2019 NFL Play by Play Data Analysis

Michael Flora

5/28/2020

Introduction and Goals:

In this project, I will be exploring, visualizing, analyzing, and modeling 2019 NFL play by play data that is publicly available online. The goal of this project is to learn something and gain interesting insights that would not have been possible without data. I hope to come to a conclusion at the end that is useful and applicable for a wide variety of people.

Here, I clear my workspace, load some useful packages, and read in the data set.

```
rm(list = ls())
library(tidyverse)
```

```
## -- Attaching packages ---------------------------------------------------------------------------- tidy
```

```
## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.1     v dplyr   1.0.0
## v tidyr   1.1.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
```

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

```
library(nflfastR)
```

```
## Loading required package: nflscrapR
```

```
## Loading required package: nnet
```

```
## Loading required package: magrittr
```

```
##
## Attaching package: 'magrittr'
```

```
## The following object is masked from 'package:purrr':
##
##     set_names
```

```
## The following object is masked from 'package:tidyr':
##
##     extract
```

```
library(ggimage)

pbp = readRDS(url('https://raw.githubusercontent.com/guga31bb/nflfastR-data/master/data/play_by_play_20
```

Here, we will look at the first 5 rows of the data set to see what we're working with. Next, we'll at the strucutre, so that we can get a better understanding of the variables used.

```r
head(pbp)
```

```
## # A tibble: 6 x 340
##   play_id game_id old_game_id home_team away_team season_type  week posteam
##     <dbl> <chr>   <chr>       <chr>     <chr>     <chr>       <int> <chr>
## 1       1 2019_0~ 2019090804  MIN       ATL       REG             1 <NA>
## 2      36 2019_0~ 2019090804  MIN       ATL       REG             1 ATL
## 3      51 2019_0~ 2019090804  MIN       ATL       REG             1 ATL
## 4      79 2019_0~ 2019090804  MIN       ATL       REG             1 ATL
## 5     100 2019_0~ 2019090804  MIN       ATL       REG             1 ATL
## 6     121 2019_0~ 2019090804  MIN       ATL       REG             1 ATL
## # ... with 332 more variables: posteam_type <chr>, defteam <chr>,
## #   side_of_field <chr>, yardline_100 <dbl>, game_date <chr>,
## #   quarter_seconds_remaining <dbl>, half_seconds_remaining <dbl>,
## #   game_seconds_remaining <dbl>, game_half <chr>, quarter_end <dbl>,
## #   drive <dbl>, sp <dbl>, qtr <dbl>, down <dbl>, goal_to_go <dbl>, time <chr>,
## #   yrdln <chr>, ydstogo <dbl>, ydsnet <dbl>, desc <chr>, play_type <chr>,
## #   yards_gained <dbl>, shotgun <dbl>, no_huddle <dbl>, qb_dropback <dbl>,
## #   qb_kneel <dbl>, qb_spike <dbl>, qb_scramble <dbl>, pass_length <chr>,
## #   pass_location <chr>, air_yards <dbl>, yards_after_catch <dbl>,
## #   run_location <chr>, run_gap <chr>, field_goal_result <chr>,
## #   kick_distance <dbl>, extra_point_result <chr>, two_point_conv_result <chr>,
## #   home_timeouts_remaining <dbl>, away_timeouts_remaining <dbl>,
## #   timeout <dbl>, timeout_team <chr>, td_team <chr>,
## #   posteam_timeouts_remaining <dbl>, defteam_timeouts_remaining <dbl>,
## #   total_home_score <dbl>, total_away_score <dbl>, posteam_score <dbl>,
## #   defteam_score <dbl>, score_differential <dbl>, posteam_score_post <dbl>,
## #   defteam_score_post <dbl>, score_differential_post <dbl>,
## #   no_score_prob <dbl>, opp_fg_prob <dbl>, opp_safety_prob <dbl>,
## #   opp_td_prob <dbl>, fg_prob <dbl>, safety_prob <dbl>, td_prob <dbl>,
## #   extra_point_prob <dbl>, two_point_conversion_prob <dbl>, ep <dbl>,
## #   epa <dbl>, total_home_epa <dbl>, total_away_epa <dbl>,
## #   total_home_rush_epa <dbl>, total_away_rush_epa <dbl>,
## #   total_home_pass_epa <dbl>, total_away_pass_epa <dbl>, air_epa <dbl>,
## #   yac_epa <dbl>, comp_air_epa <dbl>, comp_yac_epa <dbl>,
## #   total_home_comp_air_epa <dbl>, total_away_comp_air_epa <dbl>,
## #   total_home_comp_yac_epa <dbl>, total_away_comp_yac_epa <dbl>,
## #   total_home_raw_air_epa <dbl>, total_away_raw_air_epa <dbl>,
## #   total_home_raw_yac_epa <dbl>, total_away_raw_yac_epa <dbl>, wp <dbl>,
## #   def_wp <dbl>, home_wp <dbl>, away_wp <dbl>, wpa <dbl>, home_wp_post <dbl>,
## #   away_wp_post <dbl>, vegas_wp <dbl>, vegas_home_wp <dbl>,
## #   total_home_rush_wpa <dbl>, total_away_rush_wpa <dbl>,
## #   total_home_pass_wpa <dbl>, total_away_pass_wpa <dbl>, air_wpa <dbl>,
## #   yac_wpa <dbl>, comp_air_wpa <dbl>, comp_yac_wpa <dbl>,
## #   total_home_comp_air_wpa <dbl>, ...
```

Data Manipulation:

Here I will manipulate the data by creating some new columns that I plan on using later in the analysis. First, I convert the play type variable to a factor, so that it is easier to use. I create a column called good_play. In this data set, a win probability function is included, which I will use. I classify a good play as 1 when a team's win probability increases or stays the same because of a particular play and 0 when the win probability decreases after the play. I create a column that is 1 if the play type is a passing play and 0 if it is anything

else. I create a column called yardage that has 3 possible values: long, medium, and short. If there are 8 yards to go until the 1st down marker, then yardage is long. If there's 4-7 yards to go, then medium yardage. If less than 4 yards to go, then short yardage.

