Table des matières

I Université FUN
Introduction à la statistique avec R

Bruno Falissard, Christophe Lalanne

Lien vers le cours de **l'Université FUN** : ici

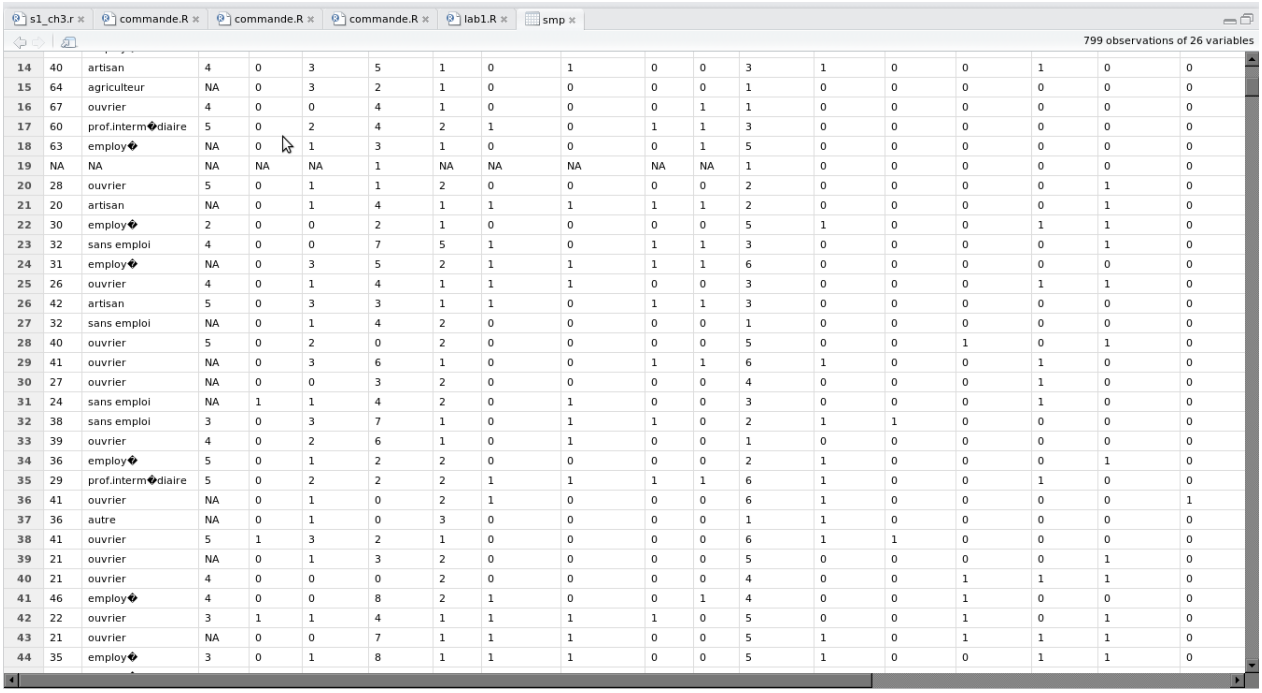
Ce cours est soumis à une licence Creative Commons :

1 Les Labs

1.1 LAB 1 : Gestion des données, DATA FRAME, Variable numériques et catégorielles

Dans cette première session, on va s'intéresser au langage de base donc :
— comment importer des données enregistrées, par exemple, dans un fichier Excel '
— comment manipuler des variables de type numérique et des variables de type catégoriel '

```
> ##Import du fichier de données smp2.csv
> smp <- read.csv2("/comptes/E131729J/XX_Université_fun/univfunR/TP/smp2.csv")
> ##Définition de l'espace de travail par défaut
> setwd("~/XX_Université_fun/univfunR/TP")
> ##Affichage des données :
> View(smp)
> View(smp)
```



s1_ch3.r | commande.R | commande.R | commande.R | lab1.R | smp — 799 observations of 26 variables

14	40	artisan	4	0	3	5	1	0	1	0	0	3	1	0	0	1	0	0
15	64	agriculteur	NA	0	3	2	1	0	0	0	0	1	0	0	0	0	0	0
16	67	ouvrier	4	0	0	4	1	0	0	0	1	1	0	0	0	0	0	0
17	60	prof.interm\u00e9diaire	5	0	2	4	2	1	0	1	1	3	0	0	0	0	0	0
18	63	employ\u00e9	NA	0	1	3	1	0	0	0	1	5	0	0	0	0	0	0
19	NA	NA	NA	NA	NA	1	NA	NA	NA	NA	NA	1	0	0	0	0	0	0
20	28	ouvrier	5	0	1	1	2	0	0	0	0	2	0	0	0	0	1	0
21	20	artisan	NA	0	1	4	1	1	1	1	1	2	0	0	0	0	1	0
22	30	employ\u00e9	2	0	0	2	1	0	0	0	0	5	1	0	0	1	1	0
23	32	sans emploi	4	0	0	7	5	1	0	1	1	3	0	0	0	0	1	0
24	31	employ\u00e9	NA	0	3	5	2	1	1	1	1	6	0	0	0	0	0	0
25	26	ouvrier	4	0	1	4	1	1	1	0	0	3	0	0	0	1	1	0
26	42	artisan	5	0	3	3	1	1	0	1	1	3	0	0	0	0	0	0
27	32	sans emploi	NA	0	1	4	2	0	0	0	0	1	0	0	0	0	0	0
28	40	ouvrier	5	0	2	0	2	0	0	0	0	5	0	0	1	0	1	0
29	41	ouvrier	NA	0	3	6	1	0	0	1	1	6	1	0	0	1	0	0
30	27	ouvrier	NA	0	0	3	2	0	0	0	0	4	0	0	0	1	0	0
31	24	sans emploi	NA	1	1	4	2	0	1	0	0	3	0	0	0	1	0	0
32	38	sans emploi	3	0	3	7	1	0	1	1	0	2	1	1	0	0	0	0
33	39	ouvrier	4	0	2	6	1	0	1	0	0	1	0	0	0	0	0	0
34	36	employ\u00e9	5	0	1	2	2	0	0	0	0	2	1	0	0	0	1	0
35	29	prof.interm\u00e9diaire	5	0	2	2	2	1	1	1	1	6	1	0	0	1	0	0
36	41	ouvrier	NA	0	1	0	2	1	0	0	0	6	1	0	0	0	0	1
37	36	autre	NA	0	1	0	3	0	0	0	0	1	1	0	0	0	0	0
38	41	ouvrier	5	1	3	2	1	0	0	0	0	6	1	1	0	0	0	0
39	21	ouvrier	NA	0	1	3	2	0	0	0	0	5	0	0	0	0	1	0
40	21	ouvrier	4	0	0	0	2	0	0	0	0	4	0	0	1	1	1	0
41	46	employ\u00e9	4	0	0	8	2	1	0	0	1	4	0	0	1	0	0	0
42	22	ouvrier	3	1	1	4	1	1	1	1	0	5	0	0	1	0	1	0
43	21	ouvrier	NA	0	0	7	1	1	1	0	0	5	1	0	1	1	1	0
44	35	employ\u00e9	3	0	1	8	1	1	1	0	0	5	1	0	0	1	1	0

Listing du nom des variables présentes dans le fichier :

```
> names(smp)
 [1] "age"          "prof"         "duree"        "discip"       "n.enfant"     "n.fratrie"
 [7] "ecole"        "separation"   "juge.enfant"  "place"        "abus"         "grav.cons"
[13] "dep.cons"     "ago.cons"     "ptsd.cons"    "alc.cons"     "subst.cons"   "scz.cons"
[19] "char"         "rs"           "ed"           "dr"           "suicide.s"    "suicide.hr"
[25] "suicide.past" "dur.interv"
```

Affichage des données V2

```
> str(smp)
'data.frame': 799 obs. of  26 variables:
 $ age          : int  31 49 50 47 23 34 24 52 42 45 ...
 $ prof         : Factor w/ 8 levels "agriculteur",..: 3 NA 7 6 8 6 3 2 6 6 ...
 $ duree        : int  4 NA 5 NA 4 NA NA 5 4 NA ...
 $ discip       : int  0 0 0 0 1 0 0 0 1 0 ...
 $ n.enfant     : int  2 7 2 0 1 3 5 2 1 2 ...
 $ n.fratrie    : int  4 3 2 6 6 2 3 9 12 5 ...
 $ ecole        : int  1 2 2 1 1 2 1 2 1 2 ...
 $ separation   : int  0 1 0 1 1 0 1 0 1 0 ...
 $ juge.enfant  : int  0 0 0 0 NA 0 1 0 1 0 ...
 $ place        : int  0 0 0 1 1 0 1 0 0 0 ...
 $ abus         : int  0 0 0 0 0 0 0 0 1 1 ...
 $ grav.cons    : int  1 2 2 1 2 1 5 1 5 5 ...
 $ dep.cons     : int  0 0 0 0 1 0 1 0 1 0 ...
 $ ago.cons     : int  1 0 0 0 0 0 0 0 0 0 ...
 $ ptsd.cons    : int  0 0 0 0 0 0 0 0 0 0 ...
 $ alc.cons     : int  0 0 0 0 0 0 0 0 1 1 ...
 $ subst.cons   : int  0 0 0 0 0 0 1 0 1 0 ...
 $ scz.cons     : int  0 0 0 0 0 0 0 0 0 0 ...
 $ char         : int  1 1 1 1 1 1 1 1 4 1 ...
 $ rs           : int  2 2 2 2 2 1 3 2 3 2 ...
 $ ed           : int  1 2 3 2 2 2 3 2 3 2 ...
 $ dr           : int  1 1 2 2 2 1 2 2 1 2 ...
 $ suicide.s    : int  0 0 0 1 0 0 3 0 4 0 ...
 $ suicide.hr   : int  0 0 0 0 0 0 1 0 1 0 ...
 $ suicide.past: int  0 0 0 0 1 0 1 0 1 0 ...
 $ dur.interv   : int  NA 70 NA 105 NA NA 105 84 78 60 ...
```

Note :

— **int** → Variable quantitative

— **factor** → Variable qualitative (level)

Résumé numérique des données :

```
> summary(smp)
      age                      prof          duree            discip          n.enfant
 Min.   :19.0   ouvrier            :227   Min.   :1.000   Min.   :0.000   Min.   : 0.000
 1st Qu.:28.0   sans emploi        :222   1st Qu.:4.000   1st Qu.:0.000   1st Qu.: 0.000
 Median :37.0   employe            :135   Median :5.000   Median :0.000   Median : 1.000
 Mean   :38.9   artisan            : 90   Mean   :4.302   Mean   :0.232   Mean   : 1.755
 3rd Qu.:48.0   prof.intermediaire: 58   3rd Qu.:5.000   3rd Qu.:0.000   3rd Qu.: 3.000
 Max.   :83.0   (Other)            : 61   Max.   :5.000   Max.   :1.000   Max.   :13.000
 NA's   :2      NA's               :  6   NA's   :223     NA's   :6       NA's   :26
   n.fratrie          ecole         separation      juge.enfant         place
 Min.   : 0.000   Min.   :1.000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
 1st Qu.: 2.000   1st Qu.:1.000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000
 Median : 3.000   Median :2.000   Median :0.0000   Median :0.0000   Median :0.0000
 Mean   : 4.287   Mean   :1.866   Mean   :0.4226   Mean   :0.2771   Mean   :0.2285
 3rd Qu.: 6.000   3rd Qu.:2.000   3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:0.0000
 Max.   :21.000   Max.   :5.000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
                  NA's   :5       NA's   :11       NA's   :5        NA's   :7
      abus          grav.cons        dep.cons         ago.cons        ptsd.cons
 Min.   :0.0000   Min.   :1.000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
 1st Qu.:0.0000   1st Qu.:2.000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000
 Median :0.0000   Median :4.000   Median :0.0000   Median :0.0000   Median :0.0000
 Mean   :0.2778   Mean   :3.643   Mean   :0.3967   Mean   :0.1665   Mean   :0.2165
 3rd Qu.:1.0000   3rd Qu.:5.000   3rd Qu.:1.0000   3rd Qu.:0.0000   3rd Qu.:0.0000
 Max.   :1.0000   Max.   :7.000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
 NA's   :7        NA's   :4
    alc.cons        subst.cons        scz.cons           char             rs
 Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :1.000   Min.   :1.000
 1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:1.000   1st Qu.:1.000
 Median :0.0000   Median :0.0000   Median :0.0000   Median :1.000   Median :2.000
 Mean   :0.1865   Mean   :0.2653   Mean   :0.0826   Mean   :1.512   Mean   :2.057
 3rd Qu.:0.0000   3rd Qu.:1.0000   3rd Qu.:0.0000   3rd Qu.:2.000   3rd Qu.:3.000
 Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :4.000   Max.   :3.000
                                                    NA's   :96      NA's   :103
       ed              dr          suicide.s        suicide.hr      suicide.past
 Min.   :1.000   Min.   :1.000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
 1st Qu.:1.000   1st Qu.:1.000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000
```

```
 Median :2.000   Median :2.000   Median :0.0000   Median :0.0000   Median :0.0000
 Mean   :1.866   Mean   :2.153   Mean   :0.7942   Mean   :0.2013   Mean   :0.2841
 3rd Qu.:3.000   3rd Qu.:3.000   3rd Qu.:1.0000   3rd Qu.:0.0000   3rd Qu.:1.0000
 Max.   :3.000   Max.   :3.000   Max.   :5.0000   Max.   :1.0000   Max.   :1.0000
 NA's   :107     NA's   :111     NA's   :41       NA's   :39       NA's   :14
   dur.interv
 Min.   :  0.00
 1st Qu.: 48.00
 Median : 60.00
 Mean   : 61.89
 3rd Qu.: 75.00
 Max.   :120.00
 NA's   :50
```

