Joshua French

Joshua French

2021-09-15

Contents

Preliminaries

I recommend you execute the following commands install packages we may use in this course.

```
# packages related to books
books = c("faraway", "alr4", "car", "rms")
install.packages(books)
# packages related to tidy/tidying data
tidy = c("broom", "tidyr", "dplyr")
install.packages(tidy)
# packages related to plotting
moreplots = c("ggplot2", "ggthemes", "lattice", "HH")
install.packages(moreplots)
# packages related to model diagnostics
diag = c("leaps", "lmtest", "gvlma", "caret")
install.packages(diag)
# packages related to workflow
workflow = c("remotes")
install.packages(workflow)
```

Lastly, we need to install the **perturb** package, which is currently not available through the `install.packages` function. To install this from the package developer's GitHub repository, we run the command below in the Console.

```
remotes::install_github(repo = "JohnHendrickx/Perturb")
```

Acknowledgments

The **bookdown** package (Xie 2021) was used to generate this book.

Chapter 1

R Foundations

Meaningful data analysis requires the use of computer software.

R statistical software is one of the most popular tools for data analysis both in academia and the broader workforce. In what follows, I will attempt to lay a foundation of basic knowledge and skills with R that you will need for data analysis. I make no attempt to be exhaustive, and many other important aspects of using R (like plotting) will be discussed later, as needed.

1.1 What is R?

- R is programming language and environment designed for statistical computing.
  - It was introduced by Robert Gentleman and Robert Ihaka in 1993.
  - It is modeled after the *S* programming language.
- R is free, open source, and runs on Windows, Macs, Linux, and other types of computers.
- R is an interactive programming language
  - You type and execute a command in the Console for immediate feedback in contrast to a compiled programming language, which compiles a program that is then executed.
- R is highly extendable.
  - Many user-created packages are available to extend the functionality beyond what is installed by default.
  - Users can write their own functions and easily add software libraries to R.

1.2 Where to get R (and R Studio Desktop)

R may be downloaded from the R Project's website. This link *should* bring you to the relevant page for downloading the software.

R Studio Desktop is a free "front end" for R provided by R Studio. R Studio Desktop makes doing data analysis with R much easier by adding an Integrated Development Environment (IDE) and providing many other features. Currently, you may download R Studio at this link. You may need to navigate the R Studio website directly if this link no longer functions.

Install R and R Studio Desktop before continuing. Then open R Studio Desktop as you continue to learn about R.

1.3 R Studio Layout

R Studio Desktop has four panes:

1. Console: the pane where the code is executed.
2. Source: the pane where you prepare commands to be executed.
3. Environment/History: the pane where you can see all the objects in your workspace, your command history, and other things.
4. The Files/Plot/Packages/Help: the pane where you navigate between directories, where plots can be viewed, where you can see the packages available to be loaded, and where you can get help.

To see all R Studio panes, press the keys `Shift + Ctrl + Alt + 0`

1.4 Running code, scripts, and comments

Code is executed in R by typing it in the Console and hitting enter.

Instead of typing all of your code in the Console and hitting enter, it's better to write your code in a Script and execute the code separately.

A new script can be obtained by executing File -> New File -> R Script or pressing "Ctrl + Shift + n" (on a PC) or "Cmd + Shift + n" on a Mac.

There are various ways to run code from a Script file. The most common ones are:

1. Highlight the code you want to run and hit the Run button at the top of the Script pane.
2. Highlight the code you want to run and press "Ctrl + Enter" on your keyboard. If you don't highlight anything, by default, R Studio runs the command the cursor currently lies on.

To save a script, click File -> Save or press "Ctrl + s" (on a PC) or "Cmd + s" (on a Mac).

A comment is a set of text ignored by R when submitted to the Console.

A comment is indicated by the `#` symbol. Nothing to the right of the `#` is executed in the Console.

To comment (or uncomment) multiple lines in R, highlight the code you want to comment and press "Ctrl + Shift + c" on a PC or "Cmd + Shift + c" on a Mac.

1.4.1 Example

Perform the following tasks:

1. Type `1+1` in the Console and hit enter.
2. Open a new Script in R Studio.
3. `mean(1:3)` in your Script file.
4. Type `# mean(1:3)` in your Script file.
5. Run the commands from the Script using an approach mentioned above.

1.5 Packages

Packages are collections of functions, data, and other objects that extend the functionality installed by default in R.

R packages can be installed using the `install.packages` function and loaded using the `library` function.

1.5.1 Example

Practice installing and loading a package by doing the following:

1. Install the set of **faraway** package by executing the command `install.packages("faraway")`.
2. Load the **faraway** package by executing the command `library(faraway)`.

1.6 Getting help

There are a number of helps to get help in R.

If you know the command for which you want help, then exectue `?command` in the Console. * e.g., `?lm` * This also may work with data sets, package names, object classes, etc.

If you want to search the documentation for a certain *topic*, then execute `??topic` in the Console. * If you need help deciphering an error, identifying packages to perform a certain analysis, how to do something better, then a web search is likely to help.

1.6.1 Example

Do the following:

1. Execute `?lm` in the Console to get help on the `lm` function, which is one of the main functions used for fitting linear models.
2. Execute `??logarithms` in the Console to search the R documentation for information about logarithms.
3. Do a web search for something along the lines of "How do I change the size of the axis labels in an R plot?"

1.7 Data types and structures

1.7.1 Basic data types

R has 6 basic ("atomic") vector types:

1. character - collections of characters. E.g., `"a"`, `"hello world!"`
2. double - decimal numbers. e.g., `1.2`, `1.0`
3. integer - whole numbers. In R, you must add `L` to the end of a number to specify it as an integer. E.g., `1L` is an integer but `1` is a double.
4. logical - boolean values, `TRUE` and `FALSE`
5. complex - complex numbers. E.g., `1+3i`
6. raw - a type to hold raw bytes.

The `typeof` function returns the R internal type or storage mode of any object.

Consider the following commands and output:

```
# determine basic data type
typeof(1)
#> [1] "double"
typeof(1L)
#> [1] "integer"
typeof("hello world!")
#> [1] "character"
```

1.7.2 Other important object types

There are other important types of objects in R that are not basic. We will discuss a few. The R Project manual provides additional information about available types.

1.7.2.1 Numeric

An object is `numeric` if it is of type `integer` or `double`. In that case, it's `mode` is said to be `numeric`.

The `is.numeric` function tests whether an object can be interpreted as numbers. We can use it to determine whether an object is `numeric`.

Some examples:

```
# is the object numeric?
is.numeric("hello world!")
#> [1] FALSE
is.numeric(1)
#> [1] TRUE
is.numeric(1L)
#> [1] TRUE
```

1.7.2.2 NULL

`NULL` is a special object to indicate an object is absent. An object having a length of zero is not the same thing as an object being absent.

1.7.2.3 NA

A "missing value" occurs when the value of something isn't known. R uses the special object `NA` to represent missing value.

If you have a missing value, you should represent that value as `NA`. Note: `"NA"` is not the same thing as `NA`.

1.7.2.4 Functions

A function is an object the performs a certain action or set of actions based on objects it receives from its arguments.

1.7.3 Data structures

R operates on data structures. A data structure is simply some sort of "container" that holds certain kinds of information

R has 5 basic data structures:

- vector
- matrix
- array
- data frame
- list

Vectors, matrices, and arrays are homogeneous objects that can only store a single data type at a time.

Data frames and lists can store multiple data types.

Vectors and lists are considered one-dimensional objects. A list is technically a vector. Vectors of a single type are atomic vectors. (https://cran.r-project.org/doc/manuals/r-release/R-lang.html#List-objects)

Matrices and data frames are considered two-dimensional objects.

Arrays can be n-dimensional objects.

This is summarized in the table below, which is based on a table in the first edition of Hadley Wickham's *Advanced R*.

dimensionality	homogeneous	heterogeneous
1d	vector	list
2d	matrix	data frame
nd	array	

1.8 Assignment

To store a data structure in the computer's memory we must assign it a name.

Data structures can be stored using the assignment operator `<-` or `=`.

Some comments:

- In general, both `<-` and `=` can be used for assignment.
- Pressing the "Alt" and "-" keys simultaneously on a PC or Linux machine (Option and - on a Mac) will insert `<-` into the R console and script files.
  - If you are creating an R Markdown file, then this command will only insert `<-` if you are in an R computing environment.
- `<-` and `=` are NOT synonyms, but can be used identically most of the time. It's safest to use `<-` for assignment.

Once an object has been assigned a name, it can be printed by executing the name of the object or using the `print` function.

1.8.1 Example

In the following code, we compute the mean of a vector and print the result.

```
# compute the mean of 1, 2, ..., 10 and assign the name m
m <- mean(1:10)
m # print m
#> [1] 5.5
print(m) # print m a different way
#> [1] 5.5
```

1.9 Vectors

A *vector* is a single-dimensional set of data of the same type.

1.9.1 Creation

The most basic way to create a vector is the `c` function.

The `c` function combines values into a vector or list.

e.g., the following commands create vectors of type numeric, character, and logical, respectively.

- `c(1, 2, 5.3, 6, -2, 4)`
- `c("one", "two", "three")`
- `c(TRUE, TRUE, FALSE, TRUE)`

1.9.2 Creating patterned vectors

R provides a number of functions for creating vectors following certain consistent patterns.

The `seq` (sequence) function is used to create an equidistant series of numeric values.

Some examples:

- `seq(1, 10)`: A sequence of numbers from 1 to 10 in increments of 1.
- `1:10`: A sequence of numbers from 1 to 10 in increments of 1.
- `seq(1, 20, by = 2)`: A sequence of numbers from 1 to 20 in increments of 2.
- `seq(10, 20, len = 100)`: A sequence of numbers from 10 to 20 of length 100.

The `rep` (replicate) function can be used to create a vector by replicating values.

Some examples:

- `rep(1:3, times = 3)`: Repeat the sequence 1, 2, 3 three times in a row.
- `rep(c("trt1", "trt2", "trt3"), times = 1:3)`: Repeat "trt1" once, "trt2" twice, and "trt3" three times.
- `rep(1:3, each = 3)`: Repeat each element of the sequence 1, 2, 3 three times.

1.9.3 Example

Execute the following commands in the Console to see what you get.

```
# vector creation
c(1, 2, 5.3, 6, -2, 4)
c("one", "two", "three")
c(TRUE, TRUE, FALSE, TRUE)
# sequences of values
seq(1, 10)
1:10
```

```
seq(1, 20, by = 2)
seq(10, 20, len = 100)
# replicated values
rep(1:3, times = 3)
rep(c("trt1", "trt2", "trt3"), times = 1:3)
rep(1:3, each = 3)
```

Vectors can be combined into a new object using the `c` function.

1.9.4 Example

Execute the following commands in the Console

```
v1 <- 1:5 # create a vector
v1 # print the vector
#> [1] 1 2 3 4 5
print(v1)
#> [1] 1 2 3 4 5
v2 <- c(1, 10, 11) # create a new vector
new <- c(v1, v2) # combine and assign the combined vectors
new # print the combined vector
#> [1]  1  2  3  4  5  1 10 11
```

1.9.5 Categorical vectors

Categorical data should be stored as a `factor` in R.

The `factor` function takes values that can be coerced to a character and converts them to an object of class `factor`.

Some examples:

```
# create some factor variables
f1 <- factor(rep(1:6, times = 3))
f1
#>  [1] 1 2 3 4 5 6 1 2 3 4 5 6 1 2 3 4 5 6
#> Levels: 1 2 3 4 5 6
f2 <- factor(c("a", 7, "blue", "blue", FALSE))
f2
#> [1] a     7     blue  blue  FALSE
#> Levels: 7 a blue FALSE
```

1.9.6 Example

Create a vector named `grp` that has two levels: `a` and `b`, where the first 7 values are `a` and the second 4 values are `b`.

1.9.7 Extracting parts of a vector

Subsets of the elements of a vector can be extracted by appending an index vector in square brackets `[]` to the name of the vector .

Let's create the numeric vector 2, 4, 6, 8, 10, 12, 14, 16.

```
# define a sequence 2, 4, ..., 16
a <- seq(2, 16, by = 2)
a
#> [1]  2  4  6  8 10 12 14 16
```

Let's access the 2nd, 4th, and 6th elements of `a`.

```
# extract subset of vector
a[c(2, 4, 6)]
#> [1]  4  8 12
```

Let's access all elements in `a` EXCEPT the 2nd, 4th, and 6th using the minus (`-`) sign in front of the index vector.

```
# extract subset of vector using minus
a[-c(2, 4, 6)]
#> [1]  2  6 10 14 16
```

Let's access all elements in `a` except elements 3 through 6.

```
a[-(3:6)]
#> [1]  2  4 14 16
```

1.10 Helpful functions

1.10.1 General functions

Some general functions commonly used to describe data objects:

- `length(x)`: length of `x`
- `sum(x)`: sum elements in `x`
- `mean(x)`: sample mean of elements in `x`
- `var(x)`: sample variance of elements in `x`
- `sd(x)`: sample standard deviation of elements in `x`
- `range(x)`: range (minimum and maximum) of elements in `x`
- `log(x)`: (natural) logarithm of elements in `x`
- `summary(x)`: a summary of `x`. Output changes depending on the class of `x`.
- `str(x)`: provides information about the structure of `x`. Usually, the class of the object and some information about its size.

1.10.2 Example

Run the following commands in the Console:

```
# common functions
x <- rexp(100) # sample 100 iid values from an Exponential(1) distribution
length(x) # length of x
sum(x) # sum of x
mean(x) # sample mean of x
var(x) # sample variance of x
sd(x) # sample standard deviation of x
range(x) # range of x
log(x) # logarithm of x
summary(x) # summary of x
str(x) # structure of x
```

1.10.3 Functions related to statistical distributions

Suppose that a random variable X has the `dist` distribution:

- `p[dist](q, ...)`: returns the cdf of X evaluated at `q`, i.e., $p = P(X \leq q)$.
- `q[dist](p, ...)`: returns the inverse cdf (or quantile function) of X evaluated at p, i.e., $q = \inf\{x : P(X \leq x) \geq p\}$.
- `d[dist](x, ...)`: returns the mass or density of X evaluated at x (depending on whether it's discrete or continuous).
- `r[dist](n, ...)`: returns an i.i.d. random sample of size `n` having the same distribution as X.
- The `...` indicates that additional arguments describing the parameters of the distribution may be required.

Execute `?Distributions` to get information about the distributions available in R by default.

1.10.4 Example

Execute the following commands in R to see the output. What is each command doing?

```
# statistical calculations
pnorm(1.96, mean = 0, sd = 1)
qunif(0.6, min = 0, max = 1)
dbinom(2, size = 20, prob = .2)
dexp(1, rate = 2)
rchisq(100, df = 5)
```

- `pnorm(1.96, mean = 0, sd = 1)` returns the probability that a standard normal random variable is less than or equal to 1.96, i.e., $P(X \leq 1.96)$.

- `qunif(0.6, min = 0, max = 1)` returns the value x such that $P(X \leq x) = 0.6$ for a uniform random variable on the interval $[0, 1]$.
- `dbinom(2, size = 20, prob = .2)` returns the probability that $P(X = 2)$ for X Binom$(n = 20, \pi = 0.2)$.
- `dexp(1, rate = 2)` evaluates the density of an exponential random variable with mean $= 1/2$ at $x = 1$.
- `rchisq(100, df = 5)` returns a sample of 100 observations from a chi-squared random variable with 5 degrees of freedom.

1.11 Data Frames

Data frames are two-dimensional data objects. Each column of a data frame is a vector (or variable) of possibly different data types. This is a *fundamental* data structure used by most of R's modeling software.

In general, I recommend *tidy data*, which means that each variable forms a column of the data frame, and each observation forms a row.

1.11.1 Creation

Data frames are created by passing vectors into the `data.frame` function.

The names of the columns in the data frame are the names of the vectors you give the `data.frame` function.

Consider the following simple example.

```
# create basic data frame
d <- c(1, 2, 3, 4)
e <- c("red", "white", "blue", NA)
f <- c(TRUE, TRUE, TRUE, FALSE)
df <- data.frame(d,e,f)
df
#>   d     e     f
#> 1 1   red  TRUE
#> 2 2 white  TRUE
#> 3 3  blue  TRUE
#> 4 4  <NA> FALSE
```

The columns of a data frame can be renamed using the `names` function on the data frame.

```
# name columns of data frame
names(df) <- c("ID", "Color", "Passed")
df
#>   ID Color Passed
#> 1  1   red   TRUE
#> 2  2 white   TRUE
```

```
#> 3  3  blue   TRUE
#> 4  4  <NA>  FALSE
```

The columns of a data frame can be named when you are first creating the data frame by using `name =` for each vector of data.

```
# create data frame with better column names
df2 <- data.frame(ID = d, Color = e, Passed = f)
df2
#>   ID Color Passed
#> 1  1   red   TRUE
#> 2  2 white   TRUE
#> 3  3  blue   TRUE
#> 4  4  <NA>  FALSE
```

1.11.2 Extracting parts of a data frame

The column vectors of a data frame may be extracted using `$` and specifying the name of the desired vector.

- `df$Color` would access the `Color` column of data frame `df`.

Part of a data frame can also be extracted by thinking of at as a general matrix and specifying the desired rows or columns in square brackets after the object name. For example, if we had a data frame named `df`:

- `df[1,]` would access the first row of `df`.
- `df[1:2,]` would access the first two rows of `df`.
- `df[,2]` would access the second column of `df`.
- `df[1:2, 2:3]` would access the information in rows 1 and 2 of columns 2 and 3 of `df`.

If you need to select multiple columns of a data frame by name, you can pass a character vector with column names in the column position of `[]`.

- `df[, c("Color", "Passed")]` would extract the `Color` and `Passed` columns of df.

1.11.3 Example

Execute the following commands in the Console:

```
# Extract parts of a data frame
df3 <- data.frame(numbers = 1:5,
                  characters = letters[1:5],
                  logicals = c(TRUE, TRUE, FALSE, TRUE, FALSE))
df3 # print df
df3$logicals # access the logicals vector of df3
df3[1, ] # access the first column of df3
```

```
df3[, 3] # access the third column of df3
df3[, 2:3] # access the column 2 and 3 of df3
df3[, c("numbers", "logicals")] # access the numbers and logical columns of df3
```

Students often can work more conveniently with vectors, so it is sometimes desirable to access a part of a data frame and assign it a new name for later use. For example, to access the `ID` column of `df2` and assign it the name `newID`, we could execute `newID <- df2$ID`.

1.11.4 Importing Data

The `read.table` function imports data from file into R as a data frame.

Usage: `read.table(file, header = TRUE, sep = ",")`

- `file` is the file path and name of the file you want to import into R.
  - If you don't know the file path, set `file = file.choose()` will bring up a dialog box asking you to locate the file you want to import.
- `header` specifies whether the data file has a header (variable labels for each column of data in the first row of the data file).
  - If you don't specify this option in R or use `header = FALSE`, then R will assume the file doesn't have any headings.
  - `header = TRUE` tells R to read in the data as a data frame with column names taken from the first row of the data file.
- `sep` specifies the delimiter separating elements in the file.
  - If each column of data in the file is separated by a space, then use `sep = " "`
  - If each column of data in the file is separated by a comma, then use `sep = ","`
  - If each column of data in the file is separated by a tab, then use `sep = "\t"`.

