

# 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 Libraries

##### 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

<b>sort(x)</b>	Return x sorted.	<b>rev(x)</b>	Return x reversed.
<b>table(x)</b>	See counts of values.	<b>unique(x)</b>	See unique values.

#### Selecting Vector Elements

##### By Position

<b>x[4]</b>	The fourth element.
<b>x[-4]</b>	All but the fourth.
<b>x[2:4]</b>	Elements two to four.
<b>x[-(2:4)]</b>	All elements except two to four.
<b>x[c(1, 5)]</b>	Elements one and five.

##### By Value

<b>x[x == 10]</b>	Elements which are equal to 10.
<b>x[x &lt; 0]</b>	All elements less than zero.
<b>x[x %in% c(1, 2, 5)]</b>	Elements in the set 1, 2, 5.

##### Named Vectors

<b>x['apple']</b>	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

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

<code>ls()</code>	List all variables in the environment.
<code>rm(x)</code>	Remove x from the environment.
<code>rm(list = ls())</code>	Remove all variables from the environment.

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

## Matrixes

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

 <code>m[2, ]</code> - Select a row	<code>t(m)</code> Transpose
 <code>m[, 1]</code> - Select a column	<code>m %*% n</code> Matrix Multiplication
 <code>m[2, 3]</code> - Select an element	<code>solve(m, n)</code> Find x in: $m * x = n$

## Lists

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

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



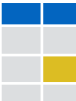
Also see the **dplyr** library.

## 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

<code>df[, 2]</code>	
<code>df[2, ]</code>	
<code>df[2, 2]</code>	

### List subsetting

<code>df\$x</code>		<code>df[[2]]</code>	
--------------------	---	----------------------	---

### Understanding a data frame

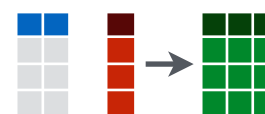
<code>View(df)</code>	See the full data frame.
<code>head(df)</code>	See the first 6 rows.

`nrow(df)`  
Number of rows.

`ncol(df)`  
Number of columns.

`dim(df)`  
Number of columns and rows.

`cbind` - Bind columns.



`rbind` - Bind rows.



## Strings

Also see the **stringr** library.

<code>paste(x, y, sep = ' ')</code>	Join multiple vectors together.
<code>paste(x, collapse = ' ')</code>	Join elements of a vector together.
<code>grep(pattern, x)</code>	Find regular expression matches in x.
<code>gsub(pattern, replace, x)</code>	Replace matches in x 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

<code>factor(x)</code>	Turn a vector into a factor. Can set the levels of the factor and the order.	<code>cut(x, breaks = 4)</code>	Turn a numeric vector into a factor but 'cutting' into sections.
------------------------	--	---------------------------------	--

## Statistics

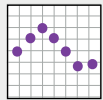
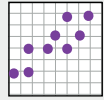
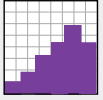
<code>lm(x ~ y, data=df)</code> Linear model.	<code>t.test(x, y)</code> Perform a t-test for difference between means.	<code>prop.test</code> Test for a difference between proportions.
<code>glm(x ~ y, data=df)</code> Generalised linear model.	<code>pairwise.t.test</code> Perform a t-test for paired data.	<code>aov</code> Analysis of variance.
<code>summary</code> 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

Also see the **ggplot2** library.

 <code>plot(x)</code> Values of x in order.	 <code>plot(x, y)</code> Values of x against y.	 <code>hist(x)</code> Histogram of x.
---	---	---

## Dates

See the **lubridate** library.

# Data Wrangling with dplyr and tidyr

## Cheat Sheet



### Syntax - Helpful conventions for wrangling

#### dplyr::tbl\_df(iris)

Converts data to tbl class. tbl's are easier to examine than data frames. R displays only the data that fits onscreen:

```
Source: local data frame [150 x 5]
  Sepal.Length Sepal.Width Petal.Length
1           5.1           3.5           1.4
2           4.9           3.0           1.4
3           4.7           3.2           1.3
4           4.6           3.1           1.5
5           5.0           3.6           1.4
..           ...           ...           ..
Variables not shown: Petal.Width (dbl),
Species (fctr)
```

#### dplyr::glimpse(iris)

Information dense summary of tbl data.

#### utils::View(iris)

View data set in spreadsheet-like display (note capital V).

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa
7	4.6	3.4	1.4	0.3	setosa
8	5.0	3.4	1.5	0.2	setosa

#### dplyr::%>%

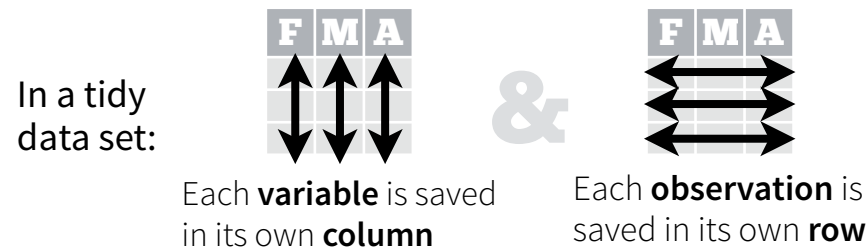
Passes object on left hand side as first argument (or . argument) of function on righthand side.

