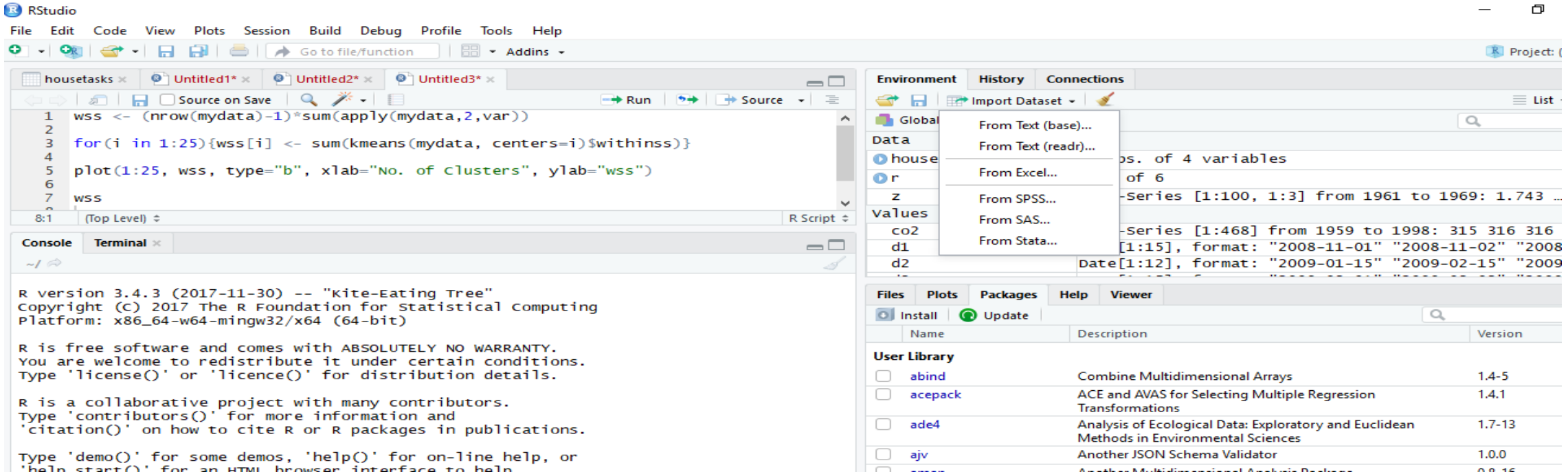
Importation des données

L'importation de données peut se faire via l'onglet "Import Data set" de la fenêtre Environnement de RStudio, ou bien en utilisant des lignes de commandes.

Selon le type de fichier txt, un fichier xls, un fichier xlsx ou encore d'un fichier csv provenant de différents tableurs etc….



Les données importées doivent être structurées selon un certain schéma.

Elles doivent correspondre à la structure utilisée par les fonctions qui seront employées pour mener les différents types d'analyses statistiques (descriptives, inférentielles, prédictives).

On va vous montrer comment importer facilement et efficacement vos données dans le logiciel R depuis le logiciel Excel. Pour cela:

- expliquer comment organiser votre espace de travail en utilisant la fonctionnalité “Project”
- exposer comment structurer vos données au format tidy avant l’importation
- faire quelque recommandations
- présenter la fonction “*read.csv2*” permettant de faire l’importation
- montrer comme visualiser les données importées

