Introduction à R

Statistiques descriptives

Pascal Bessonneau

06/2015

Les statistiques de base

Pour avoir un aperçu à partir des indicateurs les plus courants, on peut utiliser la fonction *summary*.
Cette fonction donne un résumé de la variable adaptée au type de la variable.

Les statistiques de base

```
> summary(iris)

##   Sepal.Length    Sepal.Width
##  Min.   :4.300   Min.   :2.000
##  1st Qu.:5.100   1st Qu.:2.800
##  Median :5.800   Median :3.000
##  Mean   :5.843   Mean   :3.057
##  3rd Qu.:6.400   3rd Qu.:3.300
##  Max.   :7.900   Max.   :4.400
##   Petal.Length    Petal.Width
##  Min.   :1.000   Min.   :0.100
##  1st Qu.:1.600   1st Qu.:0.300
##  Median :4.350   Median :1.300
##  Mean   :3.758   Mean   :1.199
##  3rd Qu.:5.100   3rd Qu.:1.800
##  Max.   :6.900   Max.   :2.500
##        Species
##  setosa    :50
##  versicolor:50
##  virginica :50
##
```

Les statistiques de base

Si on veut extraire ces informations, on peut utiliser la fonction *sink*.
Cette fonction permet de rediriger les résultats affichés à l'écran vers un fichier.

```
sink("resultats.txt")
summary(iris)
sink()
```

Les statistiques de base

Mais des fonctions sont disponibles pour chaque indicateur :

mean	donne la moyenne. L'argument *trim* permet d'écarter une certaine proportion des valeurs extrêmes dans le calcul de la moyenne.
min	donne le minimum
max	donne le maximum
range	donne le minimum et le maxium dans un vecteur de longueur 2
sd	donne l'écart-type
median	donne la médiane
quantile	donne les quantiles c(0,0.25,0.5,0.75,1) mais qui sont modifiables par l'option *probs*

Les statistiques de base

On peut ainsi calculer pour une variable quantitative quelques statistiques :

```
> mean(iris$Sepal.Length)

## [1] 5.843333

> sd(iris$Sepal.Length)

## [1] 0.8280661

> quantile(iris$Sepal.Length)

##   0%  25%  50%  75% 100%
##  4.3  5.1  5.8  6.4  7.9
```

Les statistiques de base

```
> r <- c(
+   mean(iris$Sepal.Length),
+   sd(iris$Sepal.Length),
+   quantile(iris$Sepal.Length)
+   )
> names(r) <- c("Moy.","EC","Min","1erQ","Méd.","3emeQ","Max")
> r

##      Moy.        EC       Min      1erQ
## 5.8433333 0.8280661 4.3000000 5.1000000
##      Méd.     3emeQ       Max
## 5.8000000 6.4000000 7.9000000
```

Les statistiques de base

```
> r <- rbind( r, c(
+   mean(iris$Sepal.Length),
+   sd(iris$Sepal.Length),
+   quantile(iris$Sepal.Length)
+   )
+ )
> rownames(r) <- c( "Sepal.Length", "Sepal.Width" )
> r

##                  Moy.        EC Min 1erQ
## Sepal.Length 5.843333 0.8280661 4.3  5.1
## Sepal.Width  5.843333 0.8280661 4.3  5.1
##              Méd. 3emeQ Max
## Sepal.Length  5.8   6.4 7.9
## Sepal.Width   5.8   6.4 7.9

> write.csv2(r,"data/Resultats.txt")
```

Les statistiques de base

Il existe une connexion de fichier particulière : "clipboard".
Dans ce cas, le résultat n'est pas écrit dans un fichier mais dans le presse-papiers par exemple pour le copier-coller dans une application tierce.

```
> write.csv2(r,"clipboard")
```

Les statistiques de base

Pour les CSVs, cela ne donne pas grand chose, mais le paquet *questionr* contient la fonction *clipcopy* qui permet d'exporter un objet au format HTML (via le paquet *R2HTML*).

```
> clipcopy(r)
```

L'objet peut alors peut alors être directement coller dans un tableur.

