Arthur Charpentier

e-mail : charpentier.arthur@uqam.ca

url : http ://freakonometrics.hypotheses.org/

Data Science pour l'Actuariat, Mars - Juin 2015

DataMining & R

avec Stéphane Tufféry

A Brief Introduction To More Advanced R

"*An expert is a man who has made all the mistakes which can be made, in a narrow field* " N. Bohr

References

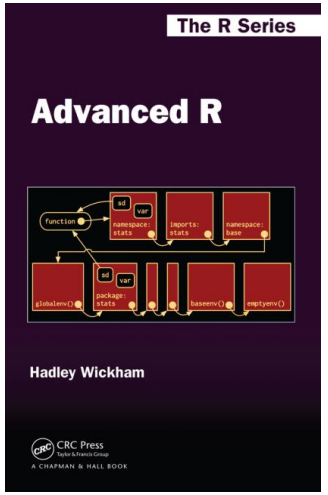
Wickam, H. Advanced R. CRC Press, 2014.
(cf http://adv-r.had.co.nz/)

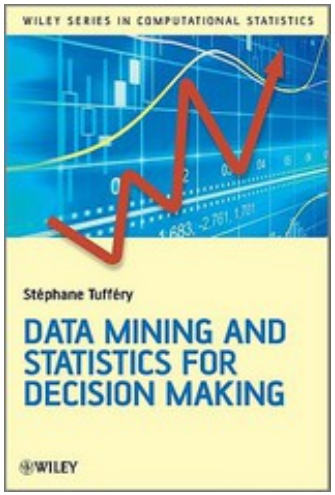
Tufféry, S. Data Mining and Statistics for Decision Making. Wiley, 2013.

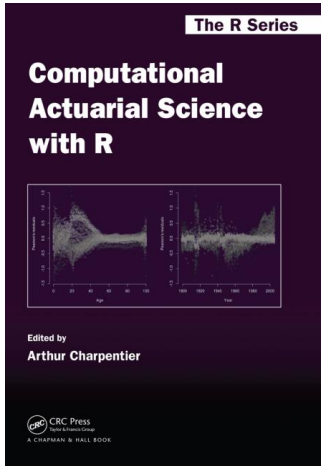
Charpentier, A. Computational Actuarial Science with R. CRC Press. 2014.

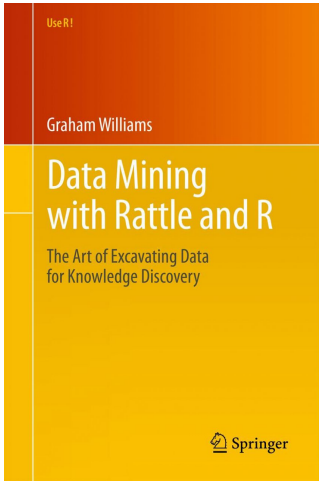
Williams, G. Data Mining with Rattle and R. Springer. 2011.

Zhao, Y. R and Data Mining : Examples and Case Studies. 2013
(cf http://cran.r-project.org/contrib/)









Additional Information, R in Insurance, 2015

Amsterdam School of Economics

Home Education Research News & Events Sections People About the ASE Alumni Contact

- News & Events
  - Events
  - Archive
  - R in Insurance
    - Talk proposal submission
    - Registration and fees
    - Programme
    - Travel and accommodation
    - Contact
  - Rob in Insurance

R in Insurance 2015

Following two successful events at Cass in Londen, R in Insurance will travel across the Channel to Amsterdam in 2015. The third conference on R in Insurance will be held at the Amsterdam School of Economics in Amsterdam, The Netherlands, on Monday, 29 June 2015.

The intended audience of the conference includes both academics and practitioners who are interested in practical aspects of insurance and more specifically the applications of R in Insurance.

Topics

This one-day conference will focus on all subdisciplines of actuarial science where professionals can use R, the lingua franca for statistical computation. Topics covered include risk management, reserving, pricing, loss modelling and the use of R in a production environment.

The central theme is how one can use R as a primary tool for insurance risk management, analysis and modelling.

Additional Information, R in Insurance, 2015

Amsterdam School of Economics

Home Education Research News & Events Sections People About the ASE Alumni Contact

- News & Events
  - Events
  - Archive
  - R in Insurance
  - Rob in Insurance
    - Programme
    - Location
    - Contact

Rob in Insurance

After a distinguished career, Rob Kaas, Professor of Actuarial Statistics at the University of Amsterdam, will retire in the summer of 2015. He will be granted emeritus status at the University of Amsterdam, where he started working as early as 1969.

Among many other aspects, Prof. Kaas is author of *Modern Actuarial Risk Theory - Using* R, one of the main textbooks in Actuarial Science that has been translated into many languages, and of over 130 scholarly research papers in Actuarial Science; Program Director of the renowned Actuarial Science Education Programs at the University of Amsterdam; Managing Editor of Insurance: Mathematics and Economics, the leading journal in Actuarial Science; and organizer of the first IME conference in Amsterdam in 1997, starting the series of what are considered the major academic conferences in Actuarial Science today.

R

R is an *interpreted language* where expressions are entered into the R console, and a program within the R system (called the interpreter) executes the code, unlike C but like Javascript.

```
> 2+3
[1] 5
```

R is an Object-Oriented Programming language. The idea is to create various objects that contain useful information, and that could be called by other functions (e.g. graphs).

```
> a <- 2+3
> print(a)
[1] 5
```

R

A package is a related set of functions, including help files, and data files, that have been bundled together and is shared among the R community

```
> install.packages("quantreg", dependencies=TRUE)
> library(quantreg)
```

S3 and S4 classes

"*Everything in S is an object. Every object in S has a class.*"

S3 is a primitive concept of classes, in R To define a class, use

```
> person3 <- function(name,age,
    weight,height){
crct<-list(name=name,age=age,
    weight=weight,height=height)
class(crct)<-"person3"
return(crct)}
```

To create a person, use

```
> JohnDoe3 <- person3(name="John",
    age=28, weight=76, height=182)
> JohnDoe3
$name
[1] "John"
$age
[1] 28
$weight
[1] 76
$height
[1] 182
attr(,"class")
[1] "person3"
```

S3 and S4 classes

If we want a function that returns the BMI (Body Mass Index), use

```
> BMI3 <- function(object,...) {
    return(object$weight*1e4/
    object$height^2)}
```

e.g.

```
> BMI3(JohnDoe3)
[1] 22.94409
```

S3 and S4 classes

The analogous S4 version is

```
> setClass("person4",
    representation(name="
    character", age="numeric",
    weight="numeric", height="
    numeric"))
> JohnDoe4 <- new("person4",name
    ="John",age=28, weight=76,
    height=182)
```

```
> JohnDoe4
An object of class "person"
Slot "name":
[1] "John"
Slot "age":
[1] 28
Slot "weight":
[1] 76
Slot "height":
[1] 182
```

Here we have

S3 and S4 classes

Observe that

```
> JohnDoe3$age
[1]  28
> JohnDoe4@age
[1]  28
```

To create our BMI function, use

```
> setGeneric("BMI4",function(
    object,separator) return(
    standardGeneric("BMI")))

> setMethod("BMI4", "person4",
function(object){return(object
    weight*1e4/objectheight^2)})

> BMI4(JohnDoe)
[1] 22.94409
```

Numbers, in R

```
> x <- exp(1)
> x
[1] 2.718282
```

About large numbers,

```
> 1/0
[1] Inf
> .Machine$double.xmax
[1] 1.797693e+308
> 2e+307<Inf
[1] TRUE
> 2e+308<Inf
[1] FALSE
```

R has a recycling rule

```
> x <- c
    (100,200,300,400,500,600,700,800)

> y <- c(1,2,3,4)
> x+y
[1] 101 202 303 404 501 602 703
    804
```

```
> for(i in 1:2){
+   nom_var <- paste0("x", i)
+   assign(nom_var, rpois(5,7))
+ }
> x1
[1] 4 6 5 8 9
> x2
[1] 6 9 5 8 5
```

Numbers, in R

About naming components of a vector

```
> x <- 1:6
> names(x) <- letters[1:6]
> x
a b c d e f
1 2 3 4 5 6
> x[2:4]
b c d
2 3 4
> x[c("b","c","d")]
b c d
2 3 4
```

It is possible to consider some active building variables

```
> x <- runif(4)
> x
[1] 0.8018263 0.1685260
    0.5900765 0.8230110
> x
[1] 0.8018263 0.1685260
    0.5900765 0.8230110
> library(pryr)
> x %<a-% runif(1)
> x
[1] 0.08434417
> x
[1] 0.9253761
> x <- 1:2
> x
[1] 0.1255551
```

Numbers ($\in \mathbb{R}$), in R

```
> (3/10-1/10)
[1] 0.2
> (3/10-1/10)==(7/10-5/10)
[1] FALSE
> (3/10-1/10)-(7/10-5/10)
[1] 2.775558e-17
> all.equal((3/10-1/10),(7/10-5/
    10))
[1] TRUE
> (eps<-.Machine$double.eps)
[1] 2.220446e-16
```

```
> set.seed(1)
> U <- runif(20)
> U[1:4]
[1] 0.2655087 0.3721239
    0.5728534 0.9082078
> options(digits = 3)
> U[1:4]
[1] 0.266 0.372 0.573 0.908
> options(digits = 22)
> U[1:4]
[1] 0.2655086631420999765396
    0.3721238996367901563644
    0.5728533633518964052200
    0.9082077899947762489319
```

