MSCS 264: Homework #3

The dataset we will use is called `bball`. You can acquire it by runnining the chunk of code below. Here, we use a techinique called webscraping. If you go to the URL, you can see that table of all the NBA players stats in 2022. The code below "scrapes" that data from the website and tidies it in R! By the end of the semester, you'll know how to do this yourself! (Just to get you started thinking about ALL the possibilities of where data can come from!)

1. Let's find the top scoring centers. To do this, follow the steps below. In each step, you will create a new dataset. Retype the name of the dataset below to print it out!

```
colnames(bball)
```

```
##  [1] "Rk"                "Player"            "Pos"
##  [4] "Age"               "Tm"                "G"
##  [7] "GS"                "MP"                "FG"
## [10] "FGA"               "FGpct"             "3P"
## [13] "3PA"               "3P%"               "2P"
## [16] "2PA"               "2P%"               "eFG%"
## [19] "FT"                "FTA"               "FTpct"
## [22] "ORB"               "DRB"               "TRB"
## [25] "AST"               "STL"               "BLK"
## [28] "TOV"               "PF"                "PTS"
## [31] "points_per_minute"
```

```
bball$Pos
```

```
##   [1] "C"  "C"  "C"  "PF" "C"  "SG" "SG" "SG" "SG" "C"  "PG" "SF" "SF" "SF" "PF"
##  [16] "PF" "SF" "PF" "PG" "SF" "PG" "SF" "PG" "PG" "PG" "SF" "SG" "C"  "C"  "PF"
##  [31] "PF" "PF" "PG" "PG" "C"  "SG" "PG" "PG" "PF" "PF" "SF" "SG" "C"  "SF" "PF"
##  [46] "SF" "PF" "SG" "SG" "SF" "SF" "SF" "PF" "PF" "PF" "PG" "SF" "C"  "C"  "C"
##  [61] "C"  "PG" "SF" "SG" "PF" "PF" "SF" "SF" "SG" "SG" "SF" "SG" "SG" "C"  "SF"
##  [76] "SF" "PF" "SF" "PG" "SG" "SG" "PG" "SF" "SF" "SG" "SG" "SG" "SG" "SF" "SG"
##  [91] "SF" "SF" "C"  "C"  "C"  "SG" "SF" "SG" "C"  "SF" "PG" "SG" "SG" "SF" "PF"
## [106] "SG" "PG" "PF" "PG" "C"  "C"  "C"  "C"  "PG" "PG" "PG" "PF" "SG" "C"  "C"
## [121] "SF" "PG" "PF" "SG" "PF" "PF" "SG" "C"  "SG" "PF" "C"  "PG" "PG" "SF" "PF"
## [136] "PG" "PF" "C"  "C"  "C"  "PF" "PF" "PF" "SF" "SF" "SF" "PF" "SG" "SG" "SG"
## [151] "SG" "SG" "SG" "PG" "C"  "C"  "SG" "PF" "C"  "PF" "PF" "C"  "SG" "C"  "PG"
## [166] "PG" "SG" "SG" "SG" "SG" "PG" "SG" "SG" "SG" "PG" "PF" "PG" "PG" "PG" "SG"
## [181] "PG" "PG" "PG" "C"  "C"  "C"  "SG" "PG" "PG" "PF" "SG" "SG" "SF" "SG" "SF"
## [196] "SG" "C"  "SF" "SF" "SF" "SF" "C"  "C"  "C"  "C"  "C"  "C"  "C"  "C"  "PF"
## [211] "PF" "PF" "PF" "PG" "SG" "SG" "SG" "SF" "PG" "SG" "PG" "SG" "PG" "PG" "C"
## [226] "PG" "PF" "PF" "PF" "C"  "PF" "SG" "SG" "SG" "PG" "SG" "PF" "C"  "PF" "PF"
## [241] "C"  "SF" "PG" "PF" "PF" "C"  "PG" "PF" "SG" "PG" "PF" "PG" "SF" "PF" "SG"
## [256] "PF" "SF" "PF" "SG" "C"  "SG" "SG" "PF" "SG" "SG" "PG" "SG" "SG" "PG" "PG"
