CSCI/DASC 6020: Written Assignment 03

Your Name

2024-09-24

Assignment Goal

The goal of this assignment is to demonstrate your understanding of exploratory data analysis.

Assignment Specification

The first step is to identify a **relevant** open-source dataset. The dataset should have at least 500 instances and contain a mix of at least ten **continuous** and **categorical** variables. The next step is to perform various exploratory data analysis tasks discussed in the class. The final step is to summarize your findings. You may use Quarto/RMarkdown or Jupyter/Python to respond to this assignment. Use this document as a template to prepare your response.

Some open-source data sources for this assignment are the following. This list is not exhaustive and you are not required to select a datset from this list.

1. Integrated Postsecondary Education Data System (IPEDS) is a system of 12 interrelated survey components conducted annually that gathers data from every college, university, and technical and vocational institution that participates in the federal student financial aid programs. IPEDS Website
2. U.S. Securities and Exchange Commission (SEC) Financial Statement Data Sets
3. United Nations (UN) Datasets
4. World Bank Open Data
5. The Library of Congress Datasets
6. NASDAQ Historical Datasets
7. The World Factbook and Guide to Country Comparisons

1 The Data Quality Report

Document the data quality report in two separate tables, one for the continuous features and another for the categorical features. Use the table format discussed in the class.

Ensure you run the following two chunks before any calculations

```
#|label: load-functions&packages
#|message: false

library(tidyverse)

#Finds the mode of a column and its respective frequency
#Input: A vector object, the column you wish to find the mode of
#Output: Data-frame with two vectors:
# x; one or more mode elements (will only return multiple if their is a tie)
# Freq; the frequency count for each element in x in the passed vector
find_mode <- function(x) {
  tab <- table(x)
  sorted_t <- tab %>%
      as.data.frame(stringsAsFactors=FALSE) %>%
      arrange(desc(Freq))
  tab1 <- sorted_t %>% slice_max(Freq)
}

#Builds a Data Quality Report (DQR) for the categorical variables of your data-set
#Input is a data frame containing the columns of your categorical data
#Output is a printed DQR
categorical.dqr <- function(a) {

  #create vectors for each column of DQR
  feature <- colnames(a)
  count <- nrow(a)
  missing <- c()
  card <- c()
  min <- c()
  mode1 <- c()
  mode2 <- c()
  mode1_freq <- c()
  mode2_freq <- c()

  #loop through each column in the data-frame and extract desired values
  for (i in feature) {
```

```
    ca <- length(unique(a[[i]]))
    mi <- sum(is.na(a[[i]]))
    mo <- find_mode(a[[i]])
    missing <- append(missing, c(mi))
    card <- append(card, c(ca))
    mode1 <- append(mode1, c(mo[1,1]))
    mode2 <- append(mode2, c(mo[2,1]))
    mode1_freq <- append(mode1_freq, c(mo[1,2]))
    mode2_freq <- append(mode2_freq, c(mo[2,2]))
  }

  #Convert percentage columns
  mode1_per <- (mode1_freq / count) * 100
  mode2_per <- (mode2_freq / count) * 100
  missing <- (missing / count) * 100

  #build and print the DQR
  df <- data.frame(feature, count, missing, card, mode1, mode1_freq, mode1_per, mode2, mode2

  print(df)
}

#Builds a Data Quality Report (DQR) for the continuous variables of your data-set
#Input is a data frame containing the columns of your continuous data
#Output is a printed DQR
continuous.dqr <- function(a) {

  #create vectors for each column of DQR
  feature <- colnames(a)
  count <- nrow(a)
  missing <- c()
  card <- c()
  Min <- c()
  Max <- c()
  Mean <- c()
  q1 <- c()
  Median <- c()
  q3 <- c()
  standard_dev <- c()

  #loop through each column in the data-frame and extract desired values
  for (i in feature) {
```

```
    ca <- length(unique(a[[i]]))
    mis <- sum(is.na(a[[i]]))
    mi <- min(a[[i]])
    ma <- max(a[[i]])
    me <- mean(a[[i]])
    st <- sd(a[[i]])
    st <- format(round(st, 1), nsmall = 1) #format these numbers to 1 decimal place
    me <- format(round(me, 1), nsmall = 1) #format these numbers to 1 decimal place
    card <- append(card, ca)
    missing <- append(missing, c(mis))
    Min <- append(Min, mi)
    Max <- append(Max, ma)
    Mean <- append(Mean, me)
    standard_dev <- append(standard_dev, st)

    #quantile command is used to gather q1, median, and q3 data
    q_all <- quantile(a[[i]], prob=c(.25,.5,.75), type=1)
    q_all <- q_all %>% as.data.frame(stringsAsFactors=FALSE)
    q1 <- append(q1, q_all[1,1])
    Median <- append(Median, q_all[2,1])
    q3 <- append(q3, q_all[3,1])
  }

  #convert to percent
  missing <- (missing / count) * 100

  #build and print DQR
  df <- data.frame(feature, count, missing, card, Min, q1, Mean, Median, q3, Max, standard_de
  print(df)
}

#Makes a histogram for each column in the data frame passed to it
#Input: data-frame of continuous functions
#Output: Histogram for each column in the data-frame
make.hist <- function(a) {
  feature <- colnames(a)
  for (i in feature) {
    hist(a[[i]], xlab = i, main = 'Frequency Distribution')
  }
}

#Creates the data-frame to show issues, luckily for us we have none and all data looks how we
```

```r
id.issues <- function(a) {

  features <- colnames(a)

  issue <- c()
  handling <- c()
  for (i in features) {
    x = "n/a"
    issue <- append(issue, c(x))
    handling <- append(handling, c(x))
  }

  df <- data.frame(features, issue, handling)
  print(df)
}
```

```r
#|label: load-datasets
x <- read.csv(file = 'game_stats_2018.csv')
y <- read.csv(file = 'game_stats_2019.csv')
gs_18 <- as.tibble(x)
gs_19 <- as.tibble(y)
(gs_18)
```

```
# A tibble: 256 x 15
    Week HomeTeam AwayTeam Total H.RushAtt H.RushYards H.PassYards H.Turnover
   <int> <chr>    <chr>    <dbl>     <int>       <int>       <int>      <int>
 1     1 PHI      ATL       44.5        27         113         132          2
 2     1 CAR      DAL       42.5        32         147         161          1
 3     1 CLE      PIT       41          38         177         197          1
 4     1 IND      CIN       47.5        22          75         319          2
 5     1 ARI      WAS       43.5        15          68         153          2
 6     1 DEN      SEA       42.5        32         146         329          3
 7     1 GNB      CHI       45          18          69         341          2
 8     1 MIA      TEN       43.5        29         120         230          2
 9     1 MIN      SFO       46.5        32         116         244          1
10     1 NOR      TAM       50          13          43         439          2
# i 246 more rows
# i 7 more variables: H.Score <int>, A.RushAtt <int>, A.RushYards <int>,
#   A.PassYards <int>, A.Turnover <int>, A.Score <int>, Result <int>
```

```
head(gs_19)
```

```
# A tibble: 6 x 15
   Week HomeTeam AwayTeam Total H.RushAtt H.RushYards H.PassYards H.Turnover
  <int> <chr>    <chr>    <dbl>     <int>       <int>       <int>      <int>
1     1 CHI      GNB       47          15          46         228          1
2     1 CAR      LAR       49.5        23         127         239          3
3     1 CLE      TEN       44          20         102         285          3
4     1 ARI      DET       45.5        23         112         308          1
5     1 DAL      NYG       44          30          89         405          0
6     1 JAX      KAN       49          16          81         350          2
# i 7 more variables: H.Score <int>, A.RushAtt <int>, A.RushYards <int>,
#   A.PassYards <int>, A.Turnover <int>, A.Score <int>, Result <int>
```

```
cat_18 <- select(gs_18, Week, HomeTeam, AwayTeam, Result)
cat_19 <- select(gs_19, Week, HomeTeam, AwayTeam, Result)
con_18 <- select(gs_18, Total, H.RushAtt, H.RushYards, H.PassYards, H.Turnover, H.Score, A.Ru
con_19 <- select(gs_19, Total, H.RushAtt, H.RushYards, H.PassYards, H.Turnover, H.Score, A.Ru
```

