Lab6

Irlanda Ayon-Moreno

4/11/2018

Lab 6: More data wrangling and outputs

Gaston Sanchez

Learning Objectives:

- Work with a more complex file structure
- Practice exporting tables
- Practice exporting R output (as is)
- Practice exporting plot images
- Learn about `"dplyr"` pipelines
- Get to know the pipe operator `%>%`
- Chain dplyr operations with the piper

Manipulating and Visualizing Data Frames

In this lab, you will continue manipulating data frames with `"dplyr"`, and plotting graphics with `"ggplot2"`. In addition, you will also use various functions to export (or save) tables, images, and R output to external files.

While you follow this lab, you may want to open these cheat sheets:

- dplyr cheatsheet
- ggplot2 cheatsheet

Filestructure

To help you better prepare for HW02, we want you to practice working with a more sophisticated file structure. Follow the steps listed below to create the necessary subdirectories like those depicted in this scheme:

```
lab06/
   README.md
   data/
   code/
   output/
   images/
```

- Open a shell terminal (e.g. command line or GitBash)

- Change your working directory to a location where you will store all the materials for this lab
- Use `mkdir` to create a directory `lab06`
- `cd` to `lab06`
- Use `mkdir` to create other subdirectories: `data`, `code`, `output`, `images`
- List the contents of `lab06` to confirm that you have all the subdirectories
- Use `touch` to create an empty `README.md` text file.
- Open the `README.md` file with a text editor (e.g. the one in RStudio) and add a brief description of what this lab is about. Save the changes.
- `cd` to `data/`
- Download the data file with the command `curl`, and the `-O` option (letter O)

```
curl -O https://raw.githubusercontent.com/ucb-stat133/stat133-spring-2018/master/data/nba2
```

- Use `ls` to confirm that the csv file is in `data/`
- Use *word count* `wc` to count the lines of the csv file
- Take a peek at the first rows of the csv file with `head`
- Take a peek at the last 5 rows of the csv file with `tail`

R script file

- Once you have the filestructure for this lab, go to RStudio and open a new `R` script file (do NOT confuse with an `Rmd` file).
- Save the `R` script file as `lab06-script.R` in the `code/` folder of `lab06/`

R script files are used to write R code only, using R syntax. In other words, you should NOT use Markdown or LaTeX syntax inside an R script file. Why? Because if you run the entire script, R will try to execute all the commands, and won't be able to recognize Markdown, LaTeX, yaml, or other syntaxes.

File Header

Let's start with some good coding practices by adding a header to the `R` script file in the form of R comments. In general, the header section should contain a title, a description of what the script is about, what are the inputs, and what are the main outputs produced when executing the code in the script. Optionally, you can also include the name of the author, the date, and other details. Something like this:

```
# Title: Short title (one sentence)
# Description: what the script is about (one paragraph or two)
# Input(s): what are the main inputs (list of inputs)
```

```
# Output(s): what are the main outputs (list of outputs)
# Author(s): First Last
# Date: mm-dd-yyyy
```

Think of the header of a script file as the yaml header used in `Rmd` files. The header should be the very first thing that appears at the top of the script file. Personally, I like to surround the header in my R script files with some delimiting characters that help the reader to visually identify main parts of the script. Here's a hypothetical example of a header:

```
# ======================================================================
# Title: Cleaning Data
# Description:
#   This script performs cleaning tasks and transformations on
#   various columns of the raw data file.
# Input(s): data file 'raw-data.csv'
# Output(s): data file 'clean-data.csv'
# Author: Gaston Sanchez
# Date: 2-27-2018
# ======================================================================
```

Another good coding practice is to avoid writing very long lines of code. Most coding style guides stick to a maximum line width of 80 characters, and this is the magic number that I also use for my scripts.

Your turn: Include a header in your R script file, respecting the width limit of 80 characters. Save this file in the `code/` folder.

