NFL 2020 Projections from Historical Data (data from 2009-2019)

Kevin Veeder

10/15/2020

Install Packages

```
install.packages("tidyverse")
install.packages("ggrepel")
install.packages("ggimage")
install.packages("nflfastR")
```

Load Packages

```
library(tidyverse)
library(ggrepel)
library(ggimage)
library(nflfastR)
```

converts numbers out of scientific notation

```
options(scipen = 9999)
```

Loading Data from the data repository

```
data <- readRDS(url('https://raw.githubusercontent.com/guga31bb/nflfastR-data/master/data/play_by_play_2019.rds'
))
```

Basics: how to look at your data

Dimensions

```
#this tells us that there are ```r rows``` rows (i.e., plays) in the data and ```r cols``` columns (variables):
dim(data)
#> [1] 48034   340
```

`str` displays the **str**ucture of the dataframe:

```
str(data[1:10])
#> tibble [48,034 × 10] (S3: tbl_df/tbl/data.frame)
#>  $ play_id     : num [1:48034] 1 36 51 79 100 121 148 185 214 239 ...
#>  $ game_id     : chr [1:48034] "2019_01_ATL_MIN" "2019_01_ATL_MIN" "2019_01_ATL_MIN" "2019_01_ATL_MIN" ...
#>  $ old_game_id : chr [1:48034] "2019090804" "2019090804" "2019090804" "2019090804" ...
#>  $ home_team   : chr [1:48034] "MIN" "MIN" "MIN" "MIN" ...
#>  $ away_team   : chr [1:48034] "ATL" "ATL" "ATL" "ATL" ...
#>  $ season_type : chr [1:48034] "REG" "REG" "REG" "REG" ...
#>  $ week        : int [1:48034] 1 1 1 1 1 1 1 1 1 1 ...
#>  $ posteam     : chr [1:48034] NA "ATL" "ATL" "ATL" ...
#>  $ posteam_type: chr [1:48034] NA "away" "away" "away" ...
#>  $ defteam     : chr [1:48034] NA "MIN" "MIN" "MIN" ...
```

