STAT 597 Final Project

Supplementing NFL Play-By-Play Data With In-Game Weather Data

Nick Romano

Motivation

The average NFL (National Football League) team is valued at approximately $2.57B. With 32 teams in the league, the collective value of the league approaches $100B. It goes without saying that there is a lot of money on the line when it comes to the sport. With only 16 games in a season, teams are fighting for a compeitive edge with the hope of making the Superbowl. Football is still well behind other sports like baseball and basketball when it comes to advanced analytics however there has been a noticable adoption of data collection and analytics in recent years. Teams are collecting more data than ever, tracking in game position of a player, player attributes, and box score statistics among much else. While some data is available to the public, much is kept proprietary.

Perhaps the greatest offshoot of the NFL has been Famtasy Football. Fantasy Football is a competition in which participants select imaginary teams from among the players in a league and score points according to the actual performance of their players. Many Fantasy Football leagues have rather large cash prizes associated with them. It is estimated that nearly 32 million Americans play Fantasy Football, with an estimated value of over $70B per year in tangible and intangible activity [5]. Fans who participate in Fantasy Football have (collectively) just as much money relying on the sport as the league itself! Those fans are fighting for a competitive edge to win their Fantasy Football leagues. In a competition based on statistics, advanced analytics become all the more useful. Such analytics, however, require the proper data to be available.

An owner (that is, a Fan participating in Fantasy Football) may choose to start or sit a player in a particular matchup based on that player's in-game performance in similar matchups or under similar conditions. For instance, I may choose to sit my usual QB in favor of another

QB for a game which is expecting cold weather because that QB has not performed well in cold weather historically. In order to make such a decision, the typical owner would have to spend hours (or days depending on their level of commitment) researching player performance. Aside from fans participating in Fantasy football, if a researcher or sports analyst ever desired to analyze the effect of weather at the game level or even at the play level, he/she would not have a clean dataset for doing so.

The NFL does provide access to game data through API endpoints but the amount of data retrievable through this API is quite vast and requires a lot of work to wrangle into a useable format. However, thanks to a group of researchers at Carnegie Mellon University, there is an R package (`nflscrapR`) which scrapes, parses, and outputs datasets retrieved using the NFL's API. Using `nflscrapR`, I can retrieve play-by-play data for any game in any season since 2009, but this data does not include weather data. For this project, I have used the methods we learned in class to scrape sub-hourly weather data from Weather Underground to create a dataset representing the exact weather conditions on each play of the game. While this project only focuses on a single team and a single season, the code produced for this project could be applied for all teams and seasons.

Data Description

Without much effort, the NFL API provides basic in-game play information, listed below:

"Drive" - Drive number
"sp" - Whether the play resulted in a score (any kind of score)
"qtr" - Quarter of Game
"down" - Down of the given play
"time" - Time at start of play
"yrdln" - Between 0 and 50
"ydstogo" - Yards to go for a first down
"ydsnet" - Total yards gained on a given drive
"posteam" - The team on offense
"AirYards" - Number of yards the ball was thrown in the air
"YardsAfterCatch" - Number of yards receiver gained after catch

"QBHit" - Binary: 1 if the QB was knocked to the ground else 0

"desc" - A detailed description of what occured during the play

A lot of additional data features can be extracted from the description field (desc). Here is an example entry in the description field: "(6:53) (Shotgun) N.Foles pass short left to Z.Ertz to ATL 8 for 10 yards (D.Jones; D.Campbell)". In this example, we can determine the remaining time in the quarter, whether the play was from shotgun formation, whether the play was a pass or not, who was involved with the play, and the result of the play, among others.The value of the `nflscrapR` package is its ability to parse the description field to extract potentially 173 additional features detailing the events of the particular play. The package also incorporates calculations for the expected points and win probability for each play.

