Set-Up Distance at Snap

Nate Rowan

10/24/2020

Objective

In this document, my goal is to see how tightly defensive backs line up against recievers at the snap, and whether this affects how well they cover routes. Some questions to consider:

- see which defensive backs play tight/soft most frequently, and which teams play tight/soft most frequently
- is there a certain "type" of DB that plays tight/soft more often. Perhaps slower DB's play softer more often so they don't get burned?
- see if there is a relationship between a team's average "softness" and their performance against the pass
- under what situations do teams play softer or tighter coverage more frequently?
- under what situations is it more favorable to play tight/soft coverage?
- relationship between epa/play and tightness
- does relationship between tightness and play success change depending on type of route (go route vs slant)
- does tightness vary for teams that play man vs zone
- does the success of tightness/softness vary for teams that play different coverage schemes

```r
### Load in packages
library(tidyverse)
```

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

```
## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.3     v dplyr   1.0.2
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
```

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

```r
library(gganimate)
```

```
## Warning: package 'gganimate' was built under R version 4.0.3
```

```r
library(janitor)
```

```
##
## Attaching package: 'janitor'
```

```
## The following objects are masked from 'package:stats':
##
##     chisq.test, fisher.test
```

```r
library(lubridate)
```

```
##
```

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

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

First, we read in the data. Only using week 1 tracking data so my laptop doesn't explode

```
### Read in Big Data Bowl data
plays_data <- read_csv(here::here("datasets/plays.csv"))
```

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   playDescription = col_character(),
##   possessionTeam = col_character(),
##   playType = col_character(),
##   yardlineSide = col_character(),
##   offenseFormation = col_character(),
##   personnelO = col_character(),
##   personnelD = col_character(),
##   typeDropback = col_character(),
##   gameClock = col_time(format = ""),
##   penaltyCodes = col_character(),
##   penaltyJerseyNumbers = col_character(),
##   passResult = col_character(),
##   isDefensivePI = col_logical()
## )
```

```
## See spec(...) for full column specifications.
```

```
players_data <- read_csv(here::here("datasets/players.csv"))
```

```
## Parsed with column specification:
## cols(
##   nflId = col_double(),
##   height = col_character(),
##   weight = col_double(),
##   birthDate = col_character(),
##   collegeName = col_character(),
##   position = col_character(),
##   displayName = col_character()
## )
```

```
games_data <- read_csv(here::here("datasets/games.csv"))
```

```
## Parsed with column specification:
## cols(
##   gameId = col_double(),
##   gameDate = col_character(),
##   gameTimeEastern = col_time(format = ""),
##   homeTeamAbbr = col_character(),
##   visitorTeamAbbr = col_character(),
##   week = col_double()
## )
```

```
week1_tracking_data <- read_csv(here::here("datasets/week1.csv"))
```

```
## Parsed with column specification:
## cols(
##   time = col_datetime(format = ""),
##   x = col_double(),
##   y = col_double(),
##   s = col_double(),
##   a = col_double(),
##   dis = col_double(),
##   o = col_double(),
##   dir = col_double(),
##   event = col_character(),
##   nflId = col_double(),
##   displayName = col_character(),
##   jerseyNumber = col_double(),
##   position = col_character(),
##   frameId = col_double(),
##   team = col_character(),
##   gameId = col_double(),
##   playId = col_double(),
##   playDirection = col_character(),
##   route = col_character()
## )
```

Next, cleaning up data. I'll probably add to this as I come across random things

```r
### Clean data

### Standardize so that offense always goes in same direction
week1_tracking_data <- week1_tracking_data %>%
  mutate(x = ifelse(playDirection == "left", 120-x, x),
         y = ifelse(playDirection == "left", 160/3 - y, y))

### Changes names from ugly camelCase to snake_case
plays_data <- janitor::clean_names(plays_data)
players_data <- janitor::clean_names(players_data)
games_data <- janitor::clean_names(games_data)
week1_tracking_data <- janitor::clean_names(week1_tracking_data)

