R_intro_3

Karel Fišer, 2017

variables & functions

```
> x <- 2
> y <- c(2,4,5)
> animals <- "cat"

> rnorm(7)
[1]  0.9594941 -0.1102855 -0.5110095 -0.9111954
[5] -0.8371717  2.4158352  0.1340882
> random_x <- rnorm(7)
> rnorm(y[2])
```

function call

```
> rnorm(7, mean = 10)
[1]  8.792934 10.277429 11.084441  7.654302
[5] 10.429125 10.506056  9.425260

> rnorm(7, mean = 0, sd = 1) # defaults

> rnorm(7, mean=10, sd=2)
> rnorm(7, sd=2, mean=10) # with names any order

> rnorm(3, 10, 2) # w/o names exact order
```

help(rnorm)

Normal {stats} R Documentation

The Normal Distribution

Description

Density, distribution function, quantile function and random generation for the normal distribution with mean equal to `mean` and standard deviation equal to `sd`.

Usage

```
dnorm(x, mean = 0, sd = 1, log = FALSE)
pnorm(q, mean = 0, sd = 1, lower.tail = TRUE, log.p = FALSE)
qnorm(p, mean = 0, sd = 1, lower.tail = TRUE, log.p = FALSE)
rnorm(n, mean = 0, sd = 1)
```

Arguments

`x, q`	vector of quantiles.
`p`	vector of probabilities.
`n`	number of observations. If `length(n) > 1`, the length is taken to be the number required.
`mean`	vector of means.
`sd`	vector of standard deviations.
`log, log.p`	logical; if TRUE, probabilities p are given as log(p).
`lower.tail`	logical; if TRUE (default), probabilities are $P[X \le x]$ otherwise, $P[X > x]$.

Details

help(rnorm)

If `mean` or `sd` are not specified they assume the default values of `0` and `1`, respectively.

The normal distribution has density

$$f(x) = 1/(\sqrt{(2\,\pi)}\,\sigma)\; e^{-((x-\mu)^2/(2\,\sigma^2))}$$

where μ is the mean of the distribution and σ the standard deviation.

`qnorm` is based on Wichura's algorithm AS 241 which provides precise results up to about 16 digits.

Value

`dnorm` gives the density, `pnorm` gives the distribution function, `qnorm` gives the quantile function, and `rnorm` generates random deviates.

The length of the result is determined by `n` for `rnorm`, and is the maximum of the lengths of the numerical parameters for the other functions.

The numerical parameters other than `n` are recycled to the length of the result. Only the first elements of the logical parameters are used.

Source

For `pnorm`, based on

Cody, W. D. (1993) Algorithm 715: SPECFUN – A portable FORTRAN package of special function routines and test drivers. *ACM Transactions on Mathematical Software* **19**, 22–32.

For `qnorm`, the code is a C translation of

Wichura, M. J. (1988) Algorithm AS 241: The percentage points of the normal distribution. *Applied Statistics*, **37**, 477–484.

For `rnorm`, see RNG for how to select the algorithm and for references to the supplied methods.

References

Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) *The New S Language*. Wadsworth & Brooks/Cole.

Johnson, N. L., Kotz, S. and Balakrishnan, N. (1995) *Continuous Univariate Distributions*, volume 1, chapter 13. Wiley, New York.

help(rnorm)

See Also

Distributions for other standard distributions, including dlnorm for the *Log*normal distribution.

Examples

```
require(graphics)

dnorm(0) == 1/sqrt(2*pi)
dnorm(1) == exp(-1/2)/sqrt(2*pi)
dnorm(1) == 1/sqrt(2*pi*exp(1))

## Using "log = TRUE" for an extended range :
par(mfrow = c(2,1))
plot(function(x) dnorm(x, log = TRUE), -60, 50,
     main = "log { Normal density }")
curve(log(dnorm(x)), add = TRUE, col = "red", lwd = 2)
mtext("dnorm(x, log=TRUE)", adj = 0)
mtext("log(dnorm(x))", col = "red", adj = 1)

plot(function(x) pnorm(x, log.p = TRUE), -50, 10,
     main = "log { Normal Cumulative }")
curve(log(pnorm(x)), add = TRUE, col = "red", lwd = 2)
mtext("pnorm(x, log=TRUE)", adj = 0)
mtext("log(pnorm(x))", col = "red", adj = 1)

## if you want the so-called 'error function'
erf <- function(x) 2 * pnorm(x * sqrt(2)) - 1
## (see Abramowitz and Stegun 29.2.29)
## and the so-called 'complementary error function'
erfc <- function(x) 2 * pnorm(x * sqrt(2), lower = FALSE)
## and the inverses
erfinv <- function (x) qnorm((1 + x)/2)/sqrt(2)
erfcinv <- function (x) qnorm(x/2, lower = FALSE)/sqrt(2)
```

