TP introduction R :

La base de la base!

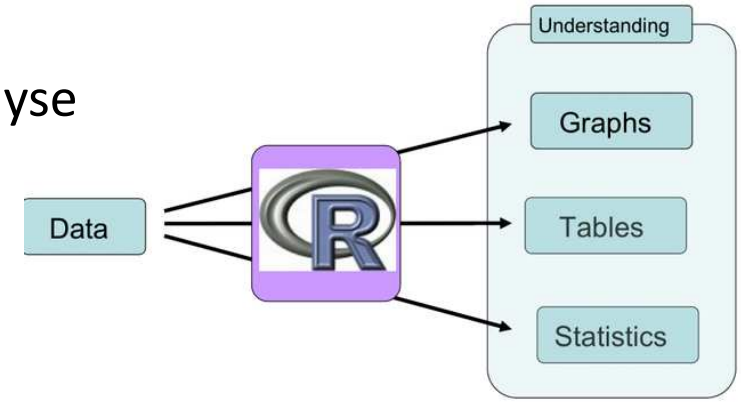
Plan

1. R et Rstudio, c'est quoi?
2. Projet R et répertoire de travail
3. R comme super calculateur
4. Le concept d'objet sous R : Affectation
5. Les types de variables
6. Quelques opérateurs relationnels et logiques utiles
7. Différentes structures de données sous R
    - Les vecteurs
    - Les tableaux et matrices
    - Les data frames
    - Les listes
8. Indexer des objets
9. Les fonctions
10. Rechercher de l'aide
11. Installer des packages

Trucs et astuces R

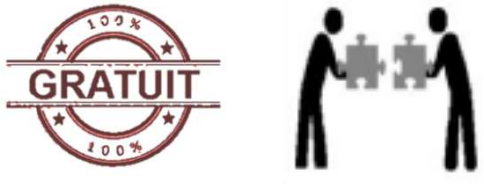
1. R c’est quoi?

Un **langage** de programmation orienté objet conçu pour l'analyse statistique, l'exploration et la visualisation de données.



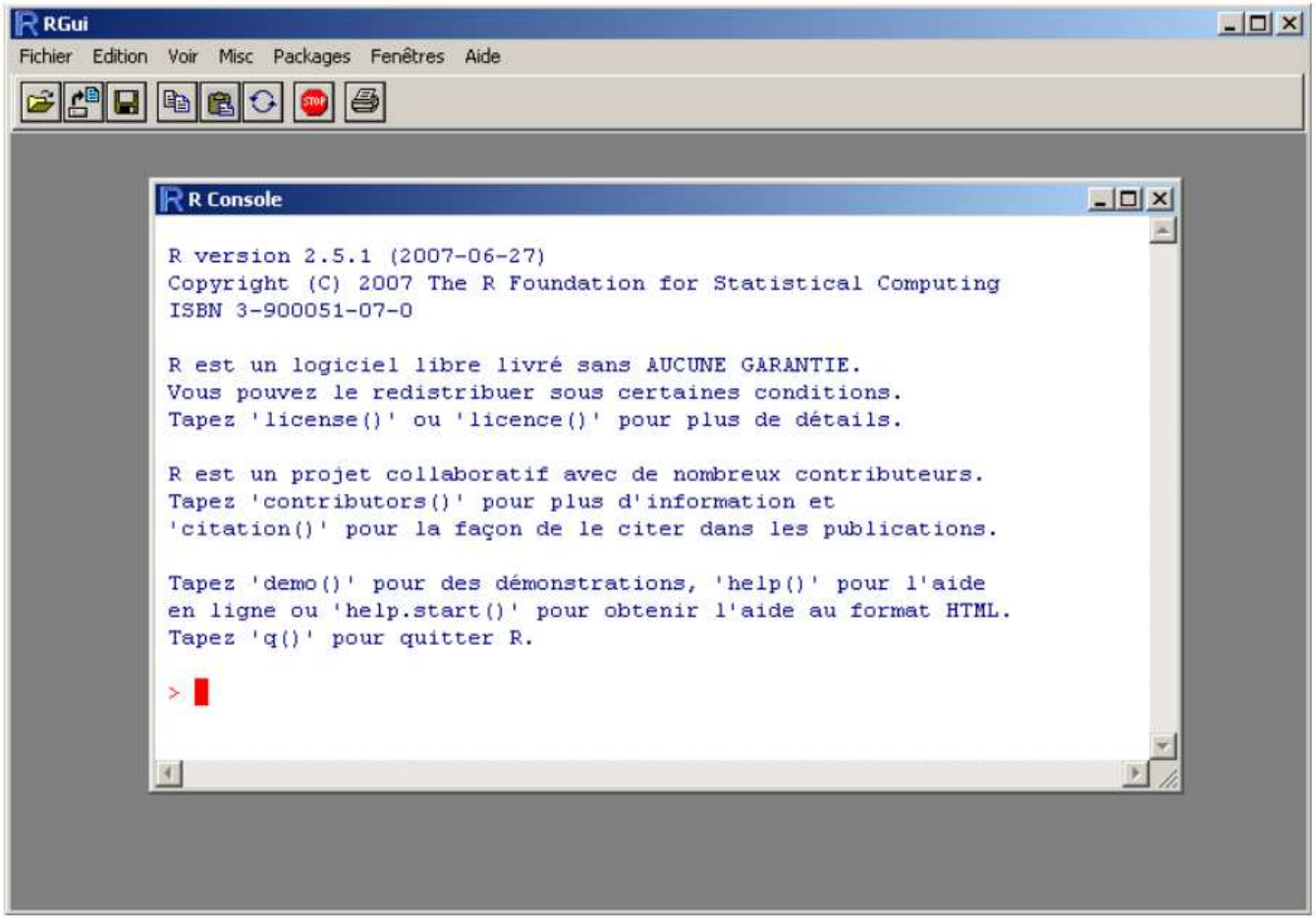
Un **logiciel** libre et gratuit (open source), compatible avec la majorité des système d'exploitation → partage de codes R facile.

Constamment mis à jour et amélioré **par** la communauté d’utilisateur, **pour** la **communauté**!



