MATH 3190 Homework 1

Focus: Notes 1-3

Due February 10, 2024

Now its time to practice what we have learned in class and learn even more! Note that your homework should be completed in R Markdown or Quarto (you can just add your answers to this document in the appropriate part) and Knitted to an html document or pdf document. You will 'turn in' this homework by uploading to your GitHub Math_3190_Assignment repository in the Homework/Homework_1 directory.

Problem 1 (25 points)

Part a (20 points)

Write two functions called `ghist` and `gbox` that are similar to my `ggraph` function that you put in your myplots.in package from Lab 2. Remember that the "in" should be replaced with your initials. The `ghist` function should create a ggplot histogram of a variable that is given as a vector. The `gbox` function should create a ggplot box plot when a single numeric vector is given or it should create side-by-side box plots if one numeric and one categorical variables are given. Allow the user to indicate whether it should be horizontal or vertical box plots. Be sure to properly document these functions.

Part b (3 points)

Add those functions to your myplots.in package. Then run the `devtools::document()` function, update the DESCRIPTION file, and install your package to verify those functions work.

Part c (2 points)

Update your GitHub myplots.in repo with the updated package. This is only worth 2 points, but I cannot verify you did part a without this, so it is actually worth much more.

Problem 2 (60 points)

Part a (9 points)

Learn about the `read.fwf()` function for use in downloading data from a URL into **R**. Learn about tools for downloading files from external servers. The `widths` and `strip.white` options will be especially useful here. Use this function to download the scores for all college basketball games for the 2023-2024 season (http://kenpom.com/cbbga24.txt) and then convert it to a tibble (load the `tidyverse` package first). The second team listed per line is the home team. It is not clear what the numbers, letters, or city names indicate after the second listed score. Notice that this is a "live" file that gets updated every day! So, your tibble size may change if you work on this assignment over the course of several days. That's fine. Give the code you used to download these data.

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.4.4     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(dplyr)

data24 <- read.fwf( "http://kenpom.com/cbbga24.txt",
                    widths = c(10, 24, 3, 24, 3, 2, 20),
                    strip.white=TRUE) |>
  as_tibble()

head(data24)
```

```
## # A tibble: 6 x 7
##   V1         V2              V3 V4             V5 V6    V7
##   <chr>      <chr>        <int> <chr>       <int> <chr> <chr>
## 1 11/06/2023 Portland St.    62 Air Force      55 ""    ""
## 2 11/06/2023 Morehead St.    73 Alabama       105 ""    ""
## 3 11/06/2023 Morgan St.      59 Arizona       122 ""    ""
## 4 11/06/2023 Alcorn St.      59 Arkansas       93 ""    ""
## 5 11/06/2023 Marist          71 Army           55 ""    ""
## 6 11/06/2023 Life            72 Austin Peay    90 ""    ""
```

Now lets practice using our tidy data/tidyverse tools! Using your `cbbga24` tibble, try doing the following:

Part b (2 points)

Use `rename()` to rename all of your variables to names that make sense.

```
cbbga24 <-  data24 |> rename( 'Date' = V1,
  'Team_1' = V2,
  'Score_1' = V3,
  'Team_2' = V4,
  'Score_2' = V5,
  'Site_Type' = V6,
  'Game_Site' = V7
  )

head(cbbga24)
```

```
## # A tibble: 6 x 7
##   Date       Team_1       Score_1 Team_2      Score_2 Site_Type Game_Site
```

```
##   <chr>      <chr>          <int> <chr>         <int> <chr>     <chr>
## 1 11/06/2023 Portland St.      62 Air Force        55 ""        ""
## 2 11/06/2023 Morehead St.      73 Alabama         105 ""        ""
## 3 11/06/2023 Morgan St.        59 Arizona         122 ""        ""
## 4 11/06/2023 Alcorn St.        59 Arkansas         93 ""        ""
## 5 11/06/2023 Marist            71 Army             55 ""        ""
## 6 11/06/2023 Life              72 Austin Peay      90 ""        ""
```

Part c (2 points)

Use `mutate()` to create a new column that gives the score differences (team1−team2).