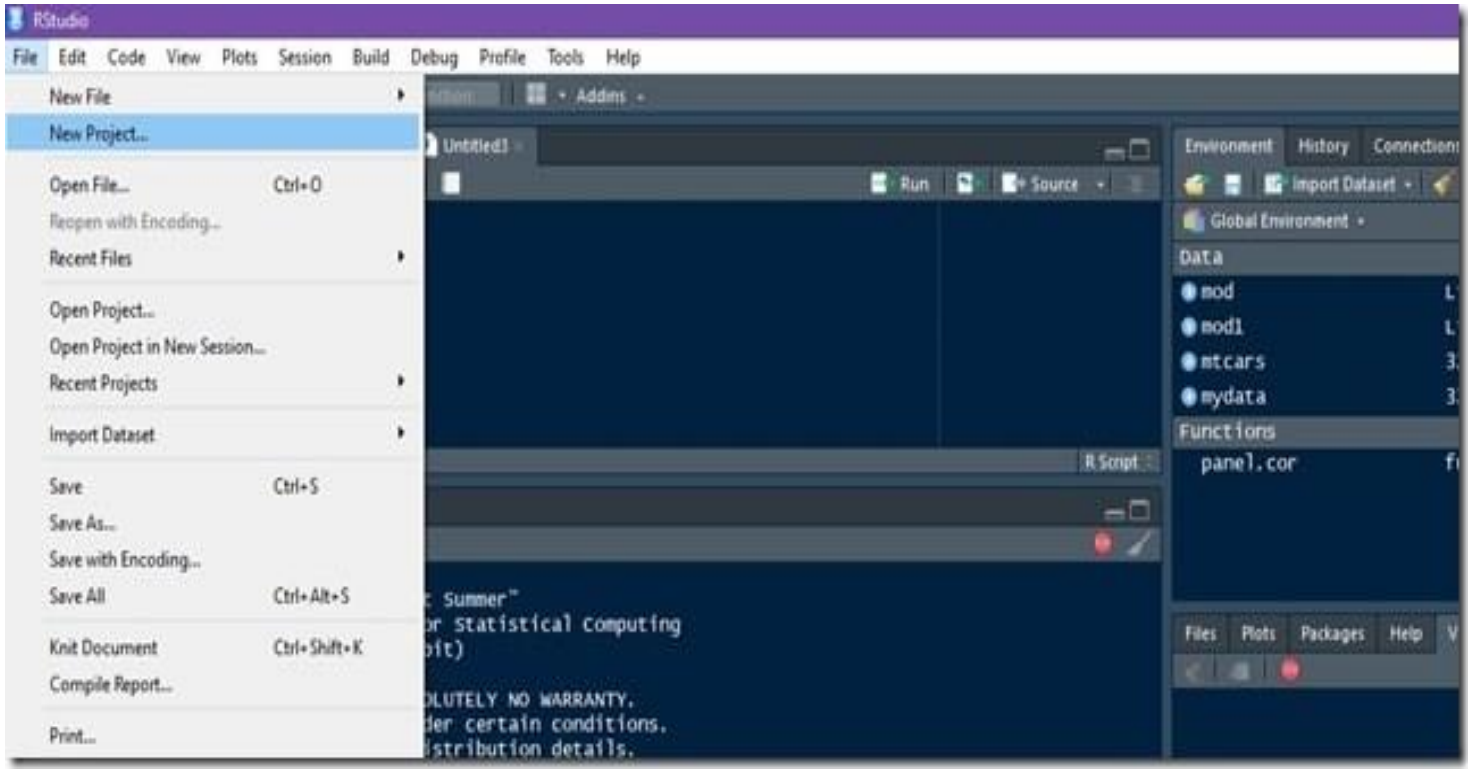
Organisation sous R

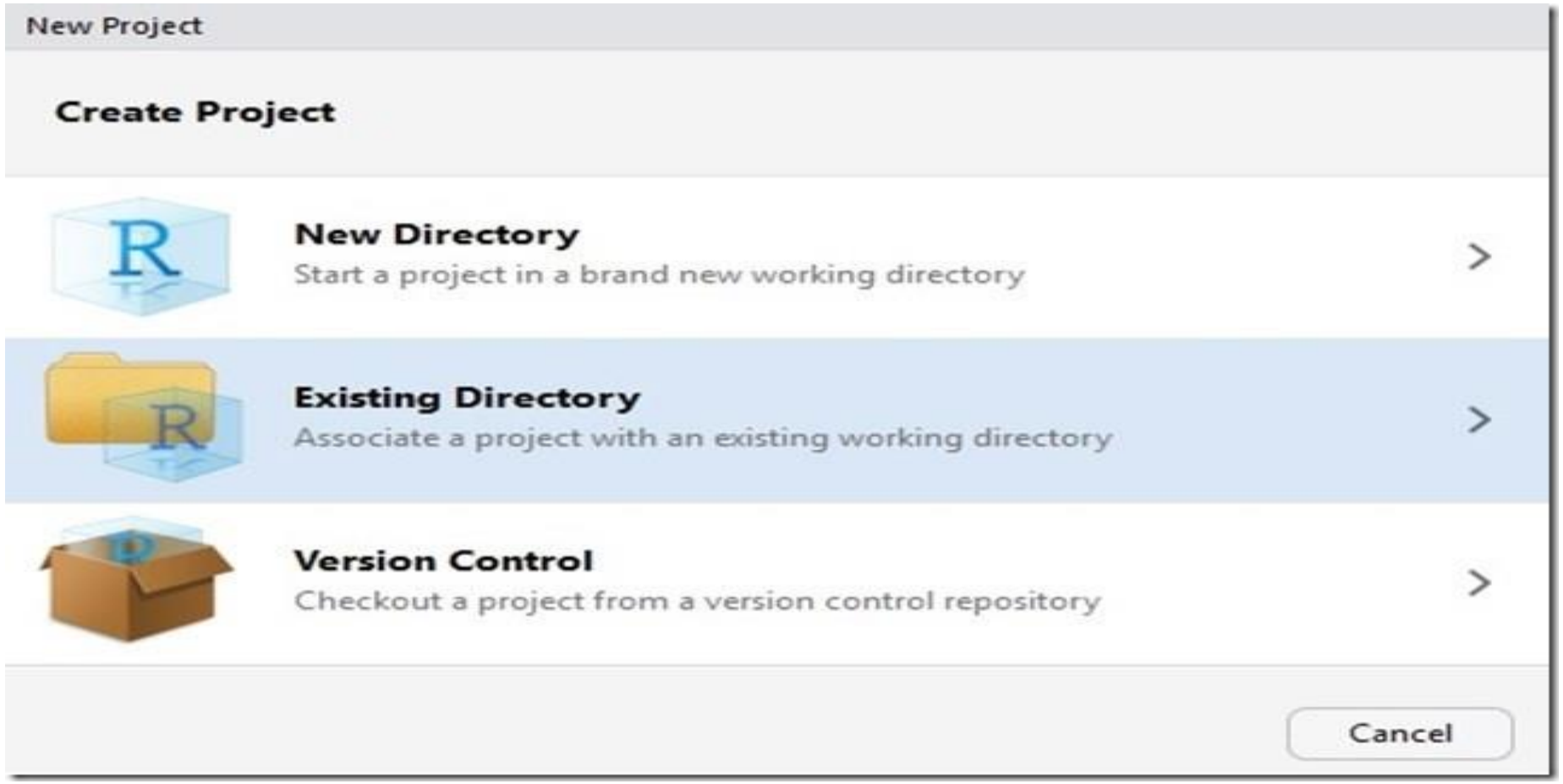
- R studio dispose d'une fonctionnalité "**Projet**" qui permet d'organiser les travaux concernant R et R Studio.
- un système de dossiers mais spécifique à R.
- C'est un peu comme si au lieu d'avoir un seul R et RStudio, vous en aviez plusieurs, un pour chaque dossier de travail.

A chaque projet R est associée un ***working directory*** (répertoire de travail) localisé au même endroit que le projet. Cela est vraiment pratique.

En plus du working directory, un ***workspace*** est associé au projet R. Il s'agit de tous les objets (données, tableau , constante etc…) que vous allez créer lors de vos analyses. L'ensemble des éléments composant le workspace sont visibles dans la fenêtre "Environnement".

En plus du working directory et du workspace, un fichier contenant l'**historique** des commandes est également créé avec le projet R




New Project
Create Project
New Directory
Start a project in a brand new working directory
Existing Directory
Associate a project with an existing working directory
Version Control
Checkout a project from a version control repository
Cancel

Structurez vos données au format tidy avant l'importation

Les données importées doivent être structurées d'une certaine façon pour pouvoir, ensuite, être utilisées dans les différentes fonctions de R.