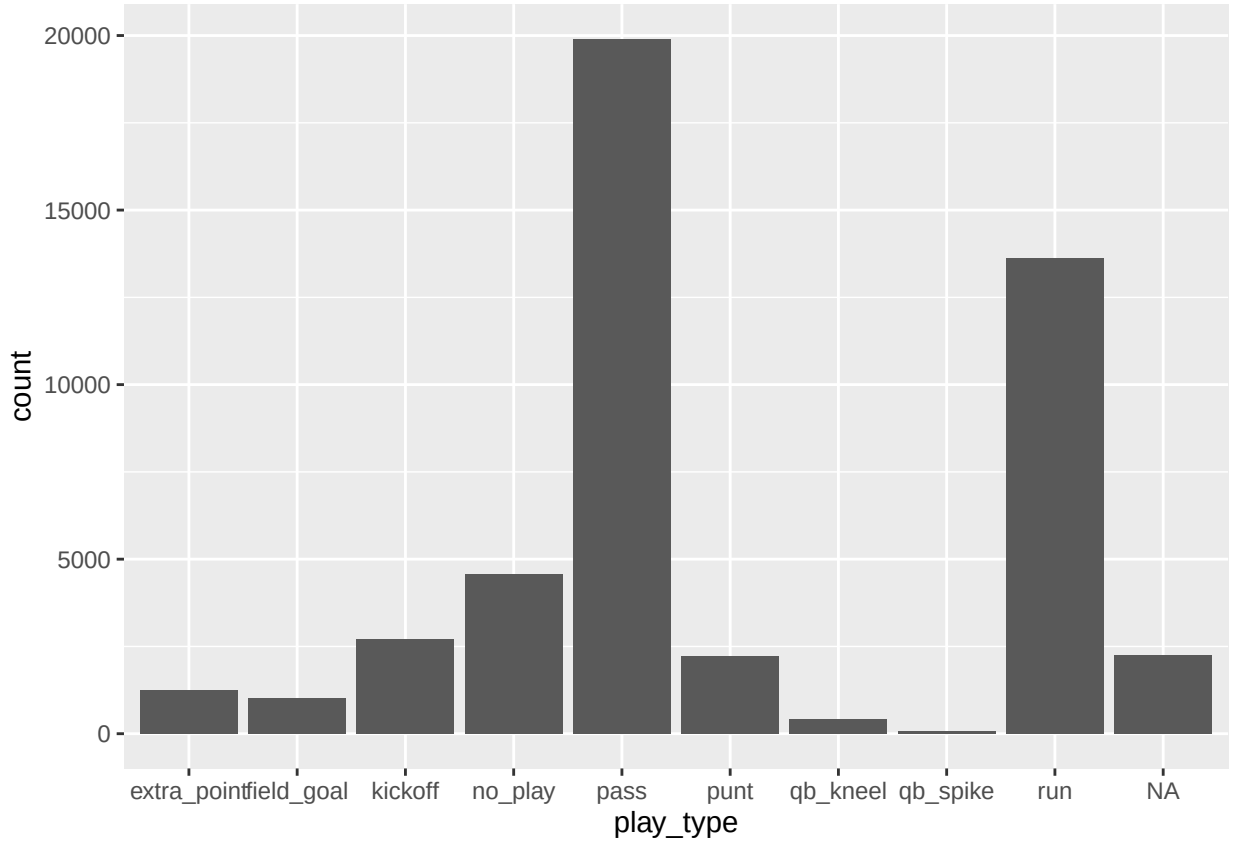
```
pbp$play_type = as.factor(pbp$play_type)

pbp$good_play = ifelse(pbp$wpa >= 0, 1, 0)

pbp$pass = ifelse(pbp$play_type == "pass", 1, 0)

pbp$yardage = case_when(pbp$ydstogo >= 8 ~ "long",
                        pbp$ydstogo >= 4 ~ "medium",
                        TRUE ~"short")
```

Data Visualization and Exploration:

In this, I will create some different types of visualizations to see if I can find anything meaningful or interesting. These graphs can provide to be useful when communicating with those who lack the jargon that goes along with analytics and statistics.

Noteworthy: Quarter 5 means Overtime

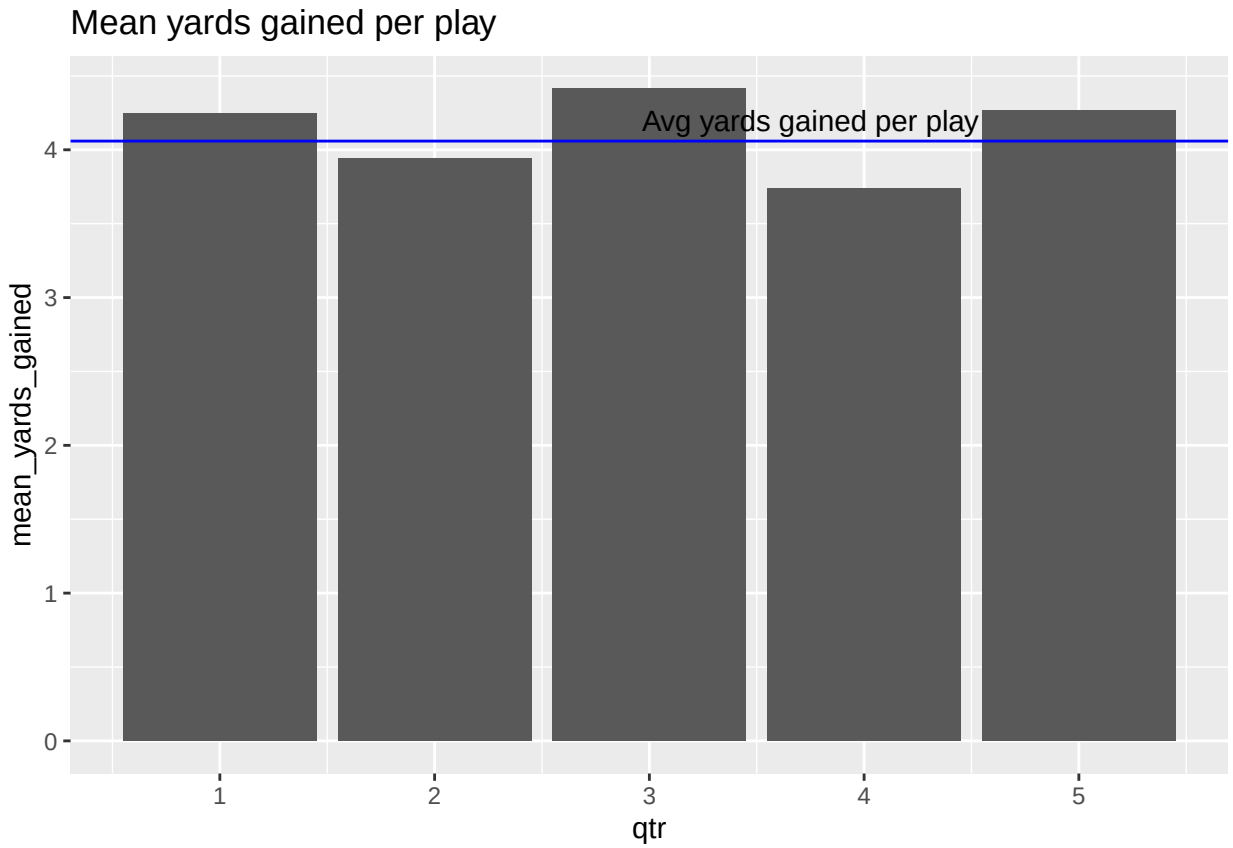
```
## Bar Chart that counts number of plays of different play types
pbp %>%
  ggplot(aes(x= play_type)) + geom_bar()
```




```
## Graph of average yards gained per play in each quarter
pbp %>% group_by(qtr) %>% summarise(mean_yards_gained = mean(yards_gained, na.rm = TRUE)) %>%
```

```
  ggplot(aes(x= qtr, y = mean_yards_gained)) + geom_bar(stat = "identity") +
  labs(title = "Mean yards gained per play") +
  geom_hline(yintercept = mean(pbp$yards_gained, na.rm = TRUE), color = "blue") +
  annotate("text", label = "Avg yards gained per play", x = 3.75, y = 4.2)
```

