Initiation à R, logiciel de Statistiques

I. Plusieurs façons de travailler
(… ou de s'enquiquiner)

II. R, logiciel de programmation en Statistiques

III. Votre premier projet en R, pas à pas

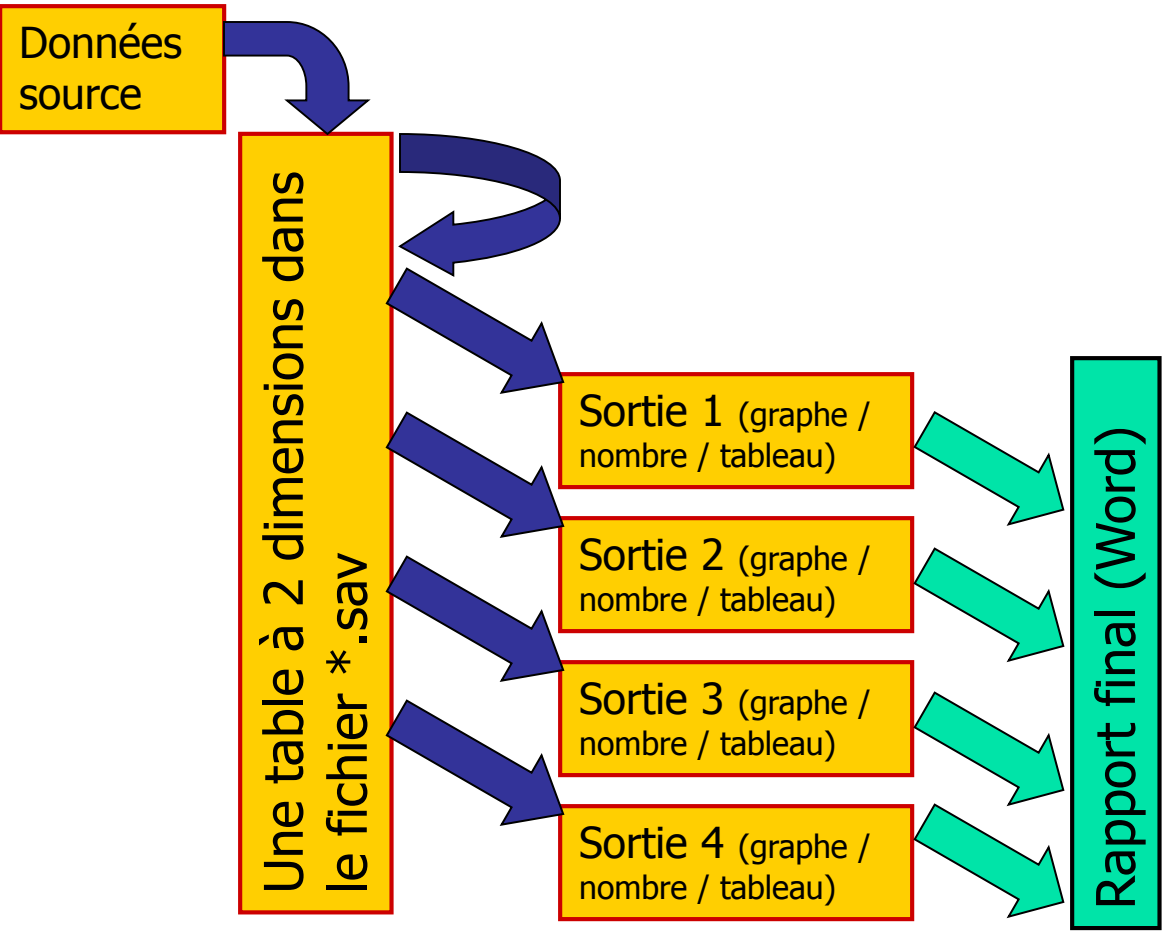
Dr Emmanuel Chazard
MCU-PH Université Lille 2
http://emmanuel.chazard.org

Plusieurs façons de travailler
(ou de s'enquiquiner…)

