Statistics with R

A fast route to Data Science

Montserrat Guillen

Dept. Econometrics, Statistics and Applied Economics &
Riskcenter, UB

Pre-requisites

Install R (cran.r-project.org)

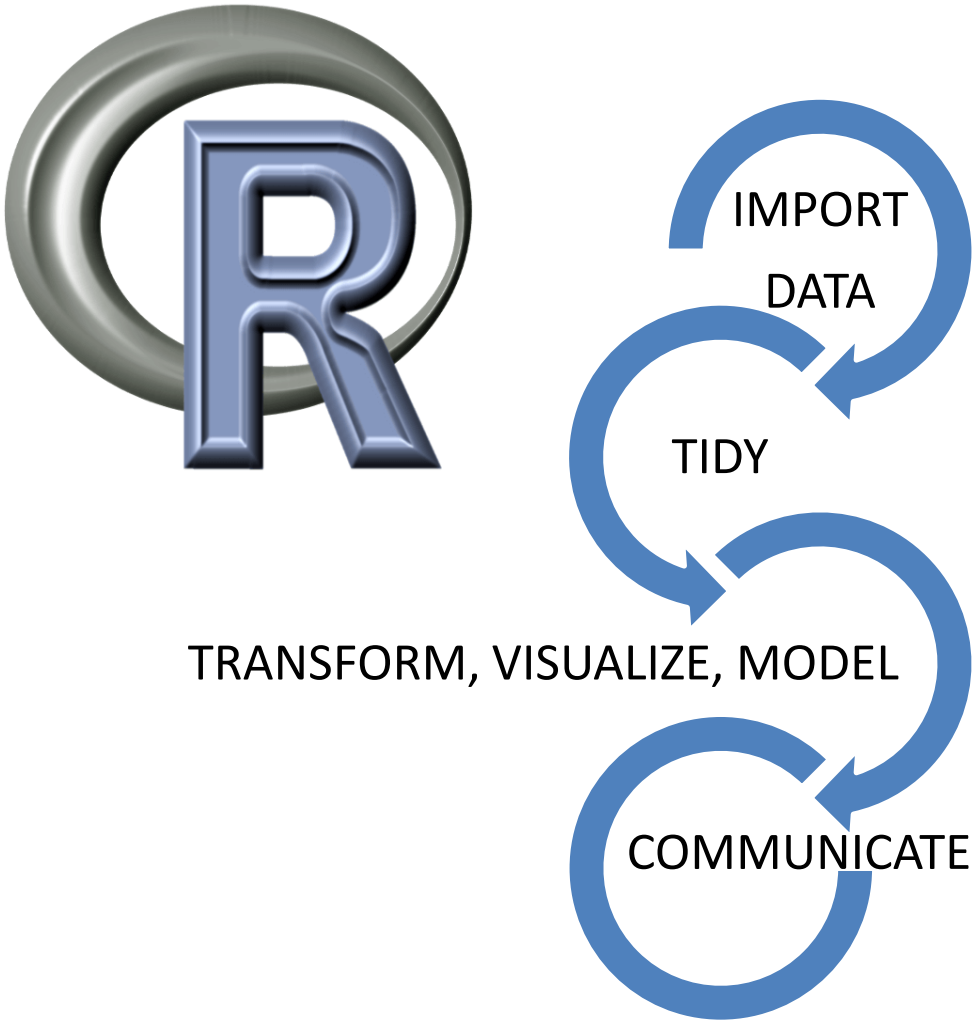
R-3.6.1 for Windows (32/64 bit)

Install R Studio (www.rstudio.com/products/rstudio/download/)

RStudio Desktop

Doc-03.pdf

What do we need?



What is R in the universe?

Our workplan today (not standard)

- **RStudio projects, working directory, scripts and packages**
- **Data structures:** vectors, matrices, data.frames, lists, objects
- **Data wrangling:** injestion+digestion
- **R programming:** if-the-else, loops. functions
- **Rmarkdown:** producing HTML, PDF, LateX, PPT

R users groups and communities

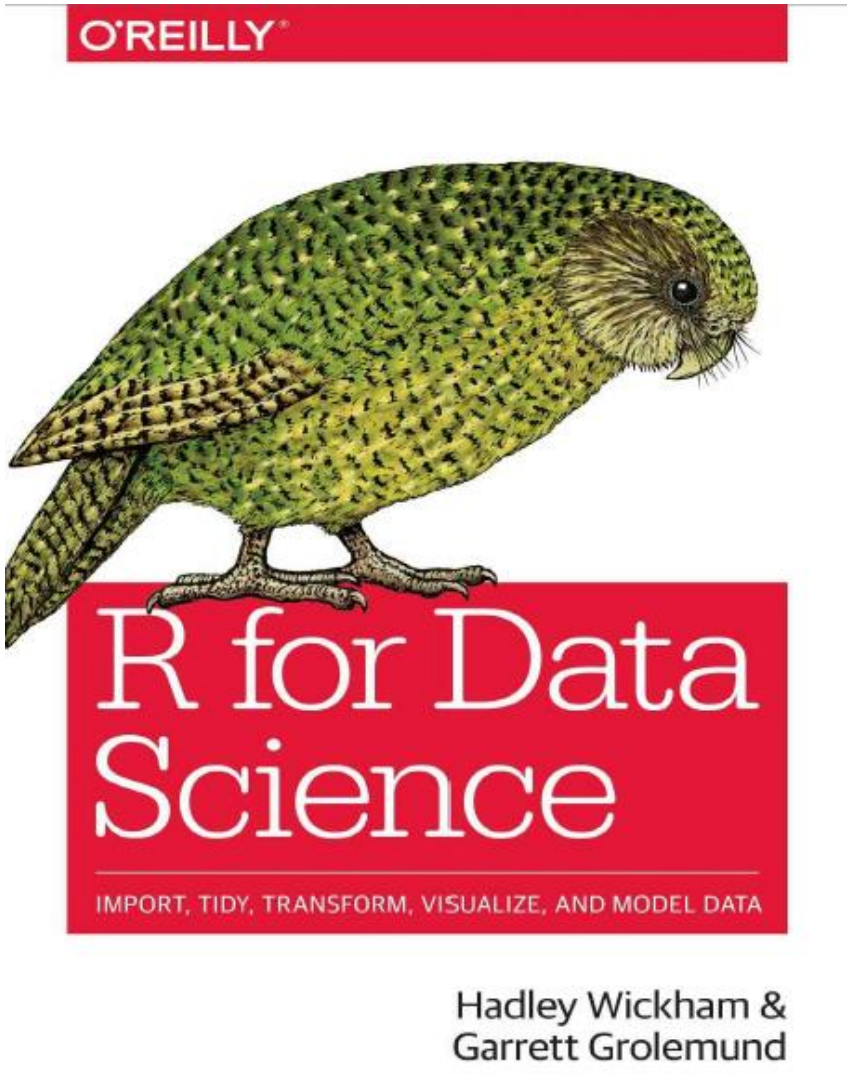
Nabble - This site has a funny name, but unlike other forums on this page it is specific to R. The user base is knowledgable & helpful. Questions here range **from R & programming to statistical methods & data mining**. As long as your question is specific, it should be answered quickly.

Stack Overflow - This is a popular forum focused on programming, with a tag for questions related to R. The community here is active & helpful. Keep in mind that this is a **place for programming questions**. If you have questions about the underlying math or methods, this isn't the place.

Cross Validated - A sister site to Stack Overflow, this forum deals with **statistics & statistical programming**. Any questions about programming R, statistical methods, data mining, big data, etc, are welcome here.

R-Help Mailing List - Before forums, there were mailing lists. This mailing list has been active since the late 90's and is still very active. **You can post a question if you subscribe to the list**. The archive is searchable & has a ton of great info.

