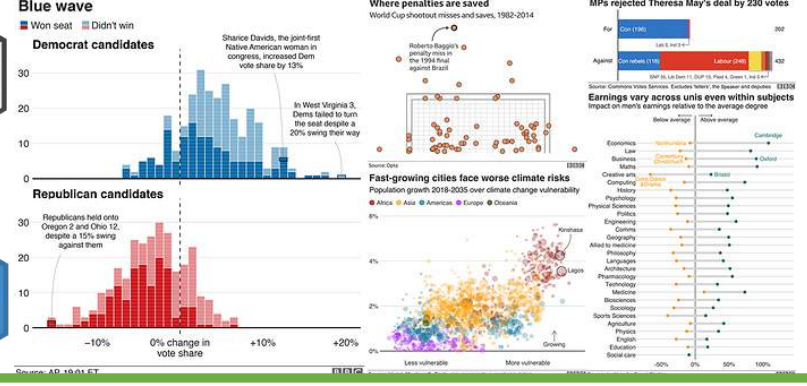


Initiation à la data science avec R

TP 0

Prise en main du logiciel R et de l'environnement RStudio

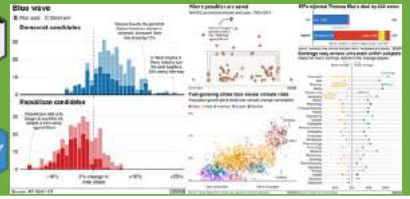
Valéry BOURNY
Olivier DURAND-DROUHIN

UniLaSalle Terre & Sciences

Parrainé par les entreprises

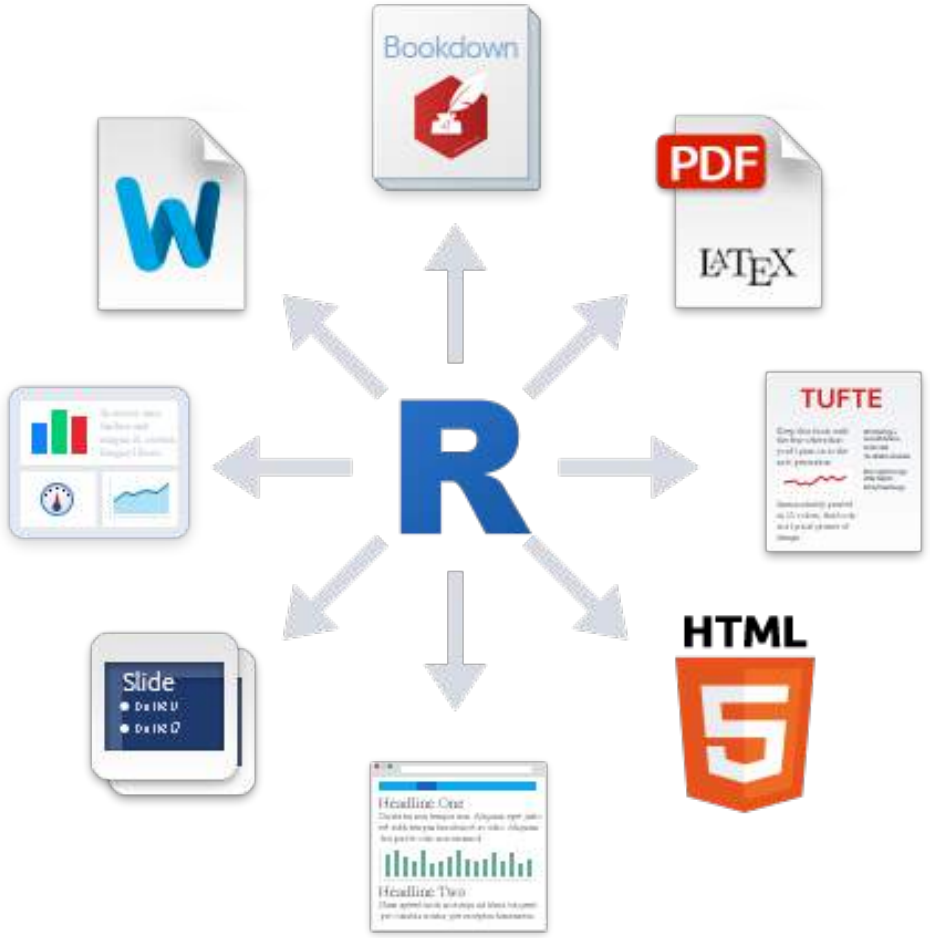
ENEDIS L'ELECTRICITE EN RESEAU

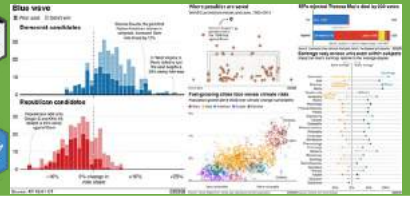
MERSEN

TP 0 : **Prise en main du logiciel R et de l'environnement RStudio**

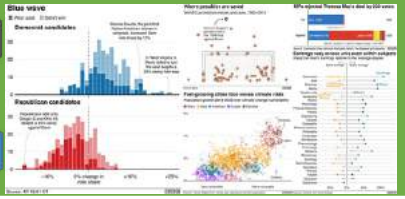
- ***R*** est langage orienté vers **le traitement de données** et **l'analyse statistique**.
- C'est un logiciel (ou plus exactement un langage de programmation):
  - **Libre** publié sous licence GNU GPL
  - **Multiplateforme** (Windows, Mac OS X, Linux)
  - **Gratuit**
  - **Très puissant** avec des fonctionnalités de base qui peuvent être étendues à l'aide de nombreuses extensions (plusieurs milliers)
  - Avec d'excellentes **capacités graphiques**
  - En constant développement avec une **communauté** de plus en plus importante
  - **Multidisciplinaire** (tous les secteurs scientifiques et sciences sociales)
  - Avec de nombreux formats d'export





TP 0 : **Prise en main du logiciel R et de l'environnement RStudio**

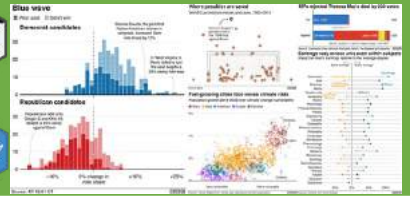
- **RStudio** est un environnement de développement intégré qui complète *R* en facilitant son usage au quotidien.
- C'est un logiciel :
  - Libre
  - Gratuit
  - qui fonctionne sous Windows, Mac OS X et Linux.
  - C'est une interface bien supérieure à celles fournies par défaut lorsqu'on installe R sous Windows ou sous Mac
- Il fournit un éditeur de script avec
  - coloration syntaxique
  - des fonctionnalités pratiques d'édition et d'exécution du code
  - un affichage simultané du code, de la console **R**, des fichiers, graphiques et pages d'aide
  - une gestion des extensions
- Il intègre de base divers outils comme par exemple la production de rapports automatisés au format **Rmarkdown**.
- Une interface uniquement anglophone.
- En constant développement.



TP 0 : **Prise en main du logiciel R et de l'environnement RStudio**

TP0

- **Installation de R et RStudio**
- **Environnement de travail**
- **Premier pas avec R et Rstudio**
- **Les packages :** tidyverse, ggplot2 et les autres
- **Création d'un document Rmarkdown**

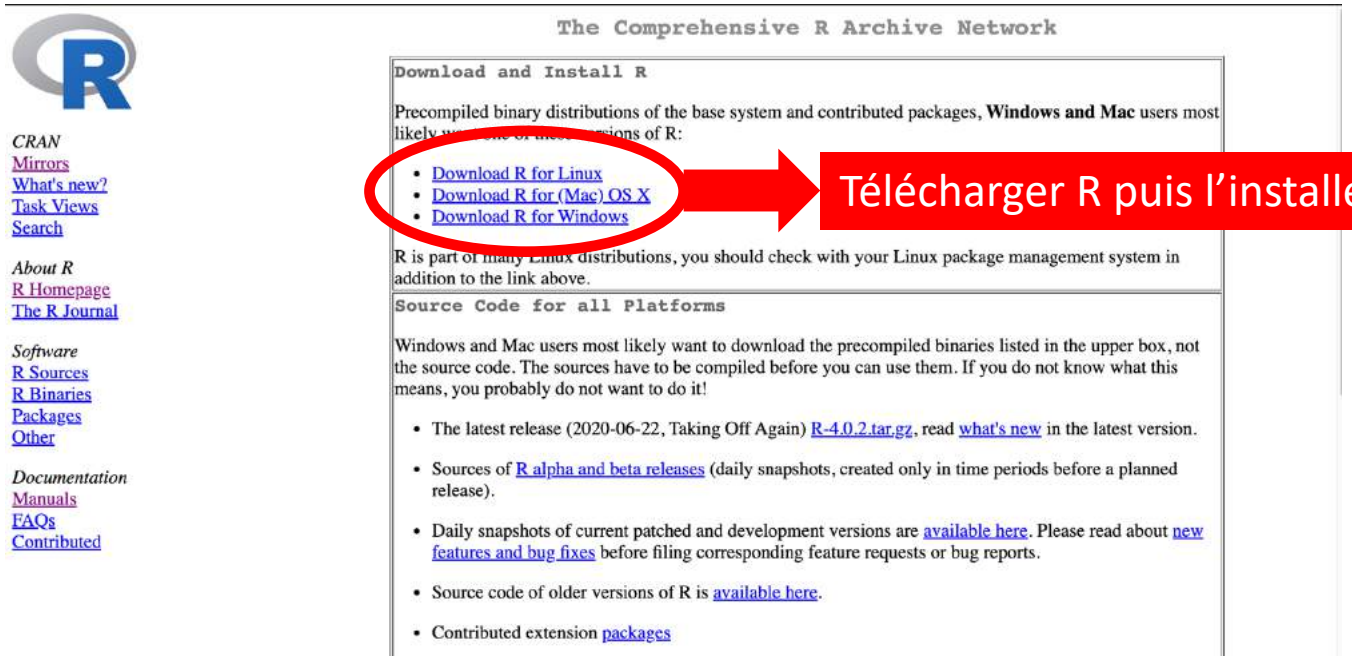


TP 0 : **Prise en main du logiciel R et de l'environnement RStudio**

➢ Installation de R et RStudio

R est un langage et un environnement d'analyse statistiques et graphiques. Il est compatible Windows/MAC/Linux.
L'adresse du site officiel est :

http://www.r-project.org/



CRAN
Mirrors
What's new?
Task Views
Search

About R
R Homepage
The R Journal

Software
R Sources
R Binaries
Packages
Other

Documentation
Manuals
FAQs
Contributed

The Comprehensive R Archive Network

Download and Install R

Precompiled binary distributions of the base system and contributed packages, **Windows and Mac** users most likely

- Download R for Linux
- Download R for (Mac) OS X
- Download R for Windows

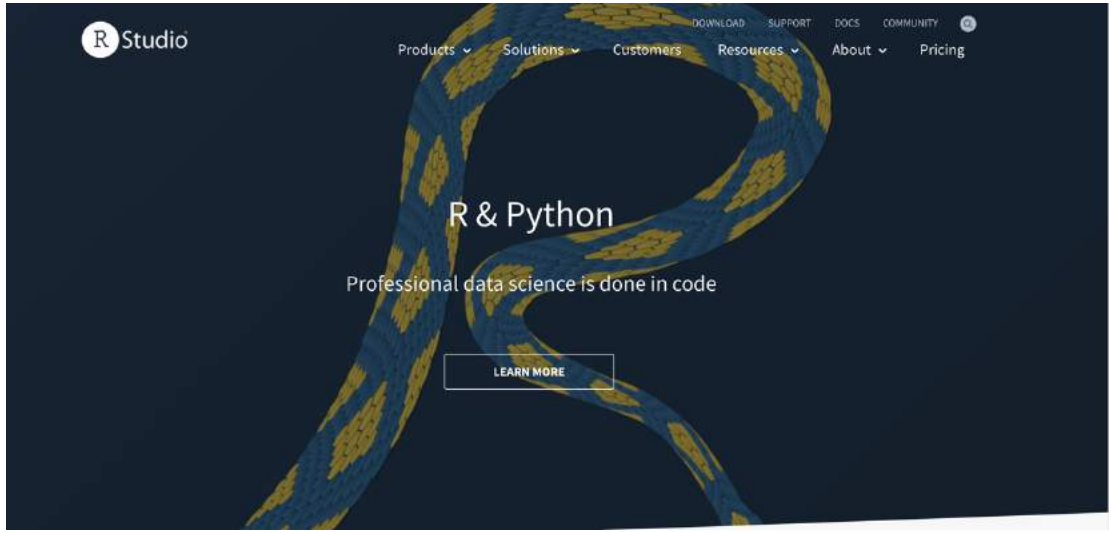
Télécharger R puis l'installer

R is part of many Linux distributions, you should check with your Linux package management system in addition to the link above.

Source Code for all Platforms

Windows and Mac users most likely want to download the precompiled binaries listed in the upper box, not the source code. The sources have to be compiled before you can use them. If you do not know what this means, you probably do not want to do it!

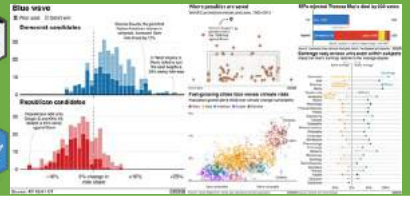
- The latest release (2020-06-22, Taking Off Again) R-4.0.2.tar.gz, read what's new in the latest version.
- Sources of R alpha and beta releases (daily snapshots, created only in time periods before a planned release).
- Daily snapshots of current patched and development versions are available here. Please read about new features and bug fixes before filing corresponding feature requests or bug reports.
- Source code of older versions of R is available here.
- Contributed extension packages



Remarque : il est préférable de commencer par installer ***R*** avant d'installer ***RStudio***

