post01-Aiden-Rafii

Title:

Learning to manipulate data using the Dplyr package and visualize that data through density curves from GGplot2

Introduction:

Two things I find really cool in this class are the dplyr package and the ggplot density curve function. Dplyr is awesome because you can basically manipulate your data however you want with this package. There are functions like filter, select, and arrange which can cut up your data. Dplyr is a perfect package to fit data frames into a vision that you have. GGplot2 is also an awesome package as after you cut up your data with Dplyr you can display your new data with ggplot2 functions. My favorite function is the density curve. It shows your the density of several variables. Essentially the plot shows you how populated certain variables are. From there you can determine what percentile your certain variable is in and how different or unique your data points are.

Motivation

Growing up I always loved to follow NBA statistics, I'd pore over them for hours at a time. However I didn't always know how to find what I wanted from the data. Dplyr is the perfect way for me to find exatly what I want from team data.I can filter by team, player, even age or experience. Once I do this I can also manipulate player stats using dplyr to find things such as points per minute, player efficiency, etc. Once I cut up this data using Dplyr, I can use the density curve function on ggplot to analyze how good certain players really are. If I want to see the distribution of players who average over 10 points a game, I can cut up the data with dplyr using filter, then I use a density curve to see the distribution of players with a ppg over 10. This is super exciting because I dont have to run a bunch of code to determine the distribution of the data, i can just look at my density curve and see what the 95th percentile for ppg is. I can also color code it as well making it way easier to see my variables and differentiate them.

Origins

In 2012 "on October 28th, 2012, Hadley Wickham started the dplyr project in github as an evolution of his data analysis package plyr (Initially the package was indeed called 'plyr2')". The plyr2 package started out as a package to combine data frames. "The initial functions of dplyr in 2012 included arrange, mutate, summarise, and subset"(http://adolfoalvarez.cl/plumbers-chains-and-famous-painters-the-history-of-the-pipe-operator-in-r/). In 2013 the pipe operator was introduced in dplyr.In September 2014 we received the present version of the Dplyr package which we use in class! Hadley Wickham not only gave us Dplyr in 2014, but 9 years before that Hadley put out GGplot2 in 2005.GGplot2 was finalized in, "25 February 2014, Hadley Wickham formally announced that 'ggplot2 is shifting to maintenance mode'" (https://en.wikipedia.org/wiki/Ggplot2) GGplot2 came out with a bunch of different graphing options, one of which were density curves.

Dplyr Use cases & examples

Before I begin I have to load the 3 packages of readr, dplyr, and ggplot 2

```
setwd("C:/Users/Amir/Desktop/stat133/stat133-hws-fall17/post001")
library(dplyr)
```

```
## 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)
library(readr)
```

Now that I did that I can now use the data that I'll be manipulating. Since we already use NBA data in class I decided to switch it up a bit and use NFL 2016 player stats. Im going to load that data into my directory here.

```
rushing_data  <- read.csv('Data/NFL_rushing_data_2016.csv', stringsAsFactors = FALSE)
receiving_datas <- read.csv('Data/NFL_receiving_data_2016.csv', stringsAsFactors = FALSE)
```

