In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [2]:
```
data = pd.read_csv("pred_metrics_df.csv")
```

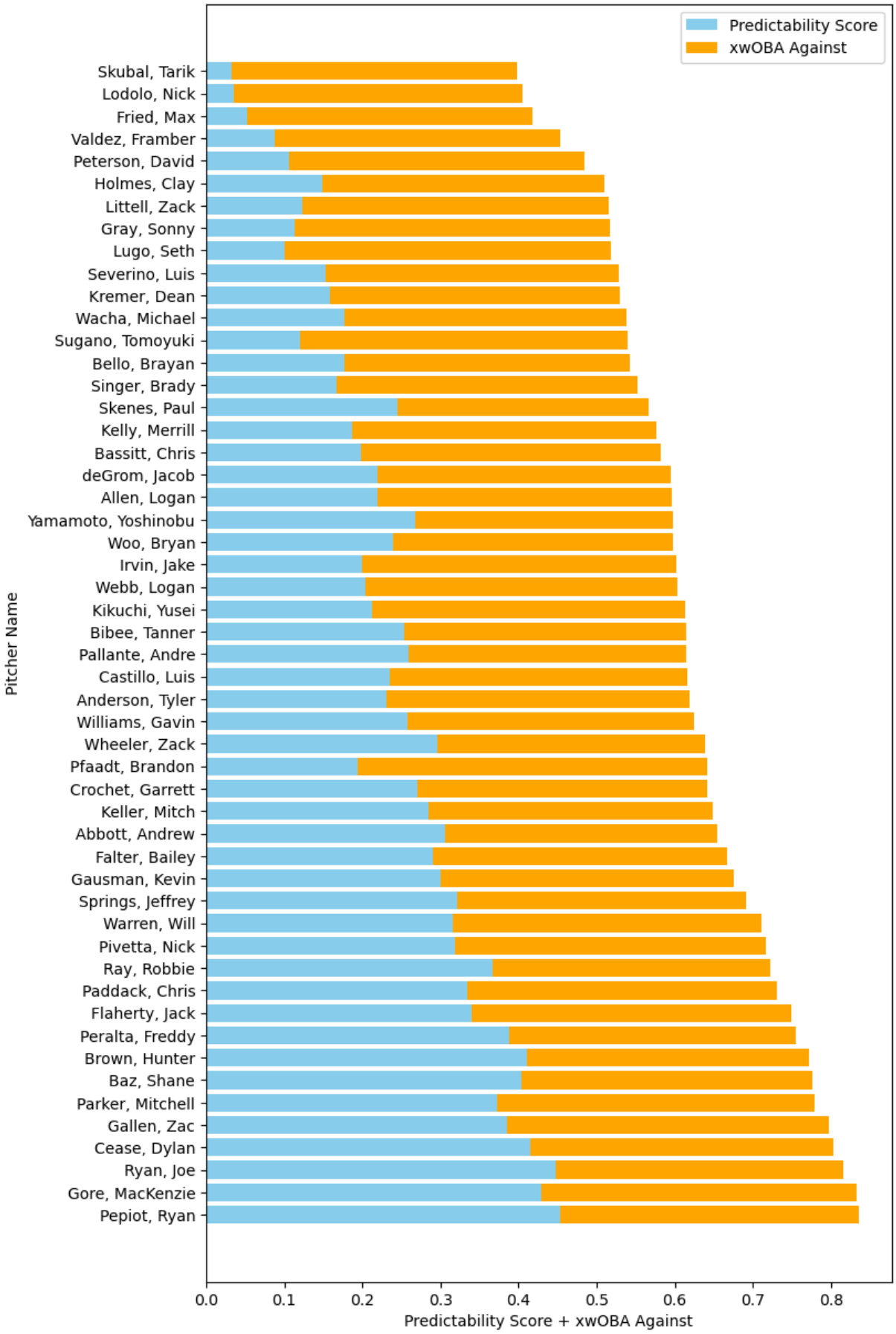
In [3]:
```
data.head()
```

Out[3]:

	mlbam_id	pitcher_name	test_accuracy	num_pitch_types	predictability_score	xw
0	686752	Pepiot, Ryan	0.544776	6	0.453731	
1	657746	Ryan, Joe	0.540000	6	0.448000	
2	669022	Gore, MacKenzie	0.542857	5	0.428571	
3	656302	Cease, Dylan	0.512500	6	0.415000	
4	686613	Brown, Hunter	0.508876	6	0.410651	

