E-cafe : premiers pas sous R

C. Hozé

Références

Tutoriel : http://user.oc-static.com/pdf/374508-effectuez-vos-etudes-statistiques-avec-r.pdf

Aide mémoire : http://www.duclert.org/

Statistiques avec R (Cornillon et al., Presses Universitaires Rennes)

Analyses de données avec R (F. Husson, S. Lê, J.Pagès), Presses Universitaires Rennes)

Le logiciel R

- **Logiciel pour les calculs statistiques et graphiques :**
  - **Plusieurs versions existent Windows, Mac, Linux…**
  - **Gratuit, Open source et collaboratif**
  - **Installé sur l'ensemble des nœuds de DGA20.**

- **Un langage de programmation simple**
  - **Proche du shell et/ou des mathématiques**
  - **fonction(arg1,arg2,…,argN)**
  - **Le résultat est un objet que l'on pourra manipuler / exporter facilement**
  - Ex: corrélation entre n variable matrice n lignes, n colonnes
- **Des arguments obligatoires et d'autres avec une valeur par défaut :**
  - **Si les options par défaut conviennent : 1 ou 2 arguments par fonction**
  - **Si on veut personnaliser : on précise chacun des arguments**
- **C'est grâce à cette personnalisation l'on peut (entre autres) faire de beaux graphiques ;-)**

R : Une infinité de fonctionnalités grâce aux packages

- **Package : module additionnel**
- **8029 packages disponibles sur le CRAN (Comprehensive R Archive Network)**

- **E-cafe à venir :**
  - **ggplot2** make beautiful graphics
  - **dplyr** : fast data manipulation.
  - **Rmarkdown** : export your report as an HTML, pdf, or MS Word doc, or a HTML or pdf slideshow
- **Quelques packages utiles pour les stats et la bio**
  - **lme4 / FactoMineR : analyses de variances (entre autres)**
  - **glmnet** pour le Lasso et l' elastic-net
  - **GenABEL** : an R library for genome wide analysis
  - **BioConductor** : série de packages dédié à la bioinformatique
  - (**Biomart**: accès à Ensembl depuis R)
- **Et même…**
  - **Rcmdr** (Windows) : pour faire du R en "clique-bouton"
  - **haven**/foreign/sas7bdat : lire base SAS ou Stata
  - **XL connect** : lire fichier excel (sans passer par un csv)
  - **ggmap** : télécharger une carte google map -> données spatiales
- **Souvent plusieurs commandes pour une même action :**

  commande de base, commande du package A, … commande du package N

Notions de base : ouvrir et quitter R

Lancer R : taper R dans un terminal

```
choze@dga12:/g2b/choze/SHO# R

R version 3.0.2 (2013-09-25) -- "Frisbee Sailing"
Copyright (C) 2013 The R Foundation for Statistical Computing
Platform: x86_64-unknown-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
>
```

Quitter R : taper **q()**

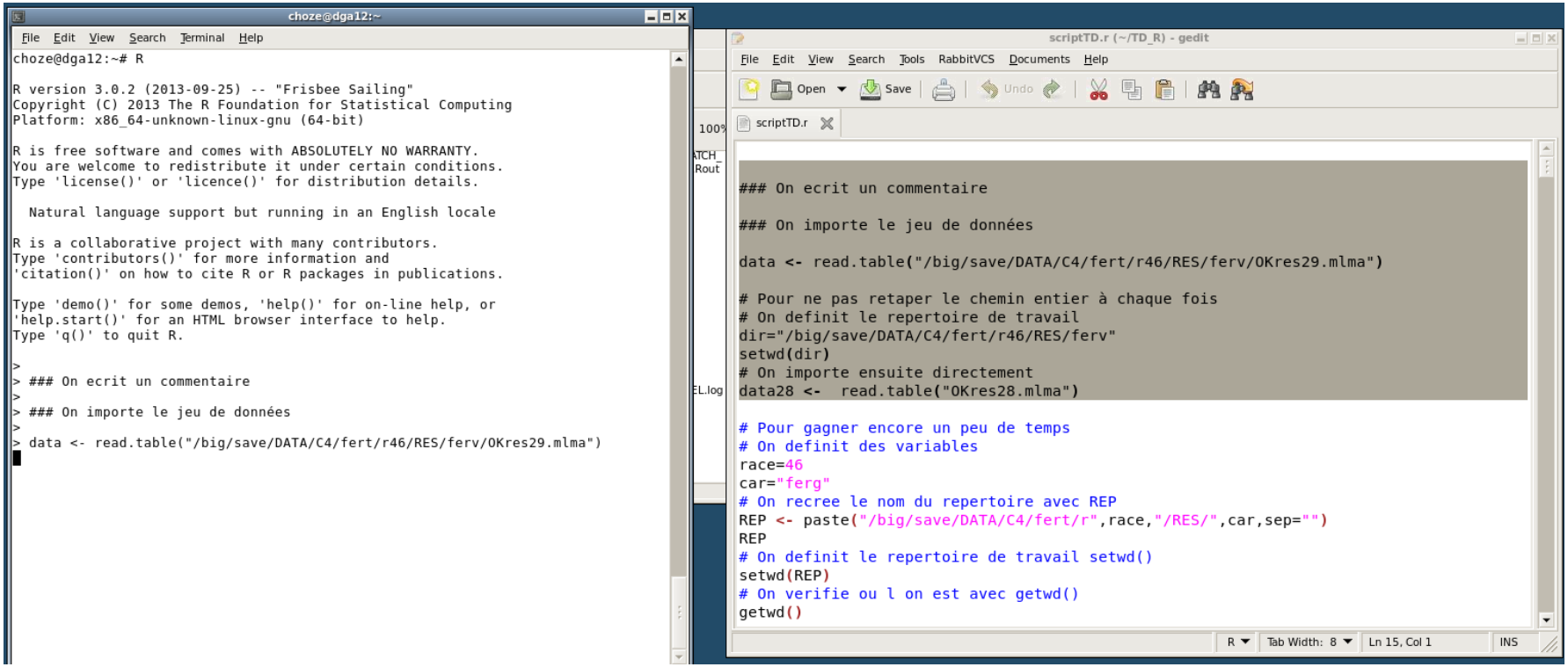
```
> q()
Save workspace image? [y/n/c]: n
```

Répondre oui, non, annuler pour sauvegarder ou pas une image de la session

Exécuter un script

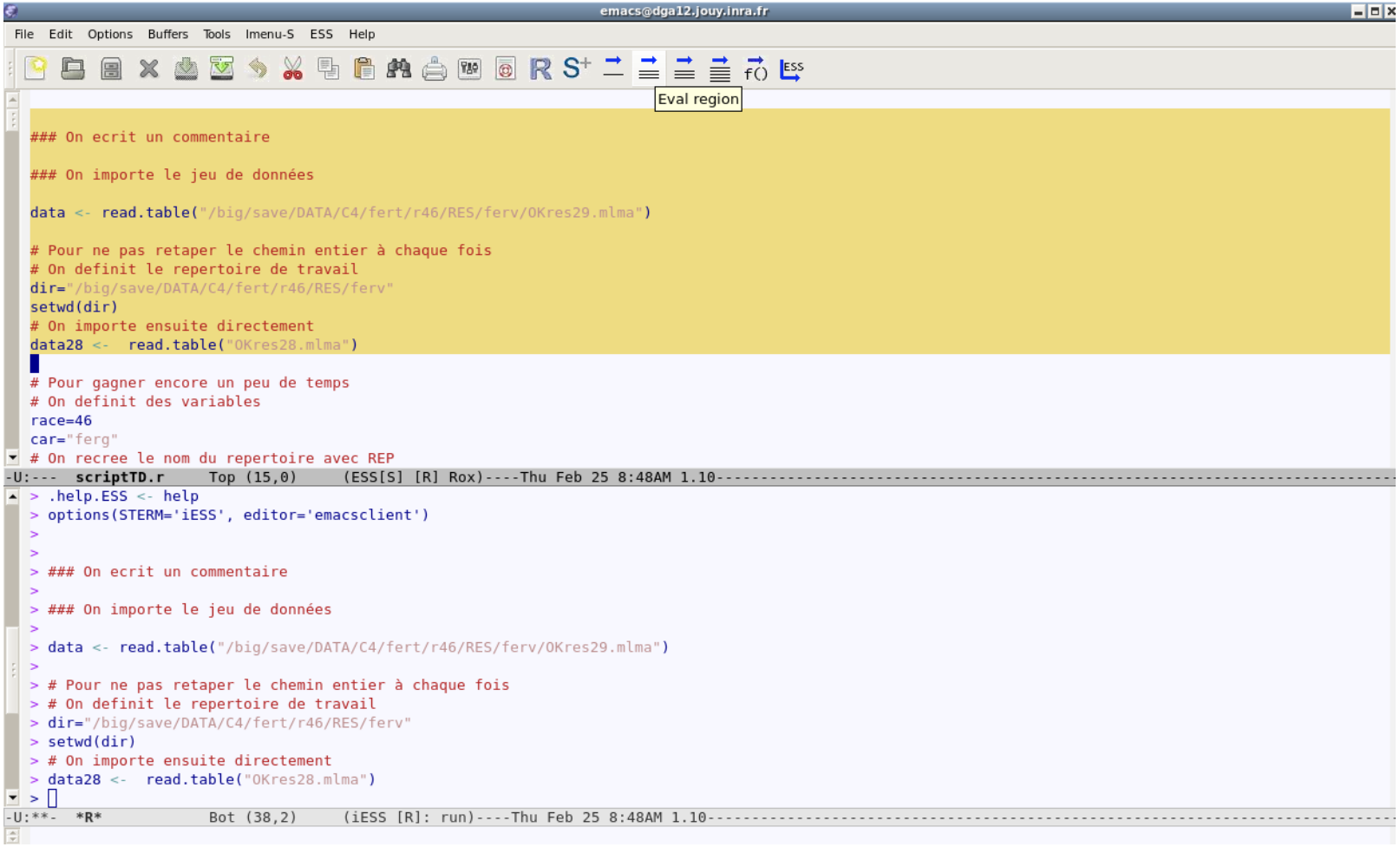
 # Dans un terminal, on ouvre R

 # On copie-colle depuis un éditeur de texte.



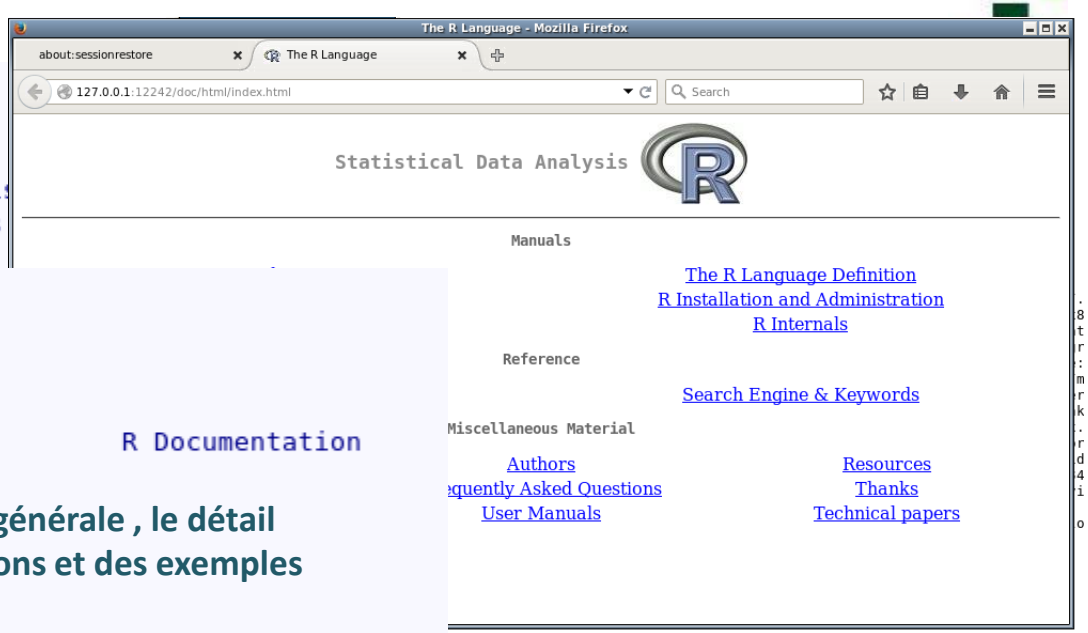
Executer un script (emacs)

On ouvre un .r avec EMACS



Notions de base : recherche d'aide

```
> #Ouvrir l'aide en ligne
>
> help.start()
If the browser launched by '/usr/bin/firefox' i
    *not* restarted, and you must switch to its
Otherwise, be patient ...
```




```
>
> # Rechercher de l'aide sur une fonction
> # On tape q pour sortir de l'aide
>
> help(print)
print                   package:base                    R Documentation

Print Values

Description:

     'print' prints its argument and returns it _invisibly_ (via
     'invisible(x)').  It is a generic function which means that new
     printing methods can be easily added for new 'class'es.

Usage:

     print(x, ...)
```

 # On a une description générale , le détail de l'ensemble des options et des exemples

On peut aussi appeler l'aide avec ?
?print
# Si on ne sait pas quel est le nom de la fonction -> on demande à google ;-)
# On peut aussi utiliser help.search() mais ce n'est pas hyper pratique….
help.search("linear models")

