Statistiques avec R

Jeanne Charoy - Automne 2021

Description du cours

- 6 TDs de 2h
- Solidifier vos connaissances en statistiques
- Apprendre à utiliser le logiciel d'analyse statistique R
  - A la fin du cours vous devriez être capable de conduire des analyses simples avec R et créer des représentations graphiques de vos données.
- Evaluation: a la fin de chaque TD, il y aura une fiche de petits exercices à compléter, pour un total sur 20 points à la fin

R pour les stats

Gratuit

Grande communauté en ligne, facile de trouver des réponses a vos questions sur stackexchange etc...

Tres flexible, possible de faire toutes sortes de test (grands nombres de packages)

- Anova
- Regression
- Mixed-effect models
- Factor analyses….

Syntaxe un peu compliquee

“Learning curve”

Installer R & RStudio

Site: https://www.r-project.org/

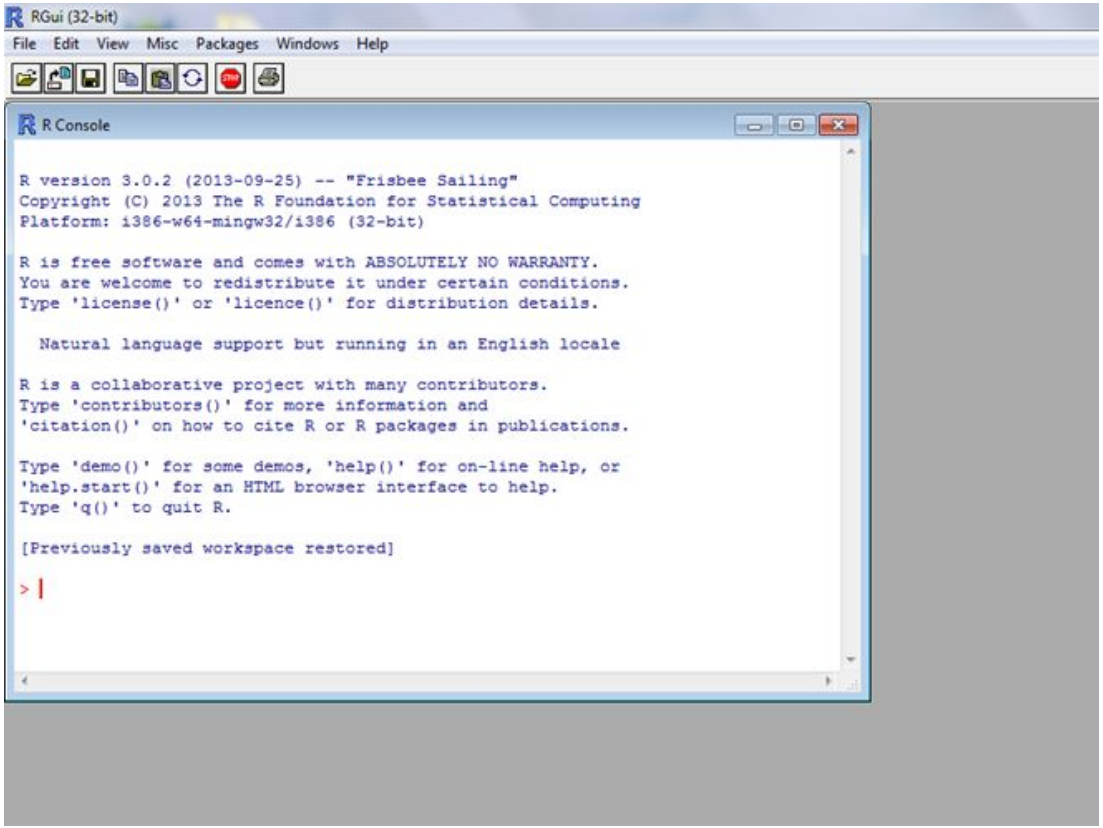
Choisis la première version miroir (maintenu par RStudio)

Download en fonction de ton OS

Ensuite va sur: https://www.rstudio.com/

Download la version Desktop qui est gratuite

R ressemble a ca



Les bases de R

Command line: R utilise une syntaxe/du code

Du coup deux trucs auxquels faire attention:

1) La syntaxe en question → connaître les noms des fonctions, comment faire des calculs etc…
2) Les typos → par exemple, la fonction Anova() n'est pas la même que la fonction anova()...