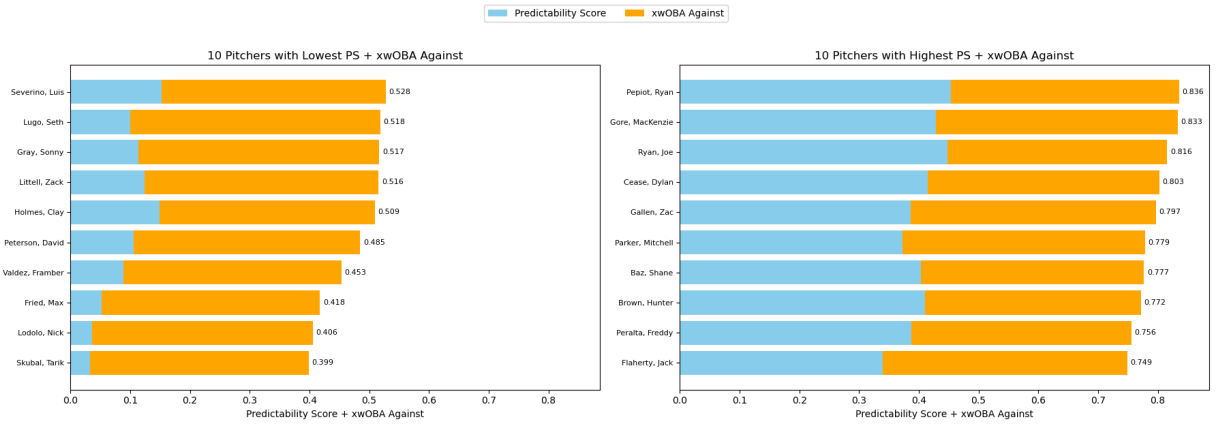
In [4]:
```
df = data.sort_values('PS_xwoba', ascending= False)
```

In [5]:
```
fig, ax = plt.subplots(figsize=(8, 15))

# First bar (predictability score)
ax.barh(df['pitcher_name'],
        df['predictability_score'],
        color='skyblue',
        label='Predictability Score')

# Second bar stacked on top of the first
ax.barh(df['pitcher_name'],
        df['xwoba_against'],
        left=df['predictability_score'],
        color='orange',
        label='xwOBA Against')

# Axis labels and legend
ax.set_xlabel('Predictability Score + xwOBA Against')
ax.set_ylabel('Pitcher Name')
ax.legend()
plt.show();
```