Variable names

```
names(data)
#>   [1] "play_id"
#>   [2] "game_id"
#>   [3] "old_game_id"
#>   [4] "home_team"
#>   [5] "away_team"
#>   [6] "season_type"
#>   [7] "week"
#>   [8] "posteam"
#>   [9] "posteam_type"
#>  [10] "defteam"
#>  [11] "side_of_field"
#>  [12] "yardline_100"
#>  [13] "game_date"
#>  [14] "quarter_seconds_remaining"
#>  [15] "half_seconds_remaining"
#>  [16] "game_seconds_remaining"
#>  [17] "game_half"
#>  [18] "quarter_end"
#>  [19] "drive"
#>  [20] "sp"
#>  [21] "qtr"
#>  [22] "down"
#>  [23] "goal_to_go"
#>  [24] "time"
#>  [25] "yrdln"
#>  [26] "ydstogo"
#>  [27] "ydsnet"
#>  [28] "desc"
#>  [29] "play_type"
#>  [30] "yards_gained"
#>  [31] "shotgun"
#>  [32] "no_huddle"
#>  [33] "qb_dropback"
#>  [34] "qb_kneel"
#>  [35] "qb_spike"
#>  [36] "qb_scramble"
#>  [37] "pass_length"
#>  [38] "pass_location"
#>  [39] "air_yards"
#>  [40] "yards_after_catch"
#>  [41] "run_location"
#>  [42] "run_gap"
#>  [43] "field_goal_result"
#>  [44] "kick_distance"
#>  [45] "extra_point_result"
#>  [46] "two_point_conv_result"
#>  [47] "home_timeouts_remaining"
#>  [48] "away_timeouts_remaining"
#>  [49] "timeout"
#>  [50] "timeout_team"
#>  [51] "td_team"
#>  [52] "posteam_timeouts_remaining"
#>  [53] "defteam_timeouts_remaining"
#>  [54] "total_home_score"
#>  [55] "total_away_score"
#>  [56] "posteam_score"
#>  [57] "defteam_score"
#>  [58] "score_differential"
#>  [59] "posteam_score_post"
#>  [60] "defteam_score_post"
#>  [61] "score_differential_post"
#>  [62] "no_score_prob"
#>  [63] "opp_fg_prob"
#>  [64] "opp_safety_prob"
#>  [65] "opp_td_prob"
#>  [66] "fg_prob"
#>  [67] "safety_prob"
#>  [68] "td_prob"
#>  [69] "extra_point_prob"
#>  [70] "two_point_conversion_prob"
#>  [71] "ep"
```

```
#>  [72] "epa"
#>  [73] "total_home_epa"
#>  [74] "total_away_epa"
#>  [75] "total_home_rush_epa"
#>  [76] "total_away_rush_epa"
#>  [77] "total_home_pass_epa"
#>  [78] "total_away_pass_epa"
#>  [79] "air_epa"
#>  [80] "yac_epa"
#>  [81] "comp_air_epa"
#>  [82] "comp_yac_epa"
#>  [83] "total_home_comp_air_epa"
#>  [84] "total_away_comp_air_epa"
#>  [85] "total_home_comp_yac_epa"
#>  [86] "total_away_comp_yac_epa"
#>  [87] "total_home_raw_air_epa"
#>  [88] "total_away_raw_air_epa"
#>  [89] "total_home_raw_yac_epa"
#>  [90] "total_away_raw_yac_epa"
#>  [91] "wp"
#>  [92] "def_wp"
#>  [93] "home_wp"
#>  [94] "away_wp"
#>  [95] "wpa"
#>  [96] "home_wp_post"
#>  [97] "away_wp_post"
#>  [98] "vegas_wp"
#>  [99] "vegas_home_wp"
#> [100] "total_home_rush_wpa"
#> [101] "total_away_rush_wpa"
#> [102] "total_home_pass_wpa"
#> [103] "total_away_pass_wpa"
#> [104] "air_wpa"
#> [105] "yac_wpa"
#> [106] "comp_air_wpa"
#> [107] "comp_yac_wpa"
#> [108] "total_home_comp_air_wpa"
#> [109] "total_away_comp_air_wpa"
#> [110] "total_home_comp_yac_wpa"
#> [111] "total_away_comp_yac_wpa"
#> [112] "total_home_raw_air_wpa"
#> [113] "total_away_raw_air_wpa"
#> [114] "total_home_raw_yac_wpa"
#> [115] "total_away_raw_yac_wpa"
#> [116] "punt_blocked"
#> [117] "first_down_rush"
#> [118] "first_down_pass"
#> [119] "first_down_penalty"
#> [120] "third_down_converted"
#> [121] "third_down_failed"
#> [122] "fourth_down_converted"
#> [123] "fourth_down_failed"
#> [124] "incomplete_pass"
#> [125] "touchback"
#> [126] "interception"
#> [127] "punt_inside_twenty"
#> [128] "punt_in_endzone"
#> [129] "punt_out_of_bounds"
#> [130] "punt_downed"
#> [131] "punt_fair_catch"
#> [132] "kickoff_inside_twenty"
#> [133] "kickoff_in_endzone"
#> [134] "kickoff_out_of_bounds"
#> [135] "kickoff_downed"
#> [136] "kickoff_fair_catch"
#> [137] "fumble_forced"
#> [138] "fumble_not_forced"
#> [139] "fumble_out_of_bounds"
#> [140] "solo_tackle"
#> [141] "safety"
#> [142] "penalty"
#> [143] "tackled_for_loss"
```

```
#> [144] "fumble_lost"
#> [145] "own_kickoff_recovery"
#> [146] "own_kickoff_recovery_td"
#> [147] "qb_hit"
#> [148] "rush_attempt"
#> [149] "pass_attempt"
#> [150] "sack"
#> [151] "touchdown"
#> [152] "pass_touchdown"
#> [153] "rush_touchdown"
#> [154] "return_touchdown"
#> [155] "extra_point_attempt"
#> [156] "two_point_attempt"
#> [157] "field_goal_attempt"
#> [158] "kickoff_attempt"
#> [159] "punt_attempt"
#> [160] "fumble"
#> [161] "complete_pass"
#> [162] "assist_tackle"
#> [163] "lateral_reception"
#> [164] "lateral_rush"
#> [165] "lateral_return"
#> [166] "lateral_recovery"
#> [167] "passer_player_id"
#> [168] "passer_player_name"
#> [169] "receiver_player_id"
#> [170] "receiver_player_name"
#> [171] "rusher_player_id"
#> [172] "rusher_player_name"
#> [173] "lateral_receiver_player_id"
#> [174] "lateral_receiver_player_name"
#> [175] "lateral_rusher_player_id"
#> [176] "lateral_rusher_player_name"
#> [177] "lateral_sack_player_id"
#> [178] "lateral_sack_player_name"
#> [179] "interception_player_id"
#> [180] "interception_player_name"
#> [181] "lateral_interception_player_id"
#> [182] "lateral_interception_player_name"
#> [183] "punt_returner_player_id"
#> [184] "punt_returner_player_name"
#> [185] "lateral_punt_returner_player_id"
#> [186] "lateral_punt_returner_player_name"
#> [187] "kickoff_returner_player_name"
#> [188] "kickoff_returner_player_id"
#> [189] "lateral_kickoff_returner_player_id"
#> [190] "lateral_kickoff_returner_player_name"
#> [191] "punter_player_id"
#> [192] "punter_player_name"
#> [193] "kicker_player_name"
#> [194] "kicker_player_id"
#> [195] "own_kickoff_recovery_player_id"
#> [196] "own_kickoff_recovery_player_name"
#> [197] "blocked_player_id"
#> [198] "blocked_player_name"
#> [199] "tackle_for_loss_1_player_id"
#> [200] "tackle_for_loss_1_player_name"
#> [201] "tackle_for_loss_2_player_id"
#> [202] "tackle_for_loss_2_player_name"
#> [203] "qb_hit_1_player_id"
#> [204] "qb_hit_1_player_name"
#> [205] "qb_hit_2_player_id"
#> [206] "qb_hit_2_player_name"
#> [207] "forced_fumble_player_1_team"
#> [208] "forced_fumble_player_1_player_id"
#> [209] "forced_fumble_player_1_player_name"
#> [210] "forced_fumble_player_2_team"
#> [211] "forced_fumble_player_2_player_id"
#> [212] "forced_fumble_player_2_player_name"
#> [213] "solo_tackle_1_team"
#> [214] "solo_tackle_2_team"
#> [215] "solo_tackle_1_player_id"
```

```
#> [216] "solo_tackle_2_player_id"
#> [217] "solo_tackle_1_player_name"
#> [218] "solo_tackle_2_player_name"
#> [219] "assist_tackle_1_player_id"
#> [220] "assist_tackle_1_player_name"
#> [221] "assist_tackle_1_team"
#> [222] "assist_tackle_2_player_id"
#> [223] "assist_tackle_2_player_name"
#> [224] "assist_tackle_2_team"
#> [225] "assist_tackle_3_player_id"
#> [226] "assist_tackle_3_player_name"
#> [227] "assist_tackle_3_team"
#> [228] "assist_tackle_4_player_id"
#> [229] "assist_tackle_4_player_name"
#> [230] "assist_tackle_4_team"
#> [231] "pass_defense_1_player_id"
#> [232] "pass_defense_1_player_name"
#> [233] "pass_defense_2_player_id"
#> [234] "pass_defense_2_player_name"
#> [235] "fumbled_1_team"
#> [236] "fumbled_1_player_id"
#> [237] "fumbled_1_player_name"
#> [238] "fumbled_2_player_id"
#> [239] "fumbled_2_player_name"
#> [240] "fumbled_2_team"
#> [241] "fumble_recovery_1_team"
#> [242] "fumble_recovery_1_yards"
#> [243] "fumble_recovery_1_player_id"
#> [244] "fumble_recovery_1_player_name"
#> [245] "fumble_recovery_2_team"
#> [246] "fumble_recovery_2_yards"
#> [247] "fumble_recovery_2_player_id"
#> [248] "fumble_recovery_2_player_name"
#> [249] "return_team"
#> [250] "return_yards"
#> [251] "penalty_team"
#> [252] "penalty_player_id"
#> [253] "penalty_player_name"
#> [254] "penalty_yards"
#> [255] "replay_or_challenge"
#> [256] "replay_or_challenge_result"
#> [257] "penalty_type"
#> [258] "defensive_two_point_attempt"
#> [259] "defensive_two_point_conv"
#> [260] "defensive_extra_point_attempt"
#> [261] "defensive_extra_point_conv"
#> [262] "season"
#> [263] "cp"
#> [264] "cpoe"
#> [265] "series"
#> [266] "series_success"
#> [267] "series_result"
#> [268] "order_sequence"
#> [269] "start_time"
#> [270] "time_of_day"
#> [271] "stadium"
#> [272] "weather"
#> [273] "nfl_api_id"
#> [274] "play_clock"
#> [275] "play_deleted"
#> [276] "play_type_nfl"
#> [277] "special_teams_play"
#> [278] "st_play_type"
#> [279] "end_clock_time"
#> [280] "end_yard_line"
#> [281] "fixed_drive"
#> [282] "fixed_drive_result"
#> [283] "drive_real_start_time"
#> [284] "drive_play_count"
#> [285] "drive_time_of_possession"
#> [286] "drive_first_downs"
#> [287] "drive_inside20"
```

```
#> [288] "drive_ended_with_score"
#> [289] "drive_quarter_start"
#> [290] "drive_quarter_end"
#> [291] "drive_yards_penalized"
#> [292] "drive_start_transition"
#> [293] "drive_end_transition"
#> [294] "drive_game_clock_start"
#> [295] "drive_game_clock_end"
#> [296] "drive_start_yard_line"
#> [297] "drive_end_yard_line"
#> [298] "drive_play_id_started"
#> [299] "drive_play_id_ended"
#> [300] "away_score"
#> [301] "home_score"
#> [302] "location"
#> [303] "result"
#> [304] "total"
#> [305] "spread_line"
#> [306] "total_line"
#> [307] "div_game"
#> [308] "roof"
#> [309] "surface"
#> [310] "temp"
#> [311] "wind"
#> [312] "home_coach"
#> [313] "away_coach"
#> [314] "stadium_id"
#> [315] "game_stadium"
#> [316] "success"
#> [317] "passer"
#> [318] "passer_jersey_number"
#> [319] "rusher"
#> [320] "rusher_jersey_number"
#> [321] "receiver"
#> [322] "receiver_jersey_number"
#> [323] "pass"
#> [324] "rush"
#> [325] "first_down"
#> [326] "aborted_play"
#> [327] "special"
#> [328] "play"
#> [329] "passer_id"
#> [330] "rusher_id"
#> [331] "receiver_id"
#> [332] "name"
#> [333] "jersey_number"
#> [334] "id"
#> [335] "qb_epa"
#> [336] "xyac_epa"
#> [337] "xyac_mean_yardage"
#> [338] "xyac_median_yardage"
#> [339] "xyac_success"
#> [340] "xyac_fd"
```

Select

Head + Manipulation

```
#"`desc`" is the variable that lists the description of what happened on the play

#these are the first 6 rows from a week 1 game, ATL @ MIN
data %>%
  select(posteam, defteam, desc, rush, pass) %>%
  head()
#> # A tibble: 6 x 5
#>   posteam defteam desc                                           rush  pass
#>   <chr>   <chr>   <chr>                                         <dbl> <dbl>
#> 1 <NA>    <NA>    GAME                                              0     0
#> 2 ATL     MIN     5-D.Bailey kicks 65 yards from MIN 35 to end zone…     0     0
#> 3 ATL     MIN     (15:00) 2-M.Ryan sacked at ATL 17 for -8 yards (5…     0     1
#> 4 ATL     MIN     (14:20) 24-D.Freeman right tackle to ATL 21 for 4…     1     0
#> 5 ATL     MIN     (13:41) (Shotgun) 2-M.Ryan scrambles left end to …     0     1
#> 6 ATL     MIN     (12:59) 5-M.Bosher punt is BLOCKED by 50-E.Wilson…     0     0
```

Easier way to read in this data

```
data %>% select(posteam, defteam, desc, rush, pass) %>% head()
```