Cette structure peut être résumée par deux grands principes :

- Chaque variable mesurée doit correspondre à une colonne.
- Si plusieurs observations ont été faites pour une variable donnée, ces observations doivent être sur des lignes différentes.

```
my_iris

##   id Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1  1          5.1         3.5          1.4         0.2
## 2  2          4.9         3.0          1.4         0.2
## 3  3          4.7         3.2          1.3         0.2
## 4  4          4.6         3.1          1.5         0.2
## 5  5          5.0         3.6          1.4         0.2
```

Exemple :
nous ayons relevé deux indicateurs biologiques (la créatinine et la glycémie) de 50 patients à deux temps différents (au cours de la première et de la 3ème semaine de septembre). Dans une première approche, les données pourraient être reportées dans un tableau sous cette forme :

id_patient	créatinine_semaine1 (mL/min)	créatinine_semaine3 (mL/min)	glycémie_semaine1 (g/L)	glycémie_semaine3 (g/L)
p1	0,95	1,11	99,41	103,61
p2	0,59	0,74	92,37	118,88
p3	0,75	1,13	89,28	89,25
....				
p10	0,35	0,66	89,77	105,13
....				
p50	0,81	0,67	107,76	100,2

Ce format ne correspond pas aux critères énoncés. Chaque variable mesurée (créatinine et glycémie) n'est pas contenue dans une seule colonne mais dans deux.

De ce fait, le deuxième critère n'est pas respecté non plus puisque les deux mesures d'une même variable (la mesure de la glycémie de la première semaine et la mesure de la glycémie la deuxième semaine), ne sont pas sur des lignes différentes, mais sur une même ligne.

Les données pourraient également avoir été reportées sous la forme transposée du premier tableau, mais cela ne correspond pas non plus aux critères énoncés.

id_patient	p1	p2	p3	....	p10	....	p50
glycémie_semaine1 (g/L)	0,95	0,59	0,75		0,35		0,81
glycémie_semaine3 (g/L)	1,11	0,74	1,13		0,66		0,67
créatinine_semaine1 (mL/min)	99,41	92,37	89,28		89,77		107,76
créatinine_semaine3 (mL/min)	103,61	118,88	89,25		105,13		100,2

id_patient	temps	glycémie (g/L)	créatinine (mL/min)
p1	s1	0,95	99,41
p1	s3	1,11	103,61
p2	s1	0,59	92,37
p2	s3	0,74	118,88
p3	s1	0,75	89,28
p3	s3	1,13	83,25
....	....	....	....
....	....	....	....
p10	s1	0,35	89,77
p10	s3	0,66	105,13
....	....	....	....
....	....	....	....
p50	s1	0,81	107,76
p50	s3	0,67	100,2

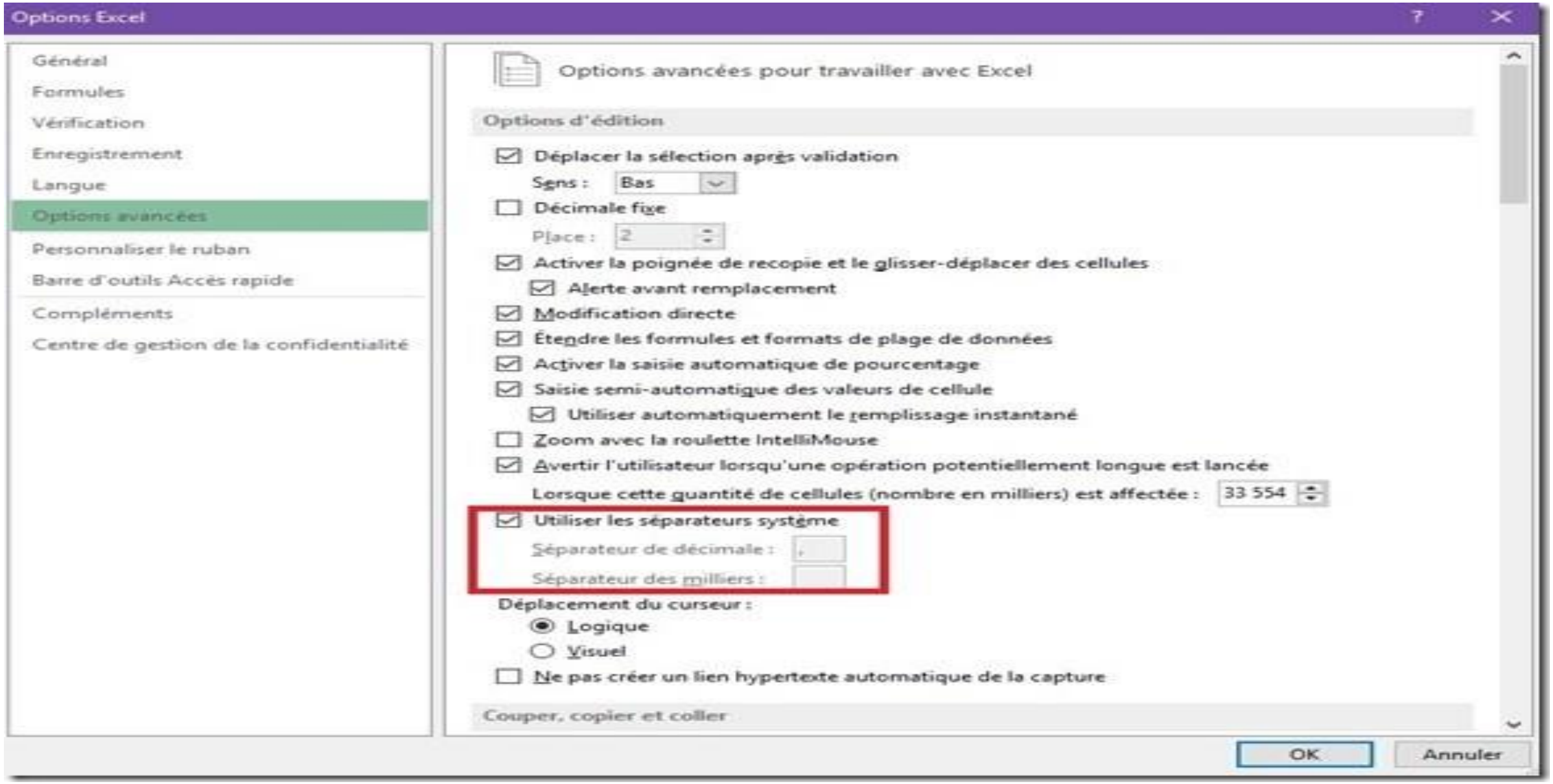
chaque constante sanguine ne correspond qu'à une seule colonne ; cela a nécessité la création d'une colonne "temps". Et chaque mesure d'une constante sanguine donnée, pour un patient donné, est reportée dans une ligne différente. Ce format est appelé **tidy data** (données rangés). Les grands principes ce format "tidy" ont été défini par Hadley Wickham.

https://cran.r-project.org/web/packages/tidyr/vignettes/tidy-data.html

Recommandations avant d'importer les bases de données

Pour importer vos données depuis Excel dans le logciciel R, il est indispensable de bien respecter le format du séparateur décimal défini dans Excel.