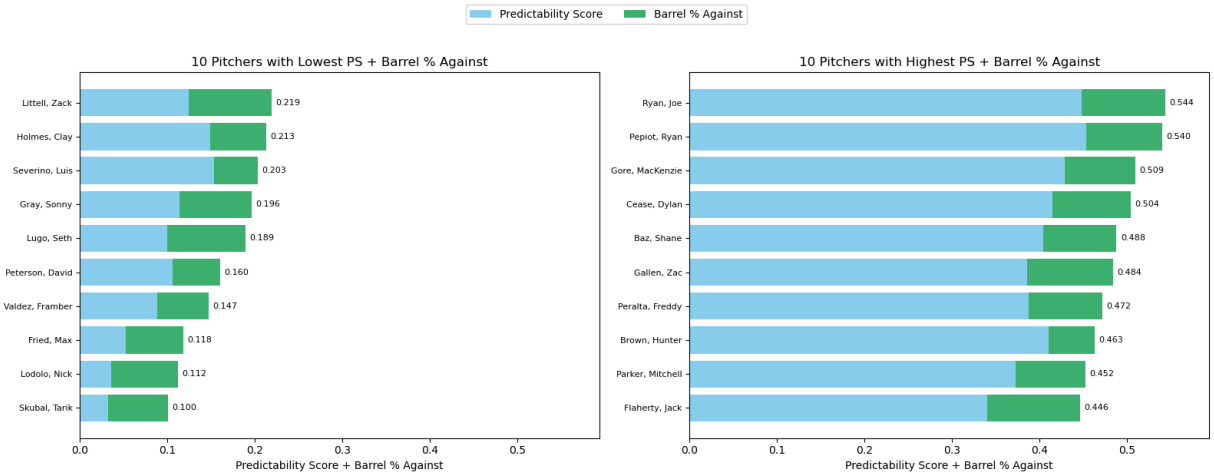
In [6]:
```
# Step 1: Get top and bottom 10 sorted ascending for clean bar order
top10_df = df.nlargest(10, 'PS_xwoba').sort_values('PS_xwoba', ascending=Tru
```

```
bottom10_df = df.nsmallest(10, 'PS_xwoba').sort_values('PS_xwoba', ascending

# Step 2: Calculate global max for consistent x-axis
global_max = max(top10_df['PS_xwoba'].max(), bottom10_df['PS_xwoba'].max())

# Step 3: Plot function
def plot_with_labels(ax, sub_df, title, max_val):
    ax.barh(sub_df['pitcher_name'],
            sub_df['predictability_score'],
            color='skyblue',
            label='Predictability Score')
    ax.barh(sub_df['pitcher_name'],
            sub_df['xwoba_against'],
            left=sub_df['predictability_score'],
            color='orange',
            label='xwOBA Against')

    ax.set_title(title)
    ax.set_xlabel('Predictability Score + xwOBA Against')
    ax.tick_params(axis='y', labelsize=8)

    # Apply same axis limit for both charts
    ax.set_xlim(0, max_val + 0.05)

    # Annotate totals
    for i, total in enumerate(sub_df['PS_xwoba']):
        ax.text(total + 0.005, i, f"{total:.3f}", va='center', fontsize=8)

# Step 4: Create figure with subplots
fig, axes = plt.subplots(ncols=2, figsize=(18, 6), sharex=True)

plot_with_labels(axes[0], bottom10_df, '10 Pitchers with Lowest PS + xwOBA A
plot_with_labels(axes[1], top10_df, '10 Pitchers with Highest PS + xwOBA Aga

# Step 5: One legend above both plots
handles, labels = axes[0].get_legend_handles_labels()
fig.legend(handles, labels, loc='upper center', ncol=2, bbox_to_anchor=(0.5,

plt.tight_layout(rect=[0, 0, 1, 0.95])
plt.show();
```



In [7]:
```
# Top/bottom 10 by PS_barrel, sorted ascending
top10_b = df.nlargest(10, 'PS_barrel').sort_values('PS_barrel', ascending=Tr
```

```
bot10_b = df.nsmallest(10, 'PS_barrel').sort_values('PS_barrel', ascending=T

# Calculate a global max for consistent x-axis
global_max = max(top10_b['PS_barrel'].max(), bot10_b['PS_barrel'].max())

def plot_ps_metric(ax, sub_df, title, metric_col, metric_label, metric_color
    ax.barh(sub_df['pitcher_name'],
            sub_df['predictability_score'],
            color='skyblue',
            label='Predictability Score')
    ax.barh(sub_df['pitcher_name'],
            sub_df[metric_col],
            left=sub_df['predictability_score'],
            color=metric_color,
            label=metric_label)

    ax.set_title(title)
    ax.set_xlabel('Predictability Score + ' + metric_label)
    ax.tick_params(axis='y', labelsize=8)

    # Apply same x-axis range to both
    ax.set_xlim(0, max_val + 0.05)

    # Annotate totals
    for i, total in enumerate(sub_df['PS_barrel']):
        ax.text(total + 0.005, i, f"{total:.3f}", va='center', fontsize=8)

# Create subplots
fig, axes = plt.subplots(ncols=2, figsize=(16, 6), sharex=True)

plot_ps_metric(
    axes[0], bot10_b,
    '10 Pitchers with Lowest PS + Barrel % Against',
    metric_col='barrel_percent_against',
    metric_label='Barrel % Against',
    metric_color='mediumseagreen',
    max_val=global_max
)

plot_ps_metric(
    axes[1], top10_b,
    '10 Pitchers with Highest PS + Barrel % Against',
    metric_col='barrel_percent_against',
    metric_label='Barrel % Against',
    metric_color='mediumseagreen',
    max_val=global_max
)

# Single legend above both
handles, labels = axes[0].get_legend_handles_labels()
fig.legend(handles, labels, loc='upper center', ncol=2, bbox_to_anchor=(0.5,

plt.tight_layout(rect=[0, 0, 1, 0.95])
plt.show();
```