Matrices, in R

```
> (N<-rpois(20,5))
 [1] 9 3 6 3 4 4 1 4 8 4 5 5 5 3
     7 6 7 2 6 4
> (M=matrix(N,4,5))
     [,1] [,2] [,3] [,4] [,5]
[1,]    9    4    8    5    7
[2,]    3    4    4    3    2
[3,]    6    1    5    7    6
[4,]    3    4    5    6    4
> dim(N)=c(4,5)
> N
     [,1] [,2] [,3] [,4] [,5]
[1,]    9    4    8    5    7
[2,]    3    4    4    3    2
[3,]    6    1    5    7    6
[4,]    3    4    5    6    4
```

```
> M>.6
[,1]  [,2]  [,3]  [,4]  [,5]
[1,] FALSE FALSE TRUE  TRUE  TRUE
[2,] FALSE TRUE  FALSE FALSE TRUE
[3,] FALSE TRUE  FALSE TRUE  FALSE
[4,] TRUE  TRUE  FALSE FALSE TRUE
> M[1]
[1] 9
> M[1,1]
[1] 9
> M[1,]
[1] 9 4 8 5 7
> M[,1]
[1] 9 3 6 3
```

Matrices, in R

```
> u<-1:24
> dim(u)=c(6,4)
> u
     [,1] [,2] [,3] [,4]
[1,]    1    7   13   19
[2,]    2    8   14   20
[3,]    3    9   15   21
[4,]    4   10   16   22
[5,]    5   11   17   23
[6,]    6   12   18   24
> str(u)
 int [1:6, 1:4] 1 2 3 4 5 6 7 8
   9 10 ...
> colnames(u)=letters[1:4]
> rownames(u)=LETTERS[10:15]
```

```
> u[c("K","N"),]
  a  b  c  d
K 2  8 14 20
N 5 11 17 23
> u[c(1,2),]
  a  b  c  d
J 1  7 13 19
K 2  8 14 20
> u[c("K",5),]
Error in u[c("K", 5), ] :
    subscript out of bounds
```

Memory Issues, in R

R holds all the objets in 'memory', and only limited amount of memory can be used.

classical R message : cannot allocate vector of size ____ MB

big datasets are often larger then the size of the RAM that is available

on Windows, the limits are 2Gb and 4Gb for 32-bit and 64-bit respectively

```
> rm(list=ls())
> memory.size()
[1] 15.05
> memory.limit()
[1] 3583
> memory.limit(size=8000)
Error in memory.limit(size = 8000) :
  don't be silly!: your machine has a 4Gb address limit
> memory.limit(size=4000)
[1] 4000
```

Memory Issues, in R

```
> x3 <- 1:1e3
> x4 <- 1:1e4
> x5 <- 1:1e5
> x6 <- 1:1e6
```

```
> object.size(x3)
4024 bytes
> object.size(x4)
40024 bytes
> object.size(x5)
400024 bytes
> object.size(x6)
4000024 bytes
```

```
> z1 <- matrix(0,1e+06,3)
> z2 <- matrix(as.integer(0),1e
    +06,3)
```

```
> object.size(z1)
24000112 bytes
> object.size(z2)
12000112 bytes
```

```
> z3 <- matrix(0,1e+07,30)
Error: cannot allocate vector of
    size 2.2 Gb
```

Memory Issues, in R

Alternative : matrices are stored on disk, rather than in RAM, and only elements needed are read from the disk, and catched in RAM

```
> z1 <- matrix(as.integer(5), 3, 4)
> object.size(z1)
248 bytes
> z1 <- matrix(as.integer(5), 30, 40)
> object.size(z1)
5000 bytes
> z1 <- matrix(as.integer(5), 300, 400)
> object.size(z1)
480200 bytes
```

```
> z2 <- big.matrix(3, 4, type="integer", init=5)
> object.size(z2)
664 bytes
> z2 <- big.matrix(300, 400, type="integer", init=5)
> object.size(z2)
664 bytes
> z2
An object of class "big.matrix"
Slot "address":
<pointer: 0x1aa0e220>
```

Integers, in R

```
> (x_num=c(1,6,10))
[1]  1  6 10
> (x_int=c(1L,6L,10L))
[1]  1  6 10
> object.size(x_num)
72 bytes
> object.size(x_int)
56 bytes
> typeof(x_num)
[1] "double"
> typeof(x_int)
[1] "integer"
```

```
> is.integer(x_num)
[1] FALSE
> is.integer(x_int)
[1] TRUE
> str(x_num)
 num [1:3] 1 6 10
> str(x_int)
 int [1:3] 1 6 10
```

```
> c(1,c(2,c(3,c(4,5))))
[1] 1 2 3 4 5
```

Factors, in R

```
> (x <- c("A","A","B","B","C"))
[1] "A" "A" "B" "B" "C"
> (x <- factor(x))
[1] A A B B C
Levels: A B C
> unclass(x)
[1] 1 1 2 2 3
attr(,"levels")
[1] "A" "B" "C"
> model.matrix(~0+x)
  xA xB xC
1  1  0  0
2  1  0  0
3  0  1  0
4  0  1  0
5  0  0  1
```

```
> x[1]
[1] A
Levels: A B C
> x[1,drop=TRUE]
[1] A
Levels: A
> x <- factor(x, labels=c("Young
    ", "Adult", "Senior"))
> x
[1] Young Young Adult Adult
    Senior
Levels: Young Adult Senior
> relevel(x,"Senior")
[1] Young Young Adult Adult
    Senior
Levels: Senior Young Adult
```

Factors, in R

```
> cut(U,breaks=2,labels=c("small","large"))
 [1] small small large large small large large large large small small small
[13] large small large small large large small large
Levels: small large
> table(cut(U,breaks=c(0,.3,.8,1),labels=c("small","medium","large")))
small medium large
    5     11     4
```

```
> x <- factor(c("b","a","b"))
> levels(x)
[1] "a" "b"
> x[3]="c"
Warning Message :
In `[<-.factor`(`*tmp*`, 3, value = "c") :
  invalid factor level, NA generated
> x
[1] b    a    <NA>
Levels: a b
```

Factors, in R

```
> X <- runif(20)
> group <- rep(c("A", "B"), c(8,
    12))
> mean(X[group=="A"])
[1] 0.526534
> mean(X[group=="B"])
[1] 0.5185935
> tapply(X,group,mean)
        A         B
0.5265340 0.5185935
> sapply(split(X,group),mean)
        A         B
0.5265340 0.5185935
```

Lists, in R

```
> x <- list(1:5,c(1,2,3,4,5),
+ a="test", b=c(TRUE,FALSE),
+ rpois(5,8))
> x
[[1]]
[1] 1 2 3 4 5
[[2]]
[1] 1 2 3 4 5
$a
[1] "test"
$b
[1]  TRUE FALSE
[[5]]
[1] 11 12  5  6  3
```

```
> str(x)
List of 5
 $  : int [1:5] 1 2 3 4 5
 $  : num [1:5] 1 2 3 4 5
 $ a: chr "test"
 $ b: logi [1:2] TRUE FALSE
 $  : int [1:5] 11 12 5 6 3
```

```
> names(x)
[1] ""  ""  "a" "b" ""
```

Lists, in R

```
> x <- list(list(1:5,c
    (1,2,3,4,5)),a="test",b=c(
    TRUE,FALSE),rpois(5,8))
> x
[[1]]
[[1]][[1]]
[1] 1 2 3 4 5
[[1]][[2]]
[1] 1 2 3 4 5
$a
[1] "test"
$b
[1]  TRUE FALSE
[[4]]
[1] 10  3  8 10  9
```

```
> is.list(x)
> is.recursive(x)
> x[[3]]
[1]  TRUE FALSE
> x["b"]
[1]  TRUE FALSE
> x[[1]]
[[1]]
[1] 1 2 3 4 5
[[2]]
[1] 1 2 3 4 5
> x[[1]][[1]]
[1] 1 2 3 4 5
```

Lists, in R

```
> list(abc=1)$a
[1] 1
> list(abc=1,a=2)$a
[1] 2
> list(bac=1)$a
NULL
> list(abc=1,b=2)$a
[1] 1
> list(bac=1,a=2)$a
[1] 2
> list(bac=1)$a
NULL
> list(bac=1)$b
[1] 1
```

Lists are standard with S3 functions

```
> f <- function(x) { return(x*
    (1-x)) }
> optim.f <- optimize(f,
    interval=c(0, 1), maximum=
    TRUE)
> names(optim.f)
[1] "maximum"   "objective"
> optim.f$maximum
[1] 0.5
```

Characters and Strings, in R

```
> cities <- c("New York, NY", "
    Los Angeles, CA", "Boston,
    MA")
> substr(cities, nchar(cities)
    -1, nchar(cities))
[1] "NY" "CA" "MA"
> unlist(strsplit(cities, ", "))
    [seq(2,6,by=2)]
[1] "NY" "CA" "MA"
```

```
"Be carefull of 'quotes'"
'Be carefull of "quotes"'
```

```
> library(stringr)
> tweet <- "Emerging #climate
    change, environment and
    sustainability concerns for
    #actuaries Apr 9 #Toronto.
    Register TODAY http://bit.ly
    /CIAClimateForum"
> hash <- "#[a-zA-Z]{1,}"
> str_extract(tweet, hash)
[1] "#climate"
> str_extract_all(tweet,hash)
[[1]]
[1] "#climate"    "#actuaries" "#
    Toronto"
```

