

PROJECT PROPOSAL



(Sports Betting Analysis Tool)

<https://github.com/cis3296f22/bettR>

<https://bettR.shinyapps.io/bettR/>

Arun Agarwal
Jared Stefanowicz
Anubhav Kundu
Kevin Jang

Table of Contents

Project Proposal	3
Project Abstract	3
High Level Requirement	3
Conceptual Design	5
Proof of Concept	5
Background	6
Required Resources	6
Project Design	6
Vision	6
Persona Joe, college student with interests in sports betting	7
Persona Alex, a fantasy football league player	7
Persona Michael, a middle-age project management leader	7
Persona John, recent Statistics college graduate	8
Feature List	9
Class Diagram	10
Sequence Diagram	11
Automated Test Results	13
Project Progress	14
Week 2 Progress	15
Week 3 Progress	16
Week 4 Progress	19

Project Proposal


Project Abstract

With the legalization of sports betting, many sports fans are putting their money where their mouth is for the first time. It can be argued that most users lose money, however, due to following useless heuristics or pundits. The goal of this project is to create a web dashboard displaying the current spreads for sports games along with models from multiple publicly available sites side-by-side along with their relative historical performances. Users would also be able to enter their bets on the site and receive what their expected payout/loss and how accurate they need to be to break even. This site would allow users to have statistically informed insight into their betting and help society mitigate the negative externalities of betting through education.

High Level Requirement

Describe the requirements – i.e., what the product does and how it does it from a user point of view – at a high level.

The product displays a list of games for the day for the user to look through. Upon seeing a game that the user wishes to bet on, the user will then select that game from a dropdown box and the odds of this game's winners and losers will be displayed from another screen within the app that the user can navigate to. Once the user reaches the screen on the specific game they wish to place a bet on, the user would then input the total bets of that game and how much they intend to place on that game on which outcome. From there, the app will then calculate the probabilities and expected return on that bet and display it to the user for their analysis and confidence.



[Games](#) [Predictions](#) [Resources](#) [About](#)

←


Week 4
 (Nov 7 - 13 2022)

→


Mon, November 7 2022

 Brooklyn Nets


View

 Philadelphia 76ers


View

 Brooklyn Nets


View

 Philadelphia 76ers


View

 Brooklyn Nets


View

 Philadelphia 76ers

View


 Brooklyn Nets

View


 Philadelphia 76ers

View


Fri, November 11 2022

 Brooklyn Nets


View

 Philadelphia 76ers


View

 Brooklyn Nets


View

 Philadelphia 76ers


View

 Brooklyn Nets


View

 Philadelphia 76ers

View

 Brooklyn Nets

View

 Philadelphia 76ers


View


[Games](#) [Predictions](#) [Resources](#) [About](#)


Past Year




Source	Start Date	90 Day Accuracy	1 Year Accuracy	Overall Accuracy
Betr Composite	11-04-2022	82%	82%	82%
ESPN BPI	09-01-2015	71%	77%	75%
FiveThirtyEight	09-01-2015	78%	80%	84%
TeamRankings	09-01-2015	63%	72%	70%


[Games](#) [Predictions](#) [Resources](#) [About](#)

November 7th, 2022





New York Nets at Philadelphia 76ers
 7:00 PM
 Wells Fargo Center
 Philadelphia, PA

Source	Home Win %	Away Win %	90 Day Accuracy	1 Year Accuracy	Overall Accuracy
Betr Composite	78%	22%	82%	82%	82%
ESPN BPI	85%	15%	71%	77%	75%
FiveThirtyEight	82%	18%	78%	80%	84%
TeamRankings	74%	16%	63%	72%	70%

Bet Screener

Amount

Spread

Probability

☒ Use Betr Projection

\$150

-500

78%

Expected Winnings
\$ -9.60
 Weekly for 1 Month: ~~\$-38.40~~
 Weekly for 6 Months: ~~\$-249.60~~
 Weekly for 1 Year: ~~\$-499.20~~
 Take -354 Odds or Lower

Conceptual Design

We will create a website/web app to inform users about sports betting. The landing page has a generic article about how hard it is to make money on sports betting. The next page displays what different publicly available websites say the statistical odds are of different games for the current day. Lastly, a third page allows users to input their bets and receive an output of their projected winnings/losses over time.

Python with Selenium will be used to scrape model outputs from the web. There are a number of APIs for grabbing live sports odds. One that we plan to use is linked below. The data will then be processed in R and output into an interactive web dashboard using RShiny.

<https://rapidapi.com/theoddsapi/api/live-sports-odds/>

<https://shiny.rstudio.com>

<https://dash.plotly.com/introduction>

We realized that a big differentiator for us is going to be: 1) Combining data from multiple sources, and 2) Combining and creating really easy betting explanations. There are a lot of tools online, but they require you to have advanced knowledge and enter your own probabilities. We are going to focus more on comparing model performances across multiple sites and visualizing that alongside our aggregated visual models. Another differentiating project idea we have is the ability to make a “Betting Wrapped” reporting feature, similar to Spotify Wrapped. It would allow a user to log in and import their historical bets, and we would grade how well they performed statistically and how lucky the user got, being above or below the expected value.