Data Quality Report for categorical functions in game_stats_2018.csv

```
#|label: categorical data quality report
categorical.dqr(cat_18)
```

```
   feature count missing card mode1 mode1_freq mode1_per mode2 mode2_freq
1     Week   256       0   17     1         16   6.25000     2         16
2 HomeTeam   256       0   32   ARI          8   3.12500   ATL          8
3 AwayTeam   256       0   32   ARI          8   3.12500   ATL          8
4   Result   256       0    3    -1        133  51.95312  <NA>         NA
  mode2_per
1     6.250
2     3.125
3     3.125
4        NA
```

Data Quality Report for continuous functions in game_stats_2018.csv

```
continuous.dqr(con_18)
```

```
       feature count missing card  Min  q1  Mean Median    q3   Max
1        Total   256       0   45 36.5  43  46.5     46  49.5  63.5
2    H.RushAtt   256       0   37  6.0  21  26.5     26  32.0  53.0
3  H.RushYards   256       0  142 14.0  77 117.1    108 149.0 323.0
4  H.PassYards   256       0  179 57.0 198 258.7    249 313.0 471.0
5   H.Turnover   256       0    6  0.0   0   1.3      1   2.0   5.0
6      H.Score   256       0   42  0.0  17  24.4     24  31.0  54.0
7    A.RushAtt   256       0   37  9.0  19  25.3     24  30.0  49.0
8  A.RushYards   256       0  135 22.0  76 111.8    105 139.0 273.0
9  A.PassYards   256       0  168 90.0 188 250.2    241 300.0 478.0
10  A.Turnover   256       0    7  0.0   0   1.5      1   2.0   6.0
11     A.Score   256       0   44  0.0  16  22.2     22  28.0  51.0
   standard_dev
1           4.9
2           7.9
3          54.6
4          82.1
5           1.2
6          10.6
7           7.4
8          48.9
9          82.4
10          1.3
11          9.9
```

Data Quality Report for categorical functions in game_stats_2019.csv

```
categorical.dqr(cat_19)
```

```
   feature count missing card mode1 mode1_freq mode1_per mode2 mode2_freq
1     Week   256       0   17     1         16     6.250     2         16
2 HomeTeam   256       0   32   ARI          8     3.125   ATL          8
3 AwayTeam   256       0   32   ARI          8     3.125   ATL          8
4   Result   256       0    3     1        128    50.000  <NA>         NA
  mode2_per
1     6.250
2     3.125
3     3.125
4        NA
```

Data Quality report for continuous functions in game_stats_2019.csv

```
continuous.dqr(con_19)
```

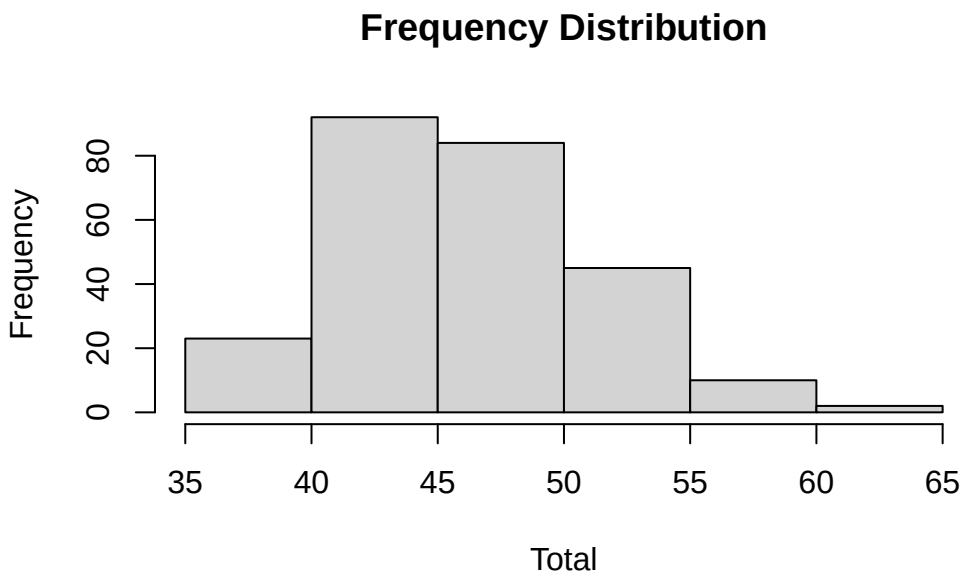
```
       feature count missing card Min    q1  Mean Median  q3   Max standard_dev
1        Total   256       0   38  35  42.5  45.1   45.5  48  55.5          4.0
2    H.RushAtt   256       0   38   7  21.0  25.9   25.0  31  47.0          7.4
3  H.RushYards   256       0  154  17  74.0 111.8  104.0 144 285.0         53.2
4  H.PassYards   256       0  168  77 195.0 255.5  247.0 309 517.0         77.6
5   H.Turnover   256       0    8   0   1.0   1.5    1.0   2   7.0          1.4
6      H.Score   256       0   42   0  16.0  22.7   23.0  30  53.0          9.9
7    A.RushAtt   256       0   37   9  21.0  26.4   26.0  32  48.0          7.3
8  A.RushYards   256       0  138  20  75.0 114.0  106.0 145 285.0         52.3
9  A.PassYards   256       0  166  82 196.0 248.0  243.0 300 458.0         75.1
10  A.Turnover   256       0    6   0   0.0   1.3    1.0   2   5.0          1.2
11     A.Score   256       0   44   0  16.0  22.9   23.0  30  59.0         10.4
```

