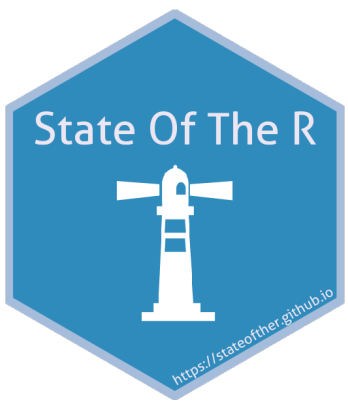
# Statistique avec R

## Premiers Pas

Mathieu Emily, Marie-Pierre Etienne, Magalie Houée-Bigot, François Husson

https://github.com/MarieEtienne/FormationContinueR

Formation Inter Entreprise, Mars 2019

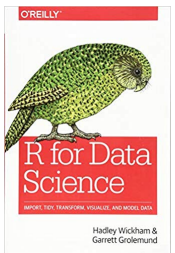



# Plan

# Préambule

Ce cours s'inspire grandement de

- R for Data science (Wickham & Grolemund, 2016), https://r4ds.had.co.nz/




- R pour la statistique et la science des données (Cornillon et al., 2018), https://r-stat-sc-donnees.github.io/


- Cours de Julien Chiquet https://github.com/jchiquet/CourseAdvancedR

# Plan

# Plan

# Qu'appelle-t-on R?

R est à la fois un **logiciel**, un **langage** et un **environnement** informatique dédié au calcul et à l'analyse statistique.

R est un projet open source (projet GNU)

R est un logiciel multi-plateforme (Linux, Mac, Windows)

# La structure R

R est composé d'un socle (base) et de bibilothèques de fonctions thématiques regroupées sous le nom de **package**

Il est possible de connecter R avec d'autres langages : C, Fortran, Java, Javascript, Python...

Il est possible d'appeler des fonctions R depuis Matlab, Excel, SAS, SPSS...

Des connectiques pour tous les types de bases de données : RODBC, RMySQL,ROracle, RJDBC, RMongo...

# Les packages

R a été pensé comme un **langage ouvert et modulaire**.

L'ensemble des fonctionnalités de R est inclus dans des **packages**.

A l'installation de R, les packages de base sont déjà installés dans votre environnement

R permet à n'importe qui de proposer et déposer son package sur le serveur du **CRAN**. De nombreux chercheurs utilisent R donc les nouvelles méthodes sont implémentées.

Aujourd'hui, $13903$ packages sont disponibles
https://cran.r-project.org/web/packages/

Si vous avez besoin d'une fonctionnalité spécifique, vous pouvez avoir besoin d'installer un package!

# R histoire

L'ère pré-R

- 1970's développement de S au Bell labs
- 1980's développement de S-PLUS au AT&T. Lab

Les débuts

- 1993 développement de R sur le modèle de S par Robert Gentleman et Ross Ihaka au département de statistique de l'université d'Auckland.
- 1995 dépôts des codes sources sous licence GNU/GPL

Le succès

- 1997 élargissement du groupe
- 2002 la fondation R dépose ses statuts sous la présidence de Gentleman et Ihaka

Développement entièrement bénévole

- "R development core team" (12aine de personnes)
- Participation de nombreux chercheurs ($\approx$ 13900 packages)

# La communauté R

La page web de la **fondation** R

- les statuts, des liens, des références.
- http://www.r-project.org/

La page web du **CRAN** (Comprehensive R Arxiv Network)

- binaires d'installation, packages, documentations, . . .
- http://cran.r-project.org/

La **conférence** annuelle des utilisateurs de R :

- l'édition 2019 se déroule début juillet en France (Toulouse)
- l'édition 2009 s'est déroulée à Agrocampus Ouest

Depuis 2012, il existe une version française : "les rencontres R".

- l'édition 2018 s'est déroulée à Agrocampus Ouest

The **R journal** propose des articles sur :

- de nouvelles extensions, des applications, des actualités.
- http://journal.r-project.org/

# Plan

# Installer R

## Allez sur http://cran.r-project.org/ et choisissez votre système d'exploitation

*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 want one of these versions of R:

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

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 (2019-03-11, Great Truth) R-3.5.3.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

**Questions About R**

- If you have questions about R like how to download and install the software, or what the license terms are, please read our answers to frequently asked questions before you send an email.

**What are R and CRAN?**

R is 'GNU S', a freely available language and environment for statistical computing and graphics which provides a wide variety of statistical and graphical techniques: linear and nonlinear modelling, statistical tests, time series analysis, classification, clustering, etc. Please consult the R project homepage for further information.

CRAN is a network of ftp and web servers around the world that store identical, up-to-date, versions of code and documentation for R. Please use the CRAN mirror nearest to you to minimize network load.

**Submitting to CRAN**

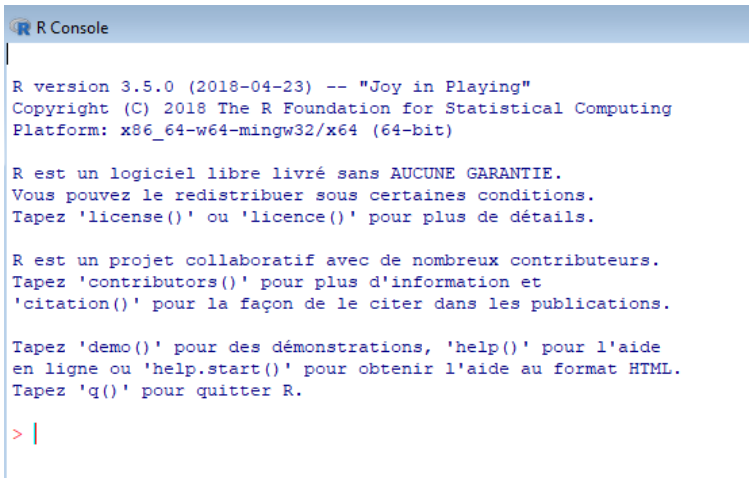
To "submit" a package to CRAN, check that your submission meets the CRAN Repository Policy and then use the web form.

If this fails, upload to ftp://CRAN.R-project.org/incoming/ and send an email to CRAN-submissions@R-project.org following the policy. Please do not attach submissions to emails, because this will clutter up the mailboxes of half a dozen people.

Note that we generally do not accept submissions of precompiled binaries due to security reasons. All binary distribution listed above are compiled by selected maintainers, who are in charge for all binaries of their platform, respectively.

# Lancement de R

Cliquer sur l'icône pour lancer R, une fenêtre, appelée **Console** s'ouvre :




R attend une instruction : ceci est indiqué par $>$ en début de ligne. Cette instruction doit être validée par **Entrée** pour être exécutée.

- instruction correcte, R exécute et redonne la main $>$
- instruction incomplète, R retourne $+$, il faut alors compléter l'instruction ou sortir avec **Echap**

# R et RStudio

## R

- Element de base : c'est le coeur de l'outil
- Utilisation "bas niveau"
- Convivialité réduite

## RStudio

RStudio est un IDE (integrated development environment) pour R.
Principaux avantages de convivialité:

- Editeur de code intégré
- Débogage
- Outil de visualisation de l'environnement de travail

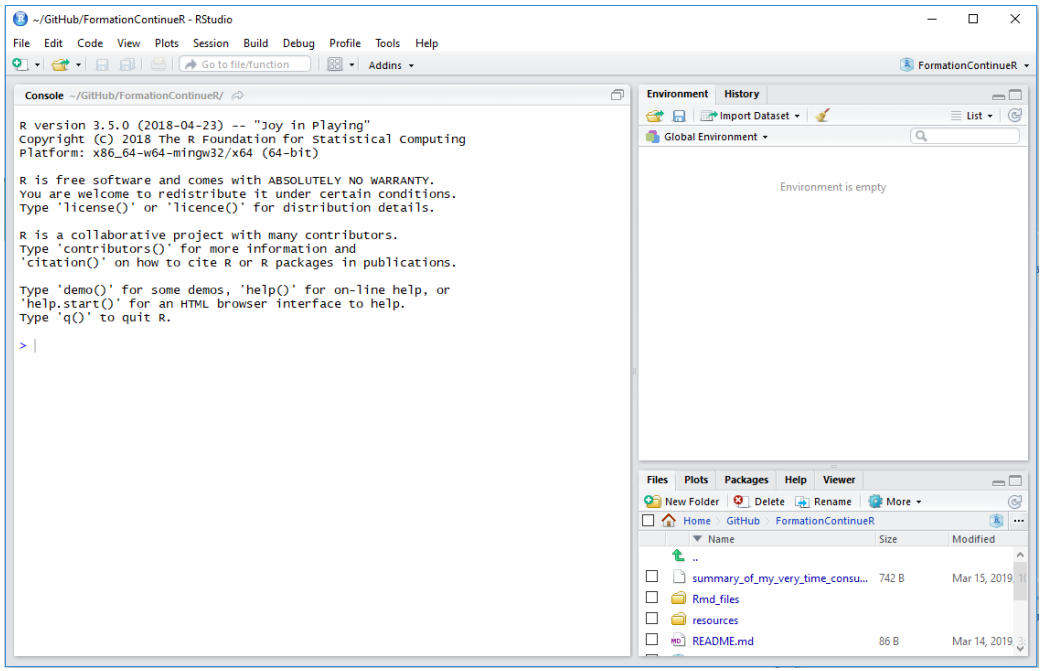
# Plan

# Installer RStudio

Allez sur https://www.rstudio.com/products/RStudio/

# Lancement de RStudio

Cliquer sur l'icône RStudio pour lancer RStudio




RStudio est divisé en 3 (4) quadrants :

- Console
- Espace de travail, historique, importation, ...
- Visualisation (graphiques), aide, ...
- Editeur de texte, de codes, ... (4ème quadrant à insérer )

# Console et environnement de travail

L'environnement de travail permet de :

- faire des calculs

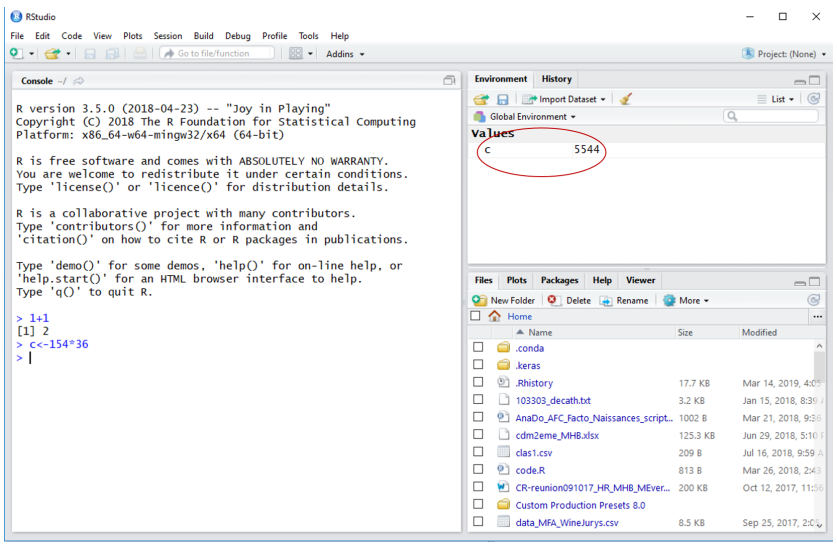
```
1+1
```

```
## [1] 2
```

- stocker les calculs dans des **variables** ou **objets**

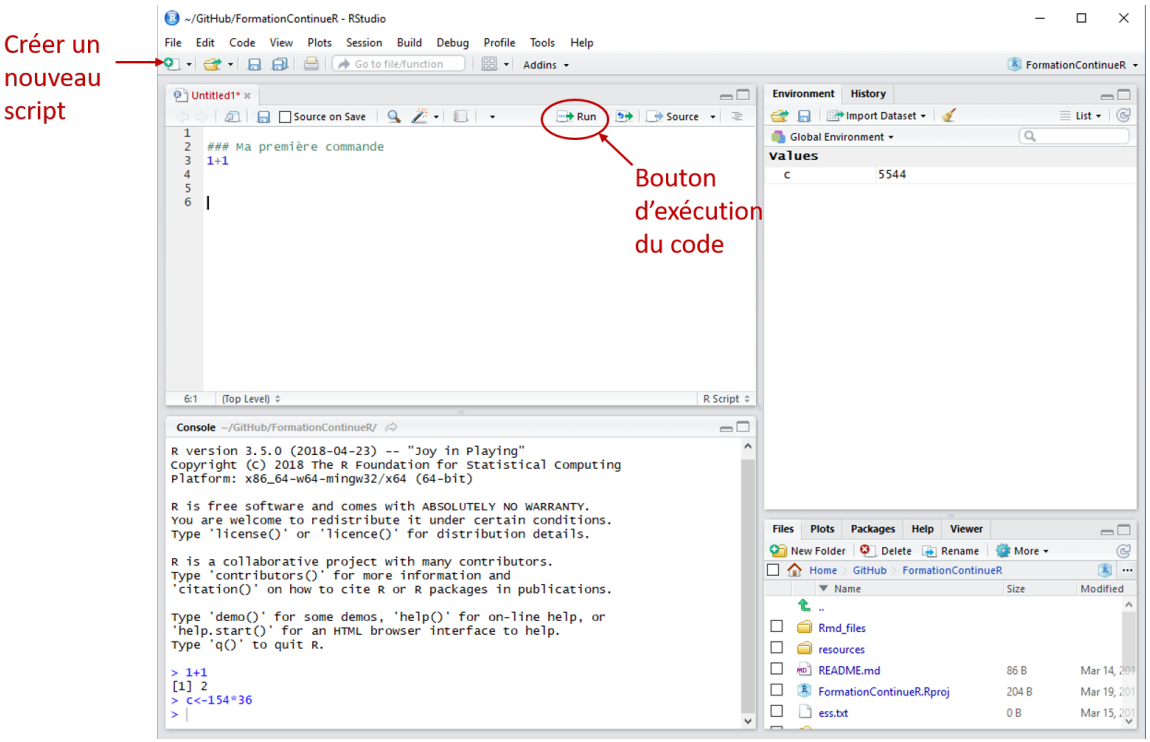
```
c<-154*36
c
```

```
## [1] 5544
```



# Plan

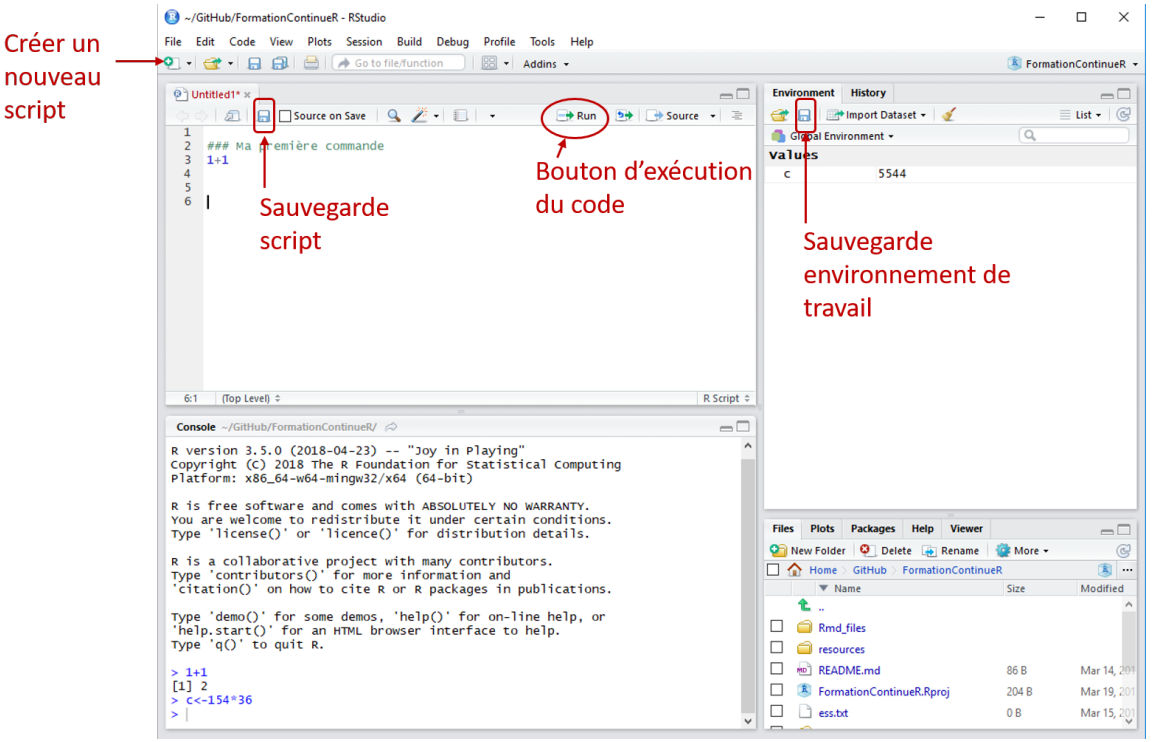
# Faire un script




Le code se tape dans la fenêtre de script et s'exécute directement.

- # pour insérer des commentaires
- **Shift+Alt+k** pour les raccourcis clavier

# Faire un script




- Possibilité de sauvegarder le script (.R),
- Possibilité de sauvegarder l'environnement de travail (.RData).

# Les Projets dans RStudio

RStudio dispose d'une fonctionnalité très pratique pour organiser son travail en différents projets.

L'idée est de réunir tous les fichiers, documents (données, scripts, ...) relatifs à un même projet dans un répertoire dédié.

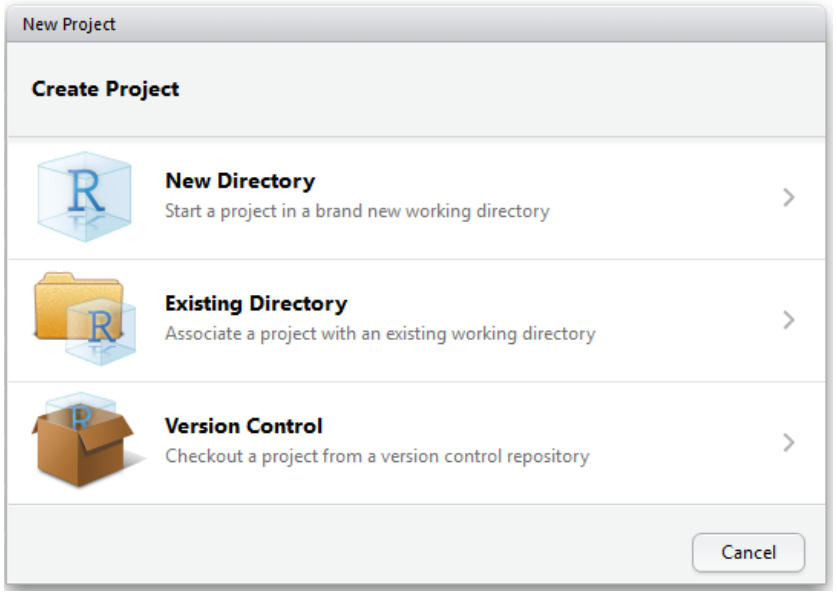
## Les avantages

- Facilite l'accès aux fichiers de données à importer : le répertoire de travail de R est défini comme étant le répertoire du projet
- L'onglet Files de l'interface permet de naviguer dans les fichiers du projet
- Les objets créés (et sauvegardés dans le fichier .Rdata) lors d'une précédente séance de travail sont chargés en mémoire
- Les scripts ouverts lors d'une précédente séance de travail sont automatiquement ouverts

Lorsque l'on ouvre un projet RStudio, on revient à l'état de notre projet tel qu'il était la dernière fois que l'on a travaillé dessus.