Here is an example reading a csv (comma separated file) with a header:

```
# import data as data frame
dtf <- read.table(file = "https://raw.githubusercontent.com/jfrench/DataWrangleViz/master/data/co
                  header = TRUE,
                  sep = ",")
str(dtf)
#> 'data.frame':    50 obs. of  7 variables:
#>  $ state_name: chr  "Alabama" "Alaska" "Arizona" "Arkansas" ...
#>  $ state_abb : chr  "AL" "AK" "AZ" "AR" ...
#>  $ deaths    : int  3831 142 6885 2586 19582 2724 5146 782 19236 9725 ...
#>  $ population: num  387000 96500 498000 238000 2815000 ...
#>  $ income    : int  25734 35455 29348 25359 31086 35053 37299 32928 27107 28838 ...
#>  $ hs        : num  82.1 91 85.6 82.9 80.7 89.7 88.6 87.7 85.5 84.3 ...
#>  $ bs        : num  21.9 27.9 25.9 19.5 30.1 36.4 35.5 27.8 25.8 27.3 ...
```

Note that the `read_table` function in the **readr** package and the `fread` function in the **data.table** package are perhaps better ways of reading in tabular data and use similar syntax.

1.12 Logical statements

1.12.1 Basic comparisons

Sometimes we need to know if the elements of an object satisfy certain conditions. This can be determined using the logical operators `<`, `<=`, `>`, `>=`, `==`, `!=`.

- `==` means equal to.
- `!=` means NOT equal to.

1.12.2 Example

Execute the following commands in R and see what you get. What is each statement performing?

```
# logical statements
# a <- seq(2, 16, by = 2) # creating the vector a
a
a > 10
a <= 4
a == 10
a != 10
```

1.12.3 And and Or statements

More complicated logical statements can be made using `&` and `|`.

- `&` means "and"
  - Only `TRUE & TRUE` returns `TRUE`. Otherwise the `&` operator returns `FALSE`.
- `|` means "or"
  - Only a single value in an `|` statement needs to be true for `TRUE` to be returned.

Note that:

- `TRUE & TRUE` returns `TRUE`
- `FALSE & TRUE` returns `FALSE`
- `FALSE & FALSE` returns `FALSE`
- `TRUE | TRUE` returns `TRUE`
- `FALSE | TRUE` returns `TRUE`
- `FALSE | FALSE` returns `FALSE`

```
# relationship between logicals & (and), | (or)
TRUE & TRUE
#> [1] TRUE
FALSE & TRUE
#> [1] FALSE
FALSE & FALSE
#> [1] FALSE
TRUE | TRUE
#> [1] TRUE
FALSE | TRUE
#> [1] TRUE
FALSE | FALSE
#> [1] FALSE
```

1.12.4 Example

Execute the following commands in R and see what you get.

```
# complex logical statements
(a > 6) & (a <= 10)
(a <= 4) | (a >= 12)
```

1.13 Subsetting with logical statements

Logical statements can be used to return parts of an object satisfying the appropriate criteria. Specifically, we pass logical statements within the square brackets used to access part of a data structure.

1.13.1 Example

Execute the following code:

```
# accessing parts of a vector using logicals
a
a < 6
a[a < 6]
a == 10
a[a == 10]
(a < 6) | ( a == 10)
a[(a < 6) | ( a == 10)]
# accessing parts of a data frame
# create a logical vector based on whether
# a state_abb in dtf is "CA" or "CO"
ca_or_co <- is.element(dtf$state_abb, c("CA", "CO"))
ca_or_co
```

```
# access the CA and CA rows of dtf
dtf[ca_or_co,]
```

1.14 Ecosystem debate

We will typically work with **base** R, which are commands and functions R offers by default.

The **tidyverse** (https://www.tidyverse.org) is a collection of R packages that provides a unified framework for data manipulation and visualization.

Since this course focuses more on modeling than data manipulation, I will typically focus on approaches in **base** R. I will use functions from the **tidyverse** when it greatly simplifies analysis, data manipulation, or visualization.

Chapter 2

Data exploration

Based on Chapter 1 of LMWR2, Chapter 1 of ALR4

2.1 Data analysis process

1. Define a statistical question of interest.
2. Collect relevant data.
3. Analyze the data.
4. Interpret your analysis.
5. Make a decision.

"The formulation of a problem is often more essential than its solution, which may be merely a matter of mathematical or experimental skill" - Albert Einstein

2.1.1 Problem Formulation

- Understand the physical background.
  - Statisticians often work in collaboration with others and need to understand something about the subject area.
- Understand the objective.
  - What are your goals?
  - Make sure you know what the client wants.
- Put the problem into statistical terms.
  - This is often the most challenging step and where irreparable errors are sometimes made.
  - That a statistical method can read in and process the data is not enough. The results of an inept analysis may be meaningless.

2.1.2 Data collection

Data collection:

- How the data were collected has a crucial impact on what conclusions can be made.
  - Are the data observational or experimental?
  - Are the data a sample of convenience or were they obtained via a designed sample survey?
- Is there nonresponse bias?
  - The data you do not see may be just as important as the data you do see.
- Are there missing values?
  - This is a common problem that is troublesome and time consuming to handle.
  - How are the data coded? How are the qualitative variables represented?
- What are the units of measurement?
- Beware of data entry errors and other corruption of the data.
  - Perform some data sanity checks.

2.2 Data exploration

An initial exploration of the data should be performed prior to any formal analysis or modeling.

Initial data analysis should consist of numerical summaries and appropriate plots.

2.2.1 Numerical summaries of data

Statistics can be used to numerically summarize aspects of the data:

- mean
- standard deviation (SD)
- maximum and minimum
- correlation
- other measures, as appropriate

2.2.2 Visual summaries of data

Plots can provide a useful visual summary of the data.

- For one numerical variable: boxplots, histograms, density plots, etc.
- For one categorial variable: bar charts.
- For two numerical variables: scatter plots.
- For one numerical and one categorical variables: parallel boxplots or density plots that distinguish between category level.
- For two categorical variables: panels of bar charts.
- For three or more variables: one or two variable plots with distinguishing colors or line types, interactive and dynamic graphics.

Good graphics are essential in data analysis.

- They help us to understand our data structure so that we can avoid mistakes.
- They help us decide on a model.
- They help communicate the results of our analysis.
- Graphics can be more convincing than text at times.

2.2.3 What to look for

When summarizing the data, look for:

- outliers
- data-entry errors
- skewness
- unusual distributions
- patterns or structure

2.3 Kidney Example

The National Institute of Diabetes and Digestive and Kidney Diseases conducted a study on 768 adult female Pima Native Americans living near Phoenix. The following variables were recorded:

- `pregnant` - number of times pregnant
- `glucose` - plasma glucose concentration at 2 hours in an oral glucose tolerance test
- `diastolic` - diastolic blood pressure (mm Hg)
- `triceps` - triceps skin fold thickness (mm)
- `insulin` - 2-hour serum insulin (mu U/ml)
- `bmi` - body mass index (weight in kg/(height in m2))
- `diabetes` - diabetes pedigree function
- `age` - age (years)
- `test` - test whether the patient showed signs of diabetes (coded zero if negative, one if positive).

The data may be obtained from the UCI Repository of machine learning databases at https://archive.ics.uci.edu/ml.

Let's load and examine the structure of the data

```
data(pima, package = "faraway")
str(pima) # structure
#> 'data.frame':    768 obs. of  9 variables:
#>  $ pregnant : int  6 1 8 1 0 5 3 10 2 8 ...
#>  $ glucose  : int  148 85 183 89 137 116 78 115 197 125 ...
#>  $ diastolic: int  72 66 64 66 40 74 50 0 70 96 ...
#>  $ triceps  : int  35 29 0 23 35 0 32 0 45 0 ...
```

```
#>  $ insulin  : int  0 0 0 94 168 0 88 0 543 0 ...
#>  $ bmi      : num  33.6 26.6 23.3 28.1 43.1 25.6 31 35.3 30.5 0 ...
#>  $ diabetes : num  0.627 0.351 0.672 0.167 2.288 ...
#>  $ age      : int  50 31 32 21 33 30 26 29 53 54 ...
#>  $ test     : int  1 0 1 0 1 0 1 0 1 1 ...
head(pima) # first six rows
#>   pregnant glucose diastolic triceps insulin  bmi
#> 1        6     148        72      35       0 33.6
#> 2        1      85        66      29       0 26.6
#> 3        8     183        64       0       0 23.3
#> 4        1      89        66      23      94 28.1
#> 5        0     137        40      35     168 43.1
#> 6        5     116        74       0       0 25.6
#>   diabetes age test
#> 1    0.627  50    1
#> 2    0.351  31    0
#> 3    0.672  32    1
#> 4    0.167  21    0
#> 5    2.288  33    1
#> 6    0.201  30    0
tail(pima) # last six rows
#>     pregnant glucose diastolic triceps insulin  bmi
#> 763        9      89        62       0       0 22.5
#> 764       10     101        76      48     180 32.9
#> 765        2     122        70      27       0 36.8
#> 766        5     121        72      23     112 26.2
#> 767        1     126        60       0       0 30.1
#> 768        1      93        70      31       0 30.4
#>     diabetes age test
#> 763    0.142  33    0
#> 764    0.171  63    0
#> 765    0.340  27    0
#> 766    0.245  30    0
#> 767    0.349  47    1
#> 768    0.315  23    0
```

2.3.1 Numerically summarizing the data

The `summary` command is a useful way to numerically summarize a data frame.

The `summary` function will compute the minimum, 0.25 quantile, mean, median, 0.75 quantile, and maximum of a `numeric` variable.

The `summary` function will count the number of values having each level for a `factor` variable.

Let's summarize the `pima` data frame.

```
summary(pima)
#>     pregnant         glucose        diastolic
#>  Min.   : 0.000   Min.   :  0.0   Min.   :  0.00
#>  1st Qu.: 1.000   1st Qu.: 99.0   1st Qu.: 62.00
#>  Median : 3.000   Median :117.0   Median : 72.00
#>  Mean   : 3.845   Mean   :120.9   Mean   : 69.11
#>  3rd Qu.: 6.000   3rd Qu.:140.2   3rd Qu.: 80.00
#>  Max.   :17.000   Max.   :199.0   Max.   :122.00
#>     triceps          insulin            bmi
#>  Min.   : 0.00   Min.   :  0.0   Min.   : 0.00
#>  1st Qu.: 0.00   1st Qu.:  0.0   1st Qu.:27.30
#>  Median :23.00   Median : 30.5   Median :32.00
#>  Mean   :20.54   Mean   : 79.8   Mean   :31.99
#>  3rd Qu.:32.00   3rd Qu.:127.2   3rd Qu.:36.60
#>  Max.   :99.00   Max.   :846.0   Max.   :67.10
#>     diabetes           age             test
#>  Min.   :0.0780   Min.   :21.00   Min.   :0.000
#>  1st Qu.:0.2437   1st Qu.:24.00   1st Qu.:0.000
#>  Median :0.3725   Median :29.00   Median :0.000
#>  Mean   :0.4719   Mean   :33.24   Mean   :0.349
#>  3rd Qu.:0.6262   3rd Qu.:41.00   3rd Qu.:1.000
#>  Max.   :2.4200   Max.   :81.00   Max.   :1.000
```

2.3.2 Cleaning the data

Cleaning data involves finding and correcting data quality issues.

The `pima` data set has some odd characterics:

- The minimum `diastolic` blood pressure is zero.
  - That's generally an indication of a health problem.
- The `test` variable appears to be `numeric` but should be a `factor` (categorical) variable.
- Many other variables have unusual zeros.
  - Look for anything unusual or unexpected, perhaps indicating a data-entry error.

Let's look at the first 40 sorted `diastolic` values.

```
sort(pima$diastolic)[1:40]
#>  [1]  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
#> [18]  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
#> [35]  0 24 30 30 38 40
```

The first 35 values of `diastolic` are zero. That's a problem.

- It seems that 0 was used in place of a missing value.

- This is very bad since 0 is a real number and this problem may be overlooked, which can lead to faulty analysis!
- This is why we must check our data carefully for things that don't make sense.

The value for missing data in R is `NA`.

Several variables share this problem. Let's set the 0s that should be missing values to `NA`.

```
pima$diastolic[pima$diastolic == 0]  <- NA
pima$glucose[pima$glucose == 0] <- NA
pima$triceps[pima$triceps == 0]  <- NA
pima$insulin[pima$insulin == 0] <- NA
pima$bmi[pima$bmi == 0] <- NA
```

The `test` variable is a categorical variable, not numerical.

- R thinks the `test` variable is `numeric`.
- In R, a categorical variable is a `factor`.
- We need to convert the `test` variable to a `factor`.

Let's convert `test` to a factor.

```
 pima$test <- factor(pima$test)
 summary(pima$test)
#>   0   1
#> 500 268
```

500 of the cases were negative and 268 were positive. We can provide more descriptive labels using the `levels` function.

We change the `0` and `1` levels to `negative` and `positive` to make the data more descriptive. A `summary` of the updates `test` variable shows why this is useful.

```
levels(pima$test) <- c("negative", "positive")
summary(pima$test)
#> negative positive
#>      500      268
```

2.4 Visualizing data with base graphics

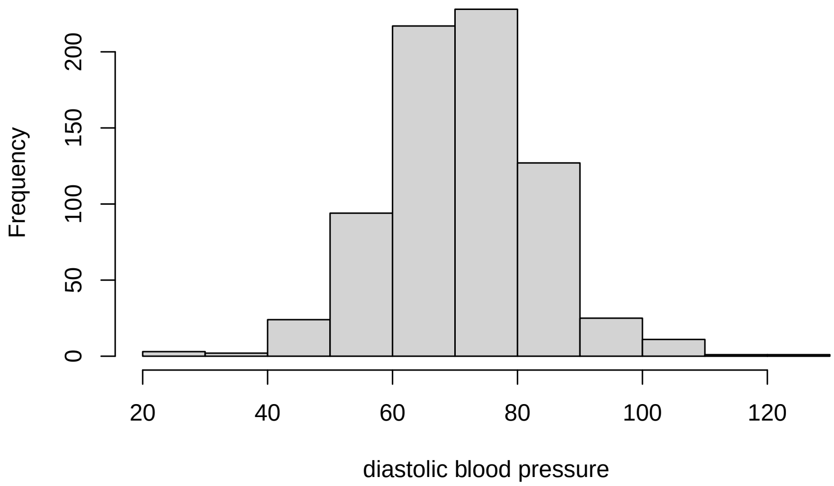
2.4.1 Histograms

The `hist` function can be used create a histogram of a numerical vector.

- The labels of the plot can be customized using the `xlab` and `ylab` arguments.
- The main title of the plot can be customized using the `main` argument.

Here is a slightly customized histogram of diastolic blood pressure.

```
hist(pima$diastolic, xlab = "diastolic blood pressure", main = "")
```

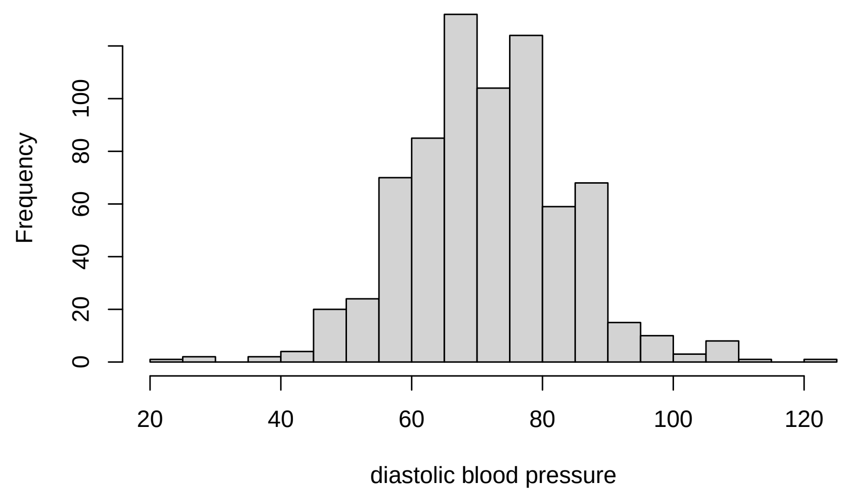


The histogram is approximately bell-shaped and centered around 70.

We can change the number of breaks in the histogram by specifying the `breaks` argument of the `hist` function.

Consider how the plot changes below.

```
hist(pima$diastolic, xlab = "diastolic blood pressure", main = "", breaks = 20)
```



2.4.2 Density plots

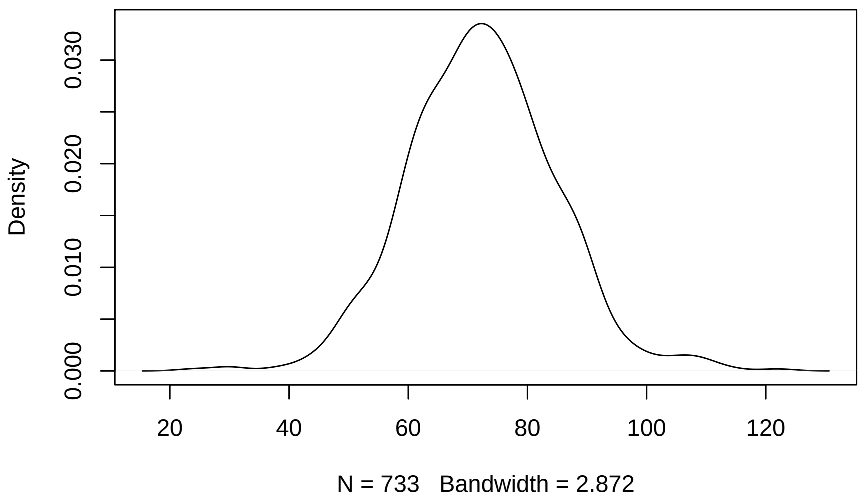
Many people prefer the density plot over the histogram because the histogram is more sensitive to its options.

A density plot is essentially a smoothed version of a histogram.

- It isn't as blocky.
- It sometimes has weird things happen at the boundaries.

The `plot` and `density` function can be combined to construct a density plot.

```
plot(density(pima$diastolic, na.rm = TRUE), main = "")
```



In the example above, we have to specify `na.rm = TRUE` so that the density is only computed using the non-missing values.

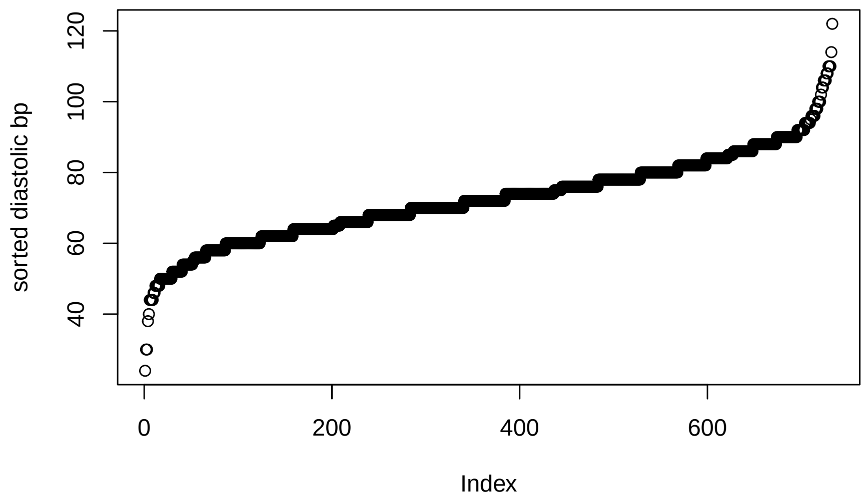
2.4.3 Index plots

An index plot is a scatter plot of a `numeric` variable versus the index of each value (i.e., the position of the value in the vector).

- This is most useful for sorted vectors.

