Predicting Pitch Outcomes for Major League Pitchers in 2024

Introduction to Data Science | Fall 2024

Sara Clokey

2024-11-18

Quick Survey!

Data Science in Baseball

- Relatively new phenomenon - last 20 years
- Sabermetrics: "the search for objective knowledge about baseball" (Costa, Huber, and Saccoman (2019))
- First wave: developing new statistics (Tango, Lichtman, and Dolphin (2007))
- Second wave: advanced tracking technology (Healey (2017))
- Third (current) wave: using models for predicting player performance

Research Question at Hand

- How can we predict pitch outcomes using sabermetric variables?
- How can we assess relative pitcher performance?

Data

- Pitch level data from Major League Baseball for the first half of the 2024 season ("CSAS 2025: Data Challenge" (2024))
- After processing:
  - Over 343,000 data points
  - 68 variables
- Potential limitation

Baseball Basics

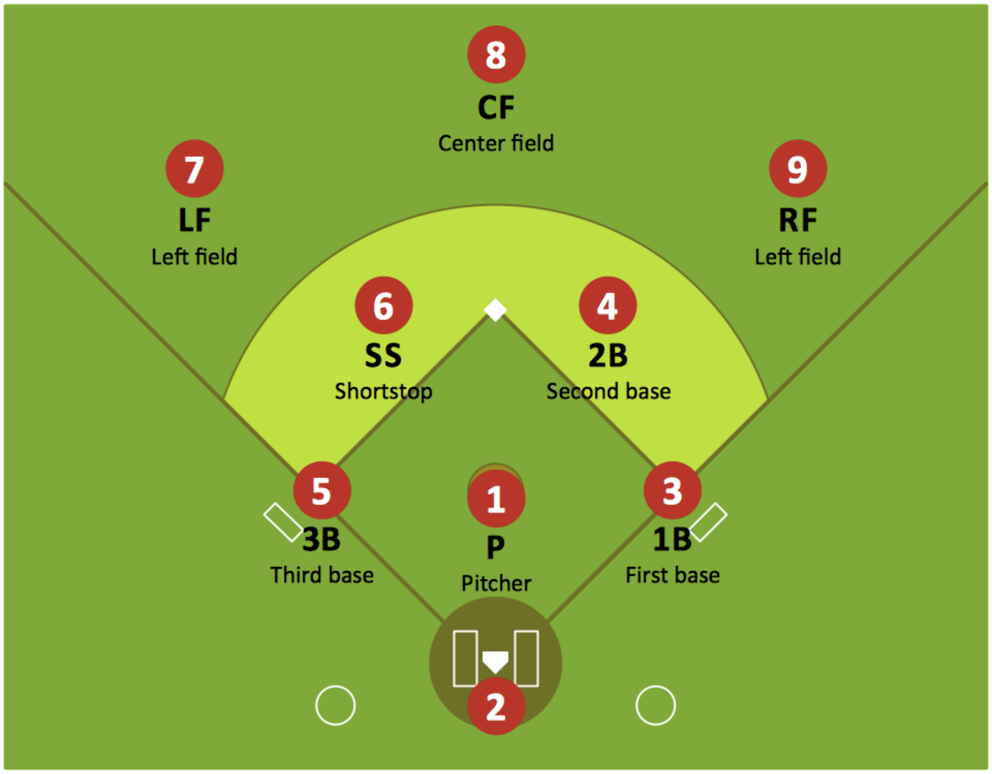


Figure 1: Baseball Field Positioning

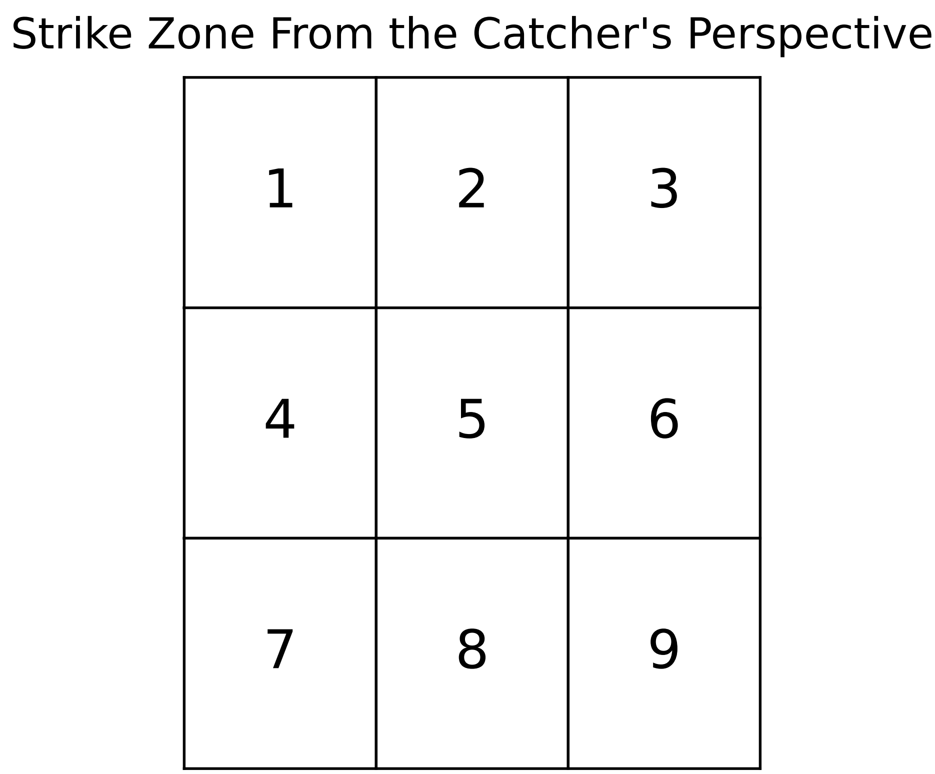
Defining Pitch Outcomes

- The result of every pitch
  - If batter hits the ball:
    * fielded out (ball is caught or thrown to a base)
    * hit
  - If batter does not hit the ball:
    * ball (called by umpire outside of the strike zone)
