Introduction

The world of Pokémon began in 1995 with the pair of games Pokémon Red and Green. For Westerners, the latter would become known as Pokémon Blue. These two games introduced the turn-based battling system that became the foundation of the Pokémon games. Numerous other games have attempted to copy the Pokémon battling format, but none have been able to equal its widespread appeal and dedicated player base. With each successive iteration, new items, types, and of course Pokémon are added to the Pokémon lexicon. The world of Pokémon has continued to grow and evolve into one of the largest video games franchises to date. The most recent iterations of Pokémon games, Pokémon Sun and Moon, have continued the tradition of adding layers onto an already complex system of battling.

Since 2011 the program Pokémon Showdown has offered a 'stripped' version of Pokémon games, and is available at http://pokemonshowdown.com/. This 'stripped' version allows players to exclusively battle one another, replicating the most recent iteration of the Pokémon games in the process. This has allowed players to hone and test their Pokémon battling skills through the years. With well-over 20,000 daily registered users and counting, this program has become the go-to program to test and practice Pokémon battling strategies in the ultimate pursuit of becoming the very best that no one ever was.

To begin formal analysis of Pokémon battling however, the battline system must be rigorously detailed for laymen and game theorists alike.

A Brief Overview

First, there needs to be a formal model. At its core, Pokémon battling takes place exclusively between two players. Each of the two players has a team composed of six different Pokémon. Depending on the battle format, the team of six Pokémon is either dictated by the player or randomly assigned. Regardless of format, each turn each player simmultaneously makes a decision. The decisions are then executed, with priority given first to priority moves and then, if neither player decided on a priority move, by a comparative assessment of each Pokémon's speed. Once a player's Pokémon loses all of its health, an event referred to as fainting, the player must switch into another Pokémon. If a player has no other Pokémon to switch into, the battle ends. As would be expected, the player whose Pokémon have all fainted loses the battle, and the other player wins.

There are a number of parameters and variables to consider beyond those already detailed. To begin with, each Pokémon has at most four moves to chose from for any turn of a battle. Additionally, whenever a player has at least two Pokémon that have not fainted, they have the choice to switch into another Pokémon. Doing so counts as the player's action for the turn. Thus, players almost always have five different choices to make each turn. However, if switches are considered distinct choices, the number of potential decisions is at most nine for a given turn. Hence, for the purposes of analysis, the supremum of decisions for any given turn is nine.

There are a number of other parameters to consider, though these parameters are case-specific, i.e. dependent upon Pokémon, held item, Pokémon ability, etc. Such parameters will be further detailed in the methodology section, along with further specifications on specific moves and Pokémon types.

That being said, Pokémon must be formally denoted as a specific type of game in game theoretic terms. With this in mind, it bears noting that there are only two outcomes to any battle: One player wins and the other loses. Because of this, Pokémon is by definition a zero-sum game. However, it is important to note that there are distinct ways that a player may win or lose a game. Notably, the same set of decisions made by one player over multiple battles may not lead to the same outcome in each battle. That is to say, uniqueness of strategies is derived from the decisions of both players for any given battle.

A point glossed over previously is the existence of different end games. Though the game ends when all of the Pokémon on one team have fainted, any player has the choice to forfeit the game any time before this occurs. Thus, the two ways to win or lose a battle are either by forfeiture or by having all of their Pokémon faint; the latter outcome is referred to as a "normal" game. Additionally, there is the potential for a game to result in a draw. However, draws result in a neutral payoff as neither player raises or lowers their ranking after a draw.

Within the scope of game theoretic terms, it is vital to note that each player is able to see all past decisions made over the course of a battle, and that battles span over multiple turns. Speaking to the former point, players are able to recall not only their past decisions but also those of their opponent, including how much damage was done by a specific move during a previous turn. As this information is available at any time during the battle, Pokémon battling is a perfect recall game. Furthermore as each game is composed of a sequence of turns, Pokémon battling is also a sequential game.

The points noted on information allude to a unique trait of Pokémon battling that further details the type of game Pokémon battling is: An incomplete information game. Though the extent of incompleteness is format specific, each player is given limited information about the opposing players team at the beginning of the battle. After each turn, players are able learn not only the different moves each opposing Pokémon has, but also their abilities, held items, and sometimes what other Pokémon compose the opposing team. Generally speaking, there is almost always new information revealed each turn. The extent to which player strategies are revealed from information is an ongoing topic of discussion, both here and in current game theory literature. Naturally, the analysis of such a topic relies upon at least one apriori assumption, namely that player strategies are revealed by their past decisions. However, whether player strategies are rational and consistent is another matter entirely; nonetheless, at the very least it is important to note that incorporating such information into players own decision-making complicates analysis and is a central topic in analyzing Pokémon battling.

