

(QUICK) INTRODUCTION TO R AND LITERATE PROGRAMMING

ARTHUR ALLIGNOL

INTRODUCTION

WHAT IS R?

- R is *an environment for statistical computing and graphics*
 - free software (GPL license)
 - Available for Linux, Solaris, Windows, Mac OSX
- Derived from the S language
- <http://www.r-project.org/>
- The strength of R is its community
 - ≥ 6000 packages additional packages implementing the latest statistical techniques
 - <http://cran.r-project.org/>

WHAT IS R?

- R is an interactive language
 - Statements converted to machine instructions as they are encountered
 - Flexible but slower than compiled code
- Object oriented (kinda)
- R provides excellent graphics functionality (base graphics and additional packages)
- Can be used both as a glorified calculator or as a "real" programming language

BASICS

R AS A CALCULATOR

```
1 + 1
```

```
[1] 2
```

```
2 + 3 * 4
```

```
[1] 14
```

```
sqrt(9)
```

```
[1] 3
```

```
pi
```

```
[1] 3.141593
```

R AS A SMART CALCULATOR

```
x <- 5          #allocate value to an object
x = 5           #as above
print(x)        # objects in console
```

```
[1] 5
```

```
x              #same
```

```
[1] 5
```

```
a <- sqrt(81)   #square root
b <- exp(2)     #exponential function
c <- log(50)    #logarithm
a * b + c       #basic operations
```

```
[1] 70.41353
```

GETTING HELP

- At the R prompt

```
help("fun")  
?fun
```

- The **sos** package (see next slide for installing packages) and the `findfn` function
- On the internet
 - [R mailing list](#)
 - [stackoverflow](#)
 - [Cross Validated](#)

PACKAGES

- Install package

```
install.packages("new_package")  
install.packages(c("new_package1", "new_package2"))
```

- Update packages

```
update.packages()
```

- Packages available from, e.g., [CRAN](#)
- CRAN [task views](#) provide list of packages grouped by theme

OPERATIONS

VECTOR OPERATIONS

```
# define vector by means of the  
# specification of its single elements  
W <- c(1,2,3,5,9,7)  
W
```

```
[1] 1 2 3 5 9 7
```

```
# vector with a 3 (repeated 5 times)  
X <- rep(3,5)  
X
```

```
[1] 3 3 3 3 3
```

```
# combine vectors  
A <- c(W,X)  
A
```

```
[1] 1 2 3 5 9 7 3 3 3 3 3
```

VECTOR OPERATIONS

```
# vector with values from 1 to 10 with stepsize 2
Y <- seq(0,10,2)
Y
```

```
[1]  0  2  4  6  8 10
```

```
#equals seq(0,10,1)
Z <- 0:10
Z
```

```
[1]  0  1  2  3  4  5  6  7  8  9 10
```

```
# element-wise addition
W+Y
```

```
[1]  1  4  7 11 17 17
```

```
# element-wise multiplication
W*Y
```

```
[1]  0  4 12 30 72 70
```

```
# element-wise multiplication with a scalar
5*W
```

```
[1]  5 10 15 25 45 35
```

VECTOR OPERATIONS

```
length(W)           #determine length of vector
```

```
[1] 6
```

```
t(W)%%Y             #transposition and vector multiplication
```

```
      [,1]  
[1,] 188
```

```
W[4]                 #select specific element
```

```
[1] 5
```

VECTOR OPERATIONS

```
W[W>2]
```

```
[1] 3 5 9 7
```

```
W[W>2 & W<7]
```

```
[1] 3 5
```

```
which(W>2)      #determine indices where a condition holds
```

```
[1] 3 4 5 6
```

MATRIX OPERATIONS

```
A=matrix(0,nrow=2,ncol=3)    #Initialize 2x3 matrix including 0
A
```

	[,1]	[,2]	[,3]
[1,]	0	0	0
[2,]	0	0	0

```
A[1,3]=5    #allocate single value at position (1,3)
A
```

	[,1]	[,2]	[,3]
[1,]	0	0	5
[2,]	0	0	0

```
A[,2]=c(1,2)    #allocate values to a whole column (analogously with rows)
A
```

	[,1]	[,2]	[,3]
[1,]	0	1	5
[2,]	0	2	0

MATRIX OPERATIONS

```
A[-3]           # A without the third element
```

```
[1] 0 0 2 5 0
```

```
B=matrix(c(1,1,1,2,2,2,3,3,3),3,3) #Initialize matrix with single values  
B
```

```
      [,1] [,2] [,3]  
[1,]    1    2    3  
[2,]    1    2    3  
[3,]    1    2    3
```

```
C=diag(c(1,2,3)) #diagonal matrix  
C
```

```
      [,1] [,2] [,3]  
[1,]    1    0    0  
[2,]    0    2    0  
[3,]    0    0    3
```