I. Exemple de SPSS
II. Exemple de SASS
III. L'esprit R !

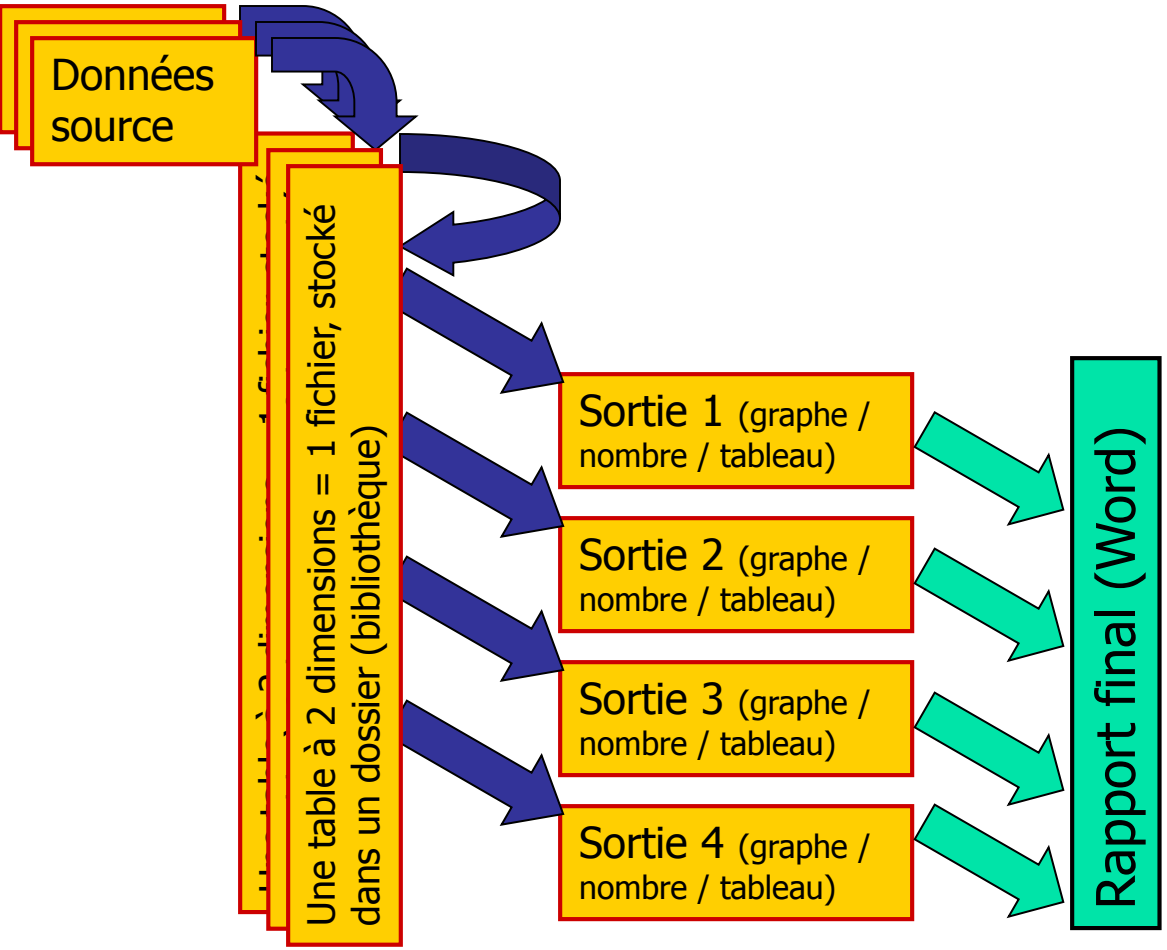
Dr Emmanuel Chazard
MCU-PH Université Lille 2
http://emmanuel.chazard.org

Plusieurs façons de travailler : exemple de SPSS



- Les données sont chargées à la main => une seule table à 2 dimensions
- Plusieurs actions de modification (« compute »)
- Plusieurs actions de sortie statistique (à la souris, ou en script mémorisable)
- Les résultats doivent être exportés pas à pas vers un fichier Word par exemple
- Traitements limités à une table à 2 dimensions, étude très classique.
- Selon les changements de stratégie ou de données, vous reprendrez à des étapes variables