    * strike (called by umpire inside the strike zone or batter swings and misses)

Pitch Zone Visualization

```
import matplotlib.pyplot as plt

fig, axs = plt.subplots(3, 3, figsize=(3, 4),
```

```
                                gridspec_kw={'wspace': 0, 'hspace': 0})

fig.suptitle("Strike Zone From the Catcher's Perspective", fontsize=16)

axs[0, 0].text(0.5, 0.5, '1', fontsize=20, ha='center', va='center')
axs[0, 1].text(0.5, 0.5, '2', fontsize=20, ha='center', va='center')
axs[0, 2].text(0.5, 0.5, '3', fontsize=20, ha='center', va='center')

axs[1, 0].text(0.5, 0.5, '4', fontsize=20, ha='center', va='center')
axs[1, 1].text(0.5, 0.5, '5', fontsize=20, ha='center', va='center')
axs[1, 2].text(0.5, 0.5, '6', fontsize=20, ha='center', va='center')

axs[2, 0].text(0.5, 0.5, '7', fontsize=20, ha='center', va='center')
axs[2, 1].text(0.5, 0.5, '8', fontsize=20, ha='center', va='center')
axs[2, 2].text(0.5, 0.5, '9', fontsize=20, ha='center', va='center')

for ax in axs.flat:
    ax.set_xticks([])
    ax.set_yticks([])

for ax in axs.flat:
    for _, spine in ax.spines.items():
        spine.set_visible(True)
        spine.set_linewidth(1)
        spine.set_edgecolor('black')

plt.tight_layout(pad=0)
plt.subplots_adjust(top=0.9)
plt.show()
```

