Package 'rfordummies'

August 18, 2014

Title Code examples to accompany the book ``R for Dummies''

Version 0.0-2

Description This package contains all the code examples in the book ``R for Dummies'' (1st edition)

Date 2014-08-18

URL http://rfordummies.com

BugReports https://github.com/andrie/rfordummies/issues

Suggests fortunes, stringr, sos, XLConnect, reshape2, ggplot2,foreign, lattice

License GPL-2 | GPL-3

LazyData true

Author Andrie de Vries [aut, cre],Joris Meys [aut]

Maintainer Andrie de Vries <apdevries@gmail.com>

NeedsCompilation no

Repository CRAN

Date/Publication 2014-08-18 11:11:22

R topics documented:

---

ch01   *Print examples of chapter 1 of 'R for Dummies'.*

Description

To print a listing of all examples of a chapter, use ch1(). To run all the examples of ch1(), use
example(ch1).

Usage

```
ch01()

ch1()
```

See Also

[toc](#)

Other Chapters: ch02, ch02, ch2; ch03, ch03, ch3; ch04, ch04, ch4; ch05, ch05, ch5; ch06, ch06,
ch6; ch07, ch07, ch7; ch08, ch08, ch8; ch09, ch09, ch9; ch10; ch11; ch12; ch13; ch14; ch15;
ch16; ch17; ch18; ch19; ch20

Examples

```
# Chapter 1 - Introducing R: The Big Picture

# Recognizing the Benefits of Using R

## It comes as free, open-source code

### It runs anywhere

### It supports extensions
```

```
### It provides an engaged community

### It connects with other languages


# Looking At Some of the Unique Features of R

## Performing multiple calculations with vectors

x <- 1:5
x
x + 2
x + 6:10

## Processing more than just statistics

## Running code without a compiler
```

---

ch02   *Print examples of chapter 2 of 'R for Dummies'.*

Description

To print a listing of all examples of a chapter, use ch2(). To run all the examples of ch2(), use example(ch2).

Usage

```
ch02()

ch2()
```

See Also

toc

Other Chapters: ch01, ch01, ch1; ch03, ch03, ch3; ch04, ch04, ch4; ch05, ch05, ch5; ch06, ch06, ch6; ch07, ch07, ch7; ch08, ch08, ch8; ch09, ch09, ch9; ch10; ch11; ch12; ch13; ch14; ch15; ch16; ch17; ch18; ch19; ch20

Examples

```
# Chapter 2 - Exploring R

# Working with a Code Editor
```

```
## Exploring RGui

### Seeing the naked R console


### Issuing a simple command

24+7+11

### Closing the console

## Not run:
quit()

## End(Not run)


## Dressing up with RStudio


# Starting Your First R Session

## Saying hello to the world

print("Hello world!")

## Doing simple math

1+2+3+4+5

## Using vectors


c(1,2,3,4,5)
1:5
sum(1:5)

## Storing and calculating values

x <- 1:5
x

y <- 10
x + y

x
y

z <- x + y
z

h <- "Hello"
h
```

```
hw <- c("Hello", "world!")
hw

paste("Hello", "world!")

## Talking back to the user

h <- "Hello"
yourname <- readline("What is your name?")
paste(h, yourname)

# Sourcing a Script

h <- "Hello"
yourname <- readline("What is your name?")
print(paste(h, yourname))


### Finding help on functions
?paste
help(paste)

# Navigating the Workspace
ls()

## Manipulating the content of the workspace

rm(z)
ls()

##Saving your work

getwd()
## Not run:
save(yourname, file="yourname.rda")

## End(Not run)

## Retrieving your work


rm(yourname)
## Not run:
load("yourname.rda")

## End(Not run)
```

---

ch03   *Print examples of chapter 3 of 'R for Dummies'.*

Description

To print a listing of all examples of a chapter, use ch3(). To run all the examples of ch3(), use example(ch3).

Usage

```
ch03()

ch3()
```

See Also

toc

Other Chapters: ch01, ch01, ch1; ch02, ch02, ch2; ch04, ch04, ch4; ch05, ch05, ch5; ch06, ch06, ch6; ch07, ch07, ch7; ch08, ch08, ch8; ch09, ch09, ch9; ch10; ch11; ch12; ch13; ch14; ch15; ch16; ch17; ch18; ch19; ch20

Examples

```
# Chapter 3 - The Fundamentals of R

# Using the Full Power of Functions

## Vectorizing your functions

baskets.of.Granny <- c(12,4,4,6,9,3)
baskets.of.Granny
sum(baskets.of.Granny)

firstnames <- c("Joris", "Carolien", "Koen")
lastname <- "Meys"
paste(firstnames,lastname)

authors <- c("Andrie","Joris")
lastnames <- c("de Vries","Meys")
paste(authors,lastnames)

## Putting the argument in a function

# print() ### This line of code leads to deliberate error for illustration
print(x = "Isn't this fun?")

print(digits=4, x = 11/7)

# Making history

## Not run:
savehistory(file = "Chapter3.Rhistory")

## End(Not run)
## Not run:
```

```
loadhistory("Chapter3.Rhistory")

## End(Not run)

# Keeping Your Code Readable

## Following naming conventions

## Choosing a clear name

paste <- paste("This gets","confusing")
paste
paste("Don't","you","think?")

## Choosing a naming style

## Structuring your code

baskets.of.Geraldine <- c(5,3,2,2,12,9)
Intro <- "It is amazing! The All Star Grannies scored
a total of"

Outro <- "baskets in the last six games!"

Total.baskets <- baskets.of.Granny +
                 baskets.of.Geraldine

Text <- paste(Intro,
              sum(Total.baskets),
              Outro)
cat(Text)
Text

cat('If you doubt whether it works,
+ just try it out.')

## Adding comments

# The All Star Grannies do it again!
baskets.of.Granny <- c(12,4,4,6,9,3) # Granny rules
sum(baskets.of.Granny) # total number of points


# Getting from Base R to More

## Finding packages

## Installing packages

## Not run:
install.packages("fortunes")

## End(Not run)
```

```
library("fortunes")
fortune("This is R")
fortune(161)
detach(package:fortunes)
```

---

ch04   *Print examples of chapter 4 of 'R for Dummies'.*

Description

To print a listing of all examples of a chapter, use ch4(). To run all the examples of ch4(), use
example(ch4).

Usage

```
ch04()
```

```
ch4()
```

See Also

toc

Other Chapters: ch01, ch01, ch1; ch02, ch02, ch2; ch03, ch03, ch3; ch05, ch05, ch5; ch06, ch06,
ch6; ch07, ch07, ch7; ch08, ch08, ch8; ch09, ch09, ch9; ch10; ch11; ch12; ch13; ch14; ch15;
ch16; ch17; ch18; ch19; ch20

Examples

```
# Chapter 4 - Getting Started with Arithmetic

# Working with Numbers, Infinity, and Missing Values

## Doing basic arithmetic

### Using arithmetic operators


baskets.of.Granny <- c(12,4,4,6,9,3)
baskets.of.Geraldine <- c(5,3,2,2,12,9)


Granny.money <- baskets.of.Granny * 120
Geraldine.money <- baskets.of.Geraldine * 145


Granny.money + Geraldine.money
```

```
baskets.of.Granny * 120 + baskets.of.Geraldine * 145

### Controlling the order of the operations
4 + 2 * 3
(4 + 2)* 3

## Using mathematical functions

### Calculating logarithms and exponentials

log(1:3)
log(1:3,base=6)

x <- log(1:3)
exp(x)

### Putting the science in scientific notation
1.33e4

4.12e-2

1.2e6 / 2e3


### Rounding numbers

round(123.456,digits=2)
round(-123.456,digits=-2)
signif(-123.456,digits=4)

### Using trigonometric functions

cos(120)
cos(120*pi/180)

## Calculating whole vectors

`+`(2,3)


##To infinity and beyond

### Using infinity

2/0
4 - Inf
is.finite(10^(305:310))

### Dealing with undefined outcomes
Inf / Inf
NaN + 4
```

```
### Dealing with missing values

x <- NA
x + 4

log(x)

is.na(x)

### Calculating infinite, undefined, and missing values


# Organizing Data in Vectors

## Discovering the properties of vectors

### Looking at the structure of a vector

str(baskets.of.Granny)
length(baskets.of.Granny)
authors <- c("Andrie", "Joris")
str(authors)

### Testing vector types



is.numeric(baskets.of.Granny)
is.integer(baskets.of.Granny)

x <- c(4L,6L)
is.integer(x)

## Creating vectors

seq(from = 4.5, to = 2.5, by = -0.5)


seq(from = -2.7, to = 1.3, length.out = 9)


baskets.of.Granny <- c(12,4,4,6,9,3)
baskets.of.Geraldine <- c(5,3,2,2,12,9)

## Combining vectors

all.baskets <-c(baskets.of.Granny, baskets.of.Geraldine)
all.baskets

## Repeating vectors
rep(c(0, 0, 7), times = 3)
```

```
rep(c(2, 4, 2), each = 3)
rep(c(0, 7), times = c(4,2))
rep(1:3,length.out=7)

# Getting Values in and out of Vectors

## Understanding indexing in R

numbers <- 30:1
numbers

## Extracting values from a vector

numbers[5]
numbers[c(5,11,3)]

indices <- c(5,11,3)
numbers[indices]
numbers[-3]
numbers[-(1:20)]
# numbers[-1:20] # NOT RUN, gives error


## Changing values in a vector


baskets.of.Granny[3] <- 5
baskets.of.Granny

baskets.of.Geraldine[c(2,4)] <- 4
baskets.of.Geraldine

Granny.copy <- baskets.of.Granny

baskets.of.Granny[4] <- 11
baskets.of.Granny

baskets.of.Granny <- Granny.copy
baskets.of.Granny

# Working with Logical Vectors

## Comparing values

baskets.of.Granny > 5
which(baskets.of.Granny > 5)

the.best <- baskets.of.Geraldine < baskets.of.Granny
which(the.best)


## Using logical vectors as indices
```

```
baskets.of.Granny[the.best]
x <- c(3, 6, 1, NA, 2)
x[x > 2]
x > 2
```