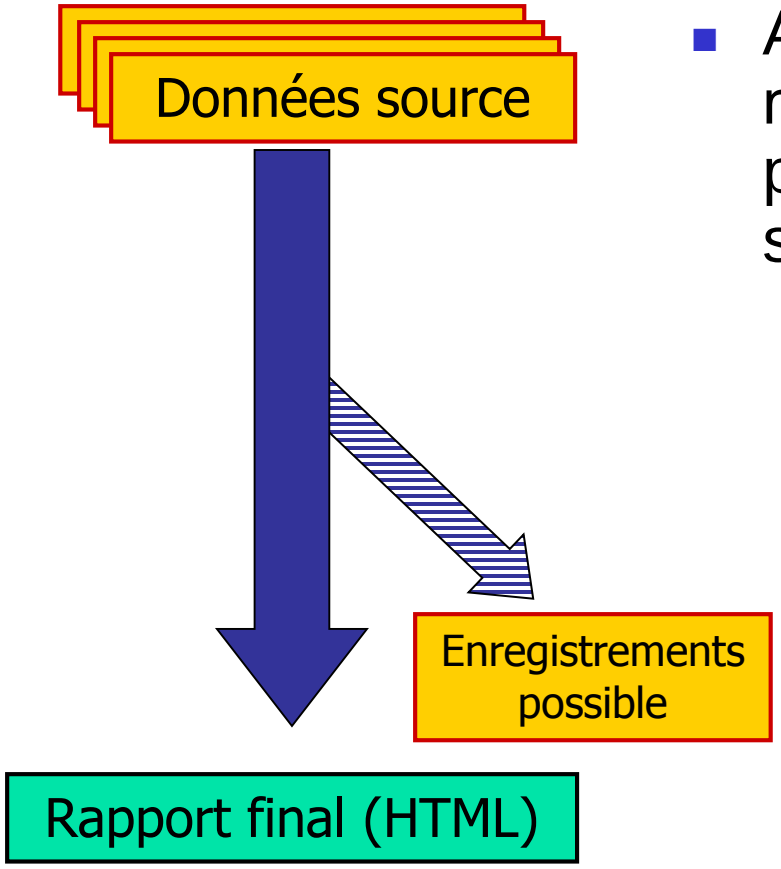
Plusieurs façons de travailler : exemple de SAS



Le principe reste le même, sauf que :

- Langage de commandes, tant pour le chargement et les calculs (DATA) que pour les sorties (PROC)
- Il est possible de travailler sur plusieurs tables simultanément
- Piètre qualité des sorties graphiques, tableaux mal présentés
- Traitements limités à plusieurs tables à 2 dimensions, langage de macro assez rigide
- Selon les changements de stratégie ou de données, vous reprendrez à des étapes variables (mais historisation)

Plusieurs façons de travailler : l'esprit R !



- Avec R, on pourrait travailler de même. A l'aide de R2HTML, il est possible avec un seul script et une seule action de :
  - Charger les données
  - Les modifier
  - Générer toutes les sorties, mises en forme dans un rapport HTML
  - Relancer le processus complet en 2 secondes à chaque modification des données ou des calculs
  - Ne pas utiliser de stockage intermédiaire (mais cela reste possible)

R, logiciel de programmation en Statistiques

I. Statistiques et programmation
II. Installer R pour Windows
III. Les packages

Dr Emmanuel Chazard
MCU-PH Université Lille 2
http://emmanuel.chazard.org

R, un logiciel de Statistiques, et un langage de programmation

- Les avantages d'un langage de programmation :
  - Possibilités infinies, totale liberté de création, puissance algorithmique, possibilité d'inventer vos propres graphiques et fonctions…
  - Un apprentissage utile (C, JavaScript, PHP, Java, Perl…)
- Les avantages d'un logiciel de Statistiques :
  - Langage moins « bas niveau », notamment pour vecteurs, tableaux et graphiques
  - Classes et fonctions clefs en main
  - Sorties graphiques très réussies et paramétrables
- Les avantages d'un projet communautaire :
  - Gratuit, facile à installer, léger en ressources
  - Très complet, très nombreuses extensions (librairies), reconnu dans le monde universitaire

Installer R sous Windows

- Site de référence :

  http://www.r-project.org

- Téléchargement du fichier binaire d'installation pour Windows :

  download > CRAN > R Binaries

  http://cran.cict.fr/bin/windows/base/R-2.10.1-win32.exe (32 Mo)

- R peut être installé sur une clef USB, et sur certains PC sans les droits administrateur

Installer et utiliser des packages

- Installer un package :
  - Voie automatique : menu interactif de R

    Menu packages > installer le package
  - Voie semi-automatique : télécharger le fichier ZIP sur le site http://www.cran-project.org