Taken together, Pokémon battling is a special case of a sequential zero sum two player game with imcomplete information and perfect recall. As Pokémon battling lends itself to a discussion of imperfect information, it is all the more vital to consider the role of decision making in sequential games. A great portion of game theory literature has explored such topics, including the revolutionary work of John Forbes Nash Jr.

Theoretics aside, there are also notes on the computation side of analyzing Pokémon. In this regard, my initial computations and analysis focus on the distributions of turn length based on win type, i.e. the length of games played when one player forfeited or had all of their Pokémon faint. After distributional computations I categorize moves by their type, specifically whether a specific move is damaging, non-damaging, or a mix. After categorizing each move individually, I then categorize sets of four moves, again using categorizations of damaging, non-damaging, or a mix. Using this categorization I gauge the effectiveness of specific move sets based on their ability to faint Pokémon and by their likelihood to increase a players probability of winning.

Overall I hope explore how information gathered turn-by-turn feeds into the decision-making process of Pokémon battling. By incorporating tenants of behavioral economics and game theory, I hope to rigorously analyze and detail phenomena that occur in Pokémon battling. However, even more specifically I want to verify the theoretical findings of Pokémon battling in a game theoretic framework, specifically the existence of at least one mixed Nash equilibrium.

Literature Review

A few points on Pokémon battling are worth noting as they relate Nash's literature on game theory mentioned previously. First, each Pokémon battle lasts a finite number of turns. Furthermore, each player has finitely many choices to make at any given turn. Thus there exists finitely many pure strategies for players to choose from each game. Thus via application of Nash's Existence Theorem, Pokémon battling has at least one Nash equilibrium (Nash 1950).

The question then is to determine what is the Nash equilibrium, possibly equilibria, in Pokémon battling. To address this question, it will be necessary to consider other propositions Nash made in relation to non-cooperative games as it bears on the existence of mixed strategies. Namely, the goal of analyzing Pokémon battling is to determine solutions, strong solutions, and sub-solutions. Importantly, non-cooperative games do not always have a solution, though they always have sub0solutions (Nash 1951). Thus, a central question will be in determining whether Pokémon battling has a solution, and if so, if it is unique. Furthermore, it will be important to determine what sub-solutions exist, along with the multiplicity of such solutions.

That being said, scant rigorous or academic research has been conducted within the scope of Pokémon-related topics. The most frequent publications focus either on Pokémon as a cultural phenomena or on strategy guides

for Pokémon games, not specifically Pokémon battling strategies. Nonetheless some of these publications have some focus on gameplay, though not to the extent of incorporating game theoretic terminology. That isn't to say that there aren't academic publications focusing on Pokémon battling.

Typically academic papers focusing on Pokémon battling have focused on the use of algorithms to simulate and play against human players in Pokémon Showdown. One paper gives a rudimentary background on Pokémon battling and focuses explicitly on 1v1 battles (Gildardo 2013). Recent literature following this included more nuance, specifically by expanding the teams to six. The most relevant work related to my analysis focused on what was at its time the most recent Pokémon battling framework (Ho et al. 2016). Though the paper focused on what is currently the previous iteration of Pokémon Showdown, the framework is fundamentally the same as that included in the data for this research. Notably, the website for Pokémon Showdown provides a hub for information on Pokémon battling basics and specific battle format descriptions. Replays and ladder ranking are available for public viewing, along with links to usage statistics and a damage calculator.

Unsurprisingly game theory vernacular has not entered into the discussion of Pokémon battling strategies, at least in any formal setting. Applying such concepts to the context of Pokémon battling offers a formal framework to discuss strategies and test hypotheses. As mentioned briefly in the introduction, there are three main areas of game theory that intersect in the analysis of Pokémon battling. These three areas of interest include the interpretation of Pokémon battling as a zero sum game, the role of incomplete information, and the implications of Pokémon being a non-cooperative game. These three topics actively influence the decision-making process found in Pokémon battling.

