Bentley University MA346 in R

Content extracted from the How to Data Website

This PDF generated on 23 November 2021

Contents

MA346 is an undergraduate data science course at Bentley University. The description from the course catalog can be found here. Topics included in the course are listed as tasks (on website) below.

Mathematical topics include functions and relations, a review of basic statistics, and (time permitting) networks, matrices, and an introduction to supervised learning.

Computing topics include Jupyter notebooks (locally and in the cloud), Python and pandas, abstraction, concatenation and merging, map-reduce, split-apply-combine, data munging, version control, and dashboards.

Communication topics include best practices for writing reports, documenting code and computational notebooks, and data visualization.

Basics

- How to do basic mathematical computations
- How to compute summary statistics

Data manipulation

- How to convert a text column into dates

Statistics in Python

- How to compute covariance and correlation coefficients

Plotting

- How to create basic plots
- How to add details to a plot
- How to change axes, ticks, and scale in a plot
- How to create a histogram
- How to create a box (and whisker) plot

This page is not yet complete. More content will be added here over time.

Content last modified on 23 July 2021.

How to do basic mathematical computations

Description

How do we write the most common mathematical operations in a given piece of software? For example, how do we write multiplication, or exponentiation, or logarithms, in Python vs. R vs. Excel, and so on?

Solution in R

For those expressions that need the Python math package, use the code `import math` beforehand to ensure that package is loaded. Alternatively, you can write `from math import *` and thus drop the `math` prefixes in the table below.

Mathematical notation	R code
$x + y$	`x+y`
$x - y$	`x-y`
xy	`x*y`
$\frac{x}{y}$	`x/y`
x^y	`x^y`
$\lvert x \rvert$	`abs(x)`
$\ln x$	`log(x)`
$\log_a b$	`log(b,a)`
e^x	`exp(x)`
π	`pi`
$\sin x$	`sin(x)`
$\sin^{-1} x$	`asin(x)`
$\sqrt{x}$	`sqrt(x)`

Other trigonometric functions are also available besides just `sin`, including `cos`, `tan`, etc.

R naturally applies these functions across vectors. For example, you can square all the entries in a vector as in the example below.

```
example.vector <- c( -3, 2, 0.5, -1, 10, 9.2, -3.3 )
example.vector ^ 2
```

```
[1]   9.00   4.00   0.25   1.00 100.00  84.64  10.89
```

Content last modified on 23 November 2021.

See a problem? Tell us or edit the source.

How to compute summary statistics

Description

The phrase "summary statistics" usually refers to a common set of simple computations that can be done about any dataset, including mean, median, variance, and some of the others shown below.

Related tasks:

- How to summarize a column (on website)
- How to summarize and compare data by groups (on website)

Solution in R

We first load a famous dataset, Fisher's irises, just to have some example data to use in the code that follows. (See how to quickly load some sample data (on website).)

```
library(datasets)
data(iris)
```

How big is the dataset? The output shows number of rows then number of columns.

```
dim(iris)  # Short for "dimensions."
```

```
[1] 150   5
```

What are the columns and their data types? Can I see a sample of each column?

```
str(iris)  # Short for "structure."
```

```
'data.frame':   150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

What do the first few rows look like?

```
head(iris) # Gives 5 rows by default.  You can do head(iris,10), etc.
```

```
  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
```

The easiest way to get summary statistics for an R `data.frame` is with the `summary` function.

```
summary(iris)
```

```
  Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
 Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
 1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
 Median :5.800   Median :3.000   Median :4.350   Median :1.300
 Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
 3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
 Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
       Species
 setosa    :50
 versicolor:50
 virginica :50
```

The columns from the original dataset are the column headings in the summary output, and the statistics computed for each are listed below those headings.

We can also compute these statistics (and others) one at a time for any given set of data points. Here, we let `xs` be one column from the above `data.frame` but you could use any vector or list.

```
xs <- iris$Sepal.Length

mean( xs )           # mean, or average, or center of mass
median( xs )         # 50th percentile
quantile( xs, 0.25 ) # compute any percentile, such as the 25th
var( xs )            # variance
sd( xs )             # standard deviation, the square root of the variance
sort( xs )           # data in increasing order
sum( xs )            # sum, or total
```

Content last modified on 26 July 2021.

See a problem? Tell us or edit the source.

How to convert a text column into dates

Description

When loading data, many software systems make intelligent guesses about the format and data type of each column, but sometimes that is not sufficient. If you have a column of text that should be interpreted as dates, how can we ask the software to convert it?

Solution in R

How to Data does not yet contain a solution for this task in R.

How to compute covariance and correlation coefficients

Description

Covariance is a measure of how much two variables "change together." It is positive when the variables tend to increase or decrease together, and negative when they upward motion of one variable is correlated with downward motion of the other. Correlation normalizes covariance to the interval $[-1, 1]$.

Solution in R

How to Data does not yet contain a solution for this task in R.

How to create basic plots

Description

Plotting is a huge topic with many options and variations, but the most foundational types of plots are a line plot and a scatterplot. How can we create those?

Related topics:

- How to add details to a plot
- How to create a histogram
- How to create a box (and whisker) plot
- How to change axes, ticks, and scale in a plot
- How to create bivariate plots to compare groups (on website)
- How to plot interaction effects of treatments (on website)

Solution in R

We will create some fake data using vectors, for simplicity. But everything we show below works also if your data is in columns of a DataFrame.