On y reviendra régulièrement pendant le cours - c'est normal de faire des erreurs en code, ce qu'il faut surtout apprendre c'est comment les identifier

RStudio window

Review de la fenetre RStudio

- Console window: tu peux y faire toutes tes operations (mais elles ne seront pas sauvegarde pour la prochaine session). Essaie de taper: 2+2
- Environment window: ou toutes tes variables/objets apparaîtront. Ecris x <- 2 dans la console et vois ce qui se passe.
- Files window: liste les fichiers dans l’espace de travail dans lequel tu te trouves.
    - Plots window: affichera tes graphs
    - Packages: pour installer de nouvelles fonctionalites sur R (plus a la slide d’apres)
    - Help: pour obtenir de l’aide sur les fonctions etc...

Library & Packages

R est super flexible

Les packages contiennent des fonctions additionnelles pas incluses dans le R de base

Open source: tout le monde peut créer et distribuer de nouveaux packages - ils sont souvent peer-reviewed.

Deux façons d'installer de nouveaux packages:

1) Depuis la fenêtre "packages"
2) En tapant install.packages('PackageName') dans la console

En utilisant une de ces méthodes, installe le package 'ggplot2'

Packages & Library

C’est pas fini.

Une fois que le package est installe il faut le “library” pour dire a R que tu vas l’utiliser pendant cette session

Deux facons:

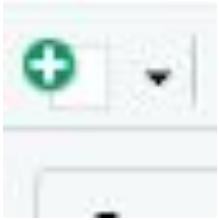
- Dans la fenetre packages, coche la case en face du package
- Ecris library(PackageName) dans la console.

Library le package ggplot

Creer un script

La console n’est pas super pratique pour garder une trace des opérations que tu as faites avec R.

A la place, crée un ‘script’ en cliquant sur ce bouton en haut a gauche (en dessous de “File”)



Mais attends un peu, d’abord on va faire un projet

Projet R

C’est en gros un dossier qui va contenir tous tes fichiers de données, tes graphs, scripts etc..

C’est bien pour rester organisé et pas perdre ses fichiers.

Fais: Files > New Project…

Choisis “New Directory”, empty project, appelle ton dossier “tdStatsM2S1” et choisis un endroit pour le sauvegarder.

Ensuite creer un script, et sauvegarde le en tant que “td1”

R est une calculatrice

Tu peux faire toutes les opérations de base +, -, *, /, et exponents

Essaie:

10+10

exp(2)

sqrt(4)

4^2

Assignments

Tu peux *assigner* des valeurs, matrices, vecteurs a un *objet* en utilisant <-

Objet: en general une lettre ou string de lettre dans laquelle tu “stores” des valeurs, formules etc…

Essaies:

X <- 2

X

X+X

Vecteurs avec R

Un vecteur est une liste d'elements de meme type (par exemple tous des integer, ou tous des lettres etc…)

La fonction pour creer un vecteur est c(...)

Essaies:

vecteur1<-c(1,2,3,4)

vecteur1

Tu peux facilement concaténer deux vecteurs:

vecteur2<-c(5,6,7,8)

c(vecteur1, vecteur2, 9, 10)

D’autres types de vecteurs

- Character vectors
  - fruit<-c(“pomme”, “poire”)
- Logical vectors
  - Pratique pour determiner si une valeur dans un autre vecteur est presente ou non
  - Vecteur1 > 10 (i.e., y a t’il une valeur superieure a 10 dans le premier vecteur?)

ATTENTION: si tu combines des elements de types differents dans un vecteur, ils seront transforme pour etre du meme type:

c(FALSE, 3) → FALSE devient 0 et TRUE devient 1, i.e. transforme en integer

c(2, “abc”) → “3”, “abc”, les deux sont des characters

Calcul simple avec des vecteurs

Les operations sont distribues sur les elements d’un vecteur.

Essaie:

a<-c(1,2,3)

5*a

Qu’obtient-on?