Rectangular data – data.frame

```
> d <- data.frame(animal=c("cat", "spider"),
  legs=c(4, 8))
> d[, 2] # d$legs
> d[2, 2] <- 7 # one leg off


> data(iris) # dataset in R
> write.table(iris, "iris.txt")
> data <- read.table("iris.txt")
```

I/O

```
> data(iris) # dataset in R
> write.table(iris, "iris.txt") # where is it?


> dir()
> getwd()
> setwd() # RStudio: Session/Set Working
  Directory/Choose Directory


> data <- read.table("iris.txt") # use Tab for
  completion
```

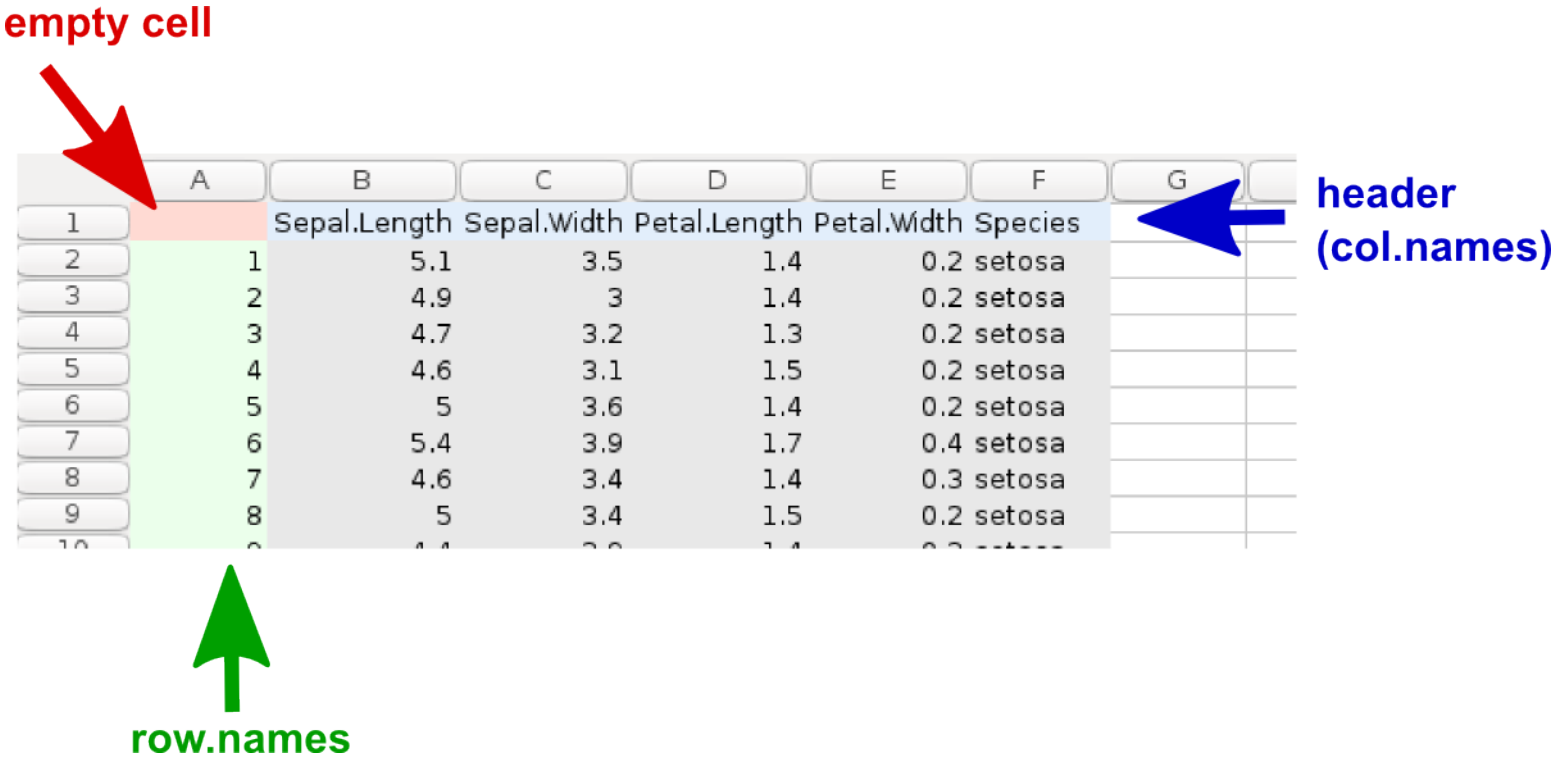
iris.txt

iris.txt

```
"Sepal.Length" "Sepal.Width" "Petal.Length" "Petal.Width" "Species"
"1" 5.1 3.5 1.4 0.2 "setosa"
"2" 4.9 3 1.4 0.2 "setosa"
"3" 4.7 3.2 1.3 0.2 "setosa"
"4" 4.6 3.1 1.5 0.2 "setosa"
"5" 5 3.6 1.4 0.2 "setosa"
"6" 5.4 3.9 1.7 0.4 "setosa"
"7" 4.6 3.4 1.4 0.3 "setosa"
"8" 5 3.4 1.5 0.2 "setosa"
"9" 4.4 2.9 1.4 0.2 "setosa"
```

	A	B	C	D	E	F	G	H
1	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species			
2	1	5.1	3.5	1.4	0.2	setosa		
3	2	4.9	3	1.4	0.2	setosa		
4	3	4.7	3.2	1.3	0.2	setosa		
5	4	4.6	3.1	1.5	0.2	setosa		
6	5	5	3.6	1.4	0.2	setosa		
7	6	5.4	3.9	1.7	0.4	setosa		
8	7	4.6	3.4	1.4	0.3	setosa		
9	8	5	3.4	1.5	0.2	setosa		
10	9	4.4	2.9	1.4	0.2	setosa		

iris.txt, iris.csv, iris.xls