```
patient.id     <- c(0, 1, 2, 3, 4, 5, 6, 7, 8, 9)
patient.height <- c(60,  64,  64,  65,  66,  66,  70,  72,  72,  76)
patient.weight <- c(141, 182, 169, 204, 138, 198, 180, 175, 244, 196)
```

We can make a line plot if we use the `type="l"` option (which is an "ell," not a number one).

```
plot(patient.id, patient.height, main="Patient heights", type="l")
```

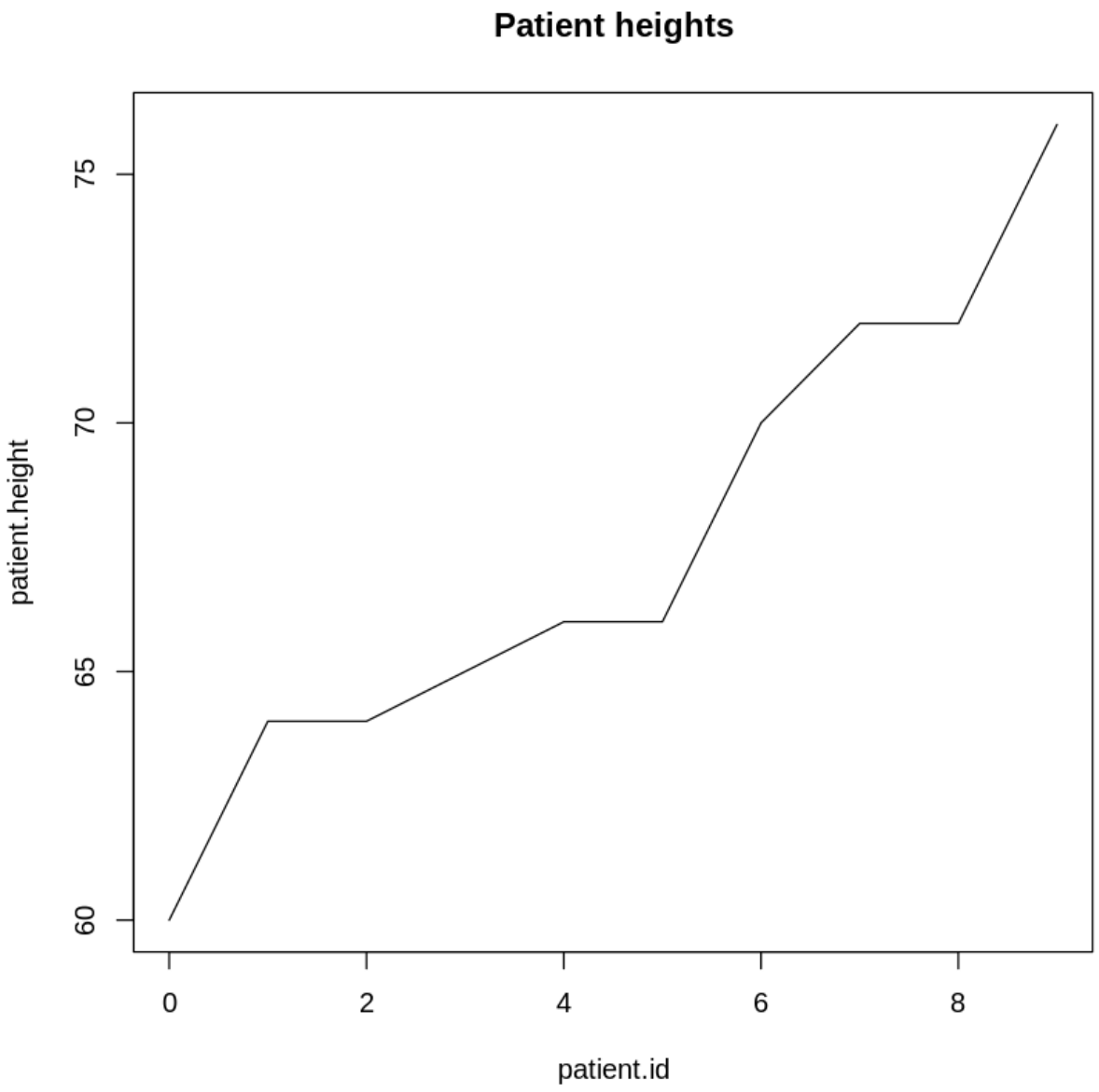

Figure 1: png

We can create a scatterplot if we have two numerical columns, such as the height and weight in the data above.