A scatter plot of sorted `numeric` values versus their index can be used to identify outliers and see whether the data has many repeated values.

```
plot(sort(pima$diastolic), ylab = "sorted diastolic bp")
```



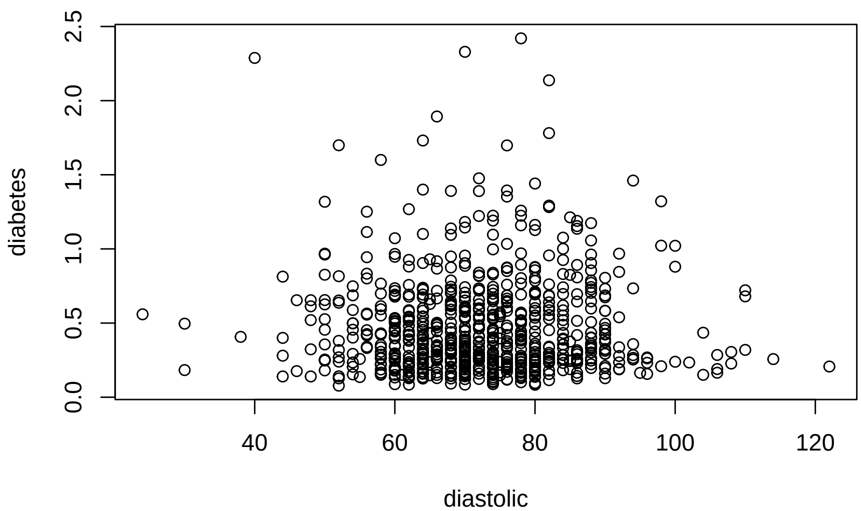
The flat spots in the plot above show that the `diastolic` variable has many repeated values.

2.4.4 Bivariate scatter plots

Bivariate scatter plots can be used to identify the relationship between two `numeric` variables.

A scatter plot of diabetes vs diastolic blood pressure is shown below.

```
plot(diabetes ~ diastolic, data = pima)
```

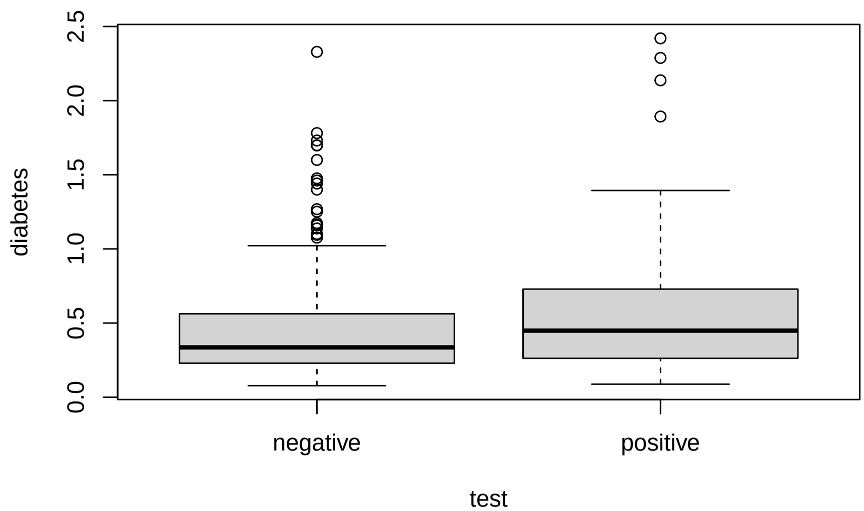


There is no clear pattern in the points, so it's difficult to claim a relationship between the two variables.

2.4.5 Bivariate boxplots

A parallel boxplot of `diabetes` score versus `test` result is shown below.

```
plot(diabetes ~ test, data = pima)
```



The median `diabetes` score seems to be a bit higher for positive tests in comparison to the negative tests.

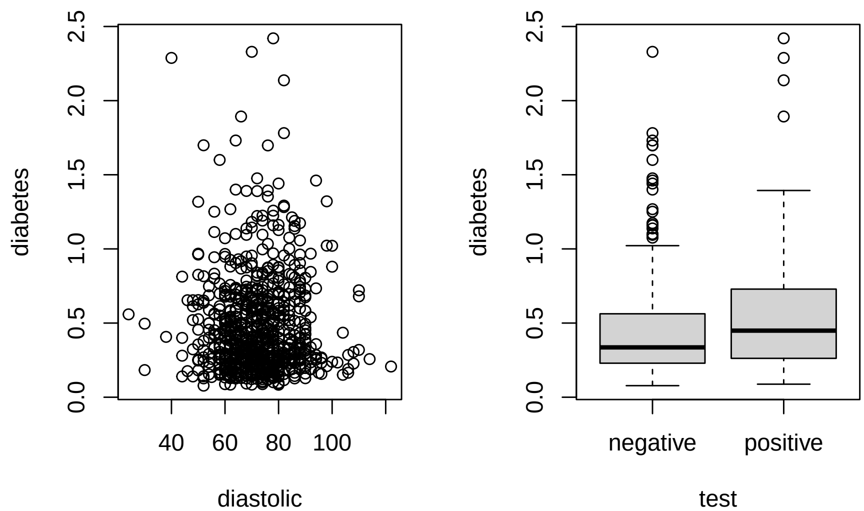
2.4.6 Multiple plots in one figure

The `par` function can be used to construct multiple plots in one figure.

- The `mfrow` argument can be used to specify the number of rows and columns of plots you need.

A 1 by 2 set of plots is shown below.

```
par(mfrow = c(1, 2))
plot(diabetes ~ diastolic, data = pima)
plot(diabetes ~ test, data = pima)
```




```
par(mfrow = c(1, 1)) # reset to a single plot
```

2.5 Visualizing data with ggplot2

The previous plots were created using R's **base** graphics system.

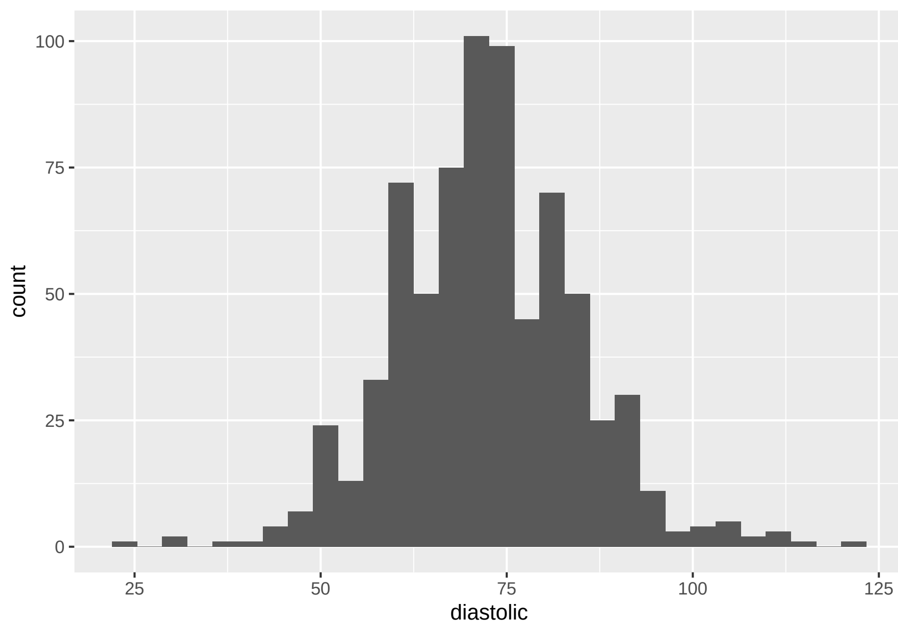
- **base** graphics are fast and simple to produce while looking professional.

A fancier alternative is to construct plots using the **ggplot2** package.

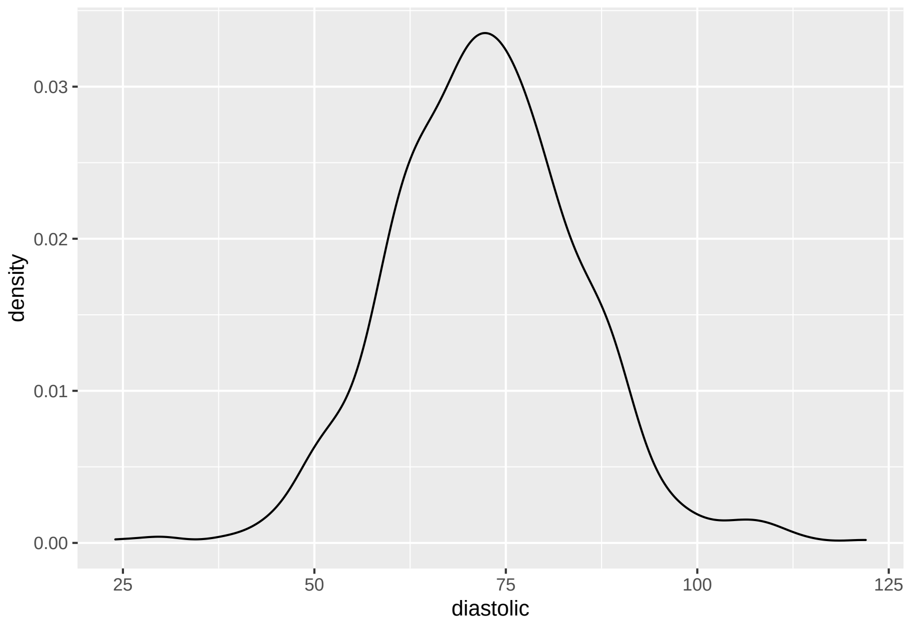
In its simplest form, to construct a (useful) plot in **ggplot2**, you need to provide:

- A `ggplot` object.
  - This is usually the object that holds your data frame.
  - The data frame is passed to `ggplot` via the `data` argument.
- A geometry object
  - Roughly speaking, this is the *kind* of plot you want.
  - e.g., `geom_hist` for a histogram, `geom_point` for a scatter plot, `geom_density` for a density plot.
- An aesthetic mapping
  - Aesthetic mappings describe how variables in the data are mapped to visual properties of a geometry.
  - This is where you specify which variable with be the `x` variable, the `y` variable, which variable will control color in the plots, etc.

```
library(ggplot2)
ggpima <-  ggplot(pima)
ggpima + geom_histogram(aes(x=diastolic))
#> `stat_bin()` using `bins = 30`. Pick better value with
#> `binwidth`.
#> Warning: Removed 35 rows containing non-finite values
#> (stat_bin).
```

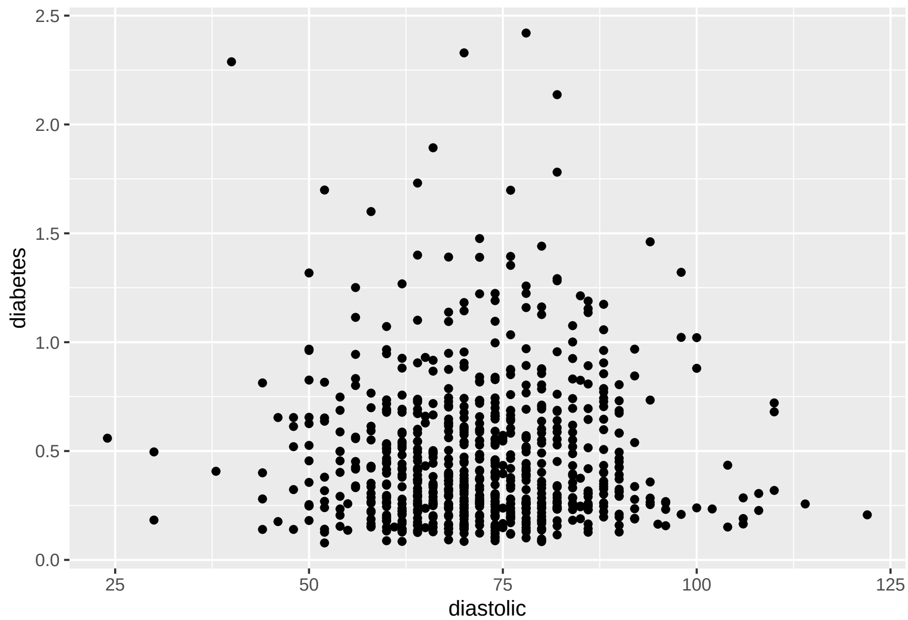



```
ggpima + geom_density(aes(x = diastolic))
#> Warning: Removed 35 rows containing non-finite values
#> (stat_density).
```




```
ggpima + geom_point(aes(x = diastolic, y = diabetes))
#> Warning: Removed 35 rows containing missing values
#> (geom_point).
```

2.5.3 A ggplot2 scatter plot

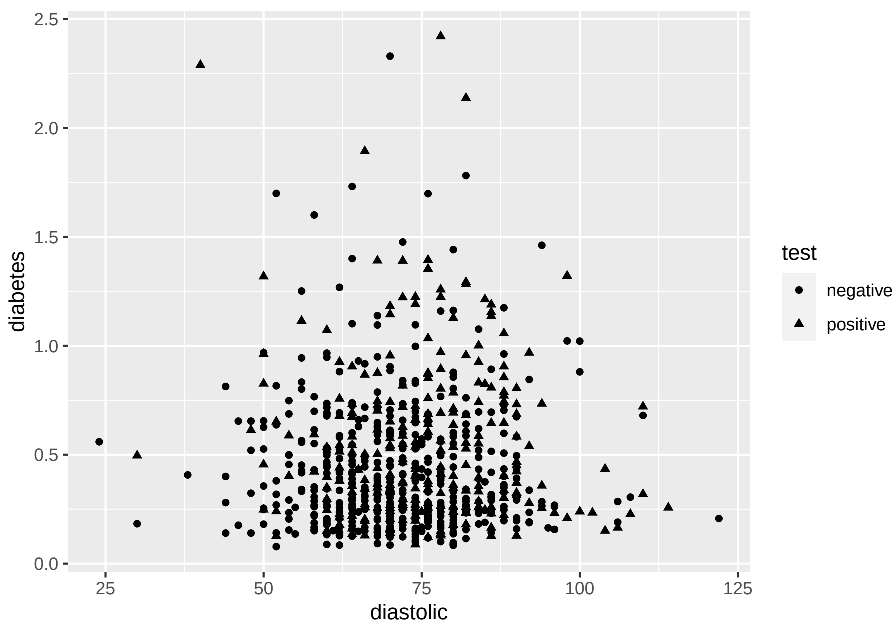


2.5.4 Scaling ggplot2 plots

In general, *scaling* is the process by which **ggplot2** maps variables to unique values. When this is done for discrete variables, **ggplot2** will often scale the variable to distinct colors, symbols, or sizes, depending on the aesthetic mapped.

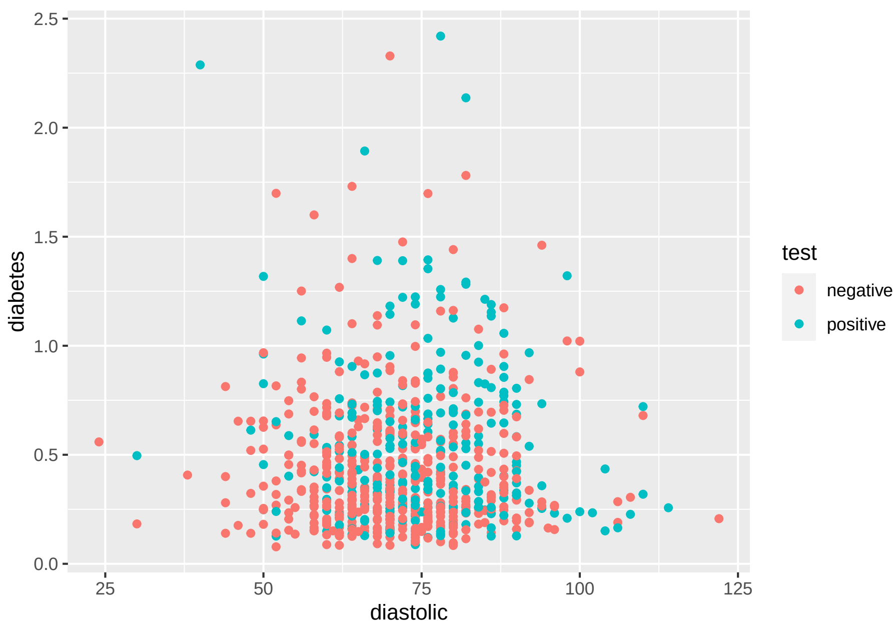
In the example below, we map the `test` variable to the `shape` aesthetic, which is then scaled to different shapes for the different `test` levels.

```
ggpima +
  geom_point(aes(x = diastolic, y = diabetes, shape = test))
#> Warning: Removed 35 rows containing missing values
#> (geom_point).
```



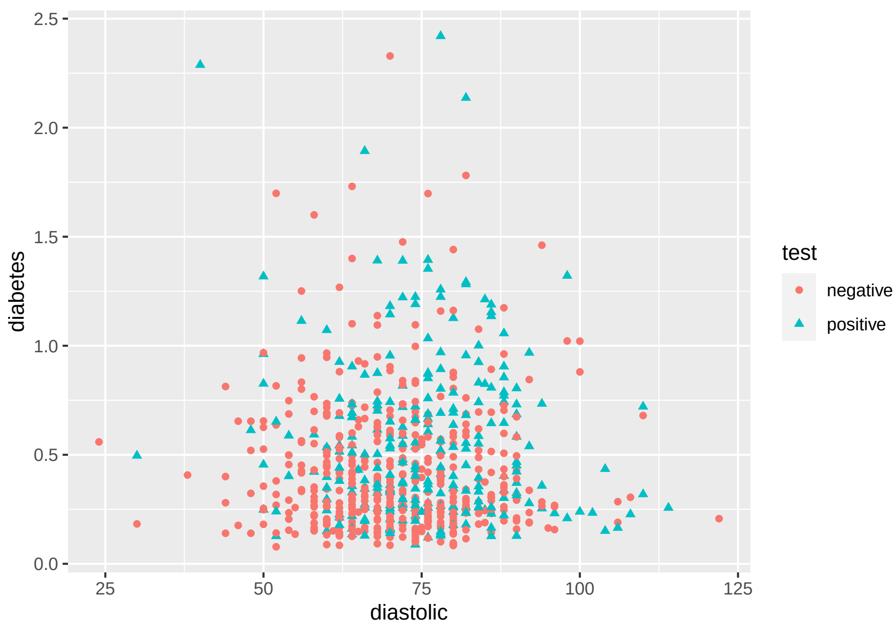
Alternatively, we can map the `test` variable to the `color` aesthetic, which creates a plot with different colors for each observation based on the `test` level.

```
ggpima +
  geom_point(aes(x = diastolic, y = diabetes, color = test))
#> Warning: Removed 35 rows containing missing values
#> (geom_point).
```



We can even combine these two aesthetic mappings in a single plot to get different colors and symbols for each level of `test`.

```
ggpima +
  geom_point(aes(x = diastolic, y = diabetes, shape = test, color = test))
#> Warning: Removed 35 rows containing missing values
#> (geom_point).
```

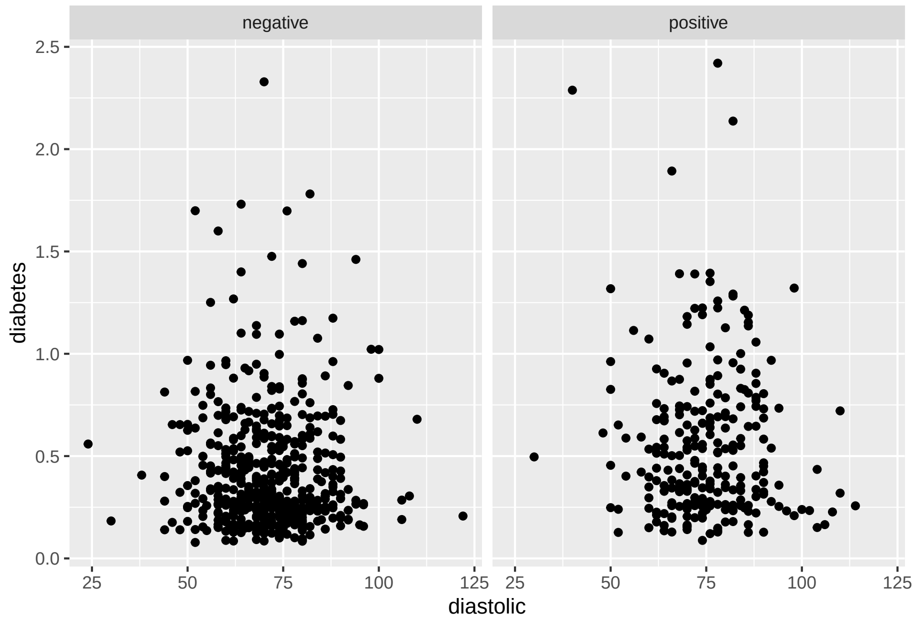


2.5.5 Facetting in ggplot2

Facetting creates separate panels (facets) of a data frame based on one or more facetting variables.