Les statistiques de base

Attention, toutes ces fonctions renvoient *NA* si une valeur ou plus est manquante (ie. égale à *NA*).

```
> min(c(3,NA,5))

## [1] NA
```

Pour ne pas avoir *NA* comme réponse, il faut utiliser un argument supplémentaire *na.rm=T*

```
> min(c(3,NA,5),na.rm=T)

## [1] 3
```

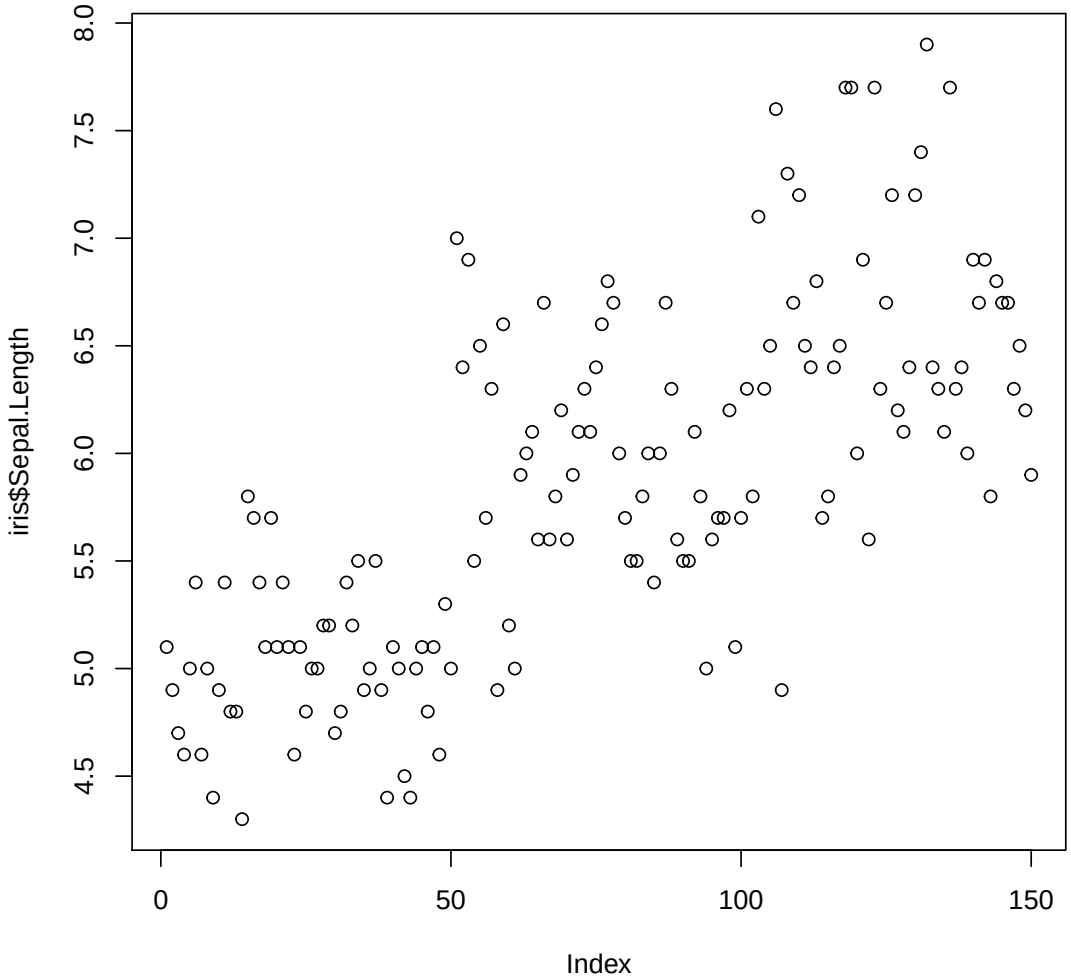
Plot

Pour obtenir une représentation graphique de la variable, il suffit de taper :

```
plot(iris$Sepal.Length)
```

Dans ce cas les valeurs de la variable sont en ordonnées et les abscisses sont les numéros d'observation.