Notions de base : les objets

- **Opérateur assigner** : " **<-** ", "=" , "->"
  - Assigner une valeur à une variable, crée l'objet s'il n'existe pas, remplace sa valeur sinon
- **Les types de variables et types d'objets**
  - **Variable** : numérique, logique, caractère, facteur
  - **Vecteur** = une série d'éléments (numérique/caractère)
  - **Matrice** = une série d'éléments de même nature (numérique/caractère) ordonnée en i lignes et j colonnes
  - **Tableau** = une série d'éléments pouvant être de nature différente ordonnée en i lignes et j colonnes
  - **Liste** = une série d'objets quel que soit leur nature
- **Les noms des objets**
  - Doivent commencer par une lettre
  - Ils peuvent contenir des nombres, des points et des underscores ("_")
  - R est sensible à la casse : M n'est pas la même chose que m
- **Pour afficher la valeur d'une variable**
  - Il suffit de taper son nom (correspond à un appel en interne à la fonction print())
  - Pour qu'elle apparaisse dans la log ou dans une boucle il faut faire appel à print.
- **Pour sélectionner des éléments : écriture mathématique**
  - **vecteur[i]** : ième élément du vecteur
  - **matrice[**i,j**], tableau[i,j]** : $i^{ème}$ ligne, $j^{ème}$ colonne
  - **matrice[m:n,] :** Lignes m à n toutes les colonnes
  - **tableau[,c(i,j,k)] :** toutes les lignes, colonnes i,j,k

Notions de base : Import de données (cas simple) et utilisation de variables

 # On écrit un commentaire avec # et on importe le jeu de données

 # Ici : les options par défaut fonctionnent : séparateur " " ou tabulation, pas d'entête, décimale ".", pas de données manquantes

```
> data <- read.table("/big/save/DATA/C4/fert/r46/RES/ferv/OKres29.mlma")
>
> # Pour ne pas retaper le chemin entier à chaque fois
> # On definit le repertoire de travail
> dir="/big/save/DATA/C4/fert/r46/RES/ferv"
> setwd(dir)
> # On importe ensuite directement
> data28 <-  read.table("OKres28.mlma")
```

```
choze@dga12:/big/save/DATA/C4/fert/r46/RES/ferv# head OKres1.mlma
1 238 0.434685 -0.439711 0.397404 0.268529
1 300 0.450041 -0.483342 0.403916 0.231447
1 316 0.160319 0.388867 0.527772 0.461238
1 324 0.289926 -0.0659716 0.433182 0.878954
1 340 0.312858 -0.862047 0.430066 0.0450214
1 357 0.317363 0.0288679 0.425547 0.945915
1 380 0.320229 0.378716 0.41772 0.364605
1 420 0.341933 0.94997 0.420564 0.0238959
1 752 0.44656 0.459655 0.394141 0.243526
1 774 0.0534398 0.557011 0.858532 0.516471
```

 # Si elle n est pas entre guillemets, R cherchera à utiliser une chaine de caractère comme une variable

```
> # Pour gagner encore un peu de temps
> # On definit des variables
> race=46
> car="ferg"
```

 # Utiliser une variable pour créer une chaine de caractère : la commande paste()

```
> # On recree le nom du repertoire avec REP
> REP <- paste("/big/save/DATA/C4/fert/r",race,"/RES/",car,sep="")
> REP
[1] "/big/save/DATA/C4/fert/r46/RES/ferg"
>
> # On definit le repertoire de travail setwd()
> setwd(REP)
> # On verifie ou l on est avec getwd()
> getwd()
[1] "/big/save/DATA/C4/fert/r46/RES/ferg"
```

 # Equivalent au shell REP=/big/save/DATA/C4/fert/r$race/RES/$car

 # On aurait pu mettre le paste directement dans le setwd sans passer par la variable REP :
setwd(paste("/big/save/DATA/C4/fert/r",race,"/RES/",car,sep="")

 # Import depuis le web : on rentre l'adresse web du fichier à la place du chemin sur le disque

Import des données : Options utiles

```
> # Si il y a un en tête et un separateur autre que espace et tabulation
> repGS3 <- "/big/C4/fert/BAYESCpi_sansPOIDS/r46/RES/ferv"
> file <- paste(repGS3,"/RES_solGS3_GWASferv5poly90pi04.csv",sep="")
>
> dataGS3 <- read.table(file,header=T,sep=";")
>
choze@dga12:/big/C4/fert/BAYESCpi_sansPOIDS/r46/RES/ferv# head RES
nomSNP;BTA;pos;solution;p;sumP;sumPelim;freq;-logP
Chr11_77901980;11;77.90198;0.0148;0.0214;0.1436;0.1423;0.79;NA
Chr11_77903846;11;77.903846;0;2e-04;0.1445;0.1428;0.388;NA
Chr11_77904163;11;77.904163;-0.1848;0.1177;0.1446;0.1428;0.04;NA
```

 # Il y a un ordre par défaut pour l es arguments
si on ne le respecte pas il faut préciser
de quel argument il s'agit :
dataGS3 <- read.table(sep=";", header=T, file="monfichier.txt")

```
> # Si on avait eu des decimales en "," et des donnees manquantes a -9999
>
> dataTEST <- read.table(file,header=T,sep=";",dec=",",na.strings="-9999")
```

```
> # Si on avait laisse des blancs au lieu des donnees manquantes
> # Et qu il n y avait qu un separateur espace
>
> dataBLANK <- read.table("/big/C3/BAYESCpi/r46/RES/ap/RES_solGS3_GWASap8_blank.csv",header=T)
Error in scan(file, what, nmax, sep, dec, quote, skip, nlines, na.strings,  :
  line 1 did not have 9 elements
>
> # On ajoute l option fill=TRUE (a utiliser avec prudence)
>
> dataBLANK <- read.table("/big/C3/BAYESCpi/r46/RES/ap/RES_solGS3_GWASap8_blank.csv",header=T,fill=TRUE)
```

```
choze@dga12:/big/C3/BAYESCpi/r46/RES/ap# head /big/C3/BAYESCpi/r46/RES/
nomSNP BTA pos solution p sumP sumPelim freq -logP
Chr13_71724838 13 71.724838 0 2e-04 0.0076 0.0055 57.963
Chr13_71726373 13 71.726373 1e-04 6e-04 0.0101 0.0074 0.054
Chr13_71726383 13 71.726383 0 2e-04 0.0122 0.0094 0.054
Chr13_71728108 13 71.728108 0 1e-04 0.0146 0.0111 5.153
Chr13_71728351 13 71.728351 -9e-04 0.0044 0.0147 0.0112 7.736
Chr13_71729062 13 71.729062 -2e-04 0.0011 0.0146 0.011 7.722
```

 # 8 éléments car champ vide pour -logP

EX 1 : Import des données et vérifications

 # On importe le fichier index qui lui nécessite des options

```
> # On se place dans le repertoire de travail
> setwd("/home/choze/TD_R/NEW_DIR")
> # On lit le fichier  contenant les index des taureaux separateur ; donnees manquante 9999
> index <- read.table("index.csv",header=TRUE,sep=";",na.strings=-9999,fill=TRUE)
```

```
> # On regarde la tête du fichier
>
> head(index)
                ANI ISU INEL Milk FatKG ProKG UHealth Otype  BODY Udder FandL
1 HOL840M003010354171 173   39  735    40    30    0.28   1.9 -0.19  1.95  0.95
2 HOL840M003010356301 192   46  634    34    38    1.20   3.2  1.30  3.00  1.00
3 HOL840M003012574873 185   39 1273    44    32    1.30   3.0  1.00  2.50  1.60
4 HOL840M003012643773 174   50 1250    48    41    1.10   1.3 -0.30  1.30  0.80
5 HOL840M003012662536 178   22  693    13    21    2.10   2.4  0.10  2.50  1.00
6 HOL840M003013924034 186   39  838    31    32    1.40   2.0 -0.10  1.80  1.40
```

 # head() ou tail() permettent de regarder quelques lignes du jeu de données

 # Rq : on pourrait faire data[1:5,]

 # On importe le fichier bull qui lui aussi nécessite des options

```
# On lit le fichier contenant les infos sur les taureaux
bull <- read.table("liste_bull.csv",header=TRUE,sep=";",fill=TRUE)
```

 # summary() donne des infos min, max, nb de niveau…

```
> summary(bull)
                ANI          PAYS       GROUPE
 HOL840M003008315818:   1   GEBV :4109   DEU:1611
 HOL840M003010354171:   1   GMACE: 236   DFS: 490
 HOL840M003010356301:   1                ESP: 548
 HOL840M003011639700:   1                FRA:1046
 HOL840M003012574873:   1                NLD: 650
 HOL840M003012643773:   1
 (Other)            :4339
```

```
      MOIS
 Min.   : 1.000
 1st Qu.: 4.000
 Median : 7.000
 Mean   : 6.644
 3rd Qu.: 9.000
 Max.   :12.000
```

```
      ANNEE
 Min.   :2007
 1st Qu.:2011
 Median :2012
 Mean   :2012
 3rd Qu.:2013
 Max.   :2015
```

```
      DANAIS
 06/06/2011:  11
 27/03/2012:  10
 28/11/2012:  10
 02/09/2011:   9
 10/09/2011:   9
 17/08/2011:   9
 (Other)   :4287
```

#Vérifier la nature des variables
Numérique: min/max/mean
Facteur: nb de modalités
Caractère : Character

Facteur : niveaux et occurrences par niveau

Numérique : min, max, mean

EX 1 : Changer le type de variable

Commande class(), connaitre le type de variable

```
> # Pour connaitre le type de variable
> class(bull$DANAIS)
[1] "factor"
> # Pour le faire sur toutes les colonnes
> sapply(bull,class)
      ANI      PAYS    GROUPE      MOIS     ANNEE    DANAIS
 "factor"  "factor"  "factor" "integer" "integer"  "factor"
```

On recode avec : as.factor(), as.numeric(), as.character()

```
> # On recode en facteur annee et mois
> # et en caractere ANI et DANAIS
> bull[,1] <- as.character(bull[,1])
> bull[,4] <- as.factor(bull[,5])
> bull[,5] <- as.factor(bull[,5])
> bull[,6] <- as.character(bull[,6])
>
```

On vérifie que les modifs ont fonctionné

```
> # On refait un class et un summary
> sapply(bull,class)
       ANI       PAYS     GROUPE       MOIS      ANNEE     DANAIS
"character"   "factor"   "factor"   "factor"   "factor" "character"
> summary(bull)
     ANI
 Length:4345
 Class :character
 Mode  :character

    PAYS       GROUPE          MOIS          ANNEE
 GEBV :4109   DEU:1611   2012   :1205   2012   :1205
 GMACE: 236   DFS: 490   2013   :1087   2013   :1087
              ESP: 548   2011   :1072   2011   :1072
              FRA:1046   2014   : 745   2014   : 745
              NLD: 650   2010   : 184   2010   : 184
                         2015   :  44   2015   :  44
                         (Other):   8   (Other):   8

    DANAIS
 Length:4345
 Class :character
 Mode  :character
```

Plus de min/max pour MOIS/ANNEE
Plus les modalités pour bull et DANAIS
Facteur , caractère

On aurait pu éviter cette manipulation :

En indiquant la nature des variables dès l'import de données : colClasses

En empêchant la convertir les chaines de caractères en facteur : stringsAsFactor

```
> # Si on veut, on peut l indiquer des l import de donnees
> bullCOL <- read.table("liste_bull.csv",header=TRUE,sep=";",fill=TRUE,
+                       colClasses=c("character",rep("factor",4),"character"))
> sapply(bullCOL,class)
       ANI       PAYS     GROUPE       MOIS      ANNEE     DANAIS
"character"   "factor"   "factor"   "factor"   "factor" "character"
> # On aurait aussi pu l empecher de transformer caractere en facteur
>
> bullFACTOR <- read.table("liste_bull.csv",header=TRUE,sep=";",fill=TRUE,stringsAsFactors=FALSE)
>
> sapply(bullFACTOR,class)
       ANI       PAYS     GROUPE       MOIS      ANNEE     DANAIS
"character" "character" "character"   "integer"   "integer" "character"
>
```

Créer une nouvelle variable,
calculer l'âge des animaux à partir des DANAIS

```
# Si on veut securiser un changement de type de variable
# Surtout pour le passage en numerique

bull$MOIS_NUM <- as.numeric(bull$MOIS)

# On etait pas oblige de lui donner un nom
# On aurait pu donner uniquement sa position
bull[,ncol(bull)+1] <- <- as.numeric(bull$MOIS)
```

 # On tape le nom ou la position de la nouvelle variable et les valeurs qu'on veut lui attribuer

```
# On calcule l age des animaux
# On cree une variable dn pour que  R réinterprete les dates de naissances
bull$dn <- as.Date(bull$DANAIS,format='%d/%m/%Y')
```

 # On lui dit que DANAIS est une date au format jj/mm/aaaa

```
> # On stocke la date du jour dans la variable Auj
> Auj <- Sys.Date()
> Auj
[1] "2016-02-25"
```

 # Utilisation de Sys.Date() pour avoir la date systeme

```
# On calcule l age en mois
bull$AgeMOIS <- as.numeric(round((Auj - bull$dn)/(365.25/12)))
```

 # On calcule et on arrondit avec round

```
> bull[10:15,]
                  ANI PAYS GROUPE MOIS ANNEE     DANAIS         dn AgeMOIS
10 HOLAUTM000146117722 GEBV    DEU 2013  2013 08/05/2013 2013-05-08      34
11 HOLAUTM000347391218 GEBV    DEU 2013  2013 27/03/2013 2013-03-27      35
12 HOLAUTM000476565619 GEBV    DEU 2011  2011 21/09/2011 2011-09-21      53
13 HOLAUTM000584233319 GEBV    DEU 2011  2011 12/10/2011 2011-10-12      52
14 HOLAUTM000624670819 GEBV    DEU 2012  2012 17/06/2012 2012-06-17      44
15 HOLAUTM000668950918 GEBV    DEU 2011  2011 24/02/2011 2011-02-24      60
>
```

