STA 402 Term Project Final Submission

Ryan Yu – STA 402 Section A

Original Statement of Assigned Task: Write a SAS macro program that accepts a year and player name specified by the user and creates a graphical and summary analysis of the on-base percentage (OBP) of that player for that year. Include separate breakdowns of this statistic by month and opposing team.

Tables and Graphs Produced By Code

As a preface, these outputs are created by the following invocation of my macro:

```
%getobpforplayer(player=TJ Friedl, year=2023)
```

The first piece of output produced by my code is a very small and simple table from PROC PRINT, containing only the season long on-base percentage (OBP) for the given player in the given year. By itself, this is not meant to be a very insightful or exciting output, but it is useful for comparisons in the further breakdowns later in the output.

Season Long OBP for TJ Friedl in 2023

OBP
0.352

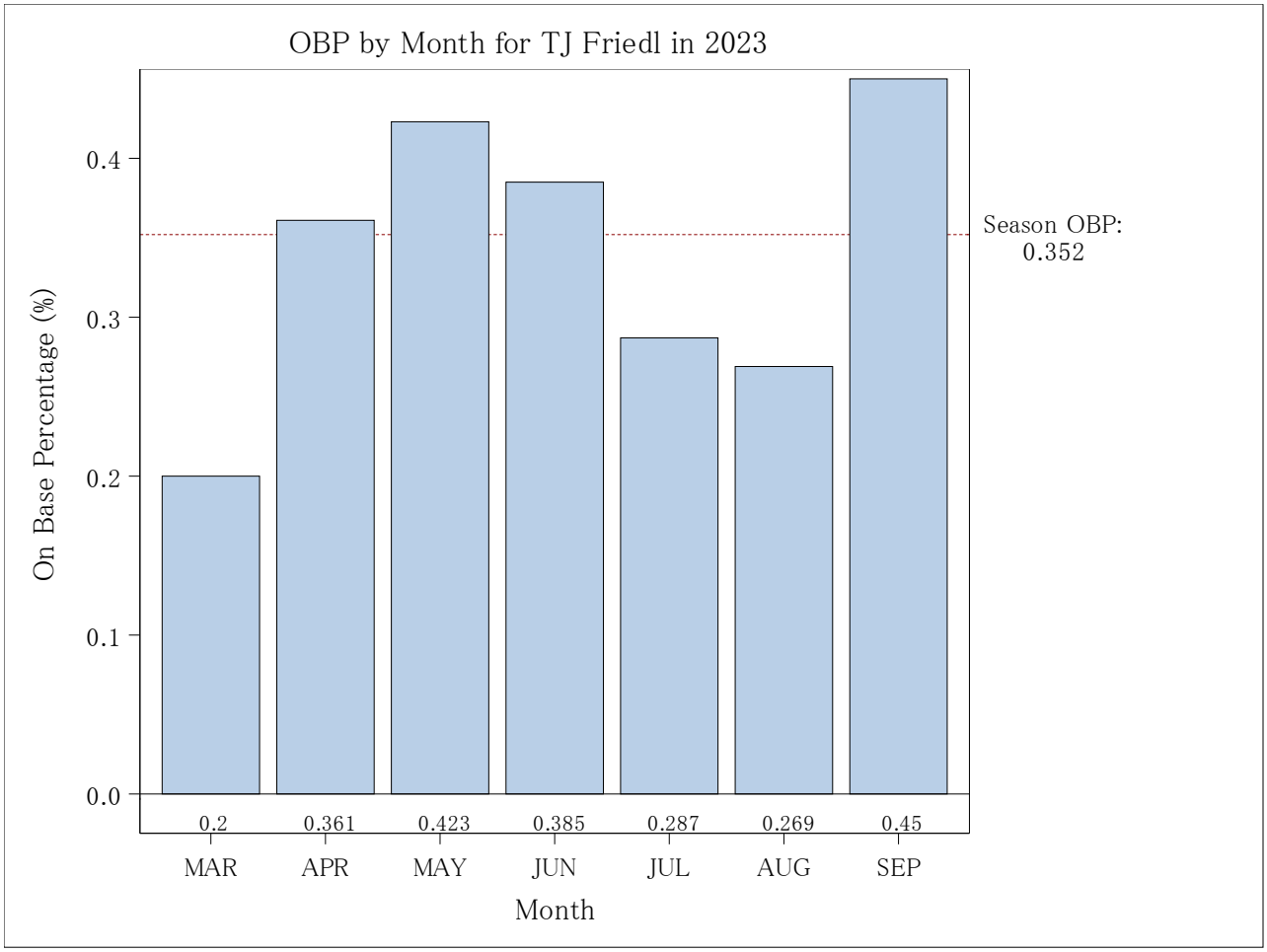
The next three sets of output are very similar in structure. They each consist of a table with three columns produced by PROC PRINT. The first column is a grouping variable, which could be a month or an opposing team as specified through the original assigned task. However, I have added additional functionality to my macro, which also outputs a breakdown of the OBP depending on whether the player is at home or away, so the grouping variable could also be the location of the game played. The second column will be the specific OBP for the subset or level of the grouping variable. The third column in the table will be the difference between the more specific OBP and the season long OBP to allow the reader to see where the player was over or underperforming.