Variables & data type

Variable quantitative (continuous): contiennent des valeurs numériques faisant référence à une unite de mesure reconnue / peut prendre un grand nombre de valeurs en général. Exemple: l'age, la taille, etc…

Variable qualitative: contiennent des valeurs qui expriment une qualité, un etat, une condition, un statut unique et exclusif. Exemple: le sexe, la religion, etc...

Categorise les variables suivantes

- Le groupe sanguin (A, B, AB, O)
- Le salaire (1000$/mois, 2030$/mois, 750$/mois etc..)
- L'affiliation politique (Repubiclain, Democrate, …)
- Score sur une echelle de Likert (e.g., echelle de 1 a 7 qui mesure le teux de satisfaction)

Sur R:

monGroupeSanguin <- O

Les types de données sur R

- Numeriques (decimales): 1.2, 3.14 etc…
- Integer: 1, 2, 44, -8
- Logical/Boolean: a proposition that is TRUE or FALSE, (e.g., 2 > 1; x == y)
- Character/Strings: “a”, “sentence one”,...
- **Factor: equivalent d’une variable catégorique**

Exercice: Avec les connaissances acquises jusqu’ici on peut creer une variable quantitative avec R. Imaginons que je veux pouvoir manipuler la variable salaireAnnuel pour lesquels j’ai collecte les donnes suivantes:

- Personne A: 10000
- B: 23000
- C: 100000

Je veux savoir a quoi s’elevera la somme totale du salaire de chacun apres 4 ans.

Revenons sur les facteurs/factors

Ce sont des variables qualitatives (categoriques). R est capable de determiner les “niveaux” de ces variables.

En reprenant le vecteur “fruit” de tout a l’heure, essaie:

fruits<-as.factor(fruits) #cette ligne transforme le type des données contenues dans le vecteur en facteur plutôt que caracteres.

Maintenant “appelle” le vecteur fruits. Tu devrais voir:

```
> fruits
[1] apple pear
Levels: apple pear
> |
```

Les elements de fruits peuvent seulement prendre ces

deux valeurs a present. Ce sont les niveaux de la catégorie “fruits’

Structure de donnees sur R

On a parler des vecteurs mais il y a aussi:

- Les listes: peuvent combiner des elements de differents type
  - Function: list()
- Les facteurs: categories
- Les matrices: des tableaux a deux dimensions ou chaque column/row est un vecteur
- Les **data frames (en gros un tableau excel):** une list de vecteurs de meme taille. Chaque column/row peut avoir un nom. Super pratique pour storer ses donnees.

Data frame

Ca ressemble a ca

Dans cet exemple: une colonne par variable et une ligne par observation

se	RT	Trial	test	img	ortho	type	fami_canon	new_canon
1	1439	90	test 1	bredento	C	nasal	FALSE	TRUE
1	1398	90	test 2	bredento	C	nasal	FALSE	TRUE
0	1098	90	test 3	bredento	C	nasal	FALSE	TRUE
1	2227	113	test 2	bytto	C	flap	FALSE	TRUE
0	699	113	test 1	bytto	C	flap	FALSE	TRUE
0	778	113	test 3	bytto	C	flap	FALSE	TRUE
0	772	125	test 1	gleanty	C	nasal	FALSE	TRUE
0	684	125	test 3	gleanty	C	nasal	FALSE	TRUE
0	1412	125	test 2	gleanty	C	nasal	FALSE	TRUE
0	1262	104	test 2	pittery	C	flap	FALSE	TRUE
0	777	104	test 1	pittery	C	flap	FALSE	TRUE
0	950	104	test 3	pittery	C	flap	FALSE	TRUE
1	1968	59	test 1	ruttow	C	flap	FALSE	TRUE
1	738	59	test 3	ruttow	C	flap	FALSE	TRUE
1	2397	59	test 2	ruttow	C	flap	FALSE	TRUE
0	1072	48	test 1	sceanetty	C	flap	FALSE	TRUE
1	1083	48	test 3	sceanetty	C	flap	FALSE	TRUE

Creer une data frame a partir de vecteurs

participants <- c(“truc”, “muche”, “bidule”)

reactionTimes<-c(1300, 500, 800)

myData<-data.frame(participants, reactionTimes)

Pour visualiser ta data frame, utilises la commande: View(myData)