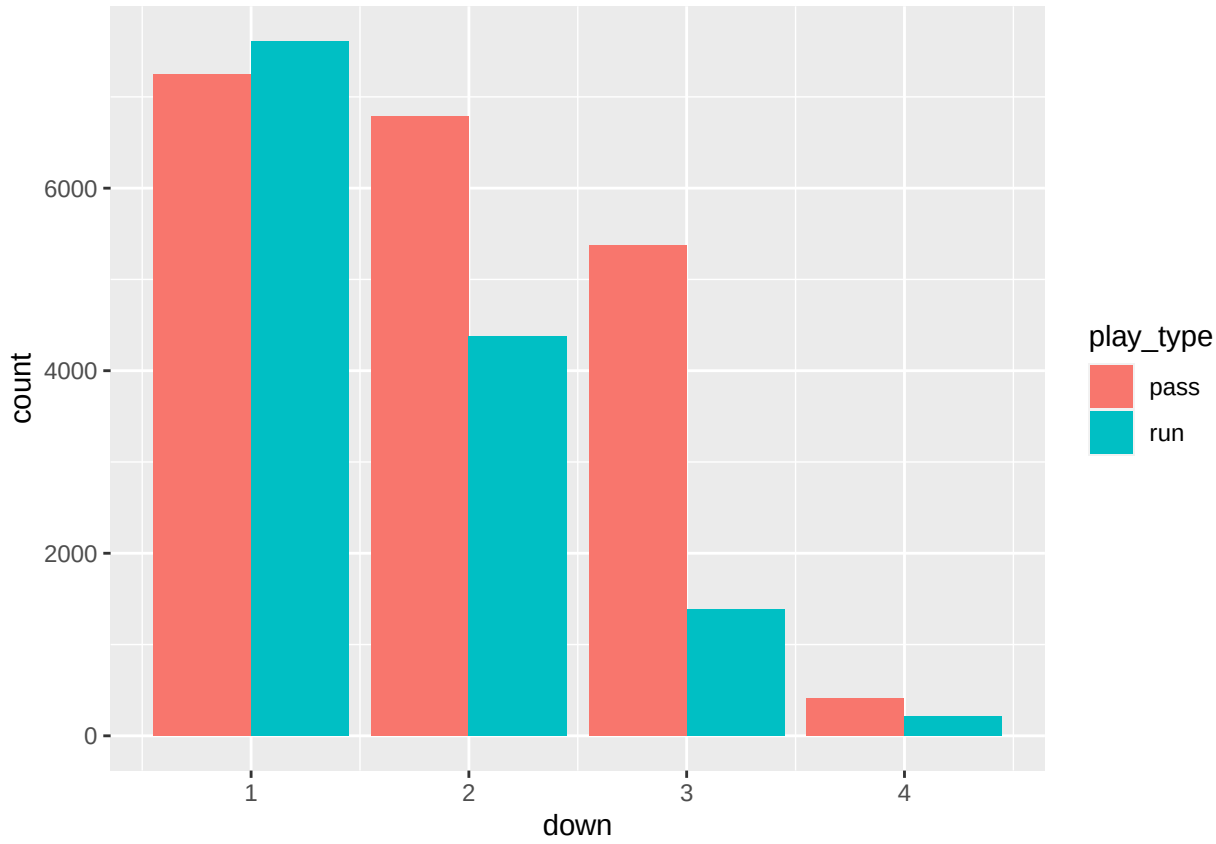
```
## `summarise()` ungrouping output (override with `.groups` argument)
```



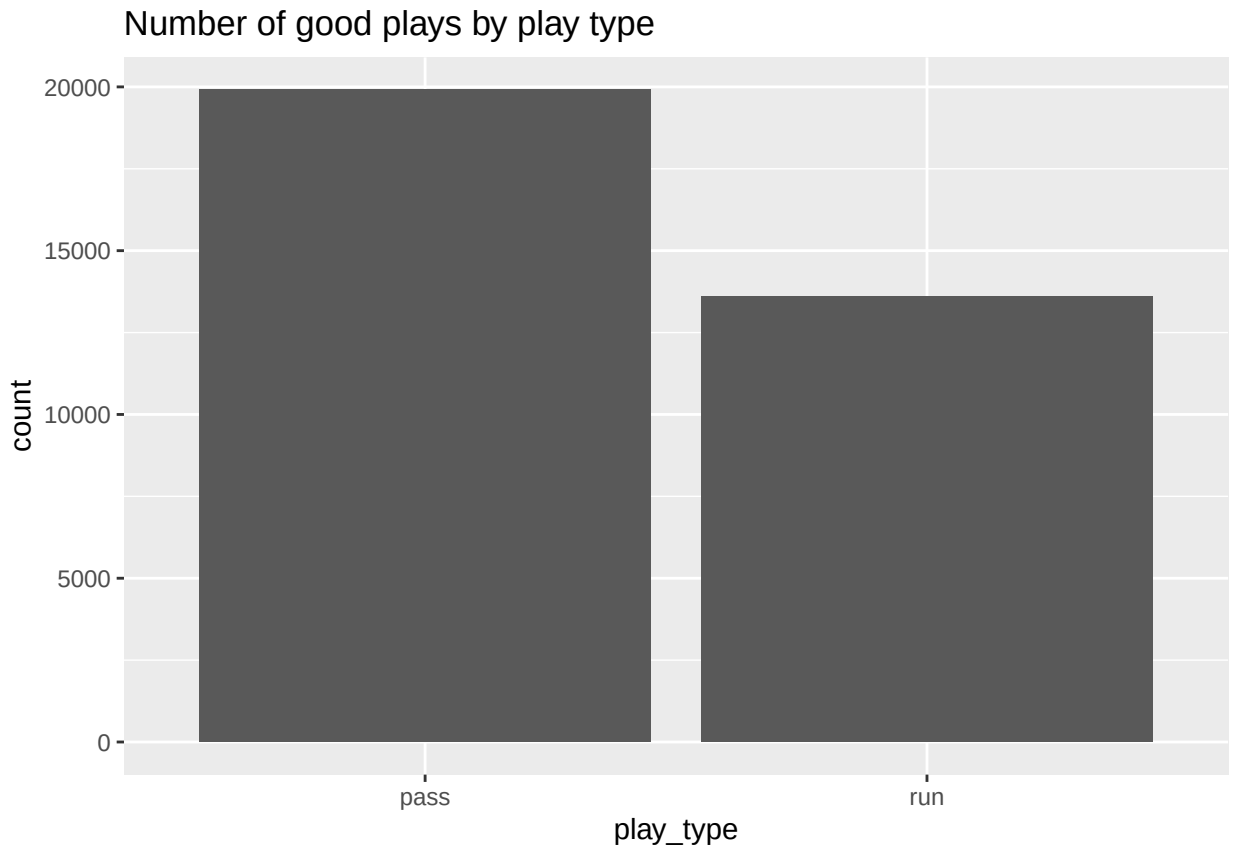

```
## Distribution of play types by quarter
pbp %>%
  ggplot(aes(x= qtr, fill = play_type)) + geom_bar() + facet_wrap(~play_type)
```


```
## Graph of pass vs run plays broken down by down
pbp %>%
  filter(play_type == "pass" | play_type == "run") %>%
  ggplot(aes(x= down, fill = play_type)) + geom_bar(position="dodge")
```

```
## Warning: Removed 116 rows containing non-finite values (stat_count).
```

