# **UNCW Baseball Technical Report**

**By: Mavin James, Will Toman, and Sean Burke**

## **I. Background of Baseball**

Baseball is a sport with deep roots in American culture. Although the exact origin of baseball is often debated, the game made its way to America in the middle of the 19th century, where it gradually evolved into what it is today. The game consists of two teams of nine players on a field holding four bases in a diamond pattern. The main objective is for the batter to hit the baseball, thrown from the pitcher on a mound, to a spot on the field where no defender can retrieve the ball. Once a batter touches all four bases, a run will be recorded. On the other hand, if the pitcher and the defending team can accumulate three outs, the inning will conclude, and the defending team will have an opportunity to bat. The team with the most runs after nine innings will be declared the winner.

The role of pitching in baseball is one of the most scrutinized positions in all of sports. While the pitcher is not fully responsible for the outcome of an at-bat, they hold most of the blame when things go wrong. There is no clock counting down that can save a pitcher in a bad situation, the man on the mound must record three outs or be replaced. Emphasizing pitcher development is a key factor in the University of North Carolina at Wilmington (UNCW) Baseball team’s success over the last few seasons.

The most important attribute of a well-equipped pitcher is control. For pitchers to succeed they must be able to throw strikes. A strike occurs when the ball is thrown through an imaginary box called the strike zone (The size of the home plate as the width and from the knees of the batter to the belt as the height). Alternatively, the pitcher can generate a swing and a miss to record a strike. If the pitcher does not record a strike, a ball is recorded. The batter is awarded first base if the pitcher throws four balls before any other outcome. Pitchers who can throw multiple different kinds of pitches are more successful. A pitcher with good movement on his breaking balls can fool the batter into swinging at impossible pitches. Quality change-ups play off the velocity of a fastball and make it hard to time when the ball crosses the batter’s plane. It is not only building pitchers who can throw well but also building a strong mentality that helps a pitcher get out of the toughest situations with men on base or when the game is on the line. The UNCW baseball team was gracious enough to allow us to work with their comprehensive dataset which included around 54,000 rows and 167 columns, built by Trackman.

## **II. Trackman Technology**

Trackman was created in 2003 initially to track the flight of a golf ball. After years of providing game-changing data to the game of golf, Trackman has now expanded to other sports such as baseball. The game of baseball has long been considered the sport to revolutionize sports analytics, and Trackman continues to make data more available. Trackman’s value is showcased by its presence in all 30 Major League Baseball (MLB) stadiums. College baseball teams are starting to realize the value it can bring to them and are increasingly adopting the technology as well.

The system that UNCW has sits atop the press box, a place high enough to track the entirety of the ball's flight. From a data analyst’s perspective, Trackman data is easy to work with. A CSV file is created that can then be loaded into any preferred computer language to begin the analysis. In the file itself, there is a unique row for every pitch thrown throughout a game. In the UNCW file explored in this project, there were around 54.000 rows, each row containing many columns, ranging from basic to intricate.

In this project, we focused on a few main columns including ‘TaggedPitchType’, ‘PitchCall’, ‘PlateLocSide’, and ‘PlateLocHeight’, which are all related to the type, location, and outcome of a pitch. The more intricate columns like ‘SpinRate’ hold value, but the goal was to avoid overwhelming the audience with baseball jargon. Instead, the focus was on presenting the data in an accessible and meaningful way to highlight the value of Trackman technology.

## **III. Pitcher Situations: Leverage and Handedness**

For pitchers, no pitch is ever the same; there are countless situations a pitcher can find himself in. Some of the best pitchers in the game of baseball can adapt to whatever situation is thrown at them, even if they are not particularly responsible for it. Leverage defines the importance of a particular pitch to the outcome of a match. Taking the score and the men on base (from 0-3 men) into account, our team built 3 distinct category filters of leverage: Low, Medium, and High.

When a pitcher is thrust into a low leverage situation, the game might be out of hand. Usually, a team dominates the match, making the importance of the following pitches obsolete. Examples of a low-leverage situation might be when a team is up by more than six runs at any particular part of the game. Medium leverage situations make up a majority of the pitches. When a pitch is not critical enough to make or break a game, but is just important enough, as a collective, to define how the rest of the game will be decided. Usually, medium leverage pitches are early in the game, before the fifth inning, or when the score difference is between three and five runs in the later stages (fifth to ninth inning). When a pitcher faces the hardest challenges, late in the game, it is almost certainly a high-leverage situation. From the fifth inning onward, if the score difference is less than three runs, every pitch will be defined as a high-leverage pitch. Evaluating pitchers during the toughest times can give an insight into their mental toughness. We broke down the leverage scenarios as follows:

**UNCW Pitching:**

* If UNCW is leading, the score decreases by the number of men on base, reflecting the increased risk of allowing runs, thus adjusting the score as: ‘adjusted score’.
* If UNCW is trailing, the adjusted score remains unchanged.

The leverage metric categorizes the criticality of game situations based on the adjusted score and the inning. Leverage is classified as follows:

* **High Leverage:**
  + Occurs when the absolute value of the adjusted score is ≤ 3 and the inning is ≥ 5.
* **Medium Leverage:**
  + Applied to situations not covered by high or low leverage criteria.
* **Low Leverage:**
  + Identified when the absolute value of the adjusted score exceeds 5.