Plot

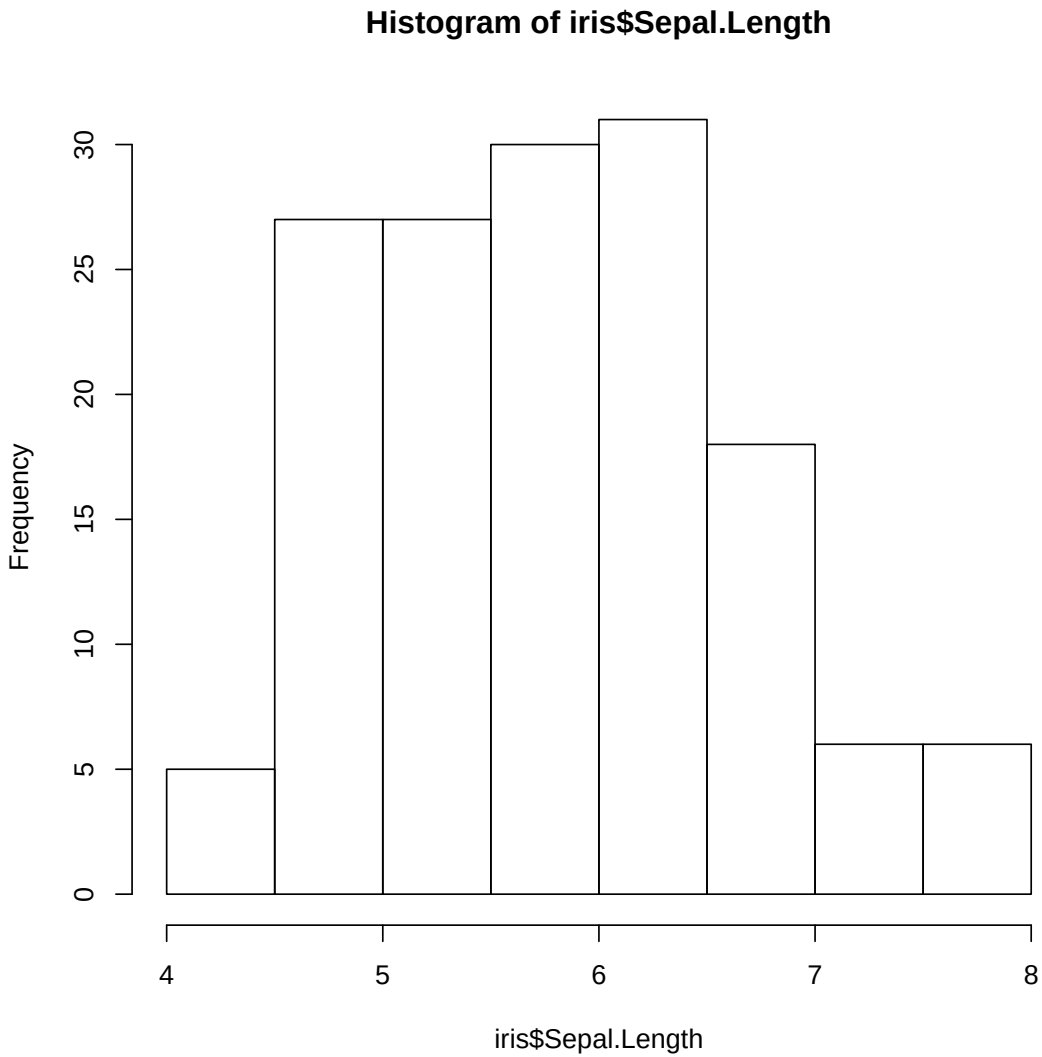


Histogramme

Pour obtenir un histogramme, il suffit de taper :

```
hist(iris$Sepal.Length)
```