Filter

```
# `filter`, which lets you filter the data to what you want. The following returns only plays that are run plays
 and pass plays

# name column describes the player most involved with the play
data %>%
  filter(rush == 1 | pass == 1) %>%
  select(posteam, desc, rush, pass, name, passer, rusher, receiver) %>%
  head()
#> # A tibble: 6 x 8
#>   posteam desc                        rush  pass name    passer  rusher receiver
#>   <chr>   <chr>                      <dbl> <dbl> <chr>   <chr>   <chr>  <chr>
#> 1 ATL     (15:00) 2-M.Ryan sacked a…     0     1 M.Ryan  M.Ryan  <NA>   <NA>
#> 2 ATL     (14:20) 24-D.Freeman righ…     1     0 D.Free… <NA>    D.Fre… <NA>
#> 3 ATL     (13:41) (Shotgun) 2-M.Rya…     0     1 M.Ryan  M.Ryan  <NA>   <NA>
#> 4 MIN     (12:53) 33-D.Cook right e…     1     0 D.Cook  <NA>    D.Cook <NA>
#> 5 MIN     (12:32) 8-K.Cousins pass …     0     1 K.Cous… K.Cous… <NA>   D.Cook
#> 6 MIN     (11:57) 8-K.Cousins pass …     0     1 K.Cous… K.Cous… <NA>   A.Thiel…
```

What if we wanted to view special teams plays?

```
data %>%
  filter(special == 1) %>%
  select(down, ydstogo, desc) %>%
  head()
#> # A tibble: 6 x 3
#>    down ydstogo desc
#>   <dbl>   <dbl> <chr>
#> 1    NA       0 5-D.Bailey kicks 65 yards from MIN 35 to end zone, Touchback.
#> 2     4       2 (12:59) 5-M.Bosher punt is BLOCKED by 50-E.Wilson, Center-47-J.…
#> 3    NA       0 (Kick formation) 5-D.Bailey extra point is GOOD, Center-58-A.Cu…
#> 4    NA       0 5-D.Bailey kicks 67 yards from MIN 35 to ATL -2. 38-K.Barner to…
#> 5    NA       0 (Kick formation) 5-D.Bailey extra point is GOOD, Center-58-A.Cu…
#> 6    NA       0 5-D.Bailey kicks 65 yards from MIN 35 to end zone, Touchback.
```

Fourth down plays?

```
data %>%
  filter(down == 4) %>%
  select(down, ydstogo, desc) %>%
  head()
#> # A tibble: 6 x 3
#>    down ydstogo desc
#>   <dbl>   <dbl> <chr>
#> 1     4       2 (12:59) 5-M.Bosher punt is BLOCKED by 50-E.Wilson, Center-47-J.…
#> 2     4      19 (2:38) 5-M.Bosher punts 33 yards to MIN 8, Center-47-J.Harris, …
#> 3     4      20 (12:33) 2-B.Colquitt punts 51 yards to ATL 17, Center-58-A.Cutt…
#> 4     4      27 (1:49) 5-M.Bosher punts 45 yards to MIN 10, Center-47-J.Harris,…
#> 5     4      10 (:49) 2-B.Colquitt punts 57 yards to ATL 33, Center-58-A.Cuttin…
#> 6     4       1 (10:56) 2-B.Colquitt punts 42 yards to ATL 10, Center-58-A.Cutt…
```

Fourth down plays that aren't special teams plays?

```
data %>%
  filter(down == 4 & special == 0) %>%
  select(down, ydstogo, desc) %>%
  head()
#> # A tibble: 6 x 3
#>    down ydstogo desc
#>   <dbl>   <dbl> <chr>
#> 1     4       5 (9:25) (Shotgun) 2-M.Ryan pass deep left to 18-C.Ridley for 20 …
#> 2     4       2 (4:39) (Punt formation) PENALTY on MIN, Delay of Game, 5 yards,…
#> 3     4       2 (1:27) (No Huddle, Shotgun) 2-M.Ryan pass short left to 11-J.Jo…
#> 4     4       1 (2:59) (Punt formation) Direct snap to 41-A.Levine.  41-A.Levin…
#> 5     4       3 (9:30) (Shotgun) 3-R.Griffin pass short left to 89-M.Andrews fo…
#> 6     4       1 (3:55) 17-J.Allen FUMBLES (Aborted) at NYJ 37, RECOVERED by NYJ…
```

To save a new dataframe of just the plays we want

we need to use `<-` to assign a new dataframe. Let's save a new dataframe that's just run plays and pass plays with non-missing EPA, called `pbp_rp`.

```
# `!is.na(epa)` means to exclude plays with missing (`na`) EPA. The `!` symbol is often used by computer folk to
 negate something, so `is.na(epa)` means "EPA is missing" and `!is.na(epa)` means "EPA is not missing", which we
 have used above.
pbp_rp <- data %>%
  filter(rush == 1 | pass == 1, !is.na(epa))
```

Group by and Summarize

Let's take a look at how various Cowboys' running backs fared on run plays in 2019:

```
#`n()`, which returns the number in a group

# summarize is useful for collapsing the data down to a summary of what you're looking at, and here, while groupi
ng by player, we're summarizing the mean of EPA, success, yardage

pbp_rp %>%
    filter(posteam == "DAL", rush == 1) %>%
    group_by(rusher) %>%
    summarize(
      mean_epa = mean(epa), success_rate = mean(success), ypc=mean(yards_gained), plays=n()
      ) %>%
    arrange(-mean_epa) %>%
    filter(plays > 20)
#> `summarise()` ungrouping output (override with `.groups` argument)
#> # A tibble: 3 x 5
#>   rusher     mean_epa success_rate   ypc plays
#>   <chr>         <dbl>        <dbl> <dbl> <int>
#> 1 D.Prescott   0.288         0.591  6.41    22
#> 2 T.Pollard   -0.0266        0.456  5.08    90
#> 3 E.Elliott   -0.0412        0.411  4.39   309

# Therefore, Prescott was much more effective as a rusher in 2019 than the running backs, and there was no meanin
gful difference between Pollard and Elliott in efficiency.
```

you'll notice that the above doesn't match up with the official stats. This is because `nflfastR` computes EPA and provides player names on plays with penalties and on two-point conversions. So if wanting to match the official stats, we need to restrict to `down <= 4` (to excluded two-point conversions, which have down listed as `NA`) and `play_type = run` (to exclude penalties, which are `play_type = no_play`)

```
pbp_rp %>%
    filter(posteam == "DAL", down <=4, play_type == 'run') %>%
    group_by(rusher) %>%
    summarize(
      mean_epa = mean(epa), success_rate = mean(success), ypc=mean(yards_gained), plays=n()
      ) %>%
    filter(plays > 20)
#> `summarise()` ungrouping output (override with `.groups` argument)
#> # A tibble: 3 x 5
#>   rusher     mean_epa success_rate   ypc plays
#>   <chr>         <dbl>        <dbl> <dbl> <int>
#> 1 D.Prescott   0.288         0.591  6.41    22
#> 2 E.Elliott   -0.0185        0.422  4.51   301
#> 3 T.Pollard   -0.0210        0.453  5.29    86

#Now we can see that Zeke has 301 carries at 4.5 yards/carry, and Pollard has 86 carries for 5.3 yards/carry.
```

Manipulating columns: mutate, if_else, and case_when

Let's say we want to make a new column, named `home`, which is equal to 1 if the team with the ball is the home team. Let's introduce another extremely useful function, `if_else`:

```
#`mutate` is R's word for creating a new column (or overwriting an existing one)

pbp_rp %>%
  mutate(
    home = if_else(posteam == home_team, 1, 0)
  ) %>%
  select(posteam, home_team, home) %>%
  head(10)
#> # A tibble: 10 x 3
#>    posteam home_team  home
#>    <chr>   <chr>     <dbl>
#>  1 ATL     MIN           0
#>  2 ATL     MIN           0
#>  3 ATL     MIN           0
#>  4 MIN     MIN           1
#>  5 MIN     MIN           1
#>  6 MIN     MIN           1
#>  7 ATL     MIN           0
#>  8 ATL     MIN           0
#>  9 ATL     MIN           0
#> 10 MIN     MIN           1
```

we've created a new column called `home` . The above uses `if_else` , which uses the following pattern: condition (in this case, `posteam == home_team`), value if condition is true (in this case, if `posteam == home_team` , it is 1), and value if the condition is false (0). So we could use this to, for example, look at average EPA/play by home and road teams:

```
pbp_rp %>%
  mutate(
    home = if_else(posteam == home_team, 1, 0)
  ) %>%
  group_by(home) %>%
  summarize(epa = mean(epa))
#> `summarise()` ungrouping output (override with `.groups` argument)
#> # A tibble: 2 x 2
#>    home     epa
#>   <dbl>   <dbl>
#> 1     0  0.0213
#> 2     1 -0.0164
```