A central factor involved in the decision making process as it relates to game theory is the role of information, specifically how players take into account imperfect and/or incomplete information. By its nature information is revealed each turn such as the four moves an opposing Pokémon has, its ability, its held item, and what other Pokémon the opposing player has on their team. As a result Pokémon is not a perfect information game. As such Zermelo's theorem implies that there does not exist a pure Nash equilibrium; this does not exclude the possibility of a winning strategy though (Schwalbe et al. 2001).

Both imperfect and incomplete information apply to Pokémon battling and reveal insight into decision making process of players. An especially interesting addition to this topic is the implications of assymmetrical information. In this regard, the concepts of sunk costs and signalling enter into the equation. Especially for the Random battle format, each player is given minimal information at the onset of the game. After each turn, the player gains more information. When players receive this information they add it into their decision making process, though doing so it not always advantageous.

Concepts of game theory are not the only relevant ideas for analyzing Pokémon battling. In this vein, exploring whether a player will switch Pokémon will necessarily invoke ideas found in both game theory and behavioral economics. One specific concept engrained in behavioral economics that relates to Pokémon battling is the idea of "keeping doors open". In Chapter 6 of Ariel's work Predictably Irrational, results indicates that players prefer to keep options available even if doing so incurs costs and/or reduces their payoff. In the context of Pokémon battling, players may decide to preemptively switch Pokémon in the hopes of having that Pokémon later in the battle. However, whether this adversely influences the player is subject to inquiry.

A an ending point of interest, taking into account an opposing players strategy as deduced from the information revealed each turn can easily rub against the issues of level-K thinking. Knowing that a player is likely to repeat the same decision made the previous turn, players can optimize their decision assumming the same decision is going to be made. However, if the opposing player incorporates such knowledge into their decision the turn the player decides to act off the opposing player's status quo decision, suboptimal decisions may actually be made. That being said, the inclusion of level-k thinking is closely aligned to recent behavioral game theory literature. A central finding of this literature has emphasized the role of iterated reasoning, which essentializes player adaptation to other players decision making (Wunder et al. 2011).

References

- Ariely, D. (2009). Predictably irrational: The hidden forces that shape our decisions. New York, NY: Harper. Chapter 8: Keeping Doors Open (pp. 139-154).
- Schwalbe, U., & Walker, P. (2001). Zermelo and the Early History of Game Theory. Games and Economic Behavior, 34(1), 123-137. doi:10.1006/game.2000.0794
- Smogon University (n.d.). Retrieved October 31, 2016, from http://www.smogon.com/
- Pokémon Showdown! battle simulator (n.d.). Retrieved October 31, 2016, from http://pokemonshowdown.com/
- Pokémon Showdown Github Master Repository (2016). Zarel/Pokemon-Showdown. Retrieved October 31, 2016, from https://github.com/Zarel/Pokemon-Showdown/tree/master/data
- Wunder, M., Kaisers, M., Yaros, J., Littman, M., (May, 2011) Using Iterated Reasoning to Predict Opponent Strategies. Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011), Tumer, Yolum, Sonenberg and Stone (eds.), 593-600.
- Ho, H., Ramesh, V. (2016) Percymon: A Pokemon Showdown Artifical Intelligence. Retrieved October 31, 2016, from: http://robots.stanford.edu/cs221/2016/restricted/projects/vramesh2/final.pdf
- Sanchez-Ante, Gildardo (Dec., 2013) Sistemas Inteligentes: Reportes Finales Ago-Dic 2013. Retrieved October 31, 2016, from: https://www.researchgate.net/profile/Gildardo_Sanchez-Ante/publication/259343975_Sistemas_Inteligentes_Reportes_Finales_Ago-Dic_2013/links/0c96052b1d0b582e95000000.pdf#page=140
- Pokémon Company International (Nov. 21, 2014) Pokémon Omega Ruby & Pokémon Alpha Sapphire: The Official Hoenn Region Strategy Guide.
- Nash, J. F., Jr. (1950, January 15). Equilibrium Points in n-Person Games. Proceedings of the National Academy of Sciences of the United States of America, 36(1), 48-49. Retrieved November 29, 2016, from http://www.sscnet.ucla.edu/polisci/faculty/chwe/austen/nash1950.pdf
- Nash, J. F., Jr. (1951, September). Non-Cooperative Games. The Annals of Mathematics, Second Series, 54(2), 286-295. Retrieved November 29, 2016, from http://lcm.csa.iisc.ernet.in/gametheory/Classics/NCG.pdf

Methodology

The data used is a compilation of battle logs from the Pokémon Showdown servers. Each battle log is stored as a separate .json file. The data spans across the year 2015, composed of four different months of data. The four months are March, June, September, and December. There are no lapses in data, i.e. each day of each month has numerous battle logs to account for. No dramatic overhaul was done to the Pokémon battling format at this time, though some minor adjustments were made. Furthermore only ranked games are included in the dataset, indicating that each player stands to gain or lose from the battle.

