Vue d'ensemble
Le système
Un exemple
A l'aide !
Premiers pas
Les vecteurs
Tests
Les matrices
Affichage
La fin...pour aujourd'hui

INTRODUCTION AU LOGICIEL R

Variables et mise en jambe

Anne Dubois, Julie Bertrand, Emmanuelle Comets

emmanuelle.comets@inserm.fr

INSERM UMR738

Introduction à R - Novembre 2009

E. Comets

Vue d'ensemble
Le système
Un exemple
A l'aide !
Premiers pas
Les vecteurs
Tests
Les matrices
Affichage
La fin...pour aujourd'hui

Ou trouver R ?

Sur le site www.cran.r-project.org :

- dans la rubrique Software $\longrightarrow$ R binaries $\longrightarrow$ Windows $\longrightarrow$ base
- télécharger l'exécutable (.exe)
- lancer l'installation par un double clic et suivre les instructions

Vous pouvez télécharger des extensions :

- dans la rubrique Software $\longrightarrow$ Packages $\longrightarrow$ MonPackage
- télécharger MonPackage.zip
- installer le package dans R ("install from local zip")
- charger le package ("load package")

Introduction à R - Novembre 2009

E. Comets

Vue d'ensemble
Le système
Un exemple
A l'aide !
Premiers pas
Les vecteurs
Tests
Les matrices
Affichage
La fin...pour aujourd'hui

Présentation

- Le langage S
  - est un langage de programmation interactif et facile à apprendre
  - est un langage statistique contenant un très grand choix de techniques statistiques, y compris les plus récentes
  - est parmi les logiciels statistiques les plus utilisés (SAS, SPSS, STATISTICA,...)
  - dispose de modules et de librairies spécialisées disponibles sur Internet
- Mise en oeuvre de S
  - version commerciale : Splus
  - version graticielle : R
  - R a été fortement influencé par le langage S et Scheme, et il a été plus ou moins développé par des transfuges de Splus

Introduction à R - Novembre 2009

E. Comets

Vue d'ensemble
Le système
Un exemple
A l'aide !
Premiers pas
Les vecteurs
Tests
Les matrices
Affichage
La fin...pour aujourd'hui

Pourquoi utiliser R ?

- Séquence pub : R...
  - ... permet de faire des graphiques très flexibles et d'une qualité exceptionnelle
  - ... permet de construire facilement vos propres fonctions
  - ... est facile à interfacer avec d'autres langages
  - ... est aussi un logiciel mathématique (calcul matriciel, intégration numérique, optimisation, ...)
- Beaucoup de développements en R (librairies spécialisées)
- Pourquoi utiliser R plutôt que SAS ?
  - la plupart des analyses statistiques possibles en SAS sont aussi faisables en R
  - les graphes sont bien plus jolis
  - R est très facile à utiliser
  - R est gratuit
- Pourquoi utiliser R plutôt que Excel ?
  - Excel n'est pas un logiciel statistique !

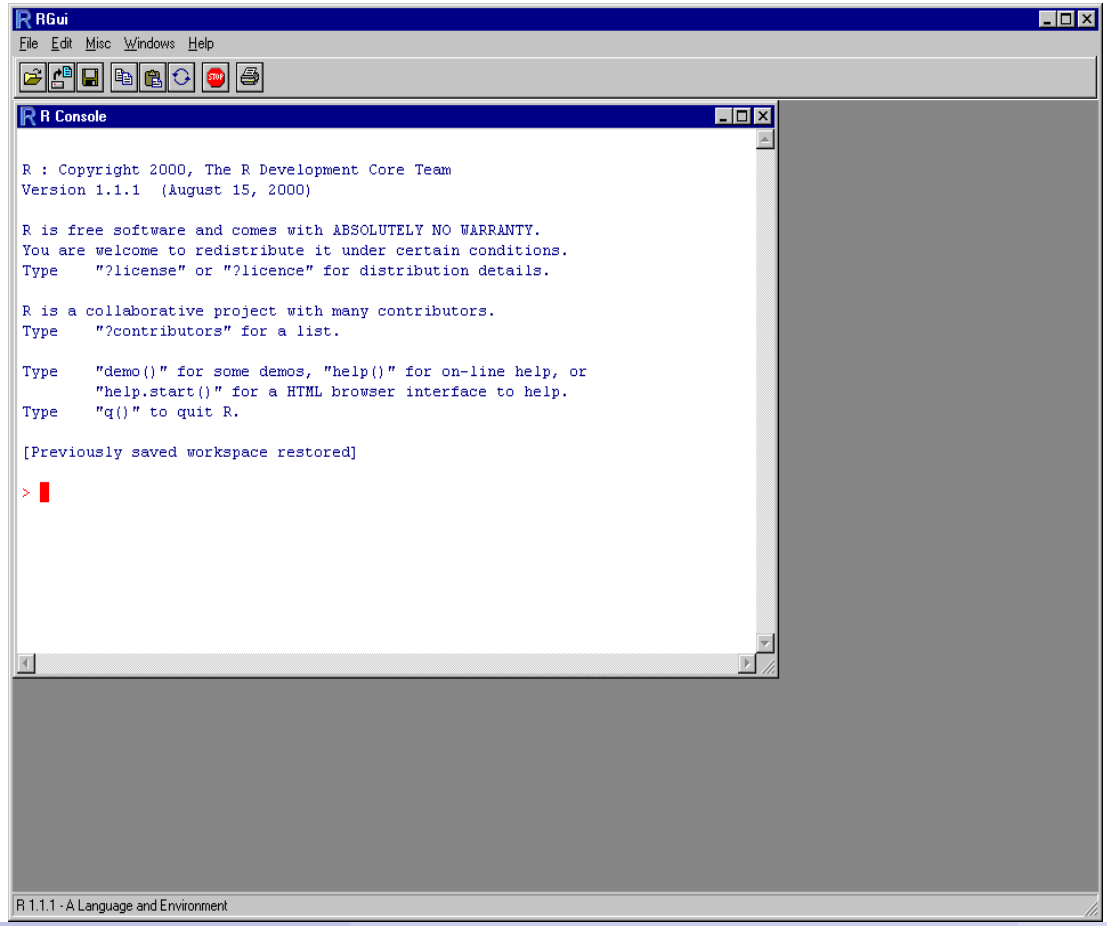
Introduction à R - Novembre 2009

E. Comets

Vue d'ensemble
Le système
Un exemple
A l'aide !
Premiers pas
Les vecteurs
Tests
Les matrices
Affichage
La fin...pour aujourd'hui

Comment utiliser R (1)

- Lancer R
  - double-clic sur l'icône
- Taper des commandes
  - fenêtre terminal
- Interrompre des calculs
  - touche ESC



Introduction à R - Novembre 2009
E. Comets

Vue d'ensemble
Le système
Un exemple
A l'aide !
Premiers pas
Les vecteurs
Tests
Les matrices
Affichage
La fin...pour aujourd'hui

Comment utiliser R (2)

- Comme une calculatrice :

taper exactement l'instruction ci-dessous dans la fenêtre terminal

```
10+2
```

Ce qu'on voit dans la fenêtre terminal :

```
> 10*2
[1] 20
```

- Attention, ne pas taper le signe supérieur ">", qui est en fait l'indicateur (ou *prompt*) disant que le système est prêt à écouter les commandes.
- R renvoie le résultat (ici, 20) précédé de [1]

Dans la suite des exemples, nous mettrons toujours le *prompt* pour différencier d'un résultat rendu par R mais il ne faudra pas le taper.