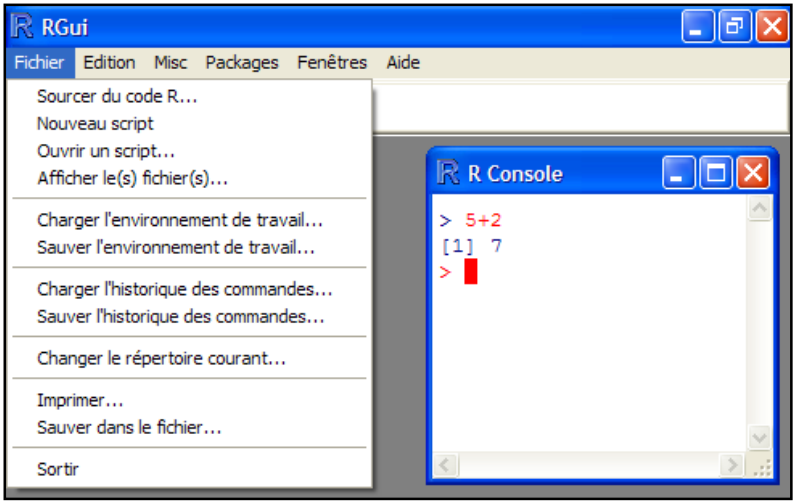
    (*download > CRAN > Packages*) puis

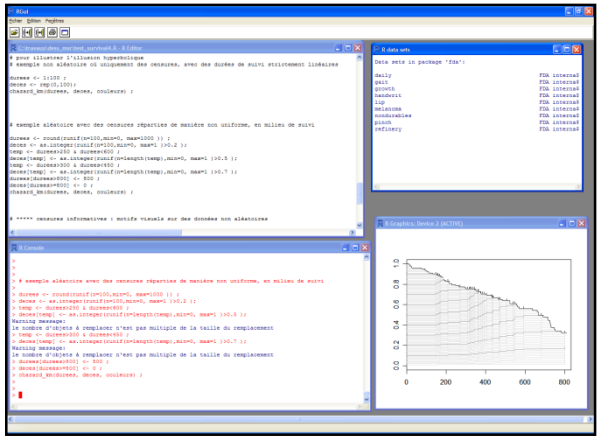
    Menu packages > installer le package depuis le fichier ZIP
- Utiliser un package :
  - Une fois installé, le package doit être chargé par la commande *library(nomdupackage)*
  - Certains packages contiennent des jeux de données. Voir pour ce faire *data(nomdupackage)*

Votre premier projet en R, pas à pas

I. La console,
II. Les variables, print()
III. Les fonctions, l'aide
IV. Quelques mots sur le langage lui-même
V. Chargement de données
VI. Les vecteurs
VII. Graphiques exploratoires
VIII. La boucle FOR
IX. Génération d'un rapport complet avec R2HTML

La console R





- La console :
  - reçoit vos commandes
  - affiche parfois des sorties texte
  - « Fichier > charger/sauver l'environnement de travail » permet de garder un état de la mémoire vive. Ceci est facultatif.
- La fenêtre de script
  - Permet de mémoriser / éditer / réutiliser votre script
  - *Contrôle* + *R* pour exécuter une portion de script (ligne courante ou portion sélectionnée)
- Attention : R est sensible à la casse

Les variables, print()

- Le type le plus simple : les scalaires, nombres ou chaînes de caractère.

Trop fastoche ?

OK, alors c'est à vous maintenant !

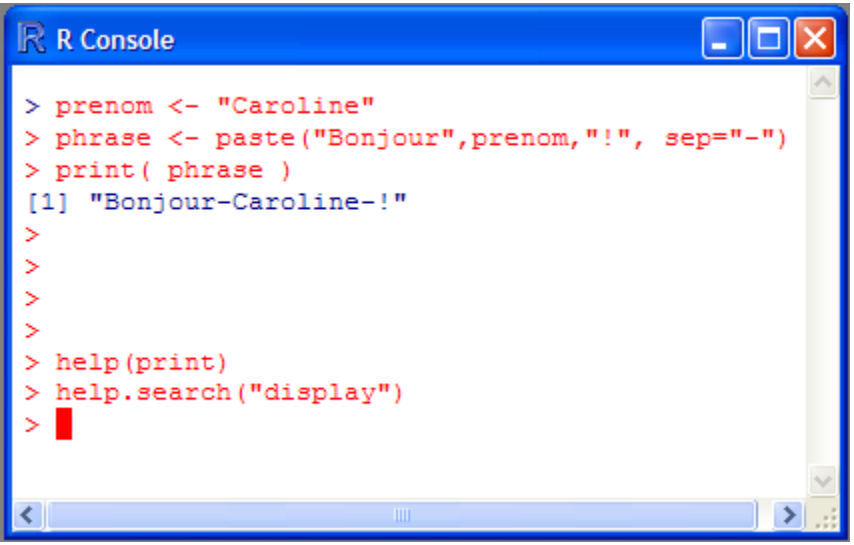
Petits rappels :

