Fantasy Football Analytics

Isaac T. Petersen

To our daughter, Maisie.

Table of contents

Preface

This is a book in progress—it is incomplete. I will continue to add to and update it as I am able.

Open Access

This is an open-access book. This means that it is freely available for anyone to access.

License

Figure 1 Creative Commons License

The online version of this book is licensed under the Creative Commons Attribution License[1]. In short, you can use my work as long as you cite it.

[1] https://creativecommons.org/licenses/by/4.0/

Citation

The APA-style citation for the book is:

Petersen, I. T. (2024). *Fantasy football analytics*. Version 0.0.1. University of Iowa Libraries. https://github.com/isaactpetersen/Fantasy-Football-Analytics-Textbook. [INSERT DOI LINK]

The BibTeX citation for the book is:

```
@book{petersenFantasyFootballAnalytics,
  title = {Fantasy football analytics},
  author = {Petersen, Isaac T.},
  year = {2024},
  publisher = {{University of Iowa Libraries}},
  note = {Version 0.0.1},
  doi = {INSERT},
  isbn = {INSERT},
 url = {https://github.com/isaactpetersen/Fantasy-Football-Analytics-Textbook}
}
```

Accessibility

I strive to follow principles of accessibility[2] (archived at https://perma.cc/8XJ9-Q6QJ) to make the book content accessible to people with visual impairments and physical disabilities. If there are additional ways I can make the content more accessible, please let me know.

How to Contribute

This is an open-access textbook. My goal is to share data analysis strategies for free! Anyone is welcome to contribute to the project. If you would like to contribute, feel free to open an issue[3] or create a pull request[4] on

[2] https://bookdown.org/yihui/rmarkdown-cookbook/html-accessibility.html
[3] https://github.com/isaactpetersen/Fantasy-Football-Analytics-Textbook/issues
[4] https://github.com/isaactpetersen/Fantasy-Football-Analytics-Textbook/pulls

GitHub. The GitHub repository for the book is located here: https://github.com/isaactpetersen/Fantasy-Football-Analytics-Textbook. If you have data or analysis examples that are you willing to share and include in the book, feel free to contact me.

Acknowledgments

I thank Dr. Benjamin Motz, who provided consultation and many helpful resources based on his fantasy football statistics class. I also thank key members of FantasyFootballAnalytics.net[5], including Val Pinskiy, Andrew Tungate, Dennis Andersen, and Adam Peterson, who helped develop and provide fantasy football-related resources and who helped sharpen my thinking about the topic. I also thank Professor Patrick Carroll, who taught me the value of statistics for answering important questions.

[5] http://fantasyfootballanalytics.net

1

Introduction

1.1 About this Book

How can we use information to make predictions about uncertain events? This book is about empiricism (basing theories on observed data) and judgment, prediction, and decision making in the context of uncertainty. The book provides an introduction to modern analytical techniques used to make informed predictions, test theories, and draw conclusions from a given dataset.

This book was originally written for a undergraduate-level course entitled, "Fantasy Football: Predictive Analytics and Empiricism". The chapters provide an overview of topics that each could have its own class and textbook, such as causal inference[1], factor analysis[2], cluster analysis[3], principal component analysis[4], machine learning[5], cognitive biases[6], modern portfolio theory[7], data visualization[8], simulation[9], etc. The book gives readers an overview of the breadth of the approaches to prediction and empiricism. As a consequence, the book does not cover any one technique or approach in great depth.

1.2 What is Fantasy Football?

Fantasy football is an online game where participants assemble (i.e., "draft") imaginary teams composed of real-life National Football League (NFL) players. In this game, participants compete against their opponents (e.g.,

[1] causal-inference.qmd
[2] factor-analysis.qmd
[3] cluster-analysis.qmd
[4] pca.qmd
[5] machine-learning.qmd
[6] cognitive-bias.qmd
[7] modern-portfolio-theory.qmd
[8] data-visualization.qmd
[9] simulation.qmd

friends/coworkers/classmates), accumulating points based on players' actual statistical performances in games. The goal is to outscore one's opponent each week to win matches and ultimately claim victory in the league.

1.3 Why Focus on Fantasy Football?

I was fortunate to have an excellent instructor who taught me the value of learning statistics to answer interesting and important questions. That is, I do not find statistics intrinsically interesting; rather, I find them interesting because of what they allow me to do. Many students find statistics intimidating in part because of how it is typically taught—with examples like dice rolls and coin flips that are (seemingly irrelevant and) boring to students. My contention is that applied examples are a more effective lens to teach many concepts in psychology and data analysis. It can be more engaging and relatable to learn statistics in the applied context of sports, a domain that is more intuitive to many. Many people play fantasy sports. This book involves applying statistics to a particular domain (football). People actually want to learn statistical principles and methods when they can apply them to interesting questions (e.g., sports). In my opinion [and supported by evidence; Motz (2013)], this is a much more effective way of engaging people and teaching statistics than in the context of abstract coin flips and dice rolls. Fantasy football relies heavily on prediction—trying to predict which players will perform best and selecting them accordingly. In this way, fantasy football provides a plethora of decision making opportunities in the face of uncertainty, and a wealth of data for analyzing these decisions. However, unlike many other applied domains in psychology, fantasy football (1) allows a person to see the accuracy of their predictions on a timely basis and (2) provides a safe environment for friendly competition. Thus, it provides a unique domain to evaluate—and improve—the accuracy of various prediction models.

1.4 Educational Value

Skills in data analysis are highly valuable. This book includes practical and conceptual tools that build a foundation for critical thinking. The book aims to help readers evaluate theory in the light of evidence (and vice versa) and to refine decision making in the context of uncertainty. Readers will learn about the ways that psychological science (and related disciplines) poses questions,

formulates hypotheses, designs studies to test those questions, and interprets the findings, collectively with the aim to answer questions, improve decision making, and solve problems.

Of course, this is not a traditional psychology textbook. However, the book incorporates important psychological concepts, such as cognitive biases in judgment and prediction, etc. In the modern world of big data, research and society need people who know how to make sense of the information around us. Psychology is in a prime position to teach applied statistics to a wide variety of students, most of whom will not have careers as psychologists. Psychology can teach the importance of statistics given humans' cognitive biases. It can also teach about how these biases can influence how people interpret statistics. This book will teach readers the applications of statistics (prediction) and research methods (empiricism) to answer questions they find interesting, while applying scientific and psychological rigor.

1.5 Learning Objectives

This book aims to help readers accomplish the following learning objectives:

- Apply empirical inference and appreciate the value it provides over speculative supposition.
- Ask educated questions when confronted with decisions in the face of uncertainty.
- Understand human decision making, including common heuristics and cognitive biases and how to mitigate them analytically.
- Engage in critical thinking about causality, including devising plausible alternative explanations for observed effects.
- Understand causal inference including confounding, causal pathways, and counterfactuals.
- Think empirically about human behavior and performance.
- Describe the strengths and weaknesses of humans versus computers in prediction scenarios.
- Manipulate and summarize datasets.
- Critically evaluate the strengths and limitations of different statistical models and methodologies used in predicting uncertain events, enhancing their understanding of statistical inference and model selection.
- Use various analytical techniques for predicting the outcome of uncertain events, and for uncovering latent causes of patterns in observed data.
- Interpret findings from various statistical approaches and evaluate the accuracy of predictions.

- Engage in iterative problem-solving processes, refining analytical approaches based on feedback and outcomes, and adapting strategies accordingly.
- Communicate statistical findings and analyses in both written and oral formats, demonstrating proficiency in presenting complex information to diverse audiences.
- Make sense of big data.
- Use practical analytical skills that can be applied in future research and job settings.

1.6 Disclosures

I am the Owner of Fantasy Football Analytics, LLC, which operates https://fantasyfootballanalytics.net.

1.7 Disclaimer

"*This material probably won't win you fantasy football championships. You could take what we learn and apply it to fantasy football and you might become 5 percent more likely to win. Or… Consider the broader relevance of this. You could learn data analysis and figure out ways to apply it to other systems. And you could be making a six-figure salary within the next five years.*" – Benjamin Motz, Ph.D.

2

Intro to Football and Fantasy

This chapter provides a brief primer on (American) football and fantasy football. If you are already familiar with fantasy football, feel free to skip this chapter.

2.1 Football

2.1.1 The Objective

The goal in football is for a team to score more points than their opponent. A game lasts 60 minutes, and it is separated into four 15-minute quarters. The team with the most points when the time runs out wins.

2.1.2 The Roster

2.1.2.1 Overview

Each team has 11 players on the field at a time. The particular players who are on the field will depend on the situation, but usually includes one of the three subsets of players:

1. Offense
2. Defense
3. Special Teams

An example formation is depicted in Figure Figure 2.1.

2.1.2.2 Offense

The offense is on the field when the team has the ball.

Players on offense include:

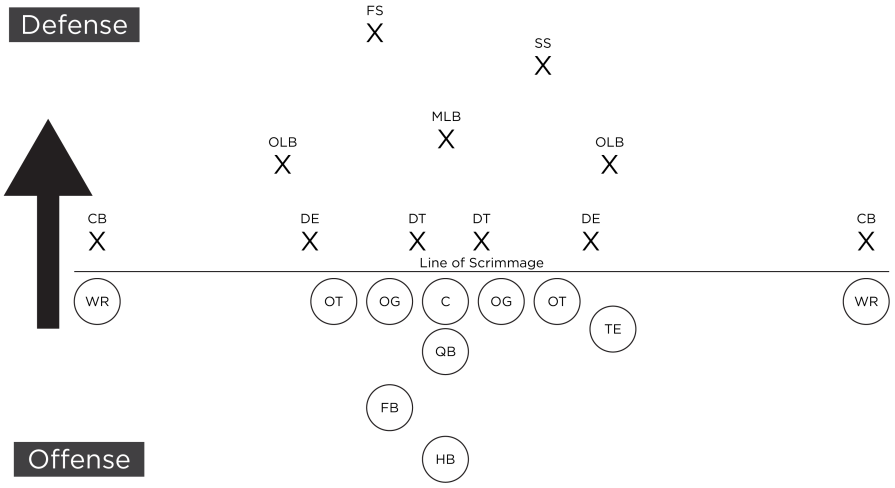


Figure 2.1 An Example Football Formation for the Offense and Defense. The solid line indicates the line of scrimmage. The arrow indicates the direction the offense tries to advance the ball.