Comment utiliser R (3)

- Comme une calculatrice (suite) :

R connaît de nombreux noms de fonctions mathématiques

```
> log(2)+exp(0.5)*(4**2)+sqrt(4)+cos(pi)
[1] 28.07269
```

log, exp, sqrt, cos sont des fonctions R (voir plus loin).

- Comme un langage :

On peut affecter le calcul précédent à un objet :

```
> x<-log(2)+exp(0.5)*(4**2)+sqrt(4)+cos(pi)
> x
[1] 28.07269
```

x contient maintenant le résultat du calcul et peut être utilisé comme tel :

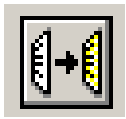
```
> log(x)
[1] 3.334797
```

Fichier de commande

Il est conseillé de créer un fichier de commandes (fichier "script")

- Fichier $\longrightarrow$ Nouveau script
- enregistrer le script dans votre répertoire de travail

Pour soumettre une ligne de commande

- placer le curseur sur cette ligne ou la sélectionner
- cliquer sur 

A partir de maintenant, habituez-vous à taper toutes vos commandes dans un fichier script, et sauvez-le à la fin de la séance.

Le script sert de fichier de commande dans le langage R et peut être utilisé comme un "programme", composé de commandes qui s'exécutent l'une après l'autre et de fonctions (modules) qui peuvent être appelées.

Introduction à R - Novembre 2009

E. Comets

Vue d'ensemble
Le système
Un exemple
A l'aide !

Premiers pas
Les vecteurs
Tests
Les matrices
Affichage

La fin...pour aujourd'hui

Utilisation de Tinn-R

On peut également télécharger le logiciel Tinn-R :
`http://www.sciviews.org/Tinn-R/Tinn-R_1.18.5.6_setup.exe`

- Installer Tinn-R
- Spécifier l'exécutable R associé : R $\longrightarrow$ Rgui $\longrightarrow$ Initiate preferred
- Suivre les instructions de la FAQ:
  `http://www.sciviews.org/Tinn-R/Tinn-R_FAQ.html`

L'avantage principal de Tinn-R est la coloration syntaxique, qui permet de mieux s'y retrouver.

Un exemple

Pour introduire R, nous allons voir un panorama rapide de ce que nous pouvons faire avec le logiciel.

Le problème : On a observé les vitesses de décomposition d'un médicament dans plusieurs milieux (pH, température). On vous donne un fichier contenant ces données. Dans un éditeur de texte, ces données ressemblent à :

```
Heure   Obs     pH      Temp
1       12.6808271380506        3       10
2       21.1737611264806        3       10
3       40.440773331342 3       10
6       40.6003130905758        3       10
12      65.316532922766 3       10
24      67.6825382118089        3       10
....
```

Elles sont sauvées dans mon répertoire de travail sous le nom decomposition.dat.

Lecture d'un fichier

Notion : lecture, entrée/sortie

Ces données sont structurées en une table, on va les lire avec une commande appelée read.table et les mettre dans un objet qu'on appelle mydat.

```
mydat<-read.table("decomposition.dat",header=T)
```

Note : pour utiliser read.table, on a juste besoin de lui donner le nom du fichier (avec son chemin si besoin)

Options : ici j'ai aussi donné le paramètre optionnel header=T, qui lui dit que la première ligne donne le titre des colonnes; souvent de nombreuses options existent et on ne spécifie que ceux qui changent de la valeur par défaut

Introduction à R - Novembre 2009

E. Comets

Vue d'ensemble
Le système
Un exemple
A l'aide !
Premiers pas
Les vecteurs
Tests
Les matrices
Affichage
La fin...pour aujourd'hui

A quoi ressemblent mes données?

Notion : affichage

Pour visualiser les données, je peux taper le nom de l'objet dans lequel je les ai mises :

```
> mydat
...
```

mais il y en a beaucoup et je ne vois plus le début. Je peux utiliser la commande head pour voir juste les premières lignes :

```
> head(mydat)
     Heure      Obs pH Temp
[1,]     1 12.68083  3   10
[2,]     2 21.17376  3   10
[3,]     3 40.44077  3   10
[4,]     6 40.60031  3   10
[5,]    12 65.31653  3   10
[6,]    24 67.68254  3   10
```

Tiens d'ailleurs, combien j'en ai ?

Notion : fonctions de base

Je compte le nombre de lignes et de colonnes avec dim :

```
> dim(mydat)
[1] 150   4
```

Combien ai-je de temps différents ?

```
> unique(mydat$Heure)
[1]  1  2  3  6 12 24
> length(unique(mydat$Heure))
[1] 6
```

Combien ai-je de "sujets" (ici, combien de fois je trouve le temps 1 par exemple) ?

```
> length(mydat$Heure[mydat$Heure==1])
[1] 25
```

Il y en a beaucoup, résumons !

Notion : statistiques descriptives

J'aimerais bien avoir une idée, par exemple, de mon échelle de temps et de l'étendue de mes observations. La commande summary, qui s'applique à de nombreux objets de R peut servir à ça :

```
> summary(mydat)
     Heure            Obs                pH          Temp
 Min.   : 1.0   Min.   :  8.516   Min.   : 3   Min.   :10
 1st Qu.: 2.0   1st Qu.: 28.748   1st Qu.: 5   1st Qu.:20
 Median : 4.5   Median : 44.515   Median : 7   Median :30
 Mean   : 8.0   Mean   : 46.122   Mean   : 7   Mean   :30
 3rd Qu.:12.0   3rd Qu.: 63.181   3rd Qu.: 9   3rd Qu.:40
 Max.   :24.0   Max.   :117.419   Max.   :11   Max.   :50
```

La commande str donne un autre type de résumé incluant le type de chaque variable.

Introduction à R - Novembre 2009

E. Comets

Vue d'ensemble
Le système
Un exemple
A l'aide !

Premiers pas
Les vecteurs
Tests
Les matrices
Affichage

La fin...pour aujourd'hui

Un petit histogramme (1)