MATRIX OPERATIONS

`B * C` `#element-wise operations`

	<code>[,1]</code>	<code>[,2]</code>	<code>[,3]</code>
<code>[1,]</code>	1	0	0
<code>[2,]</code>	0	4	0
<code>[3,]</code>	0	0	9

`B + C`

	<code>[,1]</code>	<code>[,2]</code>	<code>[,3]</code>
<code>[1,]</code>	2	2	3
<code>[2,]</code>	1	4	3
<code>[3,]</code>	1	2	6

`B %% C` `# matrix multiplication`

	<code>[,1]</code>	<code>[,2]</code>	<code>[,3]</code>
<code>[1,]</code>	1	4	9
<code>[2,]</code>	1	4	9
<code>[3,]</code>	1	4	9

MATRIX OPERATIONS

A

```
      [,1] [,2] [,3]
[1,]    0    1    5
[2,]    0    2    0
```

```
(A <- rbind(A, c(0, 1, 5))) #add row
```

```
      [,1] [,2] [,3]
[1,]    0    1    5
[2,]    0    2    0
[3,]    0    1    5
```

```
(A <- cbind(A, c(0, 1, 4))) #add column
```

```
      [,1] [,2] [,3] [,4]
[1,]    0    1    5    0
[2,]    0    2    0    1
[3,]    0    1    5    4
```

CONTROL FLOW

FOR LOOP

```
for (i in 1:3) {  
  print(i)  
}
```

```
[1] 1  
[1] 2  
[1] 3
```

```
item <- c("Apple", "Orange", "Tomato")  
for (i in item)  
  print(i)
```

```
[1] "Apple"  
[1] "Orange"  
[1] "Tomato"
```

WHILE LOOP

```
i <- 5  
while(i < 10){  
  print(i)  
  i <- i + 1  
}
```

```
[1] 5  
[1] 6  
[1] 7  
[1] 8  
[1] 9
```

IF-THEN-ELSE

```
x <- 8  
y <- 7  
if (x < y) {  
  print(x)  
} else {  
  print(y)  
}
```

```
[1] 7
```

IFELSE()

```
(z <- ifelse(x < y, x, y))
```

```
[1] 7
```

- ifelse works with vectors

```
a <- c(3, 7, 8)
b <- c(4, 6, 9)
ifelse(a < b, a, b)
```

```
[1] 3 6 8
```

DISTRIBUTIONS

NORMAL DISTRIBUTION

```
# Value of a N(5,34) density function of at point x=2  
dnorm(x=2, mean = 5, sd = 7)
```

```
[1] 0.05199096
```

```
# Value of a N(0,1) distribution function at point x=2  
pnorm(q=2, mean = 0, sd = 1)
```

```
[1] 0.9772499
```

```
# p-quantile of a N(0,1) distribution  
qnorm(p=0.25, mean = 0, sd = 1)
```

```
[1] -0.6744898
```

```
# Simulation of 50 N(0,1)-distributed random variables  
x <- rnorm(n=50, mean = 0, sd = 1)
```

OTHER DISTRIBUTIONS

```
rexp(n = 5, rate = 1) # exponential distribution
```

```
[1] 1.1623165 1.3561561 0.9712323 2.1551756 1.1152585
```

```
punif(1.5, 1, 2) # Uniform distribution [1,2]
```

```
[1] 0.5
```

See the [distribution help page](#) and the [Distribution task view](#) for all distributions available

MATH FUNCTIONS

MATH FUNCTIONS

```
x <- abs(rnorm(50, 2, 2)) #abs: absolute value

sqrt(x)      #square root
exp(x)       #exponential function
log(x)       #logarithm
sin(x)       #sinus
cos(x)       #cosinus
sort(x)      #sort entries
quantile(x, 0.25) #sample 25%-quantile

mean(x)      #sample mean
var(x)       #sample variance
sd(x)        #sample standard deviation
sqrt(var(x))
median(x)    #sample median
min(x)       #sample minimum
max(x)       #sample maximum
summary(x)   #Overview of important sample parameters
```

USER DEFINED FUNCTIONS

A FUNCTION WITH ONE PARAMETER

```
myFun <- function(x) {  
  x^2 + 1  
}  
myFun(3)
```

```
[1] 10
```