Puis aller sur RStudfio

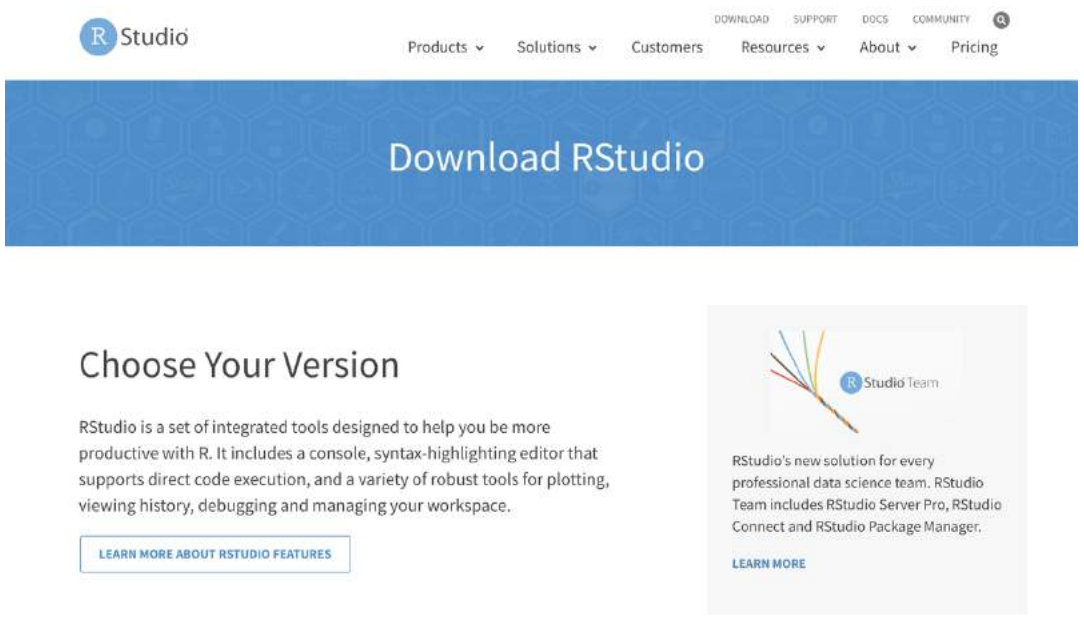
https://rstudio.com/

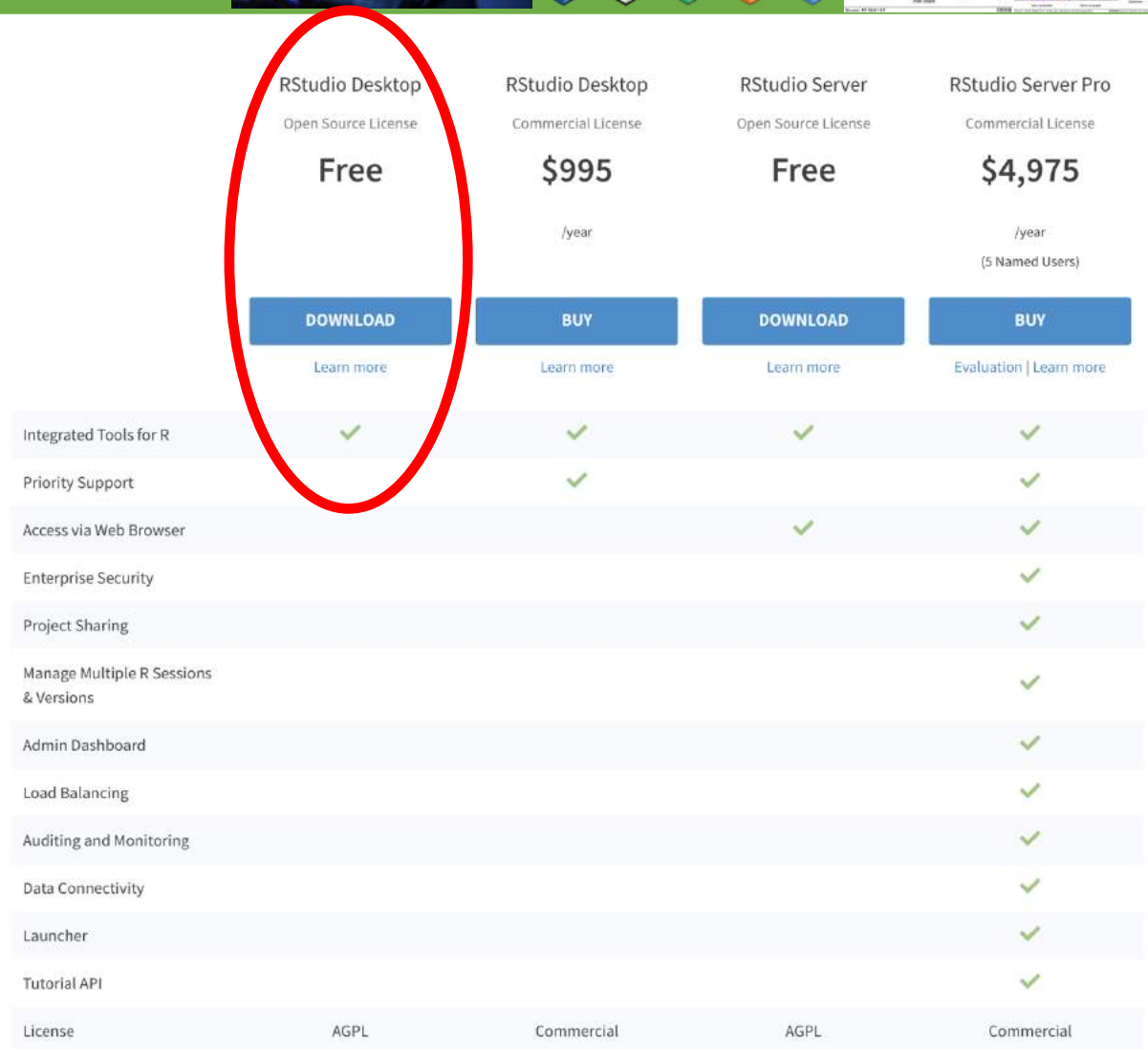


TP 0 : **Prise en main du logiciel R et de l'environnement RStudio**

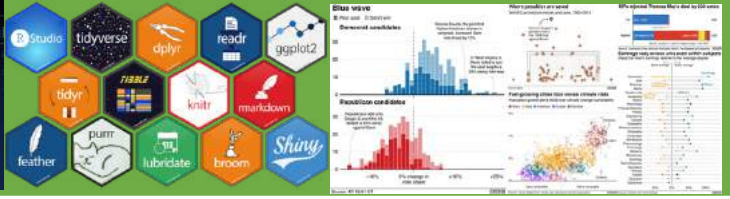
➢ Installation de R et RStudio

https://rstudio.com/products/rstudio/download/





Télécharger l'édition Open Source de RStudio Desktop



TP 0 : **Prise en main du logiciel R et de l'environnement RStudio**

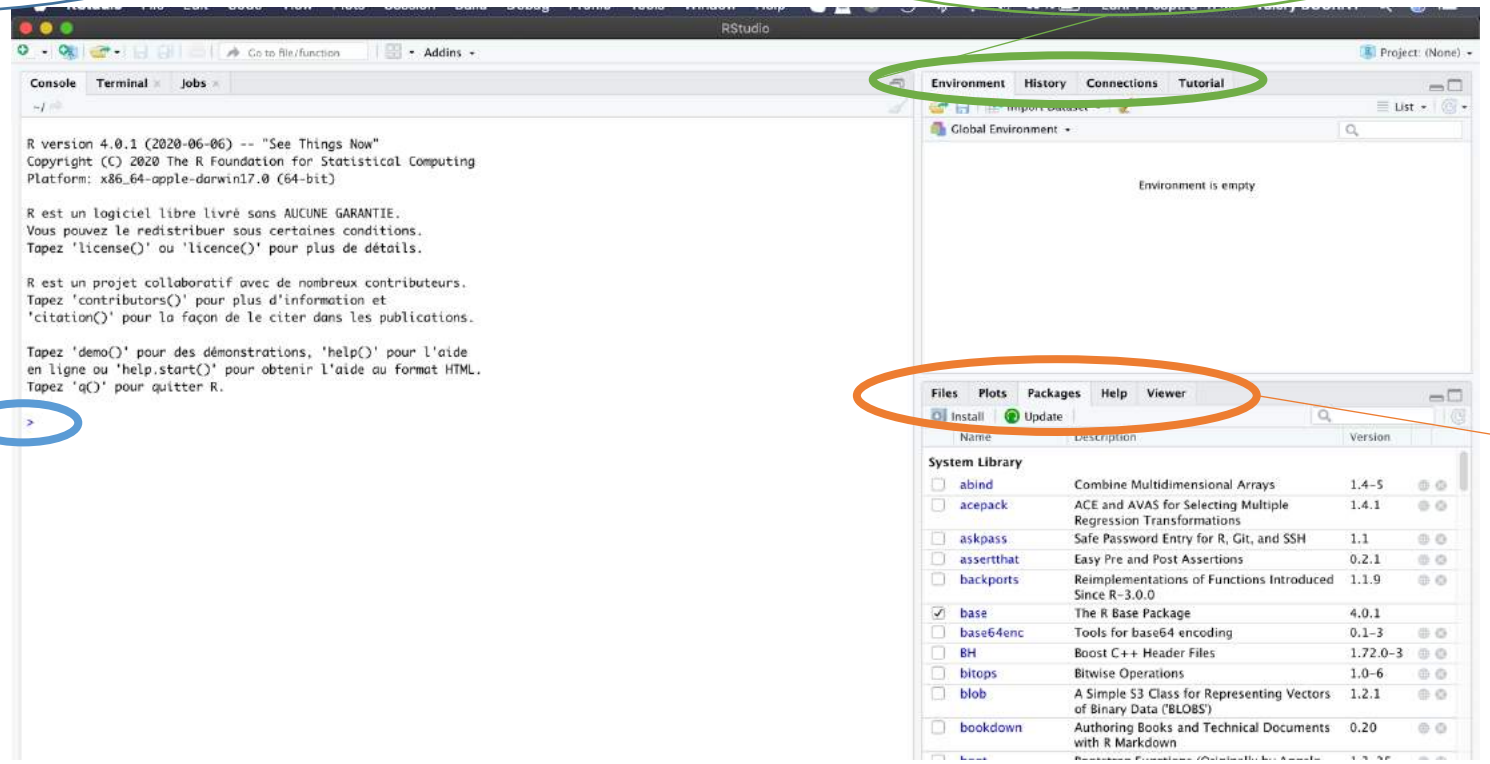
➢ Environnement de travail

Lancer RStudio

L'écran principal est découpé en trois grandes zones

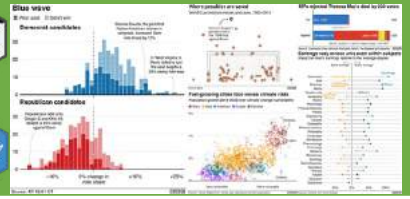
Console ou ligne de commande R
(C'est la zone d'interaction avec R)

Navigateur d'environnement de travail (onglet *Environment*), d'historique des commandes (onglet *History*)



Navigateur de fichiers du projet (onglet *Files*, de packages (onglet *Packages*), de graphiques (onglet *Plots*), de l'aide en ligne (onglet *Help*)

Invite de commande (ou *prompt)*
R est disponible et en attente de votre prochaine commande



TP 0 : **Prise en main du logiciel R et de l'environnement RStudio**

➢ Environnement de travail

Saisir les lignes de code suivantes

```
> x<-2*5
> y<-0.3^2
> z<-x-y
> Text<-"Hello"
```

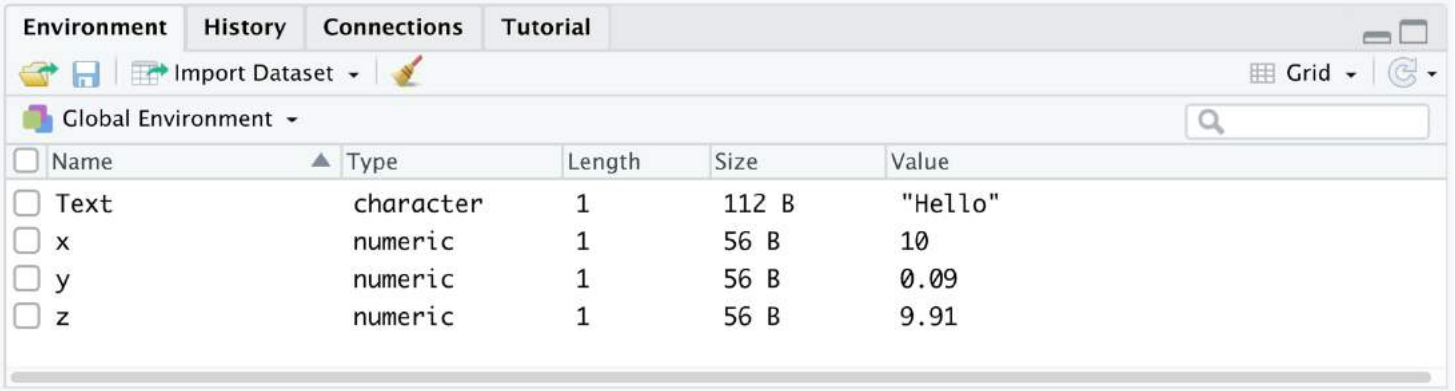
- Le résultat d'une opération peut être stocker dans une *variable* à l'aide de l'opérateur d'affectation **<-**
- les résultats des analyses sont eux aussi stockés dans des objets et sont dès lors manipulables => conduite des analyses plus facile et rapide
- Les nombres sont saisies « à l'anglaise » (en utilisant le **.** comme séparateur pour les décimales).

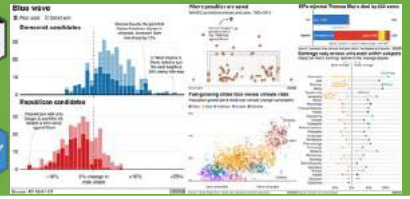
Afficher les variables x, y, z et Text

```
> x
[1] 10
```

- Par défaut l'affichage est réduit au minimum et c'est l'utilisateur qui demande à voir des résultats supplémentaires ou plus détaillés.
- **[1]** : la première (et unique) coordonnée du vecteur x est 1.

Consulter l'environnement global
Consulter l'historique





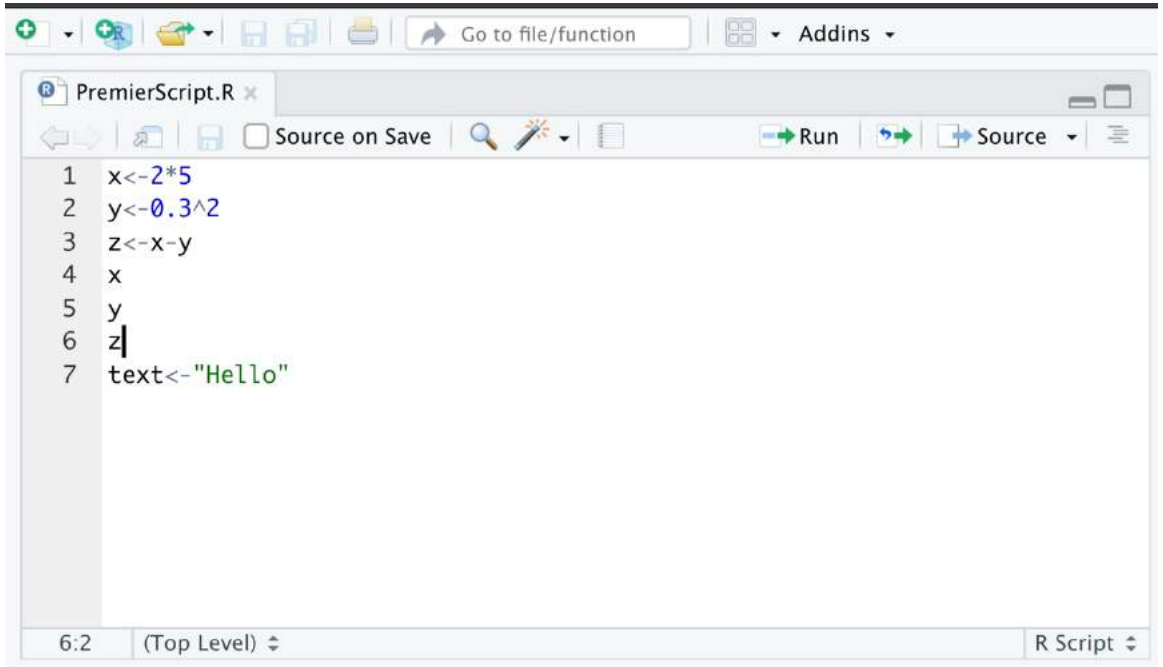
TP 0 : **Prise en main du logiciel R et de l'environnement RStudio**

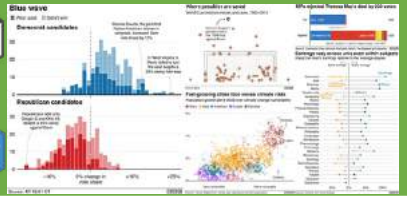
➢ Premier pas avec R et RStudio — Les scripts

- Les commandes sont plutôt regroupées dans **des scripts** (de simples fichiers texte).

- Un script regroupe **une suite d'instructions** (les commandes) effectuant **les différentes opérations d'une analyse.**

- Un script présente de nombreux avantages :
  - Il garde par ordre chronologique l'ensemble des étapes d'une analyse, de l'importation des données à leur analyse en passant par les manipulations et les recodages;
  - A tout moment on peut revenir en arrière et modifier ce qui a été fait;
  - Il est très rapide de ré exécuter une suite d'opérations complexes;
  - On peut très facilement mettre à jour les résultats en cas de modification des données sources;
  - Le script garantit, sous certaines conditions, **la reproductibilité des résultats obtenus**.

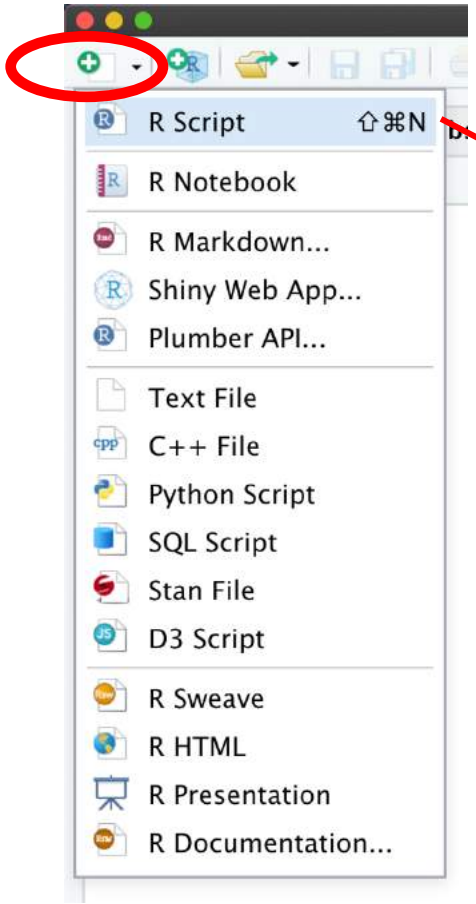


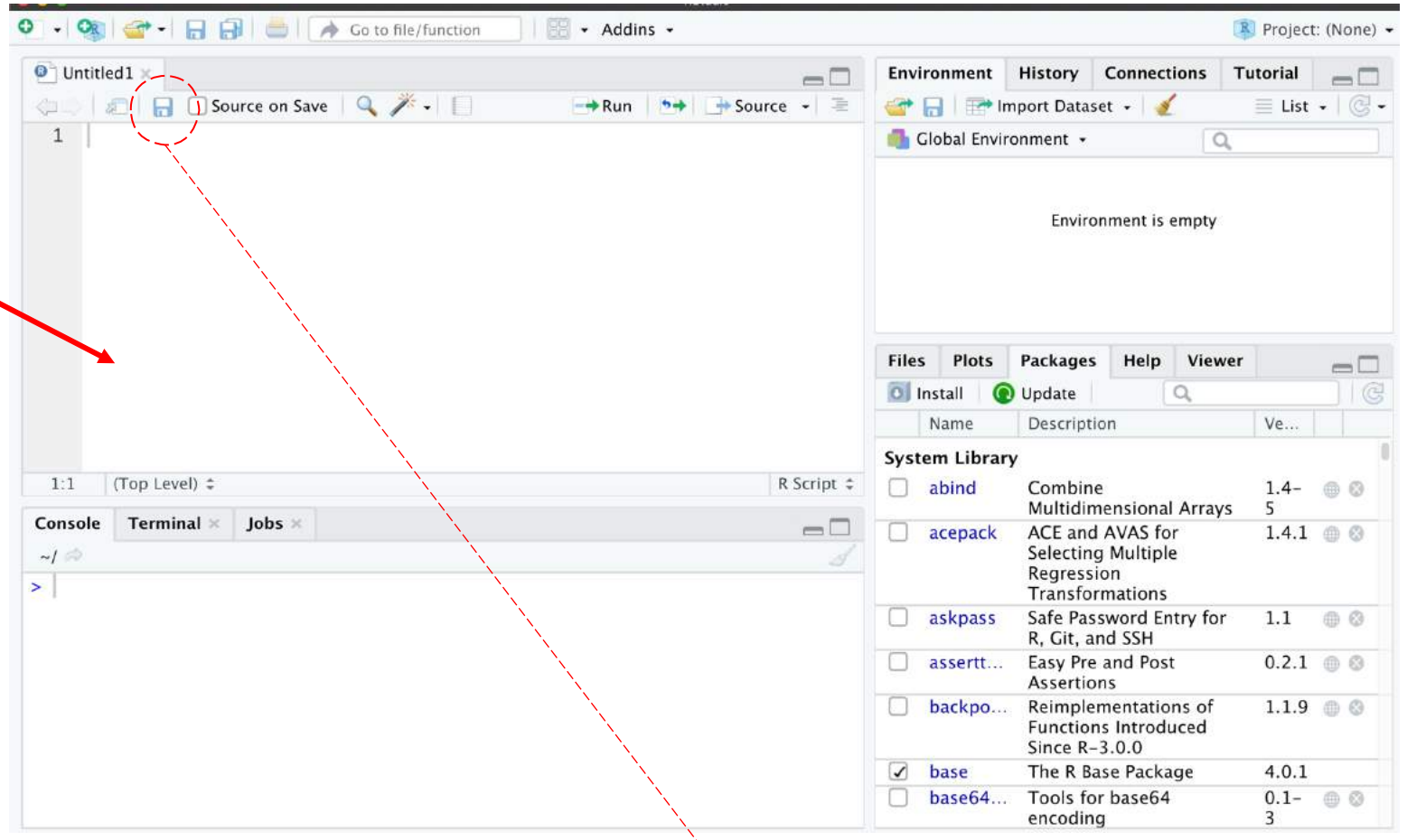


TP 0 : **Prise en main du logiciel R et de l'environnement RStudio**

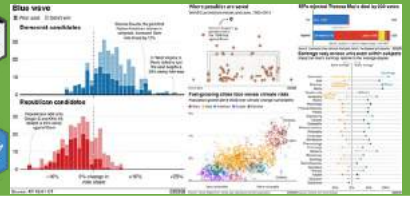
➢ Premier pas avec R et RStudio — Les scripts

Créer un script





On peut enregistrer notre script à tout moment dans un fichier avec l'extension *.R*
(que vous pouvez ouvrir avec n'importe quel éditeur de texte).

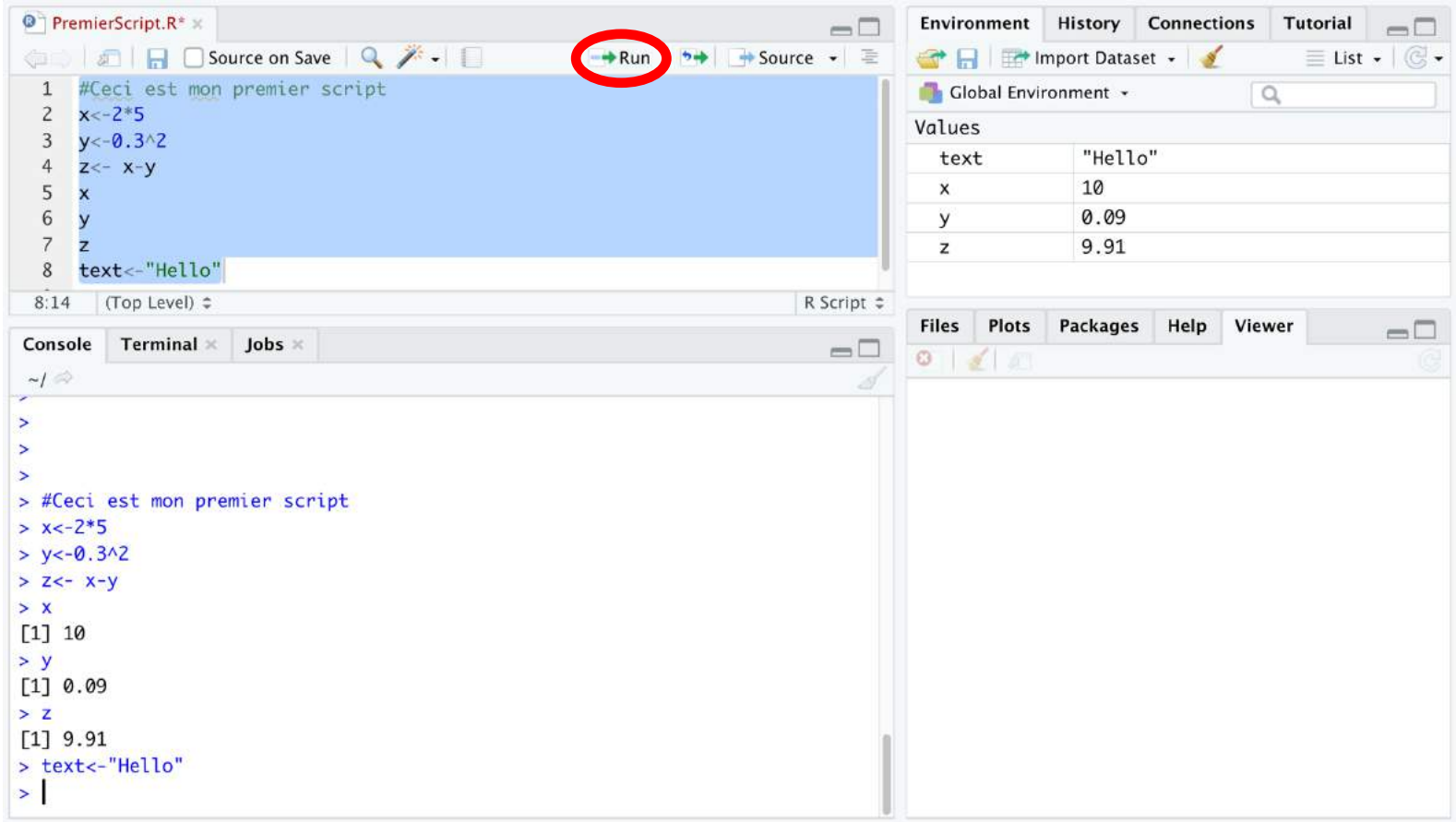


TP 0 : **Prise en main du logiciel R et de l'environnement RStudio**

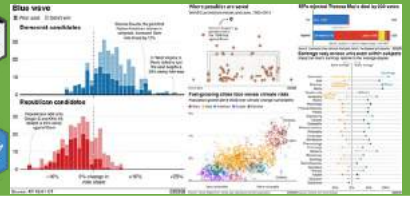
➢ Premier pas avec R et RStudio

Les scripts

Taper votre premier script et l'exécuter



- On peut sélectionner plusieurs lignes avec la souris ou le clavier et cliquer sur *Run* (ou utiliser le raccourci clavier Ctrl + Entrée (Cmd + Entrée sous Mac)), et l'ensemble des lignes est exécuté d'un coup.
- Les commentaires # sont un élément très important d'un script. Il s'agit de texte libre, ignoré par R, et qui permet de décrire les étapes du script, sa logique, les raisons pour lesquelles on a procédé de telle ou telle manière …