Specific usage statistics are found in a subsidiary website, found at http://sweepercalc.com/stats/. The usage statistics track the frequency of use for specific Pokémon, items, abilities, and a host of other relevant variables for Pokémon battling.

A number of links redirect users to the host site of this game: Smogon University. The host site can be found at http://www.smogon.com/. This website offers a wide variety of resources, similar to those found at the Pokémon Showdown website. Most importantly the Smogon forums are a prominent site for discussion of Pokémon battling strategies.

Pokémon Battling Basics

The Pokémon battle starts with Pokémon being sent out. For the purposes of the data used, one Pokémon is sent out for each opponent, totalling two Pokémon being out at any given time. Following this, each Pokémon has 4 moves to choose from, along with the option to switch to a different Pokémon (when applicable). After both players make a decision, the moves are weighted for priority and speed to determine the order of play. If both players decide not to switch one Pokémon will attack the other, after which the next Pokémon will do

the same if it has not fainted. After each move has been executed the turn ends and the process is repeated. When one of the Pokémon faints, the player whose Pokémon fainted will be prompted to select another Pokémon from the bench. The first player to lose all of their Pokémon loses the battle.

Battle Formats

The data used for this study include two different Pokémon battling formats. The two formats are known as Over Used and Random Battles, abbreviated as OU and Randbats respectively. Both formats have teams of six Pokémon and only allow one Pokémon to be out at any given time. While both battle formats are subsets of what are known as single battles, each has their own unique spin on the Pokémon battling format.

Random Battles are the most frequently played format. Neither player gets to decide on their initial Pokémon nor do they have any input on the composition of the team. The format uses an algorithm to determine team compositions. However it is important to note that there are restrictions to the Randbats format that center around team composition and move composition for specific Pokémon.

The Over Used battle format includes team composition. By including team composition, players are able to decide what Pokémon to include on their team, the moves of each Pokémon, and other factors such as held items and abilities. Further restrictions to the OU format include banning specific Pokémon. The restricted Pokémon are included in the "Uber" tier along with the Pokémon Mega-Rayquaza. Additionally certain "hidden" abilities are locked for Pokémon.

Pokémon Attributes

Generally, there are a number of factors that are specific to each Pokémon. Some of these factors are considered static, meaning that they do not change over the course of the battle. These types of factors are noted as "Fixed" Attributes. However there are some factors that are generally regarded as Fixed Attributes but are affected by certain moves, at least when the Pokémon is sent out. These types of factors are considered "Mixed Attributes". Additionally there are attributes that are inherently influenced throughout the course of the battle. These are noted as "Variable Attributes". The terminology is largely taken from the Ho et al. paper for ease of translation.

Pokémon Fixed Attributes

Fixed attributes include the type(s) of the Pokémon, the four moves the Pokémon has learned, the one item the Pokémon holds, the Pokémon's one selected ability, the level of the Pokémon, and the Pokémon's baseline stats. The latter factor is divided into six categories. These categories include (baseline) Health, Attack, Special Attack, Defense, Special Defense, and Speed. There is further nuance with the inclusion of Pokémon natures and Individual Values, or IVs. These factors influence the base stats of each Pokémon. However due to the sheer number of trivial combinations of IV spreads and nature choices, these two factors will not be a pivotal aspect of framework used.

Pokémon Types

The type(s) of each Pokémon influence not only the potential weaknesses of each Pokémon, but also influence the amount of damage certain type-specific moves are able to do. Each Pokémon has at least one and at most two types. If a Pokémon uses a damaging move whose type corresponds to type of the Pokémon that used it, that Pokémon gets a same type attack bonus, abbreviated as a "stab" bonus. This causes the move to do 50% more damage, potentially 100% if the Pokémon also has the ability Adaptability.

Pokémon Mixed Attributes

Pokémon Variable Attributes

Variable Attributes include the current health of the Pokémon, the status of the Pokémon, the volatile status of the Pokémon, boost data of the Pokémon, and whether the Pokémon in question is currently active.