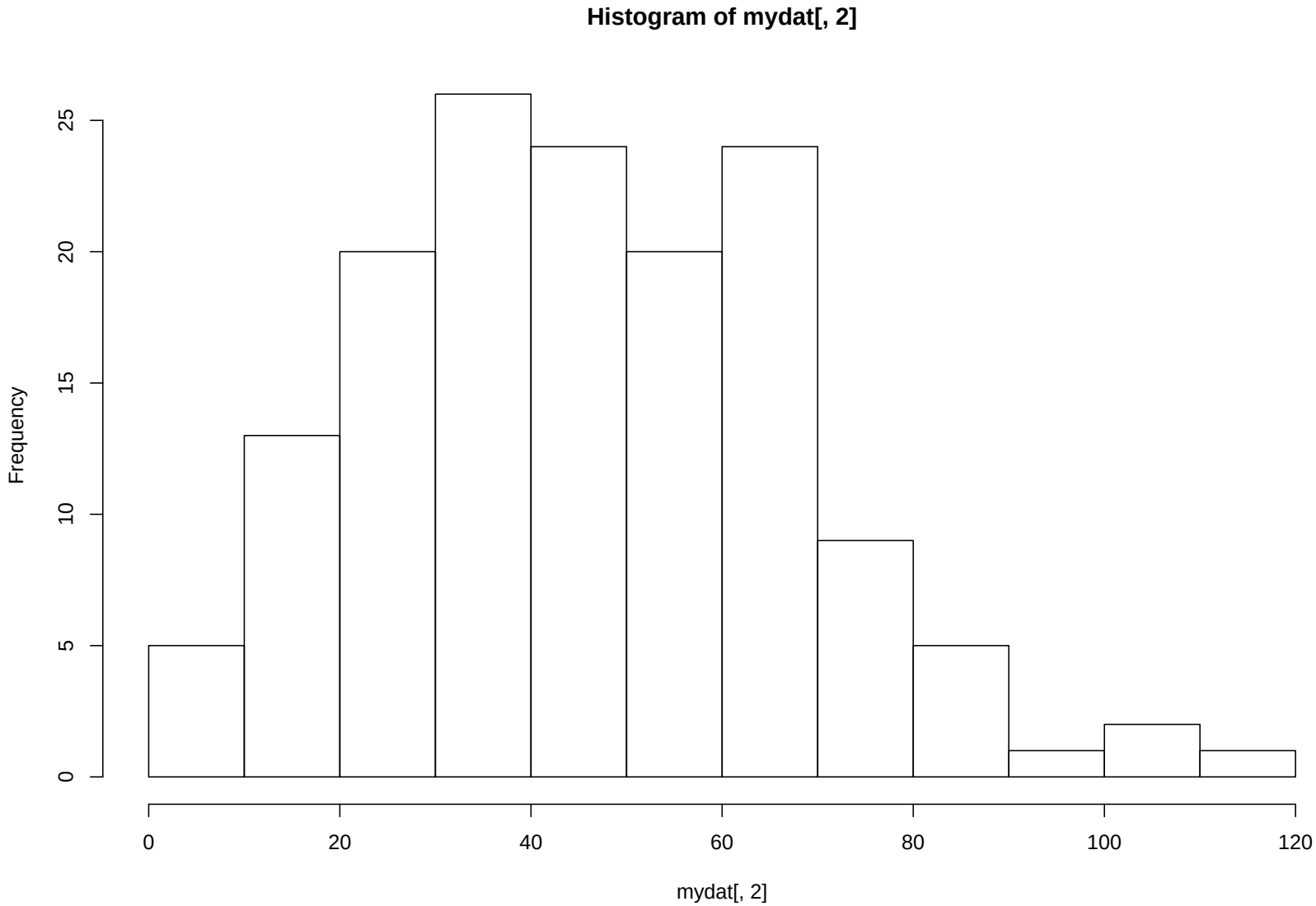
Notion : graphe

Oui, bon, mais tout ça, ça n'est pas très parlant... Je peux aussi représenter mes observations comme un histogramme :

```
> hist(mydat[,2])
```

Note : nous reviendrons abondamment sur la notation mydat[,2], qui fait référence à la 2e colonne de la table mydat. On peut également ici utiliser `mydat$Obs` comme dans un des transparents précédents.

Un petit histogramme (2)



Introduction à R - Novembre 2009

E. Comets

Vue d'ensemble
Le système
Un exemple
A l'aide !

Premiers pas
Les vecteurs
Tests
Les matrices
Affichage

La fin...pour aujourd'hui

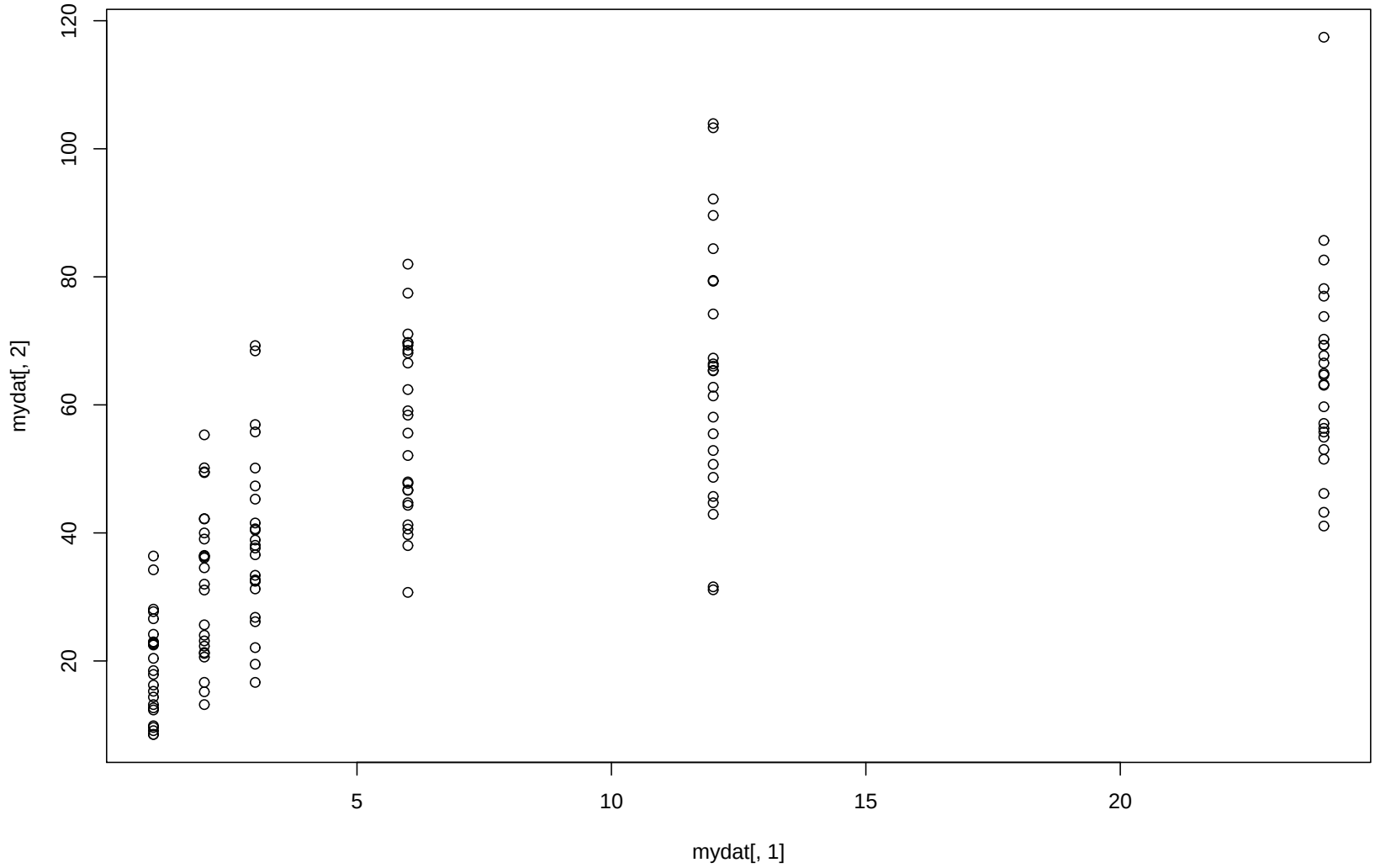
Hmm, comment traiter ces données?

Notion : graphe

Bon maintenant j'en fais quoi, de ces données... si je faisais un graphe des observations en fonction du temps?

```
> plot(mydat[,1],mydat[,2])
```

Graphe



Introduction à R - Novembre 2009
E. Comets

Vue d'ensemble
Le système
Un exemple
A l'aide !
Premiers pas
Les vecteurs
Tests
Les matrices
Affichage
La fin...pour aujourd'hui

Tiens, on dirait une droite...