La commande summuray fonctionne également pour des variables seules (pas seulement pour les data frames)
Pour isoler une variable dans un data frame, *xxx$yyy* (x : nom du data frame, y : nom de la variable)

```
> summary(smp$age)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
   19.0    28.0    37.0    38.9    48.0    83.0       2
```

On peut toujours écrire *smp$age* mais dans ce cas, **R** va nous renvoyer l'ensemble des observations

```
> smp$age
  [1] 31 49 50 47 23 34 24 52 42 45 31 NA 21 40 64 67 60 63 NA 28 20 30 32 31 26
 [26] 42 32 40 41 27 24 38 39 36 29 41 36 41 21 21 46 22 21 35 45 38 19 21 27 40
 [51] 39 47 24 36 39 22 38 37 29 23 36 42 56 28 36 38 43 29 64 25 51 35 30 37 26
 [76] 36 58 32 30 26 27 23 24 39 43 39 26 44 37 40 24 46 26 38 37 30 39 36 39 28
[101] 27 51 48 47 41 35 25 31 44 40 29 34 49 57 33 35 32 34 46 45 31 42 48 34 34
[126] 64 50 53 49 53 37 42 55 32 33 40 29 32 23 61 39 30 37 30 39 49 44 40 56 43
[151] 27 21 44 50 50 20 37 42 27 22 25 20 21 19 25 24 49 24 26 35 22 24 23 46 26
[176] 41 51 20 30 37 49 28 28 51 40 33 25 29 40 43 35 50 44 35 24 43 26 45 42 45
[201] 48 45 34 31 40 22 42 38 38 40 46 26 29 25 40 43 28 29 32 28 57 31 71 33 24
[226] 22 25 26 52 33 38 39 41 52 33 39 59 33 50 58 23 41 43 42 22 57 41 30 66 49
[251] 46 28 59 35 44 83 34 49 60 56 46 62 41 27 53 48 66 66 55 61 43 54 38 51 51
[276] 50 56 53 49 41 44 64 42 52 72 43 30 32 43 25 27 25 52 39 42 59 46 62 50 24
[301] 43 32 67 28 44 19 20 23 26 28 31 42 57 30 36 53 33 25 22 42 25 32 23 45 48
[326] 35 37 38 24 47 61 38 27 27 26 30 47 37 30 41 29 37 28 47 26 50 23 60 37 48
[351] 41 28 54 61 33 31 25 66 26 29 29 53 24 48 40 47 40 41 54 25 36 44 32 27 31
[376] 34 34 71 20 54 39 50 36 37 43 28 21 35 36 53 36 38 66 62 38 24 49 21 34 29
[401] 36 29 33 34 57 65 25 36 31 54 49 42 30 20 23 21 23 39 45 29 21 54 77 23 32
[426] 58 49 26 40 51 62 45 41 30 52 20 36 34 35 30 46 79 66 19 41 51 26 56 33 39
[451] 72 45 59 21 41 43 55 26 49 29 26 28 77 61 63 30 49 48 45 32 56 48 64 73 33
[476] 74 54 27 49 45 27 53 62 54 37 56 60 33 34 32 44 49 46 67 39 59 63 81 38 58
[501] 42 73 48 41 28 44 45 46 50 27 56 46 42 25 23 26 19 24 24 32 23 24 33 21 33
[526] 41 24 31 19 25 51 39 22 20 30 34 28 20 20 33 24 32 37 25 24 29 19 37 56 49
[551] 60 29 22 20 49 33 30 29 25 62 41 33 44 60 24 24 33 27 45 33 44 23 23 35 36
[576] 28 24 27 27 28 27 40 52 19 31 21 33 23 30 23 31 48 24 24 26 32 29 38 23 50
[601] 26 47 38 24 24 19 25 31 33 26 38 23 37 19 49 33 30 38 30 26 27 21 31 19 26
[626] 28 49 35 25 32 27 20 30 25 21 54 27 22 39 21 54 49 23 36 59 50 24 47 42 41
[651] 33 46 23 19 39 38 40 39 40 44 26 48 47 23 25 20 45 44 57 39 55 19 34 28 33
[676] 19 33 27 46 47 22 27 26 52 56 44 63 34 41 38 37 58 37 24 60 26 21 52 20 37
[701] 32 32 58 49 32 37 46 50 44 47 37 38 50 56 30 34 43 55 43 31 55 41 68 45 48
[726] 42 71 38 46 65 51 57 57 71 40 43 71 48 34 69 43 35 62 34 51 48 36 44 49 74
[751] 19 56 57 65 52 77 29 37 45 40 72 27 56 35 30 37 30 40 54 26 48 83 32 22 48
[776] 67 58 37 24 34 39 38 39 56 35 26 70 68 42 41 40 26 50 27 28 44 31 38 71


> table(smp$age)

19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45
15 15 18 13 23 30 22 30 25 21 20 25 18 22 26 20 16 18 25 24 23 22 23 19 17 19 17
46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72
16 12 17 23 16 11 10  8 11  6 14  9  7  6  7  5  7  4  5  3  6  4  2  1  1  6  3
73 74 77 79 81 83
 2  2  3  1  1  2
```

On peut cependant afficher une observation précise grâce au crochets.

```
> smp$age[1]
[1] 31
```

On peut également demander à R d'afficher une suite de données (par exemple les dix premières)

```
> smp$age[1:10]
 [1] 31 49 50 47 23 34 24 52 42 45
```

Calcul de la valeur minimale pour la variable age (Il faut préciser le second paramètre sinon il n'enlève pas les valeurs manquante et nous renvoi NA)

```
> min(smp$age, na.rm =TRUE)
[1] 19
```

Et maintenant pour les fonctions de base *max* et *mean*

```
> max(smp$age, na.rm = TRUE)
[1] 83
> mean(smp$age, na.rm = TRUE)
[1] 38.89962
```

Dans le cadre des variable binaire, on souhaite renvoyer les premières valeurs de la variable abus :

```
> smp$abus[1:10]
 [1] 0 0 0 0 0 0 0 0 1 1
```

Ou alors

```
> head(smp$abus, n=10)
 [1] 0 0 0 0 0 0 0 0 1 1
```

Si l'on souhaite retrouver les modalité de cette variable binaire :

```
> unique(smp$abus)
[1]  0  1 NA
```

Pour connaitre le nombre total d'observations :

```
> length(smp$abus)
[1] 799
```

Ce qui correspond globalement au nombre de lignes du tableau *smp*

```
> nrow(smp)
[1] 799
```

Le tableau d'effectifs associés à chaque modalités (On remarque que la somme des effectifs n'est pas égales à 799, cela provient du fait que les variables NA ne sont pas affichées)

```
> table(smp$abus)

  0   1
572 220
```

Pour les afficher les valeurs manquantes :

```
> table(smp$abus, useNA = "always")

   0    1 <NA>
 572  220    7
```

Si l'on fait :

```
> summary(smp$abus)
    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
  0.0000  0.0000  0.0000  0.2778  1.0000  1.0000       7
```

On remarque que la variable abus est traités comme une variable numérique. Si l'on souhaite la traiter comme une variable qualitative, nous allons utiliser la commande factor qui va retourner les modes possibles :

```
> head(smp$abus)
[1] 0 0 0 0 0 0
> ##Renvoie les modes possibles.
> head(factor(smp$abus))
[1] 0 0 0 0 0 0
Levels: 0 1
```

Nous allons créer une nouvelle variable pour traiter ABUS comme un variable qualitative :

```
> abus <- factor(smp$abus)
> table(abus, useNA="always")
abus
   0    1 <NA>
 572  220    7
```

Nous pouvons donner des noms à ces niveaux :

```
> abus <- factor(smp$abus, levels =c(0,1), labels=c("Non","Oui"))
```

Afficher les données de Abus sans les valeurs non renseignées sous forme d'une tableau :

```
> table(abus)
abus
Non Oui
572 220
```

Pour affichier les valeurs de Abus avec les valeurs non renseignées

```
> table(abus, useNA="always")
abus
 Non  Oui <NA>
 572  220    7
```

On peut voir que les modalités Non (resp Oui) ont été associées aux valeurs 0 (resp 1).

Nous pouvons dès lors regarder une autre variable qualitative :

```
> names(smp)
 [1] "age"          "prof"         "duree"        "discip"       "n.enfant"     "n.fratrie"
 [7] "ecole"        "separation"   "juge.enfant"  "place"        "abus"         "grav.cons"
[13] "dep.cons"     "ago.cons"     "ptsd.cons"    "alc.cons"     "subst.cons"   "scz.cons"
[19] "char"         "rs"           "ed"           "dr"           "suicide.s"    "suicide.hr"
[25] "suicide.past" "dur.interv"
```

Et prenons par exemple le nombre d'enfants. Si l'on regarde les premières observations de cette variable :

```
> head(smp$n.enfant)
[1] 2 7 2 0 1 3
```

On peut remarquer que l'on a des valeurs numériques. On peut donc utiliser la commande *summary()* car **R** considère que c'est une variable numérique.

```
> summary(smp$n.enfant)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
  0.000   0.000   1.000   1.755   3.000  13.000      26
```

On peut créer une nouvelle variable :

```
> gauss <- smp$n.enfant
> head(gauss)
[1] 2 7 2 0 1 3
> summary(gauss)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
  0.000   0.000   1.000   1.755   3.000  13.000      26
```

Si l'on souhaite étudier la répartition des effectifs, nous devons alors entrer la commande :

```
> table(gauss)
gauss
  0   1   2   3   4   5   6   7   8   9  10  11  13
214 220 125 101  55  31   7   7   7   2   2   1   1
```

Si l'on souhaite retourner le nombre d'enfant dont l'âge est supérieur à 4 ans.

```
> table(gauss > 4)

FALSE  TRUE
  715    58
```

Un autre test :

```
> table(gauss <=4)

FALSE  TRUE
   58   715
```

Si l'on souhaite définir une nouvelle variable correspondant à l'age des enfants dans une variable **mais** sous forme de facteurs (et non plus de nombres), nous ferons :

```
> smp$n.enfant.cat <-factor(smp$n.enfant)
## Affichage comme tableau d'effectifs
> table(smp$n.enfant.cat)

  0   1   2   3   4   5   6   7   8   9  10  11  13
214 220 125 101  55  31   7   7   7   2   2   1   1
```

On peut retourner le nombre de niveau/mode de cette variable :

```
> levels(smp$n.enfant.cat)
 [1] "0"  "1"  "2"  "3"  "4"  "5"  "6"  "7"  "8"  "9"  "10" "11" "13"
> nlevels(smp$n.enfant.cat)
[1] 13
```

On peut à présent rassembler (agréger) les derniers niveau à partir de 5, on va considérer que les niveau 6 à 13 représentent une modalité unique - On redéfinit donc un nouveau mode :

```
> levels(smp$n.enfant.cat)[6:13]<-"+5"
## Affichage comme tableau d'effectifs
> table(smp$n.enfant.cat)

  0   1   2   3   4  +5
214 220 125 101  55  58
```

Les effectifs ont donc été agrégés dans la même classe.
Il serait possible de faire la même commande pour une variable numérique

Nous allons maintenant sauvegarder notre fichier smp au format R. On donne le nom du Data Frame sur lequel nous travallions ainsi que le noms du fichier.

```
> save(smp, file="smp_v1.rda")
```

On peut également sauvegarder l'historique des commandes que nous avons saisies

```
> savehistory("commande.R")
```

1.2 LAB 2 : Indexation critériée d'observations, sélection de variables, graphiques univariés

Dans cette deuxième session, nous allons nous intéresser à la sélection indexée d'observations ou à la restriction d'un tableau de données à un certain nombre de variables, ce qui est souvent plus pratique, soit pour faire des analyses statistiques, soit pour faire des représentations graphiques.