To accomplish this, we had to curate a couple of features into the dataset. Originally, the dataset did not include key metrics like Cumulative runs scored, actual score difference, men on base, adjusted leverage score, and a leverage category. The creation of the score used simple arithmetic to find when a run occurred and accumulate it, for each team respectively, until the game is over. After the score was defined, we needed to find how many men were on base during each pitch. There are nine scenarios where a batter can get on base, so finding these scenarios and accumulating them throughout an inning can give us the total number of men on base for the whole inning. Unfortunately, some scenarios eliminate runners on base as well. If a runner scores they are no longer on base, thus when a run scored we subtract it from the total men on base. Another way a base runner can be eliminated is by recording an out on the base paths. Defined as ‘caught stealing’ or ‘picked off’. After these scenarios are considered, the pitch-specific runners on base are recorded. Once the inning is over the men on base tally resets to zero. Taking the score and the men on base during a specific pitch allows us to define ‘adjusted score’ as defined before.

Another situation a pitcher can face is the handiness of the batter. Batters can either hit left-handed, or right-handed. Depending on what handiness a batter utilizes, the batter will either be on the left side or the right side of the plate. Typically, it is well regarded as an advantage for the pitcher if the batter and the pitcher have the same handiness (Eg. Left vs. left) due to the tough visual angle the batter has as the ball is released. However, a batter with the opposite handiness (Eg. Left vs. right) will have an easier time seeing the ball released from the pitcher's hand. Utilizing the ‘BatterSide’ and the ‘PitcherSide’ columns we can observe whether there is an advantage for the pitcher or the batter. Pitchers who succeed against both types of batters are well-equipped to be players who thrive as starting pitchers. Meanwhile, a pitcher who may specialize against only one particular batter's handiness may be utilized as a situational substitute pitcher. Finding what situations a pitcher thrives on is key in game management by the coaching staff.

## **IV. Visualizations**

In the game of baseball, visualizations are extremely popular and play a crucial role in analyzing the extensive amount of data produced every game. When looking at the Trackman dataset we used in this project many possible visualizations came to mind. Ultimately, we chose to focus on three that we felt would have the most impact on our audience. First, a heatmap function that displays the entire pitch arsenal of a pitcher, allowing the user filter capabilities based on what they are looking for. Second, a scatterplot that highlights pitch location and performance across various leverage levels, emphasizing how pitchers perform in pressure situations. Lastly, a tree of pie charts highlighting the pitch type most likely to be used given certain situations in an at-bat. When combined the three provide a foundational look at the capabilities when using a Trackman dataset.

**Heatmaps Displaying Pitcher’s Pitch Arsenal**

The first visualization created utilizes a function to allow for reusability and adaptability. The function creates a grid of plots grouped by the type of pitch thrown. The visualization allows for interpretation and comparison between each pitch.

The function’s signature is as follows:

def plot\_pitcher\_data\_heat(pitchers\_df, pitcher\_firstname, pitcher\_lastname, pitch\_call = None)

The signature of the function plot\_pitcher\_data\_heat defines its purpose and outlines the parameters required for its execution. The first parameter, pitchers\_df, is the pandas data frame produced by the CSV file from Trackman. This function can be used not only with the CSV file from UNCW but with any CSV file generated by Trackman. The second and third parameters are pretty self-explanatory, they are the first and last names of the pitcher who the user is interested in exploring. The pitch\_call parameter is optional and provides the capability of sorting for a specific outcome. For example, if pitch\_call = “StrikeSwinging” is entered the only pitches used to create the heatmaps will be those that produced a swing and miss. When a pitch\_call is not provided the heatmaps show the location for every outcome. When the user enters a pitcher’s name the function accepts any case to allow for easier use, this is accomplished in these three lines:

pitcher\_name = (pitcher\_lastname + ', ' + pitcher\_firstname).lower()

pitchers\_df.loc[:, 'Pitcher'] = pitchers\_df['Pitcher'].str.lower()

pitcher\_data = pitchers\_df[pitchers\_df['Pitcher'] == pitcher\_name].copy()

In the lines above the input of the pitcher’s first and last names are read in, combined, and converted to lowercase to create a variable named pitcher\_name. Then the next line uses a slice to get every row in the dataframe and convert the ‘pitcher’ column to lowercase. Once these two lines are complete the last line simplifies the data frame to only the rows where the pitcher’s name in the input matches the pitcher’s name in the dataframe. This simplified data frame is assigned to the variable pitcher\_data and a copy is created to ensure the original data frame is not altered. Once the refined data frame is created for the specified pitcher, a conditional statement is used to apply the filter of pitch outcome, and the pitcher\_data data frame is refined even more.

if pitch\_call:

pitcher\_data = pitcher\_data[pitcher\_data['PitchCall'] == pitch\_call]

After these few steps some data processing is done to enhance the clarity of the final output. The processing includes sorting by pitch type, getting a count of how many pitches of that type are thrown, and making the names of the pitches uppercase. All of these attributes make the final product easier to process. The next chunk of code highlighted below really displays the flexibility of the function.