Inspiration for a two-session journey to Statistics with R



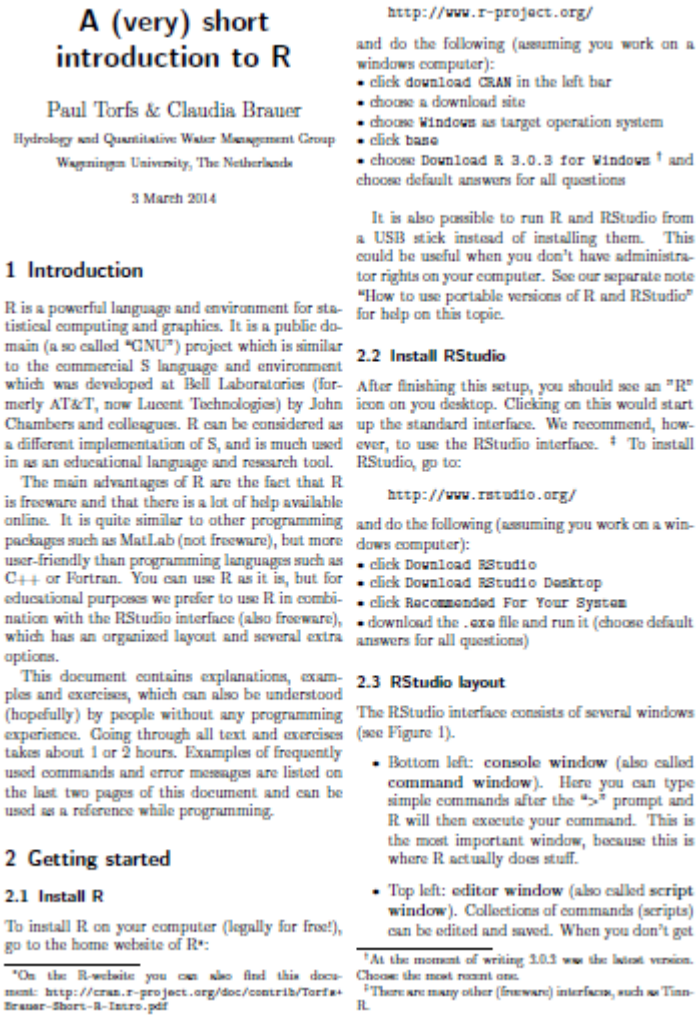
Online version of this book is available at **http://r4ds.had.co.nz**

The source of the book is available at **https://github.com/hadley/r4ds**

February, 2017

Doc-01.pdf

Starting point
Statistics with R



A (very) short introduction to R

Paul Torfs & Claudia Brauer

Hydrology and Quantitative Water Management Group
Wageningen University, The Netherlands

3 March 2014

1 Introduction

R is a powerful language and environment for statistical computing and graphics. It is a public domain (a so called "GNU") project which is similar to the commercial S language and environment which was developed at Bell Laboratories (formerly AT&T, now Lucent Technologies) by John Chambers and colleagues. R can be considered as a different implementation of S, and is much used in as an educational language and research tool.

The main advantages of R are the fact that R is freeware and that there is a lot of help available online. It is quite similar to other programming packages such as MatLab (not freeware), but more user-friendly than programming languages such as C++ or Fortran. You can use R as it is, but for educational purposes we prefer to use R in combination with the RStudio interface (also freeware), which has an organized layout and several extra options.

This document contains explanations, examples and exercises, which can also be understood (hopefully) by people without any programming experience. Going through all text and exercises takes about 1 or 2 hours. Examples of frequently used commands and error messages are listed on the last two pages of this document and can be used as a reference while programming.

2 Getting started

2.1 Install R

To install R on your computer (legally for free!), go to the home website of R*:

`http://www.r-project.org/`

and do the following (assuming you work on a windows computer):

- click `download CRAN` in the left bar
- choose a download site
- choose `Windows` as target operation system
- click `base`
- choose `Download R 3.0.3 for Windows` † and choose default answers for all questions

It is also possible to run R and RStudio from a USB stick instead of installing them. This could be useful when you don't have administrator rights on your computer. See our separate note "How to use portable versions of R and RStudio" for help on this topic.

2.2 Install RStudio

After finishing this setup, you should see an "R" icon on you desktop. Clicking on this would start up the standard interface. We recommend, however, to use the RStudio interface. ‡ To install RStudio, go to:

`http://www.rstudio.org/`

and do the following (assuming you work on a windows computer):

- click `Download RStudio`
- click `Download RStudio Desktop`
- click `Recommended For Your System`
- download the `.exe` file and run it (choose default answers for all questions)

2.3 RStudio layout

The RStudio interface consists of several windows (see Figure 1).

- Bottom left: **console window** (also called **command window**). Here you can type simple commands after the ">" prompt and R will then execute your command. This is the most important window, because this is where R actually does stuff.
- Top left: **editor window** (also called **script window**). Collections of commands (scripts) can be edited and saved. When you don't get

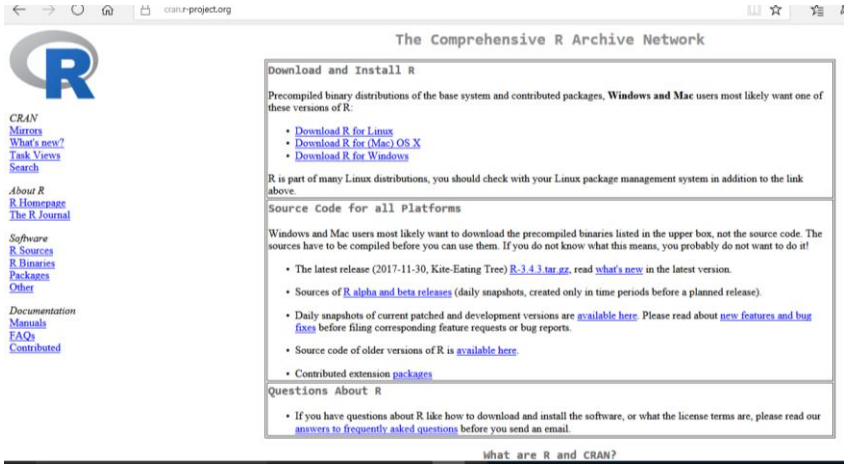
*On the R-website you can also find this document: `http://cran.r-project.org/doc/contrib/Torfs+Brauer-Short-R-Intro.pdf`

†At the moment of writing 3.0.3 was the latest version. Choose the most recent one.

‡There are many other (freeware) interfaces, such as Tinn-R.