```
plot(patient.height, patient.weight, main = "Height vs. Weight")
```

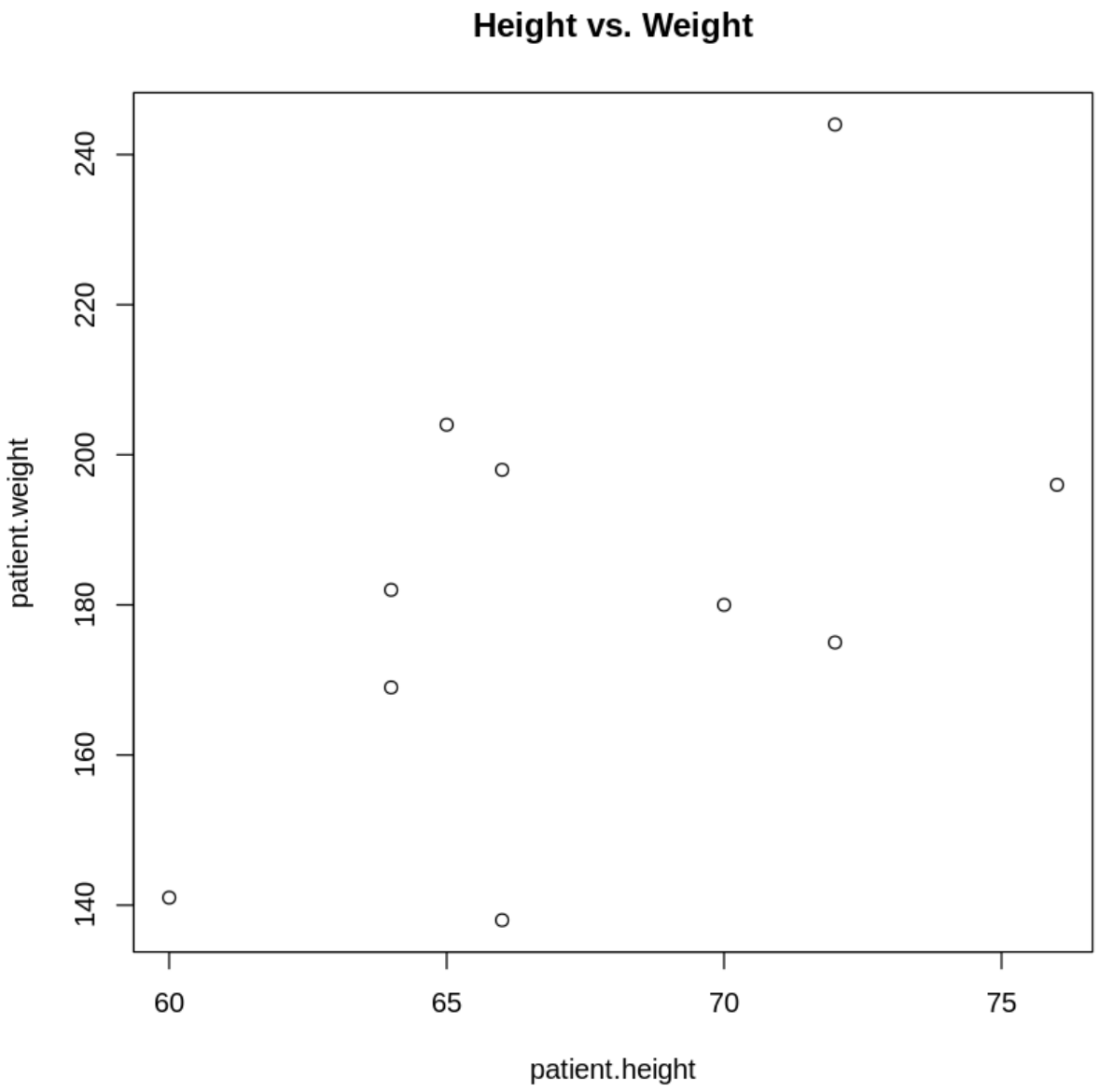

Figure 2: png

Content last modified on 14 September 2021.

See a problem? Tell us or edit the source.

How to add details to a plot

Description

After making a plot, we might want to add axis labels, a title, gridlines, or text. Plotting packages provide tons of tools for this sort of thing. What are some of the essentials?

Related topics:

- How to create basic plots
- How to create a histogram
- How to create a box (and whisker) plot
- How to change axes, ticks, and scale in a plot
- How to create bivariate plots to compare groups (on website)
- How to plot interaction effects of treatments (on website)

Solution in R

How to Data does not yet contain a solution for this task in R.

How to change axes, ticks, and scale in a plot

Description

The mathematical markings and measurements in a plot can make a big difference on its readability and usefulness. These include the range of each axis, which points on that axis are marked with tick marks, and whether the axes use linear or logarithmic scaling. How can we customize these options?

Related topics:

- How to create basic plots
- How to create a histogram
- How to create a box (and whisker) plot
- How to add details to a plot
- How to create bivariate plots to compare groups (on website)
- How to plot interaction effects of treatments (on website)

Solution in R

How to Data does not yet contain a solution for this task in R.

How to create a histogram

Description

A histogram is a very common and useful data visualization. It displays an approximation of the distribution in single series of data points (one variable) by grouping the data into bins, each bin draw as a vertical bar. How can we create such a visualization?

Related topics:

- How to create basic plots
- How to create a box (and whisker) plot
- How to add details to a plot
- How to create bivariate plots to compare groups (on website)
- How to plot interaction effects of treatments (on website)

Solution in R

We will create some random data, but that's just for demonstration purposes. You can apply the answer below to any data. Simply replace the data variable with your real data (a list, a column of a dataframe, etc.).