- Quarterback (QB)
- Running Back (RB)
  - Halfback or Tailback
  - Fullback (FB)
- Wide Receiver (WR)
- Tight End (TE)
- Offensive Linemen (OL)
  - Center (C)
  - Offensive Guard (OG)
  - Offensive Tackle (OT)

Quarterbacks are the most important player on the offense. They help lead the team down the field. Quarterbacks receive the ball from the Center at the beginning of the play, and they can either hand the ball off (typically to a Running Back or Fullback), pass the ball (typically to a Wide Receiver or Tight End), or run the ball. Quarterbacks tend to have a strong arm for throwing the ball far and accurately. Some quarterbacks are fast and are considered "dual threats" to pass or run.

Running Backs take a hand-off from the Quarterback to execute a running play. They may also catch short passes from the Quarterback or help protect (i.e., block for) the Quarterback from the defensive players who are trying to tackle the Quarterback. Halfbacks or Tailbacks tend to be quick and agile. Fullbacks tend to be strong and powerful.

Wide Receivers catch passes from the Quarterback to execute a passing play. On running plays, they provide protection for the player running the ball (e.g., the Running Back) so the ball carrier can get as far as possible without being tackled. Wide receivers tend to be tall, fast, have good hands (can catch the ball well), and can jump high.

Tight Ends block for running and passing plays, and they catch passes from the Quarterback. Tight ends tend to be strong and have good hands.

Offensive Linemen block for running and passing plays. On passing plays, they provide protection for the Quarterback so the Quarterback has time to pass the ball without being tackled. On running plays, they provide protection for the player running the ball (e.g., the Running Back) so the ball carrier can get as far as possible without being tackled. Offensive Linemen tend to be large so they can provide adequate protection for the Quarterback and Running Back.

2.1.2.3 Defense

The defense is on the field when the team does not have the ball (i.e., when the opposing team has the ball).

Players on defense include:

- Defensive Linemen (DL)
  - Defensive End (DE)
  - Defensive Tackle (DT)
- Linebacker (LB)
  - Middle (or Inside) Linebacker (MLB)
  - Outside Linebacker (OLB)
- "The Secondary" / Defensive Back (DB)
  - Cornerback (CB)
  - Safety (S)
    - Free Safety (FS)
    - Strong Safety (SS)

The players on the defense attempt to tackle the offensive players for as short of gains as possible and attempt to prevent completed passes.

On passing plays, Defensive Linemen try to apply pressure to the Quarterback and try to sack them. On rushing plays, Defensive Linemen try to tackle the ball carrier. Defensive Linemen tend to be large yet quick so they can apply pressure to the Quarterback.

Linebackers are versatile in that, on a given play, they may attempt to a) "blitz" to sack the Quarterback, b) stop the Running Back, or c) prevent a completed pass. Linebackers tend to be strong yet agile.

Defensive Backs are specialist pass defenders. The main role of Cornerbacks is to cover the Wide Receivers. Safeties serve as the last line of defense for longer passes. Defensive Backs tend to be quick and agile.

2.1.2.4 Special Teams

The special teams involves specialist players who are on the field during all kicking plays including kickoffs, field goals, and punts.

Players on special teams include:

- Kicker (K)
- Punter (P)
- Holder
- Long Snapper
- Punt Returner
- Kick Returner
- and other players intended to block for or to tackle the ball carrier

On a field goal attempt, the Long Snapper snaps the ball to the Holder, who holds the ball for the Kicker. The Kicker attempts field goals and, during kickoffs, kicks the ball to the opposing team. During kickoffs, the Kick Returner catches the kicked ball and returns it for as many yards as possible. During a punt play, the Long Snapper snaps the ball to the Punter who kicks (i.e., punts) the ball to the opposing team. The Punt Returner catches the punted ball and returns it for as many yards as possible.

2.1.3 The Field

The football field is rectangular and is 120 yards long and 53 1/3 yards wide. At each end of the 120-yard field is a team's end zone. Each end zone is 10 yards long. Thus, the distance from one end zone to the other end zone is 100 yards. Behind each end zone is a field goal post. A diagram of a football field is depicted in Figure Figure 2.2.

2.1.4 The Gameplay

At the beginning of the game, there is a coin flip to determine which teams receives the ball first and which team takes which side of the field. During

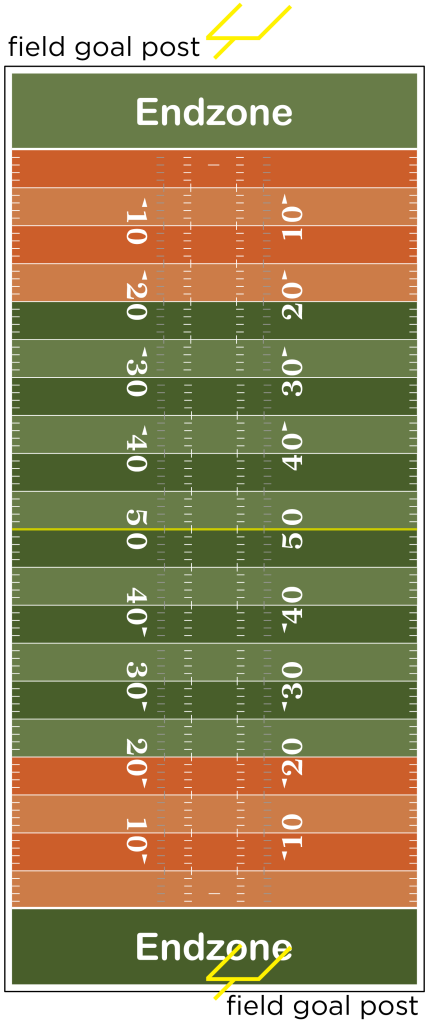


Figure 2.2 A Diagram of a Football Field. The yard markers depict the distance from the nearest end zone. The orange shaded area is called the "red zone", where chances of scoring points are highest. The original figure was modified to depict field goal posts. (Figure retrieved from https://commons.wikimedia.org/wiki/File:American_football_field.svg)

the kickoff, the kicking team kicks the ball to the receiving team, who has the option to return the kick. The offense starts their possession at the 25 yard line—if there is no return (i.e., a touchback)—or wherever the kick returner is tackled or goes out of bounds.

The team with the ball (i.e., the offense) has four opportunities ("downs") to advance the ball 10 yards. A team can advance the ball either by running it or passing and catching it. At the end of a rushing play, the ball advances to wherever the ball carrier is tackled or goes out of bounds (i.e., wherever the player is "down"). At the end of a passing play, if the thrown ball is caught (i.e., a completed pass), the ball advances to wherever the ball carrier is tackled or goes out of bounds. If the thrown ball is not caught in bounds before the ball hits the ground (i.e., an incomplete pass), the ball does not advance. Wherever the ball is advanced to dictates where the next play begins. The yard position on the field where the next play takes place from is known as the "line of scrimmage". Neither team can cross the line the line of scrimmage until the next play begins. To begin the play, the ball is placed on the line of scrimmage and the Center gives (or "snaps") the ball to the Quarterback.

If the team advances the ball 10 or more yards within four downs, the team receives a "first down" and is awarded a new set of downs—four more downs to advance the ball 10 more yards. If the team advances the ball all the way to the other team's end zone, they score a touchdown. If the team fails to advance the ball 10 or more yards within four downs, the team loses the ball, and the other team takes possession at that spot on the field. There are risks of giving the other team the ball with a short distance to score. Thus, on fourth down, instead of trying to advance the ball for a first down, a team may choose to kick a field goal—to get points—or to punt.

A field goal involves a kicker kicking the ball with an intent to kick the ball through the field goal posts ("uprights"). To score points by making a field goal, the kicked ball must go between the uprights (extended vertically) and over the cross bar.

Punting involves a punter kicking the ball to the other team with an intent to give their opponent worse field position, thus making it harder for the other team to score. The punting team tries to pin the opponent as close as possible to the opponent's end zone (i.e., as far as possible from the own team's end zone), so they have a longer distance to go to score a touchdown.

There are multiple ways that ball possession can switch from the offense to the other team. After scoring a touchdown, field goal, or safety, there is a kick-off, in which the scoring team kicks the ball to the opponent. Another way that the ball swithces possession to the other team is if the team commits a turnover. The defense can force a turnover by an interception, fumble recovery, or turnover on downs. A turnover due to an interception occurs when the defense catches the the quarterback's pass. A turnover due to a fumble recovery occurs when an offensive player, who had possession of the ball, loses

the ball before being down or scoring a touchdown and the ball is recovered by the opponent. A turnover on downs occurs when the team attempts to achieve the remainder of the needed 10 yards to go on fourth down but fails.

Other football-related situations include tackles for loss and sacks. A tackle for loss occurs when a ball carrier is tackled behind the line of scrimmage. A sack occurs when a Quarterback is tackled with the ball behind the line of scrimmage. A pass defended occurs when a defensive player knocks down the ball in the air so that the indended receiver cannot catch the ball.

2.1.5 The Scoring

The goal of the team with the ball (i.e., the offense) is to score points. It can do this by either advancing the ball into the other team's end zone (6 points) or by kicking a field goal (3 points). Advancing the ball in the other team's end zone is called a touchdown. After a touchdown, the offense chooses to attempt either a point-after-touchdown (PAT) or a two-point conversion. A PAT is a short kick attempt from the 15-yard line that, if it goes through the goal posts ("uprights"), is worth 1 point. A two-point conversion is a single-scoring opportunity from the 3-yard line (i.e., 3 yards away from the end zone). If the offense scores (i.e., advances the ball into the end zone) from the 3-yard line, the team is awarded 2 points.

A team can kick a field goal from any distance as long as the kick goes through the goal posts. The current record for the longest field goal is 66 yards (by Justin Tucker in 2021).

A safety occurs when the offense is tackled with the ball in their own end zone. When a safety occurs, the opposing team (i.e., defense) is awarded two points.

2.1.6 Glossary of Terms

- running play ("run") or rushing play (or "rush")
- passing play (or "pass")
- passing attempt
- rushing attempt
- passing completion
- passing incompletion
- passing yards
- rushing yards
- receving yards
- reception
- touchdown
- passing touchdown

- rushing touchdown
- receiving touchdown
- two-point conversion
- block
- kickoff
- field goal
- point after touchdown (PAT)
- extra point returned
- punt
- fumble lost
- fumble forced
- fumble recovery
- interception
- tackle
- tackle solo
- tackle assist
- tackle for loss
- sack
- pass defended
- safety

2.2 Fantasy Football

2.2.1 Overview of Fantasy Football

As noted in the Introduction[1], fantasy football is an online game where participants assemble (i.e., "draft") imaginary teams composed of real-life National Football League (NFL) players.[2] The participants are in charge of managing and making strategic decisions for their imaginary team to have the best possible team that will score the most points. Thus, the participants are called "managers". Managers make decisions such as selecting which players to draft, selecting which players to play (i.e., "start") on a weekly basis, identifying players to pick up from the remaining pool of available players (i.e., waiver wire), and making trades with other teams. Fantasy football relies heavily on prediction—trying to predict which players will perform best and selecting them accordingly.