End thesis Chapter 1

MARKDOWN TEMPLATE

R Markdown Basics

Here is a brief introduction into using *R Markdown*. *Markdown* is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. *R Markdown* provides the flexibility of *Markdown* with the implementation of **R** input and output. For more details on using *R Markdown* see http://rmarkdown.rstudio.com.

Be careful with your spacing in *Markdown* documents. While whitespace largely is ignored, it does at times give *Markdown* signals as to how to proceed. As a habit, try to keep everything left aligned whenever possible, especially as you type a new paragraph. In other words, there is no need to indent basic text in the Rmd document (in fact, it might cause your text to do funny things if you do).

Lists

It's easy to create a list. It can be unordered like

- Item 1
- Item 2

or it can be ordered like

1. Item 1
2. Item 2

Notice that I intentionally mislabeled Item 2 as number 4. *Markdown* automatically figures this out! You can put any numbers in the list and it will create the list. Check it out below.

To create a sublist, just indent the values a bit (at least four spaces or a tab). (Here's one case where indentation is key!)

1. Item 1
2. Item 2
3. Item 3
    - Item 3a
    - Item 3b

Line breaks

Make sure to add white space between lines if you'd like to start a new paragraph. Look at what happens below in the outputted document if you don't:

Here is the first sentence. Here is another sentence. Here is the last sentence to end the paragraph. This should be a new paragraph.

Now for the correct way:

Here is the first sentence. Here is another sentence. Here is the last sentence to end the paragraph.

This should be a new paragraph.

R chunks

When you click the **Knit** button above a document will be generated that includes both content as well as the output of any embedded **R** code chunks within the document. You can embed an **R** code chunk like this (`cars` is a built-in **R** dataset):

```
summary(cars)
```

```
##      speed           dist
##  Min.   : 4.0   Min.   :  2.00
##  1st Qu.:12.0   1st Qu.: 26.00
##  Median :15.0   Median : 36.00
##  Mean   :15.4   Mean   : 42.98
##  3rd Qu.:19.0   3rd Qu.: 56.00
##  Max.   :25.0   Max.   :120.00
```

Inline code

If you'd like to put the results of your analysis directly into your discussion, add inline code like this:

> The `cos` of 2π is 1.

Another example would be the direct calculation of the standard deviation:

> The standard deviation of `speed` in `cars` is 5.2876444.

One last neat feature is the use of the `ifelse` conditional statement which can be used to output text depending on the result of an **R** calculation:

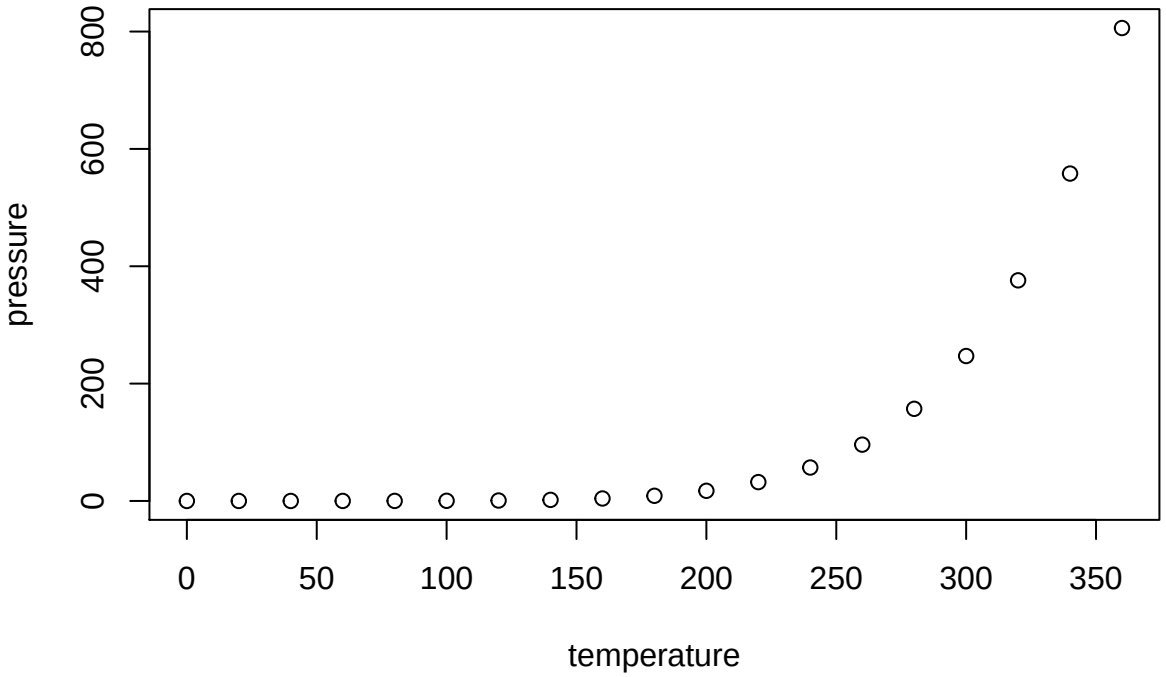
> The standard deviation is less than 6.