Below, we facet the data by the `test` result.

```
ggpima +
  geom_point(aes(x = diastolic, y = diabetes)) +
  facet_grid(~ test)
#> Warning: Removed 35 rows containing missing values
#> (geom_point).
```



2.5.6 Summary of ggplot2

To create a **ggplot2** plot:

- Create a`ggplot` object using the `ggplot` function.
  - Specify the data frame the data is contained in (e.g., the data frame is `pima`).
- Specify the geometry for the plot (the kind of plot you want to produce)
- Specify the aesthetics using `aes`.
  - The aesthetic specifies what you see, such as position in the x or y direction or aspects such as shape or color.
  - The aesthetic can be specified in the geometry, or if you have consistent aesthetics across multiple geometries, in the `ggplot` statement.

The advantage of **ggplot2** is more apparent in producing complex plots involving more than two variables.

- **ggplot2** makes it easy to plot the data for each group with different colors, symbols, line types, etc.
- **ggplot2** will automatically provide a legend mapping the attributes to the different groups.
- **ggplot2** makes it easy to create separate panels with plots for the observations having a certain characteristic.

2.6 Summary of data exploration

You should use both numerical and graphical summaries of data **prior** to modeling data.

Data exploration helps us to:

- Gain understanding about our data
- Identify problems or unusual features of our data
- Identify patterns in our data
- Decide on a modeling approach for the data
- etc.

Chapter 3

Review of probability, random variables, and random vectors

3.1 Probability Basics

Points ω in Ω are called **sample outcomes**, **realizations**, or **elements**.

A **set** is a (possibly empty) collection of elements.

- Sets are denoted as a set of elements between curly braces, i.e., $\{\omega_1, \omega_2, \ldots\}$, where the ω_i are elements of Ω.

Set A is a subset of set B if every element of A is an element of B.

- This is denoted as $A \subseteq B$, meaning that A is a subset of B.
- Subsets of Ω are **events**.

The **null set** or **empty set**, $\emptyset$, is the set with no elements, i.e., $\{\}$.

- The empty set is a subset of any other set.

A function P that assigns a real number $P(A)$ to every event A is a probability distribution if it satisfies three properties:

1. $P(A) \geq 0$ for all $A \in \Omega$
2. $P(\Omega) = P(\omega \in \Omega) = 1$
3. If $A_1, A_2, \ldots$ are disjoint, then $P\left(\bigcup_{i=1}^{\infty} A_i\right) = \sum_{i=1}^{\infty} P(A_i)$.

A set of events $\{A_i : i \in I\}$ are **independent** if

$$P\left(\cap_{i \in J} A_i\right) = \prod_{i \in J} P(A_i)$$

for every finite subset $J \subseteq I$.

3.2 Random Variables

A **random variable** Y is a mapping/function

$$Y : \Omega \to \mathbb{R}$$

that assigns a real number $Y(\omega)$ to each outcome ω.

- We typically drop the (ω) part for simplicity.

The **cumulative distribution function (CDF)** of Y, F_Y, is a function $F_Y : \mathbb{R} \to [0, 1]$ defined by

$$F_Y(y) = P(Y \leq y).$$

- The subscript of F indicates the random variable the CDF describes.
- E.g., F_X denotes the CDF of the random variable X and F_Y denotes the CDF of the random variable Y.
- The subscript can be dropped when the context makes it clear what random variable the CDF describes.

The support of Y, $\mathcal{S}$, is the smallest set such that $P(Y \in \mathcal{S}) = 1$.

3.2.1 Discrete random variables

Y is a **discrete** random variable if it takes countably many values $\{y_1, y_2, \dots\} = \mathcal{S}$.

The **probability mass function (pmf)** for Y is $f_Y(y) = P(Y = y)$, where $y \in \mathbb{R}$, and must have the following properties:

1. $0 \leq f_Y(y) \leq 1$.
2. $\sum_{y \in \mathcal{S}} f_Y(y) = 1$.

Additionally, the following statements are true:

- $F_Y(c) = P(Y \leq c) = \sum_{y \in \mathcal{S}: y \leq c} f_Y(y)$.
- $P(Y \in A) = \sum_{y \in A} f_Y(y)$ for some event A.
- $P(a \leq Y \leq b) = \sum_{y \in \mathcal{S}: a \leq y \leq b} f_Y(y)$.

The expected value, mean, or first moment of Y is defined as

$$E(Y) = \sum_{y \in \mathcal{S}} y f_Y(y),$$

assuming the sum is well-defined.

The **variance** of Y is defined as

$$var(Y) = E(Y - E(Y))^2 == \sum_{y \in \mathcal{S}} (y - E(Y))^2 f_Y(y).$$

The **standard deviation** of Y is

$$SD(Y) = \sqrt{var(Y)}.$$

3.2.1.1 Example (Bernoulli)

A random variable $Y \sim \mathsf{Bernoulli}(\pi)$ if $\mathcal{S} = 0, 1$ and $P(Y = 1) = \pi$, where $\pi \in (0, 1)$.

The pmf of a Bernoulli random variable is

$$f_Y(y) = \pi^y (1 - \pi)^{(1-y)}.$$

Determine $E(Y)$ and $var(Y)$.

3.2.2 Continuous random variables

Y is a continuous random variable if there exists a function $f_Y(y)$ such that:

1. $f_Y(y) \geq 0$ for all y,
2. $\int_{-\infty}^{\infty} f_Y(y)dy = 1$,
3. $a \leq b$, $P(a < Y < b) = \int_a^b f_Y(y)dy$.

The function f_Y is called the **probability density function (pdf)**.

Additionally, $F_Y(y) = \int_{-\infty}^{y} f_Y(y)dy$ and $f_Y(y) = F'_Y(y)$ for any point y at which F_Y is differentiable.

The expected value of a continuous random variables Y is defined as

$$E(Y) = \int_{-\infty}^{\infty} y f_Y(y)dy = \int_{y \in \mathcal{S}} y f_Y(y).$$

assuming the integral is well-defined.

The **variance** of a continuous random variable Y is defined by

$$var(Y) = E(Y - E(Y))^2 = \int_{-\infty}^{\infty} (y - E(Y))^2 f_Y(y)dy = \int_{y \in \mathcal{S}} (y - E(Y))^2 f_Y(y)dy.$$

The **standard deviation** of Y is

$$SD(Y) = \sqrt{var(Y)}.$$

3.2.3 Useful facts for transformation of random variables

Let Y be a random variable and $c \in \mathbb{R}$ be a constant. Then:

- $E(cY) = cE(Y)$
- $E(c + Y) = c + E(Y)$
- $var(cY) = c^2 var(Y)$
- $var(c + Y) = var(Y)$

3.3 Multivariate distributions

3.3.1 Basic properties

Let $Y_1, Y_2, \ldots, Y_n$ denote n random variables with supports $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_n$, respectively.

If the random variables are jointly (all) discrete, then the joint pmf $f(y_1, \ldots, y_n) = P(Y_1 = y_1, \ldots, Y_n = y_n)$ satisfies the following properties:

1. $0 \leq f(y_1, \ldots, y_n) \leq 1$,
2. $\sum_{y_1 \in \mathcal{S}_1} \cdots \sum_{y_n \in \mathcal{S}_n} f(y_1, \ldots, y_n) = 1$,
3. $P((Y_1, \ldots, Y_n) \in A) = \sum_{(y_1, \ldots, y_n) \in A} f(y_1, \ldots, y_n)$.

In this context,

$$E(Y_1 \cdots Y_n) = \sum_{y_1 \in \mathcal{S}_1} \cdots \sum_{y_n \in \mathcal{S}_n} y_1 \cdots y_n f(y_1, \ldots, y_n).$$

In general,

$$E(g(Y_1, \ldots, Y_n)) = \sum_{y_1 \in \mathcal{S}_1} \cdots \sum_{y_n \in \mathcal{S}_n} g(y_1, \ldots, y_n) f(y_1, \ldots, y_n),$$

where g is a function of the random variables.

If the random variables are jointly continuous, then $f(y_1, \ldots, y_n) = P(Y_1 = y_1, \ldots, Y_n = y_n)$ is the joint pdf if it satisfies the following properties:

1. $f(y_1, \ldots, y_n) \geq 0$,
2. $\int_{y_1 \in \mathcal{S}_1} \cdots \int_{y_n \in \mathcal{S}_n} f(y_1, \ldots, y_n) dy_n \cdots dy_1 = 1$,
3. $P((Y_1, \ldots, Y_n) \in A) = \int \cdots \int_{(y_1, \ldots, y_n) \in A} f(y_1, \ldots, y_n) dy_n \ldots dy_1$.

In this context,

$$E(Y_1 \cdots Y_n) = \int_{y_1 \in \mathcal{S}_1} \cdots \int_{y_n \in \mathcal{S}_n} y_1 \cdots y_n f(y_1, \ldots, y_n) dy_n \ldots dy_1.$$

In general,

$$E(g(Y_1, \ldots, Y_n)) = \int_{y_1 \in \mathcal{S}_1} \cdots \int_{y_n \in \mathcal{S}_n} g(y_1, \ldots, y_n) f(y_1, \ldots, y_n) dy_n \cdots dy_1,$$

where g is a function of the random variables.

3.3.2 Marginal distributions

If the random variables are jointly discrete, then the marginal pmf of Y_1 is

$$f_{Y_1}(y_1) = \sum_{y_2 \in \mathcal{S}_2} \cdots \sum_{y_n \in \mathcal{S}_n} f(y_1, \ldots, y_n).$$

Similarly, if the random variables are jointly continuous, then the marginal pdf of Y_1 is

$$f_{Y_1}(y_1) = \int_{y_2 \in \mathcal{S}_2} \cdots \int_{y_n \in \mathcal{S}_n} f(y_1, \ldots, y_n) dy_n \cdots dy_2.$$

3.3.3 Independence of random variables

Random variables X and Y are independent if

$$F(x, y) = F_X(x) F_Y(y).$$

Alternatively, X and Y are independent if

$$f(x, y) = f_X(x) f_Y(y).$$

3.3.4 Conditional distributions

Let X and Y be random variables. The conditional distribution of X given $Y = y$, denoted $X|Y = y$ is

$$f(x|y) = f(x, y)/f_Y(y).$$

3.3.5 Covariance

The covariance between random variables X and Y is

$$cov(X, Y) = E[(X - E(X))(Y - E(Y))] = E(XY) - E(X)E(Y).$$

3.3.6 Useful facts for transformations of multiple random variables

Let a and b be scalar constants. Then:

- $E(aY) = aE(Y)$
- $E(a + Y) = a + E(Y)$
- $E(aY + bZ) = aE(Y) + bE(Z)$
- $var(aY) = a^2 var(Y)$
- $var(a + Y) = var(Y)$
- $cov(aY, bZ) = abcov(Y, Z)$.
- $var(Y + Z) = var(Y) + var(Z) + 2cov(Y, Z)$.

3.4 Random vectors

3.4.1 Definition

Let $\mathbf{y} = (Y_1, Y_2, \ldots, Y_n)^T$ be an $n \times 1$ vector of random variables. $\mathbf{y}$ is a random vector.

- A vector is always defined to be a column vector, even if the notation is ambiguous.

3.4.2 Mean, variance, and covariance

The mean of a random vector is

$$E(\mathbf{y}) = \begin{pmatrix} E(Y_1) \\ E(Y_2) \\ \vdots \\ E(Y_n) \end{pmatrix}.$$

The variance (covariance) of a random vector is

$$\begin{aligned} var(\mathbf{y}) &= E(\mathbf{y}\mathbf{y}^T) - E(\mathbf{y})E(\mathbf{y})^T \\ &= \begin{pmatrix} var(Y_1) & cov(Y_1, Y_2) & \dots & cov(Y_1, Y_n) \\ cov(Y_2, Y_1) & var(Y_2) & \dots & cov(Y_2, Y_n) \\ \vdots & \vdots & \vdots & \vdots \\ cov(Y_n, Y_1) & cov(Y_n, Y_2) & \dots & var(Y_n) \end{pmatrix}. \end{aligned}$$

Let $\mathbf{x} = (X_1, X_2, \dots, X_n)^T$ and $\mathbf{y} = (Y_1, Y_2, \dots, Y_n)^T$ be $n \times 1$ random vectors.

The covariance between two random vectors is

$$cov(\mathbf{x}, \mathbf{y}) = E(\mathbf{x}, \mathbf{y}^T) - E(\mathbf{x})E(\mathbf{y})^T.$$

3.5 Properties of transformations of random vectors

Define:

- $\mathbf{a}$ to be an $n \times 1$ vector of constants
- A to be an $m \times n$ matrix of constants
- $\mathbf{x} = (X_1, X_2, \dots, X_n)^T$ to be an $n \times 1$ random vector
- $\mathbf{y} = (Y_1, Y_2, \dots, Y_n)^T$ to be an $n \times 1$ random vector
- $\mathbf{z} = (Z_1, Z_2, \dots, Z_n)^T$ to be an $n \times 1$ random vector
- $0_{n \times n}$ to be an $n \times n$ matrix of zeros.

Then:

- $E(A\mathbf{y}) = AE(\mathbf{y}), E(\mathbf{y}A^T) = E(\mathbf{y})A^T$.
- $E(\mathbf{x} + \mathbf{y}) = E(\mathbf{x}) + E(\mathbf{y})$
- $var(A\mathbf{y}) = A\ var(\mathbf{y})A^T$
- $cov(\mathbf{x} + \mathbf{y}, \mathbf{z}) = cov(\mathbf{x}, \mathbf{z}) + cov(\mathbf{y}, \mathbf{z})$
- $cov(\mathbf{x}, \mathbf{y} + \mathbf{z}) = cov(\mathbf{x}, \mathbf{y}) + cov(\mathbf{x}, \mathbf{z})$
- $cov(A\mathbf{x}, \mathbf{y}) = A\ cov(\mathbf{x}, \mathbf{y})$
- $cov(\mathbf{x}, A\mathbf{y}) = cov(\mathbf{x}, \mathbf{y})A^T$
- $var(a) = 0_{n \times n}$
- $cov(\mathbf{a}, \mathbf{y}) = 0_{n \times n}$
- $var(\mathbf{a} + \mathbf{y}) = var(\mathbf{y})$

3.6 Multivariate normal (Gaussian) distribution

3.6.1 Definition

$\mathbf{y} = (Y_1, \ldots, Y_n)^T$ has a multivariate normal distribution with mean $E(\mathbf{y}) = \mu$ (an $n \times 1$ vector) and covariance $\text{var}(\mathbf{y}) = \mathbf{\Sigma}$ (an $n \times n$ matrix) if the joint pdf is

$$f(\mathbf{y}) = \frac{1}{(2\pi)^{n/2}|\mathbf{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{y} - \mu)^T \Sigma^{-1}(\mathbf{y} - \mu)\right),$$

where $|\mathbf{\Sigma}|$ is the determinant of $\mathbf{\Sigma}$. Note that $\mathbf{\Sigma}$ must be symmetric and positive definite.

We would denote this as $\mathbf{y} \sim N(\mu, \mathbf{\Sigma})$.

3.6.2 Useful facts

Important fact: A linear function of a multivariate normal random vector (i.e., $\mathbf{a} + A\mathbf{y}$) is also multivariate normal (though it could collapse to a single random variable if A is a $1 \times n$ vector).

Application: Suppose that $\mathbf{y} \sim N(\mu, \Sigma)$. For an $m \times n$ matrix of constants A, $A\mathbf{y} \sim N(A\mu, A\Sigma A^T)$.

3.7 Example 1

3.7.1 Bernoulli distribution

A random variable $Y \sim \text{Bernoulli}(\theta)$ when $\mathcal{S} = \{0, 1\}$ and the pmf of a Bernoulli random variable is

$$f(y \mid \theta) = \theta^y (1 - \theta)^{(1-y)}.$$

- Determine $E(Y)$

- Determine $var(Y)$

3.7.2 Binomial distribution

A random variable $Y \sim \text{Bin}(n, \theta)$ when $\mathcal{S} = \{0, 1, 2, \ldots, n\}$ and the pmf is

$$f(y \mid \theta) = \binom{n}{y} \theta^y (1-\theta)^{(n-y)}.$$

Alternatively, let $Y_1, Y_2, \ldots, Y_n \overset{i.i.d.}{\sim} \text{Bernoulli}(\theta)$. Then $Y = \sum_{i=1}^n Y_i \sim \text{Bin}(n, \theta)$.

- Determine $E(Y)$

- Determine $var(Y)$

Assume $Y \sim \mathrm{Bin}(20, 0.4)$.

- Determine $F(8)$.

- Determine $P(8 \leq Y \leq 10)$.

3.7.3 Poisson Distribution

$Y \sim \text{Poisson}(\theta)$ when $\Omega = \{0, 1, 2, \ldots\}$ and

$$f(y \mid \theta) = \frac{1}{y!}\theta^y e^{-\theta}.$$

- Determine $E(Y)$

- Determine $var(Y)$

Assume $Y \sim$ Poisson(4).

- Determine $F(12)$

- Determine $P(15 \leq Y \leq 20)$.

3.8 Example 2

Gasoline is to be stocked in a bulk tank once at the beginning of each week and then sold to individual customers. Let Y_1 denote the proportion of the capacity of the bulk tank that is available after the tank is stocked at the beginning of the week. Because of the limited supplies, Y_1 varies from week to week. Let Y_2 denote the proportion of the capacity of the bulk tank that is sold during the week. Because Y_1 and Y_2 are both proportions, both variables are between 0 and 1. Further, the amount sold, y_2, cannot exceed the amount available, y_1. Suppose the joint density function for Y_1 and Y_2 is given by

$$f(y_1, y_2) = 3y_1;\ 0 \leq y_2 \leq y_1 \leq 1.$$

3.8.1 Problem 1

Determine $P(0 \leq Y_1 \leq 0.5;\ 0.25 \leq Y_2)$

3.8.2 Problem 2

Determine f_{Y_1} and f_{Y_2}

3.8.3 Problem 3

Determine $E(Y_1)$ and $E(Y_2)$

3.8.4 Problem 4

Determine $var(Y_1)$ and $var(Y_2)$

3.8.5 Problem 5

Determine $E(Y_1 Y_2)$

3.8.6 Problem 6

Determine $cov(Y_1, Y_2)$

3.8.7 Problem 7

Determine the mean and variance of $\mathbf{a}^T\mathbf{y}$, where $\mathbf{a} = (1, -1)^T$ and $\mathbf{y} = (Y_1, Y_2)^T$. This is the expectation and variance of the difference between the amount of gas available and the amount of gas sold.