```
cbbga24 <- cbbga24 |>
  mutate(Margin = Score_1-Score_2 )

head(cbbga24)
```

```
## # A tibble: 6 x 8
##   Date       Team_1       Score_1 Team_2      Score_2 Site_Type Game_Site Margin
##   <chr>      <chr>          <int> <chr>         <int> <chr>     <chr>      <int>
## 1 11/06/2023 Portland St.      62 Air Force        55 ""        ""             7
## 2 11/06/2023 Morehead St.      73 Alabama         105 ""        ""           -32
## 3 11/06/2023 Morgan St.        59 Arizona         122 ""        ""           -63
## 4 11/06/2023 Alcorn St.        59 Arkansas         93 ""        ""           -34
## 5 11/06/2023 Marist            71 Army             55 ""        ""            16
## 6 11/06/2023 Life              72 Austin Peay      90 ""        ""           -18
```

Part d (2 points)

Use `arrange()` to sort the data set by the home team.

```
cbbga24 <- cbbga24 |> arrange(Team_1)

head(cbbga24)
```

```
## # A tibble: 6 x 8
##   Date       Team_1            Score_1 Team_2 Score_2 Site_Type Game_Site Margin
##   <chr>      <chr>               <int> <chr>    <int> <chr>     <chr>      <int>
## 1 11/06/2023 Abilene Christian      64 Oklah~      59 ""        ""             5
## 2 11/10/2023 Abilene Christian      64 N.C. ~      84 ""        ""           -20
## 3 11/17/2023 Abilene Christian      77 San J~      71 "N"       "St. Tho~      6
## 4 11/20/2023 Abilene Christian      69 Misso~      87 "N"       "St. Tho~    -18
## 5 11/29/2023 Abilene Christian      71 UT Ar~      86 ""        ""           -15
## 6 12/21/2023 Abilene Christian      73 Arkan~      83 ""        ""           -10
```

Response: The Home Team is typically listed first in column order for sports data sets. However when there is a 'n' neutral site type the teams would then be ordered alphabetically. Yet this does not seem to be the case for this data set.
The `Site_Type` column is mostly empty with what appear to be irregular or unusually coded site types, e.g. '2N'. Some entries appear to have some correspondance with a game site state-wise, but are in fact coded with an 'n', e.g. the 'away-team' is Idaho St. and game site is Idaho Falls, ID, but is typed 'n'. This data is treated simply by ordering the first column. The `Site_Type` data would need to be updated and reviewed for any use in ordering.

Part e (2 points)

Use `select()` to remove the extra variable(s) that had that irrelevant information at the end of each line. Note: you can select every variable except one by using the "!".

```
cbbga24 <- cbbga24 |> select(!c(Site_Type, Game_Site))
head(cbbga24)
```

```
## # A tibble: 6 x 6
##   Date       Team_1            Score_1 Team_2       Score_2 Margin
##   <chr>      <chr>               <int> <chr>          <int>  <int>
## 1 11/06/2023 Abilene Christian      64 Oklahoma St.      59      5
## 2 11/10/2023 Abilene Christian      64 N.C. State        84    -20
## 3 11/17/2023 Abilene Christian      77 San Jose St.      71      6
## 4 11/20/2023 Abilene Christian      69 Missouri St.      87    -18
## 5 11/29/2023 Abilene Christian      71 UT Arlington      86    -15
## 6 12/21/2023 Abilene Christian      73 Arkansas          83    -10
```

Part f (2 points)

Put parts a-e all together in one piping expression (with 5 pipes) and save this as a new object in **R**.