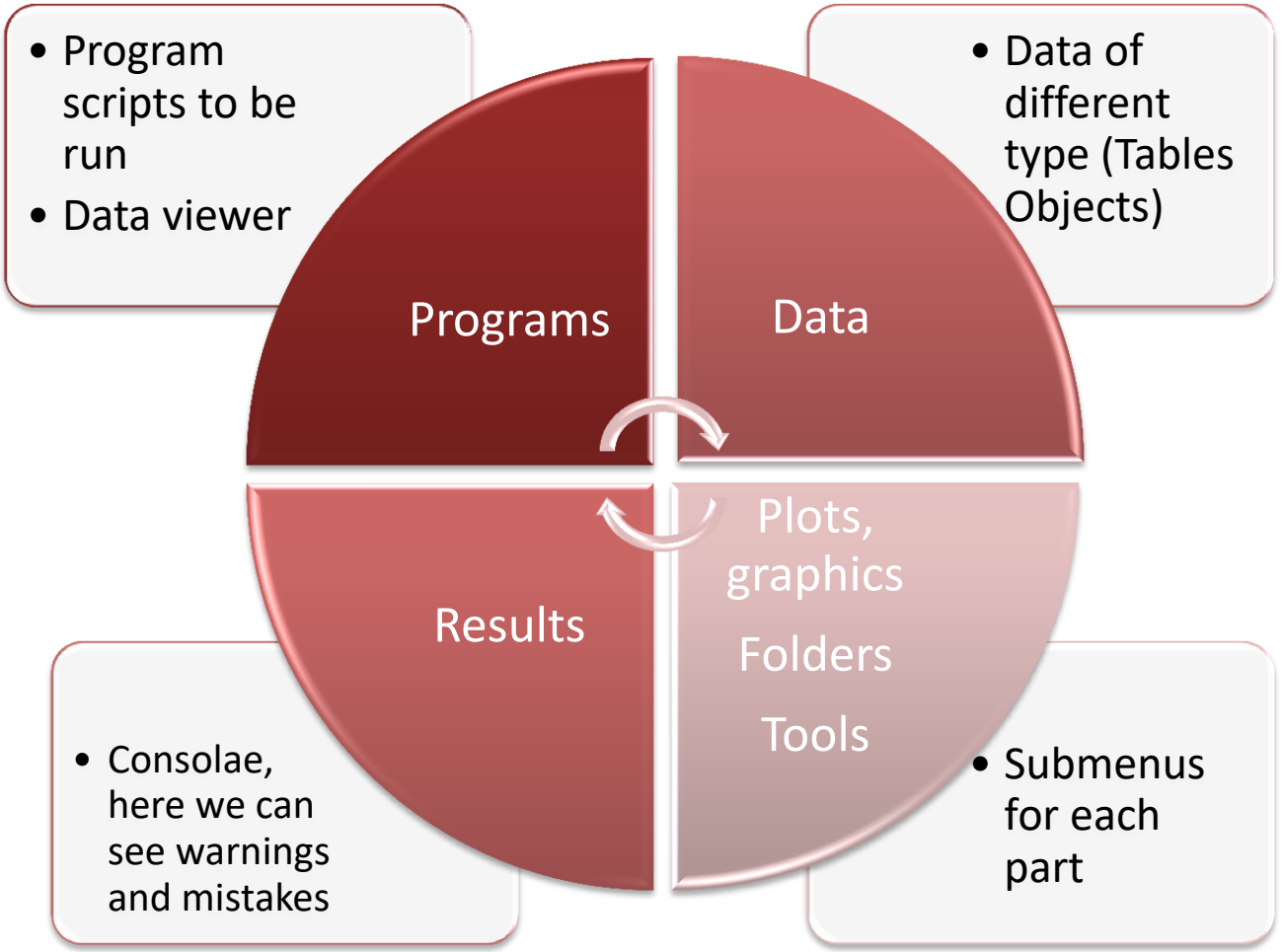
This paper can be downloaded from **https://cran.r-project.org/doc/contrib/Torfs+Brauer-Short-R-Intro.pdf**