Proof of Concept

Linked below are galleries of RShiny and Dash dashboards and an RShiny example repo.

<https://shiny.rstudio.com/gallery/>

<https://dash.gallery/Portal/>

<https://github.com/gpilgrim2670/SwimMap>

Here is a link to an old Amazon web scraper Jared made using Selenium and a web scraper used to pull sports forecasts from FiveThirtyEight.

<https://github.com/jaredStef/AmazonScraper>

https://github.com/jmhayes3/fivethirtyeight_scraper

Instructions on how to run code:

1. Install R-Studio Desktop Open Source Edition on your preferred platform [here](#)
2. Clone the Git Repository

3. Open the app.R file in RStudio
4. Press the green Run App button on the top right of the code editing section
5. Bettr will run and open in your browser
6. Instructions on how to use BettR are/will be provided within the application/browser

NOTE: Steps remain the same for Windows, Mac, and Unix.

Background

There are existing sites which provide individual models of what team is going to win. Our web app would show all the models from multiple websites. There are betting odds calculators which take in the odds, your bet, and win probability to give you expected payoff. These tools are simplistic and most users don't know what these values actually are. Our website would allow users to get a realistic picture of their long-term betting prospects (most likely losing money) and allows them to compare themselves to statistical consensus.

<https://www.vegasinsider.com/betting-odds-calculator/>
<https://projects.fivethirtyeight.com/2022-nfl-predictions/games/>
<https://www.fangraphs.com/livescoreboard.aspx>

Required Resources

We need Python, R, the required libraries (pandas, beautifulsoup, tidyverse, and others), and an internet connection. No hardware is required. We may potentially need to purchase a VSC to have our data be updated daily automatically. For any packages and libraries necessary to run the app, you may use the command `<install.packages('NAME_OF_PACKAGE')>` inside the console for R. If you are running the app from RStudio, RStudio will recognize what libraries are necessary to install and will prompt you for approval to install from the top of the code editing section of the IDE.

Project Design

Vision

FOR amateur sports betters and enthusiasts

WHO need statistically informed sports betting predictions and resources to improve

BETTR is a website

THAT provides game predictions, profitability calculators, and educational resources

UNLIKE Odds Jam, FiveThirtyEight, and ESPN

OUR PRODUCT displays and compares information from multiple sources across sports.

Personas

Persona Joe, college student with interests in sports betting

Joe is a college student who just got into sports betting through his friends. He doesn't fully understand betting spreads or have a solid strategy, but he feels like he has a good understanding of his team. He bets based on what industry experts have been saying on ESPN+ and The Athletic. He realizes at the end of the month that he has been losing money, but he decides to keep with it. He finds BettR online and uses it to screen his bets, learn more, and compare predictions from multiple sites.

(Jared)

Persona Alex, a fantasy football league player

Alex has been part of a fantasy football league among his friends. He's been seeing a lot of success with his league and has tried to get into sports betting to test the waters. While he has seen some success, he has also had some failures. In order to better understand what bets he should make, Alex has decided to check out some free web apps like BettR to learn what he's doing right and wrong.

(Kevin)

Persona Michael, a middle-age project management leader

Michael is a middle-age project management leader at a mid-size technology company. He majored in General business in college and performed relatively average. From middle school onwards, he displayed a strong interest in all sports, especially Football and Basketball. He started by just watching the games that were displayed on his TV, and moved on to playing the sports himself and understanding the specific details of what makes players and teams perform well. In college, he learned about the idea of sports betting. He already had some interest in gambling, so it became a hobby for him to learn (and later participate in) sports betting. He had to take an introduction to Python course in college, and it was through that course that he realized the possibilities of analyzing sports through computational means. While he never pursued a career in Computer Science/Sports Analysis, he picked up a few skills on the side while reading about sports betting. Once he got to work at a technology company as a project manager, he had to be able to understand a lot of technical terminology in order to maintain his position. Thus, he ended up building his foundational coding skills to the point where he might be able to do some very simple data cleaning on his own. However, it was not nearly enough for him to create sports betting models on his own nor through online tutorials.

While Michael is not capable of getting into the backend side of sports betting, he knows sports betting tools already exist online such as 'Odds' and 'ESPN'. Unfortunately, he finds some of these a bit complex to use and sometimes notices differences among various sports betting websites. He doesn't know why this is, and more importantly, he is worried that he is not catching all the important distinctions that may exist between the sites. Understanding these differences/distinctions may make the difference between winning and losing a sport's wager. Michael discovers a new player in the sports betting application field, BettR. He loves that the program immediately provides the user safe betting tips and helpful information such as how to understand betting data, something he always struggled

with. Furthermore, BettR compares the sports data off of multiple of the main sports betting applications, which allows Michael to easily understand the most important information he needs as well as be able to analyze why distinctions exist. As he is someone that likes to make \$10-\$100 sports wagers with his friend group, this application might make a large difference in his performance with betting against his friends. Michael sees the amazing potential in BettR to serve as a starting point for individuals entering the sports betting arena, and he imagines his sport's betting skills growing as the site grows with additional features (once they are added/improved upon in the future).