```
cbbga24 <-  read.fwf( "http://kenpom.com/cbbga24.txt",
                      widths = c(10, 24, 3, 24, 3, 2, 20),
                      strip.white=TRUE) |>
  as_tibble() |>
  rename( 'Date' = V1,
  'Team_1' = V2,
  'Score_1' = V3,
  'Team_2' = V4,
  'Score_2' = V5,
  'Site_Type' = V6,
  'Game_Site' = V7) |>
  mutate(Margin = Score_1-Score_2 ) |>
  arrange(Team_1) |>
  select(!c(Site_Type, Game_Site))

head(cbbga24)
```

```
## # A tibble: 6 x 6
##   Date       Team_1            Score_1 Team_2       Score_2 Margin
##   <chr>      <chr>               <int> <chr>          <int>  <int>
## 1 11/06/2023 Abilene Christian      64 Oklahoma St.      59      5
## 2 11/10/2023 Abilene Christian      64 N.C. State        84    -20
## 3 11/17/2023 Abilene Christian      77 San Jose St.      71      6
## 4 11/20/2023 Abilene Christian      69 Missouri St.      87    -18
## 5 11/29/2023 Abilene Christian      71 UT Arlington      86    -15
## 6 12/21/2023 Abilene Christian      73 Arkansas          83    -10
```

Part g (3 points)

Use `filter()` to reduce the data down to only games played in 2023 (you could use the `lubridate` package for this, since it specializes in dealing with dates, but some base **R** packages will also work). Save this in a

new tibble. **We will use this tibble with only the 2023 years from here on out**.

```
library(lubridate)
cbbga23 <- cbbga24 |>
  mutate(Date = mdy(Date)) |>
  #Run only once, a second run will not be able to parse the transformed dates again
  filter(between(Date, mdy("01/01/2023"), mdy("12/31/2023")))

head(cbbga23)
```

```
## # A tibble: 6 x 6
##   Date       Team_1            Score_1 Team_2       Score_2 Margin
##   <date>     <chr>               <int> <chr>          <int>  <int>
## 1 2023-11-06 Abilene Christian      64 Oklahoma St.      59      5
## 2 2023-11-10 Abilene Christian      64 N.C. State        84    -20
## 3 2023-11-17 Abilene Christian      77 San Jose St.      71      6
## 4 2023-11-20 Abilene Christian      69 Missouri St.      87    -18
## 5 2023-11-29 Abilene Christian      71 UT Arlington      86    -15
## 6 2023-12-21 Abilene Christian      73 Arkansas          83    -10
```

Part h (4 points)

Write a function that will filter the tibble to only games played by a given team. Demonstrate your function by displaying games played by SUU.

```
byTeamGames <- function(data, team, ungroup){
  newTibble <- data |>
    rowwise() |>
    filter(any(c(Team_1, Team_2) == team)) |>
    ungroup() #if there is some other (non-lubridate parsed) formatting
  return(newTibble)
}

suu23 = byTeamGames(cbbga23, team = 'Southern Utah', ungroup=TRUE)

suu23
```

```
## # A tibble: 13 x 6
##    Date       Team_1           Score_1 Team_2              Score_2 Margin
##    <date>     <chr>              <int> <chr>                 <int>  <int>
##  1 2023-12-30 Antelope Valley       78 Southern Utah            95    -17
##  2 2023-11-29 Cal Baptist           91 Southern Utah            66     25
##  3 2023-12-09 Idaho St.             74 Southern Utah            82     -8
##  4 2023-11-09 Life Pacific          73 Southern Utah           108    -35
##  5 2023-12-22 Middle Tennessee      63 Southern Utah            69     -6
##  6 2023-11-06 Southern Utah         72 Cal St. Bakersfield      73     -1
##  7 2023-11-14 Southern Utah         84 Utah St.                 93     -9
##  8 2023-11-21 Southern Utah         53 Louisiana Tech           67    -14
##  9 2023-11-22 Southern Utah         74 Texas St.                67      7
## 10 2023-12-02 Southern Utah         63 Seattle                  73    -10
## 11 2023-12-05 Southern Utah         86 Utah                     88     -2
## 12 2023-12-16 Southern Utah         74 Northern Arizona         76     -2
## 13 2023-12-19 Southern Utah         88 Montana St.              89     -1
```

Part i (7 points)

Use `summarize()` to extract SUU's win/loss record and winning percentage for their 2023 games. Hint: using the `case_when()` function inside of a `mutate()` function to create a new variable that indicates whether SUU won or lost is helpful.