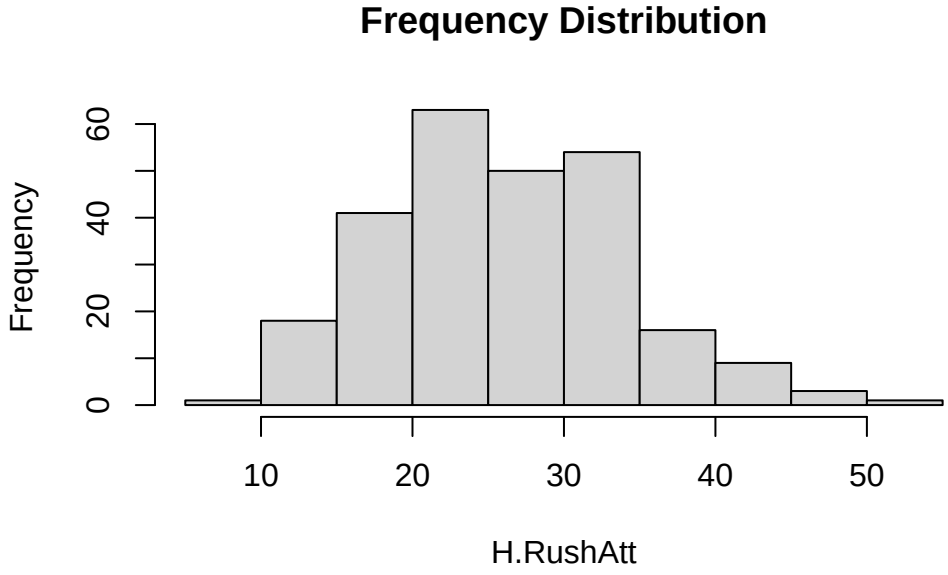
2 Histograms of Continuous Features

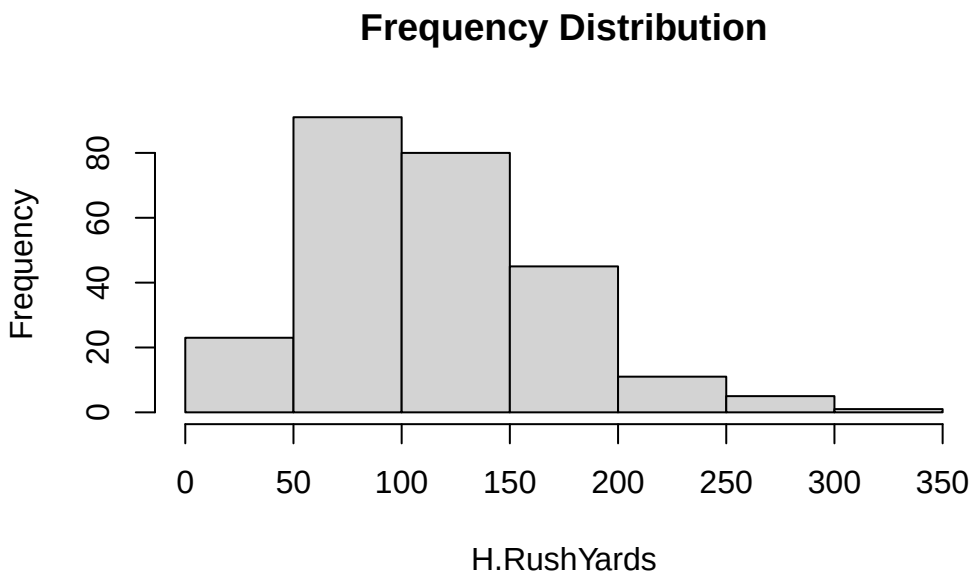
Create a **histogram** for each continuous feature. What probability distributions the histograms reveal? For example, uniform, normal (unimodal), unimodal (skewed right), unimodal (skewed left), exponential, and multimodal.

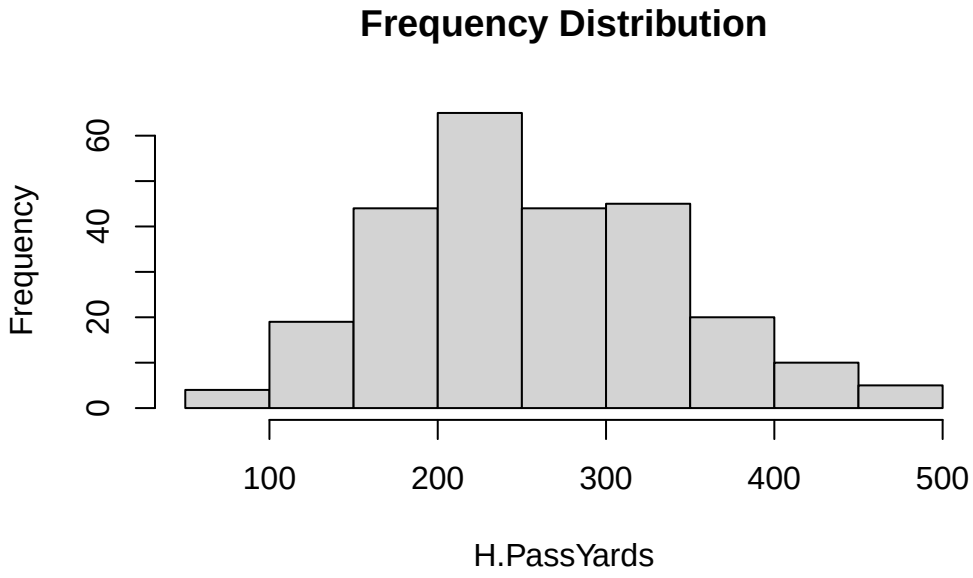
Histogram for continuous features in game_stats_2018.csv