Les fonctionnalités sont organisé sous la forme de modules ou ”packages”

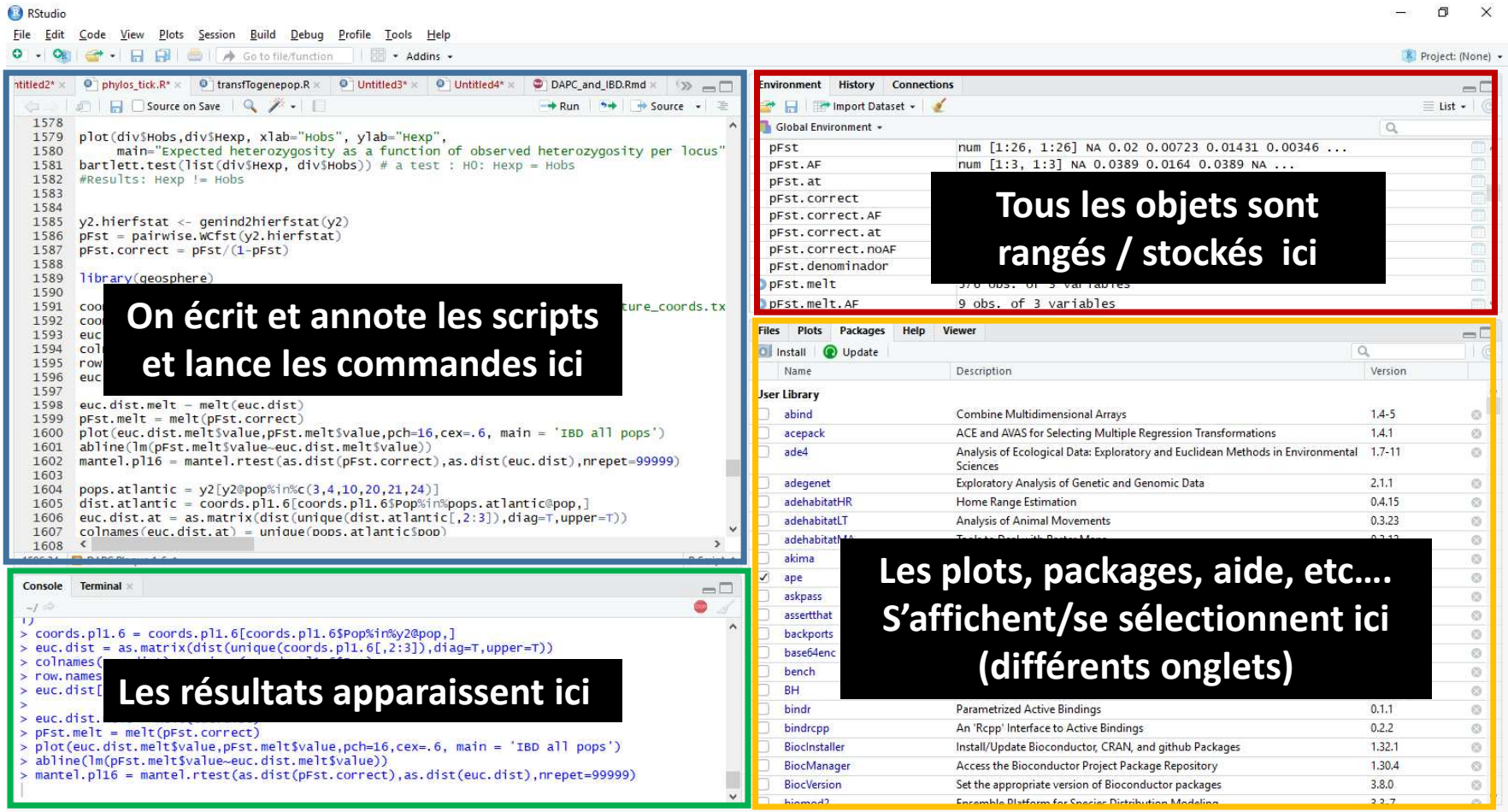
Voici R…


RGui
Fichier Edition Voir Misc Packages Fenêtres Aide
R Console
R version 2.5.1 (2007-06-27)
Copyright (C) 2007 The R Foundation for Statistical Computing
ISBN 3-900051-07-0
R est un logiciel libre livré sans AUCUNE GARANTIE.
Vous pouvez le redistribuer sous certaines conditions.
Tapez 'license()' ou 'licence()' pour plus de détails.
R est un projet collaboratif avec de nombreux contributeurs.
Tapez 'contributors()' pour plus d'information et
'citation()' pour la façon de le citer dans les publications.
Tapez 'demo()' pour des démonstrations, 'help()' pour l'aide
en ligne ou 'help.start()' pour obtenir l'aide au format HTML.
Tapez 'q()' pour quitter R.
>

RStudio c'est quoi?

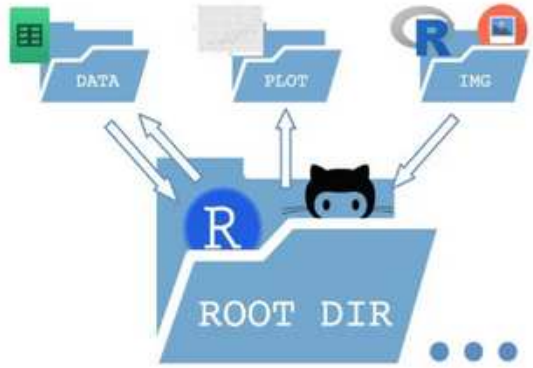
http://www.rstudio.com/

Interface plus ergonomique que R (personnalisable, visuel, intuitif).



2. Projet R et répertoire de travail

... Etre organisé pour s'y retrouver...



Les **projets RStudio** permettent d'accéder facilement à tous les fichiers (scripts, données, graphiques, rapports automatisés…) utilisé pour une analyse.

Ces fichiers sont regroupés, reliés et enregistrés dans un **répertoire de travail** dédié.

Lorsque vous ré-ouvrirez votre projet **Rstudio,** vous retrouverez votre projet à l'état dans lequel vous l'avez laissé la dernière fois que vous avez travaillé dessus.

L'utilisation de projets facilite la reproductibilité et le partage de données, de scripts, et de leur documentation.

2. Projet R et répertoire de travail

Création d'un projet pour le TP :

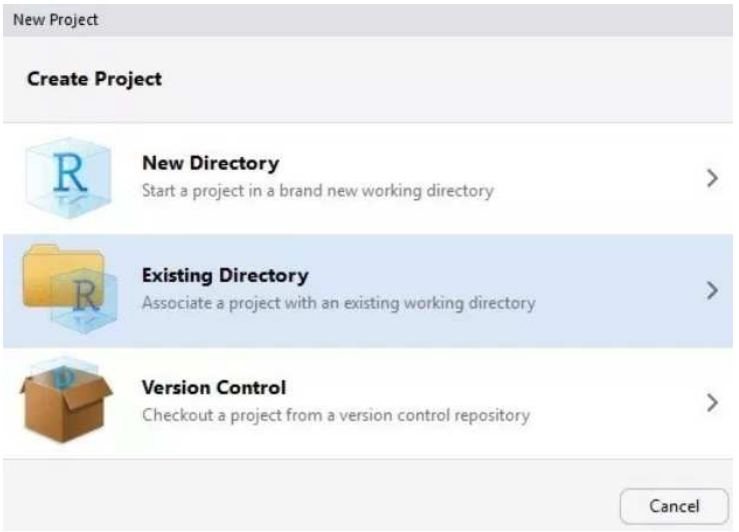
1. Créer un dossier nommé "TP_R" sur votre bureau :

Clic droit sur Bureau →Nouveau / Dossier / "TP_R"