[1] intro.qmd

[2] Fantasy leagues are also available for baseball[3], basketball[4], and many other sports.

2.2.2 The Fantasy League

A fantasy football "league" is composed of various imaginary (i.e., "fantasy") teams—and their associated manager. In the fantasy league, the managers' fantasy teams play against each other. A fantasy league is commonly composed of 8, 10, or 12 fantasy teams, but leagues can have more or fewer teams.

2.2.3 The Roster of a Fantasy Team

On a given roster, a manager has a "starting lineup" and a "bench". Each week, the manager decides which players on their roster to put in the starting lineup, and which to keep on the bench. In many leagues, a starting lineup is composed of offensive players, a kicker, and defense/special teams:

Offensive players:

Table 2.1 Offensive Players in the Starting Lineup

Position	Typical Number of Players in Starting Lineup
Quarterback (QB)	1
Running Back (RB)	2
Wide Receiver (WR)	2
Tight End (TE)	1
Flex Position	1

A "flex position" is a flexible position that can involve a player from various positions: e.g., a Running Back, Wide Receiver, or Tight End.

Kickers:

- one Kicker (K)

Defense/Special Teams:

- one Team Defense (DST/D/DEF) or multiple Individual Defensive Players (IDP)

2.2.4 Scoring

2.2.4.1 Scoring Overview

In the game of fantasy football, managers accumulate points on a weekly basis based on players' actual statistical performances in NFL games. Managers receive points for only those players who are on their starting lineup (not players on their bench). A manager's goal is to outscore their opponent each week to win matches and ultimately claim victory in the league. Scoring settings can differ from league to league.

Below are common scoring settings for fantasy leagues.

2.2.4.2 Offensive Players

Table 2.2 Common Scoring Settings for Offensive Players

Statistical category	Points
Rushing or receiving TD	6
Returning a kick or punt for a TD	6
Returning or recovering a fumble for a TD	6
Passing TD	4
Passing INT	−2
Fumble lost	−2
Rushing, passing, or receiving 2-point conversion	2
Rushing or receiving yards	1 point per 10 yards
Passing yards	1 point per 25 yards

Note: "TD" = touchdown; "INT" = interception

Other common (but not necessarily standard) statistical categories include:

- receptions (called "point per reception" [PPR] leagues)
- return yards
- passing attempts
- rushing attempts

2.2.4.3 Kickers

Table 2.3 Common Scoring Settings for Kickers

Statistical category	Points
FG made: 50+ yards	5
FG made: 40–49 yards	4
FG made: 39 yards or less	3
Rushing, passing, or receiving 2-point conversion	2
Point after touchdown attempt made	1
Point after touchdown attempt missed	−1
Missed FG: 0–39 yards	−2
Missed FG: 40–49 yards	−1

Note: "FG" = field goal

2.2.4.4 Team Defense/Special Teams

Table 2.4 Common Scoring Settings for Team Defense/Special Teams

Statistical category	Points
Defensive or special teams TD	3
Interception	2
Fumble recovery	2
Blocked punt, PAT, or FG	2
Safety	2
Sack	1

Note: "TD" = touchdown; "PAT" = point after touchdown; "FG" = field goal

2.2.4.5 Individual Defensive Players

Table 2.5 Common Scoring Settings for Individual Defensive Players

Statistical category	Points
Tackle solo	1
Tackle assist	0.5
Tackle for loss	1
Sack	2
Interception	4

Statistical category	Points
Fumble forced	2
Fumble recovery	2
TD	6
Safety	2
Pass defended	1
Blocked kick	2
Extra point returned	2

Note: "TD" = touchdown

Other common (but not necessarily standard) statistical categories include:

- turnover return yards

2.2.4.6 Common Scoring Abbreviations

- "TD" = touchdown
- "INT" = interception
- "yds" = yards
- "ATT" = attempts
- "2-pt conversion" = two-point conversion
- "FG" = field goal
- "PAT" = point after touchdown (i.e., extra point/point after attempt)

3

Getting Started with R for Data Analysis

The book uses R for statistical analyses (http://www.r-project.org). R is a free software environment; you can download it at no charge here: https://cran.r-project.org.

3.1 Initial Setup

To get started, follow the following steps:

1. Install R: https://cran.r-project.org
2. Install RStudio Desktop: https://posit.co/download/rstudio-desktop
3. After installing RStudio, open RStudio and run the following code in the console to install several key R packages:

```
install.packages(c("petersenlab","tidyverse","psych"))
```

> **Note 1: If you are in Dr. Petersen's class**
>
> If you are in Dr. Petersen's class, also perform the following steps:
>
> 1. Set up a free account on GitHub.com[a].
> 2. Download GitHub Desktop: https://desktop.github.com
>
> [a] https://github.com

3.2 Installing Packages

You can install R packages using the following syntax:

```
install.packages("INSERT_PACKAGE_NAME_HERE")
```

For instance, you can use the following code to install the `nflreadr` package:

```
install.packages("nflreadr")
```

3.3 Load Packages

```
library("nflreadr")
library("nflfastR")
```

```
Attaching package: 'nflfastR'

The following objects are masked from 'package:nflreadr':

    load_pbp, load_player_stats
```

```
library("nfl4th")
library("nflplotR")
library("progressr")
library("lubridate")
```

```
Attaching package: 'lubridate'

The following objects are masked from 'package:base':

    date, intersect, setdiff, union
```

```
library("tidyverse")
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr   1.1.4     v readr   2.1.5
v forcats 1.0.0     v stringr 1.5.1
v ggplot2 3.5.1     v tibble  3.2.1
v purrr   1.0.2     v tidyr   1.3.1

-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

3.4 Download Football Data

3.4.1 Players

```
nfl_players <- progressr::with_progress(
  nflreadr::load_players())
```

3.4.2 Teams

```
nfl_teams <- progressr::with_progress(
  nflreadr::load_teams(current = TRUE))
```

3.4.3 Player Info

3.4.4 Rosters

A Data Dictionary for rosters is located at the following link: https://nflreadr.nflverse.com/articles/dictionary_rosters.html

```
nfl_rosters <- progressr::with_progress(
  nflreadr::load_rosters(seasons = TRUE))
```

```
nfl_rosters_weekly <- progressr::with_progress(
  nflreadr::load_rosters_weekly(seasons = TRUE))
```

3.4.5 Game Schedules

A Data Dictionary for game schedules data is located at the following link: https://nflreadr.nflverse.com/articles/dictionary_schedules.html

```
nfl_schedules <- progressr::with_progress(
  nflreadr::load_schedules(seasons = TRUE))
```

3.4.6 The Combine

A Data Dictionary for data from the combine is located at the following link: https://nflreadr.nflverse.com/articles/dictionary_combine.html

```
nfl_combine <- progressr::with_progress(
  nflreadr::load_combine(seasons = TRUE))
```

3.4.7 Draft Picks

A Data Dictionary for draft picks data is located at the following link: https://nflreadr.nflverse.com/articles/dictionary_draft_picks.html

```
nfl_draftPicks <- progressr::with_progress(
  nflreadr::load_draft_picks(seasons = TRUE))
```

3.4.8 Depth Charts

A Data Dictionary for data from weekly depth charts is located at the following link: https://nflreadr.nflverse.com/articles/dictionary_depth_charts.html

```
nfl_depthCharts <- progressr::with_progress(
  nflreadr::load_depth_charts(seasons = TRUE))
```

3.4.9 Play-By-Play Data

To download play-by-play data from prior weeks and seasons, we can use the `load_pbp()` function of the `nflreadr` package. We add a progress bar using

the `with_progress()` function from the `progressr` package because it takes a while to run. A Data Dictionary for the play-by-play data is located at the following link: https://nflreadr.nflverse.com/articles/dictionary_pbp.html

> **Note 2: Downloading play-by-play data**
>
> Note: the following code takes a while to run.

```
nfl_pbp <- progressr::with_progress(
  nflreadr::load_pbp(seasons = TRUE))
```

3.4.10 4th Down Data

```
nfl_4thdown <- nfl4th::load_4th_pbp(seasons = 2014:2023)
```

3.4.11 Participation

A Data Dictionary for the participation data is located at the following link: https://nflreadr.nflverse.com/articles/dictionary_participation.html

```
nfl_participation <- progressr::with_progress(
  nflreadr::load_participation(
    seasons = TRUE,
    include_pbp = TRUE))
```

3.4.12 Historical Weekly Actual Player Statistics

We can download historical week-by-week actual player statistics using the `load_player_stats()` function from the `nflreadr` package. A Data Dictionary for statistics for offensive players is located at the following link: https://nflreadr.nflverse.com/articles/dictionary_player_stats.html. A Data Dictionary for statistics for defensive players is located at the following link: https://nflreadr.nflverse.com/articles/dictionary_player_stats_def.html.