Histogramme

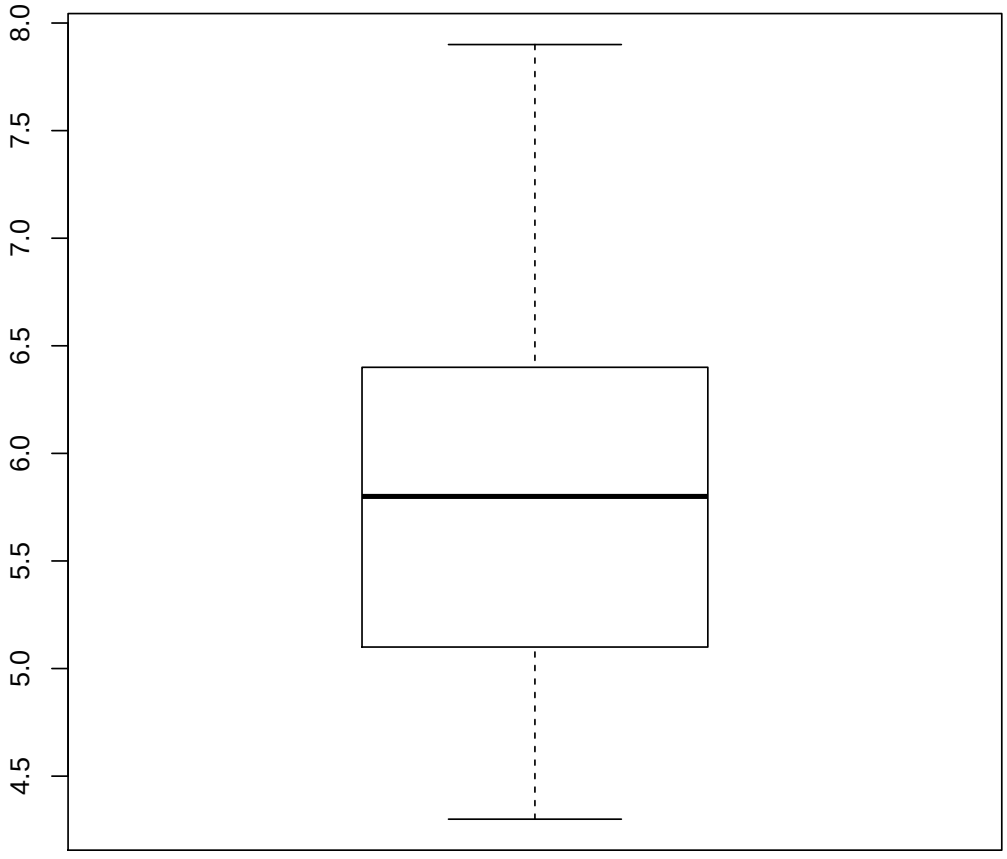


Boxplots

Pour obtenir un boxplot, il suffit de taper :

```
boxplot(iris$Sepal.Length)
```

