Data manipulation

Contents

Introduction

This is the first programming practical. If you haven't yet done so, open the project file `02_Data_manipulation.Rproj` in RStudio. You can choose to write the answers to your exercises in either an `.R` file or in an `.Rmd` file. Example answer files are provided in the project directory (`example_answers.Rmd` and `example_answers.R`). You can open these from the files pane and use them as a starting point. While working through the exercises, write down your code in one of these files. Use proper style and provide comments so you can read it back later and still understand what is happening.

The practicals always start with the packages we are going to use. Be sure to run these lines in your session to load their functions before you continue.

```
library(ISLR)
library(tidyverse)
library(haven)
```

Data types

There are several data types in `R`. Here is a table with the most common ones:

Type	Short	Example
Integer	int	0, 1, 2, 3, -4, -5
Numeric / Double	dbl	0.1, -2.5, 123.456
Character	chr	"dav is a cool course"
Logical	lgl	TRUE / FALSE
Factor	fct	low, medium, high

The `class()` function can give you an idea about what type of data each variable contains.

---

1. **Run the following code in R and inspect their data types using the `class()` function. Try to guess beforehand what their types will be!**

```
object_1 <- 1:5
object_2 <- 1L:5L
object_3 <- "-123.456"
object_4 <- as.numeric(object_2)
object_5 <- letters[object_1]
object_6 <- as.factor(rep(object_5, 2))
object_7 <- c(1, 2, 3, "4", "5", "6")
```

the factor data type is special to R and uncommon in other programming languages. It is used to represent categorical variables with fixed possible values. For example, when there is a multiple choice question with 5 possible choices (a to e) and 10 students answer the question, we may get a result as in `object_6`.

Vectors can have only a single data type. Note that the first three elements in `object_7` have been converted. We can convert to different data types using the `as.<class>()` functions.

---

2. **Convert `object_7` back to a vector of numbers using the `as.numeric()` function**

Lists

A list is a collection of objects. The elements may have names, but it is not necessary. Each element of a list can have a different data type, unlike vectors.

---

3. **Make a list called `objects` containing object 1 to 7 using the `list()` function.**

A special type of list is the `data.frame`. It is the same as a list, but each element is forced to have the same length. The elements of a `data.frame` are the columns of a dataset. In the tidyverse, `data.frames` are called `tibbles`.

---

4. **Make a data frame out of `object_1`, `object_2`, and `object_5` using the `data.frame()` function**

Loading data

We are going to use a dataset from Kaggle - the Google play store apps data by user `lava18`. We have downloaded it into the data folder already from https://www.kaggle.com/lava18/google-play-store-apps (downloaded on 2018-09-28).

Tidyverse contains many data loading functions – each for their own file type – in the packages `readr` (default file types) and `haven` (external file types such as from SPSS or Stata). The most common file type is `csv`, which is what we use here.

---

1. **Use the function `read_csv()` to import the file "data/googleplaystore.csv" and store it in a variable called `apps`.**

If necessary, use the help files. These import functions from the tidyverse are fast and safe: they display informative errors if anything goes wrong. `read_csv()` also displays a message with information on how each column is imported: which variable type each column gets.

---

2. **Did any column get a variable type you did not expect?**

---

3. **Use the function `head()` to look at the first few rows of the `apps` dataset**