TP 0 : **Prise en main du logiciel R et de l'environnement RStudio**

➢ Premier pas avec R et RStudio

Les objets

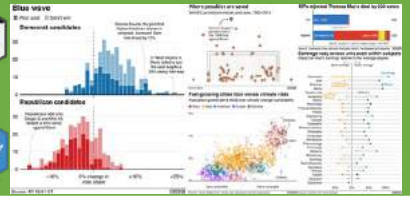
Saisir le script suivant

Exécuter le

PremierScript.R | Production.R*

Source on Save

```
1  #Saisie de la production en courant d'un parc éolien P1 par mois
2  P1<-c(11,24,10,12,11,15,7,11,8,13,9,20) #vecteur numeric
3  mois<-c("janvier", "février", "mars","avril", "mai","juin","juillet","août",
4          "septembre","octobre","novembre","décembre") #vecteur character
5  ProdP1<-data.frame(Mois=mois,Prod=P1) #jeux de données (list)
6
7  #Affichage simple des variables P1
8  P1
9  mois
10 ProdP1
11
12 #Affichage du 4ème mois et de sa production
13 mois[4]
14 P1[4]
15 ProdP1[4,]
16
17 #Affichage des valeurs extrêmes
18 min(P1)
19 max(P1)
20
21 #Calcul de la production totale de P1 sur 1 an
22 Ptot<-sum(P1)
23 Ptot
24
25 #Calcul de la production cumulée de P1 sur 1 an
26 Pcum<-cumsum(P1)
27 Pcum
28
29 #Calcul de la moyenne de la production de P1
30 Pmoy<-mean(P1)
31 Pmoy
32
```

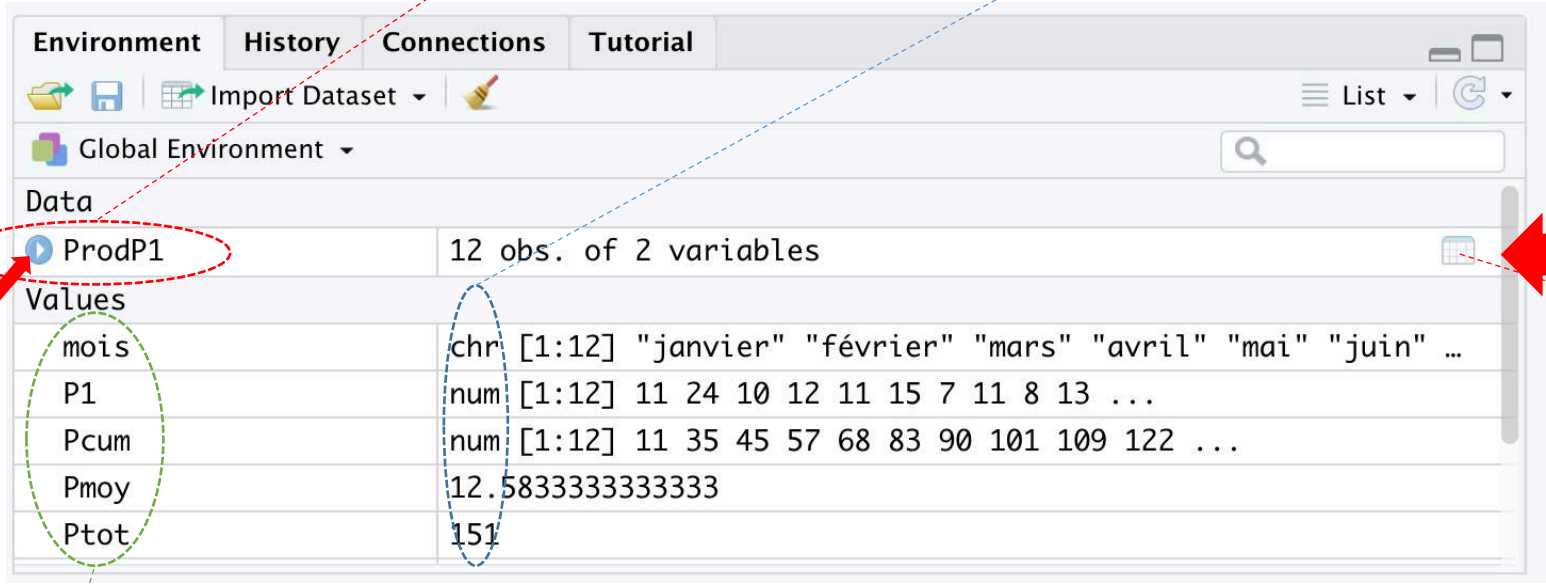


TP 0 : **Prise en main du logiciel R et de l'environnement RStudio**

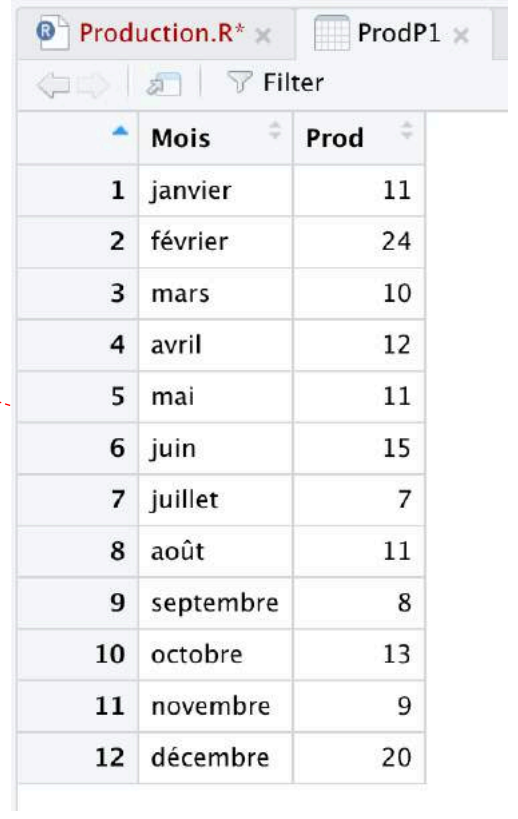
➢ Premier pas avec R et RStudio — Les objets

- Les **modes** (ou types) d'un objet dans R sont :
  - null (objet vide) : NULL
  - logical (booléen) : TRUE, FALSE
  - numeric (réel, entier)
  - complex (nombre complexe)
  - character (chaîne de caractères)
  - …

- Les **data-frames** (ou jeu de données) : ce sont des listes particulières dont les composantes sont de même longueur mais dont les modes peuvent différés.



Cliquer

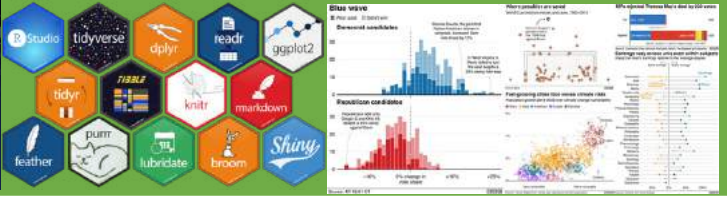


Production.R* | ProdP1

Filter

	Mois	Prod
1	janvier	11
2	février	24
3	mars	10
4	avril	12
5	mai	11
6	juin	15
7	juillet	7
8	août	11
9	septembre	8
10	octobre	13
11	novembre	9
12	décembre	20

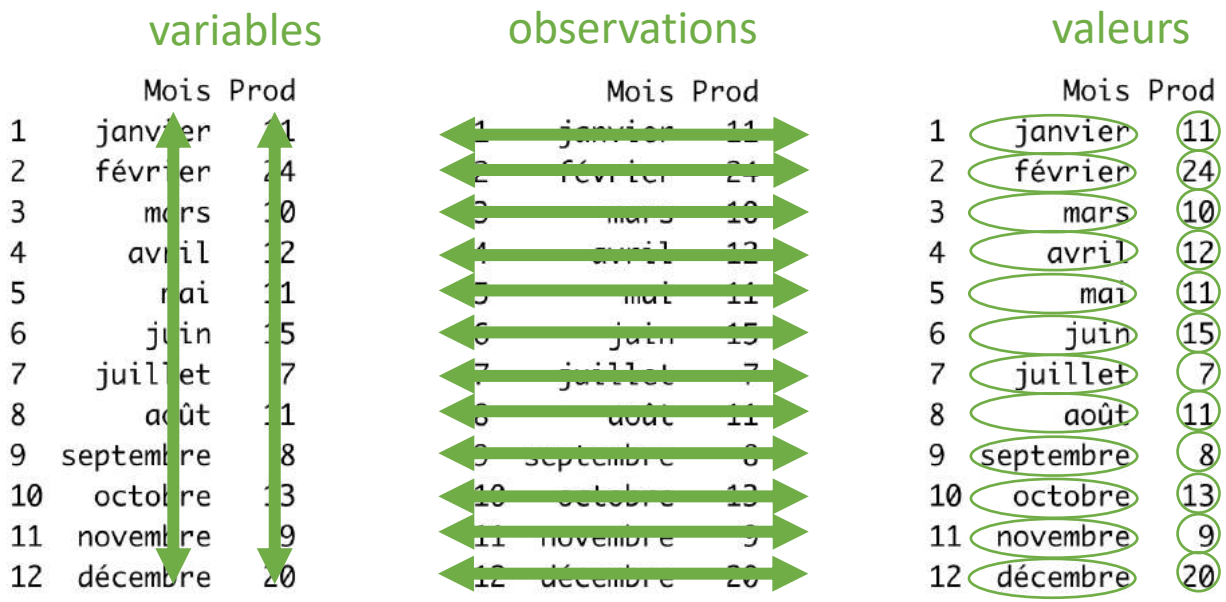
- Les **vecteurs** sont des objets à mode unique composés de valeurs appelées composantes (ou coordonnées). Leur longueur est le nombre d'éléments de ceux-ci.



TP 0 : **Prise en main du logiciel R et de l'environnement RStudio**

➢ Premier pas avec R et RStudio — Les objets

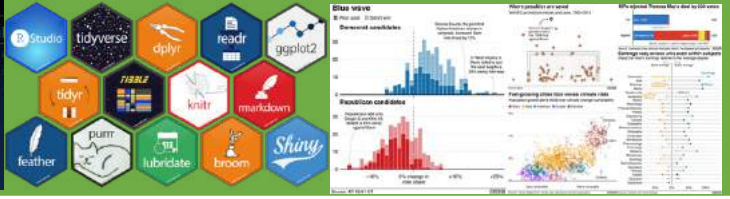
Concaténation des vecteurs de mêmes tailles et éventuellement de modes différents ➡ Un tableau de données constitué de variables quantitatives et/ou qualitative mesurées sur les mêmes individus (très utilisé en analyse des données)



*Le tableau de données est **au format tidy (bien rangé)** : chaque variable doit avoir sa propre colonne, chaque observation doit avoir sa propre ligne, chaque valeur doit avoir sa propre cellule.*

```
Console  Terminal  Jobs
~/
> #Affichage simple des variables P1
> P1
 [1] 11 24 10 12 11 15  7 11  8 13  9 20
> mois
 [1] "janvier"   "février"   "mars"      "avril"     "mai"       "juin"
 [7] "juillet"   "août"      "septembre" "octobre"   "novembre"  "décembre"
> ProdP1
        Mois Prod
1    janvier   11
2    février   24
3       mars   10
4      avril   12
5        mai   11
6       juin   15
7    juillet    7
8       août   11
9  septembre    8
10   octobre   13
11  novembre    9
12  décembre   20
>
> #Affichage du 4ème mois et de sa production
> mois[4]
[1] "avril"
> P1[4]
[1] 12
> ProdP1[4,]
   Mois Prod
4 avril   12
>
> #Affichage des valeurs extrêmes
> min(P1)
[1] 7
> max(P1)
[1] 24
>
> #Calcul de la production totale de P1 sur 1 an
> Ptot<-sum(P1)
> Ptot
[1] 151
>
> #Calcul de la production cumulée de P1 sur 1 an
> Pcum<-cumsum(P1)
> Pcum
 [1]  11  35  45  57  68  83  90 101 109 122 131 151
>
> #Calcul de la moyenne de la production de P1
> Pmoy<-mean(P1)
> Pmoy
[1] 12.58333
```

Extraction/Sélection de composantes



TP 0 : **Prise en main du logiciel R et de l'environnement RStudio**

➢ Premier pas avec R et RStudio — Les objets

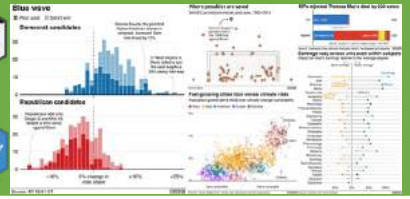
- Les **fonctions** : elles permettent d'effectuer des calculs, d'obtenir des résultats et d'accomplir des actions. une fonction a un *nom*, elle prend en entrée entre parenthèses un ou plusieurs *arguments* (ou *paramètres*) et retourne un *résultat*.

Nom *Argument* *Résultat*

```
> Pcum<-cumsum(P1)
> Pcum
 [1]  11  35  45  57  68  83  90 101 109 122 131 151
```

- Des fonctions sont prédéfinies dans *R* (*min*, *max*, *mean*, *c*, …)
- Possibilité également de créer ses propres fonctions

```
Console  Terminal  Jobs
~/
> #Affichage simple des variables P1
> P1
 [1] 11 24 10 12 11 15  7 11  8 13  9 20
> mois
 [1] "janvier"   "février"   "mars"      "avril"     "mai"       "juin"
 [7] "juillet"   "août"      "septembre" "octobre"   "novembre"  "décembre"
> ProdP1
        Mois Prod
1    janvier   11
2    février   24
3      mars   10
4     avril   12
5       mai   11
6      juin   15
7    juillet    7
8      août   11
9  septembre    8
10   octobre   13
11  novembre    9
12  décembre   20
>
> #Affichage du 4ème mois et de sa production
> mois[4]
[1] "avril"
> P1[4]
[1] 12
> ProdP1[4,]
   Mois Prod
4 avril   12
>
> #Affichage des valeurs extrêmes
> min(P1)
[1] 7
> max(P1)
[1] 24
>
> #Calcul de la production totale de P1 sur 1 an
> Ptot<-sum(P1)
> Ptot
[1] 151
>
> #Calcul de la production cumulée de P1 sur 1 an
> Pcum<-cumsum(P1)
> Pcum
 [1]  11  35  45  57  68  83  90 101 109 122 131 151
>
> #Calcul de la moyenne de la production de P1
> Pmoy<-mean(P1)
> Pmoy
[1] 12.58333
```



TP 0 : **Prise en main du logiciel R et de l'environnement RStudio**

➢ Premier pas avec R et RStudio

Les objets

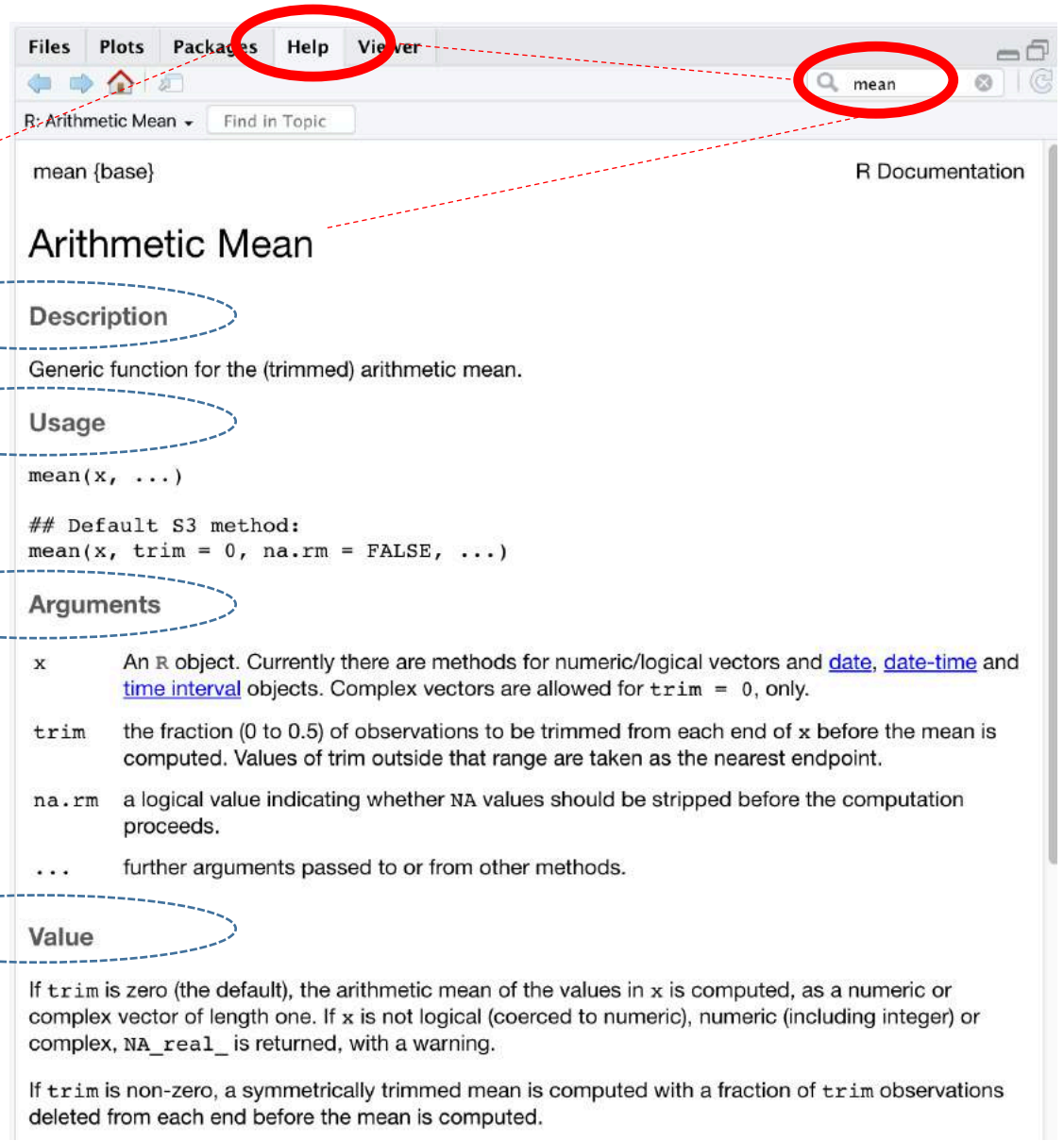
Pour obtenir de l'aide sur une fonction :

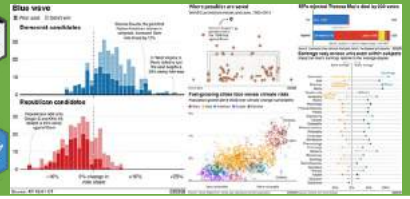
```
> help("mean")
```

ou

```
> ?max
```

ou





TP 0 : **Prise en main du logiciel R et de l'environnement RStudio**

➢ Premier pas avec R et RStudio

Les objets → Saisir les commandes

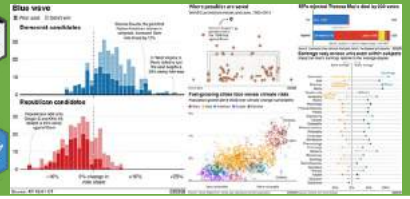
```
33  str(ProdP1)
34  str(ProdP1$Mois)
35  str(ProdP1$Prod)
36
37  summary(ProdP1)
```

```
> str(ProdP1)
'data.frame':   12 obs. of  2 variables:
 $ Mois: chr  "janvier" "février" "mars" "avril" ...
 $ Prod: num  11 24 10 12 11 15 7 11 8 13 ...
> str(ProdP1$Mois)
 chr [1:12] "janvier" "février" "mars" "avril" "mai" ...
> str(ProdP1$Prod)
 num [1:12] 11 24 10 12 11 15 7 11 8 13 ...
> summary(ProdP1)
     Mois                Prod
 Length:12          Min.   : 7.00
 Class :character   1st Qu.: 9.75
 Mode  :character   Median :11.00
                    Mean   :12.58
                    3rd Qu.:13.50
                    Max.   :24.00
```

- la fonction ***str()*** renvoie un descriptif plus détaillé de la structure du tableau. Elle liste les différentes variables, indique leur mode et affiche les premières valeurs.