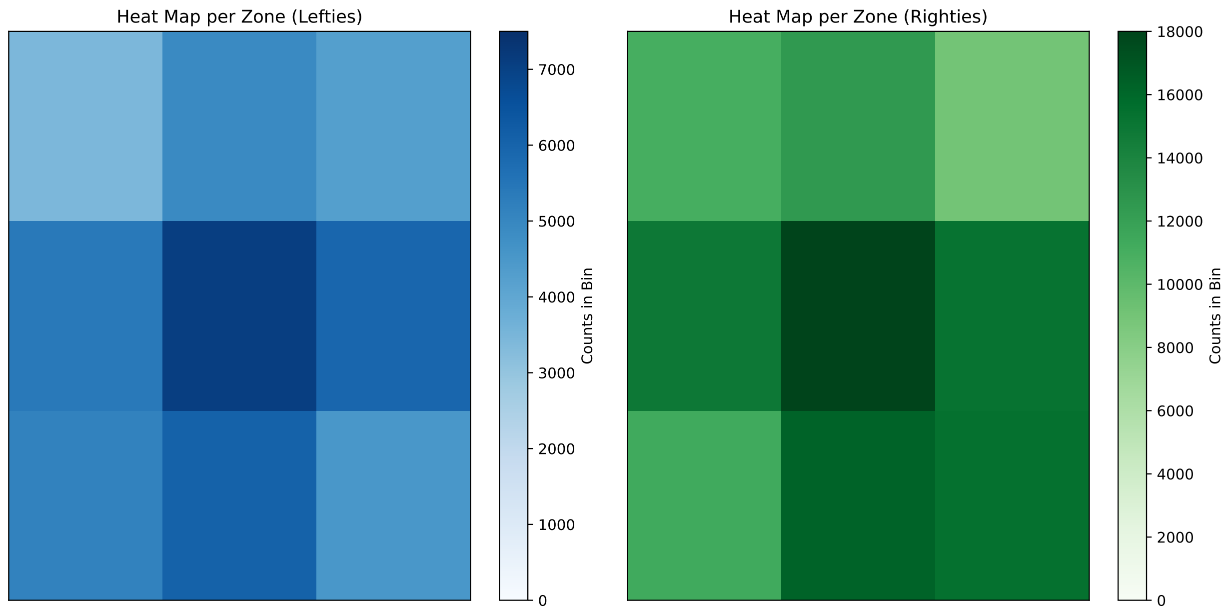


Pitch Zone Usage in MLB Data

```python
import os
import pandas as pd
import pyarrow as pa

file_path = (
    'data/statcast_pitch_swing_data_20240402_20240630.arrow'
)

table = pa.ipc.open_file(file_path).read_all()

df = table.to_pandas()

df.rename(columns={'type': 'category'}, inplace=True)

missing_percentage = df.isnull().mean() * 100

high_missing_columns = missing_percentage[missing_percentage >= 65].index
```

```python
df = df.drop(high_missing_columns, axis=1)

df = df.drop(['bat_speed', 'swing_length'], axis=1)


are_fielder2_equal = (df['fielder_2'] == df['fielder_2_1']).all()

df = df.drop(['game_year', 'fielder_2_1'], axis=1)

out_of_range = df[(df['release_speed'] < 60) | (df['release_speed'] > 105)]
unique_out_of_range = out_of_range['release_speed'].unique()

df.drop(
    df[(df['release_speed'] < 60) | (df['release_speed'] > 105)].index,
    inplace=True
)

unique_home_teams = df['home_team'].unique()

unique_away_teams = df['away_team'].unique()

clean_df = df.dropna()

def assign_x_coord(row):
    if row.zone in [1, 4, 7]:
        return 1
    if row.zone in [2, 5, 8]:
        return 2
    if row.zone in [3, 6, 9]:
        return 3

def assign_y_coord(row):
    if row.zone in [1, 2, 3]:
        return 3
    if row.zone in [4, 5, 6]:
        return 2
    if row.zone in [7, 8, 9]:
        return 1

clean_df_zones = clean_df.copy().loc[df.zone <= 9]

clean_df_zones['zone_x'] = clean_df_zones.apply(assign_x_coord, axis=1)
clean_df_zones['zone_y'] = clean_df_zones.apply(assign_y_coord, axis=1)
```

```python
clean_df_lefties = clean_df_zones[clean_df_zones['p_throws'] == 'L']
clean_df_righties = clean_df_zones[clean_df_zones['p_throws'] == 'R']

plt.figure(figsize=(12, 6))

plt.subplot(1, 2, 1)
plt.hist2d(
    clean_df_lefties.zone_x, clean_df_lefties.zone_y, bins=3, cmap='Blues',
    vmin=0, vmax=7500
)
plt.title('Heat Map per Zone (Lefties)')
plt.gca().get_xaxis().set_visible(False)
plt.gca().get_yaxis().set_visible(False)
cb_left = plt.colorbar()
cb_left.set_label('Counts in Bin')


plt.subplot(1, 2, 2)
plt.hist2d(
    clean_df_righties.zone_x, clean_df_righties.zone_y, bins=3, cmap='Greens',
    vmin=0, vmax=18000
)
plt.title('Heat Map per Zone (Righties)')
plt.gca().get_xaxis().set_visible(False)
plt.gca().get_yaxis().set_visible(False)
cb_right = plt.colorbar()
cb_right.set_label('Counts in Bin')

plt.tight_layout()
plt.show()
```



Pitch Groups

- Create groups based on the goal of a pitch
- Fastballs: focus on velocity
- Breaking balls: focus on horizontal and vertical movement
- Offspeed: focus on disrupting batter's timing
- Knuckleball: focus on a lack of spin

Velocity

- The maximum speed of a given pitch at any point from its release to the time it crosses home plate
- Measured in miles per hour (MPH)
- Focus of fastball group