Par défaut, le séparateur décimal défini par Excel est la virgule. Pour le vérifier vous pouvez aller dans le menu Fichier, puis Options (tout en bas) puis Options Avancées. Si ce n'est pas le cas, changez l'option pour utiliser la virgule.


Options Excel
Général
Formules
Vérification
Enregistrement
Langue
Options avancées
Personnaliser le ruban
Barre d'outils Accès rapide
Compléments
Centre de gestion de la confidentialité
Options avancées pour travailler avec Excel
Options d'édition
Déplacer la sélection après validation
Sens : Bas
Décimale fixe
Place : 2
Activer la poignée de recopie et le glisser-déplacer des cellules
Alerte avant remplacement
Modification directe
Étendre les formules et formats de plage de données
Activer la saisie automatique de pourcentage
Saisie semi-automatique des valeurs de cellule
Utiliser automatiquement le remplissage instantané
Zoom avec la roulette IntelliMouse
Avertir l'utilisateur lorsqu'une opération potentiellement longue est lancée
Lorsque cette quantité de cellules (nombre en milliers) est affectée : 33 554
Utiliser les séparateurs système
Séparateur de décimale : ,
Séparateur des milliers :
Déplacement du curseur :
Logique
Visuel
Ne pas créer un lien hypertexte automatique de la capture
Couper, copier et coller
OK
Annuler

Ensuite, lorsque vous mettez en forme vos tableaux de données dans Excel vous devez utiliser le même séparateur décimal que celui défini par Excel, c'est à dire la virgule.

Si le séparateur considéré par Excel est la virgule et que vous, vous utilisez un point, alors au moment de l'importation dans le logiciel R vos données ne seront pas considérées comme "**numériques**" mais comme du **texte**. Vous ne pourrez donc pas les utiliser comme des nombres.

Si dans vos feuilles Excel, le séparateur est un point (américain, aux Etats Unis le séparateur décimal défini par défaut par Excel est un point), alors utilisez l'outil "Rechercher-Remplacer" pour remplacer les points par des virgules.

Lors de la mise en forme de vos données sous Excel,:
Ne pas utiliser d'accents ou de caractères spéciaux pour nommer une variable (ça vaut aussi pour le nom du fichier de données).

Par contre vous pouvez utiliser les tirets bas ou les points.
Ne pas nommer une variable en commença par un chiffre : pas "**1ventes" mais "ventes1**"

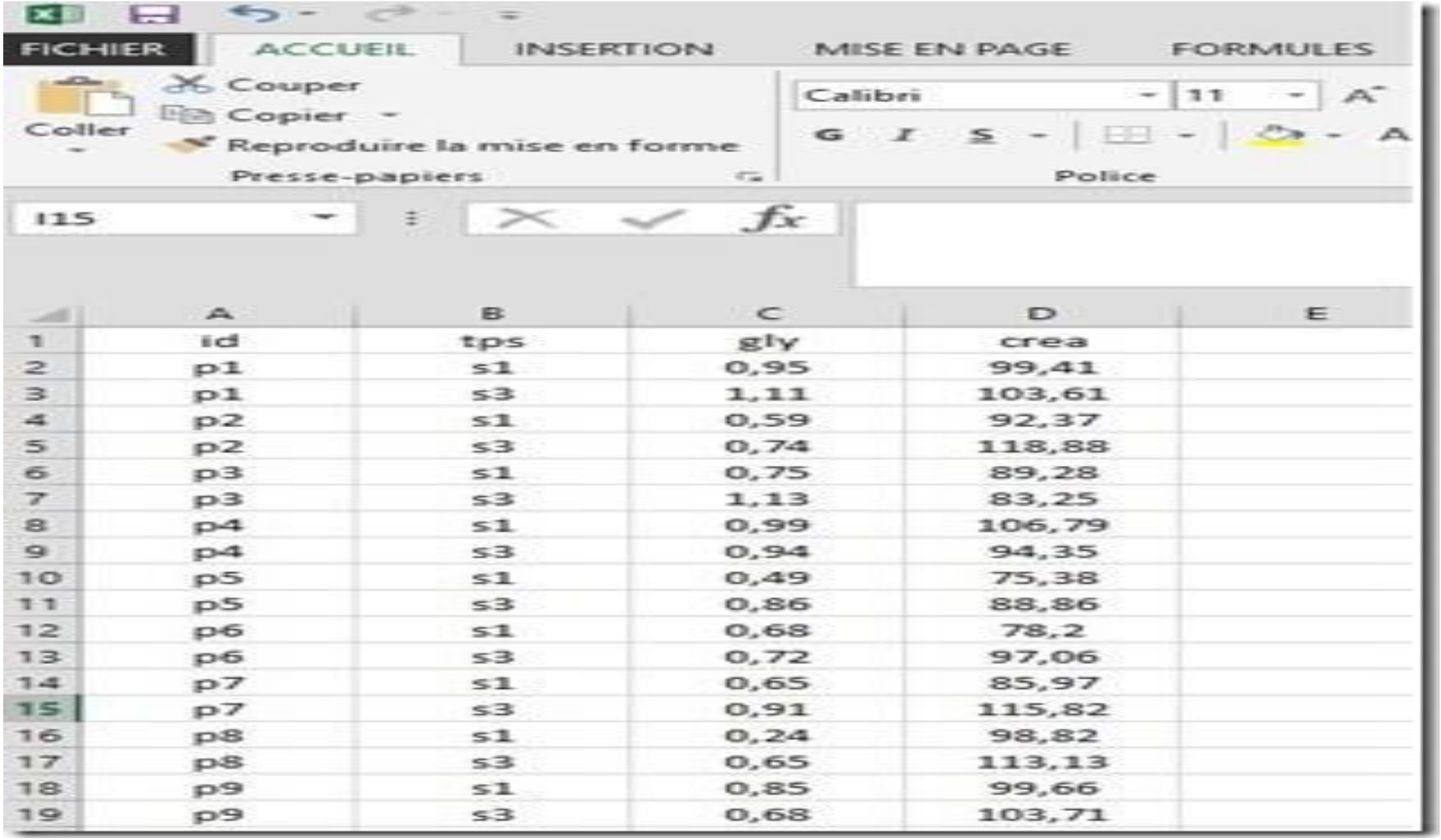
Raccourcir le nom des variables, tout en conservant leur intelligibilité. Par exemple “glycémie” pourrait devenir “glyc” plutôt que simplement “g” et “créatinine” devenir “crea” plutôt que “cre”.