# Créer un nouveau projet

Pour créer un projet, File –> New Project




Choisir *Existing directory* ou *New directory* selon l'existence ou non du dossier du projet. Créer ou sélectionner le dossier, puis cliquer sur *Create project*

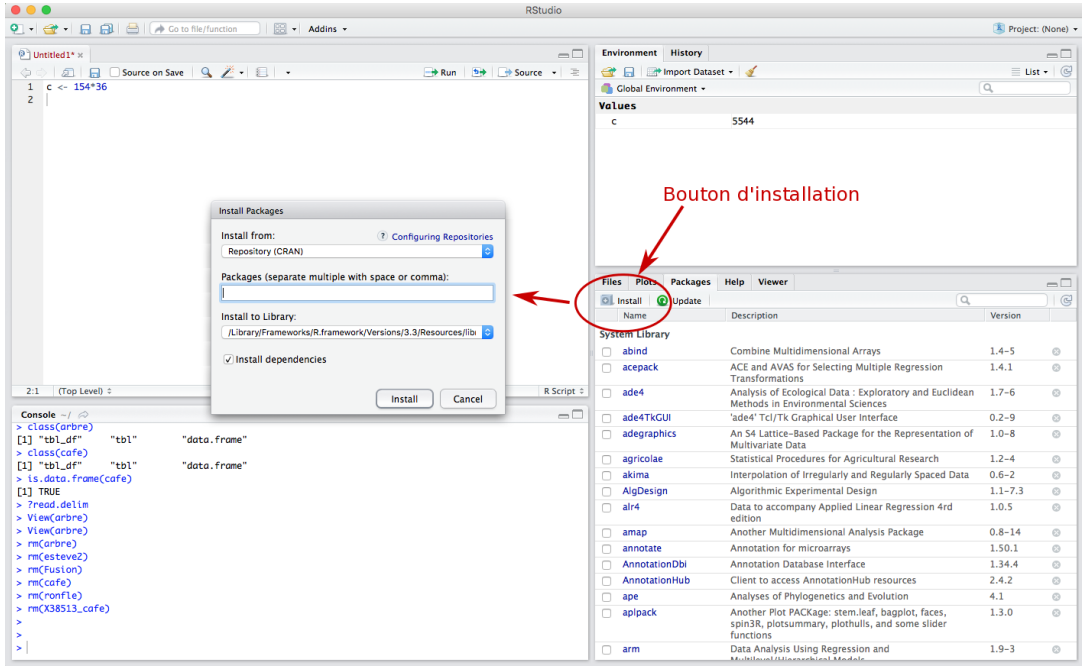
Une fois le projet créé, son nom est affiché dans un petit menu déroulant en haut à droite de l'interface de RStudio FormationContinueR (menu qui permet de passer facilement d'un projet à un autre).

# Plan

# Installer un package

Le but de l'installation est de télécharger et placer au bon endroit les codes R contenus dans le package

- En mode interactif, directement à l'aide d'un site "miroir":




- En mode ligne de commande

```
install.packages('ggplot2')
```

- Vérifier si le package est disponible et l'installer uniquement si besoin

```
if (! require('FactoMineR')) install.packages('FactoMineR')
```

# Charger le package

Il ensuite charger le package dans votre environnement lorsque l'on souhaite l'utiliser.

- En mode interactif

en cochant la case du package dans le menu Packages de RStudio

- dans la console : fonction library ou require

```
library('FactoMineR')
```

Le mode console est le mode compatible avec la production de documents.

# Exercice

- Installer le package `dplyr`
- Charger le package `dplyr`

# L'aide

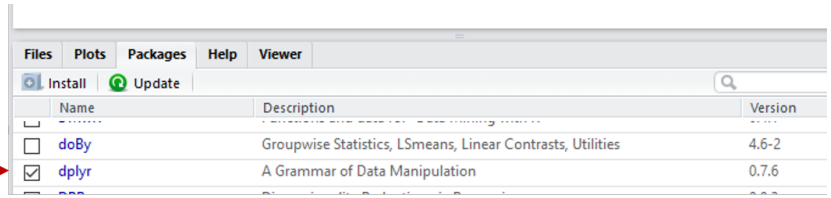
Pour obtenir de l'aide :

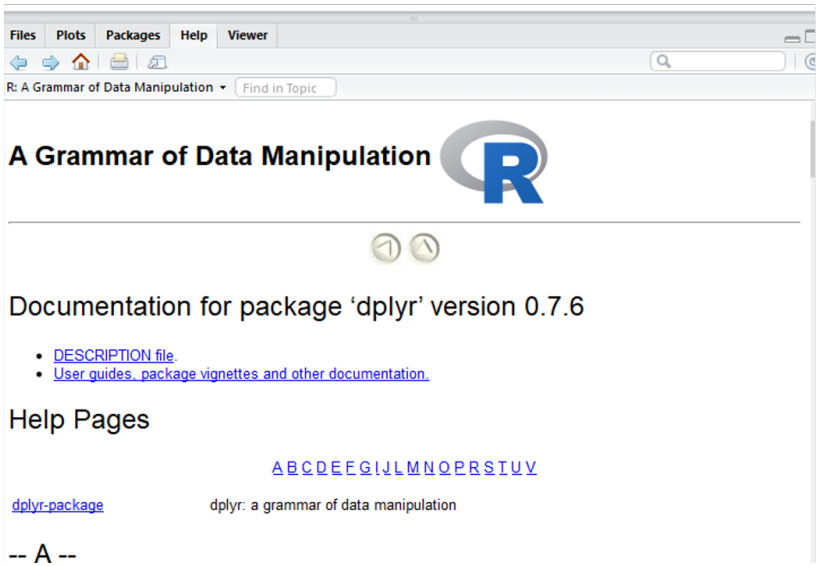
- en ligne de code dans la console

```
help(dplyr) # lance l'aide associée à la commande dplyr
help.start()  # lande l'aide HTML
```

- en mode interactif

Cliquer sur le nom du package dans l'onglet Packages



# L'aide

La plupart des packages propose une documentation intitulée **vignettes**, décrivant l'utilisation du package

Pour accéder à la (aux) vignette(s) d'un package, taper dans la console :

```
browseVignettes("dplyr")
```

Il y a toujours des exemples que l'on peut exécuter directement.

# Plan

# Rapports automatisés : R Markdown

L'extension **rmarkdown** permet de générer des documents de manière dynamique en mélangeant texte mis en forme et résultats produits par du code R.

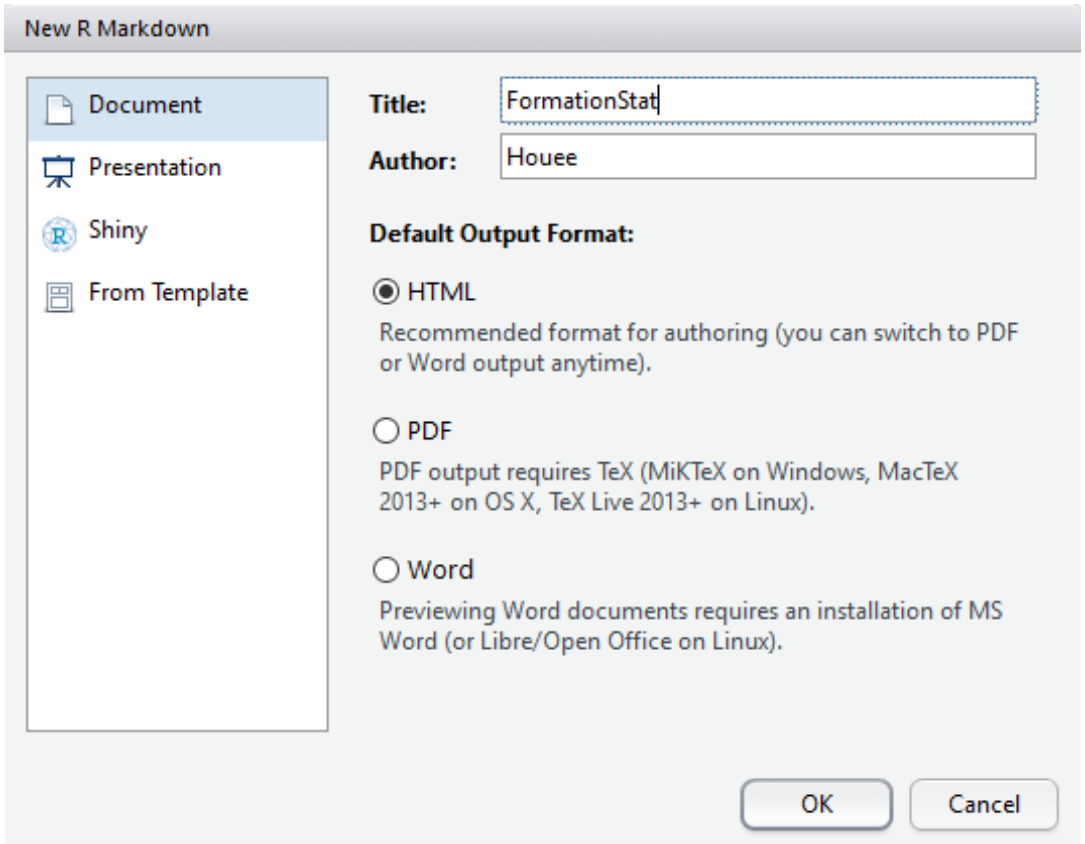
Les documents générés peuvent être au format HTML, PDF, Word, ...

Créer un R Markdown :

- New File –> R Markdown...
- Préciser le nom du document et le format de sortie souhaité

Le fichier créé a une extension .rmd

# Création du document



# Eléments d'un document R Markdown

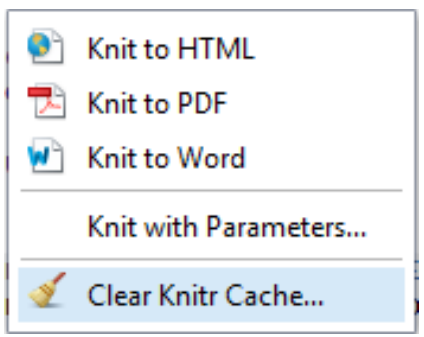
- en-tête délimité par 3 tirets
- texte du document
- blocs de code R

````
---
title: "FormationStat"
author: "Houee"
date: "21 mars 2019"
output: html_document
---

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

## R Markdown

This is an R Markdown document. Markdown is a simple formatting s
and MS Word documents. For more details on using R Markdown see

When you click the **Knit** button a document will be generated t
well as the output of any embedded R code chunks within the docur
chunk like this:

```{r cars}
summary(cars)
```

## Including Plots

You can also embed plots, for example:

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

Note that the `echo = FALSE` parameter was added to the code chur
code that generated the plot.
````

# Compiler un document

On peut à tout moment compiler, ou plutôt “tricoter” (Knit), un document R Markdown pour obtenir et visualiser le document généré.

Pour cela, il suffit de cliquer sur le bouton Knit et de choisir le format de sortie voulu :




*Pour la génération du format PDF, vous devez avoir une installation fonctionnelle de LaTeX sur votre système.*

# Exercice

- Créer votre R Markdown pour la formation

# Plan

# Les objets

Principaux types de données :

- booléen ("logical") : TRUE, FALSE
- numeric : integer ou double
- caractères (character) : "bonjour"
- vide (null) : NULL
- complexe (complex) : 2+0i,2i
- binaires (raw)

Structuration des données :

- tous les éléments sont de même type (vecteur, matrice, tableaux)
- de types différents : liste, data.frame

*Le tableau individus/variable est la structure en statistique: chaque colonne représente une variable. Tous les éléments d'une colonne sont donc de même mode mais les colonnes peuvent être de modes différents (variables qualitatives, quantitatives)*

# Les objets

Tout objet de R possède :

- un **mode** par exemple "logical", "numeric", "character", "list", "function"
- une **classe** :
  - vecteurs (vector),
  - facteurs (factor),
  - matrice (matrix),
  - liste (list),
  - fonction (function),
  - tableau de données (data frame)

```
X<-c(1:5,10,12) # création d'un vecteur intitulé X
X # affichage de X
```

```
## [1]  1  2  3  4  5 10 12
```

```
is.vector(X) # X est-il un vecteur?
```

```
## [1] TRUE
```

```
class(X) # quelle est la classe de l'objet?
```

# Les opérateurs logiques dans R

```
?Comparison
# et
?base::Logic
```

| Opérateurs | | Opérateurs | |
|---|---|---|---|
| $<$ | inférieur strictement à | $!=$ | différent de |
| $>$ | supérieur strictement à | %in% | appartient à |
| $==$ | égal à | is.na | est manquant |
| $<=$ | inférieur ou égal à | !is.na | n'est pas manquant |

# Plan

# Bonnes pratiques pour les fichiers

- Préférer un format `csv`, éviter les formats propriétaires `xlsx`
- Eviter les accents, espaces et caractères spéciaux dans les noms de fichier et les noms de variables
- Garder une trace de toutes les opérations de pré traitement des données (typiquement dans un fichier `RMarkdown`)

# Plan

# Importer un fichier en mode interactif

File - Import Dataset - From Text (base) - choix du fichier

# Exercice

- Importer le fichier `SamaresEq.txt` dans la variable `SamaresEq_base` et vérifier la classe de l'objet obtenu
- Importer le fichier `SamaresEq.txt` avec l'option From text (readr) `SamaresEq_readr` et vérifier la classe de l'objet obtenu

# Importer un fichier en ligne de commande

```
mon_fichier <- "../../Datasets/SamaresEq.txt"
SamaresEq_base <- read.table(file = mon_fichier, sep = " ", header = TRUE, dec = '.')
head(SamaresEq_base, n=3)
```

```
##   Site NomSite Distance Arbre Poids Surface Largeur Longueur CircArbre
## 1    4 Gornies      6.3     1  11.0 0.73816   0.427  2.31502      18.5
## 2    4 Gornies      6.3     1  15.0 0.69958   0.489  2.15873      18.5
## 3    4 Gornies      6.3     1  14.6 0.75847   0.444  2.27595      18.5
```

- `file` indique le nom complet du fichier (potentiellement avec le chemin d'accès)
- `sep` décrit le type de séparateur utilisé dans le fichier - `dec` décrit le signe pour les décimales (utile pour les fichiers en Français)

La version ligne de commande préférable dans une optique de science reproductible.

# Importer un fichier en ligne de commande

En utilisant le package readr

```
mon_fichier <- "../../Datasets/SamaresEq.txt"
SamaresEq_readr <- read_delim(file = mon_fichier, delim = " ")
SamaresEq_readr
```

```
## # A tibble: 2,380 x 9
##    Site NomSite Distance Arbre Poids Surface Largeur Longueur CircArbre
##   <dbl> <chr>      <dbl> <dbl> <dbl>   <dbl>   <dbl>    <dbl>     <dbl>
## 1     4 Gornies      6.3     1  11     0.738   0.427     2.32      18.5
## 2     4 Gornies      6.3     1  15     0.700   0.489     2.16      18.5
## 3     4 Gornies      6.3     1  14.6   0.758   0.444     2.28      18.5
## 4     4 Gornies      6.3     1  13.8   0.779   0.474     2.26      18.5
## # ... with 2,376 more rows
```

- `file` indique le nom complet du fichier (potentiellement avec le chemin d'accès)
- `delim` décrit le type de séparateur utilisé dans le fichier

La version ligne de commande préférable dans une optique de science reproductible.

# Importer depuis une `url` en ligne de commande

Même approche pour une importation depuis une `url` : Le fichier decathlon

```
monFichier <- 'https://husson.github.io/img/decathlon.csv'
decathlon <- read.table(file = monFichier, header = TRUE, sep = ';')
```

# Exercice

- Importer le jeu de données vin
- Créer un fichier `RMarkdown` nommé `Formation_R_exercices` et insérer le code pour créer le tableau `vins_base` contenant le jeu de données vin en utilisant la commande `read.table`
- Insérer le code pour créer le tableau `vins_readr` contenant le jeu de données vin en utilisant la commande `read_delim` du package `readr`.

# Solution

- Importer le jeu de données vin

```
vins_base <- read.table(file = 'http://factominer.free.fr/livre/vins.csv',
                        sep = ';', header =  TRUE, encoding = 'latin1')
```

# Solution

- Créer un fichier `RMarkdown` nommé `Formation_R_exercices.Rmd` et insérer le code pour créer le tableau `vins_base` contenant le jeu de données vin en utilisant la commande `read.table`
  - New File – R Markdown
  - Modifier auteur et titre
  - save as `Formation_R_exercices.Rmd`

# Solution

- Insérer le code pour créer le tableau `vins_readr` contenant le jeu de données vin en utilisant la commande `read_delim` du package `readr`.

```
library(readr)
vins_readr <- vins <- read_delim("http://factominer.free.fr/livre/vins.csv",
                                 delim = ";",
                                 escape_double = FALSE, locale = locale(encoding = "latin1"),
                                 trim_ws = TRUE)
```

# Vérifier l'importation

- Afficher le fichier dans son ensemble ( éviter pour des fichiers longs)

```
SamaresEq_base
```

- Afficher les premières ou dernières lignes

```
head(SamaresEq_base, n = 2)
```

```
##   Site NomSite Distance Arbre Poids Surface Largeur Longueur CircArbre
## 1    4 Gornies      6.3     1    11 0.73816   0.427  2.31502      18.5
## 2    4 Gornies      6.3     1    15 0.69958   0.489  2.15873      18.5
```

```
tail(SamaresEq_base, n = 3)
```

```
##      Site   NomSite Distance Arbre Poids Surface Largeur Longueur
## 2378    5 StLaurent     11.3    29  41.6 1.00891 0.56836  2.40750
## 2379    5 StLaurent     11.3    29  35.3 0.94044 0.58594  2.29955
## 2380    5 StLaurent     11.3    29  26.7 0.76706 0.48735  2.18422
##      CircArbre
## 2378        26
## 2379        26
## 2380        26
```

Dans le cas d'objet `tibble` obtenu avec l'importation du package `readr`, l'affichage du tableau ne donne que les premières lignes

```
SamaresEq_readr
```

# Importation d'un jeu de données à l'aide une requête SQL (base de données)

- Connexion à une base SQL.

```
library(RODBC)
# Liste les tables de la base de données connectée
sqlTables(connect_base, tableType = "TABLE")
# Liste les champs de la table  DonneesTotales
sqlColumns(connect_base, sqtable = "DonneesTotales")
```

- Executer une requête sur la base

```
# execute une requete SQL
OtoYFT <- sqlQuery( channel = connect_base,
                    query =
"
SELECT * FROM DonneesTotales
WHERE (DonneesTotales.ProblemeSp = 'Ok' AND DonneesTotales.REC_Sp='Y'
AND DonneesTotales.Otolithe = 'OT')
")
# Liste les champs de la table  DonneesTotales
sqlColumns(connect_base, sqtable = "DonneesTotales")
```

```
## [1] "Importation.html"   "Importation.Rmd"    "Manip_standard.Rmd"
## [4] "Manip_tidy.html"    "Manip_tidy.Rmd"
```

# Plan

# La structure des tableaux de données dans R

Un tableau de données est un objet `data.frame`.

- Connaître les dimensions d'un tableau

```
dim(SamaresEq_base)
```

```
## [1] 2380    9
```

- Connaître les noms de variables

```
colnames(SamaresEq_base)
```

```
## [1] "Site"      "NomSite"   "Distance"  "Arbre"     "Poids"     "Surface"
## [7] "Largeur"   "Longueur"  "CircArbre"
```

- Accéder à une variable

```
head(SamaresEq_base$Poids, n=5)
```

```
## [1] 11.0 15.0 14.6 13.8 12.2
```

- Accéder à une ligne

```
SamaresEq_base[2, ]
```

```
##   Site NomSite Distance Arbre Poids Surface Largeur Longueur CircArbre
## 2    4 Gornies      6.3     1    15 0.69958   0.489  2.15873      18.5
```

# Plan

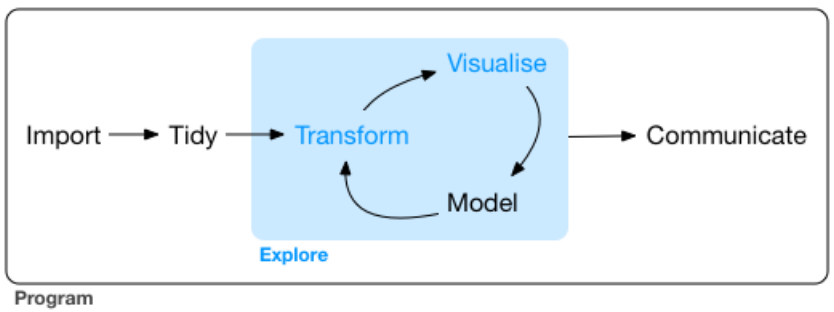
# Présentation

Le `tidyverse` est ensemble de packages développés pour {*faciliter*} *la manipulation de données dans* `R`.