- Une opération très importante est l'accès aux variables du tableau (à ses colonnes) pour pouvoir les manipuler, effectuer des calculs, etc. On utilise pour cela l'opérateur ***$***, qui permet d'accéder aux colonnes du tableau.

- La fonction ***summary()*** permet d'obtenir d'un coup plusieurs indicateurs classiques de statistique.



TP 0 : **Prise en main du logiciel R et de l'environnement RStudio**

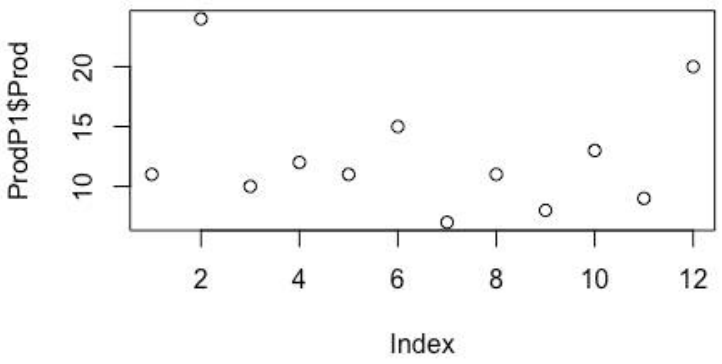
➢ Premier pas avec R et RStudio

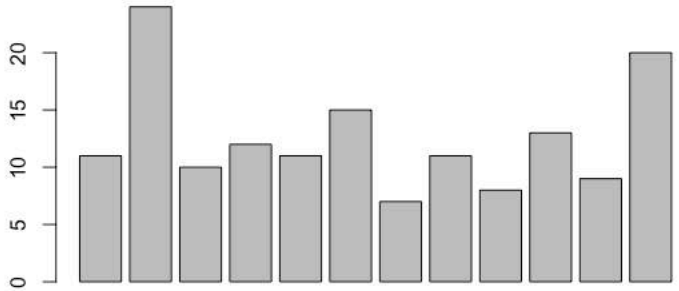
Visualiser les données → Point de départ d'une analyse statistique

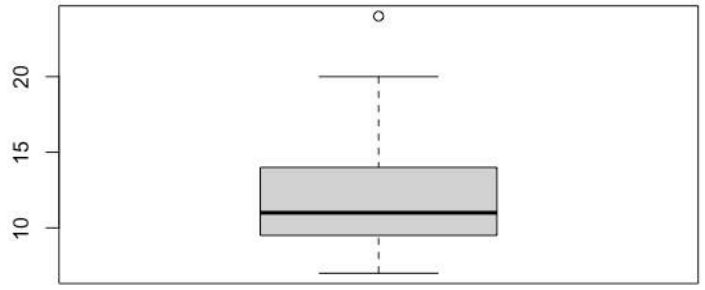
- *R* offre des graphiques de toute nature https://www.r-graph-gallery.com/
- Les fonctions graphiques conventionnelles : ***plot()***, ***boxplot()***, ***barplot()***, ***hist()*** …

```
39  #Graphiques de la production sur les 12 mois
40  plot(ProdP1$Prod)
41  barplot(ProdP1$Prod)
42  boxplot(ProdP1$Prod)
```

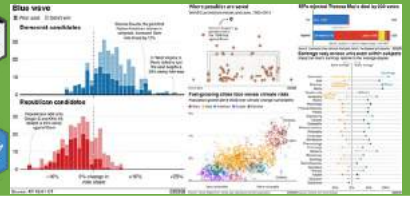
← Saisir les commandes





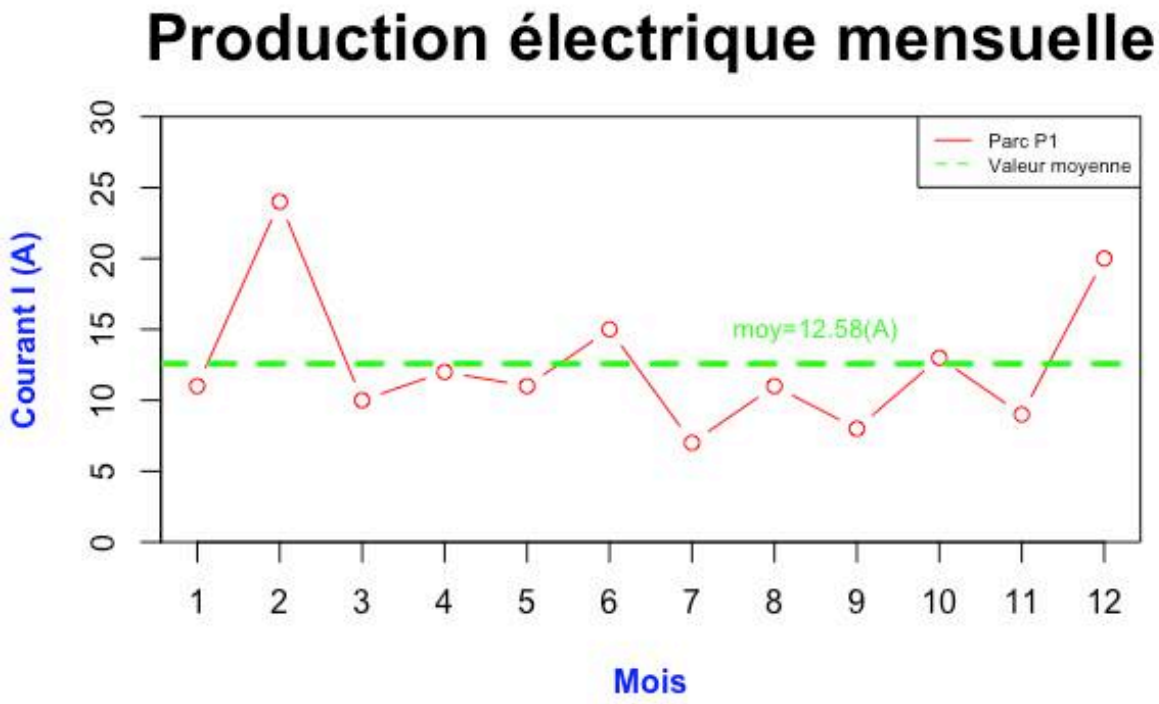


- Grande possibilité de personnaliser et d'améliorer le graphique.



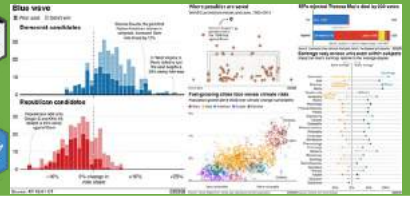
TP 0 : **Prise en main du logiciel R et de l'environnement RStudio**

➢ Premier pas avec R et RStudio — Visualiser les données




```
48  plot(ProdP1$MoisNum, ProdP1$Prod,type="b",
49    col="red", xlim=c(1,12),ylim=c(0,30),
50     yaxs="i",
51    main = "Production électrique mensuelle", cex.main = 2,
52    xlab="Mois",ylab="Courant I (A)", col.lab="blue",xaxp = c(1,12,11),
53    font.lab=2)
54
55  abline(h=mean(ProdP1$Prod),col="green", lty=2, lwd=3)
56  legend("topright", legend=c("Parc P1","Valeur moyenne"), col=c("red","green"),
57         lty=1:2, cex=0.6)
58  text(8.5,15,paste0("moy=",round(mean(ProdP1$Prod),2), "(A)"), col="green",
59       cex=0.8)
```

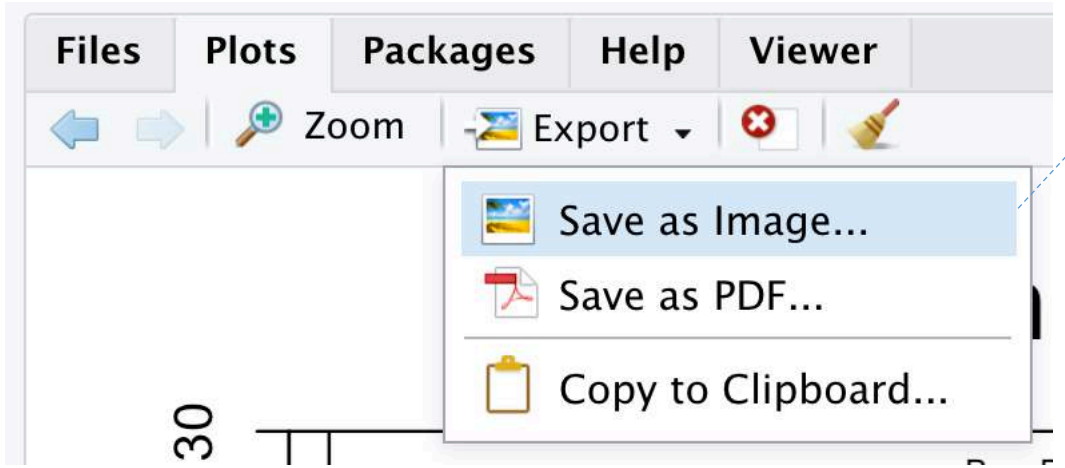
➡ De nombreux arguments dans les fonctions graphiques

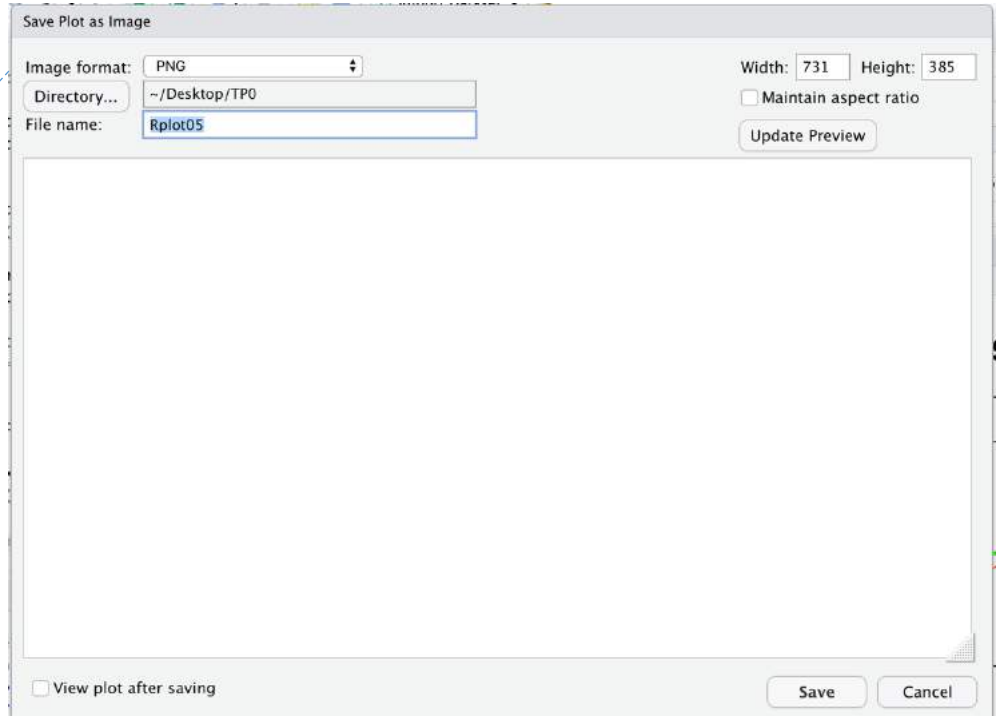


TP 0 : **Prise en main du logiciel R et de l'environnement RStudio**

➢ Premier pas avec R et RStudio — Visualiser les données

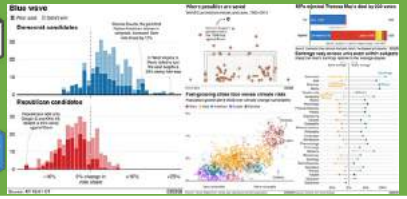
- Possibilité de sauvegarder le graphique





- *R* dispose de l'extension (package) ***ggplot2*** pour la production de graphes des plus élégants, des plus polyvalents et des plus sophistiqués de plus en plus utilisée

- *ggplot2* est l'une des composantes du ***tidyverse***.



TP 0 : **Prise en main du logiciel R et de l'environnement RStudio**

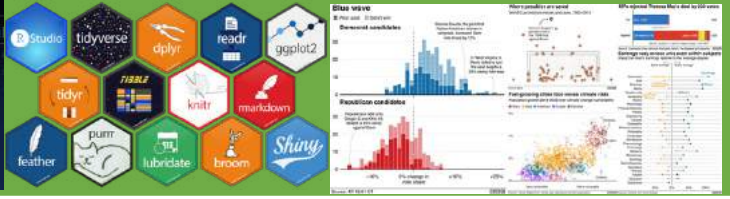
➢ **Les packages :** tidyverse, ggplot2 et les autres

- L'installation de base de *R* permet de faire énormément de choses, mais le langage dispose en plus d'un système d'extensions (les ***packages***) permettant d'ajouter facilement de nouvelles fonctionnalités.

  ➡ Plus de 8 000 packages

- La plupart des extensions sont développées et maintenues par la communauté des utilisateurs de R, et diffusées via un réseau de serveurs nommé ***CRAN*** (*Comprehensive R Archive Network*).

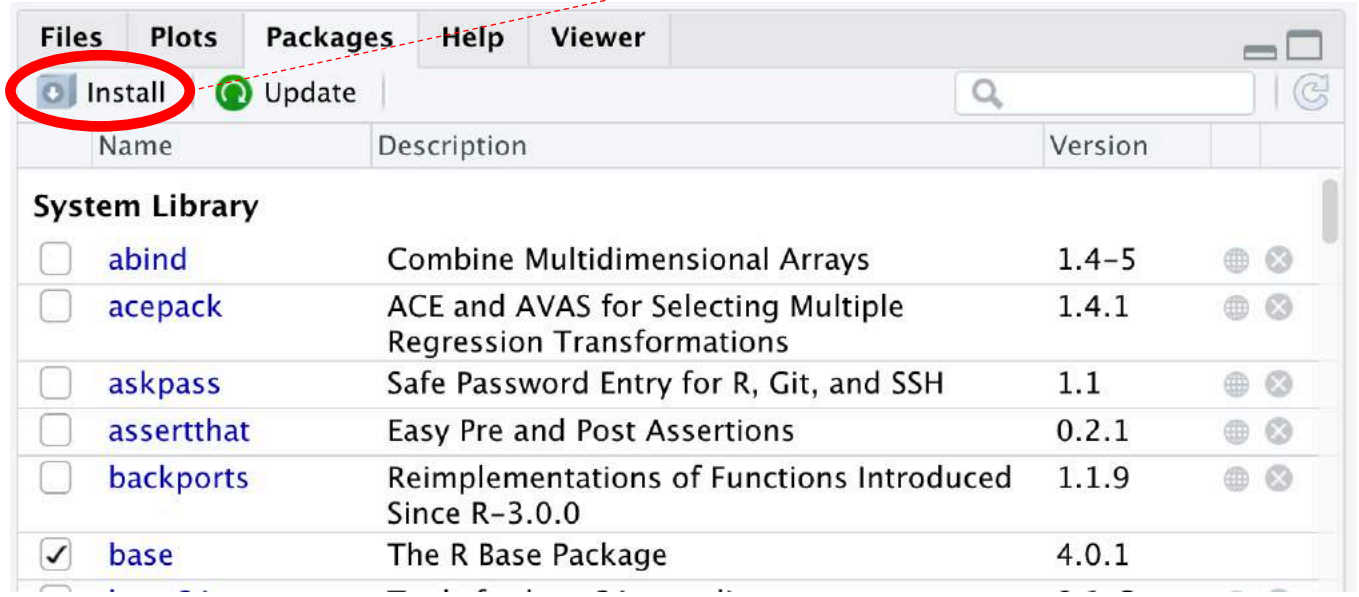
  https://cran.r-project.org/

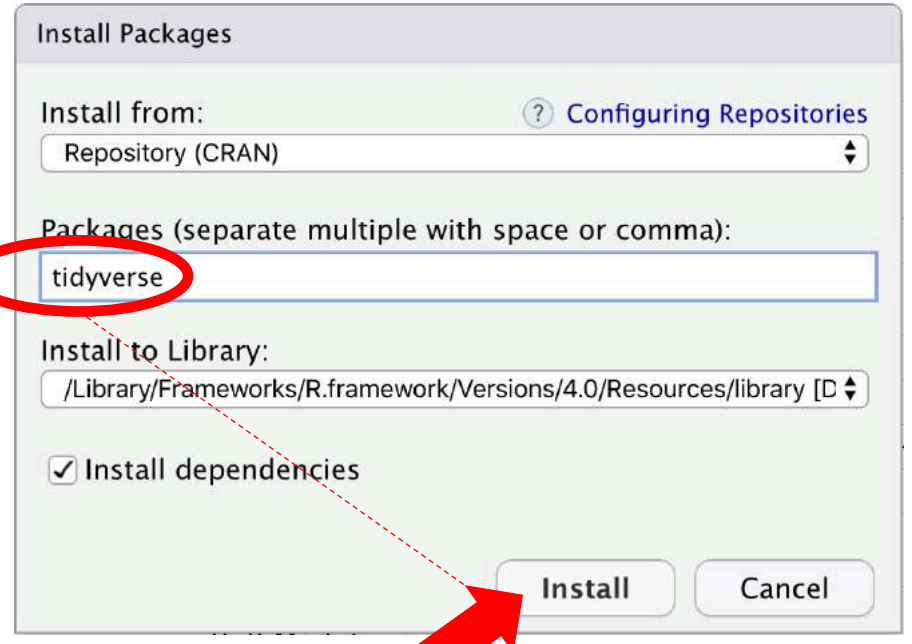


TP 0 : **Prise en main du logiciel R et de l'environnement RStudio**

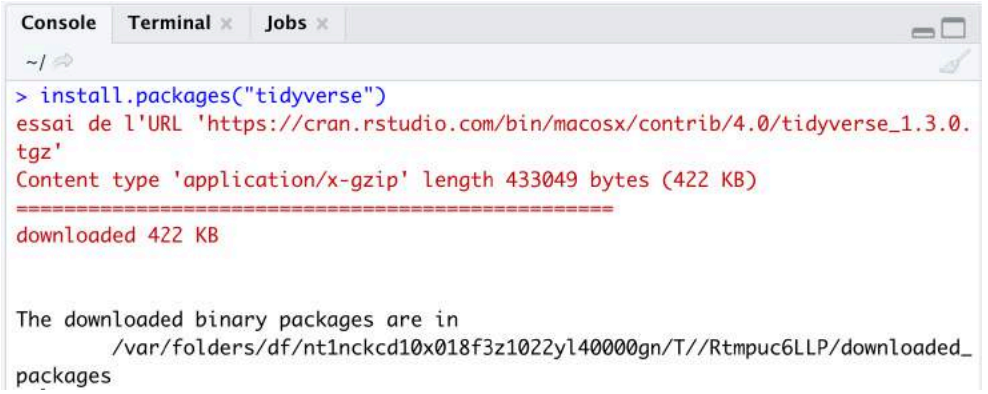
➢ **Les packages :** tidyverse, ggplot2 et les autres **Installer**

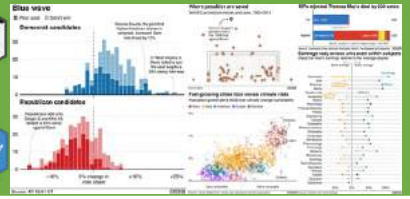
Installer une extension va télécharger l'ensemble des fichiers nécessaires depuis l'une des machines du *CRAN*, puis les installer sur le disque dur de votre ordinateur. Vous n'avez besoin de le faire qu'une fois.





