

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

class(iris)

Find the class an object belongs to.

Using Libraries

install.packages("haven")

Download and install a package from CRAN.

library(haven)

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

haven::read_dta

Use a particular function from a package.

data(iris)

Load a build-in dataset into the environment.

Working Directory

getwd()

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

setwd("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.

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) {  
  print(i + 10)  
}
```

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(arg) {  
  Do something  
}
```

Example

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

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.integer</code>	1L, 0L, 1L	Integers.
<code>as.double</code>	1, 0, 1	Floating point numbers.
<code>as.character</code>	"1", "0", "1"	Character strings. Generally preferred to factors.
<code>as.factor</code>	"1", "0", "1", Levels: "0", "1"	Character strings with set levels.

Math 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>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.

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

Matrixes

```
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

Lists

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

`l[[2]]`

Second element of l.

`l[1]`

New list with only the first element.

`l$x`

Element named x.

`l["y"]`

New list with only element named y.

Also see the **dplyr** library.

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

Matrix subsetting

`df[, 2]`



`df[2,]`



`df[2, 2]`



List subsetting

`df$x`



`df[[2]]`



Understanding a data frame

`View(df)`

See the full data frame.

`head(df)`

See the first 6 rows.

Strings

See the **stringr** library.

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(x ~ y, data = df)`
Linear model.

`glm(x ~ y, data = df)`
Generalized linear model.

`summary(model)`
Get more detailed information out a model.

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

See the **ggplot2** library.

Dates

See the **lubridate** library.