Characters and Strings, in R

```
> str_locate(tweet, hash)
     start end
[1,]    10  17
> str_locate_all(tweet, hash)
[[1]]
     start end
[1,]    10  17
[2,]    71  80
[3,]    88  95
> urls <- "http://([\\da-z\\.-]+)\\.([a-z\\.]{2,6})/[a-zA-Z0-9]{1,}"
> str_extract(tweet,urls)
[1] "http://bit.ly/CIAClimateForum"
```

```
> email="^([a-z0-9_\\.-]+)@([\\da-z\\.-]+)\\.([a-z\\.]{2,6})$"
> grep(pattern = email, x = "charpentier.arthur@uqam.ca")
[1] 1
> grepl(pattern = email, x = "charpentier.arthur@uqam.ca")
[1] TRUE
> str_detect(pattern = email, string=c("charpentier.arthur@uqam.ca","@freakonometrics"))
[1]  TRUE FALSE
```

Characters and Strings, in R

Consider a simple sentence

```
> ex_sentence = "This is 1 simple sentence, just to play with, then
    we'll play with 4, and that will be more difficult"
> ex_sentence
[1] "This is 1 simple sentence, just to play with, then we'll play
    with 4, and that will be more difficult"
```

The first step is to create a corpus

```
> library(tm)
> ex_corpus <- Corpus(VectorSource(ex_sentence))
> ex_corpus
<<VCorpus (documents: 1, metadata (corpus/indexed): 0/0)>>
> inspect(ex_corpus)
[[1]]
<<PlainTextDocument (metadata: 7)>>
This is 1 simple sentence, just to play with, then we'll play with 4,
    and that will be more difficult
```

Characters and Strings, in R

Here we have one *document* in that corpus. We see if some documents do contain some specific words

```
> grep("hard", ex_sentence)
integer(0)
> grep("difficult", ex_sentence)
[1] 1
```

Since here we do not need the corpus structure (we have only one sentence) we can use more basic functions

```
> library(stringr)
> word(ex_sentence,4)
[1] "simple"
```

Characters and Strings, in R

To get the list of all the words

```
> word(ex_sentence,1:20)
 [1] "This"       "is"         "1"          "simple"     "sentence," "
    just"       "to"         "play"       "with,"      "then"       "we'll"
[12] "play"       "with"       "4,"         "and"        "that"       "
    will"       "be"         "more"       "difficult"
> ex_words <- strsplit(ex_sentence, split=" ")[[1]]
> ex_words
 [1] "This"       "is"         "1"          "simple"     "sentence," "
    just"       "to"         "play"       "with,"      "then"       "we'll"
[12] "play"       "with"       "4,"         "and"        "that"       "
    will"       "be"         "more"       "difficult"
> grep(pattern="w",ex_words,value=TRUE)
[1] "with,"  "we'll"  "with"   "will"
```

Characters and Strings, in R

We can count the occurence of w's or i's in each word

```
> str_count(ex_words,"w")
 [1] 0 0 0 0 0 0 0 0 1 0 1 0 1 0 0 0 1 0 0 0
> str_count(ex_words,"i")
 [1] 1 1 0 1 0 0 0 0 1 0 0 0 1 0 0 0 1 0 0 2
```

or get all the words with a l

```
> grep(pattern="l",ex_words,value=TRUE)
[1] "simple"    "play"      "we'll"     "play"      "will"      "
    difficult"
> grep(pattern="l{2}",ex_words,value=TRUE)
[1] "we'll" "will"
```

or get all the words with an a or an i

```
> grep(pattern="[ai]",ex_words,value=TRUE)
 [1] "This"      "is"        "simple"    "play"      "with,"     "
    play"
```

Characters and Strings, in R

or a punctuation symbol

```
> grep(pattern="[[:punct:]]",ex_words,value=TRUE)
[1] "sentence," "with,"     "we'll"     "4,"
```

It is possible, here, to create some WordCloud, e.g.

```
> require(wordcloud)
> wordcloud(ex_corpus)
> cols<-colorRampPalette(c("lightgrey",
    "blue"))(40)
> wordcloud(words = ex_corpus, max.
    words = 40, random.order=FALSE,
    scale = c(5, 0.5), colors=cols)
```

Characters and Strings, in R

The corpus can be used to generate a list of words along with counts of their occurrence.

```
> tdm <- TermDocumentMatrix(ex_
    corpus)
> inspect(tdm)
<<TermDocumentMatrix (terms: 14,
     documents: 1)>>
Non-/sparse entries: 14/0
Sparsity           : 0%
Maximal term length: 9
Weighting          : term
    frequency (tf)
```

```
                Docs
Terms            1
  and            1
  difficult      1
  just           1
  more           1
  play           2
  sentence,      1
  simple         1
  that           1
  then           1
  this           1
  we'll          1
  will           1
  with           1
  with,          1
```

Characters and Strings, in R

Note that the Corpus should be cleaned. This involves the following steps :

- convert all text to lowercase
- expand all contractions
- remove all punctuation
- remove all *noise words*

We start with

```
> inspect(ex_corpus)
<<VCorpus (documents: 1, metadata (corpus/indexed): 0/0)>>

[[1]]
<<PlainTextDocument (metadata: 7)>>
This is 1 simple sentence, just to play with, then we'll play with 4,
    and that will be more difficult
```

Characters and Strings, in R

The first step might be to fix contractions

```
> fix_contractions <- function(doc) {
+   doc <- gsub("won't", "will not", doc)
+   doc <- gsub("n't", " not", doc)
+   doc <- gsub("'ll", " will", doc)
+   doc <- gsub("'re", " are", doc)
+   doc <- gsub("'ve", " have", doc)
+   doc <- gsub("'m", " am", doc)
+   doc <- gsub("'s", "", doc)
+   return(doc)
+ }
> ex_corpus <- tm_map(ex_corpus, fix_contractions)
> inspect(ex_corpus)
[[1]]
[1] This is 1 simple sentence, just to play with, then we will play
    with 4, and that will be more difficult
```

Characters and Strings, in R

Then we can remove numbers

```
> ex_corpus <- tm_map(ex_corpus, removeNumbers)
> inspect(ex_corpus)
[[1]]
[1] This is  simple sentence, just to play with, then we will play
    with , and that will be more difficult
```

as well as punctuation

```
> gsub("[[:punct:]]", "", ex_sentence)
[1] "This is 1 simple sentence just to play with then well play with
    4 and that will be more difficult"
> ex_corpus <- tm_map(ex_corpus,gsub, pattern= "[[:punct:]]",
    replacement = " ")
> inspect(ex_corpus)
[[1]]
[1] This is  simple sentence  just to play with  then we will play
    with   and that will be more difficult
```

Characters and Strings, in R

Then, we usually remove stop words,

```
> stopwords("en")[sample(1:length(stopwords("en")),size=10)]
 [1] "can't"     "could"     "because"   "i've"      "there's"   "who's"
     "for"       "couldn't"  "we've"     "him"
> ex_corpus <- tm_map(ex_corpus, removeWords, words=stopwords("en"))
> inspect(ex_corpus)
[[1]]
[1] This   simple sentence  just  play      will play      will
    difficult
```

We should also convert all words to lower case

```
> ex_corpus <- tm_map(ex_corpus, tolower)
> inspect(ex_corpus)
[[1]]
[1] this   simple sentence  just  play      will play      will
    difficult
```

Characters and Strings, in R

And finally, we stem the text

```
> library(SnowballC)
> ex_corpus <- tm_map(ex_corpus, stemDocument)
> inspect(ex_corpus)
[[1]]
[1] this   simple sentence  just  play      will play      will
    difficult
```

Characters and Strings, in R

We now have a clean list of words, it is possible to create some WordCloud

```
> wordcloud(ex_corpus[[1]])
> wordcloud(words = ex_corpus[[1]], max.words =
    40, random.order=FALSE, scale = c(5, 0.5),
    colors=cols)
```

Dates, in R

```
> (some.dates <- as.Date(c("16/10/12","19/11/12"),
    format="%d/%m/%y"))
[1] "2012-10-16" "2012-11-19"
> (sequence.date <- seq(from=some.dates[1],to=some.
    dates[2],by=7))
[1] "2012-10-16" "2012-10-23" "2012-10-30" "2012-11-06
    " "2012-11-13"
> format(sequence.date,"%b")
[1] "oct" "oct" "oct" "nov" "nov"
> weekdays(some.dates)
[1] "Tuesday" "Monday"
> Sys.setlocale("LC_TIME", "fr_FR")
[1] "fr_FR"
> weekdays(some.dates)
[1] "Mardi" "Lundi"
```

Symbolic Expressions, in R

Consider a regression model, $Y_i = \beta_0 + \beta_1 X_{1,i} + \beta_2 X_{2,i} + \beta_3 X_{3,i} + \varepsilon_i$. The code to fit such a model is based on

```
> fit <- lm(formula = y ~ x1 + x2 + x3, data=df)
```

In a formula, + stands for inclusion (not for summation), and - for exclusion.