(Arun)

Persona John, recent Statistics college graduate

John recently got out of college with a degree in Statistics. John is also a fan of sports betting and wants to use his statistics knowledge to be successful. He is looking for a simple web application that will make it easy for him to get his data. John turns to BettR, he uses his knowledge of statistics, the helpful tips from BettR, and the data from BettR to make smart betting decisions.

(Anubhav)

Feature ListAggregated Game Predictions:

Users will be able to view our aggregated predictions. These will be created by scraping predictions from multiple sites across the web (ESPN, TeamRankings, FiveThirtyEight, etc.) and weighting them based on their historical accuracy. We will also show which sites perform best on their own. This is the main differentiating feature of the site. It is inspired by FiveThirtyEight's political polling predictions, which weigh multiple polls similarly. (Example linked below) FiveThirtyEight also put out an article comparing their predictive rankings to ESPN's, but not historical accuracy.

<https://projects.fivethirtyeight.com/2022-election-forecast/senate/pennsylvania/>
<https://fivethirtyeight.com/features/where-fivethirtyeight-and-espn-2022-23-nba-forecasts-agree-and-disagree/>

Bet Screening:

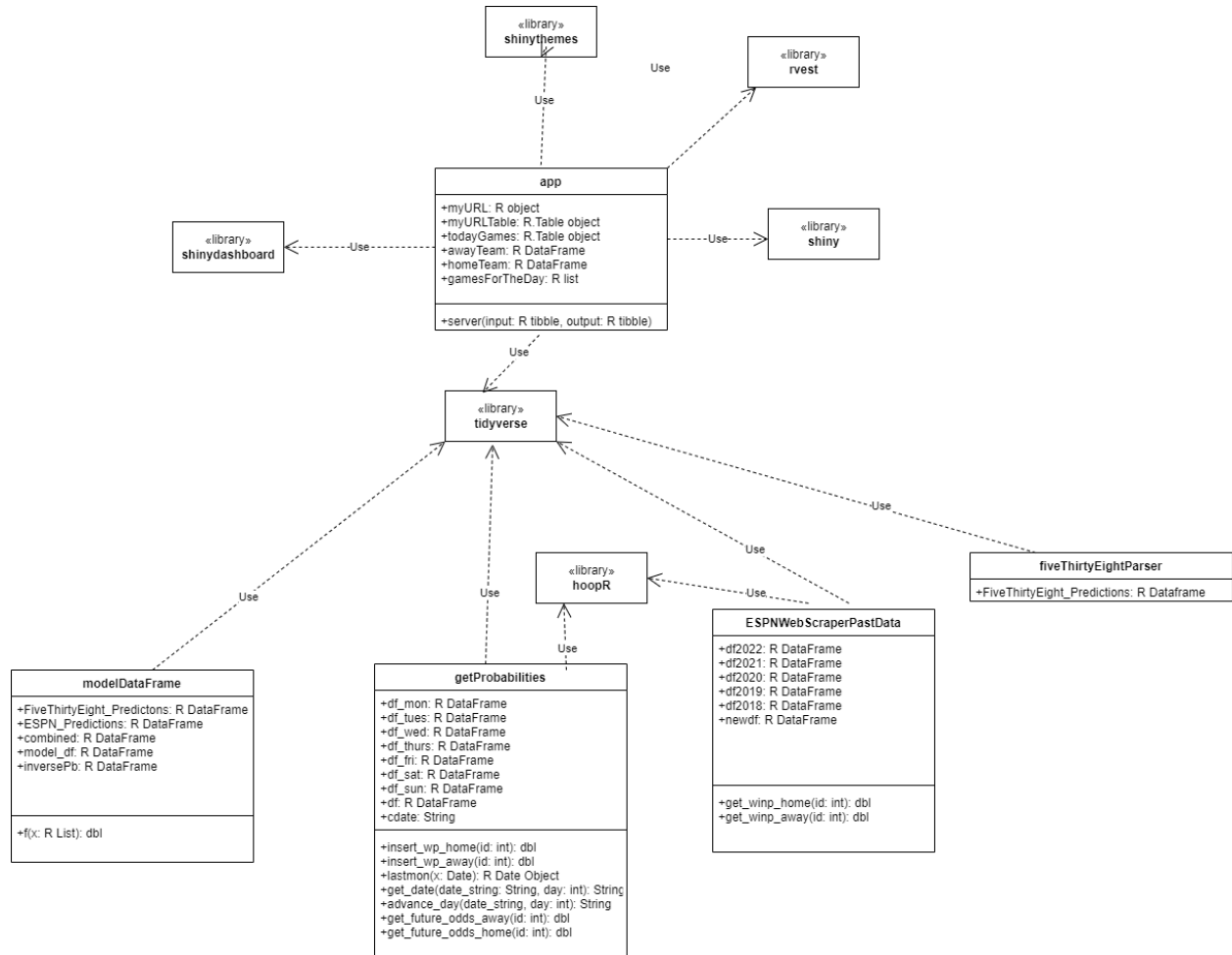
Users will have the ability to enter their bet amount and spread and receive a bet recommendation and how much money they are expected to make or lose. This would work similar to the calculator linked below. The difference is, users would be able to use our percentage from the aggregated predictions feature above.