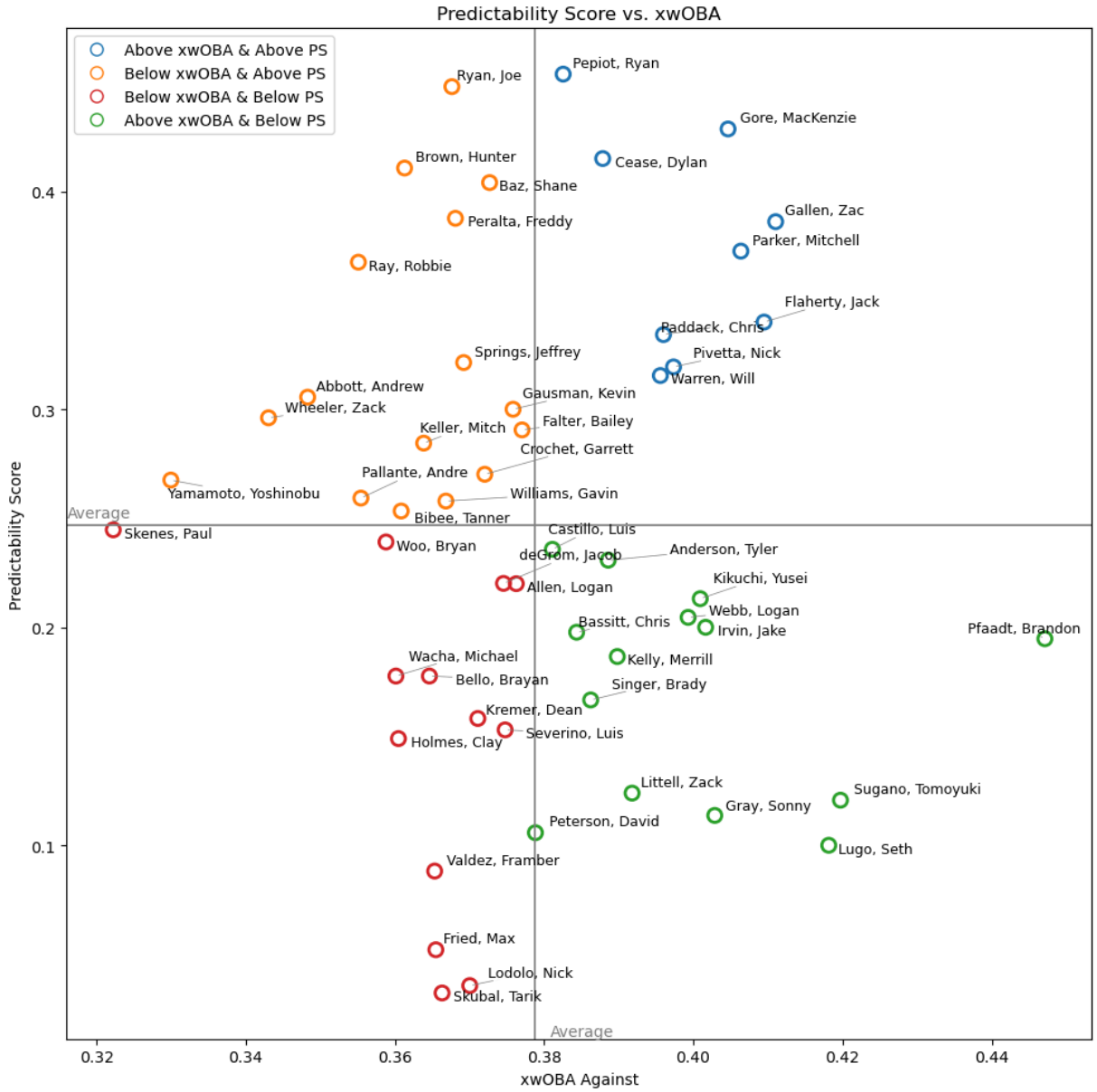


In [8]:
```
!pip install adjustText
```

```
Collecting adjustText
  Downloading adjustText-1.3.0-py3-none-any.whl.metadata (3.1 kB)
Requirement already satisfied: numpy in /opt/anaconda3/lib/python3.13/site-p
ackages (from adjustText) (2.1.3)
Requirement already satisfied: matplotlib in /opt/anaconda3/lib/python3.13/s
ite-packages (from adjustText) (3.10.0)
Requirement already satisfied: scipy in /opt/anaconda3/lib/python3.13/site-p
ackages (from adjustText) (1.15.3)
Requirement already satisfied: contourpy>=1.0.1 in /opt/anaconda3/lib/python
3.13/site-packages (from matplotlib->adjustText) (1.3.1)
Requirement already satisfied: cycler>=0.10 in /opt/anaconda3/lib/python3.1
3/site-packages (from matplotlib->adjustText) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in /opt/anaconda3/lib/pytho
n3.13/site-packages (from matplotlib->adjustText) (4.55.3)
Requirement already satisfied: kiwisolver>=1.3.1 in /opt/anaconda3/lib/pytho
n3.13/site-packages (from matplotlib->adjustText) (1.4.8)
Requirement already satisfied: packaging>=20.0 in /opt/anaconda3/lib/python
3.13/site-packages (from matplotlib->adjustText) (24.2)
Requirement already satisfied: pillow>=8 in /opt/anaconda3/lib/python3.13/si
te-packages (from matplotlib->adjustText) (11.1.0)
Requirement already satisfied: pyparsing>=2.3.1 in /opt/anaconda3/lib/python
3.13/site-packages (from matplotlib->adjustText) (3.2.0)
Requirement already satisfied: python-dateutil>=2.7 in /opt/anaconda3/lib/py
thon3.13/site-packages (from matplotlib->adjustText) (2.9.0.post0)
Requirement already satisfied: six>=1.5 in /opt/anaconda3/lib/python3.13/sit
e-packages (from python-dateutil>=2.7->matplotlib->adjustText) (1.17.0)
Downloading adjustText-1.3.0-py3-none-any.whl (13 kB)
Installing collected packages: adjustText
Successfully installed adjustText-1.3.0
```

In [9]:
```
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.lines import Line2D
from adjustText import adjust_text  # pip install adjustText

# Calculate averages
ps_mean   = df["predictability_score"].mean()
xwoba_mean = df["xwoba_against"].mean()
```