- Variable <- valeur
- print(valeur)
- paste(chaine1, chaine2…)

- J'affecte la valeur 13 à la variable *coucou*
- J'affiche *coucou*
- Si je saisis une expression sans l'affecter, alors le système comprend « *print()* »
- J'affecte le résultat du calcul « 7+15*2 » à *coucou*
- J'affiche *coucou*
- J'affiche le résultat de ce même calcul
- J'affecte la valeur « Emmanuel » à la variable *prenom*
- J'affiche *prenom*
- J'utilise la fonction *paste()* pour former « Bonjour Emmanuel ! » (en réalité print(paste(…)))

Les fonctions, l'aide

- Les fonctions :
  - S'appèlent par leur nom suivi de parenthèses
  - Entre les parenthèses, on peut placer des paramètres. Particularité de R, les paramètres peuvent être nommés => ordre indifférent
  - Elles font quelque chose, et retournent une valeur



- Deux fonctions utiles pour tout connaître sur les autres :
  - *help(nom)* ou *?nom*, si vous connaissez le nom de la fonction ou de la structure. (guillemets parfois nécessaires)
  - *help.search(motclef)* ou *??motclef* pour une recherche dans le texte (guillemets obligatoires)

Quelques mots sur le langage lui-même

- L'apparence : un langage procédural
  La réalité : une mécanique de POO
  - Classes, héritage, accès aux champs (names)…
  - Un *print()* implicite (toString)
  - Une redéfinition de nombreuses fonctions (*summary(), print(), plot()*…)
  - Une surdéfinition quasi-permanente (comportement des fonctions différent selon la nature des paramètres)
  - Des itérateurs (le contenu d'un objet de la classe *data.frame* se parcourt comme un tableau…)
- Vitesse d'exécution
  - R est programmé en C
  - Les fonctions natives programmées en C sont très rapides, l'exécution du script peut être plus lente => utilisez l'existant
  - Les itérations FOR sont plutôt lentes, alors que la manipulation directe de vecteurs est très rapide (utilisez des tableaux booléens)

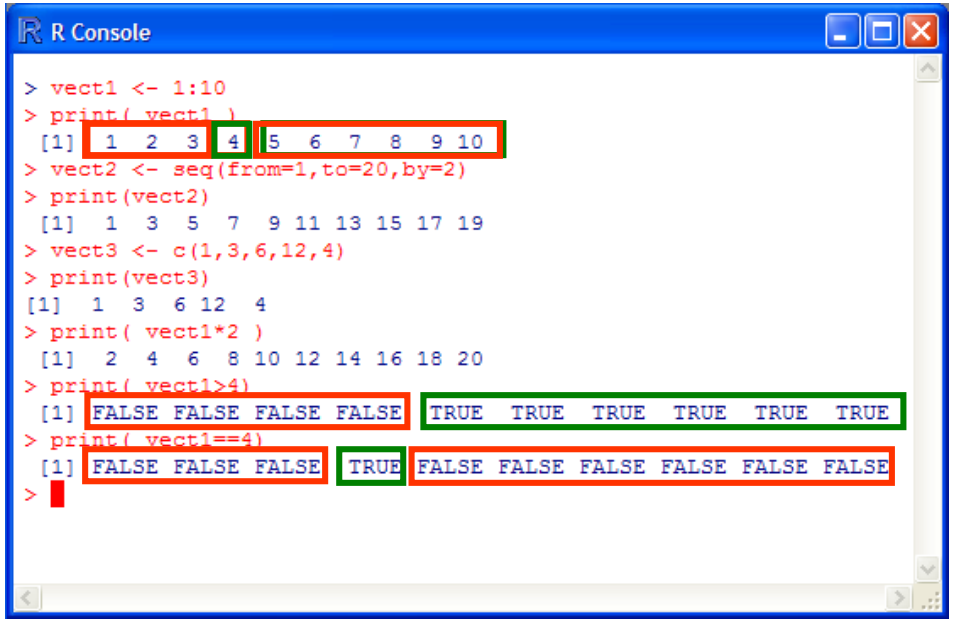
Chargement des données

- Ouvrez l'aide de la fonction *read.table()*
- Chargez les données contenues dans le fichier *data.txt* dans la variable *tableau* à l'aide de la fonction *read.table()* qui retourne un data.frame
- Explorez rapidement le contenu de *tableau* à l'aide des fonctions *print() str() summary() names()*
- On peut accéder au contenu d'un data.frame comme si c'était un tableau à 2 dimensions :