Combining logical statements

```
min.baskets <- baskets.of.Granny == min(baskets.of.Granny)
max.baskets <- baskets.of.Granny == max(baskets.of.Granny)
min.baskets | max.baskets
```

```
x[!is.na(x)]
```

Summarizing logical vectors

```
sum(the.best)
any(the.best)
all(the.best)
```

Powering Up Your Math with Vector Functions

Using arithmetic vector operations

Summarizing a vector
```
min(baskets.of.Granny)
```

```
max(baskets.of.Granny)
sum(baskets.of.Granny,baskets.of.Geraldine)
```

```
x <- c(3,6,2,NA,1)
sum(x)
sum(x,na.rm=TRUE)
```

Cumulating operations

```
cumsum(baskets.of.Granny)
cummax(baskets.of.Geraldine)
cummin(x)
```

Calculating differences

```
diff(baskets.of.Granny)
diff(x)
```

Recycling arguments

```
Granny.pointers <- c(10,2,4,0,4,1,4,2,7,2,1,2)
points <- Granny.pointers * c(2,3)
points
sum(points)

sum(Granny.pointers * c(2,3))



round(diff(baskets.of.Granny) / baskets.of.Granny * 100 )
round(diff(baskets.of.Granny) / baskets.of.Granny[1:5] * 100)
```

---

ch05   *Print examples of chapter 5 of 'R for Dummies'.*

Description

To print a listing of all examples of a chapter, use ch5(). To run all the examples of ch5(), use example(ch5).

Usage

```
ch05()

ch5()
```

See Also

toc

Other Chapters: ch01, ch01, ch1; ch02, ch02, ch2; ch03, ch03, ch3; ch04, ch04, ch4; ch06, ch06, ch6; ch07, ch07, ch7; ch08, ch08, ch8; ch09, ch09, ch9; ch10; ch11; ch12; ch13; ch14; ch15; ch16; ch17; ch18; ch19; ch20

Examples

```
# Chapter 5 - Getting Started with Reading and Writing

# Using Character Vectors for Text Data

## Assigning a value to a character vector

x <- "Hello world!"
is.character(x)
length(x)
nchar(x)

## Creating a character vector with more than one element
```

```
x <- c("Hello", "world!")
length(x)
nchar(x)

## Extracting a subset of a vector

letters
LETTERS
letters[10]
LETTERS[24:26]
tail(LETTERS, 5)
head(letters, 10)

## Naming the values in your vectors

### Looking at how named vectors work

str(islands)
islands[c("Asia", "Africa", "Antarctica")]
names(islands)[1:9]
names(sort(islands, decreasing=TRUE)[1:6])

## Creating and assigning named vectors

month.days <- c(31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31)
names(month.days) <- month.name
month.days
names(month.days[month.days==31])

# Manipulating Text

## String theory: Combining and splitting strings

### Splitting text

pangram <- "The quick brown fox jumps over the lazy dog"
pangram
strsplit(pangram, " ")

words <- strsplit(pangram, " ")[[1]]
words

### Changing text case

unique(tolower(words))
toupper(words[c(4, 9)])
tolower("Some TEXT in Mixed CASE")

### Concatenating text

paste("The", "quick", "brown", "fox")
paste(c("The", "quick", "brown", "fox"))
```

```
paste(words, collapse=" ")
paste(words, collapse="_")
paste(LETTERS[1:5], 1:5, sep="_", collapse="---")
paste("Sample", 1:5)
paste(c("A", "B"), c(1, 2, 3, 4), sep="-")
paste(c("A"), c(1, 2, 3, 4, 5), sep="-")
```

Sorting text

```
sort(letters, decreasing=TRUE)
sort(words)
```

Finding text inside text

Searching for individual words

```
head(state.name)
```

Searching by position

```
head(substr(state.name, start=3, stop=6))
```

Searching by pattern

```
grep("New", state.name)
state.name[29]
state.name[grep("New", state.name)]
state.name[grep("new", state.name)]
```

Searching for multiple words

```
state.name[grep(" ", state.name)]
state.name[grep("East", state.name)]
```

Substituting text

```
gsub("cheap", "sheep's", "A wolf in cheap clothing")
x <- c("file_a.csv", "file_b.csv", "file_c.csv")
y <- gsub("file_", "", x)
y
gsub(".csv", "", y)
```

Extending text functionality with stringr

```
## Not run:
install.packages("stringr")

## End(Not run)
library(stringr)
```

```
## Revving up with regular expressions

rwords <- c("bach", "back", "beech", "beach", "black")
grep("beach|beech", rwords)
rwords[grep("beach|beech", rwords)]
rwords[grep("be(a|e)ch", rwords)]
rwords[grep("b(e*|a*)ch", rwords)]


# Factoring in Factors

## Creating a factor

directions <- c("North", "East", "South", "South")
factor(directions)
factor(directions, levels= c("North", "East", "South", "West"))
factor(directions, levels= c("North", "East", "South", "West"), labels=c("N", "E", "S", "W"))

## Converting a factor

directions <- c("North", "East", "South", "South")
directions.factor <- factor(directions)
directions.factor
as.character(directions.factor)
as.numeric(directions.factor)

numbers <- factor(c(9, 8, 10, 8, 9))
as.character(numbers)
as.numeric(numbers)
as.numeric(as.character(numbers))

## Looking at levels

str(state.region)
levels(state.region)
levels(state.region) <- c("NE", "S", "NC", "W")
head(state.region)
nlevels(state.region)
length(levels(state.region))
levels(state.region)[2:3]

## Distinguishing data types

head(state.region)
table(state.region)
state.region

## Working with ordered factors

status <- c("Lo", "Hi", "Med", "Med", "Hi")
ordered.status <- factor(status, levels=c("Lo", "Med", "Hi"), ordered=TRUE)
ordered.status
table(status)
```

```
table(ordered.status)
```

---

ch06   *Print examples of chapter 6 of 'R for Dummies'.*

Description

To print a listing of all examples of a chapter, use ch6(). To run all the examples of ch6(), use example(ch6).

Usage

```
ch06()

ch6()
```

See Also

[toc](#)

Other Chapters: [ch01](#), [ch01](#), [ch1](#); [ch02](#), [ch02](#), [ch2](#); [ch03](#), [ch03](#), [ch3](#); [ch04](#), [ch04](#), [ch4](#); [ch05](#), [ch05](#), [ch5](#); [ch07](#), [ch07](#), [ch7](#); [ch08](#), [ch08](#), [ch8](#); [ch09](#), [ch09](#), [ch9](#); [ch10](#); [ch11](#); [ch12](#); [ch13](#); [ch14](#); [ch15](#); [ch16](#); [ch17](#); [ch18](#); [ch19](#); [ch20](#)

Examples

```
# Chapter 6 - Going on a Date with R

# Working with Dates

xd <- as.Date("2012-07-27")
xd
str(xd)
weekdays(xd)
xd + 7
xd + 0:6
weekdays(xd + 0:6)

startDate <- as.Date("2012-01-01")
xm <- seq(startDate, by="2 months", length.out=6)
xm
 months(xm)
quarters(xm)

Sys.localeconv()

as.Date("27 July 2012", format="%d %B %Y")
```

```
as.Date("27/7/12", format="%d/%m/%y")

# Adding Time Information to Dates

apollo <- "July 20, 1969, 20:17:39"
apollo.fmt <- "%B %d, %Y, %H:%M:%S"
xct <- as.POSIXct(apollo, format=apollo.fmt, tz="UTC")
xct

format(xct, "%d/%m/%y")
format(xct, "%S minutes past %I %p, on %d %B %Y")

# Performing Operations on Dates and Times

## Addition and subtraction

24*60*60
xct + 7*86400
xct + 3*60*60
xct - 7*86400
as.Date(xct) - 7

## Comparison of dates

Sys.time()
Sys.time() < xct

dec.start <- as.POSIXct("1950-01-01")
dec <- seq(dec.start, by="10 years", length.out=4)
dec
dec > xct

## Extraction

xlt <- as.POSIXlt(xct)
xlt
xlt$year
xlt$mon
unclass(xlt)
```

Description

To print a listing of all examples of a chapter, use ch7(). To run all the examples of ch7(), use example(ch7).