### Change birthdates to a datetime object
players_data <- players_data %>%
  mutate(birth_date = lubridate::parse_date_time(birth_date,
                                                 orders = c("y-m-d", "m/d/y")))
```

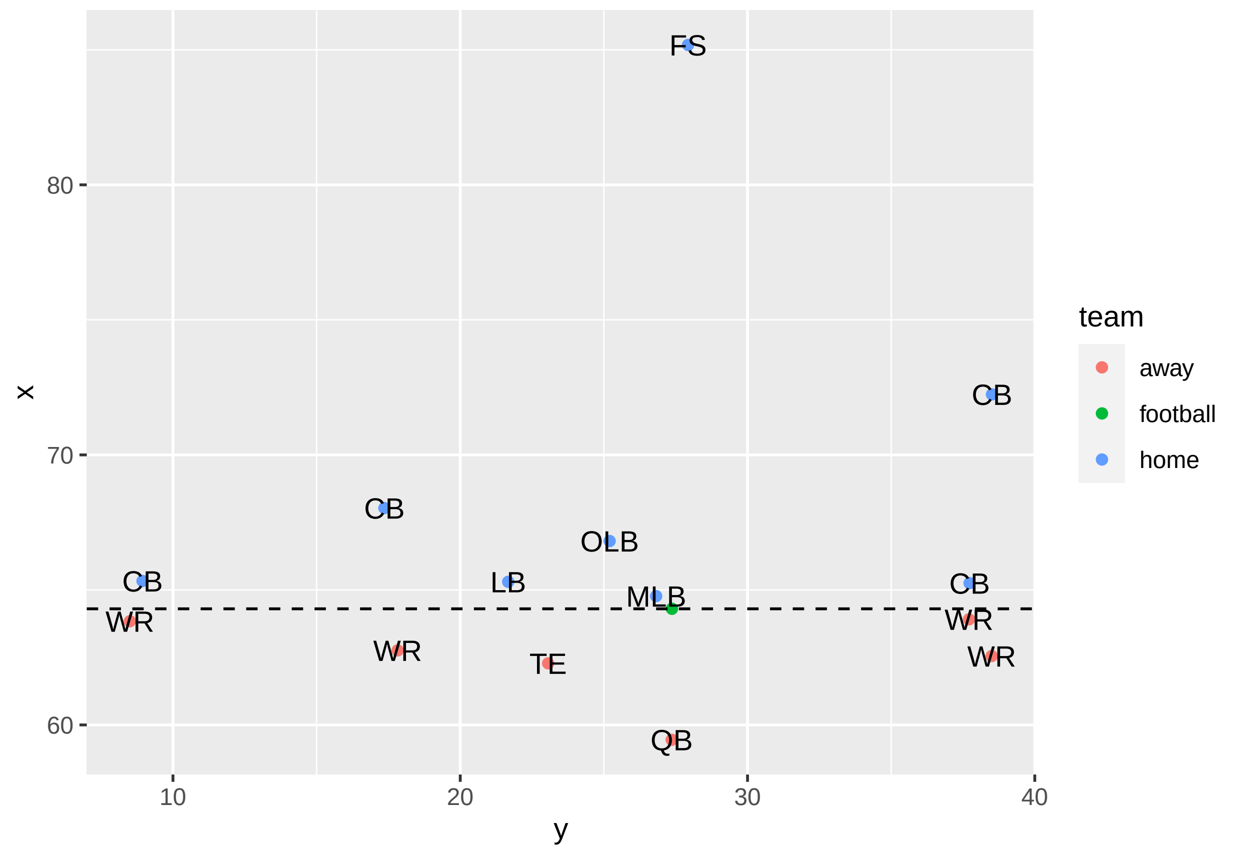
Join the games, plays, and tracking data together

```r
### Joining data together

### Join plays and games data
plays_data <- left_join(plays_data, games_data, by = "game_id")

### Filter so I just have week 1 plays data
week1_plays_data <- plays_data %>%
  filter(week == 1)

### Join plays and tracking data
plays_tracking_data <- left_join(week1_plays_data,
                                     week1_tracking_data,
                                     by = c("game_id", "play_id"))

### Nest the tracking data into a list of dataframes
plays_tracking_data <- plays_tracking_data %>%
  nest(tracking_data = x:route)
```

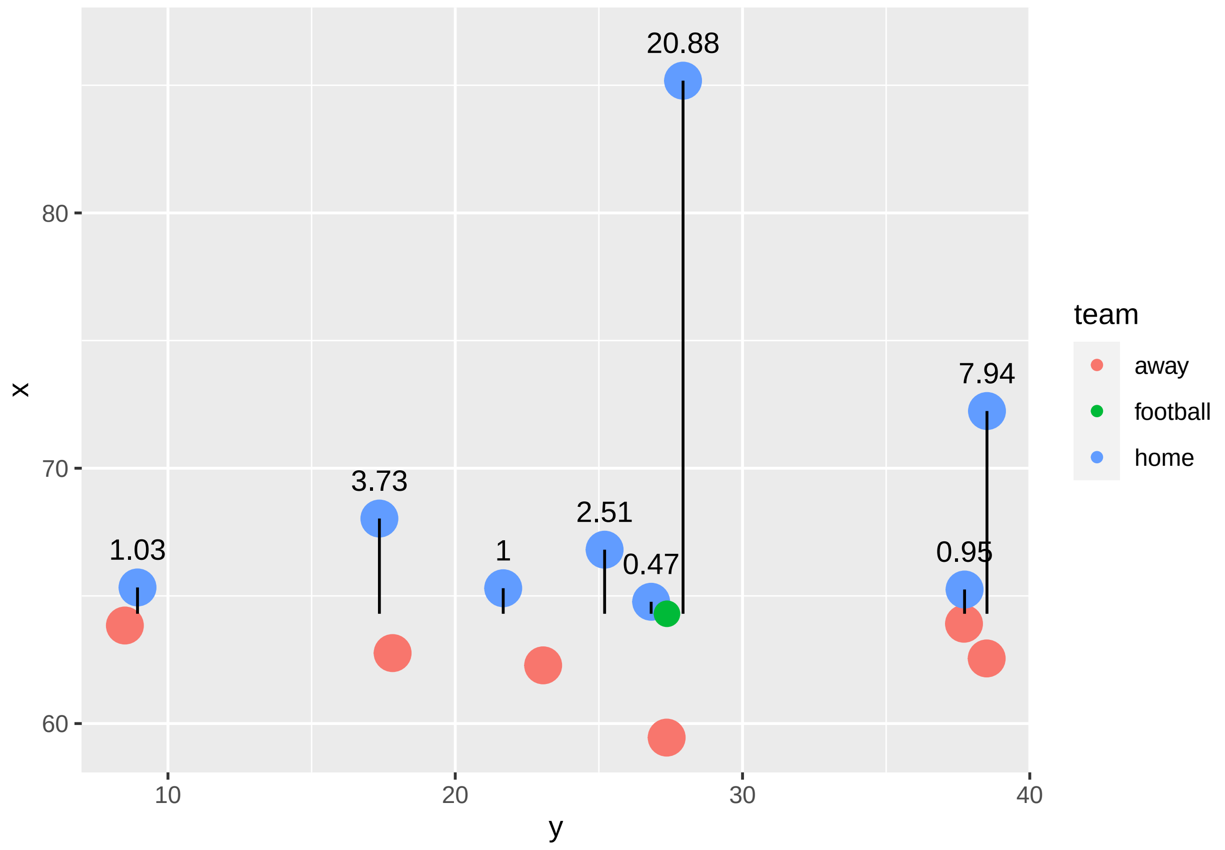
Now, the meat and potatoes. Calculating the distance of each defender from the line of scrimmage for a random play in week 1.

