

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)

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

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

Vectors Functions

sort(x)

Return x sorted.

rev(x)

Return x reversed.

table(x)

See counts of values.

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]`

Element which are equal to 10.

`x[which(x==10)]`

Element 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 Statement

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

Example

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

Functions

```
func_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
<code>df <- read.table('file.txt')</code>	<code>write.table(df, 'file.txt')</code>	Read and write a delimited text file.
<code>df <- read.csv('file.csv')</code>	<code>write.csv(df, 'file.csv')</code>	Read and write a comma separated value file. This is a special case of <code>read.table/write.table</code> .
<code>load('file.RData')</code>	<code>save(df, file = 'file.RData')</code>	Read and write a n R data file, a file type special for R.

Conditions

<code>a == b</code>	Are equal	<code>a > b</code>	Greater than	<code>a >= b</code>	Greater than or equal to	<code>is.na(a)</code>	Is missing
<code>a != b</code>	Not equal	<code>a < b</code>	Less than	<code>a <= b</code>	Less than or equal to	<code>is.null(a)</code>	Is null
<code>c e</code>	c or e	<code>c & y</code>	c and y				

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>	<code>TRUE, FALSE, TRUE</code>	Boolean values (TRUE or FALSE)
<code>as.numeric</code>	<code>1, 0, 1</code>	Integer or floating point numbers
<code>as.integer</code>	<code>1, 0, 1</code>	Integers
<code>as.character</code>	<code>'1', '0', '1'</code>	Character strings. Generally preferred to factors
<code>as.factor</code>	<code>'1', '0', '1'</code> <code>levels: '1', '0'</code>	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.

Lists

```
l <-list(x = 1:5, y = c('a', 'b'))
```

A list is a collection of elements which can be of different types.

<code>l[[2]]</code>	<code>l[1]</code>	<code>l\$x</code>	<code>l['y']</code>
Second element of l.	New list with only the first element.	Element named x.	New list with only element named y.

Miscellaneous

Arithmetics

```
16 = 3*5 + 1
```

16 % 3 Quotient (result 5).

16 % 3 Remainder (result 1).

Permutations

```

sample(x, size, replace = FALSE)           Give a sample of the specified size from
                                              the elements of x.

      sample(c(1:5), 10, replace = TRUE)

sort(x, decreasing = FALSE)                Sort a vector or factor.

      sort(c(5, 1, 7, 3)) (Result 1 3 5 7)

order(x, decreasing = FALSE)               Returns a permutation rearranging x.

      order(c(5, 1, 7, 3)) (Result 3 1 4 2)

```

Memo

[illegible]

Strings Also see the **stringr** package.

<code>cat(x, y, sep = '')</code>	Join and print multiple vectors together.
<code>cat(x, collapse = '')</code>	Join and print elements of a vector together.
<code>grep(pattern, x)</code>	Find regular expression matches in <code>x</code> .
<code>gsub(pattern, replace, x)</code>	Replace matches in <code>x</code> with a string.
<code>toupper(x)</code>	Convert to uppercase.
<code>tolower(x)</code>	Convert to lowercase.
<code>nchar(x)</code>	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

<code>lm(y ~ x, data=df)</code> Linear model.	<code>t.test(x, y)</code> Perform a t-test for	<code>prop.test</code> Test for a
--	---	--------------------------------------

<code>glm(y ~ x, data=df)</code> Generalized linear model.	<code>prop.test(x, n)</code> Perform a z-test for difference between proportions.	<code>prop.test(x, n, conf.level=0.95)</code> Perform a z-test for difference between proportions with confidence level.
---	--	---

summary or fivenum	pairwise.t.test	prop.test
Get more detailed information out a model.	Perform a t-test for paired data.	Test the equality of proportions.
aov		
Analysis of variance		

<pre>lm(y ~ x, data=df)</pre> <p>Linear model.</p> <pre>glm(y ~ x, data=df)</pre> <p>Generalized linear model.</p>	<pre>t.test(x, y)</pre> <p>Perform a t-test for difference between means.</p>	<pre>prop.test</pre> <p>Test for a difference between proportions</p>
--	---	---

summary or fivenum	pairwise.t.test	prop.test
Get more detailed information out a model.	Perform a t-test for paired data.	Test the equality of proportions.
aov		
Analysis of variance		

<pre>lm(y ~ x, data=df)</pre> <p>Linear model.</p> <pre>glm(y ~ x, data=df)</pre> <p>Generalized linear model.</p>	<pre>t.test(x, y)</pre> <p>Perform a t-test for difference between means.</p>	<pre>prop.test</pre> <p>Test for a difference between proportions</p>
--	---	---

summary or fivenum	pairwise.t.test	prop.test
Get more detailed information out a model.	Perform a t-test for paired data.	Test the equality of proportions.
aov		
Analysis of variance		

Distributions

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

Plotting Also see the **ggplot2** package.

The figure shows three plots side-by-side, each with a title and a description:

- plot(x)**: Values of x in order. The plot shows a sequence of points connected by lines, representing the order of values in the vector x.
- plot(x, y)**: Values of x against y. The plot shows points scattered in a 2D space, representing the relationship between two variables.
- hist(x)**: Histogram of x. The plot shows a bar chart representing the frequency distribution of the values in x.