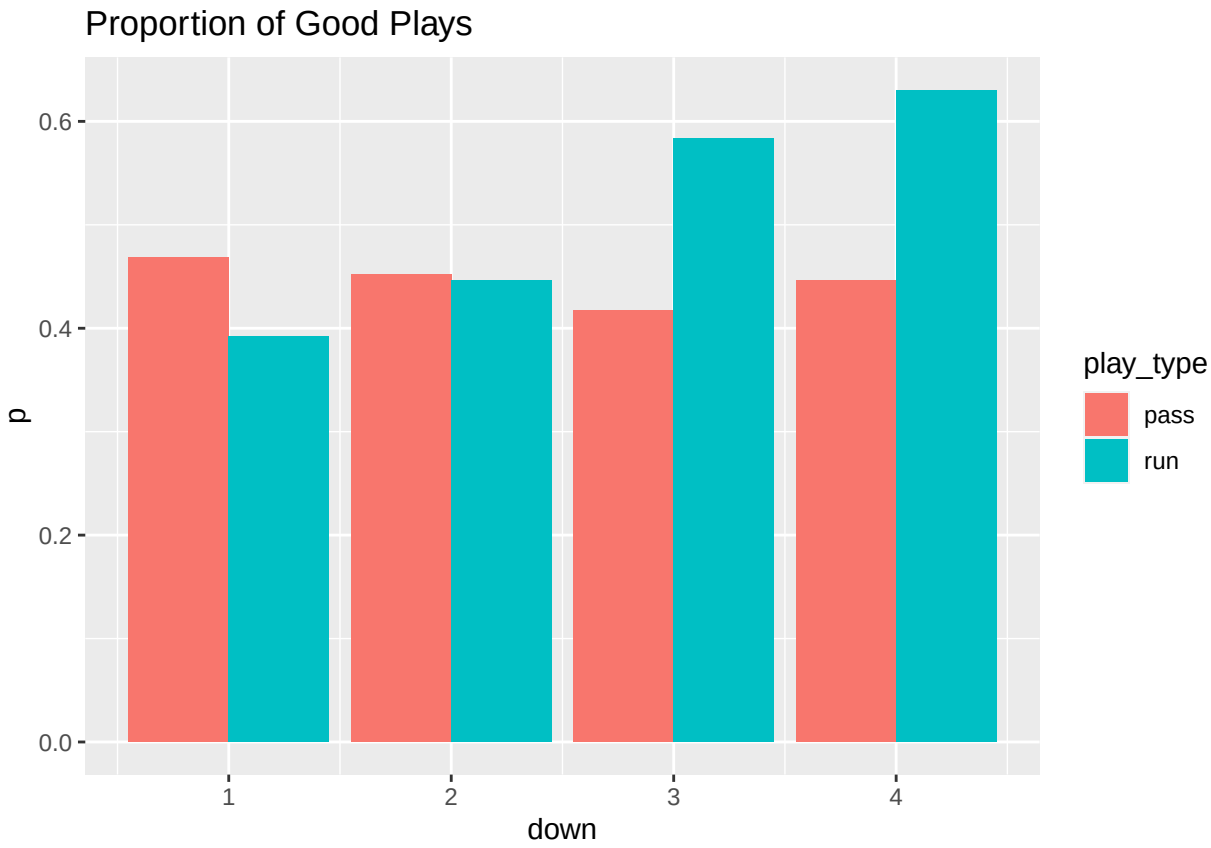



```
## Graph showing number of good plays broken down by pass vs run plays
pbp %>%
  filter(play_type == "pass" | play_type == "run") %>%
  ggplot(aes(x= play_type, fill = good_play)) + geom_bar(position="dodge") +
  labs(title = "Number of good plays by play type")
```



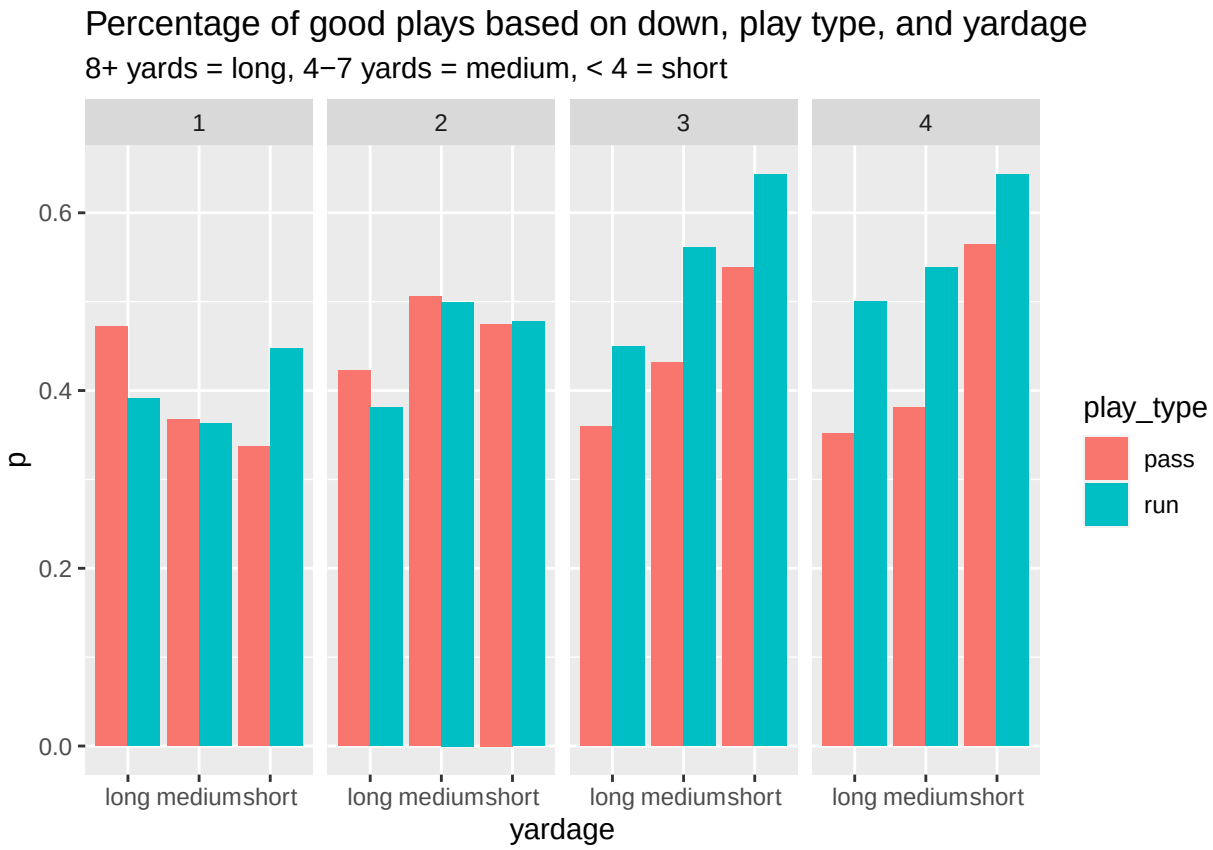

```
## Graph of proportion of good plays grouped by down & play type
pbp %>%
  filter(play_type == "pass" | play_type == "run", is.na(good_play) == FALSE, down != 5) %>%
  group_by(down, play_type) %>%
  summarise(p = sum(good_play) / length(good_play)) %>%
  ggplot(aes(x = down, y = p, fill = play_type)) +
  geom_bar(stat= "identity", position = "dodge") +
  labs(title = "Proportion of Good Plays")
```

```
## `summarise()` regrouping output by 'down' (override with `.groups` argument)
```




```
## Graph of proportion of good plays grouped by down, play type, and yardage
pbp %>%
  filter(play_type == "pass" | play_type == "run", is.na(good_play) == FALSE, down != 5) %>%
  group_by(yardage, play_type, down) %>%
  summarise(p = sum(good_play) / length(good_play)) %>%
  ggplot(aes(x= yardage, y = p, fill = play_type)) +
  geom_bar(stat = "identity", position = "dodge") +
  facet_grid(~down) +
  labs(title = "Percentage of good plays based on down, play type, and yardage",
       subtitle = "8+ yards = long, 4-7 yards = medium, < 4 = short")
```

```
## `summarise()` regrouping output by 'yardage', 'play_type' (override with `.groups` argument)
```