```
data <- rnorm(1000)
```

We can use R's hist() function to create the histogram.

```
hist(data)
```

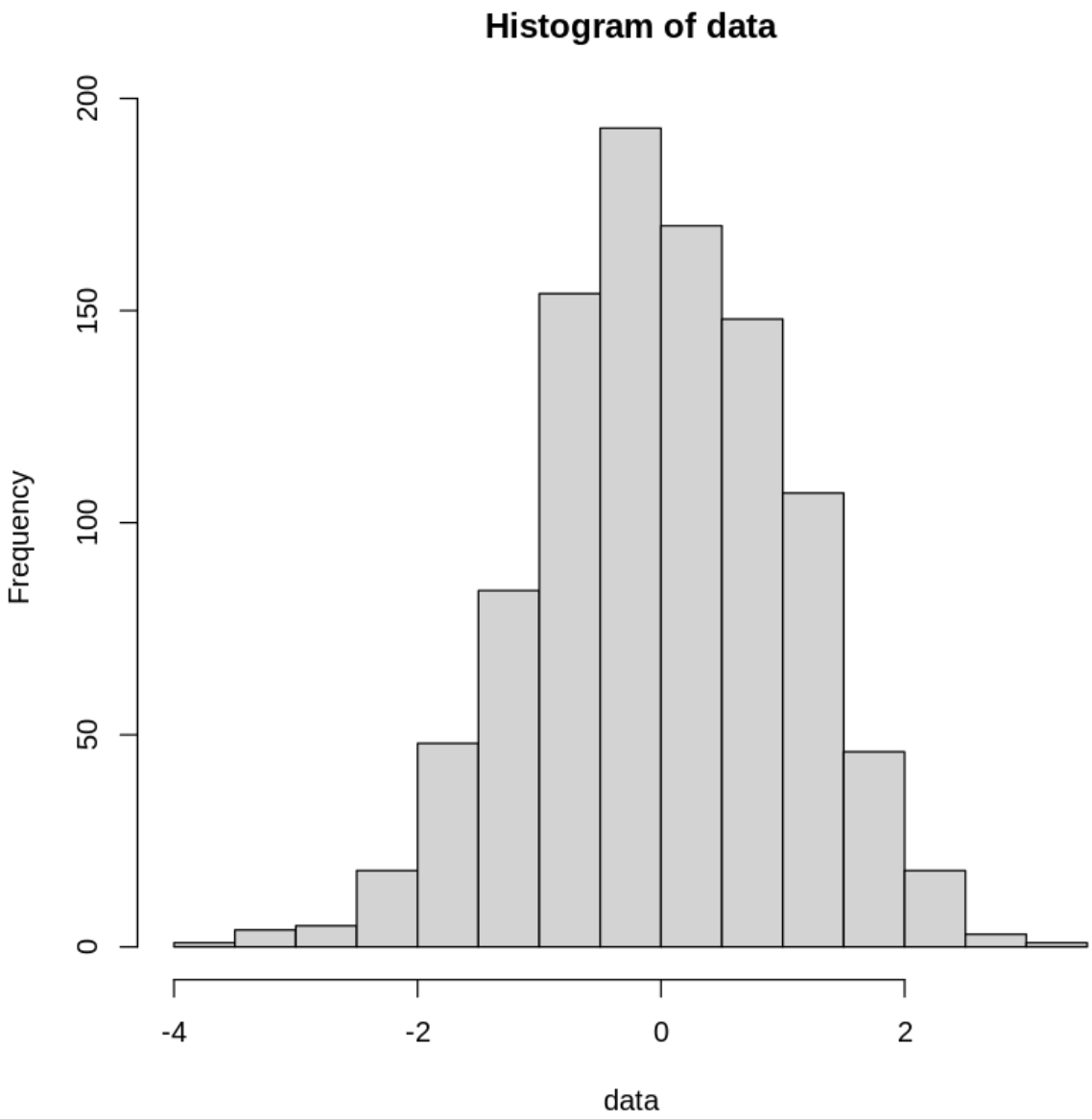

Figure 3: png

The y axis in a histogram is frequency, or the number of occurences. You can change it to probabilities instead.