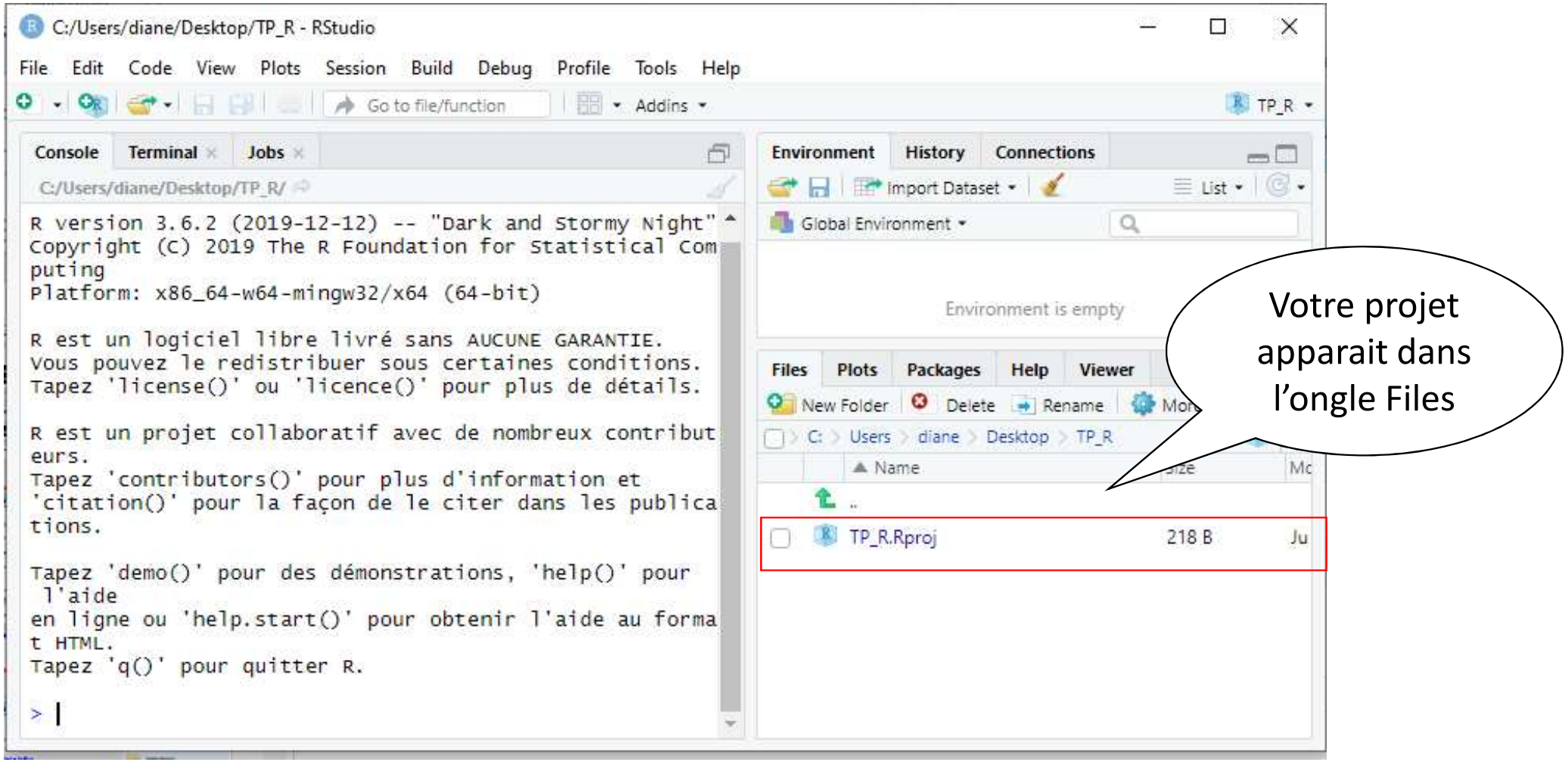
2. Dans R Studio : File / New Project / Existing directory

Sélectionner le dossier "TP_R"

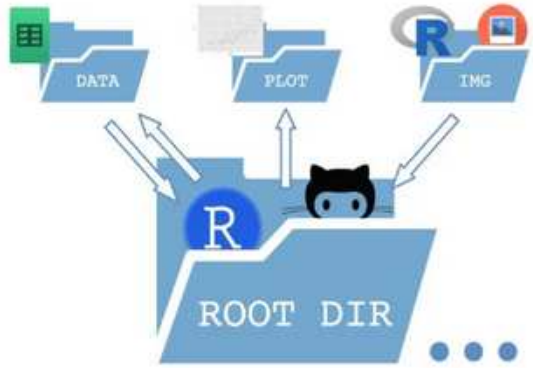
puis cliquer sur "**Create project**"



2. Projet R et répertoire de travail



2. Projet R et répertoire de travail



→ Ce dossier devient votre **répertoire de travail par défaut**
(à vous de l'organiser).

C'est là que R :

- **Ira chercher** vos données (créez par exemple un sous-dossier « data »)
- **Sauvegardera** :

- Vos scripts
- Vos graphiques (créez un sous-dossier « plots »)
- Les fichiers de sortie de R (créez un sous-dossier « outputs »)
- Un fichier **"history"** (historique des commandes) et un **"environment"** (données importées, objets et modèles créés etc..) **spécifiques au projet**

2. Projet R et répertoire de travail

Création d'un Script :

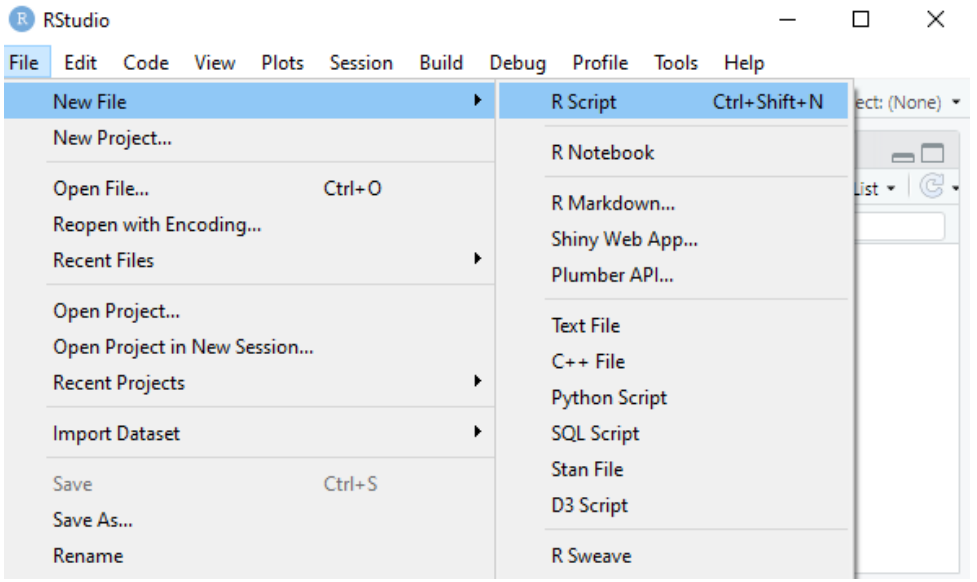
Fichier texte regroupant l'ensemble des commandes nécessaires pour réaliser un projet.

Permet de relancer une analyse avec un minimum d'effort, facilement échangeable.