Vous allez écrire le nom des variables à de nombreuses reprises dans les lignes de commandes pour réaliser vos analyses statistiques et cela vous simplifiera la tâche si les noms des variables sont courts.

Ne pas conserver les unités dans les noms des variables. Pour garder néanmoins l'information, le mieux est de faire un "**code book".** Il s'agit d'un tableau avec le nom de la variable dans les données d'origine, son unité, le nom dans le fichier importé, et les valeurs qu'elle peut prendre (par exemple le min et max pour les variables numériques et les différentes modalités possibles pour des variables catégorielles).

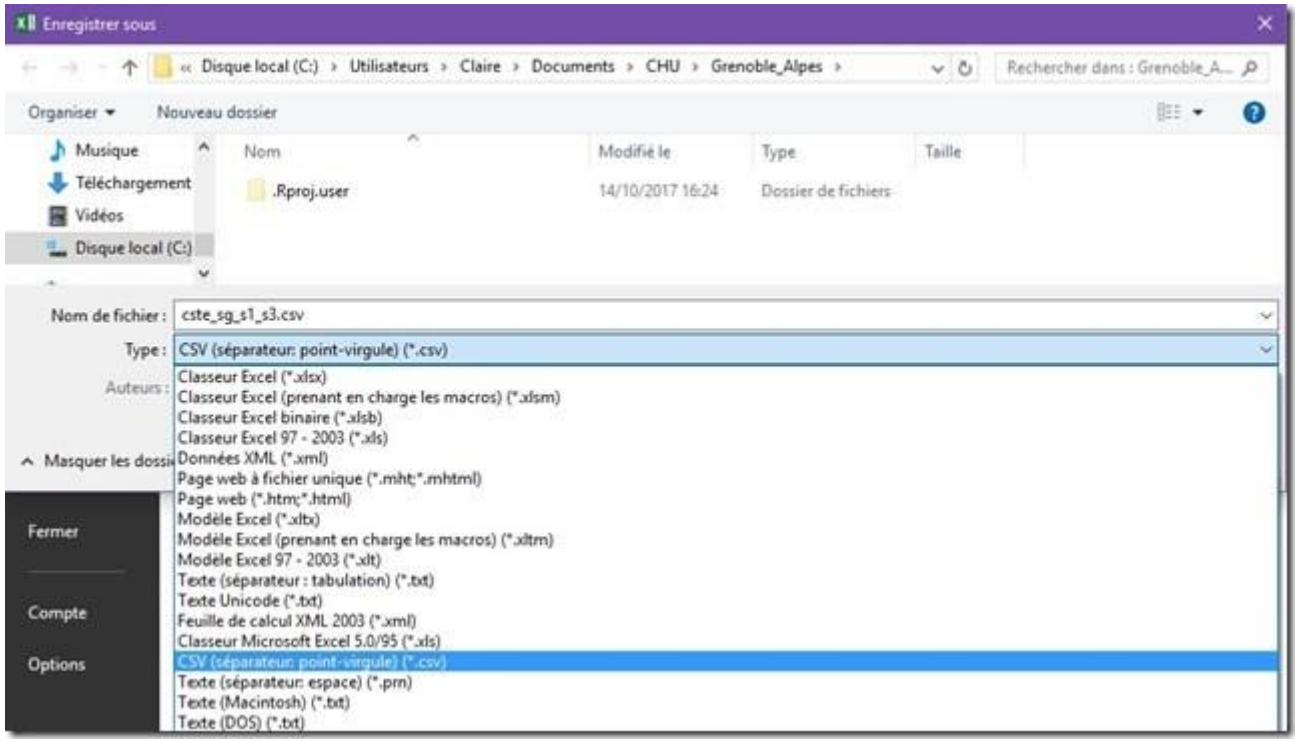
Ne pas arrondir vos données, car l'information originale sera perdue lors de l'importation. Par exemple, si vous ne gardez que 2 chiffres après la virgule vous ne pourrez plus avoir accès à plus de précision après l'importation dans le logiciel R. Il est donc préférable de gérer l'arrondi dans R.

Ne pas recoder en variable numérique une variable catégorielle. Par exemple, vous pourriez être tenté de coder la semaine 1 par un “1” et la semaine “3” par un 3. Cela engendrerait des manipulations supplémentaires à réaliser sous R.



	A	B	C	D	E
1	id	tps	gly	crea	
2	p1	s1	0,95	99,41	
3	p1	s3	1,11	103,61	
4	p2	s1	0,59	92,37	
5	p2	s3	0,74	118,88	
6	p3	s1	0,75	89,28	
7	p3	s3	1,13	83,25	
8	p4	s1	0,99	106,79	
9	p4	s3	0,94	94,35	
10	p5	s1	0,49	75,38	
11	p5	s3	0,86	88,86	
12	p6	s1	0,68	78,2	
13	p6	s3	0,72	97,06	
14	p7	s1	0,65	85,97	
15	p7	s3	0,91	115,82	
16	p8	s1	0,24	98,82	
17	p8	s3	0,65	113,13	
18	p9	s1	0,85	99,66	
19	p9	s3	0,68	103,71	

Une fois le fichier de données mis en forme, il est nécessaire de le sauvegarder au format CSV (séparateur: point-virgule (*.csv). Sauvegarder le dans le dossier associé à votre projet R



Si vous ne savez pas quels sont les caractères utilisés dans votre fichier à importer, ouvrez le fichier dans le bloc note : c**lique droit sur le fichier –> Ouvrir avec –> Bloc note**.