My goal for this post is to find out which NFL team got the least proportion of their total yards receiving from their players other than their wide receivers. In the NFL the 3 most prominent players that catch passes and garner receiving yards are the Wide Receiver (WR), the Running Back(RB), and the Tight End (TE). These 3 are called skill positions. I can reach my goal by using the many functions Dplyr has to offer. I have to group the the receiving yards by WR and team, as well as by team. Then I have to find the proportion of the Wide Receiver yards to the Total Team yards.

```
#here we use the Dplyr functions rename to fix up the starting names we received from the data set. We also use a pipe operator to add another function(select) in order to get the columns we want from our data. I used select here to get rid of the 'rank' column which was absolutely useless to us.
receiving_data <- receiving_datas%>%rename('Name'= X, 'Team' = 'Tm','Games' = 'G', 'Games_Started' = 'GS', 'Receiving_Yards' = 'Yds', 'Receiving_Touchdowns' = 'TD')%>%select(2:17)

#here I use mutate and replace in order to change up our column values. The data was pretty messed up when I first got it, the wide receiver values were both lowercase and upper case so my group function wouldn't work the way I wanted it to. I use replace to change all the lower case values to upper case in order to allow my group function to do the work it needs to do. Mutate allows me to change my columns or even add some.
receiving_data = receiving_data%>%mutate(Pos = replace(Pos, Pos == 'wr', 'WR'))

#Here I use groupby team and position in order to get the sum of the yards by each team by position. I use summarise in conjuction with groupby in order to sum up the yards value of the variable I am grouping by.
receiving_data_by_team_and_position<- receiving_data%>%group_by(Team, Pos)%>%summarise(total_yards = sum(Receiving_Yards))

#Here I want to see not just the grouped value by team and position but also the total value of a team's receiving yards. Therefore I just group_by team and use summarise and sum again. Because the data I pulled from Football reference wasn't that clean I had to remove 3 team values that didnt exist using slice.
receiving_data_by_team <- receiving_data%>%group_by(Team)%>%summarise(total_yards = sum(Receiving_Yards))%>%slice(3:n())

#Here I basically clean up my team_and_position data frame by using filter so I only have my wide receiver yard values. I do this with using filter. I also select Team and Total Yards because I no longer need the position column. I then rename the total_yards column to wide_receiver_yards to avoid any confusion
receiving_data_by_WR<- receiving_data_by_team_and_position%>%filter(Pos == 'WR')%>%select('Team', total_yards)%>%rename('wide_receiver_yards' = 'total_yards')

#Here I use right join in order to join my two tables(team yards, and wide receiver yards) by team. The values from the right all get funneled into the leftmost column value which is team. So now I have a combined table with the total wide receiver yards and total receiving yards for each team.
team_totalyds_WR_yards <- receiving_data_by_WR%>%right_join(receiving_data_by_team)
```

```
## Joining, by = "Team"
```

```
team_totalyds_WR_yards
```

```
## # A tibble: 32 x 3
## # Groups:   Team [?]
##     Team wide_receiver_yards total_yards
##    <chr>               <int>       <int>
##  1   ARI                2377        3942
##  2   ATL                3167        4960
##  3   BAL                2643        4307
##  4   BUF                1686        3061
##  5   CAR                2340        3952
##  6   CHI                2956        4009
##  7   CIN                2735        4206
##  8   CLE                1744        3693
##  9   DAL                1657        3799
## 10   DEN                2423        3680
## # ... with 22 more rows
```

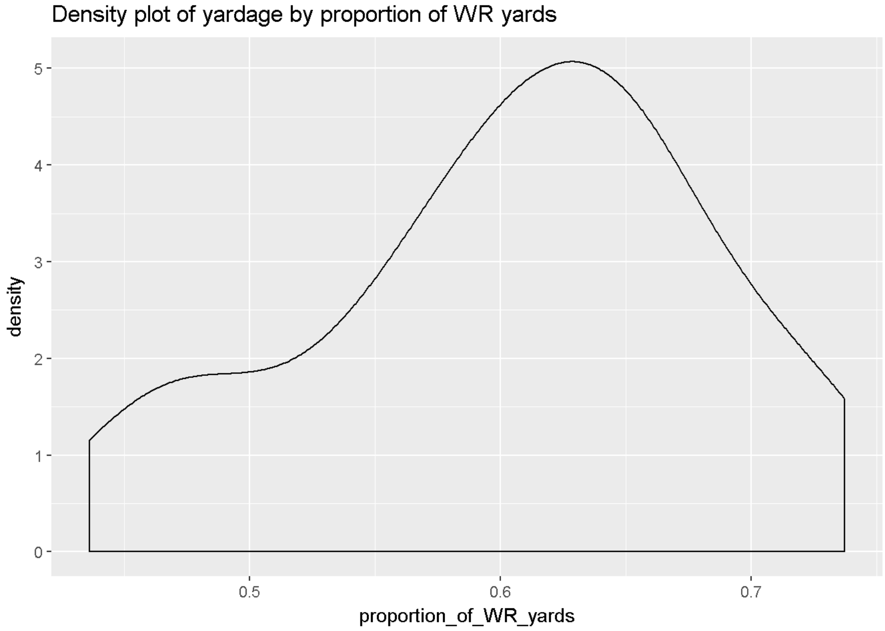
```
#Here I use mutate again in order to add a new column, our final value! I use mutate to add a proportion column which has the proportion of total wide receiver yards per team to their total receiving yards.
team_receiver_proportion <- team_totalyds_WR_yards%>%mutate(proportion_of_WR_yards = wide_receiver_yards/total_yards)

#Lastly, I arrange the table in ascending order using arrange in order to see who got proportionally the least amount of yards from their receivers.
team_receiver_proportion%>%arrange(proportion_of_WR_yards)
```

```
## # A tibble: 32 x 4
## # Groups:   Team [32]
##     Team wide_receiver_yards total_yards proportion_of_WR_yards
##    <chr>               <int>       <int>                  <dbl>
##  1   DAL                1657        3799              0.4361674
##  2   NWE                2029        4414              0.4596738
##  3   CLE                1744        3693              0.4722448
##  4   PHI                1767        3726              0.4742351
##  5   KAN                1938        3898              0.4971780
##  6   SEA                2235        4278              0.5224404
##  7   BUF                1686        3061              0.5508004
##  8   HOU                1888        3418              0.5523698
##  9   SFO                1754        3166              0.5540114
## 10   IND                2556        4491              0.5691383
## # ... with 22 more rows
```