3. Créer/ouvrir un nouveau script :

File / New File / R Script

Un script nommé par défaut "Untitled1"
vient de s'ouvrir dans la fenêtre édition



4. Enregistrer votre script
(par défaut dans le répertoire de travail) :

File / Save As... → nommez le " Script_TPR"

2. Projet R et répertoire de travail

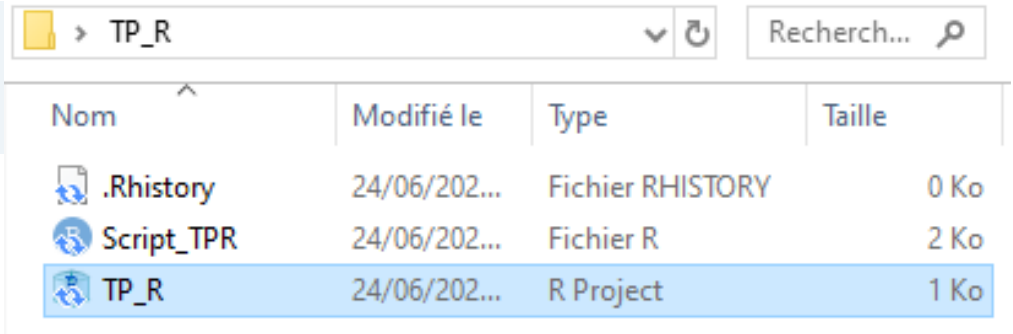
5. Dans la fenêtre édition, demander à R l'adresse de votre repertoire de travail actuel :

```
getwd()                                          ← Commande (instruction à R)
[1] "C:/Users/diane/Desktop/TP_R"                ← Sortie R (réponse, dans console)
```

Lancer la commande en tapant : « ctrl » + « entrée »

6. Obtenir le contenu du répertoire de travail :

```
dir()
[1] "Script_TPR.R" "TP_R.Rproj"
```



2. Projet R et répertoire de travail

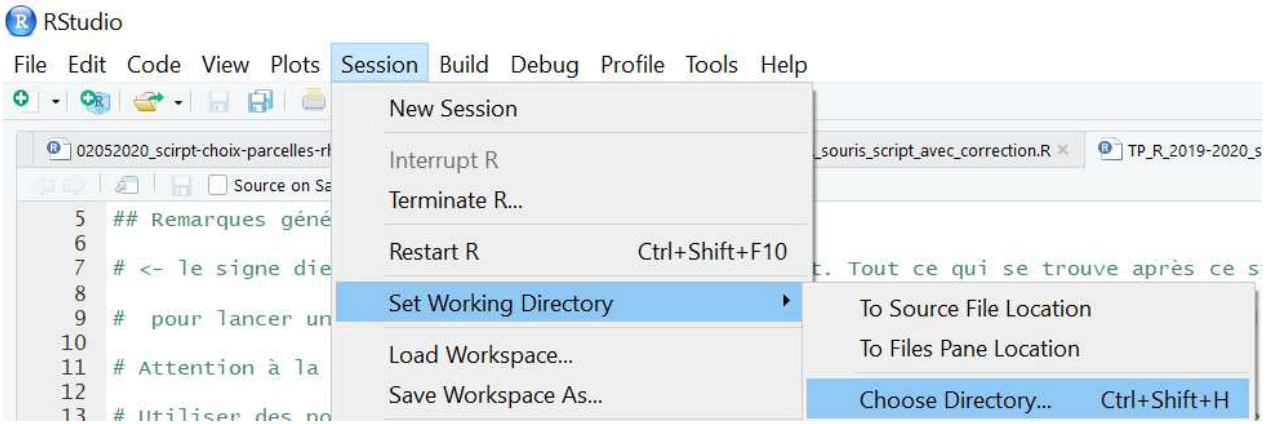
NB: Si vous devez changer de repertoire de travail, utilisez la fonction setwd().

Exemple: Si vous devez charger des données qui ne se trouvent pas à la racine du répertoire de travail mais dans un sous dossier ”data”.

Copier-coller l’adresse du dossier dans la fenêtre édition et remplacer \ par / :

```
# Definir le repertoire de travail
setwd(" C:/Users/diane/Desktop/TP_R/data")
```

Ou (+ simple) →



Trucs et astuces R

Exécution de commande:

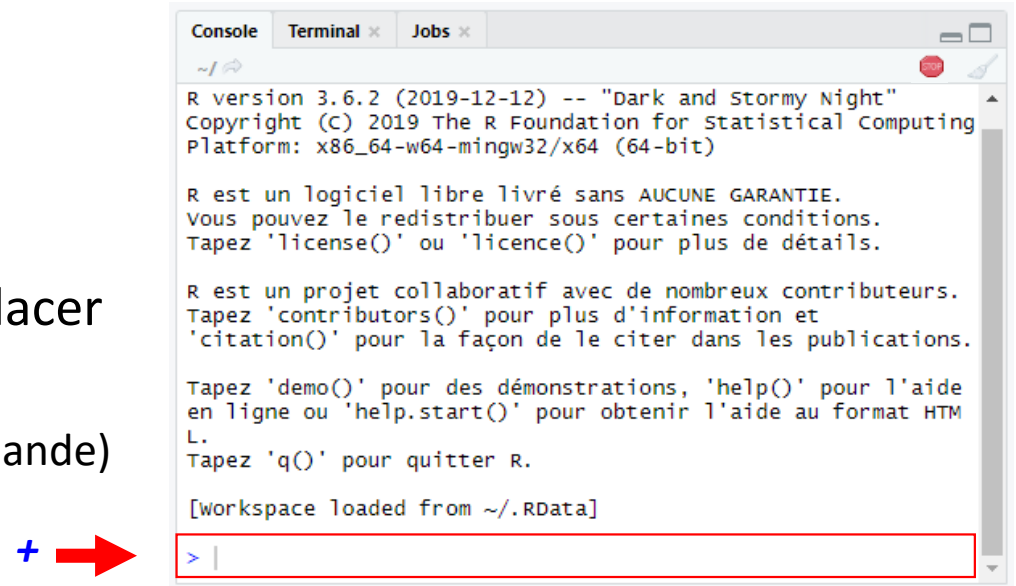
Sélectionner 1 ou plusieurs lignes du script
et tapez « ctrl » + « entrée »

« **invite de commande** » : signifie que R est disponible/attend de votre prochaine commande.

Si chevron (>) absent :
- commande incomplète (+ dans console)
-R occupé (dans barre de la console)
→ tapez « Echap » pour annuler/avorter 1 processus

Flèches « haut » et « bas » pour se déplacer dans l'historique de commande

(relancer/modifier/corriger rapidement 1commande)



Trucs et astuces R

Commentez/annotez vos scripts : Indispensable pour s'y retrouver après plusieurs semaines, comprendre, organiser et partager des scripts.