```
hist(data, prob = TRUE)
```

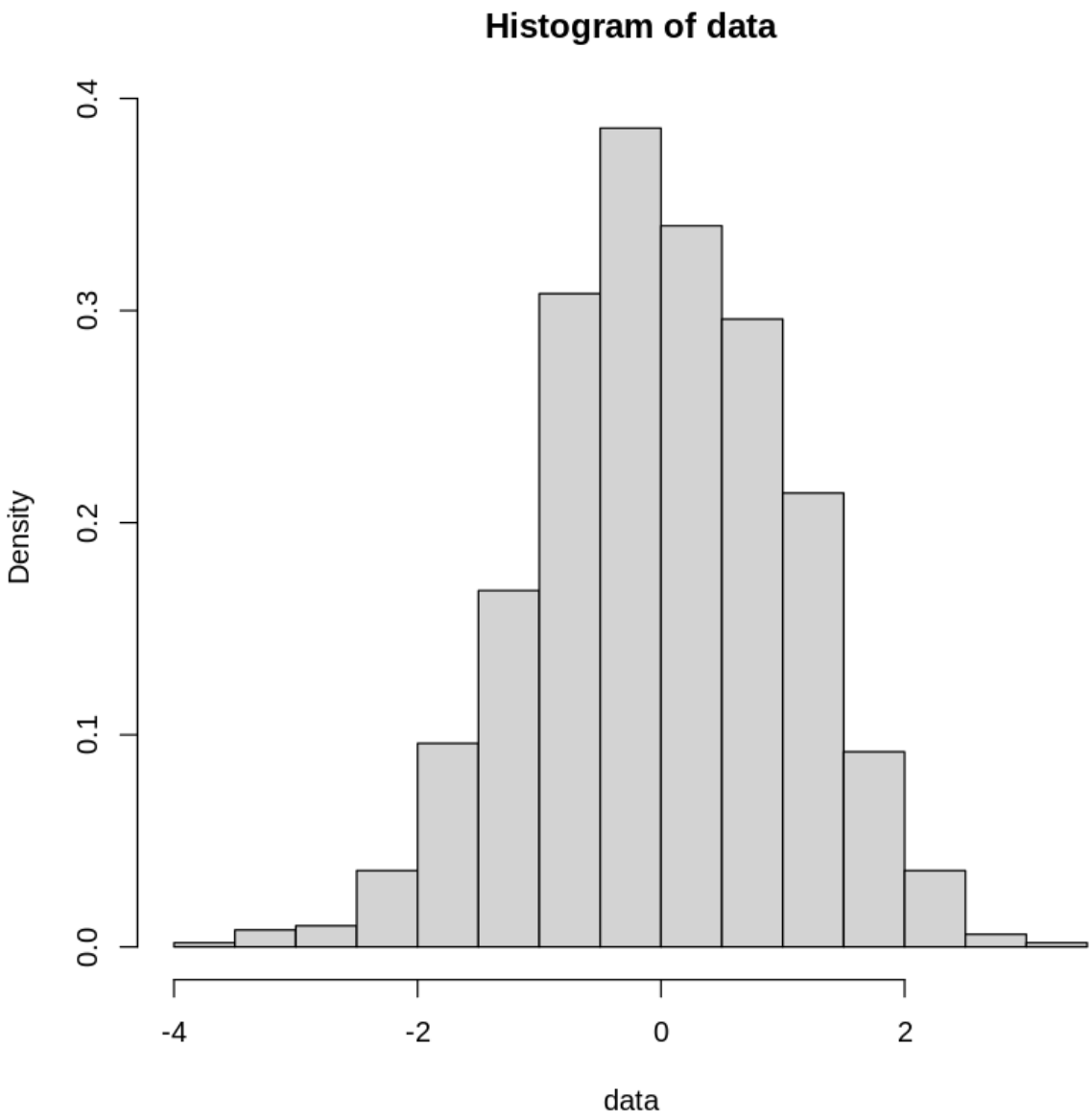

Figure 4: png

You can also choose your own bin boundaries. You might specify the number of bin breaks you want, or you can choose the exact bin breaks that you want.