Notion : régression, analyse

Pour décrire la relation entre le temps et les observations, je vais commencer par essayer une bête relation linéaire :

```
> y<-lm(mydat[,2]~mydat[,1])
> summary(y)
Call:
lm(formula = mydat[, 2] ~ mydat[, 1])
Residuals:
    Min      1Q  Median      3Q     Max
-32.769 -11.882  -2.196   9.834  50.874
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  32.2586     2.0125  16.029   <2e-16 ***
mydat[, 1]    1.7329     0.1776   9.755   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
Residual standard error: 17.45 on 148 degrees of freedom
Multiple R-squared: 0.3913,     Adjusted R-squared: 0.3872
F-statistic: 95.15 on 1 and 148 DF,  p-value: < 2.2e-16
```

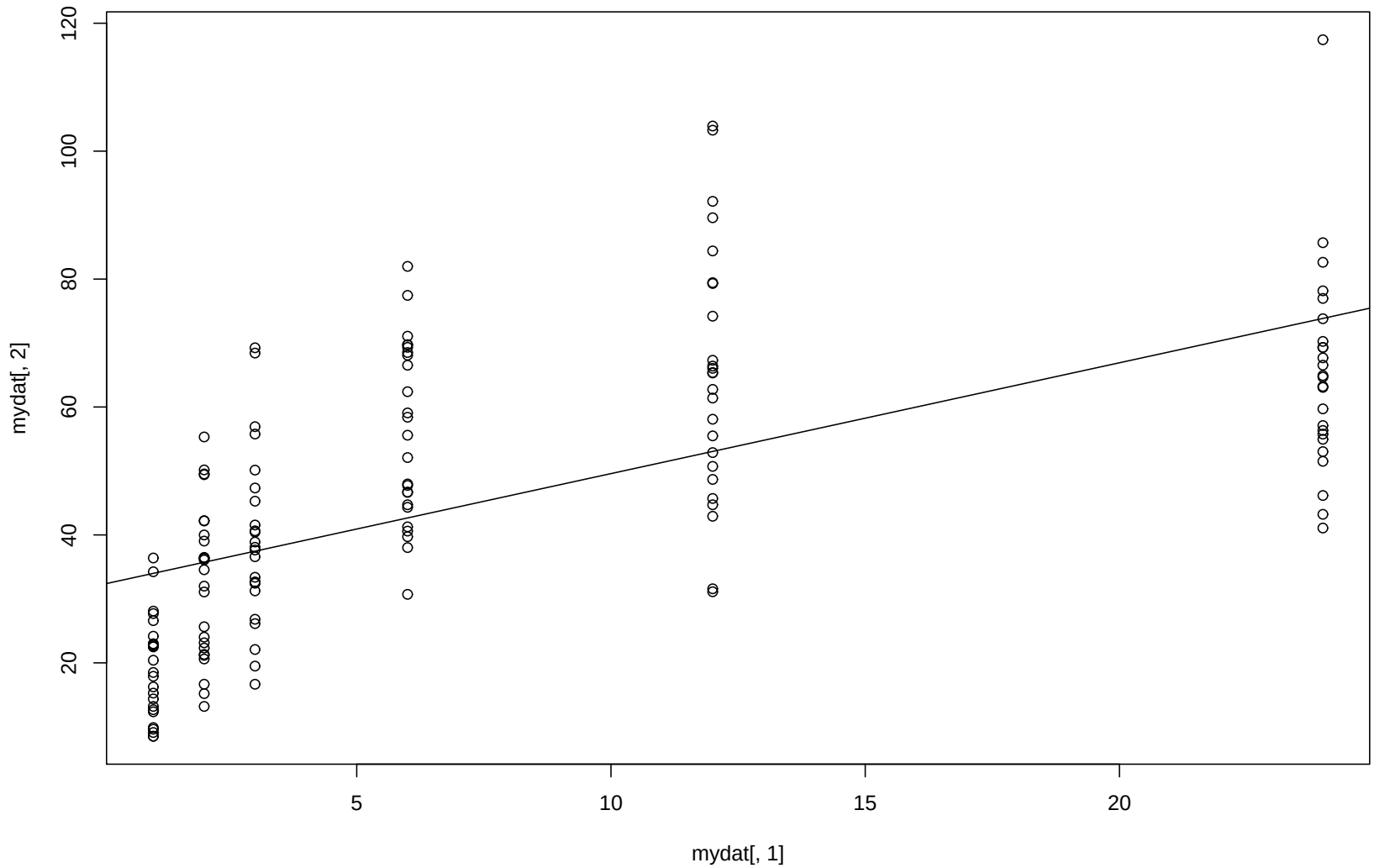
Superposons les prédictions et les observations...

Notion : diagnostic

Superposons sur notre graphe la droite de régression histoire de voir quelle tête a mon modèle... J'utilise pour ça la fonction abline à laquelle je donne comme argument le résultat de mon analyse :

```
> abline(y)
```

Résultat



Essayons d'améliorer le modèle...

Notion : comparaison

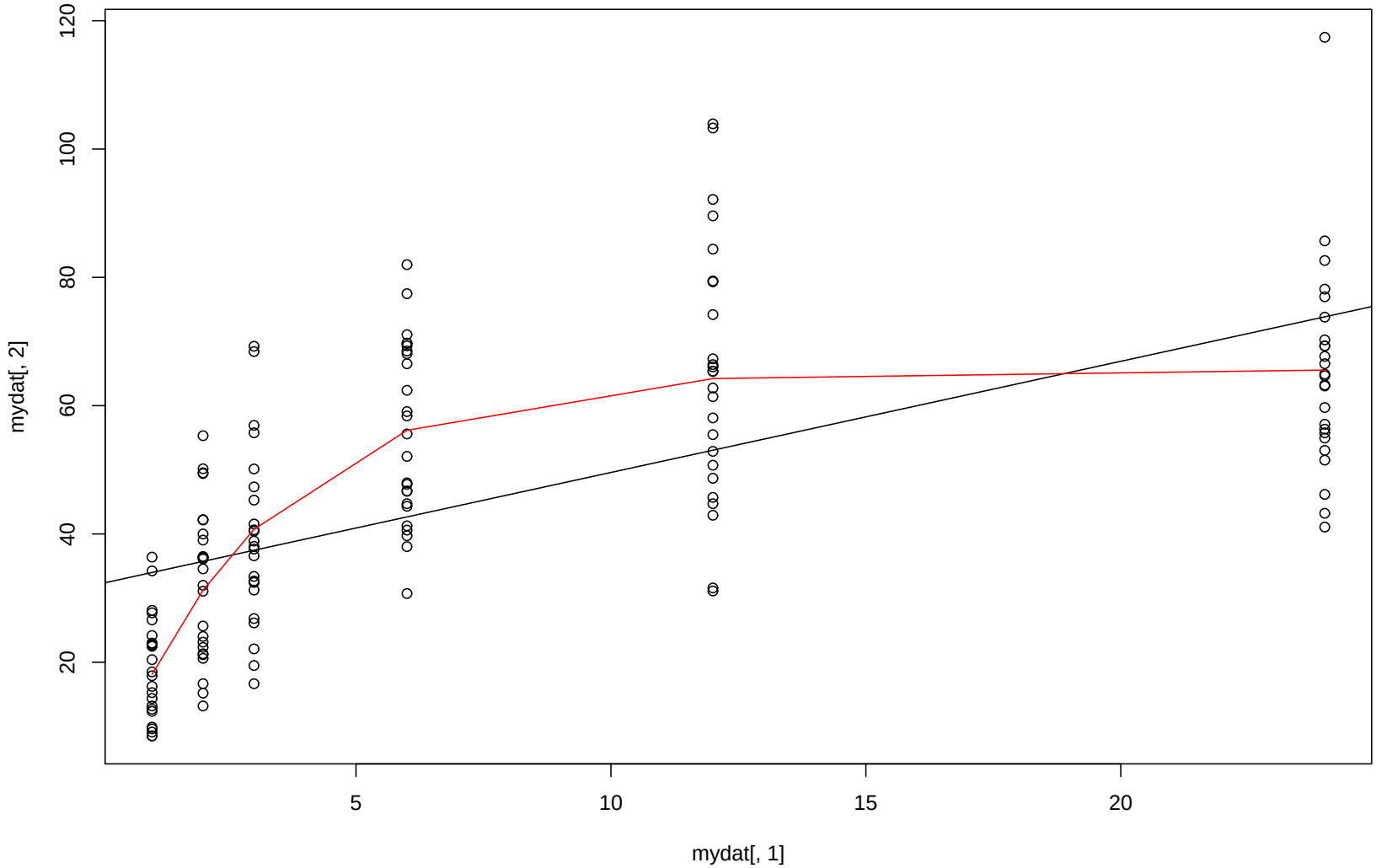
Bon d'accord, une droite ça n'a pas l'air terrible. J'ai donc envie d'essayer un modèle un peu différent, par exemple un modèle exponentiel :