A FUNCTION WITH MORE PARAMETERS

```
complicated_function <- function(x, y = 2, text = "apple") {  
  length_text <- nchar(text)  
  if (y > 0) {  
    res <- x + log(y) * length_text  
  } else {  
    res <- x + log(abs(y)) * length_text  
  }  
  
  res  
}
```

A FUNCTION WITH MORE PARAMETERS

```
(a <- complicated_function(x = 10))
```

```
[1] 13.46574
```

```
(b <- complicated_function(10, -2, "fffff"))
```

```
[1] 13.46574
```

- Careful with floating points

```
a <- sqrt(2)  
a * a - 2
```

```
[1] 4.440892e-16
```

```
a * a == 2
```

```
[1] FALSE
```


WORK WITH DATA

DATA FRAMES

```
x <- c("Franz", "Anton", "Heinrich")
y <- c(1.0, 1.7, 2.7)
z <- c(95, 85, 67)
grades_list = data.frame(name=x, grade=y, points=z) #create Data Frame
grades_list
```

	name	grade	points
1	Franz	1.0	95
2	Anton	1.7	85
3	Heinrich	2.7	67

```
# add a row
(grades_list <- rbind(grades_list,
  data.frame(name= "Maria", grade = 3.7, points = 54)) )
```

	name	grade	points
1	Franz	1.0	95
2	Anton	1.7	85
3	Heinrich	2.7	67
4	Maria	3.7	54

DATA FRAMES

```
# create new gender vector
gender <- c(rep("m", 3), "f")
grades_list <- cbind(grades_list, gender) #add column
grades_list
```

	name	grade	points	gender
1	Franz	1.0	95	m
2	Anton	1.7	85	m
3	Heinrich	2.7	67	m
4	Maria	3.7	54	f

DATA FRAMES

- Column access

```
grades_list$points
```

```
[1] 95 85 67 54
```

```
grades_list[, "points"]
```

```
[1] 95 85 67 54
```

```
grades_list[, 3]
```

```
[1] 95 85 67 54
```

```
grades_list[[3]]
```

```
[1] 95 85 67 54
```

DATA FRAMES

```
# determine mean grade of all male individuals
x <- subset(grades_list, gender == "m")$grade
# equivalently
y <- grades_list[grades_list$gender == "m", "grade"]
mean(x) == mean(y)
```

```
[1] TRUE
```

READ DATA

- The `read.table` function is used to read data into R in the form of a data frame, i.e., data with mixed modes
- `read.table` expects each field (variable) to be separated by separators (by default, spaces, tabs, newlines or carriage returns)
 - The `sep` argument can be used to specify an alternative separator
- R provides convenience functions for reading comma- and tab-separated data

<code>read.csv</code>	Separated by ,
<code>read.csv2</code>	Separated by ; decimal point ,
<code>read.delim</code>	Separated by tabs
<code>read.delim2</code>	Separated by tabs, decimal point ,

These functions are wrappers for `read.table` with the `sep` argument set appropriately

READ DATA

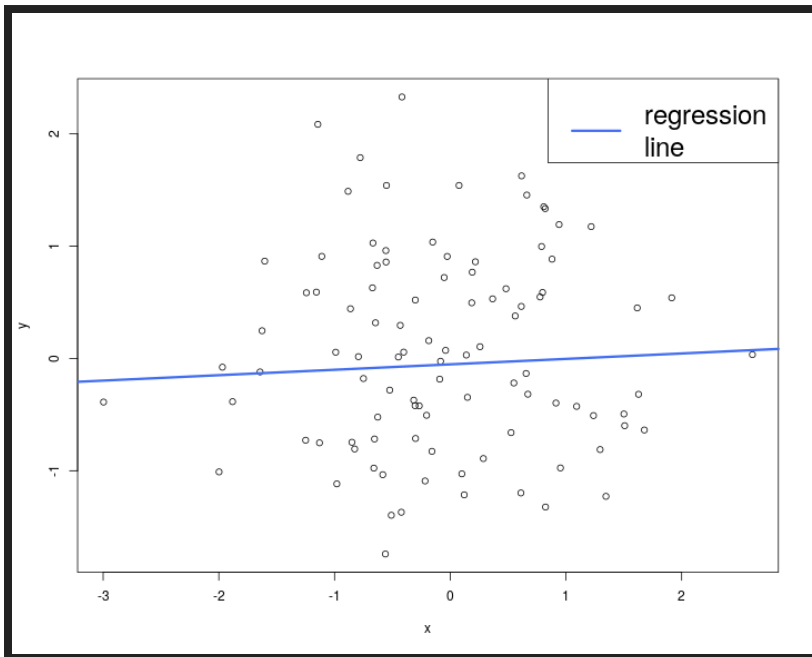
Useful options:

<code>file</code>	File to be read
<code>sep</code>	e.g., " <code>\t</code> ", " <code>,</code> ", " <code>"</code>
<code>dec</code>	Specify decimal point (default is <code>.</code>)
<code>header</code>	TRUE if the the first line are the column names (default to TRUE for <code>read.csv...</code>)
<code>col.names</code>	A vector of column names
<code>stringsAsFactors</code>	Logical. If FALSE, prevent the automatic conversion of character strings into factors
<code>na.strings</code>	By default, NA, NaN, Inf and -Inf are considered as missing values. Change this behaviour using <code>na.strings</code>
<code>skip</code> and <code>nrows</code>	Number of lines to skip and number of lines to read, respectively
<code>fill</code>	If TRUE, observations with fewer variables are filled with NAs or blanks
<code>colClasses</code>	Specify the modes of the columns to be read
<code>fileEncoding</code>	Encoding of the file. Useful for non ASCII characters from other platforms

GRAPHICS

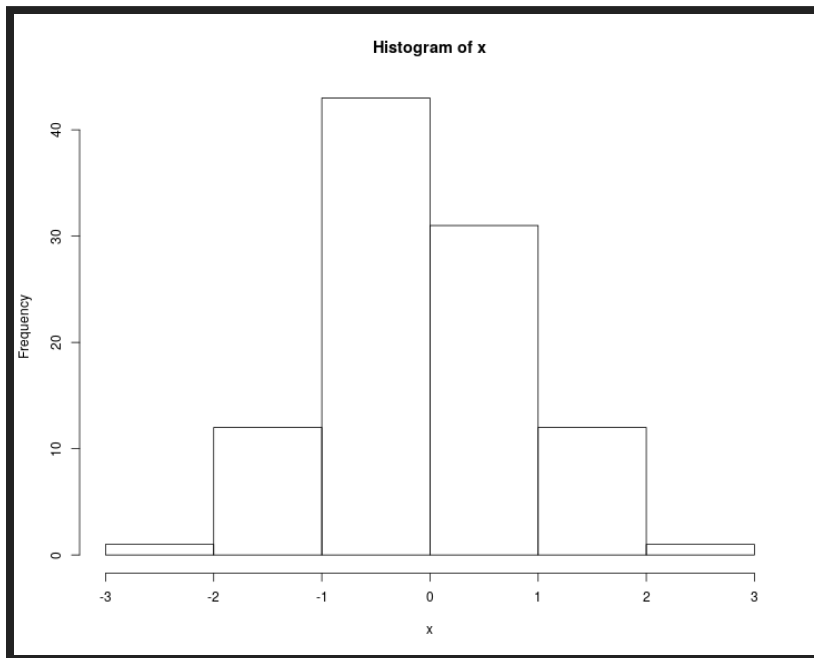
A SIMPLE GRAPHIC

```
x <- rnorm(100)
y <- rnorm(100)
plot(x, y, col = 1)
abline(lm(x~y), lwd = 3, col = "royalblue2")
legend("topright", "regression\nline", col = "royalblue2",
      lwd = 3, cex = 2)
```



ANOTHER SIMPLE GRAPHIC

```
hist(x)
```



REPRODUCIBLE RESEARCH

LITERATE PROGRAMMING

Literate programming is an approach to programming introduced by Donald Knuth (1970s) in which a program is given as an explanation of the program logic in a natural language, such as English, interspersed with snippets of macros and traditional source code, from which a compilable source code can be generated,

i.e., writing documentation containing computer code

- For statistician it means being able to
 - combine programming code with report text (article, presentation) in a single self-documenting file
 - document the code and its results, including interpretation of the results
 - allow an analysis to be rerun and the report (article, presentation) to be re-typeset by running a single command

REAL LIFE DATA

```
library(fortunes)
fortune("Tolstoy")
```

```
Happy families are all alike; every unhappy family is unhappy in its own
way.
Leo Tolstoy
```

```
and every messy data is messy in its own way - it's easy to define the
characteristics of a clean dataset (rows are observations, columns are
variables, columns contain values of consistent types). If you start to
look at real life data you'll see every way you can imagine data being
messy (and many that you can't)!
```

```
-- Hadley Wickham (answering 'in what way messy data sets are messy')
R-help (January 2008)
```