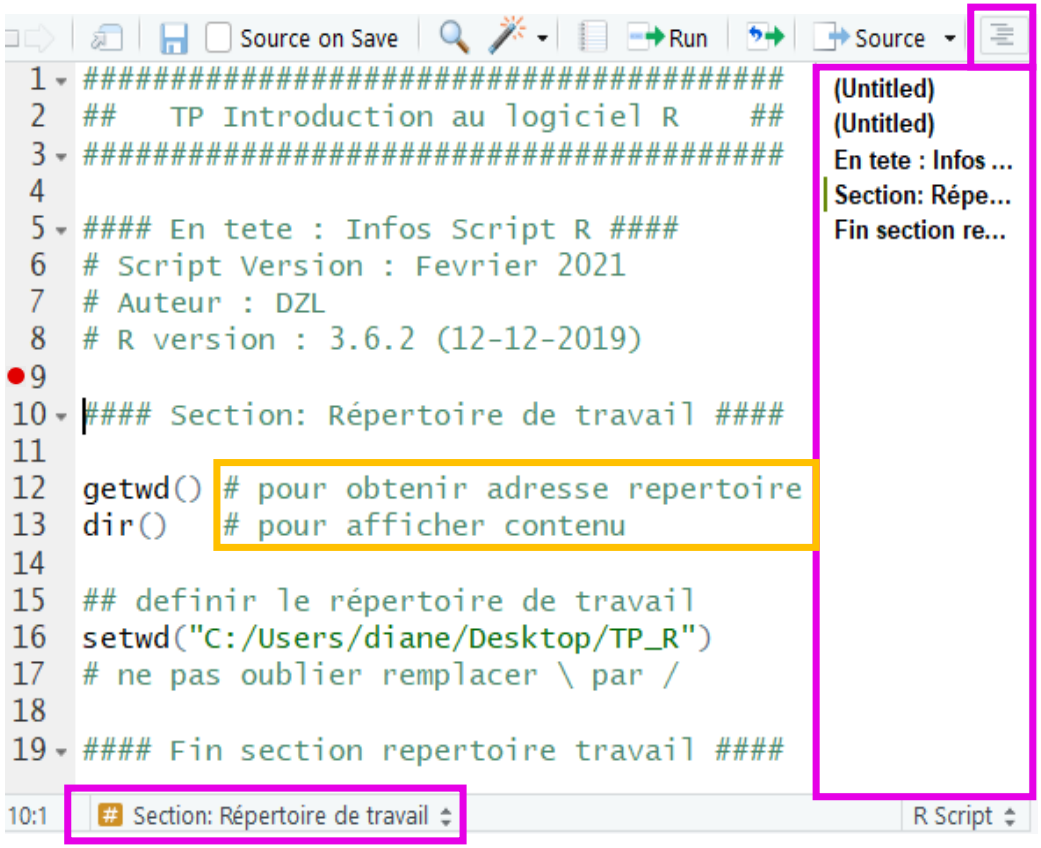
Symbole **#** (« Alt Gr » + « # ») → tout ce qui suit ne sera pas exécuté par R.

En tête : noter les informations essentielles (nom du projet, auteur, date, version R, packages utilisés…)

A partir de 4 « # » ou « - » en fin de commentaire, R crée une **section** dans le script → **Hiérarchisation** des (sous)titres.

Améliore **lisibilité** et **navigation** (masquer/déplier, début/fin section)

Clic à gauche n° ligne → insérer/enlever un **repère** ●



3. R comme super calculateur

Exemple simple avec les opérateurs de base :

```
8876.5 + 4 * ( ( 112 / 39 ) - 27^2 )
[1] 5971.987
```

NB:

- Les décimales sont des « . » (pas des « , »)
- R respecte toujours les priorités d'opérations
- Symbole ⊗ = erreur sur la ligne.

  Ex:
  ```
  ⊗  2  getwd()
  ```
- Si la console affiche **+**, R attend que vous complétiez la commande.

  Ex:
  ```
  4*
  +
  3
  [1] 12
  ```

Principaux opérateurs arithmétiques:

Opérateur	Fonction
+	Addition
-	Soustraction
*	Multiplication
/	Division
^	Exposant

Autres fonctions utiles :

sin(2*pi/3)

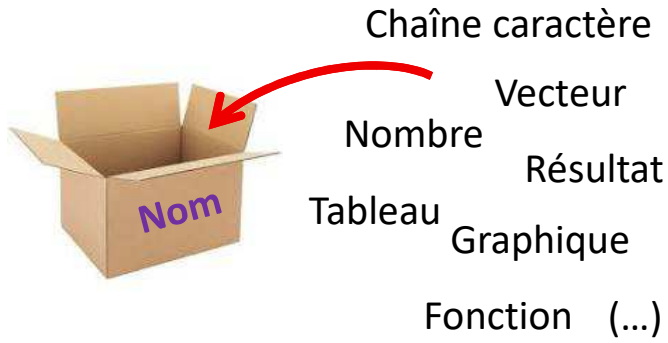
sqrt(3)/2

log(1)

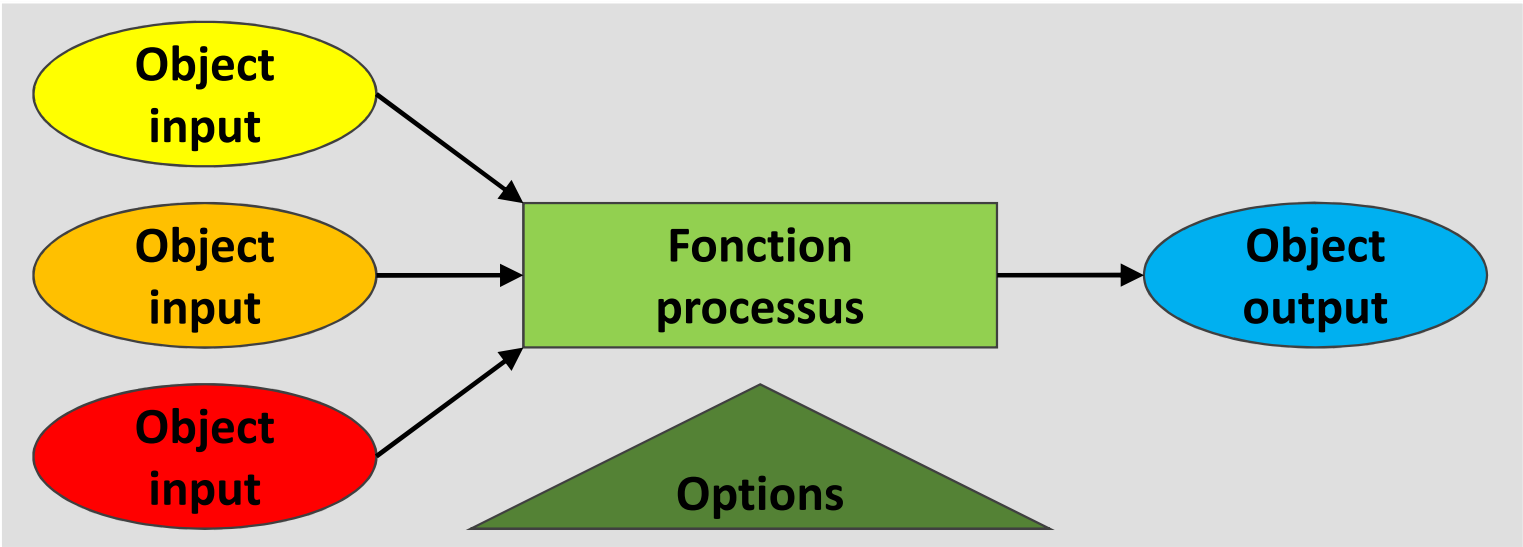
log(exp(1)) (…etc…)

4. Concept d'objet sous R : Affectation

Quand R est utilisé, les variables, les données, les fonctions, les résultats (etc), sont **stockés** en mémoire (**Environement**) sous forme d'**objets** qui ont chacun un **nom**.



