|  |  |
| --- | --- |
| Game Planning shiny app  Data Analytics Report | Abstract  This report outlines the development and features of an Advanced Scouting Shiny App designed to analyze Tampa Bay Rays pitchers, including its interactive visualizations, statistical analysis capabilities, current limitations, and potential enhancements.  Greg Halperin and Javaris Hall |

Contents

[Introduction 2](#_Toc184301225)

[Shiny Apps 2](#_Toc184301226)

[Data 3](#_Toc184301227)

[Sources/Packages 3](#_Toc184301228)

[Collection 4](#_Toc184301229)

[Filtering 4](#_Toc184301230)

[Shiny App Features 4](#_Toc184301231)

[Inputs 4](#_Toc184301232)

[Outputs 6](#_Toc184301233)

[Error Handling and Performance 10](#_Toc184301234)

[Reactive Framework 10](#_Toc184301235)

[Limitations/Future Considerations 10](#_Toc184301236)

[Bibliography 12](#_Toc184301237)

# Introduction

This Advance Scouting Shiny App was developed to improve offensive game planning for an upcoming series against the Tampa Bay Rays.

It includes customizable filters, pitch-by-pitch data analysis, and interactive visuals including heatmaps, movement charts, and usage patterns. This project aims to give our team a competitive advantage through data-driven analysis, allowing coaches and players to understand opposing pitchers’ patterns, weaknesses, and tendencies, allowing us to develop targeted offensive strategies.

This document will explore how the app was developed, its many features, and some limitations and future considerations of the app, along with some recommendations for how to attack a couple of the starting pitchers for our upcoming series.

# Shiny Apps

According to *Teach Data Science*, a Shiny app is “a powerful and flexible R package that makes it easy to build interactive web applications and dynamic dashboards straight from R. These apps can be hosted on a standalone webpage or embedded in R Markdown documents. Not only does Shiny allow you to build these web apps from R, but it enables their construction using only R code.”[[1]](#footnote-1) Shiny Apps are a great way to visualize information in an easy-to-understand interface without requiring additional web development languages.

A Shiny App was chosen for this project for this exact reason. It allows a platform for our players and coaches to interact with the data in a digestible format and reach it from anywhere. Once published, all that is needed is the URL of the app, and users can access it from any device at any time they want.

# Data

## Sources/Packages

* Required Libraries
  + library(devtools)
  + library(baseballr)
    - Install via: devtools::install\_github(repo = "BillPetti/baseballr")
    - This version includes an update to correctly load data from Baseball Savant.[[2]](#footnote-2)
    - May require uninstalling previous version of baseballr to reinstall using the above code.
  + library(dplyr)
  + library(paletteer)
  + library(ggplot2)
  + library(ggpubr)
  + library(plotly)
  + library(shiny)
  + library(shinythemes)
  + library(DT)

## Collection

* Player ID information was collected from Chadwick Bureau Register Player Lookup using the Chadwick\_player\_lu() function in BaseballR.
* Pitch-by-pitch data was collected for all Rays pitchers during the 2024 season from Baseball Savant using the statcast\_search() function in BaseballR.

## Filtering

* The data was focused solely on the Rays Pitchers that pitched in September of the 2024 season.
* Approximately 70 columns in the dataset was filtered out, including: fielding data, umpire data, score information, and any columns with only null values.

# Shiny App Features

## Inputs

**Team Selection (Select Input)**

* Displays available Teams.
* Controls the filtering of available pitchers.

**Pitcher Selection (Select Input)**

* Dynamically updates based on selected team.
* Filters to show only pitchers from the selected team.
* Pitcher choice will affect available Pitch Types.

**Date (Date Range Input)**

* Select a custom Date Range for analysis.
* Default range: min/max dates from the dataset.
* Includes error handling for invalid date selections.

**Batter Side (Radio Buttons)**

* Options: "Both", "Right Handed", "Left Handed".
* Used in filtering pitch data for splits analysis.

**Visual (Checkbox)**

* Controls display of different analytical visuals.
* Allows multiple visualization selections simultaneously.
* Options include Pitch Metrics, Usage Charts, Heatmaps, Movement Charts, and Location Charts.

**Pitch Type (Checkbox)**

* Dynamically updates based on selected pitcher's repertoire.
* Required for certain visualizations (heatmaps, movement plots, location plots)
* Includes error handling for missing selections.

## Outputs

**Pitch Metrics**

* Implementation: Uses the DT package in R for enhanced table functionality**.**
* The Data Table includes Pitch Type, Pitch Usage, Pitch Movement Metrics, and Pitch Result Metrics.
* Conditional Formatting: The below metrics are conditionally formatted so users can quickly identify positive metrics from negative ones:
  + Red is positive for hitters, and Blue is positive for pitchers.
  + Custom Breakpoints for the below Metrics:
    - Exit Velocity
      * Less than 80 mph = Blue
      * Greater than 90 mph = Red
    - OPS
      * Less than .600 = Blue
      * Greater than .900 = Red
    - wOBA
      * Less than .250 = Blue
      * Greater than .370 = Red
    - Hard Hit %
      * Less than 25% = Blue
      * Greater than 40% = Red
    - Barrel %
      * Less than 4% = Blue
      * Greater than 10% = Red
    - Whiff %
      * Greater than 30 = Blue
      * Less than 20 = Red
    - The breakpoints were determined based on Baseball Savant league averages.
* Data Validation:
  + Minimum 5 pitches required for metric calculations.
  + Null Value handling for pitches with fewer than 5 occurrences.
* Reactive Dependencies:
  + Updates with pitcher, date range, and batter side selections.
  + Not dependent on pitch type selection.