REAL LIFE DATA

	A	IV	IW	IX	IV	IZ	JA	JB	JC	JD	JE
1		Mittelwert	Mittelwert	Mittelwert	Mittelwert	Mittelwert	Mittelwert	Mittelwert	Mittelwert	Mittelwert	Mittelwert
2		PTR50	PTR50	PTR50	PTR50	BasePost	BasePost	BasePost	BasePost	BasePost	BasePost
3		ACMmean	ABPmean	Puls	CO2	ACEsyst	ACEdiast	ACEmean	ACE_RI	ACE_PI	ACMmean
4		cm/s	mmHg	Schläge/min		cm/s	cm/s	cm/s			cm/s
5											
6											
7											
8											
24	M16	76.371167	82.266667	80.72	38.86	21.662667	7.9630833	12.978992	0.6304667	1.0567333	74.523167
25	M17	63.063	88.45	61.313333	44.806667	22.201667	6.6906583	11.139333	0.6969	1.3963333	54.4775
26	M18	58.430167	80.633333	59.263333	36.986667	23.46575	5.98675	11.139975	0.7432	1.5663333	59.007667
27	M19	60.676	60.356667	85.326667	33.746667	22.81125	5.2969583	10.882025	0.7523	1.5543333	58.070833
28	M20	ND	ND	ND	ND	0	0	0	0	0	ND
29	M21	ND	ND	ND	ND	0	0	0	0	0	ND
30	M22	ND	ND	ND	ND	0	0	0	0	0	ND
31	M23	ND	ND	ND	ND	0	0	0	0	0	ND
32	C01	63.2555	68.05	62.553333	40.116667	24.255	7.417025	11.974783	0.6934333	1.409	63.1785
33	C02	78.398833	75.466667	64.183333	30.736667	25.461333	9.5011583	14.1372	0.6257667	1.1303333	83.108667
34	C03	81.851	65.386667	66.006667	40.083333	23.69675	8.90505	13.329983	0.6204667	1.12	81.260667
35	C04	44.826833	60.243333	71.69	40.226667	23.69675	8.8088	12.192308	0.6270333	1.2216667	42.106167
36	C05	60.881333	69.37	57.713333	40.006667	25.050667	5.79425	10.963517	0.7676667	1.76	61.664167
37	C06	61.895167	64.723333	56.256667	44.95	24.511667	2.8602292	9.50565	0.8815667	2.337	60.188333
38	C07	67.272333	82.496667	68.973333	31.786667	20.46275	7.546	11.96965	0.6287667	1.0839667	63.961333
39	C08	79.8875	61.253333	63.253333	39.96	22.11825	6.7554667	11.578875	0.6934333	1.3286667	86.329833
40	C09	56.081667	69.716667	72.756667	35.846667	20.798342	5.6672	11.413967	0.7303333	1.3773333	57.904
41	C10	70.968333	70.63	59.98	33.92	24.210083	8.3211333	12.3816	0.6562333	1.3261	69.6465
42	C11	65.4885	57.193333	54.646667	39.116667	22.913917	6.966575	11.174625	0.6917333	1.4503333	63.550667
43	C12	71.212167	68.53	57.23	33.403333	24.172225	4.4589417	9.51335	0.8033333	2.051	68.748167
44	C13	62.498333	67.456667	68.316667	35.743333	21.5985	5.026175	8.9525333	0.7446667	1.78	59.931667

45	C14	55.388667	96.22	59.04	34.883333	18.501817	6.5822167	10.1409	0.6438667	1.1763333	55.183333
46	C15	64.397667	69.9	58.443333	39.443333	18.057142	6.2703667	10.144108	0.6583667	1.1783333	59.700667
47	C16	50.832833	73.603333	53.75	37.706667	23.31175	9.4556	13.023267	0.5929333	1.0642	48.715333
48	C17	56.877333	64.573333	62.143333	44.436667	21.278308	7.2688	11.825275	0.6631667	1.1966667	56.646333
49	C18	71.853833	99.513333	71.756667	33.19	23.472167	9.3413833	14.218692	0.6012667	0.9945667	69.081833
50	C19	56.120167	79.136667	79.21	35.913333	20.748933	6.1086667	9.9721417	0.7038333	1.4726667	52.206
51	C20	ND	ND	ND	ND	0	0	0	0	0	ND
52											
53											
54	Mittelwert M	63.362895	78.68807	69.080175	38.029123	23.416612	6.6535396	11.807241	0.7116263	1.4604316	59.353491
55	Standardab M	11.028062	11.74354	11.654379	2.8089418	2.2251624	2.1207966	1.7868907	0.0849892	0.3577957	10.566291
56											
57	Mittelwert C	64.209895	71.761228	63.57386	37.445789	22.542966	7.0028967	11.495391	0.6856772	1.3925351	63.321693
58	Standarab C	9.889746	11.079763	7.0194203	3.9793285	2.1215214	1.8141646	1.5395569	0.0741666	0.3570025	11.328758