Usage

```
ch07()

ch7()
```

See Also

toc

Other Chapters: ch01, ch01, ch1; ch02, ch02, ch2; ch03, ch03, ch3; ch04, ch04, ch4; ch05, ch05, ch5; ch06, ch06, ch6; ch08, ch08, ch8; ch09, ch09, ch9; ch10; ch11; ch12; ch13; ch14; ch15; ch16; ch17; ch18; ch19; ch20

Examples

```
# Chapter 7
# Working in More Dimensions

# Adding a Second Dimension

## Discovering a new dimension

### Creating your first matrix

first.matrix <- matrix(1:12, ncol=4)
first.matrix
matrix(1:12, ncol=4, byrow=TRUE)

### Looking at the properties

str(first.matrix)
dim(first.matrix)
length(first.matrix)
my.array <- array(1:24, dim=c(3,4,2))
baskets.of.Granny <- c(12,4,5,6,9,3)
baskets.of.Geraldine <- c(5,4,2,4,12,9)
baskets.team <- rbind(baskets.of.Granny, baskets.of.Geraldine)

attributes(my.array)
attr(baskets.team,'season') <- '2010-2011'
attr(baskets.team,'season')
attr(baskets.team,'season') <- NULL

## Combining vectors into a matrix

baskets.of.Granny <- c(12,4,5,6,9,3)
baskets.of.Geraldine <- c(5,4,2,4,12,9)
```

```
baskets.team <- rbind(baskets.of.Granny, baskets.of.Geraldine)

baskets.team

cbind(1:3, 4:6, matrix(7:12, ncol=2))

# Using the Indices

## Extracting values from a matrix

### Using numeric indices

first.matrix[1:2, 2:3]
first.matrix[2:3,]

### Dropping values using negative indices

first.matrix[-2,-3]

nr <- nrow(first.matrix)
id <- nr*2+2
first.matrix[-id]

first.matrix[-(2 * nrow(first.matrix) + 2)]


### Juggling dimensions

first.matrix[-c(1, 3), ]
first.matrix[2, , drop=FALSE]

## Replacing values in a matrix

first.matrix[3, 2] <- 4
first.matrix

first.matrix[2, ] <- c(1,3)
first.matrix

first.matrix[1:2, 3:4] <- c(8,4,2,1)
first.matrix

# Naming Matrix Rows and Columns

## Changing the row and column names

rownames(baskets.team) <- c('Granny','Geraldine')
rownames(baskets.team)
colnames(baskets.team) <- c('1st','2nd','3th','4th','5th','6th')
baskets.team

colnames(baskets.team)[3] <- '3rd'
```

```
baskets.copy <- baskets.team
colnames(baskets.copy) <- NULL
baskets.copy

## Using names as indices

baskets.team[, c("2nd","5th")]

baskets.team["Granny",]

# Calculating with Matrices

## Using standard operations with matrices
first.matrix + 4

second.matrix <- matrix(1:3, nrow=3, ncol=4)

first.matrix + second.matrix

# first.matrix + second.matrix[,1:3] # gives error for illustration
# Error in first.matrix + second.matrix[, 1:3] : non-conformable arrays

first.matrix + 1:3

## Calculating row and column summaries

rowSums(baskets.team)

## Doing matrix arithmetic

### Transposing a matrix

t(first.matrix)

t(1:10)

t(first.matrix[2,])

### Inverting a matrix

square.matrix <- matrix(c(1,0,3,2,2,4,3,2,1),ncol=3)
solve(square.matrix)

### Multiplying two matrices

first.matrix %*% t(second.matrix)

first.matrix %*% 1:4
1:3 %*% first.matrix

# Adding More Dimensions

## Creating an array
```

```
### Using the creator functions

my.array <- array(1:24, dim=c(3,4,2))
my.array

### Changing the dimensions of a vector


my.vector <- 1:24
dim(my.vector) <- c(3,4,2)
identical(my.array, my.vector)

## Using dimensions to extract values

my.array[2,3,1]

my.array[, 3, 2, drop=FALSE]


my.array[2, , ]


# Combining Different Types of Values in a Data Frame

## Creating a data frame from a matrix

### Using the function as.data.frame

baskets.df <- as.data.frame(t(baskets.team))

### Looking at the structure of a data frame

baskets.df
str(baskets.df)

### Counting values and variables

nrow(baskets.df)
length(baskets.df)

## Creating a data frame from scratch

### Making a data frame from vectors

employee <- c('John Doe','Peter Gynn','Jolie Hope')
salary <- c(21000, 23400, 26800)
startdate <- as.Date(c('2010-11-1','2008-3-25','2007-3-14'))

employ.data <- data.frame(employee, salary, startdate)

str(employ.data)
```

```
### Keeping characters as characters

employ.data <- data.frame(employee, salary, startdate, stringsAsFactors=FALSE)
str(employ.data)

## Naming variables and observations

### Working with variable names

colnames(employ.data)
names(employ.data)

names(employ.data)[3] <- 'firstday'
names(employ.data)

### Naming observations

rownames(employ.data)
rownames(employ.data) <- c('Chef','BigChef','BiggerChef')
employ.data

# Manipulating Values in a Data Frame

## Extracting variables, observations, and values

### Pretending it's a matrix

baskets.df['3rd', 'Geraldine']
baskets.df[, 1]

str(baskets.df[, 1, drop=FALSE])

### Putting your dollar where your data is

baskets.df$Granny

## Adding observations to a data frame

### Adding a single observation

result <- rbind(baskets.df, c(7,4))
result

baskets.df <- rbind(baskets.df,'7th' = c(7,4))
baskets.df

### Adding a series of new observations using rbind

new.baskets <- data.frame(Granny=c(3,8),Geraldine=c(9,4))
rownames(new.baskets) <- c('8th','9th')
baskets.df <- rbind(baskets.df, new.baskets)

### Adding a series of values using indices
```

```
baskets.df[c('8th','9th'), ] <- matrix(c(3,8,9,4), ncol=2)
baskets.df[c('8th','9th'), ] <- c(3,8,9,4)
```

Adding variables to a data frame

Adding a single variable

```
baskets.of.Gabrielle <- c(11,5,6,7,3,12,4,5,9)
baskets.df$Gabrielle <- baskets.of.Gabrielle
```

```
head(baskets.df, 4)
```

Adding multiple variables using cbind

```
new.df <- data.frame(
   Gertrude  =  c(3,5,2,1,NA,3,1,1,4),
   Guinevere =  c(6,9,7,3,3,6,2,10,6)
)
```

```
head(cbind(baskets.df, new.df), 4)
```

Combining Different Objects in a List

Creating a list

Creating an unnamed list

```
baskets.list <- list(baskets.team, '2010-2011')
baskets.list
```

Creating a named list

```
baskets.nlist <- list(scores=baskets.team, season='2010-2011')
baskets.nlist
```

Playing with the names of elements

```
names(baskets.nlist)
```

Getting the number of elements

```
length(baskets.list)
```

Extracting elements from lists

Using [[]]

```
baskets.list[[1]]
baskets.nlist[['scores']]
```

Using []

```
baskets.list[-1]
baskets.nlist[names(baskets.nlist)=='season']

## Changing the elements in lists

### Changing the value of elements

baskets.nlist[[1]] <- baskets.df
baskets.nlist[['scores']] <- baskets.df
baskets.nlist$scores <- baskets.df

baskets.nlist[1] <- list(baskets.df)

baskets.list[1:2] <- list(baskets.df, '2009-2010')

### Removing elements

baskets.nlist[[1]] <- NULL
baskets.nlist$scores <- NULL
baskets.nlist['scores'] <- NULL

baskets.nlist <- list(scores=baskets.df, season='2010-2011')
baskets.nlist['scores'] <- list(NULL)
baskets.nlist

### Adding extra elements using indices

baskets.nlist$players <- c('Granny','Geraldine')
baskets.nlist[['players']] <- c('Granny','Geraldine')
baskets.nlist['players'] <- list(c('Granny','Geraldine'))

baskets.list[[3]] <- c('Granny','Geraldine')
baskets.list[3] <- list(c('Granny','Geraldine'))

### Combining lists


baskets.list <- list(baskets.team,'2010-2011')
players <- list(rownames(baskets.team))

c(baskets.list, players)

## Reading the output of str() for lists

str(baskets.list)

## Seeing the forest through the trees
```

Description

To print a listing of all examples of a chapter, use ch8(). To run all the examples of ch8(), use example(ch8).

Usage

```
ch08()
```

```
ch8()
```

See Also

toc

Other Chapters: ch01, ch01, ch1; ch02, ch02, ch2; ch03, ch03, ch3; ch04, ch04, ch4; ch05, ch05, ch5; ch06, ch06, ch6; ch07, ch07, ch7; ch09, ch09, ch9; ch10; ch11; ch12; ch13; ch14; ch15; ch16; ch17; ch18; ch19; ch20