```r
### Calculate distance from line of scrimmage/reciever at snap

### First, we are going to do it for a random play to get a feel
### for how to do the calculation

### Set seed for reproducibility
set.seed(123)

### Pull out the game ID and play ID of a random play
ex_ids <-  plays_tracking_data %>%
  filter(pass_result == "C") %>%
  slice_sample(n = 1) %>%
  select(game_id, play_id)

### Pull the game ID of the random play
example_game_id <- ex_ids %>%
  select(game_id) %>%
  flatten_dbl()

### Pull the play ID of the random play
example_play_id <- ex_ids %>%
  select(play_id) %>%
  flatten_dbl()

### Get the tracking data of the random play
example_play <- plays_tracking_data %>%
  filter(play_id == example_play_id, game_id == example_game_id) %>%
  unnest(cols = tracking_data)

### Great, now we have our random play selected. I'll start of by plotting
### the players' position at the snap

### Pull the tracking data at the snap
snap_data <- example_play %>%
  filter(event == "ball_snap")

### Pull out the location of the football
football_location <- snap_data %>%
  filter(display_name == "Football") %>%
  select(x, y)

### Pull the "x" location of the football
football_x <- football_location %>%
  select(x) %>%
  flatten_dbl()

### Pull the "y" location of the football
football_y <- football_location %>%
  select(y) %>%
  flatten_dbl()

### Add the x and y location of the football as a column to the
### dataframe. This will be used when we calculate the distance between
### each player and the line of scrimmage
snap_data <- snap_data %>%
  add_column(
    football_x = football_x,
    football_y = football_y
  )

### Plot of all players and their position, with the dotted line
### representing the line of scrimmage
snap_data %>%
  ggplot(aes(x = y, y = x)) +
    geom_point(aes(color = team)) +
    geom_hline(linetype = "dashed", yintercept = football_x) +
    geom_text(aes(label = position))
```

```
## Warning: Removed 1 rows containing missing values (geom_text).
```



```
### Cool. Now let's calculate how far each player is from the
### line of scrimmage
snap_data <- snap_data %>%
  mutate(
    xdist_ball = round(x - football_x, 2),
    ydist_ball = round(y - football_y, 2)
  )

### Plot a line segment with the vertical distance of each
### defender from the line of scrimmage.
### One thing that will have to be changed: in the geom_segment
### function, I specify that I only want to plot a segment for the
### home team, since that is the defense in this play. However, home
### won't always be defense, so I'll need to generalize this. Probably
### by adding a flag to each player indicating if they are on
### offense or defense
### lol that was a long comment
snap_data %>%
  ggplot(aes(x = y, y = x)) +
    geom_point(aes(color = team, size = team)) +
    geom_segment(
      data = filter(snap_data, team == "home"),
      aes(x = y, y = x, xend = y, yend = football_x)
    ) +
    geom_text(
      data = filter(snap_data, team == "home"),
      aes(label = xdist_ball), nudge_y = 1.5
    ) +
    scale_size_manual(values = c(6, 4, 6), guide = FALSE)
```



The next step is going to be to take everything that I did up there, and throw it into functions so that this can be done for all the plays in the dataset.