REAL LIFE DATA

The problems

- Bad structure
- Non ascii characters (e.g., Umlaut)
- Variable names with spaces; on several lines, ...
- Colour coding
- No consistent definition of missing values
- Free text
- (Wrong input, e.g., individuals who die twice)
- ...

Often data need to be transformed/reshaped for fitting a particular model

THE BIG ADVANTAGE OF LITERATE PROGRAMMING

Changes are **a lot** easier

- Data analysis is done by the doctors messed up with Z. They are sending you a new data set
- Z is included in a lot of regression models
- You did the data transformation in Excel → no trace of it
- You copy-pasted the results of the models from the R/SAS console in a Word document
- It's Friday 5pm, the results **have** to be available on Monday morning

⇒ Have a nice weekend

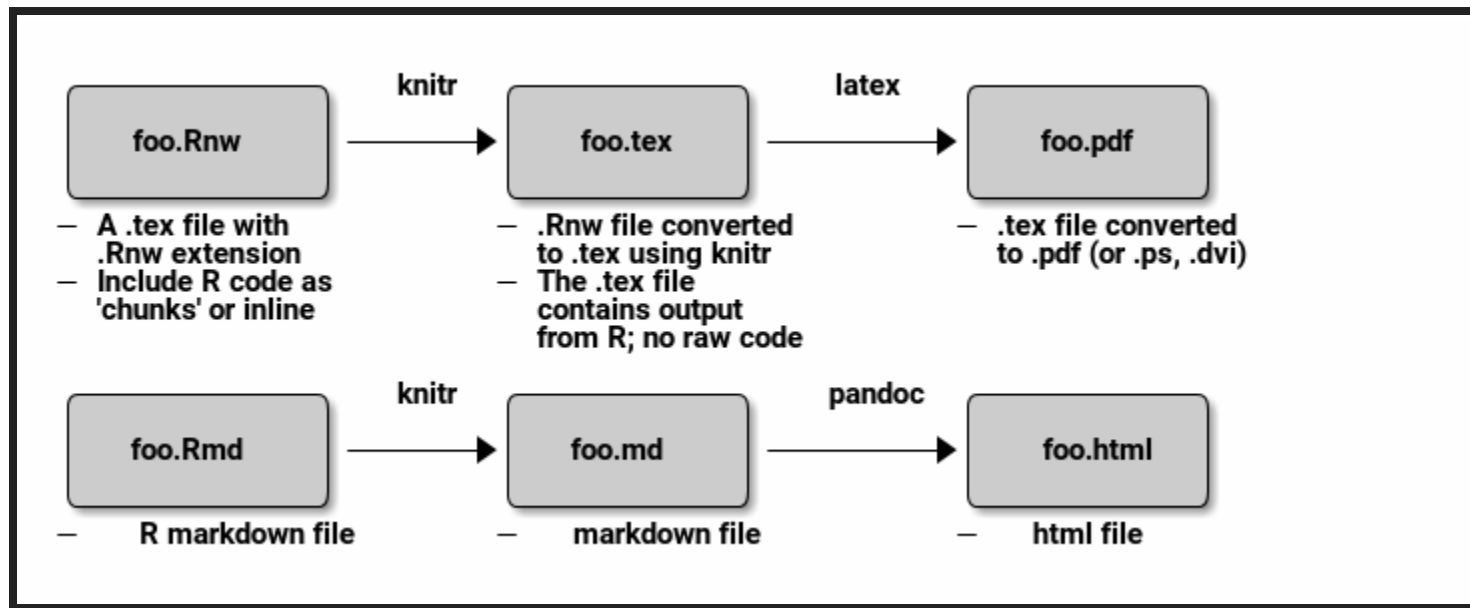
LITERATE PROGRAMMING IN R

2 main tools

- **Sweave** (2002): a tool that allows to embed the R code for complete data analyses in latex documents. The purpose is to create dynamic reports, which can be updated automatically if data or analysis change.
 - **Sweave** is part of every R installation
- **knitr**: "The knitr package was designed to be a transparent engine for dynamic report generation with R, solve some long-standing problems in Sweave, and combine features in other add-on packages into one package."
 - e.g., other formats (html, markdown)