Importer votre tableau de données en utilisant la fonction read.csv2

```
AP1 <- read.csv2("data/AirPassengers1.csv")
```

construire automatiquement le chemin d'accès, on peut utiliser la fonction **here** du package **here**

```
library(here)
AP1 <- read.csv2(here::here("data","AirPassengers1.csv"))
```

Code Preview:

```
library(readxl)
X1_baseIMC <- read_excel("G:/cours iset/master bi/cours atelier R
pour M1BI/bases/1_baseIMC.csv")
View(X1_baseIMC)
```

Si votre fichier .csv utilise la virgule comme séparateur de colonnes et le point comme séparateur de décimales, utilisez la fonction read.csv

Voici l'aide sur ces deux fonctions :

```
read.csv2(file, header = TRUE, sep = ";", quote = "\"",
          dec = ",", fill = TRUE, comment.char = "", ...)
```

```
read.csv(file, header = TRUE, sep = ",", quote = "\"",
         dec = ".", fill = TRUE, comment.char = "", ...)
```

Importer un ficher txt

Les données à importer peuvent également être au format.txt. Dans ce cas, les colonnes sont généralement séparées par une tabulation :

```
> donnees<-read.table(file=file.choose(), header=T, sep="\t", dec=".")
```

Paramètres de la fonction read.table

- **file= file.choose :** emplacement et nom du fichier à lire
- **header=T:** Valeur logique indiquant si le fichier contient le nom des variables sur la première ligne.
- **sep**: Les valeurs sur chaque ligne sont séparées par ce type de caractère.
- **dec:** séparateur décimal pour les nombres.

14_plasma2 - Bloc-notes

Fichier Edition Format Affichage ?

age	sexe	fum	mc	vitamin	cal	graisse	fibres	alcool	colesterol	betadiet	retdiet	betaplasma	retplasma
64	2	2	21.4838	1	1298.8	57	6.3	0	170.3	1945	890	200	
76	2	1	23.87631	1	1032.5	50.1	15.8	0	75.8	2653	451		
38	2	2	20.0108	2	2372.3	83.6	19.1	14.1	257.9	6321	660	328	
40	2	2	25.14062	3	2449.5	97.5	26.5	0.5	332.6	1061	864		
72	2	1	20.98504	1	1952.1	82.6	16.2	0	170.8	2863	1209		
40	2	2	27.52136	3	1366.9	56	9.6	1.3	154.6	1729	1439		
65	2	1	22.01154	2	2213.9	52	28.7	0	255.1	5371	802		
58	2	1	28.75702	1	1595.6	63.4	10.9	0	214.1	823	2571		
35	2	1	23.07662	3	1800.5	57.8	20.3	0.6	233.6	2895	944		
55	2	2	34.96995	3	1263.6	39.6	15.5	0	171.9	3307	493		
66	2	2	20.94647	1	1460.8	58	18.2	1	137.4	1714	535		
40	2	1	36.43161	2	1638.2	49.3	14.9	0	130.7	2031	492		
57	1	1	31.73039	3	2072.9	106.7	9.6	0.9	420	1982	1105		
66	2	1	21.78854	1	987.5	35.6	10.3	0	254.9	2120	1047		
66	1	1	27.31916	3	1574.3	75	7.1	0	361.5	1388	980		
64	1	2	31.44674	3	2868.5	128.8	15	20	379.5	3888	1545		
62	1	2	25.90246	1	1751.1	80.7	8.4	14.1	160.3	2194	242		
75	1	2	29.15264	1	1407.6	35	20.8	7	144.1	3470	479		
68	2	1	38.18727	3	1628.5	78.6	11.6	0	512.3	2108	921		
57	1	2	25.89669	3	1101.4	48.5	8.5	5	197.2	1157	445		
56	1	2	24.45884	3	2433.6	127.6	19.9	7.1	271.2	1739	926		
30	2	2	22.72121	3	1437.3	61.5	8.8	2.3	160.9	1008	695		

Paramètre	Description
file	Le nom du fichier, doit être une chaîne de caractères. Il peut être précédé du chemin relatif ou absolu. Attention (utile pour les utilisateurs de Windows) le caractère “\” est proscrit, et doit être remplacé par “/” ou bien “\\”. À noter qu’il est possible de saisir une adresse web (URL) en guise de chaîne de caractère.
header	Valeur logique (`header=FALSE` par défaut) indiquant si la première ligne contient les noms de variables.
sep	Le séparateur de champ dans le fichier (chaîne vide par défaut, ce qui est au final traduit par une espace comme séparation). Par exmple, `sep=";"` si les champs sont séparés par un point-virgule, ou encore `sep="\t"` s’ils sont séparés par une tabulation.
dec	Le caractère employé pour les décimales (par défaut, `dec="."`).

Fonction	Séparateur de champs	Séparateur décimal
`read.csv()`	`","`	`"."`
`read.csv2()`	`";"`	`","`
`read.delim()`	`"\t"`	`"."`
`read.delim2()`	`"\t"`	`","`

La fonction scan

La fonction scan() est beaucoup plus souple que read.table().
Son emploi est requis dès que les données ne sont pas organisées comme un tableau.
La nature des variables peut être spécifiée en renseignant le paramètre what.
On retrouve la plupart des paramètres de la fonction read.table().
Le tableau ci-après présente les principaux

Paramètre	Description
file	Le nom du fichier, doit être une chaîne de caractères. Il peut être précédé du chemin relatif ou absolu. Attention (utile pour les utilisateurs de Windows) le caractère "\" est proscrit, et doit être remplacé par "/" ou bien "\\". À noter qu'il est possible de saisir une adresse web (URL) en guise de chaîne de caractère.
what	Permet de précider le type des données lues.
nmax	Si présent, indique le nombre de données à lire, ou, si `what` est une liste, le nombre maximum de lignes à lire.
n	Le nombre de données à lire (pas de limite par défaut).
sep	Le séparateur de champ dans le fichier (chaîne vide par défaut, ce qui est au final traduit par une espace comme séparation). Par exmple, `sep=";"` si les champs sont séparés par un point-virgule, ou encore `sep="\t"` s'ils sont séparés par une tabulation.
dec	Le caractère employé pour les décimales (par défaut, `dec="."`).