`if_else` is nice if you're creating a new column based on a simple condition. But what if you need to do something more complicated? `case_when` is a good option.

```
# `case_when`: we have condition (for the first one, air yards less than 0), followed by `~`, followed by assignm
ent (for the first one, "Negative").

pbp_rp %>%
  filter(!is.na(cp)) %>%
  mutate(
    depth = case_when(
      air_yards < 0 ~ "Negative",
      air_yards >= 0 & air_yards < 10 ~ "Short",
      air_yards >= 10 & air_yards < 20 ~ "Medium",
      air_yards >= 20 ~ "Deep"
    )
  ) %>%
  group_by(depth) %>%
  summarize(cp = mean(cp))
#> `summarise()` ungrouping output (override with `.groups` argument)
#> # A tibble: 4 x 2
#>   depth       cp
#>   <chr>    <dbl>
#> 1 Deep     0.367
#> 2 Medium   0.573
#> 3 Negative 0.847
#> 4 Short    0.718

# In the above, we created 4 bins based on air yards and got average completion probability (`cp`) based on the `
nflfastR` model. Unsurprisingly, `cp` is lower the longer downfield a throw goes.
```

Basic Figures

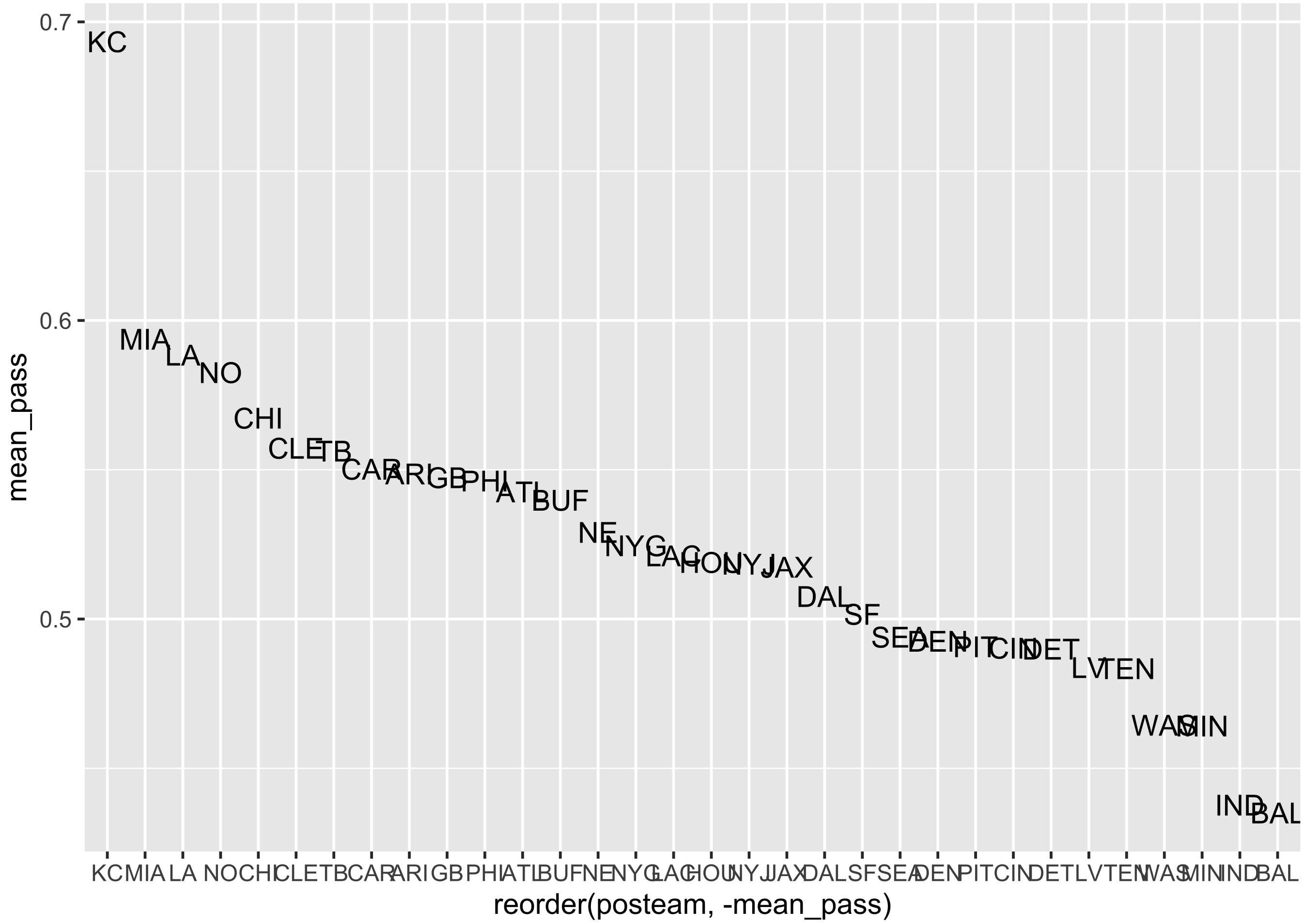
Most PASS HEAVY teams in the first half on early downs with win probability between 20 and 80

(excluding the final 2 minutes of the half when everyone is pass-happy)

```
# `arrange`, which sorts the data by the variable(s) given

# The minus sign in front of `mean_pass` means to sort in descending order.

schotty <- pbp_rp %>%
    filter(wp > .20 & wp < .80 & down <= 2 & qtr <= 2 & half_seconds_remaining > 120) %>%
    group_by(posteam) %>%
    summarize(mean_pass = mean(pass), plays = n()) %>%
    arrange(-mean_pass)
#> `summarise()` ungrouping output (override with `.groups` argument)
schotty
#> # A tibble: 32 x 3
#>    posteam mean_pass plays
#>    <chr>       <dbl> <int>
#>  1 KC          0.693   375
#>  2 MIA         0.594   288
#>  3 LA          0.588   328
#>  4 NO          0.583   321
#>  5 CHI         0.567   312
#>  6 CLE         0.557   271
#>  7 TB          0.556   320
#>  8 CAR         0.550   269
#>  9 ARI         0.549   319
#> 10 GB          0.547   285
#> # … with 22 more rows
```

Plotting our figure

```

# the "reorder" sorts the teams according to pass rate, with the "-" again saying to do it in descending order.
 "aes" is short for "aesthetic", which is R's weird way of asking which variables should go on the x and y axes.

ggplot(schotty, aes(x=reorder(posteam,-mean_pass), y=mean_pass)) +
        geom_text(aes(label=posteam))
```



Loading multiple seasons

Because all the data is stored in the data repository, it is very easy to use data from multiple seasons. The repository page (https://github.com/guga31bb/nflfastR-data) has instructions for loading multiple seasons:

```
# `map_df` stitches together the output from running a function repeatedly with different inputs. In this case, t
he function is simply reading one season's data, and the inputs are the list of seasons we want: in the above, 20
15 through 2019. But all you need to know how to do is change the range of seasons to get whichever seasons you w
ant.

seasons <- 2015:2019
pbp <- map_df(seasons, function(x) {
  readRDS(
    url(
      paste0("https://raw.githubusercontent.com/guga31bb/nflfastR-data/master/data/play_by_play_",x,".rds")
    )
  )
})
```

```
pbp %>%
  group_by(season) %>%
  summarize(n = n())
#> `summarise()` ungrouping output (override with `.groups` argument)
#> # A tibble: 5 x 2
#>   season     n
#>    <int> <int>
#> 1   2015 48869
#> 2   2016 48419
#> 3   2017 47997
#> 4   2018 47874
#> 5   2019 48034

# So each season has about ~48,000 plays
```

Lets look at various play types:

```
pbp %>%
  group_by(play_type) %>%
  summarize(n = n())
#> `summarise()` ungrouping output (override with `.groups` argument)
#> # A tibble: 10 x 2
#>    play_type       n
#>    <chr>       <int>
#>  1 extra_point  6115
#>  2 field_goal   5138
#>  3 kickoff     13525
#>  4 no_play     22778
#>  5 pass        99986
#>  6 punt        12021
#>  7 qb_kneel     2090
#>  8 qb_spike      340
#>  9 run         68129
#> 10 <NA>        11071
```