$x \%>\% f(y)$  is the same as  $f(x, y)$   
 $y \%>\% f(x, ., z)$  is the same as  $f(x, y, z)$

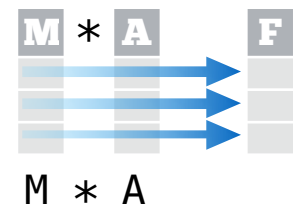
"Piping" with %>% makes code more readable, e.g.

```
iris %>%
  group_by(Species) %>%
  summarise(avg = mean(Sepal.Width)) %>%
  arrange(avg)
```

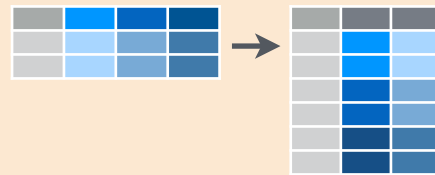
## Tidy Data - A foundation for wrangling in R



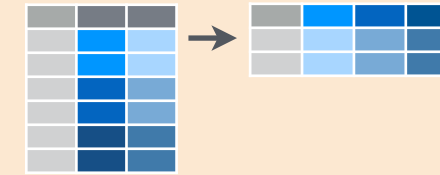
Tidy data complements R's **vectorized operations**. R will automatically preserve observations as you manipulate variables. No other format works as intuitively with R.



### Reshaping Data - Change the layout of a data set



**tidyr::gather(cases, "year", "n", 2:4)**  
Gather columns into rows.



**tidyr::spread(pollution, size, amount)**  
Spread rows into columns.



**tidyr::separate(storms, date, c("y", "m", "d"))**  
Separate one column into several.



**tidyr::unite(data, col, ..., sep)**  
Unite several columns into one.

**dplyr::data\_frame(a = 1:3, b = 4:6)**  
Combine vectors into data frame (optimized).

**dplyr::arrange(mtcars, mpg)**  
Order rows by values of a column (low to high).

**dplyr::arrange(mtcars, desc(mpg))**  
Order rows by values of a column (high to low).

**dplyr::rename(tb, y = year)**  
Rename the columns of a data frame.

### Subset Observations (Rows)



**dplyr::filter(iris, Sepal.Length > 7)**  
Extract rows that meet logical criteria.

**dplyr::distinct(iris)**  
Remove duplicate rows.

**dplyr::sample\_frac(iris, 0.5, replace = TRUE)**  
Randomly select fraction of rows.

**dplyr::sample\_n(iris, 10, replace = TRUE)**  
Randomly select n rows.

**dplyr::slice(iris, 10:15)**  
Select rows by position.

**dplyr::top\_n(storms, 2, date)**  
Select and order top n entries (by group if grouped data).

### Subset Variables (Columns)



**dplyr::select(iris, Sepal.Width, Petal.Length, Species)**  
Select columns by name or helper function.

#### Helper functions for select - ?select

- select(iris, contains("."))**  
Select columns whose name contains a character string.
- select(iris, ends\_with("Length"))**  
Select columns whose name ends with a character string.
- select(iris, everything())**  
Select every column.
- select(iris, matches(".t."))**  
Select columns whose name matches a regular expression.
- select(iris, num\_range("x", 1:5))**  
Select columns named x1, x2, x3, x4, x5.
- select(iris, one\_of(c("Species", "Genus")))**  
Select columns whose names are in a group of names.
- select(iris, starts\_with("Sepal"))**  
Select columns whose name starts with a character string.
- select(iris, Sepal.Length:Petal.Width)**  
Select all columns between Sepal.Length and Petal.Width (inclusive).
- select(iris, -Species)**  
Select all columns except Species.

#### Logic in R - ?Comparison, ?base::Logic

<	Less than	!=	Not equal to
>	Greater than	%in%	Group membership
==	Equal to	is.na	Is NA
<=	Less than or equal to	!is.na	Is not NA
>=	Greater than or equal to	&,  , !, xor, any, all	Boolean operators

## Summarise Data



`dplyr::summarise(iris, avg = mean(Sepal.Length))`

Summarise data into single row of values.

`dplyr::summarise_each(iris, funs(mean))`

Apply summary function to each column.

`dplyr::count(iris, Species, wt = Sepal.Length)`

Count number of rows with each unique value of variable (with or without weights).



Summarise uses **summary functions**, functions that take a vector of values and return a single value, such as:

<code>dplyr::first</code> First value of a vector.	<code>min</code> Minimum value in a vector.
<code>dplyr::last</code> Last value of a vector.	<code>max</code> Maximum value in a vector.
<code>dplyr::nth</code> Nth value of a vector.	<code>mean</code> Mean value of a vector.
<code>dplyr::n</code> # of values in a vector.	<code>median</code> Median value of a vector.
<code>dplyr::n_distinct</code> # of distinct values in a vector.	<code>var</code> Variance of a vector.
<code>IQR</code> IQR of a vector.	<code>sd</code> Standard deviation of a vector.