Exportation

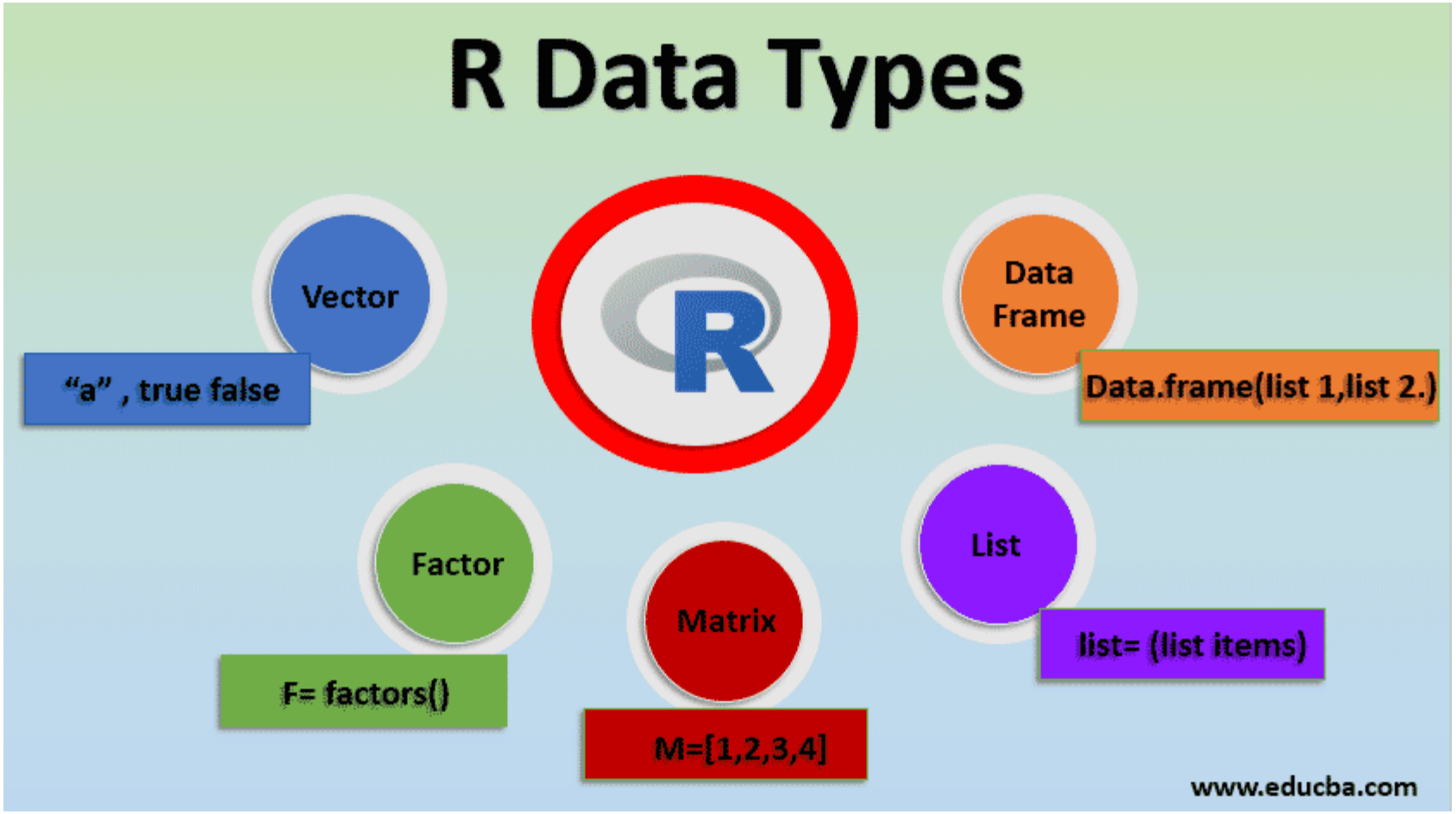
Pour enregistrer des données depuis un data.frame, un vecteur ou une matrice, la fonction write.table() peut être utilisée.

Par exemple, si le data.frame se nomme donnees, l'instruction ressemble à :

```
write.table(donnees, file = "nom_fichier.txt", sep = ";")
```

Génération (production) des données sous R

On verra lors du chapitre suivant la génération des données sous R (vecteurs, matrices, data frame, listes,,,,)



Vérifier l'importation et la structure de vos données

La vérification des données se fait soit par head ou tail:

```
head(people,7)

##    Name_FirstName Sex Age Height  Hair Blue_Eye Brown_Eye Other_Eye
## 1     Smith_Emily   F  NA   1.68 Blond        1         0         0
## 2  Jonhson_Mikael   M 253   1.93 Brown        0         1         0
## 3 Williams_Ashley   F  38   1.55 Brown        0         1         0
## 4       Jones_Alex   M  15     NA Other        0         0         1
## 5     Brown_Katie   F  79   1.63 Blond        1         0         0
## 6      Davis_John   M  56   1.70 Other        1         0         0
## 7    Miller_Megan   F  63   1.50  <NA>        0         1         0

tail(people,3)

##    Name_FirstName Sex Age Height  Hair Blue_Eye Brown_Eye Other_Eye
## 8  Wilson_Matthew   M  19   1.87 Brown        0         0         1
## 9     Moore_Grace   F  24   1.52 Blond        0         1         0
## 10    Taylor_Jack   M  -1   2.15 Brown        0         0         1
```

Pour **contrôler la structure des données utiliser la fonction str**

```
1 str(people)
```