Récupération du code épuré du précédent lab

Note : il existe deux possibilités pour chargé un fichier :

1. Appuyer sur le bouton source sachant que le script est déjà écrit dans le workspace
2. Utiliser la fonction load pour charger des données : ex : *load("smp_lab2.rda")*

Définition du répertoire de travail et import des données.

```
> setwd("[...]/StatDesR/TP")
> smp <- read.csv2("DONNEES/smp2.csv")
> names(smp)
 [1] "age"          "prof"         "duree"        "discip"
 [5] "n.enfant"     "n.fratrie"    "ecole"        "separation"
 [9] "juge.enfant"  "place"        "abus"         "grav.cons"
[13] "dep.cons"     "ago.cons"     "ptsd.cons"    "alc.cons"
[17] "subst.cons"   "scz.cons"     "char"         "rs"
[21] "ed"           "dr"           "suicide.s"    "suicide.hr"
[25] "suicide.past" "dur.interv"
> str(smp)
'data.frame': 799 obs. of  26 variables:
 $ age          : int  31 49 50 47 23 34 24 52 42 45 ...
 $ prof         : Factor w/ 8 levels "agriculteur",..: 3 NA 7 6 8 6 3 2 6 6 ...
 $ duree        : int  4 NA 5 NA 4 NA NA 5 4 NA ...
 $ discip       : int  0 0 0 0 1 0 0 0 1 0 ...
 $ n.enfant     : int  2 7 2 0 1 3 5 2 1 2 ...
 $ n.fratrie    : int  4 3 2 6 6 2 3 9 12 5 ...
 $ ecole        : int  1 2 2 1 1 2 1 2 1 2 ...
 $ separation   : int  0 1 0 1 1 0 1 0 1 0 ...
 $ juge.enfant  : int  0 0 0 0 NA 0 1 0 1 0 ...
 $ place        : int  0 0 0 1 1 0 1 0 0 0 ...
 $ abus         : int  0 0 0 0 0 0 0 0 1 1 ...
 $ grav.cons    : int  1 2 2 1 2 1 5 1 5 5 ...
 $ dep.cons     : int  0 0 0 0 1 0 1 0 1 0 ...
 $ ago.cons     : int  1 0 0 0 0 0 0 0 0 0 ...
 $ ptsd.cons    : int  0 0 0 0 0 0 0 0 0 0 ...
 $ alc.cons     : int  0 0 0 0 0 0 0 0 1 1 ...
 $ subst.cons   : int  0 0 0 0 0 0 1 0 1 0 ...
 $ scz.cons     : int  0 0 0 0 0 0 0 0 0 0 ...
 $ char         : int  1 1 1 1 1 1 1 1 4 1 ...
 $ rs           : int  2 2 2 2 2 1 3 2 3 2 ...
 $ ed           : int  1 2 3 2 2 2 3 2 3 2 ...
 $ dr           : int  1 1 2 2 2 1 2 2 1 2 ...
 $ suicide.s    : int  0 0 0 1 0 0 3 0 4 0 ...
 $ suicide.hr   : int  0 0 0 0 0 0 1 0 1 0 ...
 $ suicide.past: int  0 0 0 0 1 0 1 0 1 0 ...
 $ dur.interv   : int  NA 70 NA 105 NA NA 105 84 78 60 ...
> summary(smp)
## Trop long
```

Création d'une variable qualitative :

```
> gauss <- factor(smp$n.enfant)
> levels(gauss)[6:13]<-"5+"
> table(gauss)
gauss
  0   1   2   3   4  5+
214 220 125 101  55  58
```

Pour les variables numériques

Nous avons vu dans la session précédente que nous pouvions accéder directement aux observations d'une variable de la manière suivante :

```
> smp$age[1]
[1] 31
```

Il est également possible de réaliser cet accès à partir du Data Frame lui même en indexant ce dernier (exemple pour la première observation de la première variable (Age))

```
 [1] "age"          "prof"         "duree"        "discip"       "n.enfant"     "n.fratrie"    "
    ecole"
 [8] "separation"   "juge.enfant"  "place"        "abus"         "grav.cons"    "dep.cons"     "ago
    .cons"
[15] "ptsd.cons"    "alc.cons"     "subst.cons"   "scz.cons"     "char"         "rs"           "ed"
[22] "dr"           "suicide.s"    "suicide.hr"   "suicide.past" "dur.interv"
> smp[1,1]
[1] 31
> smp[1,"age"]  ## Méthode équivalente
[1] 31
```

Pour la seconde méthode, il est en effet plus commode de donner directement le nom de la variable (ici "age").

Pour les variables catégorielle

Nous allons nous intéresser en particulier à la variable profession dont on va afficher les six premières valeurs :

```
> head(smp$prof)
[1] autre              <NA>               prof.intermediaire ouvrier            sans emploi
[6] ouvrier
Levels: agriculteur artisan autre cadre employe ouvrier prof.intermediaire sans emploi
```

Nous allons effectuer une restriction à une modalite :

```
> table(smp$prof)

       agriculteur            artisan              autre             cadre            employe
                 6                 90                 31                 24                135
           ouvrier prof.intermediaire       sans emploi
               227                 58                222
```

Nous allons compter le nombre de personnes ayant pour profession "agriculteur" :

```
> table(smp$prof == "agriculteur")

FALSE  TRUE
  787     6
```

Nous pouvons retrouver les valeurs qui remplissent cette condition avec la fonction *which()* :

```
> which(smp$prof == "agriculteur")
[1]  15 312 384 391 439 442
```

R nous renvoit les numéro d'observation pour lesquelles la valeur de smp$prof est agriculteur.

Ce système d'indexation qui correspond à celui du dictionnaire (à une position, une valeur particulière) permet d'indexer directement les valeurs de la variable *age* pour lesquelles, la profession est agriculteur :

```
> smp$age[which(smp$prof == "agriculteur")]
[1] 64 42 37 36 35 79
```

Nous venons de récupérer l'âge des personnes ayant pour profession "agriculteur". Cette liste est donc obtenue directement à partir d'un test d'égalité logique.

Il est possible de réaliser une telle extraction sans utiliser le mécanisme d'indexation avec les crochets à l'aide de la commande *subset()* ; cette dernière prend pour argument :

1. Le nom du Dataframe
2. Le fitre que l'on veut appliquer sur les lignes (pour nous : *smp$prof == "agriculteur"*)
3. La variable sur laquelle on souhaite appliquer le filtre

```
> subset(smp, prof=="agriculteur",age)
    age
15   64
312  42
384  37
391  36
439  35
442  79
```

L'avantage est que l'on est plus obligé de préfixer le nom des variables par le nom du dataframe.
Si l'on souhaite étendre la sélection à plus d'une variable, on peut par exemple faire :

```
> names(smp)[1:5]
[1] "age"       "prof"      "duree"     "discip"    "n.enfant"

> subset(smp, prof=="agriculteur",1:5)
    age         prof duree discip n.enfant
15   64 agriculteur    NA      0         3
312  42 agriculteur     4      0         3
384  37 agriculteur     5      1         2
391  36 agriculteur     4      1         3
439  35 agriculteur     3      0         0
442  79 agriculteur     5      0         5
```

Pour reprendre l'indexation des variables 1 à 5, on peut également faire :

```
> names(smp)[1:5]
[1] "age"       "prof"      "duree"     "discip"    "n.enfant"

> subset(smp, prof=="agriculteur",c(1,3,4,5))
    age duree discip n.enfant
15   64    NA      0         3
312  42     4      0         3
384  37     5      1         2
391  36     4      1         3
439  35     3      0         0
442  79     5      0         5
```

Ou encore plus simplement :

```
> subset(smp, prof=="agriculteur",c(age,duree,discip,n.enfant))
    age duree discip n.enfant
15   64    NA      0         3
312  42     4      0         3
384  37     5      1         2
391  36     4      1         3
439  35     3      0         0
442  79     5      0         5
```

Il est également possible de rajouter des filtres sur les lignes ; On peut indexer les individus dont la profession est agriculteur et (&) dont le nombre d'enfant est supérieur à 2 :

```
> subset(smp, prof=="agriculteur" & n.enfant > 2,c(age,duree,discip,n.enfant))
    age duree discip n.enfant
15   64    NA      0        3
312  42     4      0        3
391  36     4      1        3
442  79     5      0        5
```

On peut encore s'amuser à faire des filtres beaucoups plus complexes : "Puisque l'on remarque dans les observation ci dessus qu'une des valeurs est manquante, on peut se demander si il est possible de récupérer les cas complet " c'est à dire les individus qui n'ont pas de valeur manquante sur la valeur de durée :

```
> subset(smp, prof=="agriculteur" & n.enfant > 2 & complete.cases(duree),c(age,duree,discip,n.
    enfant))
    age duree discip n.enfant
312  42     4      0        3
391  36     4      1        3
442  79     5      0        5
```

En réutilisant la variable agrégée *gauss* que l'on a définit précédemment :

```
> table(smp$n.enfant)
  0   1   2   3   4   5   6   7   8   9  10  11  13
214 220 125 101  55  31   7   7   7   2   2   1   1

> gauss <- factor(smp$n.enfant)
> levels(gauss) [6:13]<- "5+"
> table(gauss)
gauss
  0   1   2   3   4  5+
214 220 125 101  55  58
```

Il est possible de stocker ce tableau dans une variable afin d'effectuer des opérations dessus :

```
> tab <- table(gauss)
> tab
gauss
  0   1   2   3   4  5+
214 220 125 101  55  58
```

Pour obtenir la somme des valeurs :

```
> sum(tab)
[1] 773
```

Pour obtenir la fréquence :

```
> tab/sum(tab)
gauss
        0         1         2         3         4         5+
0.27684347 0.28460543 0.16170763 0.13065977 0.07115136 0.07503234
```

On peut remarquer que R procède mode par mode, c'est à dire, $0.27684347 = 214/773$. Il existe une commande : *prop.table* qui permet d'obtenir les mêmes résultats :

```
> prop.table(tab)
gauss
        0         1         2         3         4         5+
0.27684347 0.28460543 0.16170763 0.13065977 0.07115136 0.07503234
```

Il est possible de passer en paramètre de cette fonction, le tableau de la variable agrégée :