<https://oddsjam.com/betting-calculators/expected-value>

Betting Education:

We will provide explanations on how betting works, different strategies, and the main theory of how to be profitable at sports betting. We will provide sited resources on the financial impact of sports betting and how to avoid compulsive gambling.

Class Diagram (Current Progress as of Week 4)



This is the UML class diagram for our current iteration of the BettR platform. Our **app.R** file currently depends on five libraries: **shiny**, **shinydashboard**, **shinythemes**, **tidyverse**, **rvest**. In the **app.R** file, there are three variables: **myURL**, **myURLTable**, **todayGames**. They are R objects and R.Table objects respectively. There is one function **server()**, which takes in two **R Tibble** parameters: **input**, **output**. The input variable retrieves the information in the table column, and passes the data to the output table. This **server()** function launches the web server that allows the user to interact with the web application.

The purpose of our **fiveThirtyEightParser.R** file is to parse and clean the probabilities from the 538 website. This file only depends on **tidyverse** to write the finalized data to a .csv file, so that it may be used for the front-end portion of the web application. The data is cleaned by managing the data retrieved from 538 to an R Dataframe named **FiveThirtyEight_Predictions**. **HoopR** is an R library containing many helper functions to retrieve data from the ESPN website. The files **ESPNWebScaperPastData.R** and **getProbabilities.R** depend on this library. The **ESPNWebScaperPastData.R** file uses **HoopR** to retrieve win probabilities from ESPN, focusing on the 2018 - 2022 seasons. This is accomplished by declaring 5 R DataFrames: **df2022**, **df2021**, **df2020**, **df2019**, and **df2018**, and merging their rows into a new R DataFrame **newdf**. The

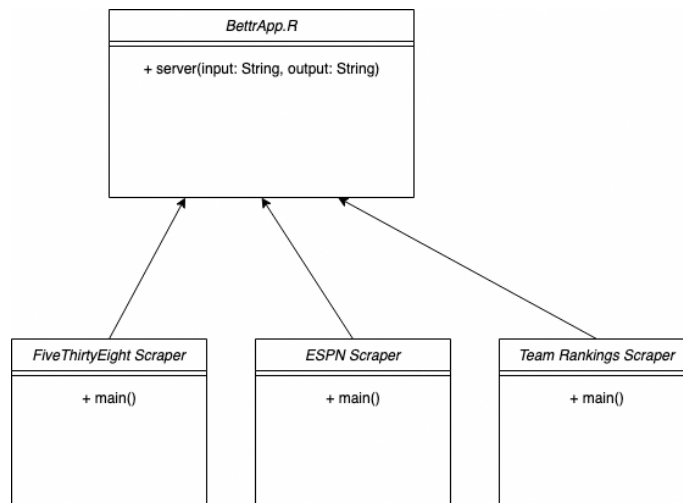
functions **get_winp_home()** and **get_winp_away()** take in an integer ID as input, and outputs the win probabilities as a double for the home and away teams respectively.

Much like the **ESPNWebScraperPastData.R**, **getProbabilities.R** contains similar functions and approaches. The purpose of this file is to retrieve win probabilities for basketball games in the current week. There are functions **get_future_odds_home()** and **get_future_odds_away()** take in an integer ID as input, and output the win probabilities as a double for the home and away teams respectively, using simple R web scraping. Those functions are utilized by **insert_wp_home()** and **insert_wp_away()**, which depending on if the NBA is in the past or future, uses the appropriate function to retrieve the odds. If it is in the past, it will use the built in **hoopR** function, if it is in the future, it will use the **get_future_odds_home()**, and **get_future_odds_away()** respectively. Since the week of NBA games starts on monday, the function **lastmon()** finds the date of the last monday, based on the current date. **lastmon()** returns a R Date object, which is unable to be used in **HoopR** functions. Therefore, the Date object that is returned is cleaned and parsed into a String of format **YYYYMMDD**. The functions **get_date()** and **advance_day()** use R's **as.Date()** implementation to get a new date from an existing date by adding an integer to it. There are currently 4 variables: **df_mon**, **df_tues**, **df_wed**, **df_thurs**, **df_fri**, **df_sat**, and **df_sun**, each corresponding to the games of that day during the week. By implementing **HoopR**'s **espn_nba_scoreboard()** function and the return value from **advance_day()**, the games for that day are inserted into the corresponding dataframe. As before, the dataframes are then merged into R DataFrame **df**, and the contents are output to a .csv file, ready to be used by the front-end.