After the table comes a graph which visualizes the information in the table. It is a bar chart produced by PROC SGPLOT, with one axis being for the grouping variable and the other for OBP. There is one category/bar per level in the grouping variable. At the base of each bar is a data label with the numerical value represented by the bar, giving the readers a bit more perspective information about the data in the chart. There is also a red dotted reference line plotted on the OBP axis at the location of the season long OBP. This gives readers a way to visually see how the batter performed when compared to their season long OBP. The reference line is also labeled with the numerical value of the season long OBP to allow comparisons with the data labels on each bar.

First for these sets of outputs is the OBP grouped by month. Note that the months appear in the normal, chronological order, as this may lead to insights such as hot or cold streaks when batting. It also gives the readers a visual representation of the batter's overall performance over time. The output is as follows:

OBP for TJ Friedl in 2023, Grouped by Month

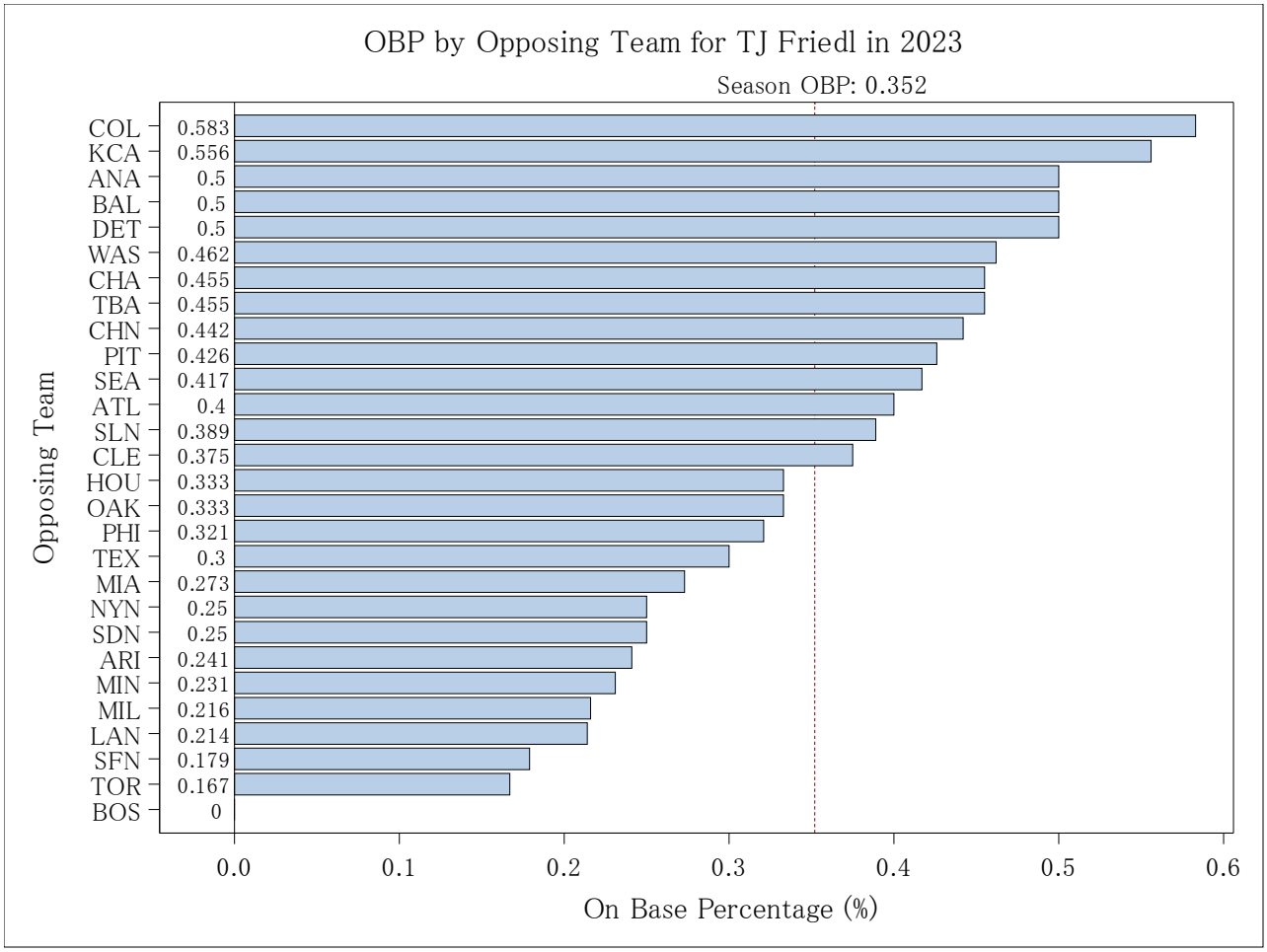
Month	Monthly OBP	Difference From Season OBP
MAR	0.200	−0.152
APR	0.361	0.009
MAY	0.423	0.071
JUN	0.385	0.033
JUL	0.287	−0.065
AUG	0.269	−0.083
SEP	0.450	0.098