Boxplots

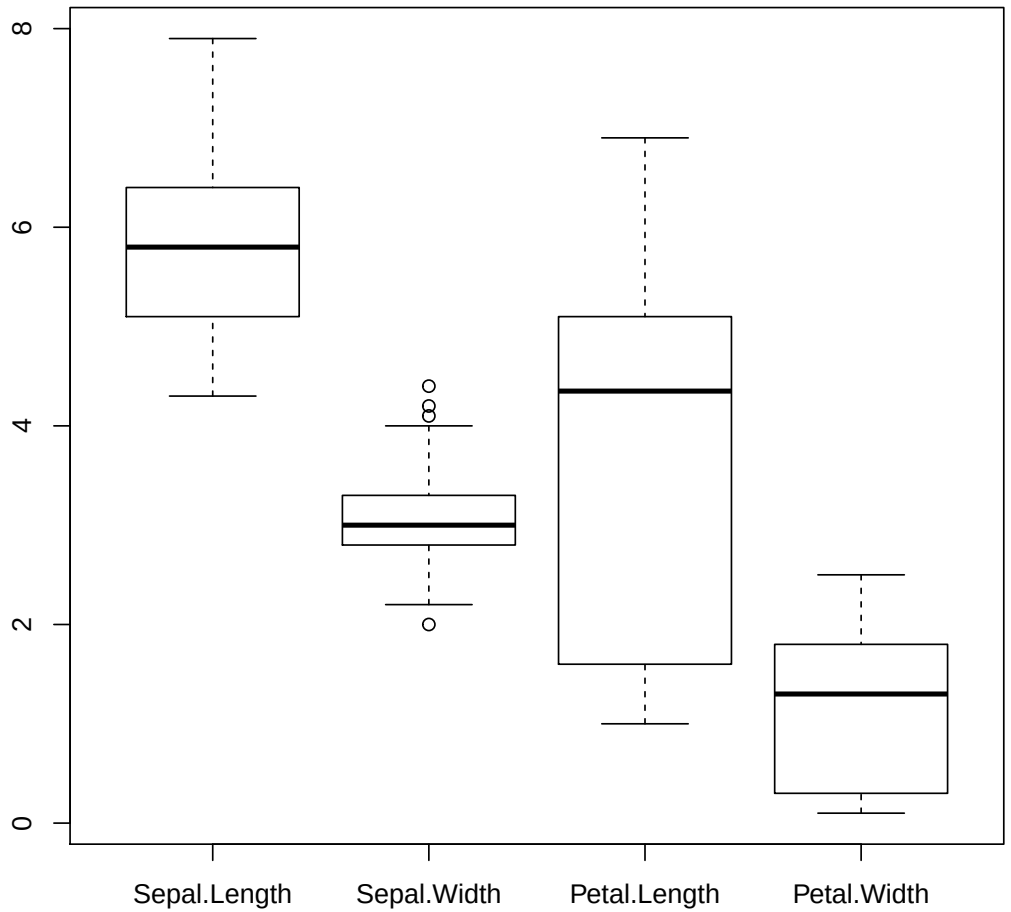


Boxplots

Pour obtenir des boxplots de plusieurs variables d'une même *data.frame*, il suffit de taper :

```
boxplot(iris[,1:4])
```

Boxplots



Corrélations

Pour analyser la covariance ou la corrélation de deux variables (ou plus), il suffit d'appeler la fonction *cov* ou *cor* avec les deux variables.

```
> cor(cg$score,cg$plaisir)

## [1] NA
```

Les méthodes disponibles pour le calcul de la corrélation sont celles de Pearson, Kendall et Spearman.

Corrélations

La gestion des valeurs manquantes est particulière dans le cas des corrélations.
En effet on peut avoir plusieurs cas de figures, pour reprendre l'aide de R.

Corrélations

Dans la majorité des cas on utilisera le *use="pairwise.complete.obs"* qui permet d'obtenir une valeur de corrélation en présence de valeurs manquantes. Dans ce cas, la corrélation est calculée pour tous les couples de valeurs ne contenant pas de *NA*.
Les alternatives sont de rendre une erreur en présence de *NA* ou de supprimer toutes les observations contenant une valeur manquante.

Corrélations

```
> cor(cg$score,cg$plaisir,use="pairwise.complete.obs")

## [1] 0.1492273
```

En fait, une matrice de corrélations peut être créée en utilisant la propriété de *cor* de se comporter différemment face à un objet de type *data.frame*.