- *Installer le package* `tidyverse`.

```
install.packages("tidyverse")
```

*D'après les créateurs dans Wickham & Grolemund (2016)*




*Charger le packahe* `tidyverse`

```
library(tidyverse)
```

*Objectif : Obtenir un code plus lisible*

# Plan

# Selectionner les individus qui satisfont une condition

## filter

```
SamaresEq_base %>% filter( Surface > 0.75) -> Grand_samares
class(Grand_samares)
```

```
## [1] "data.frame"
```

```
Grand_samares
```

```
##      Site    NomSite Distance Arbre Poids Surface Largeur Longueur
## 1       4    Gornies      6.3     1  14.6 0.75847 0.44400  2.27595
## 2       4    Gornies      6.3     1  13.8 0.77878 0.47400  2.25641
## 3       4    Gornies      6.3     1  13.3 0.75644 0.44700  2.14896
## 4       4    Gornies      6.3     1  13.7 0.78182 0.47400  2.23687
## 5       4    Gornies      6.3     1  17.1 0.89757 0.47400  2.50061
## 6       4    Gornies      6.3     1  15.3 0.78995 0.48300  2.12943
## 7       4    Gornies      6.3     1  18.3 0.83259 0.41100  2.63736
## 8       4    Gornies      6.3     2  12.2 0.82654 0.44631  2.70560
## 9       4    Gornies      6.3     2  15.8 0.77353 0.43744  2.70560
## 10      4    Gornies      6.3     3  27.1 0.77488 0.46570  2.14286
## 11      4    Gornies      6.3     3  29.2 0.83023 0.48889  2.29464
## 12      4    Gornies      6.3     3  25.2 0.77934 0.45121  2.31250
## 13      4    Gornies      6.3     3  30.2 0.75881 0.40580  2.45536
## 14      4    Gornies      6.3     3  23.2 0.76863 0.48696  2.20536
## 15      4    Gornies      6.3     3  24.5 0.78916 0.45411  2.34821
## 16      4    Gornies      6.3     4  22.7 0.87292 0.47810  2.54825
## 17      4    Gornies      6.3     4  27.7 0.94552 0.51572  2.63664
## 18      4    Gornies      6.3     4  28.6 0.92876 0.47556  2.74652
## 19      4    Gornies      6.3     4  26.6 0.84011 0.49986  2.32801
```

# Selectionner les individus qui satisfont une condition

## filter

```
SamaresEq_base %>% as_tibble %>% filter( Surface > 0.75) -> Grand_samares
class(Grand_samares)
```

```
## [1] "tbl_df"     "tbl"        "data.frame"
```

```
Grand_samares
```

```
## # A tibble: 1,517 x 9
##    Site NomSite Distance Arbre Poids Surface Largeur Longueur CircArbre
##   <int> <fct>      <dbl> <int> <dbl>   <dbl>   <dbl>    <dbl>     <dbl>
## 1     4 Gornies      6.3     1  14.6   0.758   0.444     2.28      18.5
## 2     4 Gornies      6.3     1  13.8   0.779   0.474     2.26      18.5
## 3     4 Gornies      6.3     1  13.3   0.756   0.447     2.15      18.5
## 4     4 Gornies      6.3     1  13.7   0.782   0.474     2.24      18.5
## # ... with 1,513 more rows
```

```
SamaresEq_readr %>% as_tibble %>% filter( Surface > 0.75) -> Grand_samares
class(Grand_samares)
```

```
## [1] "tbl_df"     "tbl"        "data.frame"
```

```
Grand_samares
```

```
## # A tibble: 1,517 x 9
##    Site NomSite Distance Arbre Poids Surface Largeur Longueur CircArbre
##   <dbl> <chr>      <dbl> <dbl> <dbl>   <dbl>   <dbl>    <dbl>     <dbl>
## 1     4 Gornies      6.3     1  14.6   0.758   0.444     2.28      18.5
```

# Exercice

- Sélectionner toutes les observations du site Gornies
- Sélectionner toutes les observations correspondantes à des largeurs supérieures à 0.45 mais le longueur inférieure à 2.32
- Sélectionner toutes les observations qui ne proviennent ni du site `Gornies` ni du site `StEtienne`.

# Solution

## `filter`

- Sélectionner toutes les observations du site Gornies

```
SamaresEq_readr %>% filter(Site == 'Gornies')
```

- Sélectionner toutes les observations correspondantes à des largeurs supérieures à 0.45 mais le longueur inférieure à 2.32

```
SamaresEq_readr %>% filter(Largeur > 0.45 & Longueur < 2.32)
```

- Sélectionner toutes les observations qui ne proviennent ni du site `Gornies` ni du site `StEtienne`.

```
SamaresEq_readr %>% filter(  ! NomSite %in% c('Gornies', 'StEtienne') )
```

```
## # A tibble: 1,660 x 9
##    Site NomSite  Distance Arbre Poids Surface Largeur Longueur CircArbre
##   <dbl> <chr>       <dbl> <dbl> <dbl>   <dbl>   <dbl>    <dbl>     <dbl>
## 1     1 Grenou_1        0     1  16.8   0.839   0.458     2.46        57
## 2     1 Grenou_1        0     1  25.4   0.809   0.433     2.48        57
## 3     1 Grenou_1        0     1  21.7   0.774   0.441     2.36        57
## 4     1 Grenou_1        0     1  14.9   0.860   0.453     2.64        57
## # ... with 1,656 more rows
```

# Plan

# Sélectionner certaines variables

`select`

Pour modifier les variables présentes dans le jeu de données

- Ne garder que la variable `NomSite`

```
SamaresEq_base %>% select(NomSite)
```

- Supprimer la variable `Site` et `Arbre`

```
SamaresEq_base %>% select(-Site, -Arbre)
```

# Exercice

`select`

-A partir de la table de données `SamaresEq_base`, créer une table `SamaresEq_base_gornies` qui contient les information concernant la largeur et la longueur de chaque samare uniquement pour les arbres du Site `Gornies`

# Solution

## select

-A partir de la table de données SamaresEq_base, créer une table SamaresEq_base_gornies qui contient les information concernant la largeur et la longueur de chaque samare uniquement pour les arbres du Site Gornies

```
SamaresEq_base %>% filter( NomSite = 'Gornies') %>%  select(-Site, -Arbre) -> SamaresEq_base_go
head(SamaresEq_base)
```

# Créer des nouvelles variables

## mutate

```
SamaresEq_readr %>%
  mutate(dispersion = Surface / Poids,
         log_disp = log( dispersion )) -> SamaresEq_disp
SamaresEq_disp %>% select(-Site, -Arbre, -Distance, -CircArbre)
```

```
## # A tibble: 2,380 x 7
##   NomSite Poids Surface Largeur Longueur dispersion log_disp
##   <chr>   <dbl>   <dbl>   <dbl>    <dbl>      <dbl>    <dbl>
## 1 Gornies  11     0.738   0.427     2.32     0.0671    -2.70
## 2 Gornies  15     0.700   0.489     2.16     0.0466    -3.07
## 3 Gornies  14.6   0.758   0.444     2.28     0.0520    -2.96
## 4 Gornies  13.8   0.779   0.474     2.26     0.0564    -2.87
## # ... with 2,376 more rows
```

# Exercice

`mutate`

- A partir de la table de données `SamaresEq_readr`, ajouter une variable `larg_x_long` contenant le produit de la largeur et de la longueur et une colonne `diff_surf` qui calcule la différence entre la variable précédemment définie et la surface présente dans la table.

# Solution

## select

- A partir de la table de données SamaresEq_readr, ajouter une variable larg_x_long contenant le produit de la largeur et de la longueur et une variable diff_surf qui calcule la différence entre la variable précédemment définie et la surface présente dans la table.

```
SamaresEq_readr %>% mutate(larg_x_long = Largeur * Longueur,
                           diff_surf = larg_x_long - Surface)  %>%
  select(-Site, -Arbre, -Distance, -CircArbre)
```

## On peut limiter l'affichage

```
SamaresEq_readr %>% mutate(larg_x_long = Largeur * Longueur,
                           diff_surf = larg_x_long - Surface)  %>%
  select(-Site, -Arbre, -Distance, -CircArbre)  %>%
  print(n = 3)
```

```
## # A tibble: 2,380 x 7
##   NomSite Poids Surface Largeur Longueur larg_x_long diff_surf
##   <chr>   <dbl>   <dbl>   <dbl>    <dbl>       <dbl>     <dbl>
## 1 Gornies  11     0.738   0.427     2.32       0.989     0.250
## 2 Gornies  15     0.700   0.489     2.16       1.06      0.356
## 3 Gornies  14.6   0.758   0.444     2.28       1.01      0.252
## # ... with 2,377 more rows
```

# Résumer des variables

## summarise

- Calculer des moyennes

```
SamaresEq_readr %>%
  summarise( longueur_m  = mean(Longueur, na.rm = TRUE))
```

```
## # A tibble: 1 x 1
##   longueur_m
##        <dbl>
## 1       2.49
```

- Calculer le nombre d'observations, les médianes pour plusieurs variables

```
SamaresEq_readr %>%
  summarise_at( vars(Largeur, Longueur), funs(n(), median))
```

```
## # A tibble: 1 x 4
##   Largeur_n Longueur_n Largeur_median Longueur_median
##       <int>      <int>          <dbl>           <dbl>
## 1      2380       2380          0.445            2.48
```

# Résumer des variables

## summarise

- Calculer les moyennes de toutes les variables quantitatives

```
SamaresEq_readr %>%
  summarise_if(is.numeric, mean, na.rm=TRUE)
```

```
## # A tibble: 1 x 8
##    Site Distance Arbre Poids Surface Largeur Longueur CircArbre
##   <dbl>    <dbl> <dbl> <dbl>   <dbl>   <dbl>    <dbl>     <dbl>
## 1  3.98     8.99  10.9  26.6   0.823   0.451     2.49      26.2
```

# Exercice

`summarise`

- Calculer la moyenne et l'écart-type pour les variables `Surface` et `disp`.

# Exercice

## summarise

- Calculer la moyenne et l'écart-type pour les variables `Surface` et `dispersion`.

```
SamaresEq_disp %>%
  summarise_at( vars(Surface, dispersion),  funs( sd, mean), na.rm = TRUE)
```

```
## # A tibble: 1 x 4
##   Surface_sd dispersion_sd Surface_mean dispersion_mean
##        <dbl>         <dbl>        <dbl>           <dbl>
## 1      0.179       0.00948        0.823          0.0330
```

# Plan

# Calculer des moyennes pour chaque groupe

## group_by

- Calculer la dispersion moyenne pour chaque site

```
SamaresEq_disp %>% group_by( NomSite) %>%
  summarise( Surface_m = mean (Surface)) %>%
  print(n = 3)
```

```
## # A tibble: 7 x 2
##   NomSite  Surface_m
##   <chr>        <dbl>
## 1 Gornies      0.849
## 2 Grenou_1     0.810
## 3 Grenou_2     0.853
## # ... with 4 more rows
```

# Calculer des effectifs pour chaque groupe

## group_by

- Calculer les effectifs par Site et par arbre

```
SamaresEq_disp %>% group_by( NomSite, Arbre ) %>%
summarise( n_obs = n())  %>%   print(n = 3)
```

```
## # A tibble: 119 x 3
## # Groups:   NomSite [?]
##   NomSite Arbre n_obs
##   <fct>   <int> <int>
## 1 Gornies     1    20
## 2 Gornies     2    20
## 3 Gornies     3    20
## # ... with 116 more rows
```

# Exercice

- Pour chaque site et chaque arbre, donner le nombre de samares échantillonés et leur poids moyen.
- Pour chaque site, donner le nombre d'arbres échantillonnés.

# Solution

- Pour chaque site et chaque arbre, donner le nombre de samares échantillonés et leur poids moyen.

```
SamaresEq_disp %>% group_by( NomSite, Arbre ) %>%
summarise( n_obs = n(), poids_m = mean(Poids))  %>%   print(n = 3)
```

```
## # A tibble: 119 x 4
## # Groups:   NomSite [?]
##   NomSite Arbre n_obs poids_m
##   <chr>   <dbl> <int>   <dbl>
## 1 Gornies     1    20    13.4
## 2 Gornies     2    20    11.5
## 3 Gornies     3    20    24.6
## # ... with 116 more rows
```

- Pour chaque site, donner le nombre d'arbres échantillonnés.

```
SamaresEq_disp %>% group_by( NomSite) %>% summarise( n_Arbre = n_distinct(Arbre))
```

```
## # A tibble: 7 x 2
##   NomSite  n_Arbre
##   <fct>      <int>
## 1 Gornies       29
## 2 Grenou_1      14
## 3 Grenou_2      15
## 4 Grenou_3      10
## # ... with 3 more rows
```

# Plan

# Sauvegarder dans un format texte

## write_csv

```
write_csv(SamaresEq_disp, path = "../../Datasets/SamaresEq_disp.csv")
```

# Plan

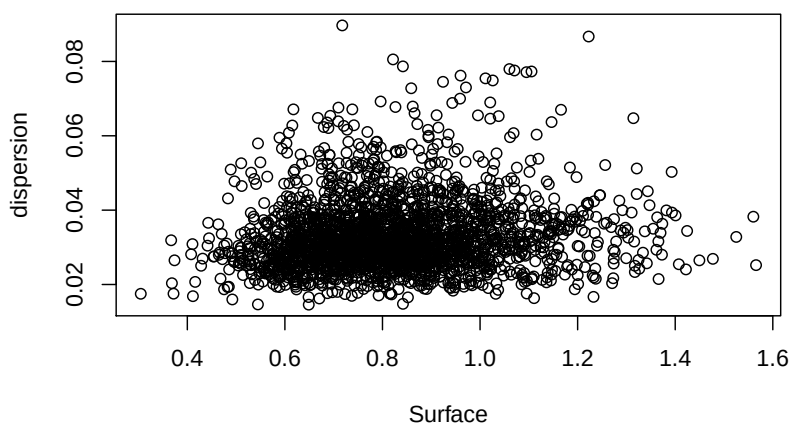
# Plan

# Scatter plot

```
head(SamaresEq_disp)
```

```
## # A tibble: 6 x 11
##    Site NomSite Distance Arbre Poids Surface Largeur Longueur CircArbre
##   <dbl> <chr>      <dbl> <dbl> <dbl>   <dbl>   <dbl>    <dbl>     <dbl>
## 1     4 Gornies      6.3     1  11     0.738   0.427     2.32      18.5
## 2     4 Gornies      6.3     1  15     0.700   0.489     2.16      18.5
## 3     4 Gornies      6.3     1  14.6   0.758   0.444     2.28      18.5
## 4     4 Gornies      6.3     1  13.8   0.779   0.474     2.26      18.5
## # ... with 2 more rows, and 2 more variables: dispersion <dbl>,
## #   log_disp <dbl>
```

```
plot(dispersion~Surface, data = SamaresEq_disp)
```

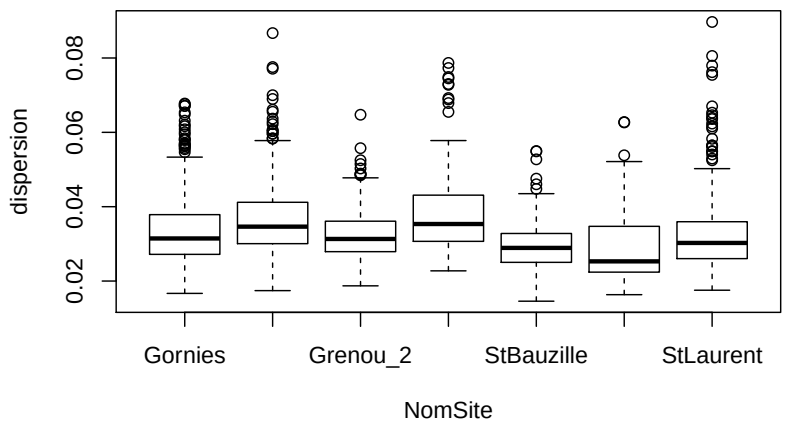


# Boîtes à moustaches

```
plot(dispersion~NomSite, data=SamaresEq_disp)
```

la variable `NomSite` n'est pas un facteur.

```
SamaresEq_disp %>% mutate(NomSite = as.factor(NomSite)) -> SamaresEq_disp
plot(dispersion~NomSite, data=SamaresEq_disp)
```



# Plan

# Les idées derrière ggplot2

ggplot2 propose de construire des graphiques en suivant une grammaire graphique. Hadley Wickham créateur du package a écrit ce papier pour expliquer son approche.

Un graphique est composé

- d'un jeu de données
- dont on veut représenter certains aspects
- en utilisant une forme adaptée

# Les idées derrière ggplot2

Un graphique est composé

- d'un jeu de données : `ggplot`
- dont on veut représenter certains aspects : `aes`
- en utilisant une forme adaptée : `geom` et `stat`

La côte est difficile mais la vue vaut le détour !

# Les idées derrière ggplot2

Un graphique est composé

- d'un jeu de données : `ggplot`
- dont on veut représenter certains aspects : `aes`
- en utilisant une forme adaptée : `geom` et `stat`

La côte est difficile mais la vue vaut le détour !

# Liens utiles

Le site de référence

Elegant graphics for data analysis.

Une galerie inspirante

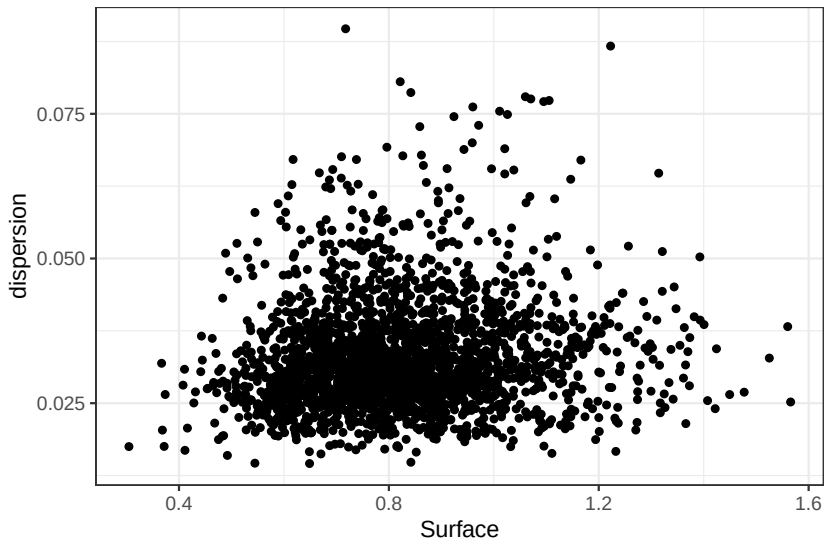
les extensions de ggplot2

Cheatsheet

# Plan

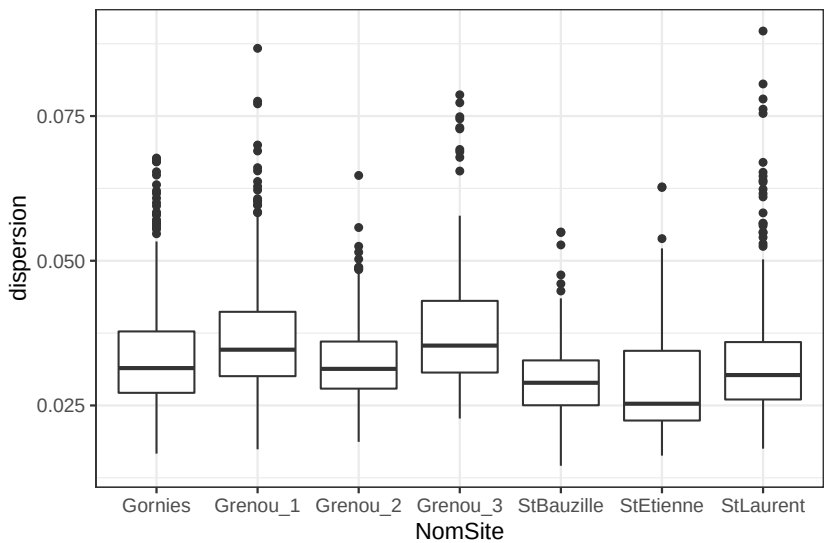
# Scatter plot

```
library(ggplot2)
ggplot(data = SamaresEq_disp) + aes( x = Surface, y = dispersion) + geom_point()
```