$$y_i = \alpha\left(1 - e^{-\lambda t_i}\right) + \varepsilon_i$$

On utilisera par exemple la fonction nls pour cela :

```
> y2<-nls(Obs~alpha*(1-exp(-lambda*Heure)),mydat,
  start=c(alpha=100,lambda=0.02))
> summary(y2)
Formula: Obs ~ alpha * (1 - exp(-lambda * Heure))
Parameters:
       Estimate Std. Error t value Pr(>|t|)
alpha  65.57698    2.08804   31.41   <2e-16 ***
lambda  0.32352    0.03038   10.65   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
Residual standard error: 14.15 on 148 degrees of freedom
Number of iterations to convergence: 12
Achieved convergence tolerance: 1.756e-06
```

C'est déjà mieux...



C'est bien joli, mais...

...que se passerait-il si j'avais pris des mesures jusqu'à 48 heures?

Notion : prédiction

J'utilise la fonction de prédiction associée à la fonction nls et qui porte le nom générique de predict. J'ai juste besoin (on le reverra) de créer un nouveau tableau de données :

```
> newdata<-data.frame(Heure=c(48),Obs=c(0))
> newdata
> predict(y2,newdata)
[1] 65.57696
```

On peut même la comparer avec celle prédite par le modèle linéaire :

```
> y<-lm(Obs~Heure,mydat)
> predict(y,newdata)
[1] 115.4367
```

Et...

...si je m'intéressais plutôt à des zones de pH neutre ?

Notion : sélection

On peut facilement sélectionner par exemple les données correspondant à pH7 et les mettre dans un nouveau jeu de données

```
> subdat<-mydat[mydat$pH==7,]
> subdat
...
```

En fait, le labo vient de me signaler qu'il avait fait une erreur de dosage : au temps 2 h de la manip à pH=3 et à température de 20 degrés, ce n'est pas 39.03988 qu'il faut lire mais 32.0445.

Notion : remplacement

Plusieurs façons de procéder :

```
> mydat[8,2]<-32.0445
> mydat[mydat$Heure==2 & mydat$pH==3 & mydat$Temp==20,2]<-32.0445
```

Et...

...un collègue américain arrive au labo et il a du mal à faire la conversion en degrés farenheit.

Notion : ajout de colonne (modification)

Je vais lui rajouter une colonne Temp.Far :

```
> amerdat<-transform(mydat,Temp.Far=((9*Temp)/5)+32)
> head(amerdat)
  Heure      Obs pH Temp Temp.Far
1     1 12.68083  3   10       50
2     2 21.17376  3   10       50
3     3 40.44077  3   10       50
4     6 40.60031  3   10       50
5    12 65.31653  3   10       50
6    24 67.68254  3   10       50
```

Pour aller plus loin...

Nous pouvons manipuler le jeu de données, en extraire des bouts, faire des jeux de données bootstrap en le rééchantillonnant, voir ce qui se passerait si on avait des données manquantes.
Nous pouvons également sur cet exemple faire des diagnostics plus poussés (résidus, valeurs aberrantes, ...) et comparer les modèles avec un test.
Nous pouvons ajuster un modèle tenant compte des différentes conditions expérimentales afin d'évaluer l'influence du pH ou de la température sur les paramètres du modèle.
Nous pouvons tester si le modèle d'erreur est adéquat.

Note : nous reviendrons sur les détails des fonctions utilisées dans ce cours et sur la façon de spécifier les variables.

Introduction à R - Novembre 2009

E. Comets

Vue d'ensemble
Le système
Un exemple
A l'aide !
Premiers pas
Les vecteurs
Tests
Les matrices
Affichage
La fin...pour aujourd'hui

Que peut-on faire en R?

- Gestion de bases de données
  - lecture
  - croisement de bases :faire correspondre plusieurs bases avec un identifiant commun
  - modifications : changement d'unité, calculs
- Graphes
  - visualiser les données
  - faire des graphes de résultats pour un papier
- Tableaux
  - exportation des résultats sous forme d'un tableau pour intégration dans un papier ou un mémoire
- Des programmes
- Des dizaines d'autres choses, comme :
  - faire ses comptes
  - gérer sa collection de DVD
  - créer des interfaces graphiques
  - interfacer une wii-mote (véridique)

Introduction à R - Novembre 2009

E. Comets

Vue d'ensemble
Le système
Un exemple
A l'aide !
Premiers pas
Les vecteurs
Tests
Les matrices
Affichage
La fin...pour aujourd'hui

Quelques conseils

- S'habituer à taper ses commandes dans un fichier texte
  - permet de corriger facilement ses erreurs sans tout retaper
  - permet de garder la trace de son travail d'une session sur l'autre
  - premiers pas vers la programmation
- Voir aussi manuel PDF
  - index avec hyperliens pour les fonctions décrites
- Bouée de secours absolument indispensable
  - habituez-vous à consulter l'aide en ligne!

Utilisation de l'aide

Pour éditer la fiche d'aide correspondant à une fonction, par exemple `lm()` (régression linéaire) :

- `help(lm)`
- ou `?lm`

On obtient une fiche comprenant les sections :

Description : une brève description de l'usage de la fonction

Usage : les arguments de la fonction et leurs valeurs par défaut

Arguments : les paramètres devant être passés à la fonction

Details : des remarques particulières sur l'usage de la fonction

Value : le type d'objet retourné par la fonction

See Also : autres rubriques d'aide proches ou similaires

Examples : des exemples d'application

Recherche

Quand on ne sait pas trop ce qu'on cherche... Par exemple, j'aimerais trouver une fonction qui fasse des permutations :

- `help.search("permutation")`
- ou `??permutation`

```
Help files with alias or concept or title matching 'permutation'
using fuzzy matching:
base::order                Ordering Permutation
base::sample               Random Samples and Permutations
combinat::permn            Generates all permutations of the elements
e1071::permutations        All Permutations of Integers 1:n
gbm::relative.influence    Methods for estimating relative influence
gdata::resample            Consistent Random Samples and Permutations
gtools::combinations       Enumerate the Combinations or Permutations
                           the Elements of a Vector
Matrix::pMatrix-class      Permutation matrices
zoo::ORDER                 Ordering Permutation
```

Note: le premier élément est le nom de la librairie (voir plus loin).

Introduction à R - Novembre 2009

E. Comets

Vue d'ensemble
Le système
Un exemple
A l'aide !
Premiers pas
Les vecteurs
Tests
Les matrices
Affichage
La fin...pour aujourd'hui

Exercice

- Premiers pas
  - lancer R
  - ouvrir un nouveau script
  - donner le résultat du calcul de 10!
  - calculer la valeur du logarithme népérien de 2
  - calculer la valeur du logarithme de 2 en base 10 ($\log_{10}(x) = \log_e(x)/\log_e(10)$)
- Aide
  - consulter la fiche d'aide de R sur la fonction log
  - calculer la valeur du logarithme de 2 en base 10 d'une autre manière en utilisant cette page d'aide

Variables en R

Une variable est caractérisée par son type et sa structure.