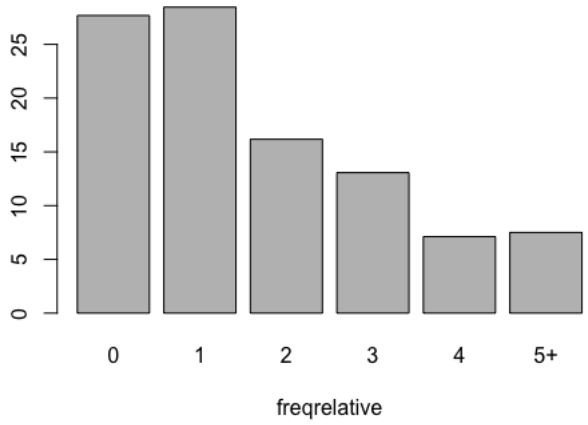
```
> prop.table(table(gauss))
gauss
          0           1           2           3           4          5+
0.27684347  0.28460543  0.16170763  0.13065977  0.07115136  0.07503234
```

On peut avoir envi de représenter les résultats avec un nombre finis de chiffres après la virgule ; pour ce faire, on utilise la commande *round()* :

```
> round(prop.table(table(gauss)),3)
gauss
    0     1     2     3     4    5+
0.277 0.285 0.162 0.131 0.071 0.075
```

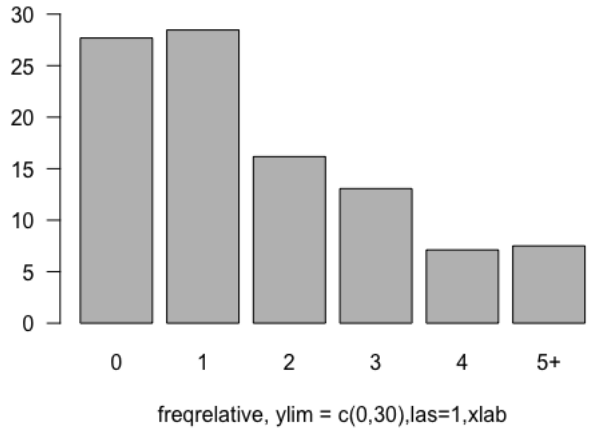
Il est également possible de réaliser des représentations graphiques simples sous forme d'un barplot. Par exemple avec la variable catégoricielle *gauss*, pour représenter les fréquences relatives obtenues sous forme de pourcentages (ne pas oublier de multiplier par 100) :

```
> barplot(prop.table(table(gauss))*100,xlab = "freqrelative")
```



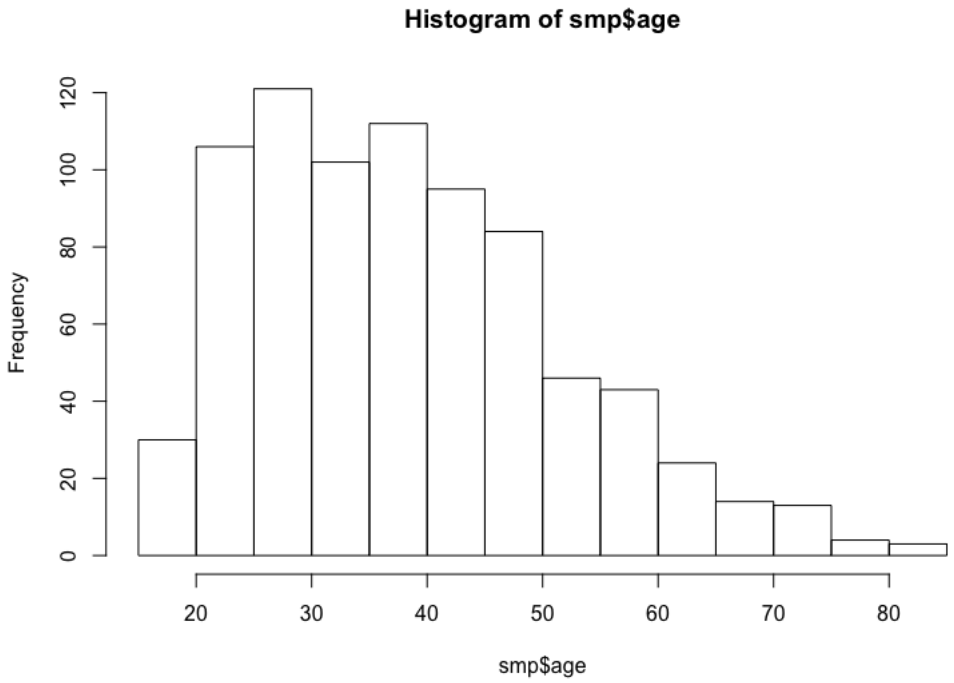
On peut augmenter la valeur de l'axe des ordonnées pour améliorer notre graphique. Tant qu'à faire, on pourrait orienter différemment les labels de cet axe :

```
> barplot(freqrelative,ylim = c(0,30), xlab = "freqrelative, ylim = c(0,30)")
barplot(freqrelative,ylim = c(0,30),las=1,xlab = "freqrelative, ylim = c(0,30),las=1,xlab")
```



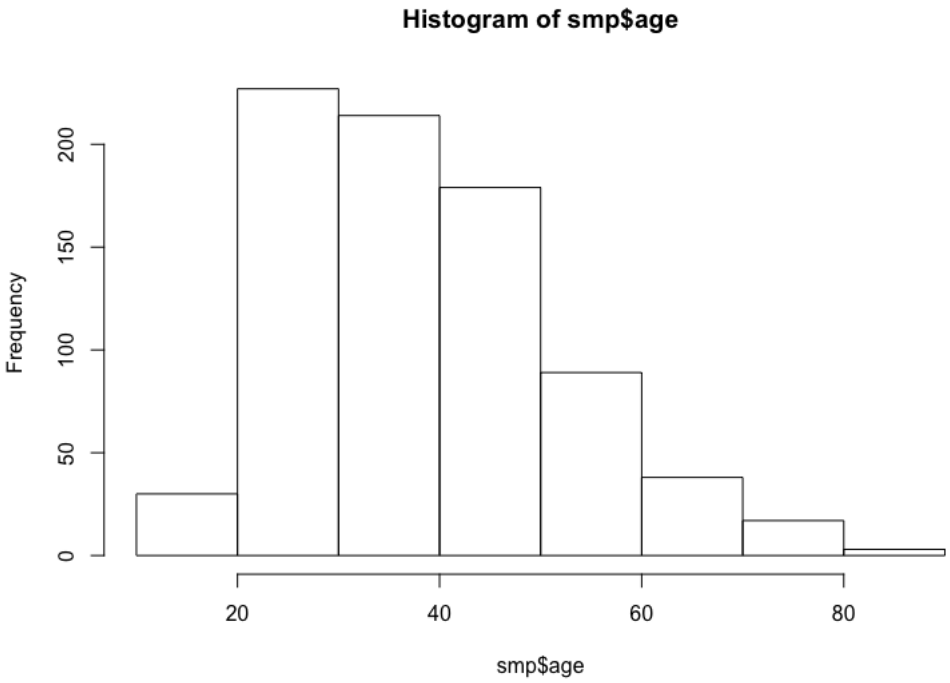
Pour représenter les variables numériques, il nous sera possible d'utiliser des histogrammes. Par exemple, pour la variable *age* :

```
> head(smp$age)
[1] 31 49 50 47 23 34
> summary(smp$age)
   Min. 1st Qu.  Median    Mean 3rd Qu.
   19.0    28.0    37.0    38.9    48.0
   Max.    NA's
   83.0       2
> hist(smp$age)
```



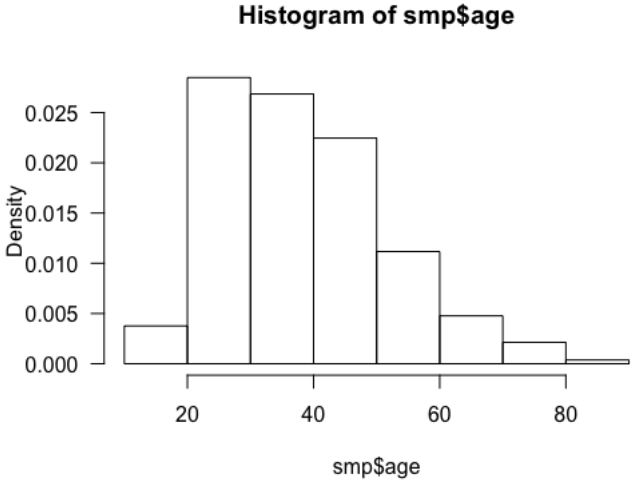
Pour réduire le nombre d'intervalles de classes, on peut faire :

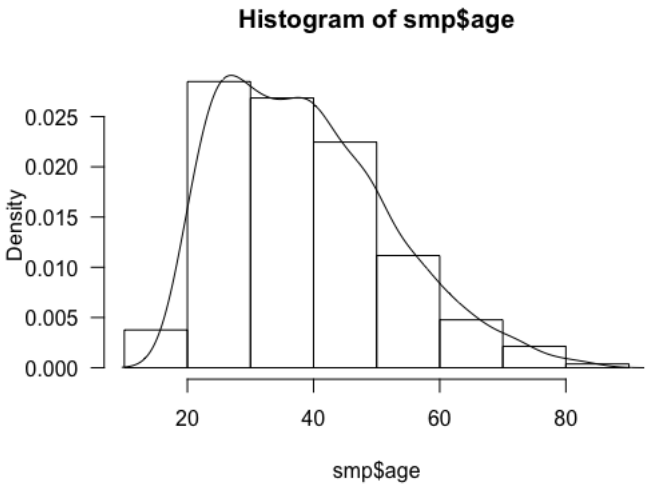
```
> hist(smp$age, nclass = 8)
```



Nous pouvons également sur le même graphique, tracer une ligne de densité non paramétrique. Il faudra au préalable transformer cet histogramme en histogramme de densité :

```
> hist(smp$age, nclass = 8, prob=TRUE, las=1)
> lines(density(smp$age, na.rm = TRUE))
```



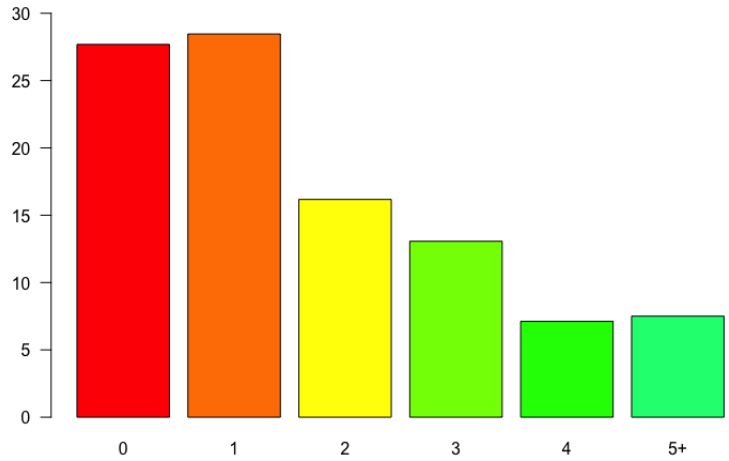

Histogram of smp$age
Density
0.025
0.020
0.015
0.010
0.005
0.000
20
40
60
80
smp$age

1.3 LAB 3 : Langage R Markdown - Génération d'un rapport automatique

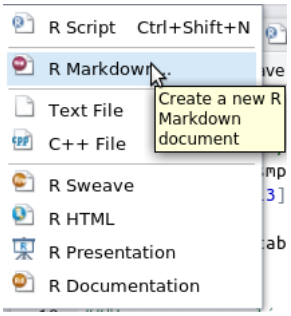
Nous avons vu à la session précédente qu'il était possible d'enregistrer toutes les commandes que l'on tape dans la console R dans un fichier de commandes ou fichier de script R, ce qui nous permet évidemment de rejouer l'analyse et faciliter la reproductibilité des résultats.
Nous allons voir qu'on peut également utiliser le langage R Markdown qui est un langage de formatage de documents qui permet de générer des rapports automatiques que l'on peut exporter au format HTML, au format PDF ou alors au format Microsoft Word.
Rmarkdown est intégré dans Rstudio depuis les versions les plus récentes et nous n'avons plus besoin d'installer le package pour pouvoir compiler des documents au format HTML.