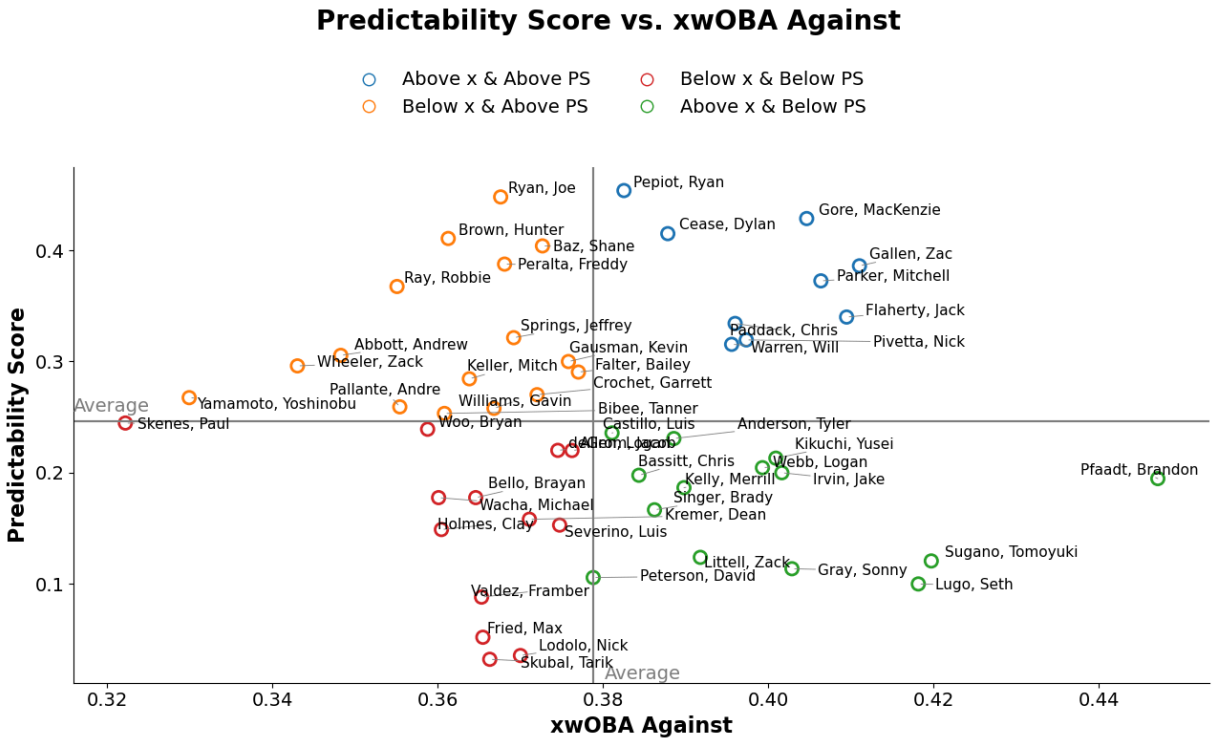
```
# Assign colors by quadrant
def color_for_row(row):
    ps_above    = row["predictability_score"] >= ps_mean
    xwoba_above = row["xwoba_against"]       >= xwoba_mean
    if  ps_above and  xwoba_above: return "tab:blue"
    if  ps_above and not xwoba_above: return "tab:orange"
    if not ps_above and not xwoba_above: return "tab:red"
    return "tab:green"

df = df.copy()
df["color"] = df.apply(color_for_row, axis=1)

fig, ax = plt.subplots(figsize=(10, 10))

# Scatter
ax.scatter(df["xwoba_against"], df["predictability_score"],
           facecolors="none", edgecolors=df["color"], s=80, linewidths=2)

# Reference lines
ax.axvline(xwoba_mean, color="gray", linewidth=1.2)
ax.axhline(ps_mean,   color="gray", linewidth=1.2)

# Average labels
ax.text(ax.get_xlim()[0], ps_mean + 0.002, "Average", color="gray", va="bott
ax.text(xwoba_mean + 0.002, ax.get_ylim()[0], "Average", color="gray", va="b

# Labels with adjustText
texts = []
for _, r in df.iterrows():
    texts.append(ax.text(r["xwoba_against"], r["predictability_score"],
                         r["pitcher_name"], fontsize=9))