**Pitch Usage Chart**

* Implementation: Created using ggplot2 with custom heat map styling.
* Visual Features:
  + Heat map visual of pitch usage by count.
  + Color gradient from light blue (lower percentage) to light red (higher percentage).
  + Percentage label in each cell.
  + Ordered by pitch type and count situation (i.e. 0-0 or 1-2)
* Reactive Dependencies:
  + Updates with pitcher, date range, and batter side selections.
  + Not dependent on pitch type selection.

**Location Heatmaps**

* Implementation: Multiple ggplot2 plots arranged using ggarrange.
  + Color palette created using paletteer.
* Strike Zone overlay with proper dimensions.
* Shown from the pitcher's perspective.
* Density: Used stat\_density2d\_filled to make the heatmap.
* Error Handling:
  + A minimum of 5 pitches is required per pitch type.
  + If one of the selected pitches does not meet the criteria, it will not appear on the grid.
* Layout:
  + Dynamic grid arrangement based on number of selected pitches.
  + Automatic scaling for multiple plots.

**Movement Chart**

* Implementation: Interactive plot using Plotly.
* Data Processing:
  + Calculates average movement metrics by pitch type.
  + Compares league averages for same-handed pitchers.
* Visual Elements:
  + Individual pitch scatter plot.
  + Pitcher averages (diamond markers)
  + League averages (star markers)
* Interactive Features:
  + Hover Text with detailed pitch metrics.
    - Displays Pitch Type, Velocity, Vertical Movement, Horizontal Movement, Spin Rate, and Result.
  + Customized legend grouping.
    - Allows the user to select only the pitcher averages, league averages, or individual pitches.
  + Zoom and pan capabilities from Plotly.

**Location Chart**

* Implementation: Interactive strike zone visualization using Plotly.
* Features:
  + Strike zone overlay with proper dimensions.
  + Home plate rendering.
  + Color-coded pitch locations.
  + Shown from the pitcher’s perspective.
* Interactive Elements:
  + Hover information.
    - Displays Pitch Type, Velocity, Vertical Movement, Horizontal Movement, Spin Rate, and Result.
  + Pitch Type Filtering.
  + Zoom and pan functionality from Plotly.

## Error Handling and Performance

* Safe Rendering to provide error messages in case of Null values.
* Data Validation
  + Checks for sufficient data points.
  + Handles missing values.
  + Provides user feedback for invalid states.

## Reactive Framework

* All outputs utilize Shiny’s reactive programming model.
* Efficient updates based on input changes.

# Limitations/Future Considerations

While this Shiny App provides valuable insights for game planning against the Tampa Bay Rays pitchers, there are several limitations and areas for improvement.

Currently, the app’s scope is limited to the Rays pitchers from the 2024 season, limiting the use cases for this application. To enhance the app’s utility a few improvements should be implemented. The first would be to increase the dataset to encompass all teams and pitchers that pitched in the MLB during the 2024 season. This would allow the team to use this app for every upcoming series. Adding a larger date range would also help identify trends across seasons and increase the significance of the data sample. In advance scouting, it is important to focus on recent trends; however, increasing the available date range allows for larger projects focusing on player development to be completed.

Another way to increase the value of the application would be to implement a pitch sequencing analysis and tunneling visualizations to offer deeper insights into pitcher tendencies. Adding to the app’s diversity would also be important. The addition of a “Hitters” tab focusing on other teams’ hitters and how to attack them would be integral in our pitching game plan. Visuals similar to the data metrics table and the heatmaps would be included in this analysis along with spray charts, and a breakdown of their baserunning and bunting capabilities.

The next steps should focus on technical improvements and organizational intregration. Introducing automated data updates, optimizing app performance for larger data sets, and developing API integration for real-time updates would improve the functionality of the app for our players and staff.

From an organizational perspective, this app could become a more comprehensive tool by incorporating some of the future considerations mentioned above as well as introducing predictive analysis and video integration. These would help create a more cohesive organization and enhance the organization’s competitive advantage through advanced analytics.

Overall, this project is a good start, but the key to maximizing its potential lies in addressing these limitations, developing new visuals and tabs, and maintaining its user-friendly interface for coaches, players, and staff.

# Bibliography

*Acquiring and Analyzing Baseball Data*. (n.d.). Billpetti.github.io. <https://billpetti.github.io/baseballr/>

Baseball Reference. (2024). *Tampa Bay Rays*. Baseball-Reference.com. <https://www.baseball-reference.com/teams/TBR/2024.shtml>

Baseball Savant. (2019). *Baseball Savant: Trending MLB Players, Statcast and Visualizations*. Baseballsavant.com. <https://baseballsavant.mlb.com/>

BillPetti. (2024, November 14). *Can’t assign 94 names to a 113 column data.table · Issue #354 · BillPetti/baseballr*. GitHub. <https://github.com/BillPetti/baseballr/issues/354>

chadwickbureau. (2016). *GitHub - chadwickbureau/register: Chadwick persons register*. GitHub. <https://github.com/chadwickbureau/register>

*Shiny - Articles*. (n.d.). Shiny.posit.co. <https://shiny.posit.co/r/articles/>

1. [An Introduction to R Shiny · Teach Data Science](https://teachdatascience.com/shiny1/) [↑](#footnote-ref-1)
2. <https://github.com/BillPetti/baseballr/issues/354> [↑](#footnote-ref-2)