## [271] "PG" "SF" "PG" "C"  "C"  "C"  "SG" "SF" "PF" "SG" "SG" "SG" "SG" "C"  "C"
## [286] "SF" "C"  "PG" "SF" "SF" "PF" "PF" "PF" "PF" "C"  "SG" "SG" "SG" "SG" "SF"
## [301] "PG" "SF" "SG" "SF" "SF" "SF" "PG" "PG" "PG" "PG" "SF" "SF" "SG" "C"  "PF"
## [316] "PF" "SF" "SG" "C"  "SG" "SF" "SF" "SF" "C"  "SG" "SF" "C"  "SG" "SF" "SG"
## [331] "SF" "SF" "SG" "C"  "C"  "C"  "SF" "SF" "SF" "PG" "SF" "PG" "C"  "PF" "SF"
## [346] "SF" "SF" "SF" "SF" "SG" "PF" "SG" "SG" "PF" "PF" "PF" "PF" "SF" "PF" "PF"
```

```
## [361] "PF" "SF" "SG" "SG" "SG" "PF" "PG" "PG" "PG" "C"  "PG" "PG" "C"  "PF" "PF"
## [376] "SF" "C"  "SG" "PG" "PG" "C"  "C"  "C"  "PG" "SF" "SF" "SF" "C"  "SG" "SF"
## [391] "SF" "SF" "SF" "SF" "PF" "PG" "PF" "SF" "SF" "SF" "SG" "SG" "C"  "C"  "C"
## [406] "SG" "SF" "PF" "SF" "SF" "SG" "C"  "SG" "SG" "SG" "SG" "SF" "SG" "PG" "C"
## [421] "SG" "SG" "SG" "PG" "PG" "SF" "SF" "C"  "C"  "C"  "SG" "SG" "C"  "PG" "SG"
## [436] "SF" "PF" "PF" "PF" "PG" "PF" "SF" "PG" "C"  "SF" "SF" "SF" "SF" "SF" "SF"
## [451] "SF" "SG" "SG" "SG" "PG" "SG" "PG" "SG" "SG" "SG" "SG" "SG" "PG" "PF" "SF"
## [466] "PF" "C"  "SF" "SG" "SF" "PG" "PG" "SG" "SG" "SG" "C"  "SF" "SG" "C"  "C"
## [481] "C"  "SG" "PG" "SG" "PF" "SG" "C"  "C"  "C"  "C"  "C"  "SG" "SG" "PG" "PF"
## [496] "PF" "SG" "PF" "PF" "PG" "PG" "PG" "PG" "PG" "SF" "PG" "C"  "SF" "SF" "C"
## [511] "C"  "PF" "SG" "SF" "PG" "PF" "SG" "PF" "C"  "SG" "SG" "C"  "SF" "SF" "SF"
## [526] "PF" "PF" "PF" "PF" "SG" "C"  "SF" "PF" "SG" "SF" "C"  "SF" "SG" "SF" "C"
## [541] "SF" "SG" "PG" "PF" "PF" "PG" "PG" "PG" "SG" "C"  "PF" "PF" "PF" "SF" "C"
## [556] "C"  "PF" "SF" "SG" "PG" "SF" "PF" "C"  "PF" "PF" "PF" "C"  "C"  "PG" "SF"
## [571] "SF" "SG" "SF" "PF" "PG" "SG" "C"  "PG" "SG" "PF" "SG" "SF" "SF" "SF" "SG"
## [586] "C"  "C"  "C"  "SG" "SG" "SG" "SG" "SG" "PG" "PG" "PG" "PG" "C"  "C"  "PF"
## [601] "PG" "PG" "PG" "PG" "SG" "SG" "PG" "PG" "PF" "PF" "C"  "C"  "SG" "SG" "PG"
## [616] "SG" "SF" "SG" "SG" "PG" "SF" "SG" "SG" "C"  "SG" "SG" "PF" "PF" "PF" "PF"
## [631] "C"  "SG" "PG" "PF" "SG" "PG" "PG" "PG" "PG" "PG" "PG" "PG" "PG" "PF" "PF"
## [646] "PF" "SF" "SF" "SG" "SG" "SG" "SG" "SG" "SG" "SG" "PG" "SG" "PG" "PF" "C"
## [661] "SF" "PG" "SG" "PG" "PF" "SF" "PF" "SG" "PF" "C"  "C"  "C"  "SG" "SG" "PG"
## [676] "PG" "PG" "PG" "SG" "SG" "PF" "PF" "PF" "PF" "PF" "SG" "C"  "PF" "PF" "PF"
## [691] "SF" "C"  "SG" "PF" "SG" "SG" "SG" "C"  "C"  "SG" "SG" "SG" "PF" "PG" "SF"
## [706] "PG" "C"  "PF" "SF" "C"  "PF" "PG" "SG" "SG" "PG" "PG" "PG" "PG" "PG" "C"
## [721] "SF" "SG" "PG" "PG" "SF" "SF" "SG" "PG" "PG" "SG" "SG" "SG" "C"  "SG" "SG"
## [736] "SF" "PG" "PG" "PF" "SF" "PG" "PF" "C"  "SF" "PF" "SF" "PF" "PF" "PF" "PG"
## [751] "C"  "SF" "SG" "PG" "PF" "PF" "SG" "PF" "PF" "PF" "PG" "C"  "C"  "C"
```