# Boîtes à moustaches

```
ggplot(data = SamaresEq_disp) +
  aes( x = NomSite, y = dispersion) +
  geom_boxplot()
```

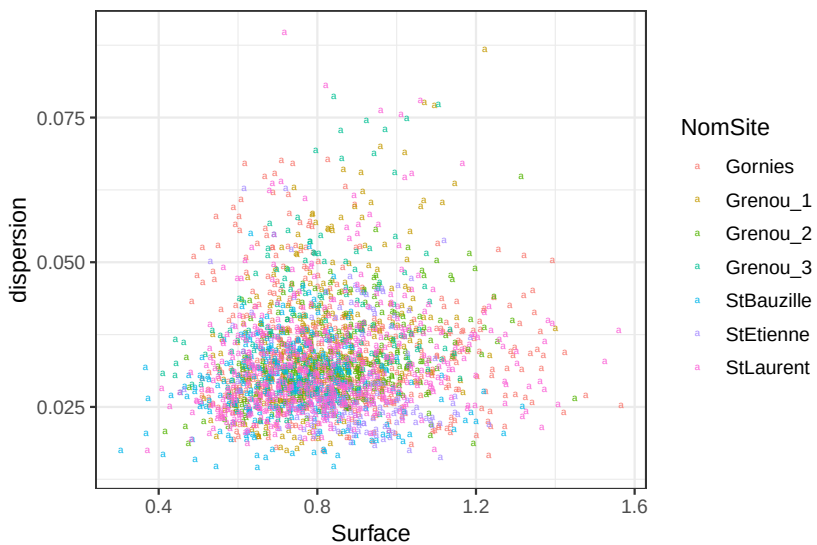


# Plan

# Ajouter des couleurs

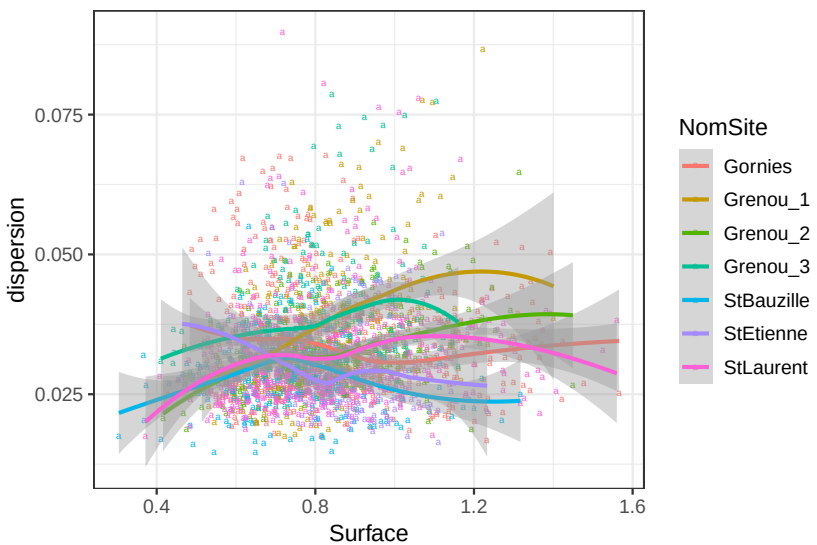
Ajouter de la couleur pour le Nom du Site et changer le symbole.

```
ggplot(data = SamaresEq_disp) +
  aes( x = Surface, y = dispersion, col = NomSite) +
  geom_point( shape = 'a')
```



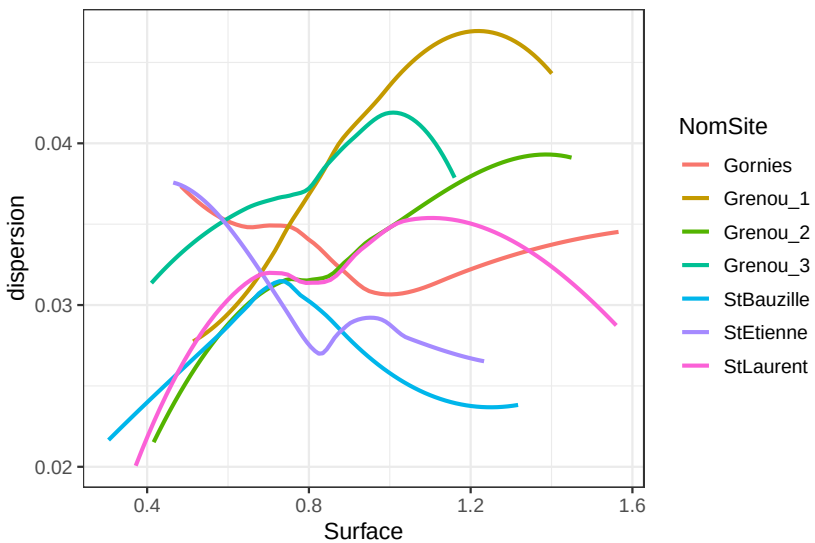
# Ajouter une tendance

```
ggplot(data = SamaresEq_disp) +
  aes( x = Surface, y = dispersion, col = NomSite) +
  geom_point( shape = 'a') +
  geom_smooth()
```



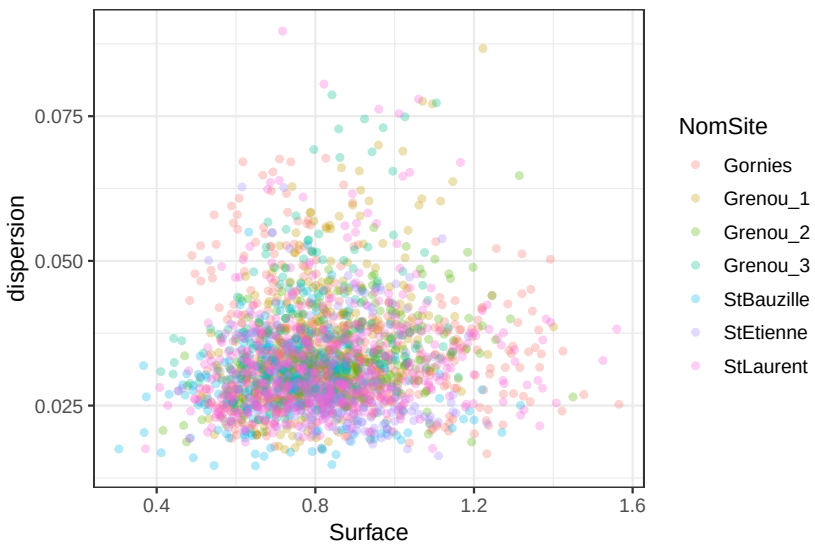
# Ajouter une tendance seule

```
ggplot(data = SamaresEq_disp) +
  aes( x = Surface, y = dispersion, col = NomSite) +
  geom_smooth(se = FALSE)
```



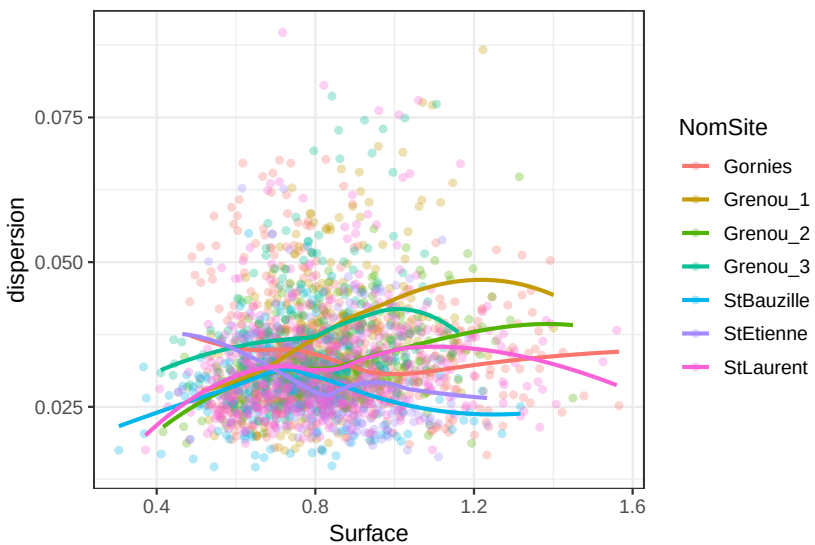
# Jouer avec la transparence

```
ggplot(data = SamaresEq_disp) +
  aes( x = Surface, y = dispersion, col = NomSite) +
  geom_point( alpha= 0.3)
```



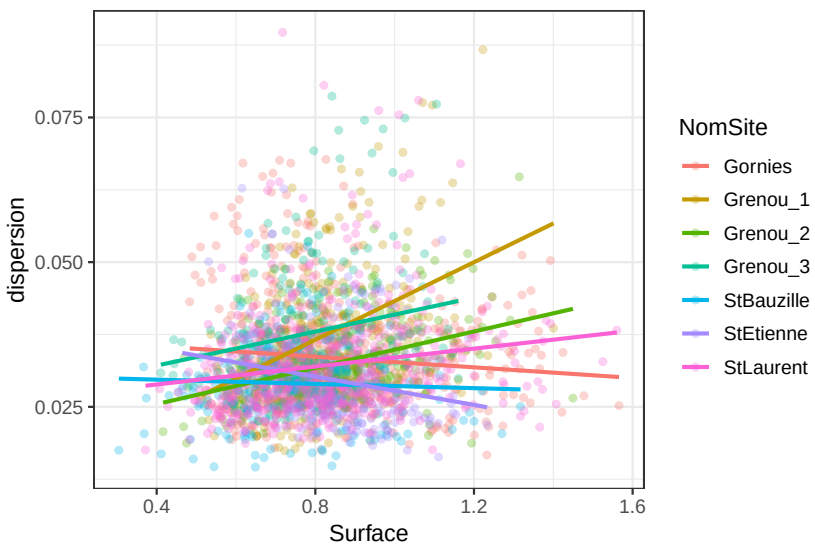
# Tous ensemble

```
ggplot(data = SamaresEq_disp) +
  aes( x = Surface, y = dispersion, col = NomSite) +
  geom_point( alpha= 0.3) +
  geom_smooth( se = FALSE)
```



# Avec une tendance linéaire

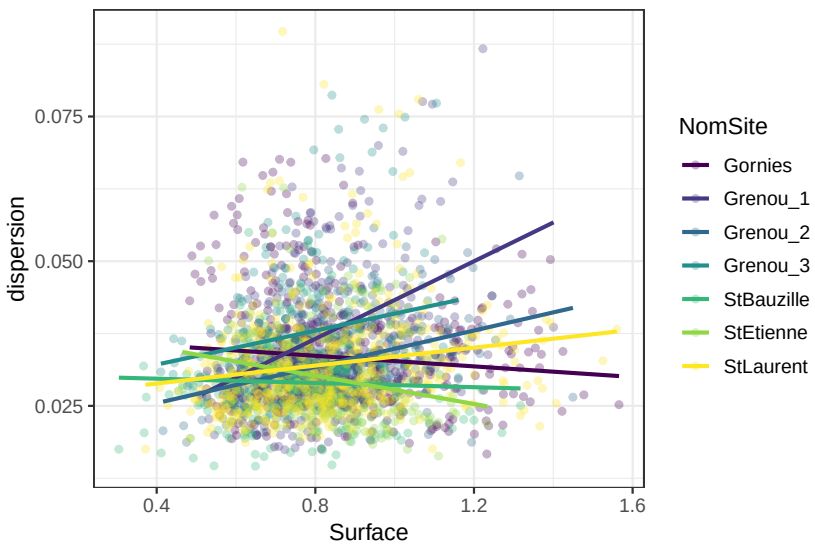
```
ggplot(data = SamaresEq_disp) +
  aes( x = Surface, y = dispersion, col = NomSite) +
  geom_point( alpha= 0.3) +
  geom_smooth(method = 'lm', se = FALSE)
```



# Plan

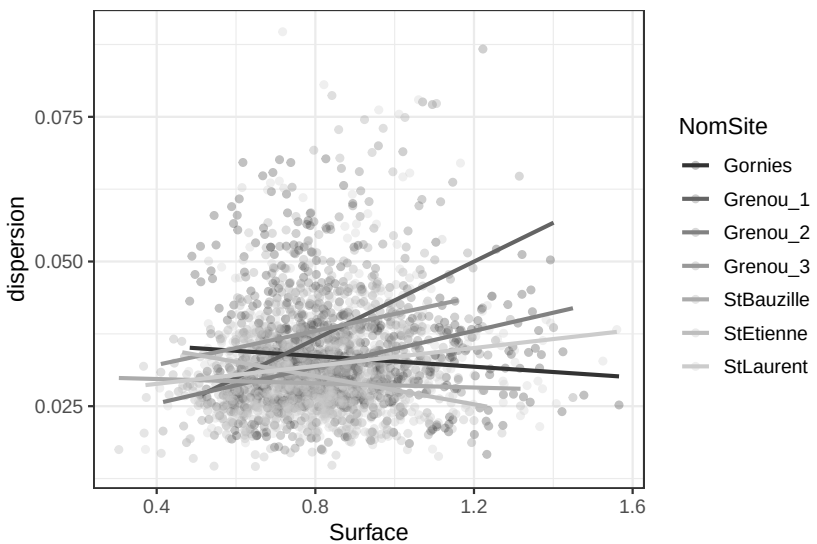
# Une palette compatible daltnonisme

```
ggplot(data = SamaresEq_disp) +
  aes( x = Surface, y = dispersion, col = NomSite) +
  geom_point( alpha= 0.3) +
  geom_smooth(method = 'lm', se = FALSE) +
  scale_color_viridis_d()
```



# En noir et blanc

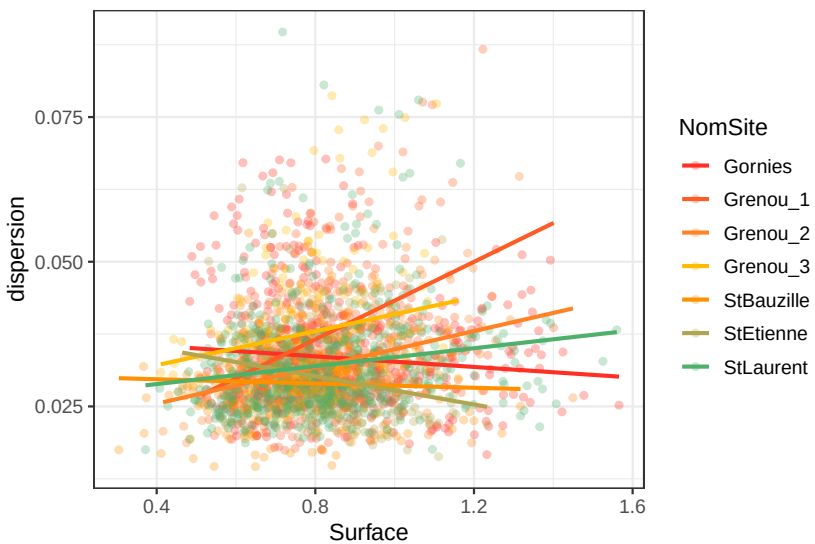
```
ggplot(data = SamaresEq_disp) +
  aes( x = Surface, y = dispersion, col = NomSite) +
  geom_point( alpha= 0.3) +
  geom_smooth(method = 'lm', se = FALSE) +
  scale_color_grey()
```



# Utiliser ses propres couleurs

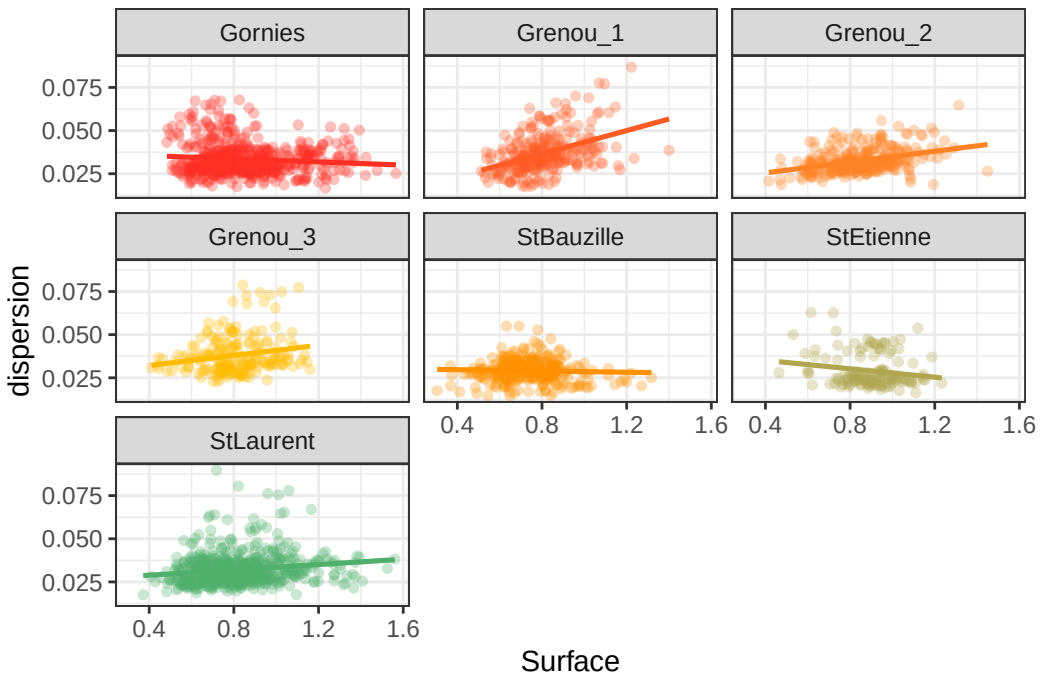
## Un site utile pour la spécification des couleurs

```
palette <- c('#FF2F24', '#FF5B24', '#FF8324', '#FFBb00', '#FF9100', '#B0A64F', '#4FB06C' )

ggplot(data = SamaresEq_disp) +
  aes( x = Surface, y = dispersion, col = NomSite) +
  geom_point( alpha= 0.3) +
  geom_smooth(method = 'lm', se = FALSE) +
  scale_color_manual(values = palette)
```



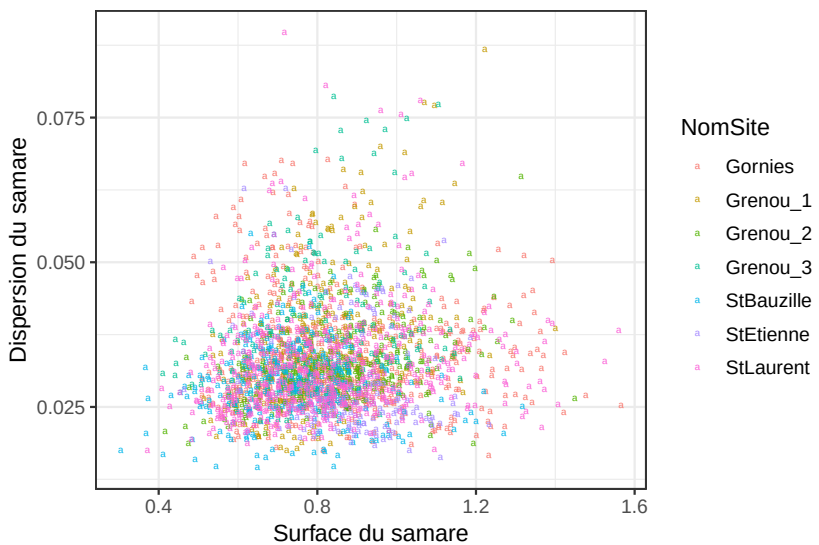
# Un graphique par site

```
ggplot(data = SamaresEq_disp) +
  aes( x = Surface, y = dispersion, col = NomSite) +
  facet_wrap(~NomSite) + theme( legend.position = 'none' ) +
  geom_point( alpha= 0.3) +
  geom_smooth(method = 'lm', se = FALSE) +
  scale_color_manual(values = palette)
```



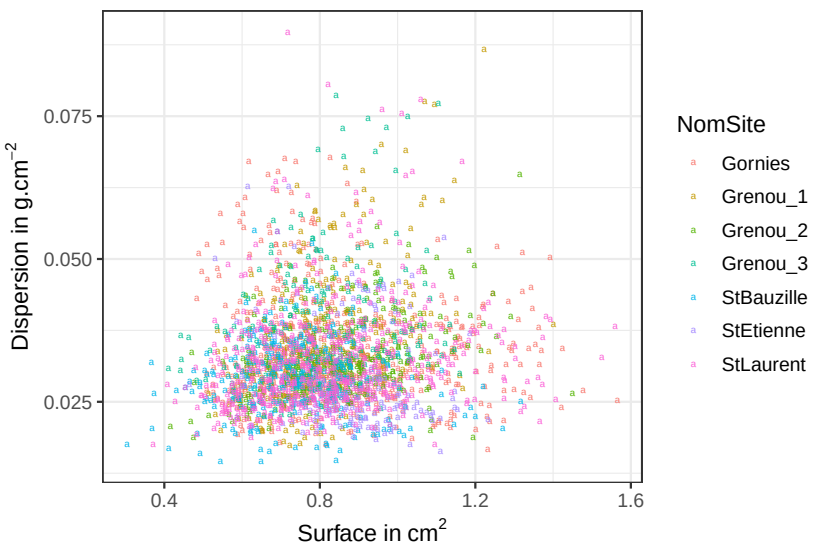
# Nommer les axes

```
ggplot(data = SamaresEq_disp) +
  aes( x = Surface, y = dispersion, col = NomSite) +
  geom_point( shape = 'a')  +
  labs(y = 'Dispersion du samare', x = 'Surface du samare')
```



# Intégrer des exposants et des indices dans les noms des axes

```
ggplot(data = SamaresEq_disp) +
  aes( x = Surface, y = dispersion, col = NomSite) +
  geom_point( shape = 'a')  +
  labs(y = expression ("Dispersion in"~g.cm^{-2}), x =  expression ("Surface in"~cm^{2}))
```



# Plan

# Exercice

Etude du rendement en fonction de l'indice de diversité,

Données issues de Kirwan et al. (2014)

- Charger le fichier Biomass_diversity.csv dans un tableau nommé `biomass`.