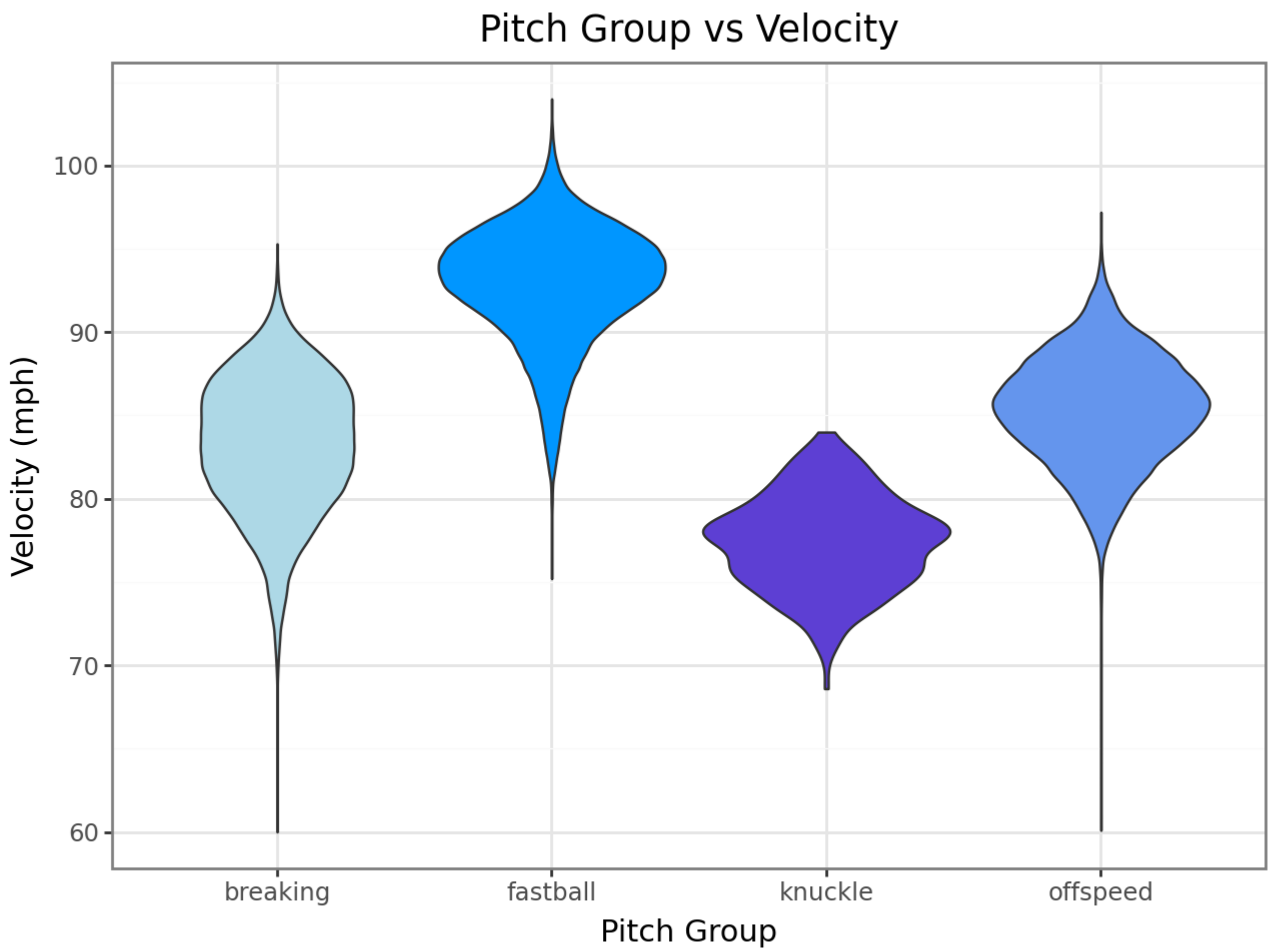
Velocity in MLB Data

```
bad_events = ['single', 'walk', 'home_run', 'double', 'field_error',
              'hit_by_pitch', 'catcher_interf', 'triple', 'sac_fly',
              'sac_bunt', 'stolen_base_2b']
good_events = ['strikeout', 'field_out', 'force_out',
               'grounded_into_double_play', 'double_play', 'fielders_choice',
               'caught_stealing_home', 'fielders_choice_out',
```

```
               'caught_stealing_2b', 'strikeout_double_play',
               'caught_stealing_3b', 'other_out',
               'pickoff_caught_stealing_home', 'pickoff_caught_stealing_3b',
               'pickoff_3b', 'sac_fly_double_play', 'pickoff_1b', 'triple_play']

def classify_pitch_outcome(row):
    if row['events'] in bad_events or row['category'] == 'B':
        return '0'
    elif row['events'] in good_events or row['category'] == 'S':
        return '1'
    else:
        return 'None'

clean_df['pitch_outcome'] = clean_df.apply(classify_pitch_outcome, axis=1)

clean_df = clean_df.drop(
    clean_df[clean_df['pitch_type'].isin(['PO', 'EP', 'FA', 'CS'])].index
)

outcome_counts = clean_df['pitch_outcome'].value_counts()

pitch_group_mapping = {
    'FC': 'fastball', 'FF': 'fastball', 'FS': 'fastball', 'SI': 'fastball',
    'FO': 'fastball', 'SL': 'breaking', 'ST': 'breaking', 'CU': 'breaking',
    'SC': 'breaking', 'KC': 'breaking', 'SV': 'breaking', 'CH': 'offspeed',
    'KN': 'knuckle'
}

clean_df['pitch_group'] = clean_df['pitch_type'].apply(
    lambda x: pitch_group_mapping.get(x, 'unknown')
)

from plotnine import *

clean_filtered_df = clean_df[clean_df['pitch_group'] != 'unknown']

colors = ['#ADD8E6', '#0096FF', '#5D3FD3', '#6495ED']

violin_plot = (
    ggplot(clean_filtered_df,
           aes(x='pitch_group', y='release_speed', fill='pitch_group'))
    + geom_violin(show_legend=False)
    + scale_fill_manual(values=colors)
    + labs(title='Pitch Group vs Velocity', x='Pitch Group', y='Velocity (mph)')
```

```
    + theme_bw()
)

violin_plot.show()
```

```
C:\Users\18607\AppData\Local\Temp\ipykernel_13680\2694131178.py:20: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/ind
```