```
hist(data, breaks = 8)                # Specify number of bin breaks
hist(data, breaks = c(seq(-5, 5, 1))) # Choose exact bin breaks
```

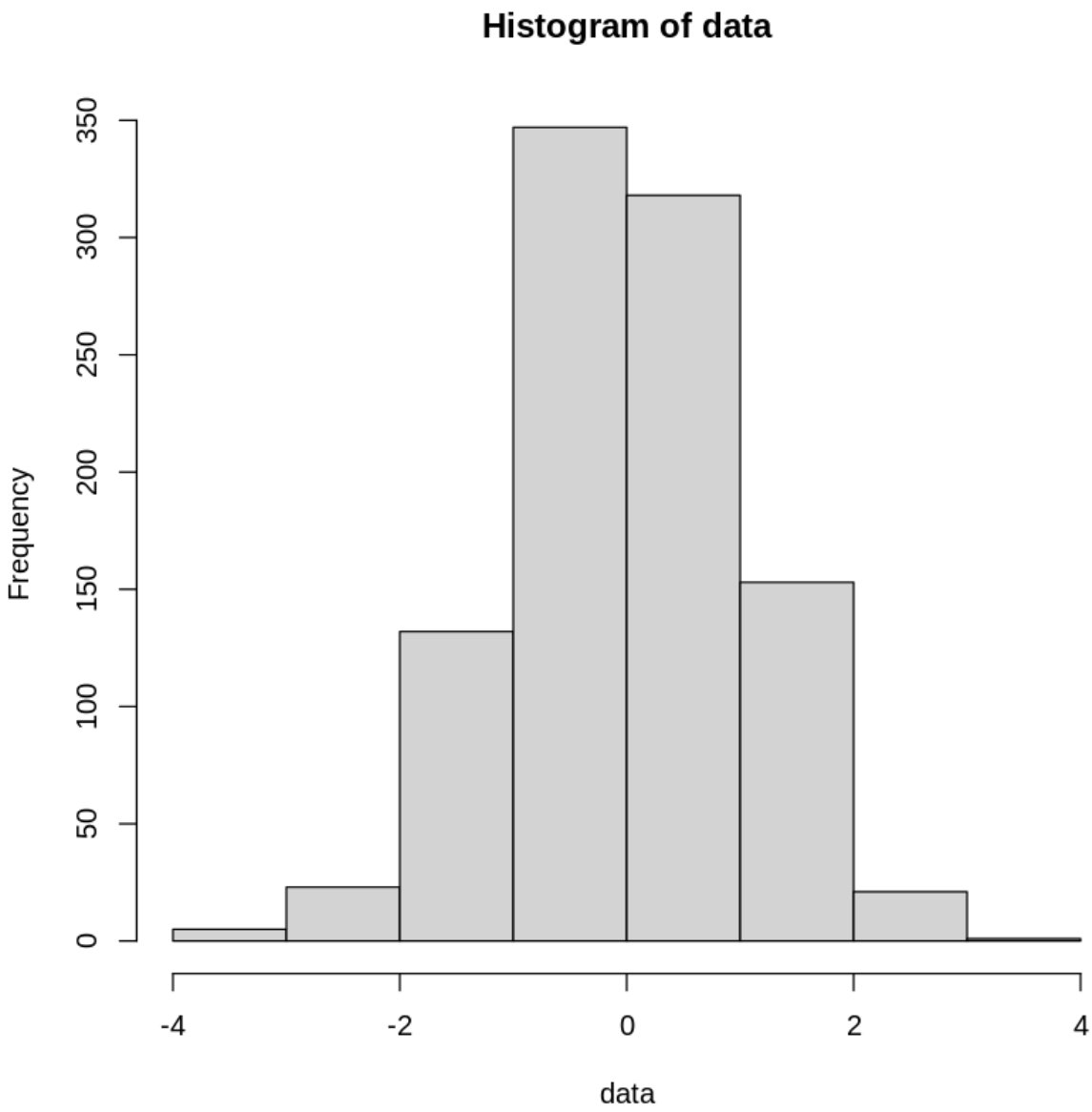
Histogram of data
Frequency
0
50
100
150
200
250
300
350
-4
-2
0
2
4
data

Figure 5: png

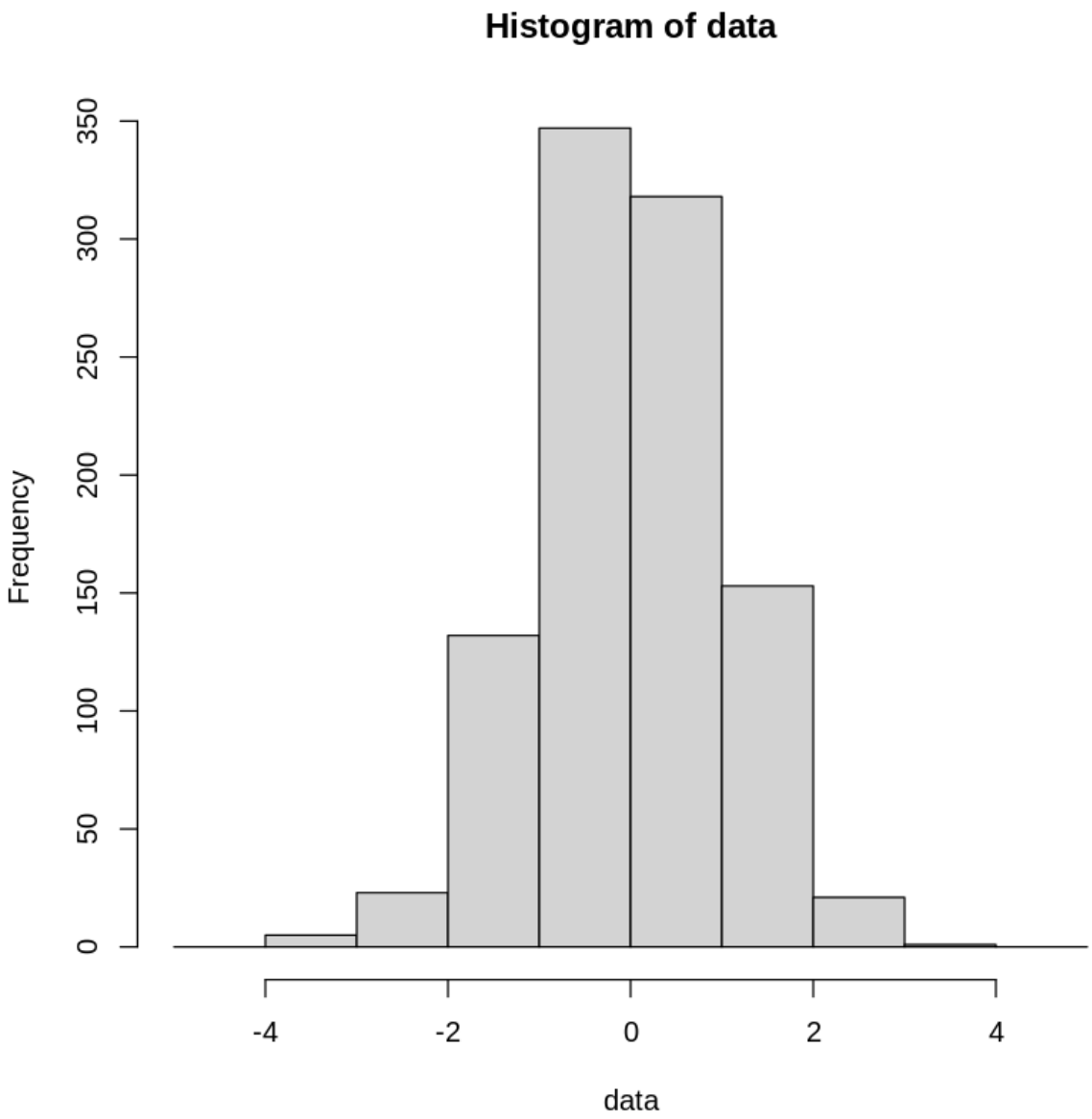

Figure 6: png

Content last modified on 14 September 2021.

See a problem? Tell us or edit the source.

How to create a box (and whisker) plot

Description

A box plot, or a box and whisker plot, shows the quartiles of a single variable from a dataset (one of which is the median) and may also show the outliers. It is a simplified way to see the distribution of a variable. Sometimes multiple box plots (one for each of several variables) are shown side-by-side on a plot, to compare the variables. How can we create such graphs?

Related topics:

- How to create basic plots
- How to add details to a plot
- How to create a histogram
- How to change axes, ticks, and scale in a plot
- How to create bivariate plots to compare groups (on website)
- How to plot interaction effects of treatments (on website)

Solution in R

We will create some fake data using vectors, for simplicity. But everything we show below works also if your data is in columns of a DataFrame.