Pour voir la liste des variables dans ta data frame: str(myData)

Tu peux aussi voir dans la fenetre Environment

Manipuler les donnees dans une dataframe

- Access a column by name: $
  - myData$participants
- Access a value with an index
  - myData$participants[2] (i.e., je veux le valeur de la deuxieme case de la colonne participant)
- Change a value based on index:
  - myData$participant[2] <- “machin”
- Find a value based on a criteria (i.e., conditional selection)
  - myData[myData$reactionTimes > 500,]
  - Qu’est ce qu’on obtient?
- Get data for all rows that meet a certain column criteria:
  - myData[myData$reactionTimes>500,]

Les operateurs conditionnels

Superieur a, inferieur a: >, <

Egal: ==

Superieur ou egal, inferieur ou egal: <=, >=

“ET”, “OU” et “NON”: &, |, !

Par exemple “n’est pas egal a” s’ecrit !=

Subset a dataframe + add a variable

Cree une nouvelle dataframe qui ne contient qu'un subset des valeurs de la dataframe originale:

myData_subset<-subset(myData, reactionTimes>500)

Ajoute une variable

myData_subset$accuracy<-c(1, 0.75)

Ajoute une variable qui est une transformation de variables existantes:

myData_subset$accuracyPercent<-myData_subset$accuracy * 100

Fonction sur R

A function is a serial set of actions designed to perform a specific task (Think: copy, paste, average, compute p-values etc…)

- E.g., plot(height, weight)
  - Plot is the ***function name***
  - Height and weight are the **"*arguments*"** – the values that are passed to the function

Many functions use *positional matching*, based on where an argument is in the parentheses

- In plot function, R assumes that the first argument is the x-variable and the second argument is the y-variable

Fonctions sur R

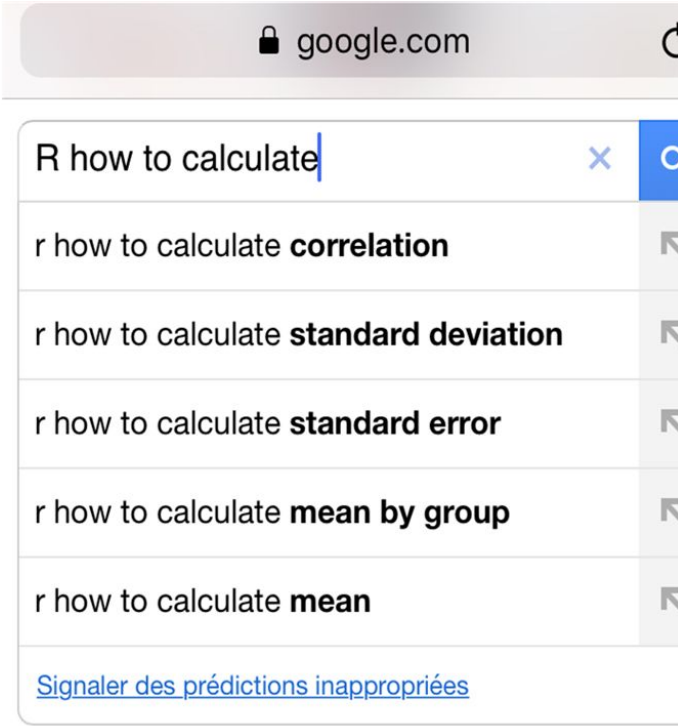
Some functions also have *optional arguments*, which are used to specify or change an aspect of the function

- E.g. plot(height, weight, color= “red”)
- color= “red” indicates the color the points should be on the plot, but it doesn’t have to be specified for the function to work.

A l'aide

Pour t'aider a faire les opérations dont tu as besoin sur R, n'hesite pas a cherhcer de l'aide. Tu peux:

1) Utiliser la fonction help() sur R, ou la fonction ?
   a) Essaie help(c) et ?data.frame
2) Utiliser la documentation R & les manuels
   a) https://www.rdocumentation.org/
   b) https://cran.r-project.org/manuals.html
3) Utiliser Google/stackexchange
   a) C'est ok de copier les bouts de codes/function stats etc trouver sur stack… ca sert a rien de reinventer le roue. Ce qu'il faut pas c'est piquer les donnees des autres. Cela dit si tu utilises des package particulier pour tes analyses, c'est mieux de les citer.