  tableau[sélection_de_lignes , sélection_de_colonnes]

Les vecteurs

- Dans un data.frame, chaque ligne ou chaque colonne constitue un vecteur (=liste de valeurs)



- Génération de vecteurs manuellement :
  - À l'aide de « : »
  - À l'aide de la fonction seq()
  - À l'aide de la fonction c()
- Opérations et comparaisons sur des vecteurs : l'opération s'applique à chacun des éléments et retourne un vecteur de même taille

Accès aux colonnes d'un data.frame

- Les data.frames :
  - sont des objets dont les champs sont les variables
  - toutefois tout est fait pour qu'ils ressemblent à des tableaux à 2 dimensions dont les lignes sont les individus et les colonnes sont les variables
- Trois façons d'accéder à une colonne :
  - *tableau[, 2]* (style tableau, déconseillé)
  - *tableau[, "sexe"]* (style tableau, conseillé car on peut mettre une variable à la place du nom de colonne)
  - *tableau$sexe* ou *tableau$"sexe"* (style objet : léger mais non dynamique)
- Pour accéder à plusieurs colonnes en même temps :
  - *tableau[, c(2,3)]*
  - *tableau[, 2:3]*
  - *tableau[, c("sexe", "calories")]*

Accès aux lignes d'un data.frame

- On peut accéder par le numéro de la ligne
  - *tableau[4 ,]*
  - *tableau[c(4,5,6) ,]*
  - *tableau[4:6 ,]*
- On peut accéder par un vecteur de sélection booléen, de la longueur du tableau :
  - *tableau[c(F,F,F,T,T,F) ,]*
  - *tableau[c(0,0,0,1,1,0) ,]*
  - *tableau[tableau$sexe==2 ,]*
    équivalent à SELECT * FROM tableau WHERE sexe=2 ;
  - *tableau[tableau[, "sexe"]==2 , "calories"]*
    équivalent à SELECT calories FROM tableau WHERE sexe=2 ;

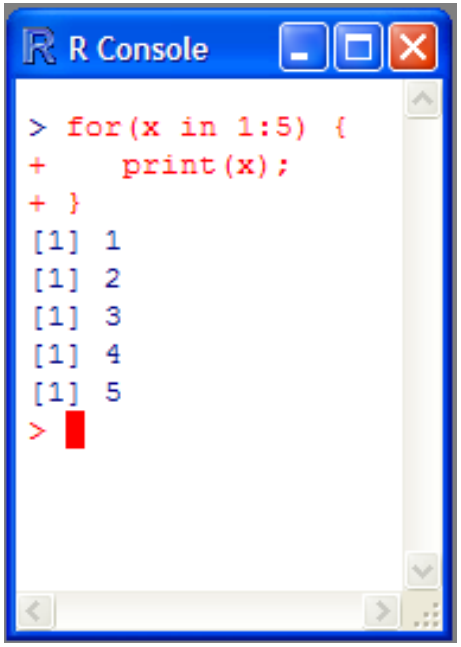
Exercices

- Les séquences :
  - Affichez l'aide de la fonction *seq()*
  - Générez un vecteur 10,13,16…34
- Les mesures du BMI chez les femmes :
  - Depuis le tableau *tableau*, générez un vecteur TRUE,FALSE,FALSE… indiquant si oui ou non la ligne concerne une femme (sexe=2)
  - Utilisez ce vecteur pour afficher, parmi le tableau *tableau*, les seules lignes concernant des femmes (NB : utilisez le vecteur booléen précédent comme sélecteur de lignes)
  - De même, listez les valeurs des BMI chez les hommes

Exercice : analyses univariées	Exercice : analyses bivariées	Exercice : réorganiser en fonctions

