R Worskhop Notes

2018-04-04

These are brief notes on data import, tidying and cleaning. For much more detail and explanation see Hadley Wickham's book: `R for Data Science` (R4DS) at (`http://r4ds.had.co.nz/wrangle-intro.html`).

Load the tidyverse

Before importing or tidying data run...

```
library(tidyverse)
```

...to load the `tidyverse` set of packages first.

Importing data

From Excel files

Uses use the `read_excel()` function from the `readxl` package. It's installed with the `tidyverse` but you need to load it before using it with...

```
library(readxl)
```

Then to read a sheet called "my-sheet-name-or-number" from an Excel workbook called "my-excel-file.xls", skipping the first 5 rows, into an R object called mydata use...

```
mydata <- read_excel("my-excel-file.xls", sheet = "my-sheet-name-or-number",
    skip = 5)
```

From comma-separated value files (csv)

Use the `read_csv()` function from the `readr` package. This is loaded automatically when the `tidyverse` is loaded.

To read a csv file called "my-csv-file.csv" which includes a header row into a new R object called mydata use...

```
mydat <- read_csv("my-csv-file.csv")
```

Tidying data

Filling down blanks

To fill down any blank cells in the `patient` and `breed` columns of `mydata` use...

```
mydata <- mydata %>% fill(patient, breed)
```

Selecting or removing columns

The `select` verb lets you choose or remove columns. To remove the name column from mydata use...

```
mydata <- mydata %>% select(-name)
```

To just select `subject` and all columns between `age` and `glucose` in mydata use...

```
mydata <- mydata %>% select(subject, age:glucose)
```

Renaming columns

To `rename` the column `sex (m/f)` to `sex` use...

```
mydata <- mydata %>% rename(sex = ‘sex (m/f)‘)
```

Note that the back ticks “ let you choose columns that have names that would not normally be allowed in R.

Separating a columns into 2 (or more) new columns

A column that contains data that should be in two separate columns can be split using `separate()`. To split a column called `sex_status` into two columns, `sex` and `status` using _ as a separator use...

```
mydata <- mydata %>% separate(sex_status, into = c("sex",
    "status"), sep = "_")
```

This would split “male_neutered” in one column into “male” and “neutered” in two separate columns etc.

Creating and changing column content

To make a new column use `mutate()`. For example if you have a column `weight` you can add a column log_wt containing `log(weight)` like this...

```
mydata <- mydata %>% mutate(log_wt = log(weight))
```

Mutate can also overwrite columns (but be careful). So to convert height in inches to height in cm in the same column you could use...

```
mydata <- mydata %>% mutate(height = height *
    2.54)
```

Gathering data in several columns

Sometimes data at different times are stored in one column per time. To convert theses into a column for measurement time and a column for the actual data use the `gather` verb. For example if there were three columns `w_1`, `w_2` and `w_3` all holding weight data at each week these could be converted to a column for week and a column for weight. . .

```
mydata <- mydata %>% gather(key = "week", value = "weight",
    w_1, w_2, w_3)
```

This can be tricky at first so for more detail see R4DS (`http://r4ds.had.co.nz/tidy-data.html#gathering`)

Filtering observations

To remove (or include) observations that meet a certain criteria use filter. For example to select only the observations from mydata where colour is "blue" use. . .

```
mydata <- mydata %>% filter(colour == "blue")
```

Note that to check if a column equals 'something' you use two equals `==`. You can compare numbers using `>` and `<`. To check if a column has values in a list use `%in%` like this. . .

```
mydata <- mydata %>% filter(colour %in% c("blue",
    "red", "green"))
```

A `!` in front of the test makes it opposite. The following keeps all the observations where the colour **isn't** 'blue'.

```
mydata <- mydata %>% filter(!colour == "blue")
```

Missing values in R are coded as `NA`. There's a special function, `is.na()` to test for missing data. So the following keeps all rows where the colour **isn't** missing

```
mydata <- mydata %>% filter(!is.na(colour))
```