- Types
  - entier : ...,-2, -1, 0, 1, 2, ...
  - réel (numérique)
  - caractère : chaîne de caractères
  - catégorie (facteur) : variable numérique ou qualitative non ordonnée
  - booléen (valeurs possibles TRUE/FALSE) : variable à 2 modalités vrai/faux
- Structures
  - vecteur : une série d'éléments de même type
  - matrice/array : un tableau à plus d'une dimension
  - data frame : une table structurée
  - liste : structure la plus flexible composée d'éléments indépendants de nature et de taille potentiellement différentes

L'unité de base de R est le vecteur : un scalaire (entier ou réel) est considéré comme un vecteur de taille 1, et une matrice (tableau) comme une collection de vecteurs (les colonnes).

Introduction à R - Novembre 2009
E. Comets
Vue d'ensemble
Le système
Un exemple
A l'aide !
Premiers pas
Les vecteurs
Tests
Les matrices
Affichage
La fin...pour aujourd'hui

Premières commandes : création d'un objet

Création de l'objet a, un scalaire de valeur 1

```
> a=1
```

ou

```
> a<-1
```

Lorsque l'on tape

```
> a
```

on obtient

```
> [1] 1
```

On peut aussi créer 2 objets identiques d'un coup

```
> a <-b <- 2
> a
> [1] 2
> b
> [1] 2
```

Création d'un vecteur

On utilise les fonctions `c`, `seq` et/ou `rep`, et l'opérateur `":"`

```
> a<-c(1,3,5)
> a
[1] 1 3 5
> a<-1:6
> a
[1] 1 2 3 4 5 6
> b<-seq(1,5,by=0.5)
> b
[1] 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0
> d<-rep(3,5)
> d
[1] 3 3 3 3 3
```

Introduction à R - Novembre 2009
E. Comets
Vue d'ensemble
Le système
Un exemple
A l'aide !
Premiers pas
Les vecteurs
Tests
Les matrices
Affichage
La fin...pour aujourd'hui

Type d'un vecteur (1)

En gros, le type d'un objet est déterminé par la taille mémoire nécessaire pour le stocker. Pour un ordinateur, un entier prend moins de place qu'un nombre réel.
On utilise la fonction `typeof` pour connaître le type d'un objet :

```
> typeof(a)
[1] "integer"
> typeof(d)
[1] "double"
```

Le type du vecteur s'ajuste automatiquement aux éléments, en prenant le type permettant de stocker tous les éléments.

```
> cha<-c(1:4,1.5,"toto")
> typeof(cha)
[1] "character"
```

Type d'un vecteur (2)

Attention, en ajoutant une chaîne de caractère dans notre vecteur `cha`, nous avons transformé tous les éléments en des caractères. Du coup ces éléments ne sont plus des nombres :

```
> cha
[1] "1"    "2"    "3"    "4"    "1.5"  "toto"
> log(cha)
Erreur dans log(cha) : Argument non numérique pour une fonction
mathématique
```

Le calcul du `log` donne donc une erreur. On peut utiliser la fonction `as.double` pour transformer ces chaînes de caractères en nombre ("to cast" en langage informatique) :

```
> log(as.double(cha))
[1] 0.0000000 0.6931472 1.0986123 1.3862944 0.4054651        NA
Warning message:
NAs introduits lors de la conversion automatique
```

Exemples

1) vecteur numérique :

```
> x <- seq(15,25, by=5)
> x
[1] 15 20 25
```

2) vecteur de chaînes de caractères :

```
> prenom <- c("John","Paul","George")
> prenom
[1] "John"   "Paul"   "George"
```

3) vecteur logique :

```
> test <- (x>=20)
> test
[1] FALSE  TRUE  TRUE
> sum(test)
[1] 2
```

4) vecteur de facteurs :

```
> fact<-as.factor(rep(4:5, each=2))
> fact
[1] 4 4 5 5
Levels: 4 5
```

Introduction à R - Novembre 2009
E. Comets
Vue d'ensemble
Le système
Un exemple
A l'aide !
Premiers pas
Les vecteurs
Tests
Les matrices
Affichage
La fin...pour aujourd'hui

Création d'un vecteur - exercice

Créer le vecteur :

```
10   3   4   5   6  10 100 100  10  20  30  40
```

- à la main
- en utilisant `seq` et `rep`

Une autre commande un peu plus compliquée vector :

```
> a<-vector(length=10,mode="numeric")
> a
 [1] 0 0 0 0 0 0 0 0 0 0
```

(vector nous rendra surtout service pour créer des listes)

Introduction à R - Novembre 2009
E. Comets
Vue d'ensemble
Le système
Un exemple
A l'aide !
Premiers pas
Les vecteurs
Tests
Les matrices
Affichage
La fin...pour aujourd'hui

Manipulation des vecteurs

Sélectionner un élément d'un vecteur

```
> x<-10:16
> x
[1] 10 11 12 13 14 15 16
> x[2]
[1] 11
```

Sélectionner le 2e et 4e élément de x :

```
> x[c(2,4)]
[1] 11 13
```

Sélection négative : tous les éléments sauf le 4e

```
> x[-4]
[1] 10 11 12 14 15
```

Introduction à R - Novembre 2009
E. Comets
Vue d'ensemble
Le système
Un exemple
A l'aide !
Premiers pas
Les vecteurs
Tests
Les matrices
Affichage
La fin...pour aujourd'hui

Exercice

A chaque fois, afficher le résultat demandé

- Créer un vecteur x contenant les éléments 1, 4 et 5
- Créer un vecteur y contenant les chiffres impairs de 1 à 9
  - afficher le 2ème élément de y
  - afficher tous les éléments de y sauf le 2ème
- Créer un vecteur xy contenant le premier, quatrième et cinquième élément de y
  - utilisez le vecteur x pour cela
- Afficher les éléments 2 à 4 de y

Opérations sur les vecteurs

Les opérations arithmétiques :

```
+, -, *, /, ^ (ou **)
```

Les opérations logiques :

```
==, !=, <, <=, >, >=
```

Toutes ces opérations se font élément par élément :

```
> x <- 1:3
> x**2
[1] 1 4 9
> x/c(2,4,6)
[1] 0.5 0.5 0.5
```

R est un langage vectoriel $\Rightarrow$ l'objet de base est le vecteur.

Introduction à R - Novembre 2009
E. Comets
Vue d'ensemble
Le système
Un exemple
A l'aide !
Premiers pas
Les vecteurs
Tests
Les matrices
Affichage
La fin...pour aujourd'hui

Opérations sur deux vecteurs

Exemple : multiplication:

```
> x<-1:5
> x*x
[1]  1  4  9 16 25
```

Chaque élément de x est multiplié par lui-même $\Rightarrow$ on a calculé le carré de x.

Vecteurs de longueur inégale : le plus petit vecteur est "recyclé" pour s'ajuster au plus grand

```
> y<-1:3
> x*y
[1]  1  4  9  4 10
Message d'avis : la longueur de l'objet le plus long n'est pas
un multiple de la longueur de l'objet le plus court in: x * y
```

Un message d'erreur apparaît ici.

Introduction à R - Novembre 2009
E. Comets
Vue d'ensemble
Le système
Un exemple
A l'aide !
Premiers pas
Les vecteurs
Tests
Les matrices
Affichage
La fin...pour aujourd'hui

Exercice

A chaque fois, afficher le résultat demandé

- Reprendre le vecteur y précédent
  - afficher les chiffres pairs de 2 à 10 à partir de y
  - afficher les chiffres pairs de 2 à 8 (trouver 2 écritures possibles en R pour obtenir ce résultat)
- Créer un vecteur x contenant les éléments 2, 3, et 4
  - multiplier y par x
  - voir le message d'avertissement; qu'a fait R et à quoi correspondent les chiffres qu'il nous donne?