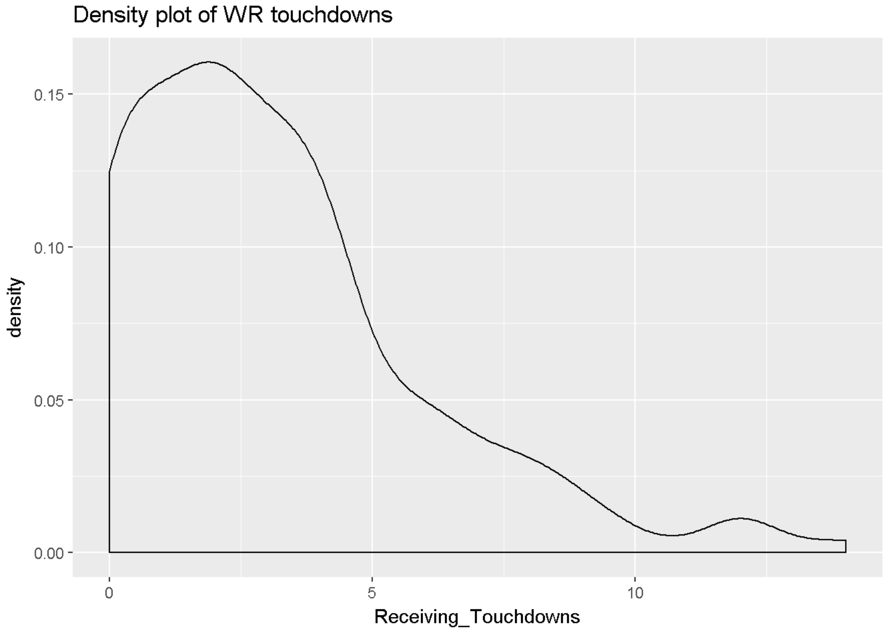
And the answer is the Dallas Cowboys!! When looking at the dallas cowboys roster we notice they have a superstar runningback(Ezekiel Elliott) and a very good tight end in Jason Witten. These two are probably the reason that the Cowboys are number one in this category! Also quarterback and coaching preference can play a huge part in this. Now that we mutated our data a bunch with Dplyr and used some cool tricks we can visualize our new table with GGplot2 functions, most notably our density curve function.