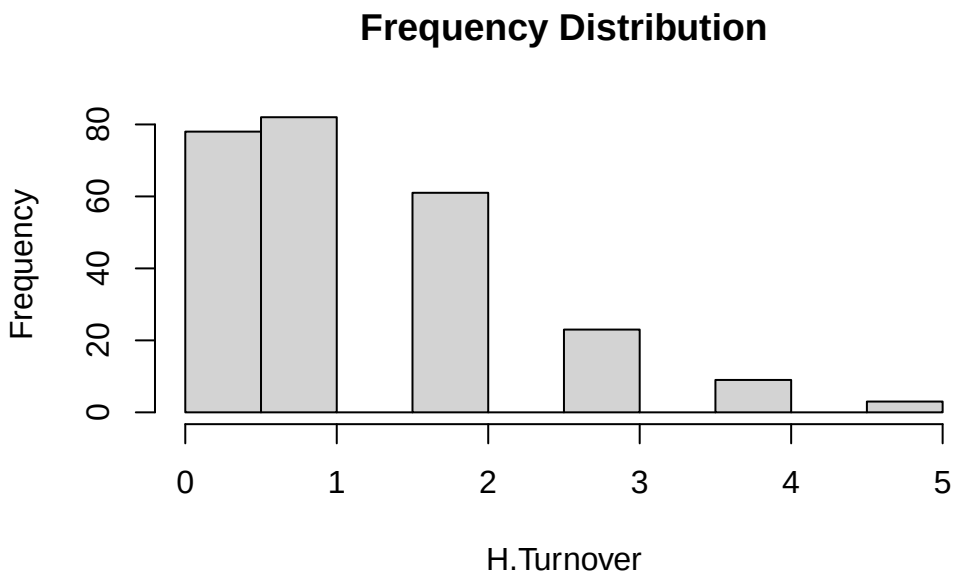
```
make.hist(con_18)
```

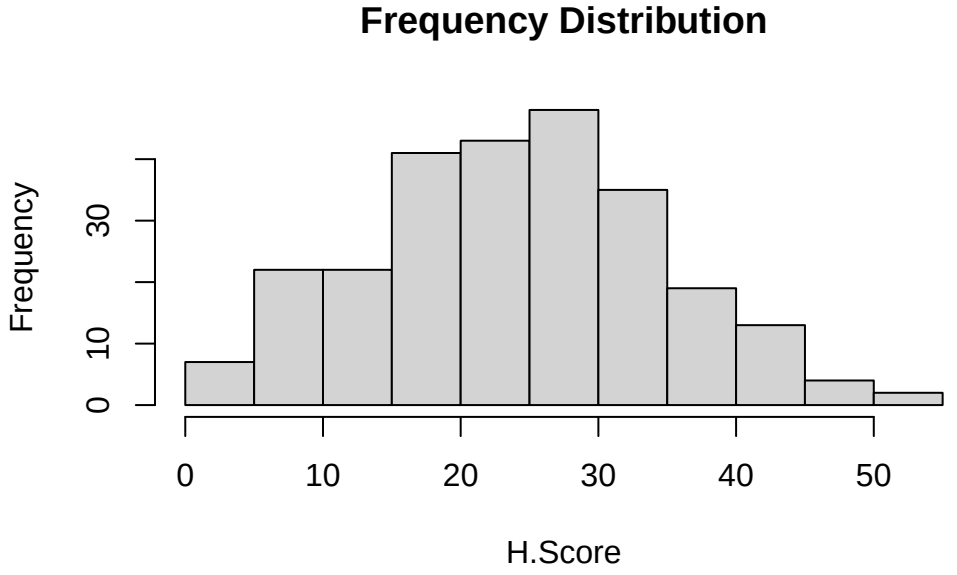

Frequency Distribution
Frequency
0
20
40
60
80
35
40
45
50
55
60
65
Total

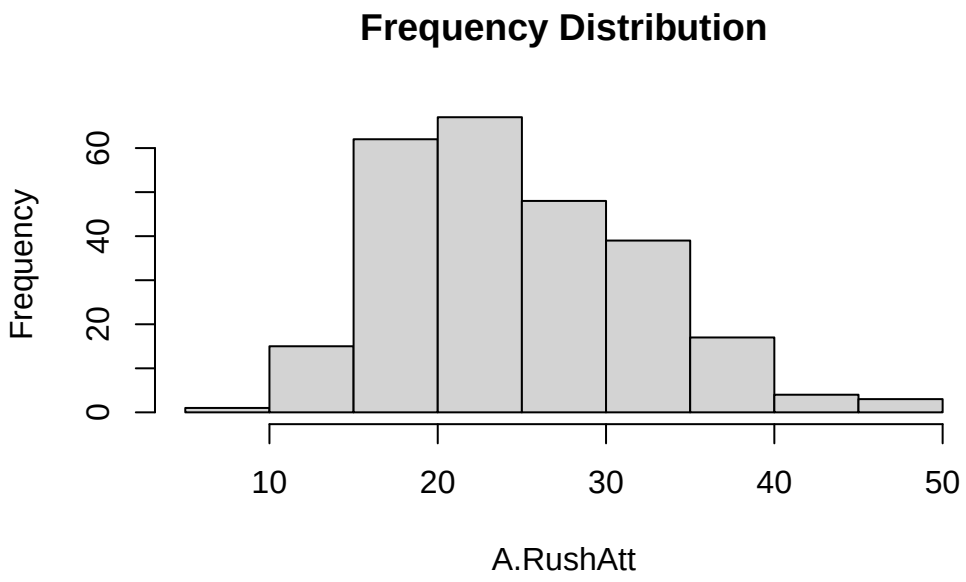

Frequency Distribution
Frequency
0
20
40
60
10
20
30
40
50
H.RushAtt

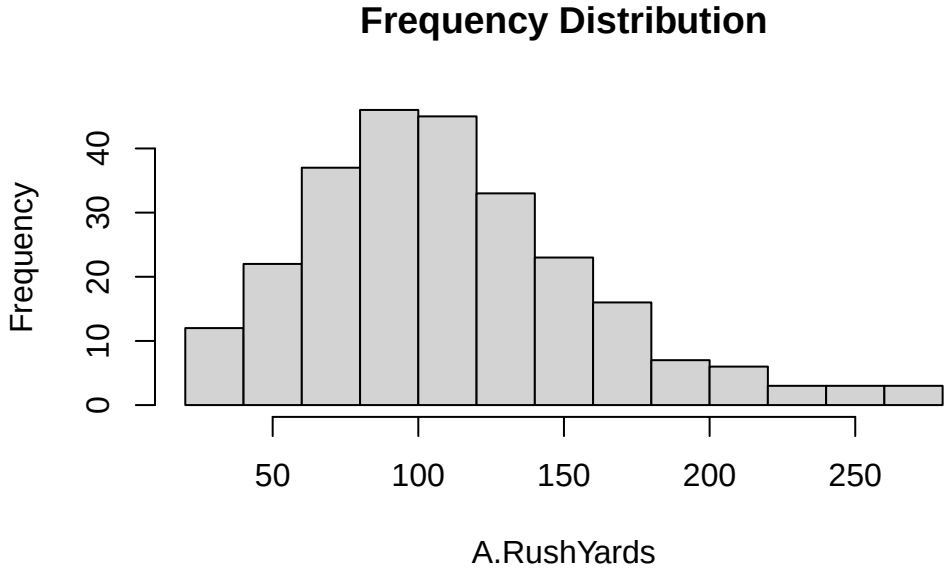

Frequency Distribution
Frequency
0
20
40
60
80
0
50
100
150
200
250
300
350
H.RushYards