CRAN is …. the place
Comprehensive R Archive Network

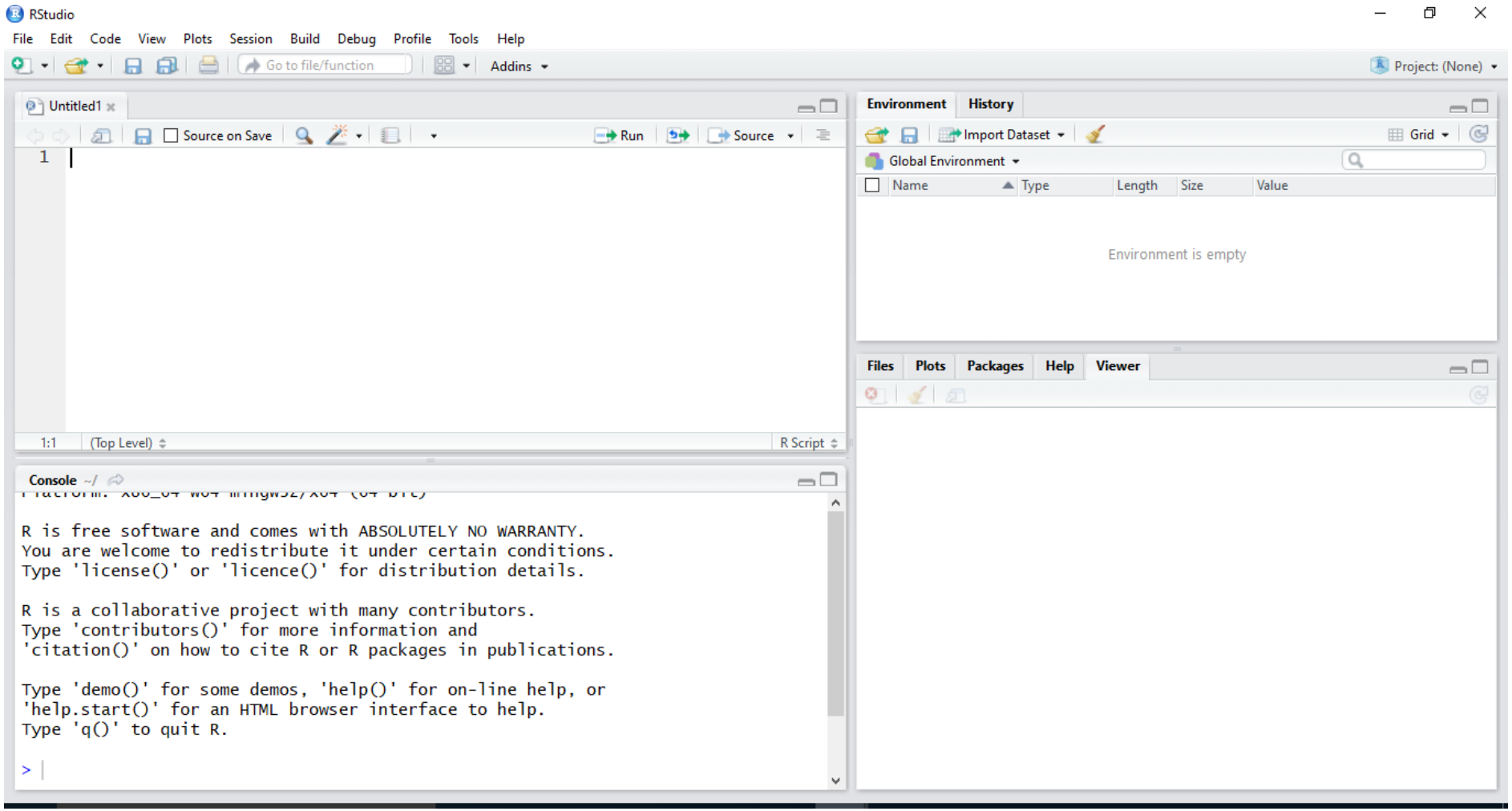


Doc-02.pdf

Global concepts in statistical softwares



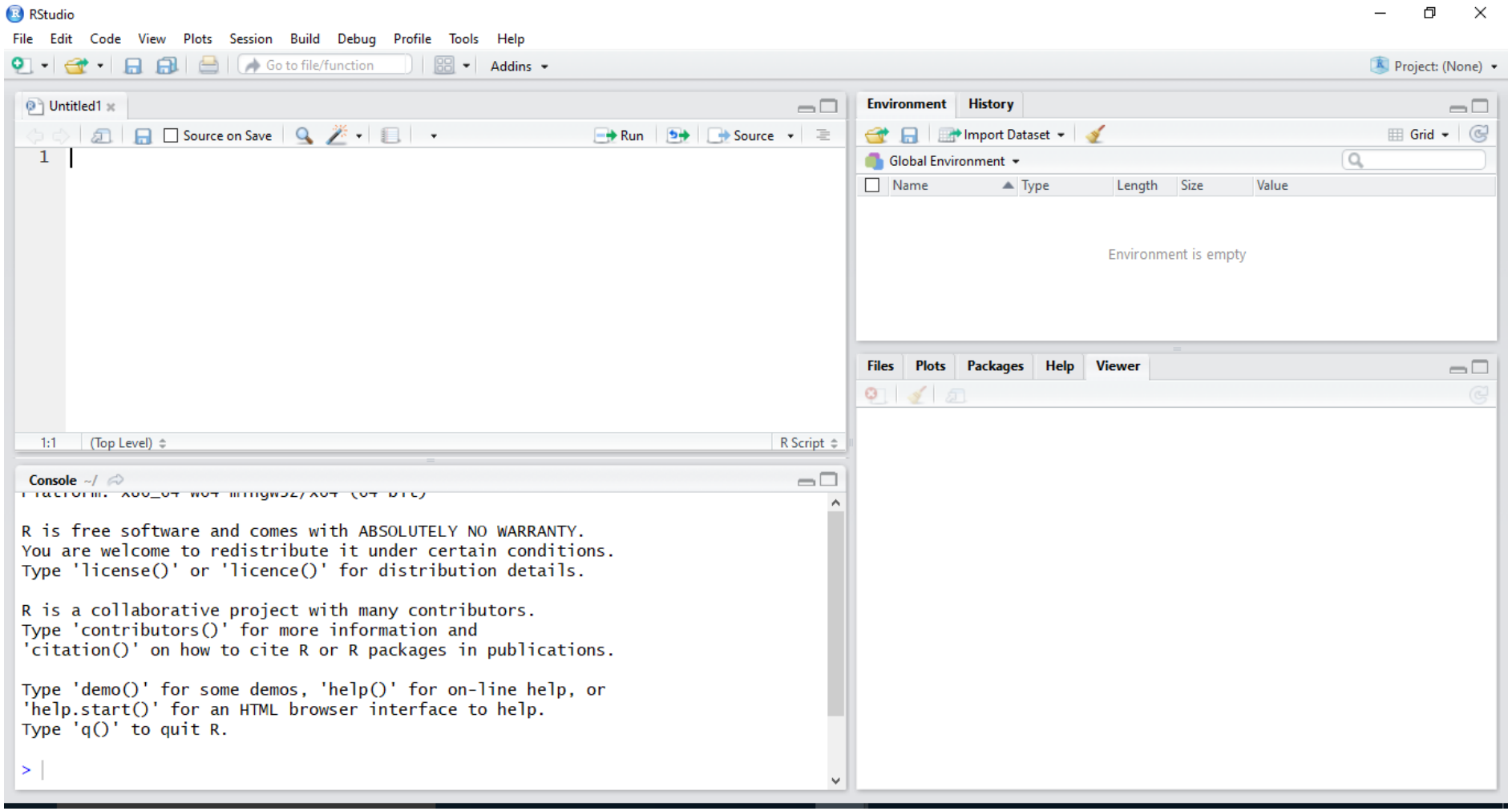
How does R Studio look like?



Our workplan today (not standard)

- **RStudio projects, working directory, scripts and packages**
- **Data structures:** vectors, matrices, data.frames, lists, objects
- **Data wrangling:** injestion+digestion
- **R programming:** if-the-else, loops. functions
- **Rmarkdown:** producing HTML, PDF, LateX, PPT

Hands on example: an Rstudio project



Super-fast start in R : launch RStudio

Create a new project File>New project
Choose New directory > Empty project
Create script file File > New file

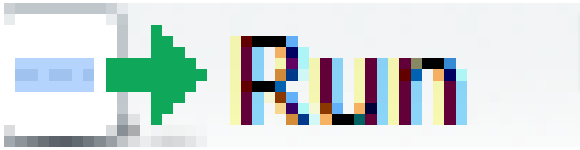
Copy paste from here:

```
Bank1 <- read.table('http://www.ub.edu/rfa/docs/DATA/bank.csv',
header=TRUE, sep=';')

View(Bank1)
names(Bank1)
```

Upper case/Lower case

Select and then
CTRL+Intro
Or click on RUN



Prog-01a.R

RStudio project

- An **Rstudo project** gives solid workflow that will serve in the future.
- Create an RStudio project for **each data analysis** project.
- Keep **data** files there; and load them later.
- Keep **scripts** there; edit them, and run them in bits or as a whole.
- Save your **outputs** (plots and cleaned data) there.
- Use ***relative paths***, not absolute paths.
- Everything you need is in **one place**, and cleanly separated from all the other projects that you are working on.

  To me this is an advantage (compactness) and a disadvantage (wrong for big data sets).

First: regression models

```
RegModel.1 <-lm(balance~age, data=Bank1)

summary(RegModel.1)

plot( Bank1$age, Bank1$balance)

Bank1$agesq=Bank1$age*Bank1$age
RegModel.2 <-lm(balance~age+agesq+loan, data=Bank1)
summary(RegModel.2)

summary(Bank1$balance)

Bank2<-subset(Bank1, balance>100)

RegModel.3 <-lm(balance~age+agesq+loan, data=Bank2)
summary(RegModel.3)

####  Exit save YES
```

Prog-01b.R

A case study: bank telemarketing

Decision Support Systems xxx (2014) xxx–xxx

Contents lists available at ScienceDirect

Decision Support Systems

journal homepage: www.elsevier.com/locate/dss

A data-driven approach to predict the success of bank telemarketing

Sérgio Moro [a,*], Paulo Cortez [b], Paulo Rita [a]

[a] ISCTE-IUL, Business Research Unit (BRU-IUL), Lisboa, Portugal
[b] ALGORITMI Research Centre, Univ. of Minho, 4800-058 Guimarães, Portugal

ARTICLE INFO

Article history:
Received 1 November 2013
Received in revised form 28 February 2014
Accepted 4 March 2014
Available online xxxx

Keywords:
Bank deposits
Telemarketing
Savings
Classification
Neural networks
Variable selection

ABSTRACT

We propose a data mining (DM) approach to predict the success of telemarketing calls for selling bank long-term deposits. A Portuguese retail bank was addressed, with data collected from 2008 to 2013, thus including the effects of the recent financial crisis. We analyzed a large set of 150 features related with bank client, product and social-economic attributes. A semi-automatic feature selection was explored in the modeling phase, performed with the data prior to July 2012 and that allowed to select a reduced set of 22 features. We also compared four DM models: logistic regression, decision trees (DTs), neural network (NN) and support vector machine. Using two metrics, area of the receiver operating characteristic curve (AUC) and area of the LIFT cumulative curve (ALIFT), the four models were tested on an evaluation set, using the most recent data (after July 2012) and a rolling window scheme. The NN presented the best results (AUC = 0.8 and ALIFT = 0.7), allowing to reach 79% of the subscribers by selecting the half better classified clients. Also, two knowledge extraction methods, a sensitivity analysis and a DT, were applied to the NN model and revealed several key attributes (e.g., Euribor rate, direction of the call and bank agent experience). Such knowledge extraction confirmed the obtained model as credible and valuable for telemarketing campaign managers.

http://dx.doi.org/10.1016/j.dss.2014.03.001 Doc-04.pdf

The data

Bank Marketing Data Set

Download: Data Folder, Data Set Description

Abstract: The data is related with direct marketing campaigns (phone calls) of a Portuguese banking institution. The classification goal is to predi

Data Set Characteristics:	Multivariate	**Number of Instances:**	45211	**Area:**	Business
Attribute Characteristics:	Real	**Number of Attributes:**	17	**Date Donated**	2012-02-14
Associated Tasks:	Classification	**Missing Values?**	N/A	**Number of Web Hits:**	475542

Source:

[Moro et al., 2014] S. Moro, P. Cortez and P. Rita. A Data-Driven Approach to Predict the Success of Bank Telemarketing. Decision Support Syste

https://archive.ics.uci.edu/ml/datasets/Bank+Marketing?package=regsel&version=0.2

What is an R package?