Lab activity: importer des data et jouer avec

Commence par récupérer le fichier csv “chronolex.csv” et store le dans ton le dossier du projet que tu as créé précédemment

(R est capable de lire different format, .xcel, .txt… mais csv est le plus R-friendly et moins gourmand en memoire que xcel.

Ensuite importe les données dans R avec la function read.csv().

Tu peux aussi faire: import dataset > from csv > browse

Import data

To check that the data was imported correctly

> fix(chronolex) or >View(chronolex)

Should display as an excel sheet like window:

–With **fix** you can edit individual points in the window

–With **View** you can't but you can search the dataset

The str() function is also helpful to see the list of the variables in your data frame.

Descriptive stats with R

Les stats descriptives = structure et représente l'information contenue dans les données (e.g., moyenne, mediane, frequence de certaines valeurs etc…)

Toujours commencer par explorer votre base de données avec ce genre de stats avant de passer aux statistiques inférentielles (i.e., t-test, anova, regression etc..). C'est tres utile pour s'assurer qu'il n'y pas de truc bizarre dans les donnees, avoir une bonne vue d'ensemble de ce qu'on etudie, quels analyses seront utiles/pertinentes et eviter de perdre du temps plus tard.

Chronolex (trouve sur OSF)

Description: Chronolex (Ferrand, Brysbaert, Keuleers, New, Bonin, Méot, Augustinova, & Pallier, 2011) involved the collection of naming, lexical decision and progressive demasking data for 1,826 monosyllabic monomorphemic French words. Thirty-seven participants (psychology students from Blaise Pascal University) were tested in the naming task, 35 additionnal participants in the lexical decision task and 33 additionnal participants (from the same pool) were tested in the progressive demasking task.

Ferrand, L., Brysbaert, M., Keuleers, E., New, B., Bonin, P., Méot, A., Augustinova, M., & Pallier, C. (2011). Comparing word processing times in naming, lexical decision, and progressive demasking: Evidence from Chronolex. Frontiers in Psychology, 2:306. doi: 10.3389/fpsyg.2011.00306

La base de donnees contient des informations sur la frequence des mots, le nombre de phonemes, le nombre de voisins etc…

Moyenne, ecart type etc...

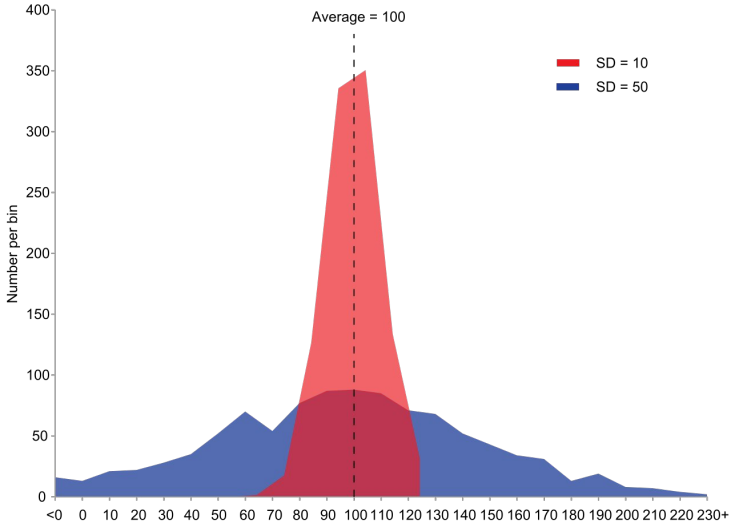
Moyenne: mean() - mesure la moyenne arithmetique d'un echantillon

Ecart type: sd() - mesure la dispersion des valeurs d'un echantillon autour de la moyenne. C'est la racine carree de la variance

Variance: var() - aussi une mesure de la dispersion, c'est la moyenne des carrees des ecarts a la moyenne

Mediane: median() - valeur qui separe la moitie inferieure de la moitie superieure d'un ensemble de valeurs.