Corrélations

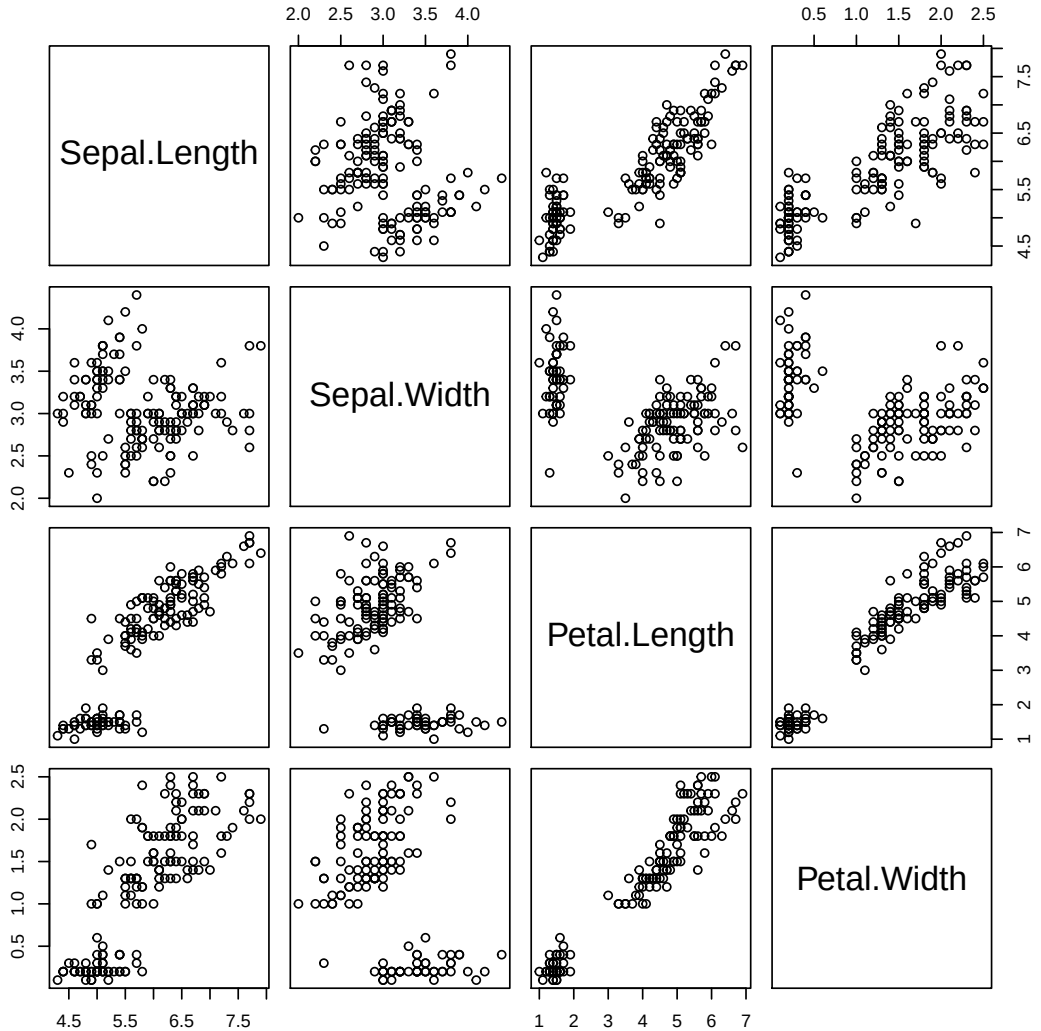
```
> cor(cg[,8:12],use="pairwise.complete.obs")

##                  score    plaisir performance
## score       1.0000000 0.14922727  0.46193763
## plaisir     0.1492273 1.00000000  0.31990943
## performance 0.4619376 0.31990943  1.00000000
## estime_soi  0.1203776 0.12368912  0.42288147
## anx_sociale 0.1236460 0.08192793  0.04367632
##             estime_soi anx_sociale
## score        0.1203776  0.12364597
## plaisir      0.1236891  0.08192793
## performance  0.4228815  0.04367632
## estime_soi   1.0000000 -0.17216954
## anx_sociale -0.1721695  1.00000000
```

Corrélations - Graphiques

On peut également passer une *data.frame* à *plot* pour avoir une table des corrélations graphiques.

```
plot(iris[,1:4])
```

Corrélations - Graphiques



Test de corrélation

Le but est ici de vous illustrer un test pour vous montrer l'utilisation des objets.