```
#adding new win/loss columns with strings and dummy-coding
suu23 <- suu23 |>
mutate(Result = case_when(
    Team_1 == 'Southern Utah' & Margin > 0 ~ 1,
    Team_1 == 'Southern Utah' & Margin < 0 ~ 0,
    Team_2 == 'Southern Utah' & Margin < 0 ~ 1,
    Team_2 == 'Southern Utah' & Margin > 0 ~ 0)) |>
  mutate(WinLoss = case_when(
    Result == 1 ~ 'Win',
    Result == 0 ~ 'Loss'))

#creating summary tibble
suu23perform <- suu23 |> filter(Team_1 == 'Southern Utah'| Team_2 == 'Southern Utah') |>
  summarize(
  Wins = sum(WinLoss == 'Win'),
  Losses = sum(WinLoss == 'Loss'),
  Percent_Win = round(Wins/(Wins+Losses)*100,2) |> paste0("%")
)

suu23perform
```

```
## # A tibble: 1 x 3
##    Wins Losses Percent_Win
##   <int>  <int> <chr>
## 1     5      8 38.46%
```

Part j (7 points)

Generalize this by writing a function that will do this for ***a given*** team.

Create a tibble with this information for ***all*** teams. (???)

Arrange this tibble by winning percentage (descending). The `add_row()` function may be useful here.

```
team_Record <- function(data, team) {

 temp_Tibble <- data |>
  mutate(Result = case_when(
    Team_1 == team & Margin > 0 ~ 1,
    Team_1 == team & Margin < 0 ~ 0,
    Team_2 == team & Margin < 0 ~ 1,
    Team_2 == team & Margin > 0 ~ 0)) |>
  mutate(WinLoss = case_when(
    Result == 1 ~ 'Win',
    Result == 0 ~ 'Loss'))

   #creating summary tibble
result_Tibble <-  temp_Tibble |>
```

```
  filter(Team_1 == team | Team_2 == team) |>
  summarize(
    Team = team,
    Wins = sum(WinLoss == 'Win'),
    Losses = sum(WinLoss == 'Loss'),
    Percent_Win = round(Wins/(Wins+Losses)*100,2) %>% paste0("%"))

  return(result_Tibble)
}
```

```
#Testing function team_Record with Oklahoma State
OKState_Rec = team_Record(cbbga23, 'Oklahoma St.')
OKState_Rec
```

```
## # A tibble: 1 x 4
##   Team          Wins Losses Percent_Win
##   <chr>        <int>  <int> <chr>
## 1 Oklahoma St.     7      5 58.33%
```

```
#seems to work, especially if you don't try Abilene Christian, with the same record as SUU
```

```
#and now for a totally different function that does something else to make that all team summary tibble

all_team_Record <- function(data) {


  unique_teams <- unique(c(data$Team_1, data$Team_2)) # vector of unique teams
  result_Tibble <- tibble() # tibble to store results

  for (team in unique_teams) {
    new_row <- team_Record(data, team)
    result_Tibble <- rbind(result_Tibble, new_row) #add_row() doesn't work here, IDKW
  }

  return(result_Tibble)
}

#now run nested-function function to obtain result record for all states in 2024

allTeamsResults <- all_team_Record(cbbga24) |>
  arrange(desc(Percent_Win))#for 2024