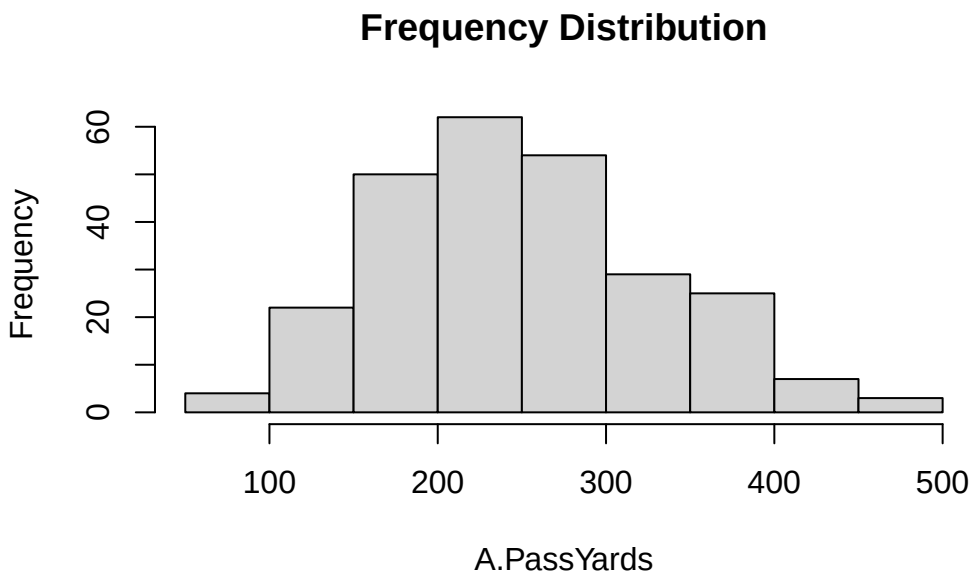

Frequency Distribution
Frequency
0
20
40
60
100
200
300
400
500
H.PassYards

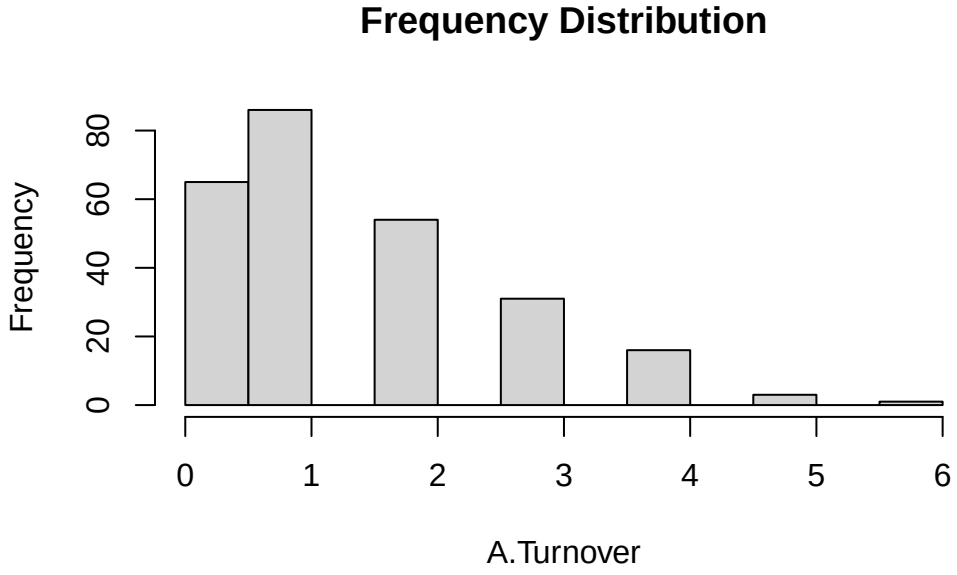

Frequency Distribution
Frequency
0
20
40
60
80
0
1
2
3
4
5
H.Turnover

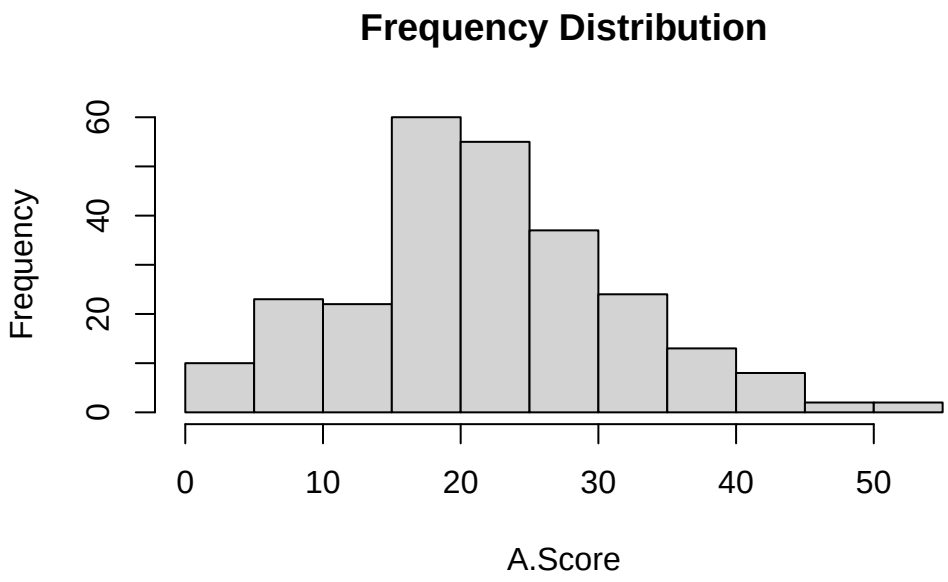

Frequency Distribution
Frequency
0
10
30
0
10
20
30
40
50
H.Score


Frequency Distribution
Frequency
0
20
40
60
10
20
30
40
50
A.RushAtt


Frequency Distribution
Frequency
0
10
20
30
40
50
100
150
200
250
A.RushYards


Frequency Distribution
Frequency
0
20
40
60
100
200
300
400
500
A.PassYards


Frequency Distribution
Frequency
0
20
40
60
80
0
1
2
3
4
5
6
A.Turnover



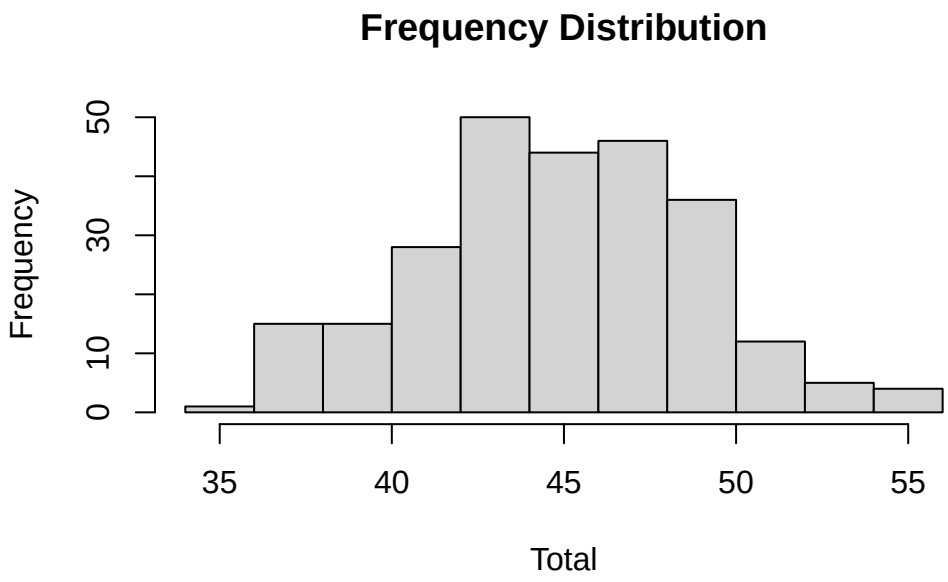
Features that exhibit a **normal** distribution: H.PassYards (there is a very very slight second peak that may cause this to be considered multimodal but I'm choosing to consider it normal for this assignment), and H.Score

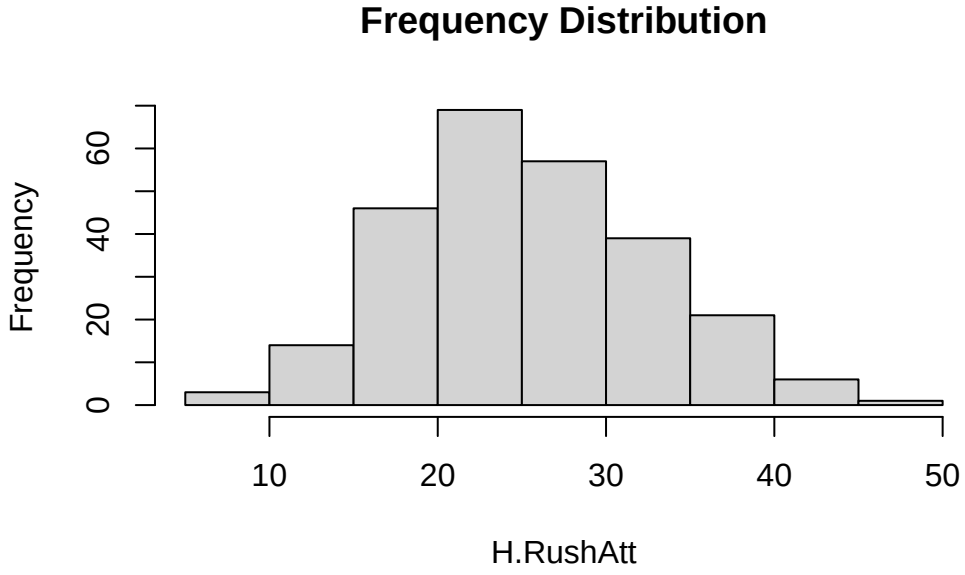
Features that exhibit a **skewed right** distribution: Total, H.RushYards, H.Turnover, A.RushAtt, A.RushYards, A.PassYards (this one was close to being normal), A.Turnover, A.Score

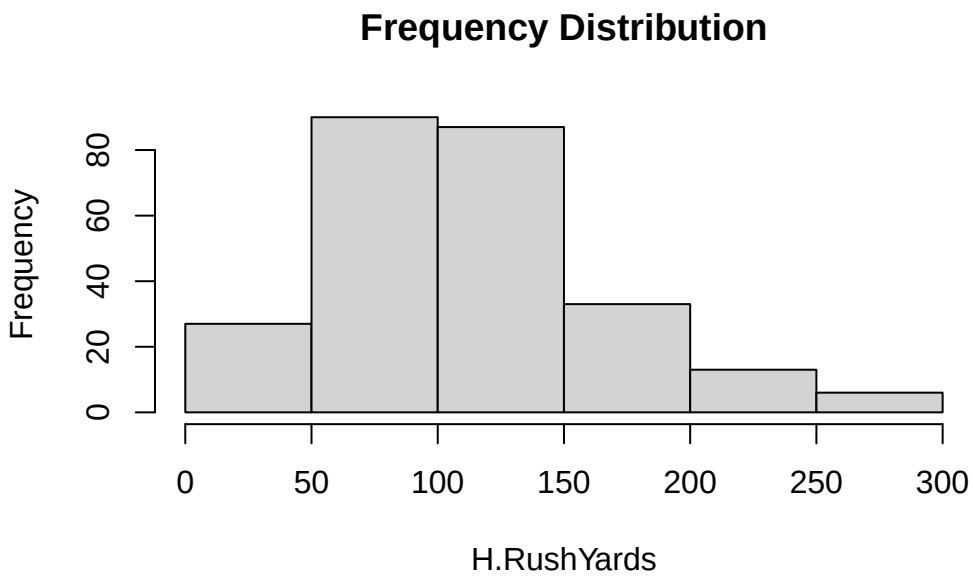
These distributions seem, from a quick glance, consistent with what we would expect. Most team stats (aka home and away rush/pass yards and attempts, etc.) will be skewed right as higher scoring (and therefore higher valued and higher Total) games will be less common. Home score being normal is interesting, we'd expect it to be less skewed right than away score yet it being normal may be a trend in higher scoring home teams which we can potentially verify in game_stats_2019.csv. Home pass yards may be the most interesting distribution especially in correlation to home score. Like home score we'd expect it to be less skewed right than its away counterpart yet home rush yards is skewed right. These two things together may suggest a rise in passing and scoring in the NFL especially for home teams.