Ggplot2 density curve use cases and examples

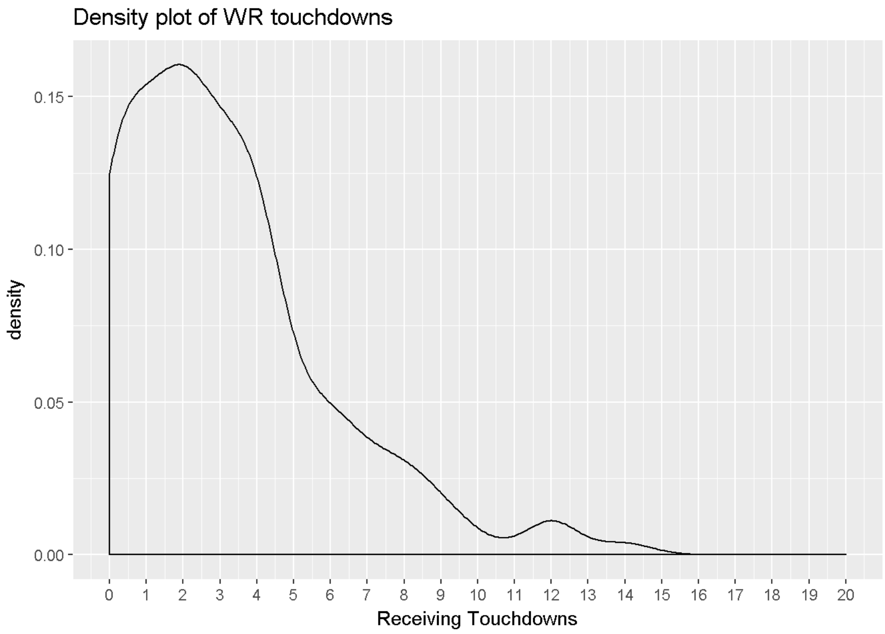
```
#Here is our basic density curve based on the proportion of wide_receiver yards. We can tell that the highest number of teams lie between the proportions of .6 and .65 of their total receiving yards made by their wide receiver.
ggplot(data = team_receiver_proportion, aes(x = proportion_of_WR_yards))+geom_density()+ggtitle("Density plot of yardage by proportion of WR yards")
```



```
#Now we do receiving touchdowns by player in order to see the amount of touchdowns most players get. We see here that actually most receivers get between 3-5 receiving touchdowns.
receiving_TD_by_WR<- receiving_data%>%filter(Pos == 'WR')
graph <- ggplot(data = receiving_TD_by_WR, aes(x = Receiving_Touchdowns))+geom_density()+ggtitle("Density plot of WR touchdowns")
graph
```

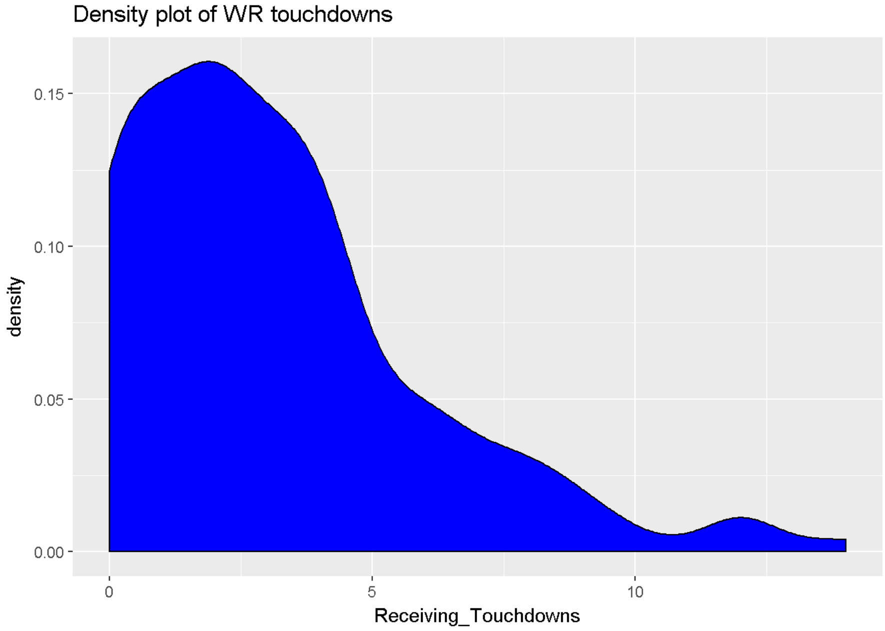


```
#We can also change up the axis ticks to get a more specific answer. We do this by changing breaks and limits. Now we see that most wide receivers get closer to 4 touchdowns. Limit is your maximum X value and you can use your breaks to determine by what values you'd like to increase x on your graph
graph2 <- graph + scale_x_continuous(name = "Receiving Touchdowns",
                              breaks = seq(0, 20, 1),
                              limits=c(0, 20))
graph2
```