 # On regarde les lignes 10 à 15

Créer une nouvelle variable,

Faire des classes à partir d'une variable numérique

 # On transforme l'âge en mois en classe d'âge

```
> # Pour ça on doit d abord definir les points de coupure
> # de 0 à max age par pas de 3.
> seq(from=0,to=max(bull[,8]),by=3)
 [1]   0   3   6   9  12  15  18  21  24  27  30  33  36  39  42  45  48  51  54
[20]  57  60  63  66  69  72  75  78  81  84  87  90  93  96  99 102
>
> # Puis on coupe, là encore on est pas oblige de preciser l ordre
> bull$AgeClasse <- cut(x= bull[,8], breaks= seq(0,max(bull[,8]),3))
>
```

 # On regarde le résultat avec un head()

```
> head(bull)
                  ANI  PAYS GROUPE MOIS ANNEE     DANAIS         dn AgeMOIS
1 HOL840M003010354171 GMACE    DEU 2014  2014 04/01/2014 2014-01-04      26
2 HOL840M003010356301  GEBV    DEU 2014  2014 19/01/2014 2014-01-19      25
3 HOL840M003012574873  GEBV    DEU 2013  2013 28/12/2013 2013-12-28      26
4 HOL840M003012643773  GEBV    DEU 2013  2013 28/07/2013 2013-07-28      31
5 HOL840M003012662536  GEBV    DEU 2013  2013 16/06/2013 2013-06-16      32
6 HOL840M003013924034  GEBV    DEU 2013  2013 11/12/2013 2013-12-11      27
  AgeClasse
1   (24,27]
2   (24,27]
3   (24,27]
4   (30,33]
5   (30,33]
6   (24,27]
```

 # On regarde le résultat avec un summary()

```
> summary(bull)
     ANI              PAYS       GROUPE          MOIS          ANNEE
 Length:4345        GEBV :4109   DEU:1611   2012   :1205   2012   :1205
 Class :character   GMACE: 236   DFS: 490   2013   :1087   2013   :1087
 Mode  :character                ESP: 548   2011   :1072   2011   :1072
                                 FRA:1046   2014   : 745   2014   : 745
                                 NLD: 650   2010   : 184   2010   : 184
                                            2015   :  44   2015   :  44
                                            (Other):   8   (Other):   8
    DANAIS                dn                 AgeMOIS          AgeClasse
 Length:4345        Min.   :2007-09-12   Min.   :  11.00   (51,54]: 387
 Class :character   1st Qu.:2011-11-15   1st Qu.:  30.00   (48,51]: 357
 Mode  :character   Median :2012-09-24   Median :  41.00   (42,45]: 307
                    Mean   :2012-10-17   Mean   :  40.42   (36,39]: 300
                    3rd Qu.:2013-09-13   3rd Qu.:  52.00   (54,57]: 299
                    Max.   :2015-03-31   Max.   : 102.00   (45,48]: 296
                                                           (Other):2399
```

Réordonner et renommer les variables

 # On crée un nouveau dataset qui contient certaines colonnes du dataset initial
 # Ici on sélectionne les colonnes par leurs noms

```
> # On cree un nouveau jeu de donnees
> # On selectionne les colonnes et l ordre dans lequel on les veut
>
> bull2 <- bull[,c("ANI","GROUPE","PAYS","AgeMOIS")]
>
> dim(bull2)
[1] 4345    4
> head(bull2)
                 ANI GROUPE  PAYS AgeMOIS
1 HOL840M003010354171    DEU GMACE      26
2 HOL840M003010356301    DEU  GEBV      25
3 HOL840M003012574873    DEU  GEBV      26
4 HOL840M003012643773    DEU  GEBV      31
5 HOL840M003012662536    DEU  GEBV      32
6 HOL840M003013924034    DEU  GEBV      26
```

 # On aurait pu les choisir à partir de leur numéros

```
> # On pouvait aussi choisir directement quels colonnes par leur numero
>
> head(bull[,c(1,3,2,8)])
                 ANI GROUPE  PAYS AgeClasse
1 HOL840M003010354171    DEU GMACE   (24,27]
2 HOL840M003010356301    DEU  GEBV   (24,27]
3 HOL840M003012574873    DEU  GEBV   (24,27]
4 HOL840M003012643773    DEU  GEBV   (30,33]
5 HOL840M003012662536    DEU  GEBV   (30,33]
6 HOL840M003013924034    DEU  GEBV   (24,27]
```

 # Eliminer une colonne du fichier bull

```
> # On aurait pu garder le même jeu de donnees
> # et seulement supprimer la colonne danais devenu inutile
>
> bull <- bull[,-c(6)]
>
> head(bull)
                 ANI  PAYS GROUPE MOIS ANNEE AgeMOIS AgeClasse
1 HOL840M003010354171 GMACE    DEU    1  2014      26   (24,27]
2 HOL840M003010356301  GEBV    DEU    1  2014      25   (24,27]
3 HOL840M003012574873  GEBV    DEU   12  2013      26   (24,27]
4 HOL840M003012643773  GEBV    DEU    7  2013      31   (30,33]
5 HOL840M003012662536  GEBV    DEU    6  2013      32   (30,33]
6 HOL840M003013924034  GEBV    DEU   12  2013      26   (24,27]
>
```

 # Les noms sont faux, la colonne pays est en 2
 # On renomme les variables de bull2

```
> # Le vecteur des noms est names
> names(bull2)
[1] "ANI"       "PAYS"      "ORI_INDEX" "AgeMOIS"
>
> # On change chacune des valeurs du vecteur
> names(bull2) <- c("ANI","PAYS","GROUPE","AgeMOIS")
> # Pour changer un seul nom, on selectionne son numero
> names(bull2)[3]
[1] "GROUPE"
> names(bull2)[3] <- c("ORI_INDEX")
>
> # On regarde a nouveau les noms
> names(bull2)
[1] "ANI"       "PAYS"      "ORI_INDEX" "AgeMOIS"
>
```

 # On change la valeur du 3ème élément

Sélection des lignes : principe

```
> #######  Principe ######
> y=c(15,5,9,2,68)
> y
[1] 15  5  9  2 68
>
> # On garde les elements de y>10
> y[which(y>10)]
[1] 15 68
>
```

```
> # Il interprete comme une selection
> # de lignes classiques
> which(y>10)
[1] 1 5
>
> # On garde les elements pour lesquels
> # la condition est remplie
> y>10
[1]  TRUE FALSE FALSE FALSE  TRUE
> # La selection est equivalente a
> y[c(TRUE,FALSE,FALSE,FALSE,TRUE)]
[1] 15 68
> # Et a
> y[c(1,5)]
[1] 15 68
>
```

 # which facultatif que si on redirige

```
> # Si on redirige le jeu de donnees
> # Le which est facultatif
> z <- y[y>10]
>
>  # Et on garde si > 60
> z[which(z>60)]
[1] 68
>
> # Mais NA si on fait
> z[which(y>60)]
[1] NA
>
> # Car l equivalence T/F est :
> z>60
[1] FALSE  TRUE
> y>60
[1] FALSE FALSE FALSE FALSE  TRUE
>
```

```
> # Du coup il cherche le 5eme de Z
> # alors que Z n a que  2 elements
> z[5]
[1] NA
> # Mais dans y ça marchait
> y[which(y>60)]
[1] 68
> y[2]
[1] 5
```

 # A garder en tête pour la sélection sur une colonne du jeu de données

Sélection des données

```
> # Fusion de l info taureau avec l info index
> # Fusionner deux fichiers entre eux
> # Si le nom de la colonne est identique entre les deux fichiers
> tot <- merge(bull,index,by="ANI")
> dim(tot)
[1] 4345   25
> # On peut selectionner lignes et colonnes en meme temps
> # On garde que les FRA et on selectionne les colonnes
> totFRA <- tot[which(tot$GROUPE=="FRA"),c(1,2,3,9,10,11)]
>
> dim(totFRA)
[1] 1046    6
>
```

Double condition :

On ne double pas les et "&" ni les ou "|"

```
> # On garde les ISU > 200
> totTOPISU <- tot[which(tot$ISU>200),]
> # On garde les top ISU non francais
> totTOPFR <- tot[which((tot$ISU>200)&(tot$GROUPE!="FRA")),]
> # On garde les top ISU ou INEL > 200
> totTOP <- tot[which((tot$ISU>200)|(tot$INEL>200)),]
>
```

Elément d une liste

```
> # On garde les elements d une liste
> # Par exemple francais et danois
> selec <- c("FRA","DFS")
> tot_FRA_DFS <- tot[which(tot$GROUPE %in% selec),]
```

Négation d une condition

```
> # La negation de la condition est possible
> # On elimine les ani avec NA pour ISU
> tot_sansNA <- tot[- which(is.na(tot$ISU)),]
>
```

```
> ### La encore attention au jeu de donnees
>  ### Car en interne il fait la liste des lignes
> # OK si :
> head(tot[which(tot$ISU>200),c(1,10)])
                    ANI ISU
13  HOL840M003015021963 203
188 HOLDEUM000121478656 201
418 HOLDEUM000356230019 203
421 HOLDEUM000356230573 202
494 HOLDEUM000356948488 201
518 HOLDEUM000357196821 203
> head(totFRA[which(totFRA$ISU>200),c(1,5)])
                     ANI ISU
13   HOL840M003015021963 203
709  HOLDEUM000538239227 210
1963 HOLFRAM002229184160 202
2076 HOLFRAM002923863182 205
2088 HOLFRAM002930983673 201
2145 HOLFRAM002942482318 207
> # Mais pas de message d erreur si on fait :
> head(totFRA[which(tot$ISU>200),c(1,5)])
                     ANI ISU
101  HOLCANM000011792387 187
1971 HOLFRAM002234683917 168
2210 HOLFRAM003554162527 138
```

 # Pourtant les animaux gardés ne sont pas les bons

Modifier certaines lignes

 # On regroupe les classes d'âge <12 mois et supérieur à 36 mois

```
> # On repasse en caractere pour faire les modifs
> bull$AgeClasse <- as.character(bull$AgeClasse)
>
> # On fait une seule classe pour les >36mois
> bull$AgeClasse[which(bull$AgeMOIS>36)] <- ">36"
>
```

 # On ne peut pas créer de nouvelles modalités sur une variable facteur

 # On change les valeurs des éléments sélectionnés

```
> # Revient a remplacer les éléments correspondants aux criteres
> bull$AgeClasse[which(bull$AgeMOIS<=12)]
[1] "(9,12]" "(9,12]" "(9,12]" "(9,12]" "(9,12]" "(9,12]" "(9,12]"
```

 # Sur le principe on peut remplacer le vecteur des éléments sélectionnés par n' importe quel vecteur de la même dimension

```
> # 7 élements on peut remplacer par n importe quel vecteur de 7 éléments
> # Si on mets un vecteur à 8 éléments : on a un message d'erreur
> bull$AgeClasse[which(bull$AgeMOIS<=12)] <- c("A","B","C","D","E","F","G","H")
Warning message:
In bull$AgeClasse[which(bull$AgeMOIS <= 12)] <- c("A", "B", "C",  :
  number of items to replace is not a multiple of replacement length
```

 # On crée un vecteur qui répète L fois "<12" avec rep

```
> # Pour avoir le nombre d elements on utilise length()
> L=length(bull$AgeClasse[which(bull$AgeMOIS<=12)])
> bull$AgeClasse[which(bull$AgeMOIS<=12)] <- rep("<12",L)
```

 # On explicite ici ce que R a fait en interne sur le >36

```
> # On regarde si cela a marche
>  bull[c(4328:4337),]
                    ANI PAYS GROUPE MOIS ANNEE     DANAIS         dn AgeMOIS AgeClasse
4328 REDNLDM000875886770 GEBV    NLD 2015  2015 11/03/2015 2015-03-11      12       <12
4329 REDNLDM000876695386 GEBV    NLD 2014  2014 11/03/2014 2014-03-11      24    (21,24]
4330 REDNLDM000881380093 GEBV    NLD 2011  2011 02/03/2011 2011-03-02      60       >36
4331 REDNLDM000882228011 GEBV    NLD 2011  2011 19/06/2011 2011-06-19      56       >36
```

 # On remplace par le même vecteur + 5

```
# On rajoute 5 points d'ISU a tous les francais

tot[which(tot$GROUPE=="FRA"),c("ISU")]     <- tot[which(tot$GROUPE=="FRA"),c("ISU")] + 5
```

Statistiques variables qualitatives

 # Nb de niveaux et nombre d'occurences

```
> # On peut faire un summary pour avoir une idee
> # mais ne donnera pas tout si > 100 modalités
> summary(bull[,5])
2007 2008 2009 2010 2011 2012 2013 2014 2015
   2    1    5  184 1072 1205 1087  745   44
>
> # On peut recuperer le nombre de niveau et leurs valeurs
> nlevels(bull[,5])
[1] 9
> levels(bull[,5])
[1] "2007" "2008" "2009" "2010" "2011" "2012" "2013" "2014" "2015"
> # On peut compter le nb d'occurence par niveau
> table(bull[,5])

2007 2008 2009 2010 2011 2012 2013 2014 2015
   2    1    5  184 1072 1205 1087  745   44
>
```