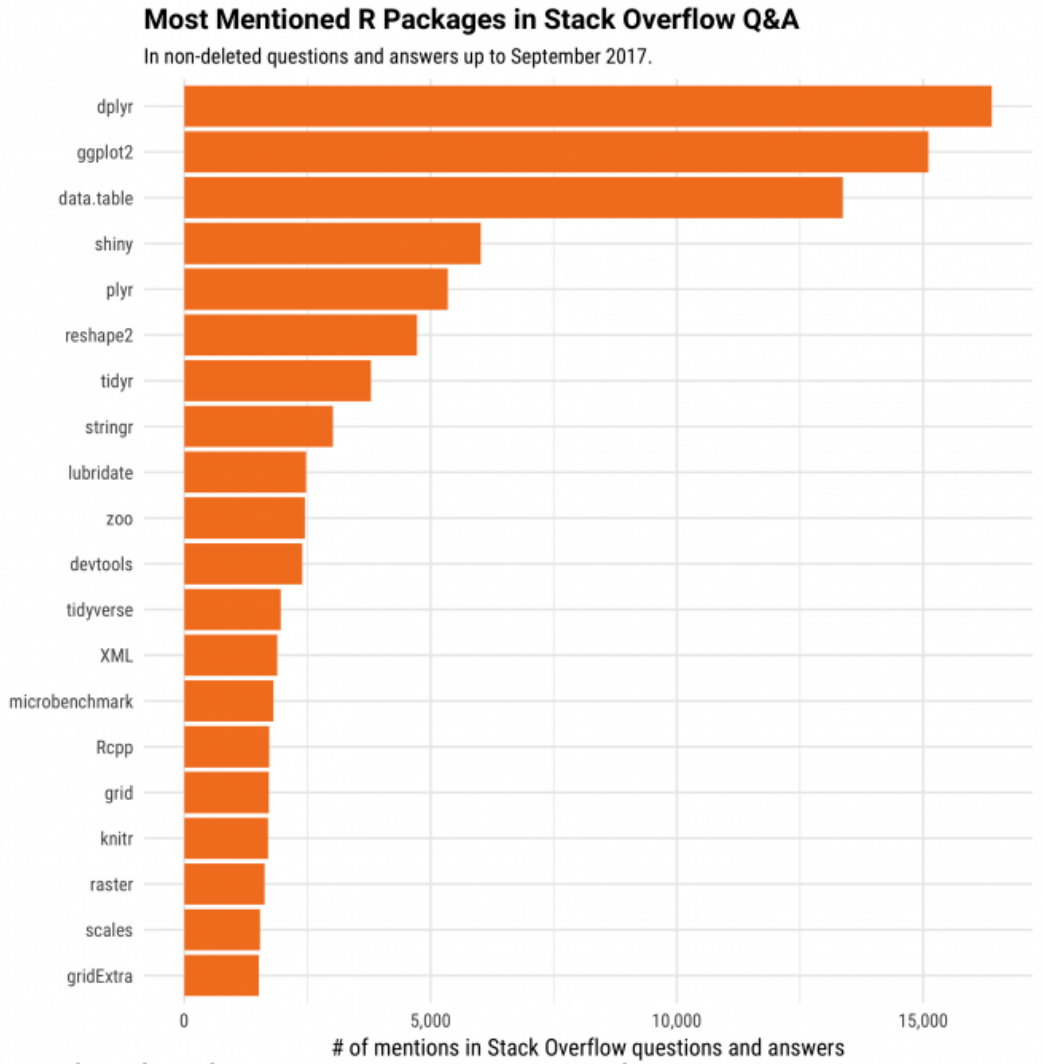
In R, the fundamental unit of **shareable code** is the package.

A package bundles together **code**, **data**, **documentation**, and **tests**, and is easy to share with others.

- Open code
- Can have conflicts (names!!)
- May not work with newer versions of R

Popular R packages

The famous packages:
(tidyverse)
dplyr
ggplot2
data.table
shiny
plyr
reshape2 ….



https://stackoverflow.blog/2017/10/10/impressive-growth-r/

Brief overview of R packages

The tidyverse is an opinionated collection of R packages designed for data science. All packages share an underlying philosophy and common APIs.

Project Site Link ›

Shiny makes it incredibly easy to build interactive web applications with R. Shiny has automatic "reactive" binding between inputs and outputs and extensive pre-built widgets.

Project Site Link ›

rmarkdown lets you insert R code into a markdown document. R then generates a final document, in a wide variety of formats, that replaces the R code with its results.

Project Site Link ›

Sparklyr is an R interface to Apache Spark, a fast and general engine for big data processing. This package connects to local and remote Apache Spark clusters, a 'dplyr' compatible back-end, and an interface to Spark's ML algorithms.

Project Site Link ›

ggplot 2 is an enhanced data visualization package for R. Create stunning multi-layered graphics with ease.

Project Site Link ›

Packages in the R Studio webpage

R Packages

Our developers create popular packages to expand the features of R. Includes ggplot2, dplyr, R Markdown & more.

Learn More

Using an R package

You install them from CRAN with

```
install.packages("xx")
```

You use them in R with

```
library("xx")
```

You get help on them with

```
package?xx
help(package = "xx").
```

```
devtools::session_info()
detach("package:xx", unload = T)
```

How to see source code of a function/method in R?

If it's an internal function of R (e.g. from base package), just type the function name

If it's a function in a package, e.g. to see the code of *plotMA* in the *DESeq2* package, first find the object it belongs to using **showMethods()**

….then to get source code by **getMethod()**

```
> getMethod("plotMA","DESeqDataSet")
```

Journal of Statistical Software

November 2017, Volume 82, Issue 4. *doi: 10.18637/jss.v082.i04*

ThresholdROC: Optimum Threshold Estimation Tools for Continuous Diagnostic Tests in R

Sara Perez-Jaume
University of Barcelona

Konstantina Skaltsa
University of Barcelona

Natàlia Pallarès
IDIBELL

Josep L. Carrasco
University of Barcelona

Abstract

We introduce an R package that estimates decision thresholds in diagnostic settings with a continuous marker and two or three underlying states. The package implements parametric and non-parametric estimation methods based on minimizing an overall cost function, as well as confidence interval estimation approaches to account for the sampling variability of the cut-off. Further features of the package include sample size determination and estimation of diagnostic accuracy measures. We used randomly generated data and two real datasets to illustrate the capabilities and characteristics of the package.

Keywords: ROC curve, threshold estimation, cost function, diagnostic tests, R package, bootstrap.

Second: using an R package

We will create a wordcloud from our course webpage:
http://www.ub.edu/datascience/postgraduate/

Prog-02.R

Prog-02.R

Prog-02.R

```
# Instalación y carga de paquetes y lectura de la información de
las palabras utilizadas en una página web
install.packages('RCurl')
library(RCurl)
install.packages('XML')
library(XML)

html <-getURL("http://www.ub.edu/datascience/postgraduate", followlocation
= TRUE)


# Es importante hacer un "parse" del contenido html
doc = htmlParse(html, asText=TRUE)
plain.text <- xpathSApply(doc, "//p", xmlValue)
cat(paste(plain.text, collapse = "\n"))
plain.text<-gsub("á", "a", plain.text)
plain.text<-gsub("é", "e", plain.text)
plain.text<-gsub("í", "i", plain.text)
plain.text<-gsub("ó", "o", plain.text)
plain.text<-gsub("ú", "u", plain.text)
```

Prog-02.R

Prog-02.R

```
plain.textclean <- sapply(plain.text, function(x) iconv(enc2utf8(x), sub =
"byte"))

# Instalación y carga del paquete necesario para construir una matriz de
términos con la que poder trabajar
install.packages("tm")
library(tm)
# Creamos un Corpus, que es un objeto de R que permite trabajar con datos
textuales
myCorpus = Corpus(VectorSource(plain.textclean))

#Pasamos la mayúsclas a minúsculas
myCorpus = tm_map(myCorpus, tolower)

# Eliminamos los signos de puntuación
myCorpus = tm_map(myCorpus, removePunctuation)

# Eliminamos los números
myCorpus = tm_map(myCorpus, removeNumbers)

# Eliminamos algunas palabras sin significado propio (artículos,..)
myCorpus = tm_map(myCorpus,removeWords, stopwords('english'))
myCorpus= tm_map(myCorpus,removeWords, c('will'))
```

Prog-02.R

Prog-02.R

```
# Creamos la matriz de términos
myDTM <- DocumentTermMatrix(myCorpus)

# La convertimos en una matriz de datos
m = as.matrix(myDTM)

# Ordenamos las filas en función de su mayor o menor frecuencia
v = sort(colSums(m), decreasing = TRUE)


# Instalación y carga del paquete necesario para obtener la nube de
términos
#install.packages('wordcloud')
library(wordcloud)

set.seed(4363)
# Finalmente creamos la nube:
wordcloud(names(v),v, min.freq = 100,
          colors=brewer.pal(6,"Dark2"),random.order=FALSE)
```