g = sns.FacetGrid(pitcher\_data, col="TaggedPitchType", col\_wrap=len(pitch\_types), height=5, aspect=1, col\_order = sorted\_pitch\_types)

def conditional\_plot(data, \*\*kwargs):

pitch\_type = data["TaggedPitchType"].iloc[0] #Get the pitch type being plotted

if len(data) < 5: #If fewer than 5 pitches, use scatterplot

sns.scatterplot(

x="PlateLocSide",

y="PlateLocHeight",

data=data,

s=75, #Size of scatter points

\*\*kwargs

)

else: #Heatmap

sns.kdeplot(

x="PlateLocSide",

y="PlateLocHeight",

data=data,

fill=True,

cmap="coolwarm", #Color gradient for heat map

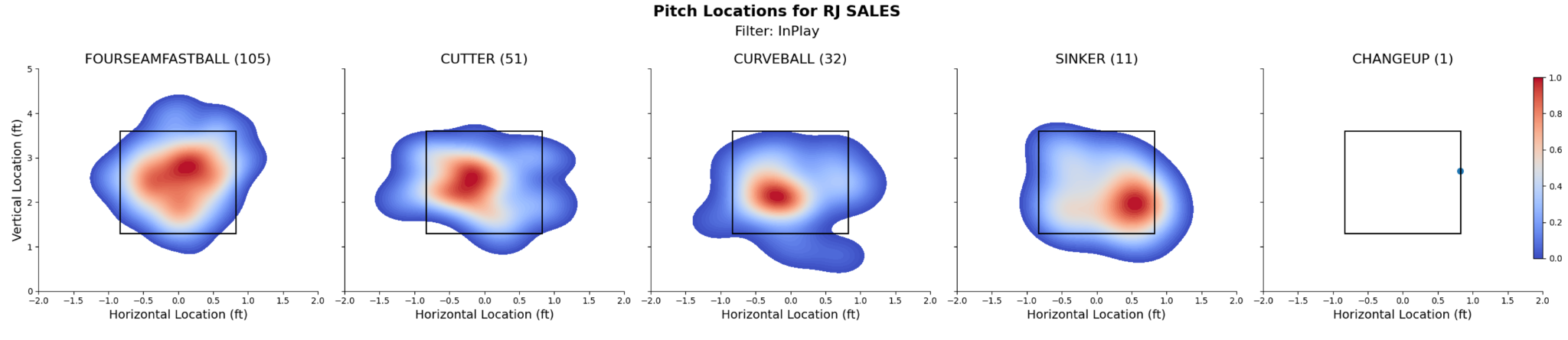
levels=50,

\*\*kwargs

)

g.map\_dataframe(conditional\_plot)

In this code chunk a function is created within the larger function to make the subplots in the facetgrid more meaningful. When there are less than five data points for a certain pitch type a heatmap is not created, and even if one was created there would not be much meaning to it. The conditional creates a scatterplot of those few pitches instead of a heatmap to give some context.The facetgrid generates a subplot for each 'TaggedPitchType', arranged in descending order from left to right based on the frequency of pitches thrown.. Some more code was added to contextualize the display and below is a sample output:



The output above comes from the following:



The pitcher specified here is RJ Sales and the pitches are sorted to display only those that are in play, meaning the pitches that are hit in the field of play. The visualization shows that RJ Sales threw five pitches (four-seam fastball, cutter, curveball, sinker, and changeup). The legend on the right tells the user that red is for high density areas and blue is for the lesser areas. The x axis is the location of the pitch horizontally relative to home plate and the y axis is the location of the pitch vertically relative to home plate. A ball hit in play is not always necessarily a bad thing as long as the hit is to a fielder. The ultimate goal would be to throw a pitch for a strike every time but that is not realistic, so sometimes pitching to contact is valuable. This graphic shows that RJ Sales threw 105 four-seam fastballs that were put in play. The heatmap shows that these pitches were thrown in the middle of the zone. Although hard to control, if Sales could spot the pitch in the corners of the zone instead he would miss more bats for sure. The interpretations of the outputs of this function are endless and emphasize areas where a pitcher is succeeding and struggling, which can lead to player development efforts.

**Scatter Plots Displaying Different Leverages**

From the very first discussion of this project we were interested in incorporating leverage into a visualization. Creating one with some meaning was a challenge. We ended up creating a function that has a similar appearance to the one above. The signature of the function is highlighted below:

def plot\_pitcher\_data\_scatter\_lev(pitchers\_df, pitcher\_firstname, pitcher\_lastname):

The function takes in the same first 3 parameters as the function mentioned in the section above, sorting by the outcome is not an option in this case. The way the data frame is narrowed down to a certain pitcher is also the same as the function in the previous section. The code below is where the function begins to differ:

leverage\_order = ["Low", "Medium", "High"]

pitcher\_data['PitchCall'] = pitcher\_data['PitchCall'].replace({'StrikeCalled': 'Strike', 'StrikeSwinging': 'Strike'})

pitcher\_data['leverage'] = pd.Categorical(pitcher\_data['leverage'], categories=leverage\_order, ordered=True)

Starting from the first line an order for the leverage is created and defined as ‘leverage\_order’, to ensure the subplots are displayed in that order. Next the ‘PitchCall’ column is altered by placing both ‘StrikeCalled’ and ‘StrikeSwinging' into the same category. Then lastly the ‘leverage’ column is made categorical and the ‘leverage\_order’ list is used to enforce a strict order. The next code chunk is what brings some true meaning to the final product.

total\_pitches = pitcher\_data.groupby('leverage').size()

strike\_pct = (pitcher\_data[pitcher\_data['PitchCall'] == 'Strike'].groupby('leverage').size() / total\_pitches \* 100).reindex(leverage\_order, fill\_value=0)

ball\_pct = (pitcher\_data[pitcher\_data['PitchCall'] == 'BallCalled'].groupby('leverage').size() / total\_pitches \* 100).reindex(leverage\_order, fill\_value=0)