La variable `H` est l'indice de diversité de Shannon et `HARV_YIELD` le rendement de la parcelle.

- Quelle est le type de la variable `YEAR` ?
- Créer une variable qualitative (un facteur) `Year_fact`.
- Représenter la variabilité des rendements en fonction des pays, et des couples pays-années.
- Faire un graphique du rendement en fonction de l'indice de diversité
- Colorier les données selon l'année et indiquer par des symboles différents les différents pays.
- Ajuster une droite de régression par pays
- Modifier le nom des axes pour une publication en Français.

# Solution

Etude du rendement en fonction de l'indice de diversité,

Données issues de Kirwan et al. (2014)

- Charger le fichier Biomass_diversity.csv dans un tableau nommé `biomass`.

```
biomass <- readr::read_csv(file = '../../Datasets/Biomass_diversity.csv')
```

- Quelle est le type de la variable `YEAR` ?

```
biomass
```

```
## # A tibble: 864 x 7
##   COUNTRY    YEAR  PLOT     G     L HARV_YIELD     H
##   <chr>     <dbl> <dbl> <dbl> <dbl>      <dbl> <dbl>
## 1 Belgium    2003    12     1     0       2.69     1
## 2 Iceland_a  2003    12     1     0       2.82     1
## 3 Sweden_a   2003    12     1     0       2.61     1
## 4 Belgium    2004    12     1     0       3.44     1
## # ... with 860 more rows
```

```
class(biomass$YEAR)
```

```
## [1] "numeric"
```

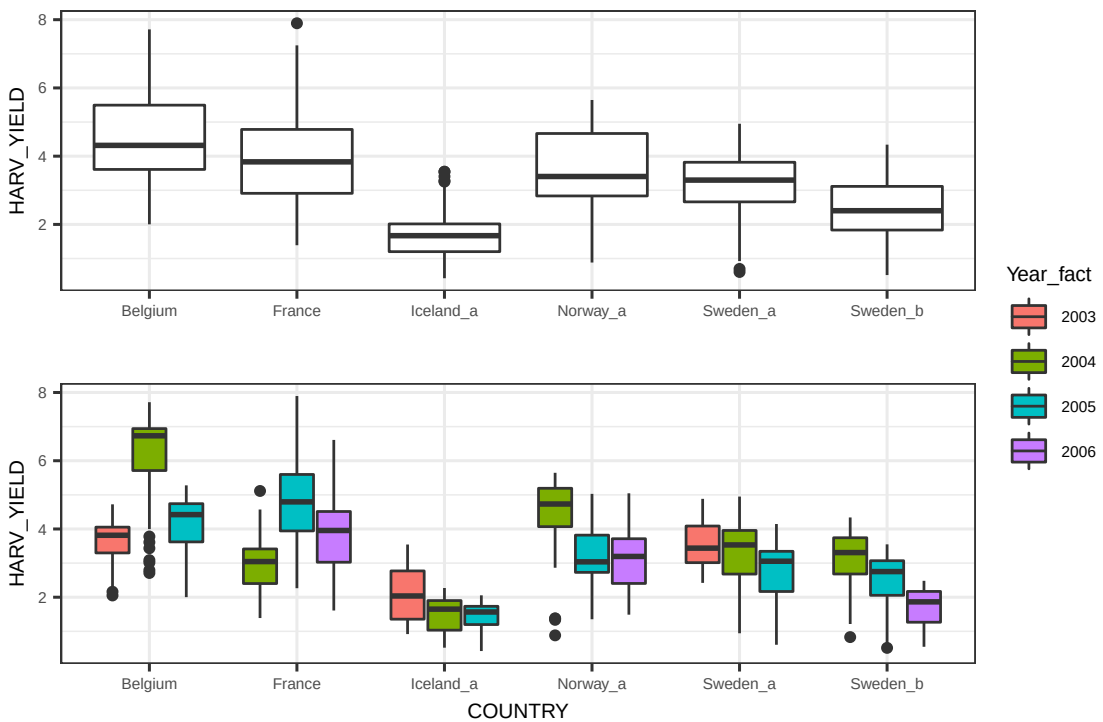
- Créer une variable qualitative `Year_fact`.

```
biomass %>%  mutate(Year_fact = as.factor(YEAR)) -> biomass
```

# Solution

- Représenter la variabilité des rendements en fonction des pays, et des couples pays-années.

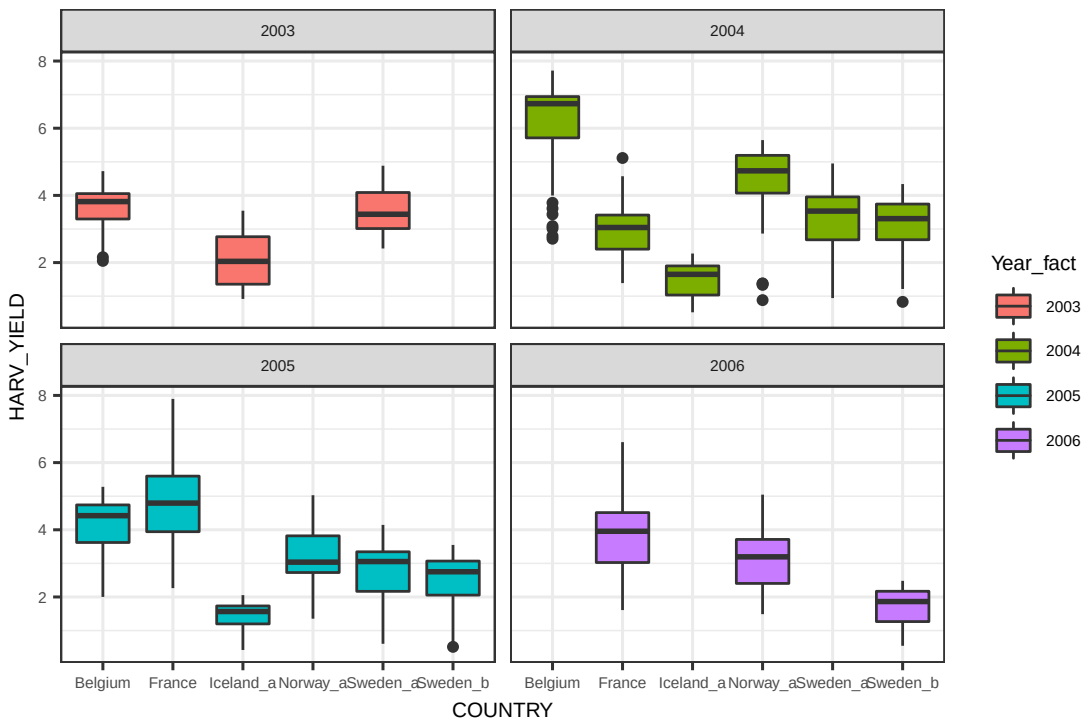
```
p1 <- biomass %>% ggplot() + aes(y = HARV_YIELD, x = COUNTRY ) + geom_boxplot() +
  xlab('') + theme( text = element_text(size=8))
p2 <- biomass %>% ggplot() +  aes(y = HARV_YIELD, x = COUNTRY) +
  geom_boxplot(aes(fill = Year_fact))  + theme( text = element_text(size=8))
library(ggpubr)
ggarrange(p1, p2, nrow= 2,  common.legend = TRUE, legend = 'right')
```



# Solution

- Représenter la variabilité des rendements en fonction des pays, et des couples pays-années.

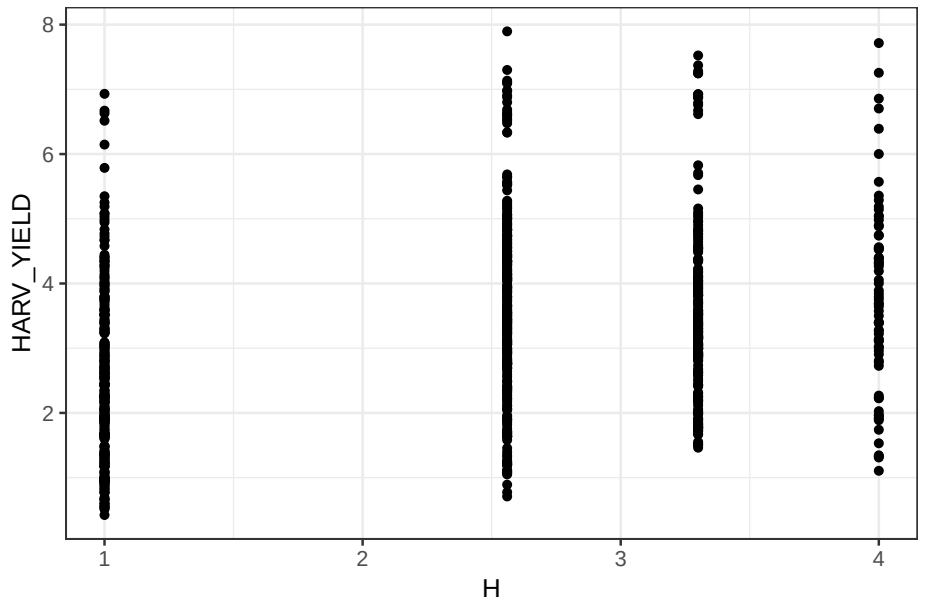
```
biomass %>% ggplot() +  aes(y = HARV_YIELD, x = COUNTRY) +
 facet_wrap(~Year_fact)  +
 geom_boxplot(aes(fill = Year_fact))  + theme( text = element_text(size=8))
```



# Solution

- Faire un graphique du rendement en fonction de l'indice de diversité

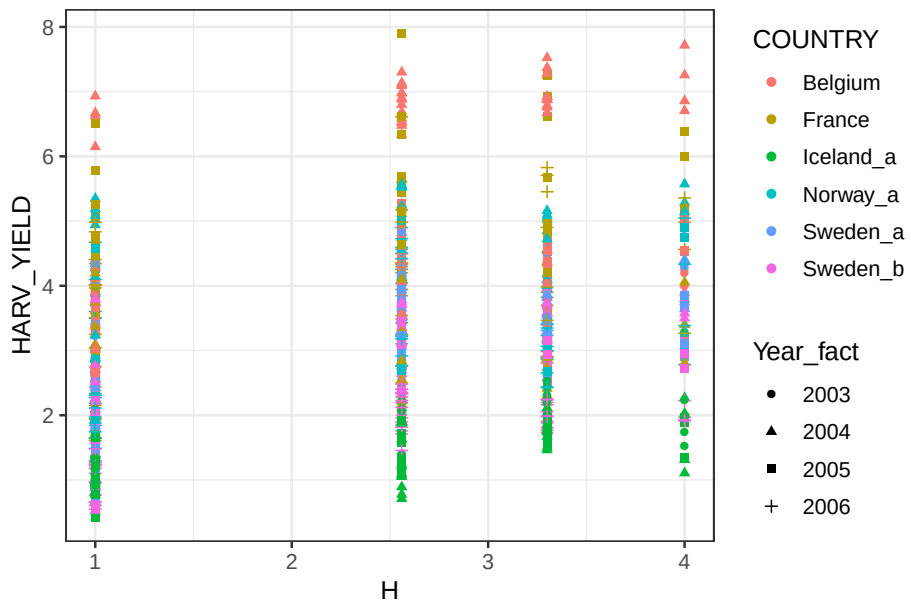
```
biomass %>% ggplot() + aes(y = HARV_YIELD, x = H ) + geom_point()
```



# Solution

- Colorer les données selon le pays et indiquer par des symboles différents les différentes années.

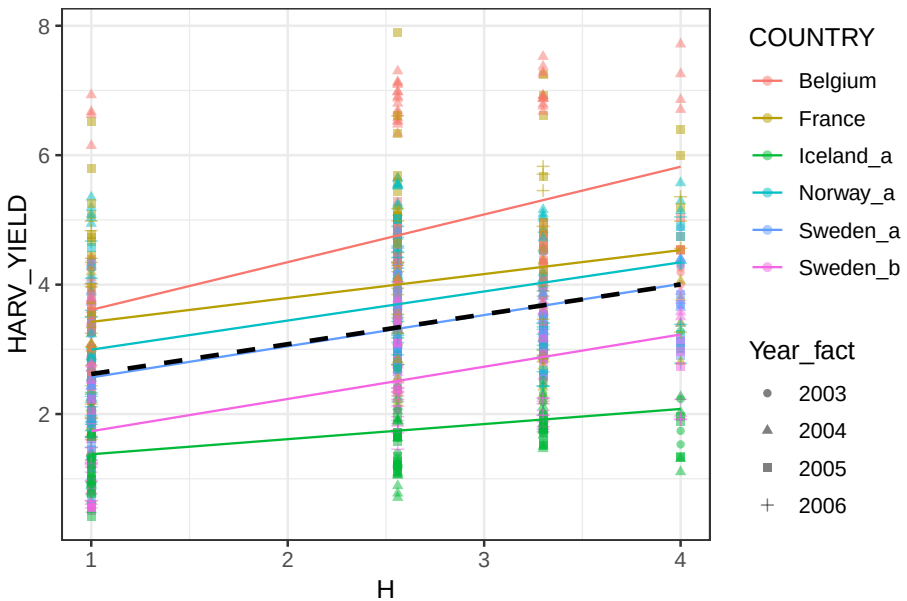
```
biomass %>% ggplot() +
  aes(y = HARV_YIELD, x = H ) +
  geom_point(aes(col = COUNTRY, shape = Year_fact))
```



# Solution

- Ajuster une droite de régression par pays

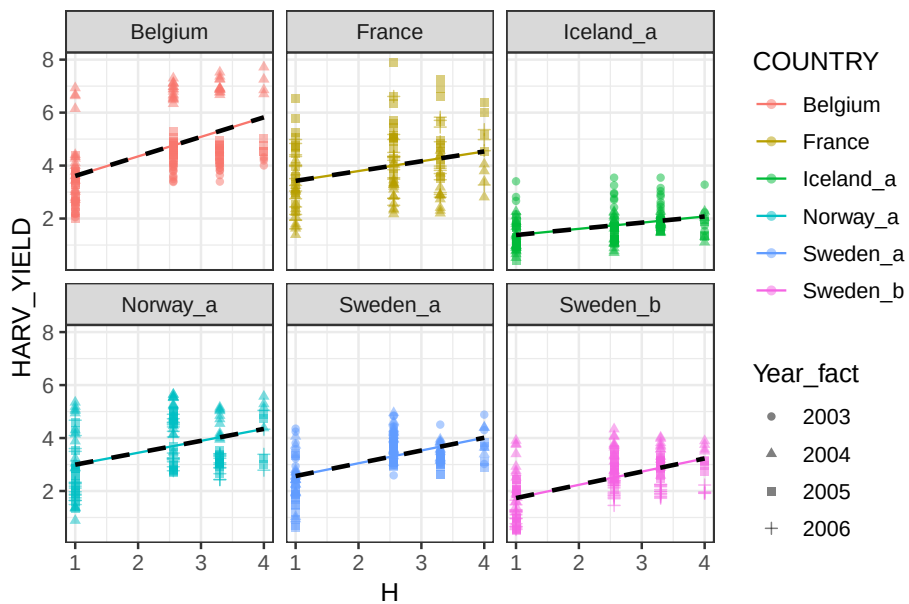
```
biomass %>% ggplot() +   aes(y = HARV_YIELD, x = H) +
  geom_point(  aes(col = COUNTRY, shape = Year_fact ), alpha = 0.5 ) +
  geom_smooth(method="lm", se= F, size = 0.5, aes(col = COUNTRY, group = COUNTRY)) +
  geom_smooth(method = 'lm',size = 1, linetype = 'dashed', colour = 'black', se = F)
```



# Solution

- Ajuster une droite de régression par pays

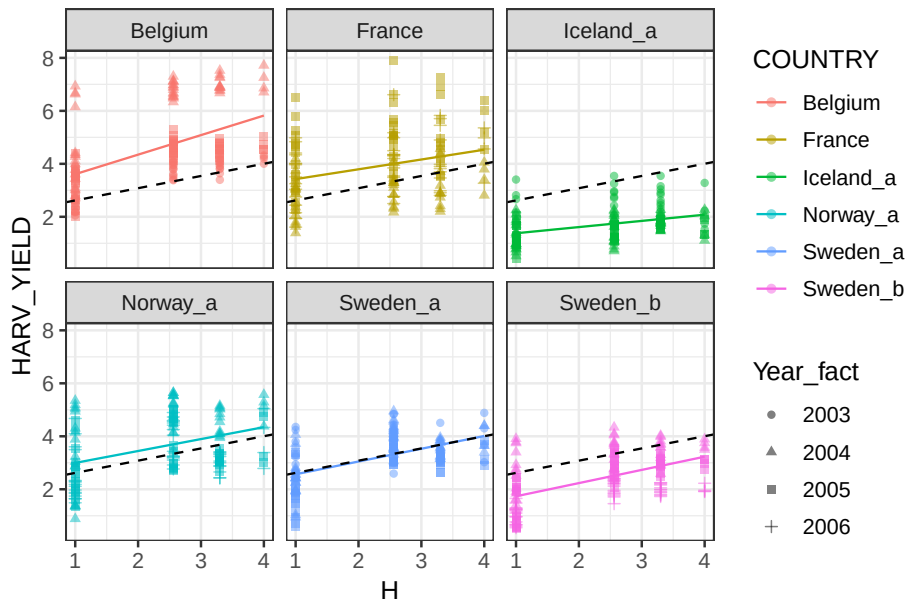
```
biomass %>% ggplot() +   aes(y = HARV_YIELD, x = H) +
  geom_point(  aes(col = COUNTRY, shape = Year_fact ), alpha = 0.5 ) + facet_wrap(~COUNTRY) +
  geom_smooth(method="lm", se= F, size = 0.5, aes(col = COUNTRY, group = COUNTRY)) -> p

p  +   geom_smooth(method = 'lm',size = 1, linetype = 'dashed', colour = 'black', se = F)
```



# Solution

- Ajuster une droite de régression par pays

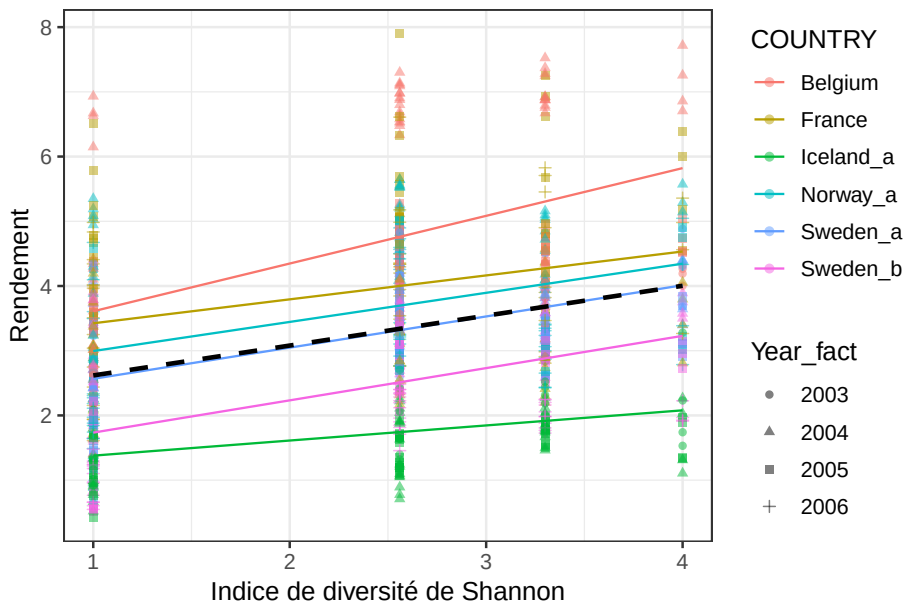
```
reg_coef <- coef(lm(HARV_YIELD ~ H, data = biomass))

p + geom_abline( intercept = reg_coef[1], slope = reg_coef[2], linetype = 'dashed', colour = 'b
```



# Solution

- Modifier le nom des axes pour une publication en Français.

```
biomass %>% ggplot() +   aes(y = HARV_YIELD, x = H) +
    geom_point(  aes(col = COUNTRY, shape = Year_fact ), alpha = 0.5 ) +
  geom_smooth(method="lm", se= F, size = 0.5, aes(col = COUNTRY, group = COUNTRY)) +
  geom_smooth(method = 'lm',size = 1, linetype = 'dashed', colour = 'black', se = F)  +
  labs( x = 'Indice de diversité de Shannon', y = 'Rendement')
```



# Plan

# Plan

# Test de comparaison de 2 moyennes

Question : Les poids des poulpes mâles et femelles sont-ils égaux ?