Both are very well integrated with the Rstudio editor

KNITR



KNITR WITH A LATEX

```
\documentclass{article}

\begin{document}

Some R Code:
<<>>=
x <- rnorm(1000)
@

Let's display an histogram of {\tt x}
<<fig_path = "graphics", caption = "An histogram of x", out.width = ".5\\linewidth", fig.align =
hist(x)
@

\end{document}
```

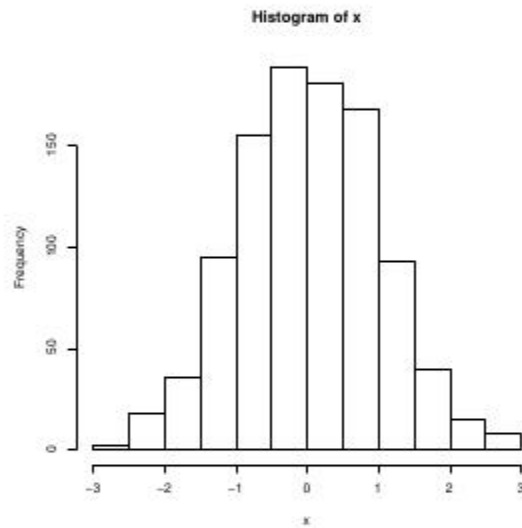
KNITR WITH LATEX

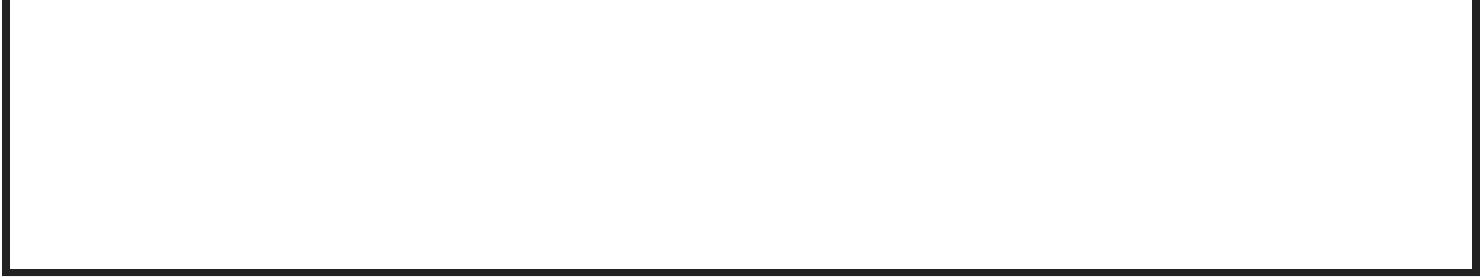
Some R Code:

```
x <- rnorm(1000)
```

Let's display an histogram of x

```
hist(x)
```





KNITR WITH MARKDOWN

```
## A simple graphic
```

```
With a list before
```

- item 1
- item 2

```
And some R code
```

```
```{r graphics2, out.width = "100px", fig.width = 10, fig.height = 8}  
x <- rnorm(100)
y <- rnorm(100)
plot(x, y, col = 1)
abline(lm(x~y), lwd = 3, col = "royalblue2")
legend("topright", "regression\nline", col = "royalblue2",
 lwd = 3, cex = 2)
```

```
and an equation
```

```
$$Y = \beta_0 + \sum_{i = 1}^p \beta_i Z_i$$
```

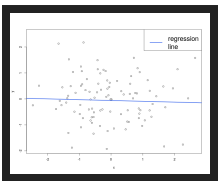
# A SIMPLE GRAPHIC

With a list

- item 1
- item 2

And some R code