Examples

```
# Chapter 8
# Putting the Fun in Functions

# Moving from Scripts to Functions

## Making the script

x <- c(0.458, 1.6653, 0.83112)
percent <- round(x * 100, digits = 1)
result <- paste(percent, "%", sep = "")
print(result)

# source('pastePercent.R') # Only after saving

## Transforming the script

addPercent <- function(x){
  percent <- round(x * 100, digits = 1)
  result <- paste(percent, "%", sep = "")
  return(result)
}

## Using the function

ls()

### Formatting the numbers

new.numbers <- c(0.8223, 0.02487, 1.62, 0.4)
addPercent(new.numbers)

### Playing with function objects
```

```
ppaste <- addPercent
ppaste

## Reducing the number of lines

### Returning values by default

# AddPercent function without last return - not written in book
addPercent <- function(x){
  percent <- round(x * 100, digits = 1)
  result <- paste(percent, "%", sep = "")
}

print( addPercent(new.numbers) )

addPercent <- function(x){
  percent <- round(x * 100, digits = 1)
  paste(percent, "%", sep = "")
}

addPercent <- function(x){
  if( !is.numeric(x) ) return(NULL)
  percent <- round(x * 100, digits = 1)
  paste(percent, "%", sep = "")
}

### Breaking the walls

odds <- function(x) x / (1-x)

odds(0.8)

addPercent <- function(x) paste(round(x * 100, digits = 1), "%", sep = "")

# Using Arguments the Smart Way

## Adding more arguments

percentages <- c(58.23, 120.4, 33)
addPercent(percentages/100)

### Adding the mult argument

addPercent <- function(x, mult){
  percent <- round(x * mult, digits = 1)
  paste(percent, "%", sep = "")
}

addPercent(percentages, mult = 1)

### Adding a default value
```

```
# addPercent(new.numbers) # Gives error for illustrative purposes
# Error in x * mult : 'mult' is missing

addPercent <- function(x, mult = 100){
  percent <- round(x * mult, digits = 1)
  paste(percent, "%", sep = "")
}

addPercent(new.numbers)

addPercent(percentages, 1)

## Conjuring tricks with dots

addPercent <- function(x, mult = 100, ...){
  percent <- round(x * mult, ...)
  paste(percent, "%", sep = "")
}

addPercent(new.numbers, digits = 2)
addPercent(new.numbers)


addPercent <- function(x, mult = 100, digits = 1){
  percent <- round(x * mult, digits = digits)
  paste(percent, "%", sep = "")
}

## Using functions as arguments

### Applying different ways of rounding

addPercent <- function(x, mult = 100, FUN = round, ...){
  percent <- FUN(x * mult, ...)
  paste(percent, "%", sep = "")
}

addPercent(new.numbers, FUN = signif, digits = 3)

### Using anonymous functions

profits <- c(2100, 1430, 3580, 5230)
rel.profit <- function(x) round(x / sum(x) * 100)
addPercent(profits,
               FUN = function(x) round(x / sum(x) * 100) )

addPercent(profits / sum(profits))

# Coping with Scoping



## Crossing the borders
```

```
### Creating a test case

x <- 1:5
test <- function(x){
  cat("This is x:", x, "\n")
  rm(x)
  cat("This is x after removing it:",x,"\n")
}

test(5:1)

### Searching the path

## Using internal functions

calculate.eff <- function(x, y, control){
  min.base <- function(z) z - mean(control)
  min.base(x) / min.base(y)
}


half <- c(2.23, 3.23, 1.48)
full <- c(4.85, 4.95, 4.12)
nothing <- c(0.14, 0.18, 0.56, 0.23)
calculate.eff(half, full, nothing)

# Dispatching to a Method

## Finding the methods behind the function

print

### Using methods with UseMethod

small.one <- data.frame(a = 1:2, b = 2:1)
print.data.frame(small.one)

### Using default methods

print.default(small.one)

## Doing it yourself

### Adapting the addPercent function

addPercent.character <- function(x){
  paste(x,"%",sep="")
}

# Not written out in the book - needed for rest code #
addPercent.numeric <- function(x, mult = 100, FUN = round, ...){
  percent <- FUN(x * mult, ...)
```

```
  paste(percent, "%", sep = "")
}

addPercent <- function(x,...){
  UseMethod("addPercent")
}

addPercent(new.numbers, FUN = floor)

addPercent(letters[1:6])

# Adding a default function

# addPercent(small.one) # Gives error on purpose
# Error in UseMethod("addPercent") :
#  no applicable method for 'addPercent' applied to an object of class "data.frame"

addPercent.default <- function(x){
  cat('You should try a numeric or character vector.\n')
}
```

---

ch09   *Print examples of chapter 9 of 'R for Dummies'.*

Description

To print a listing of all examples of a chapter, use ch9(). To run all the examples of ch9(), use example(ch9).

Usage

```
ch09()

ch9()
```

See Also

[toc](toc)

Other Chapters: [ch01](ch01), [ch01](ch01), [ch1](ch1); [ch02](ch02), [ch02](ch02), [ch2](ch2); [ch03](ch03), [ch03](ch03), [ch3](ch3); [ch04](ch04), [ch04](ch04), [ch4](ch4); [ch05](ch05), [ch05](ch05), [ch5](ch5); [ch06](ch06), [ch06](ch06), [ch6](ch6); [ch07](ch07), [ch07](ch07), [ch7](ch7); [ch08](ch08), [ch08](ch08), [ch8](ch8); [ch10](ch10); [ch11](ch11); [ch12](ch12); [ch13](ch13); [ch14](ch14); [ch15](ch15); [ch16](ch16); [ch17](ch17); [ch18](ch18); [ch19](ch19); [ch20](ch20)

Examples

```
# Chapter 9
# Controlling the Logical Flow

#Making Choices with if Statements
```

```
priceCalculator <- function(hours, pph=40){
    net.price <- hours * pph
    round(net.price)
}

priceCalculator <- function(hours, pph=40){
    net.price <- hours * pph
    if(hours > 100) {
      net.price <- net.price * 0.9
    }
    round(net.price)
}

priceCalculator(hours = 55)
priceCalculator(hours = 110)

priceCalculator <- function(hours, pph=40){
    net.price <- hours * pph
    if(hours > 100) net.price <- net.price * 0.9
    round(net.price)
}

?'if'
?"if"
?`if`

## Doing Something Else with an if...else Statement

priceCalculator <- function(hours, pph=40, public=TRUE){
    net.price <- hours * pph
    if(hours > 100) net.price <- net.price * 0.9
    if(public) {
      tot.price <- net.price * 1.06
    } else {
      tot.price <- net.price * 1.12
    }
    round(tot.price)
}

priceCalculator(25,public=TRUE)
priceCalculator(25,public=FALSE)

priceCalculator <- function(hours, pph=40, public=TRUE){
    net.price <- hours * pph
    if(hours > 100) net.price <- net.price * 0.9
    if(public) tot.price <- net.price * 1.06 else
                tot.price <- net.price * 1.12
    round(tot.price)
}

priceCalculator <- function(hours, pph=40, public=TRUE){
    net.price <- hours * pph
    if(hours > 100) net.price <- net.price * 0.9
```

```
    tot.price <- net.price * if(public) 1.06 else 1.12
    round(tot.price)
}

# Vectorizing Choices

## Looking at the problem

priceCalculator(c(25,110))
priceCalculator(110)
c(25, 110) > 100

## Choosing based on a logical vector

### Understanding how it works

ifelse(c(1,3) < 2.5 , 1:2 , 3:4)

### Trying it out

my.hours <- c(25,110)
my.hours * 40 * ifelse(my.hours > 100, 0.9, 1)

### Adapting the function

priceCalculator <- function(hours,pph=40,public){
    net.price <- hours * pph
    net.price <- net.price * ifelse(hours > 100 , 0.9, 1)
    tot.price <- net.price * ifelse(public, 1.06, 1.12)
    round(tot.price)
}

clients <- data.frame(
  hours = c(25, 110, 125, 40),
  public = c(TRUE,TRUE,FALSE,FALSE)
)

with(clients, priceCalculator(hours, public = public))

# Making Multiple Choices

## Chaining if...else statements


# Code example # NOT run
#if(client=='private'){
#  tot.price <- net.price * 1.12      # 12% VAT
#} else {
#  if(client=='public'){
#    tot.price <- net.price * 1.06    # 6% VAT
#  } else {
#    tot.price <- net.price * 1    # 0% VAT
#  }
```

```
#}

# Code example # NOT run
#if(client=='private'){
#    tot.price <- net.price * 1.12
#} else if(client=='public'){
#    tot.price <- net.price * 1.06
#} else {
#    tot.price <- net.price
#}

# Code example # NOT run
#VAT <- ifelse(client=='private', 1.12,
#          ifelse(client == 'public', 1.06, 1)
#       )
#tot.price <- net.price * VAT
#

## Switching between possibilities

### Making choices with switch

# Code example # NOT run
# VAT <- switch(client, private=1.12, public=1.06, abroad=1)



### Using default values in switch

# Code example # NOT run
# VAT <- switch(client, private=1.12, public=1.06, 1)

client <- 'other'
switch(client, private=1.12, public=1.06, 1)


# Looping Through Values

## Constructing a for loop

## Calculating values in a for loop

### Using the values of the vector

priceCalculator <- function(hours, pph=40, client){
    net.price <- hours * pph *
                    ifelse(hours > 100, 0.9, 1)

    VAT <- numeric(0)
    for(i in client){
      VAT <- c(VAT,switch(i, private=1.12, public=1.06, 1))
      }
```

```
    tot.price <- net.price * VAT
    round(tot.price)
}
```

```
clients$type <- c('public','abroad','private','abroad')
priceCalculator(clients$hours, client=clients$type)
```

Using loops and indices

```
nclient <- length(client)
VAT <- numeric(nclient)
for(i in seq_along(client)){
  VAT[i] <- switch(client[i], private=1.12, public=1.06, 1)
}
VAT
```

Looping without Loops: Meeting the Apply Family

```
songline <- 'Get out of my dreams...'
for(songline in 1:5) print('...Get into my car!')
```

```
songline
```

Looking at the family features

Meeting three of the members

Applying functions on rows and columns

Counting birds

```
counts <- matrix(c(3,2,4,6,5,1,8,6,1), ncol=3)
colnames(counts) <- c('sparrow','dove','crow')
counts
```

```
apply(counts, 2, max)
```

Adding extra arguments

```
counts[2, 2] <- NA
apply(counts,2,max)
apply(counts, 2, max, na.rm=TRUE)
```

Applying functions to listlike objects

Applying a function to a vector

Using switch on vectors

```
sapply(c('a','b'), switch, a='Hello', b='Goodbye')
```

```
#### Replacing a complete for loop with a single statement

priceCalculator <- function(hours, pph=40, client){
  net.price <- hours * pph * ifelse(hours > 100, 0.9, 1)

  VAT <- sapply(client, switch, private=1.12, public=1.06, 1)

  tot.price <- net.price * VAT
  round(tot.price)
}

### Applying a function to a data frame

sapply(clients,class)

### Simplifying results (or not) with sapply

sapply(clients, unique)

### Getting lists using lapply

sapply(clients[c(1,3), ], unique)

lapply(clients[c(1,3), ], unique)
```

---

ch10   *Print examples of chapter 10 of 'R for Dummies'.*

Description

To print a listing of all examples of a chapter, use ch10(). To run all the examples of ch10(), use example(ch10).