Next is the output grouped by opposing team. This data is presented in the order of descending OBP, so the reader can easily see the teams against which the batter performed the best or the worst. The reader can also compare to the reference line to see how many teams against which the batter over or underperformed when compared to the whole season. The output is as follows:

OBP for TJ Friedl in 2023, Grouped by Opposing Team

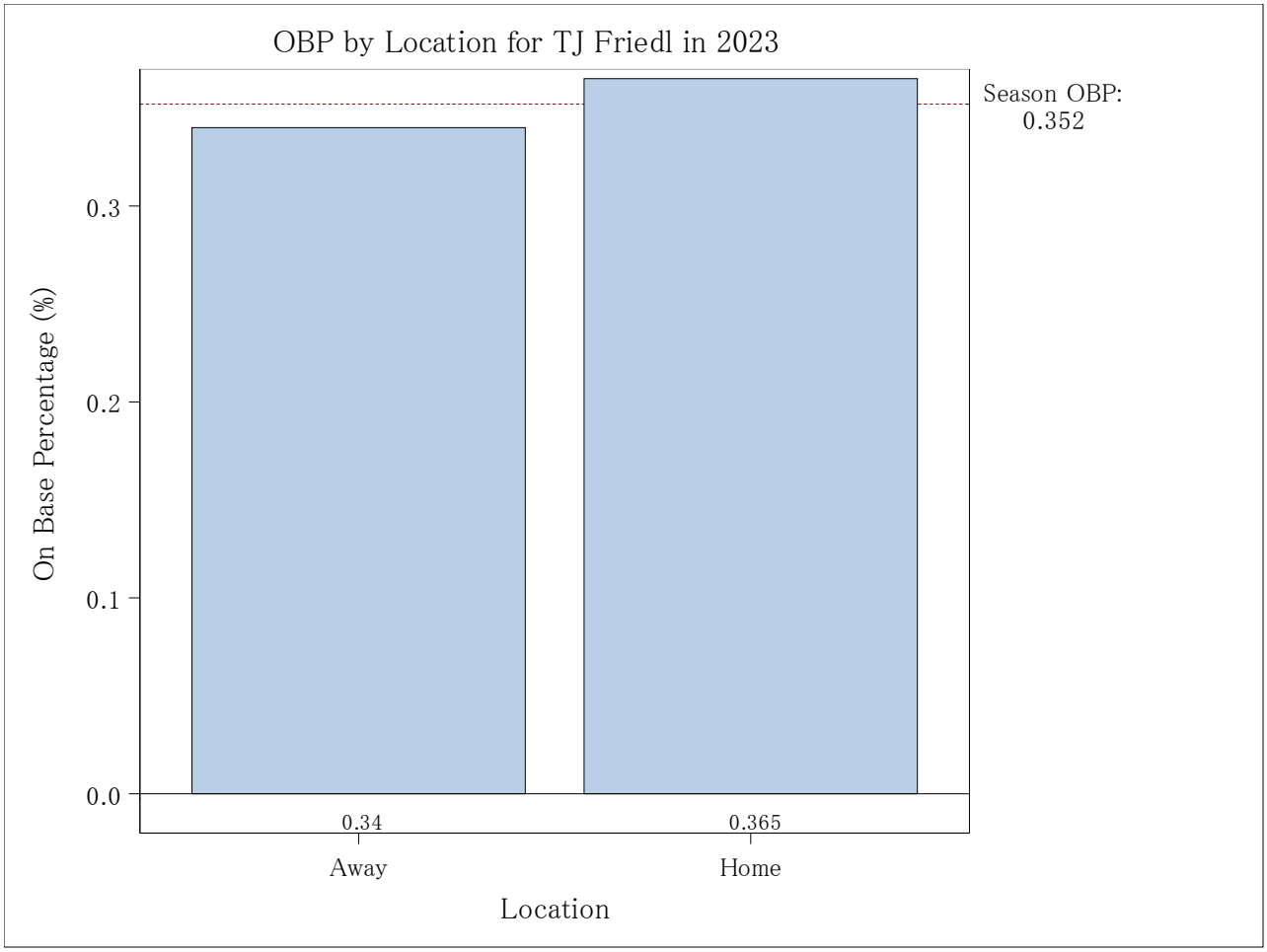
Opposing Team	OBP Against Team	Difference From Season OBP
COL	0.583	0.231
KCA	0.556	0.204
ANA	0.500	0.148
BAL	0.500	0.148
DET	0.500	0.148
WAS	0.462	0.110
CHA	0.455	0.103
TBA	0.455	0.103
CHN	0.442	0.090
PIT	0.426	0.074
SEA	0.417	0.065
ATL	0.400	0.048
SLN	0.389	0.037
CLE	0.375	0.023
HOU	0.333	-0.019
OAK	0.333	-0.019
PHI	0.321	-0.031
TEX	0.300	-0.052
MIA	0.273	-0.079
NYN	0.250	-0.102
SDN	0.250	-0.102
ARI	0.241	-0.111
MIN	0.231	-0.121
MIL	0.216	-0.136
LAN	0.214	-0.138
SFN	0.179	-0.173
TOR	0.167	-0.185
BOS	0.000	-0.352



Lastly, we have the OBP grouped by location. This was not required by the original assigned task, but I still thought it would be interesting to see if certain players performed better when playing at home vs playing away.

OBP for TJ Friedl in 2023, Grouped by Location

Location	Location OBP	Difference From Season OBP
Away	0.340	-0.012
Home	0.365	0.013



