la fonction `sqrt()`
   `sqrt(x)`

6. *Calculer une puissance* : avec le symbole ^, par exemple :
   `x ^ 3`

7. *Agrandir* (ajouter des éléments à la liste décrite par un vecteur) : si on veut produire un vecteur x' constitué de x complété par une sixième valeur 6 par exemple on utilise la fonction de concaténation `c()` :
   `c(x,6)`
   Si on veut compléter x par $(1,1,1,1,1)$ on peut utiliser l'une ou l'autre des commandes suivantes :
   `c(x,1,1,1,1,1)`
   `c(x,rep(1,5))`
   `c(x,c(1,1,1,1,1))`
   La fonction `rep()` permet de créer une suite d'éléments de même valeur (premier argument) et de longueur (deuxième argument) fixées. Essayer également de manipuler la fonction `seq()`, bien utile en pratique. Tester par exemple les commandes :
   `seq(1,10,2)`
   `seq(from=1,to=10,by=2)`
   `seq(from=1,to=2,by=.2)`

8. *Extraire :* Si on souhaite extraire les éléments de `x` placés en position 2 et 4 on exécute :
   `x[c(2,4)]`
   Si on souhaite extraire les éléments de x en positions 1 à 4 (resp. >2, resp. $\neq 2$, resp. $=2$, resp. (compris dans $]2,4]$) ....: on exécute l'une des commandes suivantes :
   `x[1:4]`
   `x[(x>2)]`
   `x[(x!=2)]`
   `x[(x==2)]`
   `x[(x>2)&(x<=4)]`

Avec plusieurs vecteurs

1. *Ajouter des vecteurs* : pour ajouter (ou soustraire) `x` et `y` (l'élément en position i du résultat est la somme des éléments en position i, de `x` et de `y`) on exécute : `x+y` ou `x-y`

2. *"Multiplier des vecteurs"* : pour multiplier (ou diviser) deux à deux les éléments des vecteurs `x` et `y` (attention il ne s'agit pas de la multiplication usuelle matricielle entre un vecteur ligne et un vecteur colonne ou entre un vecteur colonne et un vecteur ligne. Nous y reviendrons selon le besoin) :
`x*y` ou `x/y`

3. *Réunir deux vecteurs* : dans un même vecteur colonne avec la fonction `c()` ou dans une matrice à cinq lignes et deux colonnes (une pour `x` et l'autre pour `y`) avec la fonction `cbind()` ou dans une matrice à 2 lignes et cinq colonnes avec `rbind()`. Tester les commandes:
`cbind(x,y)`
`rbind(x,y)`
`t(cbind(x,y))`
Que fait la fonction `t()` ?

3 Un petit exercice de statistiques avec R : Ex 1.1 du workbook

Vous avez à présent tous les outils et commandes permettant d'effectuer les calculs demandés dans Ex 1.1 et corrigés dans vos fiches de Td.

1. Avant tout : quelle est la variable observée ? Quelles sont les données collectées (taille de l'échantillon ? individus observés ?) De quelle nature est la variable en question que nous conviendrons d'appeler X ?

2. Créer un vecteur que l'on nommera `ModalitesX` et contenant la série Pannes donnée dans l'exercice. Faire de même avec la série Semaines que l'on affectera au vecteur nommé `EffectifsX`.

3. Calculer le nombre hebdomadaire moyen de pannes observées sur les 100 semaines (moyenne empirique de l'échantillon des 100 observations de la variable nombre hebdomadaire de pannes et décrit dans un tableau en effectifs) et l'affecter à `Xmoy`.

4. Calculer la variance empirique de l'échantillon (notée s^2 dans le cours) et sa variance empirique corrigée (notée s'^2 dans le cours et définie par $s'^2 = \frac{n}{n-1}s^2$) et les affecter à `Xvar` et `Xvarc`.

5. Calculer l'écart-type empirique de l'échantillon s et son écart-type empirique corrigé s'.

6. Calculer les fréquences des modalités de la variable X et les envoyer dans un vecteur nommé `freqX`. Calculer la somme de ses éléments. Commentaire ?

7. Représenter graphiquement la distribution observée avec un diagramme en barres en exécutant :
`barplot(freqX, names.arg=ModalitesX, col=4,main="fréquences observées des nb. hebd. de pannes", ylab="fréquences",xlab="nombre hebdomadaire de pannes")`

8. Calculer les fréquences cumulées des modalités et les envoyer dans un vecteur nommé `cumfreqX` et calculer son maximum avec la fonction `max()`. Commentaire ?

9. A l'aide de la lecture simultanée des vecteurs `cumfreqX` et `ModalitesX`, indiquer les trois quartiles de la distribution de la variable X.

10. Quelle représentation graphique usuelle permet-elle de représenter les quartiles et leurs positions relatives ? La représentation de cet objet ainsi que celle de la fonction de répartition empirique (qui dans le cas d'une variable discrète est une fonction en escalier) étant très facile à réaliser sous R avec des données brutes nous le ferons dans la partie suivante.

4 Le même que précédemment avec données brutes

Dans l'exercice précédent ne vous ont pas été données les 100 observations de la variable X mais un tableau en effectifs indiquant les nombres de fois où dans l'échantillon observé (100 valeurs entières $\in \{0, 1, ..., 7\}$)) on a observé chaque modalité (de 0 à 7) de la variable X. Nous allons refaire les questions du paragraphe précédent à l'aide de l'échantillon des données brutes soit la liste des 100 valeurs observées pour X.

1. Créer le vecteur `X` qui contiendra la liste des 100 observations à l'aide de la fonction `rep()` en exécutant :
   `X<-c(rep(0,42),rep(1,35),rep(2,11),rep(3,5),rep(4,3),rep(5,2),6,7)`

2. En utilisant R comme une calculatrice et avec les données brutes, retrouver la valeur moyenne, la variance et la variance corrigée de l'échantillon observé. Comparer ces valeurs avec celles précédemment obtenues. Appliquer ensuite les fonctions `mean()`, `var()` au vecteur X. Que calcule la fonction `var()`?

3. Calculer les effectifs observés de la variable avec la fonction `table()` puis diviser par `length(X)` pour obtenir les fréquences.

4. Retrouver le diagramme en barre précédemment réalisé en utilisant les données brutes.

5. Calculer les déciles (puis les quartiles) à l'aide des commandes suivantes et réaliser le boxplot de X
   `quantile(X,seq(0.1,0.9,0.1))`
   `boxplot(X)`

6. Pour calculer les fréquences cumulées on utilise la commande :
   `cumsum(table(X))/length(X))`
   puis pour les représenter on exécute :
   `plot(ecdf(X),main="fonction de répartition empirique",xlab="nombre de pannes",`
   `ylab="fréquences cumulées")` (*ecdf pour empirical cumulative distribution)*