adjust_text(texts, arrowprops=dict(arrowstyle="-", color='gray', lw=0.5))

# Legend
legend_elements = [
    Line2D([0], [0], marker='o', color='tab:blue',  label='Above xwOBA & Abo
           markerfacecolor='none', linestyle='None', markersize=8),
    Line2D([0], [0], marker='o', color='tab:orange',label='Below xwOBA & Abo
           markerfacecolor='none', linestyle='None', markersize=8),
    Line2D([0], [0], marker='o', color='tab:red',   label='Below xwOBA & Bel
           markerfacecolor='none', linestyle='None', markersize=8),
    Line2D([0], [0], marker='o', color='tab:green', label='Above xwOBA & Bel
           markerfacecolor='none', linestyle='None', markersize=8),
]
ax.legend(handles=legend_elements, loc='upper left')

ax.set_title("Predictability Score vs. xwOBA")
ax.set_xlabel("xwOBA Against")
ax.set_ylabel("Predictability Score")

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

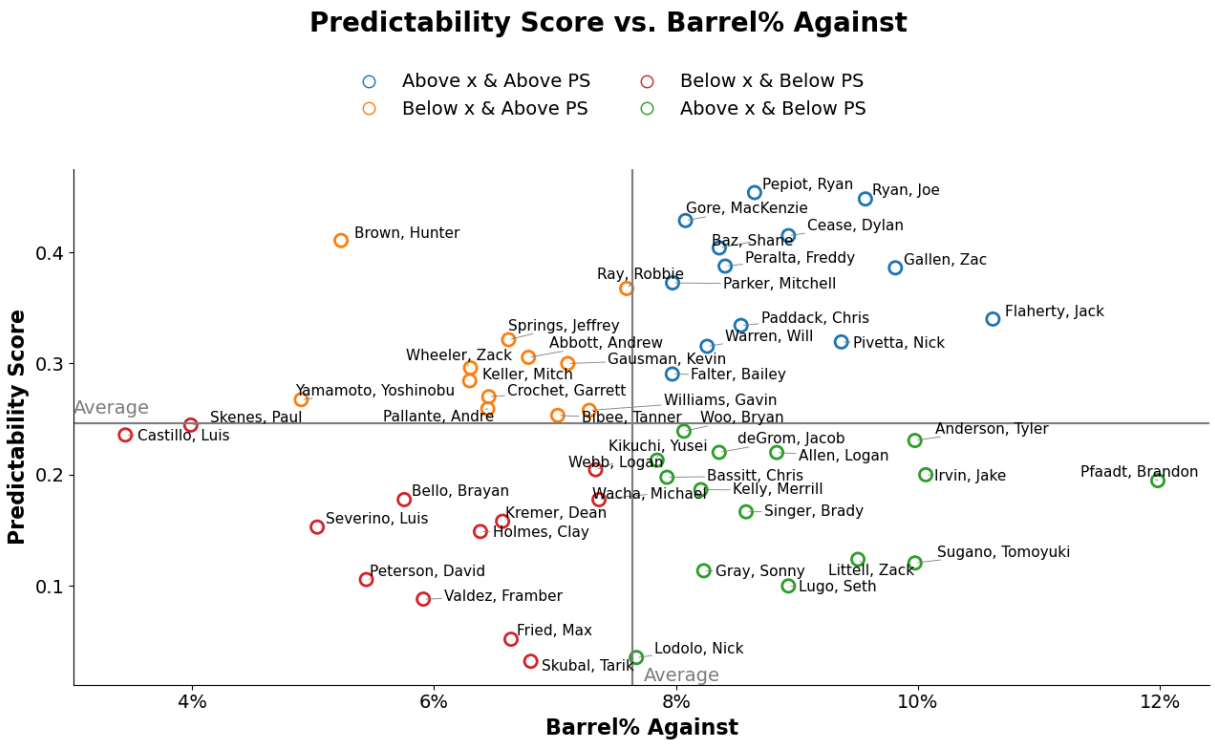


In [10]:
```
import matplotlib.pyplot as plt
from matplotlib.lines import Line2D
from matplotlib.ticker import FuncFormatter
from adjustText import adjust_text  # pip install adjustText