Installer le package tidyverse



TP 0 : **Prise en main du logiciel R et de l'environnement RStudio**

➢ **Les packages :** tidyverse, ggplot2 et les autres **Charger**

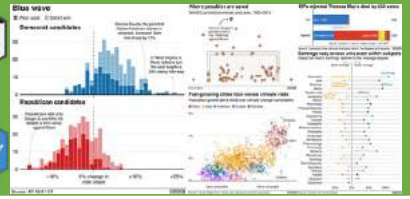
Une fois l'extension installée, il faut la "charger" avant de pouvoir utiliser les fonctions qu'elle propose. Ceci se fait avec la fonction ***library()***. Par exemple, pour pouvoir utiliser les fonctions de *tidyverse*, vous devrez exécuter la commande suivante :

```
1  library(tidyverse)
```


```
Console   Terminal ×   Jobs ×
~/
> library(tidyverse)
── Attaching packages ─────────────────────────────────── tidyverse 1.3.0 ──
✓ ggplot2 3.3.2     ✓ purrr   0.3.4
✓ tibble  3.0.3     ✓ dplyr   1.0.2
✓ tidyr   1.1.2     ✓ stringr 1.4.0
✓ readr   1.3.1     ✓ forcats 0.5.0
── Conflicts ────────────────────────────────────── tidyverse_conflicts() ──
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
Warning messages:
1: le package 'tidyverse' a été compilé avec la version R 4.0.2
2: le package 'ggplot2' a été compilé avec la version R 4.0.2
3: le package 'tibble' a été compilé avec la version R 4.0.2
4: le package 'purrr' a été compilé avec la version R 4.0.2
5: le package 'dplyr' a été compilé avec la version R 4.0.2
```

Rq: regrouper en début de script toute une série d'appels à *library()* qui permettent de charger tous les packages utilisés dans le script.

TP 0 : **Prise en main du logiciel R et de l'environnement RStudio**



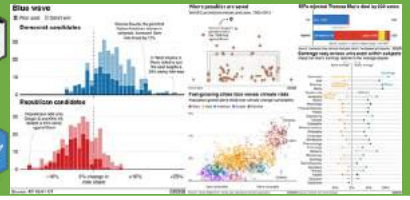
➢ **Les packages :** tidyverse, ggplot2 et les autres

Le tidyverse

- Le terme *tidyverse* est une contraction de *tidy* (« bien rangé ») et de *universe*.
- **Ce package est conçue pour la science des données.**

- Le package *tidyverse* fournit des fonctions avec une syntaxe cohérente, qui fonctionnent bien ensemble, et qui retournent des résultats prévisibles.
- Elles abordent un très grand nombre d'opérations courantes dans *R* :
  - *ggplot2* (visualisation)
  - *dplyr* (manipulation des données)
  - *tidyr* (remise en forme des données)
  - *purrr* (programmation)
  - *readr* (importation de données)
  - *tibble* (tableaux de données)
  - *forcats* (variables qualitatives)
  - *stringr* (chaînes de caractères)

https://www.tidyverse.org/packages/



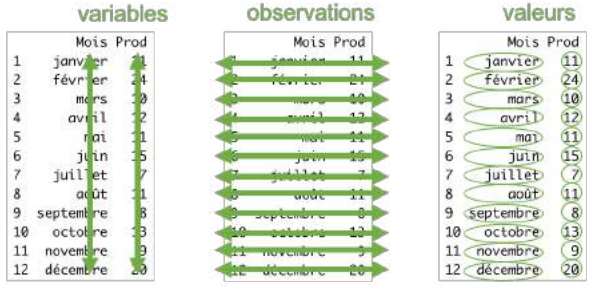
TP 0 : **Prise en main du logiciel R et de l'environnement RStudio**

➢ **Les packages :** tidyverse, ggplot2 et les autres

Le tidyverse

- Le *tidyverse* est en partie fondé sur le concept de ***tidy data*** (développé à l'origine par H. Wickham en 2014) https://www.jstatsoft.org/article/view/v059i10);
- C'est un modèle d'organisation des données qui vise à faciliter le travail souvent long et fastidieux de nettoyage et de préparation préalable à la mise en œuvre de méthodes d'analyse;
- Les extensions du *tidyverse* sont prévues pour fonctionner avec des données *tidy;*

- Les principes d'un jeu de données *tidy* sont les suivants :
  - chaque variable est une colonne
  - chaque observation est une ligne
  - chaque valeur est dans une cellule différente



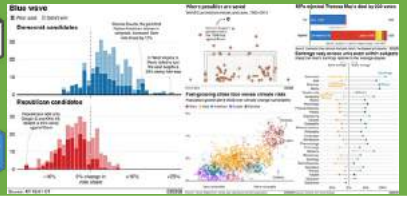
- Les extensions du *tidyverse* travaillent avec des tableaux de données au format ***tibble*** (qui est une évolution plus moderne du classique *data frame* du R de base) ➡ géré par l'extension *tibble*

- Pour autant, les *tibbles* restent compatibles avec les *data frames*. On peut ainsi facilement convertir un *data frame* en *tibble* avec *as_tibble()* (et inversement avec la fonction *as.data.frame())* :

```
61  ProdP1<-as_tibble(ProdP1)
62  ProdP1
```

```
Console   Terminal ×   Jobs ×
~/
> ProdP1<-as_tibble(ProdP1)
> ProdP1
# A tibble: 12 x 3
   Mois       Prod MoisNum
   <chr>     <dbl>   <dbl>
 1 janvier      11       1
 2 février      24       2
 3 mars         10       3
 4 avril        12       4
 5 mai          11       5
 6 juin         15       6
 7 juillet       7       7
 8 août         11       8
 9 septembre     8       9
10 octobre      13      10
11 novembre      9      11
12 décembre     20      12
```

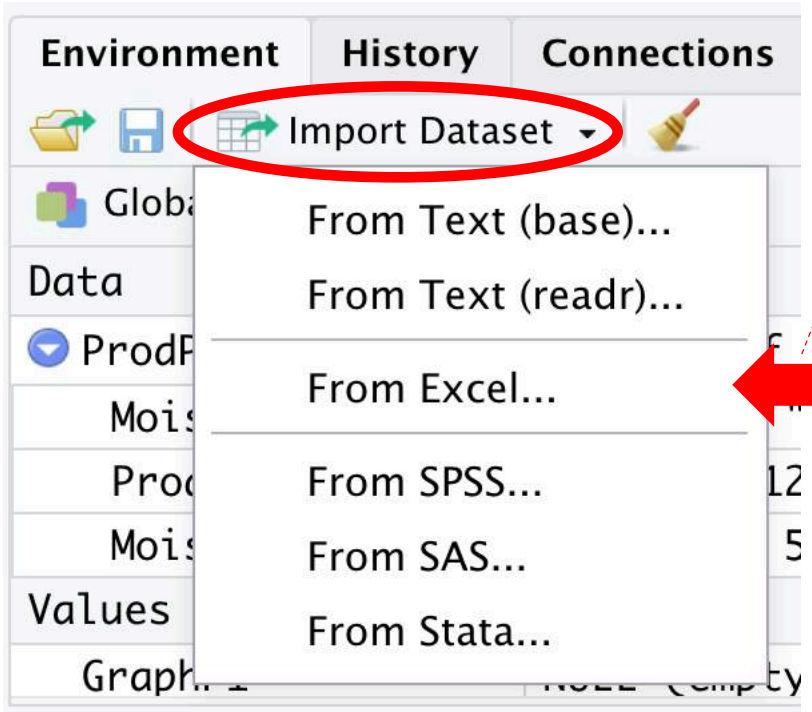
TP 0 : **Prise en main du logiciel R et de l'environnement RStudio**

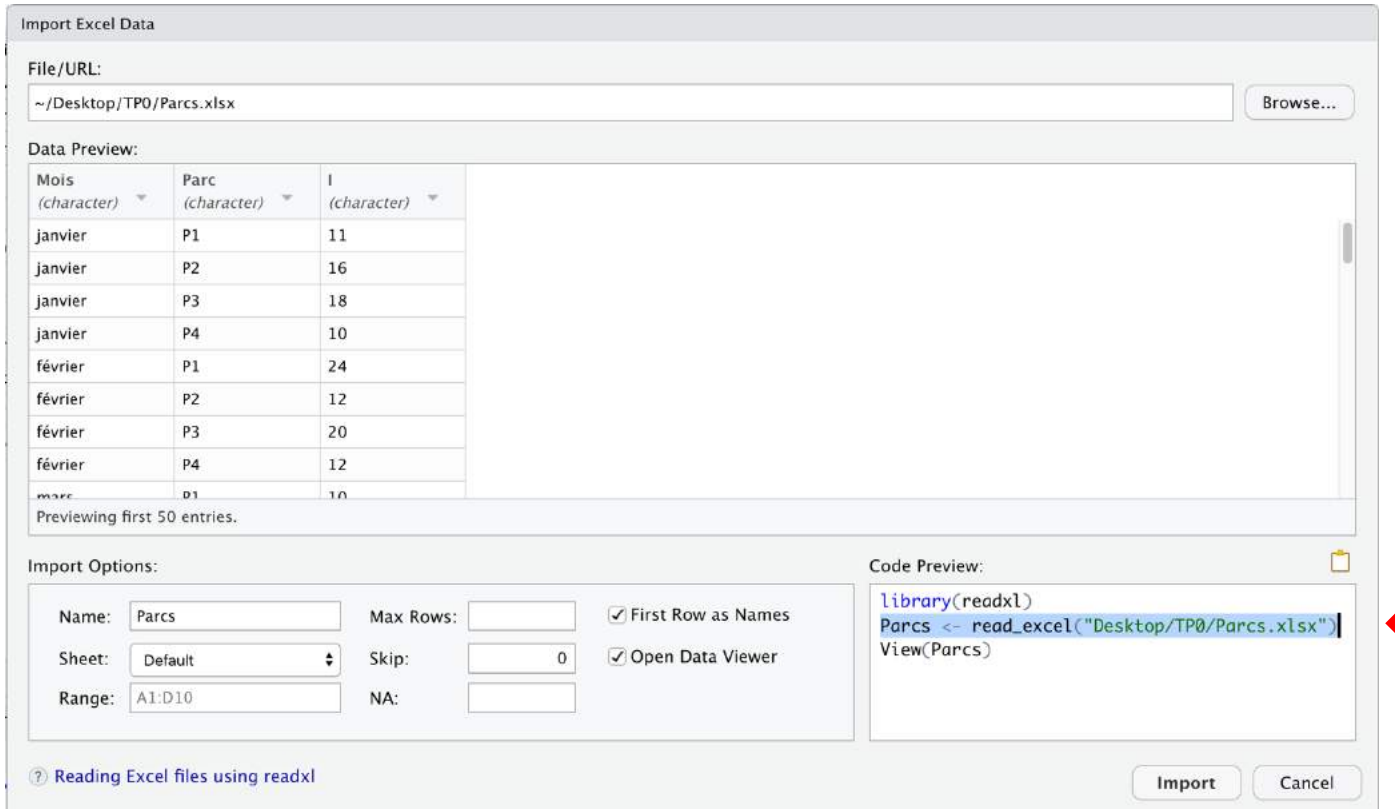


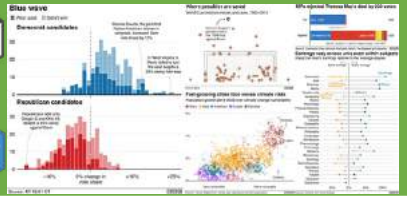
➢ **Les packages :** tidyverse, ggplot2 et les autres

Le tidyverse

- *R* n'est pas prévu pour la saisie de données, mais il bénéficie de nombreuses fonctions et packages permettant l'import de données depuis un grand nombre de formats (***CSV***, *Excel*, et bien d'autres);
- *RStudio* propose une interface permettant d'importer un fichier de données de manière interactive :







TP 0 : **Prise en main du logiciel R et de l'environnement RStudio**

➢ **Les packages :** tidyverse, ggplot2 et les autres

Le tidyverse

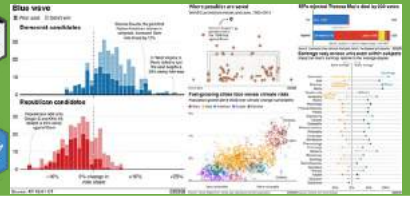
ParcsTP0.R × Parcs ×

Source on Save

```
1  ## Chargement des extensions ------------------------
2  library(tidyverse)
3  library(readxl)
4
5  ## Chargement des données ----------------------------
6  Parcs <- read_excel("Desktop/TP0/Parcs.xlsx")
```

Remarques :

- On peut exporter un tableau de données dans *R* vers un fichier dans différents formats;
- Par exemple les fonctions *write_csv()* et *write_csv2()* permettent d'enregistrer un *data frame* ou un tibble dans un fichier au format texte délimité par des virgules (*csv*) ou par des point virgules (*csv2*);
- Il n'existe pas de fonctions permettant d'enregistrer directement au format *xls* ou *xlsx*. On passe alors par un fichier *CSV*.
- Si vous travaillez sur des données de grandes dimensions, les formats texte peuvent être lents à exporter et importer. Dans ce cas, l'extension ***feather*** peut être utile : elle permet d'enregistrer un *data frame* au format *feather*, qui n'est pas le plus compact mais qui est extrêmement rapide à lire et écrire (https://github.com/wesm/feather)



TP 0 : **Prise en main du logiciel R et de l'environnement RStudio**

➢ **Les packages :** tidyverse, ggplot2 et les autres

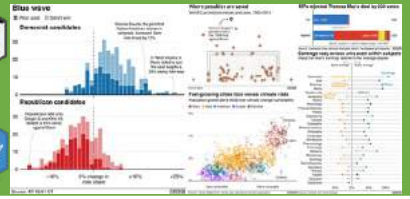
Le tidyverse

- Il peut être nécessaire de **recoder** les variables. Par exemple, avec la fonction *readxl()* les données d'un fichier Excel sont importées au format *character;*
- On peut convertir un vecteur d'un type en un autre en utilisant les fonctions *as.numeric()*, *as.character()* ou *as.logical()*. Les valeurs qui n'ont pas pu être converties sont automatiquement transformées en *NA (valeur manquante)*;
- On peut également créer un facteur (variable catégorielle) à partir d'une variable textuelle avec la fonction *factor().* Les facteurs prennent leurs valeurs dans un ensemble de modalités prédéfinies et ne peuvent en prendre d'autres. La liste des valeurs possibles est donnée par la fonction *levels().*

```
 8 ## Recodage des données ------------------------------
 9 Parcs <- as_tibble(Parcs)
10 Parcs$I <- as.numeric(Parcs$I)
11 Parcs$Parc <-factor(Parcs$Parc)
```

```
> Parcs$I <- as.numeric(Parcs$I)
Warning message:
NAs introduits lors de la conversion automatique
> Parcs$Parc <-factor(Parcs$Parc)
> levels(Parcs$Parc)
[1] "P1" "P2" "P3" "P4"
```

Nous reviendrons sur les valeurs manquantes (***NA***) et leur gestion



TP 0 : **Prise en main du logiciel R et de l'environnement RStudio**

➢ Les packages : tidyverse, ggplot2 et les autres

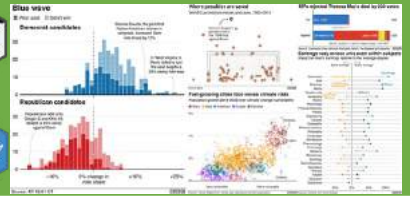
Le tidyverse

- L'extension ***dplyr*** du *tidyverse* facilite le traitement et la manipulation de données;
- Elle propose une syntaxe claire et cohérente, sous formes de **verbes**, pour la plupart des opérations sur les tables de données;
- *dplyr* part du principe que les données sont organisées selon le modèle des *tidy data.*

Les verbes d'actions :

- ***slice()*** : pour sélectionner des lignes
- ***select()*** : pour sélectionner des colonnes par leur nom ou leur position
- ***filter()*** : pour filtrer des lignes en fonction de conditions sur les colonnes
- ***rename()*** : pour renommer des colonnes
- ***arrange()*** : pour ordonner une table en fonction d'une variable;
- ***mutate()*** : pour modifier ou créer colonnes en fonction des autres;
- ***summarize()*** : pour résumer une table en la réduisant à une unique ligne (à ne pas confondre avec *summary*)

Il est très fréquent d'enchaîner plusieurs verbes d'action. Ceux-ci peuvent s'utiliser les uns après les autres, en utilisant le symbole **%>%** (***pipe*** en anglais), que l'on pourrait traduire par « et puis ».



TP 0 : **Prise en main du logiciel R et de l'environnement RStudio**

➢ **Les packages :** tidyverse, ggplot2 et les autres

Le tidyverse

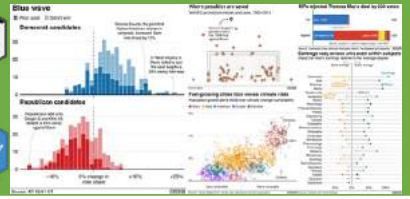
```
13 ## Exemples des verbes d'actions ---------------------
14 # Les données du parcs P1 rangées dans le tibble P1
15 P1<-Parcs %>%
16   filter(Parc=="P1")
```

```
> P1
# A tibble: 12 x 3
   Mois      Parc      I
   <chr>     <fct> <dbl>
 1 janvier   P1       11
 2 février   P1       24
 3 mars      P1       10
 4 avril     P1       12
 5 mai       P1       11
 6 juin      P1       15
 7 juillet   P1        7
 8 août      P1       11
 9 septembre P1        8
10 octobre   P1       13
11 novembre  P1        9
12 décembre  P1       20
```

```
18 # Ajout d'une colonne pour la puissance produite (tension de 20kV)
19 Parcs <- Parcs %>%
20   mutate(Puissance=I*20000)
```

```
                                [*20000)

                                [ Puissance
   <chr>   <fct> <dbl>      <dbl>
 1 janvier P1       11     220000
 2 janvier P2       16     320000
 3 janvier P3       18     360000
 4 janvier P4       10     200000
 5 février P1       24     480000
 6 février P2       12     240000
 7 février P3       20     400000
 8 février P4       12     240000
 9 mars    P1       10     200000
10 mars    P2       20     400000
# … with 38 more rows
```



TP 0 : **Prise en main du logiciel R et de l'environnement RStudio**

➢ **Les packages :** tidyverse, ggplot2 et les autres

Le tidyverse

- Les opérateurs de comparaison :

Opérateur de comparaison	Signification
==	égal à
!=	différent de
>	strictement supérieur à
<	strictement inférieur à
>=	supérieur ou égal à
<=	inférieur ou égal à

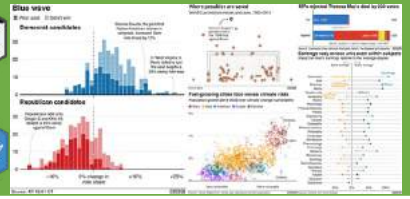
- A combiner avec les opérateurs logiques :

Opérateur logique	Signification
&	et logique
|	ou logique
!	négation logique

```
22  # A déterminer
23  Parcs %>%
24    filter(Mois=="janvier" | Mois=="février") %>%
25    filter(Puissance>220000 & Puissance<=400000) %>%
26    select(-I) %>%
27    arrange(desc(Puissance))
```

?

```
> Parcs %>%
+   filter(Mois=="janvier" | Mois=="février") %>%
+   filter(Puissance>220000 & Puissance<=400000) %>%
+   select(-I) %>%
+   arrange(desc(Puissance))
# A tibble: 5 x 3
  Mois    Parc  Puissance
  <chr>   <fct>     <dbl>
1 février P3       400000
2 janvier P3       360000
3 janvier P2       320000
4 février P2       240000
5 février P4       240000
```



TP 0 : **Prise en main du logiciel R et de l'environnement RStudio**

➢ **Les packages :** tidyverse, ggplot2 et les autres

Le tidyverse

- La puissance de ces commandes est décuplée par la possibilité de faire des traitements par groupe à l'aide de la fonction ***group_by()***
- La fonction *groupe_by()* permet de définir des groupes de lignes à partir des valeurs d'une ou plusieurs colonnes.