La boucle FOR

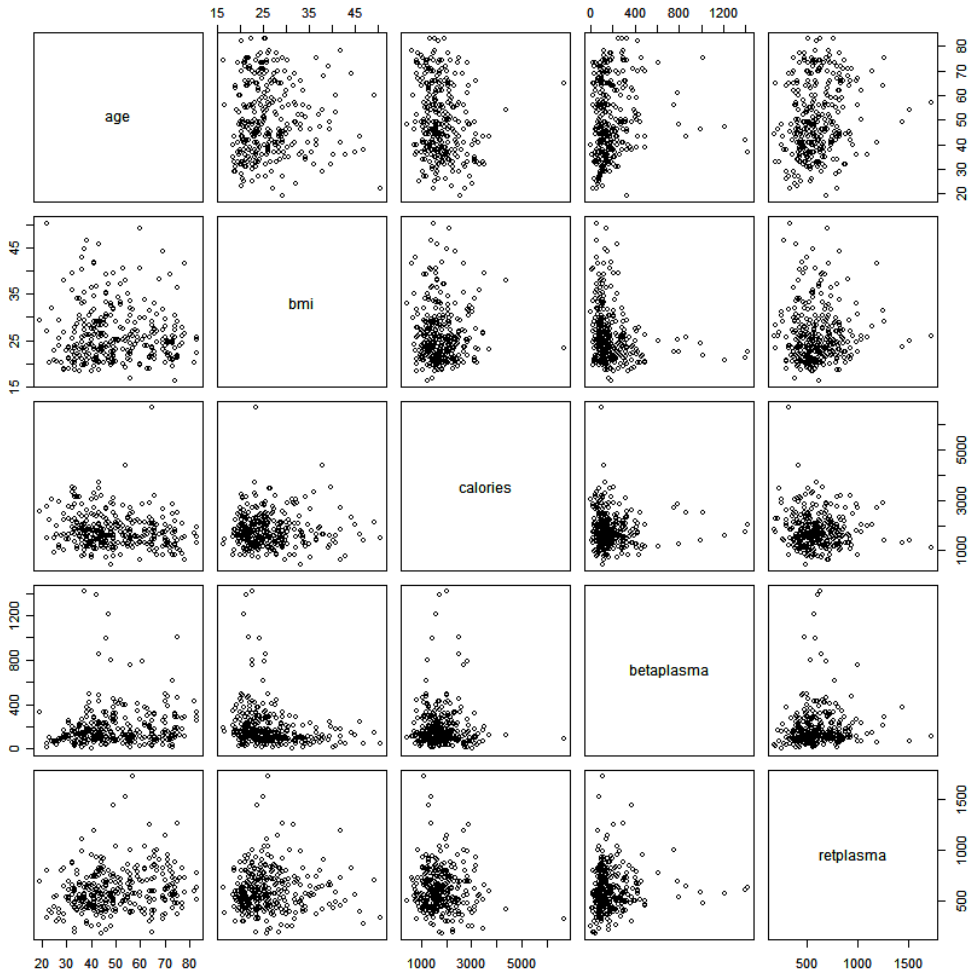
- La boucle FOR de R diffère des langages classiques (mais les vecteurs sont plus vite écrits)
- Structure :
  - Le mot clef « for »
  - Entre parenthèses :
    - le nom d'une nouvelle variable
    - le mot clef « in »
    - un vecteur de valeurs que cette variable prendra tour à tour
  - Entre accolades :
    - une série d'instructions qui sera répétée
    - À chaque itération de la boucle, la variable (ici x) prendra un nouvelle valeur dans le vecteur spécifié.



La boucle FOR : Exercice

- Mise en pratique : nous allons automatiser l'étude des variables quantitatives et qualitatives
- Générez un vecteur *noms_var_quanti_cont*
- Générez un vecteur *noms_var_quanti_disc*
- Générez un vecteur *noms_var_quali*
- Utilisez ces deux vecteurs pour encapsuler les scripts précédents dans des boucles

Pairs plot



- La fonction *pairs()* permet de générer des plots systématiques de toutes les variables quantitatives entre elles
- Appliquez cette fonction au jeu de données limité aux variables quantitatives définies dans *noms_var_quanti_cont*.