	A	B	C	D	E	F	G
1		Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species	
2	1	5.1	3.5	1.4	0.2	setosa	
3	2	4.9	3	1.4	0.2	setosa	
4	3	4.7	3.2	1.3	0.2	setosa	
5	4	4.6	3.1	1.5	0.2	setosa	
6	5	5	3.6	1.4	0.2	setosa	
7	6	5.4	3.9	1.7	0.4	setosa	
8	7	4.6	3.4	1.4	0.3	setosa	
9	8	5	3.4	1.5	0.2	setosa	

```
> read.table("iris.txt")
> read.table("iris.csv", sep=",", header=TRUE,
  row.names=1)
> library(xlsx)
> read.xlsx("iris.xls")
```

read.table()

```
header = FALSE,
sep = "",
quote = "\"'",
dec = ".",
row.names, col.names,
...
check.names = TRUE,
fill = !blank.lines.skip,
...
stringsAsFactors = default.stringsAsFactors()
...
```

read.table - SAM

```
> sam <- read.table("LB-7K.final.sam", fill=TRUE,
  nrows=1000000)

> sam[255000, 10]
[1]
"CCGGGGACTCCATCAAGGCCATCGCATCCATTGCCGACAANGANNNNNNN
NNCANNNTCATGGGGATGCAGGAGAAGAAGCTGGGCTCGCTCCCCTACCAC
"

> sam[255000, 11]
[1] "8@<D;D:@FFHBH<E?<@93:???1CGDG>FFGHG"
```

read.table → t.test

```
> data <- read.table("iris.txt")
> t.test(data[,1], data[,3])

    Welch Two Sample t-test

data:  data[, 1] and data[, 3]
t = 13.0984, df = 211.543, p-value < 2.2e-16
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 1.771500 2.399166
sample estimates:
mean of x mean of y
 5.843333  3.758000
```