Variables quantitatives, summary vs table

```
> summary(bull[,1])      # Il affiche les 100 premières modalités
HOL840M003008315818 HOL840M003010354171 HOL840M003010356301 HOL840M003011639700
                  1                   1                   1                   1
HOL840M003012574873 HOL840M003012643773 HOL840M003012662536 HOL840M003013177255
                  1                   1                   1                   1
HOL840M003013177322 HOL840M003013177354 HOL840M003013924034 HOL840M003014561936
                  1                   1                   1                   1
                                   [...]

HOLCANM000011087870 HOLCANM000011087874 HOLCANM000011144095 HOLCANM000011219853
                  1                   1                   1                   1
HOLCANM000011329554 HOLCANM000011354336 HOLCANM000011471404 HOLCANM000011471410
                  1                   1                   1                   1
HOLCANM000011591477 HOLCANM000011591482 HOLCANM000011591506 HOLCANM000011596114
                  1                   1                   1                   1
HOLCANM000011692021 HOLCANM000011696756 HOLCANM000011696774             (Other)
                  1                   1                   1                4246
> 
```

```
> table(bull[1])   # Il affiche tout

HOL840M003008315818 HOL840M003010354171 HOL840M003010356301 HOL840M003011639700
                  1                   1                   1                   1
HOL840M003012574873 HOL840M003012643773 HOL840M003012662536 HOL840M003013177255
                  1                   1                   1                   1
HOL840M003013177322 HOL840M003013177354 HOL840M003013924034 HOL840M003014561936
                                   [...]
REDNLDM000938582683 REDNLDM000938582931 REDNLDM000940628944 REDNLDM000946716663
                  1                   1                   1                   1
REDNLDM000946768378 REDNLDM000946768563 REDNLDM000949046020 REDNLDM000949046710
                  1                   1                   1                   1
REDUSAM000056629618 REDUSAM000071692203 REDUSAM000072002090 REDUSAM000072002107
                  1                   1                   1                   1
REDUSAM000072002143
                  1
```

Statistiques variables qualitatives

Nb de niveaux et nombre d'occurences

```
> # On peut faire un summary pour avoir une idee
> # mais ne donnera pas tout si > 100 modalités
> summary(bull[,5])
2007 2008 2009 2010 2011 2012 2013 2014 2015
   2    1    5  184 1072 1205 1087  745   44
>
> # On peut recuperer le nombre de niveau et leurs valeurs
> nlevels(bull[,5])
[1] 9
> levels(bull[,5])
[1] "2007" "2008" "2009" "2010" "2011" "2012" "2013" "2014" "2015"
> # On peut compter le nb d'occurence par niveau
> table(bull[,5])

2007 2008 2009 2010 2011 2012 2013 2014 2015
   2    1    5  184 1072 1205 1087  745   44
>
```

On ajoute prop.table pour avoir les proportions

```
> # Avoir les proportions
> prop.table(table(bull[,5],bull[,3]))

                DEU          DFS          ESP          FRA          NLD
2007 0.0000000000 0.0000000000 0.0004602992 0.0000000000 0.0000000000
2008 0.0000000000 0.0000000000 0.0002301496 0.0000000000 0.0000000000
2009 0.0000000000 0.0000000000 0.0009205984 0.0002301496 0.0000000000
2010 0.0000000000 0.0002301496 0.0032220944 0.0289988493 0.0098964327
2011 0.0909090909 0.0257767549 0.0340621404 0.0607594937 0.0352128884
2012 0.1150747986 0.0271576525 0.0331415420 0.0621403913 0.0398158803
2013 0.1037974684 0.0292289988 0.0294591484 0.0504027618 0.0372842348
2014 0.0589182969 0.0301495972 0.0237054085 0.0342922900 0.0243958573
2015 0.0020713464 0.0002301496 0.0009205984 0.0039125432 0.0029919448
```

Table de contingence table()

```
> # Faire des tables de contingences
> table(bull[,5],bull[,3])

      DEU DFS ESP FRA NLD
2007    0   0   2   0   0
2008    0   0   1   0   0
2009    0   0   4   1   0
2010    0   1  14 126  43
2011  395 112 148 264 153
2012  500 118 144 270 173
2013  451 127 128 219 162
2014  256 131 103 149 106
2015    9   1   4  17  13
```

Et round avec une option, pour arrondir au 3ème chiffre

```
> # Arrondir les proportions
> round(prop.table(table(bull[,5],bull[,3])),3)

       DEU   DFS   ESP   FRA   NLD
2007 0.000 0.000 0.000 0.000 0.000
2008 0.000 0.000 0.000 0.000 0.000
2009 0.000 0.000 0.001 0.000 0.000
2010 0.000 0.000 0.003 0.029 0.010
2011 0.091 0.026 0.034 0.061 0.035
2012 0.115 0.027 0.033 0.062 0.040
2013 0.104 0.029 0.029 0.050 0.037
2014 0.059 0.030 0.024 0.034 0.024
2015 0.002 0.000 0.001 0.004 0.003
```

Statistiques variables quantitatives

```
> # Des premières infos avec un summary
> summary(index[,1:3])
                  ANI             ISU              INEL
 HOL840M003008315818:   1   Min.   : 79.0   Min.   :-31.00
 HOL840M003010354171:   1   1st Qu.:146.0   1st Qu.: 19.00
 HOL840M003010356301:   1   Median :158.0   Median : 29.00
 HOL840M003011639700:   1   Mean   :158.2   Mean   : 29.07
 HOL840M003012574873:   1   3rd Qu.:171.0   3rd Qu.: 40.00
 HOL840M003012643773:   1   Max.   :222.0   Max.   :100.00
 (Other)            :4339   NA's   :57
> # On peut le faire direct sur une colonne
> summary(index[,3])
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 -31.00   19.00   29.00   29.07   40.00  100.00
> # On peut recuperer min, max, moyennes
> min(index[,3]);max(index[,3]);mean(index[,3])
[1] -31
[1] 100
[1] 29.07388

> # Ecart types et variances
> var(index[,3]); sd(index[,3])
[1] 258.6573
[1] 16.08283
> # Les quantiles
> quantile(index[,3])
  0%  25%  50%  75% 100%
 -31   19   29   40  100
```

 # Le résultat est un vecteur, si on veut le quantile 75%, il suffit de sélectionner le 4ème élément quantile(index[,3])[4]

Stat variables quantitatives

```
> # Faire une correlation entre deux variables
>  cor(index[,3],index[,4])
[1] 0.574486
>
> # Entre l ensemble des variables
> cor(index[,2:6])
      ISU      INEL      Milk     FatKG     ProKG
ISU     1        NA        NA        NA        NA
INEL   NA 1.0000000 0.5744860 0.7795982 0.9456368
Milk   NA 0.5744860 1.0000000 0.3310595 0.7325512
FatKG  NA 0.7795982 0.3310595 1.0000000 0.5688462
ProKG  NA 0.9456368 0.7325512 0.5688462 1.0000000
```

 # Par défaut il ne traite pas les variables avec NA

```
> # Eliminer toutes les lignes avec donnees manquantes
> indexNA <- na.omit(index)
> dim(index);dim(indexNA)
[1] 4345   17
[1] 4109   17
> # Refaire la correlation
> cor(indexNA[,2:6])
            ISU      INEL      Milk     FatKG     ProKG
ISU   1.0000000 0.6368403 0.3834687 0.4680691 0.6178532
INEL  0.6368403 1.0000000 0.5735257 0.7810592 0.9457695
Milk  0.3834687 0.5735257 1.0000000 0.3321382 0.7313480
FatKG 0.4680691 0.7810592 0.3321382 1.0000000 0.5711992
ProKG 0.6178532 0.9457695 0.7313480 0.5711992 1.0000000
```

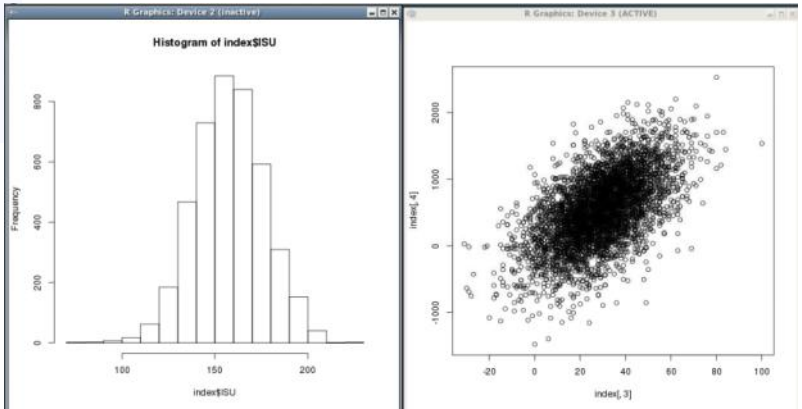
```
> # Histogramme
> hist(index$ISU)
> > # On ouvre une nouvelle fenetre
> # pour pas ecraser le graph du dessus
> x11()
> # Nuage de points
> plot(index[,3],index[,4])
>
```

 # On change l option par defaut pour garder toutes les observations avec infos sur les 2 colonnes
 # On évite ainsi de supprimer toute la ligne.

```
> # Ou plus subtilement lui rajouter l option
> cor(index[,2:6],use= "pairwise.complete.obs")
            ISU      INEL      Milk     FatKG     ProKG
ISU   1.0000000 0.6362305 0.3828836 0.4664378 0.6173888
INEL  0.6362305 1.0000000 0.5744860 0.7795982 0.9456368
Milk  0.3828836 0.5744860 1.0000000 0.3310595 0.7325512
FatKG 0.4664378 0.7795982 0.3310595 1.0000000 0.5688462
ProKG 0.6173888 0.9456368 0.7325512 0.5688462 1.0000000
>
```

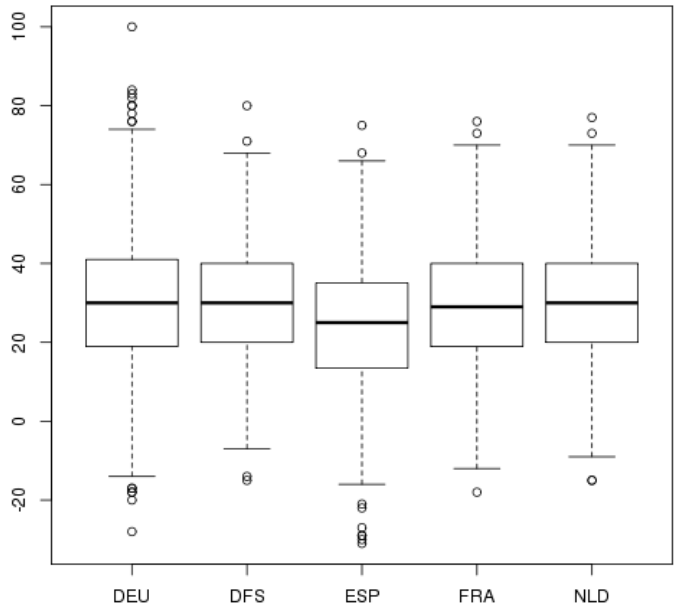
```
> # Pour les commandes var, mean, sd, syntaxe differente
>  var(index[2])
    ISU
ISU  NA
>  var(index[2],na.rm=TRUE)
        ISU
ISU 346.3748
```

 # Avec les options par défaut les graph sont moches sous R aussi ;-)



Statistiques par groupe

```
# Pour faire des boites Ã_ moustaches
boxplot(tot$INEL~tot$GROUPE)
```



 # Moyenne de l INEL pour chacun des pays

```
> by(tot$INEL,tot$GROUPE,mean)
tot$GROUPE: DEU
[1] 29.73991
------------------------------------------------------------
tot$GROUPE: DFS
[1] 29.62041
------------------------------------------------------------
tot$GROUPE: ESP
[1] 23.75912
------------------------------------------------------------
tot$GROUPE: FRA
[1] 29.55258
------------------------------------------------------------
tot$GROUPE: NLD
[1] 30.72154
```

 # Fonction by : applique la fonction à chaque niveau du facteur,

Pour avoir les moyennes sur plusieurs colonnes colMeans

 # Moyenne des colonnes 10 à 14 pour chacun des pays

```
> by(tot[,10:14],tot$GROUPE,colMeans)
tot$GROUPE: D
      ISU      INEL      Milk     FatKG     ProKG
       NA  29.73991 618.93669  32.64929  22.92055
------------------------------------------------------------
tot$GROUPE: DFS
      ISU      INEL      Milk     FatKG     ProKG
       NA  29.62041 286.59184  31.83469  21.04286
------------------------------------------------------------
tot$GROUPE: ESP
      ISU      INEL      Milk     FatKG     ProKG
       NA  23.75912 702.02920  28.84672  18.88869
------------------------------------------------------------
tot$GROUPE: FRA
      ISU      INEL      Milk     FatKG     ProKG
158.76769  29.55258 591.71128  29.32027  23.40344
------------------------------------------------------------
tot$GROUPE: NLD
      ISU      INEL      Milk     FatKG     ProKG
       NA  30.72154 352.38000  34.20769  21.88308
```

Tri sur une ou plusieurs colonnes et export d'un fichier

```
> #### On veut le top INEL
>
> totTRI <- tot[order(tot$INEL),c(1,2,3,10,11,12)]
> # Le top se trouve a la fin
> head(totTRI)
                    ANI PAYS GROUPE AgeClasse ISU INEL
4305 REDNLDM000881237953 GEBV    ESP       >36  95  -31
1624 HOLESPM002702894296 GEBV    ESP       >36  82  -30
1514 HOLESPM000503132869 GEBV    ESP       >36  92  -29
1692 HOLESPM003303396470 GEBV    ESP       >36  84  -29
4137 REDDEUM000814888607 GEBV    DEU       >36  96  -28
1513 HOLESPM000503130968 GEBV    ESP       >36  96  -27
```