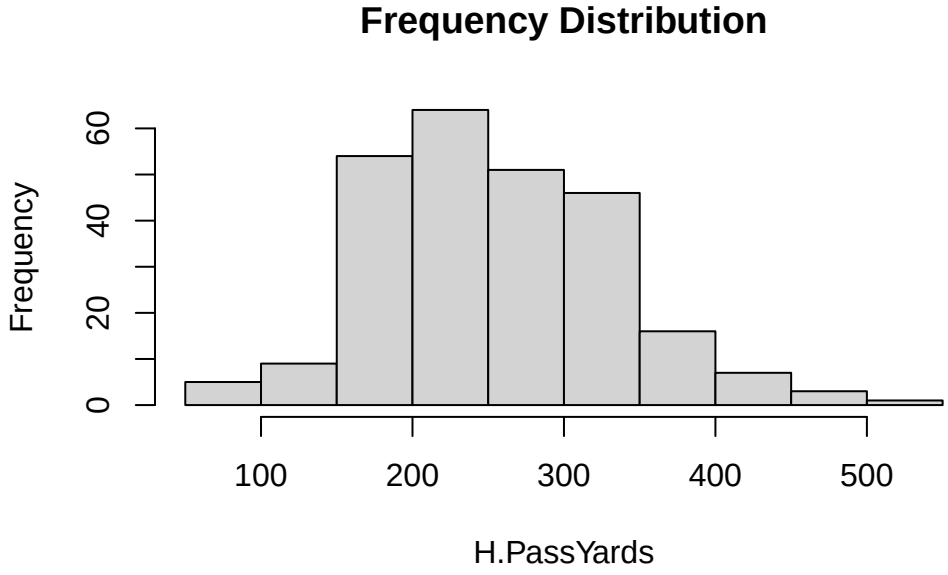
Histogram for continuous features in game_stats_2019.csv

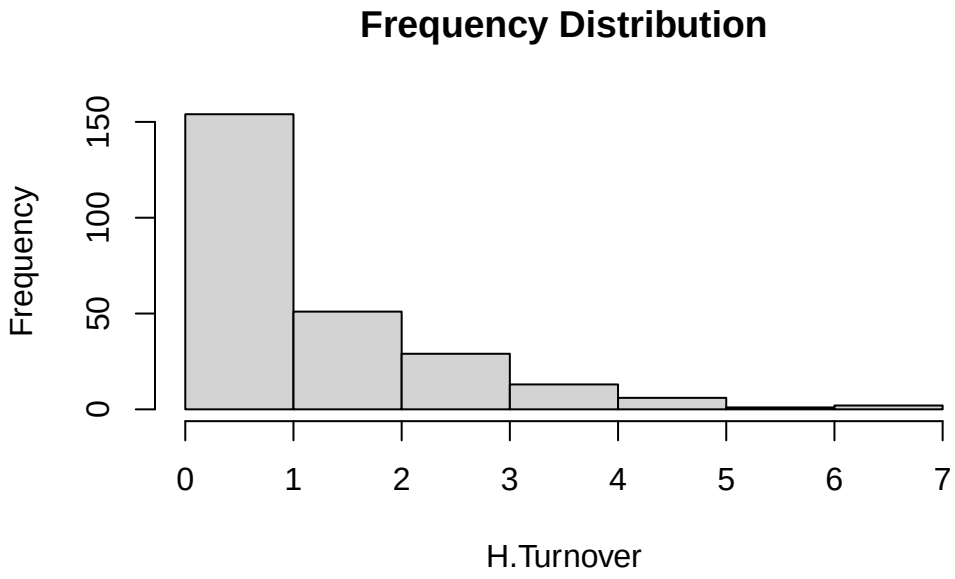
```
make.hist(con_19)
```

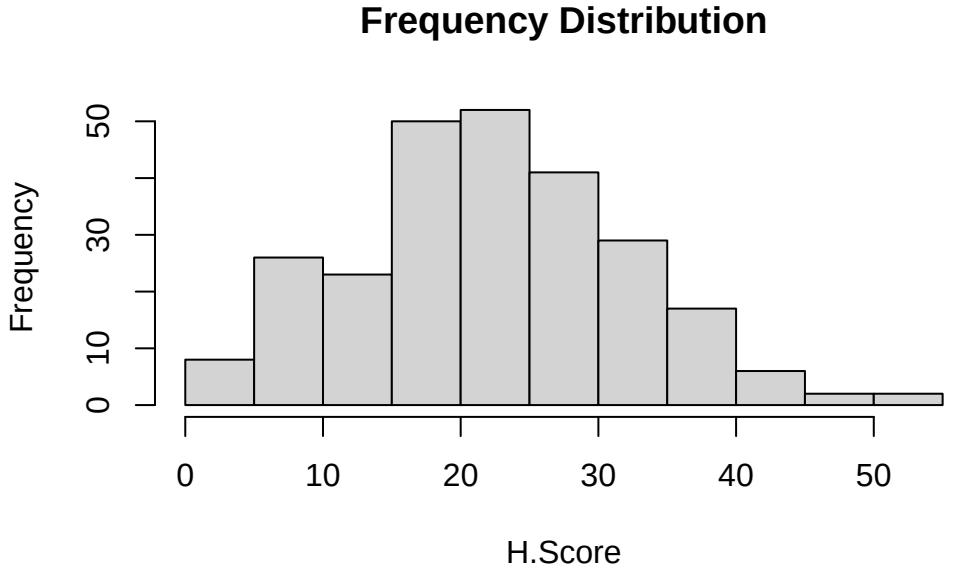

Frequency Distribution
Frequency
0
10
30
50
35
40
45
50
55
Total

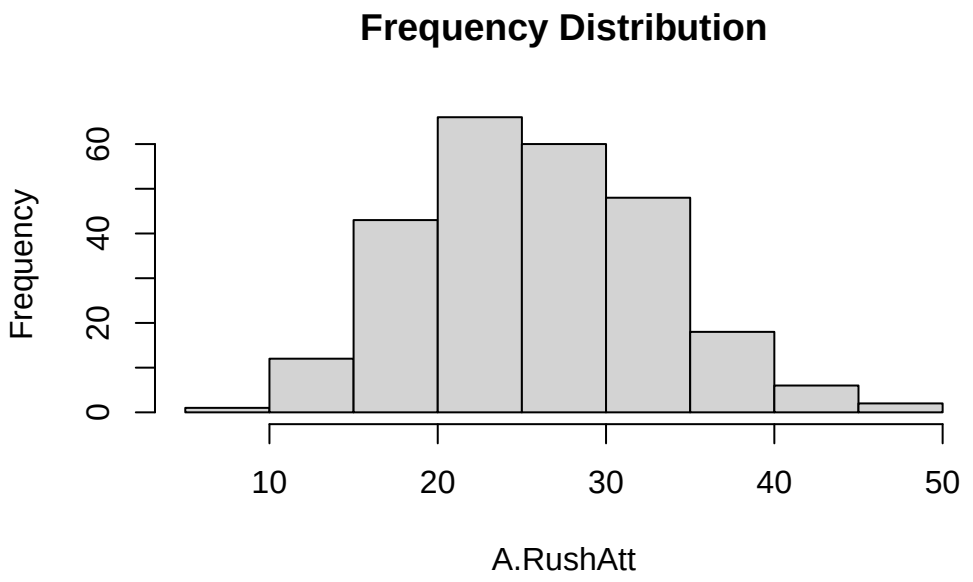

Frequency Distribution
Frequency
0
20
40
60
10
20
30
40
50
H.RushAtt

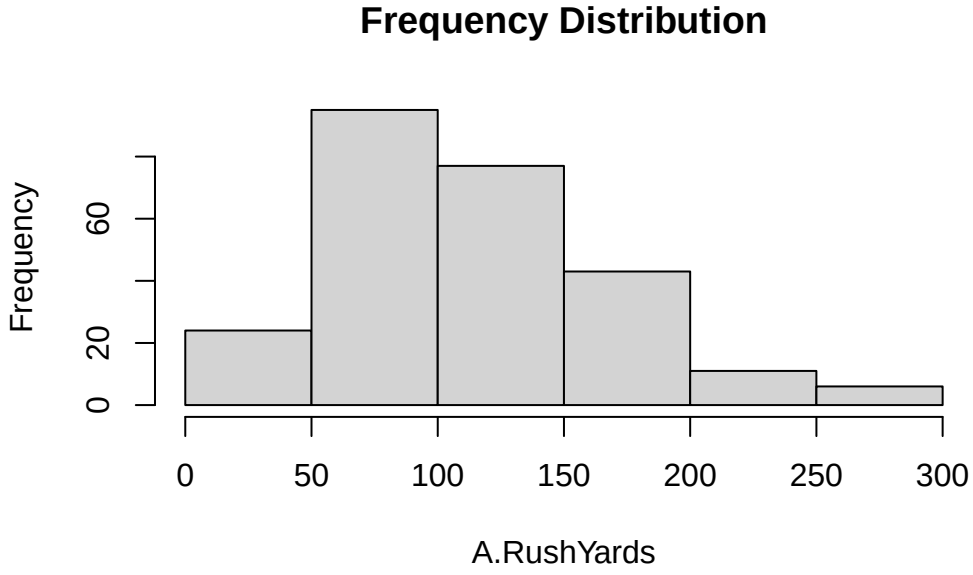

Frequency Distribution
Frequency
0
20
40
60
80
0
50
100
150
200
250
300
H.RushYards