To illustrate the use of categorical variables, consider

```
> set.seed(123)
> df <- data.frame(Y=rnorm(50), X1=as.factor(sample(LETTERS[1:4],size
    =50, replace=TRUE)), X2=as.factor(sample(1:3,size=50,replace=TRUE)
    ))
> tail(df,3)
           Y X1 X2
48  -0.557   B  2
49   0.950   C  2
50  -0.498   A  3
```

Symbolic Expressions, in R

```
> reg <- lm(Y~X1+X2,data=df)
> model.matrix(reg)[47:50,]
   (Intercept) X1B X1C X1D X22 X23
47           1   0   0   0   1   0
48           1   0   0   0   1   0
49           1   0   0   0   0   0
50           1   0   1   0   1   0
```

```
> reg <- lm(Y~X1*X2,data=df)
> model.matrix(reg)[47:50,]
   (Intercept) X1B X1C X1D X22 X23 X1B:X22 X1C:X22 X1D:X22 X1B:X23
47           1   1   0   0   0   1       0       0       0       1
48           1   1   0   0   1   0       1       0       0       0
49           1   0   1   0   1   0       0       1       0       0
50           1   0   0   0   0   1       0       0       0       0
```

Functions, in R

```
> factorial
function (x)
gamma(x + 1)
<bytecode: 0x1708aa7c>
<environment: namespace:base>
```

```
> gamma
function (x) .Primitive("gamma")
```

```
> x<-rexp(6)
> sum(x)
[1] 5.553364
> .Primitive("sum")(x)
[1] 5.553364
> cppFunction('double sum_C(
    NumericVector x) {
+   int n = x.size();
+   double total = 0;
+   for(int i = 0; i < n; ++i) {
+     total += x[i];
+   }
+   return total;
+ }')
> sum_C(x)
[1] 5.553364
```

Functions, in R

```
> f <- function(x) x^2
> f
function(x) x^2
> formals(f)
$x
> body(f)
x^2
```

```
> x <- 10
> f <- function() x<-5
> f()
> x
[1] 10
```

```
> f <- function(){
+     x<-5
+     return(x)
+ }
> f()
[1] 5
> x
[1] 10
> f <- function(){
+     x<<-5
+     return(x)
+ }
> f()
[1] 5
> x
[1] 5
```

Functions, in R

```
> x=function(y) y/2
> x
function(y) y/2
> x <- 5
> x(x)
Error: could not find function "
    x"

> x=function(y) y/2
> y=function(){
+    x <- 10
+    x(x)
+ }
> y()
[1] 5
```

```
> names_list <- function(...){
+    names(list(...))
+ }
> names_list(a=5,b=7)
[1] "a" "b"
```

Replacement functions act like they modify their arguments in place

```
> 'second<-' <- function(x,
    value) {
+    x[2] <- value
+    return(x)
+ }
> x <- 1:8
> second(x) <- 5
> x
 [1] 1 5 3 4 5 6 7 8
```

Functions, in R

```
> f <- function(x,m=0,s=1){
+ H<-function(t) 1-pnorm(t,m,s)
+ integral<-integrate(H,lower=x,upper=Inf)$value
+ res<-H(x)/integral
+ return(res)
+ }
> f(0:1)
[1] 1.2533141 0.3976897
Warning :
In if (is.finite(lower)) { :
the condition has length > 1 and only the first
    element will be used
> Vectorize(f)(0:1)
[1] 1.253314 1.904271
> sapply(0:1,"f")
[1] 1.253314 1.904271
```

Functions, in R

```
> fibonacci <- function(n){
+   if(n<2) return(1)
+   fibonacci(n-2)+fibonacci(n-1)
+ }
> fibonacci(20)
[1] 10946
> system.time(fibonacci(30))
   user  system elapsed
  3.687   0.000   3.719
```

Functions, in R

It is possible to use Memoisation : all previous inputs are stored... tradeoff speed and memory

```
> library(memoise)
> fibonacci <- memoise(function(n){
+   if(n<2) return(1)
+   fibonacci(n-2)+fibonacci(n-1)
+ })
> system.time(fibonacci(30))
   user  system elapsed
  0.004   0.000   0.004
```

Functions, in R

```
> binorm <- function(x1,x2,r=0){
+ exp(-(x1^2+x2^2-2*r*x1*x2)/(2*(1-r^2)))/(2*pi*sqrt(1-r^2))
+ }
```

```
> u <- seq(-2,2)
> binorm(u,u)
[1] 0.00291 0.05854 0.15915 0.05854 0.00291
> outer(u,u,binorm)
          [,1]    [,2]    [,3]    [,4]    [,5]
[1,] 0.00291 0.0130 0.0215 0.0130 0.00291
[2,] 0.01306 0.0585 0.0965 0.0585 0.01306
[3,] 0.02153 0.0965 0.1591 0.0965 0.02153
[4,] 0.01306 0.0585 0.0965 0.0585 0.01306
[5,] 0.00291 0.0130 0.0215 0.0130 0.00291
```

```
> (uv<-expand.grid(u,u))
> matrix(binorm(uv$Var1,uv$Var2),5,5)
```

```
> "%pm%" <- function(x,s) x + c(qnorm(.05),qnorm(.95))*s
> 100 %pm% 10
[1]  83.55146 116.44854
```

```
> f(0:1)
[1] 1.2533141 0.3976897
Warning :
In if (is.finite(lower)) { :
the condition has length > 1 and only the first element will be used
> Vectorize(f)(0:1)
[1] 1.253314 1.904271
> sapply(0:1,"f")
[1] 1.253314 1.904271
```

Functions, in R

```
> f<-function(x) dlnorm(x)
> integrate(f,0,Inf)
1 with absolute error < 2.5e-07
> integrate(f,0,1e5)
1.819813e-05 with absolute error < 3.6e-05
> integrate(f,0,1e3)$value+integrate(f,1e3,1e5)$value
[1] 1
```

Functions, in R

```
> set.seed(1)
> u <- runif(1)
> if(u>.5) {("greater than 50%")} else {("smaller than 50%")}
[1] "smaller than 50%"
> ifelse(u>.5,("greater than 50%"),("smaller than 50%"))
[1] "smaller than 50%"
> u
[1] 0.2655087

> v_x <- runif(1e5)
> sqrt_x <- NULL
> system.time(for(x in v_x) sqrt_x <- c(sqrt_x,sqrt(x)))
       user      system     elapsed
     23.774       0.224      24.071
```

Functions, in R

```
> sqrt_x <- numeric(length(v_x))
> system.time(for(x in seq_along(v_x)) sqrt_x[i] <- sqrt(x[i]))
   user  system elapsed
  1.320   0.000   1.328
>
> system.time(Vectorize(sqrt)(v_x))
   user  system elapsed
  0.008   0.000   0.009
>
> sqrt_x <- sapply(v_x,sqrt)
> system.time(unlist(lapply(v_x,sqrt)))
   user  system elapsed
  0.300   0.000   0.299
```

Functions, in R

```
> library(parallel)
> (allcores <- detectCores(all.tests=TRUE))
[1] 4
> system.time(unlist(mclapply(v_x,sqrt,mc.cores=4)))
   user  system elapsed
  0.396   0.224   0.362
```

Functions, in R

Write a function to generate random numbers drawn from a compound Poisson, $X = Y_1 + \cdots + Y_N$ with $N \sim \mathcal{P}(\lambda)$ and Y_i i.i.d. $\mathcal{E}(\alpha)$.

```
> rN.Poisson <- function(n) rpois(n,5)
> rX.Exponential <- function(n) rexp(n,2)
```

```
> rcpd1 <- function(n,rN=rN.Poisson,rX=rX.Exponential){
    V <- rep(0,n)
    for(i in 1:n){
      N <- rN(1)
      if(N>0){V[i] <- sum(rX(N))}
    }
    return(V)}
```

Functions, in R

```
> rcpd1 <- function(n,rN=rN.Poisson,rX=rX.Exponential){
    V <- rep(0,n)
    for (i in 1:n) V[i] <- sum(rX(rN(1)))
    return(V)}
```

```
> rcpd2 <- function(n,rN=rN.Poisson,rX=rX.Exponential){
  N <- rN(n)
   X <- rX(sum(N))
   I <- factor(rep(1:n,N),levels=1:n)
   return(as.numeric(xtabs(X ~ I)))}
```

Functions, in R

```
> rcpd3 <- function(n,rN=rN.Poisson,rX=rX.Exponential){
  N <- rN(n)
  X <- rX(sum(N))
  I <- factor(rep(1:n,N),levels=1:n)
  V <- tapply(X,I,sum)
  V[is.na(V)] <- 0
  return(as.numeric(V))}
```

Functions, in R

```
> rcpd4 <- function(n,rN=rN.Poisson,rX=rX.Exponential){
   return(sapply(rN(n), function(x) sum(rX(x))))}
```

```
> rcpd5 <- function(n,rN=rN.Poisson,rX=rX.Exponential){
   return(sapply(Vectorize(rX)(rN(n)),sum))}
```

```
> rcpd6 <- function(n,rN=rN.Poisson,rX=rX.Exponential){
  return(unlist(lapply(lapply(t(rN(n)),rX),sum)))}
```