Your turn

create data outside R and load it in

packages

```
> install.packages("ggplot2") # CRAN
> # in RStudio Packages/Install Packages

> source("http://bioconductor.org/biocLite.R")
> biocLite("flowCore") # Bioconductor

> library(devtools)
> install_github("username/packagename") # github


> library(ggplot2)
```

packages - info

```
> library(help=ggplot2) # description
> library(ggplot2); ls("package:ggplot2")
> data(package="ggplot2")
```

where packages live:

```
> .libPaths()
> find.package("ggplot2", lib.loc=NULL, quiet =
  TRUE)
```

packages – where to start

- Google
- PubMed, StackOverflow

- CRAN and Bioconductor views

- vignettes
- (manuals), help
- explore the code (functions)

CRAN Task Views

Bayesian	Bayesian Inference
ChemPhys	Chemometrics and Computational Physics
ClinicalTrials	Clinical Trial Design, Monitoring, and Analysis
Cluster	Cluster Analysis & Finite Mixture Models
DifferentialEquations	Differential Equations
Distributions	Probability Distributions
Econometrics	Computational Econometrics
Environmetrics	Analysis of Ecological and Environmental Data
ExperimentalDesign	Design of Experiments (DoE) & Analysis of Experimental Data
Finance	Empirical Finance
Genetics	Statistical Genetics
Graphics	Graphic Displays & Dynamic Graphics & Graphic Devices & Visualization
HighPerformanceComputing	High-Performance and Parallel Computing with R
MachineLearning	Machine Learning & Statistical Learning
MedicalImaging	Medical Image Analysis
MetaAnalysis	Meta-Analysis
Multivariate	Multivariate Statistics
NaturalLanguageProcessing	Natural Language Processing
NumericalMathematics	Numerical Mathematics
OfficialStatistics	Official Statistics & Survey Methodology
Optimization	Optimization and Mathematical Programming
Pharmacokinetics	Analysis of Pharmacokinetic Data
Phylogenetics	Phylogenetics, Especially Comparative Methods
Psychometrics	Psychometric Models and Methods
ReproducibleResearch	Reproducible Research
Robust	Robust Statistical Methods
SocialSciences	Statistics for the Social Sciences
Spatial	Analysis of Spatial Data
SpatioTemporal	Handling and Analyzing Spatio-Temporal Data
Survival	Survival Analysis
TimeSeries	Time Series Analysis
WebTechnologies	Web Technologies and Services
gR	gRaphical Models in R

vignette

Using Time Dependent Covariates and Time Dependent Coefficients in the Cox Model

Terry Therneau Cindy Crowson
Mayo Clinic

February 26, 2013

1 Introduction

One of the strengths of the Cox model is its ability to encompass covariates that change over time, due to the theoretical foundation in martigales. A *martingale* (original definition) is a betting strategy in games of chance. One of the simplest and best known is doubling the bet each time you lose. For instance consider the following game of roulette:

Bet	Outcome	Win	Running total
R \$1	Red	2	1
R \$1	Black	0	0
R \$2	Black	0	-2
B \$4	Red	0	-6
R \$8	Black	0	-14
B \$16	Black	32	2
B \$1	Red	0	1
B \$2	Black	4	3
⋮	⋮	⋮	⋮

At the end of each cycle of bets the player is another \$1 ahead. The problem is that a modest sequence of losses will exhaust their stake.

The rule for time dependent covariates in a Cox model is simple and essentially the same as that for gambling: you cannot look into the future. A covariate may change in any way based on past data or outcomes, but it may not reach "forward" in time. One of the more well known examples of this error is analysis by response: at the end of a trial a survival curve is made comparing those who had an early response to treatment (shrinkage of tumor, lowering of cholesterol, or whatever), and it discovered that response predicts survival. The problem arises because subjects are classified as responders or non-responders from the beginning of the study, i.e., they are placed into group A or B before the response has occurred. As a consequence, any

early deaths that occur before response can be assessed will be assigned to the non-responder group, even deaths that have nothing to do with the condition under study.