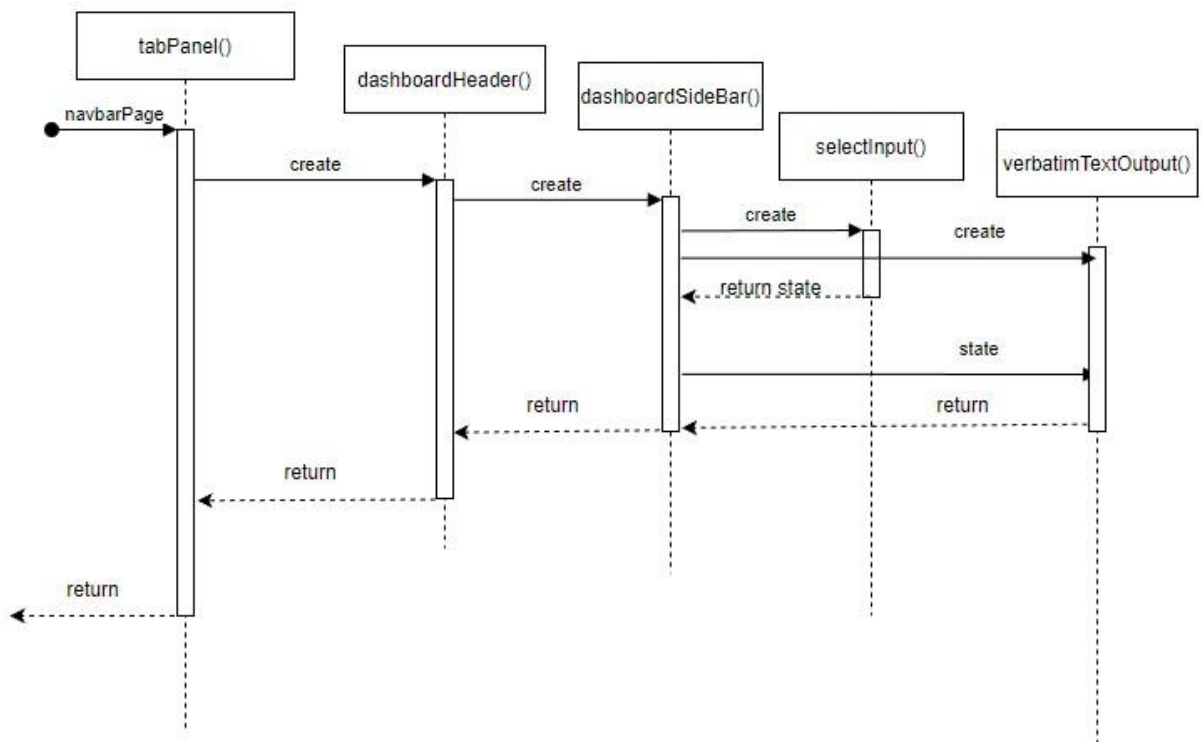
Another file in the design is **model_Dataframe.R**, which produces the dataframe with win probabilities that directly gets used to build the model for our application. It starts by cleaning the 538 and ESPN datasets to put them in the same format. It then appends them together and groups rows/instances according to NBA team pairs. This is done using a function called **f**, which taken in a row from the data frame as input and outputs the probability of the similar team pair that exists in the data frame (for example, if the inputted row is for the team pair 'BOS' and 'ATL', it outputs the win probability for the row with the team pair 'ATL' and 'BOS'). After taking the average of the win probabilities, the combined dataframe is outputted to a csv, which can be integrated into the user interface.

Theoretical Design (Plan by End of Project):

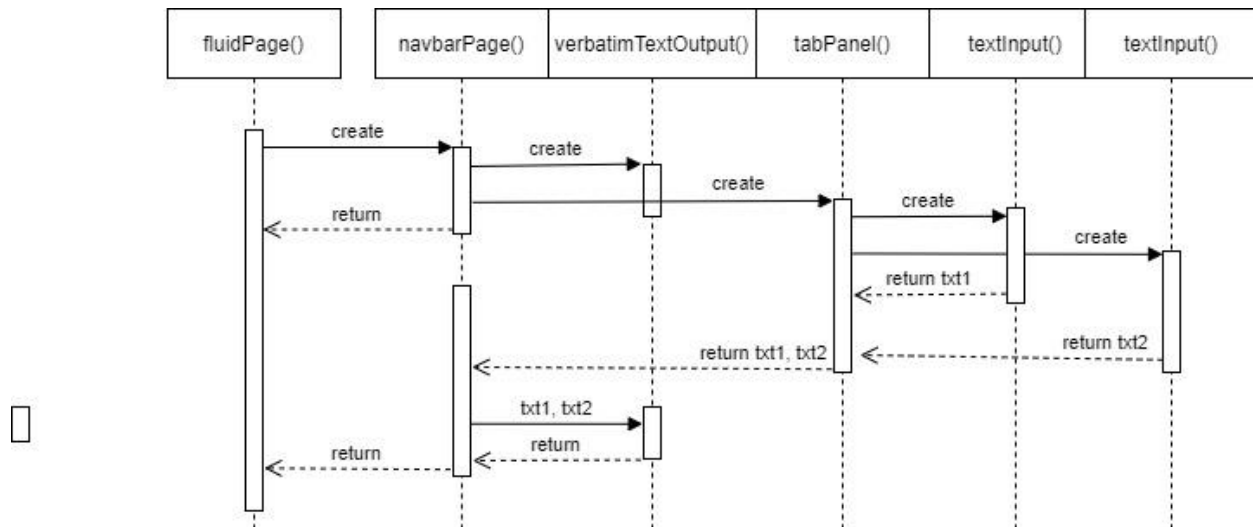
In the future, each website and data source will have its own web scraper written in Python. These will pass data to our R Server by writing to CSV files periodically. The writing to CSV files will be done in a main function in each of the scraper classes, as displayed. Our R server will then combine local HTML files and complete Data Science tasks on the data to present it to the user on the website.