Figures with QB stats

```
qbs <- pbp %>%
  filter(week <= 17, !is.na(epa)) %>%
  group_by(id, name) %>%
  summarize(
    epa = mean(qb_epa),
    cpoe = mean(cpoe, na.rm = T),
    n_dropbacks = sum(pass),
    n_plays = n(),
    team = last(posteam)
  ) %>%
  ungroup() %>%
  filter(n_dropbacks > 100 & n_plays > 1000)
#> `summarise()` regrouping output by 'id' (override with `.groups` argument)

qbs
#> # A tibble: 40 x 7
#>    id                         name            epa   cpoe n_dropbacks n_plays team
#>    <chr>                      <chr>         <dbl>  <dbl>       <dbl>   <int> <chr>
#>  1 32013030-2d30-3031-3935… T.Brady     0.195    1.09         3181    3280 NE
#>  2 32013030-2d30-3032-3035… D.Brees     0.199    4.46         2965    3071 NO
#>  3 32013030-2d30-3032-3132… J.McCown    0.00259  0.260        1152    1188 PHI
#>  4 32013030-2d30-3032-3134… C.Palmer    0.142    2.19         1607    1648 ARI
#>  5 32013030-2d30-3032-3238… E.Manning   0.00570 -1.57         2808    2860 NYG
#>  6 32013030-2d30-3032-3239… B.Roethlis… 0.188    2.72         2559    2619 PIT
#>  7 32013030-2d30-3032-3239… P.Rivers    0.139    2.27         3301    3352 LAC
#>  8 32013030-2d30-3032-3334… A.Smith     0.0961   1.62         2151    2259 WAS
#>  9 32013030-2d30-3032-3334… A.Rodgers   0.137    0.680        3152    3234 GB
#> 10 32013030-2d30-3032-3336… R.Fitzpatr… 0.0980  -0.921        2204    2285 MIA
#> # … with 30 more rows

# First, we're grouping by `id` and `name` to make sure we're getting unique players; i.e., if two players have t
he same name (like Javorius Allen and Josh Allen both being J.Allen), we are also using their id to differntiate
 them

# `qb_epa` is equal to EPA in all instances except for when a pass is completed and a fumble is lost, in which ca
se a QB gets "credit" for the play up to the spot the fumble was lost (making EPA function like passing yards).

# The `last` part in the `summarize` comment gets the last team that a player was observed playing with.

# `n_dropbacks > 100` makes sure we're only including quarterbacks (making sure they hit a certain number of drop
backs)

#The `ungroup()` near the end is good practice after grouping to make sure you don't get weird behavior with the
 data you created down the line.
```

Real life example: let’s make a win total model

Get team wins each season (using Lee Sharpe's famous `games` file)

```
games <- readRDS(url("http://www.habitatring.com/games.rds"))
str(games)
#> tibble [5,839 × 35] (S3: tbl_df/tbl/data.frame)
#>  $ game_id         : chr [1:5839] "1999_01_MIN_ATL" "1999_01_KC_CHI" "1999_01_PIT_CLE" "1999_01_OAK_GB" ...
#>  $ season          : int [1:5839] 1999 1999 1999 1999 1999 1999 1999 1999 1999 1999 ...
#>  $ game_type       : chr [1:5839] "REG" "REG" "REG" "REG" ...
#>  $ week            : int [1:5839] 1 1 1 1 1 1 1 1 1 1 ...
#>  $ gameday         : chr [1:5839] "1999-09-12" "1999-09-12" "1999-09-12" "1999-09-12" ...
#>  $ weekday         : chr [1:5839] "Sunday" "Sunday" "Sunday" "Sunday" ...
#>  $ gametime        : chr [1:5839] NA NA NA NA ...
#>  $ away_team       : chr [1:5839] "MIN" "KC" "PIT" "OAK" ...
#>  $ away_score      : int [1:5839] 17 17 43 24 14 3 10 30 25 28 ...
#>  $ home_team       : chr [1:5839] "ATL" "CHI" "CLE" "GB" ...
#>  $ home_score      : int [1:5839] 14 20 0 28 31 41 19 28 24 20 ...
#>  $ location        : chr [1:5839] "Home" "Home" "Home" "Home" ...
#>  $ result          : int [1:5839] -3 3 -43 4 17 38 9 -2 -1 -8 ...
#>  $ total           : int [1:5839] 31 37 43 52 45 44 29 58 49 48 ...
#>  $ old_game_id     : chr [1:5839] "1999091210" "1999091206" "1999091213" "1999091208" ...
#>  $ away_rest       : int [1:5839] 7 7 7 7 7 7 7 7 7 7 ...
#>  $ home_rest       : int [1:5839] 7 7 7 7 7 7 7 7 7 7 ...
#>  $ away_moneyline  : int [1:5839] NA NA NA NA NA NA NA NA NA NA ...
#>  $ home_moneyline  : int [1:5839] NA NA NA NA NA NA NA NA NA NA ...
#>  $ spread_line     : num [1:5839] -4 -3 -6 9 -3 5.5 3.5 7 -3 9.5 ...
#>  $ away_spread_odds: int [1:5839] NA NA NA NA NA NA NA NA NA NA ...
#>  $ home_spread_odds: int [1:5839] NA NA NA NA NA NA NA NA NA NA ...
#>  $ total_line      : num [1:5839] 49 38 37 43 45.5 49 38 44.5 37 42 ...
#>  $ under_odds      : int [1:5839] NA NA NA NA NA NA NA NA NA NA ...
#>  $ over_odds       : int [1:5839] NA NA NA NA NA NA NA NA NA NA ...
#>  $ div_game        : int [1:5839] 0 0 1 0 1 0 1 1 1 0 ...
#>  $ roof            : chr [1:5839] "dome" "outdoors" "outdoors" "outdoors" ...
#>  $ surface         : chr [1:5839] "astroturf" "grass" "grass" "grass" ...
#>  $ temp            : int [1:5839] NA 80 78 67 NA 76 NA 73 75 NA ...
#>  $ wind            : int [1:5839] NA 12 12 10 NA 8 NA 5 3 NA ...
#>  $ away_coach      : chr [1:5839] "Dennis Green" "Gunther Cunningham" "Bill Cowher" "Jon Gruden" ...
#>  $ home_coach      : chr [1:5839] "Dan Reeves" "Dick Jauron" "Chris Palmer" "Ray Rhodes" ...
#>  $ referee         : chr [1:5839] "Gerry Austin" "Phil Luckett" "Bob McElwee" "Tony Corrente" ...
#>  $ stadium_id      : chr [1:5839] "ATL00" "CHI98" "CLE00" "GNB00" ...
#>  $ stadium         : chr [1:5839] "Georgia Dome" "Soldier Field" "Cleveland Browns Stadium" "Lambeau Field"
 ...
```

To start, we want to create a dataframe where each row is a team-season observation, listing how many games they won: I'm going to just take the home and away results and bind together

```
home <- games %>%
  filter(game_type == 'REG') %>%
  select(season, week, home_team, result) %>%
  rename(team = home_team)
home %>% head(5)
#> # A tibble: 5 x 4
#>   season  week team  result
#>    <int> <int> <chr>  <int>
#> 1   1999     1 ATL       -3
#> 2   1999     1 CHI        3
#> 3   1999     1 CLE      -43
#> 4   1999     1 GB         4
#> 5   1999     1 IND       17
```

Note that we used `rename` to change `home_team` to `team`

```
away <- games %>%
  filter(game_type == 'REG') %>%
  select(season, week, away_team, result) %>%
  rename(team = away_team) %>%
  mutate(result = -result)
away %>% head(5)
#> # A tibble: 5 x 4
#>   season  week team  result
#>    <int> <int> <chr>  <int>
#> 1   1999     1 MIN        3
#> 2   1999     1 KC        -3
#> 3   1999     1 PIT       43
#> 4   1999     1 OAK       -4
#> 5   1999     1 BUF      -17
```