 # On veut les lignes triées selon la colonne INEL et les colonnes 1,2,3,10,11,12

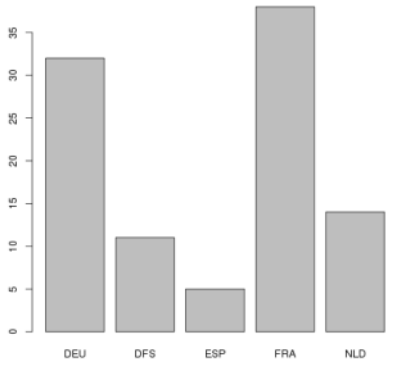
 # On s'aperçoit que dans l'ordre croissant,
 # On ajoute l'option decreasing=TRUE

```
> # On trie en sens inverse
> totTRI <- tot[order(tot$INEL,decreasing=TRUE),c(1,2,3,9,10,11)]
> totTRI <- tot[order(-tot$INEL),c(1,2,3,10,11,12)]
> # On trie sur INEL et ISU
> totTRI <- tot[order(tot$ISU,tot$INEL,decreasing=TRUE),c(1,2,3,9,10,11)]
> # On cree la liste les 100premiers individus colonne 1
> listeTOP <- totTRI[1:100,1]
> # On exporte le fichier sans guillemets, noms colonnes ni nom de lignes
> write.table(listeTOP,"listeANItopINEL.txt",quote=F,col.names=F,row.names=F)
```

 # Si on veut trier sur plusieurs colonnes

 # On récupére les numeros des 100 premiers

```
> ## Connnaitre la repartition par pays du top ISU puis barplot
> table(totTRI[1:100,3])

DEU DFS ESP FRA NLD
 32  11   5  38  14
>
> barplot(table(totTRI[1:100,3]))
```



EX : Exploitation des sorties de GS3

Les options du merge

```
> # On fusionne d abord la carte et les frequences
> # Les noms des var
> head(carte)
  V4 V4.1            V1 V2       V3
1  1    1 Chr6_53981154  6 53981154
2  2    2 Chr6_53981307  6 53981307
3  3    3 Chr6_53981324  6 53981324
4  4    4 Chr6_53981328  6 53981328
5  5    5 Chr6_53981371  6 53981371
6  6    6 Chr6_53981391  6 53981391
> head(freq)
  V1     V2
1  1 35.840
2  2 36.469
3  3 36.469
4  4 36.483
5  5 36.309
6  6  0.040
>
```

 # Si le nom de variable est différent entre les 2 dataset
 # On remplace le by de tout à l'heure par by.x , by.y

```
> # Pour fusionner alors que les noms sont differents
> # On precise le nom (ou la position) de la variable sur la
> # Par defaut R garde les donnnees comunnes aux 2 fichiers
> carteB <- merge(carte,freq,by.x="V4",by.y="V1")
>
>
> # Si les noms sont identiques R les renomme
> head(carteB)
  V4 V4.1            V1 V2.x       V3   V2.y
1  1    1 Chr6_53981154    6 53981154 35.840
2  2    2 Chr6_53981307    6 53981307 36.469
3  3    3 Chr6_53981324    6 53981324 36.469
4  4    4 Chr6_53981328    6 53981328 36.483
5  5    5 Chr6_53981371    6 53981371 36.309
6  6    6 Chr6_53981391    6 53981391  0.040
>
```

 # Le merge classique ne peut fusionner les fichiers que 2 à 2

```
> # On fusionne ensuite avec les resultats du BAYES
> # pour retrouver les positions sur le genome des SNP et freq
> analyseSNP <- merge(data[which(data[,1]==3),],carteB,by.x="level",by.y="V4")
>
```

```
> #### On va rechercher les resultats du GWAS
>
> NAMEgcta <- paste(RES_gcta,"_",BTA,"_",car,".csv",sep="")
> resGCTA.tmp <- read.table(NAMEgcta)
> resGCTA <- resGCTA.tmp[which((resGCTA.tmp[2]>=minP)&(resGCTA.tmp[2]<=maxP)),c(2,8)]
>
```

 # Par défaut R conserve uniquement les éléments communs aux 2 fichiers
 # On ajoute les options all, all.x all.y pour conserver l'ensemble des éléments de X et/ou Y

```
### On refusionne avec resGCTA
# Par defaut R garde les donnnees communes aux 2 fichiers, on change paramètres all
resSOMME <- merge(analyseSNP,resGCTA,by.x="V3",by.y="V2",all.x=T,all.y=F)
```

Manipulation de lignes

```
> # On fait un premier fichier avec proba > 1.1 * pi
>
> resPROBA <- resSOMME[(resSOMME$p>1.1*PI),c(10,11,1,4,6,13,14,12)]
> # On fait un second fichier avec -logP different de NA
> resLOGP <-   resSOMME[!is.na(resSOMME$V8),c(10,11,1,4,6,13,14,12)]
```

 # On aurait pu faire le tout en une seule étape avec un "ou"

```
> # On les concatene
> resGLOBAL <- rbind(resPROBA,resLOGP)
> print(nrow(resGLOBAL))
[1] 18103
>
```

 # rbind permet de concaténer les lignes,
Il existe aussi cbind pour accoler deux fichiers sur la base des colonnes

```
> # On elimine les eventuels doublons
>
> resFINAL <- unique(resGLOBAL)
> print(nrow(resFINAL))
[1] 18091
>
> # Si on avait voulu faire le sort sur certaines colonnes BTA,pos
> # On met le unique sur le choix des lignes
> resFINAL <- resGLOBAL[!duplicated(resGLOBAL[,c(2,3)]),]
> dim(resFINAL)
[1] 18091     8
```

 # unique permet d'éliminer les doublons

 # On élimine les lignes non dupliquées pour col 2 et 3

```
> # On remet des noms + lisibles
> names(resFINAL) <- c( "nomSNP","BTA","pos","solution","p","sumP","logP","freq")
> head(resFINAL)
          nomSNP BTA      pos     solution         p      sumP logP   freq
2  Chr6_53981307   6 53981307 1.150810e-06 0.0001375 0.0061250   NA 36.469
8  Chr6_53981502   6 53981502 3.328864e-08 0.0001375 0.0068125   NA  0.455
9  Chr6_53981530   6 53981530 2.572730e-07 0.0002125 0.0069375   NA  0.669
10 Chr6 53981537   6 53981537 3.137708e-06 0.0002000 0.0070250   NA 36.162
```

```
> # On exporte le tout
> # Cette fois on met separateur ";" et on garde nom des colonnes
> write.table(resFINAL,"RES_GS3.csv",sep=";",quote=F,col.names=TRUE,row.names=FALSE)
>
```

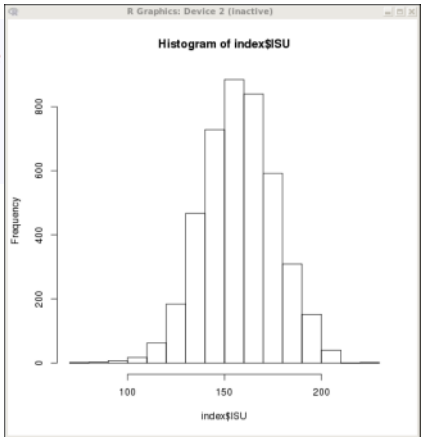
Aller + loin

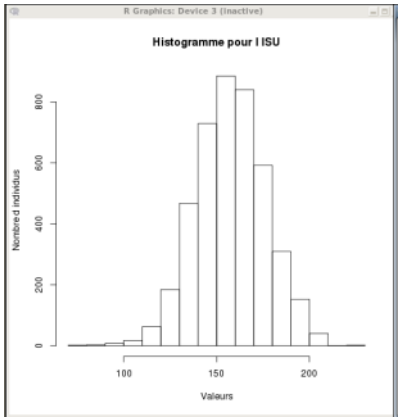
Personnalisation des graphiques
Lancement en batch et passage d arguments
Programmation if/for

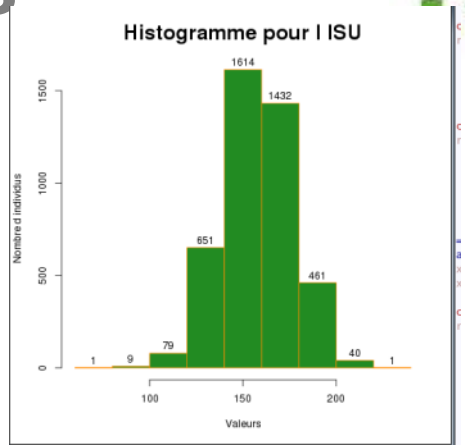
Apply et autres fonctions
Quelques commandes utiles

Améliorer son graph : titre et noms des axes Et options spécifiques à l'histogramme

```
### Faire un histogramme #


hist(index$ISU)
```







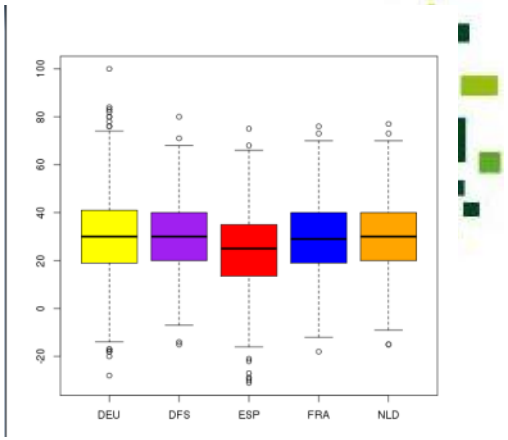

```
# On ajoute titre et noms des axes
hist(index$ISU,main="Histogramme pour l ISU",xlab="Valeurs",ylab="Nombre d individus")
```

```
# On rajoute encore des options
hist(index$ISU,main="Histogramme pour l ISU",xlab="Valeurs",ylab="Nombre d individus",

     col="forestgreen",          # Couleur  de remplissage

     border="darkorange",        # Couleur des contours

     nclass=10,                  # Nombre de classes

     label=TRUE,                 # Afficher le nombre par niveau

     cex.main=2)                 # Augmenter la taille du titre
```

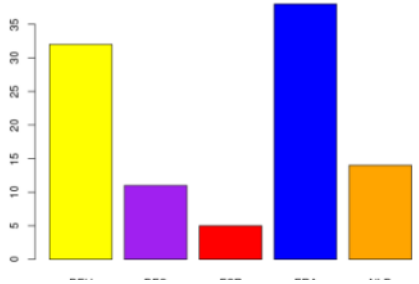
Améliorer son graph : choisir ses couleurs

```
# On choisit soi mÃªme ses couleurs en definissant un vecteur de couleurs

colPAYS <- c("yellow","purple","red","blue","orange")
```

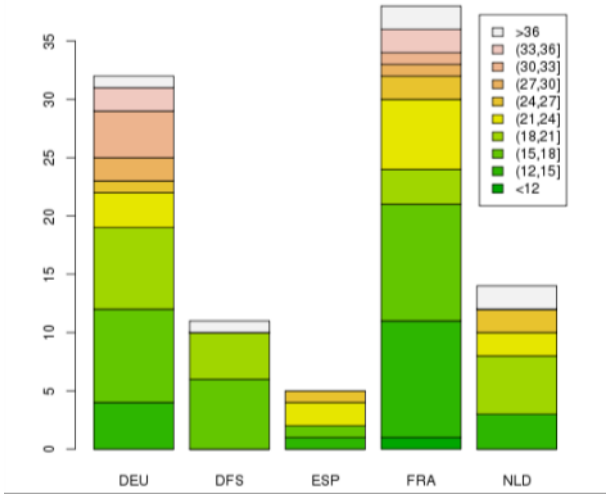
```
> # Pour faire des boites à moustaches
> boxplot(tot$INEL~tot$GROUPE,col=colPAYS)
```




```
> ## Connnaitre la repartition par pays du top ISU puis barplot
> CompoTOPpays <- table(totTRI[1:100,3])
> CompoTOPpays

DEU DFS ESP FRA NLD
 32  11   5  38  14
> barplot(CompoTOPpays,col=colPAYS)
```



```
> # Connaitre la repartition par pays et date de naissance
> CompoTOP <- table(totTRI[1:100,4],totTRI[1:100,3])

> CompoTOP

        DEU DFS ESP FRA NLD
  <12     0   0   0   1   0
  (12,15] 4   0   1  10   3
  (15,18] 8   6   1  10   0
  (18,21] 7   4   0   3   5
  (21,24] 3   0   2   6   2
  (24,27] 1   0   1   2   2
  (27,30] 2   0   0   1   0
  (30,33] 4   0   0   1   0
  (33,36] 2   0   0   2   0
  >36     1   1   0   2   2
> barplot(CompoTOP,col=terrain.colors(nrow(CompoTOP)),legend=T)
>
```

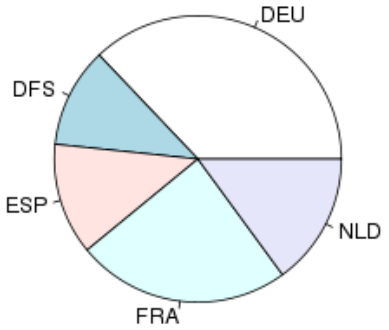


 # Il existe plusieurs "palettes" par defaut : la + classique est rainbow
 # terrain.colors est un peu moins aggressive pour les yeux que le rainbow

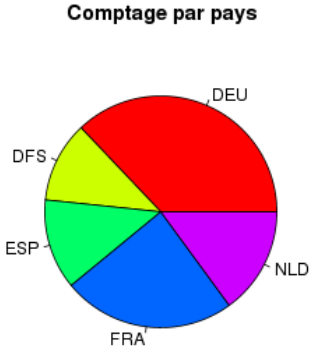
Améliorer son graph : diagramme circulaire