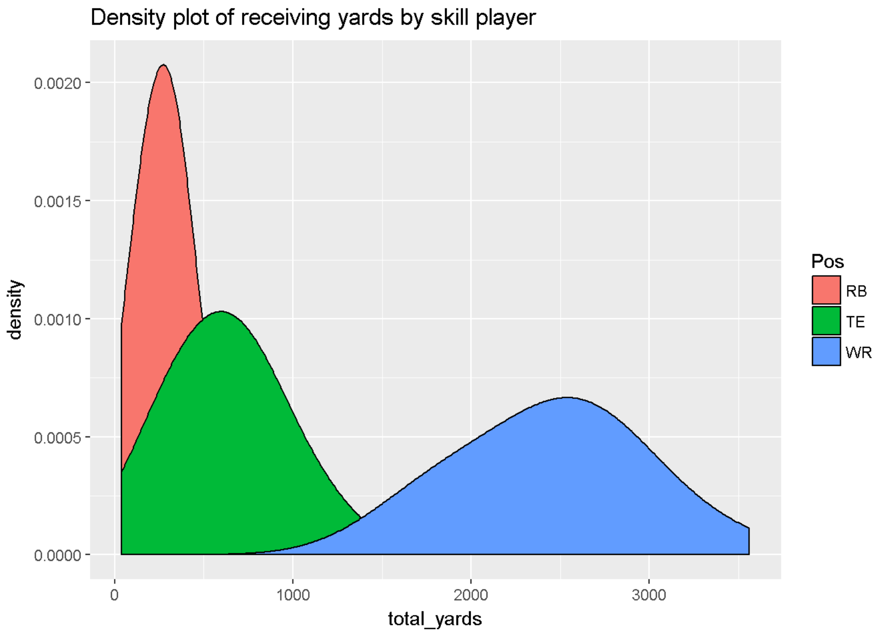


```
#Another cool thing about density curves is you can also change the color of the line and the color of the filling
inside the density curve, this is done by using fill, and line inside the geom_density piece of code.
ggplot(data = receiving_TD_by_WR, aes(x = Receiving_Touchdowns))+geom_density(fill = 'blue', line = 'gold')+ggtitl
e("Density plot of WR touchdowns")
```

```
## Warning: Ignoring unknown parameters: line
```



```
#Here we want to make 3 density curves on one graph to compare receiving yards by postion so we use filter again t
o get the 3 skil positions we want
receiving_data_by_team_and_skill <- receiving_data_by_team_and_position%>%filter(Pos == 'RB'|Pos ==  'WR'|Pos == '
TE')

#With our density curves we can use group to create a seperate density curve for each positional group we want and
we can make it so that each one is a specific color as well. We can now clearly see that on average WR will have m
ore receiving yards, then its tight ends, then its running backs. We can also smooth out our density curves by usi
ng adjust! The higher value you go the lower density you'll have on your y axis.
ggplot(data=receiving_data_by_team_and_skill,aes(x=total_yards, group=Pos, fill= Pos)) +
    geom_density(adjust=2)+ggtitle("Density plot of receiving yards by skill player")
```



Conclusion

In Conclusion we've learned how to cut up and manipulate our data frames using dplyr with slice, select, and mutate. We've even learned how to clean up our data if we're given faulty values like when we used replace and rename. We've also learned to find the portions of data that we want with filter. And we learned how to use mutate and replace in conjuction with each other. In regards to our density curves we have now learned what they're used for, how to use and make them in ggplot as well as how to put multiple density curves on one axis and how to label, title, and color them! All these techniques can be used for things such as manipulating weather data, sports, and even monetary accounts!

References

References I used to learn more info and put together this post:

My data came from https://www.pro-football-reference.com/years/2016/receiving.htm I learned how to use mutate and replace together with https://stackoverflow.com/questions/28013850/change-value-of-variable-with-dplyr I learned how to use slice in regards to N with https://stackoverflow.com/questions/42237955/remove-the-first-n-rows-from-each-factor-level-in-an-r-data-frame/42238006 I learned how to join tables from the right side with http://dplyr.tidyverse.org/reference/join.html I learned how to make multiple density curves from http://www.r-graph-gallery.com/135-stacked-density-graph/ I learned how to create a basic density curve by watching https://www.youtube.com/watch?v=BmyS3Z5mfxE I learned about the history of Dplyr by reading http://adolfoalvarez.cl/plumbers-chains-and-famous-painters-the-history-of-the-pipe-operator-in-r/ I learned about the history of ggplot by reading https://en.wikipedia.org/wiki/Ggplot2