Nous allons repartir du code écrit précédemment :

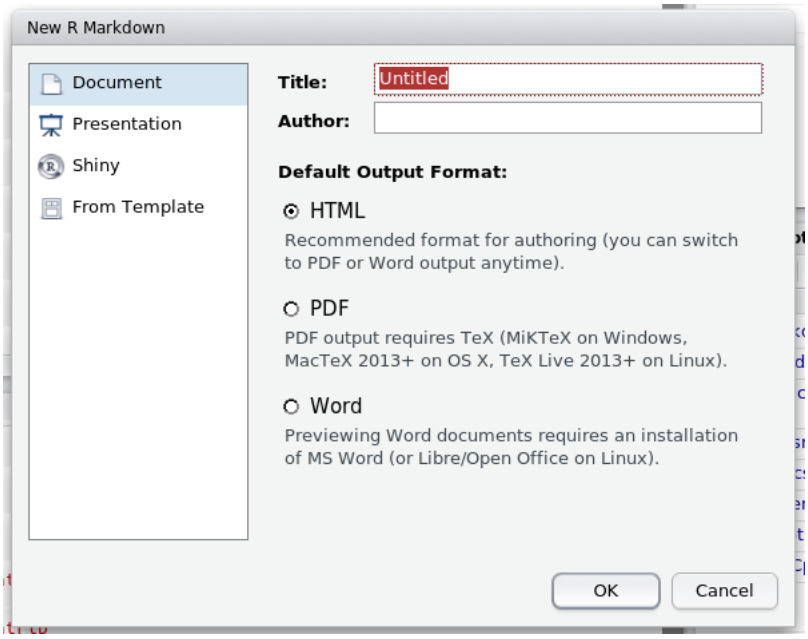
```
> setwd("~/Desktop/DIVERS_TEMPLATES/StatDesR/TP")
> smp <- read.csv2("DONNEES/smp2.csv")
> names(smp)
 [1] "age"          "prof"         "duree"        "discip"       "n.enfant"
 [6] "n.fratrie"    "ecole"        "separation"   "juge.enfant"  "place"
[11] "abus"         "grav.cons"    "dep.cons"     "ago.cons"     "ptsd.cons"
[16] "alc.cons"     "subst.cons"   "scz.cons"     "char"         "rs"
[21] "ed"           "dr"           "suicide.s"    "suicide.hr"   "suicide.past"
[26] "dur.interv"
> n.enfant.cat <- factor(smp$n.enfant)
> levels(n.enfant.cat) [6:13]<- "5+"
> table(n.enfant.cat)
n.enfant.cat
  0   1   2   3   4  5+
214 220 125 101  55  58
## Création d'un tableau de fréquence
> prop.table(table(n.enfant.cat))
n.enfant.cat
         0          1          2          3          4         5+
0.27684347 0.28460543 0.16170763 0.13065977 0.07115136 0.07503234
## Création d'un tableau de fréquence en pourcentage
> prop.table(table(n.enfant.cat))*100
n.enfant.cat
        0         1         2         3         4        5+
27.684347 28.460543 16.170763 13.065977  7.115136  7.503234
> barplot(prop.table(table(n.enfant.cat))*100,ylim = c(0,30),las=1,col = rainbow(12))
```



Nous allons à présent générer un autre document : le RMarkdown
Il faut donc installer les packages *markdown* et *rmarkdown* sur le site du CRAN ou directement via RSTUDIO.



Nous pouvons voir (sur la figure ci dessous qu'il existe plusieurs formats possibles). Nous allons choisir le premier : *Document* et la sortie au format *HTML*
Une fois les champs Titres et Auteurs remplies, R va générer un document standard pour la sortie choisie.

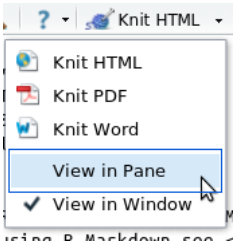


Le logiciel va alors nous générer un modèle de document avec une option *output* qui spécifie le type de document qui va être compilé :

```
---
title: "Exemple d'analyse"
author: "LATIF Mehdi"
date: "31/12/2017"
output: html_document
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```
```{r cars}
summary(cars)
```
```

Il est toujours possible de modifier de langage pour l'output :



En cliquant ici, rmarkdown.rstudio.com, nous trouverons les principales commandes pour la mise en forme d'un fichier rmarkdown.

Nous pouvons remarquer la présence de structure R que l'on appelle Chunk dans lesquelles se trouvent des suites de sequences d'instruction.

```
```{r, echo = TRUE}
summary(cars)
```
```{r pressure, echo=FALSE}
plot(pressure)
```
```

L'option echo permet d'afficher ou non la commande qui va être évaluée dans ce Chunk.

Nous rajoutons un phrase d'introduction et nous cliquons sur *Chunks/Insert Chunks* qui signifie à RSTUDIO qu'il faut évaluer les lignes de codes présent entre les accolades :

```
---
title: "Exemple_d_analyse"
author: "LATIF Mehdi"
date: "11/10/2016"
output: html_document
---
#Analyse des données sur la santé mentale en prison :
```{r}

```
```

Nous allons alors compléter le code markdown avec le code écrit plus haut :

```
---
title: "Exemple d'analyse"
author: "LATIF Mehdi"
date: "31/12/2017"
output:
  pdf_document: default
  html_document: default
---
#### Importation du data frame :
Ici, on a placé dans le chunk l'option *echo=FALSE* ce qui permet d'évaluer le bout de code sans l'afficher.
```{r echo=FALSE}
setwd("~/Desktop/DIVERS_TEMPLATES/StatDesR/TP")
smp <- read.csv2("DONNEES/smp2.csv")
```
Même si les commandes ne sont pas affichées, les commandes sont cependant évaluées. **La preuve**, on accède à la variable *smp*

Affichage du nom des variables présentes dans le DF :
On utilise l'option *eval = c(1,3)* pour signifier au logiciel que l'on souhaite évaluer seulement les lignes 1 et 3
```{r, eval = c(1,3)}
names(smp)
summary(smp)
str(smp)
```
Création d'une variable catégoricielle : n.enfant.cat :
```

````
```{r}
n.enfant.cat <- factor(smp$n.enfant)
levels(n.enfant.cat) [6:13]<- "5+"
table(n.enfant.cat)
```
#### Calcul de la table des fréquences (en %) :
```{r, eval = c(2)}
prop.table(table(n.enfant.cat))
prop.table(table(n.enfant.cat))*100
```
#### Affichage d'un diagramme en barre :
```{r echo = FALSE}
barplot(prop.table(table(n.enfant.cat))*100,ylim = c(0,30),las=1,col = rainbow(12), main = "
    fréquences (en %)")
```
````

Compilons ce code dans un fichier markdown et c'est magique. On obtient le résultat FirstMD.html.

1.4 LAB 4 : Tests d'associations et graphiques bivariés

Dans la session précédente, nous avons vu comment manipuler un dataframe, les variables contenues dans ce dernier et que l'on pouvait produire des résumés numériques pour des variables numériques ou qualitatives ainsi que des graphiques élémentaires à l'aide du langage RMarkdown.
Nous allons cette fois ci nous intéresser au croisement de deux variables, soit qualitatives, c'est à dire un **tableau de contigence**, soit une variable qualitative et une numérique ce qui donnera lieu à la comparaison de deux moyennes.

On rappelle que l'on peut charger directement un fichier de données à l'aide de la commande *load()*. Prenons celui que nous avons créé lors du LAB 1.

```
> setwd("~/Desktop/DIVERS_TEMPLATES/StatDesR/TP")
> load("smp_v1.rda")
> names(smp)
 [1] "age"          "prof"         "duree"        "discip"       "n.enfant"     "n.fratrie"
 [7] "ecole"        "separation"   "juge.enfant"  "place"        "abus"         "grav.cons"
[13] "dep.cons"     "ago.cons"     "ptsd.cons"    "alc.cons"     "subst.cons"   "scz.cons"
[19] "char"         "rs"           "ed"           "dr"           "suicide.s"    "suicide.hr"
[25] "suicide.past" "dur.interv"   "n.enfant.cat"
> cat("Nombre de variables : ",ncol(smp), " nombre d'observations : ", nrow(smp))
Nombre de variables :  27  nombre d'observations :  799
```

Nous retrouvons bien notre dataframe avec 799 observations et 27 variables.

Nous pouvons afficher un tableau d'effectif simple avec la commande *table()*. Par exemple, pour la variable consommation de substance :

```
> table(smp$subst.cons)

  0   1
587 212
> table(smp$subst.cons,useNA = "always")

   0    1 <NA>
 587  212    0
```

On peut également croisé cette dernière avec la variable abus qui est une variable binaire :

```
> table(smp$subst.cons, smp$abus)

      0   1
  0 441 140
  1 131  80
```

On obtient donc un tableau de contingence avec les modalités de la première variables qui apparaissent en ligne et celles de la seconde variable qui apparaissent en colonne (ici, 0 ou 1).

Nous allons stocker ce tableau de contingence dans une variable de type tableau :

```
> tab <- table(smp$subst.cons,smp$abus)
> tab

      0   1
  0 441 140
  1 131  80
```

On peut également calculer les fréquence grâce à la fonction *prop.table()*.

```
> prop.table(tab)

            0         1
  0 0.5568182 0.1767677
  1 0.1654040 0.1010101
```

Dans ce cas là, il nous est possible de spécifier sur quelle dimension (les lignes (dim $= 1$) ou les colonnes (dim $= 2$)) nous souhaitons effectuer le calcul des fréquences. Pour ce faire, nous allons utiliser l'option *margin*

— *margin = 1* tous les effectifs vont être rapportés aux totaux lignes.

— *margin = 2* tous les effectifs vont être rapportés aux totaux colonnes.

Dans le cas où *margin = 1*,

```
> prop.table(tab, margin = 1)

            0         1
  0 0.7590361 0.2409639
  1 0.6208531 0.3791469
```

Dans ce cas, 0.76 correspond aux 441 observations rapportées à l'ensemble des individus qui remplissent la modalité 0 de la variable *subst.cons*, donc l'effectif ligne.

Dans le cas où *margin = 2*,

```
> prop.table(tab, margin = 2)

            0         1
  0 0.7709790 0.6363636
  1 0.2290210 0.3636364
```

Dans ce cas, 0.77 correspond ici à 441 rapporté à l'effectif total pour cette première colonne.

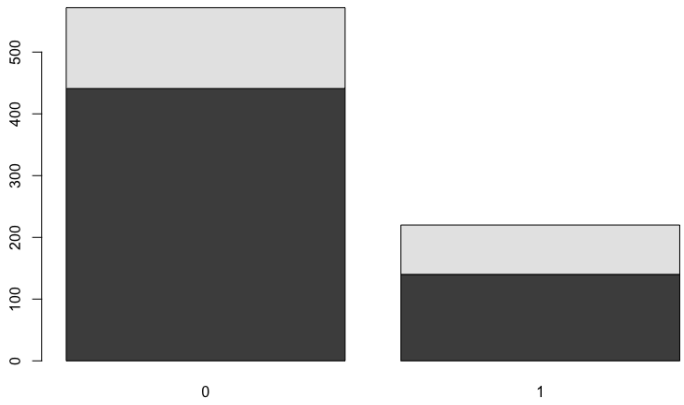
Plutôt que d'utiliser *table()*, nous pouvons également utiliser la commande *xtabs()* et qui présente l'avantage de fonctionner avec des formules dont on fera un plus large usage lorsque nous effectuerons des tests et pour les modèles.