allTeamsResults
```

```
## # A tibble: 719 x 4
##    Team            Wins Losses Percent_Win
##    <chr>          <int>  <int> <chr>
##  1 Connecticut       22      2 91.67%
##  2 Grand Canyon      22      2 91.67%
##  3 Purdue            22      2 91.67%
##  4 Coppin St.         2     19 9.52%
##  5 Indiana St.       22      3 88%
```

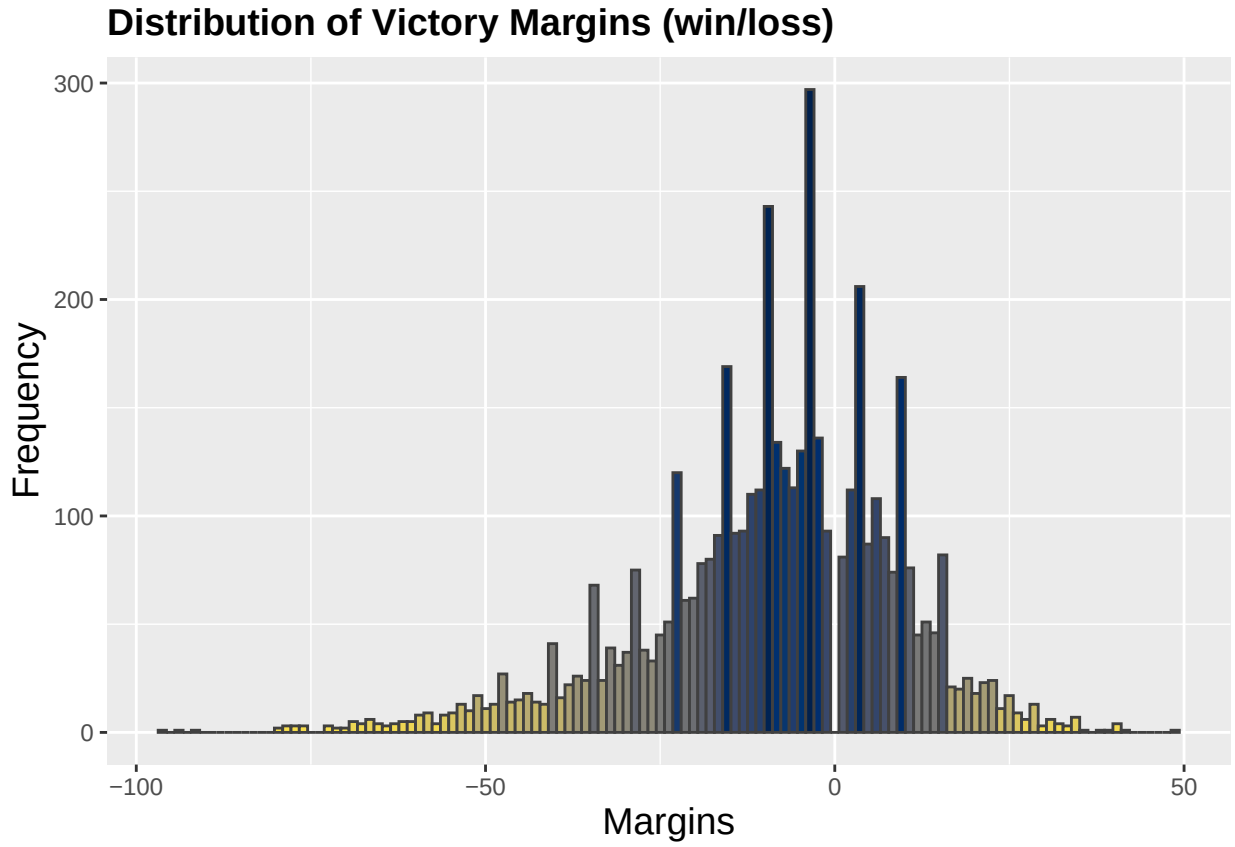
```
##  6 James Madison     22      3 88%
##  7 Samford           22      3 88%
##  8 Houston           21      3 87.5%
##  9 McNeese St.       21      3 87.5%
## 10 South Carolina    21      3 87.5%
## # i 709 more rows
```

Response: Above is the tibble summarizing the results for the *2024* data. This would also work for the 2023 data, but this was not specified in the question. Unfortunately the `add_row()` function was not working for my particular code setup, although my reading of the description suggested it was more adapted to more uses of a tibble. The `rbind()` is used instead.

Part k (8 points)

Write two functions that generate appropriate graphs for the basketball data. These two graphs could be anything you'd like and should use `ggplot2` and they should show something meaningful.