a. Create a dataset that includes only players who play at the center position (Pos is equal to C). Name it bball1.

```
bball1 <- filter(bball, Pos == "C")
bball1
```

```
## # A tibble: 135 x 31
##    Rk    Player      Pos   Age   Tm        G    GS    MP    FG   FGA FGpct  `3P`
##    <chr> <chr>       <chr> <chr> <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
##  1 1     Precious A~ C     22    TOR      73    28  1725   265   603 0.439    56
##  2 2     Steven Ada~ C     28    MEM      76    75  1999   210   384 0.547     0
##  3 3     Bam Adebayo C     24    MIA      56    56  1825   406   729 0.557     0
##  4 5     LaMarcus A~ C     36    BRK      47    12  1050   252   458 0.55     14
##  5 8     Jarrett Al~ C     23    CLE      56    56  1809   369   545 0.677     1
##  6 22    Deandre Ay~ C     23    PHO      58    58  1713   442   697 0.634     7
##  7 23    Udoka Azub~ C     22    UTA      17     6   195    37    49 0.755     0
##  8 27    Mo Bamba    C     23    ORL      71    69  1824   296   617 0.48    107
##  9 36    Charles Ba~ C     21    PHI      23     0   168    30    47 0.638     0
## 10 48    Khem Birch  C     29    TOR      55    28   991    97   200 0.485     0
## # ... with 125 more rows, and 19 more variables: `3PA` <dbl>, `3P%` <dbl>,
## #   `2P` <dbl>, `2PA` <dbl>, `2P%` <dbl>, `eFG%` <dbl>, FT <dbl>, FTA <dbl>,
## #   FTpct <dbl>, ORB <dbl>, DRB <dbl>, TRB <dbl>, AST <dbl>, STL <dbl>,
## #   BLK <dbl>, TOV <dbl>, PF <dbl>, PTS <dbl>, points_per_minute <dbl>
## # i Use `print(n = ...)` to see more rows, and `colnames()` to see all variable names
```

b. Make your dataset only include the columns Player, Tm (Team), and PTS (points scored). Name it bball2.

```
bball2 <- select(bball1, Player, Tm, PTS)
bball2
```

```
## # A tibble: 135 x 3
##    Player            Tm      PTS
##    <chr>             <chr> <dbl>
##  1 Precious Achiuwa  TOR     664
##  2 Steven Adams      MEM     528
##  3 Bam Adebayo       MIA    1068
##  4 LaMarcus Aldridge BRK     607
##  5 Jarrett Allen     CLE     904
##  6 Deandre Ayton     PHO     997
##  7 Udoka Azubuike    UTA      80
##  8 Mo Bamba          ORL     756
##  9 Charles Bassey    PHI      69
## 10 Khem Birch        TOR     247
## # ... with 125 more rows
## # i Use 'print(n = ...)' to see more rows
```

c. Using arrange, print out the dataset from most to least points scored. (It will by default print out the first 10, which is all you need!). Name it bball3.

```
bball3 <- arrange(bball2, desc(PTS))
bball3
```

```
## # A tibble: 135 x 3
##    Player             Tm      PTS
##    <chr>              <chr> <dbl>
##  1 Joel Embiid        PHI    2079
##  2 Nikola Jokić       DEN    2004
##  3 Karl-Anthony Towns MIN    1818
##  4 Jonas Valančiūnas  NOP    1314
##  5 Nikola Vučević     CHI    1288
##  6 Christian Wood     HOU    1218
##  7 Bam Adebayo        MIA    1068
##  8 Bobby Portis       MIL    1052
##  9 Rudy Gobert        UTA    1027
## 10 Kevin Love         CLE    1007
## # ... with 125 more rows
## # i Use 'print(n = ...)' to see more rows
```