Nous utilisons tilde ($\sim$) pour dénoter la relation entre deux variables. Ici c'est une relation qui est complètement symétrique donc les deux variables jouent le même rôle. Pour utiliser cette fonction, on indique après le tilde, le nom des variables que l'on souhaite insérer dans notre tableau de contingence séparées d'un + et ensuite, le nom du dataframe dans lequel elles se trouvent :

```
> xtabs(~ subst.cons + abus, smp)
           abus
subst.cons   0   1
         0 441 140
         1 131  80
```

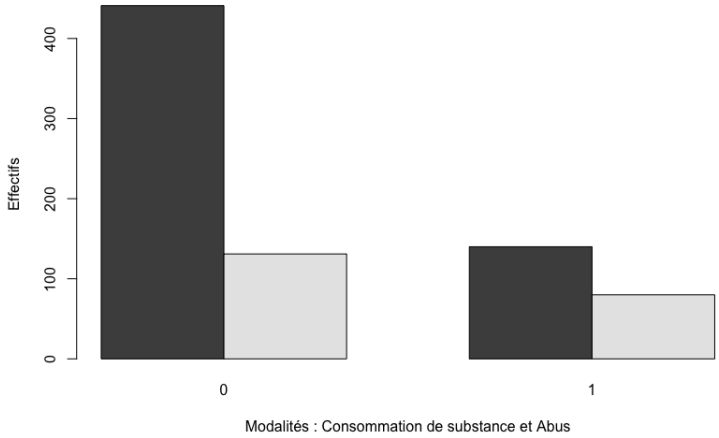
On obtient alors le même tableau d'effectif avec cette fois ci, le nom des variables qui apparaissent dans le tableau ; la première variables (subst.cons) est toujours représenter en ligne et la seconde (abus) en colonne. Dès lors, il est possible de représenter graphiquement les tableaux de contingence, La seule différence est qu'ici, par défaut, **R** va superposer les modalités :

```
> barplot(xtabs(~ subst.cons + abus, smp))
```



Pour afficher les modalités côte à côte, on utilise l'attribut *beside* = *TRUE* :

```
> barplot(xtabs(~ subst.cons + abus, smp),beside = TRUE, xlab="Modalités : Consommation de
    substance et Abus", ylab = "Effectifs")
```



Pour réaliser un test du χ^2, on utilise la méthode *chisq.test()* sur un tableau de contingence qui par défaut, inclut une correction de continuité :

```
> tab

    0   1
0 441 140
1 131  80
> chisq.test(tab)

  Pearson's Chi-squared test with Yates' continuity correction

data:  tab
X-squared = 14.052, df = 1, p-value = 0.0001779
```

Le résultat du test du χ^2 est ici 14.052, le degré de liberté[1] est de 1 et le degré de significativité associée (p-value) est égal à 0.0001779.

1. le degré de liberté se calcule tout simplement de la manière suivante :

$$\text{ddl} = \text{Nb de variables} - 1$$

Il est possible de stocker le résultat de ce test dans une variable, il nous suffira de l'appeler pour récupérer les résultats du test :

```
> res <- chisq.test(tab)
> res

        Pearson's Chi-squared test with Yates' continuity correction

data:  tab
X-squared = 14.052, df = 1, p-value = 0.0001779
```

Dans ce cas, il est possible d'accéder directement aux différents résultats obtenus par ce test :

— Si l'on souhaite afficher les effectifs observés :

```
> res$observed

     0   1
0 441 140
1 131  80
```

— Si l'on souhaite afficher les effectifs attendus sous hypothèse d'indépendance :

```
> res$expected

          0         1
0 419.6111 161.38889
1 152.3889  58.61111
```

— Si l'on souhaite afficher le test appliqué :

```
> res$method
[1] "Pearson's Chi-squared test with Yates' continuity correction"
```

— Si l'on souhaite afficher le résultat du test :

```
> res$statistic
X-squared
  14.0517
```

— Si l'on souhaite afficher le dégré de liberté (ou autres paramètres obtenus par le test)

```
> res$parameter
df
 1
```

— Si l'on souhaite afficher le dégré de significativité (p-value)

```
> res$p.value
[1] 0.0001778528
```

Si par la suite, on souhaite réaliser un test de Fisher, on utilisera la fonction *fisher.test()* avec la même syntaxe :

```
> tab

     0   1
0 441 140
```

```
  1 131  80
> fisher.test(tab)

  Fisher's Exact Test for Count Data

data:  tab
p-value = 0.0002193
alternative hypothesis: true odds ratio is not equal to 1
95 percent confidence interval:
 1.351339 2.728231
sample estimates:
odds ratio
  1.921985
```

Considérons maintenant la variable âge. Nous allons chercher à décrire l'âge en fonction de la variable correspondant à la consommation de substance (subst.cons)
Premièrement, nous allons afficher les modalités de la variable d'âge et le tableau d'effectif de subst.cons (ne pas oublier d'afficher au cas où, les valeurs non renseignées) :

```
 > head(smp$age)
[1] 31 49 50 47 23 34
> table(smp$subst.cons)

  0   1
587 212
> table(smp$subst.cons, useNA = "always")

   0    1 <NA>
 587  212    0
```

Ce qui va nous intéresser maintenant, c'est de décrire la variable âge en fonction des deux modalité de la variable consommation de substance (0 ou 1) c'est-à-dire oui il y a consommation ou non il n'y a pas consommation ; Cela revient à calculer des moyennes conditionnelles.
Pour se faire, nous allons pouvoir utiliser la fonction *tapply()*. Nous allons indiquer en paramètre le nom de la variable numérique que l'on souhaite caractériser (age), le nom de la variable catégorielle (subst.cons), ou critère de classification, et la commande que l'on souhaite utiliser, dans le cas présent, le calcul de la moyenne (mean)

```
> tapply(smp$age, smp$subst.cons, mean)
 0  1
NA NA
```

On voit ici que **R** nous renvoi les valeurs manquantes ; Cela signifie qu'il existe des valeurs manquantes pour l'une des deux variable.

```
> table(smp$age, useNA = "always")

  19   20   21   22   23   24   25   26   27   28   29   30   31   32   33   34   35   36   37   38
  39
  15   15   18   13   23   30   22   30   25   21   20   25   18   22   26   20   16   18   25   24
  23
  40   41   42   43   44   45   46   47   48   49   50   51   52   53   54   55   56   57   58   59
  60
  22   23   19   17   19   17   16   12   17   23   16   11   10    8   11    6   14    9    7    6
   7
  61   62   63   64   65   66   67   68   69   70   71   72   73   74   77   79   81   83 <NA>
   5    7    4    5    3    6    4    2    1    1    6    3    2    2    3    1    1    2    2
> table(smp$subst.cons, useNA = "always")

   0    1 <NA>
 587  212    0
```

On doit donc dire au logiciel de supprimer ces valeurs à l'aide de la fonction *na.rm = TRUE*

```
> tapply(smp$age, smp$subst.cons, mean,na.rm = TRUE)
       0        1
41.97099 30.36967
```

On obtient donc la moyenne de l'âge pour les détenus ne consommant pas de substance $(41,97099)$ et celle pour ceux qui en consomment $(30,36967)$.

Pour réaliser un test de Student, nous allons utiliser la commande *t.test()* en y indiquant les deux échantillons que l'on souhaite comparer. Dans notre cas, on souhaite étudier les âges pour lesquels il y a consommation de substance (*smp$age[smp$subst.cons == 0]*) et ceux pour lesquels il n'y a pas de consommation de substance (*smp$age[smp$subst.cons == 1]*).

```
> t.test(smp$age[smp$subst.cons == 0], smp$age[smp$subst.cons==1])

  Welch Two Sample t-test

data:  smp$age[smp$subst.cons == 0] and smp$age[smp$subst.cons == 1]
t = 15.24, df = 666.83, p-value < 2.2e-16
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 10.10664 13.09600
sample estimates:
mean of x mean of y
 41.97099  30.36967
```

On remarque que **R** n'effectue pas directement le test de Student mais une variante de ce dernier, le test de Welsh [2], qui comme nous l'avons expliquer dans le cours, ne nécessite pas l'égalité des variances comme condition de validité.
Si nous souhaitons réaliser un vrai test de Student, nous devons spécifier l'égalité des variances (*var.equal = TRUE*)de la manière suivante :

```
> t.test(smp$age[smp$subst.cons == 0], smp$age[smp$subst.cons == 1],var.equal = TRUE)

  Two Sample t-test

data:  smp$age[smp$subst.cons == 0] and smp$age[smp$subst.cons == 1]
t = 11.785, df = 795, p-value < 2.2e-16
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
  9.668959 13.533684
sample estimates:
mean of x mean of y
 41.97099  30.36967
```

2. En statistiques, le test t de Welch est une adaptation du test t de Student. Il peut être utilisé notamment pour tester statistiquement l'hypothèse d'égalité de deux moyennes avec deux échantillons de variances inégales.
Le test t de Welch définit le "t" statistique par la formule suivante :

$$t = \frac{\overline{X}_1 - \overline{X}_2}{\sqrt{\frac{s_1^2}{N_1} + \frac{s_2^2}{N_2}}}$$

où $\overline{X}$, s^2 et N correspondent respectivement à la moyenne, à sa variance et à la taille de l'échantillon. Contrairement au test t de Student, le dénominateur n'est "pas" basé sur une estimation de l'ensemble des variances.

Le calcul des degrés de liberté ν associés à cette estimation de la variance est approché par l'équation de Welch-Satterthwaite :

$$\nu = \frac{\left(\frac{s_1^2}{N_1} + \frac{s_2^2}{N_2}\right)^2}{\frac{s_1^4}{N_1^2 \cdot \nu_1} + \frac{s_2^4}{N_2^2 \cdot \nu_2}} = \frac{\left(\frac{s_1^2}{N_1} + \frac{s_2^2}{N_2}\right)^2}{\frac{s_1^4}{N_1^2 \cdot (N_1 - 1)} + \frac{s_2^4}{N_2^2 \cdot (N_2 - 1)}}.$$

Ainsi $\nu_i = N_i - 1$, les degrés de liberté sont associés à la n-ième estimation de la variance.

Précédemment, nous avons vu que la fonction *xtabs()* nous permettait d'utiliser une notation par formule ce qui est assez pratique puisque cela nous permet de décrire la relation entre plusieurs variables à l'aide d'une formule :

```
> xtabs(age~subst.cons,smp)
subst.cons
    0     1
24595  6408
```

Dans notre cas, les variables jouent un rôle **asymétrique** : Nous avons une *variable réponse*, c'est à dire une variable dépendante : l'âge et une *variable explicative* : la consommation de substances.
L'objectif est donc d'écrire une relation permettant d'*expliquer l'âge en fonction de la consommation de substances*.
Note : On rappelle que le paramètre smp dans un test permet d'indiquer le data frame dans lequel se trouvent les variables.

```
> t.test(age~subst.cons,smp)

  Welch Two Sample t-test

data:  age by subst.cons
t = 15.24, df = 666.83, p-value < 2.2e-16
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 10.10664 13.09600
sample estimates:
mean in group 0 mean in group 1
       41.97099        30.36967
```

Nous pouvons donc remarquer que nous obtenons les mêmes résultats qu'avec le test de Student réalisé plus haut, moyennant le fait que nous ne sommes plus obligés de préfixer les variables par le nom du dataframe.