```
x <- rnorm(100)
y <- rnorm(100)
plot(x, y, col = 1)
abline(lm(x~y), lwd = 3, col = "royalblue2")
legend("topright", "regression\nline", col = "royalblue2",
 lwd = 3, cex = 2)
```

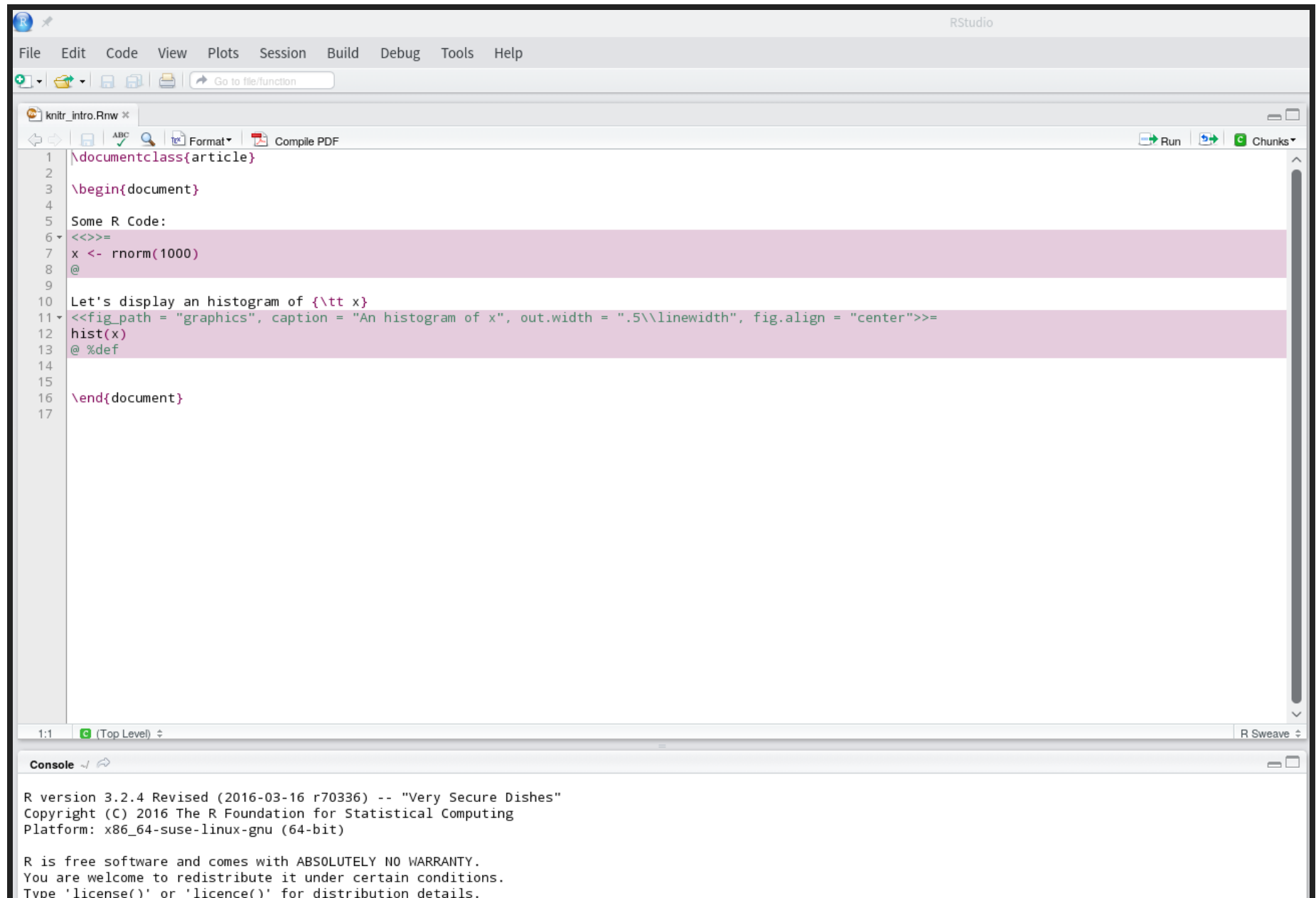


and an equation

$$Y = \beta_0 + \sum_{i=1}^p \beta_i Z_i$$



# KNITR AND RSTUDIO



The screenshot displays the RStudio environment with a Knitr Rnw file open. The editor window shows the following content:

```
1 \documentclass{article}
2
3 \begin{document}
4
5 Some R Code:
6 <<=>
7 x <- rnorm(1000)
8 @
9
10 Let's display an histogram of {\tt x}
11 <<fig_path = "graphics", caption = "An histogram of x", out.width = ".5\\linewidth", fig.align = "center">=
12 hist(x)
13 @ %def
14
15
16 \end{document}
17
```

The console window at the bottom shows the R startup message:

```
R version 3.2.4 Revised (2016-03-16 r70336) -- "Very Secure Dishes"
Copyright (C) 2016 The R Foundation for Statistical Computing
Platform: x86_64-suse-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.
```

```
type "contrib()") or "citation()" or "demo()".
```

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.

Error in tools:::httpdPort <= 0L :

comparison (4) is possible only for atomic and list types

> |