The weather data used to supplement the play-by-play data is scrapped from Weather Underground for the date of the game. Weather observations are available at sub-hour granularity and include the following features:

"time" - time of observation

"temp" - temperature in F

"wind_dir" - wind direction (e.g. SW)

"wind_speed" - wind speed in mph

"wind_gust" - max. wind gust observed since last observation in mph

"precip" - precipitation observed since last observation in inches "condition" - weather condition (e.g. cloudy, sunny, rainy)

Approach & Challenges

My goal with this project was to complete a proof of concept on a subset of the play-by-play data showing how weather data can be warngled and merged with the play-by-play data providing researchers, sports analysts, or football fans alike play-by-play weather data that can be incorporated into whatever analyses they desire. With the code of this POC in place, it can be extended to all games and all seasons dating back to 2009.

To join the two data sets, special consideration had to be taken. The weather and play-by-play data sets do not share an identical attribute that can be used as a key, however, the two data

sets do share a common attribute: time. The goal was to join the rows of weather data to rows of nfl data on the nearest time of play. This presented several challenges:

1. Only the time of the play *in the game* was available.

   For each play, we have the time of the play in the game (e.g. 6:54 in the second quarter) but not the actual time of day (e.g. 1:47pm). In order to match a weather observation with a play we need the actual time of the day. The approximate time of day that the play took place can be calculated using the game start time and the time elapsed in the game with each play. In approximating the time of day that the play took place, I had to make the following assumptions:

   a. Play frequency is consistent between first and second halves
   b. Each half takes approximately the same amount of time (same number of commerical breaks, game stops, etc.).
   c. Halftime lasts for 15 minutes (that is the time between end of first half and beginning of second is 15 minutes)

2. NFL API does not provide the time of game start or game end.

   As mentioned in (1), to get the time that the play occured, I needed the game start time. I discovered that it is quite difficult to find the actual time the game started. Past schedules are indicators of when the game was scheduled to start but sometimes games do not start on time. Using the scheduled start time would introduce some uncertainty in the approximated time that the play took place. The NFL does provide a Game Report in PDF form which is a breakdown of the entire game. This Game Report includes the exact time of kickoff as well as the duration of the game. For this project, I manually extracted the game start times and duration. This presents a limitation of the approach presented herein.

3. Weather observation times are at a different granularity than play time.

   Plays can occur anywhere from 30s to 5+ minutes apart from one another but weather observations are sub-hourly, occuring anywhere from 5 to 30 minutes apart from one another. When joining the datasets, we want to use the time of

the play to match with the closest time weather observation. This was one of the more challenging parts of the project because none of the packages included in the `tidyverse` a fuzzy join. Instead I found it easier to turn to SQL to perform the join. More on this in the next section.

Implementation

Approximating the Time of Play

To approximate the time each play took place, four things are required: (1) start time of first half, (2) start time of the second half, (3) game duration, and (4) seconds remaining in the half when the play occured. Using the ECDF of seconds remaining (i.e. the percentage of time remaining in the half) and the duration of each half, an approximate time for each play can be determined. The pseudo code below describes this calculation for the first half. The table following presents a window of the play-by-play dataset with the approximated time of play.

```
game_length       # game duration including game stopage
halftime          # length of halftime
half1_start       # time at start of first half
half1_seconds_rem # seconds remaining at time of play in first half

# determin approximate duration of one half in seconds
half_length <- (game_length - halftime)/2

# determine percentile rank of half_seconds_remaining
ECDF <- ecdf(half1_seconds_rem)(half1_seconds_rem)

# elapsed time (in seconds) from game start for each play
elaspsed_time_sec = half_length - (half_length*ECDF)

# get approximate time of occurence for each play
times <- (half_start + elaspsed_time_sec)
```