Required Packages

The next thing that you need to include in your script file are the required packages. Not all script files need packages, but many do. When this is the case, loading the packages should be the first lines of code to be executed.

Include the commands to load the following packages in your script:

```
# packages
library(readr)    # importing data
library(dplyr)    # data wrangling
```

```
## Warning: package 'dplyr' was built under R version 3.4.2
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(ggplot2)  # graphics
```

In addition to loading the packages, sometimes you will also need to load code from other script files. We won't do that today, but you should know that this is very common as the complexity and size of your projects grow.

Exporting some data tables

After the header, and the loading-packages sections, the next part in your lab script involves importing the data. In addition to importing a data table, you are also going to practice exporting tables. That is, writing data tables to external files.

- We want you to work with relative paths. To execute the commands that read input(s) and write output(s), you will need to set the working directory. On the menu bar go to **Session**, then select the option **Set Working Directory**, and choose **To Source File Location**.
- Use `read_csv()` from the package `"readr"` to import the data `nba2017-players.csv` in R. Do this by specifying a relative path.

```
nba2017_players <- read_csv(file = "../data/nba2017-players.csv")
```

```
## Parsed with column specification:
## cols(
##   player = col_character(),
##   team = col_character(),
##   position = col_character(),
##   height = col_integer(),
##   weight = col_integer(),
##   age = col_integer(),
##   experience = col_integer(),
##   college = col_character(),
##   salary = col_double(),
##   games = col_integer(),
##   minutes = col_integer(),
##   points = col_integer(),
##   points3 = col_integer(),
##   points2 = col_integer(),
##   points1 = col_integer()
## )
```

- Use the imported tibble to create a data frame `warriors` by selecting rows—e.g. `filter()`—of Golden State Warriors, arranging rows by salary in increasing order.

```
warriors <- arrange(filter(nba2017_players, team == "GSW"), salary)
head(warriors, 3)
```

```
## # A tibble: 3 x 15
##                 player  team position height weight   age experience
##                  <chr> <chr>    <chr>  <int>  <int> <int>      <int>
## 1          Matt Barnes   GSW       SF     79    226    36         13
```

```
## 2        Patrick McCaw   GSW       SG     79    185    21          0
## 3 James Michael McAdoo   GSW       PF     81    230    24          2
## # ... with 8 more variables: college <chr>, salary <dbl>, games <int>,
## #   minutes <int>, points <int>, points3 <int>, points2 <int>,
## #   points1 <int>
```

- Use the function `write.csv()` to export (or save) the data frame `warriors` to a data file `warriors.csv` in the `data/` directory. You will need to use a relative path to specify the `file` argument. Also, see how to use the argument `row.names` to avoid including a first column of numbers.

```
write.csv(warriors, file = '../data/warriors.csv', row.names = FALSE)
write.csv(warriors, file = '../data/warriors2.csv', row.names = TRUE)
```

- Create another data frame `lakers` by selecting rows of Los Angeles Lakers, this time arranging rows by experience (decreasingly).

```
lakers <- filter(nba2017_players, team == "LAL") %>%
  arrange(desc(experience))
head(lakers, 3)
```

```
## # A tibble: 3 x 15
##              player  team position height weight   age experience
##               <chr> <chr>    <chr>  <int>  <int> <int>      <int>
## 1 Metta World Peace   LAL       SF     78    260    37         16
## 2         Luol Deng   LAL       SF     81    220    31         12
## 3      Corey Brewer   LAL       SF     81    186    30          9
## # ... with 8 more variables: college <chr>, salary <dbl>, games <int>,
## #   minutes <int>, points <int>, points3 <int>, points2 <int>,
## #   points1 <int>
```

- Now use the function `write_csv()` to export (or save) the data frame `lakers` to a data file `lakers.csv` in the `data/` directory. You will also need to use a relative path to specify the `file` argument.

```
write_csv(lakers, path = '../data/lakers.csv')
```

Exporting some R output

After exporting the tables to the corresponding csv files, you will produce some summary statistics, and then save the generated output to external text files. To do this, you will have to learn about the `sink()` function, which sends R output to a specified file.