Usage

```
ch10()
```

See Also

[toc](toc)

Other Chapters: [ch01](ch01), [ch01](ch01), [ch1](ch1); [ch02](ch02), [ch02](ch02), [ch2](ch2); [ch03](ch03), [ch03](ch03), [ch3](ch3); [ch04](ch04), [ch04](ch04), [ch4](ch4); [ch05](ch05), [ch05](ch05), [ch5](ch5); [ch06](ch06), [ch06](ch06), [ch6](ch6); [ch07](ch07), [ch07](ch07), [ch7](ch7); [ch08](ch08), [ch08](ch08), [ch8](ch8); [ch09](ch09), [ch09](ch09), [ch9](ch9); [ch11](ch11); [ch12](ch12); [ch13](ch13); [ch14](ch14); [ch15](ch15); [ch16](ch16); [ch17](ch17); [ch18](ch18); [ch19](ch19); [ch20](ch20)

Examples

```
# Chapter 10
# Debugging Your Code
```

```
# NOTE : Much code is commented out, as they generate
# errors on purpose. Uncomment the code and run the
# line to see the error and try the debugging out

# Knowing What to Look For

# Reading Errors and Warnings

## Reading error messages

# "a" + 1
# Error in "a" + 1 : non-numeric argument to binary operator

# data.frame(1:10,10:1,)
# Error in data.frame(1:10, 10:1, ) : argument is missing, with no default

## Caring about warnings (or not)

x <- 1:10
y <- if (x < 5 ) 0 else 1

x <- 4
sqrt(x - 5)

plot(1:10, 10:1, color='green')


# Going Bug Hunting

## Calculating the logit

# checks input and does logit calculation
logit <- function(x){
  x <- ifelse(x < 0 | x > 1, "NA", x)
  log(x / (1 - x) )
}
# transforms percentage to number and calls logit
logitpercent <- function(x){
  x <- gsub("%", "", x)
  logit(as.numeric(x))
}


## Knowing where an error comes from

# logitpercent('50%')
# Error in 1 - x : non-numeric argument to binary operator

# traceback()

## Looking inside a function
```

```
### Telling R which function to debug

# debug(logit)
# logitpercent('50%')

### Stepping through the function

### Start browsing from within the function

logit <- function(x){
  x <- ifelse(x < 0 | x > 1, "NA", x)
  browser()
  log(x / (1 - x) )
}

# logit(50)

# Generating Your Own Messages

## Creating errors

logit <- function(x){
  if( any(x < 0 | x > 1) ) stop('x not between 0 and 1')
  log(x / (1 - x) )
}


# logitpercent(c('50%','150%'))
# Error in logit(as.numeric(x)/100) : x not between 0 and 1

## Creating warnings
# Function wrapped around for illustrative purposes
# In book only body is given
logit <- function(x){
  x <- ifelse(x < 0 | x > 1, NA, x )
  if( any(is.na(x)) ) warning('x not between 0 and 1')
  log(x / (1 - x) )
}

logitpercent(c('50%','150%'))

# Recognizing the Mistakes You're Sure to Make

## Starting with the wrong data

## Having your data in the wrong format


### Dropping dimensions when you don't expect it

rowsum.df <- function(x){
```

```
    id <- sapply(x,is.numeric)
    rowSums(x[, id])
}

# rowsum.df(sleep)

### Messing up with lists
strsplit('this is a sentence',' ')[2]

strsplit('this is a sentence',' ')

strsplit('this is a sentence',' ')[[1]][2]

customer <- c('Johan Delong','Marie Petit')
namesplit <- strsplit(customer,' ')

paste(namesplit[2],collapse='.')

paste(namesplit[[2]],collapse='.')


### Mixing up factors and numeric vectors

cyl.factor <- as.factor(mtcars$cyl)

median(as.numeric(cyl.factor))

as.numeric(levels(cyl.factor))[cyl.factor]
```

---

ch11   *Print examples of chapter 11 of 'R for Dummies'.*

Description

To print a listing of all examples of a chapter, use ch11(). To run all the examples of ch11(), use
example(ch11).

Usage

```
ch11()
```

See Also

[toc](toc)

Examples

```
# Chapter 11 - Getting Help


# Finding Information in the R Help Files

## When you know exactly what you're looking for

?date

## When you don't know exactly what you're looking for

??date


# Searching the Web for Help with R

## Not run:
RSiteSearch("cluster analysis")

## End(Not run)

## Not run:
install.packages("sos")

## End(Not run)
library("sos")
## Not run:
findFn("cluster")

## End(Not run)

# Getting Involved in the R Community

## Using the R mailing lists

## Discussing R on Stack Overflow and Stack Exchange

## Tweeting about R

# Making a Minimal Reproducible Example

dput(cars[1:4, ])

## Creating sample data with random values

set.seed(1)
x <- rnorm(5)
x

cards <- c(1:9, "J", "Q", "K", "A")
suits <- c("Spades", "Diamonds", "Hearts", "Clubs")
```

```
deck <- paste(rep(suits, each=13), cards)
set.seed(123)
sample(deck, 7)

set.seed(5)
sample(LETTERS[1:3], 12, replace=TRUE)

set.seed(42)
dat <- data.frame(
   x = sample(1:5),
   y = sample(c("yes", "no"), 5, replace = TRUE)
)
dat

dput(cars[1:4, ])

## Producing minimal code

## Providing the necessary information

sessionInfo()
```

---

ch12   *Print examples of chapter 12 of 'R for Dummies'.*

Description

To print a listing of all examples of a chapter, use ch12(). To run all the examples of ch12(), use example(ch12).

Usage

```
ch12()
```

See Also

[toc](toc)

Other Chapters: [ch01](ch01), [ch01](ch01), [ch1](ch1); [ch02](ch02), [ch02](ch02), [ch2](ch2); [ch03](ch03), [ch03](ch03), [ch3](ch3); [ch04](ch04), [ch04](ch04), [ch4](ch4); [ch05](ch05), [ch05](ch05), [ch5](ch5); [ch06](ch06), [ch06](ch06), [ch6](ch6); [ch07](ch07), [ch07](ch07), [ch7](ch7); [ch08](ch08), [ch08](ch08), [ch8](ch8); [ch09](ch09), [ch09](ch09), [ch9](ch9); [ch10](ch10); [ch11](ch11); [ch13](ch13); [ch14](ch14); [ch15](ch15); [ch16](ch16); [ch17](ch17); [ch18](ch18); [ch19](ch19); [ch20](ch20)

Examples

```
# Chapter 12
# Getting Data into and out of R

# NOTE : Most of the code depends on actions, directories
# and the presence of files. Code that isn't runnable is
# commented out.
```

```
# Getting Data into R

## Entering data in the R text editor

elements <- data.frame()
# elements <- edit(elements)

# print(elements)

## Using the Clipboard to copy and paste
# Reminder : This only works on Windows

## Not run:
# x <- readClipboard()

## End(Not run)
# x
## Not run:
# x <- readClipboard()

## End(Not run)
# x
# x <- read.table(file = "clipboard", sep = "\t", header=TRUE)
# x

## Reading data in CSV files

### Using read.csv() to import data

# elements <- read.csv(file.path("f:", "elements.csv"))
# str(elements)
# elements <- read.csv(file.path("f:", "elements.csv"), stringsAsFactors=FALSE)
# str(elements)

### Using read.table() to import tabular data in text files

## Reading data from Excel
## Not run:
# install.packages("XLConnect")

## End(Not run)
# library("XLConnect")
# excel.file <- file.path("~/Elements.xlsx")

# elements <- readWorksheetFromFile(excel.file, sheet=1)
# elements <- readWorksheetFromFile(excel.file, sheet="Sheet1")

## Working with other data types

# library(foreign)
# read.spss(file="location/of/myfile")
```

```
# Getting Your Data out of R


# writeClipboard(names(iris))

# write.table(head(iris), file="clipboard", sep="\t", row.names=FALSE)

# Working with Files and Folders

## Understanding the working directory
getwd()

# setwd("F:/git/roxygen2")
# getwd()
# setwd("F:\\git\\stringr")
# getwd()

file.path("f:", "git", "surveyor")

# setwd(file.path("F:", "git", "roxygen2"))
# getwd()

file.path("F:", "git", "roxygen2", "roxygen2", "README.md" )

## Manipulating files

# list.files(file.path("F:", "git", "roxygen2"))
my.file <- tempfile()
my.file
write.csv(iris, file=my.file)
list.files(tempdir())

file.iris <- read.csv(my.file)

file.remove(my.file)
list.files(tempdir())
```

---

ch13   *Print examples of chapter 13 of 'R for Dummies'.*

Description

To print a listing of all examples of a chapter, use ch13(). To run all the examples of ch13(), use example(ch13).

Usage

```
ch13()
```

See Also

toc

Other Chapters: ch01, ch01, ch1; ch02, ch02, ch2; ch03, ch03, ch3; ch04, ch04, ch4; ch05, ch05, ch5; ch06, ch06, ch6; ch07, ch07, ch7; ch08, ch08, ch8; ch09, ch09, ch9; ch10; ch11; ch12; ch14; ch15; ch16; ch17; ch18; ch19; ch20