Functions, in R

```
> n <- 100
> library(microbenchmark)
> options(digits=1)
> microbenchmark(rcpd1(n),rcpd2(n),rcpd3(n),rcpd4(n),rcpd5(n),rcpd6(n
    ))
Unit: microseconds
      expr  min   lq mean median   uq  max neval  cld
 rcpd1(n)  756  794  875    829  884 1624   100   c
 rcpd2(n) 1292 1394 1656   1474 1578 6799   100    d
 rcpd3(n)  629  677  741    707  756 2079   100  bc
 rcpd4(n)  482  515  595    540  561 5095   100 ab
 rcpd5(n)  613  663  864    694  744 9943   100   c
 rcpd6(n)  426  453  496    469  492 1020   100 a
```

Functions, in R

```
> n <- 50
> X <- matrix(rnorm(m * n, mean = 10, sd = 3), nrow = m)
> group <- rep(c("A","B"), each = n / 2)
>
> system.time(for(i in 1:m) t.test(X[i, ] ~ group)$stat)
utilisateur      syst me          coul
      1.028        0.000       1.030
> system.time(for(i in 1:m) t.test(X[i, group == "A"], X[i, group ==
    "B"])$stat)
utilisateur      syst me          coul
      0.224        0.000       0.222
```

Functions, in R

```
> f <- function(x) log(x)
> f("x")
Error in log(x) : non-numeric argument to mathematical function
> try(f("x"))
Error in log(x) : non-numeric argument to mathematical function
> inherits(try(f("x"), silent=TRUE), "try-error")
[1] TRUE
> x=2:4
> a=0
> try(a<-f("x"), silent=TRUE)
> a
[1] 0
> try(a<-f(x), silent=TRUE)
> a
[1] 0.6931472 1.0986123 1.3862944
```

Functions, in R

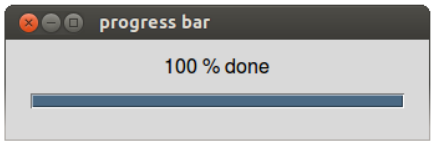
```
> power <- function(exponent) {
+   function(x) {
+     x ^ exponent
+   }
+ }
> square <- power(2)
> square(4)
[1] 16
> cube <- power(3)
> cube(4)
[1] 64
> cube
function(x) {
    x ^ exponent
  }
<environment: 0x9c810a0>
```

```
> x=1:10
> g=function(f) f(x)
> g(mean)
[1] 5.5
```

Progress Bar, in R

```
> v_x <- runif(2e4)
> sqrt_x <- NULL
> pb <- txtProgressBar(min = 0,
    max = total, style = 3)
|   0%
> for(i in seq_along(v_x)){
+   sqrt_x <- c(sqrt_x,sqrt(x[i
    ]))
+   if(i %% 1e3==0)
    setTxtProgressBar(pb, i%/%1
    e3)
+ }
|==========================| 100%
```

```
> library(tcltk)
> total <- 20
> pb <- tkProgressBar(title = "
    progress bar", min = 0,max =
     total, width = 300)
> for(i in seq_along(v_x)){
+   sqrt_x <- c(sqrt_x,sqrt(x[i])
    )
+   if(i %% 1e3==0)
    setTkProgressBar(pb, i%/%1e3
    , label=paste( round(i%/%1e3
    /total*100, 0),"% done"))
+ }
```



Data Frames, in R

```
> df <- data.frame(x=1:3,y=letters[1:3])
> str(df)
'data.frame': 3 obs. of  2 variables:
 $ x: int  1 2 3
 $ y: Factor w/ 3 levels "a","b","c": 1 2 3
> class(df)
[1] "data.frame"
```

```
> cbind(df,z=9:7)
  x y z
1 1 a 9
2 2 b 8
3 3 c 7
```

```
> df$z <- 5:3
> df
  x y z
1 1 a 5
2 2 b 4
3 3 c 3
```

Data Frames, in R

```
> cbind(df,z=9:7)
  x y z z
1 1 a 5 9
2 2 b 4 8
3 3 c 3 7
> df$z<-5:3
> df
  x y z
1 1 a 5
2 2 b 4
3 3 c 3
```

```
> df <- data.frame(x=1:3,y=
    letters[1:3],xy=19:17)
> df[1]
  x
1 1
2 2
3 3
> df[,1,drop=FALSE]
  x
1 1
2 2
3 3
```

Data Frames, in R

```
> df[,1,drop=TRUE]
[1] 1 2 3
> df[[1]]
[1] 1 2 3
> df[[1]]
[1] 1 2 3
> df$x
[1] 1 2 3
> df[,"x"]
[1] 1 2 3
> df[["x"]]
[1] 1 2 3
> df[["x",exact=FALSE]]
[1] 1 2 3
```

```
> set.seed(1)
> df[sample(nrow(df)),]
  x y xy
1 1 a 19
3 3 c 17
2 2 b 18
> set.seed(1)
> df[sample(nrow(df),nrow(df)*2,
    replace=TRUE),]
    x y xy
1   1 a 19
2   2 b 18
2.1 2 b 18
3   3 c 17
1.1 1 a 19
3.1 3 c 17
```

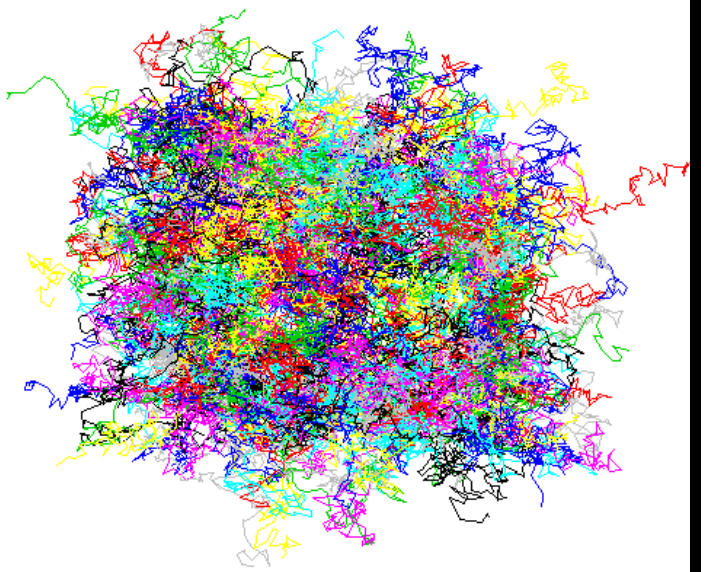
Data Frames, in R

```
> rm(list=ls())
> library(RCurl)
> dropbox_ldf <- "https://dl.dropboxusercontent.com/s/l7rolinwojn37e2
    /ldf_json_2.RData"
> dropbox_df  <- "https://dl.dropboxusercontent.com/s/wzr19v9pyl7ah0j
    /df_json_2.RData"
> dropbox_dt  <- "https://dl.dropboxusercontent.com/s/nlwi1mmsxr5f23k
    /dt_json_2.RData"
> source_https <- function(loc,...){
+   curl=getCurlHandle()
+   curlSetOpt(cookiejar="cookies.txt",useragent="Mozilla/5.0",
     followlocation=TRUE)
+   tmp <- getURLContent(loc,  .opts=list(ssl.verifypeer=FALSE),curl=
     curl)
+   fch <- source(tmp)
+   fch
+ }
```

Data Frames, in R

```
> source_https(dropbox_ldf)
>  load("df_json_2.RData")
> tail(df)
```

	Pers_Id	Traj_Id	lat	lon
159996158	10000	2000091	3.860666	−2.6781690
159996159	10000	2000091	3.983418	−2.2454256
159996160	10000	2000091	3.929773	−2.0908522
159996161	10000	2000091	3.967067	−1.8922986
159996162	10000	2000091	3.881188	−2.1948032
159996163	10000	2000091	2.989197	0.0869032



Data Frames, in R

```
> n <- nrow(df)
> system.time(df$first<-c(1,df$
    Traj_Id[2:n]!=
df$Traj_Id[1:(n-1)]))
   user     system    elapsed
   3.12       1.23       4.37
> system.time(df$last<-c(df$Traj
    _Id[2:n]!=
df$Traj_Id[1:(n-1)],1))
   user    system     elapsed
   2.90      6.23       31.58
> object.size(df)
6399847720 bytes
```

```
> lat_0=0
> lon_0=0
> system.time(df$test <- (((lat_
    0-df$lat)2+(lon_0-df$lon)2)
    <=1)&(df$last==1))
Error: cannot allocate vector of
     size 1.2 Gb
```

```
> df$first <- NULL
> df$last <- NULL
> object.size(df)
3839908904 bytes
```