Table 1: Sample Play-by-Play Data (Windowed)

play_id	time	posteam	defteam	drive	down	ydstogo	yrdln
52	09:05 PM	ATL	PHI	1	1	10	ATL 25
75	09:05 PM	ATL	PHI	1	1	15	ATL 20

play_id	time	posteam	defteam	drive	down	ydstogo	yrdln
104	09:08 PM	ATL	PHI	1	2	5	ATL 30
125	09:09 PM	ATL	PHI	1	1	10	ATL 41
146	09:10 PM	ATL	PHI	1	1	10	PHI 39
168	09:11 PM	ATL	PHI	1	2	10	PHI 39
190	09:12 PM	ATL	PHI	1	3	10	PHI 39
214	09:13 PM	ATL	PHI	1	1	6	PHI 6
235	09:14 PM	ATL	PHI	1	2	1	PHI 1
256	09:15 PM	ATL	PHI	1	3	1	PHI 1
278	09:16 PM	ATL	PHI	1	4	1	PHI 1
299	09:17 PM	PHI	ATL	2	1	10	PHI 2
320	09:18 PM	PHI	ATL	2	2	8	PHI 4
344	09:19 PM	PHI	ATL	2	3	4	PHI 8

Scraping Weather Data From Weather Underground

Weather Underground provides historical weather observations at sub-hour granularity. I used the `rvest` package to scrape weather data from www.wunderground.com/history/ for the date of the game. The site displays a table at the bottom of the page that contains the sub-hourly weather observations. The weather observations include: time, temperature, wind direction, wind speed, max. wind gust, precipitation, and observed condition. After retrieving the raw html from the site using the `read_html()` function I was able to parse the html table into a data frame using `html_table()`. The table still required additional wrangling to get the data into usable format. I performed the following steps to obtain the data in the format presented in Table 2:

1. Remove all non-alphanumeric characters using `str_replace_all()` and the regular expression `[^0-9a-zA-Z ]+`.
2. For `temp`, `wind_speed`, `wind_gust`, `precip` columns: remove all non-numeric characters using `str_replace_all()` and the regular expression `[^0-9]` and convert to numeric tyoe using `as.numeric()`. This removes any text for

units (e.g. F or mph) before converting to numeric type.

3. Convert the time column to POSIXct objects using `as.POSIXct()` and `format=%Y-%m-%d %I%M %p` to format time as 24-hour time. I formatted the time this way to match the format of the time in the play-by-play table. Also, formatting time as a POSIXct object makes it easier to perform aritmetic with the time (useful when joining the data sets on time).

Table 2: Sample Weather Observations Data

time	temp	wind_dir	wind_speed	wind_gust	precip	condition
08:22 PM	79	SSW	5	0	0	Cloudy
08:35 PM	79	SSW	5	0	0	Cloudy
08:54 PM	79	SSW	9	0	0	Cloudy
09:54 PM	80	SSW	6	0	0	Cloudy
10:50 PM	79	S	7	0	0	Mostly Cloudy
10:54 PM	79	SSW	6	0	0	Mostly Cloudy
11:37 PM	78	S	6	0	0	Fair
11:54 PM	78	S	6	0	0	Fair
12:42 AM	78	SSW	6	0	0	Cloudy
12:48 AM	79	SSW	5	0	0	Cloudy
12:54 AM	78	SSW	7	0	0	Cloudy
01:04 AM	79	SSW	7	0	0	Cloudy
01:54 AM	78	SSW	7	0	0	Mostly Cloudy
02:38 AM	78	SSW	5	0	0	Cloudy
02:54 AM	78	SSW	5	0	0	Cloudy

Merging the Data Sets

The objective of this project was to wrangle, clean, and finally combine the NFL Play-by-Play and Weather data sets to produce a supplemented NFL Play-by-Play data set that includes weather data for each play. The challenge with combining the two data sets is that there is no

column that can be used to join on. Rather, the goal is to left join the Weather data on to the NFL play data by the closest matching time of observation in the weather table to the time in the NFL play data. To do this I found the `sqldf` package to be quite useful. With this package I can write a query string in SQL syntax format to perform the join operation on two data frames. Below is a pseudo code representation of the SQL query I used. Using SQL I was able to perform a subquery to find the row in the weather table with the smallest difference of time compared to the NFL play times (i.e. the weather observation with the time closest to the time in the NFL play table). I then left joined the weather table on the NFL table where the time differences were equal. The final result is displayed in Table 3.

```
SELECT
    s.*
    w.*,
    (SELECT
        ABS(s.time-w.time) as diff
     FROM
        weather w
     ORDER BY
        diff
     LIMIT
        1
    ) AS time_diff
FROM
    s18_updated s
LEFT JOIN
    weather w
ON
    time_diff = ABS(s.time - w.time)
```

