2. A Tidyverse Primer

Owen Hale

Sort the `mtcars` data frame by the columns `gear` and `mpg`.

```
#base R order function
mtcars[order(mtcars$gear, mtcars$mpg),]

#dplyr arrange function
library(dplyr)
arrange(.data = mtcars, gear, mpg)
```

Sort the same dataset and get the first 10 rows:

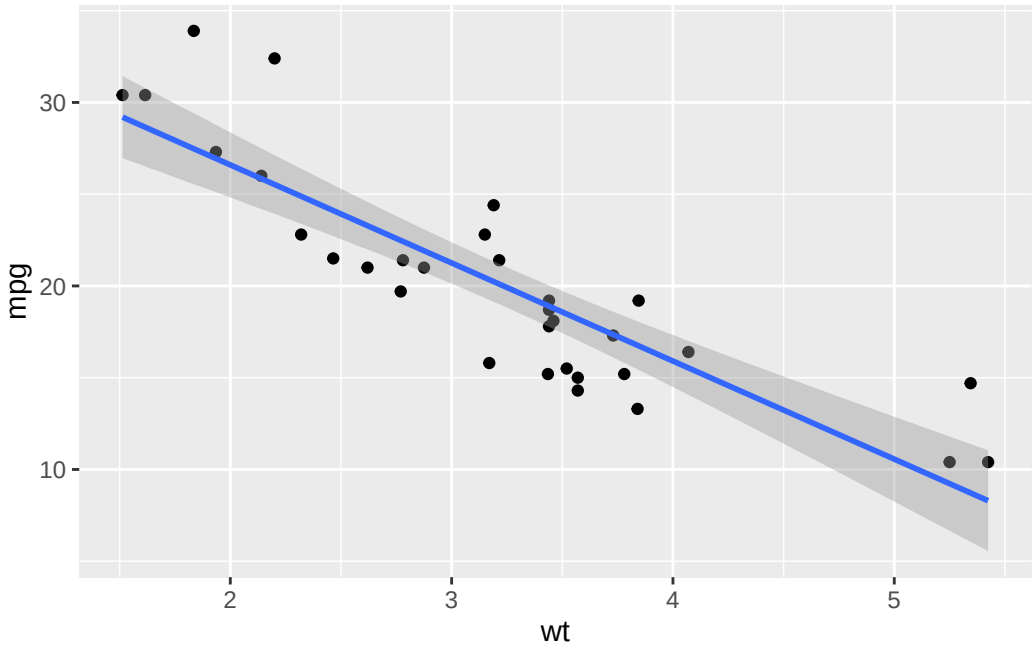
```
# base R
small_mtcars <- arrange(mtcars, gear)
small_mtcars <- slice(small_mtcars, 1:10)

# compact base R
small_mtcars <- slice(arrange(mtcars, gear), 1:10)

# tidyverse with pipe
small_mtcars <-
  mtcars %>%
  arrange(gear) %>%
  slice(1:10)
```

Make a scatter plot with a regression line:

```
library(ggplot2)
ggplot(mtcars, aes(x = wt, y = mpg)) +
  geom_point() +
  geom_smooth(method = lm)
```



Functional programming:

Functional programming helps to replace loops with tidyverse functions.

Calculate log ratio of `fuel efficiency` to `car weight`:

```
# Base R
n <- nrow(mtcars)
ratios <- rep(NA_real_, n)
for (car in 1:n) {
  ratios[car] <- log(mtcars$mpg[car]/mtcars$wt[car])
}

# tidyverse
ratios <- log(mtcars$mpg/mtcars$wt)
```

The `purrr` package is full of functional programming tools. Use it to take the square root of the data using the `map` function.

```
map(head(mtcars$mpg, 3), sqrt)
```

Different versions of the map function can be used when we know what variable type it will return. For example, here it returns a double-precision number:

```
map_dbl(head(mtcars$mpg, 3), sqrt)
```

Additionally, we can apply functions to two vectors. Here, we calculate the log ratio of two vectors from earlier:

```
log_ratios <- map2_dbl(mtcars$mpg, mtcars$wt, compute_log_ratio)
```

Temporary, anonymous functions can be applied to our vectors using `map2()` and the variables `.x` and `.y`.

```
map2_dbl(mtcars$mpg, mtcars$wt, ~ log(.x/.y))
```

Tibbles

A tibble is a rectangular object that is easier to work with than a normal data frame.

- allow invalid column names
- force user to use the entire column name to avoid confusion
- maintain two dimensions even with just one column
- better printing

Sample workflow:

Here is an example data processing pipeline using the tidyverse:

```
# load packages
library(tidyverse)
library(lubridate)

# link to data
url <- "https://data.cityofchicago.org/api/views/5neh-572f/rows.csv?accessType=DOWNLOAD&bo

all_stations <-
  read_csv(url) %>% # load data
  dplyr::select(station = stationname, date, rides) %>% # select columns & filter data
  mutate(date = mdy(date), rides = rides / 1000) %>% # convert dates into char, convert ri
  group_by(date, station) %>%
  summarize(rides = max(rides), .groups = "drop") # max rides for each day & station
```