Quantile: quantile() - divisent les donnees en intervalles contenant le meme nombre de donnees (la mediane est le quantile qui separe les donnees en deux)



Quantile avec differentes coupures

To get different percentage points of a variable:

```
> pvec <- seq(0,1,0.1)
```

{sequence from 0 – 1 in .1 increments}

```
> pvec
 [1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
> quantile(chronolex$Nphon,pvec)
  0%  10%  20%  30%  40%  50%  60%  70%  80%  90% 100%
   1    2    3    3    3    3    3    4    4    4    6
```

Valeurs manquantes (i.e., missing values)

Certaines fonction ne vont pas fonctionner si il y a des valeurs manquantes dans les donnees (i.e., les valeurs manquantes sont specifies avec “NA” - cherche des instances de NA dans chronolex en faisant View(chronolex))

```
> mean(chronolex$FLPrt)
```

→ what do you get?

Pour eliminer les valeurs manquantes, tu peux utiliser l’argument **na.rm=TRUE** (rm est l’abbreviation de remove)

```
> mean(chronolex$FLPrt,na.rm=TRUE)
```

→ what do you get now?

Dealing with missing values

Pour voir combien il y a de valeurs manquantes pour une variable tu peux utiliser la fonction is.na

is.na(chronolex$FLPrt)

Mais c'est pas super pratique si il y a beaucoup de donnees

Plus simple c'est d'utiliser la function summary() - en plus ca te donnes plein d'autres infos interessantes (moyenne etc.)

summary(chronolex$FLPrt)

Recoding variables

Il n'y pas vraiment de variables categoriques dans la bases chronolex, mais c'est possible d'en creer en recodant des variables quantitatives.

Imagine que tu veux traiter la frequence comme une variable categorique plutot que continue, avec les niveaux "high" et "low" (pour haute freq et basse freq). Tu as besoin de savoir comment "decouper" la variable en deux (pour faire les deux categories high et low). On pourrait faire par rapport a la moyenne mais comme tu peux voir elle est beaucoup plus eleve que la mediane, probablement a cause de certaines valeurs tres haute (le max est ~26000) mais qui restent rare. Du coup c'est mieux de couper par rapport a la mediane dand ce cas.

```
summary(chronolex$freqfilms)
```

```
Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
```

```
   0.010     3.002    11.845   325.182    55.562 25988.480
```

Comme la mediane est 11.8 tu decides que toutes valeurs inferieures a 19 sera considere "low frequency" et toutes valeurs superieures "high frequency"

Recoding variables

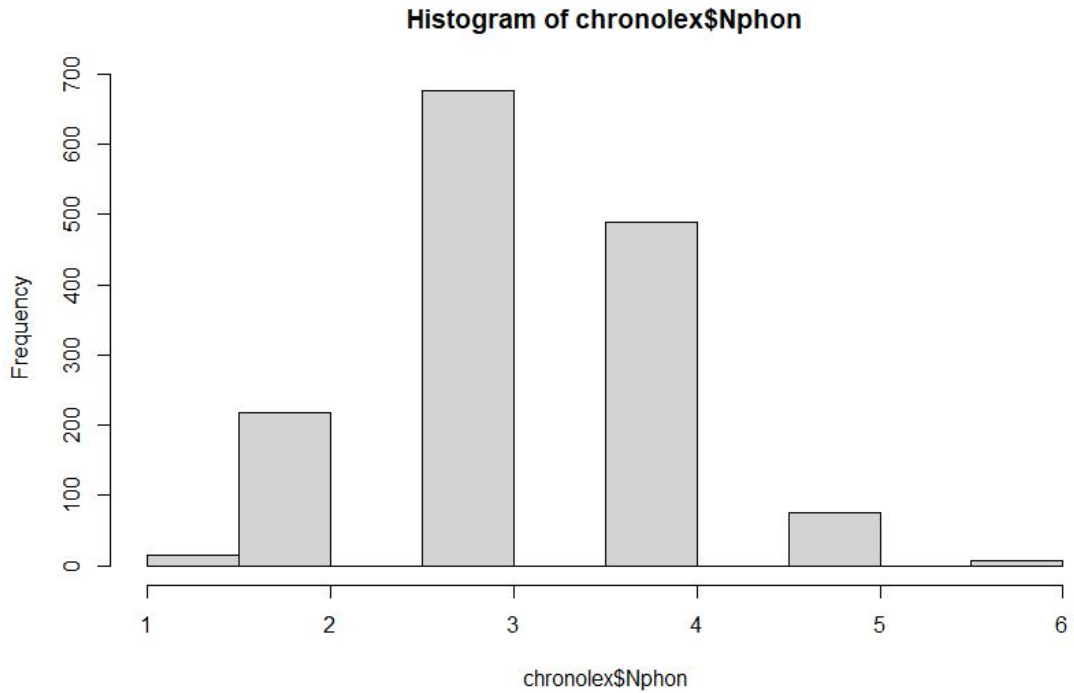
```
chronolex$freqfilmsCAT[chronolex$freqfilms >19]<-"high"

chronolex$freqfilmsCAT[chronolex$freqfilms <19]<-"low"
```