For away teams, we need to flip the result since result is given from the perspective of the home team. Now let's make a columns called `win` based on the result.

```
results <- bind_rows(home, away) %>%
  arrange(week) %>%
  mutate(
    win = case_when(
      result > 0 ~ 1,
      result < 0 ~ 0,
      result == 0 ~ 0.5
    )
  )

results %>% filter(season == 2019 & team == 'SEA')
#> # A tibble: 16 x 5
#>    season  week team  result   win
#>     <int> <int> <chr>  <int> <dbl>
#>  1   2019     1 SEA        1     1
#>  2   2019     2 SEA        2     1
#>  3   2019     3 SEA       -6     0
#>  4   2019     4 SEA       17     1
#>  5   2019     5 SEA        1     1
#>  6   2019     6 SEA        4     1
#>  7   2019     7 SEA      -14     0
#>  8   2019     8 SEA        7     1
#>  9   2019     9 SEA        6     1
#> 10   2019    10 SEA        3     1
#> 11   2019    12 SEA        8     1
#> 12   2019    13 SEA        7     1
#> 13   2019    14 SEA      -16     0
#> 14   2019    15 SEA        6     1
#> 15   2019    16 SEA      -14     0
#> 16   2019    17 SEA       -5     0

# Note: `results %>% filter(season == 2019 & team == 'SEA')` part at the end isn't actually for saving the data i
n a new form, but just making sure the previous step did what I wanted
```

Now that we have the dataframe we wanted, we can get team wins by season easily:

this is the 10 seasons with the most wins:

```
team_wins <- results %>%
  group_by(team, season) %>%
  summarize(
    wins = sum(win),
    point_diff = sum(result)) %>%
  ungroup()
#> `summarise()` regrouping output by 'team' (override with `.groups` argument)

team_wins %>%
  arrange(-wins) %>%
  head(10)
#> # A tibble: 10 x 4
#>    team  season  wins point_diff
#>    <chr>  <int> <dbl>      <int>
#>  1 NE      2007    16        315
#>  2 CAR     2015    15        192
#>  3 GB      2011    15        201
#>  4 PIT     2004    15        121
#>  5 BAL     2019    14        249
#>  6 IND     2005    14        192
#>  7 IND     2009    14        111
#>  8 JAX     1999    14        179
#>  9 NE      2003    14        110
#> 10 NE      2004    14        177
```

Get team EPA by season

Let's start by getting data from every season from the `nflfastR` data repository:

```
seasons <- 1999:2019
pbp <- map_df(seasons, function(x) {
  readRDS(
    url(
      paste0("https://raw.githubusercontent.com/guga31bb/nflfastR-data/master/data/play_by_play_",x,".rds")
    )
  ) %>%
    filter(rush == 1 | pass == 1, week <= 17, !is.na(epa), !is.na(posteam), posteam != "") %>%
    select(season, posteam, pass, defteam, epa)
})

#note that we are only keeping regular season games with week<=17
```

Getting EPA/play on offense and defense: We know we need to group by team, season, and pass

```
pbp %>%
  group_by(posteam, season, pass) %>%
  summarize(epa = mean(epa)) %>%
  head(4)
#> `summarise()` regrouping output by 'posteam', 'season' (override with `.groups` argument)
#> # A tibble: 4 x 4
#> # Groups:   posteam, season [2]
#>   posteam season  pass     epa
#>   <chr>    <int> <dbl>   <dbl>
#> 1 ARI       1999     0 -0.201
#> 2 ARI       1999     1 -0.162
#> 3 ARI       2000     0 -0.242
#> 4 ARI       2000     1 -0.0678
```

`pivot_wider` helps us to get each team-season on the same row

Reference page for `pivot_wider`: (https://tidyr.tidyverse.org/reference/pivot_wider.html (https://tidyr.tidyverse.org/reference/pivot_wider.html))

```
pbp %>%
  group_by(posteam, season, pass) %>%
  summarize(epa = mean(epa)) %>%
  pivot_wider(names_from = pass, values_from = epa) %>%
  head(10)
#> `summarise()` regrouping output by 'posteam', 'season' (override with `.groups` argument)
#> # A tibble: 10 x 4
#> # Groups:   posteam, season [10]
#>    posteam season    `0`      `1`
#>    <chr>    <int>  <dbl>    <dbl>
#>  1 ARI       1999 -0.201 -0.162
#>  2 ARI       2000 -0.242 -0.0678
#>  3 ARI       2001 -0.177  0.0740
#>  4 ARI       2002 -0.134 -0.0662
#>  5 ARI       2003 -0.219 -0.120
#>  6 ARI       2004 -0.116 -0.0833
#>  7 ARI       2005 -0.262  0.00971
#>  8 ARI       2006 -0.164  0.0341
#>  9 ARI       2007 -0.116  0.0108
#> 10 ARI       2008 -0.136  0.138
```

Now let’s rename to something more sensible (offense) and save the variable:

Note that variable names that are numbers need to be surrounded in tick marks for this to work.

```
offense <- pbp %>%
  group_by(posteam, season, pass) %>%
  summarize(epa = mean(epa)) %>%
  pivot_wider(names_from = pass, values_from = epa) %>%
  rename(off_pass_epa = `1`, off_rush_epa = `0`)
#> `summarise()` regrouping output by 'posteam', 'season' (override with `.groups` argument)

head(offense, 20)
#> # A tibble: 20 x 4
#> # Groups:   posteam, season [20]
#>    posteam season off_rush_epa off_pass_epa
#>    <chr>    <int>        <dbl>        <dbl>
#>  1 ARI       1999      -0.201      -0.162
#>  2 ARI       2000      -0.242      -0.0678
#>  3 ARI       2001      -0.177       0.0740
#>  4 ARI       2002      -0.134      -0.0662
#>  5 ARI       2003      -0.219      -0.120
#>  6 ARI       2004      -0.116      -0.0833
#>  7 ARI       2005      -0.262       0.00971
#>  8 ARI       2006      -0.164       0.0341
#>  9 ARI       2007      -0.116       0.0108
#> 10 ARI       2008      -0.136       0.138
#> 11 ARI       2009      -0.143       0.0374
#> 12 ARI       2010      -0.167      -0.246
#> 13 ARI       2011      -0.0782     -0.0789
#> 14 ARI       2012      -0.239      -0.267
#> 15 ARI       2013      -0.106       0.0317
#> 16 ARI       2014      -0.166       0.0564
#> 17 ARI       2015      -0.0587      0.261
#> 18 ARI       2016      -0.0925      0.0404
#> 19 ARI       2017      -0.207      -0.0863
#> 20 ARI       2018      -0.187      -0.267
```

and then do the same for defense:

```
defense <- pbp %>%
  group_by(defteam, season, pass) %>%
  summarize(epa = mean(epa)) %>%
  pivot_wider(names_from = pass, values_from = epa) %>%
  rename(def_pass_epa = `1`, def_rush_epa = `0`)
#> `summarise()` regrouping output by 'defteam', 'season' (override with `.groups` argument)

head(defense, 20)
#> # A tibble: 20 x 4
#> # Groups:   defteam, season [20]
#>    defteam season def_rush_epa def_pass_epa
#>    <chr>    <int>        <dbl>        <dbl>
#>  1 ARI       1999     -0.00937      0.00386
#>  2 ARI       2000      0.0286       0.188
#>  3 ARI       2001     -0.0689       0.0803
#>  4 ARI       2002     -0.0192       0.165
#>  5 ARI       2003     -0.0627       0.191
#>  6 ARI       2004     -0.132       -0.0335
#>  7 ARI       2005     -0.0618      -0.0420
#>  8 ARI       2006     -0.171        0.0550
#>  9 ARI       2007     -0.178        0.0156
#> 10 ARI       2008     -0.0778       0.102
#> 11 ARI       2009     -0.0628      -0.0342
#> 12 ARI       2010     -0.0988       0.0536
#> 13 ARI       2011     -0.0565       0.0203
#> 14 ARI       2012     -0.0864      -0.102
#> 15 ARI       2013     -0.200       -0.0312
#> 16 ARI       2014     -0.136       -0.00474
#> 17 ARI       2015     -0.245       -0.0177
#> 18 ARI       2016     -0.177       -0.0351
#> 19 ARI       2017     -0.169       -0.0539
#> 20 ARI       2018      0.0182       0.0383
```

