

Base R

Cheat Sheet

Getting Help

Accessing the help files

?mean

Get help of a particular function.

help.search('weighted mean')

Search the help files for a word or phrase.

help(package = 'dplyr')

Find help for a package.

More about an object

str(iris)

Get a summary of an object's structure.

class(iris)

Find the class an object belongs to.

Using Packages

install.packages('dplyr')

Download and install a package from CRAN.

library(dplyr)

Load the package into the session, making all its functions available to use.

dplyr::select

Use a particular function from a package.

data(iris)

Load a built-in dataset into the environment.

Working Directory

getwd()

Find the current working directory (where inputs are found and outputs are sent).

setwd('C://file/path')

Change the current working directory.

Use projects in RStudio to set the working directory to the folder you are working in.

Vectors

Creating Vectors

c(2, 4, 6)	2 4 6	Join elements into a vector
2:6	2 3 4 5 6	An integer sequence
seq(2, 3, by=0.5)	2.0 2.5 3.0	A complex sequence
rep(1:2, times=3)	1 2 1 2 1 2	Repeat a vector
rep(1:2, each=3)	1 1 1 2 2 2	Repeat elements of a vector

Vector Functions

sort(x)

Return x sorted.

table(x)

See counts of values.

rev(x)

Return x reversed.

unique(x)

See unique values.

Selecting Vector Elements

By Position

x[4]

The fourth element.

x[-4]

All but the fourth.

x[2:4]

Elements two to four.

x[-(2:4)]

All elements except two to four.

x[c(1, 5)]

Elements one and five.

By Value

x[x == 10]

Elements which are equal to 10.

x[x < 0]

All elements less than zero.

x[x %in% c(1, 2, 5)]

Elements in the set 1, 2, 5.

Named Vectors

x['apple']

Element with name 'apple'.

Programming

For Loop

```
for (variable in sequence) {  
  Do something  
}
```

Example

```
for (i in 1:4){  
  j <- i + 10  
  print(j)  
}
```

While Loop

```
while (condition) {  
  Do something  
}
```

Example

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

If Statements

```
if (condition) {  
  Do something  
} else {  
  Do something different  
}
```

Example

```
if (i > 3){  
  print('Yes')  
} else {  
  print('No')  
}
```

Functions

```
function_name <- function(var) {  
  Do something  
  return(new_variable)  
}
```

Example

```
square <- function(x) {  
  squared <- x*x  
  return(squared)  
}
```

Reading and Writing Data

Also see the **readr** package.

Input	Output	Description
df <- read.table('file.txt')	write.table(df, 'file.txt')	Read and write a delimited text file.
df <- read.csv('file.csv')	write.csv(df, 'file.csv')	Read and write a comma separated value file. This is a special case of read.table/write.table.
load('file.Rdata')	save(df, file = 'file.Rdata')	Read and write an R data file, a file type special for R.

Conditions

a == b	Are equal	a > b	Greater than	a >= b	Greater than or equal to	is.na(a)	Is missing
a != b	Not equal	a < b	Less than	a <= b	Less than or equal to	is.null(a)	Is null

Types

Converting between common data types in R. Can always go from a higher value in the table to a lower value.

<code>as.logical</code>	TRUE, FALSE, TRUE	Boolean values (TRUE or FALSE).
<code>as.numeric</code>	1, 0, 1	Integers or floating point numbers.
<code>as.character</code>	'1', '0', '1'	Character strings. Generally preferred to factors.
<code>as.factor</code>	'1', '0', '1', levels: '1', '0'	Character strings with preset levels. Needed for some statistical models.

Maths Functions

<code>log(x)</code>	Natural log.	<code>sum(x)</code>	Sum.
<code>exp(x)</code>	Exponential.	<code>mean(x)</code>	Mean.
<code>max(x)</code>	Largest element.	<code>median(x)</code>	Median.
<code>min(x)</code>	Smallest element.	<code>quantile(x)</code>	Percentage quantiles.
<code>round(x, n)</code>	Round to n decimal places.	<code>rank(x)</code>	Rank of elements.
<code>signif(x, n)</code>	Round to n significant figures.	<code>var(x)</code>	The variance.
<code>cor(x, y)</code>	Correlation.	<code>sd(x)</code>	The standard deviation.

Variable Assignment

```
> a <- 'apple'
> a
[1] 'apple'
```

The Environment

```
ls()
# List all variables in the environment.

rm(x)
# Remove x from the environment.

rm(list = ls())
# Remove all variables from the environment.
```

You can use the environment panel in RStudio to browse variables in your environment.

Matrices

```
m <- matrix(x, nrow = 3, ncol = 3)
# Create a matrix from x.

m[2, ] # - Select a row
m[, 1] # - Select a column
m[2, 3] # - Select an element

t(m) # Transpose
m %*% n # Matrix Multiplication
solve(m, n) # Find x in: m * x = n
```

Lists

```
l <- list(x = 1:5, y = c('a', 'b'))
# A list is a collection of elements which can be of different types.

l[[2]] # 1 [1]
# Second element of l.

l[[1]] # 1 [1]
# New list with only the first element.

l$x # 1 $x
# Element named x.

l[["y"]] # 1 [1]
# New list with only element named y.
```

Also see the `dplyr` package.

Data Frames

```
df <- data.frame(x = 1:3, y = c('a', 'b', 'c'))
# A special case of a list where all elements are the same length.
```

x	y
1	a
2	b
3	c

List subsetting

```
df$x # 1 2 3
df[[2]] # 1 2 3
```

Understanding a data frame

View(df) See the full data frame.

head(df) See the first 6 rows.

Matrix subsetting

```
df[, 2]
# 1 2
```

nrow(df) Number of rows.

```
df[, 1]
# 1 2
```

ncol(df) Number of columns.

```
df[2, 2]
# 2
```

dim(df) Number of columns and rows.

cbind - Bind columns.



rbind - Bind rows.



Strings

Also see the **stringr** package.

```
paste(x, y, sep = ' ') # Join multiple vectors together.
paste(x, collapse = ' ') # Join elements of a vector together.
grep(pattern, x) # Find regular expression matches in x.
gsub(pattern, replace, x) # Replace matches in x with a string.
toupper(x) # Convert to uppercase.
tolower(x) # Convert to lowercase.
nchar(x) # Number of characters in a string.
```

Factors

```
factor(x) # Turn a vector into a factor. Can set the levels of the factor and the order.
cut(x, breaks = 4) # Turn a numeric vector into a factor by 'cutting' into sections.
```

Statistics

```
lm(y ~ x, data=df) # Linear model.
glm(y ~ x, data=df) # Generalised linear model.
summary # Get more detailed information out a model.
t.test(x, y) # Perform a t-test for difference between means.
prop.test # Test for a difference between proportions.
pairwise.t.test # Perform a t-test for paired data.
aov # Analysis of variance.
```

Distributions

	Random Variates	Density Function	Cumulative Distribution	Quantile
Normal	<code>rnorm</code>	<code>dnorm</code>	<code>pnorm</code>	<code>qnorm</code>
Poisson	<code>rpois</code>	<code>dpois</code>	<code>ppois</code>	<code>qpois</code>
Binomial	<code>rbinom</code>	<code>dbinom</code>	<code>pbinom</code>	<code>qbinom</code>
Uniform	<code>runif</code>	<code>dunif</code>	<code>punif</code>	<code>qunif</code>

Plotting

Also see the **ggplot2** package.



`plot(x)` Values of x in order.



`plot(x, y)` Values of x against y.



`hist(x)` Histogram of x.

Dates

See the **lubridate** package.