## Group Data

`dplyr::group_by(iris, Species)`

Group data into rows with the same value of Species.

`dplyr::ungroup(iris)`

Remove grouping information from data frame.

`iris %>% group_by(Species) %>% summarise(...)`

Compute separate summary row for each group.



## Make New Variables



`dplyr::mutate(iris, sepal = Sepal.Length + Sepal.Width)`

Compute and append one or more new columns.

`dplyr::mutate_each(iris, funs(min_rank))`

Apply window function to each column.

`dplyr::transmute(iris, sepal = Sepal.Length + Sepal.Width)`

Compute one or more new columns. Drop original columns.

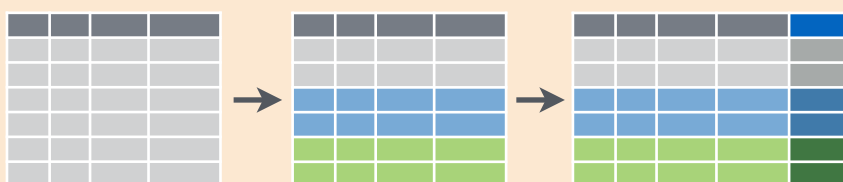


Mutate uses **window functions**, functions that take a vector of values and return another vector of values, such as:

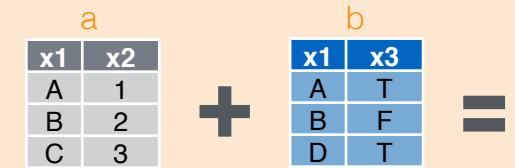
<code>dplyr::lead</code> Copy with values shifted by 1.	<code>dplyr::cumall</code> Cumulative <b>all</b>
<code>dplyr::lag</code> Copy with values lagged by 1.	<code>dplyr::cumany</code> Cumulative <b>any</b>
<code>dplyr::dense_rank</code> Ranks with no gaps.	<code>dplyr::cummean</code> Cumulative <b>mean</b>
<code>dplyr::min_rank</code> Ranks. Ties get min rank.	<code>cumsum</code> Cumulative <b>sum</b>
<code>dplyr::percent_rank</code> Ranks rescaled to [0, 1].	<code>cummax</code> Cumulative <b>max</b>
<code>dplyr::row_number</code> Ranks. Ties got to first value.	<code>cummin</code> Cumulative <b>min</b>
<code>dplyr::ntile</code> Bin vector into n buckets.	<code>cumprod</code> Cumulative <b>prod</b>
<code>dplyr::between</code> Are values between a and b?	<code>pmax</code> Element-wise <b>max</b>
<code>dplyr::cume_dist</code> Cumulative distribution.	<code>pmin</code> Element-wise <b>min</b>

`iris %>% group_by(Species) %>% mutate(...)`

Compute new variables by group.



## Combine Data Sets



Mutating Joins

x1	x2	x3
A	1	T
B	2	F
C	3	NA

x1	x3	x2
A	T	1
B	F	2
D	T	NA

x1	x2	x3
A	1	T
B	2	F

x1	x2	x3
A	1	T
B	2	F
C	3	NA
D	NA	T

`dplyr::left_join(a, b, by = "x1")`

Join matching rows from b to a.

`dplyr::right_join(a, b, by = "x1")`

Join matching rows from a to b.

`dplyr::inner_join(a, b, by = "x1")`

Join data. Retain only rows in both sets.

`dplyr::full_join(a, b, by = "x1")`

Join data. Retain all values, all rows.

Filtering Joins

x1	x2
A	1
B	2

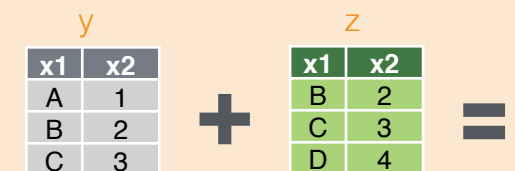
x1	x2
C	3

`dplyr::semi_join(a, b, by = "x1")`

All rows in a that have a match in b.

`dplyr::anti_join(a, b, by = "x1")`

All rows in a that do not have a match in b.



Set Operations

x1	x2
B	2
C	3

x1	x2
A	1
B	2
C	3
D	4

x1	x2
A	1

`dplyr::intersect(y, z)`

Rows that appear in both y and z.

`dplyr::union(y, z)`

Rows that appear in either or both y and z.

`dplyr::setdiff(y, z)`

Rows that appear in y but not z.

Binding

x1	x2
A	1
B	2
C	3

x1	x2	x1	x2
A	1	B	2
B	2	C	3
C	3	D	4

`dplyr::bind_rows(y, z)`

Append z to y as new rows.

`dplyr::bind_cols(y, z)`

Append z to y as new columns.

Caution: matches rows by position.