Note the use of `>` here, which signifies a quotation environment that will be indented.

As you see with `$2 \pi$` above, mathematics can be added by surrounding the mathematical text with dollar signs. More examples of this are in [Mathematics and Science] if you uncomment the code in [Math].

Including plots

You can also embed plots. For example, here is a way to use the base **R** graphics package to produce a plot using the built-in `pressure` dataset:



Note that the `echo=FALSE` parameter was added to the code chunk to prevent printing of the **R** code that generated the plot. There are plenty of other ways to add chunk options. More information is available at http://yihui.name/knitr/options/.

Another useful chunk option is the setting of `cache=TRUE` as you see here. If document rendering becomes time consuming due to long computations or plots that are expensive to generate you can use knitr caching to improve performance. Later in this file, you'll see a way to reference plots created in **R** or external figures.

Loading and exploring data

Included in this template is a file called `flights.csv`. This file includes a subset of the larger dataset of information about all flights that departed from Seattle and Portland in 2014. More information about this dataset and its **R** package is available at http://github.com/ismayc/pnwflights14. This subset includes only Portland flights and only rows that were complete with no missing values. Merges were also done with the `airports` and `airlines` data sets in the `pnwflights14` package to get more descriptive airport and airline names.

We can load in this data set using the following command:

```
flights <- read.csv("data/flights.csv")
```

The data is now stored in the data frame called `flights` in **R**. To get a better feel for the variables included in this dataset we can use a variety of functions. Here we can see the dimensions (rows by columns) and also the names of the columns.

```
dim(flights)
```

```
## [1] 52808    16
```

```
names(flights)
```

```
##  [1] "month"        "day"          "dep_time"     "dep_delay"
##  [5] "arr_time"     "arr_delay"    "carrier"      "tailnum"
##  [9] "flight"       "dest"         "air_time"     "distance"
## [13] "hour"         "minute"       "carrier_name" "dest_name"
```

Another good idea is to take a look at the dataset in table form. With this dataset having more than 50,000 rows, we won't explicitly show the results of the command here. I recommend you enter the command into the Console ***after*** you have run the **R** chunks above to load the data into **R**.

```
View(flights)
```

While not required, it is highly recommended you use the `dplyr` package to manipulate and summarize your data set as needed. It uses a syntax that is easy to understand using chaining operations. Below I've created a few examples of using `dplyr` to get information about the Portland flights in 2014. You will also see the use of the `ggplot2` package, which produces beautiful, high-quality academic visuals.

We begin by checking to ensure that needed packages are installed and then we load them into our current working environment:

```
# List of packages required for this analysis
pkg <- c("dplyr", "ggplot2", "knitr", "bookdown", "devtools")
# Check if packages are not installed and assign the
# names of the packages not installed to the variable new.pkg
new.pkg <- pkg[!(pkg %in% installed.packages())]
# If there are any packages in the list that aren't installed,
# install them
if (length(new.pkg))
  install.packages(new.pkg, repos = "http://cran.rstudio.com")
# Load packages (thesisdown will load all of the packages as well)
library(thesisdown)
```

The example we show here does the following:

- Selects only the `carrier_name` and `arr_delay` from the `flights` dataset and then assigns this subset to a new variable called `flights2`.
- Using `flights2`, we determine the largest arrival delay for each of the carriers.

```
flights2 <- flights %>%
  select(carrier_name, arr_delay)
max_delays <- flights2 %>%
  group_by(carrier_name) %>%
  summarize(max_arr_delay = max(arr_delay, na.rm = TRUE))
```