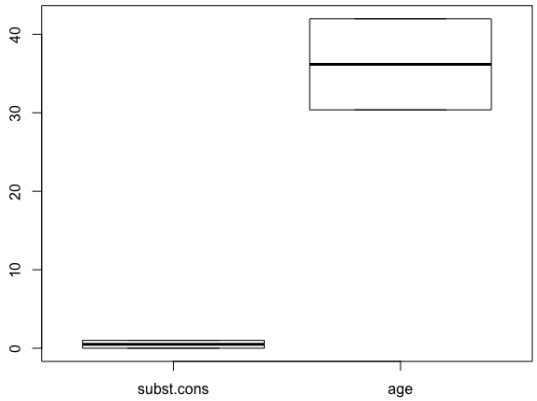
Nous avons vu qu'il nous était possible de calculer des moyennes conditionnelles à l'aide de la commande *t.apply()*. Si l'on souhaite effectuer ce calcul tout en conservant un attribut sous forme d'une formule, nous pouvons utiliser la commande *aggregate()* avec comme paramètre, la formule que l'on a définit précédemment pour les variables asymétriques. On doit cependant spécifier en attribut, la valeur que l'on souhaite calculer : ici, la moyenne (*mean*).

```
> aggregate(age ~ subst.cons, smp, mean)
  subst.cons      age
1          0 41.97099
2          1 30.36967
```

Note : Contrairement à la fonction *t.apply()*, dans *aggregate()*, nous n'avons pas besoin pas besoin de spécifier que l'on souhaite supprimer les valeurs manquantes ($na.rm = TRUE$) car cette commande est activée par défaut.

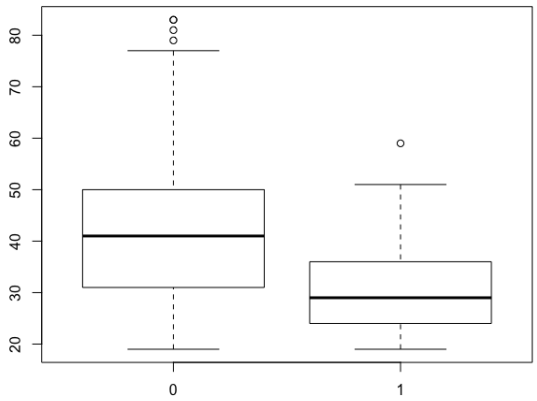
Enfin, un autre avantage de la fonction *aggregate()* est qu'elle nous renvoie directement un dataframe que l'on peut réutiliser pour réaliser des graphiques :

```
> boxplot(aggregate(age ~ subst.cons, smp, mean))
```



Si l'on souhaite générer une représentation graphique des distributions conditionnelles, il est encore une fois possible d'utiliser la notation par formule dans la fonction *boxplot()* :

```
> boxplot(age~subst.cons,smp)
```



Cette fois-ci, nous obtenons une représentation en forme de boites à moustaches pour chacune des modalités de la variable *subst.cons* avec en ordonnée, les valeurs prises par la variable *age*.

Note : Il existe beaucoup d'autres commandes qui peuvent nous permettre de réaliser des graphiques plus élaborer.

1.5 LAB 5 : ANOVA, Régression linéaire et logistique

Après avoir vu les principaux tests d'associations dans le cas où on croise deux variables, on va s'intéresser aux modèles d'ANOVA et des régressions linéaires et logistiques.

```
> setwd("~/Desktop/DIVERS_TEMPLATES/StatDesR/TP")
> smp <- read.csv2("DONNEES/smp2.csv")
> cat("Nombre de variables : ",ncol(smp), " nombre d'observations : ", nrow(smp))
Nombre de variables :  26  nombre d'observations :  799
> names(smp)
 [1] "age"          "prof"         "duree"        "discip"       "n.enfant"     "n.fratrie"
 [7] "ecole"        "separation"   "juge.enfant"  "place"        "abus"         "grav.cons"
[13] "dep.cons"     "ago.cons"     "ptsd.cons"    "alc.cons"     "subst.cons"   "scz.cons"
[19] "char"         "rs"           "ed"           "dr"           "suicide.s"    "suicide.hr"
[25] "suicide.past" "dur.interv"
```

Nous allons cette fois ci nous intéresser à un sous ensemble du dataframe et en particulier pour la variable profession, les individus qui sont soit sans-emploi soit qui ont profession intermédiaire soit qui sont cadre et on va regarder simplement les variables age, nombre enfant (n.enfant).
Dans un LAB précédent, nous avions vu la commande *subset()* qui permet de définir des sous ensembles de données à partir d'un dataframe et de filtres pour selectionner les lignes à inclure dans ce sous ensemble. Dans notre exemple, nous allons donc écrire :

subset(smp, prof ==\"sans emploi\" | prof ==\"prof.intermediaire\" | prof == \"cadre\", c(age, n.enfant,prof))

Note : Les opérateurs logiques dans R sont : | pour le OU et & pour le ET.
Dès lors, nous pouvons afficher les premières observations de ce sous ensemble

```
> head(subset(smp, prof == "sans emploi" | prof == "prof.intermediaire" | prof == "cadre", c(age, n
    .enfant, prof)),10)
   age n.enfant              prof
3   50        2 prof.intermediaire
5   23        1        sans emploi
11  31        0 prof.intermediaire
17  60        2 prof.intermediaire
23  32        0        sans emploi
27  32        1        sans emploi
31  24        1        sans emploi
32  38        3        sans emploi
35  29        2 prof.intermediaire
46  38        2 prof.intermediaire
```

Et le sauvegarder dans une nouvelle variable de type dataframe :

```
> smpb<-subset(smp, prof == "sans emploi" | prof == "prof.intermediaire" | prof == "cadre", c(age,
    n.enfant, prof))
> names(smpb)
[1] "age"       "n.enfant" "prof"
> cat("Nombre de variables : ",ncol(smpb), " nombre d'observations : ", nrow(smpb))
Nombre de variables :  3  nombre d'observations :  304
> str(smpb)
'data.frame': 304 obs. of  3 variables:
 $ age     : int  50 23 31 60 32 32 24 38 29 38 ...
 $ n.enfant: int  2 1 0 2 0 1 1 3 2 2 ...
 $ prof    : Factor w/ 8 levels "agriculteur",..: 7 8 7 7 8 8 8 8 7 7 ...
```

On peut également afficher un résumé de ce nouveau dataframe :

```
> summary(smpb)
      age           n.enfant                       prof
 Min.   :19.00   Min.   : 0.000   sans emploi       :222
 1st Qu.:27.00   1st Qu.: 0.000   prof.intermediaire: 58
 Median :36.00   Median : 1.000   cadre             : 24
 Mean   :38.42   Mean   : 1.648   agriculteur       :  0
```

```
 3rd Qu.:47.25   3rd Qu.: 3.000   artisan           :  0
 Max.   :83.00   Max.   :13.000   autre             :  0
                 NA's   :11       (Other)           :  0
```

Or, on peut s'apercevoir que R à conserver pour la variable prof, les anciens niveaux qui n'ont plus lieu d'être puisqu'ils ne sont plus vérifiés.
Nous les supprimer en utilisant la commande *factor()* qui va recalculer automatiquement les niveaux de notre variable de sorte que cette fois-ci, il ne restera que les trois modalités qui nous intéressent :

```
> smpb$prof <- factor(smpb$prof, labels=c("cadre", "intermédiaire", "sans emploi"))
> summary(smpb)
      age          n.enfant                prof
 Min.   :19.00   Min.   : 0.000   cadre         : 24
 1st Qu.:27.00   1st Qu.: 0.000   intermédiaire: 58
 Median :36.00   Median : 1.000   sans emploi   :222
 Mean   :38.42   Mean   : 1.648
 3rd Qu.:47.25   3rd Qu.: 3.000
 Max.   :83.00   Max.   :13.000
                 NA's   :11
```

Nous pouvons également réaliser un tableau d'effectif pour vérifier que nous ne nous sommes pas trompé lors de la création du sous ensemble :

```
> table(smp$prof)

      agriculteur           artisan             autre            cadre           employe
                6                90                31               24               135
          ouvrier prof.intermediaire       sans emploi
              227                58               222
> table(smpb$prof)

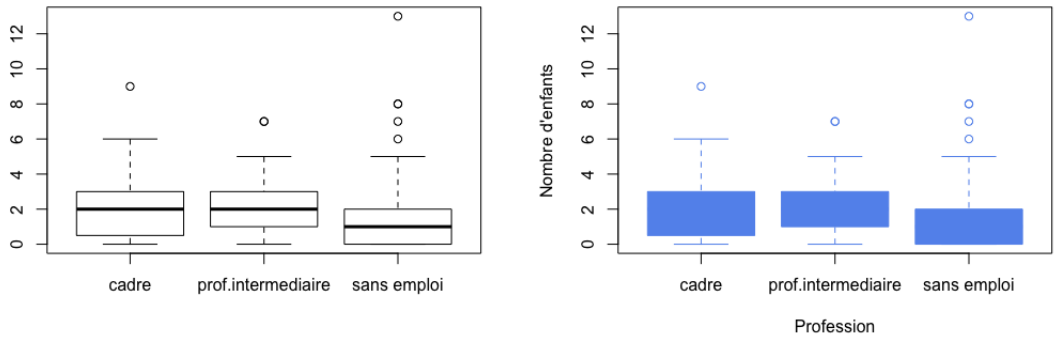
        cadre intermédiaire   sans emploi
           24           58           222
```

Si maintenant, nous souhaitons résumer le nombre d'enfant moyen en fonction de la professions dans le nouveau dataframe, nous utiliserons la commande *aggregate()* avec une formule donc on décrit le nombre d'enfants par la variable profession en utilisant le tilde ($\sim$) pour indiquer la relation entre les deux variables. (ne pas oublier de donner l'opération que l'on souhaite effectuer, ici c'est la moyenne)

```
> aggregate(n.enfant ~ prof, data=smpb, mean)
           prof n.enfant
1         cadre 2.166667
2 intermédiaire 2.107143
3   sans emploi 1.469484
```

On peut effectuer une représentation graphique de la moyenne du nombre d'enfants par profession à l'aide de boxplot ; On passera alors en paramètre la même foncton mettant en relation les variables (n.enfant $\sim$ prof, data=smpb)

```
> boxplot(n.enfant ~ prof, data=smpb, xlab="Profession", ylab="Nombre d'enfants")
> boxplot(n.enfant ~ prof, data=smpb, xlab="Profession", ylab="Nombre d'enfants", col="
    cornflowerblue", border="cornflowerblue")
```



On obtient donc, pour chacune des modalités, la distribution du nombre d'enfants, ici en ordonnée.

Nous allons maintenant nous intéresser aux modèle de régressions que nous pouvons réaliser sur de telles variables.
Tout d'abord, si l'on souhaite réaliser une ANOVA, alors la commande pour réaliser ces analyse ou les modèles de régression linéaire s'appelle *lm()*.
On peut également regarder l'aide en ligne pour cette commande en tapant la commande *help(lm)*.

```
> help(lm)
```

```
Fitting Linear Models : lm is used to fit linear models.
It can be used to carry out regression, single stratum analysis of variance and analysis of
    covariance (although aov may provide a more convenient interface for these).
lm(formula, data, subset, weights, na.action, method = "qr", model = TRUE, x = FALSE, y = FALSE, qr
    = TRUE, singular.ok = TRUE, contrasts = NULL, offset, ...)
```

On peut voir qu'il s'agit d'une commande très générale pour réaliser des modèles linéaires et cela inclut la régression linéaire et puis la régression sur variables indicatrices ou variables catégorielles qui est le cas particulier de l'ANOVA.
Nous allons donc effectuer une regression que l'on a étudié précédemment à savoir le nombre d'enfants décrit par la profession dans le data-frame smpb et puis on va stocker ce résultat-là dans une variable qu'on appellera *m*

```
> m <- lm(n.enfant ~ prof, smpb);m

Call:
lm(formula = n.enfant ~ prof, data = smpb)

Coefficients:
      (Intercept)  profintermédiaire     profsans emploi
          2.16667           -0.05952           -0.69718
```

Ici on a en fait stocké le résultat de notre analyse de régression dans la variable « m »; on peut regarder à quoi correspond "m", en fait ça nous rappelle l'instruction qu'on a tapée c.à.d. la commande lm(), la formule et le data-frame dans lequel on trouve les variables et puis des coefficients.
Dans le cadre d'une ANOVA, on utilisera la commande *drop1()* qui nous permet en donnant le nom d'une variable et en spécifiant un test de Fisher Snedecor de fournir un tableau d'analyse de variance.

```
> drop1(m, test = "F")
Single term deletions

Model:
n.enfant ~ prof
```

```
       Df Sum of Sq    RSS    AIC F value  Pr(>F)
<none>              947.74 349.96
prof    2     25.05 972.79 353.60  3.8325 0.02276 *
---
Signif. codes:  0 ?***' 0.001 ?**' 0.01 ?*' 0.05 ?.' 0.1 ? ' 1
```