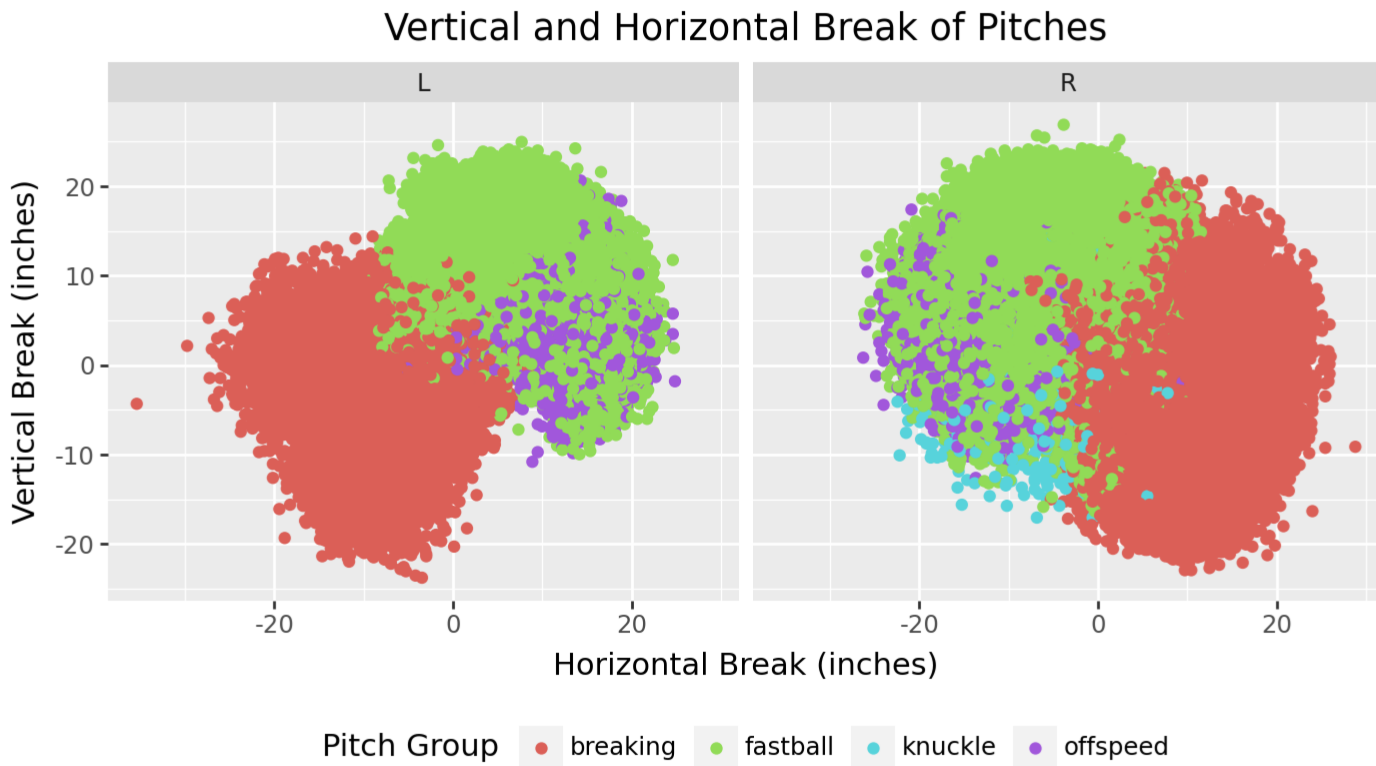
Break

- Horizontal break: distance, in inches, of lateral movement that a pitcher imparts on a ball (Maschino (2023))

- Induced vertical break: distance, in inches, of vertical movement that a pitcher imparts on a ball (Maschino (2023))
    - considered in a vaccuum due to gravity
- Focus of breaking balls pitch group

Break in MLB Data

```
clean_df['pfx_x'] = clean_df['pfx_x'] * 12
clean_df['pfx_z'] = clean_df['pfx_z'] * 12

(ggplot(clean_df, aes(x='pfx_x', y='pfx_z', color='pitch_group')) +
    geom_point() +
    labs(title='Vertical and Horizontal Break of Pitches',
         x='Horizontal Break (inches)',
         y='Vertical Break (inches)',
         color='Pitch Group') +
    coord_fixed(ratio=1) +
    theme(figure_size=(7, 5), legend_position='bottom') +
    facet_wrap('~p_throws')
)
```