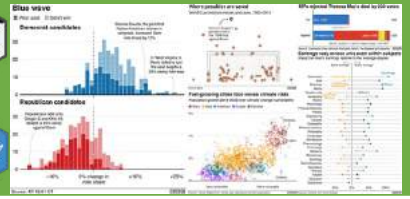
```
29  # Grouper par mois et sélectionner la puissance la plus grande
30  Parcs %>%
31    group_by(Mois) %>%
32    filter(Puissance==max(Puissance,na.rm=TRUE))
```

```
> Parcs %>%
+   group_by(Mois) %>%
+   filter(Puissance==max(Puissance,na.rm=TRUE))
# A tibble: 13 x 4
# Groups:   Mois [12]
   Mois      Parc      I Puissance
   <chr>     <fct> <dbl>     <dbl>
 1 janvier   P3       18    360000
 2 février   P1       24    480000
 3 mars      P2       20    400000
 4 avril     P3       21    420000
 5 mai       P3       23    460000
 6 juin      P2       23    460000
 7 juillet   P3       16    320000
 8 août      P3       20    400000
 9 septembre P3       19    380000
10 octobre   P2       19    380000
11 novembre  P3       18    360000
12 décembre  P1       20    400000
13 décembre  P3       20    400000
```

```
34  #
35  Parcs %>%
36    group_by(Mois) %>%
37    summarise(Pmax=max(Puissance,na.rm=TRUE),
38              Pmin=min(Puissance,na.rm=TRUE),
39              Pmoy=mean(Puissance,na.rm=TRUE),
40              Pmed=median(Puissance,na.rm=TRUE))
```

```
> Parcs %>%
+   group_by(Mois) %>%
+   summarise(Pmax=max(Puissance,na.rm=TRUE),
+             Pmin=min(Puissance,na.rm=TRUE),
+             Pmoy=mean(Puissance,na.rm=TRUE),
+             Pmed=median(Puissance,na.rm=TRUE))
`summarise()` ungrouping output (override with `.groups` argument)
# A tibble: 12 x 5
   Mois        Pmax   Pmin    Pmoy   Pmed
   <chr>      <dbl>  <dbl>   <dbl>  <dbl>
 1 août      400000 140000 270000  270000
 2 avril     420000 160000 290000  290000
 3 décembre  400000 280000 360000  380000
 4 février   480000 240000 340000  320000
 5 janvier   360000 200000 275000  270000
 6 juillet   320000 140000 225000  220000
 7 juin      460000 280000 355000  340000
 8 mai       460000 200000 295000  260000
 9 mars      400000 180000 285000  280000
10 novembre  360000 180000 246667. 200000
11 octobre   380000 240000 305000  300000
12 septembre 380000 160000 293333. 340000
```

Quand la variable contient des valeurs manquante, il faut préciser à *R* de les supprimer en ajoutant l'option *na.rm=TRUE*



TP 0 : **Prise en main du logiciel R et de l'environnement RStudio**

➢ **Les packages :** tidyverse, ggplot2 et les autres

Le tidyverse

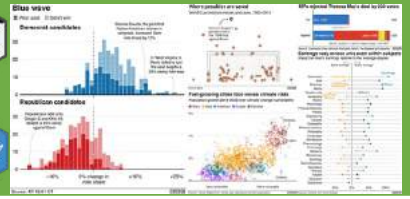
- On peut grouper selon plusieurs variables à la fois : *group_by(Parc,Mois)*

- On peut à tout moment "dégrouper" un tableau à l'aide de ***ungroup()*** :

```
42  #
43  Parcs %>%
44    group_by(Parc) %>%
45    summarise(Psum=sum(Puissance, na.rm=TRUE)) %>%
46    ungroup() %>%
47    mutate(Poucentage = Psum/sum(Psum)  * 100)
```

```
> Parcs %>%
+   group_by(Parc) %>%
+   summarise(Psum=sum(Puissance, na.rm=TRUE)) %>%
+   ungroup() %>%
+   mutate(Poucentage = Psum/sum(Psum)  * 100)
`summarise()` ungrouping output (override with `.groups` argument)
# A tibble: 4 x 3
  Parc      Psum Poucentage
  <fct>    <dbl>      <dbl>
1 P1     3020000       22.2
2 P2     3680000       27.0
3 P3     4580000       33.6
4 P4     2340000       17.2
```

- Il y a bien d'autres fonctions utiles dans l'extension *dplyr* pour la manipulation des données.



TP 0 : **Prise en main du logiciel R et de l'environnement RStudio**

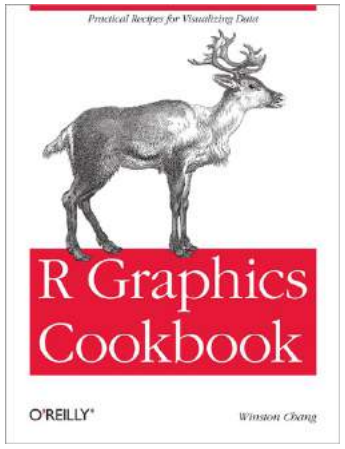
➢ **Les packages :** tidyverse, ggplot2 et les autres ggplot2

- ***ggplot2*** est une extension du *tidyverse* qui permet de générer des graphiques avec une syntaxe cohérente et puissante;
- Elle nécessite l'apprentissage d'une grammaire spécifique, mais elle permet la construction de graphiques complexes de manière efficace;
- Une des particularités de *ggplot2* est qu'elle part du principe que les données relatives à un graphique sont stockées dans un tableau de données (*data frame*, *tibble* ou autre).

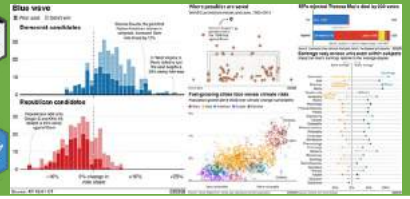
https://r-graphics.org/

www.r-graph-gallery.com

https://rstudio.com/wp-content/uploads/2015/03/ggplot2-cheatsheet.pdf



« *Cheat Sheet* » de ggplot2
ou simplement dans l'aide de RStudio



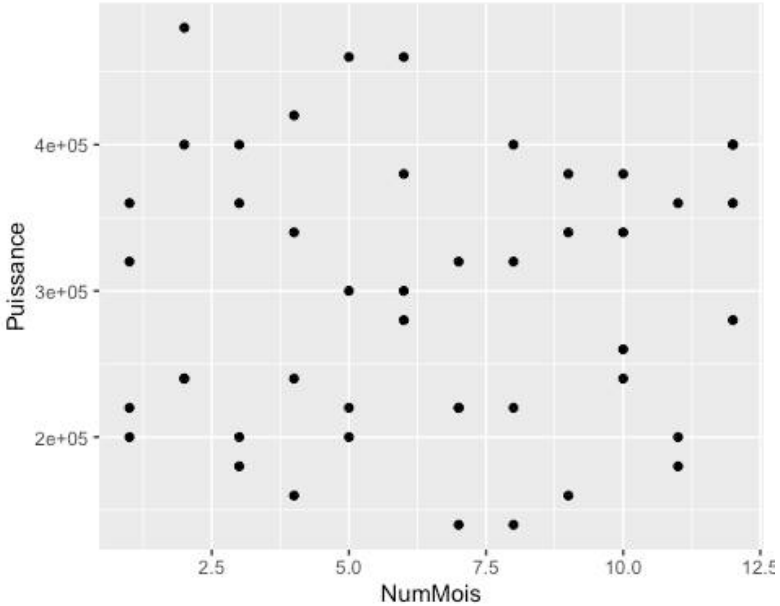
TP 0 : **Prise en main du logiciel R et de l'environnement RStudio**

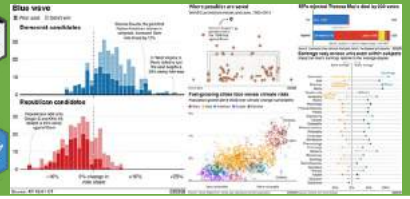
➢ **Les packages :** tidyverse, ggplot2 et les autres ggplot2

- Une représentation graphique *ggplot* se construit à partir d'un ensemble d'éléments indépendants qui constitue la grammaire de la syntaxe
- Les principaux éléments de cette grammaire sont :
  - Data (***ggplot***) : le jeu de données contenant les variables utilisées;
  - Aesthetics (***aes***) : les variables à représenter;
  - Geométrics (***geom_…***) : le type de représentation graphique souhaitée;
  - Statistics (***stat_…***) : les éventuelles transformations des données pour la représentation souhaitée;
  - Scales (***scale_…***) : permet de contrôler le lien entre les données et les aesthetics

```
49 ## Quelques graphiques avec la grammaire ggplot2
50 # Nuages de points
51 ggplot(data = Parcs)+
52   aes(x = NumMois, y = Puissance)+
53   geom_point()
```

geom_point() représente les données en nuages de points. Les aesthetics associés sont : *x, y* et également *shape*, *fill*, *color*, *size*, *group*



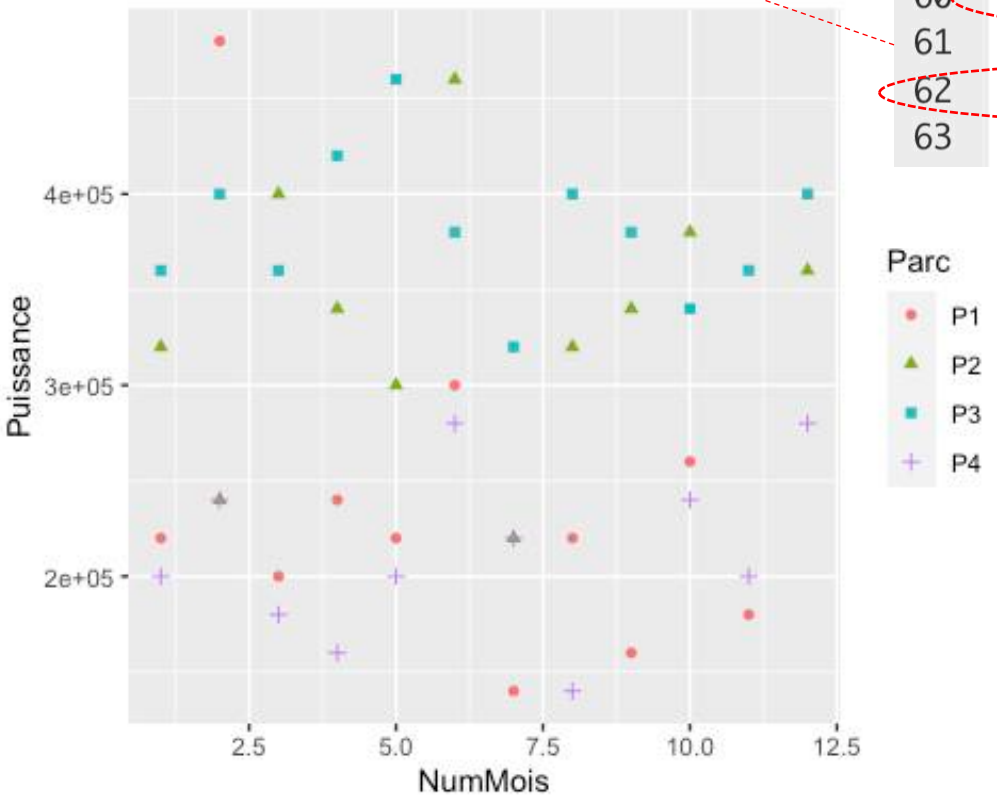


TP 0 : **Prise en main du logiciel R et de l'environnement RStudio**

➢ **Les packages :** tidyverse, ggplot2 et les autres ggplot2

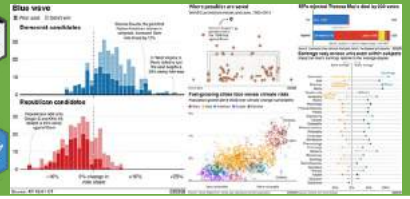
```
55  ggplot(data = Parcs)+
56    aes(x = NumMois, y = Puissance, color = Parc, shape=Parc)+
57    geom_point()
```

```
59  ggplot(data = Parcs)+
60    geom_point(aes(x = NumMois, y = Puissance,color = Parc, shape=Parc))
61
62  ggplot(data = Parcs,aes(x = NumMois, y = Puissance,color = Parc, shape=Parc))+
63    geom_point()
```



Produit le même graphique

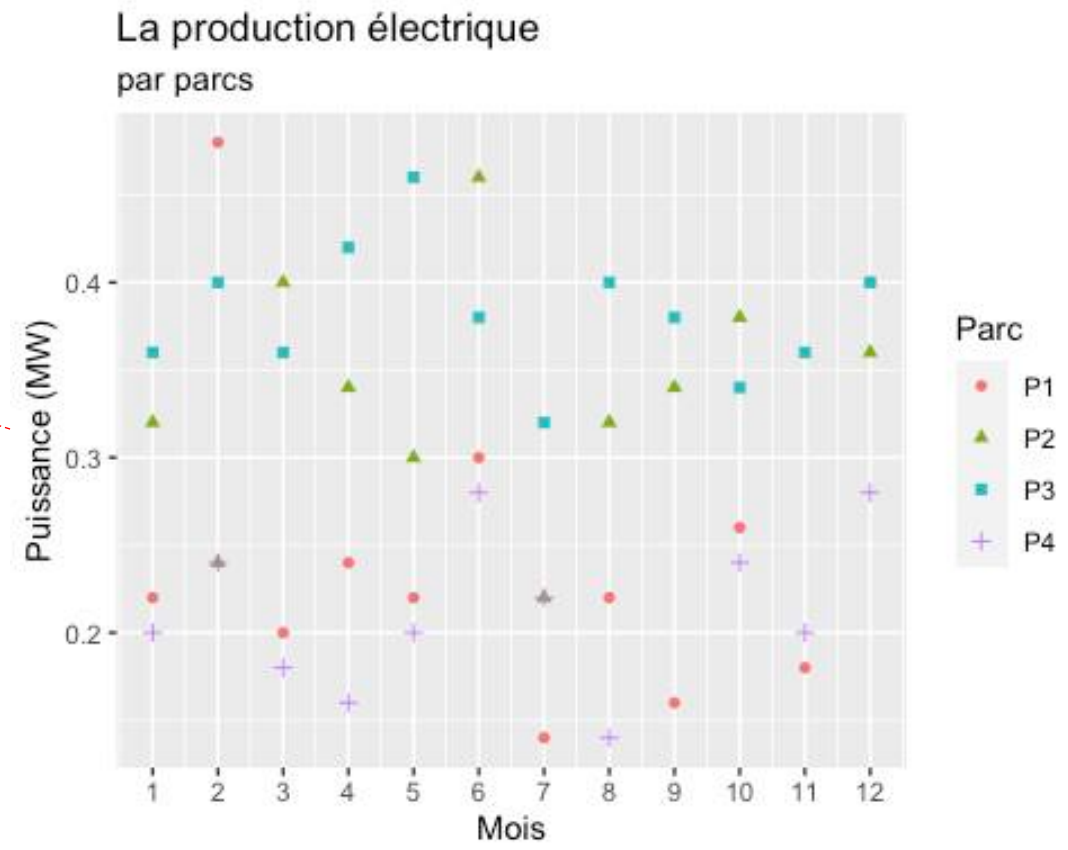
Pour simplifier un peu le code, plutôt que de déclarer les *aesthetics* dans chaque *geom*, on peut les déclarer dans l'appel à *ggplot()*. Ils seront automatiquement "hérités" par les geom ajoutés (sinon on peut redéfinir les *aesthetics* dans *geom*).

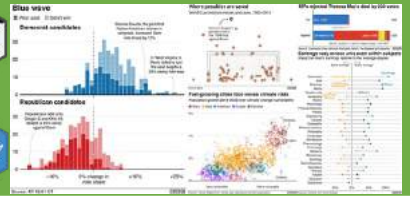


TP 0 : **Prise en main du logiciel R et de l'environnement RStudio**

➢ **Les packages :** tidyverse, ggplot2 et les autres ggplot2

```
65  Graph_Nuage <-ggplot(data = Parcs)+
66    aes(x = NumMois, y = Puissance/1E6, color = Parc, shape=Parc)+
67    geom_point()
68
69  Graph_Nuage <- Graph_Nuage+
70    ggtitle("La production électrique",subtitle = "par parcs") +
71    xlab("Mois") +
72    ylab("Puissance (MW)")+
73    scale_x_continuous(breaks=seq(1,12,by=1))
74  Graph_Nuage
```

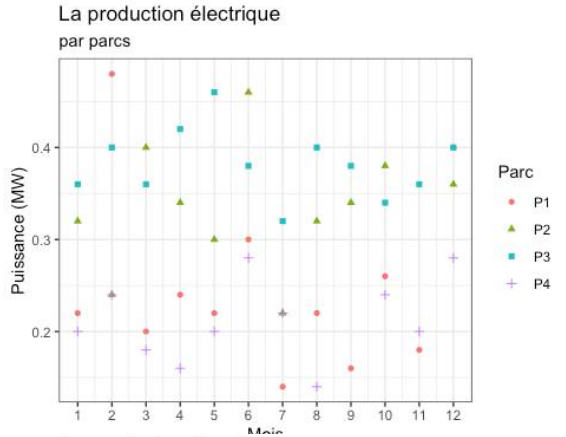


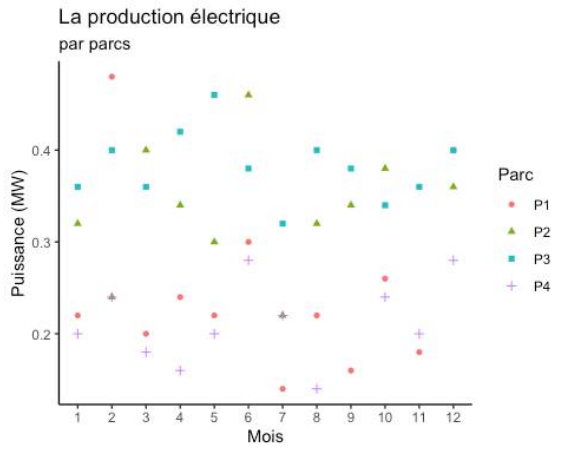


TP 0 : **Prise en main du logiciel R et de l'environnement RStudio**

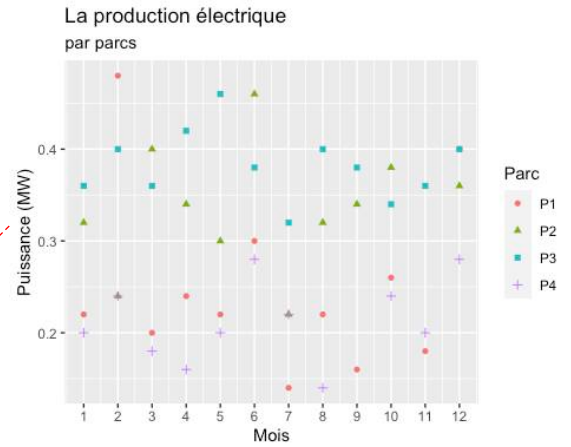
➢ **Les packages :** tidyverse, ggplot2 et les autres ggplot2

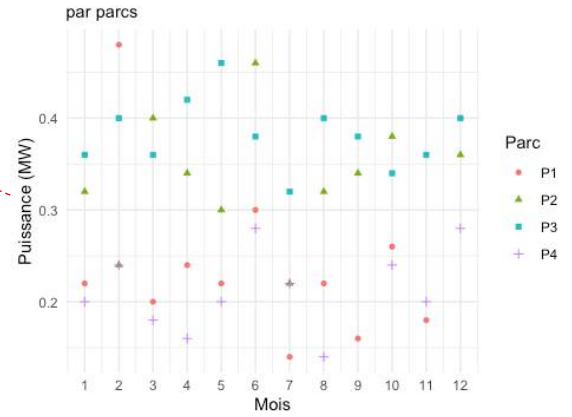
- Les thèmes permettent de contrôler l'affichage de tous les éléments du graphique qui ne sont pas reliés aux données : titres, grilles, fonds, etc.
- Il existe un certain nombre de thèmes préexistants, par exemple :



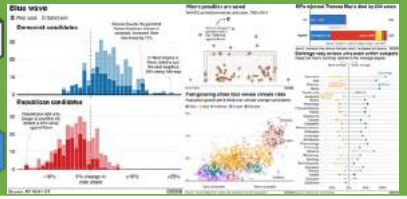



```
76  Graph_Nuage+theme_bw()
77  Graph_Nuage+theme_classic()
78  Graph_Nuage+theme_grey() #theme par défaut
79  Graph_Nuage+theme_minimal()
```