plt.rcParams.update({
    "font.size": 14,
    "axes.titleweight": "bold",
    "axes.labelweight": "semibold",
    "axes.spines.top": False,
    "axes.spines.right": False,
})

def plot_quadrant_scatter(df, x_col, x_label, title, out_path=None, fig_size
    data = df.copy()
    ps_mean = data["predictability_score"].mean()
    x_mean  = data[x_col].mean()

    # quadrant colors
```

```
    def assign_color(r):
        ps_above = r["predictability_score"] >= ps_mean
        x_above  = r[x_col] >= x_mean
        if  ps_above and  x_above:  return "tab:blue"
        if  ps_above and not x_above: return "tab:orange"
        if not ps_above and not x_above: return "tab:red"
        return "tab:green"
    data["color"] = data.apply(assign_color, axis=1)

    fig, ax = plt.subplots(figsize=fig_size)
    fig.suptitle(title, y=0.98, fontsize=20, fontweight="bold")
    plt.subplots_adjust(top=0.88)

    # scatter
    ax.scatter(data[x_col], data["predictability_score"],
               facecolors="none", edgecolors=data["color"], s=90, linewidths

    # reference lines
    ax.axvline(x_mean, color="gray", linewidth=1.5)
    ax.axhline(ps_mean, color="gray", linewidth=1.5)

    # average labels
    ax.text(ax.get_xlim()[0], ps_mean + (ax.get_ylim()[1]-ax.get_ylim()[0])*
            "Average", color="gray", va="bottom")
    ax.text(x_mean + (ax.get_xlim()[1]-ax.get_xlim()[0])*0.01, ax.get_ylim()
            "Average", color="gray", va="bottom")

    # pitcher labels with collision-avoidance
    texts = [ax.text(r[x_col], r["predictability_score"], r["pitcher_name"],
             for _, r in data.iterrows()]
    adjust_text(texts, arrowprops=dict(arrowstyle="-", color="gray", lw=0.6)

    # axis labels
    ax.set_xlabel(x_label, fontsize=16)
    ax.set_ylabel("Predictability Score", fontsize=16)

    # conditional percentage formatting
    if format_percent:
        ax.xaxis.set_major_formatter(FuncFormatter(lambda v, _: f"{v:.0%}"))

    # legend between title and plot
    legend_elements = [
        Line2D([0], [0], marker='o', color='tab:blue',  label='Above x & Abo
               markerfacecolor='none', linestyle='None', markersize=9),
        Line2D([0], [0], marker='o', color='tab:orange',label='Below x & Abo
               markerfacecolor='none', linestyle='None', markersize=9),
        Line2D([0], [0], marker='o', color='tab:red',   label='Below x & Bel
               markerfacecolor='none', linestyle='None', markersize=9),
        Line2D([0], [0], marker='o', color='tab:green', label='Above x & Bel
               markerfacecolor='none', linestyle='None', markersize=9),
    ]
    fig.legend(handles=legend_elements, loc="upper center", ncol=2,
               frameon=False, bbox_to_anchor=(0.5, 0.92))

    plt.tight_layout(rect=[0, 0, 1, 0.86])
```

```
    if out_path:
        plt.savefig(out_path, dpi=dpi, bbox_inches="tight")
    plt.show()


# === xwOBA Against (decimal form) ===
plot_quadrant_scatter(
    df,
    x_col="xwoba_against",
    x_label="xwOBA Against",
    title="Predictability Score vs. xwOBA Against",
    out_path="ps_vs_xwoba_slide.png",
    format_percent=False
)

# === Barrel% Against (percentage form) ===
plot_quadrant_scatter(
    df,
    x_col="barrel_percent_against",  # keep as fraction, format as %
    x_label="Barrel% Against",
    title="Predictability Score vs. Barrel% Against",
    out_path="ps_vs_barrel_slide.png",
    format_percent=True
)
```




Predictability Score vs. Barrel% Against
Above x & Above PS
Below x & Above PS
Below x & Below PS
Above x & Below PS
Predictability Score
Barrel% Against
0.4
0.3
0.2
0.1
4%
6%
8%
10%
12%
Average
Average
Pepiot, Ryan
Ryan, Joe
Gore, MacKenzie
Cease, Dylan
Baz, Shane
Peralta, Freddy
Gallen, Zac
Brown, Hunter
Ray, Robbie
Parker, Mitchell
Flaherty, Jack
Paddack, Chris
Springs, Jeffrey
Warren, Will
Abbott, Andrew
Pivetta, Nick
Wheeler, Zack
Gausman, Kevin
Keller, Mitch
Falter, Bailey
Yamamoto, Yoshinobu
Crochet, Garrett
Williams, Gavin
Skenes, Paul
Pallante, Andre
Bibee, Tanner
Woo, Bryan
Castillo, Luis
Anderson, Tyler
deGrom, Jacob
Kikuchi, Yusei
Allen, Logan
Webb, Logan
Bassitt, Chris
Irvin, Jake
Pfaadt, Brandon
Kelly, Merrill
Bello, Brayan
Wacha, Michael
Singer, Brady
Severino, Luis
Kremer, Dean
Holmes, Clay
Sugano, Tomoyuki
Peterson, David
Gray, Sonny
Littell, Zack
Lugo, Seth
Valdez, Framber
Fried, Max
Lodolo, Nick
Skubal, Tarik