Sequence Diagrams



In this sequence diagram, we have a user interaction with a drop-down menu with a list of options for the user to select from. Initially, as the app starts, the first object to be created is the `navbarPage`. In this object, we establish how many navigation bars exist by the number of created objects. As the `navbarPage` creates different pages, it also creates different `tabPanel` objects that will contain code and data for. In the specific user interaction, the user interacts with a `tabPanel` object, which then creates a `dashboardHeader()` object that will then create a `dashboardSideBar` object. The point of these is to create a dashboard object that will display information for the user and then partition a portion of that screen for the drop-down menu. The drop-down menu is created with a `selectInput` object that contains a state variable. This state variable will select from an available list of objects input at creation. The `selectInput` object will return the state variable, which holds an integer value of the list object. This state variable will be sent to the `verbatimTextOutput` object, which was created right after `selectInput` object. `verbatimTextOutput` will take in the state variable and then display the state of the drop-down menu and return back to `dashboardSideBar` object. After this, control will revert to the `navbarPage` where the app code will continue.



In this sequence diagram, we display a user interaction of how a user can input an odds line of both away and home teams. Initially, the app will execute by creating a fluidPage object, which will hold all the containers and object inside of it. Then we go through a sequence of creation of the relevant object for the user. First a navbarPage to navigate between navigation pages, then a verbatimTextOutput which is where the variables will be sent to be displayed. Then the tabPanel object, and finally our two textInput objects. Each one will accept a string input from the user. If they are null, then nothing is returned and the textInput Objects are always checking for values not null. Once a value has been inserted, tabPanel will return both values all the way to the navbarPage. navbarPage will then send the txt1 and txt2 variables to the verbatimTextOutput. Once verbatimTextOutput has received these variables, it will output these two variables then return back to fluidPage which will continue execution of the code.

Automated Test Results

No tests failed.

```
==> Testing shiny application using 'shinytest'

Loading required package: shiny
— Attaching packages — tidyverse 1.3.2 —
✓ ggplot2 3.4.0      ✓ purrr 0.3.5
✓ tibble 3.1.8       ✓ dplyr 1.0.10
✓ tidyr 1.2.1        ✓ stringr 1.4.1
✓ readr 2.1.3        ✓ forcats 0.5.2
— Conflicts — tidyverse_conflicts() —
✖ dplyr::filter() masks stats::filter()
✖ dplyr::lag()     masks stats::lag()

Attaching package: 'shinydashboard'

The following object is masked from 'package:graphics':

  box

Attaching package: 'rvest'

The following object is masked from 'package:readr':

  guess_encoding

Running mytest.R
==== Comparing mytest... No changes.

Test complete
```

ESPN web scrape test

```
==> Testing R file using 'testthat'

[ FAIL 0 | WARN 0 | SKIP 0 | PASS 0 ]
[ FAIL 1 | WARN 0 | SKIP 0 | PASS 0 ]

— Error ('test_ESPNwebScraPerPastData.R:1'): (code run outside of 'test_that()') —
Error in "rbindlist(l, use.names, fill, idcol)": Item 2 has 34 columns, inconsistent with item 1 which has 33 columns. To fill missing columns use fill=TRUE.
Backtrace:
 1. base::source("../ESPNwebScraPerPastData.R", chdir = TRUE)
    at test_ESPNwebScraPerPastData.R:1:0
 5. base::rbind(df2022, df2021, df2020, df2019, df2018)
 6. data.table::rbind(deparse.level, ...)
 7. data.table::rbindlist(l, use.names, fill, idcol)

[ FAIL 1 | WARN 0 | SKIP 0 | PASS 0 ]

Test complete
```