Looking at the top 5 pass offenses and defenses:

```
#top 5 offenses
offense %>%
  arrange(-off_pass_epa) %>%
  head(5)
#> # A tibble: 5 x 4
#> # Groups:   posteam, season [5]
#>   posteam season off_rush_epa off_pass_epa
#>   <chr>    <int>        <dbl>        <dbl>
#> 1 NE        2007      0.00216        0.422
#> 2 IND       2004     -0.00125        0.413
#> 3 GB        2011     -0.114          0.413
#> 4 KC        2018      0.0209         0.349
#> 5 DEN       2013     -0.0296         0.344

#top 5 defenses
defense %>%
  arrange(def_pass_epa) %>%
  head(5)
#> # A tibble: 5 x 4
#> # Groups:   defteam, season [5]
#>   defteam season def_rush_epa def_pass_epa
#>   <chr>    <int>        <dbl>        <dbl>
#> 1 TB        2002      -0.0755       -0.292
#> 2 NE        2019      -0.164        -0.244
#> 3 JAX       2017      -0.111        -0.243
#> 4 NYJ       2009      -0.103        -0.220
#> 5 LA        2003      -0.0543       -0.214
```

Fix team names and join

Now we're ready to bind it all together. Actually, let's make sure all the team names are ready too.

```
team_wins %>%
  group_by(team) %>%
  summarize(n=n()) %>%
  arrange(n)
#> `summarise()` ungrouping output (override with `.groups` argument)
#> # A tibble: 35 x 2
#>    team      n
#>    <chr> <int>
#>  1 LV        1
#>  2 LAC       4
#>  3 LA        5
#>  4 STL      17
#>  5 SD       18
#>  6 HOU      19
#>  7 OAK      21
#>  8 ARI      22
#>  9 ATL      22
#> 10 BAL      22
#> # … with 25 more rows
```

Nope, not yet, we need to fix the Raiders, Rams, and Chargers, which are LV, LA, and LAC in `nflfastR` .

```
team_wins <- team_wins %>%
  mutate(
    team = case_when(
      team == 'OAK' ~ 'LV',
      team == 'SD' ~ 'LAC',
      team == 'STL' ~ 'LA',
      TRUE ~ team
    )
  )

#`TRUE` statement at the bottom says that if none of the above cases are found, keep team the same
```

Checking to see if all teams have the correct number of seasons:

```
team_wins %>%
  group_by(team) %>%
  summarize(n=n()) %>%
  arrange(n)
#> `summarise()` ungrouping output (override with `.groups` argument)
#> # A tibble: 32 x 2
#>    team      n
#>    <chr> <int>
#>  1 HOU      19
#>  2 ARI      22
#>  3 ATL      22
#>  4 BAL      22
#>  5 BUF      22
#>  6 CAR      22
#>  7 CHI      22
#>  8 CIN      22
#>  9 CLE      22
#> 10 DAL      22
#> # … with 22 more rows

# HOU has 3 fewer seasons because it didn't exist from 1999 through 2001, which is fine, and all the other team n
ames have 22 seasons like they should.
```

NOW we can join!

```
#just looking at seattle from seasons 2012-current
data <- team_wins %>%
  left_join(offense, by = c('team' = 'posteam', 'season')) %>%
  left_join(defense, by = c('team' = 'defteam', 'season'))

data %>%
  filter(team == 'SEA' & season >= 2012)
#> # A tibble: 9 x 8
#>   team  season  wins point_diff off_rush_epa off_pass_epa def_rush_epa
#>   <chr>  <int> <dbl>      <int>        <dbl>        <dbl>        <dbl>
#> 1 SEA     2012  11          167     -0.00475       0.213       -0.0738
#> 2 SEA     2013  13          186     -0.101         0.188       -0.126
#> 3 SEA     2014  12          140      0.0216        0.139       -0.231
#> 4 SEA     2015  10          146     -0.102         0.249       -0.148
#> 5 SEA     2016  10.5         62     -0.126         0.102       -0.207
#> 6 SEA     2017   9           34     -0.189         0.0569      -0.122
#> 7 SEA     2018  10           81     -0.0273        0.210       -0.131
#> 8 SEA     2019  11            7     -0.136         0.120       -0.0930
#> 9 SEA     2020  NA           NA     NA            NA           NA
#> # … with 1 more variable: def_pass_epa <dbl>
```

Next we need to create new columns for prior year EPA, and point differential

```
data <- data %>%
  arrange(team, season) %>%
  mutate(
    prior_off_rush_epa = lag(off_rush_epa),
    prior_off_pass_epa = lag(off_pass_epa),
    prior_def_rush_epa = lag(def_rush_epa),
    prior_def_pass_epa = lag(def_pass_epa),
    prior_point_diff = lag(point_diff)
  )

data %>%
  head(5)
#> # A tibble: 5 x 13
#>   team  season  wins point_diff off_rush_epa off_pass_epa def_rush_epa
#>   <chr>  <int> <dbl>      <int>        <dbl>        <dbl>        <dbl>
#> 1 ARI     1999     6       -137       -0.201      -0.162      -0.00937
#> 2 ARI     2000     3       -233       -0.242      -0.0678      0.0286
#> 3 ARI     2001     7        -48       -0.177       0.0740     -0.0689
#> 4 ARI     2002     5       -155       -0.134      -0.0662     -0.0192
#> 5 ARI     2003     4       -227       -0.219      -0.120      -0.0627
#> # … with 6 more variables: def_pass_epa <dbl>, prior_off_rush_epa <dbl>,
#> #   prior_off_pass_epa <dbl>, prior_def_rush_epa <dbl>,
#> #   prior_def_pass_epa <dbl>, prior_point_diff <int>
```

Correlations and regressions

```
data %>%
  select(-team, -season) %>%
  cor(use="complete.obs") %>%
  round(2)
#>                    wins point_diff off_rush_epa off_pass_epa def_rush_epa
#> wins               1.00       0.92         0.43         0.70        -0.29
#> point_diff         0.92       1.00         0.48         0.76        -0.33
#> off_rush_epa       0.43       0.48         1.00         0.40         0.06
#> off_pass_epa       0.70       0.76         0.40         1.00        -0.02
#> def_rush_epa      -0.29      -0.33         0.06        -0.02         1.00
#> def_pass_epa      -0.57      -0.62        -0.04        -0.10         0.30
#> prior_off_rush_epa  0.23       0.26         0.33         0.22         0.02
#> prior_off_pass_epa  0.29       0.32         0.18         0.46        -0.01
#> prior_def_rush_epa -0.12      -0.14         0.03        -0.04         0.26
#> prior_def_pass_epa -0.17      -0.20        -0.07        -0.05         0.06
#> prior_point_diff    0.36       0.41         0.22         0.36        -0.09
#>                    def_pass_epa prior_off_rush_epa prior_off_pass_epa
#> wins                      -0.57               0.23               0.29
#> point_diff                -0.62               0.26               0.32
#> off_rush_epa              -0.04               0.33               0.18
#> off_pass_epa              -0.10               0.22               0.46
#> def_rush_epa               0.30               0.02              -0.01
#> def_pass_epa               1.00              -0.10               0.00
#> prior_off_rush_epa        -0.10               1.00               0.40
#> prior_off_pass_epa         0.00               0.40               1.00
#> prior_def_rush_epa         0.14               0.05              -0.02
#> prior_def_pass_epa         0.27              -0.01              -0.09
#> prior_point_diff          -0.19               0.47               0.76
#>                    prior_def_rush_epa prior_def_pass_epa prior_point_diff
#> wins                            -0.12              -0.17             0.36
#> point_diff                      -0.14              -0.20             0.41
#> off_rush_epa                     0.03              -0.07             0.22
#> off_pass_epa                    -0.04              -0.05             0.36
#> def_rush_epa                     0.26               0.06            -0.09
#> def_pass_epa                     0.14               0.27            -0.19
#> prior_off_rush_epa               0.05              -0.01             0.47
#> prior_off_pass_epa              -0.02              -0.09             0.76
#> prior_def_rush_epa               1.00               0.31            -0.35
#> prior_def_pass_epa               0.31               1.00            -0.60
#> prior_point_diff                -0.35              -0.60             1.00
```

We've run the correlation on this dataframe, removing missing values, and then rounding to 2 digits. Not surprisingly, we see that wins in the current season are more strongly related to passing offense EPA than rushing EPA or defense EPA, and prior offense carries more predictive power than prior defense. Pass offense is more stable year to year (`0.46`) than rush offense (`0.33`), pass defense (`0.27`), or rush defense (`0.26`)