Final Documented SAS Code

```
/*  Please change the following %let folder= statement to match where you are
      storing the baseball play-by-play data
*/
%let folder=C:\Users\ryanj\OneDrive\Desktop\STA402_Final;

/*  On Base Percentage

This program reads the baseball play-by-play data for a particular year as
determined by the user, stores the data for a specific player (also designated
by the reader), and displays tabular and graphical displays breaking down
the player's On Base Percentage (OBP) by month and by opposing team. To read the
data, the program first iterates through all of the .ROS files to find the
provided player's unique ID. Then, it iterates through all of the .EVA and .EVN
files of the given year, usinh a flag system, checking the beginning of lines
for certain words or values to know what data to expect and store. After the
data files have been read, the program uses a series of PROC SORTs and PROC
MEANS to group the data and calculate the OBP for the full season, by month,
by opposing team, and by location. Lastly, the program uses PROC PRINT and PROC
SGPLOT to create the tabular and graphical outputs.

Since the program is implemented as a macro, you will need to provide macro
parameters to be able to run the program. Change the %let folder= statement
above if you have not already, and then invoke the macro as follows:

      %getobpforplayer(player=TJ Friedl, year=2023)
```

```
The possible values for the year macro parameter are 2008 to 2023, inclusive.

Additionally, the macro has an optional macro parameter rtf which establishes
the name and location of the output RTF file. By default, the RTF file is stored
in the same folder as the .zip files with the name obp.rtf. If the user wants
to specify a new file name or location, they can invoke the macro as follows:

      %getobpforplayer(player=TJ Friedl, year=2023, rtf=C:\path\filename.rtf)

The program makes the following assumptions:

      1. The player macro parameter consists of a player's full name, spelled
         correctly as in the official MLB listing of the rosters

      2. The year macro parameter is in the 4-digit form YYYY, and it a value
         between 2008 and 2023

      3. The zipped folders are all stored together in the folder designated
         above in the %let folder= statement. Each zip file is named according
         to the year, i.e. 2023eve.zip, and none of the internal files' names
         have been altered
*/

%macro getobpforplayer(player=, year=, rtf=&folder\obp.rtf);
/*************************** READ TEAM NAMES ****************************/
* use to read files directly from .zip file;
filename inzip ZIP "&folder\&year.eve.zip";

data _NULL_;
      * Read the TEAMYEAR file from the zip folder;
      infile inzip(TEAM&year);

      length line $35 team $3 league $1;
      input line;

      * pull team abbreviation and league from the line;
      nfiles+1;
      team = substr(line, 1, 3);
      league = substr(line, 5, 1);

      * save roster file and game file names as macro variables;
      call symputx("rosterfilename" || compress(nfiles), team || "&year" || ".ROS");
      call symputx("filename" || compress(nfiles),
                        "&year" || team || ".EV" || league);

      call symputx("nfiles", nfiles); * save this count for later;
run;

/*************************** GET PLAYER ID ******************************/
* read the text files and create data sets from them;
%do i=1 %to &nfiles;
      data _NULL_;
            infile inzip(&&rosterfilename&i) length=linelength;
            length line $120;
            input @1 line $varying120. linelength;

            /*  if name matches, save unique id in macro variable */
            if compare(scan(line, 3, ","), scan("&player", 1, " ")) = 0
            and compare(scan(line, 2, ","), scan("&player", 2, " ")) = 0
            then do;
                  call symputx("playerid", scan(line, 1, ","));
                  call symputx("team", substr("&&rosterfilename&i", 1, 3));
            end;
```

```
      run;
%end;

/********************** READ DATA FILES ********************************/
%do i=1 %to &nfiles;
      data DS&i;
            infile inzip(&&filename&i) length=linelength;
            length line $120 month $2 oppteam $3 outcome $14 event $50 loc $4;
            input @1 line $varying120. linelength;

            * declare and retain necessary variables;
            retain newgame 0;
            retain playlines 0;
            retain month "00";
            retain oppteam "XXX";
            retain loc "Away";


            if "&team" ^= substr("&&filename&i", 5, 3) then do;
                  oppteam = substr("&&filename&i", 5, 3);
                  loc = "Away";
            end;
            else loc = "Home";

            if substr(line, 1, 3) = "id," then do;
                  newgame = 1;
                  playlines = 0;
                  month = substr(line, 11, 2);
            end;
            /*  After seeing line with "id,", the next line we are looking for
                  the the info line with the visiting team */
            else if newgame then do;