save your work

objects:

```
> save(x, y, file="xy.rda")
> # saveRDS(x, "x.rds")
> save.image()  # "work.RData"
> load("xy.rda")
```

data:

```
> write.table(); read.table()
```

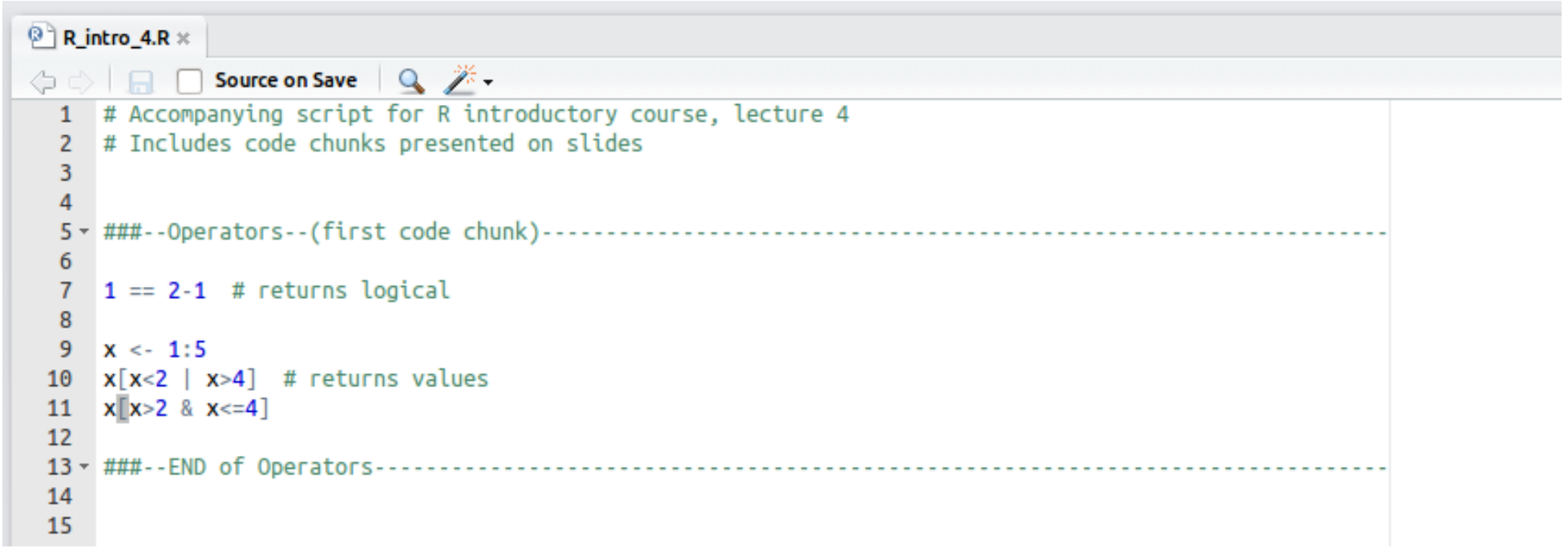
code:

```
> # use editor to save scripts
> source("iris_plots.R")
$ R CMD BATCH [options] iris_plots.R [outfile]
```

scripts

Scripts are plain text files storing your code.

Store them with ending .R (e.g. r_intro.R)



scripts

Scripts are plain text files storing your code.

- repeat your work any time → reproducible research
- repeat your work with any data → reports
- monitor changes → code versions

- have a style (e.g. google-r-style.html)
- make comments! # make the code understandable to others (e.g. your future self)

make comments

```
># comment why not what (with exceptions)
># whole line or
>x <- 2 # after code
>
># separate code into chunks:
># ----------------------
># ======================
># ---- Title ----------
>
># TODO(who)
```