get Probabilities Test

```
==> Testing R file using 'testthat'
```

```
[ FAIL 0 | WARN 0 | SKIP 0 | PASS 0 ]  
[ FAIL 0 | WARN 1 | SKIP 0 | PASS 0 ]  
[ FAIL 1 | WARN 1 | SKIP 0 | PASS 0 ]
```

```
— warning ('test_getProbabilities.R:1'): (code run outside of `test_that()`) —  
cannot open file '../getProbabilities.R': No such file or directory  
Backtrace:
```

```
1. base::source("../getProbabilities.R", chdir = TRUE)  
   at test_getProbabilities.R:1:0  
2. base::file(filename, "r", encoding = encoding)
```

```
— Error ('test_getProbabilities.R:1'): (code run outside of `test_that()`) —  
Error in `file(filename, "r", encoding = encoding)`: cannot open the connection  
Backtrace:
```

```
1. base::source("../getProbabilities.R", chdir = TRUE)  
   at test_getProbabilities.R:1:0  
2. base::file(filename, "r", encoding = encoding)
```

```
[ FAIL 1 | WARN 1 | SKIP 0 | PASS 0 ]
```

```
Test complete
```

test Model Dataframe Tests

```
==> Testing R file using 'testthat'
```

```
[ FAIL 0 | WARN 0 | SKIP 0 | PASS 0 ]  
[ FAIL 1 | WARN 0 | SKIP 0 | PASS 0 ]
```

```
— Failure ('test_modelDataframe.R:12'): get similar win probability —————  
apply(modelDf, 1, f) not equal to `inverseList`.  
Types not compatible: character is not list
```

```
[ FAIL 1 | WARN 0 | SKIP 0 | PASS 0 ]
```

```
Test complete
```


Week 2 Progress

Sprint Goal:

During this sprint, we plan to create a mockup of our user interface design, create a very basic design of our user interface using RShiny, research into methods of scraping data using Python and R (such as Selenium), determine how/what would need to be accomplished in order to scrape data from ESPN and FiveThirtyEight, start working on building the web scraper for each dataset, and, at the very least, scrape the NBA data for teams that will play in the current week. Besides this, this Sprint will need to have multiple parts of the Project Report completed, including high level requirements, a UML class diagram, and project progress.

Backlog Features:

- Home page of User Interface created
 - Option to type in name
 - Option to choose dataset
- Upcoming Games pages of User Interface created
 - Displays NBA games for today, as scraped from ESPN
- Third Page in User Interface created for future use
- Mockup of User Interface drawn/designed
- README page updated with more detailed run instructions

Tasks in Sprint	Task Status at end of Sprint	Assigned To
Create front end framework	Completed	Kevin
Create tabs for front end interaction	Completed	Kevin
Create main page functionality for front end	Completed	Kevin
Create upcoming games page	Completed	Kevin
Create web scrape of ESPN for upcoming games	Completed	Kevin
Implement web scrape output to upcoming games for the day	Completed	Kevin
Update main page with gambling help resources	Completed	Kevin
Create user interaction functionality for selecting upcoming games	Completed	Kevin
Created host for web app for user access	Completed	Kevin

Created simple UML diagram detailing current progress in app.R	Completed	Anubhav
Create web scraper for ESPN data	In Progress	Anubhav
Shared previously written web scraping code and assisted members with setup and site URLs	Completed	Jared
Read research related to sports gambling to inform feature creation	Completed	Jared
Created Mock design for user interface	Completed	Jared
Created and typed up overall product architecture and plan to help keep team on track	Completed	Jared
Meet with all team members to discuss how to fairly and efficiently divide work for this week	Completed	Arun
Add in some of the parts for the Project Proposal Report, including conceptual design, required resources, persona Michael, and project progress. Also make various adjustments to all other parts of the document	Completed	Arun
Refamiliarize with R basics	Completed	Arun
Research into how to do web scraping in R and Python (ex: using Selenium)	Completed	Arun
Look into Jared's old web scraper to see how it could be implemented into our project	Completed	Arun
Write basics for the web scraper model for the ESPN and FiveThirtyEight Data in Python	Partially Completed	Arun

Week 3 Progress

Sprint Goal:

During this sprint, we plan to finish the basic design of the scrapers for the ESPN and FiveThirtyEight current and past data, improve upon the design of our product (adding relevant data tables to the webpage, improving look, etc.), and research/write out the resources/tips data that will be included on the front page of our application. The user interface will include an info/home page section with advice from the Web (information collected but not yet formatted), an expected profit/loss calculator, a page to hold data read in (basic functionality implemented at this point), and a model comparison and composite model page. Besides this, this Sprint will need to have multiple parts of the Project Report completed, including 2 Sequence Diagrams, 2 Updated Class Diagrams, and Week 3 Progress.