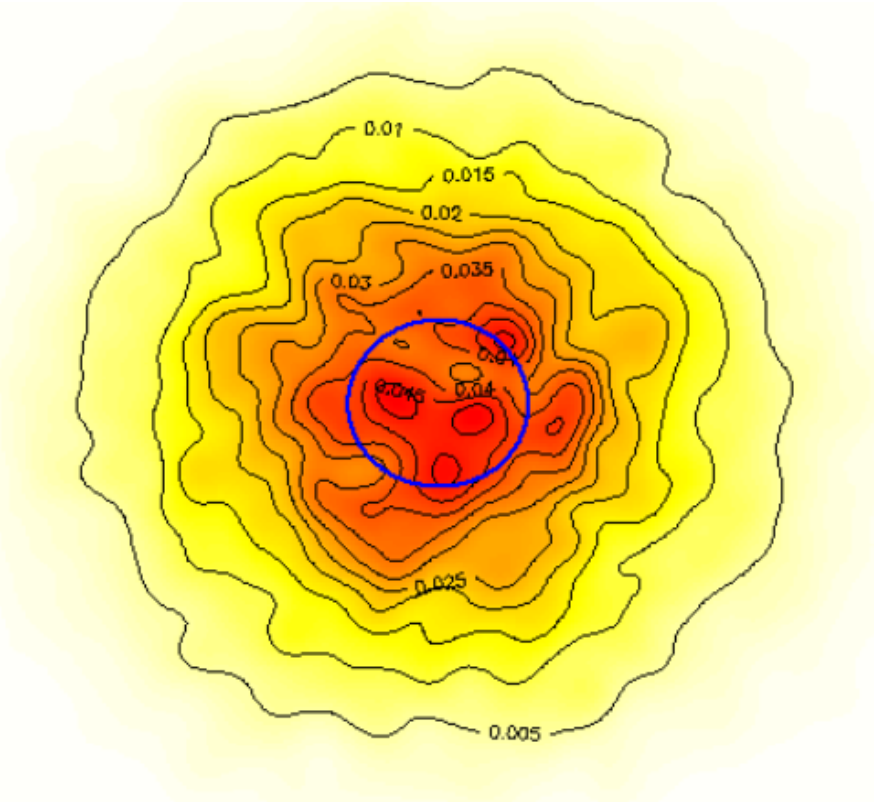
Data Frames, in R

```
> system.time(df$test <- ((
(lat_0-df$lat)2+(lon_0-df$lon)2)
    <=1)
&(c(df$Traj_Id[2:n]!=df$Traj_Id
    [1:(n-1)],1)==1))
   user      system     elapsed
   9.39       11.10       41.17
> system.time(list_Traj <-
    unique(df$Traj_Id[df$test==
    TRUE]))
   user      system     elapsed
   0.72        0.25        0.98
```

```
> system.time(base <- df[(df$
    Traj_Id %in% list_Traj)& (c
    (1,df$Traj_Id[2:n]!=df$Traj_
    Id[1:(n-1)])==1),c("lat","
    lon")])
   user      system     elapsed
   11.7         2.7        14.4
> head(base)
                lat       lon
556     -0.9597891 2.469243
2866    -0.9597891 2.469243
4321    -0.9597891 2.469243
5677    -0.9597891 2.469243
9403    -0.9597891 2.469243
10432   -0.9597891 2.469243
> nrow(base0
[1]  63453
```

Data Frames, in R

```
> X <- base[,c("lon","lat")]
> library(KernSmooth)
> kde2d <- bkde2D(X, bandwidth=c(bw.ucv(
    X[,1]),bw.ucv(X[,2])),gridsize = c
    (251L, 251L))
> image(x=kde2d$x1, y=kde2d$x2,z=kde2d$
    fhat,col=
rev(heat.colors(100)))
> contour(x=kde2d$x1, y=kde2d$x2,z=kde2d
    $fhat, add=TRUE)
```



Databases, in R

Consider the gapminderDataFiveYear.txt dataset, inspired from stat545-ubc

```
> gdf <- read.delim("gapminderDataFiveYear.txt")
> head(gdf,4)
      country year      pop continent lifeExp gdpPercap
1 Afghanistan 1952  8425333      Asia  28.801  779.4453
2 Afghanistan 1957  9240934      Asia  30.332  820.8530
3 Afghanistan 1962 10267083      Asia  31.997  853.1007
4 Afghanistan 1967 11537966      Asia  34.020  836.1971
> str(gdf)
'data.frame': 1704 obs. of  6 variables:
 $ country  : Factor w/ 142 levels "Afghanistan",..: 1 1 1 1 1 1 ...
 $ year     : int  1952 1957 1962 1967 1972 1977 1982 1987 1992 ...
 $ pop      : num  8425333 9240934 10267083 11537966 13079460 ...
 $ continent: Factor w/ 5 levels "Africa","Americas",..: 3 3 3 3 ...
 $ lifeExp  : num  28.8 30.3 32 34 36.1 ...
 $ gdpPercap: num  779 821 853 836 740 ...
```

Databases, in R

One can consider `tbl_df()` to get an improved data frame (called local dataframe)

```
> gtbl <- tbl_df(gdf)
> gtbl
Source: local data frame [1,704 x 6]

       country year      pop continent lifeExp gdpPercap
1  Afghanistan 1952  8425333      Asia  28.801  779.4453
2  Afghanistan 1957  9240934      Asia  30.332  820.8530
3  Afghanistan 1962 10267083      Asia  31.997  853.1007
4  Afghanistan 1967 11537966      Asia  34.020  836.1971
5  Afghanistan 1972 13079460      Asia  36.088  739.9811
6  Afghanistan 1977 14880372      Asia  38.438  786.1134
..         ...  ...      ...       ...     ...       ...
```

Databases, in R

For instance, to reproduce

```
> subset(gdf, lifeExp < 30)
      country year      pop continent lifeExp gdpPercap
1 Afghanistan 1952  8425333      Asia  28.801  779.4453
1293   Rwanda 1992  7290203    Africa  23.599  737.0686
```

use

```
> filter(gtbl, lifeExp < 30)
Source: local data frame [2 x 6]

      country year      pop continent lifeExp gdpPercap
1 Afghanistan 1952  8425333      Asia  28.801  779.4453
2      Rwanda 1992  7290203    Africa  23.599  737.0686
```

Databases, in R

The %>% operator can be used to generate (conveniently) datasets

```
> gtbl %>%
+    filter(country == "Italy") %>%
+    select(year, lifeExp)
Source: local data frame [12 x 2]

   year lifeExp
1  1952  65.940
2  1957  67.810
3  1962  69.240

11 2002  80.240
12 2007  80.546
```

which is (almost) the same as

```
> gdf[gdf$country == "Italy", c("year", "lifeExp")]
```

Local Data Frames, in R

```
> load("ldf_json_2.RData")
> system.time( ldepart <- ldf
    %>% group_by(Traj_Id) %>%
+ summarise(first_lat=head(lat
    ,1),
first_lon=head(lon,1)) )
   user      system      elapsed
  60.82        1.46        80.54
```

```
> system.time( larrive <- ldf
    %>% group_by(Traj_Id) %>%
+ summarise(last_lat=tail(lat,1)
    ,last_lon=tail(lon,1)) )
   user      system      elapsed
  60.81        0.31        62.15
> lat_0=0
> lon_0=0
> system.time( system.time(
    larrive <- mutate(larrive,
    dist=(last_lat-lat_0)2+(last
    _lon-lon_0)2) ))
   user      system      elapsed
   0.08        0.00        0.09
```

Local Data Frames, in R

```
> system.time( lfin <- filter(larrive,dist<=1) )
   user      system     elapsed
   0.05        0.00        0.04
> system.time( lbase <- left_join(lfin,ldepart) )
Joining by: "Traj_Id"
   user      system     elapsed
   0.53        0.05        0.66
> head(lbase)
Source: local data frame [63,453 x 6]

  Traj_Id    last_lat     last_lon        dist   first_lat  first_lon
1       8  0.41374639  0.491248980 0.412511638 -0.9597891   2.469243
2      36  0.58774352  0.003360806 0.345453735 -0.9597891   2.469243
3      54  0.34479069 -0.358867800 0.247666719 -0.9597891   2.469243
4      71 -0.04341135  0.014686416 0.002100236 -0.9597891   2.469243
5     117 -0.05103682 -0.070353141 0.007554322 -0.9597891   2.469243
```

```
17 6        130  -0.56196768  -0.715720445  0.828063425  -0.9597891   2.469243
```

Databases, in R

As in stat545-ubc.github.io consider the two following datasets (from superheroes.RData)

```
> load("superheroes.RData")
> superheroes
      name alignment  gender          publisher
1  Magneto       bad    male             Marvel
2    Storm      good  female             Marvel
3 Mystique       bad  female             Marvel
4   Batman      good    male                 DC
5    Joker       bad    male                 DC
6 Catwoman       bad  female                 DC
7  Hellboy      good    male Dark Horse Comics
```

for the superheroes,

Databases, in R

and for the publishers, consider

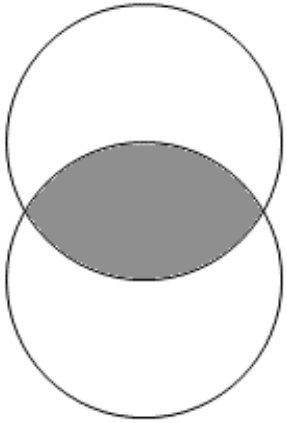
```
> publishers
  publisher yr_founded
1        DC       1934
2    Marvel       1939
3     Image       1992
```

There are many ways to merge those databases.