Chapter 4

Useful matrix facts

A **matrix** is a two-dimensional array of values, symbols, or other objects (depending on the context). We will assume that our matrices contain numbers or random variables. Context will make it clear which is being represented.

- Matrices are commonly denoted by bold capital letters like $\mathbf{A}$ or $\mathbf{B}$, but this will sometimes be simplified to capital letters like A or B.

4.1 Notation

A matrix $\mathbf{A}$ with m rows and n columns (an $m \times n$ matrix) will be denoted as

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{1,1} & \mathbf{A}_{2,1} & \cdots & \mathbf{A}_{1,n} \\ \mathbf{A}_{2,1} & \mathbf{A}_{2,1} & \cdots & \mathbf{A}_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}_{m,1} & \mathbf{A}_{m,2} & \cdots & \mathbf{A}_{m,n} \end{bmatrix},$$

where $\mathbf{A}_{i,j}$ denotes the element in row i and column j of matrix $\mathbf{A}$.

A **column vector** is a matrix with a single column. A **row vector** is a matrix with a single row.

- Vectors are commonly denoted with bold lowercase letters such as $\mathbf{a}$ or $\mathbf{b}$, but this may be simplified to lowercase letters such as a or b.

A $p \times 1$ column vector $\mathbf{a}$ may constructed as

$$\mathbf{a} = [a_1, a_2, \ldots, a_p]^T = \begin{bmatrix} a_1 & a_2 & \cdots & a_p \end{bmatrix}^T = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_p \end{bmatrix}.$$

4.2 Basic mathematical properties

4.2.1 Addition and subtraction

Consider matrices **A** and **B** with identical sizes $m \times n$.

We add **A** and **B** by adding the element in position i, j of **B** with the element in position i, j of A, i.e.,

$$(\mathbf{A} + \mathbf{B})_{i,j} = \mathbf{A}_{i,j} + \mathbf{B}_{i,j}.$$

Similarly, if we subtract **B** from matrix **A**, then we subtract the element in position i, j of **B** from the element in position i, j of **A**, i.e.,

$$(\mathbf{A} - \mathbf{B})_{i,j} = \mathbf{A}_{i,j} - \mathbf{B}_{i,j}.$$

Example:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} + \begin{bmatrix} 2 & 9 & 1 \\ 1 & 3 & 1 \end{bmatrix} = \begin{bmatrix} 3 & 11 & 4 \\ 5 & 8 & 7 \end{bmatrix}.$$

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} - \begin{bmatrix} 2 & 9 & 1 \\ 1 & 3 & 1 \end{bmatrix} = \begin{bmatrix} -1 & -7 & 2 \\ 3 & 2 & 5 \end{bmatrix}.$$

4.2.2 Scalar multiplication

A matrix multiplied by a scalar value $c \in \mathbb{R}$ is the matrix obtained by multiplying each element of the matrix by c. If **A** is a matrix and $c \in \mathbb{R}$, then

$$(c\mathbf{A})_{i,j} = c\mathbf{A}_{i,j}.$$

Example:

$$3 \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} = \begin{bmatrix} 3 \cdot 1 & 3 \cdot 2 & 3 \cdot 3 \\ 3 \cdot 4 & 3 \cdot 5 & 3 \cdot 6 \end{bmatrix} = \begin{bmatrix} 3 & 6 & 9 \\ 12 & 15 & 18 \end{bmatrix}.$$

4.2.3 Matrix multiplication

Consider two matrices **A** and **B**. The matrix product **AB** is only defined if the number of columns in **A** matches the number of rows in **B**.

Assume **A** is an $m \times n$ matrix and **B** is an $n \times p$ matrix. **AB** will be an $m \times p$ matrix and

$$(\mathbf{AB})_{i,j} = \sum_{k=1}^{n} \mathbf{A}_{i,k}\mathbf{B}_{k,j}.$$

Example:

$$\begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\end{bmatrix}\begin{bmatrix}1 & 4\\ 2 & 5\\ 3 & 6\end{bmatrix} = \begin{bmatrix}1\cdot 1+2\cdot 2+3\cdot 3 & 1\cdot 4+2\cdot 5+3\cdot 6\\ 4\cdot 1+5\cdot 2+6\cdot 3 & 4\cdot 4+5\cdot 5+6\cdot 6\end{bmatrix} = \begin{bmatrix}14 & 32\\ 32 & 77\end{bmatrix}.$$

4.2.4 Associative property

Addition and multiplication satisfy the associative property for matrices. Assuming that the matrices $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$ have the sizes required to do the operations below, then

$$(\mathbf{A}+\mathbf{B})+\mathbf{C} = \mathbf{A}+(\mathbf{B}+\mathbf{C})$$

and

$$(\mathbf{AB})\mathbf{C} = \mathbf{A}(\mathbf{BC}).$$

4.2.5 Distributive property

Matrix operations satisfy the distributive property. Assuming that the matrices $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$ have the sizes required to do the operations below, then

$$\mathbf{A}(\mathbf{B}+\mathbf{C}) = \mathbf{AB}+\mathbf{AC} \quad \text{and} \quad (\mathbf{A}+\mathbf{B})\mathbf{C} = \mathbf{AC}+\mathbf{BC}.$$

4.2.6 No commutative property

In general, matrix multiplication does not satisfy the commutative property, i.e.,

$$AB \neq BA,$$

even when the matrix sizes allow the operation to be performed.

Example:

$$\begin{bmatrix}1 & 2\end{bmatrix}\begin{bmatrix}1\\ 2\end{bmatrix} = \begin{bmatrix}5\end{bmatrix}$$

$$\begin{bmatrix}1\\ 2\end{bmatrix}\begin{bmatrix}1 & 2\end{bmatrix} = \begin{bmatrix}1 & 2\\ 2 & 4\end{bmatrix}$$

4.3 Transpose and related properties

4.3.1 Definition

The **transpose** of a matrix, denoted T as a superscript, exchanges the rows and columns of the matrix. More formally, the i, j element of $\mathbf{A}^T$ is the j, i element of $\mathbf{A}$, i.e., $(\mathbf{A}^T)_{i,j} = \mathbf{A}_{j,i}$.

Example:

$$\begin{bmatrix} 2 & 9 & 3 \\ 4 & 5 & 6 \end{bmatrix}^T = \begin{bmatrix} 2 & 4 \\ 9 & 5 \\ 3 & 6 \end{bmatrix}.$$

4.3.2 Transpose and mathematical operations

Assume that the matrices $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$ have the sizes required to perform the operations below. Additionally, assume that $c \in \mathbb{R}$ is a scalar constant.

The following properties are true:

- $c^T = c$
- $(\mathbf{A} + \mathbf{B})^T = \mathbf{A}^T + \mathbf{B}^T$
- $(\mathbf{AB})^T = \mathbf{B}^T\mathbf{A}^T$, which can be extended to $(\mathbf{ABC})^T = \mathbf{C}^T\mathbf{B}^T\mathbf{A}^T$, etc.
- $(\mathbf{A}^T)^T = \mathbf{A}$

4.4 Special matrices

4.4.1 Square matrices

A matrix is **square** if the number of rows equals the number of columns. The **diagonal elements** of an $n \times n$ square matrix $\mathbf{A}$ are the elements $\mathbf{A}_{i,i}$ for $i = 1, 2, \ldots, n$. Any non-diagonal elements of $\mathbf{A}$ are called off-diagonal elements.

4.4.2 Identity matrix

The $n \times n$ identity matrix $\mathbf{I}_{n\times n}$ is 1 for its diagonal elements and 0 for its off-diagonal elements. Context often makes it clear what the dimensions of an identity matrix are, so $\mathbf{I}_{n\times n}$ is often simplified to $\mathbf{I}$ or I.

Example:

$$\mathbf{I}_{3\times 3} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

4.4.3 Symmetric

A matrix $\mathbf{A}$ is **symmetric** if $\mathbf{A} = \mathbf{A}^T$, i.e., $\mathbf{A}_{i,j} = \mathbf{A}_{j,i}$ for all potential i, j.

- A symmetric matrix must be square.

4.4.4 Idempotent

A matrix is **idempotent** if $\mathbf{AA} = \mathbf{A}$

- An idempotent matrix must be square.

4.5 Matrix inverse

An $n \times n$ matrix $\mathbf{A}$ is invertible if there exists a matrix $\mathbf{B}$ such that $\mathbf{AB} = \mathbf{BA} = \mathbf{I}_{n\times n}$. The inverse of $\mathbf{A}$ is denoted $\mathbf{A}^{-1}$.

- Inverse matrices only exist for square matrices.

Some other properties related to the inverse operator:

- If $n \times n$ matrices $\mathbf{A}$ and $\mathbf{B}$ are invertible then $(\mathbf{AB})^{-1} = \mathbf{B}^{-1}\mathbf{A}^{-1}$.
- If $\mathbf{A}$ is invertible then $(\mathbf{A}^{-1})^T = (\mathbf{A}^T)^{-1}$.

4.6 Matrix derivatives

We start with some basic calculus results.

Let $f(b)$ be a function of a scalar value b and $\frac{df(b)}{db}$ denote the derivative of the function with respect to b. Assume x is a fixed value. Then the following is true:

$f(b)$	$\frac{df(b)}{db}$
bx	x
b^2	$2b$
xb^2	$2bx$

Now lets look at the deriviate of a scalar function with respect to a vector.

Let $f(\mathbf{b})$ be a function of a $p\times 1$ column vector $= [b_1, b_2, \ldots, b_p]^T$. The derivative of $f(\mathbf{b})$ with respect to $\mathbf{b}$ is denoted $\frac{\partial f(\mathbf{b})}{\partial \mathbf{b}}$ and

$$\frac{\partial f(\mathbf{b})}{\partial \mathbf{b}} = \begin{bmatrix} \frac{\partial f(\mathbf{b})}{\partial b_1} \\ \frac{\partial f(\mathbf{b})}{\partial b_2} \\ \vdots \\ \frac{\partial f(\mathbf{b})}{\partial b_p} \end{bmatrix}.$$

Assume $\mathbf{X}$ is a fixed matrix. The following is true:

$f(\mathbf{b})$	$\frac{\partial f(\mathbf{b})}{\partial \mathbf{b}}$
$\mathbf{b}^T\mathbf{X}$	$\mathbf{X}$
$\mathbf{b}^T\mathbf{b}$	$2\mathbf{b}$
$\mathbf{b}^T\mathbf{X}\mathbf{b}$	$2\mathbf{X}\mathbf{b}$

Chapter 5

Defining a linear model

Based on Chapter 2 of LMWR2, Chapter 2 and 3 of ALR4

5.1 Background and terminology

Regression models are used to model the relationship between:

- one or more **response** variables and
- one or more **predictor** variables.

The distinction between these two types variables is their purpose in the model.

- Predictor variables are used to predict the value of the response variable.

Response variables are also known as **outcome**, **output**, or **dependent** variables.

Predictor variables are also known as **explanatory**, **regressor**, **input**, **dependent**, or **feature** variables.

Note: Because the variables in our model are often interrelated, describing these variables as independent or dependent variables is vague and is best avoided.

A distinction is sometimes made between **regression models** and **classification models**. In that case:

- Regression models attempt to predict a numerical response.
- Classification models attempt to predict the category level a response will have.

5.2 Goals of regression

The basic goals of a regression model are to:

1. *Predict* future or unknown response values based on specified values of the predictors.
    - What will the selling price of a home be?
2. *Identify relationships* (associations) between predictor variables and the response.
    - What is the general relationship between the selling price of a home and the number of bedrooms the home has?

With our regression model, we also hope to be able to:

1. *Generalize* our results from the sample to the a larger population of interest.
    - E.g., we want to extend our results from a small set of college students to all college students.
2. *Infer causality* between our predictors and the response.
    - E.g., if we give a person a vaccine, then this causes the person's risk of catching the disease to decrease.

A "true model" doesn't exist for real data. Thus, finding the true model should not be the goal of a regression analysis. A regression analysis should attempt to find a model that adequately describes the relationship between the response and relevant predictor variables (either in terms of prediction, association, generalization, causality, etc.)

5.3 Regression for Pearson's height data

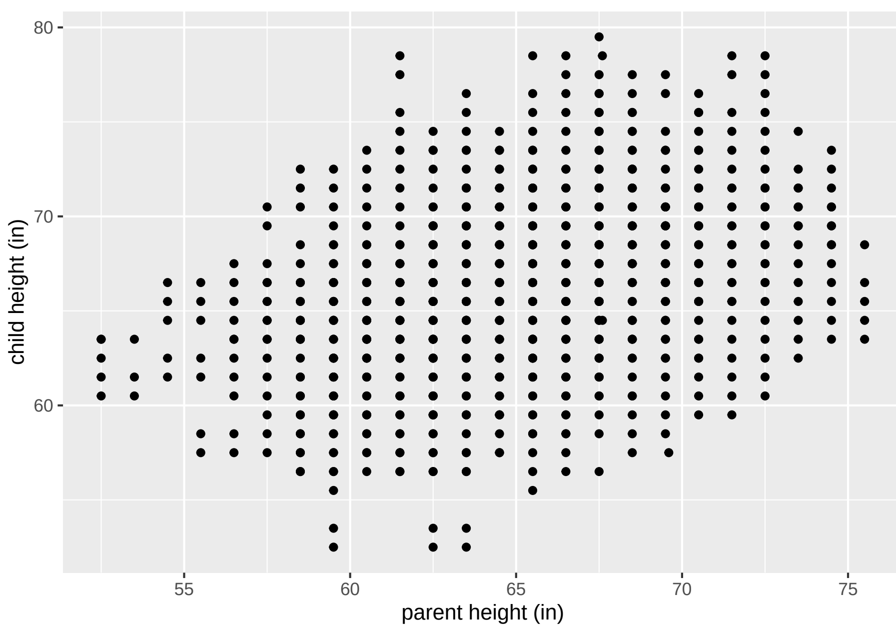
Wachsmuth, Wilkinson, and Dallal (2003) compiled child and parent height data from English familes tabulated by Pearson and Lee (1897) and Pearson and Lee (1903). The data are available in the `PearsonLee` data set in the **HistData** package (Friendly 2021). The `PearsonLee` data frame includes the variables:

- `child`: child height (inches).
- `parent`: parent height (inches).
- `gp`: a factor with levels `fd` (father/daughter), `fs` (father/son), `md` (mother/daughter), `ms` (mother/son) indicating the parent/child relationship.
- `par`: a factor with levels `Father`, `Mother` indicating the parent measured.
- `chl`: a factor with levels `Daughter`, `Son` indicating the child's relationship to the parent.

It is natural to wonder whether the height of a parent could explain the height of their child. We can consider a regression analysis that regresses child's height (the response variable) on parent's height (the predictor variable). The additional variables `gp`, `par`, and `chl` could also be used as predictor variables in our analysis. We perform an informal (linear) regression analysis visually using **ggplot2** (Wickham et al. 2021).

Consider a plot of child's height versus parent's height.

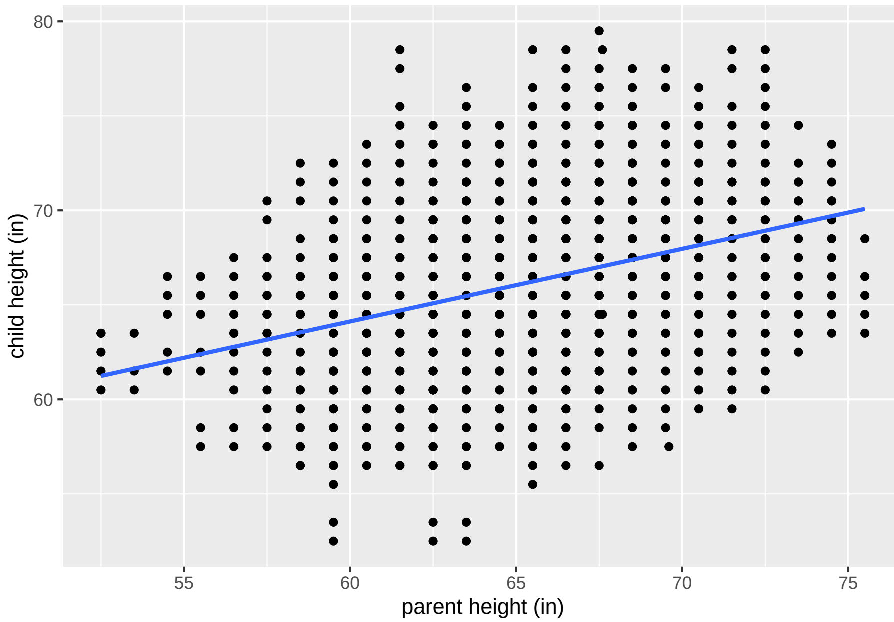
```
data(PearsonLee, package = "HistData") # load data
library(ggplot2) # load ggplot2 package
# create ggplot object for repeated use
# we'll be using common aesthetics across multiple geometries
# so we put them in the ggplot function
# also improve the x, y labels
ggheight <- ggplot(data = PearsonLee,
                   mapping = aes(x = parent, y = child)) +
             xlab("parent height (in)") + ylab("child height (in)")
ggheight + geom_point() # scatter plot of child vs parent height
```



We see a positive linear association between parent height and child height: as the height of the parent increases, the height of the child also tends to increase.

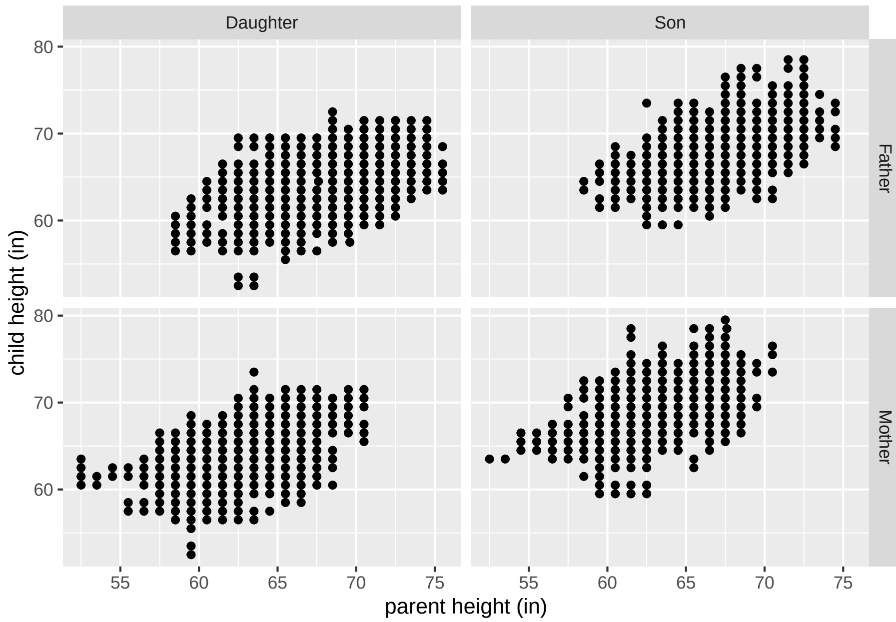
A simple linear regression model describes the relationship between a response and a predictor variable using the "best fitting" straight line (we'll formalize what best means later). We add the estimated simple linear regression model to our previous plot below using the `geom_smooth` function. The line fits reasonably well.

```
ggheight + geom_point() +
  geom_smooth(method = lm, formula = y ~ x, se = FALSE) # add estimated line
```



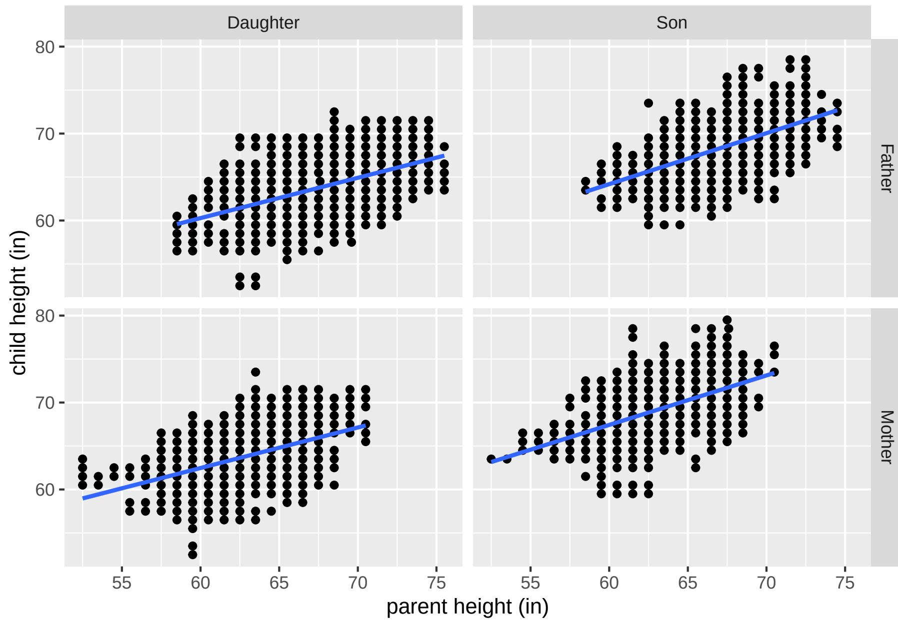
We may also wonder whether the type of parent (father/mother) or child (daughter/son) affects the relationship. We facet our scatter plots based on the `par` and `chl` variables below. While the overall patterns are similar, we notice that Father heights tend to be larger than Mother heights and Son heights tend to be larger than Daughter heights.

```
ggheight + geom_point() +
  facet_grid(par ~ chl) # facet the data by parent/child type
```



Having seen the previous graphic, we may wonder whether we can better model the relationship between parent and child height by accounting for which parent and child were measured. An interaction model assumes that the intercept and slope of each combination of parent/child is the different. We fit and plot an interaction model below.

```
ggheight + geom_point() + facet_grid(par ~ chl) +
       geom_smooth(method = lm, formula = y ~ x, se = FALSE) # add interaction  model to data
```



Other questions we could explore are whether the slopes across the different parent/child combinations are the same, whether the variability of the data is constant as parent height changes, predicting heights outside the range of the observed data, the precision of our estimated model, etc.