Les fonctions utiles (1)

1) ordre : `sort, rev, order, rank`

```
> sort(1:10,decreasing=T)
[1] 10  9  8  7  6  5  4  3  2  1
```

2) aggrégation : `cbind, rbind`

```
> a <- 1:2
> b <- 3:4
> rbind(a,b)
  [,1] [,2]
a    1    2
b    3    4
> cbind(a,b)
     a b
[1,] 1 3
[2,] 2 4
```

D'ici peu nous allons utiliser ces fonctions pour créer des matrices.

Les fonctions utiles (2)

3) propriétés : `length, names, mode`

```
> length(a)
[1] 2
> mode(a)
[1] "numeric"
```

4) fonctions mathématiques : `exp, sqrt, cos, sum, ...`

```
> log(x)
[1] 2.708050 2.995732 3.218876
> sqrt(x)
[1] 3.872983 4.472136 5.000000
> sum(x)
[1] 60
```

5) fonctions de manipulation : `unique, duplicated`

```
> unique(c(1,1,2,2))
[1] 1 2
```

Introduction à R - Novembre 2009
E. Comets
Vue d'ensemble
Le système
Un exemple
A l'aide !
Premiers pas
Les vecteurs
Tests
Les matrices
Affichage
La fin...pour aujourd'hui

Quelques fonctions mathématiques

trunc	partie entière
round	arrondi à un nombre donné de décimales
signif	arrondi à un nombre donné de chiffres significatifs
floor	entier n tel que $n \leq x$
ceiling	entier n tel que $n \geq x$
abs	valeur absolue
sum	somme de toutes les composantes
prod	produit de toutes les composantes
cumprod,cumsum	produit/somme cumulée
diff	différences successives
%%	reste de la division (modulo)
sqrt	racine carrée
min, max	valeur minimale et maximale

Introduction à R - Novembre 2009
E. Comets

Vue d'ensemble
Le système
Un exemple
A l'aide !
Premiers pas
Les vecteurs
Tests
Les matrices
Affichage
La fin...pour aujourd'hui

Expressions logiques

Tests d'égalité : == (attention, 2 "=")

Tests de non-égalité : `!=`

Relations d'ordre : `<,<=,>=,>`

Test simultané de deux conditions `& (et), | (ou)`

Test séquentiel de deux conditions `&& (et), || (ou)`
(la deuxième expression n'est évaluée que si la première est vraie)

Echanger valeurs TRUE et FALSE : !

Introduction à R - Novembre 2009
E. Comets
Vue d'ensemble
Le système
Un exemple
A l'aide !
Premiers pas
Les vecteurs
Tests
Les matrices
Affichage
La fin...pour aujourd'hui

Test sur les composantes

Les opérations logiques `==, !=, <, <=, >, >=` appliquées à un vecteur renvoient un vecteur de booléens (valeurs vrai/faux) :

```
> x <- 1:10
> x>5
 [1] FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE
```

Ce booléen peut être utilisé pour sélectionner les indices de x correspondant à un test vrai (valeurs TRUE du vecteur de booléen) :

```
> x[x>5]
[1]  6  7  8  9 10
```

Introduction à R - Novembre 2009
E. Comets

Vue d'ensemble
Le système
Un exemple
A l'aide !
Premiers pas
Les vecteurs
Tests
Les matrices
Affichage
La fin...pour aujourd'hui

Les données manquantes (1)

Elles sont codées en NA (aussi NaN pour les données numériques) :

```
> y <- c(1:4,NA,rep(0.8,2),NA)
> y
[1] 1.0 2.0 3.0 4.0  NA 0.8 0.8  NA
```

Attention, un test impliquant une donnée NA donnera un résultat NA :

```
> y <- c(1:4,NA,rep(0.8,2),NA)
> y[y>0]
[1] 1.0 2.0 3.0 4.0  NA 0.8 0.8  NA
```

Les données manquantes (2)

Pour "tester" la présence de données manquantes :

```
>is.na(y)
[1] FALSE FALSE FALSE FALSE  TRUE FALSE FALSE  TRUE
> y[is.na(y)==F]
[1] 1.0 2.0 3.0 4.0 0.8 0.8
> y[!is.na(y)]      #même chose
[1] 1.0 2.0 3.0 4.0 0.8 0.8
```

Donc pour afficher uniquement les éléments positifs et non manquants de y, il faut faire un double test ("ET") :

```
> y[!is.na(y) & y>0]
[1] 1.0 2.0 3.0 4.0 0.8 0.8
```

Attention, certaines fonctions retournent NA si on les applique un vecteur contenant des NA :

```
> mean(y)
[1] NA
> mean(y,na.rm=T)
[1] 1.933333
```

Introduction à R - Novembre 2009
E. Comets

Vue d'ensemble
Le système
Un exemple
A l'aide !

Premiers pas
Les vecteurs
Tests
Les matrices
Affichage

La fin...pour aujourd'hui

Exercice : manipulation de vecteurs

1. Créer un vecteur vec de taille 10 avec la commande `rnorm(10)`).
2. Calculer le nombre d'éléments positifs de vec.
3. Calculer yvec dont les éléments sont composés du logarithme des éléments de vec.
4. Créer le vecteur vec2 avec seulement les éléments non manquants de yvec.
5. Afficher le nombre d'éléments de vec2.

Introduction à R - Novembre 2009
E. Comets
Vue d'ensemble
Le système
Un exemple
A l'aide !
Premiers pas
Les vecteurs
Tests
Les matrices
Affichage
La fin...pour aujourd'hui

Création d'une matrice (1)

En partant d'un vecteur, avec remplissage par colonne (par défaut) :

```
> x<-matrix(1:9,ncol=3)
> x
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
```

ou avec remplissage par ligne :

```
> x<-matrix(1:9,ncol=3,byrow=T)
> x
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
```

Création d'une matrice (2)

A partir d'un vecteur `vec`, `dim(vec)<-c(n,m)` crée une matrice à `n` lignes et `m` colonnes :

```
> x<-1:12
> dim(x)<-c(3,4)
> x
     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12
```

Ici encore par défaut le vecteur `x` vient se ranger en colonnes de gauche à droite.

Création d'une matrice (3)

A partir de 2 ou plusieurs vecteurs, soit ligne par ligne (`rbind`), soit colonne par colonne (`cbind`).

```
> x<-1:4
> y<-x**2;z<-x**3
> mat<-rbind(x,y,z)
> mat
  [,1] [,2] [,3] [,4]
x    1    2    3    4
y    1    4    9   16
z    1    8   27   64
> mat<-cbind(x,y,z)
> mat
     x  y  z
[1,] 1  1  1
[2,] 2  4  8
[3,] 3  9 27
[4,] 4 16 64
```

Bien sûr il faut que les vecteurs aient la même taille (sinon échec avec message d'erreur).