                  if index(line,"info,visteam") then do;
                        if substr(line, 14, 3) ^= "&team"
                        then oppteam = substr(line, 14, 3);
                        newgame = 0;
                        playlines = 1;
                  end;
            end;
            /*  After seeing line with visiting team, the next lines we care about
                  are the play lines */
            else if playlines then do;
                  if index(line, "play,") and scan(line, 4, ",") = "&playerid" then do;
                        * if it is the player at bat, get the event;
                        event = scan(line, 7, ",");

                        /*  assign outcome of play based on event */
                        * Hit By Pitch - counts towards numerator;
                        if substr(event, 1, 2) = 'HP' then do;
                              outcome = 'HBP';
                              numerator = 1;
                              denominator = 1;
                        end;
                        * Sac fly - counts towards denominator;
                        else if index(event, '/SF') then do;
                              outcome = 'SF';
                              denominator = 1;
                        end;
                        * cases that start with characters with other meanings;
                        * i.e. wild pitch, no play, stolen base, etc.;
                        * the play lines still have unique id, but do not count to OBP;
                        * added check for empty string due to how first() works later;
                        else if index(event,'WP') or index(event,'NP')
```

```
                        or index(event,'DI') or index(event,'SB') or index(event,'SH')
                        or index(event,'CS') or index(event,'BK') or index(event,'PO')
                        or index(event,'PB') or index(event, 'OA') or event = " "
                        then do;
                              delete;
                        end;
                        * walk - counts for numerator and denominator;
                        else if first(event) = 'I' or first(event) = 'W' then do;
                              outcome = 'W';
                              numerator = 1;
                              denominator = 1;
                        end;
                        * reach by error - counts towards denominator;
                        else if first(event) = 'E' or index(event, 'FLE') then do;
                              outcome = 'E';
                              denominator = 1;
                        end;
                        * out - counts towards denominator;
                        * check if first character is a number;
                        else if compress(first(event),"123456789") = " " then do;
                              outcome = 'O';
                              denominator = 1;
                        end;
                        * strikeout - counts towards denominator;
                        else if first(event) = 'K' then do;
                              outcome = 'K';
                              denominator = 1;
                        end;
                        * Hit - counts towards both numerator and denominator;
                        else if first(event) = 'S' or first(event) = 'D'
                        or first(event) = 'T' or first(event) = 'H' then do;
                              outcome = 'H';
                              numerator = 1;
                              denominator = 1;
                        end;
                        * Fielders Choice - counts towards denominator;
                        else if index(event, 'FC') then do;
                              outcome = 'FC';
                              denominator = 1;
                        end;
                        * A catch all for cases that are not caught above;
                        else do;
                              outcome = 'Other';
                              result = event;
                        end;

                        keep month oppteam loc outcome numerator denominator result;

                        output;
                  end;
            end;
      run;
%end;

/********************** CREATE FORMAT FOR GRAPH DISPLAYS ******************/
proc format;
      /* create my own format called 'monthname' to convert between number
         and 3 character month abbrevaiations

         Note that we don't need to do this for the different teams since they
         are already stored in the datasets as 3 character abbreviations
      */
      value monthname
```

```
            1 = "JAN"
            2 = "FEB"
            3 = "MAR"
            4 = "APR"
            5 = "MAY"
            6 = "JUN"
            7 = "JUL"
            8 = "AUG"
            9 = "SEP"
            10 = "OCT"
            11 = "NOV"
            12 = "DEC"
            other = "Not a month";
run;

/************************** PROCESS DATA SETS ******************************/
* stack all datafiles into 1;
data allfiles;
      set DS:;