A useful function in the `knitr` package for making nice tables in *R Markdown* is called `kable`. It is much easier to use than manually entering values into a table by copying and pasting values into Excel or LaTeX. This again goes to show how nice reproducible documents can be! (Note the use of `results="asis"`, which will produce the table instead of the code to create the table.) The `caption.short` argument is used to include a shorter title to appear in the List of Tables.

```
kable(max_delays,
      col.names = c("Airline", "Max Arrival Delay"),
      caption = "Maximum Delays by Airline",
      caption.short = "Max Delays by Airline",
      longtable = TRUE,
      booktabs = TRUE)
```

Table 1: Maximum Delays by Airline

Airline	Max Arrival Delay
Alaska Airlines Inc.	338
American Airlines Inc.	1539
Delta Air Lines Inc.	651
Frontier Airlines Inc.	575
Hawaiian Airlines Inc.	407
JetBlue Airways	273
SkyWest Airlines Inc.	421
Southwest Airlines Co.	694
United Air Lines Inc.	472
US Airways Inc.	347
Virgin America	366

The last two options make the table a little easier-to-read.

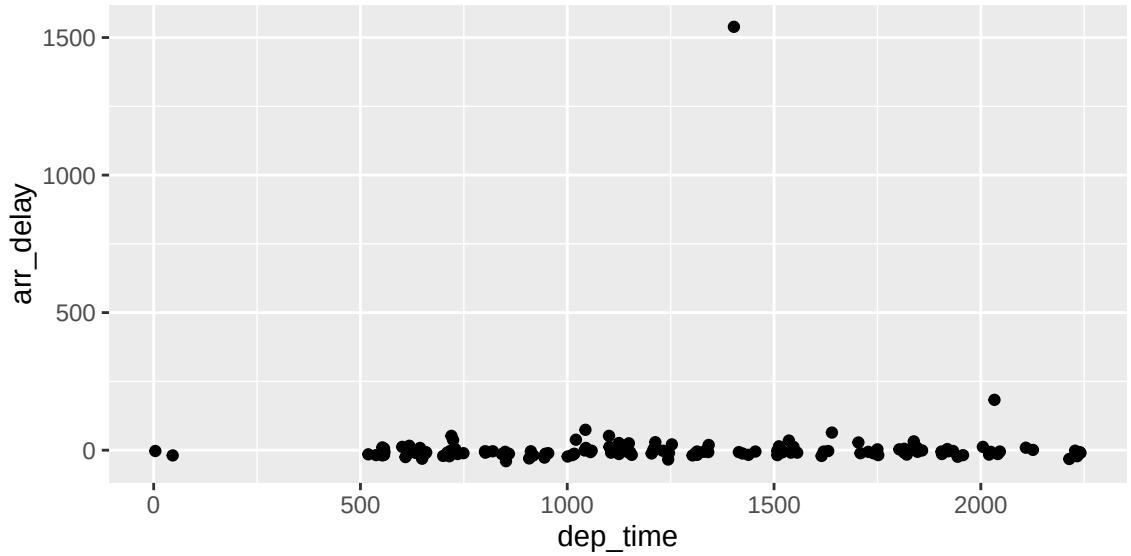
We can further look into the properties of the largest value here for American Airlines Inc. To do so, we can isolate the row corresponding to the arrival delay of 1539 minutes for American in our original `flights` dataset.

```
flights %>% filter(arr_delay == 1539,
                   carrier_name == "American Airlines Inc.") %>%
  select(-c(month, day, carrier, dest_name, hour,
            minute, carrier_name, arr_delay))
```

```
##   dep_time dep_delay arr_time tailnum flight dest air_time distance
## 1     1403      1553     1934  N595AA   1568  DFW      182     1616
```

We see that the flight occurred on March 3rd and departed a little after 2 PM on its way to Dallas/Fort Worth. Lastly, we show how we can visualize the arrival delay of all departing flights from Portland on March 3rd against time of departure.

```
flights %>% filter(month == 3, day == 3) %>%
  ggplot(aes(x = dep_time, y = arr_delay)) + geom_point()
```



Additional resources

- *Markdown* Cheatsheet - https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet
- *R Markdown* Reference Guide - https://www.rstudio.com/wp-content/uploads/2015/03/rmarkdown-reference.pdf
- Introduction to `dplyr` - https://cran.rstudio.com/web/packages/dplyr/vignettes/introduction.html
- `ggplot2` Documentation - http://docs.ggplot2.org/current/