Travaux Pratiques N°2

Feryal WINDAL

Septembre 2021

Résumé des instructions

Commande	Description
`head("data")`	Affiche les 6 premières lignes de `data`.
`tail("data")`	Affiche les 6 dernières lignes de `data`.
`str("objet")`	Affiche la structure de `objet`.
`levels("factor")`	Affiche les niveaux de `factor`.
`table("vector")`	Affiche la table des effectifs de `vector`.
`cumsum("vector")`	Cumule l'une après l'autre les coordonnées de `vector`.
`sum("vector")`	Calcule la somme des coordonnées de `vector`.
`prop.table("vector")`	Affiche la table des fréquences de `vector`.
`min("vector")`	Calcule la valeur minimale de `vector`.
`max("vector")`	Calcule la valeur maximale de `vector`.
`tapply(x,grp,fonct)`	Applique la fonction `fonct` aux groupes constitués à partir du vecteur `x` grâce aux modalités du facteur `grp`.
`as.data.frame("matrix")`	Transforme `matrix` en un tableau de données.
`rownames("matrix")` `rownames("dframe")`	Affiche le nom des lignes de `matrix`. Affiche le nom des lignes de `dframe`.
`colnames("matrix")` `colnames("dframe")`	Affiche le nom des colonnes de `matrix`. Affiche le nom des colonnes de `dframe`.
	Commandes pour les représentations graphiques
`plot(x,y)`	Trace le nuage de points de `y` en fonction de `x`.
`barchart(...)`	Trace un diagramme.
`barplot(x)`	Trace un diagramme en bâtons de `x`.
`pie(x)`	Trace un diagramme circulaire de `x`.
`hist()`	Trace un histogramme, avec des options pour définir les classes : • `nclass=...` pour fixer un nombre de classes • `br=c(...,...,...)` pour donner les bornes des classes. Attention de ne pas utiliser `nclass` et `br` en même temps.
`boxplot(x)` `boxplot(x~fac)`	Trace une boîte à moustaches de `x`. Trace une boîte à moustaches pour chaque groupe de valeurs de `x` défini par un niveau de `fac`.

	Commandes pour les caractéristiques de position
`summary(vector)` `summary(dframe)`	Donne certaines caractéristiques de position de `vector`. Donne certaines caractéristiques de position pour chaque colonne de `dframe`.
`mean(vector)`	Calcule la moyenne de `vector`.
`range(vector)`	Calcule les valeurs minimale et maximale de `vector`.
`median(vector)`	Calcule la médiane de `vector`.
`quantile(vector,probs,type)`	Calcule les quantiles de `vector` de niveau les valeurs de `probs` de type « `type` ». Par défaut « `type=7` » et `probs=c(0,.25,.5,.75,1)`.

Commande	Description
	Commandes pour les caractéristiques de dispersion
`var(vector)`	Calcule la variance corrigée de `vector`.
`sd(vector)`	Calcule l'écart-type corrigé de `vector`.
	Commandes pour les caractéristiques de forme
`kurtosis(vector)`	Calcule l'aplatissement de `vector`.
`skewness(vector)`	Calcule l'asymétrie de `vector`.
	Commandes pour manipuler les données
`make.groups(x,y,...)`	Empile les vecteurs `x`, `y`,... dans un tableau de données en créant un facteur pour identifier chacun des groupes.

Fonction `factor`

In this exercise, you will discover how the factor function works, which was mentioned in the course reminders.
On three varieties of apples rated 1, 2 and 3, the juiciness of each apple is raised. Juiciness is an index between 0 and 10. There are four apples per variety that have been tested. Variety 1 is Golden Delicious, Variety 2 is Calville Apple and Variety 3 is Belle de Boskoop. Presumably, the question you might ask yourself would be : What is the juiciest variety of apple ? You will not attempt to answer this question here. This is because it is an application of a statistical technique known as analysis of variance that you do not yet know. The goal of this exercise is to show you how to use the factor function. The results obtained are listed in the following table :

Apple variety	Juiciness	Apple variety	juiciness
1	4	2	7
1	6	2	6
1	3	3	6
2	7	3	5

1. Enter the data under R by introducing two variables :
   — a first variable that you will note `Variety`
   — and a second variable that you will note `Juicy`
   Át the end of this operation, build a `data.frame` whose name is `Apple`.
2. Give the structure of the dataset `Apple` that you just built in the previous question. What do you see ? We must therefore transform the variable `Variety` into a `factor`.
3. To transform a numeric or integer type vector, you can use the `factor` function. So to transform the variable `Variety` which is currently in `numeric` mode, you type the following command line :

   ```
   > Variety <-factor (Variety)
   ```

   then

   ```
   > Apples <-data.frame(Variety, Jutosite)
   > rm(Variety)
   > rm(Jutosite)
   ```

   What is the nature of the `Apples` dataset ? What are the modes of the two variables that make up the `Apples` dataset ?