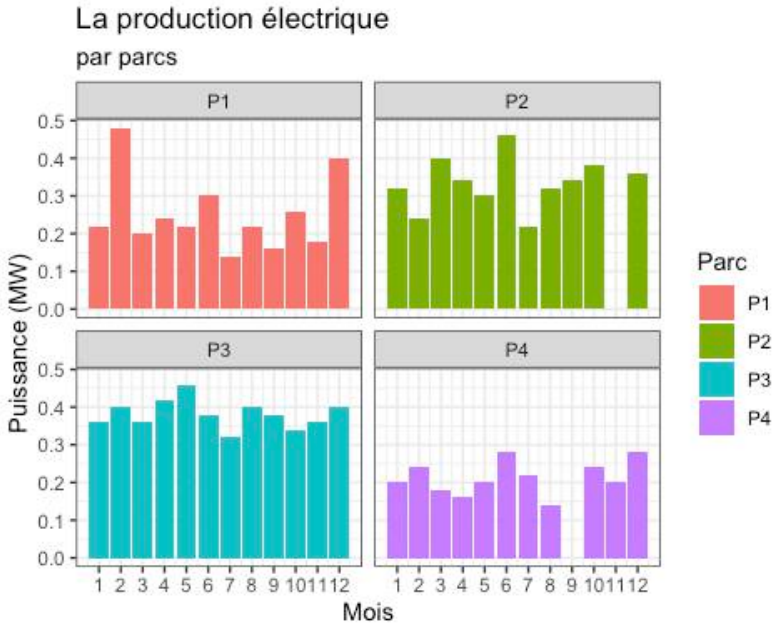


Néanmoins le graphique en nuages de points n'est pas explicite et pertinent avec ces données



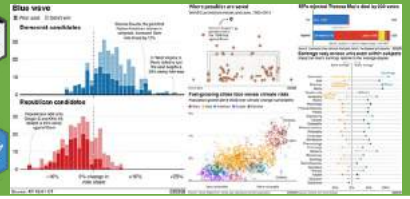
TP 0 : **Prise en main du logiciel R et de l'environnement RStudio**

➢ **Les packages :** tidyverse, ggplot2 et les autres ggplot2




```
81  # Graphique en bâton
82  ggplot(data = Parcs,aes(x = NumMois, y = Puissance/1E6, fill=Parc))+
83    geom_bar(stat="identity")+
84    facet_wrap(~ Parc)+
85    ggtitle("La production électrique",subtitle = "par parcs") +
86    xlab("Mois") +
87    ylab("Puissance (MW)")+
88    scale_x_continuous(breaks=seq(1,12,by=1))+
89    theme_bw()
```

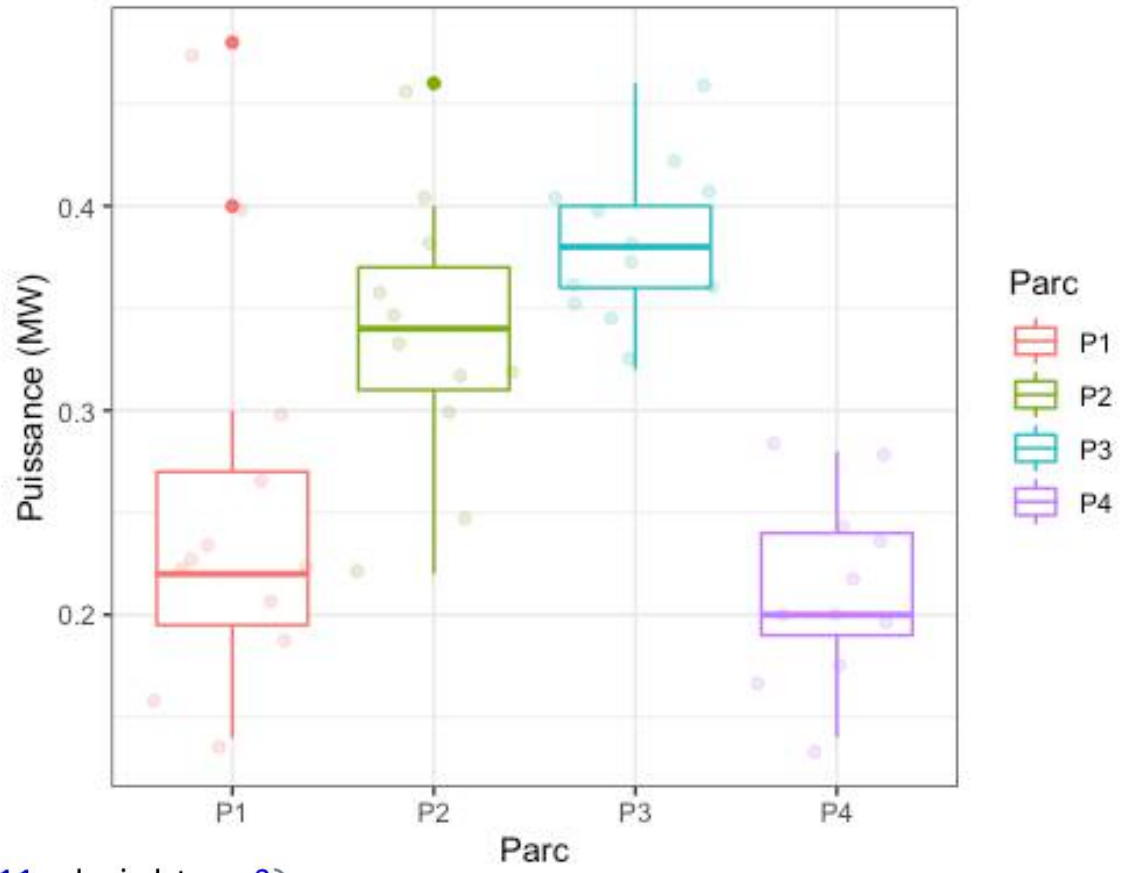
- Le *faceting* permet d'effectuer plusieurs fois le même graphique selon les valeurs d'une ou plusieurs variables qualitatives.
- C'est ce que permettent les fonctions *facet_wrap()* et *facet_grid();*
- Les deux fonctions prennent en paramètre une formule de la forme ***~ variable***, où variable est le nom de la variable en fonction de laquelle on souhaite faire les différents graphiques;
- Avec *facet_wrap()*, les différents graphiques sont affichés les uns à côté des autres et répartis automatiquement dans la page;
- Pour *facet_grid()*, les graphiques sont disposés selon une grille. La formule est alors de la forme variable en ligne ~ variable en colonne. Si on n'a pas de variable dans l'une des deux dimensions, on met un point (.) `94 facet_grid(. ~ Parc)+`



TP 0 : **Prise en main du logiciel R et de l'environnement RStudio**

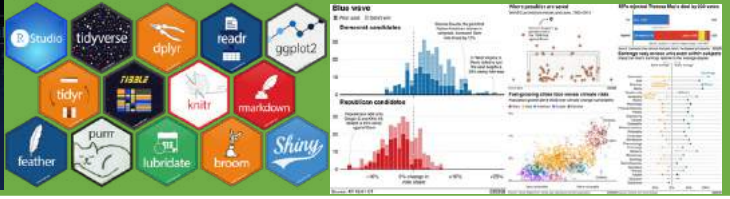
➢ **Les packages :** tidyverse, ggplot2 et les autres ggplot2

```
102  # Boites à moustaches
103  Graph_moustaches <- ggplot(data = Parcs,aes(x = Parc, y = Puissance/1E6,
104                                              color=Parc))+
105    geom_boxplot()+
106    geom_jitter(alpha = 0.2,show.legend=FALSE)+
107    theme(legend.position="bottom")+
108    theme_bw()+
109    ylab("Puissance (MW)")
110  Graph_moustaches
```



- Exporter de façon programmatique :

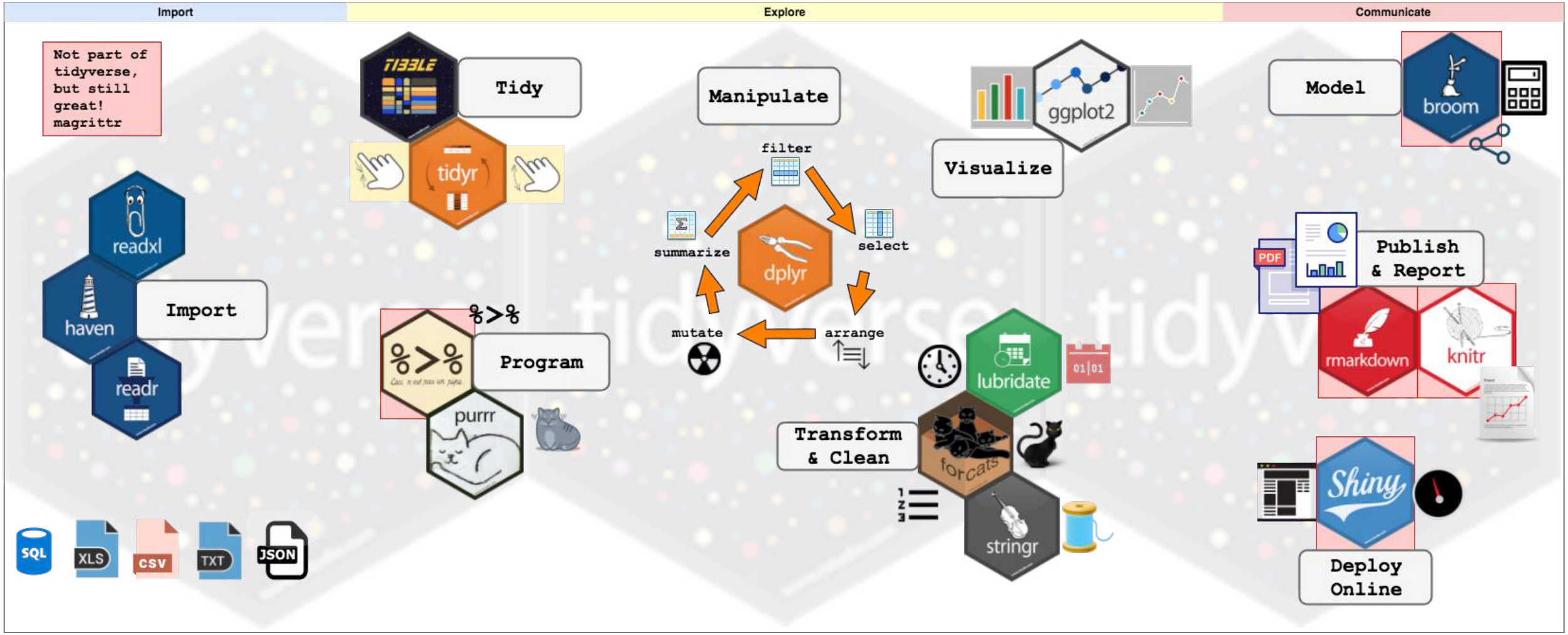
```
112  ggsave("Graph_moustaches.pdf",plot = Graph_moustaches,width = 11, height = 8)
```

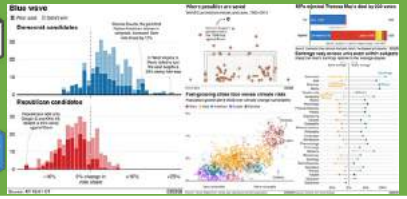


TP 0 : **Prise en main du logiciel R et de l'environnement RStudio**

➢ **Les packages :** tidyverse, ggplot2 et les autres

Et les autres



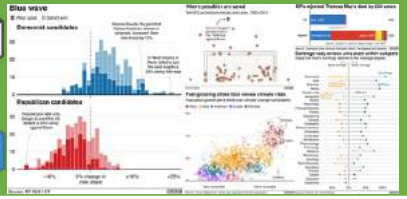


TP 0 : **Prise en main du logiciel R et de l'environnement RStudio**

➢ Création d'un document Rmarkdown

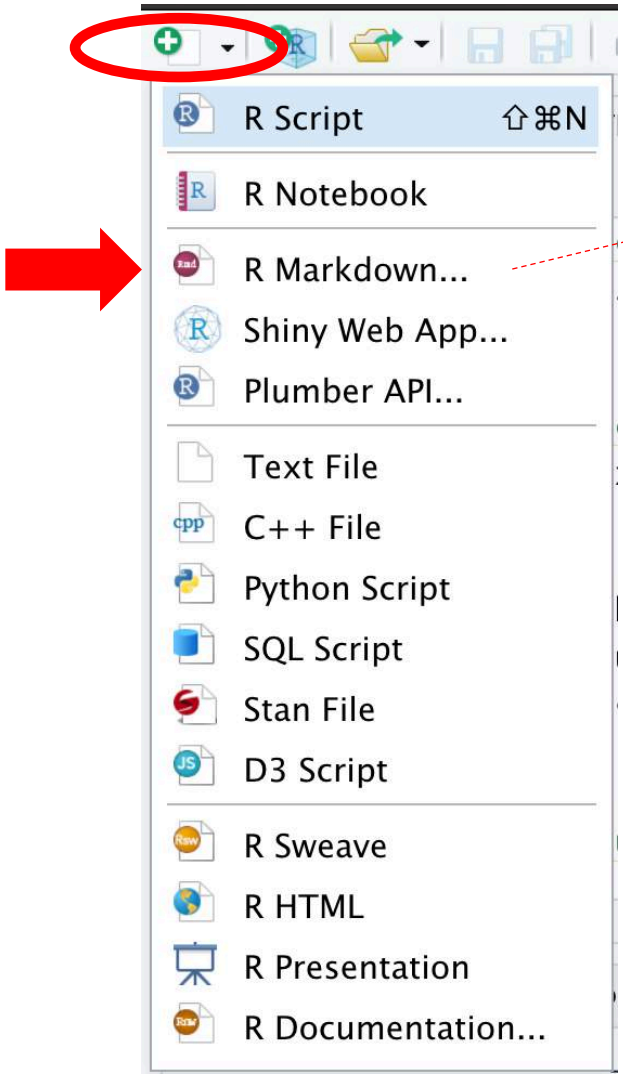
- Produire facilement des rapports automatisés dans divers format (**Word**, **PDF**, **HTML**, …)
- C'est donc un outil très pratique pour l'exportation, la communication et la diffusion de résultats d'analyse.

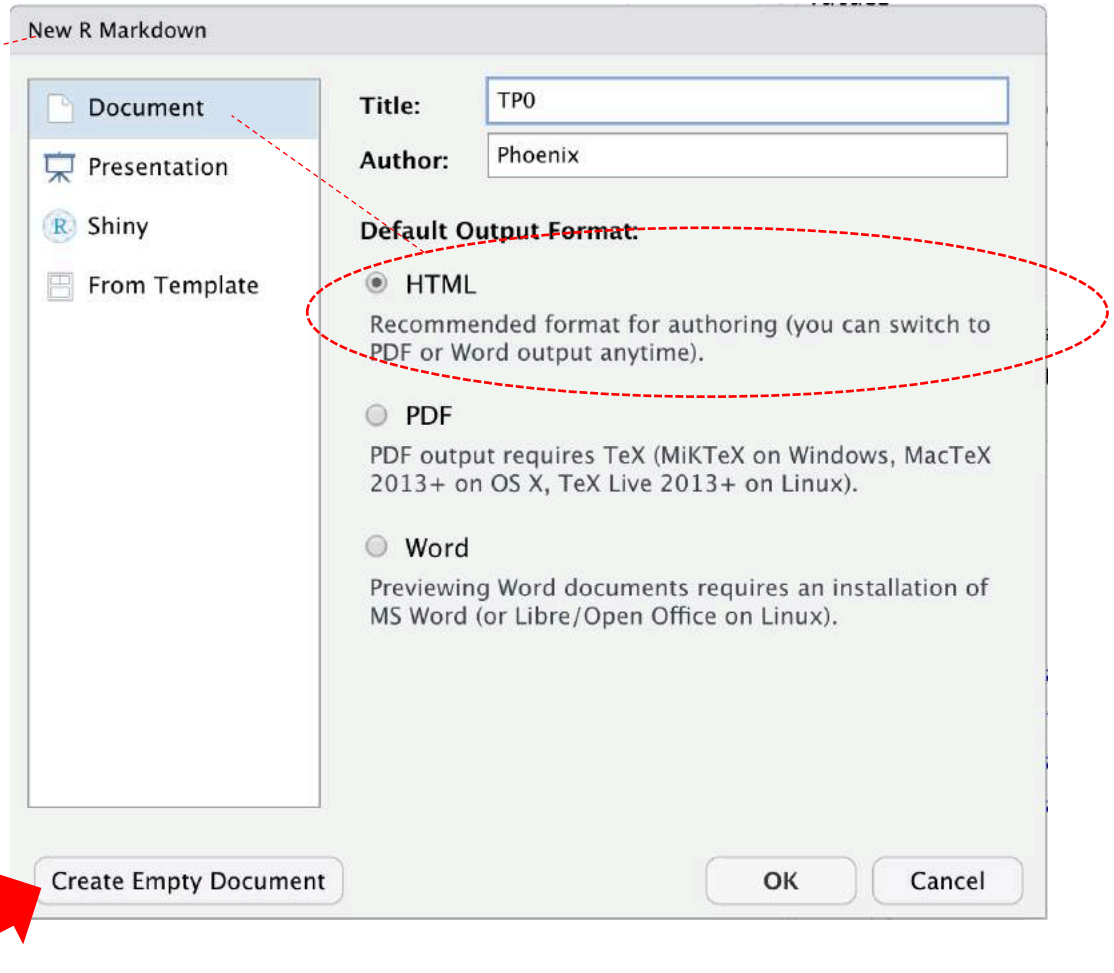
Rmarkdown ne fait pas partie du *tidyverse*, mais elle est installée et chargée par défaut par *RStudio*