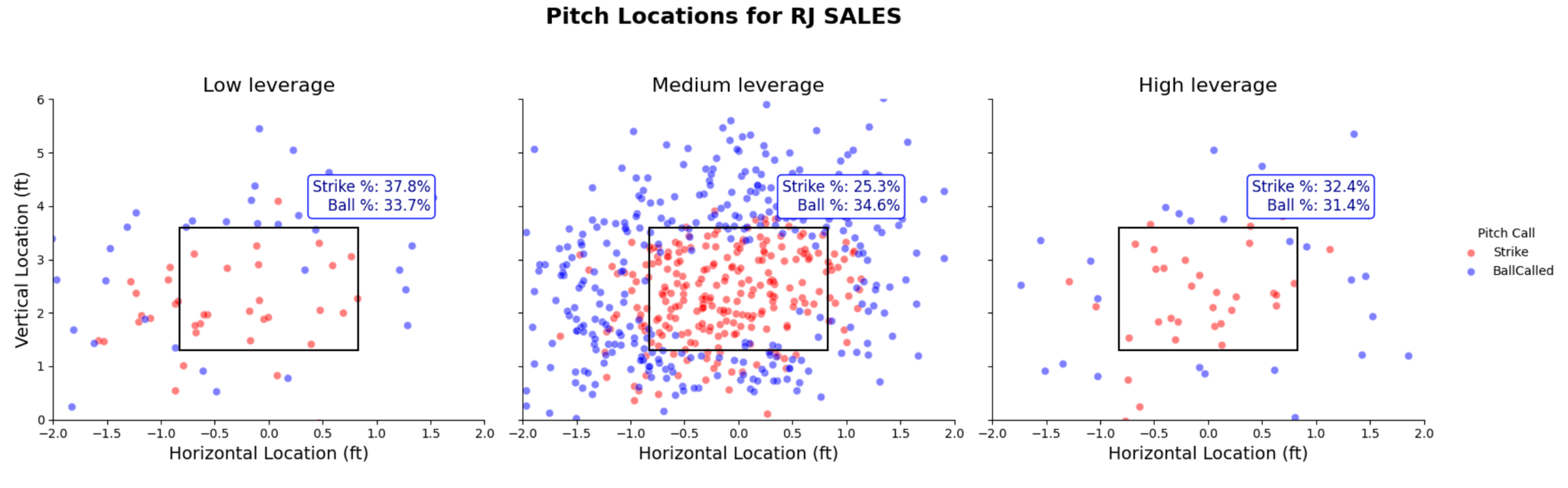
relevant\_data = pitcher\_data[pitcher\_data['PitchCall'].isin(['Strike', 'BallCalled'])]

These lines of code provide a way to compare control of pitches within the different leverage categories. Control is a big part of pitching, the goal is to have a ratio that includes a few more strikes than balls. Throwing all strikes may initially seem like the goal, but that is not the case, if all pitches are strikes the pitcher becomes more predictable. These percentages are then displayed on a similar facetgrid as used in the previous function and highlighted below:

​​g = sns.FacetGrid(relevant\_data, col="leverage", col\_order=leverage\_order, hue="PitchCall", palette=palette, height=5, aspect=1)

g.map\_dataframe(sns.scatterplot, x="PlateLocSide", y="PlateLocHeight", edgecolor="w", s=40, alpha = .5)

The biggest difference in this visualization is the use of a scatterplot as opposed to a heatmap. We felt the scatter plot illustrated balls and strikes better than the heatmap. Also this grid has a fixed number of three subplots, one for each leverage category. Below is a sample output:



The output comes from the following:



The biggest thing to focus on here is the ratio of balls to strikes in each leverage category. The most concerning ratio occurs in the medium leverage category, which is where most of the pitches are thrown. The ball percentage is 34.6%, which is considerably higher than that of the strike percentage. This shows that RJ Sales could use some work on control, which is something that coaches can create a development plan to work on. In the other two leverage categories not too many pitches are thrown but the ratios are more of what you would be looking for.

**Pitch Variety Chart**

The third visualization was made to show how the pitchers change what style of pitch they are throwing depending on the count. As mentioned in the presentation, the count is essentially a score for each matchup between batter and pitcher. We decided to use pie charts for this visualization because it was the easiest way to show how the percentages of each pitch change as the count changes. Because there are 12 possible counts, we knew we would need 12 separate pie charts. We decided to organize each pie chart in a diamond-ish shape field. Its vertical position within the field was determined by what pitch number(1-5) would result in those counts. The horizontal position of each chart in the field was based on whether the count went in favor of the pitcher or hitter.

Now that our design was decided it was time to code. Firstly we wanted to focus on just the top five pitchers at UNCW so we had to filter the raw data frame for that and reduce the columns for more efficiency.

#Only Include Top Five Pitchers

top\_five=df['Pitcher'].value\_counts()[:5]

df=df[df['Pitcher'].isin(top\_five.index)]

#Filter Only needed columns:

columns\_to\_keep = ['Pitcher', 'BatterSide', 'PitcherThrows', 'Inning', 'Top/Bottom', 'Outs', 'Balls', 'Strikes', 'TaggedPitchType', 'PitchCall', 'PlayResult']

df = df[columns\_to\_keep]

df=df.reset\_index(drop=True)

Next we used for loop iteration to create a new column for each possible count combination and assign a one to it when that count occurs.