Table 3: Random Sample (N=15) of Combined Dataset

play_id	play_time	weather_time	condition	temp	wind_dir	wind_speed
2208	10:35 PM	10:50 PM	Mostly Cloudy	79	S	7
2948	11:19 PM	11:37 PM	Fair	78	S	6
1662	10:13 PM	09:54 PM	Cloudy	80	SSW	6
52	09:05 PM	08:54 PM	Cloudy	79	SSW	9
75	09:05 PM	08:54 PM	Cloudy	79	SSW	9

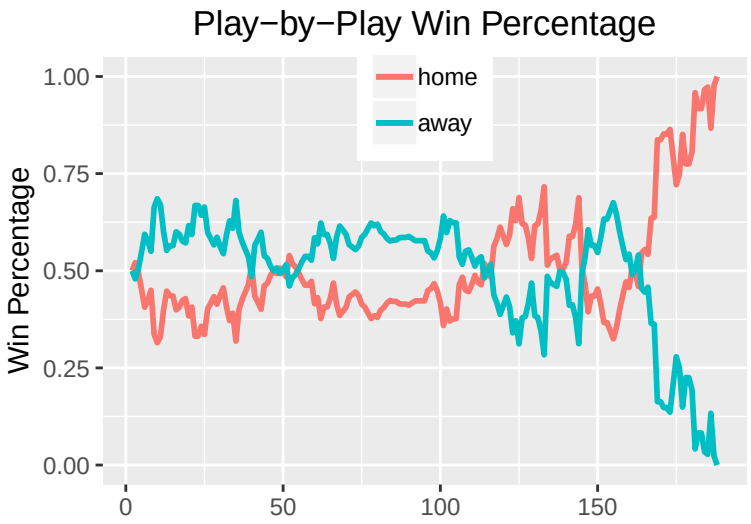
play_id	play_time	weather_time	condition	temp	wind_dir	wind_speed
146	09:10 PM	08:54 PM	Cloudy	79	SSW	9
1850	10:19 PM	09:54 PM	Cloudy	80	SSW	6
4395	12:19 AM	11:54 PM	Fair	78	S	6
2225	10:52 PM	10:50 PM	Mostly Cloudy	79	S	7
1473	10:05 PM	09:54 PM	Cloudy	80	SSW	6
545	09:27 PM	09:54 PM	Cloudy	80	SSW	6
2764	11:11 PM	10:54 PM	Mostly Cloudy	79	SSW	6
1568	10:09 PM	09:54 PM	Cloudy	80	SSW	6
1628	10:12 PM	09:54 PM	Cloudy	80	SSW	6
932	09:42 PM	09:54 PM	Cloudy	80	SSW	6