Examples

```
# C hapter 13 - Manipulating and Processing Data

# Deciding on the Most Appropriate Data Structure

# Creating Subsets of Your Data

## Understanding the three subset operators
## Understanding the five ways of specifying the subset

str(islands)
islands[]
islands[c(8, 1, 1, 42)]
islands[-(3:46)]
islands[islands < 20]
islands[c("Madagascar", "Cuba")]

## Subsetting data frames

str(iris)
iris[1:5, ]
iris[, c("Sepal.Length", "Sepal.Width")]
iris[, 'Sepal.Length']
iris[, 'Sepal.Length', drop=FALSE]
iris['Sepal.Length']
iris[1:5, c("Sepal.Length", "Sepal.Width")]

### Taking samples from data

sample(1:6, 10, replace=TRUE)

set.seed(1)
sample(1:6, 10, replace=TRUE)
sample(1:6, 10, replace=TRUE)

set.seed(1)
sample(1:6, 10, replace=TRUE)

set.seed(123)
index <- sample(1:nrow(iris), 5)
index
iris[index, ]

### Removing duplicate data
```

```
duplicated(c(1,2,1,3,1,4))
duplicated(iris)
which(duplicated(iris))
iris[!duplicated(iris), ]

index <- which(duplicated(iris))
iris[-index, ]

### Removing rows with missing data

str(airquality)
complete.cases(airquality)

x <- airquality[complete.cases(airquality), ]
str(x)
x <- na.omit(airquality)



# Adding Calculated Fields to Data

## Doing arithmetic on columns of a data frame

x <- iris$Sepal.Length / iris$Sepal.Width
head(x)

## Using with and within to improve code readability

y <- with(iris, Sepal.Length / Sepal.Width)
head(y)
identical(x, y)

iris$ratio <- iris$Sepal.Length / iris$Sepal.Width
iris <- within(iris, ratio <- Sepal.Length / Sepal.Width)
head(iris$ratio)

## Creating subgroups or bins of data

### Using cut to create a fixed number of subgroups

head(state.x77)
frost <- state.x77[, "Frost"]
head(frost, 5)
cut(frost, 3, include.lowest=TRUE)

### Adding labels to cut

cut(frost, 3, include.lowest=TRUE, labels=c("Low", "Med", "High"))

### Using table to count the number of observations

x <- cut(frost, 3, include.lowest=TRUE, labels=c("Low", "Med", "High"))
```

```
table(x)
x


# Combining and Merging Data Sets

## Creating sample data to illustrate merging

all.states <- as.data.frame(state.x77)
all.states$Name <- rownames(state.x77)
rownames(all.states) <- NULL
str(all.states)

### Creating a subset of cold states

cold.states <- all.states[all.states$Frost>150, c("Name", "Frost")]
cold.states

### Creating a subset of large states

large.states <- all.states[all.states$Area>=100000, c("Name", "Area")]
large.states

## Using the merge() function

### Using merge to find the intersection of data

merge(cold.states, large.states)

### Understanding the different types of merge

merge(cold.states, large.states, all=TRUE)


## Working with lookup tables

### Finding a match

index <- match(cold.states$Name, large.states$Name)
index

large.states[na.omit(index), ]

### Making sense of %in%

index <- cold.states$Name %in% large.states$Name
index
!is.na(match(cold.states$Name,large.states$Name))
cold.states[index, ]

# Sorting and Ordering Data

some.states <- data.frame(
```

```
      Region = state.region,
      state.x77)

some.states <- some.states[1:10, 1:3]
some.states
```

Sorting vectors

Sorting a vector in ascending order

```
sort(some.states$Population)
```

Sorting a vector in decreasing order

```
sort(some.states$Population, decreasing=TRUE)
```

Sorting data frames

Getting the order

```
order.pop <- order(some.states$Population)
order.pop

some.states$Population[order.pop]
```

Sorting a data frame in ascending order

```
some.states[order.pop, ]
order(some.states$Population)
order(some.states$Population, decreasing=TRUE)

some.states[order(some.states$Population, decreasing=TRUE), ]
```

Sorting on more than one column

```
index <- with(some.states, order(Region, Population))
some.states[index, ]
```

Sorting multiple columns in mixed order
```
index <- order(-xtfrm(some.states$Region), some.states$Population)
some.states[index, ]
```

Traversing Your Data with the Apply Functions

Using the apply() function to summarize arrays

```
str(Titanic)
apply(Titanic, 1, sum)
apply(Titanic, 3, sum)
apply(Titanic, c(3, 4), sum)
```

Using lapply() and sapply() to traverse a list or data frame

```
lapply(iris, class)
sapply(iris, class)
sapply(iris, mean)
sapply(iris, function(x) ifelse(is.numeric(x), mean(x), NA))

## Using tapply() to create tabular summaries

tapply(iris$Sepal.Length, iris$Species, mean)
with(iris, tapply(Sepal.Length, Species, mean))

### Using tapply() to create higher-dimensional tables

str(mtcars)
cars <- within(mtcars,
    am <- factor(am, levels=0:1, labels=c("Automatic", "Manual"))
)

with(cars, tapply(mpg, am, mean))
with(cars, tapply(mpg, list(gear, am), mean))

### Using aggregate()

with(cars, aggregate(mpg, list(gear=gear, am=am), mean))

# Getting to Know the Formula Interface


aggregate(mpg ~ gear + am, data=cars, mean)

aov(mpg ~ gear + am, data=cars)

library(lattice)
xyplot(mpg ~ gear + am, data=cars)


# Whipping Your Data into Shape


## Understanding data in long and wide format


## Getting started with the reshape2 package

## Not run:
install.packages("reshape2")

## End(Not run)
library("reshape2")

goals <- data.frame(
    Game = c("1st", "2nd", "3rd", "4th"),
    Venue = c("Bruges", "Ghent", "Ghent", "Bruges"),
    Granny = c(12, 4, 5, 6),
```

```
    Geraldine = c(5, 4, 2, 4),
    Gertrude = c(11, 5, 6, 7)
)

## Melting data to long format

mgoals <- melt(goals)
mgoals <- melt(goals, id.vars=c("Game", "Venue"))
mgoals

## Casting data to wide format

dcast(mgoals,  Venue + Game ~ variable, sum)
dcast(mgoals, variable ~ Venue , sum)
dcast(mgoals,  Venue ~ variable , sum)

dcast(mgoals,  Venue + variable ~ Game , sum)

library(ggplot2)
ggplot(mgoals, aes(x=variable, y=value, fill=Game)) + geom_bar()
```

---

ch14   *Print examples of chapter 14 of 'R for Dummies'.*

Description

To print a listing of all examples of a chapter, use ch14(). To run all the examples of ch14(), use example(ch14).

Usage

```
ch14()
```

See Also

toc

Other Chapters: ch01, ch01, ch1; ch02, ch02, ch2; ch03, ch03, ch3; ch04, ch04, ch4; ch05, ch05, ch5; ch06, ch06, ch6; ch07, ch07, ch7; ch08, ch08, ch8; ch09, ch09, ch9; ch10; ch11; ch12; ch13; ch15; ch16; ch17; ch18; ch19; ch20

Examples

```
# Chapter 14
# Summarizing Data

# Starting with the Right Data

## Using factors or numeric data
```

```
## Counting unique values

sapply(mtcars, function(x) length(unique(x)))

## Preparing the data

cars <- mtcars[c(1,2,9,10)]
cars$gear <- ordered(cars$gear)
cars$am <- factor(cars$am, labels=c('auto', 'manual'))
str(cars)

# Describing Continuous Variables

## Talking about the center of your data

mean(cars$mpg)
median(cars$cyl)

## Describing the variation
sd(cars$mpg)

## Checking the quantiles

### Calculating the range
range(cars$mpg)

### Calculating the quantiles
quantile(cars$mpg)

### Getting on speed with the quantile function
quantile(cars$mpg, probs=c(0.05, 0.95))

# Describing Categories

## Counting appearances

### Creating a table
amtable <- table(cars$am)
amtable

### Working with tables

## Calculating proportions
amtable/sum(amtable)
prop.table(amtable)

## Finding the center
id <- amtable == max(amtable)
names(amtable)[id]

# Describing Distributions

## Plotting histograms
```

```
### Making the plot
hist(cars$mpg, col='grey')

### Playing with breaks
hist(cars$mpg, breaks=c(5,15,25,35))

## Using frequencies or densities

### Creating a density plot

mpgdens <- density(cars$mpg)
plot(mpgdens)

### Plotting densities in a histogram
hist(cars$mpg, col='grey', freq=FALSE)
lines(mpgdens)

# Describing Multiple Variables

## Summarizing a complete dataset

### Getting the output
summary(cars)

### Fixing a problem

cars$cyl <- as.factor(cars$cyl)

## Plotting quantiles for subgroups

boxplot(mpg ~ cyl, data=cars)

## Tracking correlations

names(iris)

### Looking at relations
plot(iris[-5])

### Getting the numbers

with(iris, cor(Petal.Width, Petal.Length))

### Calculating correlations for multiple variables

iris.cor <- cor(iris[-5])
str(iris.cor)

iris.cor['Petal.Width', 'Petal.Length']

### Dealing with missing values
```

```
# Working with Tables

## Creating a two-way table

### Creating a table from two variables

with(cars, table(am, gear))

### Creating tables from a matrix

trial <- matrix(c(34,11,9,32), ncol=2)
colnames(trial) <- c('sick', 'healthy')
rownames(trial) <- c('risk', 'no_risk')
trial.table <- as.table(trial)
trial.table

### Extracting the numbers

trial.table['risk', 'sick']

##Converting tables to a data frame

trial.df <- as.data.frame(trial)
str(trial.df)

trial.table.df <- as.data.frame(trial.table)
str(trial.table.df)

## Looking at margins and proportions

### Adding margins to the table

addmargins(trial.table)
addmargins(trial.table,margin=2)

### Calculating proportions

prop.table(trial.table)

### Calculating proportions over columns and rows
prop.table(trial.table, margin=1)
```

---

ch15 *Print examples of chapter 15 of 'R for Dummies'.*

Description

To print a listing of all examples of a chapter, use ch15(). To run all the examples of ch15(), use example(ch15).