On va ensuite **agir** sur ces objets avec des **opérateurs** (arithmétiques, logiques,. . .) et des **fonctions** (aussi des objets).



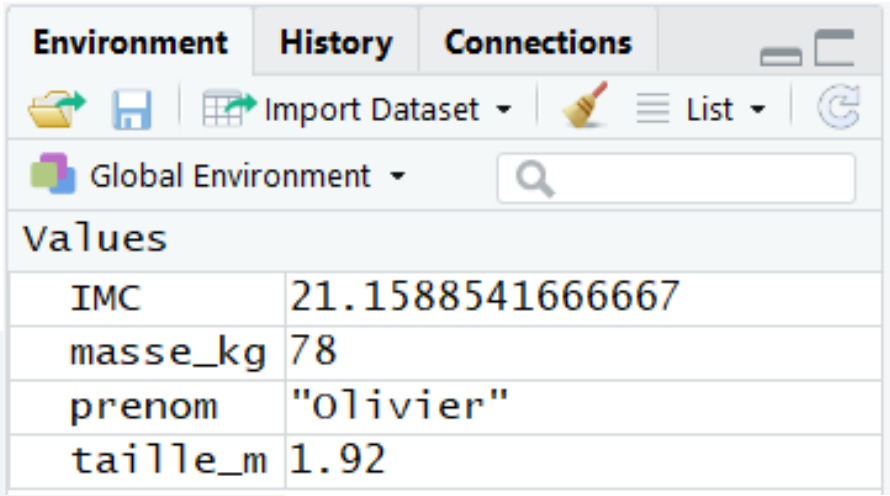
4. Concept d'objet sous R : Affectation

Créer des objets:

Format générique : nom <- contenu

Chaîne caractère
Vecteur
Nombre
Résultat
Tableau
Graphique
Fonction (…)

```
prenom <- "Olivier"   # chaine de caractères
taille_m <- 1.92      # valeur numérique décimale
masse_kg <- 78        # valeur numérique entière
```



Agir sur les objets:

```
IMC<-masse_kg/(taille_m)^2
IMC    # appeler et afficher contenu de l'objet
[1] 21.15885
round(IMC,2) # arrondir a 2 décimales
[1] 21.16
```

Trucs et astuces R

- **Choisir des noms d'objets explicites et courts**
- Ils peuvent contenir des lettres, des chiffres, les symboles . et _ (à la place des espaces)

ex: ***taille_conj1*** est préférable à ***taille_du_conjoint_numero_1*** (trop long) ou à ***t1*** (pas explicite)

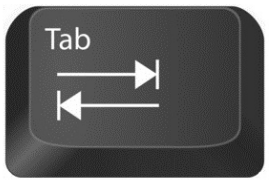
- **Attention** :

- Ils ne peuvent pas commencer par un chiffre ou un symbole

- R fait la différence entre minuscules et majuscules dans les noms d'objets (ex: x ≠ X)

- Éviter les caractères accentués (pour des questions d'encodage)

- Éviter les noms d'objets/de fonction qui existent dans R (ex: F, T, data.frame…)



NB: La touche tabulation permet de compléter automatiquement les noms d'objets/fonctions/packages … (accélère la saisie et évite les erreurs de frappe)

4. Concept d'objet sous R : Affectation

Olivier a pris 5kg pendant le confinement:

```
masse_kg <- masse_kg + 5   #  écrase la précédente valeur autant de fois qu'on lance la commande
masse_kg                                  # renvoie [1] 83  (au lieu de 78)
IMC<-masse_kg/(taille_m)^2          #  mise a jour de l'IMC
round(IMC,2)                              # renvoie [1] 22.52  (au lieu de 21.16)
```

Les objets ont tous 2 attributs intrinsèques; la classe (type de donnée) et la longueur :

```
class(prenom)            # la fonction class() renvoie [1] "character"
class(taille_m)          # la fonction class() renvoie [1] " numeric"
class(masse_kg)          # la fonction class() renvoie [1] " numeric"

taille_m+masse_kg        # renvoie [1] 84.92 (opération possible)
 prenom+taille_m         # on ne peut pas faire d'opération sur des objets de classes différentes

length(IMC)              # la fonction length() renvoie [1] 1 → 1 seule valeur (1 élément)
```

5. Les types de variables

Pour connaître le type d'une variable donnée, on peut utiliser la fonction class():
dicte les principes régissant la manipulation de cet objet.

Résultat de class	Type de variable
factor	Facteur
integer	Numérique
double	Numérique
numeric	Numérique
character	Caractères
logical	Booléenne

4 principaux types de variables dans R :

- **Numériques** : quantitatives (integer, double)
- **Caractères** : chaînes de caractères
- **Booléennes (logical)** : Vrai vs Faux (TRUE, FALSE)
- **Facteurs** : format spécial pour les variables catégorielles pouvant prendre un nombre restreint de modalités (levels). Très utilisés pour faire des groupes.

6. Quelques opérateurs relationnels et logiques utiles

R permet de tester des déclarations logiques (condition → Vraie ou Fausse?):

```
A=1      # si lance A, renvoie [1] 1
A==1     # renvoie [1] "TRUE"

A!=1     # renvoie [1] "FALSE"

A<2      # renvoie [1] "TRUE"
A>2      # renvoie [1] "FALSE"

B<-12

A&B==1           # renvoie [1] "FALSE"
A|B==12          #  renvoie [1] "TRUE"

C<-c(A, B)       # vous venez de créer un vecteur!
C                # renvoie [1] 1 12
which(C==12)     # renvoie [1] 2 (position de l'élément dans vecteur)
```

Attention! Le 1er correspond à une affirmation (équivaut à assignation), le 2nd à une question!

Opérateur	Fonction
<	Inférieur à
<=	Inférieur ou égal à
>	Supérieur à
>=	Supérieur ou égal à
==	Strictement égal à
!=	Différent de
x|y	x OU y
x&y	x ET y
!	Négation

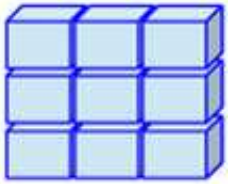
Opérateurs relationnels/de comparaison

Opérateurs logiques

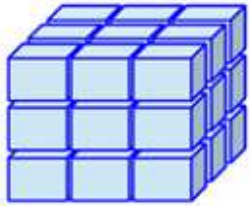
7. Différentes structures de données sous R

Selon la nature des valeurs contenues dans un objet, on distingue 5 catégories de données :

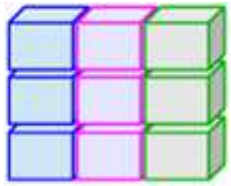
Vector: Objet 1D généré par combinaison d'éléments du même type. Structure de base R.