Backlog Features

- Home page of User Interface updated
 - Design Improved
 - Resources/Tips for Safe Betting/Gambling implemented
- Upcoming Games pages of User Interface improved
 - Displays mock NBA games for this week and allows user to select one
- Mockup of User Interface drawn/designed
- README page updated with more detailed run instructions based on raised Issues

Tasks in Sprint	Size	Task Status at end of Sprint	Assigned To
Create user interaction for selecting upcoming games	1	Completed	Kevin
Create front-end functionality for odds calculation of selected games	3	Completed	Kevin
Integrate web scraper built into user interface	3	Completed	Kevin
Create 2 sequence diagrams representing two different user interactions with our product	1	Completed	Kevin
Update main page with intended gambling warnings	3	Completed	Kevin
Update main page with basic education of odds and probabilities	3	Completed	Kevin
Pull past win probabilities using HoopR	3	Completed	Anubhav

Implement R and Python Functionality so that different libraries can be used together	3	Completed	Anubhav
Update Class Diagram (add details to description; functionality of server; what type of string is in the input, etc.)	1	Completed	Anubhav
Pull Odds from ESPN for current season	3	Completed	Anubhav, Arun
FiveThirtyEight - finish script to get data and output csv	3	Completed	Arun
Continue working on web scrapers (talk with team about bugs in Week 2 code and resolve)	3	Completed	Arun
Create release for this Week's Sprint	1	Completed	Arun
Make updates to project board (add missing items to backlog, create missing columns, adjust tasks no longer being considered; move tasks that have not yet been moved)	1	Completed	Arun
Add to project report (this document) with new sections and any adjustments	3	Completed	Arun
Meet with all team members to discuss how to fairly and efficiently divide work for this week	1	Completed	Arun
Research into how to do web scraping in R and Python (ex: using Selenium and HoopR)	1	Completed	Arun
Make significant updates to the webpage and running code portions of the README based on user comments and steps that are noticeably missing	1	Completed	Arun
Complete Final Designs/Mockups	3	Completed	Jared
Complete Resource Document	5	Completed	Jared
Begin theming and laying out frontend to match designs	8	Completed	Jared
TOTAL	54		

Velocity System:

1 - 30 minutes (XS)

3 - few hours - one day (S)

5 - 2-4 days (M)

8 - whole sprint (7 days) (L)

11 - more than one sprint (XL) - need to be broken in smaller tasks.

Estimated Velocity (At the Beginning of the Sprint)	54
Calculated Velocity (At the End of the Sprint)	54

Week 4 Progress:

Sprint Goal:

During this sprint, we plan to fully integrate the backend work with the frontend work; that is, we implement the past web-scraped data from ESPN and 538 to create a new model data frame, which gets used to make our model for the user interface. We also implement the data frame consisting of the current week's NBA game matchups and win probabilities. Besides this, this Sprint will need to implement automated testing, which is done in separate files for the user interface and backend. This Sprint also completes the automated testing and Week 4 Progress sections of the Project Report and updates the Sequence and Class Diagrams.

Backlog Features:

- Home page of User Interface updated
 - Design Improved
- Upcoming Games pages of User Interface with real data
 - Displays actual NBA games for this week and allows user to select one
 - Web Scrapers are now integrated into user interface
- Model data now integrated into user interface
- Backend script to develop model data frame is created
- Automated Tests developed for Shiny application and backend scripts (scrapers and models)
- Class Diagrams and Sequence Diagrams updated with new features

NOTE: For this Sprint, we were short one member due to a personal emergency, so he did not plan any items for this week's sprint. Thus, the tasks displayed reflect the work of 3 developers:

Tasks in Sprint	Size	Task Status at end of Sprint	Assigned To
Develop a Method to retrieve future game odds using HoopR	3	Completed	Anubhav
Create web scraper functionality to pull probabilities of future games	3	Completed	Anubhav
Compare performance and functionality between HoopR and web scraper methods of retrieving future odds data	3	Completed	Anubhav
Combine functions of retrieving past probability and future probability so that a data frame of all games can be retrieved with all the probabilities	3	Completed	Anubhav

Export games of the week data frame to CSV file to be integrated into the front end	1	Completed	Anubhav
Update UML Class Diagram with updates (model script, new scraper script, automated testing)	1	Completed	Anubhav
Fix ESPN scraper so that it only has desirable data (remove columns, remove bad rows, add full names of teams)	3	Completed	Arun
Fix FiveThirtyEight Scraper so that it only has desired data (remove columns, remove bad rows, remove older years, add full names of teams)	3	Completed	Arun
Make first version of model script to obtain model data frame	3	Completed	Arun
Make automated tests for scraper and model scripts (backend testing)	3	Completed	Arun
Add to project report with new sections and any adjustments	3	Completed	Arun
Create release for this week's sprint	1	Completed	Arun
Create control script to execute back end programs	3	Partially Completed (Started)	Arun
Add testing plan and coverage report to project report	1	Partially Completed	Kevin
Replace front end scraper with back end scraper product	5	Completed	Kevin
Implement app calculation of data from scrapers that give accurate predictions and expected value for user specified bets	5	Completed	Kevin
Create unit tests for app.R and scraper products to confirm proper starting	5	Completed	Kevin
Update UML Sequence Diagram based on provided feedback	1	Partially Completed	Kevin
Write main page HTML/CSS	3	Completed but not Integrated	Jared
Rewrite R layout to match designs	5	Completed but not Integrated	Jared
Write Resources HTML/CSS	3	Partially Completed	Jared

Total	61		
--------------	----	--	--

Estimated Velocity (At the Beginning of the Sprint)	61
Calculated Velocity (At the End of the Sprint)	45