Insights from Data Visualizations:

-Pass play is most common play -Average yards per play is higher in the 1st and 3rd quarter of games -Larger number of pass plays happen in the 2nd and 4th quarters -Number of run plays increases slighlty in the 4th quarter. I believe this is due to the fact that teams that have a lead at the end of games will run the ball to burn the clock -Larger number of fields goals occur in the 2nd quarter compared to the other quarters -More run than pass plays occur on 1st down, but more pass than run plays occur on 2nd, 3rd, and 4th down. I believe this happens because teams that don't get a decent amount of yards on 1st down are forced to "catch up" and pass plays give the best shot at accomplishing that -There is a significantly more larger number of "good" pass plays than "good" run plays according to my definition of good plays -Higher proportion of good plays on 1st down are pass plays. Pass and run plays have similar proportion of good plays on 2nd down. Higher proportion of good plays on 3rd and 4th down are run plays, which I found to be interesting. -Based on the last visualization, we can see the proportion of good plays based on play type, yardage, and down. This could be very useful in identifying what type of play to call in certain scenarios. -Higher proportion of good plays on 1st down and 10 are pass plays.

Hypothesis Testing:

After exploring the data a bit, I became curious about whether there is a difference in the proportion of passing plays between the 1st and 2nd half of the game. To do this, I will complete a paired t-test that measures the difference of each team's 1st and 2nd half proportion.

Null Hypothesis: HO: D = 0; Half has no effect on the proportion of passing plays Alternative Hypothesis: Ha: D != 0; Half does have an effect on the proportion of passing plays

```
team_quarter_pct_passing_first_half = pbp %>%
  filter(qtr == 1 | qtr == 2) %>%
  group_by(posteam) %>%
  filter(play_type != "no_play", is.na(play_type) != TRUE) %>%
  summarise(pct_passing = sum(pass) / length(pass))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
team_quarter_pct_passing_second_half = pbp %>%
  filter(qtr == 3 | qtr == 4) %>%
  group_by(posteam) %>%
  filter(play_type != "no_play", is.na(play_type) != TRUE) %>%
  summarise(pct_passing = sum(pass) / length(pass))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
t.test(team_quarter_pct_passing_first_half$pct_passing, team_quarter_pct_passing_second_half$pct_passing
       paired = TRUE, alternative = "two.sided")
```

```
##
##  Paired t-test
##
## data:  team_quarter_pct_passing_first_half$pct_passing and team_quarter_pct_passing_second_half$pct_p
## t = 0.039321, df = 31, p-value = 0.9689
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.01793345  0.01863855
## sample estimates:
## mean of the differences
##            0.0003525469
```

Based on the results of the paired t test, there is no evidence to suggest that the proportion of passing plays changes based on which half it is. The p-value os 0.9695 is extremely high. The mean difference was 0.0003456508, which is very close to 0. The 95% Confidence interval was (-0.01793267, 0.01862397). Based on these, I believe that it's safe to say that the true mean difference is extremely close to 0.

Classification Model:

I plan to attempt to predict if a play is a run or pass play based on different variables in the data set. I believe this could be a very useful tool for a defensive coordinator to utilize. If successful, a coordinator would be better equipped to call defensive audibles if analytics teams were able to quickly make predictions based on certain game scenarios.

```
## Building train and test data sets

classifier_data_set = pbp %>% filter(play_type == "pass" | play_type == "run", is.na(down) != TRUE)

n = length(classifier_data_set$play_id)

train_rows = sample(n, n/2)

train = classifier_data_set[train_rows,]
test = classifier_data_set[-train_rows,]

## Training the classifier
```

```r
## pass is 1 if pass play, 0 if not
log_m1 = glm(pass ~ wp + ydstogo + down + game_seconds_remaining, data = train, family = "binomial")
summary(log_m1)
```

```
##
## Call:
## glm(formula = pass ~ wp + ydstogo + down + game_seconds_remaining,
##     family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.0067  -1.1168   0.6312   0.9933   1.9353
##
## Coefficients:
##                            Estimate Std. Error z value Pr(>|z|)
## (Intercept)            -1.293e+00  8.555e-02  -15.12  < 2e-16 ***
## wp                     -1.332e+00  5.748e-02  -23.18  < 2e-16 ***
## ydstogo                 1.198e-01  5.350e-03   22.39  < 2e-16 ***
## down                    8.177e-01  2.480e-02   32.98  < 2e-16 ***
## game_seconds_remaining -5.025e-05  1.611e-05   -3.12  0.00181 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 22542  on 16697  degrees of freedom
## Residual deviance: 20494  on 16693  degrees of freedom
## AIC: 20504
##
## Number of Fisher Scoring iterations: 4
```

```r
log_probs1 = predict(log_m1, newdata = test, type = "response")
log_p1 = ifelse(log_probs1 > 0.5, 1, 0)
c_matrix1 = table(log_p1 == test$pass)
c_matrix1
```

```
##
## FALSE  TRUE
##  5591 11108
```

```r
log_m2 = glm(pass ~ wp + ydstogo + down, data = train, family = "binomial")
summary(log_m2)
```

```
##
## Call:
## glm(formula = pass ~ wp + ydstogo + down, family = "binomial",
##     data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.0219  -1.1246   0.6366   0.9938   1.9295
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.380837   0.080747  -17.10   <2e-16 ***
```

```
## wp          -1.340216   0.057288  -23.39   <2e-16 ***
## ydstogo      0.119712   0.005348   22.39   <2e-16 ***
## down         0.819815   0.024779   33.09   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 22542  on 16697  degrees of freedom
## Residual deviance: 20503  on 16694  degrees of freedom
## AIC: 20511
##
## Number of Fisher Scoring iterations: 4
```

```
log_probs2 = predict(log_m2, newdata = test, type = "response")
log_p2 = ifelse(log_probs2 > 0.5, 1, 0)
c_matrix2 = table(log_p2 == test$pass)
c_matrix2
```

```
##
## FALSE  TRUE
##  5595 11104
```

```
log_m3 = glm(pass ~ down + ydstogo + td_prob + fg_prob + posteam,
             data = train, family = "binomial")
summary(log_m3)
```

```
##
## Call:
## glm(formula = pass ~ down + ydstogo + td_prob + fg_prob + posteam,
##     family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.6860  -1.1600   0.6601   1.0134   2.2581
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.835221   0.149042  -5.604 2.10e-08 ***
## down         0.658993   0.026821  24.570  < 2e-16 ***
## ydstogo      0.086561   0.005704  15.174  < 2e-16 ***
## td_prob     -2.069539   0.145859 -14.189  < 2e-16 ***
## fg_prob      0.904501   0.188814   4.790 1.66e-06 ***
## posteamATL   0.343997   0.139647   2.463 0.013765 *
## posteamBAL  -0.595222   0.132999  -4.475 7.63e-06 ***
## posteamBUF  -0.370265   0.134683  -2.749 0.005975 **
## posteamCAR   0.024578   0.136086   0.181 0.856677
## posteamCHI  -0.105056   0.138582  -0.758 0.448403
## posteamCIN  -0.013933   0.136872  -0.102 0.918916
## posteamCLE  -0.134093   0.139222  -0.963 0.335465
## posteamDAL  -0.019669   0.135508  -0.145 0.884590
## posteamDEN  -0.118727   0.141107  -0.841 0.400126
## posteamDET  -0.118601   0.137300  -0.864 0.387694
## posteamGB   -0.065219   0.134536  -0.485 0.627842
## posteamHOU  -0.148972   0.133772  -1.114 0.265438
```

```
## posteamIND  -0.371050   0.138968  -2.670 0.007584 **
## posteamJAX   0.072096   0.137559   0.524 0.600204
## posteamKC    0.178409   0.134421   1.327 0.184431
## posteamLA   -0.020602   0.137460  -0.150 0.880862
## posteamLAC   0.132007   0.138228   0.955 0.339580
## posteamLV   -0.242642   0.137513  -1.764 0.077648 .
## posteamMIA   0.247042   0.140961   1.753 0.079678 .
## posteamMIN  -0.398432   0.134277  -2.967 0.003005 **
## posteamNE   -0.027271   0.134139  -0.203 0.838898
## posteamNO    0.066942   0.137578   0.487 0.626560
## posteamNYG   0.086063   0.137793   0.625 0.532247
## posteamNYJ  -0.185440   0.137672  -1.347 0.177989
## posteamPHI  -0.131662   0.133499  -0.986 0.324017
## posteamPIT  -0.260340   0.139947  -1.860 0.062847 .
## posteamSEA  -0.263185   0.132008  -1.994 0.046183 *
## posteamSF   -0.453104   0.132785  -3.412 0.000644 ***
## posteamTB    0.071447   0.137939   0.518 0.604486
## posteamTEN  -0.520920   0.134845  -3.863 0.000112 ***
## posteamWAS  -0.167498   0.143591  -1.166 0.243415
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 22542  on 16697  degrees of freedom
## Residual deviance: 20688  on 16662  degrees of freedom
## AIC: 20760
##
## Number of Fisher Scoring iterations: 4
```

```
log_probs3 = predict(log_m3, newdata = test, type = "response")
log_p3 = ifelse(log_probs3 > 0.5, 1, 0)
c_matrix3 = table(log_p3 == test$pass)
c_matrix3
```

```
##
## FALSE  TRUE
##  5902 10797
```

```
best_logistic_model_prop = 11095 / (11095 + 5565)
best_logistic_model_prop
```

```
## [1] 0.6659664
```

```
library("pROC")
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'
```

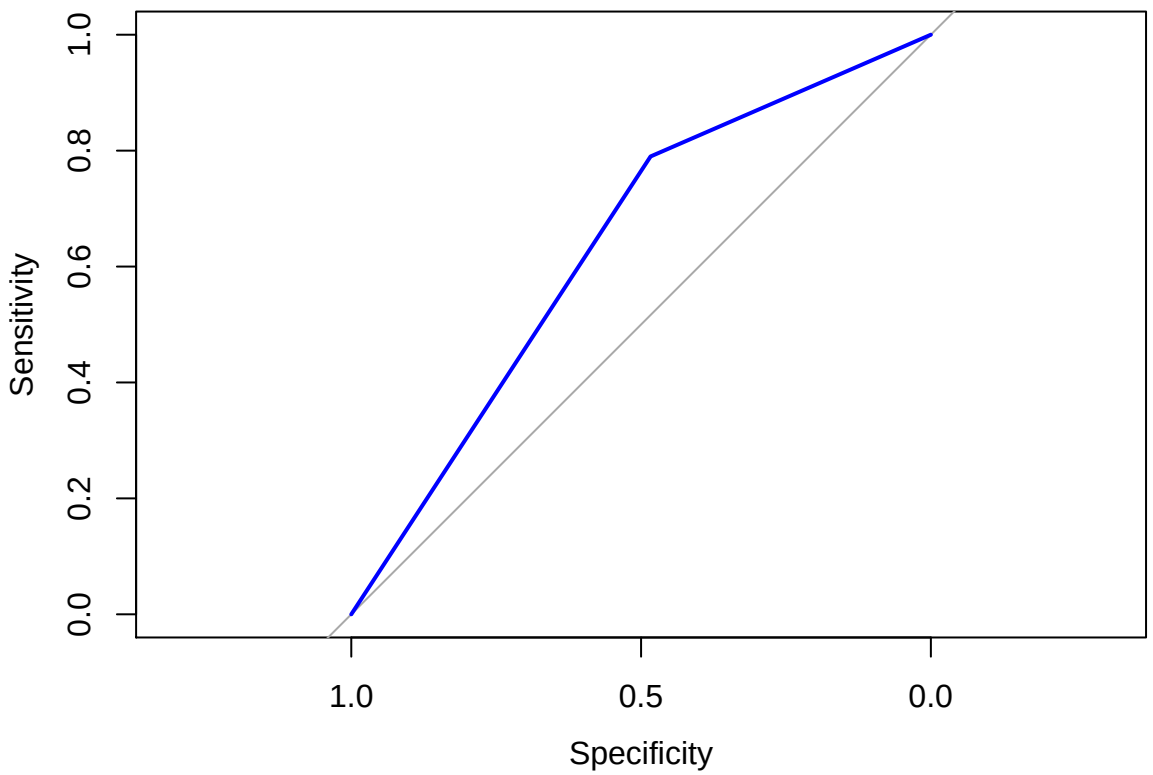
```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
ROC = roc(test$pass, log_p2)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
plot(ROC, col = "blue")
```




```
auc(ROC)
```

```
## Area under the curve: 0.6368
```

The ROC curve graphs the true positive rate versus the false positive rate. A random machine learning algorithm would be be the line, y = x. The area under the curve would be 0.5 My machine learning algoritm has an area under the curve of 0.6378. That is not a very strong measure. This means my algorithm does not do a very good job at making predictions.