d. Rename the Tm column "Team and the PTS column"Points Scored". (You can just have this one print out, you don't need to save it!)

```
rename(bball3, Team = Tm, Points_Scored = PTS)
```

```
## # A tibble: 135 x 3
##    Player             Team  Points_Scored
##    <chr>              <chr>         <dbl>
##  1 Joel Embiid        PHI            2079
##  2 Nikola Jokić       DEN            2004
##  3 Karl-Anthony Towns MIN            1818
##  4 Jonas Valančiūnas  NOP            1314
##  5 Nikola Vučević     CHI            1288
##  6 Christian Wood     HOU            1218
##  7 Bam Adebayo        MIA            1068
```

```
##  8 Bobby Portis       MIL            1052
##  9 Rudy Gobert        UTA            1027
## 10 Kevin Love         CLE            1007
## # ... with 125 more rows
## # i Use ‘print(n = ...)‘ to see more rows
```

2. Create a table similar to the one above, but include the point guards (Pos is equal to PG) with the most minutes played (MP). Rename the columns Team and Minutes Played.

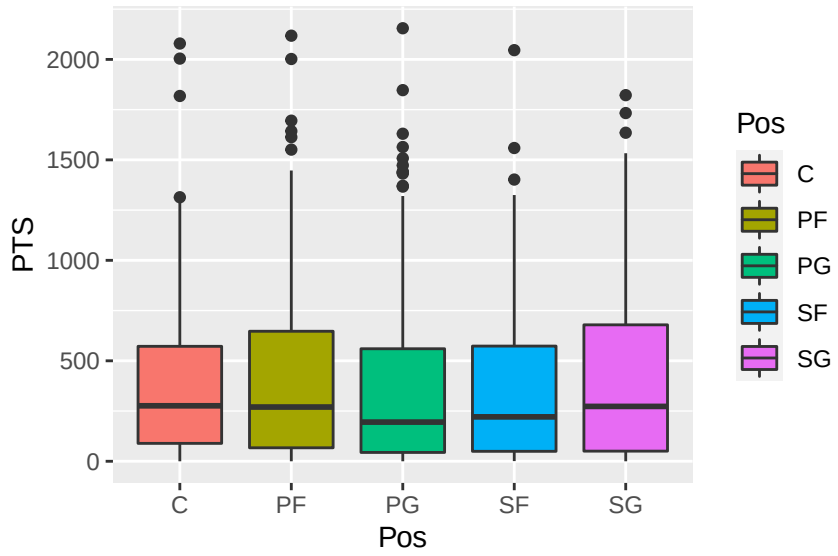
```
bball|>
  filter(Pos == "PG")|>
  select(Player, Tm, MP)|>
  rename(Team = Tm, Minutes_Played = MP)|>
  arrange(desc(Minutes_Played))
```

```
## # A tibble: 145 x 3
##    Player            Team  Minutes_Played
##    <chr>             <chr>          <dbl>
##  1 Russell Westbrook LAL             2678
##  2 Trae Young        ATL             2652
##  3 Tyrese Maxey      PHI             2650
##  4 Fred VanVleet     TOR             2462
##  5 Darius Garland    CLE             2430
##  6 LaMelo Ball       CHO             2422
##  7 James Harden      TOT             2419
##  8 Dejounte Murray   SAS             2366
##  9 Luka Dončić       DAL             2301
## 10 Marcus Smart      BOS             2296
## # ... with 135 more rows
## # i Use ‘print(n = ...)‘ to see more rows
```