```
## Loading required package: viridisLite
```

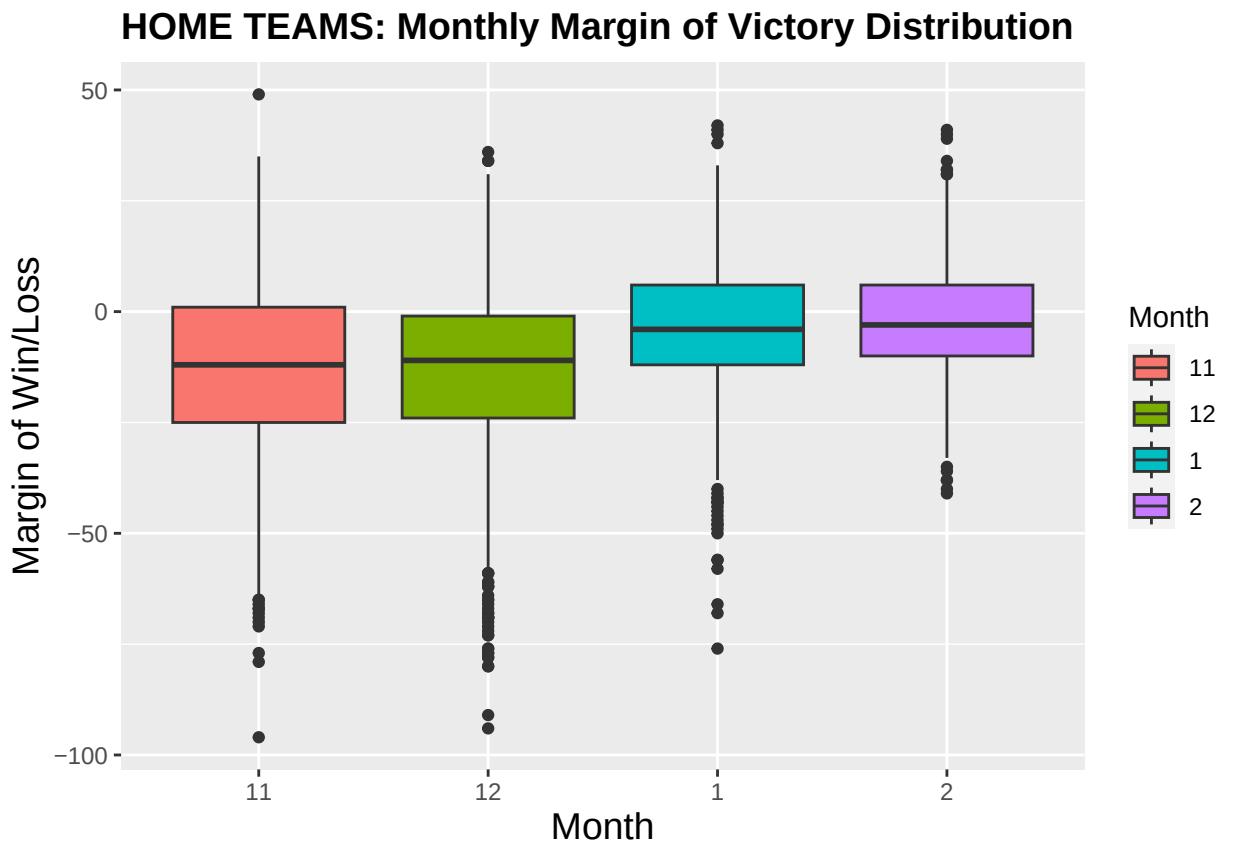