for balls in range(4): # 0 to 3 balls

for strikes in range(3): # 0 to 2 strikes

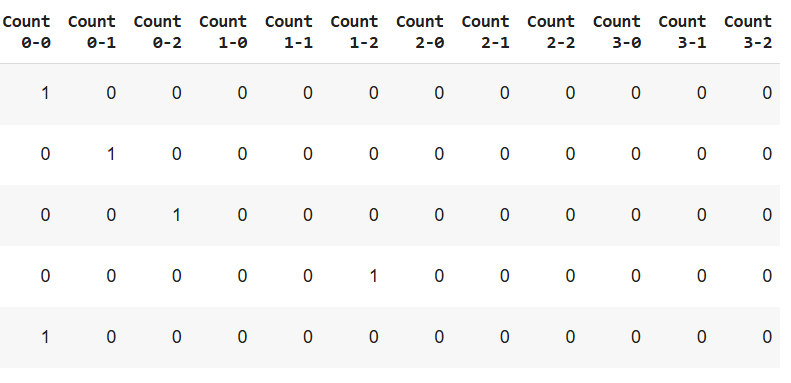
col\_name = f'Count {balls}-{strikes}' #create column name

df[col\_name] = 0 # init

df.loc[(df['Balls'] == balls) & (df['Strikes'] == strikes), col\_name] = 1 # Assign 1 when to corresponding column when that count is present in the Balls STrikes columns.

df.head()

This resulted in the creation of the following columns. You can see how the one is assigned if that pitch occurred at that count.



Now we had to find out how many times each style of pitch occurs at each count.

# Get unique pitch types

pitch\_types = df['TaggedPitchType'].unique()

# Count how many times each count appears in the dataframe.

count\_sums = df.loc[:, df.columns.str.match(r'Count')].sum().to\_frame()

# Create a column name for each type of pitch.

for pitch\_type in pitch\_types:

column\_name = pitch\_type

count\_sums[column\_name] = 0 #initalize

# go through each count possible

for balls in range(4):

for strikes in range(3):

count\_name = f'Count {balls}-{strikes}'

# Filter for specific pitch type and count

filtered\_df = df[(df['TaggedPitchType'] == pitch\_type) & (df[count\_name] == 1)]

count\_sums.loc[count\_name, column\_name] = len(filtered\_df)

Now we cleaned up for nulls and pitch types that were used too infrequently.

#Clean up a little

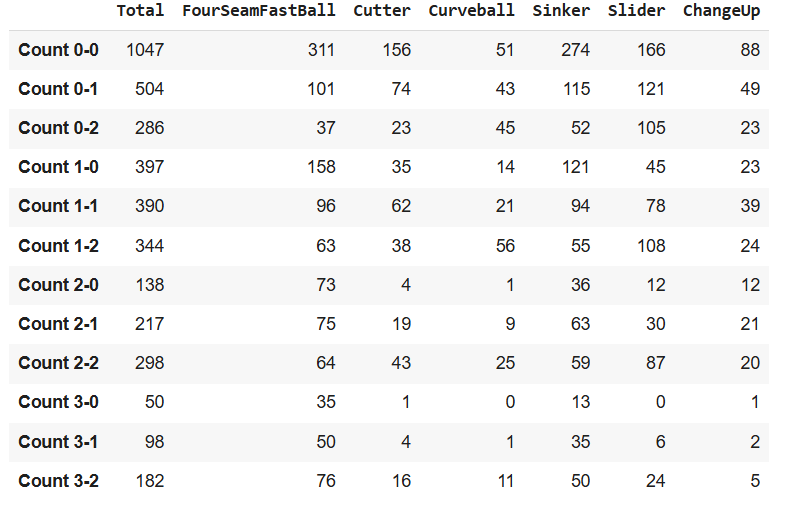
count\_sums.rename(columns={0: 'Total'}, inplace=True)

count\_sums = count\_sums.drop(columns=count\_sums.columns[count\_sums.columns.isna()])

count\_sums = count\_sums.drop(columns=['TwoSeamFastBall', 'Undefined', 'FourSeamFastball'])

count\_sums

Here is the count sums dataframe.



Lastly we had to convert these values into percentages of the total column. Resulting in the final dataframe called ‘percents’.

percents=count\_sums.copy()

percents['FourSeamFastBall']=percents['FourSeamFastBall']/percents['Total']

percents['Slider']=percents['Slider']/percents['Total']

percents['Curveball']=percents['Curveball']/percents['Total']

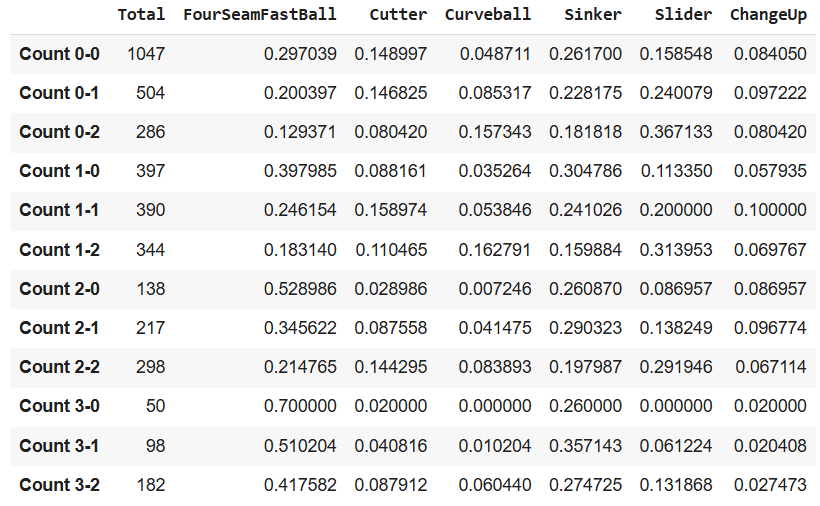
percents['Cutter']=percents['Cutter']/percents['Total']

percents['Sinker']=percents['Sinker']/percents['Total']

