Lab 5

David Schultheiss

11/1/2020

```
library(tidyverse)
```

```
## -- Attaching packages ------------------------------------- tidyverse 1.3.0 --
```

```
## v ggplot2 3.3.2     v purrr   0.3.3
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   1.0.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
```

```
## Warning: package 'ggplot2' was built under R version 3.6.2
```

```
## -- Conflicts ---------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following object is masked from 'package:base':
##
##     date
```

Question 1

You just need to load in the data and do some pre-processing. Notice that there are 25 different data files, so we will need to do something a little different here. **NOTE**, you **NEED** to make sure the `lab_5_data` folder is in the same directory as your R Markdown (or R) script. We create a vector called `files` that stores the **file path** for each files, and then use the `lapply()` function to load in every element in the vector. As a general outline, we need to

1. Load in all the data files (this has been done for you),
2. Rename all column names to remove spaces and backslashes,
3. Change the `Date`, `Start_Time`, and `End_Time` columns to be time referenced (think back to a previous lab),
4. Change the `Position_Name` column to reduce the number of positions to be `Midfielder`, `Striker`, `Goal Keeper`, `Defender`, and `Wing` (from previous lab).

Answer

```
files = list.files('/Users/davidschultheiss/Downloads/lab_5_data', pattern="*.csv", full.names=TRUE)

dat = lapply(files, read_csv, col_types = cols()) %>%
  bind_rows() %>%
  rename_all( ~str_replace_all(., " ", "_") ) %>%
  rename_all( ~str_replace_all(., "/", "_") ) %>%
  mutate(Date = mdy(Date),
         Start_Time = hms(Start_Time),
         End_Time = hms(End_Time)) %>%
  mutate(Position_Name = Position_Name %>%
           str_extract(c("Midfielder|Defender|Back|Stikder|Goal Keeper|Wing|Striker")) %>%
           str_replace_all("Back", "Defender"))
```

Question 2

Here we are going to start exploring confident intervals (sort of). Looking at the four metrics `Player_Load_Per_Minute`, `Meterage_Per_Minute`, `Maximum_Velocity`, and `Total_Distance`, we want to plot the 95% confidence interval by position over time. Note that in the R chunk statement, I have included a couple extra arguments; please do not delete these, they are only there to size your final plot. For the data construction step, our general outline is

1. Select the appropriate columns,
2. Filter out `NA`s
3. Pivot from wide to long format
4. Find the mean, lower CI, and upper CI grouping by position, date, and metric.

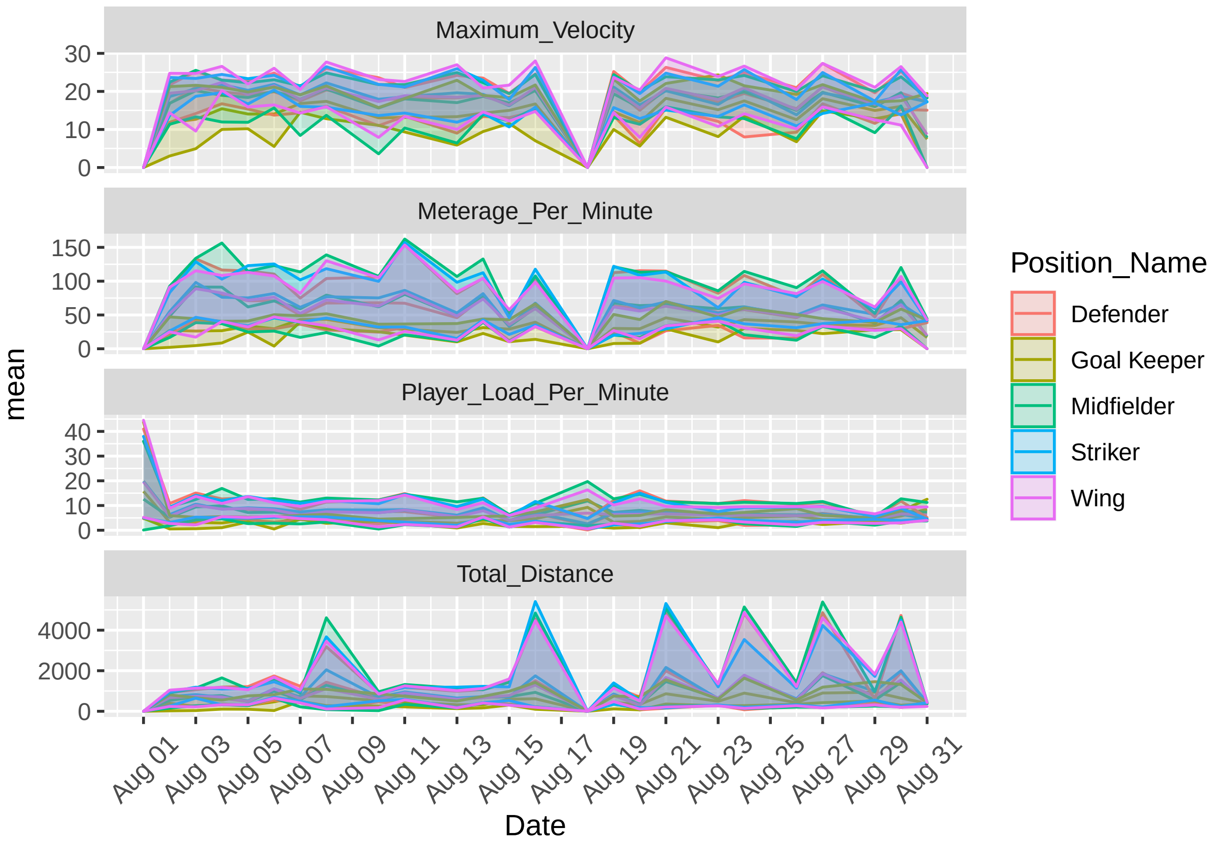
Then, to plot this, our general outline is