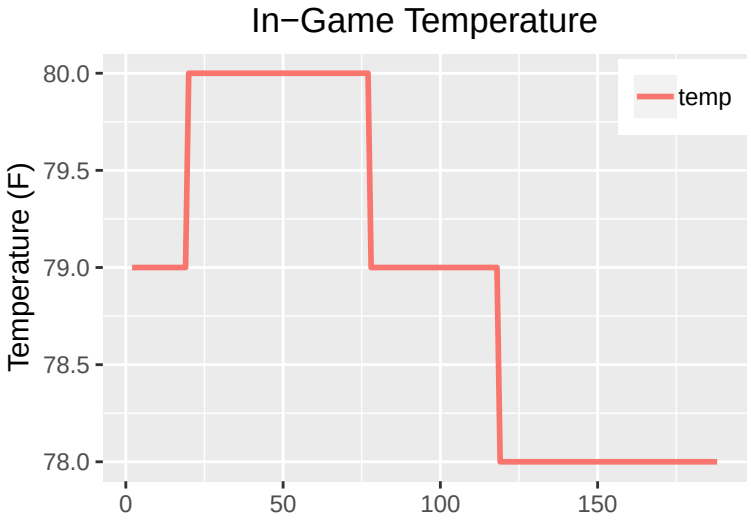
Uses Cases

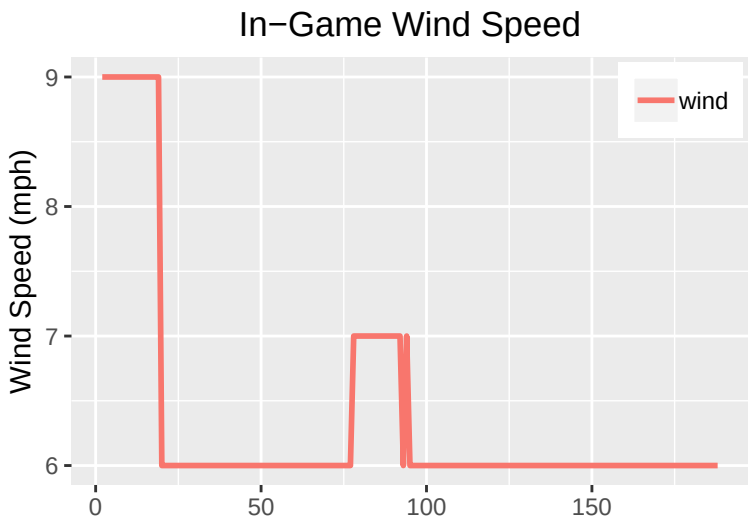
With the combined data set we now have play-by-play data for each game that includes the weather conditions at the time of each play. Researchers, sports analysts, or fans are now able to perform any analysis they desire involving the influence of weather on in game performance. I don't go into any analysis here. Instead I've listed a few potentially interesting questions that this data set can be used to answer.

1. Does in-game weather have an effect on win percentage?
2. Does in-game weather influence the number of first downs a team has? How about fourth downs?
3. When a game is played in rainy weather is a team more likely to run or pass? How about in cold weather?
4. How does in-game wind speed influence pass depth? What about wind direction?
5. In what weather condition does a given team perform best?

Visualizing the Data







References

1. nflscrapR. https://github.com/maksimhorowitz/nflscrapR
2. nflscrapR-data. https://github.com/ryurko/nflscrapR-data
3. NFL Game Center. http://www.nfl.com/liveupdate/game-center/
4. Weather Underground. www.wunderground.com/history/
5. The $70 Bilion Fantasy Football Market. https://www.forbes.com/sites/briangoff/2013/08/20/the-70-billion-fantasy-football-market/#26afeb4c755c

Packages

```
library(knitr)
library(tidyverse)
library(nflscrapR)
library(rvest)
library(sqldf)
```

Load NFL Play-By-Play Data

```
# load play by play data for 2018 regular season
s18 <- read_csv(paste("https://raw.githubusercontent.com/ryurko/nflscrapR-data",
                      "/master/play_by_play_data/regular_season/reg_pbp_2018.csv",
                      sep=""))
```

```r
# get game ids for all Philadelphia Eagles games in the 2018 season
phi_games <- scrape_game_ids(2018, teams = "PHI", weeks = 1)

# game duration in seconds
game_length <- hrs(3) + mns(19)

# datetime of game
game_datetime <- c("2018-09-06 9:05:00 PM")

s18
```