Our workplan Step 2

- RStudio projects, working directory, scripts and packages
- **Data structures: vectors, matrices, data.frames, lists, objects**
- Data wrangling: injestion+digestion
- R programming: if-the-else, loops. functions
- Rmarkdown: producing HTML, PDF, LateX, PPT

Fast-learning R when you already are a programmer

Some examples of R commands

```
> 10^2 + 36

> a = 4
> a

> rm(list=ls()) ## removes all variables from memory
```

Scalars, vectors and functions

```
> b=c(3,4,5)
> mean(b)
> rnorm(15)
> set.seed(1)
> rnorm(10)
> plot(rnorm(100))
```

Fast-learning R when you already are a programmer (II)

You can run in batch mode

```
> source("prog2.R")
```

You can change your working directory

```
> setwd('c\\project-2')
```

Data structures: vectors

```
> vec1 = c(1,4,6,8,10)
> vec1[5] ### careful Python lovers
> vec1[3] = 12
> vec2 = seq(from=0, to=1, by=0.25)
> sum(vec1)
> vec1+vec2
```

Fast-earning R when you already are a programmer (III)

Data structures: matrices

```
> mat=matrix(data=c(9,2,3,4,5,6),ncol=3)
> mat
> mat[2,] ### careful Python lovers
```

Data structures: data frames and lists

A data frame is a matrix with names above the columns. This is nice, because you can call and use one of the columns without knowing its position .

```
> t = data.frame(x = c(11,12,14),
      y = c(19,20,21), z = c(10,9,7))
> mean(t$z)
> mean(t[["z"]])
> L = list(one=1, two=c(1,2), five=seq(0, 1,length=5))
> names(L)
> L$two[2]
# note attach(t) and detach(t) for data frames
```

Miscelaneous

Reading and writing data files

```
write.table(t, file="tst0.txt", row.names=FALSE)
write.csv(t, file="file1.csv",  sep=';',  dec=',')
read.table(file="tst0.txt", header=TRUE)
```

Not available data

```
> vec = c(1,2,NA)
> max(vec, na.rm=TRUE)
> vec[is.na(vec)]<-99
> max(vec)
```

Characters and dates

```
as.date
as.character
as.numeric
> date1=strptime( c("20100225230000","20100226000000",
"20100226010000"),format="%Y%m%d%H%M%S")
```

More on vectors and arrays

Basic operators element-by-element

`*, /, +, -`

Repetition, sequence

```
> a=c(1,2)
> rep(a, times=3)

> z<- array(1:24, dim=c(2,3,4)))
> z
> dim(z)
```

Factors (Be careful!!!!!!)

```
> sex=c('M', 'W', 'W', 'W', 'W', 'M', 'M')
> sex=factor(sex)
> sex
> levels(sex)
> age=c(19, 20, 18, 18, 20, 21, 22)
> mean(age[sex=='W'])
```

Further tasks

- Create tables of frequencies and marginals
- Multidimensional tables , add names in rows and columns
- Typical functions: mean, sum, max, sd (standard deviation)
- **lapply** (calculates a function for all Items in a list a returns a list)
- **sapply** (returns a vector)
- **apply** (only for rows (1) or columns (2))
- **tapply** (function separating by a factor)

```
a <- matrix(1:12, nrow=3, ncol=4) # fills by columns
                                  #  byrow = TRUE for rows
apply(a,1,sum)
apply(a,2,mean)
tapply(age, sex, mean)
```

Basic operators: vectors (x) and matrices (A,B)

Operator	Description	Operator	Descrition
A*B	Element-wise multiplication	A %*% B	Matrix multiplication
t(A)	Transpose	A %o% B	Outer product. AB'
diag(x)	Creates diagonal MATRIX	crossprod(A,B)	A'B
diag(A)	Returns a vector containing the elements of the principal diagonal	solve(A, b)	Returns vector x in the equation b = Ax (i.e., $A^{-1}b$)
diag(k)	If k is a scalar, this creates a k x k identity matrix.	solve(A)	Inverse of A where A is a square matrix.
y<-eigen(A)	y$val are the eigenvalues y$vec are the eigenvectors	ginv(A)	Moore-Penrose Generalized Inverse of A. Requieres MASS package
y<-svd(A)	Single value decomposition of A. yd, yu, y%v (singular values, left and right sing. vectors	chol(A) y <- qr(A)	Choleski factorization of A. QR decomposition of A.
colMeans	Means of column (rows)	rowSums	Sums of rows (columns)

Let's practice

airquality is a dataset that comes with R basic packages

```
airquality
airquality[1:5,]
class(airquality[1:5,])
airquality[1,1]
airquality[1,]
class(airquality$Ozone)
sum(airquality$Ozone)
sum(airquality$Ozone[!is.na(airquality$Ozone)])
```

Prog-03.R

Things to remember

Here are five quick tips to remember when getting started in R:

Assignment. R may use the arrow operator (<-) for assignment, not necessarily a single equals (=).

Case Sensitive. The R language is case sensitive, meaning that C() and c() are two different function calls.

Help. You can help on any operator or function using the help() function or the ? operator and help with packages using the double question mark operator (??).

How To Quit. You can exit the R interactive environment by calling the q() function.

Documentation. R installs with a lot of useful documentation. You can review it in the browser by typing: *help.start()*

R for Python programmers

Batch mode

You can also run one or more R scripts in batch mode.

```
$ R CMD BATCH script_1.R script_2.R
```

You can also script inline using Rscript -e.

Example

Notice how we use a semi-colon to separate multiple commands in a single line

```
$ Rscript –e "library(knitr);knit('script.Rmd')“
```

http://ramnathv.github.io/pycon2014-r/

Our workplan Step 3

- RStudio projects, working directory, scripts and packages
- Data structures: vectors, matrices, data.frames, lists, objects
- **Data wrangling: injestion+digestion**
- R programming: if-the-else, loops. functions
- Rmarkdown: producing HTML, PDF, LateX, PPT

Data wrangling

Get to know the data frame

`head()` - see first 6 rows
`tail()` - see last 6 rows
`dim()` - see dimensions
`nrow()` - number of rows
`ncol()` - number of columns
`str()` - structure of each column
`names()` - will list the names attribute for a data frame (or any object really), which gives the column names.

What is Tidy Data?

A dataset is said to be **tidy** if it satisfies the following conditions

- observations are in rows
- variables are in columns
- contained in a single dataset.

Tidy data makes it easy to carry out data analysis.

Causes of Messiness

Here are some **examples** of the more commonly observed patterns in **untidy data**:

- Column headers are values, not variable names
- Multiple variables are stored in one column
- Variables are stored in both rows and columns
- Multiple types of experimental unit stored in the same table
- One type of experimental unit stored in multiple tables

Special features

To understand computations in R, two slogans are helpful:

> "Everything that exists is an object.
> Everything that happens is a function call."
> **John Chambers**


```
typeof()      # what is it?
length()      # how long is it? What about two
                dimensional objects?
attributes()  # does it have any metadata?
```

When **using factors in statistical modeling functions**, it is important to know what the baseline level is. This is the first factor but by default the ordering is determined by alphabetical order of words entered. You can change this by specifying the levels.

```
x <- factor(c("yes", "no", "yes"), levels =
c("yes", "no"))
```

Our workplan Step 4

- RStudio projects, working directory, scripts and packages
- Data structures: vectors, matrices, data.frames, lists, objects
- Data wrangling: injestion+digestion
- **R programming: if-the-else, loops. functions**
- Rmarkdown: producing HTML, PDF, LateX, PPT

Control Structures

These allow you to control the flow of execution of a script typically inside of a function. Common ones include:

```
if, else
for
while
repeat
break
next
return
```

Let's practice loops

```
x <- 1:15
if (sample(x, 1) <= 10)  {
          print("x is less than 10")
      }   else {
          print("x is greater than 10")
      }

ifelse(x <= 10, "x less than 10", "x greater than 10")

y <- if (sample(x, 1) < 10) { 5 } else { 0 }

for (i in 1:10) {
          print(i)
          }
```

Prog-04a.R

Functions

Basic components of a function
The body(), the code inside the function.
The formals(), the "formal" argument list, which controls how you can call the function.
The `environment()`` which determines how variables referred to inside the function are found.
args() to list arguments.