```
patient_id     <- c(0,   1,   2,   3,   4,   5,   6,   7,   8,   9)
patient_height <- c(60,  64,  64,  65,  66,  66,  70,  72,  72,  76)
patient_weight <- c(141, 182, 169, 204, 138, 198, 180, 175, 244, 196)
```

We can use R's boxplot() function to make the plot.

```
boxplot(patient_weight)
```

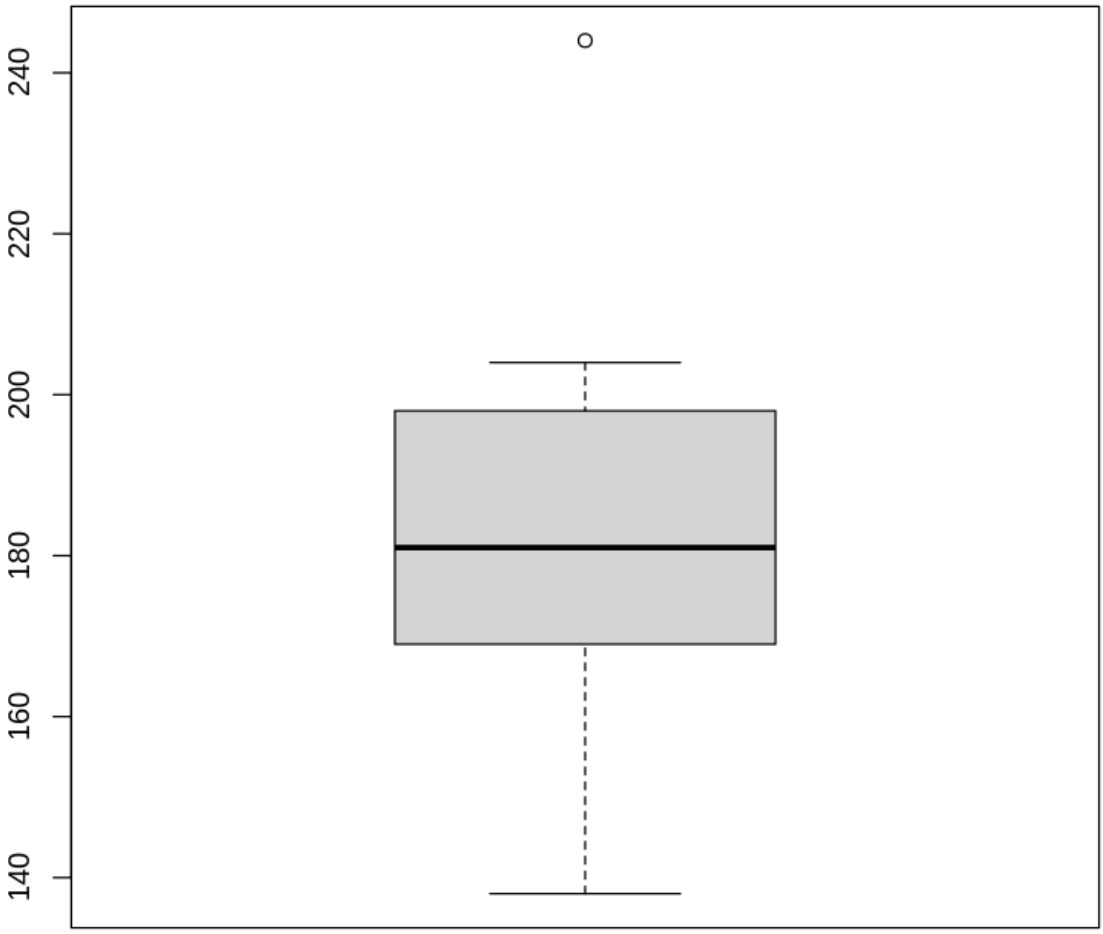

Figure 7: png

You can show more than one variable's box plot side-by-side by passing both variables into the boxplot() function.