On fait une table contenant les comptages avec table()

```
 DEU  DFS  ESP  FRA  NLD
1611  490  548 1046  650
> # On les stocke
> ComptagePays <- table(bull[,3])
>
> # On fait un diagramme
>  pie(ComptagePays)
>  x11()
```

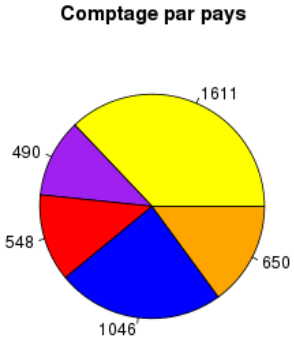



```
> # On ajoute un titre (main) et on choisit les couleurs (col) avec rainbow
>  pie(ComptagePays,main="Comptage par pays",col=rainbow(5))
>  x11()
```

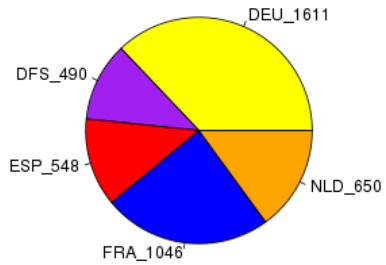



```
> # On choisit soi même ses couleurs en definissant un vecteur de couleurs
> colPAYS <- c("yellow","purple","red","blue","orange")
> # On peut decider de changer les "labels" en mettant les comptage
> pie(ComptagePays,labels=ComptagePays,main="Comptage par pays",col=colPAYS)
```

```
> # Du coup on perd l info pays, il faut rajouter une legende
> legend("topright", legend =names(ComptagePays), ,col=colPAYS, pch = 15)
> # localisation, étiquette de légende, couleur, type de point, 15 = petit carré
```




```
> # On peut mettre n importe quel vecteur de 5 elements en "label"
> # On aurait pu mettre A,B,C,D,E
> # Ou on concatene pays et nombre
> LAB <- paste(names(ComptagePays),ComptagePays,sep="_")
>
> pie(ComptagePays,labels=LAB,main="Comptage par pays",col=colPAYS)
>
>
```

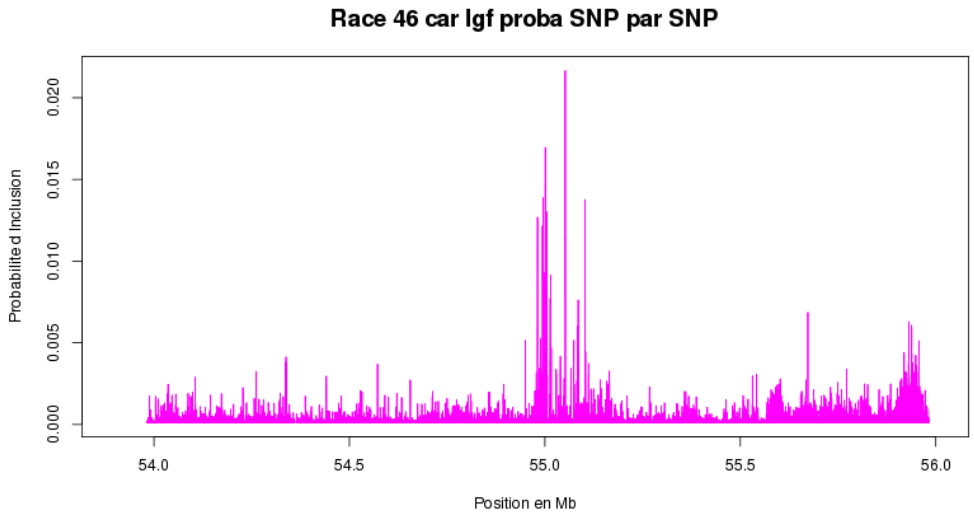


Améliorer son graph : nuages de points

```
> # On met en Mb et on fait des lignes
> minP <- floor(min(pos*10^-5))/10
> maxP <- ceiling(max(pos*10^-5))/10
>
> plot(pos*10^-6,
+      p,
+      xlim=c(minP,maxP),
+      col=BTA,
+      type="l",
+      xlab="Position en Mb",
+      ylab="Probabilite d Inclusion",
+      cex=1.5,
+      main=paste("Race",race,"car",car,"proba SNP par SNP"),
+      cex.main=3,
+      )
```

Pour arrondir à une décimale près
On arrondit en dessous et on divise
On arrondit au dessus et on divise

Abscisses
Ordonnées
limites de l axe des abscisses
couleur en fonction de la colonne BTA
On veut des lignes
Nom des axes
Taille des points ratio par rapport à 1
Nom du titre
Taille du titre



Améliorer ses graphs : ajouts d'éléments

```
> # On veut plus de marques sur l axe des abscisses
> plot(pos*10^-6,
+      p,
+      xlim=c(minP,maxP),
+      col="blue",                # couleur bleue
+      pch=".",                   # On veut des points "."
+      xlab="Position en Mb",
+      xaxt="n",                  # On ne veut pas d axe des abscisses
+      ylab="Probabilite d Inclusion",
+      cex=2,
+      main=paste("Race",race,"car",car,"proba SNP par SNP"),
+      cex.main=2,
+      )
>
```

```
# On prépare la création du nouvel axe
# avec une marque sans  texte tous les 0.01
# avec une marque avec texte tous le 0.1
> # On cree un vecteur avec les positions on veut des marques
>
> POS <- seq(minP,maxP,by=0.01)
> LAB <- array("",length(POS))
> # Un deuxieme avec les positions qu on veut afficher
> LAB10 <- seq(minP,maxP,by=0.1)
>
```

```
# On ajoute l axe en 2 fois une pour les marques tous les 0.01 puis tous les 0.1
> # On cree les axes
> axis(1, at = POS, labels =LAB, tick = TRUE, col ="black", lwd =1, las = 1, cex.axis = 1)
> axis(1, at =LAB10, labels=LAB10, tick = TRUE, col ="black", lwd =2, las = 1, cex.axis = 1)
>
```

```
# On ajoute une ligne verticale
> # On veut ajouter une ligne rose horizontale \340 p=0,01
> abline(h=0.01,col="pink")
```

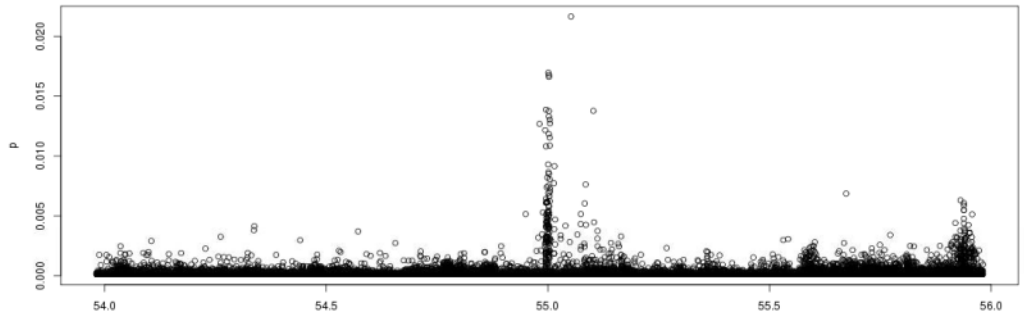
```
># On ajoute des points
> # On veut afficher les SNP qui sortent en GCTA
> points(resFINAL[!is.na(logP),c("pos")]*10^-6,resFINAL[!is.na(logP),c("p")],col="red",pch=20)
```

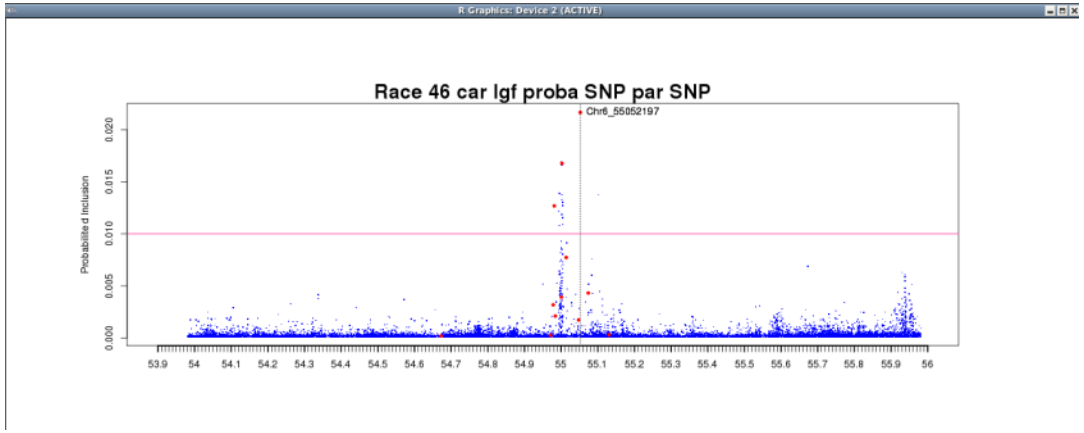
```
>  # On ajoute du texte
> # On veut le nom du SNP max
> SNPmax <- resFINAL[p==max(p,na.rm=T),]
> text(x=SNPmax[,3]*10^-6,y=SNPmax[,5],labels=SNPmax[,1],pos=4)
>
```

```
  # On ajoute une ligne verticale en pointillé
> # On veut une ligne vertical pointille à la position Max
> abline(v=(SNPmax[,3]*10^-6),col="darkgrey",lty="dotted")
```

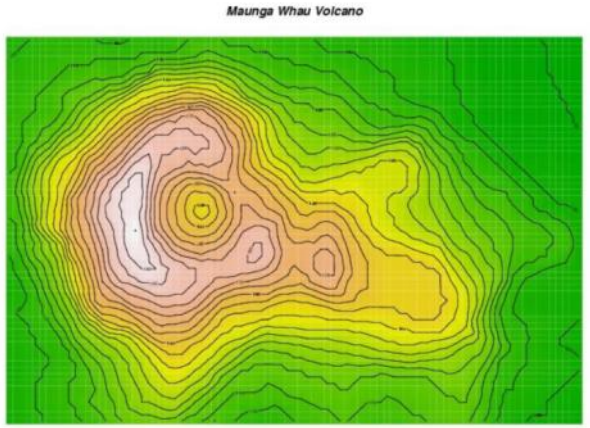
Améliorer ses graphs

avec les bonnes options on peut (quasiment) tout faire

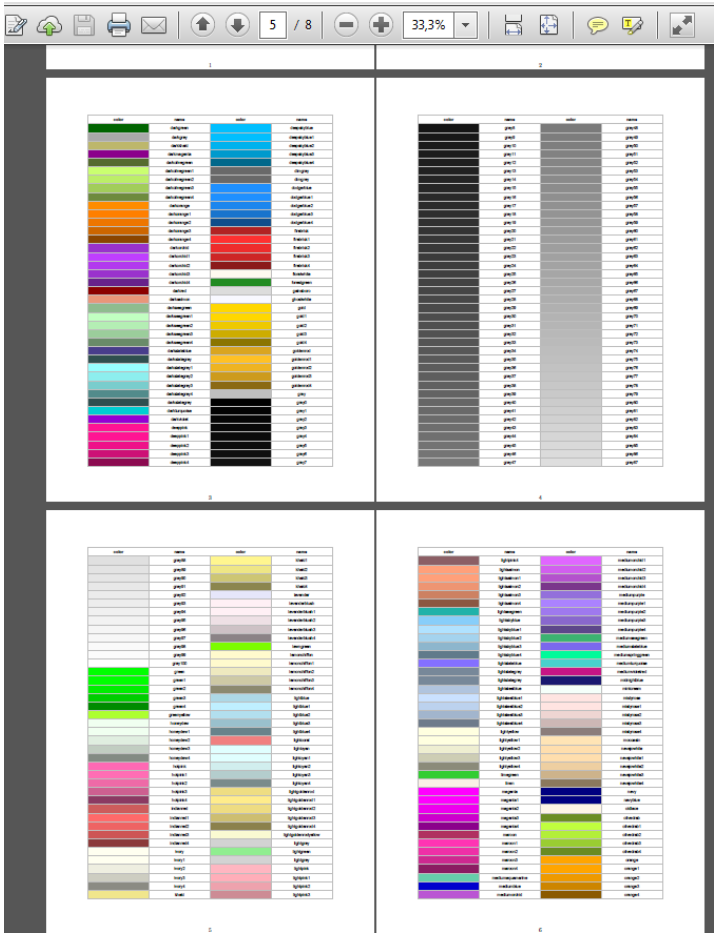




Avec les données adéquates, vous pourrez même faire des graphes aussi poussés que celui ci:



Une palette de couleurs + fournies qu'un catalogue de peinture



- http://www.stat.columbia.edu/~tzheng/files/Rcolor.pdf
- **Pour les + connaisseurs : définition des couleurs avec rgb(x,y,z)**
- **Paramètre alpha pour gérer la transparence, utilisable aussi sur couleurs par défaut**

Export de graphique et mise en page

 # Pour exporter un graph : on redirige la sortie graphique

```
pdf("fichierGraphique.pdf")
plot(res$pos,res$logP)
dev.off()
```

- **On peut définir le format et l'orientation**
- **Syntaxe identique avec jpeg(), png(), on peut définir le format et la résolution**
- **Attention avec pdf il stocke les infos pour chaque point donc cela peut être très lourd pour des graph de détection de QTL.**