      month_num = input(month, 2.); * convert month to numeric variable;
      format month_num monthname.;  * apply custom format;
      drop month;
run;

* get obp for whole season;
proc means data=allfiles noprint;
      var numerator denominator;
      output out=allfiles_obp;
run;

* sort data by month;
proc sort data=allfiles;
      by month_num;
run;

* group numerator and denominator by month;
proc means data=allfiles N noprint;
      by month_num;
      var numerator denominator;
      output out=allfiles_month;
run;

* sort data by opposing team;
proc sort data=allfiles;
      by oppteam;
run;

* group numerator and denominator by opposing team;
proc means data=allfiles N noprint;
      by oppteam;
      var numerator denominator;
      output out=allfiles_oppteam;
run;

* sort data by location;
proc sort data=allfiles;
      by loc;
run;

* group numerator and denominator by location;
proc means data=allfiles N noprint;
      by loc;
      var numerator denominator;
```

```
      output out=allfiles_loc;
run;

* calculate season long OBP and round to 3 decimal places;
data allfiles_obp_final;
      set allfiles_obp(where=(_STAT_='N'));
      obp = round(numerator / denominator, .001);
      keep obp;
run;

* store season OBP in macro variable, to be used in plots;
data _NULL_;
      set allfiles_obp_final;
      call symputx("obpseason", obp);
run;

* calculate OBPs by month and round to 3 decimal places;
data allfiles_month_final;
      set allfiles_month(where=(_STAT_='N'));
      obp = round(numerator / denominator, .001);
      diff = obp - &obpseason;
      keep month_num obp diff;
run;

* calculate OBPs by opposing team and round to 3 decimal places;
data allfiles_oppteam_final;
      set allfiles_oppteam(where=(_STAT_='N'));
      obp = round(numerator / denominator, .001);
      diff = obp - &obpseason;
      keep oppteam obp diff;
run;

* calculate OBPs by location and round to 3 decimal places;
data allfiles_loc_final;
      set allfiles_loc(where=(_STAT_='N'));
      obp = round(numerator / denominator, .001);
      diff = obp - &obpseason;
      keep loc obp diff;
run;

/*************************** GENERATE OUTPUTS ****************************/

ods rtf bodytitle file="&rtf";

* show season long OBP;
title "Season Long OBP for &player in &year";
proc print data=allfiles_obp_final noobs label;
      label obp="OBP";
run;

* show OBPs by month;
title "OBP for &player in &year, Grouped by Month";
proc print data=allfiles_month_final noobs label;
      label obp="Monthly OBP" month_num="Month" diff="Difference From Season OBP";
run;

* show bar chart of OBP by month, overlaid with horizontal line for season OBP;
title "OBP by Month for &player in &year";
proc sgplot data=allfiles_month_final;
      refline &obpseason / axis=y lineattrs=(thickness=3 color=darkred pattern=dash)
                label=("Season OBP:(*ESC*){unicode '000a'x}      &obpseason");
                        * use unicode for new line inside label;
```

```
      * bar chart, category=month;
    vbar month_num / response=obp datalabel=obp datalabelpos=bottom;
    xaxis label="Month";
    yaxis label="On Base Percentage (%)";
run;

* sort data for print so that order matches plot;
proc sort data=allfiles_oppteam_final;
      by descending obp;
run;

* show OBPs by opposing team;
title "OBP for &player in &year, Grouped by Opposing Team";
proc print data=allfiles_oppteam_final noobs label;
      label obp="OBP Against Team" oppteam="Opposing Team"
              diff="Difference From Season OBP";
run;

/* show bar chart of OBP by opposing team, overlaid with vertical line for
   season OBP */
title "OBP by Opposing Team for &player in &year";
proc sgplot data=allfiles_oppteam_final;
      refline &obpseason / axis=x lineattrs=(thickness=3 color=darkred pattern=dash)
                  label=("Season OBP: &obpseason")
                           labelpos=min;
                           * use unicode for new line inside label;
    hbar oppteam / response=obp datalabel=obp categoryorder=respdesc
                        datalabelpos=left;  * bar chart, category=opposing team;
    xaxis label="On Base Percentage (%)";
      * type and fitpolicy options force SAS to show all group (team) labels;
    yaxis label="Opposing Team" type=discrete fitpolicy=none;
run;

* show OBPs by location;
title "OBP for &player in &year, Grouped by Location";
proc print data=allfiles_loc_final noobs label;
      label obp="Location OBP" loc="Location" diff="Difference From Season OBP";
run;

* show bar chart of OBP by location, overlaid with horizontal line for season OBP;
title "OBP by Location for &player in &year";
proc sgplot data=allfiles_loc_final;
      refline &obpseason / axis=y lineattrs=(thickness=3 color=darkred pattern=dash)
                label=("Season OBP:(*ESC*){unicode '000a'x}      &obpseason");
                        * use unicode for new line inside label;

    * bar chart, category=month;
    vbar loc / response=obp datalabel=obp datalabelpos=bottom;
    xaxis label="Location";
    yaxis label="On Base Percentage (%)";
run;

ods rtf close;

%mend getobpforplayer;
```