Variables defined inside functions exist in a different environment than the global environment. However, if a variable is not defined inside a function, it will look one level above.

Let's practice functions

```
add <- function(a, b) {
            return(a + b)
              }
vector <- c(3, 4, 5, 6)
sapply(vector, add, 1)

# Functions with pre defined values
temp <- function(a = 1, b = 2) { return(a + b) }

# Functions usually return the last value it computed
f <- function(x) { if (x < 10) { 0 } else { 10 } }

# This is a constructor function, i.e. a function that creates another one.
make.power <- function(n) { pow <- function(x) { x^n } pow }

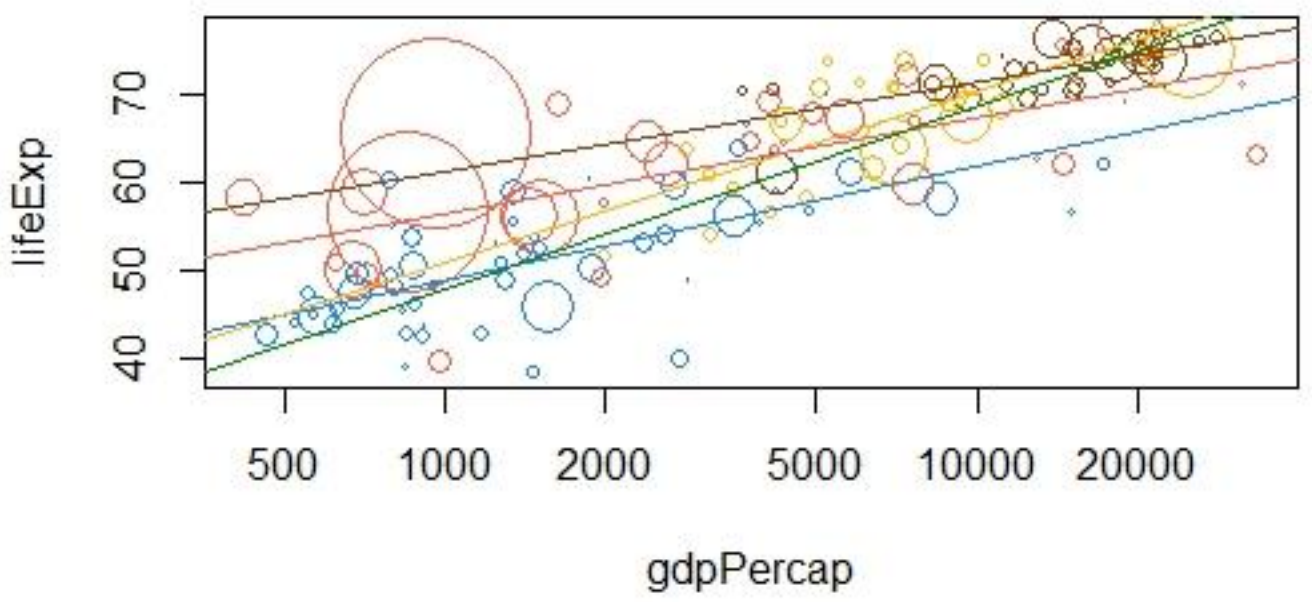
cube <- make.power(3)
square <- make.power(2)
```

Prog-04b.R

Our workplan Step 4

- RStudio projects, working directory, scripts and packages
- Data structures: vectors, matrices, data.frames, lists, objects
- Data wrangling: injestion+digestion
- **R programming: if-the-else, loops, functions…. a detour to graphics**
- Rmarkdown: producing HTML, PDF, LateX, PPT

Data visualization

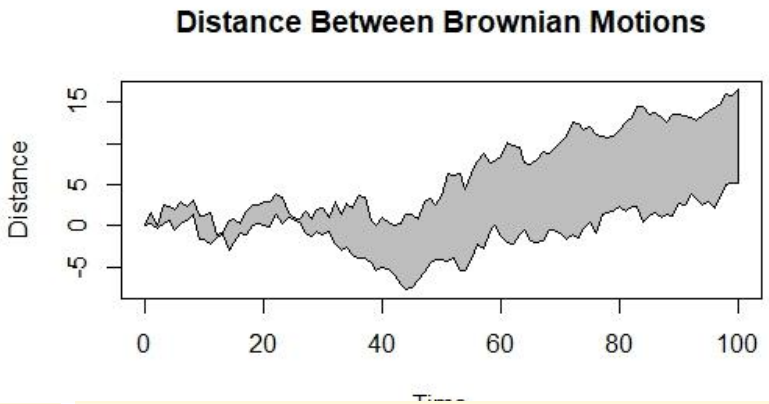


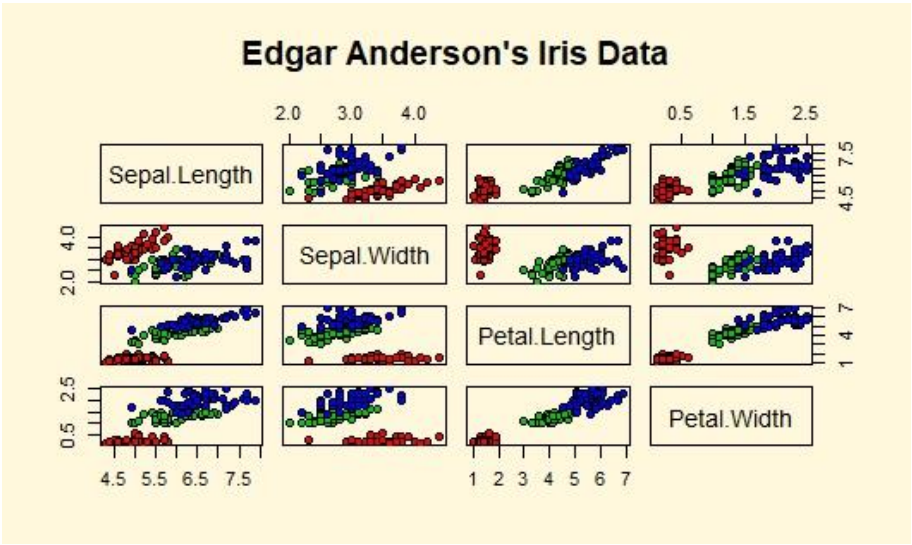
Gapminder, 1982 World data

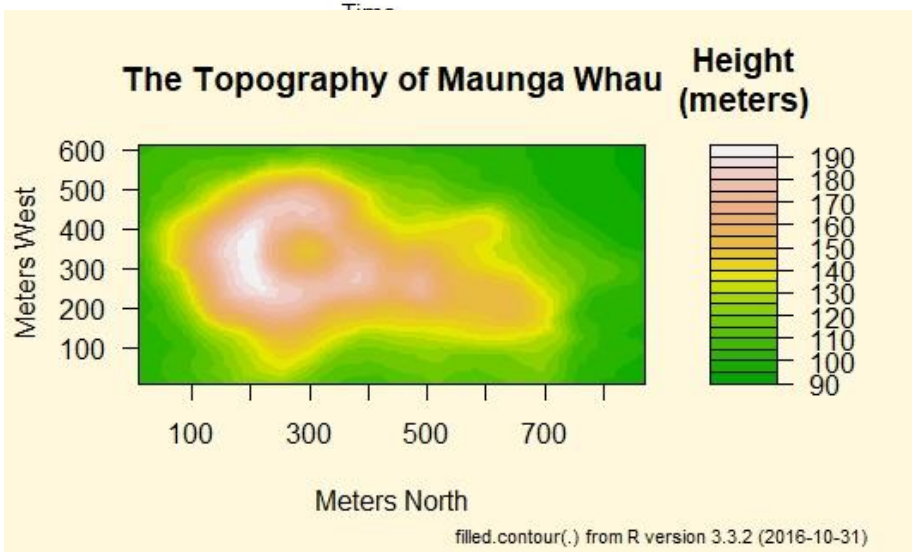
Prog-05.R

A few pictures

```
demo(image)
demo(persp)
demo(graphics)
```







Our workplan Step 5

- RStudio projects, working directory, scripts and packages
- Data structures: vectors, matrices, data.frames, lists, objects
- Data wrangling: injestion+digestion
- R programming: if-the-else, loops, functions …. a detour to graphics
- **Rmarkdown: producing HTML, PDF, LateX, PPT**