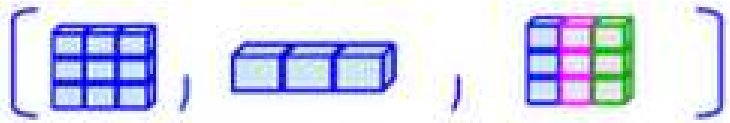
Matrix: Objet 2D (L*C) généré en combinant plusieurs vecteurs du même type. On peut faire des opérations sur les matrices (+, -, *…).



Array: Objet ≥3D généré en combinant plusieurs matrices/vecteurs du même type. Vecteur=array 1D, matrice=array 2D.



Data frame: Objet 2D généré en combinant plusieurs vecteurs (colonnes) de n'importe quel type, avec la contrainte 1 colonne =1 type. Format le plus utilisé.



List : Une liste stocke des collections d'objets de tous types.

7.1. Les vecteurs

Objet de base incontournable dans R → tout est vecteur !

- Une valeur seule = vecteur de longueur 1.
- Matrices, tableaux, data frame sont composés de vecteurs de longueurs identiques.
- Unité de base dans les calculs…

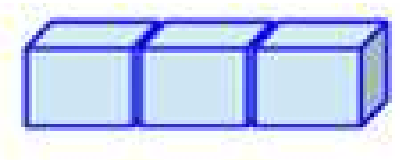
➔ Un vecteur est une entité constituée d'une suite (ordonnée) de valeurs semblables

Format générique : nom.du.vecteur <- c (valeur1, valeur2, valeur3, …)

La fonction c() signifie combiner ou concaténer.

Les valeurs doivent être de même nature (class) : numeric, character ou logical.

La longueur d'1 vecteur = nombre d'éléments qu'il contient.



7.1. Les vecteurs

APPLICATION: On a demandé à 2 échantillons d'individus de nous indiquer leur poids.

Création de vecteurs:

 # Echantillon 1:

```
poids <- c (77, 58, 66, 89)
poids                          # renvoie le vecteur [1] 77  58  66  89

names (poids) <- c ("Marc", "Sophie", "Julie", "Francois")  # names () attribue étiquettes aux valeurs
poids                          # renvoie
```

```
Marc   Sophie    Julie Francois
  77      58       66       89
```

```
class(poids) ; length(poids)  # renvoie [1] "numeric"  [1] 4
```

 # Echantillon 2:

```
pds_suite<-c (44, 61, 101)
names (pds_suite) <- c ("Marie", "Clara", "Theo")
```

7.1. Les vecteurs

On veut combiner ces 2 vecteurs en 1 seul; comment faire?

```
poids_tous <- c (poids, pds_suite)
poids_tous          # renvoie
```

```
Marc Sophie Julie Francois Marie Clara Theo
  77     58    66       89    44    61   101
```

```
class(poids_tous) ; length(poids_tous)    # renvoie [1] "numeric"  [1] 7
```

On nous fourni la taille des individus en m. On veut la taille en cm; comment faire?

```
taille_tous_m<- c(1.76, 1.56, 1.64, 1.97, 1.47, 1.73, 1.88)
names(taille_tous_m) <- c("Marc", "Sophie", "Julie", "Francois","Marie", "Clara", "Theo")

taille_tous_cm<- c(taille_tous_m)*100  # l'opération s'applique a chacun des termes du vecteur
taille_tous_cm
```

```
Marc Sophie Julie Francois Marie Clara Theo
 176    156   164      197   147   173   188
```

7.1. Les vecteurs

Le confinement est passé par là…

En moyenne, les individus ont pris du poids, mais pas tous le même nombre de kg.

Comment prendre cela en compte pour calculer leur poids après confinement?

Marc	*Sophie*	*Julie*	*Francois*	*Marie*	*Clara*	*Theo*
+5	*+1*	*+3*	*-2*	*0*	*+6*	*+2*

```
prise_poids <- c (5, 1, 3, -2, 0, 6, 2)        # on crée un vecteur de prise de poids
poids_ap <- poids_tous + prise_poids            # on additionne 2 vecteurs terme à terme
poids_ap                                        # renvoie 82  59  69  87  44  67  103
class(poids_ap) ; length(poids_ap)              # renvoie [1] "numeric"  ; [1] 7
```

Calculer le poids moyen pris pour cet échantillon avec les fonctions sum() et length().

```
prise_poids_moy<- sum(prise_poids)/length(prise_poids)
prise_poids_moy                 # renvoie [1] 2.142857
mean(prise_poids)               # la fonction mean() renvoie [1] 2.142857
```

7.1. Les vecteurs

Calculez l'IMC avant et après confinement, et le delta (différence) entre ces valeurs.

```
IMC_av <-poids_tous / (taille_tous_m)^2
IMC_av
```

```
    Marc   Sophie    Julie Francois    Marie    Clara     Theo
24.85795 23.83300 24.53896 22.93282 20.36189 20.38157 28.57628
```

```
IMC_ap <- poids_ap / (taille_tous_m)^2
IMC_ap
```

```
    Marc   Sophie    Julie Francois    Marie    Clara     Theo
26.47211 24.24392 25.65437 22.41748 20.36189 22.38631 29.14215
```

```
IMC_delta <-IMC_ap - IMC_av
IMC_delta
```

```
     Marc     Sophie      Julie   Francois      Marie      Clara       Theo
1.6141529  0.4109139  1.1154075 -0.5153444  0.0000000  2.0047446  0.5658669
```

7.1. Les vecteurs

Générer des séries de valeurs:

Soit ”x” un vecteur contenant 6 entiers allant de 1 à 6 :

```
x <- 1:6              # renvoie [1]  1  2  3  4  5  6   → ordre croissant
x <- 6:1              # renvoie [1]  6  5  4  3  2  1   → ordre décroissant

# On peut aussi générer des séquences avec la fonction seq()
x <- seq (from=1, to=6, by=1)              # by = défini le pas de sélection
x <- seq (from=1, to=6, length.out=6)      # length.out = nombre de valeurs voulues

# La fonction rep() permet de répéter des valeurs un certain nombre de fois
rep(1:6, 1)           # renvoie [1]  1  2  3  4  5  6
```

NB: rep() fonctionne aussi avec d'autre type d'éléments :

```
rep (TRUE, 5)                 # renvoie [1] TRUE TRUE TRUE TRUE TRUE
rep (c ("A", "B"), 3)         # renvoie [1] "A" "B"  "A" "B"  "A" "B"
```

7.1. Les vecteurs

Générez les vecteurs suivants, chacun de 2 façons différentes :

```
[1] 10.0 12.5 15.0
```