Say you are interested in exporting the summary statistics of `height` and `weight`, exactly in the same way they are displayed by R:

```
summary(nba2017_players[ ,c('height', 'weight')])
```

```
##      height          weight
##  Min.   :69.00   Min.   :150.0
##  1st Qu.:77.00   1st Qu.:200.0
```

```
##  Median :79.00   Median :220.0
##  Mean   :79.15   Mean   :220.2
##  3rd Qu.:82.00   3rd Qu.:240.0
##  Max.   :87.00   Max.   :290.0
```

One naive option to "export" this output would be to manually copy the text displayed on the console, and then paste it to a text file. While this may work, it is labor intensive, error prone, and highly irreproducible. A better way to achieve this task is with the `sink()` function. Here's how:

```
# divert output to the specified file
sink(file = '../output/summary-height-weight.txt')
summary(nba2017_players[ ,c('height', 'weight')])
```

```
##      height          weight
##  Min.   :69.00   Min.   :150.0
##  1st Qu.:77.00   1st Qu.:200.0
##  Median :79.00   Median :220.0
##  Mean   :79.15   Mean   :220.2
##  3rd Qu.:82.00   3rd Qu.:240.0
##  Max.   :87.00   Max.   :290.0
```

```
sink()
```

The fist call to `sink()` opens a connection to the specified file, and then all outputs are diverted to that location. The second call to `sink()`, i.e. the one without any arguments, closes the connection.

While you are `sink()`ing output to a specified file, all the results will be sent to such file. In other words, nothing will be printed on the console. Only after the sinking process has finished and the connection is closed, you will be able to execute commands and see results displayed on R's console.

Why sinking?

Why would you ever want to `sink()` R outputs to a file? Why not simply display them as part of your `Rmd` file? One good reason for diverting output to an external file is for convenience. In practice, the reports and documents (e.g. papers, executive summaries, slides) of a data analysis project won't contain everything that you tried, explored, analyzed, and graphed. There will be many intermediate results that, while relevant for a specific stage of the analysis cycle, are innecessary for the final report. So a good way to keep these intermediate outputs is by exporting them with `sink()`.

Your turn:

- Export the output of `str()` on the data frame with all the players. `sink()` the output, using a relative path, to a text file `data-structure.txt`, in the `output/` folder.

```
sink(file = '../output/data-structure.txt')
str(nba2017_players)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    441 obs. of  15 variables:
##  $ player    : chr  "Al Horford" "Amir Johnson" "Avery Bradley" "Demetrius Jackson" ...
##  $ team      : chr  "BOS" "BOS" "BOS" "BOS" ...
```

```
##  $ position  : chr  "C" "PF" "SG" "PG" ...
##  $ height    : int  82 81 74 73 79 69 78 78 79 82 ...
##  $ weight    : int  245 240 180 201 205 185 235 215 225 231 ...
##  $ age       : int  30 29 26 22 31 27 26 21 20 29 ...
##  $ experience: int  9 11 6 0 9 5 4 2 0 6 ...
##  $ college   : chr  "University of Florida" NA "University of Texas at Austin" "University o
##  $ salary    : num  26540100 12000000 8269663 1450000 1410598 ...
##  $ games     : int  68 80 55 5 47 76 72 29 78 78 ...
##  $ minutes   : int  2193 1608 1835 17 538 2569 2335 220 1341 1232 ...
##  $ points    : int  952 520 894 10 262 2199 999 68 515 299 ...
##  $ points3   : int  86 27 108 1 39 245 157 12 46 45 ...
##  $ points2   : int  293 186 251 2 56 437 176 13 146 69 ...
##  $ points1   : int  108 67 68 3 33 590 176 6 85 26 ...
##  - attr(*, "spec")=List of 2
##   ..$ cols   :List of 15
##   .. ..$ player    : list()
##   .. .. ..- attr(*, "class")= chr  "collector_character" "collector"
##   .. ..$ team      : list()
##   .. .. ..- attr(*, "class")= chr  "collector_character" "collector"
##   .. ..$ position  : list()
##   .. .. ..- attr(*, "class")= chr  "collector_character" "collector"
##   .. ..$ height    : list()
##   .. .. ..- attr(*, "class")= chr  "collector_integer" "collector"
##   .. ..$ weight    : list()
##   .. .. ..- attr(*, "class")= chr  "collector_integer" "collector"
##   .. ..$ age       : list()
##   .. .. ..- attr(*, "class")= chr  "collector_integer" "collector"
##   .. ..$ experience: list()
##   .. .. ..- attr(*, "class")= chr  "collector_integer" "collector"
##   .. ..$ college   : list()
##   .. .. ..- attr(*, "class")= chr  "collector_character" "collector"
##   .. ..$ salary    : list()
##   .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
##   .. ..$ games     : list()
##   .. .. ..- attr(*, "class")= chr  "collector_integer" "collector"
##   .. ..$ minutes   : list()
##   .. .. ..- attr(*, "class")= chr  "collector_integer" "collector"
##   .. ..$ points    : list()
##   .. .. ..- attr(*, "class")= chr  "collector_integer" "collector"
##   .. ..$ points3   : list()
##   .. .. ..- attr(*, "class")= chr  "collector_integer" "collector"
##   .. ..$ points2   : list()
##   .. .. ..- attr(*, "class")= chr  "collector_integer" "collector"
##   .. ..$ points1   : list()
##   .. .. ..- attr(*, "class")= chr  "collector_integer" "collector"
##   ..$ default: list()
##   .. ..- attr(*, "class")= chr  "collector_guess" "collector"
##   ..- attr(*, "class")= chr "col_spec"
```

```
sink()
```

- Export the `summary()` of the entire data frame `warriors` to a text file `summary-warriors.txt`, in the `output/` folder (also use a relative path).

```
sink(file = "../output/summary-warriors.txt")
summary(warriors)
```

```
##     player              team              position             height
##  Length:15          Length:15          Length:15          Min.   :75.00
##  Class :character   Class :character   Class :character   1st Qu.:79.00
##  Mode  :character   Mode  :character   Mode  :character   Median :79.00
##                                                           Mean   :79.87
##                                                           3rd Qu.:81.00
##                                                           Max.   :84.00
##      weight           age          experience       college
##  Min.   :175.0   Min.   :20.00   Min.   : 0.000   Length:15
##  1st Qu.:203.5   1st Qu.:24.50   1st Qu.: 2.500   Class :character
##  Median :226.0   Median :28.00   Median : 7.000   Mode  :character
##  Mean   :223.5   Mean   :27.73   Mean   : 6.733
##  3rd Qu.:242.5   3rd Qu.:31.50   3rd Qu.:11.500
##  Max.   :270.0   Max.   :36.00   Max.   :13.000
##      salary             games         minutes         points
##  Min.   :  383351   Min.   :10.0   Min.   :  85   Min.   :  19.0
##  1st Qu.: 1093628   1st Qu.:57.5   1st Qu.: 598   1st Qu.: 214.5
##  Median : 1551659   Median :71.0   Median :1137   Median : 426.0
##  Mean   : 6579394   Mean   :63.0   Mean   :1309   Mean   : 631.5
##  3rd Qu.:11621864   3rd Qu.:76.5   3rd Qu.:2034   3rd Qu.: 675.0
##  Max.   :26540100   Max.   :79.0   Max.   :2649   Max.   :1999.0
##     points3          points2         points1
##  Min.   :  0.00   Min.   :  8.0   Min.   :  3.00
##  1st Qu.:  1.50   1st Qu.: 62.5   1st Qu.: 25.00
##  Median : 18.00   Median :155.0   Median : 44.00
##  Mean   : 65.47   Mean   :169.3   Mean   : 96.47
##  3rd Qu.: 72.50   3rd Qu.:199.5   3rd Qu.:124.50
##  Max.   :324.00   Max.   :434.0   Max.   :336.00
```

```
sink()
```