Data Visualization

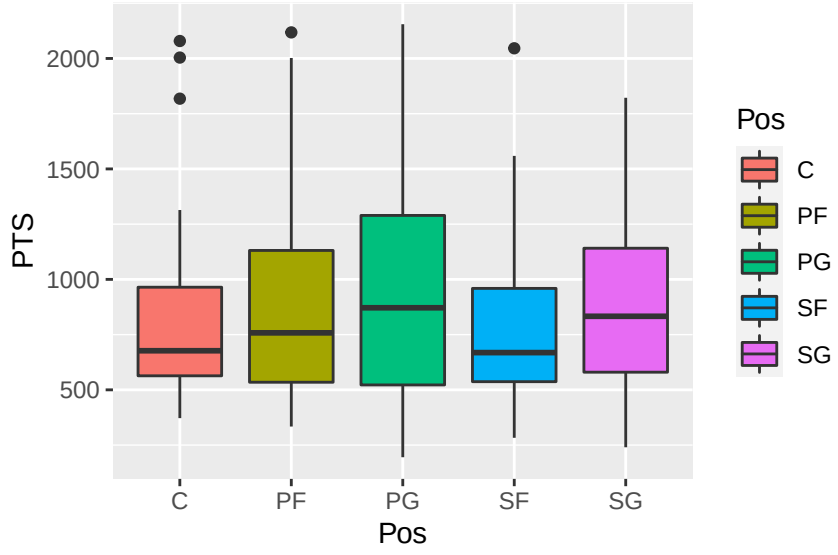
3. Create a boxplot of the number of points scored (PTS) by Position (Pos).

```
bball|>
  ggplot(aes(x = Pos, y = PTS))+
  geom_boxplot(aes(fill = Pos))
```



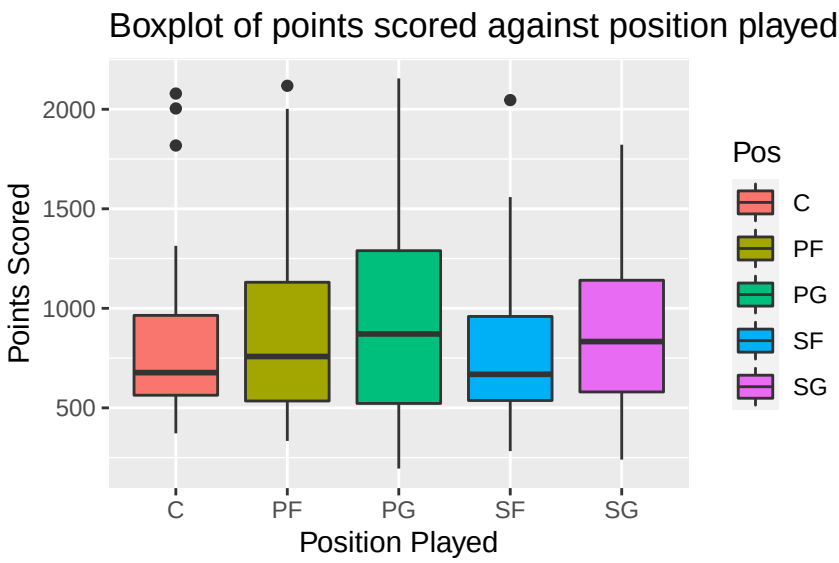
4. Notice that the number of points scored may be as low as zero! This is because we include players who may have only played for a few minutes the entire season. Let's create a new dataset, bball58 which only includes players who played in at least 58 games (variable G is number of games played in). Recreate your boxplot from 3 using this

```
bball58 <- filter(bball, G >= 58)
bball58|>
  ggplot(aes(x = Pos, y = PTS))+
  geom_boxplot(aes(fill = Pos))
```



5. Add axis labels and a title to your graph above.

```
bball58|>
  ggplot(aes(x = Pos, y = PTS))+
  geom_boxplot(aes(fill = Pos))+
  labs(title = "Boxplot of points scored against position played")+
  xlab("Position Played")+
  ylab("Points Scored")
```



6. Based on your graph, which position tended to score the most points? Which had the most variability in points scored?

Ans: The point guard seems to score the mostr as well as has the highest variability in points scored.