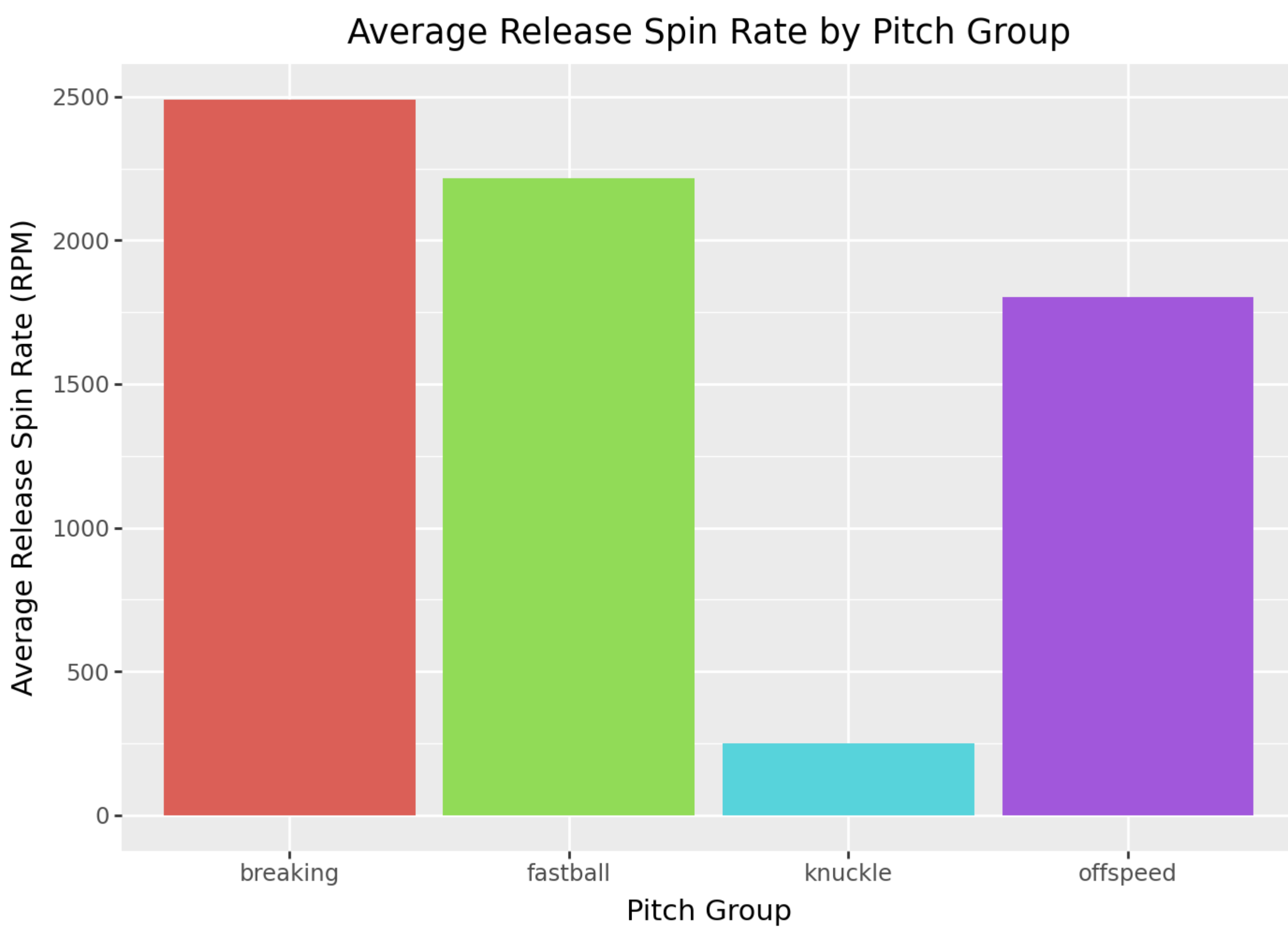
Spin Rate

- Rate of spin on a baseball after it is released in revolutions per minute (RPM)
  - RPM changes pitch trajectory; same pitch with same velocity will end up in a different place depending on how much it spins ("Spin Rate (SR)" (2024))
- Focus of knuckleball pitch group
  - Aim for lowest possible RPM

Spin Rate in MLB Data

```
avg_spin_rate_by_pitch_group = clean_filtered_df.groupby('pitch_group')[
    'release_spin_rate'].mean().reset_index()
```

```
(ggplot(avg_spin_rate_by_pitch_group,
        aes(x='pitch_group', y='release_spin_rate', fill='pitch_group'))
 + geom_bar(stat='identity', show_legend=False)
 + labs(
     title='Average Release Spin Rate by Pitch Group',
     x='Pitch Group',
     y='Average Release Spin Rate (RPM)'
   )
 + theme(figure_size=(7, 5), legend_position='bottom')
)
```



Revisiting Pitch Outcome Definition

- After analyzing some aspects of pitch performance, let's see if they affect pitch outcome
- Considered from pitcher's perspective
  - positive events sorted as 1s
  - negative events sorted as 0s
- Result is a binary variable to be predicted through model building
- Potential limitation

Prediction Model

- Using LASSO logistic regression (Tibshirani (1996))
  - Binary variable
  - Variable reduction, overfitting prevention, interpretable
- Determine if existing variables can predict whether pitch outcome is positive or negative from the pitcher's perspective