Ici, on obtient pour la variable explicative « profession » à deux degrés de liberté, la somme des carrés correspondante (25.05) et la valeur de « F » correspondant au test de l'analyse de variance (3.8325). La statistique de test est alors de 3.83 avec pour degré de significativité 0.02.
Il est également possible d'établir un modèle linéaire pour deux variables numériques ; prenons par exemple la variable nombre d'enfant et l'âge. On applique alors la commande suivante :

```
> m <- lm(n.enfant ~ age, data=smpb);m

Call:
lm(formula = n.enfant ~ age, data = smpb)

Coefficients:
(Intercept)          age
   -1.00902      0.06849
```

Cette fois ci, on obtient deux coefficients, l'intercept qui représente le terme d'ordonnée à l'origine et « age » qui va représenter la pente.
Dans le cas précédent, pour obtenir les tests associés à ce modèle linéaire, on tapera simplement la commande *summary()* : suivante :

```
> summary(m)

Call:
lm(formula = n.enfant ~ age, data = smpb)

Residuals:
    Min      1Q  Median      3Q     Max
-4.2646 -0.9087 -0.2511  0.5708  9.0094

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -1.009022   0.262696  -3.841  0.00015 ***
age          0.068488   0.006357  10.773  < 2e-16 ***
---
Signif. codes:  0 ?***' 0.001 ?**' 0.01 ?*' 0.05 ?.' 0.1 ? ' 1

Residual standard error: 1.546 on 291 degrees of freedom
  (11 observations deleted due to missingness)
Multiple R-squared:  0.2851,  Adjusted R-squared:  0.2827
F-statistic: 116.1 on 1 and 291 DF,  p-value: < 2.2e-16
```

On obtient cette fois ci un tableau avec les coefficients de régression et les tests t associés. Dans ce cas ci, la pente est évaluée à 10.77 avec un degré de significativité inférieur à 2.10^{-16}.

Depuis le début de ce LAB, nous travaillons sur un sous ensemble du dataframe smp mais il faut savoir que la commande *lm()* permet d'utiliser directement une option *subset* qui permet des modèles de régressions sur des ensembles d'observations entier ; il nous faut alors donner dans la fonction *lm()*, les filtres que l'on souhaite utiliser. Nous pouvons donc effectuer les mêmes calculs sur le dataframe complet smp en précisant les filtres que l'on veut appliquer ; Dans notre cas : *prof == "sans emploi" | prof == "prof.intermediaire" | prof == "cadre"*.

```
> m <- lm(n.enfant ~ age, data=smp, subset = (prof == "sans emploi" | prof == "prof.intermediaire"
    | prof == "cadre"));m

Call:
lm(formula = n.enfant ~ age, data = smp, subset = (prof == "sans emploi" |
    prof == "prof.intermediaire" | prof == "cadre"))

```

```
Coefficients:
(Intercept)          age
   -1.00902      0.06849

> summary(m)

Call:
lm(formula = n.enfant ~ age, data = smp, subset = (prof == "sans emploi" |
    prof == "prof.intermediaire" | prof == "cadre"))

Residuals:
    Min      1Q  Median      3Q     Max
-4.2646 -0.9087 -0.2511  0.5708  9.0094

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -1.009022   0.262696  -3.841  0.00015 ***
age          0.068488   0.006357  10.773  < 2e-16 ***
---
Signif. codes:  0 ?***' 0.001 ?**' 0.01 ?*' 0.05 ?.' 0.1 ? ' 1

Residual standard error: 1.546 on 291 degrees of freedom
  (17 observations deleted due to missingness)
Multiple R-squared:  0.2851,  Adjusted R-squared:  0.2827
F-statistic: 116.1 on 1 and 291 DF,  p-value: < 2.2e-16
```

On constate alors que les résultats sont identiques.
L'intérêt ici, c'est que l'on peut utiliser à la fois une notation par formule, on décrit la relation entre le nombre d'enfants qui est la variable de réponse et l'âge qui est la variable explicative, ces variables se trouvent dans le data-frame qui s'appelle smp. Par contre ce data-frame-là va être filtré selon les critères qui sont indiqués (dans la commande) dans l'option subset. Donc en particulier on ne va s'intéresser qu'aux individus qui remplissent les conditions profession égal soit sans emploi, soit profession intermédiaire, soit cadre.

Lorsque l'on a un modèle de régression, on peut appliquer la commande *coef()* pour afficher les coefficients de ce modèle.

```
> coef(m)
(Intercept)          age
-1.00902159  0.06848829
```

Il est également possible d'indexer ces coefficients par le numéro de position dans le vecteur *coef(m)*

```
> coef(m)[1]
(Intercept)
  -1.009022
> coef(m)[2]
       age
0.06848829

> cat("Coefficient directeur du modèle de régression linéaire : ", coef(m)[2])
Coefficient directeur du modèle de régression linéaire :  0.06848829
> cat("Ordonnée à l'origine du modèle de régression linéaire : ", coef(m)[1])
Ordonnée à l'origine du modèle de régression linéaire :  -1.009022

> coef(m)["(Intercept)"]
(Intercept)
  -1.009022
> coef(m)["age"]
       age
0.06848829
```

Si l'on souhaite obtenir les intervalles de confiance, il nous faut alors saisir la commande *confint()* sur le modèle de régression étudié :

```
> confint(m)
                  2.5 %      97.5 %
(Intercept) -1.52604619 -0.49199700
```

```
age           0.05597581   0.08100076
```

On obtient alors les intervalles de confiance à 95% (par exemple, l'intervalle de confiance à 95% pour le coefficient directeur du modèle est $[0.056, 0.081]$) ;
Il est possible d'obtenir un tableau d'analyse de variance associé à la régression à l'aide de la commande *anova()* :

```
> anova(m)
Analysis of Variance Table

Response: n.enfant
           Df Sum Sq Mean Sq F value    Pr(>F)
age         1 277.35  277.35  116.05 < 2.2e-16 ***
Residuals 291 695.44    2.39
---
Signif. codes:  0 ?***' 0.001 ?**' 0.01 ?*' 0.05 ?.' 0.1 ? ' 1
```

Lorsque l'on souhaite réaliser des prédiction sur des valeurs non nécessairement observées, on peut utliser la commande *predict()* ? Dans ce cas là, on va passer en paramètre de la fonction, le nom de la variable de laquelle est stocké le modèle de régression linéaire ainsi qu'un dataframe dans lequel nous allons indiquer pour la variable explicative choisie, les valeurs pour lesquelles on souhaite effectuer une prédiction. On pourra également obtenir les intevalles de confiance associés en ajoutant l'option *interval="confidence"* :

```
> predict(m, data.frame(age=c(20, 30, 40)))
        1         2         3
0.3607441 1.0456270 1.7305099

> predict(m, data.frame(age=c(20, 30, 40)),interval="confidence")
        fit        lwr       upr
1 0.3607441 0.06588459 0.6556037
2 1.0456270 0.83652266 1.2547313
3 1.7305099 1.55212966 1.9088901
```

On obtient alors sous la colonne « fit » les valeurs prédites, les colonnes « lwr » et « upr » représentent quant à elle, les bornes inférieures et supérieures des intervalles de confiance à 95% pour la prévision.

En dehors de la régression linéaire, on peut s'intéresser à la régression logistique.
Dans ces cas-là on prendra par exemple une variable binaire. Par exemple, nous pouvons construire une telle variable à partir de la variable représentant le nombre d'enfants. On va s'intéresser au nombre d'enfants supérieur à 2. Dans ces cas-là on codera 1 sinon on code 0. On utilise alors la commande *ifelse()* et on réalise un test logique donc « est-ce-que le nombre d'enfants est supérieur à deux ? » Dans ces cas-là on associe la valeur 1 sinon on associe la valeur 0.

```
> table(smp$n.enfant, useNA = "always")

   0    1    2    3    4    5    6    7    8    9   10   11   13 <NA>
 214  220  125  101   55   31    7    7    7    2    2    1    1   26
> smp$n.enfant.bin <- ifelse(smp$n.enfant > 2, 1, 0)
> table(smp$n.enfant.bin, useNA = "always")

   0    1 <NA>
 559  214   26
```

Il y a donc 214 individus qui ont plus de 2 enfants.

Dans le cadre de la regression logistique, nous utliserons la commande *glm()*

```
> help(glm)
```

```
Fitting Generalized Linear Models : glm is used to fit generalized linear models, specified by
    giving a symbolic description of the linear predictor and a description of the error
    distribution...
glm(formula, family = gaussian, data, weights, subset,na.action, start = NULL, etastart, mustart,
    offset, control = list(...), model = TRUE, method = "glm.fit", x = FALSE, y = TRUE, contrasts =
     NULL, ...)
```

Pour la fonction *glm()*,on utilisera avec toujours une notation par formule. La seule différence c'est qu'il faudra indiquer le type de régression qu'on souhaite effectuer.
Nous allons donc créer un modèle de régression logistique avec pour formule, le nombre d'enfants en fonction de l'âge. L'écriture de la fonction est identque mais cette fois ci, nous avons dichotomisé la variable. On prendra toujours les données dans le dataframe smp et cette fois ci, on indique en option que l'on veut effectuer une régression logistique en déclarant dans dans l'option de famille "binomiale" avec comme échelle de lien le "logit"

```
> m <- glm(n.enfant.bin ~ age, data=smp, family=binomial("logit"));m

Call:  glm(formula = n.enfant.bin ~ age, family = binomial("logit"),
    data = smp)

Coefficients:
(Intercept)          age
   -3.82709      0.06949

Degrees of Freedom: 772 Total (i.e. Null);  771 Residual
  (26 observations deleted due to missingness)
Null Deviance:      912.1
Residual Deviance: 794.2  AIC: 798.2
> summary(m)

Call:
glm(formula = n.enfant.bin ~ age, family = binomial("logit"),
    data = smp)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.8551  -0.7525  -0.5326   0.8763   2.1301

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -3.827089   0.312803 -12.235   <2e-16 ***
age          0.069487   0.007016   9.904   <2e-16 ***
---
Signif. codes:  0 ?***' 0.001 ?**' 0.01 ?*' 0.05 ?.' 0.1 ? ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 912.06  on 772  degrees of freedom
Residual deviance: 794.16  on 771  degrees of freedom
  (26 observations deleted due to missingness)
AIC: 798.16

Number of Fisher Scoring iterations: 4
```

Cette fois-ci on a la variable explicative (age) avec la valeur du coefficient de régression (0.069487) sur l'échelle du log odds [3].

3. Le logit ,accronyme de log-odds unit, est l'unité de mesure sur l'axe de la variable. La fonction logit est une fonction mathématique utilisée principalement en statistiques pour la régression logistique et en inférence bayésienne pour transformer les probabilités sur $[0, 1]$ en evidence sur $\mathbb{R}$.
Son expression est :

$$\text{logit}(p) = \ln\left(\frac{p}{1-p}\right)$$

où p est défini sur $]0; 1[$

The logit function is the inverse of the sigmoidal "logistic" function or logistic transform used in mathematics, espe-

cially in statistics. When the function's variable represents a probability p, the logit function gives the log-odds, or the logarithm of the odds $p/(1?p)$. The logit of a number p between 0 and 1 is given by the formula :

$$\text{logit}(p) = \log\left(\frac{p}{1-p}\right) = \log(p) - \log(1-p) = -\log\left(\frac{1}{p} - 1\right)$$

II Université de Nantes

III Théorie de la statistique

IV Div'R

V Représentations graphiques

VI Programmation avec R

VII Fonctions usuelles et aide mémoire