Usage

```
ch15()
```

See Also

[toc](toc)

Other Chapters: [ch01](ch01), [ch01](ch01), [ch1](ch1); [ch02](ch02), [ch02](ch02), [ch2](ch2); [ch03](ch03), [ch03](ch03), [ch3](ch3); [ch04](ch04), [ch04](ch04), [ch4](ch4); [ch05](ch05), [ch05](ch05), [ch5](ch5); [ch06](ch06), [ch06](ch06), [ch6](ch6); [ch07](ch07), [ch07](ch07), [ch7](ch7); [ch08](ch08), [ch08](ch08), [ch8](ch8); [ch09](ch09), [ch09](ch09), [ch9](ch9); [ch10](ch10); [ch11](ch11); [ch12](ch12); [ch13](ch13); [ch14](ch14); [ch16](ch16); [ch17](ch17); [ch18](ch18); [ch19](ch19); [ch20](ch20)

Examples

```
# Chapter 15
# Testing Differences and Relations

# Taking a Closer Look at Distributions

## Observing beavers
str(beaver2)

## Testing normality graphically
library(lattice)
histogram(~temp | factor(activ), data=beaver2)

## Using quantile plots

### Comparing two samples

qqplot(beaver2$temp[beaver2$activ==1],
       beaver2$temp[beaver2$activ==0])

### Using a QQ plot to check for normality

qqnorm( beaver2$temp[beaver2$activ==0], main='Inactive')
qqline( beaver2$temp[beaver2$activ==0] )

## Testing normality in a formal way

shapiro.test(beaver2$temp)
result <- shapiro.test(beaver2$temp)
result$p.value

with(beaver2, tapply(temp, activ, shapiro.test))

# Comparing Two Samples

## Testing differences

### Carrying out a t-test

t.test(temp ~ activ, data=beaver2)
```

```
activetemp <- beaver2$temp[beaver2$activ==1]
inactivetemp <- beaver2$temp[beaver2$activ==0]
t.test(activetemp, inactivetemp)

### Dropping assumptions

wilcox.test(temp ~ activ, data=beaver2)

### Testing direction

## Comparing paired data

t.test(extra ~ group, data=sleep, paired=TRUE)

# Testing Counts and Proportions

## Checking out proportions
survivors <- matrix(c(1781,1443,135,47), ncol=2)
colnames(survivors) <- c('survived','died')
rownames(survivors) <- c('no seat belt','seat belt')
survivors

result.prop <- prop.test(survivors)
result.prop

## Analyzing tables

### Testing contingency of tables
chisq.test(survivors)

### Testing tables with more than two columns
str(HairEyeColor)
HairEyeMargin <- margin.table(HairEyeColor, margin=c(1,2))
HairEyeMargin

chisq.test(HairEyeMargin)

## Extracting test results
str(result)
t.test(temp ~ activ, data=beaver2)$p.value

# Working with Models

## Analyzing variances
str(InsectSprays)

### Building the model
AOVModel <- aov(count ~ spray, data=InsectSprays)

### Looking at the object
AOVModel
```

```
## Evaluating the differences
summary(AOVModel)

### Checking the model tables
model.tables(AOVModel, type='effects')

### Looking at the individual differences
Comparisons <- TukeyHSD(AOVModel)
Comparisons$spray['D-C',]

### Plotting the differences
plot(Comparisons, las=1)

## Modeling linear relations

### Building a linear model
Model <- lm(mpg ~ wt, data=mtcars)

### Extracting information from the model

coef.Model <- coef(Model)
coef.Model

plot(mpg ~ wt, data = mtcars)
abline(a=coef.Model[1], b=coef.Model[2])

## Evaluating linear models

### Summarizing the model
Model.summary <- summary(Model)
Model.summary

coef(Model.summary)

### Testing the impact of model terms
Model.anova <- anova(Model)
Model.anova

Model.anova['wt','Pr(>F)']

## Predicting new values

### Getting the values
new.cars <- data.frame(wt=c(1.7, 2.4, 3.6))
predict(Model, newdata=new.cars)

### Having confidence in your predictions
predict(Model, newdata=new.cars, interval='confidence')
predict(Model,newdata=new.cars, interval='prediction')
```

---

ch16   *Print examples of chapter 16 of 'R for Dummies'.*

Description

To print a listing of all examples of a chapter, use ch16(). To run all the examples of ch16(), use example(ch16).

Usage

```
ch16()
```

See Also

toc

Other Chapters: ch01, ch01, ch1; ch02, ch02, ch2; ch03, ch03, ch3; ch04, ch04, ch4; ch05, ch05, ch5; ch06, ch06, ch6; ch07, ch07, ch7; ch08, ch08, ch8; ch09, ch09, ch9; ch10; ch11; ch12; ch13; ch14; ch15; ch17; ch18; ch19; ch20

Examples

```
# Chapter 16 - Using Base Graphics

# Creating Different Types of Plots


## Getting an overview of plot

large.islands <- head(sort(islands, decreasing=TRUE), 10)

plot(large.islands, main="Land area of continents and islands",
   ylab="Land area in square miles")
text(large.islands, labels=names(large.islands), adj=c(0.5, 1))


## Adding points and lines to a plot

plot(faithful)

## Adding points

short.eruptions <- with(faithful, faithful[eruptions < 3, ])

plot(faithful)
points(short.eruptions, col="red", pch=19)

## Changing the shape of points

## Changing the color
```

```
head(colors(), 10)

## Adding lines to a plot

fit <- lm(waiting~eruptions, data=faithful)

plot(faithful)
lines(faithful$eruptions, fitted(fit), col="blue")
abline(v=3, col="purple")

abline(h=mean(faithful$waiting))
abline(a=coef(fit)[1], b=coef(fit)[2])
abline(fit, col = "red")

# Different plot types

plot(LakeHuron, type="l", main='type="l"')
plot(LakeHuron, type="p", main='type=p"')
plot(LakeHuron, type="b", main='type="b"')


x <- seq(0.5, 1.5, 0.25)
y <- rep(1, length(x))
plot(x, y, type="n")
points(x, y)

with(mtcars, plot(mpg, disp))
with(mtcars, boxplot(disp, mpg))
with(mtcars, hist(mpg))

# Controlling Plot Options and Arguments

## Adding titles and axis labels

plot(faithful,
    main = "Eruptions of Old Faithful",
    xlab = "Eruption time (min)",
    ylab = "Waiting time to next eruption (min)")


## Changing plot options

### The axes label style

plot(faithful, las=1)

### The box type

plot(faithful, bty="n")

### More than one option
```

```
plot(faithful, las=1, bty="l", col="red", pch=19)

### Font size of text and axes

x <- seq(0.5, 1.5, 0.25)
y <- rep(1, length(x))
plot(x, y, main="Effect of cex on text size")
text(x, y+0.1, labels=x, cex=x)

plot(x, y, main="Effect of cex.main, cex.lab and cex.axis",
  cex.main=1.25, cex.lab=1.5, cex.axis=0.75)

## Putting multiple plots on a single page

old.par <- par(mfrow=c(1, 2))
plot(faithful, main="Faithful eruptions")
plot(large.islands, main="Islands", ylab="Area")
par(old.par)


# Saving Graphics to Image Files

setwd("~/")
getwd()
## Not run:
png(filename="faithful.png")

## End(Not run)
plot(faithful)
## Not run:
dev.off()

## End(Not run)
```

---

ch17   *Print examples of chapter 17 of 'R for Dummies'.*

Description

To print a listing of all examples of a chapter, use ch17(). To run all the examples of ch17(), use
example(ch17).

Usage

```
ch17()
```

See Also

toc

Other Chapters: ch01, ch01, ch1; ch02, ch02, ch2; ch03, ch03, ch3; ch04, ch04, ch4; ch05, ch05, ch5; ch06, ch06, ch6; ch07, ch07, ch7; ch08, ch08, ch8; ch09, ch09, ch9; ch10; ch11; ch12; ch13; ch14; ch15; ch16; ch18; ch19; ch20

Examples

```
# Chapter 17 - Creating Faceted Graphics with Lattice


# Creating a Lattice Plot

str(mtcars)

## Loading the lattice package

library("lattice")

## Making a lattice scatterplot

xyplot(mpg ~ hp | factor(cyl), data=mtcars)

## Adding trend lines

xyplot(mpg ~ hp | factor(cyl), data=mtcars,
    type=c("p", "r"))


# Changing Plot Options

## Adding titles and labels

xyplot(mpg ~ hp | factor(cyl), data=mtcars,
    type=c("p", "r"),
    main="Fuel economy vs. Performance",
    xlab="Performance (horse power)",
    ylab="Fuel economy (miles per gallon)",
)

xyplot(mpg ~ hp | factor(cyl), data=mtcars,
    type=c("p", "r"),
    main=list(
        label="Fuel economy vs. Performance given Number of Cylinders",
        cex=0.75)
)

## Changing the font size of titles and labels

xyplot(mpg ~ hp | factor(cyl), data=mtcars,
    type=c("p", "r"),
```

```
    main=list(
        label="Fuel economy vs. Performance given Number of Cylinders",
        cex=0.75),
    xlab=list(
        label="Performance (horse power)",
        cex=0.75),
    ylab=list(
        label="Fuel economy (miles per gallon)",
        cex=0.75),
    scales=list(cex=0.5)
)


## Using themes to modify plot options

xyplot(mpg ~ hp | factor(cyl), data=mtcars,
    type=c("p", "r"),
    par.settings=simpleTheme(col="red", col.line="blue")
)


# Plotting Different Types

## Making a bar chart

mtcars$cars <- rownames(mtcars)

barchart(cars ~ mpg | factor(cyl), data=mtcars,
    main="barchart",
    scales=list(cex=0.5),
    layout=c(3, 1)
)

## Making a box-and-whisker plot

bwplot(~ hp | factor(cyl), data=mtcars, main="bwplot")


# Plotting Data in Groups

## Using data in tall format

str(longley)
library("reshape2")
mlongley <- melt(longley, id.vars="Year")

str(mlongley)

xyplot(value ~ Year | variable, data=mlongley,
    layout=c(6, 1),
    par.strip.text=list(cex=0.7),
    scales=list(cex=0.7)
)
```

```
## Creating a chart with groups

mtcars$cars <- rownames(mtcars)
mtcars$am <- with(mtcars, ifelse(am==0, "Automatic", "Manual"))

barchart(cars ~ mpg | factor(cyl), data=mtcars,
    group=am,
    scales=list(cex=0.5),
    layout=c(3, 1),
)

## Adding a key

barchart(cars ~ mpg | factor(cyl), data=mtcars,
    main="barchart with groups",
    group=am,
    auto.key=TRUE,
    par.settings = simpleTheme(col=c("grey80", "grey20")),
    scales=list(cex=0.5),
    layout=c(3, 1)
)


# Printing and Saving a Lattice Plot

## Assigning a lattice plot to an object

my.plot <- xyplot(mpg ~ hp | cyl, data=mtcars)
class(my.plot)

## Printing a lattice plot in a script

xyplot(mpg ~ hp | cyl, data=mtcars)

my.plot <- xyplot(mpg ~ hp | cyl, data=mtcars)
print(my.plot)


## Saving a lattice plot to file

setwd("~/")
## Not run:
trellis.device(device="png", filename="xyplot.png")

## End(Not run)
print(my.plot)
## Not run:
dev.off()

## End(Not run)
```

ch18	*Print examples of chapter 18 of 'R for Dummies'.*

Description

To print a listing of all examples of a chapter, use ch18(). To run all the examples of ch18(), use example(ch18).