```
nfl_actualStats_offense_weekly <- progressr::with_progress(
  nflreadr::load_player_stats(
    seasons = TRUE,
    stat_type = "offense"))
```

```
nfl_actualStats_defense_weekly <- progressr::with_progress(
  nflreadr::load_player_stats(
    seasons = TRUE,
    stat_type = "defense"))

nfl_actualStats_kicking_weekly <- progressr::with_progress(
  nflreadr::load_player_stats(
    seasons = TRUE,
    stat_type = "kicking"))
```

3.4.13 Injuries

A Data Dictionary for injury data is located at the following link: https://nflreadr.nflverse.com/articles/dictionary_injuries.html

```
nfl_injuries <- progressr::with_progress(
  nflreadr::load_injuries(seasons = TRUE))
```

3.4.14 Snap Counts

A Data Dictionary for snap counts data is located at the following link: https://nflreadr.nflverse.com/articles/dictionary_snap_counts.html

```
nfl_snapCounts <- progressr::with_progress(
  nflreadr::load_snap_counts(seasons = TRUE))
```

3.4.15 ESPN QBR

A Data Dictionary for ESPN QBR data is located at the following link: https://nflreadr.nflverse.com/articles/dictionary_espn_qbr.html

```
nfl_espnQBR_seasonal <- progressr::with_progress(
  nflreadr::load_espn_qbr(
    seasons = TRUE,
    summary_type = c("season")))

nfl_espnQBR_weekly <- progressr::with_progress(
  nflreadr::load_espn_qbr(
    seasons = TRUE,
    summary_type = c("weekly")))
```

```
nfl_espnQBR_weekly$game_week <- as.character(nfl_espnQBR_weekly$game_week)

nfl_espnQBR <- bind_rows(
  nfl_espnQBR_seasonal,
  nfl_espnQBR_weekly
)
```

3.4.16 NFL Next Gen Stats

A Data Dictionary for NFL Next Gen Stats data is located at the following link: https://nflreadr.nflverse.com/articles/dictionary_nextgen_stats.html

```
nfl_nextGenStats_pass_weekly <- progressr::with_progress(
  nflreadr::load_nextgen_stats(
    seasons = TRUE,
    stat_type = c("passing")))

nfl_nextGenStats_rush_weekly <- progressr::with_progress(
  nflreadr::load_nextgen_stats(
    seasons = TRUE,
    stat_type = c("rushing")))

nfl_nextGenStats_rec_weekly <- progressr::with_progress(
  nflreadr::load_nextgen_stats(
    seasons = TRUE,
    stat_type = c("receiving")))

nfl_nextGenStats_weekly <- bind_rows(
  nfl_nextGenStats_pass_weekly,
  nfl_nextGenStats_rush_weekly,
  nfl_nextGenStats_rec_weekly
)
```

3.4.17 Advanced Stats from PFR

A Data Dictionary for PFR passing data is located at the following link: https://nflreadr.nflverse.com/articles/dictionary_pfr_passing.html

```
nfl_advancedStatsPFR_pass_seasonal <- progressr::with_progress(
  nflreadr::load_pfr_advstats(
    seasons = TRUE,
    stat_type = c("pass"),
```

```
    summary_level = c("season")))

nfl_advancedStatsPFR_pass_weekly <- progressr::with_progress(
  nflreadr::load_pfr_advstats(
    seasons = TRUE,
    stat_type = c("pass"),
    summary_level = c("week")))

nfl_advancedStatsPFR_rush_seasonal <- progressr::with_progress(
  nflreadr::load_pfr_advstats(
    seasons = TRUE,
    stat_type = c("rush"),
    summary_level = c("season")))

nfl_advancedStatsPFR_rush_weekly <- progressr::with_progress(
  nflreadr::load_pfr_advstats(
    seasons = TRUE,
    stat_type = c("rush"),
    summary_level = c("week")))

nfl_advancedStatsPFR_rec_seasonal <- progressr::with_progress(
  nflreadr::load_pfr_advstats(
    seasons = TRUE,
    stat_type = c("rec"),
    summary_level = c("season")))

nfl_advancedStatsPFR_rec_weekly <- progressr::with_progress(
  nflreadr::load_pfr_advstats(
    seasons = TRUE,
    stat_type = c("rec"),
    summary_level = c("week")))

nfl_advancedStatsPFR_def_seasonal <- progressr::with_progress(
  nflreadr::load_pfr_advstats(
    seasons = TRUE,
    stat_type = c("def"),
    summary_level = c("season")))

nfl_advancedStatsPFR_def_weekly <- progressr::with_progress(
  nflreadr::load_pfr_advstats(
    seasons = TRUE,
    stat_type = c("def"),
    summary_level = c("week")))
```

```
nfl_advancedStatsPFR <- bind_rows(
  nfl_advancedStatsPFR_pass_seasonal,
  nfl_advancedStatsPFR_pass_weekly,
  nfl_advancedStatsPFR_rush_seasonal,
  nfl_advancedStatsPFR_rush_weekly,
  nfl_advancedStatsPFR_rec_seasonal,
  nfl_advancedStatsPFR_rec_weekly,
  nfl_advancedStatsPFR_def_seasonal,
  nfl_advancedStatsPFR_def_weekly,
)
```

3.4.18 Player Contracts

A Data Dictionary for player contracts data is located at the following link: https://nflreadr.nflverse.com/articles/dictionary_contracts.html

```
nfl_playerContracts <- progressr::with_progress(
  nflreadr::load_contracts())
```

3.4.19 FTN Charting Data

A Data Dictionary for FTN Charting data is located at the following link: https://nflreadr.nflverse.com/articles/dictionary_ftn_charting.html

```
nfl_ftnCharting <- progressr::with_progress(
  nflreadr::load_ftn_charting(seasons = TRUE))
```

3.4.20 Fantasy Player IDs

A Data Dictionary for fantasy player ID data is located at the following link: https://nflreadr.nflverse.com/articles/dictionary_ff_playerids.html

```
nfl_playerIDs <- progressr::with_progress(
  nflreadr::load_ff_playerids())
```

3.4.21 FantasyPros Rankings

A Data Dictionary for FantasyPros ranking data is located at the following link: https://nflreadr.nflverse.com/articles/dictionary_ff_rankings.html

```
#nfl_rankings <- progressr::with_progress( # currently throws error
#  nflreadr::load_ff_rankings(type = "all"))

nfl_rankings_draft <- progressr::with_progress(
  nflreadr::load_ff_rankings(type = "draft"))

nfl_rankings_weekly <- progressr::with_progress(
  nflreadr::load_ff_rankings(type = "week"))

nfl_rankings <- bind_rows(
  nfl_rankings_draft,
  nfl_rankings_weekly
)
```

3.4.22 Expected Fantasy Points

A Data Dictionary for expected fantasy points data is located at the following link: https://nflreadr.nflverse.com/articles/dictionary_ff_opportunity.html

```
nfl_expectedFantasyPoints_weekly <- progressr::with_progress(
  nflreadr::load_ff_opportunity(
    seasons = TRUE,
    stat_type = "weekly",
    model_version = "latest"
  ))

nfl_expectedFantasyPoints_pass <- progressr::with_progress(
  nflreadr::load_ff_opportunity(
    seasons = TRUE,
    stat_type = "pbp_pass",
    model_version = "latest"
  ))

nfl_expectedFantasyPoints_rush <- progressr::with_progress(
  nflreadr::load_ff_opportunity(
    seasons = TRUE,
    stat_type = "pbp_rush",
    model_version = "latest"
  ))

nfl_expectedFantasyPoints_weekly$season <- as.integer(nfl_expectedFantasyPoints_weekly$season)

nfl_expectedFantasyPoints_offense <- bind_rows(
```

```
  nfl_expectedFantasyPoints_pass,
  nfl_expectedFantasyPoints_rush
)
```

3.5 Data Dictionary

Data Dictionaries are metadata that describe the meaning of the variables in a datset. You can find Data Dictionaries for the various NFL datasets at the following link: https://nflreadr.nflverse.com/articles/index.html.

3.6 Variable Names

To see the names of variables in a data frame, use the following syntax:

```
names(nfl_players)
```

```
 [1] "status"                    "display_name"
 [3] "first_name"                "last_name"
 [5] "esb_id"                    "gsis_id"
 [7] "suffix"                    "birth_date"
 [9] "college_name"              "position_group"
[11] "position"                  "jersey_number"
[13] "height"                    "weight"
[15] "years_of_experience"       "team_abbr"
[17] "team_seq"                  "current_team_id"
[19] "football_name"             "entry_year"
[21] "rookie_year"               "draft_club"
[23] "college_conference"        "status_description_abbr"
[25] "status_short_description" "gsis_it_id"
[27] "short_name"                "smart_id"
[29] "headshot"                  "draft_number"
[31] "uniform_number"            "draft_round"
[33] "season"
```

3.7 Logical Operators

3.7.1 Is Equal To: ==

```
nfl_players$position_group == "RB"
```

3.7.2 Is Not Equal To: !=

```
nfl_players$position_group != "RB"
```

3.7.3 Is Greater Than: >

```
nfl_players$weight > 300
```

3.7.4 Is Less Than: <

```
nfl_players$weight < 300
```

3.7.5 Is Greater Than or Equal To: >=

```
nfl_players$weight >= 300
```

3.7.6 Is Less Than or Equal To: <=

```
nfl_players$weight <= 300
```

3.7.7 Is In a Value of Another Vector: %in%

```
nfl_players$position_group %in% c("QB","RB","WR")
```

3.7.8 Is Not In a Value of Another Vector: `!(%in%)`

```
!(nfl_players$position_group %in% c("QB","RB","WR"))
```

3.7.9 Is Missing: `is.na()`

```
is.na(nfl_players$college_name)
```

3.7.10 Is Not Missing: `!is.na()`

```
!is.na(nfl_players$college_name)
```

3.7.11 And: `&`

```
nfl_players$position_group == "RB" & nfl_players$weight > 230
```

3.7.12 Or: `|`

```
nfl_players$position_group == "RB" | nfl_players$weight > 230
```

3.8 Subset

To subset a data frame, use brackets to specify the subset of rows and columns to keep, where the value/vector before the comma specifies the rows to keep, and the value/vector after the comma specifies the columns to keep:

```
dataframe[rowsToKeep, columnsToKeep]
```

You can subset by using any of the following:

- numeric indices of the rows/columns to keep (or drop)
- names of the rows/columns to keep (or drop)
- values of `TRUE` and `FALSE` corresponding to which rows/columns to keep

3.8.1 One Variable

To subset one variable, use the following syntax:

```
nfl_players$display_name
```

or:

```
nfl_players[,"display_name"]
```

```
-- nflverse players -------------------------------------------------------

i Data updated: 2024-03-01 01:18:40 UTC

# A tibble: 20,039 x 1
   display_name
   <chr>
 1 'Omar Ellison
 2 A'Shawn Robinson
 3 A.J. Arcuri
 4 A.J. Bouye
 5 A.J. Brown
 6 A.J. Cann
 7 A.J. Cole
 8 A.J. Cruz
 9 A.J. Dalton
10 A.J. Davis
# i 20,029 more rows
```

3.8.2 Particular Rows of One Variable

To subset one variable, use the following syntax:

```
nfl_players$display_name[which(nfl_players$position_group == "RB")]
```

or:

```
nfl_players[which(nfl_players$position_group == "RB"), "display_name"]
```

```
-- nflverse players -------------------------------------------------------

i Data updated: 2024-03-01 01:18:40 UTC

# A tibble: 1,917 x 1
   display_name
   <chr>
 1 A.J. Dillon
 2 A.J. Harris
 3 A.J. Ouellette
 4 A.J. Rose
 5 Aaron Brown
 6 Aaron Craver
 7 Aaron Dykes
 8 Aaron Green
 9 Aaron Hayden
10 Aaron Jones
# i 1,907 more rows
```

3.8.3 Particular Columns (Variables)

To subset particular columns/variables, use the following syntax:

3.8.3.1 Base R