Test de corrélation

```
> cor.test(cg$score,cg$plaisir,use="pairwise.complete.obs")

##
##  Pearson's product-moment correlation
##
## data:  cg$score and cg$plaisir
## t = 22.511, df = 22250, p-value <
## 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.1363555 0.1620487
## sample estimates:
##       cor
## 0.1492273
```

Test de corrélation

Une sortie des résultats apparait mais comme la plupart des fonctions, la fonction *cor.test* renvoie aussi un objet silencieusement. Comme il n'y a pas d'affectation R devine qu'on veut l'imprimer. Mais si on change la syntaxe :

```
> montest <- cor.test(cg$score,cg$plaisir,use="pairwise.complete.obs")
```

Test de corrélation

On peut accéder aux valeurs en utilisant directement la fonction *str* qui permet de connaître la structure des objets.

Test de corrélation

```
> str(montest)

## List of 9
##  $ statistic  : Named num 22.5
##   ..- attr(*, "names")= chr "t"
##  $ parameter  : Named int 22250
##   ..- attr(*, "names")= chr "df"
##  $ p.value    : num 5.56e-111
##  $ estimate   : Named num 0.149
##   ..- attr(*, "names")= chr "cor"
##  $ null.value : Named num 0
##   ..- attr(*, "names")= chr "correlation"
##  $ alternative: chr "two.sided"
##  $ method     : chr "Pearson's product-moment correlation"
##  $ data.name  : chr "cg$score and cg$plaisir"
##  $ conf.int   : atomic [1:2] 0.136 0.162
##   ..- attr(*, "conf.level")= num 0.95
##  - attr(*, "class")= chr "htest"
```

Test de corrélation

Par exemple, pour récupérer la corrélation et son intervalle de confiance :

```
> montest$estimate

##       cor
## 0.1492273

> montest$conf.int

## [1] 0.1363555 0.1620487
## attr(,"conf.level")
## [1] 0.95
```

Autres tests

Les tests disponibles sous R sont nombreux dans les packages de base et incalculables si on tient compte des paquets : Student, Wilcoxon, χ^2, Bartlett, Fisher, ...
Tous, ou presque, fonctionnent sur le même principe.

Les données patient

Ce sont des données recoltées par une pharmacienne sur dossiers dans deux hopitaux.
Les données portent sur le traitement de la douleur chez l'enfant.
Le but est de déterminer la qualité de la prise en charge de la douleur dans ces hopitaux.
Pour cela les évaluations de la douleur se font sur une échelle de 0 à 100.
Les pathologies (CIM2) concernées sont codées de 1 à 4, de plus en plus grave (appendicite à arthrodèse).

Les données patient

Variable	Description
UID	Identifiant
Hopital	Hopital (A ou B)
Sexe	Sexe du patient
Poids	Poids du patient
Vitaux	Nombres de prises des signes vitaux
CIM2	Pathologie (de 1 à 4, de plus en plus grave)
age	Âge du patient
dureeopmin	Durée de l'opération en minutes
postopj	Nombres de jours post-opératoires
scoliose	Présence d'une scoliose
drepano	Enfant souffrant d'une drépanocytose/sphérocytose
ACP	Pompe à morphine administrée
peridurale	Injection péridurale
periACP	Pompe à morphine administrée en péridurale
nbttt	Nombre de traitements contre la douleur
totalechelle	Total des échelles de la la douleur
nbechelle	Nombre de prises d'échelles de douleur

Tableaux de contingence

Il suffit d'utiliser la fonction *table*.

```
> patient <- read.csv2("data/patient.csv")
> table(patient$sexe)

##
##  Feminin Masculin
##      101        75
```

Tableaux de contingence

Ces tableaux peuvent être à 2 niveaux :

```
> table(patient$sexe,patient$CIM2)

##
##              1  2  3  4
##   Feminin   18 24 18 41
##   Masculin 26 22 18  9
```

Tableaux de contingence

Comme les colonnes ne sont pas nommées cela peut être difficile de s'y retrouver si les variables ont les mêmes modalités.
Il suffit dans ce cas là de modifier la syntaxe :

```
> table(Sexe=patient$sexe,CIM2=patient$CIM2)

##            CIM2
## Sexe         1  2  3  4
##   Feminin  18 24 18 41
##   Masculin 26 22 18  9
```

Exportation de tableaux de contingence

Pour l'exportation des tableaux, elle peut se faire avec :

- *xtable*
- la fonction clipcopy de *questionr*
- sous forme de CSV

Dans ce dernier cas, l'aspect de la table exportée sera donc différent car il y a une conversion vers une *data.frame*.