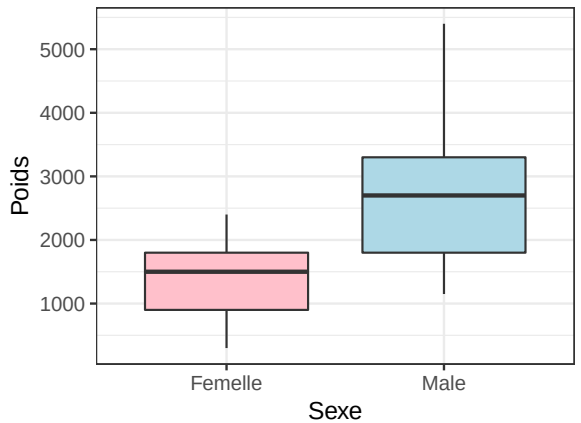
Importons et visualisons les données:

```
poulpe <- read.table("https://r-stat-sc-donnees.github.io/poulpe.csv", header=TRUE, sep=";")
summary(poulpe)
```

```
##      Poids           Sexe
##  Min.   : 300   Femelle:13
##  1st Qu.:1480   Male   :15
##  Median :1800
##  Mean   :2099
##  3rd Qu.:2750
##  Max.   :5400
```

# Visualisation des données

```
library(ggplot2)
poulpe %>% ggplot() + aes(x=Sexe,y=Poids) + geom_boxplot(fill=c("pink","lightblue"))
```




## Pour un graphe interactif en html:

```
library(plotly)
poulpe %>% ggplot() + aes(x=Sexe,y=Poids) + geom_boxplot(fill=c("pink","lightblue"))
ggplotly()
```

## Avec les lignes de code R:

```
boxplot(Poids ~ Sexe, col=c("pink","lightblue"), data=poulpe)
```

## Ou pour faire des graphes interactifs :

```
library(rAmCharts)
amBoxplot(Poids ~ Sexe, col=c("pink","lightblue"), data=poulpe)
```

# Comparaison de 2 moyennes: test de la normalité

A-t-on bien la normalité des poids pour les mâles et femelles ?

```
by(poulpe$Poids, poulpe$Sexe, shapiro.test)
```

```
## poulpe$Sexe: Femelle
##
##  Shapiro-Wilk normality test
##
## data:  dd[x, ]
## W = 0.97109, p-value = 0.9069
##
## ------------------------------------------------------------
## poulpe$Sexe: Male
##
##  Shapiro-Wilk normality test
##
## data:  dd[x, ]
## W = 0.93501, p-value = 0.3238
```

On accepte l'hypothèse de normalité des poids pour les femelles, et pour les mâles

# Comparaison de 2 moyennes : test d'égalite des variances

Quel test utiliser ? Celui avec variances égales ou inégales ?

```
var.test(Poids ~ Sexe, conf.level=.95, data=poulpe)
```

```
##
##  F test to compare two variances
##
## data:  Poids by Sexe
## F = 0.28833, num df = 12, denom df = 14, p-value = 0.03713
## alternative hypothesis: true ratio of variances is not equal to 1
## 95 percent confidence interval:
##  0.09452959 0.92444666
## sample estimates:
## ratio of variances
##          0.2883299
```

On rejette l'hypothèse d'égalité des variances $\Longrightarrow$ on considère que les variances ne sont pas égales

# Test de comparaison de 2 moyennes (suite et fin)

```
res <- t.test(Poids~Sexe, alternative="two.sided", conf.level=.95,
              var.equal=FALSE, data=poulpe)
res
```

```
##
##  Welch Two Sample t-test
##
## data:  Poids by Sexe
## t = -3.7496, df = 22.021, p-value = 0.001107
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -2010.624  -578.607
## sample estimates:
## mean in group Femelle    mean in group Male
##              1405.385              2700.000
```

On considère que les poids moyennes des mâles et femelles sont différents

Les mâles sont plus lourds (2700) que les femelles (1405.4)

# Plan

# Problématique et données

Question : Peut-on prévoir le maximum d'ozone en fonction de données climatiques (température, nébulosité, vitesse du vent, max d'ozone de la veille) ?

Importons et visualisons les données:

```
ozone <- read.table("https://r-stat-sc-donnees.github.io/ozone.txt",header=TRUE)
library(tidyverse)
ozone.m <- ozone %>% select(1:11)
ozone.m %>% select(1:4) %>% summary()
```

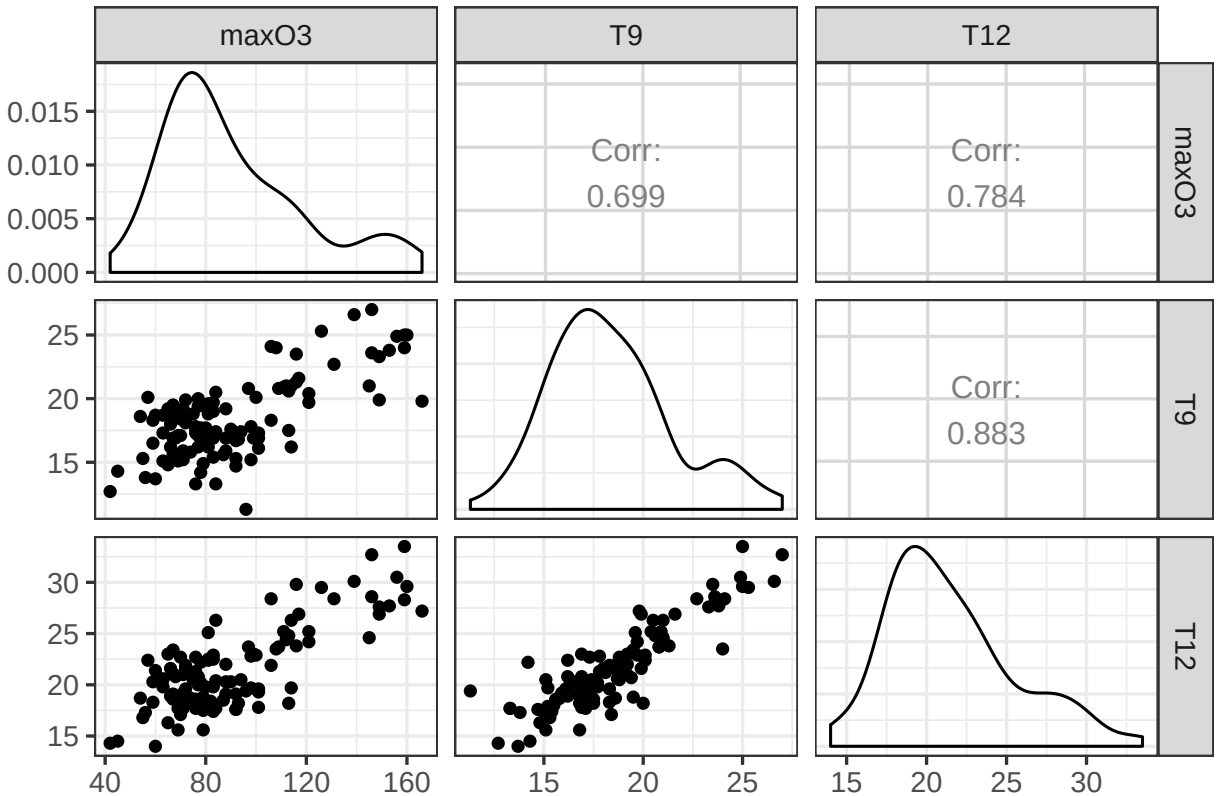
```
##      maxO3             T9              T12             T15       
##  Min.   : 42.00   Min.   :11.30   Min.   :14.00   Min.   :14.90  
##  1st Qu.: 70.75   1st Qu.:16.20   1st Qu.:18.60   1st Qu.:19.27  
##  Median : 81.50   Median :17.80   Median :20.55   Median :22.05  
##  Mean   : 90.30   Mean   :18.36   Mean   :21.53   Mean   :22.63  
##  3rd Qu.:106.00   3rd Qu.:19.93   3rd Qu.:23.55   3rd Qu.:25.40  
##  Max.   :166.00   Max.   :27.00   Max.   :33.50   Max.   :35.50
```

Avec les lignes de code R:

```
ozone <- read.table("https://r-stat-sc-donnees.github.io/ozone.txt",header=TRUE)
ozone.m <- ozone[,1:11]
summary(ozone.m[,1:4])
```

# Visualisation des liaisons par paires de variables

```
library(GGally)
ozone.m %>% select(1:3) %>% ggpairs()
```




Avec les lignes de code R :

```
pairs(ozone.m[,1:3])
```

# Construction du modèle complet

```
reg.mul <- lm(maxO3~., data=ozone.m)
summary(reg.mul)
```

```
## Call:
## lm(formula = maxO3 ~ ., data = ozone.m)
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 12.24442   13.47190   0.909   0.3656
## T9          -0.01901    1.12515  -0.017   0.9866
## T12          2.22115    1.43294   1.550   0.1243
## T15          0.55853    1.14464   0.488   0.6266
## Ne9         -2.18909    0.93824  -2.333   0.0216 *
## Ne12        -0.42102    1.36766  -0.308   0.7588
## Ne15         0.18373    1.00279   0.183   0.8550
## Vx9          0.94791    0.91228   1.039   0.3013
## Vx12         0.03120    1.05523   0.030   0.9765
## Vx15         0.41859    0.91568   0.457   0.6486
## maxO3v       0.35198    0.06289   5.597 1.88e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 14.36 on 101 degrees of freedom
## Multiple R-squared:  0.7638, Adjusted R-squared:  0.7405
## F-statistic: 32.67 on 10 and 101 DF,  p-value: < 2.2e-16
```

# Sélection de variables

```
library(FactoMineR)
select <- RegBest(ozone.m$maxO3, ozone.m[,2:11])
select$summary ; select$best
```

```
##                                 R2       Pvalue
## Model with 1 variable   0.6150674 1.512025e-24
## Model with 2 variables  0.7012408 2.541031e-29
## Model with 3 variables  0.7519764 1.457692e-32
## Model with 4 variables  0.7622198 1.763434e-32
## Model with 5 variables  0.7630603 1.449905e-31
## Model with 6 variables  0.7635768 1.130263e-30
## Model with 7 variables  0.7637610 8.556709e-30
## Model with 8 variables  0.7638390 6.076804e-29
## Model with 9 variables  0.7638407 4.066941e-28
## Model with 10 variables 0.7638413 2.545665e-27

##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  9.76225   11.10038   0.879    0.381
## T12          2.85308    0.48052   5.937 3.57e-08 ***
## Ne9         -3.02423    0.64342  -4.700 7.71e-06 ***
## maxO3v       0.37571    0.05801   6.477 2.85e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 14.23 on 108 degrees of freedom
## Multiple R-squared:  0.752,  Adjusted R-squared:  0.7451
## F-statistic: 109.1 on 3 and 108 DF,  p-value: < 2.2e-16
```

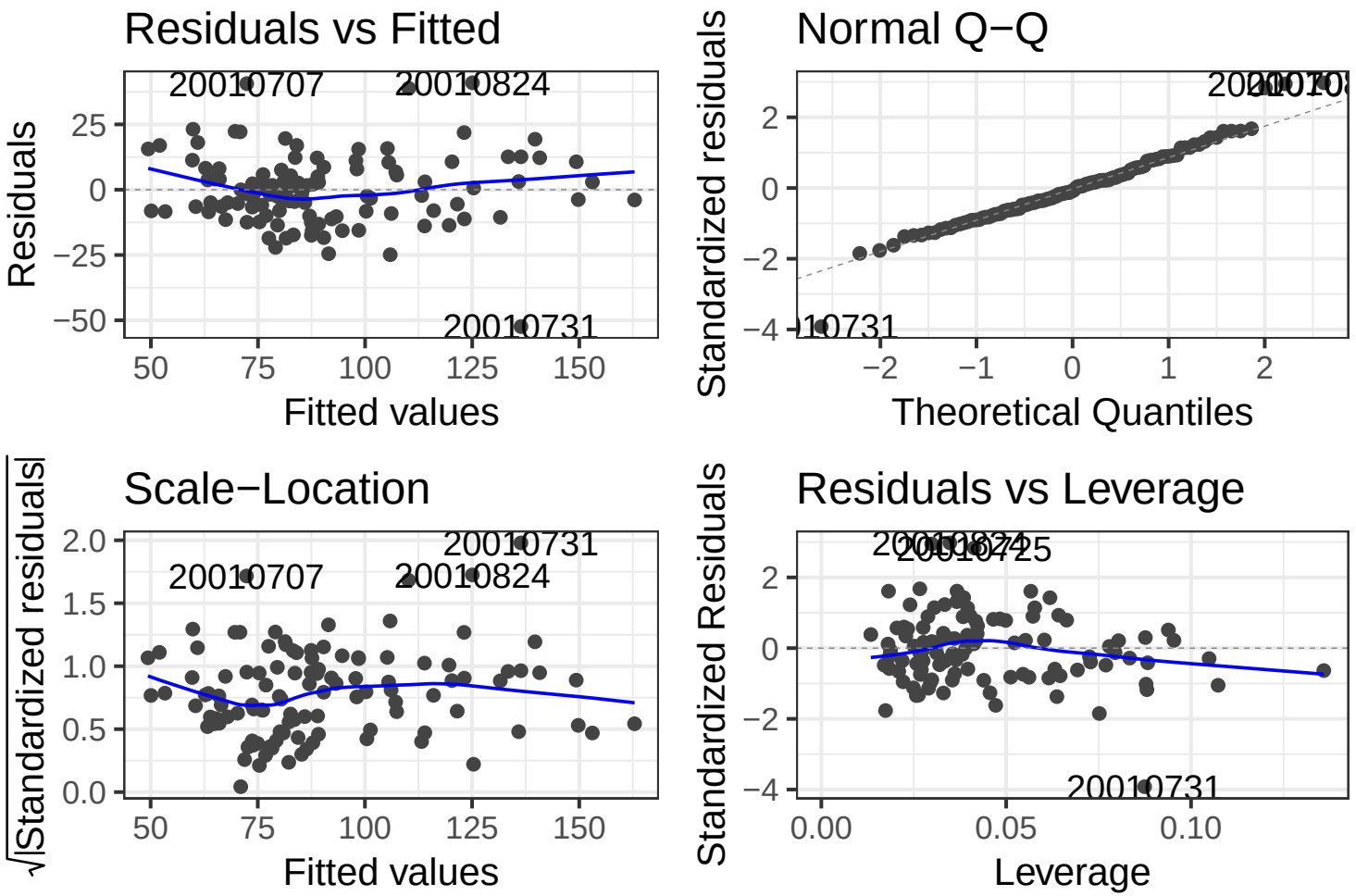
# Construction du modèle final

```
reg.fin <- lm(maxO3~T12+Ne9+Vx9+maxO3v, data=ozone.m)
summary(reg.fin)
```

```
##
## Call:
## lm(formula = maxO3 ~ T12 + Ne9 + Vx9 + maxO3v, data = ozone.m)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -52.396  -8.377  -1.086   7.951  40.933
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 12.63131   11.00088   1.148 0.253443
## T12          2.76409    0.47450   5.825 6.07e-08 ***
## Ne9         -2.51540    0.67585  -3.722 0.000317 ***
## Vx9          1.29286    0.60218   2.147 0.034055 *
## maxO3v       0.35483    0.05789   6.130 1.50e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 14 on 107 degrees of freedom
## Multiple R-squared:  0.7622, Adjusted R-squared:  0.7533
## F-statistic: 85.75 on 4 and 107 DF,  p-value: < 2.2e-16
```

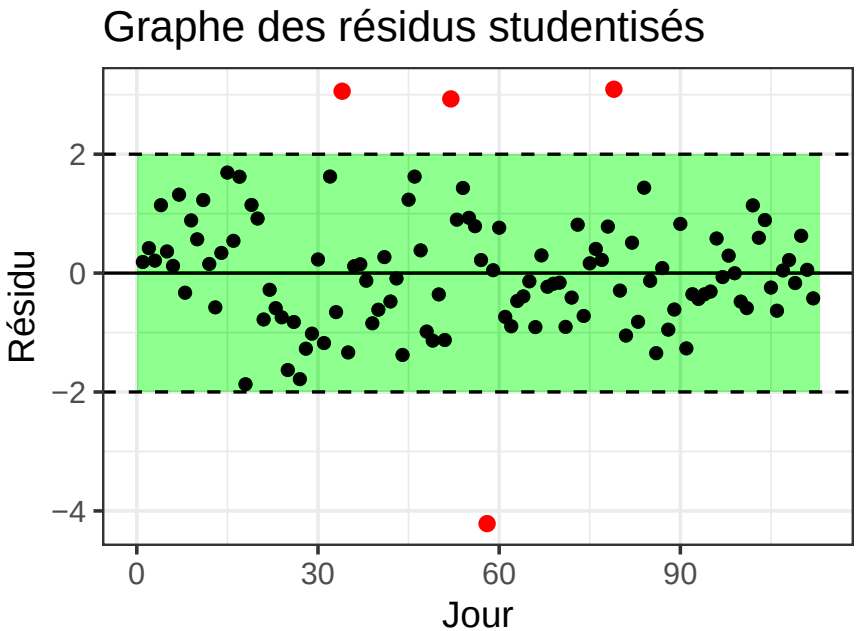
# Analyser les résidus

```
library(ggfortify)
autoplot(reg.fin)
```



# Analyser les résidus (suite)

```
residutib <- tibble(jour = 1:112, residu = rstudent(reg.fin))
residutib %>% ggplot() + aes(x=jour, y=residu) + geom_point() +
  labs(x="Jour", y="Résidu", title = "Graphe des résidus studentisés") +
  geom_abline(slope=0, intercept=c(-2,0,2), linetype=c(2,1,2)) +
  geom_rect(aes(xmin=0, xmax=113, ymin=-2, ymax=2), alpha=0.002,fill="green") +
  geom_point(data = residutib %>% filter(abs(residu)>2), cex=2, col="red")
```




## Avec les lignes de code R :

```
plot(residu,pch=15,cex=.5,ylab="Résidus",main="Graphe des résidus studentisés",ylim=c(-3,3))
abline(h=c(-2,0,2),lty=c(2,1,2))
```

# Prévoir une nouvelle valeur

Et comment prédire le maximum d'ozone pour de nouvelles valeurs ?

```
xnew <- matrix(c(19,8,2.05,70),nrow=1)
colnames(xnew) <- c("T12","Ne9","Vx9","maxO3v")
xnew <- as.data.frame(xnew)
predict(reg.fin,xnew,interval="pred")
```

```
##        fit      lwr      upr
## 1 72.51437 43.80638 101.2224
```

# Plan

# Problématique et données

Question : Y a-t-il un effet de la pluie et du vent sur le maximum d'ozone ?
Y a-t-il un effet de l'interaction de ces deux facteurs ?