```
subsetVars <- c("status","display_name","college_name")

nfl_players[,c(1,2,9)]
nfl_players[,c("status","display_name","college_name")]
nfl_players[,subsetVars]
```

Or, to drop columns:

```
dropVars <- c("status","college_name")

nfl_players[,-c(1,9)]
nfl_players[,!(names(nfl_players) %in% c("status","college_name"))]
nfl_players[,!(names(nfl_players) %in% dropVars)]
```

3.8.3.2 Tidyverse

```
nfl_players %>%
  select(status, display_name, college_name)

nfl_players %>%
  select(status:college_name)

nfl_players %>%
  select(all_of(subsetVars))
```

Or, to drop columns:

```
nfl_players %>%
  select(-status, -college_name)

nfl_players %>%
  select(-c(status:college_name))

nfl_players %>%
  select(-all_of(dropVars))
```

3.8.4 Particular Rows

To subset particular rows, use the following syntax:

3.8.4.1 Base R

```
subsetRows <- c(1,3,5)

nfl_players[c(1,3,5),]
nfl_players[subsetRows,]
nfl_players[which(nfl_players$position_group == "RB"),]
```

3.8.4.2 Tidyverse

```
nfl_players %>%
  filter(position_group == "RB")

nfl_players %>%
  filter(position_group == "RB", weight <= 250)

nfl_players %>%
  filter(position_group == "RB" | weight >= 250)
```

3.8.5 Particular Rows and Columns

To subset particular rows and columns, use the following syntax:

3.8.5.1 Base R

```
nfl_players[c(1,3,5), c(1,2,3)]
nfl_players[subsetRows, subsetVars]
nfl_players[which(nfl_players$position_group == "RB"), subsetVars]
```

3.8.5.2 Tidyverse

```
nfl_players %>%
  filter(position_group == "RB") %>%
  select(all_of(subsetVars))
```

3.9 View Data

3.9.1 All Data

To view data, use the following syntax:

```
View(nfl_players)
```

3.9.2 First 6 Rows/Elements

To view only the first six rows (if a data frame) or elements (if a vector), use the following syntax:

```
head(nfl_players)
```

```
-- nflverse players ------------------------------------------------------------

i Data updated: 2024-03-01 01:18:40 UTC

# A tibble: 6 x 33
  status display_name    first_name last_name esb_id  gsis_id suffix birth_date
  <chr>  <chr>           <chr>      <chr>     <chr>   <chr>   <chr>  <chr>
1 RET    'Omar Ellison    'Omar      Ellison   ELL711~ 00-000~ <NA>   <NA>
2 ACT    A'Shawn Robinson A'Shawn    Robinson  ROB367~ 00-003~ <NA>   1995-03-21
3 ACT    A.J. Arcuri      A.J.       Arcuri    ARC716~ 00-003~ <NA>   <NA>
4 RES    A.J. Bouye       Arlandus   Bouye     BOU651~ 00-003~ <NA>   1991-08-16
5 ACT    A.J. Brown       Arthur     Brown     BRO413~ 00-003~ <NA>   1997-06-30
6 ACT    A.J. Cann        Aaron      Cann      CAN364~ 00-003~ <NA>   1991-10-03
# i 25 more variables: college_name <chr>, position_group <chr>,
#   position <chr>, jersey_number <int>, height <dbl>, weight <int>,
#   years_of_experience <chr>, team_abbr <chr>, team_seq <int>,
#   current_team_id <chr>, football_name <chr>, entry_year <int>,
#   rookie_year <int>, draft_club <chr>, college_conference <chr>,
#   status_description_abbr <chr>, status_short_description <chr>,
#   gsis_it_id <int>, short_name <chr>, smart_id <chr>, headshot <chr>, ...
```

```
head(nfl_players$display_name)
```

```
[1] "'Omar Ellison"    "A'Shawn Robinson" "A.J. Arcuri"      "A.J. Bouye"
[5] "A.J. Brown"       "A.J. Cann"
```

3.10 Data Characteristics

3.10.1 Data Structure

```
str(nfl_players)
```

```
nflvrs_d [20,039 x 33] (S3: nflverse_data/tbl_df/tbl/data.table/data.frame)
 $ status                   : chr [1:20039] "RET" "ACT" "ACT" "RES" ...
 $ display_name             : chr [1:20039] "'Omar Ellison" "A'Shawn Robinson" "A.J. Arcuri" "A.J. Bouye" ...
 $ first_name               : chr [1:20039] "'Omar" "A'Shawn" "A.J." "Arlandus" ...
 $ last_name                : chr [1:20039] "Ellison" "Robinson" "Arcuri" "Bouye" ...
 $ esb_id                   : chr [1:20039] "ELL711319" "ROB367960" "ARC716900" "BOU651714" ...
 $ gsis_id                  : chr [1:20039] "00-0004866" "00-0032889" "00-0037845" "00-0030228" ...
 $ suffix                   : chr [1:20039] NA NA NA NA ...
 $ birth_date               : chr [1:20039] NA "1995-03-21" NA "1991-08-16" ...
 $ college_name             : chr [1:20039] NA "Alabama" "Michigan State" "Central Florida" ...
 $ position_group           : chr [1:20039] "WR" "DL" "OL" "DB" ...
 $ position                 : chr [1:20039] "WR" "DT" "T" "CB" ...
 $ jersey_number            : int [1:20039] 84 91 61 24 11 60 6 81 63 20 ...
 $ height                   : num [1:20039] 73 76 79 72 72 75 76 69 76 72 ...
 $ weight                   : int [1:20039] 200 330 320 191 226 325 220 190 280 183 ...
 $ years_of_experience      : chr [1:20039] "2" "8" "2" "8" ...
 $ team_abbr                : chr [1:20039] "LAC" "NYG" "LA" "CAR" ...
 $ team_seq                 : int [1:20039] NA 1 NA 1 1 1 1 NA NA NA ...
 $ current_team_id          : chr [1:20039] "4400" "3410" "2510" "0750" ...
 $ football_name            : chr [1:20039] NA "A'Shawn" "A.J." "A.J." ...
 $ entry_year               : int [1:20039] NA 2016 2022 2013 2019 2015 2019 NA NA NA ...
 $ rookie_year              : int [1:20039] NA 2016 2022 2013 2019 2015 2019 NA NA NA ...
 $ draft_club               : chr [1:20039] NA "DET" "LA" NA ...
 $ college_conference       : chr [1:20039] NA "Southeastern Conference" "Big Ten Conference" "American A
 $ status_description_abbr : chr [1:20039] NA "A01" "A01" "R01" ...
 $ status_short_description: chr [1:20039] NA "Active" "Active" "R/Injured" ...
 $ gsis_it_id               : int [1:20039] NA 43335 54726 40688 47834 42410 48335 NA NA NA ...
 $ short_name               : chr [1:20039] NA "A.Robinson" "A.Arcuri" "A.Bouye" ...
 $ smart_id                 : chr [1:20039] "3200454c-4c71-1319-728e-d49d3d236f8f" "3200524f-4236-7960-bf20
 $ headshot                 : chr [1:20039] NA "https://static.www.nfl.com/image/private/f_auto,q_auto/leag
 $ draft_number             : int [1:20039] NA 46 261 NA 51 67 NA NA NA NA ...
 $ uniform_number           : chr [1:20039] NA "91" "61" "24" ...
 $ draft_round              : chr [1:20039] NA NA NA NA ...
 $ season                   : int [1:20039] NA NA NA NA NA NA NA NA NA NA ...
 - attr(*, "nflverse_type")= chr "players"
 - attr(*, "nflverse_timestamp")= POSIXct[1:1], format: "2024-03-01 01:18:40"
```

3.10.2 Data Dimensions

Number of rows and columns:

```
dim(nfl_players)
```

```
[1] 20039    33
```

3.10.3 Number of Elements

```
length(nfl_players$display_name)
```

```
[1] 20039
```

3.10.4 Number of Missing Elements

```
length(nfl_players$college_name[which(is.na(nfl_players$college_name))])
```

```
[1] 12127
```

3.10.5 Number of Non-Missing Elements

```
length(nfl_players$college_name[which(!is.na(nfl_players$college_name))])
```

```
[1] 7912
```

```
length(na.omit(nfl_players$college_name))
```

```
[1] 7912
```

3.11 Create New Variables

To create a new variable, use the following syntax:

```
nfl_players$newVar <- NA
```

Here is an example of creating a new variable:

```
nfl_players$newVar <- 1:nrow(nfl_players)
```

3.12 Create a Data Frame

Here is an example of creating a data frame:

```
mydata <- data.frame(
  ID = c(1:5, 1047:1051),
  cat = sample(
    0:1,
    10,
    replace = TRUE)
)

mydata
```

```
     ID cat
1     1   1
2     2   0
3     3   1
4     4   1
5     5   1
6  1047   1
7  1048   1
8  1049   1
9  1050   0
10 1051   1
```

3.13 Recode Variables

Here is an example of recoding a variable:

```
mydata$oldVar1 <- NA
mydata$oldVar1[which(mydata$cat == 0)] <- "dog"
mydata$oldVar1[which(mydata$cat == 1)] <- "cat"

mydata$oldVar2 <- NA
mydata$oldVar2[which(mydata$cat == 0)] <- "no"
mydata$oldVar2[which(mydata$cat == 1)] <- "yes"
```

Recode multiple variables:

```
mydata %>%
  mutate(across(c(
    oldVar1:oldVar2),
    ~ case_match(
      .,
      c("dog","no") ~ 0,
      c("cat","yes") ~ 1)))
```

```
     ID cat oldVar1 oldVar2
1     1   1       1       1
2     2   0       0       0
3     3   1       1       1
4     4   1       1       1
5     5   1       1       1
6  1047   1       1       1
7  1048   1       1       1
8  1049   1       1       1
9  1050   0       0       0
10 1051   1       1       1
```

3.14 Rename Variables

```
mydata <- mydata %>%
  rename(
    newVar1 = oldVar1,
    newVar2 = oldVar2)
```

Using a vector of variable names:

```
varNamesFrom <- c("oldVar1","oldVar2")
varNamesTo <- c("newVar1","newVar2")

mydata <- mydata %>%
  rename_with(~ varNamesTo, all_of(varNamesFrom))
```

3.15 Convert the Types of Variables

One variable:

```
mydata$factorVar <- factor(mydata$ID)
mydata$numericVar <- as.numeric(mydata$cat)
mydata$integerVar <- as.integer(mydata$cat)
mydata$characterVar <- as.character(mydata$newVar1)
```

Multiple variables:

```
mydata %>%
  mutate(across(c(
    ID,
    cat),
    as.numeric))
```

```
     ID cat newVar1 newVar2 factorVar numericVar integerVar characterVar
1     1   1     cat     yes         1          1          1          cat
2     2   0     dog      no         2          0          0          dog
3     3   1     cat     yes         3          1          1          cat
4     4   1     cat     yes         4          1          1          cat
5     5   1     cat     yes         5          1          1          cat
6  1047   1     cat     yes      1047          1          1          cat
7  1048   1     cat     yes      1048          1          1          cat
8  1049   1     cat     yes      1049          1          1          cat
9  1050   0     dog      no      1050          0          0          dog
10 1051   1     cat     yes      1051          1          1          cat
```

```
mydata %>%
  mutate(across(
    ID:cat,
    as.numeric))
```

```
     ID cat newVar1 newVar2 factorVar numericVar integerVar characterVar
1     1   1     cat     yes         1          1          1          cat
2     2   0     dog      no         2          0          0          dog
3     3   1     cat     yes         3          1          1          cat
4     4   1     cat     yes         4          1          1          cat
5     5   1     cat     yes         5          1          1          cat
6  1047   1     cat     yes      1047          1          1          cat
7  1048   1     cat     yes      1048          1          1          cat
8  1049   1     cat     yes      1049          1          1          cat
9  1050   0     dog      no      1050          0          0          dog
10 1051   1     cat     yes      1051          1          1          cat
```

```
mydata %>%
  mutate(across(where(is.factor), as.character))
```

```
     ID cat newVar1 newVar2 factorVar numericVar integerVar characterVar
1     1   1     cat     yes         1          1          1          cat
2     2   0     dog      no         2          0          0          dog
3     3   1     cat     yes         3          1          1          cat
4     4   1     cat     yes         4          1          1          cat
5     5   1     cat     yes         5          1          1          cat
6  1047   1     cat     yes      1047          1          1          cat
7  1048   1     cat     yes      1048          1          1          cat
8  1049   1     cat     yes      1049          1          1          cat
9  1050   0     dog      no      1050          0          0          dog
10 1051   1     cat     yes      1051          1          1          cat
```

3.16 Merging/Joins

3.16.1 Overview

Merging (also called joining) merges two data objects using a shared set of variables called "keys." The keys are the variable(s) that uniquely identify each row (i.e., they account for the levels of nesting). In some data objects, the key might be the participant's ID (e.g., `participantID`). However, some data objects have multiple keys. For instance, in long form data objects, each participant may have multiple rows corresponding to multiple timepoints. In this case, the keys are `participantID` and `timepoint`. If a participant has multiple rows corresponding to timepoints and measures, the keys are `participantID`, `timepoint`, and `measure`. In general, each row should have a value on each of the keys; there should be no missingness in the keys.

To merge two objects, the keys must be present in both objects. The keys are used to merge the variables in object 1 (`x`) with the variables in object 2 (`y`). Different merge types select different rows to merge.

Note: if the two objects include variables with the same name (apart from the keys), `R` will not know how you want each to appear in the merged object. So, it will add a suffix (e.g., `.x`, `.y`) to each common variable to indicate which object (i.e., object `x` or object `y`) the variable came from, where object `x` is the first object—i.e., the object to which object `y` (the second object) is merged. In general, apart from the keys, you should not include variables with the same name in two objects to be merged. To prevent this, either remove or rename the shared variable in one of the objects, or include the shared variable as a key. However, as described above, you should include it as a key ***only*** if it uniquely identifies each row in terms of levels of nesting.

3.16.2 Data Before Merging

Here are the data in the `nfl_actualStats_offense_weekly` object:

```
dim(nfl_actualStats_offense_weekly)
```

```
[1] 129739     53
```

```
dim(distinct(nfl_actualStats_offense_weekly, player_id, season, week, .keep_all = TRUE))
```

```
[1] 129739     53
```

The data are structured in player-season-week form. That is, every row in the dataset is uniquely identified by the variables, `player_id`, `season`, and `week`.

Here are the data in the `nfl_expectedFantasyPoints_weekly` object:

```
dim(nfl_players)
```

```
[1] 20039    33
```

```
dim(distinct(nfl_players, gsis_id, .keep_all = TRUE))
```

```
[1] 20039    33
```

3.16.3 Types of Joins

3.16.3.1 Visual Overview of Join Types

Below is a visual that depicts various types of merges/joins. Object `x` is the circle labeled as `x`. Object `y` is the circle labeled as `y`. The area of overlap in the Venn diagram indicates the rows on the keys that are shared between the two objects (e.g., the same `player_id`, `season`, and `week`). The non-overlapping area indicates the rows on the keys that are unique to each object. The shaded blue area indicates which rows (on the keys) are kept in the merged object from each of the two objects, when using each of the merge types. For instance, a left outer join keeps the shared rows and the rows that are unique to object `x`, but it drops the rows that are unique to object `y`.

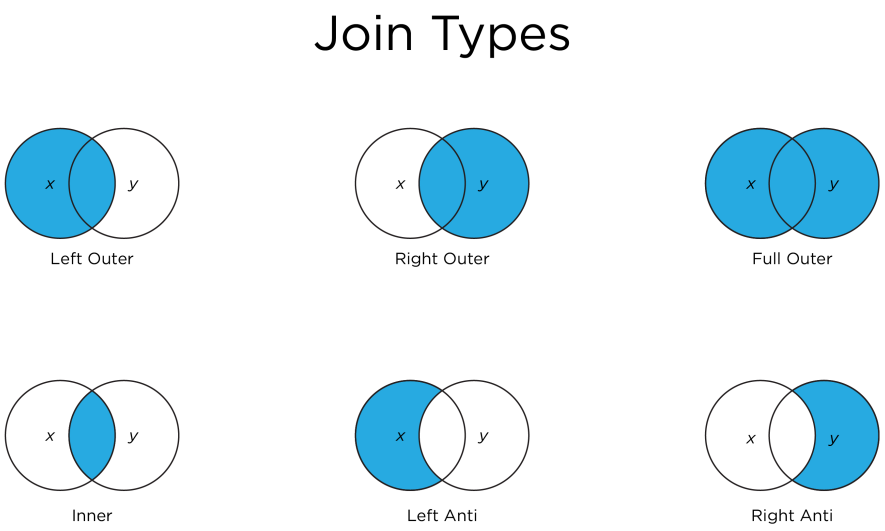


Figure 3.1 Types of merges/joins

3.16.3.2 Full Outer Join

A full outer join includes all rows in x **or** y. It returns columns from x and y. Here is how to merge two data frames using a full outer join (i.e., "full join"):

```
fullJoinData <- full_join(
  nfl_actualStats_offense_weekly,
  nfl_players,
  by = c("player_id" = "gsis_id"))

dim(fullJoinData)
```

3.16.3.3 Left Outer Join

A left outer join includes all rows in x. It returns columns from x and y. Here is how to merge two data frames using a left outer join ("left join"):

```
leftJoinData <- left_join(
  nfl_actualStats_offense_weekly,
  nfl_players,
  by = c("player_id" = "gsis_id"))

dim(leftJoinData)
```

3.16.3.4 Right Outer Join

A right outer join includes all rows in y. It returns columns from x and y. Here is how to merge two data frames using a right outer join ("right join"):

```
rightJoinData <- right_join(
  nfl_actualStats_offense_weekly,
  nfl_players,
  by = c("player_id" = "gsis_id"))

dim(rightJoinData)
```

3.16.3.5 Inner Join

An inner join includes all rows that are in **both** x **and** y. An inner join will return one row of x for each matching row of y, and can duplicate values of records on either side (left or right) if x and y have more than one matching record. It returns columns from x and y. Here is how to merge two data frames using an inner join:

```
innerJoinData <- inner_join(
  nfl_actualStats_offense_weekly,
  nfl_players,
  by = c("player_id" = "gsis_id"))

dim(innerJoinData)
```

3.16.3.6 Semi Join

A semi join is a filter. A left semi join returns all rows from x **with** a match in y. That is, it filters out records from x that are not in y. Unlike an inner

join, a left semi join will never duplicate rows of x, and it includes columns from only x (not from y). Here is how to merge two data frames using a left semi join:

```
semiJoinData <- semi_join(
  nfl_actualStats_offense_weekly,
  nfl_players,
  by = c("player_id" = "gsis_id"))

dim(semiJoinData)
```

3.16.3.7 Anti Join

An anti join is a filter. A left anti join returns all rows from x **without** a match in y. That is, it filters out records from x that are in y. It returns columns from only x (not from y). Here is how to merge two data frames using a left anti join:

```
antiJoinData <- anti_join(
  nfl_actualStats_offense_weekly,
  nfl_players,
  by = c("player_id" = "gsis_id"))

dim(antiJoinData)
```

3.16.3.8 Cross Join

A cross join combines each row in x with each row in y.

```
crossJoinData <- cross_join(
  unique(nfl_actualStats_offense_weekly[,c("player_id","player_display_name")]),
  data.frame(season = 2020:2024))

crossJoinData
dim(crossJoinData)
```

3.17 Transform Data from Long to Wide

Original data:

```
dataLong <- nfl_actualStats_offense_weekly %>%
  select(player_id, player_display_name, season, week, fantasy_points)

dim(dataLong)
```

Data widened by two variable (`season` and `week`), using `tidyverse`:

```
dataWide <- dataLong %>%
  pivot_wider(
    names_from = c(season, week),
    names_glue = "{.value}_{season}_week{week}",
    values_from = fantasy_points)

dim(dataWide)
```

3.18 Transform Data from Wide to Long

Original data:

```
dataWide <- nfl_actualStats_offense_weekly %>%
  select(player_id, player_display_name, season, week, recent_team, opponent_team)

dim(dataWide)
```

Data in long form, transformed from wide form using `tidyverse`:

```
dataLong <- dataWide %>%
  pivot_longer(
    cols = c(recent_team, opponent_team),
    names_to = "role",
    values_to = "team")

dim(dataLong)
```

3.19 Calculations

3.19.1 Historical Actual Player Statistics

In addition to week-by-week actual player statistics, we can also compute historical actual player statistics as a function of different timeframes, including season-by-season and career statistics.

3.19.1.1 Career Statistics

First, we can compute the players' career statistics using the `calculate_player_stats()`, `calculate_player_stats_def()`, and `calculate_player_stats_kicking()` functions from the `nflfastR` package for offensive players, defensive players, and kickers, respectively.

> **Note 3: Calculating players' career statistics**
>
> Note: the following code takes a while to run.