Usage

```
ch18()
```

See Also

toc

Other Chapters: ch01, ch01, ch1; ch02, ch02, ch2; ch03, ch03, ch3; ch04, ch04, ch4; ch05, ch05, ch5; ch06, ch06, ch6; ch07, ch07, ch7; ch08, ch08, ch8; ch09, ch09, ch9; ch10; ch11; ch12; ch13; ch14; ch15; ch16; ch17; ch19; ch20

Examples

```
# Chapter 18 - Looking At ggplot2 Graphics

# Installing and Loading ggplot2

## Not run:
install.packages("ggplot2")

## End(Not run)
library("ggplot2")

# Looking At Layers

ggplot(faithful, aes(x=eruptions, y=waiting)) + geom_point() + stat_smooth()

# Using Geoms and Stats

## Defining what data to use

## Mapping data to plot aesthetics

ggplot(faithful, aes(x=eruptions, y=waiting)) + geom_point() + stat_smooth()

## Getting geoms

### Creating a bar chart

ggplot(quakes, aes(x=depth)) + geom_bar()
ggplot(quakes, aes(x=depth)) + geom_bar(binwidth=50)
```

```
quakes.agg <- aggregate(mag ~ round(depth, -1), data=quakes, FUN=length)
names(quakes.agg) <- c("depth", "mag")

ggplot(quakes.agg, aes(x=depth, y=mag)) +
   geom_bar(stat="identity")
```

Making a scatterplot

```
ggplot(quakes, aes(x=long, y=lat)) + geom_point()
```

Creating line charts

```
ggplot(longley, aes(x=Year, y=Unemployed)) + geom_line()
```

Sussing Stats

Binning data

```
ggplot(quakes, aes(x=depth)) + geom_bar(binwidth=50)
ggplot(quakes, aes(x=depth)) + stat_bin(binwidth=50)
```

Smoothing data

```
ggplot(longley, aes(x=Year, y=Employed)) + geom_point()

ggplot(longley, aes(x=Year, y=Employed)) +
   geom_point() + stat_smooth()

ggplot(longley, aes(x=Year, y=Employed)) +
   geom_point() + stat_smooth(method="lm")
```

Adding Facets, Scales, and Options

Adding facets

```
ggplot(mtcars, aes(x=hp, y=mpg)) + geom_point()

ggplot(mtcars, aes(x=hp, y=mpg)) + geom_point() +
   stat_smooth(method="lm") + facet_grid(~cyl)


ggplot(mtcars, aes(x=hp, y=mpg)) +
   geom_point(aes(shape=factor(cyl), colour=factor(cyl)))

ggplot(mtcars, aes(x=hp, y=mpg)) +
   geom_point(aes(shape=factor(cyl), colour=factor(cyl))) +
   scale_shape_discrete(name="Cylinders") +
   scale_colour_discrete(name="Cylinders")
```

```
## Changing options

ggplot(mtcars, aes(x=hp, y=mpg)) + geom_point(color="red") +
    xlab("Performance (horse power") +
    ylab("Fuel consumption (mpg)") +
    ggtitle("Motor car comparison")
```

---

ch19   *Print examples of chapter 19 of 'R for Dummies'.*

Description

To print a listing of all examples of a chapter, use ch19(). To run all the examples of ch19(), use
example(ch19).

Usage

```
ch19()
```

See Also

toc

Other Chapters: ch01, ch01, ch1; ch02, ch02, ch2; ch03, ch03, ch3; ch04, ch04, ch4; ch05, ch05,
ch5; ch06, ch06, ch6; ch07, ch07, ch7; ch08, ch08, ch8; ch09, ch09, ch9; ch10; ch11; ch12;
ch13; ch14; ch15; ch16; ch17; ch18; ch20

Examples

```
# Chapter 19 - Ten Things You Can Do in R That You Would've Done in Microsoft Excel

# Adding Row and Column Totals

iris.num <- iris[, -5]

colSums(iris.num)
colMeans(iris.num)

apply(iris.num, 2, min)
apply(iris.num, 2, max)

sapply(iris.num, min)
sapply(iris.num, max)


# Formatting Numbers

format(12345.6789, digits=9, decimal.mark=",",
```

```
      big.mark=" ",small.mark=".", , small.interval=3)

x <- colMeans(mtcars[, 1:4])
format(x, digits=2, nsmall=2)

x <- seq(0.5, 0.55, 0.01)
sprintf("%.1f %%", 100*x)

set.seed(1)
x <- 1000*runif(5)
sprintf("$ %3.2f", x)

stuff <- c("bread", "cookies")
price <- c(2.1, 4)
sprintf("%s costed $ %3.2f ", stuff, price)


# Sorting Data

with(mtcars, mtcars[order(hp), ])
with(mtcars, mtcars[order(hp, decreasing=TRUE), ])

# Making Choices with If

mtcars <- within(mtcars,
   mpgClass <- ifelse(mpg < mean(mpg), "Low", "High"))

mtcars[mtcars$mpgClass == "High", ]


# Calculating Conditional Totals

with(mtcars, mean(mpg))
with(mtcars, mean(mpg[hp < 150]))
with(mtcars, mean(mpg[hp >= 150]))
with(mtcars, length(mpg[hp > 150]))


# Transposing Columns or Rows

x <- matrix(1:12, ncol=3)
x
t(x)

t(mtcars[1:4, ])


# Finding Unique or Duplicated Values

unique(mtcars$cyl)
dupes <- duplicated(iris)
head(dupes)
which(dupes)
```

```
iris[dupes, ]
iris[!dupes, ]
nrow(iris[!dupes, ])


# Working with Lookup Tables

index <- match("Toyota Corolla", rownames(mtcars))
index
mtcars[index, 1:4]


# Working with Pivot Tables

with(mtcars, tapply(hp, list(cyl, gear), mean))
aggregate(hp~cyl+gear+am, mtcars, mean)


# Using the Goal Seek and Solver

sales <- function(price) { 100 - 0.5 * price }
revenue <- function(price) { price * sales(price) }


par(mfrow=c(1, 2))
curve(sales, from=50, to=150, xname="price", ylab="Sales", main="Sales")
curve(revenue, from=50, to=150, xname="price", ylab="Revenue", main="Revenue")
par(mfrow=c(1, 1))

optimize(revenue, interval=c(50, 150), maximum=TRUE)
```

---

ch20   *Print examples of chapter 20 of 'R for Dummies'.*

Description

To print a listing of all examples of a chapter, use ch20(). To run all the examples of ch20(), use example(ch20).

Usage

```
ch20()
```

See Also

[toc](toc)

Other Chapters: ch01, ch01, ch1; ch02, ch02, ch2; ch03, ch03, ch3; ch04, ch04, ch4; ch05, ch05, ch5; ch06, ch06, ch6; ch07, ch07, ch7; ch08, ch08, ch8; ch09, ch09, ch9; ch10; ch11; ch12; ch13; ch14; ch15; ch16; ch17; ch18; ch19

Examples

```
# Chapter 20 - Ten Tips on Working with Packages

## Poking Around the Nooks and Crannies of CRAN

options("repos" = c(CRAN = "http://cran.ma.imperial.ac.uk/"))

## Finding Interesting Packages

## Installing Packages

## Not run:
install.packages("fortunes")

## End(Not run)

## Loading Packages

library("fortunes")

## Reading the Package Manual and Vignette

library(help=fortunes)
## Not run:
vignette("fortunes")

## End(Not run)

## Updating Packages

## Not run:
update.packages()

## End(Not run)

## Unloading Packages

search()
detach(package:fortunes, unload=TRUE)

## Forging Ahead with R-Forge

## Not run:
install.packages("data.table", repos="http://R-Forge.R-project.org")

## End(Not run)

## Conducting Installations from BioConductor

source("http://bioconductor.org/biocLite.R")

## Reading the R Manual
```

elements	*Periodic table of elements.*

Description

A data set containing properties of the periodic table of elements.

Format

A data frame with 118 rows and 9 variables

Details

- Atomic.no
- Name
- Symbol
- Group
- Period
- Block
- State.at.STP
- Occurrence
- Description

Source

http://en.wikipedia.org/wiki/Periodic_table

rfordummies	*A package to accompany the book "R for Dummies".*

Description

This package contains all the code examples in the book 'R for Dummies' (1st edition)

References

de Vries, A. , & Meys, J. (2012). <i>R for dummies</i>. Chichester: Wiley. http://rfordummies.com/, ISBN-13: 978-1119962847.

---

toc *Print table of contents.*

Description

Print table of contents.

Usage

```
toc()
```

Examples

```
toc()
```

Index