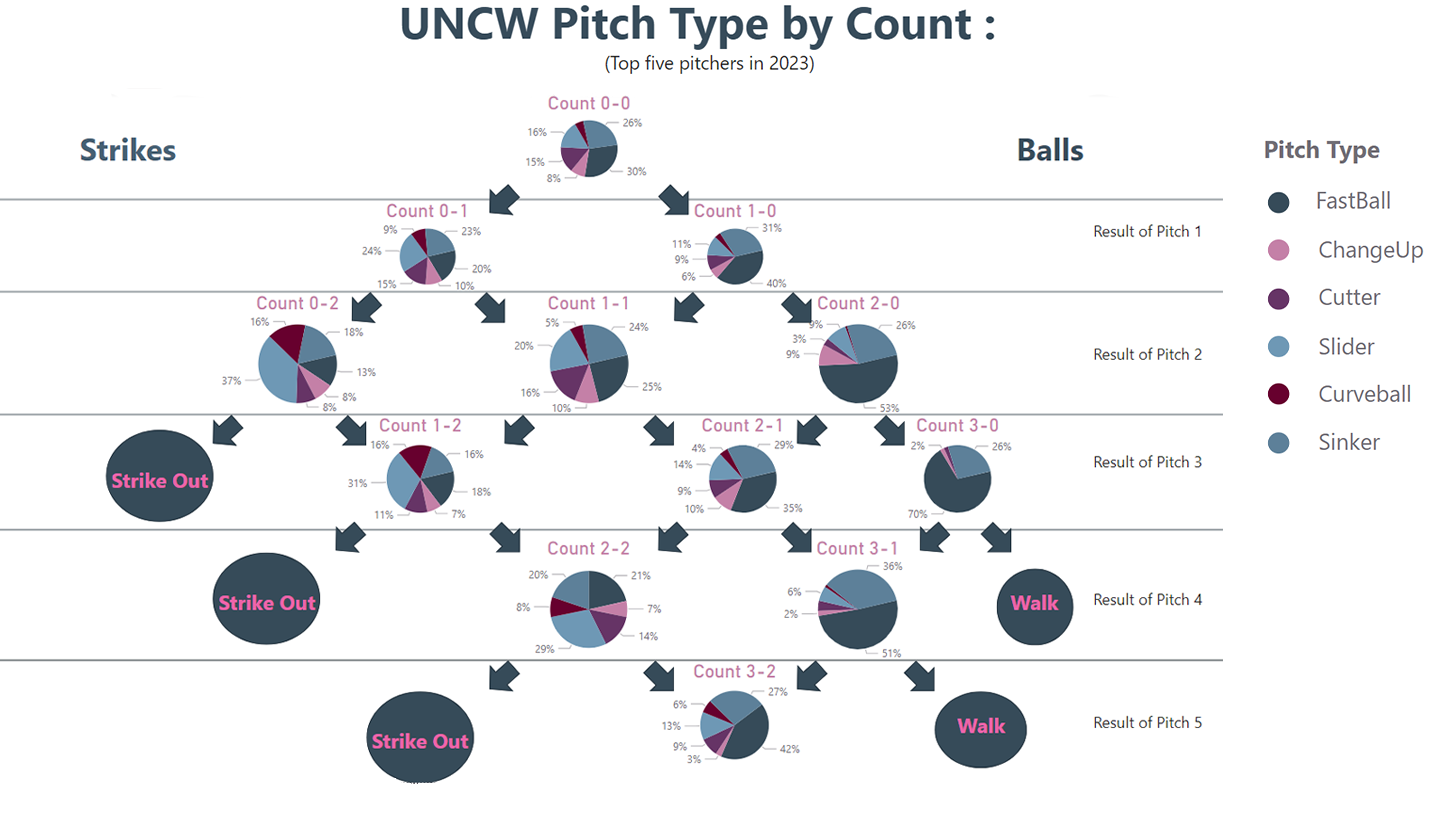
percents['ChangeUp']=percents['ChangeUp']/percents['Total']

Percents

Now we had our final dataframe.



After initially attempting to use functions to create these through matplotlib and seaborn, we decided it would be easier to export the final dataframe in a csv and use PowerBI to create the pie charts. We did this because both matplotlib and seaborn were proving difficult to achieve the diamond shape we were looking for, and PowerBI simplified this greatly. We ended up not needing the ‘percents’ dataframe in PowerBI and it made more sense to use the count\_sums dataframe. After uploading count\_sums into PowerBI we created a pie chart for each count by simply filtering for that specific row(count), and using the pitch style columns as the slices of the total column. We then manually organized them into the diamond shape. Resulting in the final visualization.



**V. Conclusion**

Data science can provide seemingly limitless insights for coaches, players, and fans alike. Through this project, we were able to determine where the UNCW pitchers were throwing the ball, what type of pitches they were throwing, and how different situations changed the way they pitched. Through our creation of the leverage columns, we were able to see how our pitchers performed under various levels of stress. Our heat map analysis allowed us to see where these pitchers were throwing the ball and whether that location was effective. Lastly our count analysis gave us insights into how our pitchers modify their pitches as the count changes. Due to the large size of our dataset, we can say that our analysis is fairly accurate, however it would be better to have data on every game UNCW has played over the course of multiple years. This would allow us more insights on rarer events and situations, whereas with our current data set, there were sometimes only a handful of data points.

We believe that our insights could be useful for multiple purposes. Firstly, it could help coaches analyze their players strengths and weaknesses in a scientific and data-backed way. For example, the coaches could see what type of pitch a specific pitcher is throwing the most strikes with, and which type of pitch is getting hit the most. Allowing the coach to focus on improving the pitch that gets hit the most.