```
boxplot(patient_height, patient_weight)
```

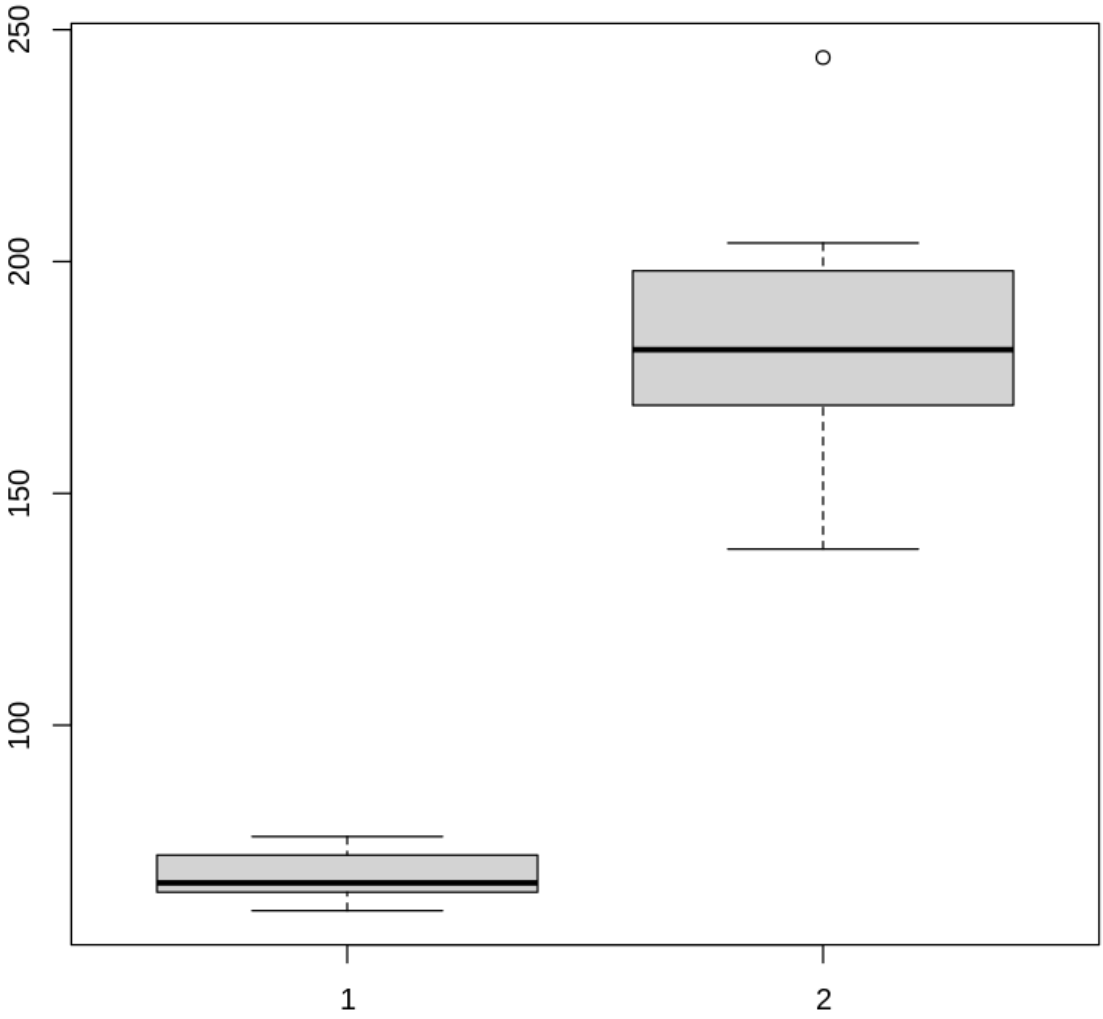

Figure 8: png

Content last modified on 14 September 2021.

See a problem? Tell us or edit the source.