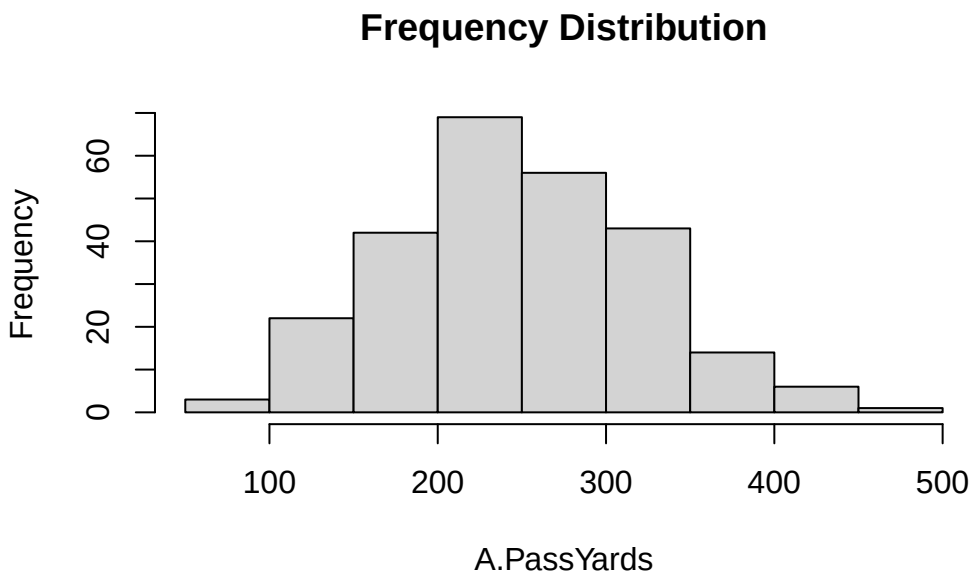

Frequency Distribution
Frequency
0
20
40
60
100
200
300
400
500
H.PassYards

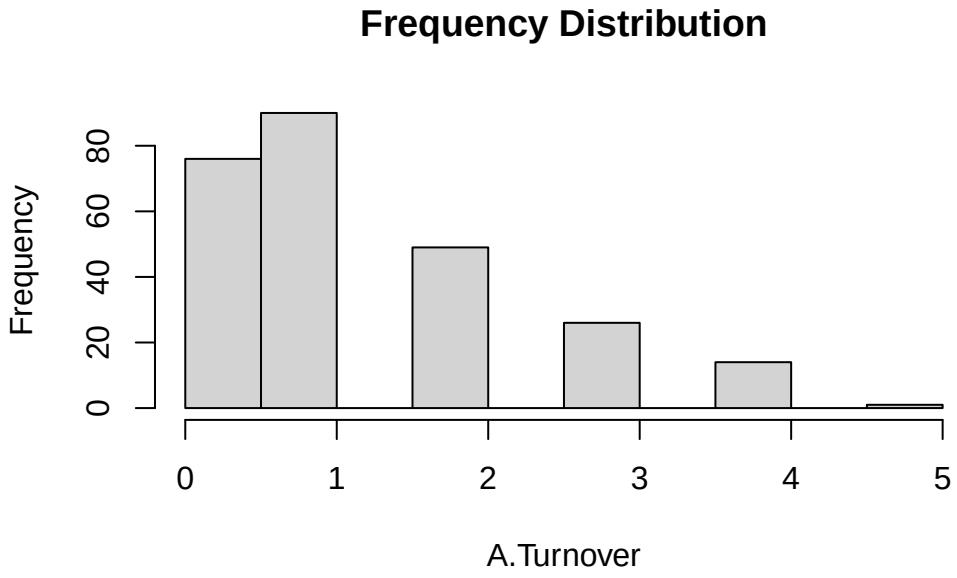

Frequency Distribution
Frequency
0
50
100
150
0
1
2
3
4
5
6
7
H.Turnover

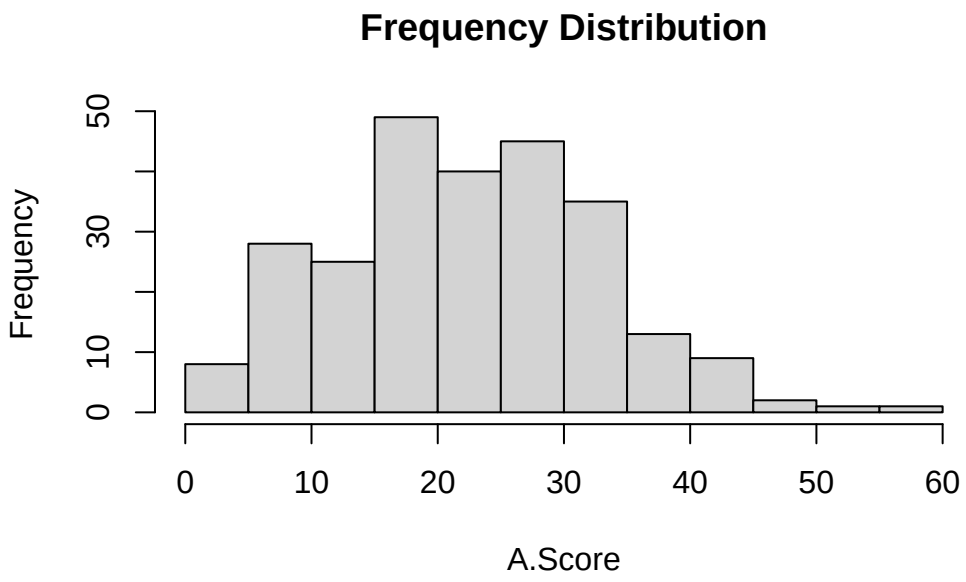

Frequency Distribution
Frequency
0
10
30
50
0
10
20
30
40
50
H.Score


Frequency Distribution
Frequency
0
20
40
60
10
20
30
40
50
A.RushAtt


Frequency Distribution
Frequency
0
20
60
0
50
100
150
200
250
300
A.RushYards


Frequency Distribution
Frequency
0
20
40
60
100
200
300
400
500
A.PassYards


Frequency Distribution
Frequency
0
20
40
60
80
0
1
2
3
4
5
A.Turnover



Features that exhibit a **normal** distribution: Total & H.Score (once again both of these could be considered multi-modal but its very small), A.RushAtt

Features that exhibit a **skewed right** distribution: H.RushAtt, H.RushYards, H.PassYards (very slightly skewed right), H.Turnover, A.RushAtt (again very slightly), A.RushYards, A.PassYards (again very slightly), A.Turnover

Features that exhibit a **multi-modal** distribution: A.Score (could be considered skewed right for simplicity but there's enough peaks ill call this one multimodal)

These once again seem consistent with expected results. Looking back at game_score_2018.csv we see Home Score remain normal yet H.PassYards is skewed right (albeit very slightly) casting some doubt on our earlier interpretation. Two more differences stand out: Total has a normal distrubtion this time (perhaps odds makers adjusted to the higher scoring saw last year), and A.Score is multimodal (still skewed right though) suggesting that the improving offenses from 2018 started preforming better on the road.

