Lecture 4

2023-08-03

Dataframes

These are like matrices.

```
offers <- data.frame(amount = c(241, 590, 533),
                     spec = c("family", "cardio", "ortho"))
View(offers)
```

Example

We can load an external package and view a larger data frame.

```
library(babynames)
# View(babynames) # file too large, don't actually load it
View(babynames[2, ]) # View the second row
View(babynames[2:6, c("name", "year")])
# Views the second to sixth row, filtered only for name and year
```

There are 2 ways of viewing a column, `babynames$prop` and `babynames[, "prop"]`. The former is a vector, and the latter is a dataframe with one column (think 2D). You have to access entries in the column differently.

```
babynames$prop[1] # getting the nth entry of a vector is simple
```

```
## [1] 0.07238359
```

```
babynames[, "prop"][1, "prop"]
```

```
## # A tibble: 1 x 1
##     prop
##    <dbl>
## 1 0.0724
```

```
# first part is still a dataframe with 2 dimensions, so we need 2 coordinates

# Vectors are different from dataframes!
```

Note that `babynames[2, ]` can be thought of as $\text{babynames}_{2,\cdot}$ in Linear Algebra Done Right.

Searching for desired data

What is the most popular name in `year`?

```
year = 1999
babies_year <- babynames[babynames$year == year, ]
# you can use View(babies.year) to see it
# babynames$ year == year is a vector of booleans

max_name_count <- max(babies_year$n)
(babies_year$name)[max_name_count == babies_year$n]
```

```
## [1] "Jacob"
```

```r
# returns vector of most popular names
```

In the above example, the most popular name is a boy's name. For a girl's name,

```r
babies_year_f <- babies_year[babies_year$sex == "F", ]
# We can do this because babies_year is still a dataframe. Then do the same
max_name_count <- max(babies_year_f$n)
(babies_year_f$name)[max_name_count == babies_year_f$n]
```

```
## [1] "Emily"
```

We can write a function for this.

```r
most_common_name <- function(babynames, year, sex){
  babies <- babynames[babynames$year == year & babynames$sex == sex, ]
  max_name_count <- max(babies$n)
  (babies$name)[babies$n == max_name_count]
}
most_common_name(babynames, 2003, "M")
```

```
## [1] "Jacob"
```

Tidyverse

Function Composition

```r
f <- function(x){
  x^2
}

g <- function(y){
  y + 2
}

# To compose both functions, we can do
f(g(5))
```

```
## [1] 49
```

```r
# but also
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.2     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.2     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

```r
5 %>% g %>% f
```

```
## [1] 49
```

```
# or
5 %>% g() %>% f()
```

```
## [1] 49
```

You can also filter a dataframe.

```
filter(babynames, year == 1880, sex == "F")
# This is equivalent to
babynames %>% filter(year == 1880, sex == "F")
# and
babynames %>% filter(year == 1880) %>% filter(sex == "F")

# To create a new dataframe with the same syntax,
babynames %>% select(year, name) # n by 2
# You lose everything else, e.g. number of occurences
```

Summarize

Summarize computes some function of a column.

```
babynames %>% summarize(pername = mean(n))
# This returns a dataframe pername with the mean of n

# For the average count for sex == "M" and sex == "F",
babynames %>% filter(sex == "M") %>% summarize(pername = mean(n))
babynames %>% filter(sex == "F") %>% summarize(pername = mean(n))

# group rows
babynames %>% group_by(sex) %>% summarize(pername = mean(n))
# returns 2 by 2 dataframe with columns sex and pername
```

Exercise

Using `gapminder`, write code to computer the number of countries in Asia. Write a function that takes in the name of the continent, and computes then number of countries on that continent.

```
library(gapminder)
countries_on_continent <- function(cont){
  on_continent <- filter(gapminder, continent == cont, year == 1992)
  dim(on_continent)[1]
}
countries_on_continent("Asia")
```

```
## [1] 33
```