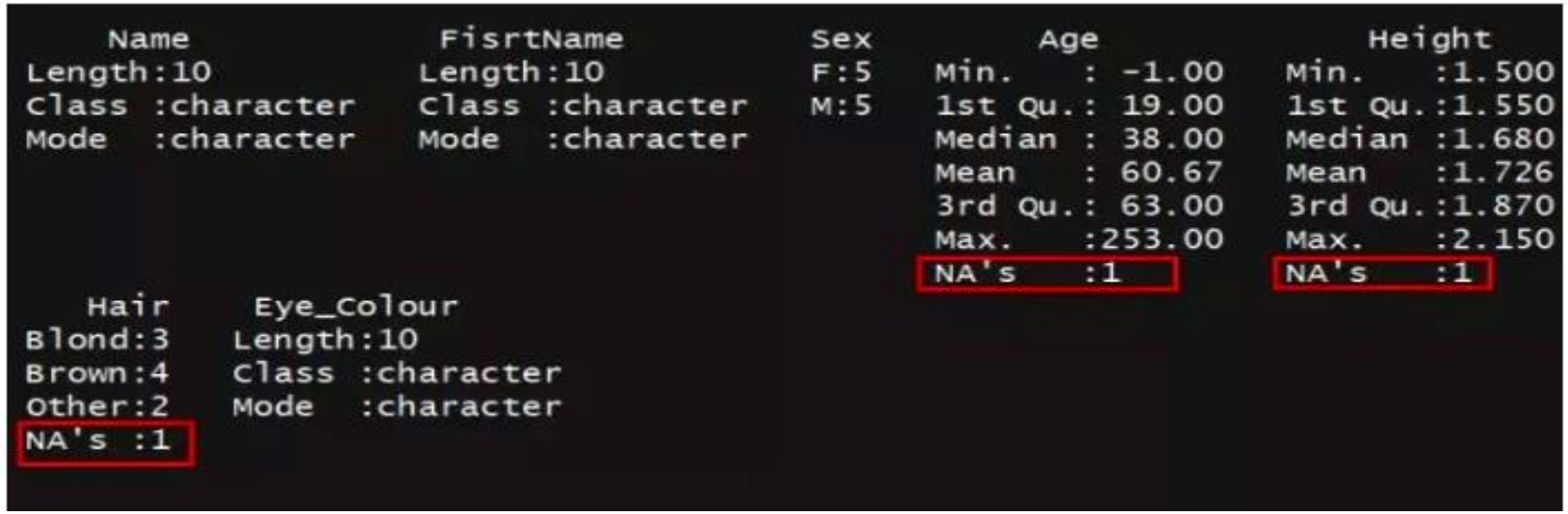
```
'data.frame':   10 obs. of  8 variables:
 $ Name_FirstName: Factor w/ 10 levels "Brown_Katie",..: 7 4 9 3 1 2 5 10 6 8
 $ Sex           : Factor w/ 2 levels "F","M": 1 2 1 2 1 2 1 2 1 2
 $ Age           : int  NA 253 38 15 79 56 63 19 24 -1
 $ Height        : num  1.68 1.93 1.55 NA 1.63 1.7 1.5 1.87 1.52 2.15
 $ Hair          : Factor w/ 3 levels "Blond","Brown",..: 1 2 2 3 1 3 NA 2 1 2
 $ Blue_Eye      : int  1 0 0 0 1 1 0 0 0 0
 $ Brown_Eye     : int  0 1 1 0 0 0 1 0 1 0
 $ Other_Eye     : int  0 0 0 1 0 0 0 1 0 1
```

Cette fonction est très pratique, car elle permet de connaître :

- le **format des données** : ici un data frame
- les **dimensions** du jeu de données : ici 10 lignes (observations) et 8 colonnes (variables)
- le **nom des variables** : ici Name_FirstName, Sex, Age etc..
- le **format de ces variables** : ici Name_FirstName, Sex et Hair sont des variables catégorielles (factor), l'Age est une variable numérique de type entier, Height est une variable numérique continue

Les valeurs manquantes (NA)

Une méthode simple pour explorer les données manquantes est d'utiliser la fonction summary



Une autre fonction très utile est la fonction `df_status()` du package `funModelling` qui renvoit, pour chaque variable, le nombre de valeurs égales à zéro, le nombre de valeurs manquantes, et le nombre de valeurs infinies (par exemple 1/0), ainsi que les pourcentages

Supprimer les lignes comportant des NA

Pour **ne conserver que les lignes du jeu de données ne comportant aucune donnée on peut utiliser la fonction Na,omit()**

```
people_sans_NA <- na.omit(people)
summary(people_sans_NA)

##      Name              Prenom           Sex         Age        
##  Length:7           Length:7           F:3   Min.   : -1.00  
##  Class :character   Class :character   M:4   1st Qu.: 21.50  
##  Mode  :character   Mode  :character         Median : 38.00  
##                                              Mean   : 66.86  
##                                              3rd Qu.: 67.50  
##                                              Max.   :253.00  
##      Height          Hair     Eye_Colour       
##  Min.   :1.520   Blond:2   Length:7           
##  1st Qu.:1.590   Brown:4   Class :character   
##  Median :1.700   Other:1   Mode  :character
```

Dans certaines situations, par exemple **lorsque les données manquantes sont totalement aléatoires**, on peut avoir envie de r**emplacer les NA par une moyenne, ou une médiane**.
Pour cela, on peut utiliser la fonction

`replace_na` du package `tidyr`

Par exemple, pour remplacer la valeur manquante de la variable Height, par la moyenne des valeurs :

```r
people_rep <- people %>%
     mutate(Height=replace_na(Height, mean(Height, na.rm=TRUE)))

summary(people_rep$Height)

    ##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    ##   1.500   1.570   1.690   1.726   1.834   2.150
```

Identifier les valeurs aberrantes

Les **valeurs numériques aberrantes** peuvent être identifiées grâce à la fonction summary

en **prêtant attention aux valeurs min et max** :

Visualisations

Pour **mettre en évidence des données aberrantes**, il peut également être intéressant de r**éaliser des visualisations**.

On peut par exemple utiliser le code suivant pour ne sélectionner que les variables numériques et réaliser un **dotplot** pour chacune d'entre elles :