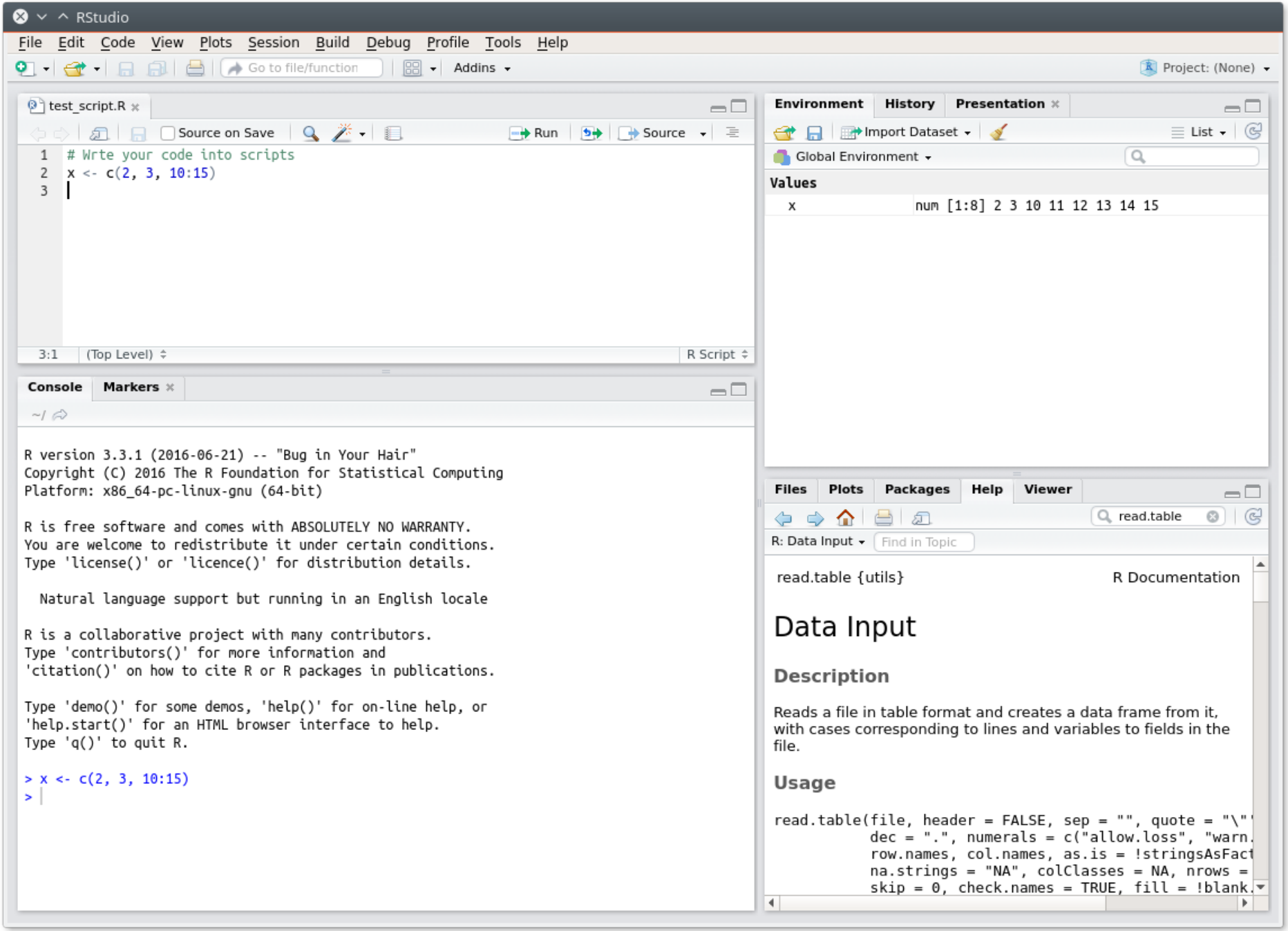
Rstudio - shortcuts

Alt + Shift + K

- Ctrl + Enter
- Ctrl + 1, Ctrl + 2
- Ctrl + → , Ctrl + Shift + →
- Ctrl + Shift + c
- Alt + -
- Esc
- Tab
- up-arrow, or Ctrl + R

https://www.rstudio.com/resources/cheatsheets/

RStudio


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Project: (None)
test_script.R
Source on Save
Run
Source
1 # Wrte your code into scripts
2 x <- c(2, 3, 10:15)
3
3:1 (Top Level)
R Script
Console
Markers
~/
R version 3.3.1 (2016-06-21) -- "Bug in Your Hair"
Copyright (C) 2016 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.
Natural language support but running in an English locale
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
> x <- c(2, 3, 10:15)
>
Environment
History
Presentation
Import Dataset
List
Global Environment
Values
x
num [1:8] 2 3 10 11 12 13 14 15
Files
Plots
Packages
Help
Viewer
read.table
R: Data Input
Find in Topic
read.table {utils}
R Documentation
Data Input
Description
Reads a file in table format and creates a data frame from it, with cases corresponding to lines and variables to fields in the file.
Usage
read.table(file, header = FALSE, sep = "", quote = "\"'
dec = ".", numerals = c("allow.loss", "warn.
row.names, col.names, as.is = !stringsAsFact
na.strings = "NA", colClasses = NA, nrows =
skip = 0, check.names = TRUE, fill = !blank.

Your turn

write a script

http://www.bioinformin.net