Introduction à R - Novembre 2009
E. Comets
Vue d'ensemble
Le système
Un exemple
A l'aide !
Premiers pas
Les vecteurs
Tests
Les matrices
Affichage
La fin...pour aujourd'hui

Création d'une matrice (4)

Matrice diagonale avec `diag(x)`

```
> y<-1:3
> diag(y)
     [,1] [,2] [,3]
[1,]    1    0    0
[2,]    0    2    0
[3,]    0    0    3
```

Eléments d'une matrice (1)

Pour récupérer la deuxième colonne de la matrice x :

```
> x<-matrix(1:9,ncol=3,byrow=T)
> x[,2]
[1] 2 5 8
```

Pour la troisième ligne :

```
> x[3,]
[1] 7 8 9
```

Pour le deuxième élément de la troisième ligne :

```
> x[3,2]
[1] 8
```

Pour avoir toute la deuxième ligne sauf l'élément de la deuxième colonne :

```
> x[2,-2]
[1] 4 6
```

Eléments d'une matrice (2)

Comme pour les vecteurs, on peut sélectionner plusieurs lignes à la fois :

```
> x[1:2,]
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
```

On peut sélectionner en même temps sur les lignes et les colonnes, on obtient les cellules situées à l'intersection des 2 conditions :

```
> x[1:2,1:2]
     [,1] [,2]
[1,]    1    2
[2,]    4    5
```

A la place de lignes consécutives (ici 1:2), on peut introduire un vecteur d'indices :

```
> x[c(1,3),]
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    7    8    9
```

Introduction à R - Novembre 2009

E. Comets

Vue d'ensemble
Le système
Un exemple
A l'aide !

Premiers pas
Les vecteurs
Tests
Les matrices
Affichage

La fin...pour aujourd'hui

Eléments d'une matrice (3)

Le vecteur d'indices peut être défini comme une condition :

```
>  x[x[,1]>1,]
     [,1] [,2] [,3]
[1,]    4    5    6
[2,]    7    8    9
```

Ce qui se lit : "on choisit parmi les lignes de x celles pour lesquelles, dans la première colonne, la valeur est supérieure strictement à 1" (dans la partie ligne) et "toutes les colonnes" (dans la partie colonne).

Introduction à R - Novembre 2009
E. Comets

Vue d'ensemble
Le système
Un exemple
A l'aide !
Premiers pas
Les vecteurs
Tests
Les matrices
Affichage
La fin...pour aujourd'hui

Exercice

1. Créer une matrice à 3 lignes et 5 colonnes ayant pour éléments 15 chiffres entre 1 et 20 (remplissage indifférent)
2. Extraire la sous-matrice formée par les 2 dernières lignes et les colonnes 2 et 4
3. Extraire les éléments inférieurs à 3 de la première colonne
4. Afficher les lignes de la matrice pour lesquelles la valeur dans la première colonne est inférieure à 3

Introduction à R - Novembre 2009
E. Comets
Vue d'ensemble
Le système
Un exemple
A l'aide !
Premiers pas
Les vecteurs
Tests
Les matrices
Affichage
La fin...pour aujourd'hui

Affichage (1)

Affichage de chaînes de caractères : fonction `cat`

```
> cat(prenom[1],"est l'ami de",prenom[2],"\n")
John est l'ami de Paul
```

Transforme les autres types de données en chaînes :

```
> cat(prenom[1],"a",length(prenom)-1,"amis\n")
John a 2 amis
```

L'unité de base en R étant le vecteur, `cat` transforme les matrices en vecteur pour les afficher :

```
> x<-matrix(1:9,ncol=3)
> x
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
> cat(x,"\n")
1 2 3 4 5 6 7 8 9
```

Introduction à R - Novembre 2009
E. Comets

Vue d'ensemble
Le système
Un exemple
A l'aide !
Premiers pas
Les vecteurs
Tests
Les matrices
Affichage
La fin...pour aujourd'hui

Affichage (2)

Affichage orienté objet, avec la fonction `print` :

```
> print(prenom)
[1] "John"   "Paul"   "George"
```

Cette fonction s'adapte au type d'objet considéré. Pour une matrice :

```
> print(x)
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
```

On verra que ce sera le cas aussi pour des objets plus complexes (résultats de tests, objet résultant de procédures statistiques,...).

Introduction à R - Novembre 2009
E. Comets
Vue d'ensemble
Le système
Un exemple
A l'aide !
Premiers pas
Les vecteurs
Tests
Les matrices
Affichage
La fin...pour aujourd'hui

Affichage (3)

Le résultat de `print` peut être renvoyé vers un autre objet (comme souvent dans R). Essayez :

```
y<-print(x)
```

Pour écrire directement dans une chaîne de caractères, on dispose de la fonction `paste` :

```
> numdat<-1
> nomfich<-paste("dataset",numdat,".dat",sep="")
> nomfich
[1] "dataset1.dat"
```

Là encore, un ou plusieurs éléments peuvent être des vecteurs :

```
> paste("data",1:3,sep=".")
[1] "data.1" "data.2" "data.3"
```

Affichage (4)

La syntaxe de `paste` est la suivante :

```
paste(..., sep = " ", collapse = NULL)
```

Les "..." représentent autant d'arguments que l'on veut, que la fonction va coller en les séparant par le ou les caractères spécifiés dans `sep`. Avec `collapse`, il est possible de demander à ce que le résultat soit concaténé dans une seule chaîne de caractères, séparées à chaque fois dans la chaîne finale par un ou plusieurs caractères :

```
> paste("data",1:3,sep="",collapse="+")
[1] "data1+data2+data3"
> paste("data",1:3,sep="X",collapse="...")
[1] "dataX1...dataX2...dataX3"
```

Introduction à R - Novembre 2009
E. Comets
Vue d'ensemble
Le système
Un exemple
A l'aide !
Premiers pas
Les vecteurs
Tests
Les matrices
Affichage
La fin...pour aujourd'hui

Affichage (5)

Deux fonctions très utiles `head` et `tail` pour afficher respectivement les premières et les dernières lignes d'un tableau/matrice.

```
> x<-matrix(1:100,ncol=5)
> head(x)
     [,1] [,2] [,3] [,4] [,5]
[1,]    1   21   41   61   81
[2,]    2   22   42   62   82
[3,]    3   23   43   63   83
[4,]    4   24   44   64   84
[5,]    5   25   45   65   85
[6,]    6   26   46   66   86
> tail(x)
      [,1] [,2] [,3] [,4] [,5]
[15,]   15   35   55   75   95
[16,]   16   36   56   76   96
[17,]   17   37   57   77   97
[18,]   18   38   58   78   98
[19,]   19   39   59   79   99
[20,]   20   40   60   80  100
```

Introduction à R - Novembre 2009

E. Comets

Vue d'ensemble

Le système

Un exemple

A l'aide !

Premiers pas

Les vecteurs

Tests

Les matrices

Affichage

La fin...pour aujourd'hui

Une fonction très utile : source

La fonction `source` permet de lire et d'exécuter les commandes écrites dans un fichier.
Essayez de taper les commandes suivantes dans un fichier toto.R :

```
a<-a+5
cat("Coucou\n")
```

Sauvez le fichier. Puis tapez :

```
a<-a+5
source("toto.R")
```

Session et répertoire de travail

Il est conseillé de créer un répertoire MonProjet pour stocker :

- les programmes R
- les jeux de données
- l'historique des commandes de la session
- les données créées au cours de la session

Pour cela, à chaque début de session

- choisir Fichier $\longrightarrow$ Changer le répertoire courant
- donner le chemin de MonProjet
- récupérer les données : Fichier $\longrightarrow$ Charger l'environnement de travail

Sauvegarder son environnement en fin de session :

- répondre OK à la question "Sauver l'environnement de travail?"

Note peut charger inutilement la mémoire de R. Peut aussi être dangereux quand on programme (variables non initialisées).

Espace de travail

Notion clé : l'organisation $\Rightarrow$ ne pas hésiter à créer des répertoires correspondant aux différents aspects et projets de travail.

Un environnement de travail avec quelques sous-répertoires pourrait ressembler à ça :

```
+--> Racine
      +--> Programmes
            +--> Resultats
      +--> Redaction
            +--> Rapport
            +--> Figures
      +--> Divers
```