Model Results

- LASSO logistic model validation with best parameters:

```
categorical_col = [
    'pitch_type',
    'stand',
    'p_throws',
    'home_team',
    'away_team',
    'bb_type',
    'inning_topbot',
    'if_fielding_alignment',
    'of_fielding_alignment',
    'pitch_group'
]

numerical_col = [
    'release_speed', 'release_pos_x', 'release_pos_z', 'batter', 'pitcher',
    'zone', 'balls', 'strikes', 'pfx_x', 'pfx_z', 'plate_x', 'plate_z',
    'outs_when_up', 'inning', 'sz_top', 'sz_bot', 'release_spin_rate',
    'release_extension', 'game_pk', 'release_pos_y', 'at_bat_number',
    'pitch_number', 'home_score', 'away_score', 'bat_score', 'fld_score',
    'spin_axis'
```

```
]

from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
import numpy as np

numerical_transformer = StandardScaler()

categorical_transformer = OneHotEncoder()

preprocessor = ColumnTransformer(
    transformers=[
        ('cat', categorical_transformer, categorical_col),
        ('num', numerical_transformer, numerical_col)
    ]
)

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import Pipeline

pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('classifier', LogisticRegression(penalty='l1', solver='saga',
                                      max_iter=1000))
])

X = X = clean_df[numerical_col + categorical_col]
y = clean_df['pitch_outcome']

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=1918)

pipeline.fit(X_train, y_train)

model = pipeline.named_steps['classifier']
intercept = model.intercept_
coefficients = model.coef_[0]

from sklearn.metrics import (
    recall_score, precision_score, f1_score, accuracy_score, confusion_matrix
)

y_pred = pipeline.predict(X_test)
```

```python
y_test = y_test.astype(int)
y_pred = y_pred.astype(int)

accuracy = accuracy_score(y_test, y_pred)
recall = recall_score(y_test, y_pred, average='binary')
precision = precision_score(y_test, y_pred, average='binary')
f1 = f1_score(y_test, y_pred, average='binary')
cm = confusion_matrix(y_test, y_pred)

print(f"Accuracy: {accuracy}")
print(f"Recall: {recall}")
print(f"Precision: {precision}")
print(f"F1 Score: {f1}")
print("Confusion Matrix:")
print(cm)
```

```
Accuracy: 0.7816103669190448
Recall: 0.731134320134245
Precision: 0.8725490196078431
F1 Score: 0.7956066118855866
Confusion Matrix:
[[24489  4264]
 [10735 29192]]
```

Coefficients associated with pitch groups:

```python
pipeline.fit(X_train, y_train)

categorical_transformer = pipeline.named_steps['preprocessor'].transformers_[0][1]

categorical_feature_names = categorical_transformer.get_feature_names_out(
    categorical_col
)

all_feature_names = numerical_col + list(categorical_feature_names)

feature_importance = pd.DataFrame({
    'Feature': all_feature_names,
    'Coefficient': coefficients
})

feature_importance['Abs_Coefficient'] = feature_importance['Coefficient'].abs()
```

```python
pitch_group_features = [
    feature for feature in categorical_feature_names if 'pitch_group' in feature
]

feature_importance_pitch_group = feature_importance[
    feature_importance['Feature'].isin(pitch_group_features)
]

print(feature_importance_pitch_group[['Feature', 'Coefficient',
                                      'Abs_Coefficient']])
```

```
                 Feature  Coefficient  Abs_Coefficient
117  pitch_group_breaking     0.010833         0.010833
118  pitch_group_fastball     0.002829         0.002829
119   pitch_group_knuckle     0.027161         0.027161
120  pitch_group_offspeed     0.019792         0.019792
```

Results Interpretation

- Overall, this model predicts pitch outcome correctly 78.2% of the time
- The pitch group variable seems to have a meaningful contribution to this prediction capability
- Based on these results, we can use pitch group sabermetrics (velocity and break) to help assess relative pitcher performance

K-Means Clustering Assessment

- Strategy used to help pitchers "maximize their effectiveness" (Andrews et al. (2021))
- Groups players with similar break and velocity tendencies
- Sort those groups into relative rankings to find pitcher priorities