Exportation de tableaux de contingence

```
> as.data.frame(table(Sexe=patient$sexe,CIM2=patient$CIM2))

##       Sexe CIM2 Freq
## 1  Feminin    1   18
## 2 Masculin    1   26
## 3  Feminin    2   24
## 4 Masculin    2   22
## 5  Feminin    3   18
## 6 Masculin    3   18
## 7  Feminin    4   41
## 8 Masculin    4    9
```

Exportation de tableaux de contingence (ancienne version)

Il y a une astuce pour contourner le problème. La classe *table* a la même structure interne qu'une *matrix*. Alors on berne R.

```
> tableau <- (table(Sexe=patient$sexe,CIM2=patient$CIM2))
> class(tableau)

## [1] "table"

> class(tableau) <- "matrix"
> class(tableau)

## [1] "matrix"
```

Exportation de tableaux de contingence (version correcte)

Depuis la première version du cours, il y a maintenant un contournement dans R :

```
> tableau <- (table(Sexe=patient$sexe,CIM2=patient$CIM2))
> as.data.frame.matrix(tableau)

##           1  2  3  4
## Feminin  18 24 18 41
## Masculin 26 22 18  9
```

Marges sur les tableaux

On peut ajouter les marges avec la fonction addmargins :

Marges sur les tableaux

```
> addmargins(tableau,1)

##            CIM2
## Sexe         1  2  3  4
##   Feminin   18 24 18 41
##   Masculin 26 22 18  9
##   Sum       44 46 36 50
```

Marges sur les tableaux

```
> addmargins(tableau,2)

##            CIM2
## Sexe          1   2   3   4 Sum
##   Feminin    18  24  18  41 101
##   Masculin   26  22  18   9  75
```

Marges sur les tableaux

```
> addmargins(tableau,1:2)

##            CIM2
## Sexe          1   2   3   4 Sum
##   Feminin    18  24  18  41 101
##   Masculin   26  22  18   9  75
##   Sum        44  46  36  50 176
```

Tableau de proportion

Cette fois c'est la fonction *prop.table* qu'on appelle en utilisant une table déjà créée :

```
> prop.table(tableau,1)

##           CIM2
## Sexe                 1         2         3
##   Feminin  0.1782178 0.2376238 0.1782178
##   Masculin 0.3466667 0.2933333 0.2400000
##           CIM2
## Sexe                 4
##   Feminin  0.4059406
##   Masculin 0.1200000
```

Proportions dans les tableaux

```
> prop.table(tableau,2)

##           CIM2
## Sexe                 1         2         3
##   Feminin  0.4090909 0.5217391 0.5000000
##   Masculin 0.5909091 0.4782609 0.5000000
##           CIM2
## Sexe                 4
##   Feminin  0.8200000
##   Masculin 0.1800000
```

Proportions dans les tableaux

On peut combiner les deux...

```
> addmargins(prop.table(tableau),1:2)

##            CIM2
## Sexe                  1           2           3
##   Feminin  0.10227273 0.13636364 0.10227273
##   Masculin 0.14772727 0.12500000 0.10227273
##   Sum      0.25000000 0.26136364 0.20454545
##            CIM2
## Sexe                  4         Sum
##   Feminin  0.23295455 0.57386364
##   Masculin 0.05113636 0.42613636
##   Sum      0.28409091 1.00000000
```

Autres fonctions

Les autres fonctions dans les tableaux de base sont *ftable*, *tabulate*, *xtabs*, ... Chacun a sa petite spécificité.
Dans les packages, il existe pléthore de fonctions pour calculer des tableaux croisés. Notamment des versions qui permettent d'ajuster la présentation comme dans *questionr*.