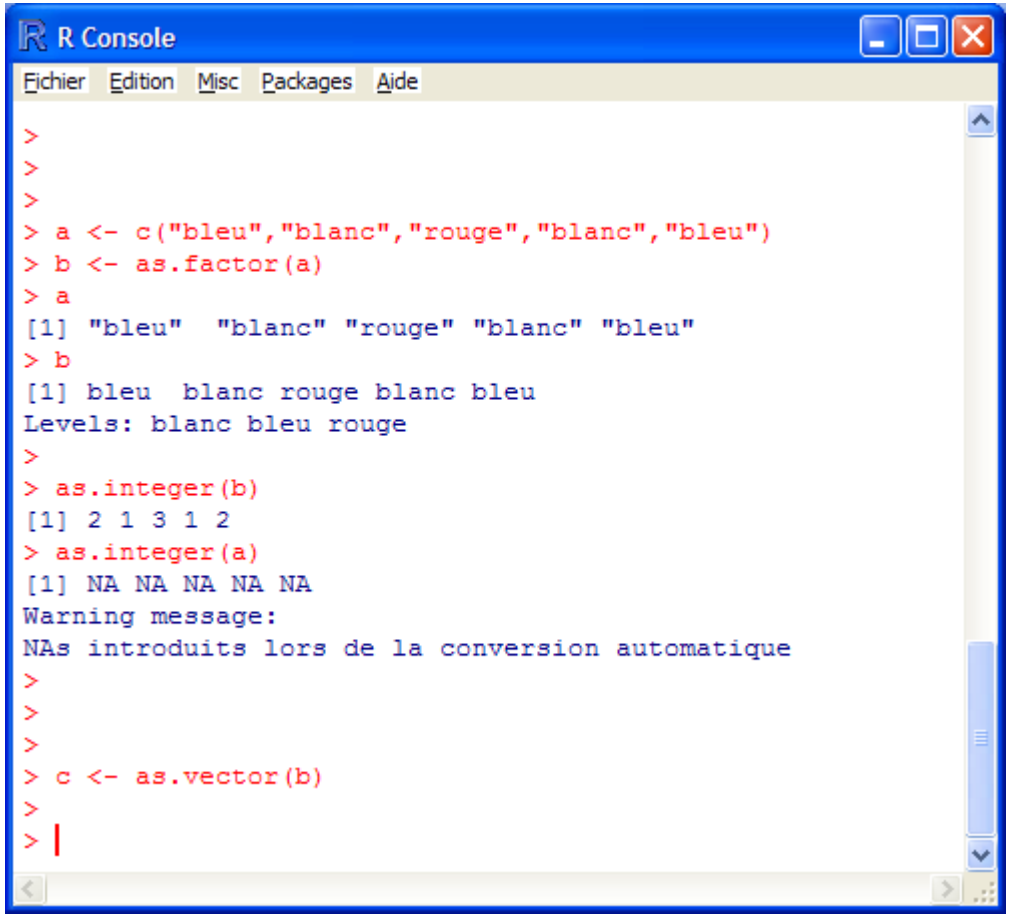
La génération d'un rapport complet avec R2HTML : Exercice

- Chargez la librairie R2HTML à l'aide de la fonction *library()* (écrivez cette ligne au début de votre script)
- En début de script, utilisez *HTMLStart()*
- En fin de script, utilisez *HTMLStop()*
- A la place de chaque *print()*, utilisez *HTML()*
- Pour les graphiques, générez-les d'abord comme d'habitude, puis invoquez la fonction *HTMLplot()*, en précisant :
  - le sous-titre (caption)
  - Le nom de l'image (GraphFileName), à générer avec *paste()* car il y a un bug, par exemple : img_univ_age, img_univ_nb_enf, etc.
- Pour insérer des titres, utilisez *HTML.title()*, où HR est un niveau de profondeur allant de 1 à 9
- Lancez le script et admirez…

Les objets en bref…

- La plupart des fonctions statistiques de R retournent des objets
- Fonctions et structures utiles sur les objets :
  - *print() summary() plot()* sont souvent redéfinis
  - *class()* permet de connaître la classe d'un objet
  - *names()* retourne le noms des champs, *monobjet$monchamp* permet d'afficher un champ

Vecteurs et facteurs…



- Par défaut, lors de la construction des dataframes, les colonnes comportant des caractères sont chargées en facteurs.
- Fonctions *as.vector()* et *as.factor()*
- Conseil : lors du chargement des données, forcer le chargement en vecteur (exemple pour read.table : *stringsAsFactors=FALSE* ou *as.is=TRUE*)