 # Si on veut plusieurs graphiques dans la même sortie,

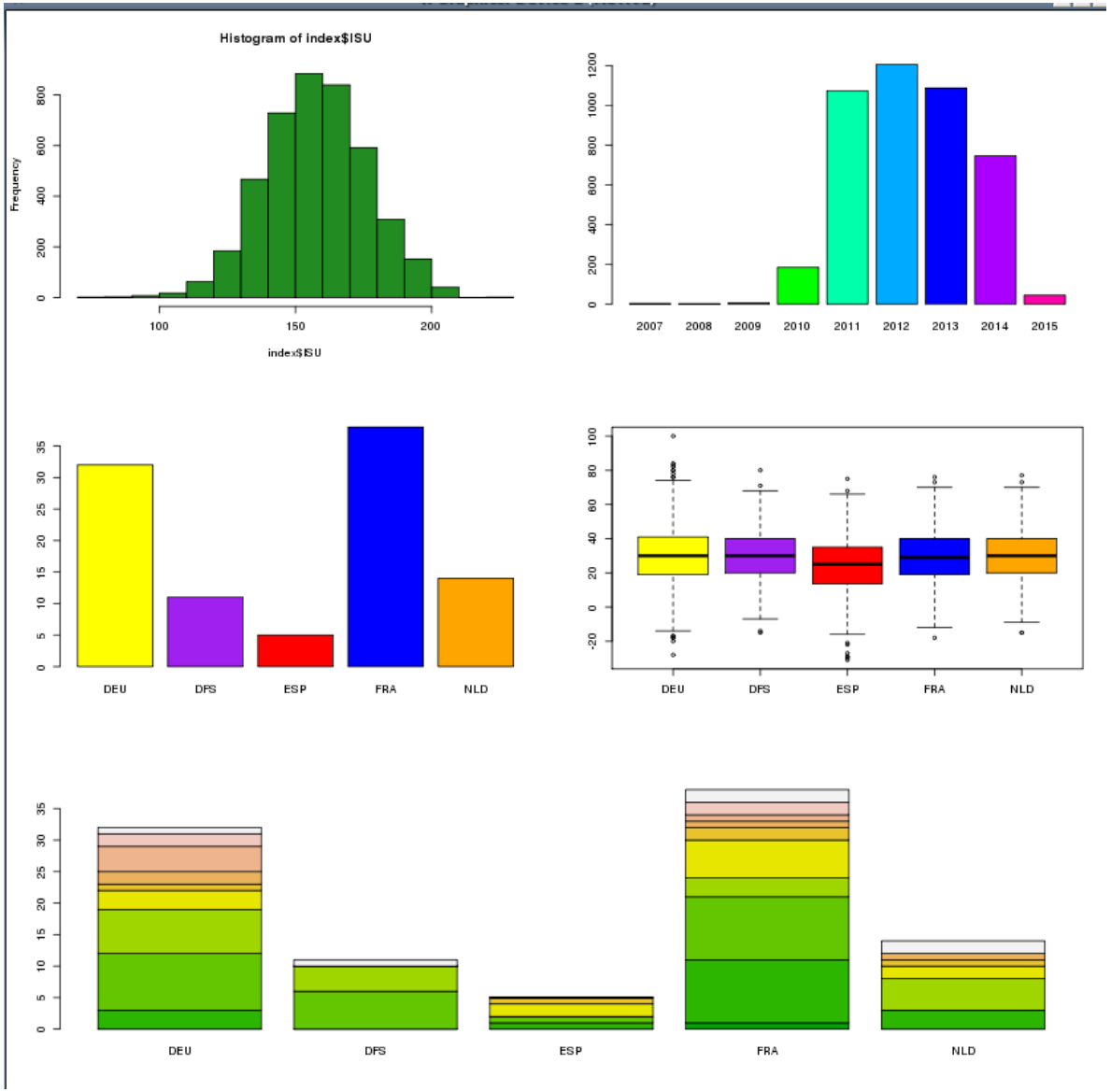
- **Avec un pdf il créera de nouvelle page jusqu'au dev.off()**
- **Pour un jpeg, png il faut indiquer la mise en page avec layout()**

```
> # On prepare une matrice qui definira la page
> # 3 graphs :  3 lignes, 1 colonne
>      matrix(1:3,3,1)
     [,1]
[1,]    1
[2,]    2
[3,]    3
```

```
> # 5 graphs : 3 lignes, 2 colonnes et 1 gros graph en bas
>      matrix(c(1,2,3,4,5,3),3,2)
     [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    3
>
```

```
 # On met le tout dans la fonction layout
layout(matrix(c(1,2,3,4,5,3),3,2))

# On redirige vers un fichier
jpeg("exLAYOUT.jpeg")

# On trace les graph
hist(index$ISU,col="forestgreen")
barplot(CompoTOPpays,col=colPAYS)
barplot(CompoTOP,col=terrain.colors(nrow(CompoTOP)))
barplot(table(tot$ANNEE),col=rainbow(nlevels(tot$ANNEE)))
boxplot(tot$INEL~tot$GROUPE,col=colPAYS)

# On cloture le fichiers
dev.off()
```

 # On peut aussi définir les marges intérieures et extérieures : par(mar(c(a,b,c,d),oma(i,j,k,l))


Histogram of index$ISU
Frequency
0
200
400
600
800
100
150
200
index$ISU
0
200
400
600
800
1000
1200
2007
2008
2009
2010
2011
2012
2013
2014
2015
0
5
10
15
20
25
30
35
DEU
DFS
ESP
FRA
NLD
-20
0
20
40
60
80
100
DEU
DFS
ESP
FRA
NLD
0
5
10
15
20
25
30
35
DEU
DFS
ESP
FRA
NLD

Lancer un script R avec passage d'arguments

Rscript

 # Dans le terminal avec Rscript

 # On indique script arg1 arg2 et > nom de la log

```
Rscript scriptEXEMPLE_TD_BATCH.r 46 lgf  > Rscript.log
```

 # Dans le script R avec Rscript

```
# On imprime les arguments
print(commandArgs())
```

 # Il y a un certain nombre d arguments par défaut

```
[1] "/bao/R/3.0.2/lib64/R/bin/exec/R" "--slave"
[3] "--no-restore"                    "--file=scriptEXEMPLE_TD_BATCH.r"
[5] "--args"                          "46"
[7] "lgf"
```

 # Les arguments que l'on veut sont en 6 et 7

```
# On lit argument 6 et 7
race <- commandArgs()[6]
car <- commandArgs()[7]

print(paste("on travaille race",race,"car",car))
```

```
[1] "on travaille race 46 car lgf"
```

 # La log est redirigé mais ne contient
que ce qu'on lui demande d'imprimer
 # Elle ne contiendra pas les éventuels messages
d'erreur.

R CMD BATCH

 # Dans le terminal avec R CMD BATCH

 # On indique -arg1 -arg2 script et nom de la log

```
R CMD BATCH -46 -lgf scriptEXEMPLE_TD_BATCHrcmdbatch.r
```

 # Dans le script R avec R CMD BATCH

```
# On imprime les arguments
print(commandArgs())
```

 # Le nombre et la valeur des arguments par défaut
est différente : save / restore

```
> # On imprime les arguments
> print(commandArgs())
[1] "/bao/R/3.0.2/lib64/R/bin/exec/R"    "-f"
[3] "scriptEXEMPLE_TD_BATCHrcmdbatch.r" "--restore"
[5] "--save"                             "--no-readline"
[7] "-46"                                "-lgf"
```

 # Les arguments que l'on veut sont en 7 et 8 et ont des tirets

```
# On lit argument 7 et 8
arg <- commandArgs()[7]
race <- as.character(substr(arg, 2,100))
arg <- commandArgs()[8]
car <- as.character(substr(arg, 2,100))

print(paste("on travaille race",race,"car",car))
```

```
> print(paste("on travaille race",race,"car",car))
[1] "on travaille race 46 car lgf"
> proc.time()
   user  system elapsed
  0.271   0.058   0.338
```

 # La log se place dans le fichier spécifié et contient tout.

 # Selon la méthode voulu : le script R est différent

Lancer un script R avec passage d'arguments

Pour avoir une solution qui marche dans les 2 cas

```
# Se lance avec
# Rscript EXEMPLE_TD_BATCH_UNIVERSEL.r 46 lgf > RscriptUNIVERSEL.log
# R CMD BATCH --no-save --no-restore '--args 66 "lgf"' EXEMPLE_TD_BATCH_UNIVERSEL.r

# On imprime les arguments
print(commandArgs())


# L option trailing only permet de ne
# retourner que les arguments apres args
args=(commandArgs(trailingOnly=TRUE))

print(args)

race <- args[1]
car <- args[2]

print(paste("on travaille race",race,"car",car))
```

```
> race <- args[1]
> car <- args[2]
>
> print(paste("on travaille race",race,"car",car))
[1] "on travaille race 66 car \"lgf\""
>
```

```
args=(commandArgs(TRUE))

# On imprime les arguments
print(commandArgs())
[1] "/bao/R/3.0.2/lib64/R/bin/exec/R"  "-f"
[3] "EXEMPLE_TD_BATCH_UNIVERSEL.r"     "--restore"
[5] "--save"                           "--no-readline"
[7] "--no-save"                        "--no-restore"
[9] "--args"                           "66"
11] "\"lgf\""
```

Un peu galère mais possible de faire passer des variables avec R CMD BATCH

```
choze@dga12:~/TD_R# BREED=46
choze@dga12:~/TD_R# car="lait"
choze@dga12:~/TD_R#
echo R CMD BATCH --no-save --no-restore \'--args race=$BREED car=\"$trait\"\'  EXEMPLE_TD_BATCH_UNIVERSEL.r EXEMPLE_TD_BATCH_UNIVERSEL.log > lanceR.sh
choze@dga12:~/TD_R# choze@dga12:~/TD_R# more lanceR.sh
R CMD BATCH --no-save --no-restore '--args race=46 car="lait "' EXEMPLE_TD_BATCH_UNIVERSEL.r EXEMPLE_TD_BATCH_UNIVERSEL.log
choze@dga12:~/TD_R# sh lanceR.sh
```

Aller + loin : Conditions et boucles
Somme des probabilités par intervalles

```
> ### On somme les proba par intervalle de 100 SNP
> NBw=100
> # On cree une variable vide
> analyseSNP$sum <- as.numeric("")
```

```
> # On regarde les colonnes qu on veut
> head(analyseSNP[,c(1,5,11,13)])
  level         p       V3 sum
1     1 0.0000750 53981154  NA
2     2 0.0001375 53981307  NA
3     3 0.0000500 53981324  NA
4     4 0.0000625 53981328  NA
5     5 0.0000875 53981371  NA
6     6 0.0001000 53981391  NA
```

```
> # On boucle sur tous les SNP
> for(i in c(1:nrow(analyseSNP)) #dim(analyseSNP)[1])
+   {
+         # On recupere les num de ligne
+         # des SNP situe NBw/2 avant et apres
+         d <- (i-NBw/2)
+         f <- (i+NBw/2)
```

```
      # Si la valeur est negative, on est
      # avant le debut de l intervalle donc on remet 1
    if(d<1)
      {
        d=1
      }
      # Si la valeur est superieure au nb de ligne, on est
      # apres le debut de l intervalle donc on remet le max
    if(f>nrow(analyseSNP))
      {
        f=nrow(analyseSNP)
      }
```

```
> head(analyseSNP[,c(1,5,11,13)])
  level         p       V3       sum
1     1 0.0000750 53981154 0.0060000
2     2 0.0001375 53981307 0.0061250
3     3 0.0000500 53981324 0.0062875
4     4 0.0000625 53981328 0.0064000
5     5 0.0000875 53981371 0.0065000
6     6 0.0001000 53981391 0.0065750
```

```
      # On recupere les positions min et max
    SNPdebut <- analyseSNP[d,11]
    SNPfin <- analyseSNP[f,11]
+
+     # On somme les proba (col 5) entre les positions du SNP min et max
+     # On remplit la colonne sum
+     analyseSNP$sum[i] <- sum(analyseSNP[which((analyseSNP[,11]>=SNPdebut)&(analyseSNP[,11]<=SNPfin)),5])
+
+   }
```

Modulo et appartenance à un vecteur

 # On imprime si i appartient au vecteur
Si on veut regarder si ce que l on fait marche, on affiche les valeurs pour certaines lignes

```
if (i %in% c(1,2,25128,25129))
  {
   print(c("i",i,"d",d,"f",f,"posD",SNPdebut,"posF",SNPfin))
 }
```

```
[1] "i"     "1"        "d"     "1"        "f"     "51"
[7] "posD"  "53981154" "posF"  "53983079"
[1] "i"     "2"        "d"     "1"        "f"     "52"
[7] "posD"  "53981154" "posF"  "53983221"
```

```
[1] "i"     "25128"    "d"     "25078"    "f"     "25129"
[7] "posD"  "55977922" "posF"  "55981298"
[1] "i"     "25129"    "d"     "25079"    "f"     "25129"
[7] "posD"  "55978017" "posF"  "55981298"
```