Pour s’assurer que la nouvelle variable freqfilmsCAT est bien un facteur, tu peux utiliser str(chronolex$freqfilms) ou bien regarder dans la fenetre Environment.

Si non, utilise: chronolex$freqfilmsCAT<-as.factor(chronolex$freqfilmsCAT)

Simple graphs: histograms

La fonction hist() sort un histogramme de la variable comme montre ici



Aggregating values based on other values

La fonction aggregate() est super utile

Par exemple, tu peux calculer le temps de reaction (rt) moyen pour une tache (la tache NMG par exemple), en fonction de la frequence des mots.

aggregate(NMGrt~freqfilmsCAT, chronolex, FUN="mean")

freqfilmsCAT <chr>	FLPrt <dbl>
high	630.6128
low	678.2814

Aggregate()

The FUN argument lets you specify the function you want applied to the variable.

It could be sum: aggregate(NMGrt ~ filmsfreqCAT, chronolex, FUN= “sum”)

Or median, sd, var, sqrt…

Bar plots

Maintenant imaginons qu'on veut faire un barplot qui represente les moyennes que l'on vient de calculer, avec les barres d'erreur autour de la moyenne.

Pour ca on a besoin:

1) Des moyennes
2) Des erreur type, calculer a partir de l'ecart type (pour creer les error bars)
    a) Erreur type en fr: c'est une mesure d'erreur d'estimation de la moyenne (cause par des erreurs d'echantillonage possible etc…). C'est l'ecart type divise par la racine du nombre de participants
    b) Donc il nous faut l'ecart type (mesurer avec la fonction sd())

Creating a bar plot

```
meanFLPrt<-aggregate(NMGrt~freqfilmsCAT, chronolex,FUN= “mean”)

sdFLPrt <-aggregate(NMGrt~freqfilmsCAT, chronolex,FUN = “mean”)
```

Il y avait 33 participant dans la tache de naming donc N = 33

```
meanNMGrt$sd <- sdNMGrt$NMGrt

meanNMG$sem <- meanNMG$sd / sqrt(33)

meanNMG
```

Maintenant on a tout ce qu’il faut.

GGPLOT

Ggplot est un super package pour faire de la visualisation de donnees - mais pas super intuitif au debut.

La premiere ligne est toujours:

ggplot(dataset, aes(x, y)) -- definit les donnees, et les axes x et y

Ensuite tu rajoutes des arguments pour definir quel genre de plot tu veux:

```
ggplot(dataset, aes(x, y))+
 geom_bar(stat="identity", position = "dodge") -- ca nous donne un barplot
```

GGPLOT

Pour ajouter les error bars

```
ggplot(meanNMG, aes(freqfilmsCAT, NMGrt))+
  geom_bar(stat = "identity", position="dodge")+
   geom_errorbar(aes(ymin=NMGrt-sem, ymax=NMGrt+sem))
```

GGPLOT

Pour ajouter les error bars qui prennent pas tout le plot

```
ggplot(meanNMG, aes(freqfilmsCAT, NMGrt))+
  geom_bar(stat = "identity", position="dodge")+
   geom_errorbar(aes(ymin=NMGrt-sem, ymax=NMGrt+sem), width=0.2)
```

A faire avant vendredi 19/11 (prochain cours)

Completer la fiche d’exercice (notee sur 2 points).

Envoyer moi vos fiches avec les reponses par mail:
jeanne.charoy@univ-grenoble-alpes.fr