- Export another `summary()` of the entire data frame `lakers` to a text file `summary-lakers.txt`, in the `output/` folder (using a relative path).

```
pwd
cd ../output/
ls
```

```
## /Users/irlandaayon-moreno/Desktop/stat133/labs/lab6/code
## data-structure.txt
## summary-height-weight.txt
## summary-lakers.txt
```

```
## summary-warriors.txt
```

```
sink(file = "../otuput/summary-lakers.txt")
summary(lakers)
sink()
```

Exporting some "base" graphs

In the same way that R output, as it appears on the console, can be exported to some files, you can do the same with graphics and plots. Actually, saving plot images is much more common than `sink()`ing output.

Base R provides a wide array of functions to save images in most common formats:

- `png()`
- `jpeg()`
- `tiff()`
- `bmp()`
- `svg()`
- `pdf()`

Similar to the writing table functions such as `write.table()` or `write.csv()`, and the `sink()` function, the graphics device functions require a file name to be provided. Here's how to save a simple scatterpot of `height` and `weight` in png format to the folder `images/`:

```
# saving a scatterplot in png format
png(filename = "../images/scatterplot-height-weight.png")
plot(nba2017_players$height, nba2017_players$weight, pch = 20,
     xlab = 'Height', ylab = 'Height')
dev.off()
```

```
## pdf
##   2
```

- The function `png()` tells R to save the image in PNG format, using the provided filename.
- Invoking `png()` will open a graphics device; not the graphics device of RStudio, so you won't be able to see the graphic.
- The `plot()` function produces the scatterplot.
- The function `dev.off()` closes the graphics device.

Your turn:

- Save another version of the scatterplot between `height` and `weight`, but now try to get an image with higher resolution. Save the plot in `images/`.

```
png(filename = '../images/scatterplot2-height-weight.png', pointsize = 20)
plot(nba2017_players$height, nba2017_players$weight, las = 1, pch = 19,
     xlab = 'Height', ylab = 'Weight')
dev.off()
```

```
## pdf
##   2
```

- Save a histogram in JPEG format of `age` with dimensions (width x height) 600 x 400 pixels. Save the plot in `images/`.

```
jpeg(filename = "../images/histogram-age",
     width = 600,
     height = 400,
     units = "px")
hist(nba2017_players$age, xlab = 'Age', las = 1, col = 'gray80')
dev.off()
```

```
## pdf
##   2
```

- Use `pdf()` to save the previous histogram of `age` in PDF format, with dimensions (width x height) 7 x 5 inches.

```
pdf(file = "../images/histogram2-age",
     width = 7,
     height = 5)
hist(nba2017_players$age, xlab = 'Age', las = 1, col = 'gray80')
dev.off()
```

```
## pdf
##   2
```

Exporting some ggplots

The package `"ggplot2"` comes with a wrapper function `ggsave()` that allows you to save ggplot graphics to a specified file. By default, `ggsave()` saves images in PDF format.

- Use `ggplot()` to make a scatterplot of `points` and `salary`, and store it in a ggplot object named `gg_pts_salary`. Then use `ggsave()` to save the plot with dimensions (width x height) 7 x 5 inches; in the `images/` folder as `points_salary.pdf`

```
gg_pts_salary <- ggplot(data = nba2017_players, aes(x = points, y = salary)) +
  geom_point()

ggsave(filename = '../images/points_salary.pdf',
       plot = gg_pts_salary,
       width = 7, height = 5, units = "in")
```