1. Choose the x variable,
2. Choose the y variable,
3. Choose if you want to color and/or fill the lines/bounds by a variable (probably should do this),
4. Make a line,
5. Use `geom_ribbon()` to create the CI, where you pass in what the lower bound should be and what the upper bound should be,
6. Make the plot pretty (e.g., proper labels, perhaps a legend is not needed, etc.).

Answer

```
dat_ci =  dat %>%
  select(Date, Position_Name, Period_Name, Player_Load_Per_Minute, Meterage_Per_Minute, Maximum_Velocity
  filter(complete.cases(.)) %>%
  filter(Period_Name != 'Session') %>%
  pivot_longer(-c(Date, Position_Name, Period_Name), names_to = 'Metric', values_to = 'Values') %>%
  group_by(Date, Position_Name, Metric) %>%
  summarise_at(vars(Values), list(mean = mean,
                                  lower = ~ quantile(.,probs = 0.025),
                                  upper = ~ quantile(.,probs = 0.975))) %>%
  ungroup()
```

```r
ggplot(dat_ci, aes(x = Date , y = mean, color = Position_Name, fill= Position_Name)) +
  geom_line() +
  geom_ribbon(aes(ymin = lower, ymax = upper), alpha=0.2) +
  facet_wrap(~Metric , scale = 'free_y', nrow = 5) +
  scale_x_date(date_breaks ='2 days', date_labels = "%b %d") +
  theme(axis.text.x =element_text(angle = 45, vjust = 0.5, size = 10))
```



Question 3

Part a)

For this question, instead of filling in/writing your own code, you will be analyzing what I did. The "Question 3" code chunk below has four different comments, each is associated with a number. You need to answer the comment that is associated with the same numbered bullet point.

1. Why do we filter out all rows where the `Period_Name` is `Session`?
2. What do these four lines of code do?
3. What do these four lines of code do?
4. What is the effect of having the `pivot_longer` statement before the `summarize_at` statement? What would happen if they were switched?

Answer

1. I'm not sure exactly what 'session' is, but all of its values are extreme outliers from the rest of the measurement periods. My guess is that this data is taken from training sessions, because of these high values. Excluding it allows us to just view data related to games.
2. They are creating new variables at the end of our dataset (vd, ve, ae, ad) by summing up different portions of existing varaibles. For instance, vd is the sum of Velocity_Band_2_Total_Distance through Velocity_Band_8_Total_Distance. Missing values are also removed.
3. The varaibles used in #2 to calculate sums are transformed into proportions. For instance, Velocity_Band_2_Total_Distance makes up 51.9% of a player's vd (sum of all velocity bands) on a certain date/period.
4. If we used summarize_at first, the code needed would demand calling many more variables. If we wanted to pivot the table afterwards, it would require calling a lot more variables. Using pivot_longer first allows for more concise code.

Part b)

Below are four figures, Figures **??**, **??**, **??**, and **??**, that are created using the constructed data from part a. **NOTE**, you will need to load in the data from Question 1 for the figures to show up. For the four figures below, answer the following:

1. Is it better to represent the data as percent of time spent in each band, or would it have been better to not transform the data and plot the raw values (i.e., the values contained in the original data)? Explain.
2. Are the four figures comparable? Explain.
3. Are the figures meaningful, and if so, what conclusions can you draw from them?
4. Is it better to have all of the y-axis on the same scale (withing figures and/or across figures), or should the y-axis be specific to each subplot?

Answer

1. The proportions help to standardize the scale and make it much easier to analyze.
2. Yes you can compare the effort put into velocity/acceleration bands with the total distance travelled for each. Velocity and acceleration are also related.

3. It would be helpful to have a data description so I knew what I was actually looking at here. I'm not sure where that is, so can only make basic observations. For acceleration effort and distance, we see that where there is more effort there is more distance. These two things occur in tandem. For velocity, this isn't true. Most distance covered comes from bands 1 and 2, while most effort comes from bands 3 and 4. If this is referring to a dead sprint, it makes sense. The first several seconds of a sprint you can cover a lot of distance, but will quickly get hit with fatigue. It takes a lot more effort to maintain a sprint after that initial burst, which would explain the higher effort bands occuring later.
4. The y-axis should be standardized so the data can be compared more easily.