Regression analysis will generally be much more complex that was is presented above, but this example hopefully gives you an idea of the kinds of questions regression analysis can help you answer.

5.4 Definition of a linear model

A **linear model** is a regression model in which the regression coefficients (to be discussed later) enter the model linearly.

- A linear model is just a specific type of regression model.

5.4.1 Basic construction and relationships

We begin by defining notation for the objects we will need and clarifying some of their important properties.

- Y denotes the response variable.
  - The response variable is treated as a random variable.
  - We will observe realizations of this random variable for each observation in our data set.

- X denotes a single predictor variable. X_1, X_2, …, X_{p-1} will denote the predictor variables when there is more than one predictor variable.
  - The predictor variables are treated as non-random varables.
  - We will observe values of the predictors variables for each observation in our data set.
- β_0, β_1, …, β_{p-1} denote **regression coefficients**.
  - Regression coefficients are statistical parameters that we will estimate from our data.
  - Like all statistical parameters, regression coefficients are treated as fixed (non-random) but unknown values.
  - Regression coefficients are not observable.
- ϵ denotes **error**.
  - The error is not observable.
  - The error is treated as a random variable.
  - The error is assumed to have mean 0, i.e., $E(\epsilon) = 0$.
  - Since $E(\epsilon) = 0$ and X is non-random, the expectation of ϵ conditional on X is also 0, i.e., $E(\epsilon|X) = 0$.
  - In this context, error doesn't mean "mistake" or "malfunction." ϵ is simply the deviation of the response from its mean.

Then a **linear model** for Y is defined by the equation

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_{p-1} X_{p-1} + \epsilon. \tag{5.1}$$

We now emphasize the relationship between the response, the mean response, and the error. The mean of the response variable will depend on the values of the predictor variables. Thus, we can only discuss the expectation of the response variable conditional on the values of the predictor variables. This is denoted as $E(Y|X_1, \ldots, X_{p-1})$.

For simplicity, assume our linear model only has a single predictor (this is an example of simple linear regression). Based on what we've presented, we have that

$$\begin{aligned}
E(Y|X) &= E(\beta_0 + \beta_1 X + \epsilon|X) && (5.2)\\
&= E(\beta_0|X) + E(\beta_1 X|X) + E(\epsilon|X) && (5.3)\\
&= \beta_0 + \beta_1 X + 0 && (5.4)\\
&= \beta_0 + \beta_1 X. && (5.5)
\end{aligned}$$

The second line follows from the fact that the expectation of a sum of random variables is the sum of the expectation of the random variables. The third line follows from the fact that the expected value of a constant (non-random) value is the constant (the regression coefficients and X are non-random) and by our assumption that the errors have mean 0 (unconditionally or conditionally on the predictor variable.)

Thus, we see that we see that for a simple linear regression model

$$Y = E(Y|X) + \epsilon.$$

For a model with multiple predictors, this extends to

$$Y = E(Y|X_1, X_2, \ldots, X_{p-1}) + \epsilon.$$

Thus, our response may be written as the sum of the mean response conditional on the predictors, $E(Y|X_1, X_2, \ldots, X_{p-1})$, and the error. This is why previously we discussed the fact that the error is simply the deviation of the response from its mean.

Alternatively, one can say that a regression model is linear if the mean function can be written as a linear combination of the regression coefficients and known values, i.e.,

$$E(Y|X_1, X_2, \ldots, X_{p-1}) = \sum_{j=0}^{p-1} c_j \beta_j,$$

where $c_0, c_1, \ldots, c_{p-1}$ are known values. In fact, the $c_i, i = 1, 2, \ldots, n$ can be any function of $X_1, X_2, \ldots, X_n$! e.g., $c_1 = X_1 * X_2 * X_3$, $c_3 = X_2^2$, $c_8 = ln(X_1)/X_2^2$.

Some examples of linear models:

- $E(Y|X) = \beta_0 + \beta_1 X^2$.
- $E(Y|X_1, X_2) = \beta_0 + \beta_1 X_1 + \beta_2 X_2$.
- $E(Y|X_1, X_2) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1 * X_2$.
- $E(Y|X_1, X_2) = \beta_0 + \beta_1 ln(X_1) + \beta_2 X_2^{-1}$.

Some examples of non-linear models:

- $E(Y|X) = \beta_0 + e^{\beta_1} X$.
- $E(Y|X) = \beta_0 + \beta_1 X/(\beta_2 + X)$.

5.4.2 As a system of equations

A linear regression analysis will model the data using a linear model. Suppose we have sampled n observations from a population. We now introduce some additional notation:

- $Y_1, Y_2, \ldots, Y_n$ denote the response values for the n observations.
- $x_{i,j}$ denotes the observed value of predictor j for observation i.
  - We use lowercase x to indicate that this is the observed value of the predictor.
- $\epsilon_1, \epsilon_2, \ldots, \epsilon_n$ denote the errors for the n observations.

The linear model relating the responses, the predictors, and the errors is defined by the system of equations

$$Y_i = \beta_0 + \beta_1 x_{i,1} + \beta_2 x_{i,2} + \cdots + \beta_{p-1} x_{i,p-1} + \epsilon_i, \quad i = 1, 2, \ldots, n. \tag{5.6}$$

Based on our previous work, we can also write Equation (5.6) as

$$Y_i = E(Y_i|X_1 = x_{i,1}, \ldots, X_{p-1} = x_{i,p-1}) + \epsilon_i, \quad i = 1, 2, \ldots, n. \tag{5.7}$$

5.4.3 Using matrix notation

The regression coefficients are said to enter the model linearly, which is why this type of model is called a linear model. To see this more clearly, we represent the model using matrices. We define the following notation:

- $\mathbf{y} = [Y_1, Y_2, \ldots, Y_n]^T$ denotes the column vector containing the n responses.
- $\mathbf{X}$ denotes the matrix containing a column of 1s and the observed predictor values, specifically,

$$\mathbf{X} = \begin{bmatrix} 1 & x_{1,1} & x_{1,2} & \cdots & x_{1,p-1} \\ 1 & x_{2,1} & x_{2,2} & \cdots & x_{2,p-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_{n,1} & x_{n,2} & \cdots & x_{n,p-1} \end{bmatrix}.$$

- $\boldsymbol{\beta} = [\beta_0, \beta_1, \ldots, \beta_{p-1}]^T$ denotes the column vector containing the p regression coefficients.
- $\boldsymbol{\epsilon} = [\epsilon_1, \epsilon_2, \ldots, \epsilon_n]^T$ denotes the column vector contained the n errors. Then the system of equations defining the linear model in (5.6) can be written as

$$\mathbf{y} = \mathbf{X}\beta + \boldsymbol{\epsilon}.$$

  Thus, a linear model can be represented as a system of linear equations using matrices. A model that cannot be represented as a system of linear equations using matrices is not a linear model.

5.5 Summarizing the components of a linear model

We have already introduced a lot of objects. To aid in making sense of their notation, their purpose in the model, whether they can be observed, and whether they are modeled as a random variable (vector) or fixed, non-random values, we summarize things below.

We've already talked about observing the response variable and the predictor variables. So these objects are observable. However, we have no way to measure the regression coefficients or the error. These are not observable.

On the other hand, we treat the response variable as a random variable. Perhaps surprisingly, we treated the predictor variables as a fixed, non-random variables. The regression coefficients are treated as fixed, non-random but unknown values. This is standard for parameters in a statistical model. The errors are also treated as random variables. In fact, since both the predictor variables and the regression coefficients are non-random, the only way for the response to be a random variable based on Equation (5.6) is for the errors to be random.

We summarize this information in the table below for the objects previously discussed using the various notations introduced.

Notation	Description	Observable	Random
Y	response variable	Yes	Yes
Y_i	response value for the ith observation	Yes	Yes
$\mathbf{y}$	the $n \times 1$ column vector of response values	Yes	Yes
X	predictor variable	Yes	No
X_j	the jth predictor variable	Yes	No
$x_{i,j}$	the value of the jth predictor variable for the ith observation	Yes	No
$\mathbf{X}$	the $n \times p$ matrix of predictor values	Yes	No
β_j	the regression coefficient associated with the jth predictor variable	No	No
$\boldsymbol{\beta}$	the $p \times 1$ column vector of regression coefficients	No	No
ϵ	the error	No	Yes
ϵ_i	the error associated with observation i	No	Yes
$\boldsymbol{\epsilon}$	the $n \times 1$ column vector of errors	No	Yes

5.6 Types of regression models

The are many "named" types of regression models. You may hear or see people use these terms when describing their model. Here is a brief overview of some common regression models.

Name	Defining characteristics
Simple	an intercept term and one predictor variable
Multiple	more than one predictor variable
Multivariate	more than one response variable
Linear	the regression coefficients enter the model linearly
Analysis of variance (ANOVA)	predictors are all categorical
Analysis of covariance (ANCOVA)	at least one quantitative predictor and at least one categorical predictor
Generalized linear model (GLM)	a type of "generalized" regression model when the responses do not come from a normal distribution.

5.7 Standard linear model assumptions

The formulation of a linear model typically makes additional assumptions beyond the ones previously mentioned, specifically, about the errors, $\epsilon_1, \epsilon_2, \ldots, \epsilon_n$.

We have already mentioned that fact that we are assuming $E(\epsilon_i) = 0$ for $i = 1, 2, \ldots, n$.

We also typically assume that the errors have constant variances, i.e.,

$$var(\epsilon_i) = \sigma^2, \quad i = 1, 2, \ldots, n,$$

and that the errors are uncorrelated, i.e.,

$$cov(\epsilon_i, \epsilon_j) = 0, \quad i, j = 1, 2, \ldots, n, \quad i \neq j.$$

Additionally, we assume that the errors are identically distributed. Formally, that may be written as

$$\epsilon_i \sim F, i = 1, 2, \ldots, n, \tag{5.8}$$

where F is some arbitrary distribution. The $\sim$ means "distributed as." In other words, Equation (5.8) means, "ϵ_i is distributed as F for i equal to $1, 2, \ldots, n$." However, it is more common to assume the errors have a normal (Gaussian) distribution. Two uncorrelated normal random variables are also independent (this is true for normal random variables, but is not generally true for other distributions). Thus, we may concisely state the typical error assumptions as

$$\epsilon_1, \epsilon_2, \ldots, \epsilon_n \stackrel{i.i.d.}{\sim} N(0, \sigma^2),$$

which combines the following assumptions:

1. $E(\epsilon_i) = 0$ for $i = 1, 2, \ldots, n$.
2. $var(\epsilon_i) = 0$ for $i = 1, 2, \ldots, n$.
3. $cov(\epsilon_i, \epsilon_j) = 0$ for $i \neq j$ with $i, j = 1, 2, \ldots, n$.
4. ϵ_i has a normal distribution for $i = 1, 2, \ldots, n$.

5.8 Mathematical interpretation of coefficients

The regression coefficients have simple mathematical interpretations in basic settings.

5.8.1 Coefficient interpretation in simple linear regression

Suppose we have a simple linear regression model, so that $E(Y|X) = \beta_0 + \beta_1 X$. The interpretations of the coefficients are:

- β_0 is the expected response when the predictor is 0, i.e., $\beta_0 = E(Y|X = 0)$.
- β_1 is the expected change in the response when the predictor increases 1 unit, i.e., $\beta_1 = E(Y|X = x_0 + 1) - E(Y|X = x_0)$.

5.8.2 Coefficient interpretation in multiple linear regression

Suppose we have a multiple linear regression model, so that $E(Y|X_1, \ldots, X_{p-1}) = \beta_0 + \beta_1 X_1 + \cdots + \beta_{p-1} X_{p-1}$. Let $\mathcal{X} = \{X_1, \ldots, X_{p-1}\}$ be the set of predictors and $\mathcal{X}_{-j} = \mathcal{X} \setminus \{X_j\}$, i.e., the set of predictors without X_j.

The interpretations of the coefficients are:

- β_0 is the expected response when all predictors are 0, i.e., $\beta_0 = E(Y|X_1 = 0, \ldots, X_{p-1} = 0)$.
- β_j is the expected change in the response when predictor j increases 1 unit and the other predictors stay the same, i.e., $\beta_j = E(Y|\mathcal{X}_{-j} = \mathbf{x}^*, X_{j+1} = x_0 + 1) - E(Y|\mathcal{X}_{-j} = \mathbf{x}^*, X_{j+1} = x_0)$ where $\mathbf{x}^* \in \mathbb{R}^{p-2}$ is a fixed vector of length $p - 2$ (the number of predictors excluding X_j).

5.9 Exercises

1. If given a set of data with several variables, how would you decide what the response variable and the predictor variables would be?
2. Which objects in the linear model formula in Equation (5.1) are considered random? Which are considered fixed?
3. Which objects in the linear model formula in Equation (5.1) are observable? Which are not observable?
4. What are the typical goals of a regression analysis?
5. List the typical assumptions made for the errors in a linear model?
6. Without using a formula, what is the basic difference between a linear model and a non-linear model?
7. In the context of simple linear regression under the standard assumptions, show that $\beta_0 = E(Y|X = 0)$.
8. In the context of simple linear regression under the standard assumptions, show that $\beta_1 = E(Y|X = x_0 + 1) - E(Y|X = x_0)$.

9. In the context of multiple linear regression under the standard assumptions, show that $\beta_0 = E(Y|X_1 = 0, \ldots, X_{p-1} = 0)$.
10. In the context of multiple linear regression under the standard assumptions, show that $\beta_j = E(Y|\mathcal{X}_{-j} = \mathbf{x}^*, X_{j+1} = x_0 + 1) - E(Y|\mathcal{X}_{-j} = \mathbf{x}^*, X_{j+1} = x_0)$ where $\mathbf{x}^*$ is a fixed vector of the appropriate size.

Chapter 6

Fitting a linear model

A linear model for a response variable Y using $p-1$ predictor variables $X_1, \ldots, X_{p-1}$ may be described by the formula

$$Y = \beta_0 + \beta_1 X_1 + \cdots + \beta_{p-1} X_{p-1} + \epsilon,$$

where $\beta_0, \beta_1, \ldots, \beta_{p-1}$ are regression coefficients and ϵ is the error.

In this chapter we focus on estimating the regression coefficients.

Fitting a regression model is the same thing as estimating the parameters of a simple linear regression model.

There are many different methods of parameter estimation in statistics: method-of-moments, maximum likelihood, Bayesian, etc.

The most common estimation method for linear models is known as **Ordinary Least Squares (OLS)** estimation. OLS estimation estimates the parameters with the values that minimize the residuals sum of squares (RSS).

6.1 OLS estimation of the simple linear regression model

In a simple linear regression context, we have n observed responses $Y_1, Y_2, \ldots, Y_n$ and predictor values $x_1, x_2, \ldots, x_n$.

Recall that in a simple linear regression model

$$E(Y|X) = \beta_0 + \beta_1 X.$$

We need to define some new notation and objects to define the RSS.

Let β_j denote the estimated value of β_j and the estimated mean response as a function of the predictor X is

$$\hat{E}(Y|X) = \hat{\beta}_0 + \hat{\beta}_1 X.$$

The i**th fitted value** is defined as

$$\hat{Y}_i = \hat{E}(Y_i|X = x_i) = \hat{\beta}_0 + \hat{\beta}_1 x_i.$$

The i**th residual** is defined as

$$\hat{\epsilon}_i = Y_i - \hat{Y}_i.$$

The RSS of a regression model is the sum of the squared residuals.

The RSS for a simple linear regression model, as a function of the estimated regression coefficients, is

$$\begin{aligned}
RSS(\hat{\beta}_0, \hat{\beta}_1) &= \sum_{i=1}^{n} \hat{\epsilon}_i^2 && (6.1)\\
&= \sum_{i=1}^{n} (Y_i - \hat{Y}_i)^2 && (6.2)\\
&= \sum_{i=1}^{n} (Y_i - \hat{E}(Y|X = x_i))^2 && (6.3)\\
&= \sum_{i=1}^{n} (Y_i - (\hat{\beta}_0 + \hat{\beta}_1 x_i))^2. && (6.4)
\end{aligned}$$

The **fitted model** is the estimated model that minimizes the RSS. In a simple linear regression context, the fitted model is known as the **line of best fit**.

In Figure 6.1, we attempt to visualize the response values, fitted values, residuals, and line of best fit in a simple linear regression context. Notice that:

- The fitted values are the value returned by the line of best fit when it is evaluated at the observed predictor values. Alternatively, the fitted value for each observation is the y-value obtained when intersecting the line of best fit with a vertical line drawn from each observed predictor value.
- The residual is the vertical distance between each response value and the fitted value.
- The RSS seeks to minimize the sum of the squared vertical distances between the response and fitted values.

6.1.1 OLS estimators of the simple linear regression coefficents

Define $\bar{x} = \sum_{i=1}^{n} x_i$ and $\bar{Y} = \sum_{i=1}^{n} Y_i$.