In the next problem, we will use a subset of the General Social Survey conducted in the US in 2016.

```
#library(socviz)
#gss_sm
gss_sm <- read_csv("~/Mscs 264 S23/Class/Data/gss_sm.csv")
```

```
## Rows: 2867 Columns: 32
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (23): degree, race, sex, region, income16, relig, marital, padeg, madeg,...
## dbl  (9): year, id, ballot, age, childs, sibs, pres12, wtssall, obama
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

See the ?gss_sm for defnitions of all variables.

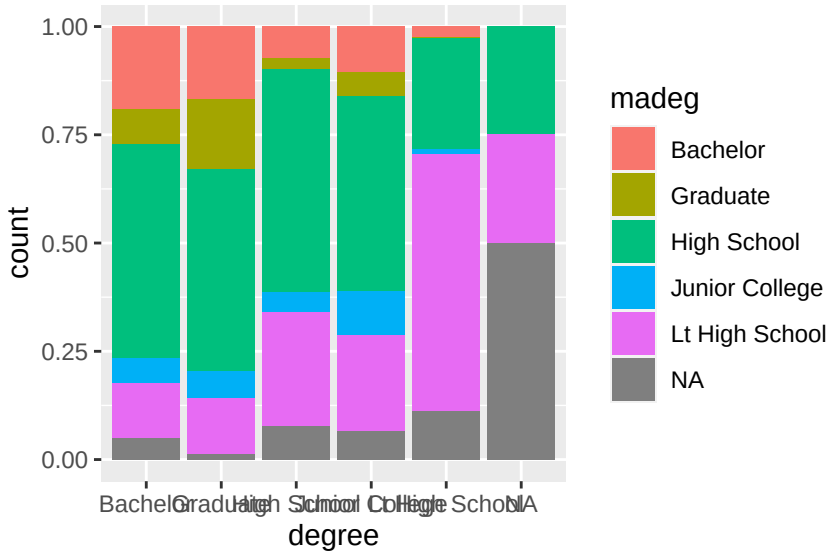
```
colnames(gss_sm)
```

```
##  [1] "year"        "id"          "ballot"      "age"         "childs"
##  [6] "sibs"        "degree"      "race"        "sex"         "region"
## [11] "income16"    "relig"       "marital"     "padeg"       "madeg"
## [16] "partyid"     "polviews"    "happy"       "partners"    "grass"
## [21] "zodiac"      "pres12"      "wtssall"     "income_rc"   "agegrp"
## [26] "ageq"        "siblings"    "kids"        "religion"    "bigregion"
## [31] "partners_rc" "obama"
```

We will examine if there is a relationahip between the highest degree eared by a respondent's mother (madeg) and the respondent's degree (degree).

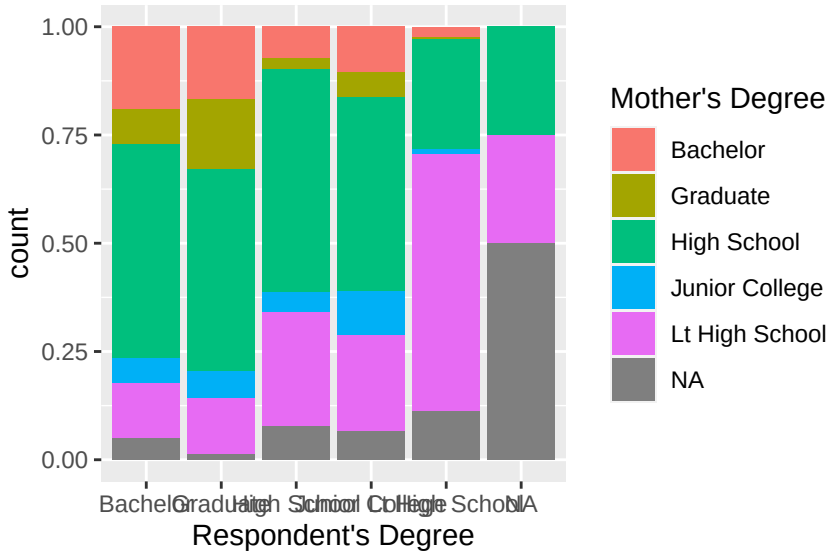
7. Create a segmented bar chart showing this relationship.

```
gss_sm|>
  ggplot()+
  geom_bar(mapping = aes(x = degree, fill = madeg), position = "fill")
```



8. Edit the x-axis and legend labels in your graph above.

```
gss_sm|>
  ggplot()+
  geom_bar(mapping = aes(x = degree, fill = madeg), position = "fill")+
  labs(fill = "Mother's Degree")+
  xlab("Respondent's Degree")
```



9. What relationship do you observe between mother's highest degree and respondent's degree?

Ans: There does seem to be some positive correlation between respondent's degree and their mother's degree. Like the higher the degree of the respondent the more likely their mothers' will have a Bachelor degree. But their is a high prevalence of high school degrees in mothers' degrees. And if a respondent has unavailable degree it's very likely that their mothers' will

have the same.

10. Create a side by side bar chart (dodge) for the same relationship. Do you think the segmented or side by side chart shows the relationship more clearly?

```
gss_sm|>
  ggplot()+
  geom_bar(mapping = aes(x = degree, fill = madeg), position = "dodge")+
  labs(fill = "Mother's Degree")+
  xlab("Respondent's Degree")
```