Rmarkdown

R Markdown provides a unified authoring framework for data science, combining your **code**, its **results**, and your **prose commentary**. R Markdown documents support dozens of output formats, like html, PDFs, Word files, slideshows, …

You need the **rmarkdown** package, but you don't need to explicitly install it or load it, as **RStudio** automatically does both when needed.

File > New file > Rmarkdown….

Choose HTML

Let's practice Rmarkdown (.Rmd)

Prog06.Rmd

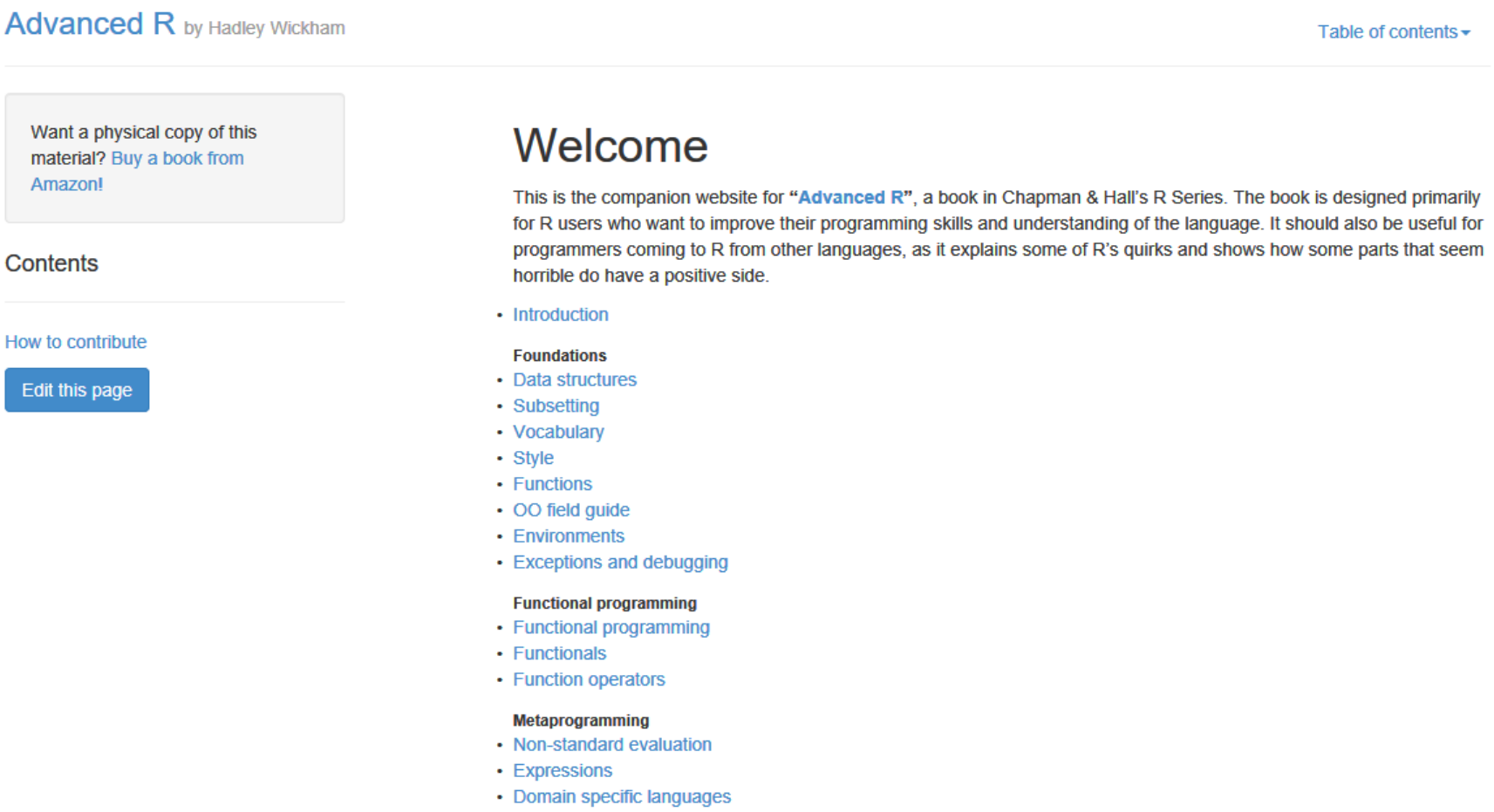
```
---
title: "Diamond sizes"
date: 2018-12-04
output: html_document
---
```

```
```{r setup, include = FALSE}
library(ggplot2)
library(dplyr)
smaller <- diamonds %>% filter(carat <= 2.5)
```
```

We have data about `r nrow(diamonds)` diamonds. Only `r nrow(diamonds) - nrow(smaller)` are larger than 2.5 carats. The price/carat by cut type is shown here:

```
```{r, echo = FALSE}
ggplot(diamonds, aes(x = price)) +
  geom_histogram(color = "black", fill = "DarkOrange", binwidth = 25) +
  scale_x_continuous(breaks = seq(0, 4000, 100)) +
  theme(axis.text.x = element_text(angle = 90)) +
  coord_cartesian(c(0,4000)) +
  facet_grid(cut~.) +
  xlab("Price") + ylab("Count")
```
```

What's next? Advanced R



Advanced R by Hadley Wickham

Table of contents

Want a physical copy of this material? Buy a book from Amazon!

Contents

How to contribute

Edit this page

Welcome

This is the companion website for **"Advanced R"**, a book in Chapman & Hall's R Series. The book is designed primarily for R users who want to improve their programming skills and understanding of the language. It should also be useful for programmers coming to R from other languages, as it explains some of R's quirks and shows how some parts that seem horrible do have a positive side.

- Introduction

Foundations

- Data structures
- Subsetting
- Vocabulary
- Style
- Functions
- OO field guide
- Environments
- Exceptions and debugging

Functional programming

- Functional programming
- Functionals
- Function operators

Metaprogramming

- Non-standard evaluation
- Expressions
- Domain specific languages

http://adv-r.had.co.nz/

Takeaways

- Basic structure of

- R programming:
  - Projects
  - Packages
  - Rmarkdown

Homework: ready for the case study?

- Get the bank data
- Find a better regression model for “account balance”
- Why do we need a predictive model in this case study?

References
Statistics with R

- *http://rstudio.com/cheatsheets*

- **Introduction to R for Python Programmers**
  http://ramnathv.github.io/pycon2014-r/
- **The Art of R Programming** Norman Matloff, WPublisher
- **R in action**, Robert I. Kabacoff, Manning Publications
- **Introductory Statistics with R**, Peter Dalgaard, Springer
- **Data Analysis and Graphics using R** , John Maindonald & W. John Braun, Cambridge University Press
- **The R Book**, Michael J. Crawley, Ed. John Wiley & Sons
- **R for dummies**, Joris Meys, Andrie de Vries Ed. John Wiley & Sons
- **Beginning R: The Statistical Programming Language**, Mark Gardener, Wrox

Statistics with R

I hope that you enjoyed!

http://www.ub.edu/riskcenter/guillen
mguillen@ub.edu
@mguillen_estany