Let's check what this looks like since 2009 relative to earlier seasons:

```
message("2009 through 2019")
#> 2009 through 2019
data %>%
  filter(season >= 2009) %>%
  select(wins, point_diff, off_pass_epa, off_rush_epa, prior_point_diff, prior_off_pass_epa, prior_off_rush_epa)
 %>%
  cor(use="complete.obs") %>%
  round(2)
#>                    wins point_diff off_pass_epa off_rush_epa prior_point_diff
#> wins               1.00       0.91         0.73         0.40             0.43
#> point_diff         0.91       1.00         0.79         0.47             0.44
#> off_pass_epa       0.73       0.79         1.00         0.37             0.39
#> off_rush_epa       0.40       0.47         0.37         1.00             0.19
#> prior_point_diff   0.43       0.44         0.39         0.19             1.00
#> prior_off_pass_epa 0.34       0.36         0.45         0.11             0.78
#> prior_off_rush_epa 0.24       0.24         0.16         0.24             0.45
#>                    prior_off_pass_epa prior_off_rush_epa
#> wins                             0.34               0.24
#> point_diff                       0.36               0.24
#> off_pass_epa                     0.45               0.16
#> off_rush_epa                     0.11               0.24
#> prior_point_diff                 0.78               0.45
#> prior_off_pass_epa               1.00               0.35
#> prior_off_rush_epa               0.35               1.00
```

```
message("1999 through 2008")
#> 1999 through 2008
data %>%
  filter(season < 2009) %>%
  select(wins, point_diff, off_pass_epa, off_rush_epa, prior_point_diff, prior_off_pass_epa, prior_off_rush_epa)
 %>%
  cor(use="complete.obs") %>%
  round(2)
#>                    wins point_diff off_pass_epa off_rush_epa prior_point_diff
#> wins               1.00       0.92         0.68         0.47             0.28
#> point_diff         0.92       1.00         0.73         0.51             0.36
#> off_pass_epa       0.68       0.73         1.00         0.47             0.34
#> off_rush_epa       0.47       0.51         0.47         1.00             0.26
#> prior_point_diff   0.28       0.36         0.34         0.26             1.00
#> prior_off_pass_epa 0.23       0.28         0.45         0.30             0.74
#> prior_off_rush_epa 0.23       0.28         0.30         0.42             0.50
#>                    prior_off_pass_epa prior_off_rush_epa
#> wins                             0.23               0.23
#> point_diff                       0.28               0.28
#> off_pass_epa                     0.45               0.30
#> off_rush_epa                     0.30               0.42
#> prior_point_diff                 0.74               0.50
#> prior_off_pass_epa               1.00               0.48
#> prior_off_rush_epa               0.48               1.00
```

!So in the more recent period, passing offense has become slightly more stable but more predictive of following-year success, while at the same time rushing offense has become substantially less stable and less predictive of future team success.!

basic regression of wins on prior offense and defense EPA/play

(we should only look at this more recent period to fit our model since it's more relevant for 2020)

```
data <- data %>% filter(season >= 2009)

fit <- lm(wins ~ prior_off_pass_epa  + prior_off_rush_epa + prior_def_pass_epa + prior_def_rush_epa, data = data)

summary(fit)
#> 
#> Call:
#> lm(formula = wins ~ prior_off_pass_epa + prior_off_rush_epa + 
#>     prior_def_pass_epa + prior_def_rush_epa, data = data)
#> 
#> Residuals:
#>     Min      1Q  Median      3Q     Max 
#> -7.7207 -1.8625  0.1089  2.1744  7.1073 
#> 
#> Coefficients:
#>                    Estimate Std. Error t value             Pr(>|t|)    
#> (Intercept)          7.8970     0.3997  19.759 < 0.0000000000000002 ***
#> prior_off_pass_epa   6.4875     1.2923   5.020          0.000000827 ***
#> prior_off_rush_epa   6.4348     2.3447   2.744              0.00638 ** 
#> prior_def_pass_epa  -3.5530     1.7315  -2.052              0.04093 *  
#> prior_def_rush_epa  -6.1005     2.4360  -2.504              0.01273 *  
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#> 
#> Residual standard error: 2.823 on 347 degrees of freedom
#>   (32 observations deleted due to missingness)
#> Multiple R-squared:  0.1684, Adjusted R-squared:  0.1588 
#> F-statistic: 17.56 on 4 and 347 DF,  p-value: 0.0000000000003874

#pretty surprised passing offense isn't higher here. How does this compare to simply using point differential?
```

```
fit2 <- lm(wins ~ prior_point_diff, data = data)

summary(fit2)
#> 
#> Call:
#> lm(formula = wins ~ prior_point_diff, data = data)
#> 
#> Residuals:
#>     Min      1Q  Median      3Q     Max 
#> -7.1042 -1.8347  0.1688  2.0713  7.4547 
#> 
#> Coefficients:
#>                  Estimate Std. Error t value            Pr(>|t|)    
#> (Intercept)      8.000000   0.148515  53.867 <0.0000000000000002 ***
#> prior_point_diff 0.012990   0.001468   8.847 <0.0000000000000002 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#> 
#> Residual standard error: 2.786 on 350 degrees of freedom
#>   (32 observations deleted due to missingness)
#> Multiple R-squared:  0.1827, Adjusted R-squared:  0.1804 
#> F-statistic: 78.26 on 1 and 350 DF,  p-value: < 0.00000000000000022

#R2 is somewhat higher for just point differential. This isn't surprising as we've thrown away special teams play
s and haven't attempted to make any adjustments for things like fumble luck that we know can improve EPA's predic
tive power.
```

Predictions - predictions from the EPA model:

```
preds <- predict(fit, data %>% filter(season == 2020)) %>%
  #was just a vector, need a tibble to bind
  as_tibble() %>%
  #make the column name make sense
  rename(prediction = value) %>%
  round(1) %>%
  #get names
  bind_cols(
    data %>% filter(season == 2020) %>% select(team)
  )

preds %>%
  arrange(-prediction) %>%
  head(20)
#> # A tibble: 20 x 2
#>    prediction team
#>         <dbl> <chr>
#>  1       11.4 BAL
#>  2       10.2 SF
#>  3        9.7 NE
#>  4        9.7 NO
#>  5        9.6 DAL
#>  6        9.4 MIN
#>  7        9.2 TEN
#>  8        9.1 KC
#>  9        8.8 TB
#> 10        8.7 IND
#> 11        8.7 PHI
#> 12        8.6 GB
#> 13        8.4 HOU
#> 14        8.2 BUF
#> 15        8.1 ARI
#> 16        8.1 SEA
#> 17        8   LA
#> 18        7.9 LAC
#> 19        7.9 LV
#> 20        7.6 ATL
```

And if we just used simple point differential to predict:

```r
preds2 <- predict(fit2, data %>% filter(season == 2020)) %>%
  #was just a vector, need a tibble to bind
  as_tibble() %>%
  #make the column name make sense
  rename(prediction = value) %>%
  round(1) %>%
  #get names
  bind_cols(
    data %>% filter(season == 2020) %>% select(team)
  )

preds2 %>%
  arrange(-prediction) %>%
  head(20)
#> # A tibble: 20 x 2
#>    prediction team
#>         <dbl> <chr>
#>  1       11.2 BAL
#>  2       10.5 NE
#>  3       10.2 SF
#>  4        9.9 KC
#>  5        9.5 DAL
#>  6        9.5 NO
#>  7        9.4 MIN
#>  8        8.9 TEN
#>  9        8.8 GB
#> 10        8.7 BUF
#> 11        8.4 LA
#> 12        8.4 PHI
#> 13        8.1 SEA
#> 14        8.1 TB
#> 15        7.9 HOU
#> 16        7.9 LAC
#> 17        7.8 ATL
#> 18        7.8 CHI
#> 19        7.8 IND
#> 20        7.8 PIT
```

We get pretty similar results, but this model doesnt account for roster changes (TB12 no longer being with the patriots, etc.) and does not incorporate schedule

---

More data sources

- Lee Sharpe: Draft Picks, Draft Values, Games, Logos, Rosters, Standings (https://github.com/leesharpe/nfldata/blob/master/DATASETS.md)
- greerre: how to get .csv file of weather & stadium data from PFR in python (https://github.com/greerre/pfr_metadata_pull)
- Parker Fleming: Introduction to College Football Data with R and cfbscrapR (https://gist.github.com/spfleming/2527a6ca2b940af2a8aa1fee9320171d)