 # On imprime les i multiples de 1000

```
# Pour suivre l avancement on peut imprimer ou il en est et l heure
if(i%%1000==0)
  {
    print(paste("SNP",i,Sys.time()))
  }
```

```
[1] "SNP 1000 2016-02-22 17:07:15"
[1] "SNP 2000 2016-02-22 17:07:23"
[1] "SNP 3000 2016-02-22 17:07:32"
[1] "SNP 4000 2016-02-22 17:07:40"
[1] "SNP 5000 2016-02-22 17:07:49"
[1] "SNP 6000 2016-02-22 17:07:58"
[1] "SNP 7000 2016-02-22 17:08:07"
[1] "SNP 8000 2016-02-22 17:08:16"
[1] "SNP 9000 2016-02-22 17:08:25"
[1] "SNP 10000 2016-02-22 17:08:33"
[1] "SNP 11000 2016-02-22 17:08:42"
[1] "SNP 12000 2016-02-22 17:08:51"
[1] "SNP 13000 2016-02-22 17:09:00"
```

```
[1] "SNP 14000 2016-02-22 17:09:09"
[1] "SNP 15000 2016-02-22 17:09:17"
[1] "SNP 16000 2016-02-22 17:09:26"
[1] "SNP 17000 2016-02-22 17:09:35"
[1] "SNP 18000 2016-02-22 17:09:44"
[1] "SNP 19000 2016-02-22 17:09:53"
[1] "SNP 20000 2016-02-22 17:10:02"
[1] "SNP 21000 2016-02-22 17:10:11"
[1] "SNP 22000 2016-02-22 17:10:21"
[1] "SNP 23000 2016-02-22 17:10:33"
[1] "SNP 24000 2016-02-22 17:10:42"
[1] "SNP 25000 2016-02-22 17:10:51"
```

 # 4mn pour 25 000 lignes c'est long !!! Il vaut mieux éviter les boucles en R…

Apply et autres fonctions (1/3)

 # Principe de base : appliquer une même fonction à plusieurs éléments

```
> # On cree une matrice
> mat = matrix(1:15,3,5)
> mat
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    4    7   10   13
[2,]    2    5    8   11   14
[3,]    3    6    9   12   15
```

```
> # APPLY = travailler par ligne (1) ou colonne (2)
>
> ## Pour recuperer la somme par colonne ou par ligne
> apply(X=mat,MARGIN=1,FUN=sum)
[1] 35 40 45
> apply(mat,2,FUN=mean)
[1]  2  5  8 11 14
```

 # Il existe d'autres fonctions de la famille des apply, lapply, sapply…
 # Les différences se font sur le format d'entrée et de sortie

```
> # Pour connaitre le type de variable
>
> # apply en travaillant par colonnes
> apply(tot[,c(1,2,6,10)],2,class)
        ANI        PAYS          dn        INEL
"character" "character" "character" "character"
>
> # ça ne marche pas car apply travaille
> # sur des matrices: tous éléments de même nature
```

```
> ## lapply travaille par liste
> ## chaque colonne est un element de la liste
> lapply(tot[,c(1,2,6,10)],class)
$ANI
[1] "character"

$PAYS
[1] "factor"

$dn
[1] "Date"

$INEL
[1] "numeric"
```

```
> # sapply idem mais renvoie un vecteur
> sapply(tot[,c(1,2,6,10)],class)
        ANI        PAYS          dn        INEL
"character"    "factor"      "Date"   "numeric"
>
```

Apply et autres fonctions (1/3)

 # La fonction by pour les statistiques par groupe fait partie de cette famille
 # Il y a aussi tapply et aggregate

```
> # Fonction by (vu tout à l heure)
> by(tot$INEL,tot$GROUPE,mean)
tot$GROUPE: DEU
[1] 29.73991
------------------------------------
tot$GROUPE: DFS
[1] 29.62041
------------------------------------
tot$GROUPE: ESP
[1] 23.75912
------------------------------------
tot$GROUPE: FRA
[1] 29.55258
------------------------------------
tot$GROUPE: NLD
[1] 30.72154
```

```
> # tapply travaille par liste et renvoie un vecteur
> tapply(tot$INEL,list(tot$GROUPE),mean)
     DEU      DFS      ESP      FRA      NLD
29.73991 29.62041 23.75912 29.55258 30.72154
>
```

```
> # aggregate travaille par liste et renvoie un tableau
> aggregate(tot$INEL,list(tot$GROUPE),mean)
  Group.1        x
1     DEU 29.73991
2     DFS 29.62041
3     ESP 23.75912
4     FRA 29.55258
5     NLD 30.72154
>
```

 # Mon préféré : aggregate : on peut fusionner et exporter résultat directement

```
> # On fait moyenne et ecart types
> moy <- aggregate(tot$INEL,list(tot$GROUPE),mean)
> sd <- aggregate(tot$INEL,list(tot$GROUPE),sd)
>
> # On fusionne, on renomme les colonnes
> statPAYS <- merge(moy,sd,by="Group.1")
> names(statPAYS)<- c("PAYS","moy","sd")
```

```
> statPAYS
  PAYS      moy       sd
1  DEU 29.73991 16.56026
2  DFS 29.62041 15.25887
3  ESP 23.75912 16.79515
4  FRA 29.55258 15.54295
5  NLD 30.72154 14.85039
>
```

```
> ## Et on l exporte
> write.table(statPAYS,"StatPAYS.txt",quote=F,col.names=T,row.names=F)
```

Apply et autres fonctions (3/3)

 # Là ou ça devient intéressant c'est qu'on peut créer ses propres fonctions

```
# Renvoyer directement moyenne, ecart type, min et max
 function(x){return(c(mean(x),sd(x),min(x),max(x)))
```

```
> aggregate(tot$INEL,list(tot[,"GROUPE"]),function(x){return(c(mean(x),sd(x),min(x),max(x)))})
  Group.1       x.1       x.2       x.3       x.4
1     DEU  29.73991  16.56026 -28.00000 100.00000
2     DFS  29.62041  15.25887 -15.00000  80.00000
3     ESP  23.75912  16.79515 -31.00000  75.00000
4     FRA  29.55258  15.54295 -18.00000  76.00000
5     NLD  30.72154  14.85039 -15.00000  77.00000
```

 # On peut créer la fonction en dehors du apply/aggregate

 # Ex : la somme des probabilités par intervalle

```
sumWINDOW <- function(X,data=resSOMME,window=nbSNP)
  {
    # On travaille sur la colonne level
    # qui correspond aux numeros de lignes
    pos <- as.numeric(X[2])
    idD <- pos - window/2
    idF <- pos + window/2
    idD[idD<1]=1
    idF[idF>nrow(data)] <- nrow(data)
    return(c(sum(data$p[idD:idF])))
  }
```

 # 3 sec contre 4 mn pour la boucle

```
> Sys.time()
[1] "2016-03-02 11:52:52 CET"
> # On applique
> SOMMEproba <- apply(resSOMME,1,sumWINDOW)
> # On fait une colonne avec les resultats
> resSOMME$SOMME  <- SOMMEproba
>
> Sys.time()
[1] "2016-03-02 11:52:54 CET"
```

```
> ## On verifie qu on a les mêmes resultats
>
> summary(resSOMME$sum - resSOMME$SOMME)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
      0       0       0       0       0       0
```

 # Un exemple en + sur un test de proportion ligne à ligne dans le script

Commandes assign(), get()
Import de fichiers avec des noms similaires

```
> for (BTA in 28:29)
+   {
+     # On imprime pour suivre l avancement
+     print(paste("on importe le chromosome",BTA))
+     # On lit le dataset
+     data <- read.table ( paste("OKres",BTA,".mlma",sep="") )
+     # On prepare le nom du jeu de donnes
+     NAMEdata <- paste("GWAS",BTA,sep="_")
+     print(NAMEdata)
+     # On assigne à NAMEdata le jeu de donnees data
+     assign(NAMEdata,data)
+     # Pour regarder la taille du jeu de donnees
+     print(dim(data))
+     print(dim(get(NAMEdata)))
+
+   }
[1] "on importe le chromosome 28"
[1] "GWAS_28"
[1] 335474      6
[1] 335474      6
[1] "on importe le chromosome 29"
[1] "GWAS_29"
[1] 412621      6
[1] 412621      6
> dim(GWAS_28);dim(GWAS_29)
[1] 335474      6
[1] 412621      6
>
```

 # On fait une boucle

 # On donne un nom temporaire

 # On donne le nouveau nom

 # On regarde la taille du dataset
 # get est l inverse de assign, chercher le jeu de données qui a le nom demandé

```
> # Tester l existence d un objet
> exists("BTA")
[1] TRUE
> exists("GWAS_28")
[1] TRUE
>
> exists("GWAS_5")
[1] FALSE
>
```

Commandes système

ls(), rm(), file.exist(), file.info(), file.copy,file.remove(),dir.create(),SysGlob(), system()

Pour regarder tous les objets de la session

```
> ls()
 [1] "BTA"        "car"        "corPAYS"    "data"       "dataBLANK"
 [6] "dataNA"     "dataSANSna" "dataTEST"   "file"       "GWAS_28"
[11] "GWAS_29"    "i"          "j"          "NAMEdata"   "nPAYS"
[16] "race"       "REP"        "repGS3"     "year"
```

Pour effacer un élément

```
> rm(data)
> ls()
 [1] "BTA"        "car"        "corPAYS"    "dataBLANK"  "dataNA"
 [6] "dataSANSna" "dataTEST"   "file"       "GWAS_28"    "GWAS_29"
[11] "i"          "j"          "NAMEdata"   "nPAYS"      "race"
[16] "REP"        "repGS3"     "year"
```

```
> # Tester l existence d un fichier
> file.exists("/big/C3/BAYESCpi/r46/RES/lgf/BayesCpi_sol_lgf_poly90_17")
[1] TRUE
>
> # Regarder date création, modif, droit d access...
> file.info("/big/C3/BAYESCpi/r46/RES/lgf/BayesCpi_sol_lgf_poly90_17")
                                                          size isdir mode
/big/C3/BAYESCpi/r46/RES/lgf/BayesCpi_sol_lgf_poly90_17 3166623 FALSE  777
                                                               mtime
/big/C3/BAYESCpi/r46/RES/lgf/BayesCpi_sol_lgf_poly90_17 2016-01-07 17:18:12
                                                               ctime
/big/C3/BAYESCpi/r46/RES/lgf/BayesCpi_sol_lgf_poly90_17 2016-01-18 10:33:55
                                                               atime uid
/big/C3/BAYESCpi/r46/RES/lgf/BayesCpi_sol_lgf_poly90_17 2016-02-24 11:52:38 503
                                                             gid     uname grname
/big/C3/BAYESCpi/r46/RES/lgf/BayesCpi_sol_lgf_poly90_17 232 pcroiseau    g2b
```

```
> # lister tous les fichiers d un repertoire
> Sys.glob("/big/C3/BAYESCpi/r46/RES/lgf/*")
 [1] "/big/C3/BAYESCpi/r46/RES/lgf/BayesCpi_sol_lgf_poly90_1"
[74] "/big/C3/BAYESCpi/r46/RES/lgf/RES_solGS3_GWASlgf7poly90pi04.csv"
[75] "/big/C3/BAYESCpi/r46/RES/lgf/RES_solGS3_GWASlgf8poly90pi04.csv"
[76] "/big/C3/BAYESCpi/r46/RES/lgf/RES_solGS3_GWASlgf9poly90pi04.csv"
> # lister ceux avec un pattern specifique
> Sys.glob("/big/C3/BAYESCpi/r46/RES/lgf/*sol*17*")
[1] "/big/C3/BAYESCpi/r46/RES/lgf/BayesCpi_sol_lgf_poly90_17"
[2] "/big/C3/BAYESCpi/r46/RES/lgf/Graph_solGS3_GWASlgf17poly90pi04.png"
[3] "/big/C3/BAYESCpi/r46/RES/lgf/RES_solGS3_GWASlgf17poly90pi04.csv"
>
```

```
> ### Creer un repertoire
> dir.create("/home/choze/TD_R/NEW_DIR")
Warning message:
In dir.create("/home/choze/TD_R/NEW_DIR") :
  '/home/choze/TD_R/NEW_DIR' already exists
> setwd("/home/choze/TD_R/NEW_DIR")
> # Copier un fichier
> file.copy("/big/C3/BAYESCpi/r46/RES/lgf/BayesCpi_sol_lgf_poly90_17","BayesCpi_sol_lgf_17")
[1] TRUE
> # Regarder ce qu il y a dans rep courant
> system("ls")
BayesCpi_sol_lgf_17  index.csv  liste_bull.csv
>  # Effacer un fichier
> file.remove("/home/choze/TD_R/NEW_DIR//BayesCpi_sol_lgf_17")
[1] TRUE
```

Installer un package / Lire une base SAS

```
# Installer un package
install.packages("haven")

# Option lib : Emplacement où l'on veut sauvegarder le package  par défaut
# Option repos pour ne pas avoir à cliquer pour choisir un miroir CRAN

install.packages("haven", repos="http://cran.us.r-project.org" , lib="/bao/R/3.2.3/library")
```

```
# sous R323

# On appelle le package haven
library(haven)
# On lit le fichier avec la commande read_sas
data <- read_sas("efftau12.sas7bdat")
```

```
> summary(data)
      solg              efsex               ntau              effiag
 Min.   :-8.310   Length:1506        Length:1506        Min.   :  10.0
 1st Qu.:-1.246   Class :character   Class :character   1st Qu.:  26.0
 Median : 0.000   Mode  :character   Mode  :character   Median :  54.0
 Mean   : 0.000                                        Mean   : 119.6
 3rd Qu.: 1.336                                        3rd Qu.: 114.0
 Max.   : 8.101                                        Max.   :1253.0
 NA's   :585                                           NA's   :585
```

```
    camp                solv               effiav
Length:1506        Min.   :-9.13146   Min.   :  10.0
Class :character   1st Qu.:-1.35558   1st Qu.:  20.0
Mode  :character   Median : 0.04691   Median : 100.0
                   Mean   : 0.00001   Mean   : 198.2
                   3rd Qu.: 1.34763   3rd Qu.: 207.8
                   Max.   : 8.61130   Max.   :2665.0
                   NA's   :36         NA's   :36
>
```

Et maintenant à vous de jouer !

Tout le script sous DGA12 et DGA20

/home/choze/TD_R/script_TD.r