```
cbbga24Month <- cbbga24 |>
  mutate(Date = mdy(Date)) |>
  mutate(Month = month(Date)) |>   #only run once, thank you...
  mutate(Month = factor(Month, levels = c(11, 12, 1, 2)))

cbbga24Month
```

```
## # A tibble: 4,607 x 7
##    Date       Team_1            Score_1 Team_2           Score_2 Margin Month
##    <date>     <chr>               <int> <chr>              <int>  <int> <fct>
##  1 2023-11-06 Abilene Christian      64 Oklahoma St.          59      5 11
##  2 2023-11-10 Abilene Christian      64 N.C. State            84    -20 11
##  3 2023-11-17 Abilene Christian      77 San Jose St.          71      6 11
##  4 2023-11-20 Abilene Christian      69 Missouri St.          87    -18 11
##  5 2023-11-29 Abilene Christian      71 UT Arlington          86    -15 11
##  6 2023-12-21 Abilene Christian      73 Arkansas              83    -10 12
##  7 2023-12-30 Abilene Christian      84 Western Kentucky      86     -2 12
##  8 2024-01-11 Abilene Christian      64 Grand Canyon          74    -10 1
##  9 2024-01-13 Abilene Christian      53 Cal Baptist           68    -15 1
## 10 2024-01-18 Abilene Christian      71 Tarleton St.          79     -8 1
## # i 4,597 more rows
```

After preparing data for dates (only run once), the boxplot:

```
#boxplot of monthly margin distributions
marginMonthly <- ggplot(cbbga24Month, aes(x = Month, y = Margin, fill = Month)) +
  geom_boxplot() +
  labs(title = "HOME TEAMS: Monthly Margin of Victory Distribution",
       x = "Month",
       y = "Margin of Win/Loss") +
  theme(axis.title=element_text(size = 14),
        plot.title =element_text(size = 14, face = "bold"))

marginMonthly
```



Part l (12 points)

Create an **R** package that contains your functions from Parts h, j, and k and your tibble that contains all the games from 2023. You can use the the `write_csv()` function to save your tibble as a .csv file and put it in a `data-raw` folder in your package. Make sure the functions are properly documented. Upload this package to your GitHub page and indicate here what you called this package.

Response: This package is called `collegeBasketball.bs` in the form convention of this class. It should be found in my Math_3190_Assignments Git repository under Homework/Homework_1.

Problem 3 (15 points)

Repeat parts b-e and g of Problem 2 using Python in R Markdown (or Quarto). First, pass the original object that you read in from the website to Python without any changes to it (you do not need to read the file from the web in Python, but you can if you'd like) and then use `pandas` to rename the columns as indicated in part b, add the columns specified in part c, arrange the data as in part d, drop the "garbage" column as in part e, and filter it down as in part g. The `pandas` functions `rename`, `assign` (instead of `mutate`), `drop` (instead of `select`) and `str.contains` (used to select the right rows) will be useful here. Be sure to follow the guide in Notes 2 to properly install Python, install the `pandas` library and to load it in **R**.

```
import numpy as np
import pandas as pd

pybbData24 = r.data24
```

```
pybbData24.head()
```

```
##            V1            V2  V3         V4   V5 V6 V7
## 0  11/06/2023  Portland St.  62  Air Force   55
## 1  11/06/2023  Morehead St.  73    Alabama  105
## 2  11/06/2023    Morgan St.  59    Arizona  122
## 3  11/06/2023    Alcorn St.  59   Arkansas   93
## 4  11/06/2023        Marist  71       Army   55
```

```
pybbData24.columns = ['date_M/D/Y', 'home_team', 'home_score', 'visit_team', 'visit_score', 'site_type'

print(pybbData24.columns)
```

```
## Index(['date_M/D/Y', 'home_team', 'home_score', 'visit_team', 'visit_score',
##        'site_type', 'game_site'],
##       dtype='object')
```

```
pybbData24.describe()
```

```
##         home_score  visit_score
## count  4607.000000  4607.000000
## mean     68.555893    77.687866
## std      12.233821    13.861789
## min      14.000000    39.000000
## 25%      60.000000    68.000000
## 50%      69.000000    77.000000
## 75%      77.000000    86.000000
## max     119.000000   146.000000
```