It would be interesting to see how my algorithm would do if there were other variables used that I do not have access to. I believe it would be beneficial for NFL teams to build similar machine learning algorithms that are trained on a data set that is more specific to the type of players and team they'd be facing. The algorithm might be better at predicting against certain QBs, for example like Patrick Mahomes or Lamar Jackson. Defensive coordinators could find this very useful as they'd be better equipped to make better decisions regarding defensive scheme, substitutions, and plays if they have can make good predictions of the next play type.

```
first_down = pbp %>% filter(down == 1) %>%
  filter(play_type == "pass" | play_type == "run") %>%
  mutate(pass = ifelse(play_type == "pass", 1, 0))
second_down = pbp %>% filter(down == 2) %>%
  filter(play_type == "pass" | play_type == "run") %>%
  mutate(pass = ifelse(play_type == "pass", 1, 0))
third_down = pbp %>% filter(down == 3) %>%
  filter(play_type == "pass" | play_type == "run") %>%
  mutate(pass = ifelse(play_type == "pass", 1, 0))
fourth_down = pbp %>% filter(down == 4) %>%
  filter(play_type == "pass" | play_type == "run" | play_type == "field_goal" | play_type == "punt") %>%
```

```
  mutate(pass = ifelse(play_type == "pass", 1, 0))

first_down$play_type = as.factor(first_down$play_type)
second_down$play_type = as.factor(second_down$play_type)
third_down$play_type = as.factor(third_down$play_type)
fourth_down$play_type = as.factor(fourth_down$play_type)


first_down_pct_passing = first_down %>%
  group_by(posteam) %>%
  summarize(pct_passing = sum(pass) / length(pass)) %>%
  mutate("down" = 1)
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
second_down_pct_passing = second_down %>%
  group_by(posteam) %>%
  summarize(pct_passing = sum(pass)/ length(pass)) %>%
  mutate("down" = 2)
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
third_down_pct_passing = third_down %>%
  group_by(posteam) %>%
  summarize(pct_passing = sum(pass)/ length(pass)) %>%
  mutate("down" = 3)
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
fourth_down_pct_passing = fourth_down %>%
  group_by(posteam) %>%
  summarize(pct_passing = sum(pass)/ length(pass)) %>%
  mutate("down" = 4)
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
fourth_down_pct_running = fourth_down %>%
  mutate("run" = ifelse(play_type == "run",1,0)) %>%
  group_by(posteam) %>%
  summarize(pct_running = sum(run)/ length(run)) %>%
  mutate("down" = 4)
```