- Use `ggplot()` to create a scatterplot of `height` and `weight`, faceting by `position`. Store this in a ggplot object `gg_ht_wt_positions` Then use `ggsave()` to save the plot with dimensions (width x height) 6 x 4 inches; in the `images/` folder as `height_weight_by_position.pdf`

```
gg_ht_wt_positions <- ggplot(data = nba2017_players, mapping = aes(x = height, y = weight)) +
  geom_point(aes(color = position)) +
  facet_grid(.~position)

ggsave(filename = "../images/height_weight_by_position.pdf",
```

```
       plot = gg_ht_wt_positions,
       width = 6, height = 5, units = "in")
```

More "dplyr"

The last part of this lab involves working the pipe operator %>% which allows you write function calls in a more human-readable way. This becomes extremely useful in "dplyr" operations that require many steps.

The behavior of "dplyr" is functional in the sense that function calls don't have side-effects. You must always save their results in order to keep them in an object (in memory). This doesn't lead to particularly elegant code, especially if you want to do many operations at once. You either have to do it step-by-step:

```
# manipulation step-by-step
dat1 <- group_by(dat, team)
dat2 <- select(dat1, team, height, weight)
dat3 <- summarise(dat2,
  avg_height = mean(height, na.rm = TRUE),
  avg_weight = mean(weight, na.rm = TRUE))
dat4 <- arrange(dat3, avg_height)
dat4
```

```
## # A tibble: 30 x 3
##    team  avg_height avg_weight
##    <chr>      <dbl>      <dbl>
##  1 BOS         78.2        220
##  2 HOU         78.3        215
##  3 SAC         78.5        217
##  4 IND         78.5        226
##  5 CHI         78.5        216
##  6 PHO         78.5        214
##  7 BRK         78.7        222
##  8 CHO         78.8        213
##  9 LAC         78.8        225
## 10 CLE         78.9        226
## # ... with 20 more rows
```

Or if you don't want to name the intermediate results, you need to wrap the function calls inside each other:

```
# inside-out style (hard to read)
arrange(
  summarise(
    select(
      group_by(dat, team),
      team, height, weight
```

```
    ),
    avg_height = mean(height, na.rm = TRUE),
    avg_weight = mean(weight, na.rm = TRUE)
  ),
  avg_height
)
```

```
## # A tibble: 30 x 3
##    team  avg_height avg_weight
##    <chr>      <dbl>      <dbl>
##  1 BOS         78.2        220
##  2 HOU         78.3        215
##  3 SAC         78.5        217
##  4 IND         78.5        226
##  5 CHI         78.5        216
##  6 PHO         78.5        214
##  7 BRK         78.7        222
##  8 CHO         78.8        213
##  9 LAC         78.8        225
## 10 CLE         78.9        226
## # ... with 20 more rows
```

This is difficult to read because the order of the operations is from inside to out. Thus, the arguments are a long way away from the function. To get around this problem, `"dplyr"` provides the `%>%` operator from `"magrittr"`.

`x %>% f(y)` turns into `f(x, y)` so you can use it to rewrite multiple operations that you can read left-to-right, top-to-bottom:

```
# using %>%
nba2017_players %>%
  group_by(team) %>%
  select(team, height, weight) %>%
  summarise(
    avg_height = mean(height, na.rm = TRUE),
    avg_weight = mean(weight, na.rm = TRUE)) %>%
  arrange(avg_height)
```

```
## # A tibble: 30 x 3
##    team  avg_height avg_weight
##    <chr>      <dbl>      <dbl>
##  1 BOS         78.2        220
##  2 HOU         78.3        215
##  3 SAC         78.5        217
##  4 IND         78.5        226
##  5 CHI         78.5        216
##  6 PHO         78.5        214
##  7 BRK         78.7        222
##  8 CHO         78.8        213
```

```
##  9 LAC          78.8        225
## 10 CLE          78.9        226
## # ... with 20 more rows
```

Use the piper operator `"%>%"` to perform the following operations:

- display the player names of Lakers `'LAL'`.

```
nba2017_players %>%
  filter(team == "LAL") %>%
  select(player)
```

```
## # A tibble: 15 x 1
##                player
##                 <chr>
##  1    Brandon Ingram
##  2      Corey Brewer
##  3  D'Angelo Russell
##  4       David Nwaba
##  5       Ivica Zubac
##  6   Jordan Clarkson
##  7     Julius Randle
##  8   Larry Nance Jr.
##  9         Luol Deng
## 10 Metta World Peace
## 11        Nick Young
## 12       Tarik Black
## 13   Thomas Robinson
## 14    Timofey Mozgov
## 15       Tyler Ennis
```

- display the name and salary of GSW point guards 'PG'.

```
nba2017_players %>%
  filter(team == "GSW" & position == "PG") %>%
  select(player, salary)
```

```
## # A tibble: 2 x 2
##             player   salary
##              <chr>    <dbl>
## 1 Shaun Livingston  5782450
## 2    Stephen Curry 12112359
```

- dislay the name, age, and team, of players with more than 10 years of experience, making 10 million dollars or less.

```
nba2017_players %>%
  filter(experience > 10 & salary <= 10000000) %>%
  select(player, age, team)
```

```
## # A tibble: 32 x 3
##               player   age  team
```

```
##                <chr> <int> <chr>
##  1     Dahntay Jones    36   CLE
##  2    Deron Williams    32   CLE
##  3       James Jones    36   CLE
##  4       Kyle Korver    35   CLE
##  5 Richard Jefferson    36   CLE
##  6     Jose Calderon    35   ATL
##  7    Kris Humphries    31   ATL
##  8     Mike Dunleavy    36   ATL
##  9       Jason Terry    39   MIL
## 10        C.J. Miles    29   IND
## # ... with 22 more rows
```

- select the name, team, height, and weight, of rookie players, 20 years old, displaying only the first five occurrences (i.e. rows).

```
nba2017_players %>%
  filter(experience == 0 & age == 20) %>%
  select(player, team, height, weight) %>%
  slice(1:5)
```

```
## # A tibble: 5 x 4
##              player  team height weight
##               <chr> <chr>  <int>  <int>
## 1      Jaylen Brown   BOS     79    225
## 2    Henry Ellenson   DET     83    245
## 3 Stephen Zimmerman   ORL     84    240
## 4   Dejounte Murray   SAS     77    170
## 5    Chinanu Onuaku   HOU     82    245
```

- create a data frame `gsw_mpg` of GSW players, that contains variables for `player` name, `experience`, and `min_per_game` (minutes per game), sorted by `min_per_game` (in descending order).

```
gsw_mpg <- nba2017_players %>%
  filter(team == "GSW") %>%
  mutate(min_per_game = minutes/games) %>%
  select(player, experience, min_per_game) %>%
  arrange(desc(min_per_game))
```

- display the average triple points by team, in ascending order, of the bottom-5 teams (worst 3pointer teams).

```
nba2017_players %>%
  group_by(team) %>%
  summarise(avg_points3 = mean(points3)) %>%
  arrange(avg_points3) %>%
  slice(1:5)
```

```
## # A tibble: 5 x 2
##    team avg_points3
```

```
##   <chr>       <dbl>
## 1   NOP    36.64286
## 2   SAC    37.20000
## 3   PHO    37.60000
## 4   CHI    37.66667
## 5   LAL    39.46667
```

- obtain the mean and standard deviation of `age`, for Power Forwards, with 5 and 10 years (including) of experience.

```
nba2017_players %>%
  filter(position == "PF" & experience >=5 & experience <=10) %>%
  summarise(
    pf_avg_age = mean(age),
    pf_sd_age = sd(age)
  )
```

```
## # A tibble: 1 x 2
##   pf_avg_age pf_sd_age
##        <dbl>     <dbl>
## 1   28.12903  1.802627
```