```
pybbData24['margin'] = pybbData24['home_score'] - pybbData24['visit_score']
#this is in-place modification of the dataframe, whereas with .assign() the data could be preserved (if

pybbData24.head()
```

```
##    date_M/D/Y     home_team  home_score  ... site_type  game_site margin
## 0  11/06/2023  Portland St.          62  ...                          7
## 1  11/06/2023  Morehead St.          73  ...                        -32
## 2  11/06/2023    Morgan St.          59  ...                        -63
## 3  11/06/2023    Alcorn St.          59  ...                        -34
## 4  11/06/2023        Marist          71  ...                         16
##
## [5 rows x 8 columns]
```

```
pybbData24.sort_values(by = 'home_team', ascending = True, inplace =True)

pybbData24
```

```
##       date_M/D/Y          home_team  ...       game_site margin
## 4593  02/10/2024  Abilene Christian  ...                     -29
```

```
## 2606  12/30/2023  Abilene Christian  ...                    -2
## 3426  01/18/2024  Abilene Christian  ...                    -8
## 721   11/17/2023  Abilene Christian  ...  St. Thomas, VI     6
## 3065  01/11/2024  Abilene Christian  ...                   -10
## ...          ...                ...  ...             ...   ...
## 3732  01/25/2024     Youngstown St.  ...                    28
## 4277  02/04/2024     Youngstown St.  ...                    -4
## 67    11/06/2023     Youngstown St.  ...                   -10
## 1073  11/24/2023     Youngstown St.  ...                    -8
## 4103  02/01/2024     Youngstown St.  ...                    11
##
## [4607 rows x 8 columns]
```

```
pybbData24 = pybbData24.drop(columns=['game_site', 'site_type'])

pybbData24
```

```
##       date_M/D/Y          home_team  ...  visit_score margin
## 4593  02/10/2024  Abilene Christian  ...           74    -29
## 2606  12/30/2023  Abilene Christian  ...           86     -2
## 3426  01/18/2024  Abilene Christian  ...           79     -8
## 721   11/17/2023  Abilene Christian  ...           71      6
## 3065  01/11/2024  Abilene Christian  ...           74    -10
## ...          ...                ...  ...          ...   ...
## 3732  01/25/2024     Youngstown St.  ...           50     28
## 4277  02/04/2024     Youngstown St.  ...           82     -4
## 67    11/06/2023     Youngstown St.  ...           72    -10
## 1073  11/24/2023     Youngstown St.  ...           77     -8
## 4103  02/01/2024     Youngstown St.  ...           77     11
##
## [4607 rows x 6 columns]
```

```
# i guess making a function to do this as required in problem 2g:

def team_filter(data, team):
    """Filters a DataFrame based on a specified team name.
    Args:
        data: A pandas DataFrame containing team data.
        team: The team name to filter by.
    Returns:
        A new DataFrame containing rows where either the home_team or visit_team column matches the giv
    """
    teamdf = data[(data["home_team"] == team) | (data["visit_team"] == team)]
    return teamdf

pySUUdata24 = team_filter(pybbData24, 'Southern Utah')

pd.set_option('display.max_columns', None)
print(pySUUdata24.head(11))
```

```
##       date_M/D/Y         home_team  home_score           visit_team  \
## 2586  12/30/2023   Antelope Valley          78        Southern Utah
## 1365  11/29/2023       Cal Baptist          91        Southern Utah
```

```
## 2764  01/04/2024      Grand Canyon          96        Southern Utah
## 1833  12/09/2023         Idaho St.          74        Southern Utah
## 276   11/09/2023      Life Pacific          73        Southern Utah
## 2385  12/22/2023  Middle Tennessee          63        Southern Utah
## 4491  02/10/2024     Southern Utah          65         Grand Canyon
## 1524  12/02/2023     Southern Utah          63              Seattle
## 2924  01/06/2024     Southern Utah          62          Utah Valley
## 578   11/14/2023     Southern Utah          84             Utah St.
## 16    11/06/2023     Southern Utah          72  Cal St. Bakersfield
##
##       visit_score  margin
## 2586           95     -17
## 1365           66      25
## 2764           75      21
## 1833           82      -8
## 276           108     -35
## 2385           69      -6
## 4491           94     -29
## 1524           73     -10
## 2924           80     -18
## 578            93      -9
## 16             73      -1
```