Utility Functions:

```r
# function to convert minutes to seconds
mns <- function(t) {return(t*60)}

# function to convert hours to seconds
hrs <- function(t) {return(t*3600)}

# function for getting the time each play occurred
getTimeOccured <- function(half_start, half_length, half_seconds_rem) {

  # determine percentile rank of half_seconds_remaining
  pctl_rank = ecdf(half_seconds_rem)(half_seconds_rem)

  # elapsed time (in seconds) from game start for each play
  elaspsed_time_sec = half_length-(half_length*pctl_rank)

  # get approximate time of occurence for each play
  times <- (half_start + elaspsed_time_sec)

  return(times)

}

# function to format time as POSIXct
formatTime <- function(t) {

  formatted <- t %>% str_pad(.,7,pad="0") %>%
    lapply(., function(x) c(paste("2018-09-06",x))) %>%
    unlist %>%
    as.POSIXct(., format="%Y-%m-%d %I%M %p", tz="EST")

  return(formatted)
```

```
}
```

Add approximate time of occurence for each play:

```
# length of halftime
halftime    <- mns(15)

# duration of one half in seconds
half_length <- (game_length - halftime)/2
half1_start <- strptime(game_datetime,
                        format = "%Y-%m-%d %I:%M:%S %p", tz = "EST")
half2_start <- half1_start + half_length + halftime

# extract seconds remaining in game, first half, and second half
game_seconds_rem <- s18 %>%
  filter(game_id %in% phi_games$game_id) %>%
  select(play_id, game_half, half_seconds_remaining)
half1_seconds_rem <- game_seconds_rem %>%
  filter(game_half=='Half1') %>%
  .$half_seconds_remaining
half2_seconds_rem <- game_seconds_rem %>%
  filter(game_half=='Half2') %>%
  .$half_seconds_remaining

# using seconds remaining and half start times, get approximate time of occurence
half1_times <- getTimeOccured(half1_start, half_length, half1_seconds_rem)
half2_times <- getTimeOccured(half2_start, half_length, half2_seconds_rem)

names(s18)[21] = "qtr_time_rem"
s18_updated <- s18 %>%
  filter(game_id %in% phi_games$game_id) %>%
  mutate(time=c(half1_times,half2_times)) %>%
  select(game_id, play_id, game_date, time, qtr_time_rem, everything())

#s18_updated
```

Scrape weather data from Wunderground:

```
# get data from website
raw <- "./tmp/weather_20180906.html" %>%
  read_html(encoding = "utf8") %>%
  html_table(fill = TRUE) %>%
  flatten()
```

```
# reformat the table
names(raw)[c(1,2,5,6,7,9,11)] = c("time","temp","wind_dir",
                                  "wind_speed","wind_gust","precip","condition")
weather <- raw %>%
  lapply(., function(x) str_replace_all(x, "[^0-9a-zA-Z ]+","")) %>%
  as_tibble() %>%
  select(time, temp, wind_dir, wind_speed, wind_gust, precip, condition) %>%
  mutate(time = formatTime(time),
         temp = unlist(lapply(temp,
                       function(x) as.numeric(str_replace_all(x, "[^0-9]+","")))),
         wind_speed = unlist(lapply(wind_speed,
                       function(x) as.numeric(str_replace_all(x, "[^0-9]+","")))),
         wind_gust = unlist(lapply(wind_gust,
                       function(x) as.numeric(str_replace_all(x, "[^0-9]+","")))),
         precip = unlist(lapply(precip,
                       function(x) as.numeric(str_replace_all(x, "[^0-9]+",""))))))

weather
```

Join NFL and Weather datasets:

```
d <- sqldf('SELECT
              s.play_id,
              s.time as play_time,
              w.time as weather_time,
              s.home_wp,
              s.away_wp,
              w.condition,
              w.temp,
              w.wind_dir,
              w.wind_speed,
              w.precip,
              (SELECT
                  ABS(s.time-w.time) as diff
               FROM
                  weather w
               ORDER BY diff
               LIMIT 1) AS time_diff
            FROM
              s18_updated s
            LEFT JOIN weather w ON time_diff=ABS(s.time-w.time)',
           method='raw')

d <- d %>% mutate(play_time=as.POSIXct(play_time,origin="1970-01-01", tz="EST"),
```

```
    weather_time=as.POSIXct(weather_time,origin="1970-01-01", tz="EST")) %>%
  select(-time_diff)
```