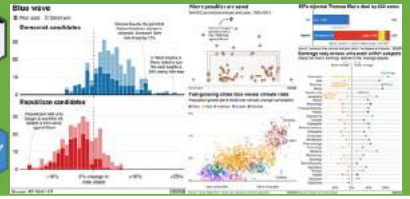


TP 0 : **Prise en main du logiciel R et de l'environnement RStudio**

➢ Création d'un document Rmarkdown





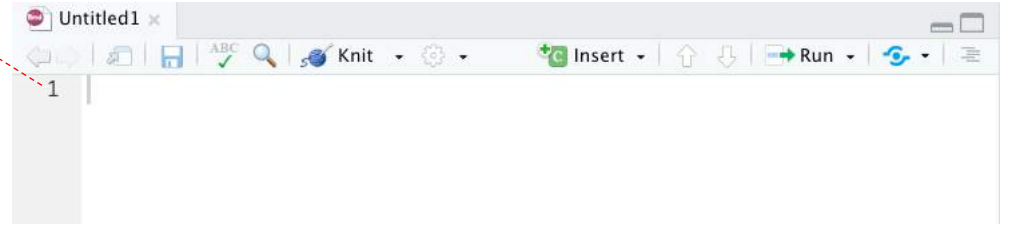


TP 0 : **Prise en main du logiciel R et de l'environnement RStudio**

➢ Création d'un document Rmarkdown

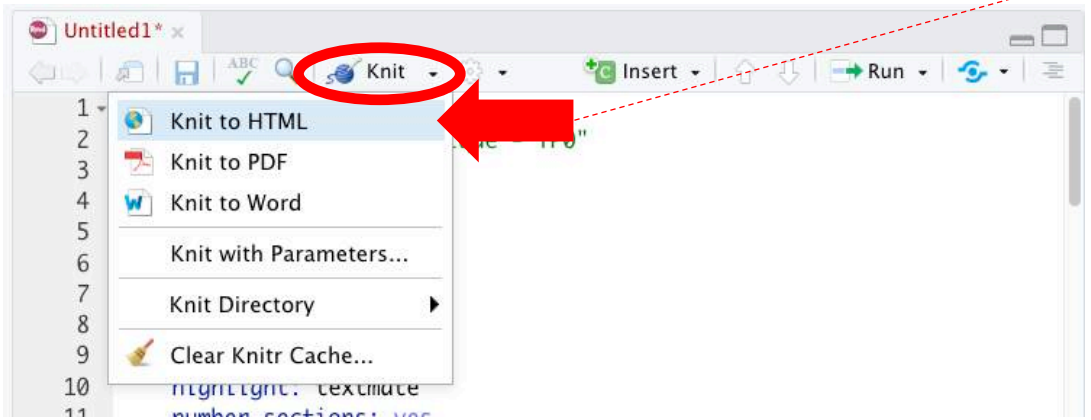
Copier le programme *TP0-HTML.txt*

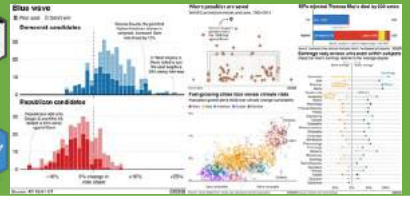
et coller le



Compiler le document au format HTML

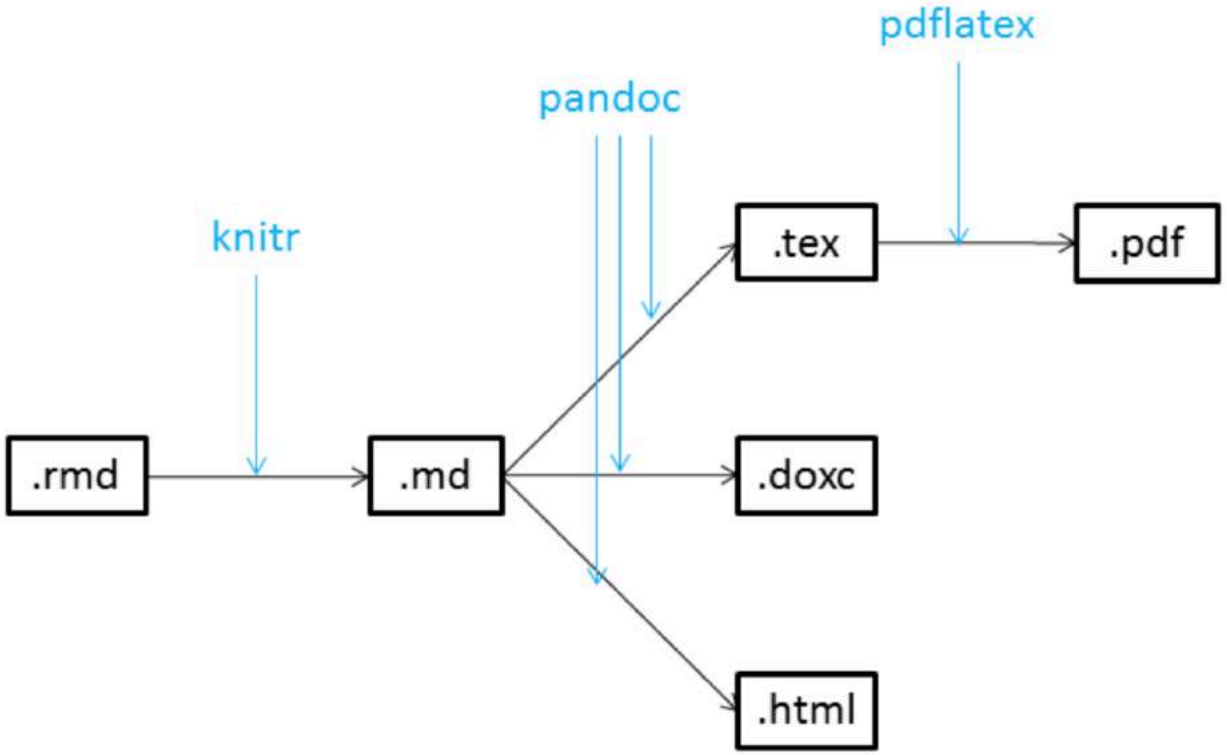
(Penser à enregistrer dans le répertoire *TP0-HTML*)





TP 0 : **Prise en main du logiciel R et de l'environnement RStudio**

➢ Création d'un document Rmarkdown



1. Chaîne de traitement
    1. package/fonction *rmarkdown*: gestion de l'ensemble de la chaîne de traitement (Rstudio -> document final)
    2. package *knitr*: interprétation du code utilisateur (R/Python/Julia) et conversion en sortie intégrable au document
    3. logiciel *Pandoc*: convertisseur de markdown (faiblement structuré) en langage balisé (fortement structuré)
2. Interpréteur de langage balisé
    1. navigateur web (*HTML*)
    2. commande (pdf)latex + lecteur (*PDF*)
    3. Word/libreoffice (*docx*)

R Markdown Reference Guide

Learn more about R Markdown at rmarkdown.rstudio.com

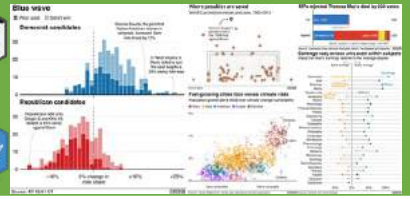
Learn more about Interactive Docs at shiny.rstudio.com/articles

Contents:

1. **Markdown Syntax**
2. Knitr chunk options
3. Pandoc options

https://rmarkdown.rstudio.com/formats.html

https://rstudio.com/wp-content/uploads/2015/03/rmarkdown-reference.pdf



TP 0 : **Prise en main du logiciel R et de l'environnement RStudio**

➢ Création d'un document Rmarkdown

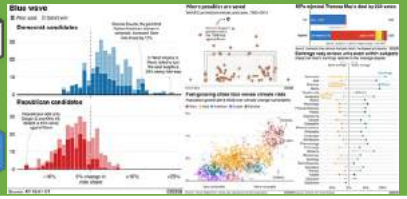
- La première partie du document est son ***en-tête***. Il se situe en tout début de document, et est délimité par trois tirets (---) avant et après;
- Cet en-tête se nomme *YAML* (*Yet Another Markup Language*);
- Il contient les métadonnées du document, comme son titre, son auteur, sa date, plus tout un tas d'options possibles qui vont permettre de configurer ou personnaliser l'ensemble du document et son rendu.

```yaml
---
title: "Production électrique"
subtitle: "TP0"
author: "Phoenix"
date: "`r Sys.Date()`"
abstract:
  Ce document est un exemple de rapport automatisé avec une sortie au format html.
output:
  html_document:

    fig_align: center
    fig_caption: yes
    highlight: textmate
    number_sections: yes
    self_contained: yes
    theme: yeti
    toc: yes
    toc_float:
      collapsed: yes
      smooth_scroll: yes
---
```

Indique que le document généré doit être au format *HTML*

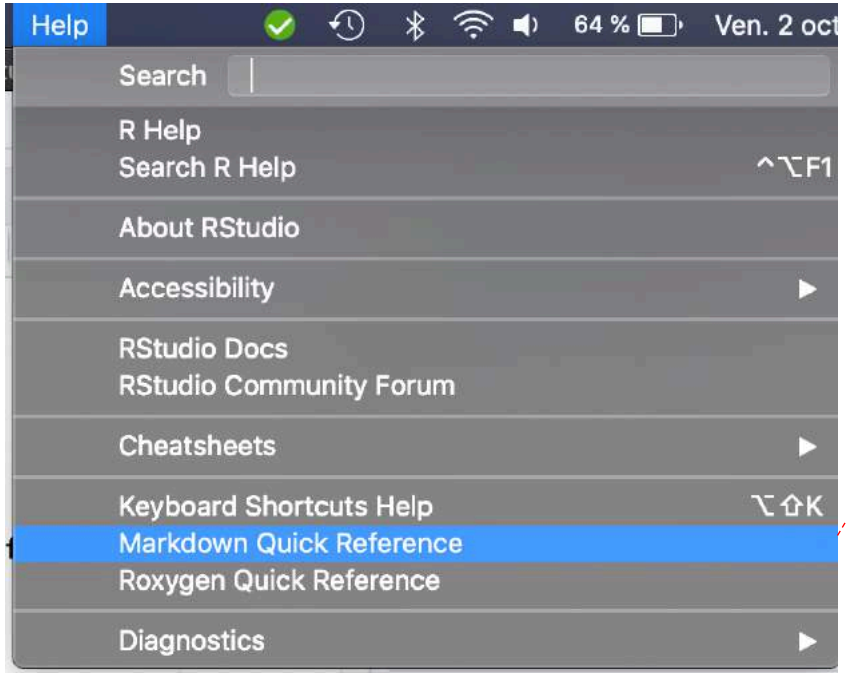
Table des matières flottante

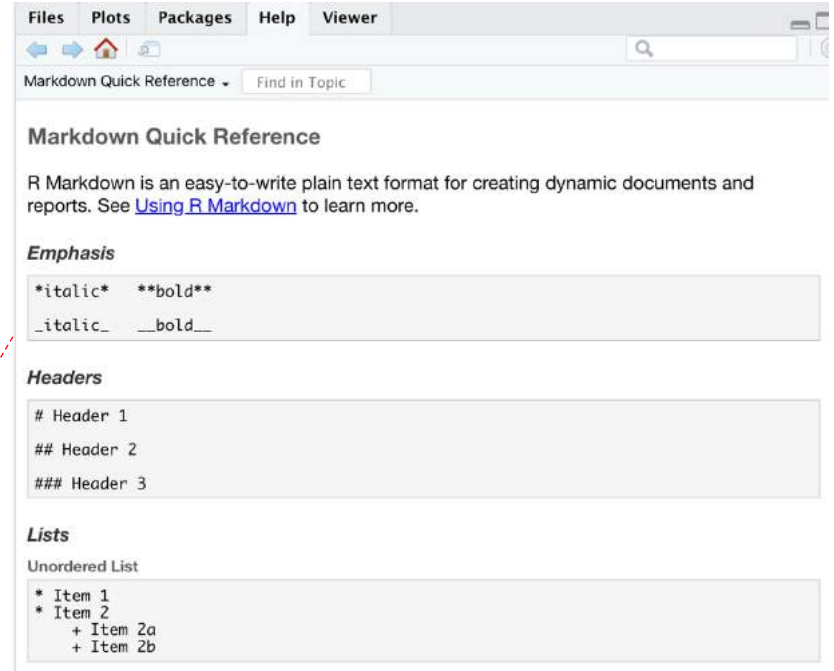


TP 0 : **Prise en main du logiciel R et de l'environnement RStudio**

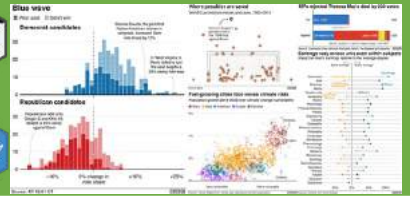
➢ Création d'un document Rmarkdown

- Le corps du document est constitué de texte qui suit la syntaxe *Markdown;*
- Il contient un balisage léger qui permet de définir des niveaux de titres ou de mettre en forme le texte.
- Dans RStudio, le menu *Help* puis ***Markdown quick reference*** donne un aperçu plus complet de la syntaxe.





+ balises spécifiques *html*



TP 0 : **Prise en main du logiciel R et de l'environnement RStudio**

➢ Création d'un document Rmarkdown

Balises légères Markdown

```
***

<div style="text-align: center">
   <img src=Logo_enedis_header.png width="50%" />
</div>

[ENEDIS](https://www.enedis.fr/enedis-en-picardie)

***

```{r setup, include=FALSE, message=FALSE, warning=FALSE}
knitr::opts_chunk$set(include=TRUE, message=FALSE, warning=FALSE, echo = FALSE,
cache=FALSE)
```

Les données
Les données **brutes** du fichier sont affichées dans le tableau ci-dessous.

```{r datas}
paged_table(Parcs)
```

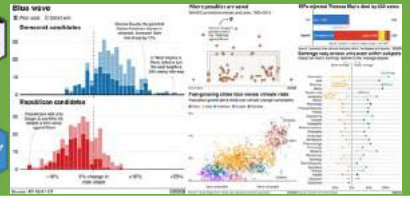
Production électrique

par parcs

Diagramme en bâton

Représentation des valeurs de la distribution de la variable quantitative *Puissance* électrique.
```

Balise spécifique *html*



TP 0 : **Prise en main du logiciel R et de l'environnement RStudio**

➢ Création d'un document Rmarkdown

En plus du texte libre au format Markdown, un document *R Markdown* **contient du code *R***. Celui-ci est inclus dans des blocs (***chunks***) délimités.

```
***

<div style="text-align: center">
   <img src=Logo_enedis_header.png width="50%" />
</div>

[ENEDIS](https://www.enedis.fr/enedis-en-picardie)

***

```{r setup, include=FALSE, message=FALSE, warning=FALSE}
knitr::opts_chunk$set(include=TRUE, message=FALSE, warning=FALSE, echo = FALSE,
cache=FALSE)
```
```

```
# Les données
Les données **brutes** du fichier sont affichées dans le tableau ci-dessous.

```{r datas}
paged_table(Parcs)
```

Production électrique

par parcs

Diagramme en bâton

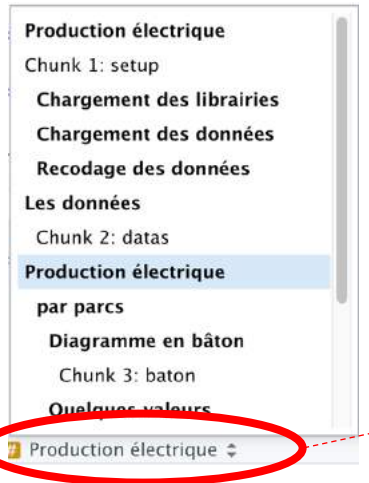
Représentation des valeurs de la distribution de la variable
électrique.
```

Exécution du bloc

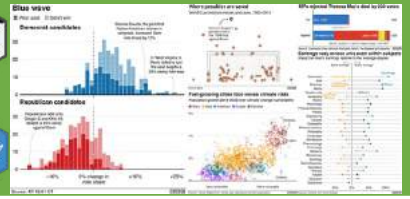
Exécution des blocs antérieurs

```{r datas}
```

Nom du bloc



Pour naviguer facilement



TP 0 : **Prise en main du logiciel R et de l'environnement RStudio**

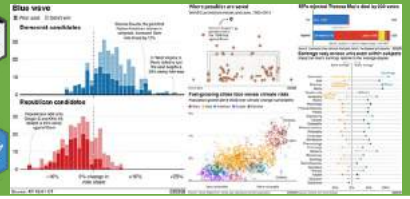
➢ Création d'un document Rmarkdown

```
23  ***
24
25  <div style="text-align: center">
26     <img src=Logo_enedis_header.png width="50%" />
27  </div>
28
29  [ENEDIS](https://www.enedis.fr/enedis-en-picardie)
30
31  ***
32
33  ```{r setup, include=FALSE, message=FALSE, warning=FALSE}
34  knitr::opts_chunk$set(include=TRUE, message=FALSE, warning=FALSE, echo = FALSE,
    cache=FALSE)
```

```
55  # Les données
56  Les données **brutes** du fichier sont affichées dans le tableau ci-dessous.
57
58  ```{r datas}
59  paged_table(Parcs)
60  ```
61
62  # Production électrique
63
64  ## par parcs
65
66  ### Diagramme en bâton
67
68  Représentation des valeurs de la distribution de la variable quantitative *Puissance*
    électrique.
```

```{r datas}

auquel on peut ajouter des options
```



TP 0 : **Prise en main du logiciel R et de l'environnement RStudio**

➢ Création d'un document Rmarkdown

Option	Valeurs	Description
echo	TRUE / FALSE	Afficher ou non le code R dans le document
eval	TRUE / FALSE	Exécuter ou non le code R à la compilation
include	TRUE / FALSE	Inclure ou non le code R et ses résultats dans le document
results	"hide", "asis", "markup", "hold"	Type de résultats renvoyés par le bloc de code
warning	TRUE / FALSE	Afficher ou non les avertissements générés par le bloc
message	TRUE / FALSE	Afficher ou non les messages générés par le bloc

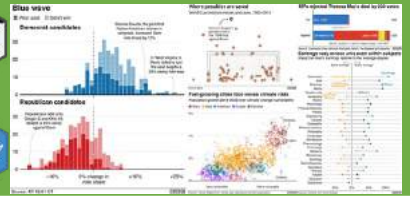
Et il y a d'autres options

- On peut vouloir appliquer une option à l'ensemble des blocs d'un document.
- On utilise la fonction :

```
33 ```{r setup, include=FALSE, message=FALSE, warning=FALSE}
34   knitr::opts_chunk$set(include=TRUE, message=FALSE, warning=FALSE, echo = FALSE,
     cache=FALSE)
```

- Par défaut *RStudio* exécute systématiquement le contenu du bloc setup avant d'exécuter celui d'un autre bloc.

TP 0 : **Prise en main du logiciel R et de l'environnement RStudio**

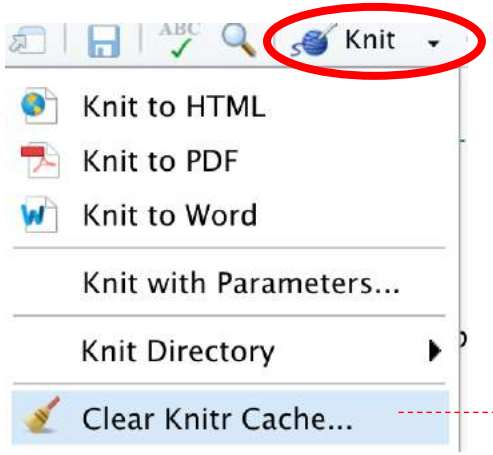


➢ Création d'un document Rmarkdown

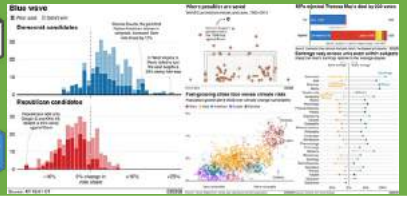
- Compiler un document *R Markdown* peut être long, car il faut à chaque fois exécuter l'ensemble des blocs de code R qui le constituent.
- On peut activer un système de mise en cache des résultats pour enregistrer les résultats des blocs.

Option	Valeurs	Description
cache	TRUE / FALSE	Mise en cache des résultats de chaque bloc

- Ce système de cache peut poser problème par exemple si les données source changent.

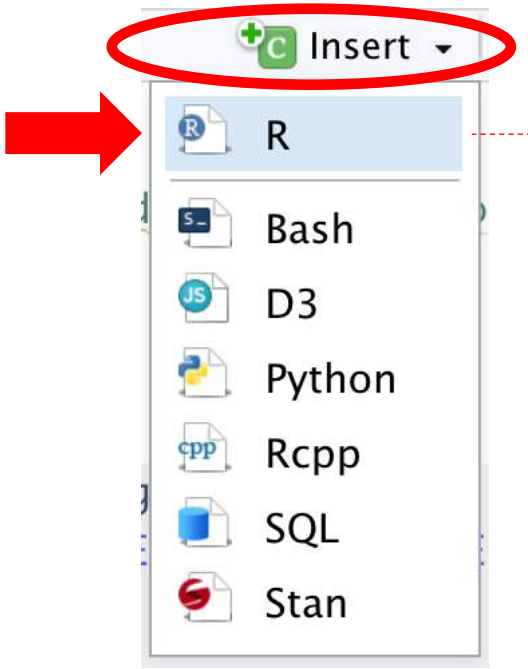


Pour vider le cache du document



TP 0 : **Prise en main du logiciel R et de l'environnement RStudio**

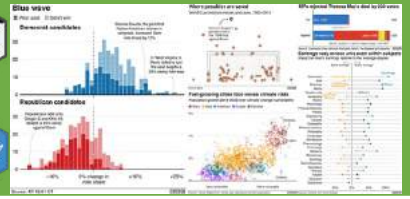
➢ Création d'un document Rmarkdown



Pour insérer un nouveau bloc

```
141 ## par mois
142
143 ```{r}
144
145 ```
```

```
143 ```{r mois}
144 tab2 <-Parcs %>%
145   group_by(NumMois)
146
147 Graph_mois <- ggplot(data=tab2,aes(x=NumMois, y=Puissance/1E6,fill=Parc))+
148   geom_bar(stat="identity")+
149   xlab("Mois") +
150   ylab("Puissance (MW)")+
151   scale_x_continuous(breaks=seq(1,12,by=1))+
152   theme_bw()
153 Graph_mois
154
155 tab3<-tab2 %>%
156   summarise(Psum=sum(Puissance, na.rm=TRUE))
157
158 kable(tab3,caption="Puissance électrique totale par mois",align=c('c','c'))
159 ```
```



TP 0 : **Prise en main du logiciel R et de l'environnement RStudio**

➢ Création d'un document Rmarkdown

Ce document est compilé :

- le texte est mis en forme,
- les blocs de code sont exécutés,
- leur résultat ajouté au document,
- et le tout est transformé dans un des différents formats possibles, ici au format *html*.

Les avantages de ce système sont nombreux :

- le code et ses résultats ne sont pas séparés des commentaires qui leur sont associés;
- le document final est **reproductible**;
- le document peut être très **facilement régénéré et mis à jour**, par exemple si les données source ont été modifiées.

A vous de jouer