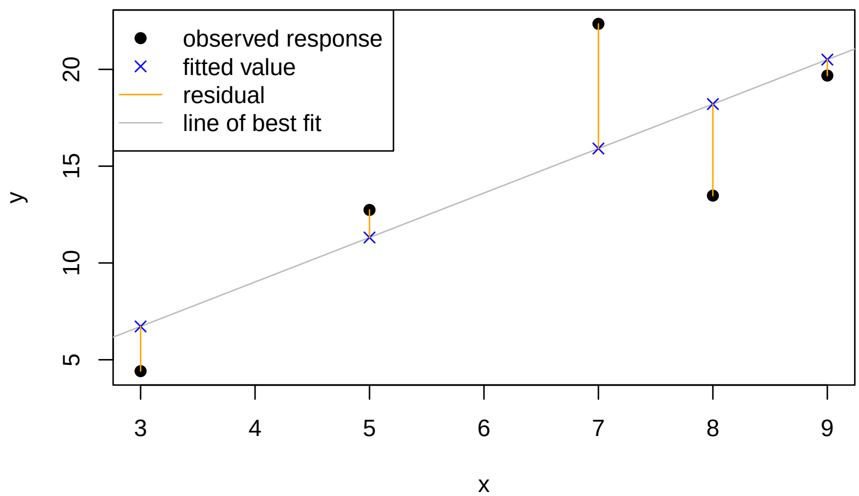


Figure 6.1: Visualization of the response values, fitted values, residuals, and line of best fit.

The OLS estimators of the regression coefficients for a simple linear regression coefficients are

$$\hat{\beta}_1 = \frac{\sum_{i=1}^n x_i Y_i - \frac{1}{n}\left(\sum_{i=1}^n x_i\right)\left(\sum_{i=1}^n Y_i\right)}{\sum_{i=1}^n x_i^2 - \frac{1}{n}\left(\sum_{i=1}^n x_i\right)^2} \tag{6.5}$$

$$= \frac{\sum_{i=1}^n (x_i - \bar{x})(Y_i - \bar{Y})}{\sum_{i=1}^n (x_i - \bar{x})^2} \tag{6.6}$$

$$= \frac{\sum_{i=1}^n (x_i - \bar{x})Y_i}{\sum_{i=1}^n (x_i - \bar{x})x_i} \tag{6.7}$$

and

$$\hat{\beta}_0 = \bar{Y} - \hat{\beta}_1 \bar{x}. \tag{6.8}$$

Thought it's already been said, we state once again that the OLS estimators of β_0 and β_1 shown above are the estimators that minimize the RSS.

6.2 Penguins simple linear regression example

We will use the `penguins` data set in the **palmerpenguins** package (Horst, Hill, and Gorman 2020) to illustrate a very basic simple linear regression analysis.

The `penguins` data set provides data related to various penguin species measured in the Palmer Archipelago (Antarctica), originally provided by Gorman, Williams, and Fraser (2014). We start by loading the data into memory.

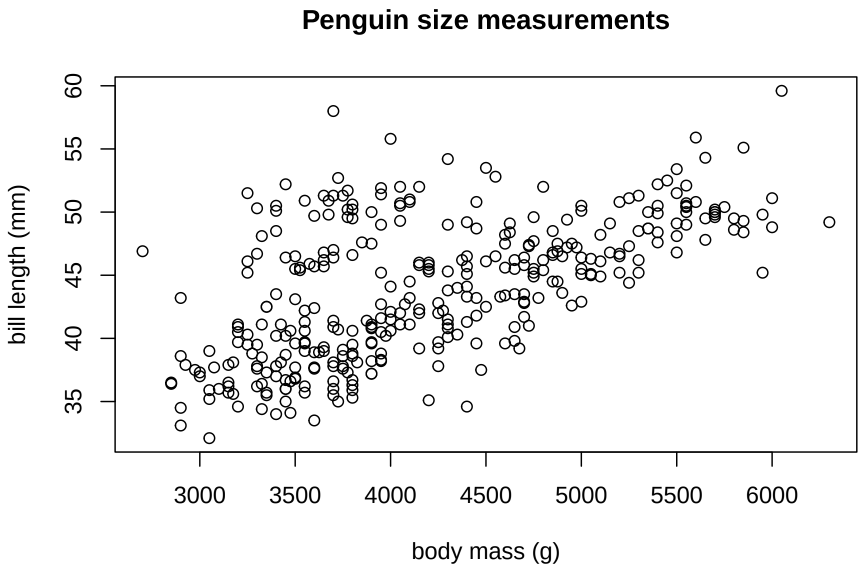
```
data(penguins, package = "palmerpenguins")
```

The data set includes 344 observations of 8 variables. The variables are:

- `species`: a `factor` indicating the penguin species
- `island`: a `factor` indicating the island the penguin was observed
- `bill_length_mm`: a `numeric` variable indicating the bill length in millimeters
- `bill_depth_mm`: a `numeric` variable indicating the bill depth in millimeters
- `flipper_length_mm`: an `integer` variable indicating the flipper length in millimeters
- `body_mass_g`: an `integer` variable indicating the body mass in grams
- `sex`: a `factor` indicating the penguin sex (`female`, `male`)
- `year`: an integer denoting the study year the penguin was observed (`2007`, `2008`, or `2009`)

We begin by creating a scatter plot of `bill_length_mm` versus `body_mass_g` (y-axis versus x-axis) in Figure **??**. We see a clear positive association between body mass and bill length: as the body mass increases, the bill length tends to increase. The pattern is linear, i.e., roughly a straight line.

```
plot(bill_length_mm ~ body_mass_g, data = penguins,
     ylab = "bill length (mm)", xlab = "body mass (g)",
     main = "Penguin size measurements")
```



We first perform a single linear regression analysis manually using the equations previously provided by regressing `bill_length_mm` on `body_mass_g`.

Using the `summary` function on the `penguins` data frame, we see that both `bill_length_mm` and `body_mass_g` have `NA` values.

```
summary(penguins)
#>       species          island    bill_length_mm
#>  Adelie   :152   Biscoe   :168   Min.   :32.10
#>  Chinstrap: 68   Dream    :124   1st Qu.:39.23
#>  Gentoo   :124   Torgersen: 52   Median :44.45
#>                                  Mean   :43.92
#>                                  3rd Qu.:48.50
#>                                  Max.   :59.60
#>                                  NA's   :2
#>  bill_depth_mm   flipper_length_mm  body_mass_g
#>  Min.   :13.10   Min.   :172.0     Min.   :2700
#>  1st Qu.:15.60   1st Qu.:190.0     1st Qu.:3550
#>  Median :17.30   Median :197.0     Median :4050
#>  Mean   :17.15   Mean   :200.9     Mean   :4202
#>  3rd Qu.:18.70   3rd Qu.:213.0     3rd Qu.:4750
#>  Max.   :21.50   Max.   :231.0     Max.   :6300
#>  NA's   :2       NA's   :2         NA's   :2
#>      sex           year
#>  female:165   Min.   :2007
#>  male  :168   1st Qu.:2007
```

```
#>  NA's   : 11   Median :2008
#>                Mean   :2008
#>                3rd Qu.:2009
#>                Max.   :2009
#>
```

We want to remove the rows of `penguins` where either `body_mass_g` or `bill_length_mm` have `NA` values. We do that below using the `na.omit` function (selecting only the relevant variables) and assign the cleaned object the name `penguins_clean`.

```
# remove rows of penguins where bill_length_mm or body_mass_g have NA values
penguins_clean <- na.omit(penguins[,c("bill_length_mm", "body_mass_g")])
```

We extract the `bill_length_mm` variable from the `penguins` data frame and assign it the name `y` since it will be the response variable. We extract the `body_mass_g` variable from the `penguins` data frame and assign it the name `y` since it will be the predictor variable. We also determine the number of observations and assign that value the name `n`.

```
# extract response and predictor from penguins_clean
y <- penguins_clean$bill_length_mm
x <- penguins_clean$body_mass_g
# determine number of observations
n <- length(y)
```

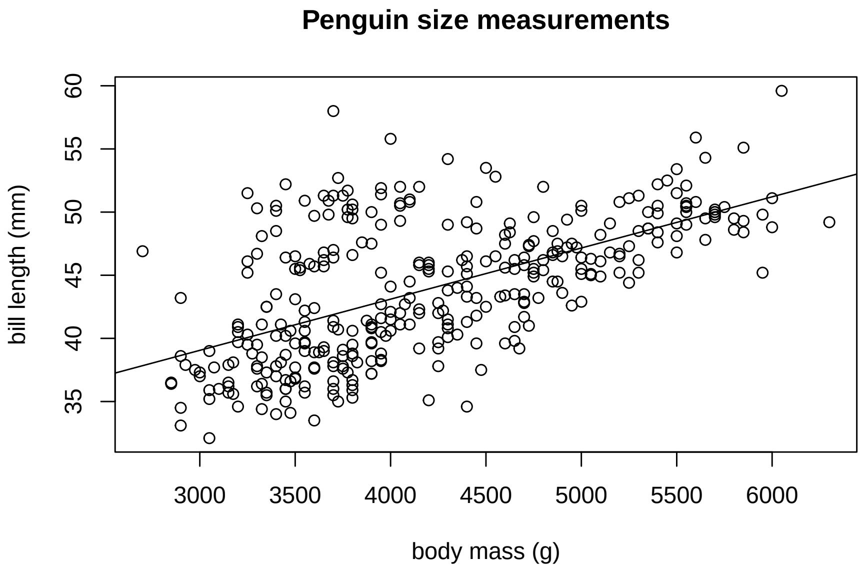
We now compute $\hat{\beta}$_1 and $\hat{beta}_0$. Note that placing `()` around the assignment operations will both perform the assign and print the results.

```
# compute OLS estimates of beta1 and beta0
(b1 <- (sum(x * y) - sum(x) * sum(y) / n)/(sum(x^2) - sum(x)^2/n))
#> [1] 0.004051417
(b0 <- mean(y) - b1 * mean(x))
#> [1] 26.89887
```

The estimated value of β_0 is $\hat{\beta}_1 - 22.02$ and the estimate value of β_1 is $\hat{\beta}_1 = -0.0011$.

We can use the `abline` function to overlay the fitted model on the observed data. Note that in simple linear regression, $\hat{\beta}_1$ corresponds to the slope of the fitted line and $\hat{\beta}_0$ will be the intercept.

```
plot(bill_length_mm ~ body_mass_g, data = penguins,
     ylab = "bill length (mm)", xlab = "body mass (g)",
     main = "Penguin size measurements")
# a is the intercept and b is the slope
abline(a = b0, b = b1)
```



The fit of the model to our data seems reasonable.

We can also compute the residuals, $\hat{\epsilon}_1, \ldots, \hat{\epsilon}_n$, the fitted values $\hat{y}_1, \ldots, \hat{y}_n$, and the associated RSS, $RSS = \sum_{i=1}^{n} \hat{\epsilon}_i^2$.

```
yhat <- b0 + b1 * x # compute fitted values
ehat <- y - yhat # compute residuals
(rss <- sum(ehat^2)) # sum of the squared residuals
#> [1] 6564.494
```

6.3 Fitting a linear model using R

We now describe how to use R to fit a linear model to data.

The `lm` function fits a linear model to data. The function has two major arguments:

- `data`: the data frame in which the model variables are stored. This can be omitted if the variables are already stored in memory.
- `formula`: a @(wilkinsonrogers1973) style formula describing the linear regression model. Assuming the `y` is the response, `x`, `x1`, `x2`, `x3` are available predictors:
  - `y ~ x` fits a simple linear regression model based on $E(Y|X) = \beta_0 + \beta_1 X$.
  - `y ~ x1 + x2` a the multiple linear regression model based on $E(Y|X_1, X_2) = \beta_0 + \beta_1 X_1 + \beta_2 X_2$.

–

An OLS fit of the simple linear regression model may be performed using the `lm` function. The `lm` function has many arguments, but at this stage, the only that are particularly relevant are the `formula` and `data` arguments:

- `data`: A data frame containing the relevant response and predictor variables.
- `formula`: A formula describing the model to fit. The notation is `y ~ x1 + x2 + ...`, where `y` is the response variable in `data`, `x1` is the first predictor to use in the model, `x2` is the second predictor, etc.

We fit a linear model regressing `body_mass_g` on `bill_length_mm` using the `penguins` data frame and store it in the object `lmod`. `lmod` is an object of class `lm`.

```
lmod <- lm(body_mass_g ~ bill_length_mm, data = penguins) # fit model
class(lmod) # class of lmod
#> [1] "lm"
```

```
(coeffs <- coefficients(lmod)) # assign and print coefficients
#>    (Intercept) bill_length_mm
#>      362.30672       87.41528
(ehat <- residuals(lmod)) # assign and print residuals
#>            1           2           3           5
#>   -30.244054  -15.210165 -635.142387 -120.447389
#>            6           7           8           9
#>  -147.727110 -137.760999  886.014418  131.832331
#>           10          11          12          13
#>   216.251642 -366.604194   33.395806 -755.074609
#>           14          15          16          17
#>    63.463584 1013.124692  138.294138 -295.277944
#>           18          19          20          21
#>   422.544004  -44.392252 -183.409466 -266.604194
#>           22          23          24          25
#>   -57.862667  299.484832  248.429695   45.980529
#>           26          27          28          29
#>   351.933998 -361.366970 -702.625442 -525.345722
#>           30          31          32          33
#>    47.374558 -565.210165  285.844972 -515.210165
#>           34          35          36          37
#>   -37.591553 -219.222806  361.014418  195.980529
#>           38          39          40          41
#>  -501.231413 -349.121139  808.565252 -402.964334
#>           42          43          44          45
#>   -28.850025 -409.256696  182.679560 -596.671973
#>           46          47          48          49
#>   776.048307 -530.074609 -665.379611  -59.256696
```

```
#>          50          51          52          53
#>   90.027059 -323.951693  432.340669   28.158581
#>          54          55          56          57
#>   16.251642 -478.133780 -281.299192 -221.502527
#>          58          59          60          61
#> -111.366970 -702.964334  100.878861 -333.032112
#>          62          63          64          65
#>  427.442336  -49.121139   94.925391 -694.222806
#>          66          67          68          69
#>  -48.782247 -115.549057  144.925391 -450.515168
#>          70          71          72          73
#>  433.734698  309.281497   67.306779 -273.951693
#>          74          75          76          77
#> -215.926411  234.450943  146.319420 -237.591553
#>          78          79          80          81
#>  285.844972   23.260249  -42.489886 -186.875308
#>          82          83          84          85
#>  587.577893  229.552611  769.417054 -272.896556
#>          86          87          88          89
#> -422.557664  264.518721  -87.930445  239.688167
#>          90          91          92          93
#> -162.760999   66.967888  344.925391   65.573859
#>          94          95          96          97
#>  626.048307 -226.739751  371.149975    7.171223
#>          98          99         100         101
#>  464.857613 -355.752392  -38.646690  303.158581
#>         102         103         104         105
#>  778.666919 -582.862667  583.395806 -750.345722
#>         106         107         108         109
#> -282.693221   13.463584  198.429695 -517.828777
#>         110         111         112         113
#>  636.353310  132.171223  251.556645 -632.693221
#>         114         115         116         117
#>  223.768587   76.048307  -19.939052 -836.536416
#>         118         119         120         121
#>  152.103444 -133.032112 -630.074609 -376.739751
#>         122         123         124         125
#> -157.862667 -426.400859 -106.299192 -389.324474
#>         126         127         128         129
#>   88.633030 -479.019471  309.959281 -721.502527
#>         130         131         132         133
#> -217.320440 -402.794888 -629.905163  -79.188917
#>         134         135         136         137
#>  834.620389 -267.828777  -55.074609 -299.290585
#>         138         139         140         141
```

```
#>    98.599141  -196.671973   417.306779  -476.400859
#>          142          143          144          145
#>  -436.366970  -118.337115  -195.108498  -622.896556
#>          146          147          148          149
#>  -121.502527   461.014418   -86.705862   -59.256696
#>          150          151          152          153
#>    83.395806   190.743304     9.959281   107.849006
#>          154          155          156          157
#>   966.929426  -169.430714   966.929426   876.726091
#>          158          159          160          161
#>   122.882895   469.039700   755.399840   252.611782
#>          162          163          164          165
#>   696.658312   712.408447   904.344703   310.298172
#>          166          167          168          169
#>  1256.793869  -165.926411  1178.120120   116.251642
#>          170          171          172          173
#>  1636.861647   399.107479   730.569286   949.446370
#>          174          175          176          177
#>   695.264283   -27.117105   640.365951   887.577893
#>          178          179          180          181
#>   707.849006  -152.286551  1109.243035    24.276924
#>          182          183          184          185
#>   816.929426   752.950674   596.319420   745.264283
#>          186          187          188          189
#>   477.742766   495.603175   806.793869   863.802476
#>          190          191          192          193
#>  1006.454977   141.421088   730.569286  -144.939052
#>          194          195          196          197
#>  1001.895537   -22.218772    51.895537   773.221787
#>          198          199          200          201
#>   726.387199  -139.701828   623.221787   812.747339
#>          202          203          204          205
#>   986.522756   414.141368   698.052341    95.264283
#>          206          207          208          209
#>   258.187898   472.882895   754.005811   108.904143
#>          210          211          212          213
#>   660.298172   311.353310   781.963315  -122.218772
#>          214          215          216          217
#>   899.107479    42.815117   541.043734   334.073589
#>          218          219          220          221
#>   984.412481   249.107479  1110.637064   535.128727
#>          222          223          224          225
#>   755.738732   217.984563   581.624423   524.276924
#>          226          227          228          229
#>   772.882895   281.624423  1189.310814    85.467618
```

```
#>          230          231          232          233
#>  1170.772621   436.522756  1636.522756   -29.396825
#>          234          235          236          237
#>   498.391233   219.209146   616.929426   462.747339
#>          238          239          240          241
#>   796.997204   443.870254   453.289566   360.467618
#>          242          243          244          245
#>   633.357344   435.467618   474.615816   410.298172
#>          246          247          248          249
#>   960.637064   597.713449   396.997204   244.378592
#>          250          251          252          253
#>   412.916785    31.793869   420.772621   248.052341
#>          254          255          256          257
#>   351.179291   486.692201   845.603175   227.950674
#>          258          259          260          261
#>  1046.658312   692.476225   469.717484   427.611782
#>          262          263          264          265
#>   933.018452   223.221787  1234.412481   485.128727
#>          266          267          268          269
#>   635.806510   -25.892521   671.111513   622.713449
#>          270          271          273          274
#>  1371.827758   436.692201   396.658312   981.963315
#>          275          276          277          278
#>   886.522756   675.670953  -927.117105  -833.070574
#>          279          280          281          282
#> -1196.710434  -805.960300 -1244.091822  -363.477244
#>          283          284          285          286
#> -1142.150994 -1096.710434  -233.409466 -1146.710434
#>          287          288          289          290
#>  -635.858632 -1106.676545  -770.824743  -857.901128
#>          291          292          293          294
#>  -799.667938  -726.778213 -1459.295157 -1732.392791
#>          295          296          297          298
#>  -968.375577  -263.138353  -468.714469 -1201.947659
#>          299          300          301          302
#> -1238.646690  -985.519741 -1144.600160  -757.901128
#>          303          304          305          306
#> -1376.778213  -889.362936  -718.375577  -427.833350
#>          307          308          309          310
#>  -737.591553  -800.214738  -727.455996  -720.485851
#>          311          312          313          314
#> -1106.845991  -614.532382  -673.273909  -107.901128
#>          315          316          317          318
#> -1762.083215  -539.024044  -695.655297  -750.892521
#>          319          320          321          322
```

```
#> -1261.744324  -839.701828 -1136.744324  -353.002796
#>          323          324          325          326
#> -1341.812102  -345.655297 -1614.193490 -1040.587519
#>          327          328          329          330
#> -1241.981548  -905.451962  -757.184883  -744.261268
#>          331          332          333          334
#>  -727.455996 -1475.384184 -1063.477244  -621.879880
#>          335          336          337          338
#>  -950.553630  -823.443355  -949.159601  -803.341688
#>          339          340          341          342
#>  -707.184883 -1240.079181  -764.871273  -923.104463
#>          343          344
#>  -703.002796  -975.553630
(yhat <- fitted(lmod)) # assign and print fitted values
#>        1        2        3        5        6        7
#> 3780.244 3815.210 3885.142 3570.447 3797.727 3762.761
#>        8        9       10       11       12       13
#> 3788.986 3343.168 4033.748 3666.604 3666.604 3955.075
#>       14       15       16       17       18       19
#> 3736.536 3386.875 3561.706 3745.278 4077.456 3369.392
#>       20       21       22       23       24       25
#> 4383.409 3666.604 3657.863 3500.515 3701.570 3754.019
#>       26       27       28       29       30       31
#> 3448.066 3911.367 3902.625 3675.346 3902.625 3815.210
#>       32       33       34       35       36       37
#> 3614.155 3815.210 3937.592 3544.223 3788.986 3754.019
#>       38       39       40       41       42       43
#> 4051.231 3649.121 3841.435 3552.964 3928.850 3509.257
#>       44       45       46       47       48       49
#> 4217.320 3596.672 3823.952 3955.075 3640.380 3509.257
#>       50       51       52       53       54       55
#> 4059.973 3823.952 3867.659 3421.841 4033.748 3378.134
#>       56       57       58       59       60       61
#> 3981.299 3771.503 3911.367 3552.964 3649.121 3483.032
#>       62       63       64       65       66       67
#> 3972.558 3649.121 3955.075 3544.223 3998.782 3465.549
#>       68       69       70       71       72       73
#> 3955.075 3500.515 4016.265 3290.719 3832.693 3823.952
#>       74       75       76       77       78       79
#> 4365.926 3465.549 4103.681 3937.592 3614.155 3526.740
#>       80       81       82       83       84       85
#> 4042.490 3386.875 4112.422 3570.447 3430.583 3622.897
#>       86       87       88       89       90       91
#> 3972.558 3535.481 3587.930 3710.312 3762.761 3483.032
#>       92       93       94       95       96       97
```

```
#> 3955.075 3334.426 3823.952 3526.740 3928.850 3692.829
#>       98       99      100      101      102      103
#> 3885.142 3255.752 4138.647 3421.841 3946.333 3657.863
#>      104      105      106      107      108      109
#> 3666.604 3675.346 3832.693 3736.536 3701.570 3692.829
#>      110      111      112      113      114      115
#> 4138.647 3692.829 4348.443 3832.693 4051.231 3823.952
#>      116      117      118      119      120      121
#> 4094.939 3736.536 3622.897 3483.032 3955.075 3526.740
#>      122      123      124      125      126      127
#> 3657.863 3876.401 3981.299 3439.324 3911.367 3754.019
#>      128      129      130      131      132      133
#> 3990.041 3771.503 4217.320 3727.795 4129.905 3579.189
#>      134      135      136      137      138      139
#> 3640.380 3692.829 3955.075 3474.291 3876.401 3596.672
#>      140      141      142      143      144      145
#> 3832.693 3876.401 3911.367 3168.337 3920.108 3622.897
#>      146      147      148      149      150      151
#> 3771.503 3788.986 3561.706 3509.257 3666.604 3509.257
#>      152      153      154      155      156      157
#> 3990.041 4392.151 4733.071 4619.431 4733.071 4523.274
#>      158      159      160      161      162      163
#> 4427.117 4330.960 4444.600 4147.388 4453.342 3937.592
#>      164      165      166      167      168      169
#> 4645.655 4339.702 4593.206 4365.926 4671.880 4033.748
#>      170      171      172      173      174      175
#> 4663.138 4400.893 4619.431 4750.554 4304.736 4427.117
#>      176      177      178      179      180      181
#> 4409.634 4112.422 4392.151 4252.287 4540.757 4575.723
#>      182      183      184      185      186      187
#> 4733.071 4497.049 4103.681 4304.736 5572.257 4654.397
#>      188      189      190      191      192      193
#> 4593.206 4086.198 4243.545 4208.579 4619.431 4094.939
#>      194      195      196      197      198      199
#> 4698.104 4322.219 4698.104 4776.778 4173.613 4339.702
#>      200      201      202      203      204      205
#> 4776.778 4287.253 4313.477 4435.859 4601.948 4304.736
#>      206      207      208      209      210      211
#> 4741.812 4427.117 4295.994 4191.096 4339.702 4138.647
#>      212      213      214      215      216      217
#> 4768.037 4322.219 4400.893 4357.185 5108.956 4365.926
#>      218      219      220      221      222      223
#> 4715.588 4400.893 4689.363 4164.871 4794.261 4532.015
#>      224      225      226      227      228      229
#> 4418.376 4575.723 4427.117 4418.376 4610.689 4514.532
```

```
#>      230      231      232      233      234      235
#> 4829.227 4313.477 4313.477 4654.397 4951.609 4505.791
#>      236      237      238      239      240      241
#> 4733.071 4287.253 4803.003 4156.130 4846.710 4514.532
#>      242      243      244      245      246      247
#> 4916.643 4514.532 4925.384 4339.702 4689.363 4252.287
#>      248      249      250      251      252      253
#> 4803.003 4680.621 4462.083 4593.206 4829.227 4601.948
#>      254      255      256      257      258      259
#> 5248.821 4488.308 4654.397 4497.049 4453.342 4007.524
#>      260      261      262      263      264      265
#> 5030.283 4147.388 4566.982 4776.778 4715.588 4164.871
#>      266      267      268      269      270      271
#> 4864.193 4400.893 5178.888 4252.287 4628.172 4488.308
#>      273      274      275      276      277      278
#> 4453.342 4768.037 4313.477 4724.329 4427.117 4733.071
#>      279      280      281      282      283      284
#> 4846.710 4330.960 4969.092 4313.477 4392.151 4846.710
#>      285      286      287      288      289      290
#> 4383.409 4846.710 4435.859 4881.677 4470.825 4907.901
#>      291      292      293      294      295      296
#> 4374.668 4776.778 4759.295 5432.393 4418.376 4663.138
#>      297      298      299      300      301      302
#> 4068.714 4601.948 4138.647 4785.520 4444.600 4907.901
#>      303      304      305      306      307      308
#> 4776.778 4689.363 4418.376 4977.833 3937.592 5100.215
#>      309      310      311      312      313      314
#> 4077.456 4820.486 4706.846 4514.532 4523.274 4907.901
#>      315      316      317      318      319      320
#> 4462.083 5039.024 4645.655 4400.893 4811.744 4339.702
#>      321      322      323      324      325      326
#> 4811.744 4803.003 4741.812 4645.655 4864.193 4715.588
#>      327      328      329      330      331      332
#> 4566.982 4855.452 4357.185 4794.261 4077.456 4925.384
#>      333      334      335      336      337      338
#> 4313.477 4671.880 4750.554 4348.443 4899.160 4453.342
#>      339      340      341      342      343      344
#> 4357.185 5240.079 4164.871 4698.104 4803.003 4750.554
methods(class="lm")
#>  [1] add1           alias          anova
#>  [4] case.names     coerce         confint
#>  [7] cooks.distance deviance       dfbeta
#> [10] dfbetas        drop1          dummy.coef
#> [13] effects        extractAIC     family
#> [16] formula        fortify        hatvalues
```

```
#> [19] influence      initialize     kappa
#> [22] labels         logLik         model.frame
#> [25] model.matrix   nobs           plot
#> [28] predict        print          proj
#> [31] qqnorm         qr             residuals
#> [34] rstandard      rstudent       show
#> [37] simulate       slotsFromS3    summary
#> [40] variable.names vcov
#> see '?methods' for accessing help and source code
```