Our leverage analysis could give coaches insights into who should be on the field in high stress situations. They can see who is performing best under stress, and make sure that player is in the game when the pressure ramps up.

Our count analysis could be useful for preparing UNCW batters for an upcoming game. For example, we could find out that at a 3-2 count the opposing pitcher is 80% likely to throw a fastball. Now during the game, if our batter finds himself in a 3-2 count against that pitcher, he can be confident a fastball is incoming. And combining this with our heatmaps, we can also prepare the player to know where the fastball is likely to land. Between the two insights, it would give our batter an advantage.

Fans of UNCW baseball could use the insights gained from our analysis to understand the team and players on a deeper level. They could use it to analyze stats on their favorite player, guess who will win a specific game, or even just to learn interesting patterns and trends of the team.

There are almost endless ways we could use this data. From machine learning models to basic player stats, we are excited about what we will do next. As for this project, we believe we achieved our objectives and discovered many real-world insights. Our Python skills developed over the course of this semester enabled us to take this massive csv, and turn it into useful information displayed in engaging and easy-to-understand visualizations.

## **V. Future Work**

After reviewing, analyzing, and dissecting the many visualizations we accumulated throughout this project, our team will begin to process the data numerically. Because the dataset offered pitch-by-pitch data we can accumulate averages of different metrics. These metrics will help us define the strengths and weaknesses of a pitcher as a whole. Metrics like Batting average allowed can showcase the pitchers ability to limit hits by taking their hits allowed and dividing it by total at-bats. As an example, we defined a function calculating the batting average allowed:

def calculate\_baa(df):

# Filter data for UNCW pitchers

uncw\_pitchers\_data = df[df['PitcherTeam'] == 'UNC\_SEA']

# Initialize a dictionary to store pitcher stats

pitcher\_stats = {}

# Sort data to process plays in order

uncw\_pitchers\_data = uncw\_pitchers\_data.sort\_values(['Date', 'Inning', 'PitchNo'])

# Get list of unique pitchers

pitchers = uncw\_pitchers\_data['Pitcher'].unique()

for pitcher in pitchers:

# Initialize stats

hits\_allowed = 0

at\_bats = 0

# Filter data for the current pitcher

pitcher\_data = uncw\_pitchers\_data[uncw\_pitchers\_data['Pitcher'] == pitcher]

# Group data by Date and Inning

grouped = pitcher\_data.groupby(['Date', 'Inning'])

for (date, inning), inning\_data in grouped:

# Iterate over plays in the inning

for idx, row in inning\_data.iterrows():

play\_result = row['PlayResult']

bborK = row['KorBB']

# Count hits allowed

if play\_result in ['Single', 'Double', 'Triple', 'HomeRun']:

hits\_allowed += 1

# Count at-bats: exclude walks, hit-by-pitches, sacrifices, and errors

if play\_result in ['Double', 'Out', 'Single','Error', 'FieldersChoice','HomeRun', 'Sacrifice' 'Triple'] or bborK == 'Strikeout':

at\_bats += 1

# Calculate BAA

baa = hits\_allowed / at\_bats if at\_bats > 0 else "N/A"

# Store stats

pitcher\_stats[pitcher] = {

'hits\_allowed': hits\_allowed,

'at\_bats': at\_bats,

'BAA': baa

}

stats\_df = pd.DataFrame.from\_dict(pitcher\_stats, orient='index')

stats\_df = stats\_df.reset\_index().rename(columns={'index': 'Pitcher'})

return stats\_df

Output:

Pitcher hits\_allowed at\_bats BAA

0 Sales, RJ 53 259 0.204633

1 Craig, Luke 24 147 0.163265

2 Kane, Connor 24 92 0.260870

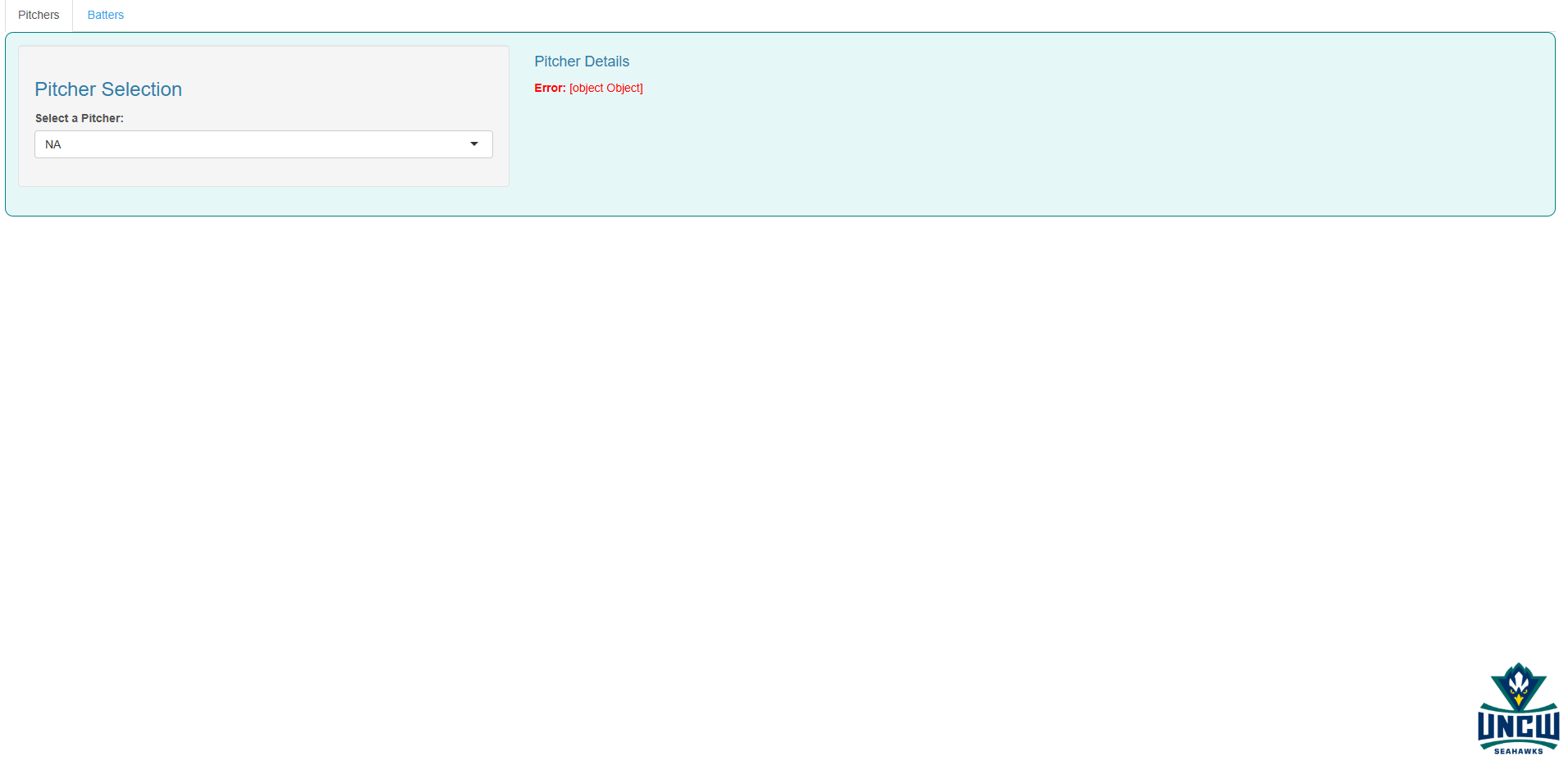
3 Shafer, Jacob 58 225 0.257778

4 Stroup, Case 18 76 0.236842

5 Baker, Trace 16 78 0.205128

6 Lawson, Ty 2 6 0.333333

There are hundreds of other metrics that can define a pitcher's abilities. Finding a select few that represent different aspects of pitching can be our first step towards using machine learning techniques. We must define key metrics or features to deliver the best prediction accuracies. Some metrics may be rudimentary and only take a couple of lines of code to process, while others may be more complex, challenging our ability to work with this dataset and coding. When we feel a proper amount of features have been created, it will be time to choose a model. Models like decision trees could help us quantify what situation-specific pitchers are most successful. If we were to explore other techniques, using K-Nearest Neighbor (KNN) would provide a simple yet effective way to classify or predict outcomes based on the proximity of data points, allowing us to leverage the similarity between observations for accurate modeling.

Several visualizations and metrics can create a cluttered environment, making it hard to understand key patterns and findings. After the conclusion of the many metrics and visualizations that will be created for every UNCW pitcher, we will construct a dashboard. Utilizing tools like R Shiny, our team can develop an interactive dashboard showcasing the key metrics for each pitcher. R Shiny is an R package that allows users to build interactive web applications directly in R. It combines the power of R with a web-based user interface, enabling users to create dashboards, data visualizations, and applications without needing web development skills. There will be some challenges working in a new environment after committing our lives to Python. We will use this as an opportunity to expand our skill sets and build much-needed flexibility and adaptability for the data science industry we will enter. Here is an example of what our dashboard looks like now. 

The user will be able to choose specific pitchers to view their key metrics and toggle between specific situations. In the future, users will be able to analyze visualizations to gain more insight into pitcher tendencies and areas for growth. There is a possibility player comparisons can be possible as well. Having a side-by-side comparison can offer the coaching staff the ability to see directly what players are performing better for their team.

After a substantial amount of unique analysis has been completed, we will attempt to collaborate with the UNCW baseball team. Sharing key findings with the staff and discussing future endeavors will help create beneficial connections and resources. The beautiful part of baseball is there is essentially an unlimited amount of data that can be utilized for research, so who knows what else the future can hold. The game of baseball can never be conquered by data scientists but as we keep immersing ourselves in it we come one step closer to uncovering deeper insights that enhance our understanding of the game. By collaborating with the UNCW baseball team, we can not only apply our findings in practical scenarios but also inspire innovative approaches to performance analysis and strategy development. This partnership has the potential to bridge the gap between data-driven research and on-field application, pushing the boundaries of what analytics can achieve in baseball.

This project explores pitching performance and game situations using Trackman data from the UNCW baseball team. Our primary focus is to analyze pitch effectiveness based on leverage situations, handedness matchups, and count-based decision-making. We developed custom metrics, including leverage categorization and adjusted scoring, to assess pitcher performance under pressure.

Currently, we have created multiple visualizations, including heatmaps of pitch locations, scatter plots of leverage-based pitch accuracy, and pie charts analyzing pitch selection across different counts. Early findings suggest that UNCW pitchers struggle with control in medium-leverage situations and adjust their pitch selection based on the count.

Future work involves calculating advanced metrics such as batting average against (BAA) and developing predictive models using machine learning. Additionally, we aim to create an interactive R Shiny dashboard to provide real-time analytics for coaches and players. This research has the potential to enhance in-game decision-making and player development at UNCW.