Remark :

`rm` for "remove"

4. You could have done otherwise. This second way is much faster and you are encouraged to use it as soon as you know a variable in your dataset is a `factor`. Type the following command lines :

```
> Variety <-factor(c(rep(1,4), rep(2,4), rep(3,4)))
> Jutosite <-c(4,6,3,5,7,8,7,6,8,6,5,6)
> Apples <-data.frame(Variety, Jutosite)
```

   What do you get ? Do you have the same result as before, ie the same structure for the `Apples` dataset ?

5. You are advised, at least in the early stages of learning statistics, not to use numbers for your `factor` levels, but rather letters. For that, you will use the option labels in the function factor. You will give a label to the numeric values 1, 2 and 3, namely 1 becomes `V1`, 2 becomes `V2` and 3 becomes `V3`, V for `Variete`. To do this, type the following command lines :

```
> Variety <-factor(c(rep(1,4),rep(2,4),rep(3,4)),labels=c("V1","V2","V3"))
> Jutosite <-c (4,6,3,5,7,8,7,6,8,6,5,6)
> Apples <-data.frame (Variety, Jutosite)
```

   What do you get ? There is something that has changed. What Can you say ?

6. Finally, there is a `as.factor` function which achieves the same result. Type the following command lines :

```
> Variety <-as.factor(c(rep(1,4),rep(2,4),rep(3,4)))
> Jutosite <-c(4,6,3,5,7,8,7,6,8,6,5,6)
> Apples <-data.frame (Variety, Jutosite)
```

   Double check that you get the same result that is expected.

7. Calculate the means for each of the groups defined by the variable `Variete` using the `tapply` function :

```
> tapply(Jutosite, Variete, mean)
```

   Proceed in the same way to obtain the standard deviation, the quantiles or apply the function `summary` to each of the groups defined by the factor `Variete`.

How to group the data ?

During one summer, a gardener collected "beans" of four different species of plants on his land. He noted on each of the "beans" the mass, size and species of the "bean". He numbered each of the "beans". Download the `"BioStatR"` library.

```
>library(BioStatR)
```

To display the `Measures` dataset, you can simply type the following command line :

```
> Mesures
```

You will be interested here in the variable `mass` of the dataset `Measures`

1. What function is used to describe this data set ?
2. Use the `plot` and `ggplot` function to represent the `size` variable against the `mass` variable. Configure for the two functions, the title of the graph and the axes.
3. Group the data of the `mass` variable into 5 classes using the `breaks = 5` option of the `hist` function.
4. Groupez les données en utilisant les classes suivantes $[0;5],]5;10],]10;15]$, $]15;20],]20;50]$
5. Comparez le résultat obtenu avec (interprétez) :

   ```
   > brk<-c(0,5,10,15,20,50)
   > table(cut(Mesures$masse,brk))
   > data.frame(table(cut(Mesures$masse,brk)))
   ```

6. Si vous cherchez à créer des groupes dont les effectifs sont équilibrés, vous pouvez par exemple utiliser la fonction `cut2` de la bibliothèque `Hmisc`. Après avoir téléchargé et installé cette bibliothèque, commentez les lignes de code suivantes et en particulier le rôle des options `g` et `m`.

   ```
   > library(Hmisc)
   > brk<-c(0,5,10,15,20,50)
   > res<-cut2(Mesures$masse,brk)
   > table(res)
   > table(cut2(Mesures$masse,g=10))
   > table(cut2(Mesures$masse,m=50))
   ```

Fonctions `attach` et `detach`

Le but de cet exercice est d'introduire les fonctions `attach` et `detach` qui permettent d'accéder plus facilement aux variables contenues dans un tableau de données.

1. Exécutez les deux lignes de commande l'une après l'autre :

   ```
   > head(Mesures$masse)
   > head(masse)
   ```

   Que constatez-vous ?

2. Vous allez utiliser les deux fonctions `attach` et `detach`. Pour cela, tapez les deux lignes de commande :

   ```
   > attach(Mesures)
   > head(masse)
   ```

   puis les suivantes :

   ```
   > detach(Mesures)
   > head(masse)
   ```

   Que se passe-t-il maintenant ?