Databases, in R

Function `inner_join(x, y)` return all rows from `x` where there are matching values in `y`

```
> inner_join(superheroes, publishers)
Joining by: "publisher"
  publisher       name alignment  gender yr_founded
1    Marvel    Magneto       bad    male        1939
2    Marvel      Storm      good  female        1939
3    Marvel  Mystique       bad  female        1939
4        DC     Batman      good    male        1934
5        DC      Joker       bad    male        1934
6        DC  Catwoman       bad  female        1934
```

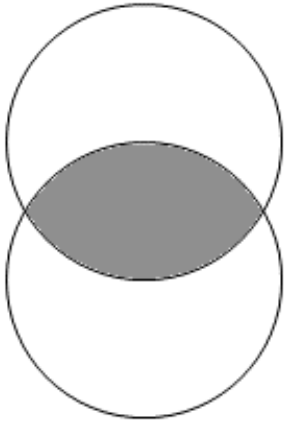


Databases, in R

Function `semi_join(x, y)` return all

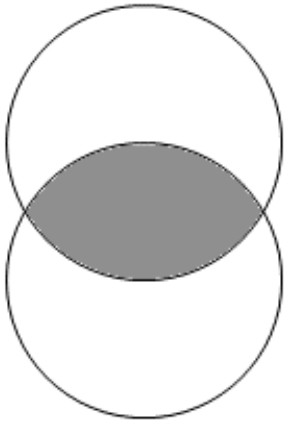
rows from `x` where there are matching values in `y`, but only columns from `x` are kept,

```
> semi_join(superheroes, publishers)
Joining by: "publisher"
      name alignment  gender publisher
1   Batman      good    male        DC
2    Joker       bad    male        DC
3 Catwoman       bad  female        DC
4  Magneto       bad    male    Marvel
5    Storm      good  female    Marvel
6 Mystique       bad  female    Marvel
```



Databases, in R

```
> inner_join(publishers, superheroes)
Joining by: "publisher"
  publisher yr_founded     name alignment  gender
1    Marvel       1939  Magneto       bad    male
2    Marvel       1939    Storm      good female
3    Marvel       1939 Mystique       bad female
4        DC       1934   Batman      good    male
5        DC       1934    Joker       bad    male
6        DC       1934 Catwoman       bad female
```

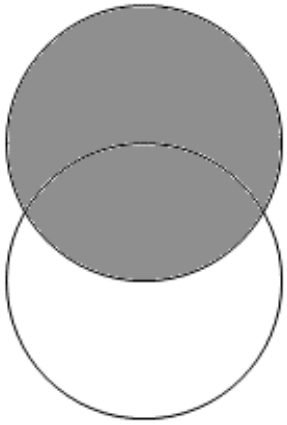


```
> semi_join(publishers, superheroes)0
Joining by: "publisher"
  publisher yr_founded
1    Marvel       1939
2        DC       1934
```

Databases, in R

Function `left_join(x, y)` return all rows from `x` and all columns from `x` and `y`

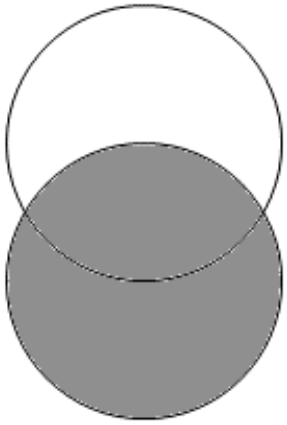
```
> left_join(superheroes, publishers)
Joining by: "publisher"
           publisher      name alignment gender yr_founded
1             Marvel   Magneto       bad   male        1939
2             Marvel     Storm      good female        1939
3             Marvel  Mystique       bad female        1939
4                 DC    Batman      good   male        1934
5                 DC     Joker       bad   male        1934
6                 DC  Catwoman       bad female        1934
7 Dark Horse Comics   Hellboy      good   male          NA
```



Databases, in R

There is no `right_join(x, y)` so we have to permutate `x` and `y`

```
> left_join(publishers, superheroes)
Joining by: "publisher"
  publisher yr_founded     name alignment gender
1        DC       1934   Batman      good   male
2        DC       1934    Joker       bad   male
3        DC       1934 Catwoman       bad female
4    Marvel       1939  Magneto       bad   male
5    Marvel       1939    Storm      good female
6    Marvel       1939 Mystique       bad female
7     Image       1992     <NA>      <NA>   <NA>
```



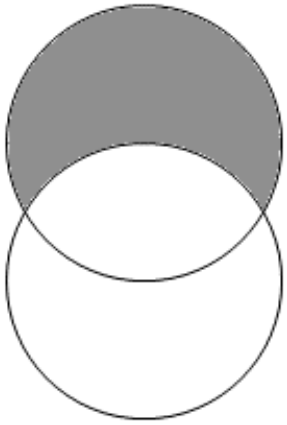
Databases, in R

One can use `anti_join(x, y)` for rows of `x` that have no match in `y`

```
> anti_join(superheroes, publishers)
Joining by: "publisher"
     name alignment gender          publisher
1 Hellboy      good   male Dark Horse Comics
```

and conversely

```
> anti_join(publishers, superheroes)
Joining by: "publisher"
  publisher yr_founded
1     Image        1992
```



Databases, in R

Note that it is possible to use a standard `merge()` function

```
> merge(superheroes, publishers, all = TRUE)
```

	publisher	name	alignment	gender	yr_founded
1	Dark Horse Comics	Hellboy	good	male	NA
2	DC	Batman	good	male	1934
3	DC	Joker	bad	male	1934
4	DC	Catwoman	bad	female	1934
5	Marvel	Magneto	bad	male	1939
6	Marvel	Storm	good	female	1939
7	Marvel	Mystique	bad	female	1939
8	Image	<NA>	<NA>	<NA>	1992

but it is much slower (in `dplyr` integrates R with C++)

There is also a `sql_join` for more advanced SQL requests).

Data Tables, in R

```
> system.time( load("dt_json_2.RData") )
      user      system     elapsed
     21.53        1.33       27.71
> system.time( setkey(dt,Traj_Id) )
      user      system     elapsed
      0.38        0.09        0.47
> system.time( depart <- dt[J(unique(Traj_Id)), mult = "first"])
      user      system     elapsed
      3.82        8.02      101.77
> system.time( arrivee <- dt[J(unique(Traj_Id)), mult = "last"] )
      user      system     elapsed
      2.62        0.52        3.39
```

Data Tables, in R

```
> lat_0=0
> lon_0=0
> system.time( arrivee[,dist:=(lat-lat_0)2+(lon-lon_0)2] )
      user      system     elapsed
      0.03        0.08        1.60
> system.time( fin <- subset(arrivee,dist <= 1) )
      user      system     elapsed
      0.04        0.00        0.78
> system.time( fin[,Pers_Id:=NULL] )
      user      system     elapsed
      0.00        0.00        0.07
> system.time( fin[,lat:=NULL] )
      user      system     elapsed
       0.0         0.0         0.2
> system.time( fin[,lon:=NULL] )
      user      system     elapsed
         0           0           0
```

Data Tables, in R

```
> system.time( base <- merge(fin,depart,all.x=TRUE) )
       user      system     elapsed
       0.02        0.00        0.17
> system.time( base <- merge(fin,depart,all.x=TRUE) )
       user      system     elapsed
       0.02        0.00        0.17
>
> head(base)
   Traj_Id       dist Pers_Id       lat      lon
1:       8 0.41251163       1 -0.9597891 2.469243
2:      36 0.34545373       1 -0.9597891 2.469243
3:      54 0.24766671       1 -0.9597891 2.469243
4:      71 0.00210023       1 -0.9597891 2.469243
5:     117 0.00755432       1 -0.9597891 2.469243
6:     130 0.82806342       1 -0.9597891 2.469243
```

Memory and Datasets, in R

Instead of loading the complete dataset in the RAM, it is also possible to load it by chunks. Consider e.g. the 'Death Master File' .info,

```
> cols <- c(1,9,20,4,15,15,1,2,2,4,2,2,4,2,5,5,7)
> noms_col <- c ("code","ssn","last_name","name_suffix","first_name",
    "middle_name","VorPCode","date_death_m","date_death_d","date_death
    _y","date_birth_m","date_birth_d","date_birth_y","state","zip_
    resid","zip_payment","blanks")
> library(LaF)
> temp <- "ssdm3"
> ssn <- laf_open_fwf( temp,column_widths = cols,column_types=rep("
    character",length(cols) ),column_names = noms_col,trim = TRUE)
> object.size(ssn)
3544 bytes
> go_through <- seq(1,nrow(ssn),by = 1e05 )
> if(go_through[ length(go_through)] != nrow( ssn)) go_through <- c(
    go_through,nrow( ssn))
```

Memory and Datasets, in R

```
> go_through <- cbind(go_through[-length(go_through)],c(go_through[-c
    (1,length(go_through)) ]-1,go_through [ length(go_through)]))
> go_through
             [,1]      [,2]
  [1,]          1    100000
  [2,]     100001    200000
  [3,]     200001    300000

[286,] 28500001 28600000
[287,] 28600001 28607398
>
> pb <- txtProgressBar(min = 0, max = nrow( go_through), style = 3)
> count_birthday <- function(s){
+   #print(s)
+   setTxtProgressBar(pb, s)
```