```
nfl_actualStats_offense_career <- nflfastR::calculate_player_stats(
  nfl_pbp,
  weekly = FALSE)

nfl_actualStats_defense_career <- nflfastR::calculate_player_stats_def(
  nfl_pbp,
  weekly = FALSE)

nfl_actualStats_kicking_career <- nflfastR::calculate_player_stats_kicking(
  nfl_pbp,
  weekly = FALSE)
```

3.19.1.2 Season-by-Season Statistics

Second, we can compute the players' season-by-season statistics.

```
seasons <- unique(nfl_pbp$season)

nfl_pbp_seasonalList <- list()
nfl_actualStats_offense_seasonalList <- list()
nfl_actualStats_defense_seasonalList <- list()
nfl_actualStats_kicking_seasonalList <- list()
```

> **Note 4: Calculating players' season-by-season statistics**
>
> Note: the following code takes a while to run.

```
pb <- txtProgressBar(
  min = 0,
  max = length(seasons),
  style = 3)

for(i in 1:length(seasons)){
  # Subset play-by-play data by season
  nfl_pbp_seasonalList[[i]] <- nfl_pbp %>%
    filter(season == seasons[i])

  # Compute actual statistics by season
  nfl_actualStats_offense_seasonalList[[i]] <-
    nflfastR::calculate_player_stats(
      nfl_pbp_seasonalList[[i]],
      weekly = FALSE)

  nfl_actualStats_defense_seasonalList[[i]] <-
    nflfastR::calculate_player_stats_def(
      nfl_pbp_seasonalList[[i]],
      weekly = FALSE)

  nfl_actualStats_kicking_seasonalList[[i]] <-
    nflfastR::calculate_player_stats_kicking(
      nfl_pbp_seasonalList[[i]],
      weekly = FALSE)

  nfl_actualStats_offense_seasonalList[[i]]$season <- seasons[i]
  nfl_actualStats_defense_seasonalList[[i]]$season <- seasons[i]
  nfl_actualStats_kicking_seasonalList[[i]]$season <- seasons[i]

  print(
    paste("Completed computing projections for season: ", seasons[i], sep = ""))

  # Update the progress bar
  setTxtProgressBar(pb, i)
}

# Close the progress bar
close(pb)
```

```
nfl_actualStats_offense_seasonal <- nfl_actualStats_offense_seasonalList %>%
  bind_rows()
nfl_actualStats_defense_seasonal <- nfl_actualStats_defense_seasonalList %>%
  bind_rows()
nfl_actualStats_kicking_seasonal <- nfl_actualStats_kicking_seasonalList %>%
  bind_rows()
```

3.19.1.3 Week-by-Week Statistics

We already load players' week-by-week statistics above. Nevertheless, we could compute players' weekly statistics from the play-by-play data using the following syntax:

```
nfl_actualStats_offense_weekly <- nflfastR::calculate_player_stats(
  nfl_pbp,
  weekly = TRUE)

nfl_actualStats_defense_weekly <- nflfastR::calculate_player_stats_def(
  nfl_pbp,
  weekly = TRUE)

nfl_actualStats_kicking_weekly <- nflfastR::calculate_player_stats_kicking(
  nfl_pbp,
  weekly = TRUE)
```

3.19.2 Historical Actual Fantasy Points

3.19.3 Player Age

```
# Reshape from wide to long format
nfl_actualStats_offense_weekly_long <- nfl_actualStats_offense_weekly %>%
  pivot_longer(
    cols = c(recent_team, opponent_team),
    names_to = "role",
    values_to = "team")

# Perform separate inner join operations for the home_team and away_team
nfl_actualStats_offense_weekly_home <- inner_join(
  nfl_actualStats_offense_weekly_long,
  nfl_schedules,
  by = c("season","week","team" = "home_team")) %>%
  mutate(home_away = "home_team")
```

```
nfl_actualStats_offense_weekly_away <- inner_join(
  nfl_actualStats_offense_weekly_long,
  nfl_schedules,
  by = c("season","week","team" = "away_team")) %>%
  mutate(home_away = "away_team")

nfl_actualStats_defense_weekly_home <- inner_join(
  nfl_actualStats_defense_weekly,
  nfl_schedules,
  by = c("season","week","team" = "home_team")) %>%
  mutate(home_away = "home_team")

nfl_actualStats_defense_weekly_away <- inner_join(
  nfl_actualStats_defense_weekly,
  nfl_schedules,
  by = c("season","week","team" = "away_team")) %>%
  mutate(home_away = "away_team")

nfl_actualStats_kicking_weekly_home <- inner_join(
  nfl_actualStats_kicking_weekly,
  nfl_schedules,
  by = c("season","week","team" = "home_team")) %>%
  mutate(home_away = "home_team")

nfl_actualStats_kicking_weekly_away <- inner_join(
  nfl_actualStats_kicking_weekly,
  nfl_schedules,
  by = c("season","week","team" = "away_team")) %>%
  mutate(home_away = "away_team")

# Combine the results of the join operations
nfl_actualStats_offense_weekly_schedules_long <- bind_rows(
  nfl_actualStats_offense_weekly_home,
  nfl_actualStats_offense_weekly_away)

nfl_actualStats_defense_weekly_schedules_long <- bind_rows(
  nfl_actualStats_defense_weekly_home,
  nfl_actualStats_defense_weekly_away)

nfl_actualStats_kicking_weekly_schedules_long <- bind_rows(
  nfl_actualStats_kicking_weekly_home,
  nfl_actualStats_kicking_weekly_away)

# Reshape from long to wide
```

```
player_game_gameday_offense <- nfl_actualStats_offense_weekly_schedules_long %>%
  distinct(player_id, season, week, game_id, home_away, team, gameday) %>% #, .keep_all = TRUE
  pivot_wider(
    names_from = home_away,
    values_from = team)

player_game_gameday_defense <- nfl_actualStats_defense_weekly_schedules_long %>%
  distinct(player_id, season, week, game_id, home_away, team, gameday) %>% #, .keep_all = TRUE
  pivot_wider(
    names_from = home_away,
    values_from = team)

player_game_gameday_kicking <- nfl_actualStats_kicking_weekly_schedules_long %>%
  distinct(player_id, season, week, game_id, home_away, team, gameday) %>% #, .keep_all = TRUE
  pivot_wider(
    names_from = home_away,
    values_from = team)

# Merge player birthdate and the game date
player_game_birthdate_gameday_offense <- left_join(
  player_game_gameday_offense,
  unique(nfl_players[,c("gsis_id","birth_date")]),
  by = c("player_id" = "gsis_id")
)

player_game_birthdate_gameday_defense <- left_join(
  player_game_gameday_defense,
  unique(nfl_players[,c("gsis_id","birth_date")]),
  by = c("player_id" = "gsis_id")
)

player_game_birthdate_gameday_kicking <- left_join(
  player_game_gameday_kicking,
  unique(nfl_players[,c("gsis_id","birth_date")]),
  by = c("player_id" = "gsis_id")
)

player_game_birthdate_gameday_offense$birth_date <- ymd(player_game_birthdate_gameday_offense$birt
player_game_birthdate_gameday_offense$gameday <- ymd(player_game_birthdate_gameday_offense$gameday

player_game_birthdate_gameday_defense$birth_date <- ymd(player_game_birthdate_gameday_defense$birt
player_game_birthdate_gameday_defense$gameday <- ymd(player_game_birthdate_gameday_defense$gameday

player_game_birthdate_gameday_kicking$birth_date <- ymd(player_game_birthdate_gameday_kicking$birt
```

```r
player_game_birthdate_gameday_kicking$gameday <- ymd(player_game_birthdate_gameday_kicking$gameday

# Calculate player's age for a given week as the difference between their birthdate and the game d
player_game_birthdate_gameday_offense$age <- interval(
  start = player_game_birthdate_gameday_offense$birth_date,
  end = player_game_birthdate_gameday_offense$gameday
) %>%
  time_length(unit = "years")

player_game_birthdate_gameday_defense$age <- interval(
  start = player_game_birthdate_gameday_defense$birth_date,
  end = player_game_birthdate_gameday_defense$gameday
) %>%
  time_length(unit = "years")

player_game_birthdate_gameday_kicking$age <- interval(
  start = player_game_birthdate_gameday_kicking$birth_date,
  end = player_game_birthdate_gameday_kicking$gameday
) %>%
  time_length(unit = "years")

# Merge with player info
player_age_offense <-  left_join(
  player_game_birthdate_gameday_offense,
  nfl_players %>% select(-birth_date, -season),
  by = c("player_id" = "gsis_id"))

player_age_defense <-  left_join(
  player_game_birthdate_gameday_defense,
  nfl_players %>% select(-birth_date, -season),
  by = c("player_id" = "gsis_id"))

player_age_kicking <-  left_join(
  player_game_birthdate_gameday_kicking,
  nfl_players %>% select(-birth_date, -season),
  by = c("player_id" = "gsis_id"))

# Add game_id to weekly stats to facilitate merging
nfl_actualStats_game_offense_weekly <- nfl_actualStats_offense_weekly %>%
  left_join(
    player_age_offense[,c("season","week","player_id","game_id")],
    by = c("season","week","player_id"))

nfl_actualStats_game_defense_weekly <- nfl_actualStats_defense_weekly %>%
```

```
  left_join(
    player_age_offense[,c("season","week","player_id","game_id")],
    by = c("season","week","player_id"))

nfl_actualStats_game_kicking_weekly <- nfl_actualStats_kicking_weekly %>%
  left_join(
    player_age_offense[,c("season","week","player_id","game_id")],
    by = c("season","week","player_id"))

# Merge with player weekly stats
player_age_stats_offense <- left_join(
  player_age_offense %>% select(-position, -position_group),
  nfl_actualStats_game_offense_weekly,
  by = c(c("season","week","player_id","game_id")))

player_age_stats_defense <- left_join(
  player_age_defense %>% select(-position, -position_group),
  nfl_actualStats_game_defense_weekly,
  by = c(c("season","week","player_id","game_id")))

player_age_stats_kicking <- left_join(
  player_age_kicking %>% select(-position, -position_group),
  nfl_actualStats_game_kicking_weekly,
  by = c(c("season","week","player_id","game_id")))

player_age_stats_offense$years_of_experience <- as.integer(player_age_stats_offense$years_of_exper
player_age_stats_defense$years_of_experience <- as.integer(player_age_stats_defense$years_of_exper
player_age_stats_kicking$years_of_experience <- as.integer(player_age_stats_kicking$years_of_exper