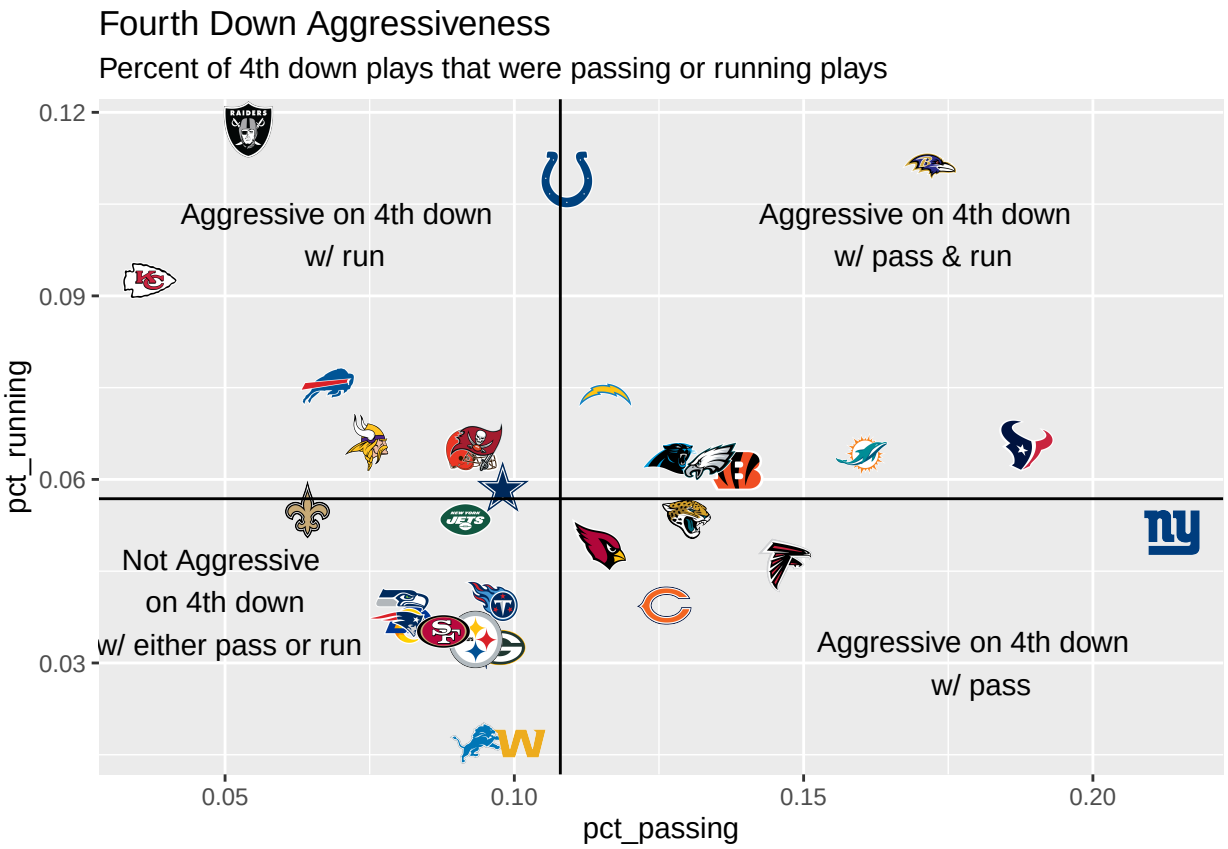
```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
fourth_down_aggressiveness = merge(fourth_down_pct_passing, fourth_down_pct_running, by = "posteam")
head(fourth_down_aggressiveness)
```

```
##   posteam pct_passing down.x pct_running down.y
## 1     ARI  0.11475410      4  0.04918033      4
## 2     ATL  0.14678899      4  0.04587156      4
## 3     BAL  0.17171717      4  0.11111111      4
## 4     BUF  0.06766917      4  0.07518797      4
## 5     CAR  0.12698413      4  0.06349206      4
## 6     CHI  0.12598425      4  0.03937008      4
```

```
## Getting rid of team logos of old teams
logos = teams_colors_logos %>% select(team_abbr, team_logo_espn) %>%
  filter(team_abbr != "LAR", team_abbr != "SD", team_abbr != "STL", team_abbr != "OAK")
```

```
## Visualizing fourth down aggressiveness of teams
ggplot(fourth_down_aggressiveness, aes(x= pct_passing, y = pct_running)) +
  geom_image(aes(image = logos$team_logo_espn), size = 0.05, asp = 16 / 9) +
  geom_hline(yintercept = mean(fourth_down_aggressiveness$pct_running)) +
  geom_vline(xintercept = mean(fourth_down_aggressiveness$pct_passing)) +
  annotate("text", x=.17, y=.1, label = "Aggressive on 4th down \n w/ pass & run") +
  annotate("text", x=.07, y=.1, label = "Aggressive on 4th down \n w/ run") +
  annotate("text", x=.05, y=.04, label = "Not Aggressive \n on 4th down \n w/ either pass or run") +
  annotate("text", x=.18, y=.03, label = "Aggressive on 4th down \n w/ pass") +
  labs(title = "Fourth Down Aggressiveness",
       subtitle = "Percent of 4th down plays that were passing or running plays")
```




```
x1= merge(first_down_pct_passing, second_down_pct_passing, by = "posteam")
x2= merge(x1, third_down_pct_passing, by = "posteam")
x3= merge(x2, fourth_down_pct_passing, by = "posteam")
```

```
## Warning in merge.data.frame(x2, fourth_down_pct_passing, by = "posteam"): column
## names 'pct_passing.x', 'down.x', 'pct_passing.y', 'down.y' are duplicated in the
## result
```

```
colnames(x3) = c("Team", "first_down_passing", "Down1", "second_down_passing", "Down2 ",
                 "third_down_passing", "Down3", "fourth_down_passing", "Down4")


x3 = x3 %>% select(Team, first_down_passing, second_down_passing, third_down_passing, fourth_down_passi
head(x3)
```

```
##   Team first_down_passing second_down_passing third_down_passing
## 1  ARI          0.5205479           0.5921450          0.8265306
## 2  ATL          0.5863454           0.7022472          0.8398058
## 3  BAL          0.3894325           0.4806202          0.6237624
## 4  BUF          0.4927235           0.5244957          0.7296137
## 5  CAR          0.5437500           0.6601671          0.8454106
## 6  CHI          0.5186916           0.6023055          0.8139535
##   fourth_down_passing
## 1          0.11475410
## 2          0.14678899
## 3          0.17171717
## 4          0.06766917
## 5          0.12698413
## 6          0.12598425
```

```
g1= ggplot(data = x3) + geom_histogram(aes(x = first_down_passing), binwidth = .01) +
  geom_vline(xintercept = mean(first_down_pct_passing$pct_passing), col = "blue", size = 1) +
  scale_x_continuous(limits = c(0,1))

g2= ggplot(data = x3) + geom_histogram(aes(x = second_down_passing), binwidth = .01) +
  geom_vline(xintercept = mean(second_down_pct_passing$pct_passing), col = "blue", size = 1) +
  scale_x_continuous(limits = c(0,1))

g3= ggplot(data = x3) + geom_histogram(aes(x = third_down_passing), binwidth = .01) +
  geom_vline(xintercept = mean(third_down_pct_passing$pct_passing), col = "blue", size = 1)+
  scale_x_continuous(limits = c(0,1))

g4= ggplot(data = x3) + geom_histogram(aes(x = fourth_down_passing), binwidth = .01) +
  geom_vline(xintercept = mean(fourth_down_pct_passing$pct_passing), col = "blue", size = 1)+
  scale_x_continuous(limits = c(0,1))

library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

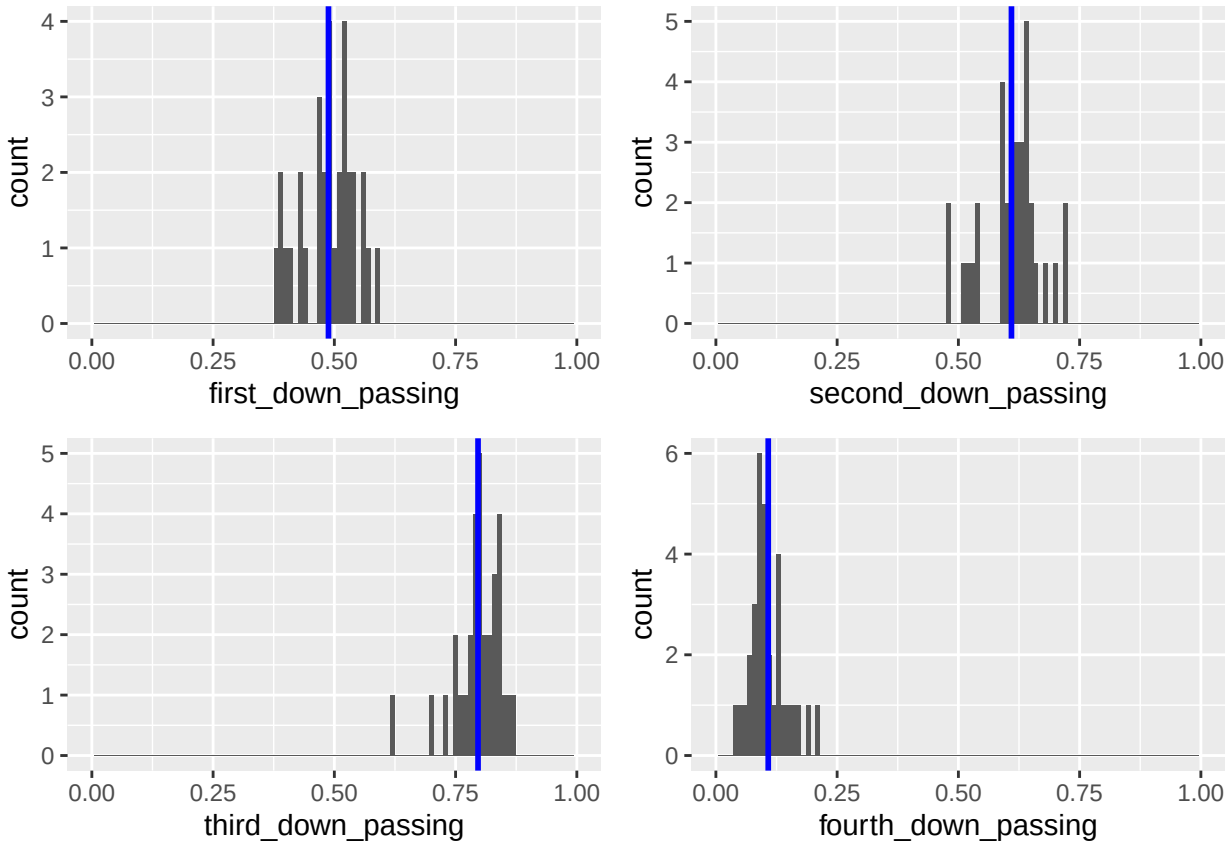
```
grid.arrange(g1,g2,g3,g4, nrow=2)
```

```
## Warning: Removed 2 rows containing missing values (geom_bar).
```

```
## Warning: Removed 2 rows containing missing values (geom_bar).
```

```
## Warning: Removed 2 rows containing missing values (geom_bar).
```

```
## Warning: Removed 2 rows containing missing values (geom_bar).
```


count
first_down_passing
second_down_passing
third_down_passing
fourth_down_passing
0.00
0.25
0.50
0.75
1.00