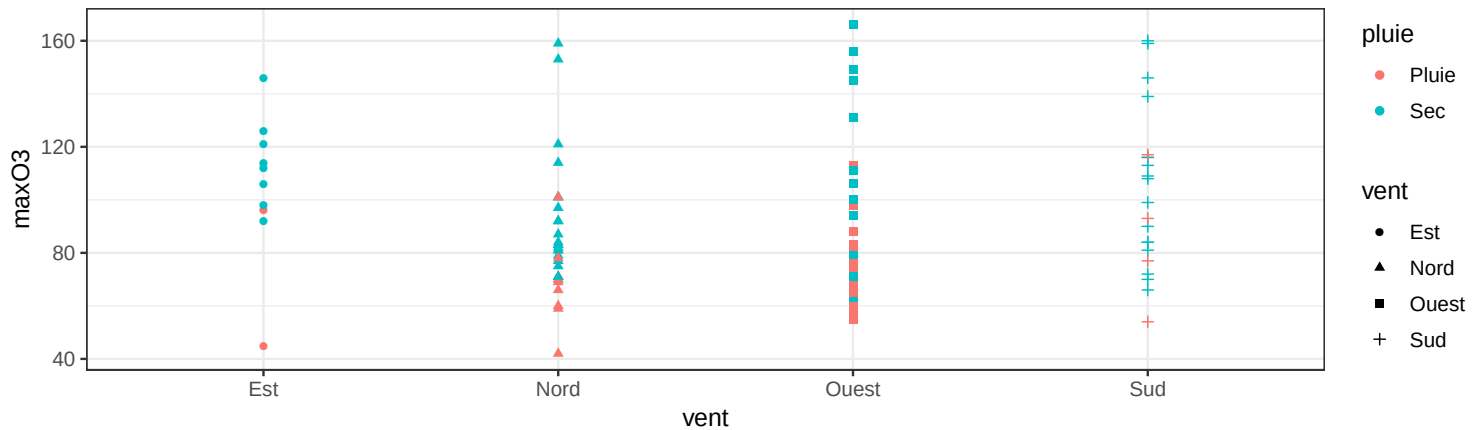
Importation des données:

```
ozone <- read.table("https://r-stat-sc-donnees.github.io/ozone.txt",header=TRUE)
summary(ozone[,c("maxO3","vent","pluie")])
```

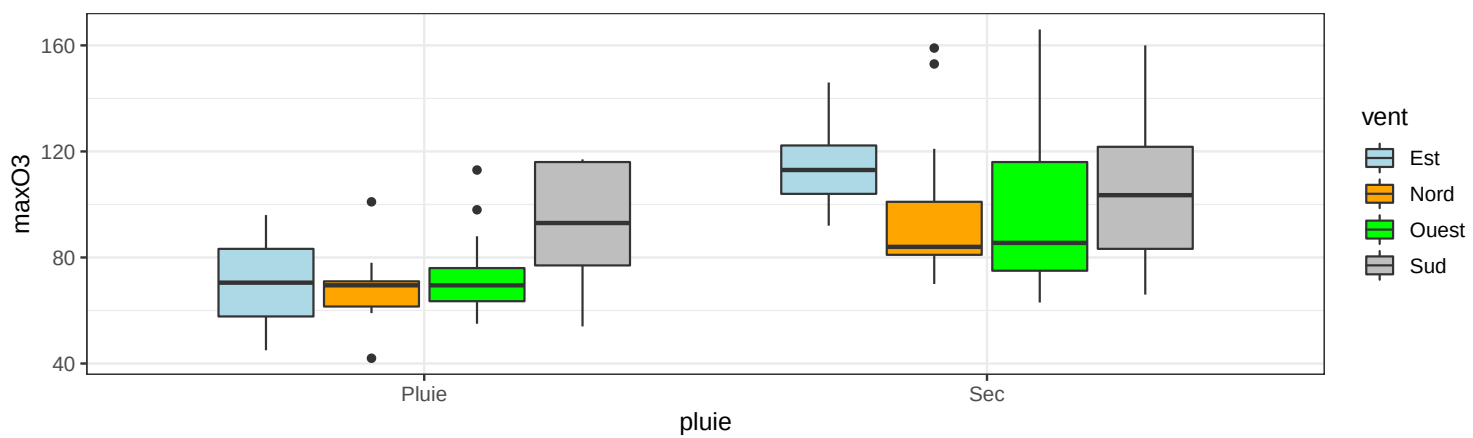
```
##      maxO3           vent      pluie
##  Min.   : 42.00   Est  :10   Pluie:43
##  1st Qu.: 70.75   Nord :31   Sec  :69
##  Median : 81.50   Ouest:50
##  Mean   : 90.30   Sud  :21
##  3rd Qu.:106.00
##  Max.   :166.00
```

# Visualisation des données avec ggplot2

```
library(ggplot2)
ozone %>% ggplot() + aes(y=maxO3, x=vent) + geom_point(aes(col=pluie, shape=vent))
```

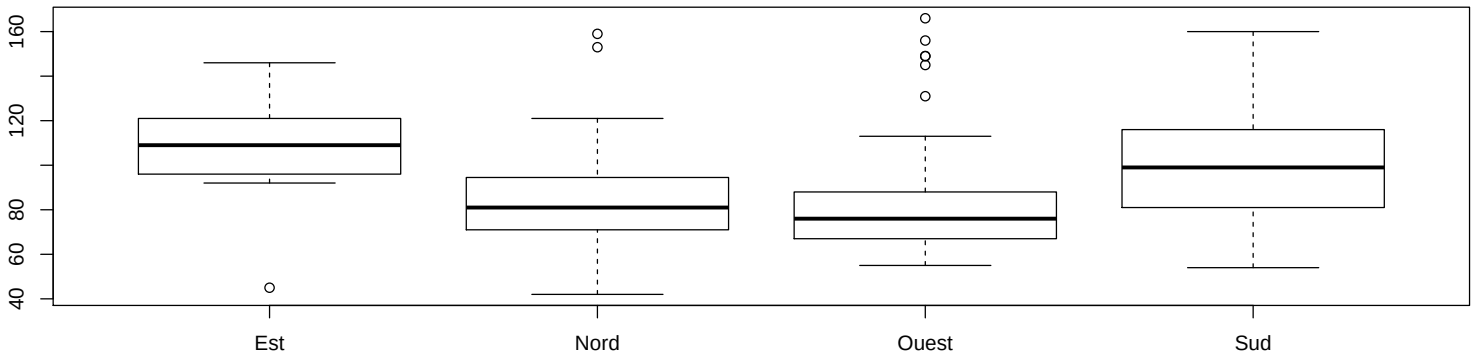



```
ozone %>% ggplot() + aes(pluie, maxO3, dodge=vent) + geom_boxplot(aes(fill=vent)) +
  scale_fill_manual(values=c("lightblue","orange","green","grey"))
```

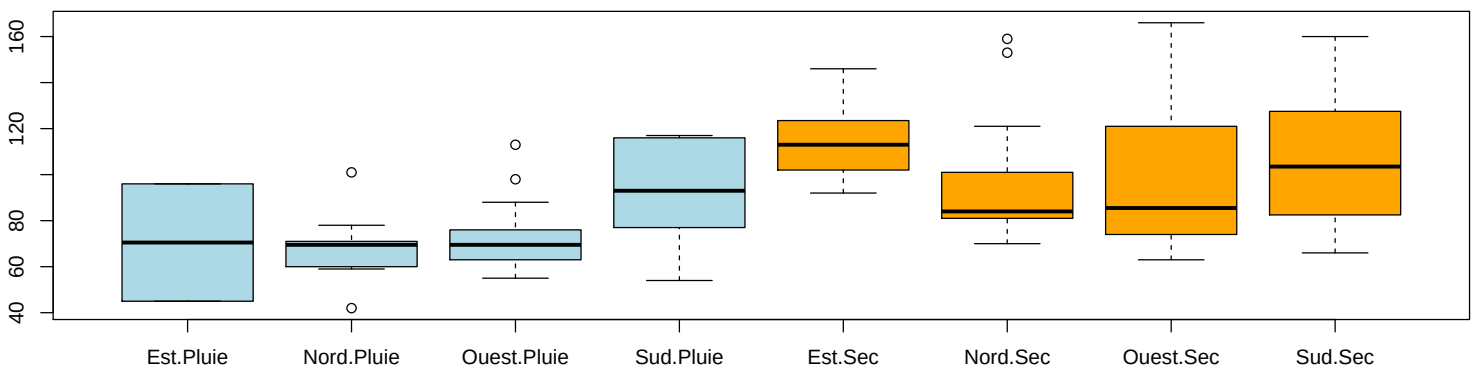


# Visualisation des données en R

```
boxplot(maxO3~vent, data = ozone)
```

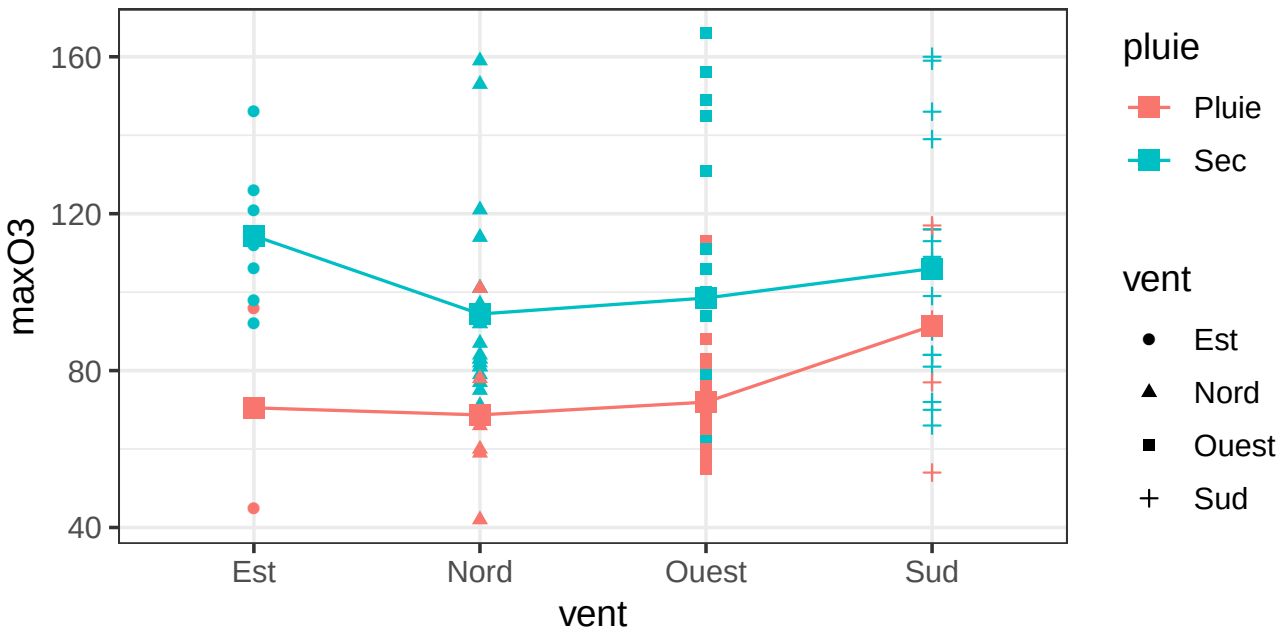



```
boxplot(maxO3~vent*pluie, data = ozone, col=c(rep("Lightblue",4),rep("orange",4)))
```



# Visualisation de l'interaction

```
ozone %>% ggplot() + aes(x = vent, y = maxO3, group = pluie) +
  geom_point(aes(color = pluie, shape=vent)) +
  stat_summary(fun.y = mean, geom = "point", size=3, shape=15,aes(color = pluie)) +
  stat_summary(fun.y = mean, geom = "line", aes(color = pluie))
```




Visualiser l'autre graphe d'interaction (une ligne brisée par direction du vent) et conserver le graphe le plus explicite

# Construction du modèle avec interaction

```
library(FactoMineR)
mod.interaction <- AovSum(maxO3 ~ vent + pluie + vent:pluie, data=ozone)
mod.interaction$Ftest
```

```
##                SS  df      MS F value    Pr(>F)
## vent         3227   3  1075.6  1.7633    0.1588
## pluie       10996   1 10996.5 18.0271 4.749e-05 ***
## vent:pluie   1006   3   335.5  0.5500    0.6493
## Residuals   63440 104   610.0
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

On accepte l'hypothèse qu'il n'y a pas d'interaction car la probabilité critique (0.649) est supérieure à 5%

Nécessité de recontruire un modèle sans interaction

# Choix d'un sous-modèle

```
modele <- AovSum(maxO3 ~ vent + pluie,data=ozone)
modele$Ftest
```

```
##              SS  df      MS F value    Pr(>F)
## vent       3791   3  1263.8  2.0982    0.1048
## pluie     16159   1 16159.4 26.8295 1.052e-06 ***
## Residuals 64446 107   602.3
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Pas d'effet vent $\Longrightarrow$ reconstruire un modèle sans l'effet du vent

# Estimation et interprétation des coefficients

```
modele <- AovSum(maxO3 ~ pluie, data=ozone)
modele$Ftest
```

```
##              SS  df      MS F value    Pr(>F)
## pluie     19954   1 19954.2  32.166 1.157e-07 ***
## Residuals 68238 110   620.3
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
modele$Ttest
```

```
##                Estimate Std. Error   t value     Pr(>|t|)
## (Intercept)    87.11796   2.419555 36.005777 1.066479e-62
## pluie - Pluie -13.72262   2.419555 -5.671545 1.156980e-07
## pluie - Sec    13.72262   2.419555  5.671545 1.156980e-07
```

# Plan

# Plan

# Problématique et données

## Importation des données:

```
decath <- read.table("https://r-stat-sc-donnees.github.io/decathlon.csv",
                     sep=";",dec=".", header=TRUE, row.names=1, check.names=FALSE)
```

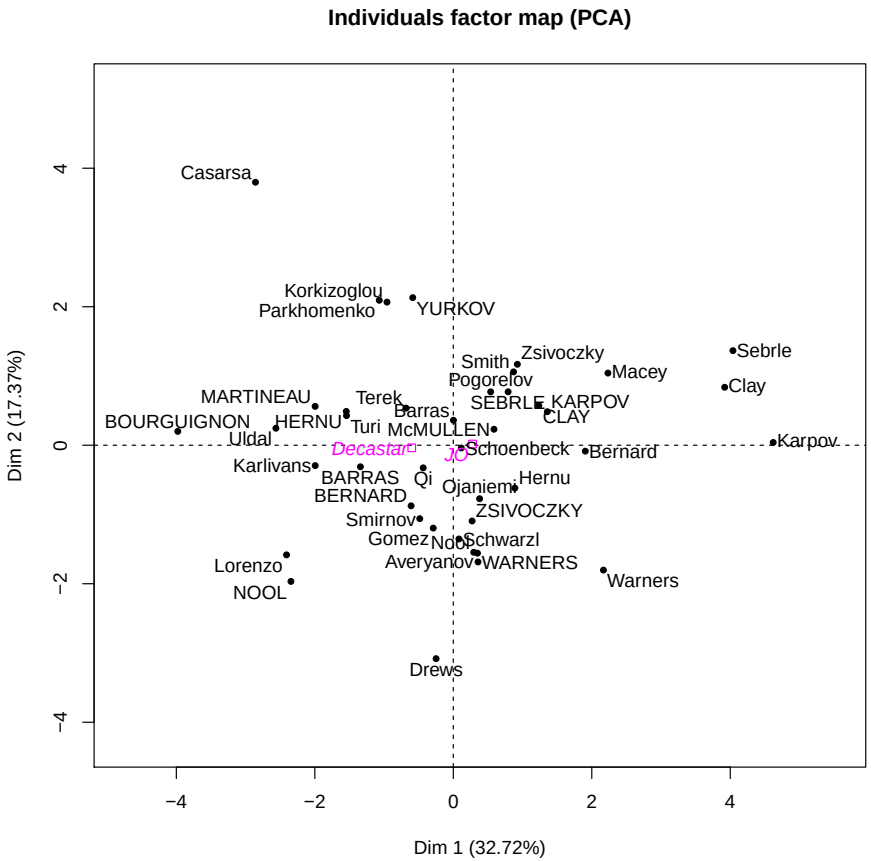
## ACP par les lignes de commande :

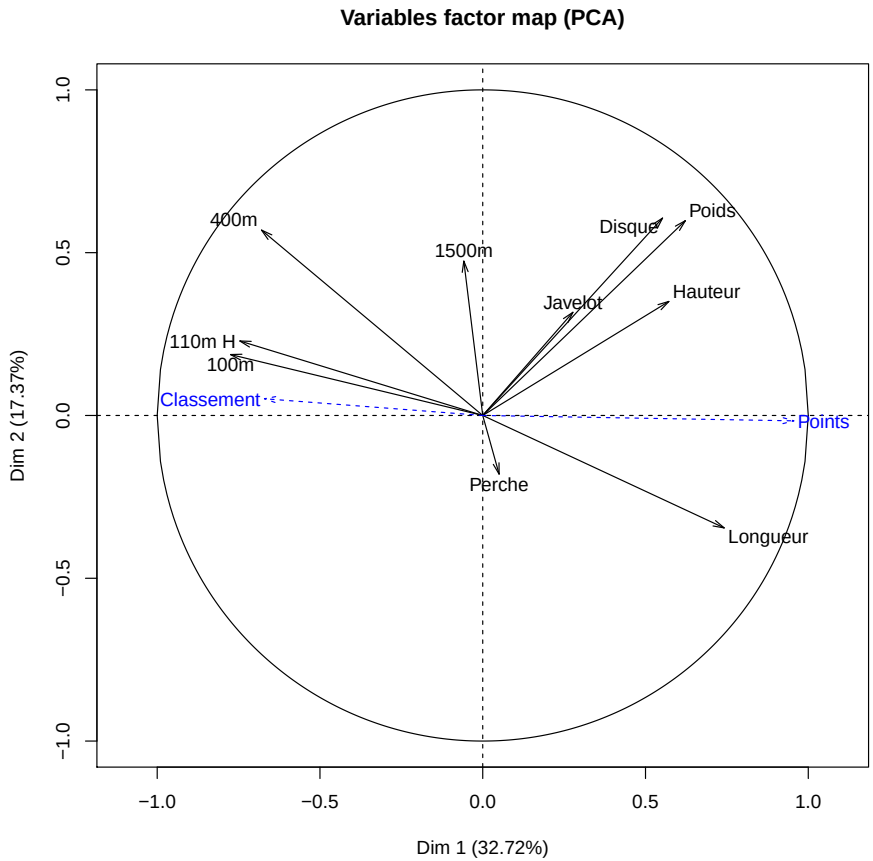
```
library(FactoMineR)
res.pca <- PCA(decath, quanti.sup=11:12, quali.sup=13)
```

## ACP par un menu déroulant et pour des graphes interactifs :

```
library(Factoshiny)
res <- PCAshiny(decath)
```

# Graphes des individus et des variables

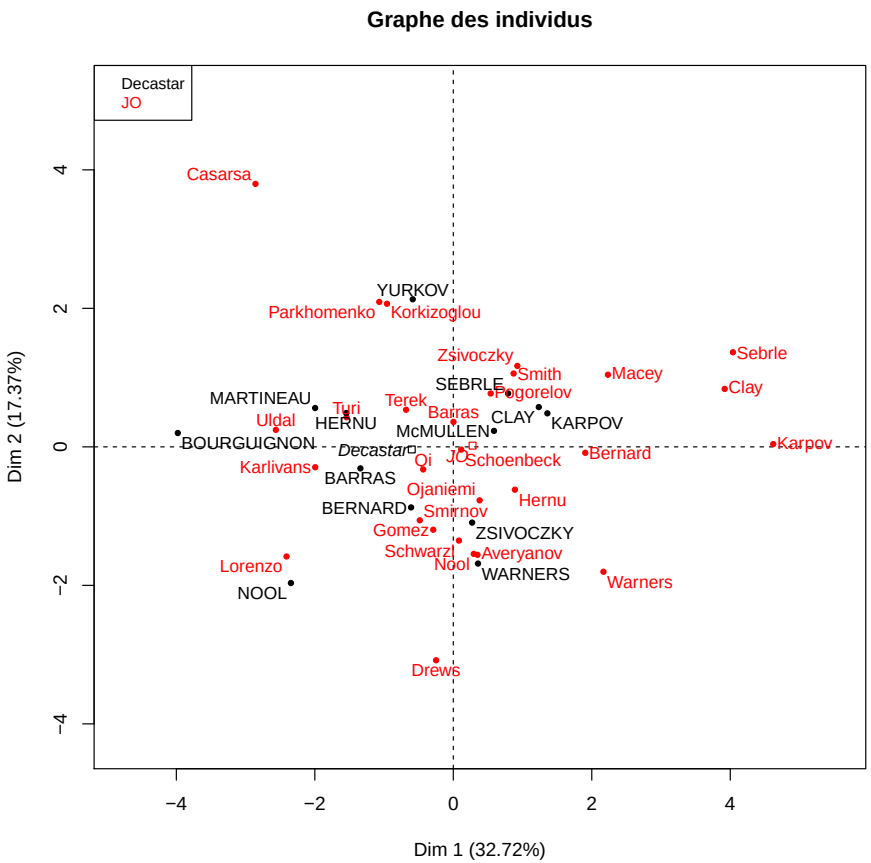


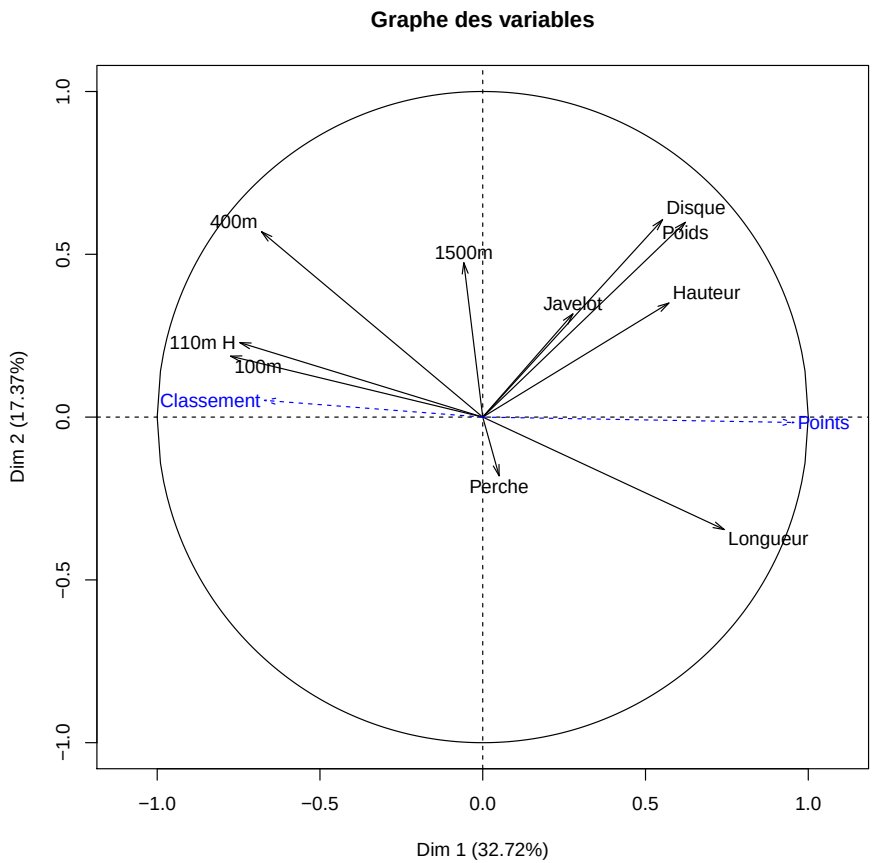



# Graphes des individus et des variables

## Possibilité de colorier les individus en fonction d'une variable qualitative :

```
plot(res.pca,habillage=13, cex=0.9, title="Graphe des individus")
plot(res.pca,choix="var", title="Graphe des variables")
```