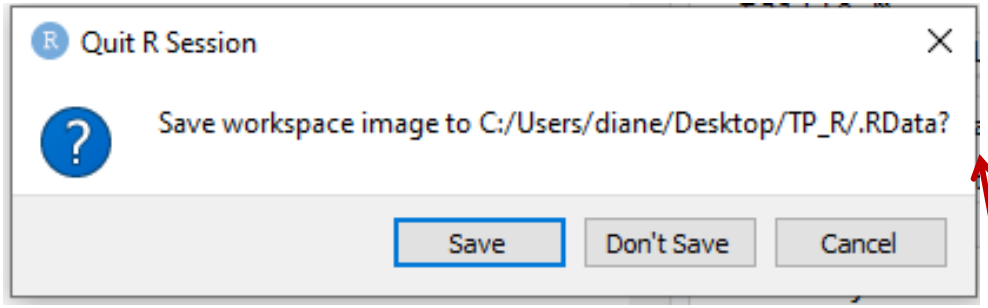
```
[1] 15 16 17 18 19 20 15 16 17 18 19 20
```

```
[1]  4  5  6  7 10  9  8
```

```
[1] "Rouge"  "Bleu"   "Rouge"  "Bleu"   "Rouge"  "Bleu"   "Vert"   "Vert"
    "Orange" "Orange" "Orange" "Orange"
```

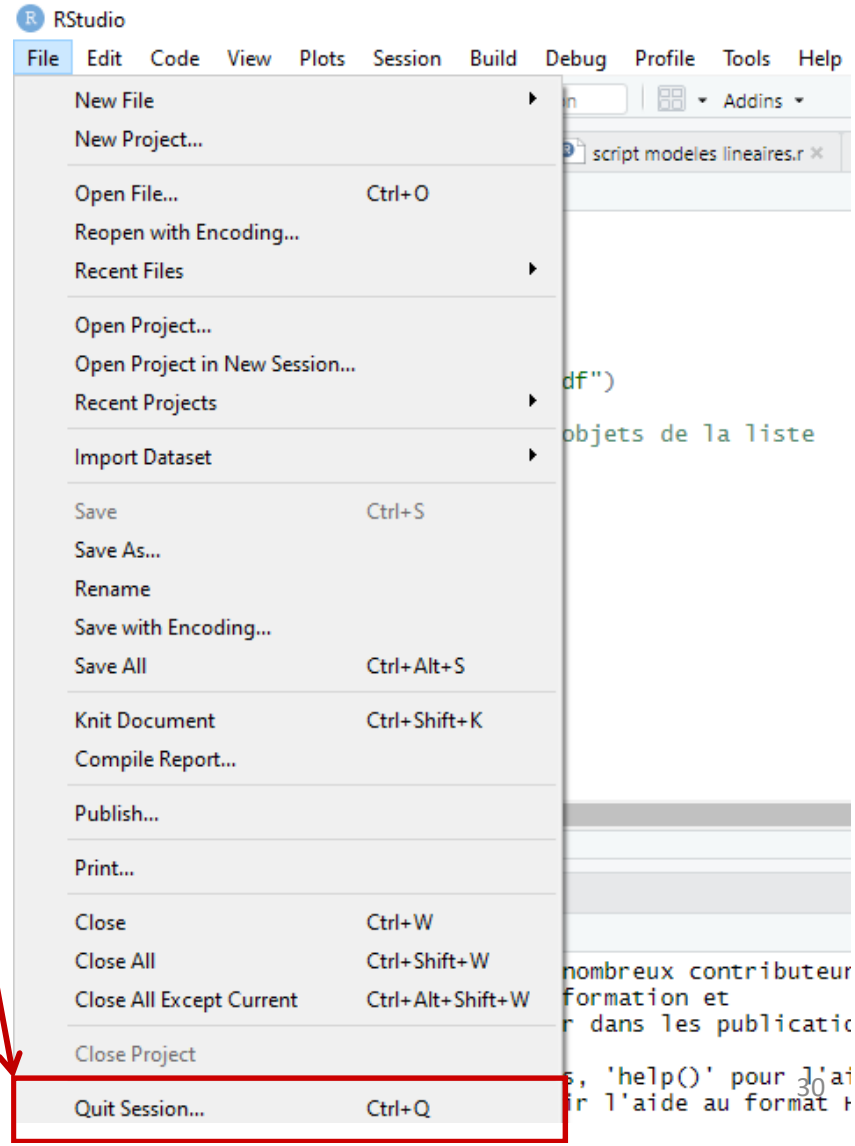
Sortez moi de là !!

Pour quitter R, rien de plus simple :



Sauver et ouvrir votre répertoire de travail, vous y trouverez :

- Un raccourci pour ouvrir votre précédente session R sauvegardée
- Un fichier de type texte (.txt ou .Rhistory) avec l'historique des commandes tapées



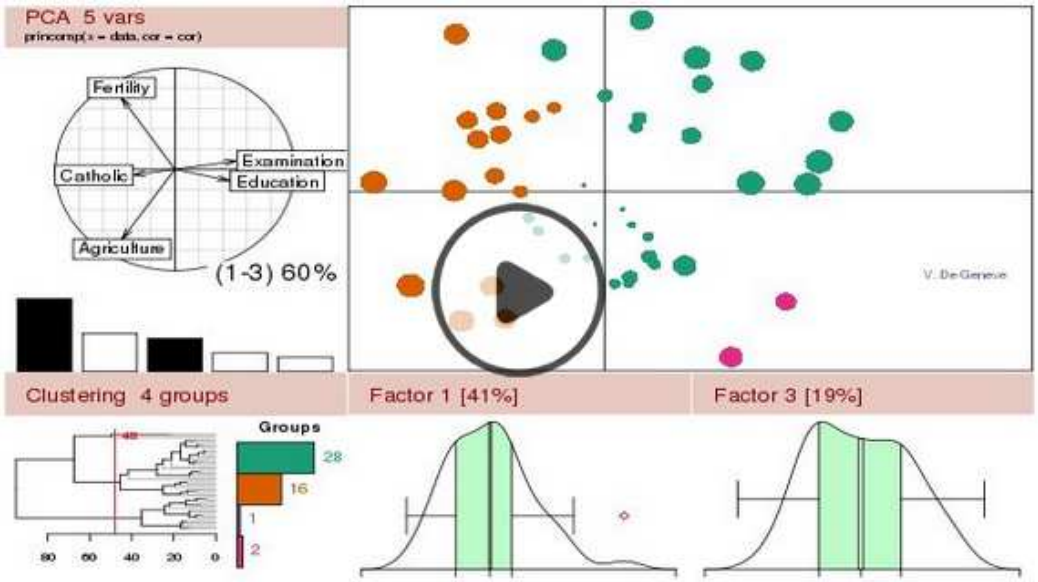
Plateforme Fun Mooc

(cours gratuits en ligne)

https://www.fun-mooc.fr

Introduction aux statistiques avec R

… inscrivez-vous !!! …



université PARIS-SACLAY

Inscription

Du 22 juillet 2021 au 20 octobre 2021

Cours

Du 13 septembre 2021 au 26 octobre 2021

Langues

Français

S'inscrire maintenant