6.3.1 Derivation of OLS simple linear regression estimators

Use calculus to derive the OLS estimator of the regression coefficients. Take the partial derivatives of $RSS(\hat{\beta}_0, \hat{\beta}_1)$ with respect to $\hat{\beta}_0$ and $\hat{\beta}_1$, set the derivatives equal to zero, and solve for $\hat{\beta}_0$ and $\hat{\beta}_1$ to find the critical points of $RSS(\hat{\beta}_0, \hat{\beta}_1)$. Technically, you must show that the Hessian matrix of $RSS(\hat{\beta}_0, \hat{\beta}_1)$ is positive definite to verify that our solution minimizes the RSS, but we won't do that here.

6.3.2 Expected value of OLS simple linear regression estimators

Determine the expected value of the OLS simple linear regression estimators.

Chapter 7

Interpreting a fitted linear model

Chapter 8

Categorical predictors

Categorical predictors can greatly improve the explanatory power or predictive capability of a fitted model when different patterns exist for different levels of the categorical variables. In what follows, we consider several common linear regression models that involve a categorical variable. To simplify our discussion, we only consider the setting where there is a single categorical variable to add to our model. Similarly, we only consider the setting where there is a single numeric variable.

We begin by defining some notation.

Let X denote a numeric regressor, with x_i denoting the value of X for observation i.

Let F denote a categorical variable with levels $L_1, L_2, \ldots, L_K$. The F stands for "factor," while the L stands for "level." The notation f_i denotes the value of F for observation i.

8.1 Indicator/dummy variables

We may recall that if $\mathbf{X}$ denotes our matrix of regressors and $\mathbf{y}$ our vector of responses, then (assuming the columns of $\mathbf{X}$ are linearly independent) the OLS solution for $\boldsymbol{\beta}$ is

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}.$$

In order to compute the estimated coefficients, both $\mathbf{X}$ and $\mathbf{y}$ must contain numeric values. How can we use a categorical predictor in our regression model when the levels are not numeric values? In order to use a categorical predictor in a regression model, we must transform it into a set of one or more **indicator** or **dummy variables**, which we explain in more detail below.

An **indicator function** is a function that takes the value 1 of a certain property is true and 0 otherwise. An indicator variable is the variable that results from applying an indicator function to each observation in a data set. Many notations exist for indicator functions. We will adopt the notation

$$I_S(x) = \begin{cases} 1 & \text{if } x \in S \\ 0 & \text{if } x \notin S \end{cases},$$

which is shorthand for a function that returns 1 if x is in the set S and 0 otherwise.

We let D_j denote the indicator (dummy) variable for factor level L_j of F. The value of D_j for observation i is denoted $d_{i,j}$, with

$$d_{i,j} = I_{\{L_j\}}(f_i),$$

i.e., $d_{i,j}$ is 1 if observation i has factor level L_j and 0 otherwise.

8.2 Common of linear models with categorical predictors

It is common to use notation like $E(Y|X_1, X_2) = \beta_0 + \beta_1 X_1 + \beta_2 X_2$ when discussing linear regression models. That notation is generally simple and convenient, but can be unclear. Asking a researcher what the estimate of β_2 is in a model is ambiguous because it will depend on the order the researcher added the variables to the model. To more closely connect each coefficient with the regressor to which it is related, we will use the notation β_X to denote the coefficient for regressor X and β_{D_j} to denote the coefficient for regressor D_j. Similarly, β_{int} denotes the intercept included in our model.

8.2.1 One-way ANOVA

8.2.1.1 Definition

A one-way analysis of variance (ANOVA) assumes a constant mean for each level of a categorical variable. A general one-way ANOVA relating a response variable Y to a factor variable F with K levels may be formulated as

$$E(Y|F) = \beta_{int} + \beta_{D_2} D_2 + \ldots + \beta_{D_K} D_K.$$

Alternatively, in terms of the individual responses, we may formulate the one-way ANOVA model as

$$Y_i = \beta_{int} + \beta_{D_2} d_{i,2} + \cdots + \beta_{D_K} d_{i,K} + \epsilon_i, \quad i = 1, 2, \ldots n.$$

This may bring up some questions that need answering.

Why does the one-way ANOVA model only contains dummy variables for the last $K-1$ levels of F? This is not a mistake. If we included dummy variables for all levels of F, then the matrix of regressors would have linearly dependent columns because the sum of the dummy variables for all K levels would equal the column of 1s for the intercept.

Why do we omit the dummy variable for the first level of F? This is simply convention. We could omit the dummy variable for any single level of F. However, it is conventional to designate one level the reference level and to omit that variable. As we will see when discussing interpretation of the coefficient, the reference level becomes the level of F that all other levels are compared to.

Could we omit the intercept instead of one of the dummy variables? Yes, you could. There is no mathematical or philosophical issues with this. However, this can create problems when you construct models including both categorical and numeric regressors. The standard approach is recommended because it typically makes our model easier to interpret and extend.

8.2.1.2 Interpretation

We interpret the coefficients of our one-way ANOVA with respect to the change in the mean response.

Suppose an observation of level L_1. We can determine that the mean response is

$$\begin{aligned} E(Y|F=L_1) &= \beta_{int} + \beta_{D_2}0 + \cdots + \beta_{D_K}0 \\ &= \beta_{int}. \end{aligned}$$

Similarly, when an observation has level L_2, then

$$\begin{aligned} E(Y|F=L_2) &= \beta_{int} + \beta_{D_2}1 + \beta_{D_3}0 + \cdots + \beta_{D_K}0 \\ &= \beta_{int} + \beta_{D_2}. \end{aligned}$$

This helps us to see the general relationship that

$$E(Y|F=L_j) = \beta_{int} + \beta_{D_j}, \quad j = 2, \ldots, K.$$

In the context of a one-way ANOVA:

- β_{int} represents the expected response for observations having the reference level.
- β_{L_j}, for $j = 2, \ldots, K$, represents the expected change in the response when comparing observations having the reference level and level L_j.
  - You can verify this by computing $E(Y|F=L_j) - E(Y|F=L_1)$ (for $j = 2, \ldots, K$).

8.2.2 Main effects models

A main effects model is also called a parallel lines model since the regression equations for each factor level produce lines parallel to one another.

A parallel lines model is formulated as

$$Y_i = \beta_{int} + \beta_X x_i + \beta_{L_2} d_{i,2} + \cdots + \beta_{L_K} d_{i,K} + \epsilon_i, \quad i = 1, 2, \ldots n.$$

When an observation has level L_1, then the expected response is $\beta_{int} + \beta_1 X$. More specifically,

$$E(Y|X = x, F = L_1) = \beta_{int} + \beta_X x + \beta_{L_2} 0 + \cdots + \beta_{L_K} 0 = \beta_{int} + \beta_X x.$$

Thus, $E(Y|X = 0, F = L_1) = \beta_{int}$.

When an observation has level L_2, the expected response is $\beta_{int} + \beta_X X + \beta_{L_2}$. More formally,

$$E(Y|X = x, F = L_2) = \beta_{int} + \beta_X x + \beta_{L_2} 1 + \beta_{L_3} 0 + \cdots + \beta_{L_K} 0 = \beta_{int} + \beta_X x + \beta_{L_2}.$$

Thus, the mean response for observations having level L_2 is $\beta_{int} + \beta_{L_2} + \beta_X x$. In general,

$$E(Y|X = x, F = L_j) = \beta_{int} + \beta_X x + \beta_{L_j}, \quad j = 2, \ldots, K.$$

Thus,

$$E(Y|X = x, F = L_j) - E(Y|X = x, F = L_1) = (\beta_{int} + \beta_X x + \beta_{L_j}) - (\beta_{int} + \beta_X x) = \beta_{L_j}.$$

In the context of parallel lines models:

- β_{int} represents the expected response for observations having the reference level when the numeric regressor $X = 0$.
- β_X is the expected change in the response when X increases by 1 unit for a fixed level of F.
- β_{L_j}, for $j = 2, \ldots, K$ represents the expected change in the response when comparing observations having level L_1 and L_j with X fixed at the same value.

8.2.3 Interaction models

An interaction model is also called a separate lines model since the regression equations for each factor level produce lines that are distinct and separate.

A separate lines model is formulated as

$$Y_i = \beta_{int} + \beta_X x_i + \beta_{L_2} d_{i,2} + \cdots + \beta_{L_K} d_{i,K} + + \beta_{X L_2} x_i d_{i,2} + \cdots + \beta_{X L_j} x_i d_{i,K} + \epsilon_i, \quad i = 1, 2, \ldots n.$$

When an observation has level L_1, then the expected response is $\beta_{int} + \beta_1 X$.
More specifically,

$$E(Y|X = x, F = L_1) = \beta_{int}+\beta_X x+\beta_{L_2}0+\cdots+\beta_{L_K}0+\beta_{XL_2}x_i0+\cdots+\beta_{XL_K}x_i0 = \beta_{int}+\beta_X x.$$

Thus, $E(Y|X = 0, F = L_1) = \beta_{int}$.

When an observation has level L_j, for $j = 2, \ldots, K$, the expected response is $\beta_{int} + \beta_X X + \beta_{L_j} + \beta_{XL_J} X$. More formally,

$$E(Y|X = x, F = L_j) = \beta_{int} + \beta_X x + \beta_{L_j} + \beta_{XL_j} x.$$

Note that

$$E(Y|X = 0, F = L_1) = \beta_{int}.$$

Additionally, we note that

$$E(Y|X = 0, F = L_j)-E(Y|X = 0, F = L_1) = (\beta_{int}+\beta_X 0+\beta_{L_j}+\beta_{XL_J}0)-(\beta_{int}+\beta_X 0) = \beta_{L_j}.$$

In the context of separate lines models:

- β_{int} represents the expected response for observations having the reference level when the numeric regressor $X = 0$.
- β_{L_j}, for $j = 2, \ldots, K$, represents the expected change in the response when comparing observations having level L_1 and L_j with $X = 0$.
- β_X represents the expected change in the response when X increases by 1 unit for observations having the reference level.
- $\beta_X L_j$, for $j = 2, \ldots, K$, represents the difference in the expected rate of change when X increases by 1 unit for observations have the baseline level in comparison to level L_j.

8.2.4 Extensions

In the models above, we have only discussed possibilities with a single numeric variable and a single factor variable. Naturally, one can consider models with multiple factor variables, multiple numeric variables, interactions between factor variables, interactions between numeric variables, etc. The models become more complicated, but the ideas are similar. One simply has to keep track of what role each coefficient plays in the model.

Chapter 9

Assessing and addressing collinearity

Friendly, Michael. 2021. *HistData: Data Sets from the History of Statistics and Data Visualization.* https://CRAN.R-project.org/package=HistData.

Gorman, Kristen B., Tony D. Williams, and William R. Fraser. 2014. "Ecological Sexual Dimorphism and Environmental Variability Within a Community of Antarctic Penguins (Genus Pygoscelis)." *PLOS ONE* 9 (3): 1–14. https://doi.org/10.1371/journal.pone.0090081.

Horst, Allison, Alison Hill, and Kristen Gorman. 2020. *Palmerpenguins: Palmer Archipelago (Antarctica) Penguin Data.* https://CRAN.R-project.org/package=palmerpenguins.

Pearson, Karl, and Alice Lee. 1897. "Mathematical Contributions to the Theory of Evolution. On Telegony in Man." *Proceedings of the Royal Society of London* 60 (359-367): 273–83.

———. 1903. "On the Laws of Inheritance in Man: I. Inheritance of Physical Characters." *Biometrika* 2 (4): 357–462.

Wachsmuth, Amanda, Leland Wilkinson, and Gerard E Dallal. 2003. "Galton's Bend." *The American Statistician* 57 (3): 190–92. https://doi.org/10.1198/0003130031874.

Wickham, Hadley, Winston Chang, Lionel Henry, Thomas Lin Pedersen, Kohske Takahashi, Claus Wilke, Kara Woo, Hiroaki Yutani, and Dewey Dunnington. 2021. *Ggplot2: Create Elegant Data Visualisations Using the Grammar of Graphics.* https://CRAN.R-project.org/package=ggplot2.

Xie, Yihui. 2021. *Bookdown: Authoring Books and Technical Documents with r Markdown.* https://CRAN.R-project.org/package=bookdown.