# Résultats

```
summary(res.pca, ncp=2)
```

```
## Call:
## PCA(X = decath, quanti.sup = 11:12, quali.sup = 13, graph = FALSE)
##
##
## Eigenvalues
##                        Dim.1   Dim.2   Dim.3   Dim.4   Dim.5   Dim.6
## Variance               3.272   1.737   1.405   1.057   0.685   0.599
## % of var.             32.719  17.371  14.049  10.569   6.848   5.993
## Cumulative % of var.  32.719  50.090  64.140  74.708  81.556  87.548
##                        Dim.7   Dim.8   Dim.9  Dim.10
## Variance               0.451   0.397   0.215   0.182
## % of var.              4.512   3.969   2.148   1.822
## Cumulative % of var.  92.061  96.030  98.178 100.000
##
## Individuals (the 10 first)
##                Dist    Dim.1    ctr   cos2    Dim.2    ctr   cos2
## Sebrle     |  4.843 |  4.038 12.158  0.695 |  1.366  2.619  0.080 |
## Clay       |  4.647 |  3.919 11.451  0.711 |  0.837  0.984  0.032 |
## Karpov     |  5.006 |  4.620 15.911  0.852 |  0.040  0.002  0.000 |
## Macey      |  3.434 |  2.233  3.719  0.423 |  1.042  1.524  0.092 |
## Warners    |  2.979 |  2.168  3.505  0.530 | -1.803  4.565  0.366 |
## Zsivoczky  |  2.566 |  0.925  0.638  0.130 |  1.169  1.918  0.207 |
## Hernu      |  1.824 |  0.889  0.589  0.238 | -0.618  0.537  0.115 |
## Nool       |  3.098 |  0.295  0.065  0.009 | -1.546  3.354  0.249 |
## Bernard    |  2.827 |  1.906  2.709  0.455 | -0.086  0.010  0.001 |
## Schwarzl   |  1.971 |  0.081  0.005  0.002 | -1.353  2.572  0.472 |
```

# Résultats (suite)

```
summary(res.pca, ncp=2)
```

```
## Variables
##              Dim.1    ctr   cos2    Dim.2    ctr   cos2
## 100m       | -0.775 18.344  0.600 |  0.187  2.016  0.035 |
## Longueur   |  0.742 16.822  0.550 | -0.345  6.869  0.119 |
## Poids      |  0.623 11.844  0.388 |  0.598 20.607  0.358 |
## Hauteur    |  0.572  9.998  0.327 |  0.350  7.064  0.123 |
## 400m       | -0.680 14.116  0.462 |  0.569 18.666  0.324 |
## 110m H     | -0.746 17.020  0.557 |  0.229  3.013  0.052 |
## Disque     |  0.552  9.328  0.305 |  0.606 21.162  0.368 |
## Perche     |  0.050  0.077  0.003 | -0.180  1.873  0.033 |
## Javelot    |  0.277  2.347  0.077 |  0.317  5.784  0.100 |
## 1500m      | -0.058  0.103  0.003 |  0.474 12.946  0.225 |
##
## Supplementary continuous variables
##              Dim.1   cos2    Dim.2   cos2
## Classement | -0.671  0.450 |  0.051  0.003 |
## Points     |  0.956  0.914 | -0.017  0.000 |
##
## Supplementary categories
##               Dist    Dim.1   cos2 v.test    Dim.2   cos2 v.test
## Decastar   |  0.946 | -0.600  0.403 -1.430 | -0.038  0.002 -0.123 |
## JO         |  0.439 |  0.279  0.403  1.430 |  0.017  0.002  0.123 |
```

# Description des dimensions

```
dimdesc(res.pca, axes=1:2)
```

```
## $Dim.1
## $Dim.1$quanti
##             correlation      p.value
## Points        0.9561543 2.099191e-22
## Longueur      0.7418997 2.849886e-08
## Poids         0.6225026 1.388321e-05
## Hauteur       0.5719453 9.362285e-05
## Disque        0.5524665 1.802220e-04
## Classement   -0.6705104 1.616348e-06
## 400m         -0.6796099 1.028175e-06
## 110m H       -0.7462453 2.136962e-08
## 100m         -0.7747198 2.778467e-09
##
##
## $Dim.2
## $Dim.2$quanti
##           correlation      p.value
## Disque      0.6063134 2.650745e-05
## Poids       0.5983033 3.603567e-05
## 400m        0.5694378 1.020941e-04
## 1500m       0.4742238 1.734405e-03
## Hauteur     0.3502936 2.475025e-02
## Javelot     0.3169891 4.344974e-02
## Longueur   -0.3454213 2.696969e-02
```

# Plan

# Description des données

Le jeu de données croise 700 prélevés décrits par les pollens de 31 espèces d'arbres. Des variables climatiques ont été mesurées : température moyenne du mois le plus froid (mtco, mean temperature of the coldest month); température moyenne du mois le plus chaud (mtwa, mean temperature of the warmest month); the growing degree-days (gdd5, the sum of daily temperatures) above 5°C; the ratio of actual evapotranspiration to potential evapotranspiration (e_pe); précipitation annuelle (pann); température moyenne annuelle (tann).

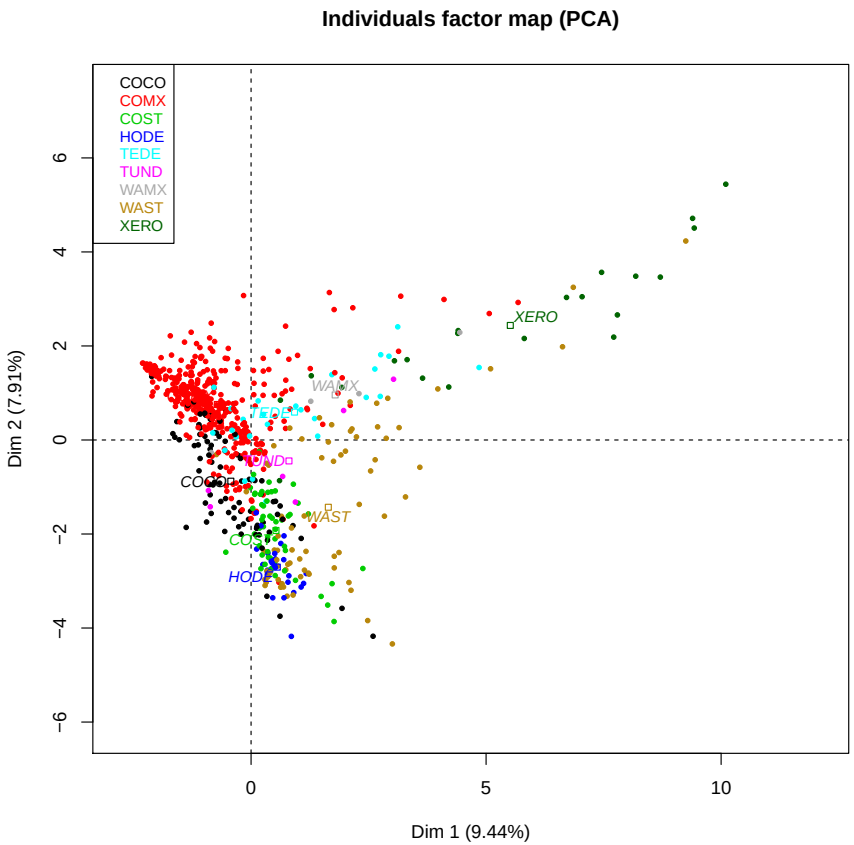
Les 700 relevés proviennent de 9 biomes différents : COCO (cool conifer forest), COMX (cool mixed forest), COST (cool steppes), HODE (hot desert), TEDE (temperate deciduous forest), TUND (tundra), WAMX (warm mixed broad-leaved forest), WAST (warm steppes), XERO (xerophytic scrubs)

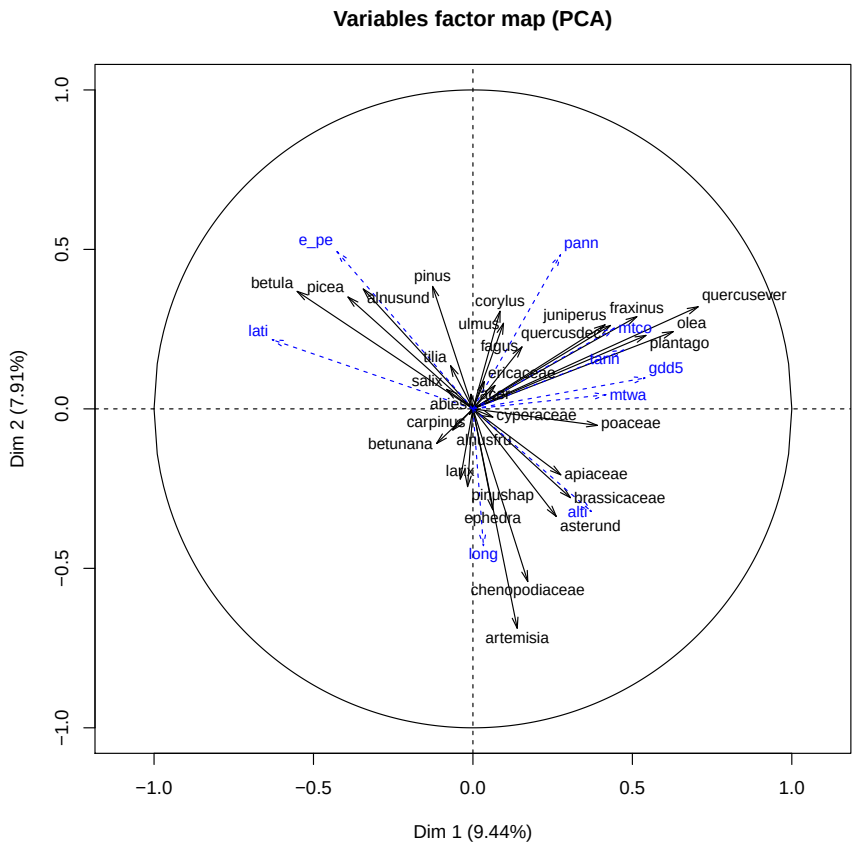
- Visualiser les 700 échantillons en fonction des concentrations de pollens (par ACP)
- Prédire la température annuelle (tann) en fonction des concentrations de pollens
- Etudier la relation entre biome et température annuelle

```
ss700 <- read.table("https://husson.github.io/img/ss700.csv", header=TRUE,
                    sep=";", row.names=1)
```

# ACP

```
library(FactoMineR)
res.pca <- PCA(ss700,quanti.sup=32:40, quali.sup=41,graph=FALSE)
plot(res.pca,hab=41,label="quali",cex=0.8)
plot(res.pca,choix="var",cex=0.8)
```



# ACP : description des dimensions

```
dimdesc(res.pca)
```

```
## $Dim.1
## $Dim.1$quanti
##                 correlation       p.value
## quercusever      0.70653170 6.559582e-107
## olea             0.62830800  3.721671e-78
## plantago         0.54289440  6.588354e-55
## gdd5             0.54004875  3.033049e-54
## fraxinus         0.51411998  1.744448e-48
## tann             0.47292243  2.703518e-40
## mtco             0.44433104  3.120251e-35
## juniperus        0.43142594  4.229579e-33
## mtwa             0.41702844  7.922163e-31
## quercusdec       0.41445975  1.962711e-30
## poaceae          0.39041209  6.602788e-27
## alti             0.37156885  2.437317e-24
## brassicaceae     0.30576165  1.293179e-16
## apiaceae         0.27572212  1.116715e-13
## pann             0.27475690  1.369452e-13
## asterund         0.26135995  2.140036e-12
## chenopodiaceae   0.17207707  4.676703e-06
## fagus            0.15400914  4.279399e-05
## artemisia        0.13896336  2.260575e-04
## ulmus            0.09591008  1.112136e-02
## corylus          0.08510370  2.434160e-02
## salix           -0.08283928  2.841042e-02
## betunana        -0.11398173  2.526650e-03
## pinus           -0.12664636  7.842611e-04
```

# Régression multiple

```
library(FactoMineR)
mod <- RegBest(ss700[,"tann"],ss700[,1:31])
mod$summary
```

```
##                                 R2        Pvalue
## Model with 1 variable   0.1337244  1.428198e-23
## Model with 2 variables  0.2407250  2.084893e-42
## Model with 3 variables  0.3114470  4.720331e-56
## Model with 4 variables  0.3813431  4.512025e-71
## Model with 5 variables  0.4332160  3.824672e-83
## Model with 6 variables  0.4790691  1.023203e-94
## Model with 7 variables  0.5172760 4.771272e-105
## Model with 8 variables  0.5414604 1.133191e-111
## Model with 9 variables  0.5606172 5.385370e-117
## Model with 10 variables 0.5799961 1.120032e-122
## Model with 11 variables 0.5973145 6.550118e-128
## Model with 12 variables 0.6061371 3.445850e-130
## Model with 13 variables 0.6144407 2.386004e-132
## Model with 14 variables 0.6195355 2.464602e-133
## Model with 15 variables 0.6241397 3.657097e-134
## Model with 16 variables 0.6291686 3.443937e-135
## Model with 17 variables 0.6330612 8.542180e-136
## Model with 18 variables 0.6368946 2.120644e-136
## Model with 19 variables 0.6389037 2.730759e-136
## Model with 20 variables 0.6402782 6.156617e-136
## Model with 21 variables 0.6410069 2.466417e-135
## Model with 22 variables 0.6414979 1.202941e-134
## Model with 23 variables 0.6420853 5.243144e-134
## Model with 24 variables 0.6424831 2.665793e-133
```

# Régression multiple

```
mod$best
```

```
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)    -4.540766   0.419324 -10.829  < 2e-16 ***
## alnusfru       -0.108785   0.038608  -2.818 0.004978 **
## artemisia       0.073286   0.011420   6.417 2.60e-10 ***
## asterund        0.095178   0.029829   3.191 0.001484 **
## betunana       -0.115918   0.041513  -2.792 0.005380 **
## carpinus        0.471085   0.047250   9.970  < 2e-16 ***
## chenopodiaceae  0.116026   0.011943   9.715  < 2e-16 ***
## corylus         0.570243   0.072462   7.870 1.40e-14 ***
## fagus           0.304931   0.103779   2.938 0.003412 **
## juniperus       0.210060   0.078342   2.681 0.007511 **
## larix          -0.172249   0.036628  -4.703 3.11e-06 ***
## olea            0.510621   0.054555   9.360  < 2e-16 ***
## pinushap       -0.084922   0.015777  -5.383 1.01e-07 ***
## pinus           0.116688   0.008417  13.863  < 2e-16 ***
## plantago        0.688050   0.125109   5.500 5.39e-08 ***
## poaceae         0.078168   0.014778   5.290 1.65e-07 ***
## quercusdec      0.208004   0.028929   7.190 1.71e-12 ***
## tilia           0.178202   0.059295   3.005 0.002750 **
## ulmus           0.771381   0.220401   3.500 0.000496 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.203 on 681 degrees of freedom
## Multiple R-squared:  0.6369, Adjusted R-squared:  0.6273
## F-statistic: 66.36 on 18 and 681 DF,  p-value: < 2.2e-16
```

# Analyse de variance

```
library(FactoMineR)
mod <- AovSum(tann ~ biome,data=ss700)
mod
```

```
## Ftest
##               SS  df      MS F value    Pr(>F)
## biome      12141   8 1517.61  49.976 < 2.2e-16 ***
## Residuals 20984 691   30.37
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Ttest
##               Estimate Std. Error  t value Pr(>|t|)
## (Intercept)    3.78679    0.46635   8.1200  < 2e-16 ***
## biome - COCO -9.29837    0.70151 -13.2547  < 2e-16 ***
## biome - COMX -2.31376    0.52629  -4.3964    1e-05 ***
## biome - COST -5.49444    0.81957  -6.7041  < 2e-16 ***
## biome - HODE -0.11811    0.98963  -0.1193  0.90503
## biome - TEDE  5.69047    1.04511   5.4449  < 2e-16 ***
## biome - TUND -7.09429    2.03812  -3.4808  0.00053 ***
## biome - WAMX  6.60221    2.47430   2.6683  0.00780 **
## biome - WAST  3.74400    0.72681   5.1513  < 2e-16 ***
## biome - XERO  8.28231    1.15853   7.1490  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

# Une carte à tester pour finir

```
library(leaflet)
pal <- colorNumeric(palette=c(low="blue",high="red"),domain=ss700["tann"])
m <- leaflet() %>% addTiles() %>%
  addCircles(ss700[,"long"],ss700[,"lati"], color=pal(ss700[,"tann"]),
             fillOpacity=1, opacity=1)
m
```

# Plan

# Les anti sèches de RStudio

R base

RStudio

RMarkdown

Importation

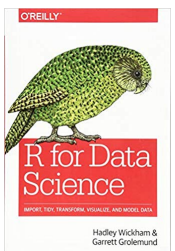
Manipulation

Visualisation

# Des livres

- A Language and Environment for Statistical Computing (R Core Team, 2017), https://www.R-project.org/


- R for Data science (Wickham & Grolemund, 2016), https://r4ds.had.co.nz/




- R pour la statistique et la science des données (Cornillon et al., 2018), https://r-stat-sc-donnees.github.io/

# Références

Cornillon, P.-A., Guyader, A., Husson, F., Jégou, N., Josse, J., Klutchnikoff, N., ... Thieurmel, B. (2018). *R pour la statistique et la science des données*. Presses universitaires de Rennes.

Kirwan, L., Connolly, J., Brophy, C., Baadshaug, O. H., Belanger, G., Black, A., ... others. (2014). The agrodiversity experiment: Three years of data from a multisite study in intensively managed grasslands. *Ecology, 2014, Vol. 95, Num. 9, P. 2680-2680*.

R Core Team. (2017). *R: A language and environment for statistical computing*. Vienna, Austria: R Foundation for Statistical Computing. Retrieved from https://www.R-project.org/

Wickham, H. (2014). *Advanced r*. CRC Press. Retrieved from http://adv-r.had.co.nz/

Wickham, H., & Grolemund, G. (2016). *R for data science: Import, tidy, transform, visualize, and model data*. " O'Reilly Media, Inc.".