Pitchers Prioritizing Movement

- Pitches in the top 30% for both horizonal and vertical break but in the bottom 20% for velocity
- The 5 pitchers with the most pitches that fit this criteria:

```python
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

features = clean_df[['release_speed', 'pfx_x', 'pfx_z']]
```

```python
features = pd.get_dummies(features, drop_first=True)

scaler = StandardScaler()
scaled_features = scaler.fit_transform(features)

pca = PCA(n_components=2)
pca_components = pca.fit_transform(scaled_features)

kmeans = KMeans(n_clusters=5, random_state=1918)
clean_df['cluster'] = kmeans.fit_predict(pca_components)

clean_df.loc[:, 'velocity_rank'] = clean_df.groupby('cluster')['release_speed']\
    .rank(pct=True)
clean_df.loc[:, 'hbreak_rank'] = clean_df.groupby('cluster')['pfx_x']\
    .rank(pct=True)
clean_df.loc[:, 'vbreak_rank'] = clean_df.groupby('cluster')['pfx_z']\
    .rank(pct=True)

clean_df.loc[:, 'vbreak_decile'] = (clean_df['vbreak_rank'] * 10).astype(int)
clean_df.loc[:, 'velocity_decile'] = (clean_df['velocity_rank'] * 10).astype(int)
clean_df.loc[:, 'hbreak_decile'] = (clean_df['hbreak_rank'] * 10).astype(int)

clean_df['velocity_improvement_candidate'] = (
    (clean_df['vbreak_decile'] >= 3) &
    (clean_df['hbreak_decile'] >= 3) &
    (clean_df['velocity_decile'] <= 2)
)

velocity_improvement_candidates = clean_df[clean_df[
    'velocity_improvement_candidate'] == True].copy()

velocity_improvement_candidates_sorted = (
    velocity_improvement_candidates.sort_values(by='pitcher')
)

top_5_velocity_improvement_pitchers = velocity_improvement_candidates_sorted[
    'pitcher'].value_counts().head(5)

print("Top 5 Pitchers Prioritizing Movement:")
print(top_5_velocity_improvement_pitchers)
```

```
Top 5 Pitchers Prioritizing Movement:
pitcher
542881    1072
```

```
596295     745
676710     686
594902     676
684007     655
Name: count, dtype: int64
```

Pitcher Names - Prioritizing Movement

- 542881: Tyler Anderson
- 596295: Austin Gomber
- 676710: Kutter Crawford
- 594902: Ben Lively
- 684007: Shota Imanaga ("MLBAMIDs for Active MLB Players" (2024))

Pitchers Prioritizing Velocity

- Pitches in the top 20% for velocity but the bottom 30% for both horizonal and vertical break
- The 5 pitchers with the most pitches that fit this criteria:

```
clean_df['break_improvement_candidate'] = (
    (clean_df['vbreak_decile'] <= 3) &
    (clean_df['hbreak_decile'] <= 3) &
    (clean_df['velocity_decile'] >= 8)
)

break_improvement_candidates = clean_df[clean_df[
    'break_improvement_candidate'] == True].copy()

break_improvement_candidates_sorted = break_improvement_candidates.sort_values(
    by='pitcher')

top_5_break_improvement_pitchers = break_improvement_candidates_sorted[
    'pitcher'].value_counts().head(5)

print("\nTop 5 Pitchers Prioritizing Velocity:")
print(top_5_break_improvement_pitchers)
```

```
Top 5 Pitchers Prioritizing Velocity:
pitcher
667755    518
665625    367
666974    364
```

```
694973    358
656557    331
Name: count, dtype: int64
```

Pitcher Names - Prioritizing Velocity

- 667755: Jose Soriano
- 665625: Elvis Peguero
- 666974: Yennier Cano
- 694973: Paul Skenes
- 656557: Tanner Houck ("MLBAMIDs for Active MLB Players" (2024))

Results Interpretation

- Players prioritizing movement average 1.45 home runs per game and an earned run average (ERA) of 3.93 ("Baseball Stats & History" (2024))
  - Need significant zone control
- Players prioritizing velocity average 0.65 home runs per game and an ERA of 2.93 ("Baseball Stats & History" (2024))
  - Generally considered elite pitchers in 2024
  - Average shorter stints in-game: arm fatigue

Conclusions

- Pitch outcome can be predicted with relative success using sabermetric statistics
- Among those meaningful predictors, sabermetrics regarding pitch differentiation (velocity and break) demonstrate clear performance differences between pitchers

References

Andrews, Chris, Andres Castillo, Alex Steinhoff, and Russell Walker. 2021. "Using Data Analytics to Improve Pitcher Performance in Major League Baseball." *AnalyticsInforms*.

"Baseball Stats & History." 2024. *Baseball Reference*.

Costa, Gabriel, Michael Huber, and John Saccoman. 2019. *Understanding Sabermetrics: An Introduction to the Science of Baseball Statistics*. McFarland & Company.

"CSAS 2025: Data Challenge." 2024. *Connecticut Data Science Lab*.

Healey, Glenn. 2017. "The New Moneyball: How Ballpark Sensors Are Changing Baseball." *Proceedings of the IEEE* 105 (11): 1999–2002.

Maschino, Jeremy. 2023. "Pitching Metrics." *Pitch Profiler*.

"MLBAMIDs for Active MLB Players." 2024. *Razzball*.

"Spin Rate (SR)." 2024. *MLB*.

Tango, Tom, Mitchel Lichtman, and Andrew Dolphin. 2007. *The Book: Playing the Percentages in Baseball*. Potomac Books.

Tibshirani, Robert. 1996. "Regression Shrinkage and Selection via the LASSO." *Journal of the Royal Statistical Society: Series B (Methodological)* 58 (1): 267–88.