# Merge player info with seasonal stats
player_seasonal_offense <- left_join(
  nfl_actualStats_offense_seasonal,
  nfl_players %>% select(-position, -position_group, -season),
  by = c("player_id" = "gsis_id")
)

player_seasonal_defense <- left_join(
  nfl_actualStats_defense_seasonal,
  nfl_players %>% select(-position, -position_group, -season),
  by = c("player_id" = "gsis_id")
)

player_seasonal_kicking <- left_join(
  nfl_actualStats_kicking_seasonal,
```

```
  nfl_players %>% select(-position, -position_group, -season),
  by = c("player_id" = "gsis_id")
)

# Calculate age
season_startdate <- nfl_schedules %>%
  group_by(season) %>%
  summarise(startdate = min(gameday, na.rm = TRUE))

player_seasonal_offense <- player_seasonal_offense %>%
  left_join(
    season_startdate,
    by = "season"
  )

player_seasonal_defense <- player_seasonal_defense %>%
  left_join(
    season_startdate,
    by = "season"
  )

player_seasonal_kicking <- player_seasonal_kicking %>%
  left_join(
    season_startdate,
    by = "season"
  )

player_seasonal_offense$age <- interval(
  start = player_seasonal_offense$birth_date,
  end = player_seasonal_offense$startdate
) %>%
  time_length(unit = "years")

player_seasonal_defense$age <- interval(
  start = player_seasonal_defense$birth_date,
  end = player_seasonal_defense$startdate
) %>%
  time_length(unit = "years")

player_seasonal_kicking$age <- interval(
  start = player_seasonal_kicking$birth_date,
  end = player_seasonal_kicking$startdate
) %>%
  time_length(unit = "years")
```

3.20 Plotting

3.20.1 Rushing Yards per Carry By Player Age

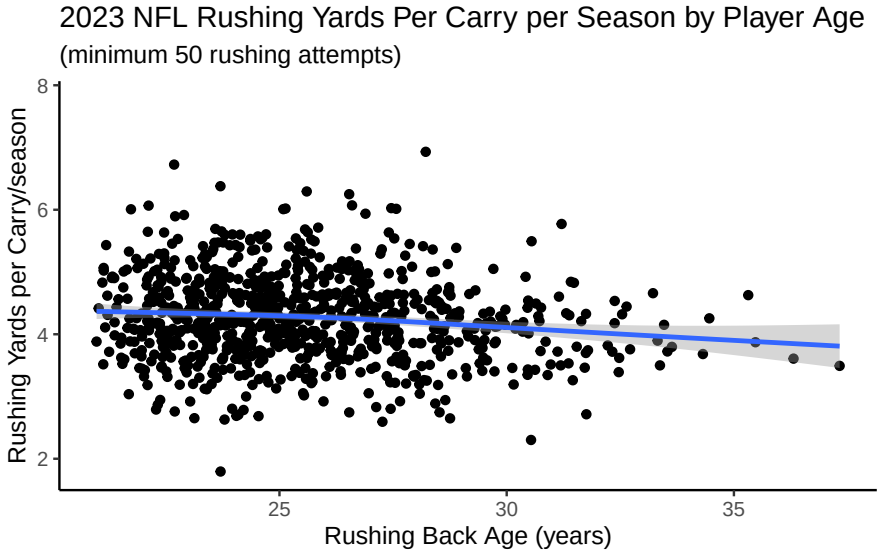
```
# Prepare Data
rushing_attempts <- nfl_pbp %>%
  dplyr::filter(
    season_type == "REG") %>%
    filter(
      rush == 1,
      rush_attempt == 1,
      qb_scramble == 0,
      qb_dropback == 0,
      !is.na(rushing_yards))

rb_yardsPerCarry <- rushing_attempts %>%
  group_by(rusher_id, season) %>%
  summarise(
    ypc = mean(rushing_yards, na.rm = TRUE),
    rush_attempts = n(),
    .groups = "drop") %>%
  ungroup() %>%
  left_join(
    nfl_players %>% select(-season),
    by = c("rusher_id" = "gsis_id")
  ) %>%
  filter(
    position_group == "RB",
    rush_attempts >= 50) %>%
  left_join(
    season_startdate,
    by = "season"
  )

rb_yardsPerCarry$age <- interval(
  start = rb_yardsPerCarry$birth_date,
  end = rb_yardsPerCarry$startdate
) %>%
  time_length(unit = "years")

# Create Plot
ggplot2::ggplot(
```

```
  data = rb_yardsPerCarry,
  ggplot2::aes(
    x = age,
    y = ypc)) +
  ggplot2::geom_point() +
  ggplot2::geom_smooth() +
  ggplot2::labs(
    x = "Rushing Back Age (years)",
    y = "Rushing Yards per Carry/season",
    title = "2023 NFL Rushing Yards Per Carry per Season by Player Age",
    subtitle = "(minimum 50 rushing attempts)"
  ) +
  ggplot2::theme_classic()
```

```
`geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```

```
Warning: Removed 865 rows containing non-finite outside the scale range
(`stat_smooth()`).
```

```
Warning: Removed 865 rows containing missing values or values outside the scale range
(`geom_point()`).
```



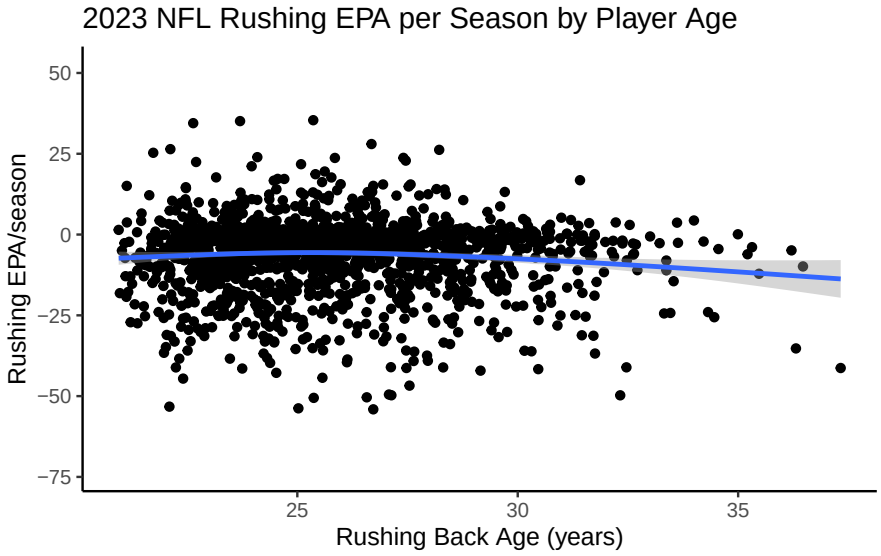

```
# Subset Data
rb_seasonal <- player_seasonal_offense %>%
```

```
  filter(position_group == "RB")

# Create Plot
ggplot2::ggplot(
  data = rb_seasonal,
  ggplot2::aes(
    x = age,
    y = rushing_epa)) +
  ggplot2::geom_point() +
  ggplot2::geom_smooth() +
  ggplot2::labs(
    x = "Rushing Back Age (years)",
    y = "Rushing EPA/season",
    title = "2023 NFL Rushing EPA per Season by Player Age"
  ) +
  ggplot2::theme_classic()
```

```
`geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```

```
Warning: Removed 2415 rows containing non-finite outside the scale range
(`stat_smooth()`).
```

```
Warning: Removed 2415 rows containing missing values or values outside the scale range
(`geom_point()`).
```



3.20.2 Defensive and Offensive EPA per Play

Expected points added (EPA) per play by the team with possession.

```
pbp_regularSeason <- nfl_pbp %>%
  dplyr::filter(
    season == 2023,
    season_type == "REG") %>%
  dplyr::filter(!is.na(posteam) & (rush == 1 | pass == 1))

epa_offense <- pbp_regularSeason %>%
  dplyr::group_by(team = posteam) %>%
  dplyr::summarise(off_epa = mean(epa, na.rm = TRUE))

epa_defense <- pbp_regularSeason %>%
  dplyr::group_by(team = defteam) %>%
  dplyr::summarise(def_epa = mean(epa, na.rm = TRUE))

epa_combined <- epa_offense %>%
  dplyr::inner_join(epa_defense, by = "team")

ggplot2::ggplot(
  data = epa_combined,
  ggplot2::aes(
    x = off_epa,
    y = def_epa)) +
  nflplotR::geom_mean_lines(
    ggplot2::aes(
      x0 = off_epa ,
      y0 = def_epa)) +
  nflplotR::geom_nfl_logos(
    ggplot2::aes(
      team_abbr = team),
      width = 0.065,
      alpha = 0.7) +
  ggplot2::labs(
    x = "Offense EPA/play",
    y = "Defense EPA/play",
    title = "2023 NFL Offensive and Defensive EPA per Play"
  ) +
  ggplot2::theme_classic() +
  ggplot2::scale_y_reverse()
```

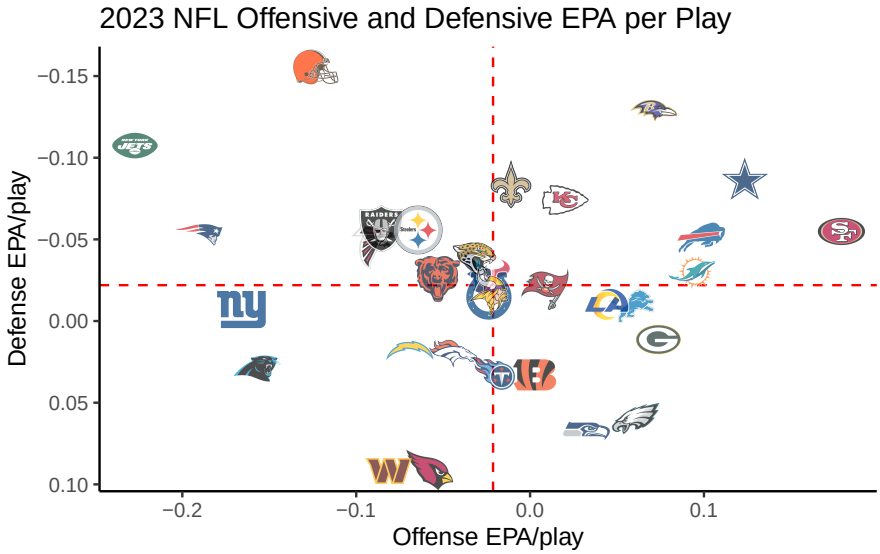


Figure 3.2 2023 NFL Offensive and Defensive EPA per Play

References

Motz, B. (2013). Fantasy football: A touchdown for undergraduate statistics education. *Proceedings of the Games, Learning, and Society Conference*, *9.0*, 222–228. https://doi.org/10.1184/R1/6686804.v1

Index