Memory and Datasets, in R

```
+    data <- ssn[ go_through[s,1]:go_through[s,2],c("date_death_m","
     date_death_d",
+                 "date_birth_m","date_birth_d")]
+    sum((data$date_death_m==data$date_birth_m) &
+          (data$date_death_d==data$date_birth_d) )
+ }
>
> system.time( data <- lapply( seq_len(nrow( go_through) ),count_
     birthday) )
   |=================================================| 100
%utilisateur      syst me          coul
       19.48         1.37        37.31
> sum( unlist(data)) /nrow(ssn)
[1] 0.001753847
```

Environments, in R

An environment is a collection of names, and each name points to an objected stored somewhere

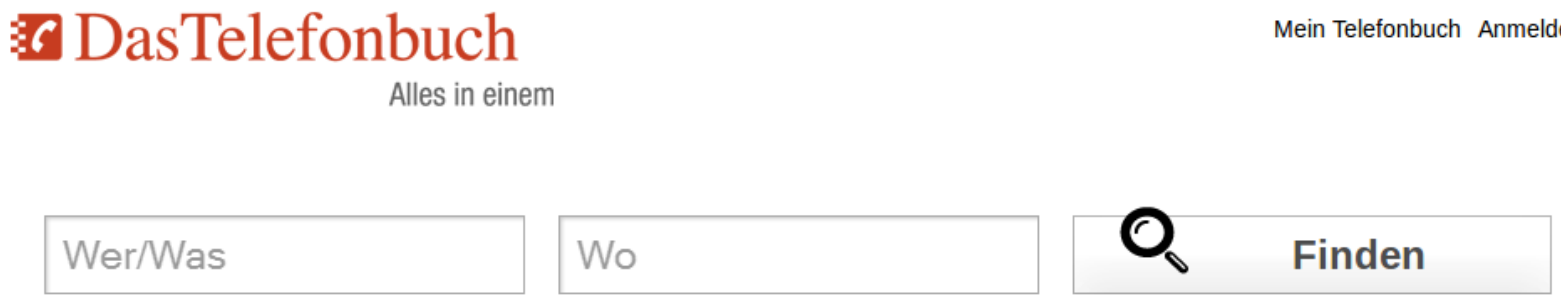
```
> a <- 1
> ls(globalenv())
[1] "a"
> environment(sd)
<environment: namespace:stats>
> find("pi")
[1] "package:base"
```

```
> e <- new.env()
> e$d <- 1
> e$f <- 1:5
> e$g <- 1:5
> ls(e)
[1] "d" "f" "g"
> str(e)
<environment: 0x8b14918>
```

Environments, in R

```
> identical(globalenv(),e)
[1] FALSE
> search()
 [1] ".GlobalEnv"                "package:memoise"
 [3] "package:microbenchmark"    "package:Rcpp"
 [5] "package:lubridate"         "package:pryr"
 [7] "package:parallel"          "package:sp"
 [9] "tools:rstudio"             "package:stats"
[11] "package:graphics"          "package:grDevices"
[13] "package:utils"             "package:datasets"
[15] "package:methods"           "Autoloads"
[17] "package:base"
```

Filling Forms & Web Scrapping



As in Munzert *et al.* (2014, http://eu.wiley.com) Consider here all people in Germany with the name Feuerstein,

```
> tb <- getForm("http://www.dastelefonbuch.de/",.params = c(kw = "
    Feuerstein",cmd = "search",ao1 = "1",reccount = "2000"))
```

Let us store that webpage on our computer

```
> write(tb, file = "phonebook_feuerstein.html")
> tb_parse <- htmlParse("phonebook_feuerstein.html", encoding = "UTF
    -8")
```

```
> xpath <- "//ul/li/a[contains(text(), 'Privat')]"
```

```
> num_results <- xpathSApply(tb_parse, xpath, xmlValue)
> num_results
[1] "\n                Privat (637)"
> num_results <- as.numeric(str_extract(num_results, "[[:digit:]]+"))
> num_results
[1] 637
```

```
> xpath <- "//div[@class='name']/a[@title]"
> surnames <- xpathSApply(tb_parse, xpath, xmlValue)
> surnames[1:3]
[1] "\n\t\t\t\tBertsch-Feuerstein Lilli"
    "\n\t\t\t\tBierig-Feuerstein Brigitte u. Feuerstein Norbert"
[3] "\n\t\t\t\tBlatt Karl u. Feuerstein-Blatt Ursula"
> xpath <- "//span[@itemprop='postal-code']"
> zipcodes <- xpathSApply(tb_parse, xpath, xmlValue)
> zipcodes[1:3]
[1] " 64625" " 68549" " 68526"
> xpath <- "//span[@itemprop='postal-code']/ancestor::div[@class='
    popupMenu']/preceding-sibling::div[@class='name']"
```

```
> names_vec <- xpathSApply(tb_parse, xpath, xmlValue)
> xpath <- "//div[@class='name']/following-sibling::div[@class='
    popupMenu']//span[@itemprop='postal-code']"
> zipcodes_vec <- xpathSApply(tb_parse, xpath, xmlValue)
> names_vec <- str_replace_all(names_vec, "(\\n|\\t|\\r| {2,})", "")
> zipcodes_vec <- as.numeric(zipcodes_vec)
```

```
> entries_df <- data.frame(plz = zipcodes_vec, name = names_vec)
> head(entries_df)
    plz                                                  name
1 64625                                  Bertsch-Feuerstein Lilli
2 68549 Bierig-Feuerstein Brigitte u. Feuerstein Norbert
3 68526                 Blatt Karl u. Feuerstein-Blatt Ursula
4 50733                                              Feuerstein
5 69207                                              Feuerstein
6 97769                                              Feuerstein
```

Now, we need a dataset that links zip codes (*Postleitzahlen*, PLZ) and geographic coordinates. We can use datasets from the OpenGeoDB project (see

http://opengeodb.org)

```
> download.file("http://fa-technik.adfc.de/code/opengeodb/PLZ.tab",
+ destfile = "geo_germany/plz_de.txt")
> plz_df <- read.delim("geo_germany/plz_de.txt", stringsAsFactors
+ = FALSE, encoding = "UTF-8")
> plz_df[1:3, ]
  X.loc_id  plz      lon      lat     Ort
1     5078 1067 13.72107 51.06003 Dresden
2     5079 1069 13.73891 51.03956 Dresden
3     5080 1097 13.74397 51.06675 Dresden
```

Now, if we merge the two

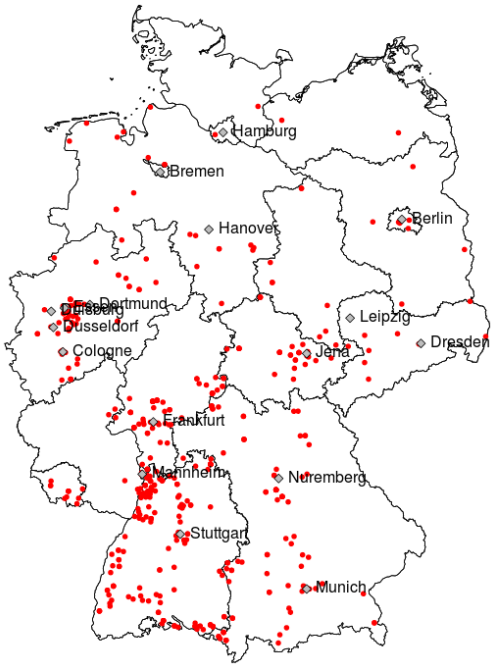
```
> places_geo <- merge(entries_df, plz_df, by = "plz", all.x = TRUE)
> places_geo[1:3, ]
    plz                 name X.loc_id      lon      lat         Ort
1 1159      Feuerstein Falk      5087 13.70069 51.04261     Dresden
2 1623    Feuerstein Regina      5122 13.29736 51.16516 Lommatzsch
3 2827 Feuerstein Wolfgang      5199 14.96443 51.13170      G rlitz
```

Now we simply need some shapefile (see slides on Spatial aspects),

```
> download.file("http://biogeo.ucdavis.edu/data/gadm2/shp/DEU_adm.zip",
+ destfile = "geo_germany/ger_shape.zip")
> unzip("geo_germany/ger_shape.zip", exdir = "geo_germany")
> projection <- CRS("+proj=longlat +ellps=WGS84 +datum=WGS84")
> map_germany <- readShapePoly(str_c(getwd(), "/geo_germany/DEU_adm0.shp"),
+ proj4string = projection)
> map_germany_laender <- readShapePoly(str_c(getwd(),"/geo_germany/DEU_adm1.shp"),
+ proj4string=projection)
> coords <- SpatialPoints(cbind(places_geo$lon, places_geo$lat))
> proj4string(coords) <- CRS("+proj=longlat +ellps=WGS84 +datum=WGS84")
> data("world.cities")
> cities_ger <- subset(world.cities,
+ country.etc == "Germany" &
```

```
14 + (world.cities$pop > 450000 |
15 + world.cities$name %in%
16 + c("Mannheim", "Jena")))
17 > coords_cities <- SpatialPoints(cbind(cities_ger$long,cities_ger$lat
       ))
```

```
1 > plot(map_germany)
2 > plot(map_germany_laender, add = TRUE)
3 > points(coords$coords.x1, coords$coords.x2, pch =
       20, col = "red")
4 > points(coords_cities, col = "black", , bg = "
     grey", pch = 23)
5 > text(cities_ger$long, cities_ger$lat, labels =
     cities_ger$name, pos = 4)
```



Similarly, consider Petersen, Gruber and Schultze