3 Identification of Data Quality Issues

Consider the missing values, irregular cardinality problems, and outliers. Summarize the **data quality issues** using a three-column table. The first column is the feature name, the second column is the associated data quality issue, and the third column describes potential handling strategies.

Luckily for us our data looks wonderful

```
id.issues(gs_18)
```

```
      features issue handling
1         Week   n/a      n/a
2     HomeTeam   n/a      n/a
3     AwayTeam   n/a      n/a
4        Total   n/a      n/a
5    H.RushAtt   n/a      n/a
6  H.RushYards   n/a      n/a
7  H.PassYards   n/a      n/a
8   H.Turnover   n/a      n/a
9      H.Score   n/a      n/a
10   A.RushAtt   n/a      n/a
11 A.RushYards   n/a      n/a
12 A.PassYards   n/a      n/a
13  A.Turnover   n/a      n/a
14     A.Score   n/a      n/a
15      Result   n/a      n/a
```

```
id.issues(gs_19)
```

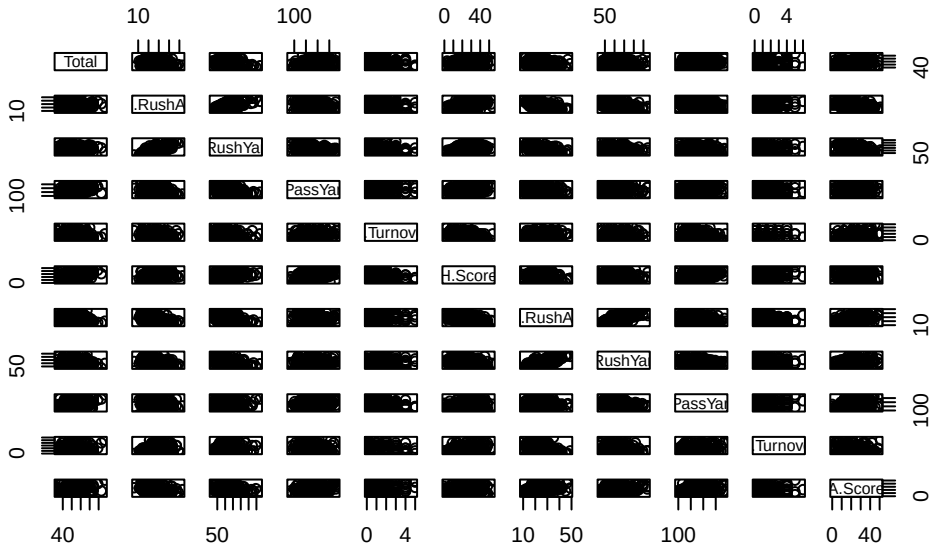
```
      features issue handling
1         Week   n/a      n/a
2     HomeTeam   n/a      n/a
3     AwayTeam   n/a      n/a
4        Total   n/a      n/a
5    H.RushAtt   n/a      n/a
6  H.RushYards   n/a      n/a
7  H.PassYards   n/a      n/a
8   H.Turnover   n/a      n/a
9      H.Score   n/a      n/a
10   A.RushAtt   n/a      n/a
11 A.RushYards   n/a      n/a
12 A.PassYards   n/a      n/a
13  A.Turnover   n/a      n/a
14     A.Score   n/a      n/a
15      Result   n/a      n/a
```

4 Scatterplot Matrix

Construct the **scatterplot matrix** for the continuous features and comment on what you observed.

Scatter plot matrix for game_scores_18.csv

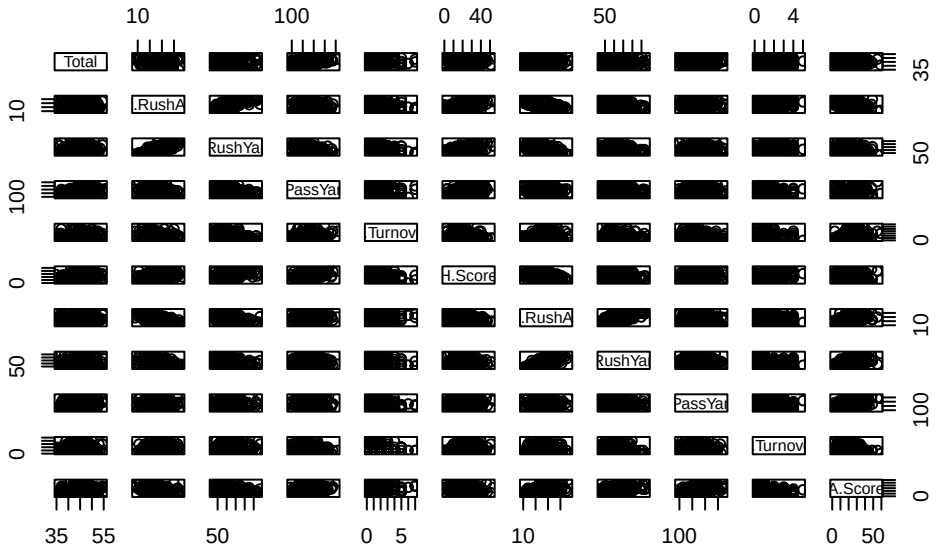
```
plot(con_18)
```



It is extremely hard to gather anything from this

Scatter plot matrix for game_scores_2019.csv

```
plot(con_19)
```



Same with this unfortunately

5 Visualizing Pairs of Categorical Features

Use multiple **barplot visualizations**.

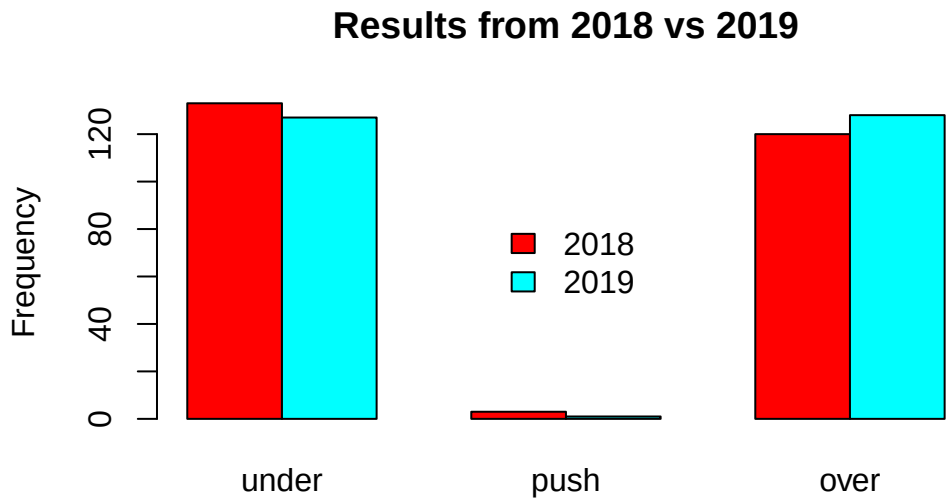
Lets make a quick plot to view the Results from 2018 vs 2019, keep in mind that result can be three different values -1 if the final score was lower than the total (under), 0 if the final score was the same as the total (push), and 1 if the final score was higher than the total (over)

```
tab <- table(cat_18$Result)
tab2 <- table(cat_19$Result)
x <- as.data.frame(tab)
y <- as.data.frame(tab2)
under <- c(x[1,2],y[1,2])
push <- c(x[2,2], y[2,2])
over <- c(x[3,2], y[3,2])
df <- data.frame(under, push, over)
print(df)
```

```
  under push over
1   133    3  120
```

```
2    127     1   128
```

```
barplot(height=as.matrix(df), beside=TRUE, main="Results from 2018 vs 2019", ylab="Frequency"
legend("center", c("2018", "2019"), cex=1.0, bty="n", fill=rainbow(2))
```



We can see here the under was more frequent in 2018 than 2019 and the inverse is true for the over

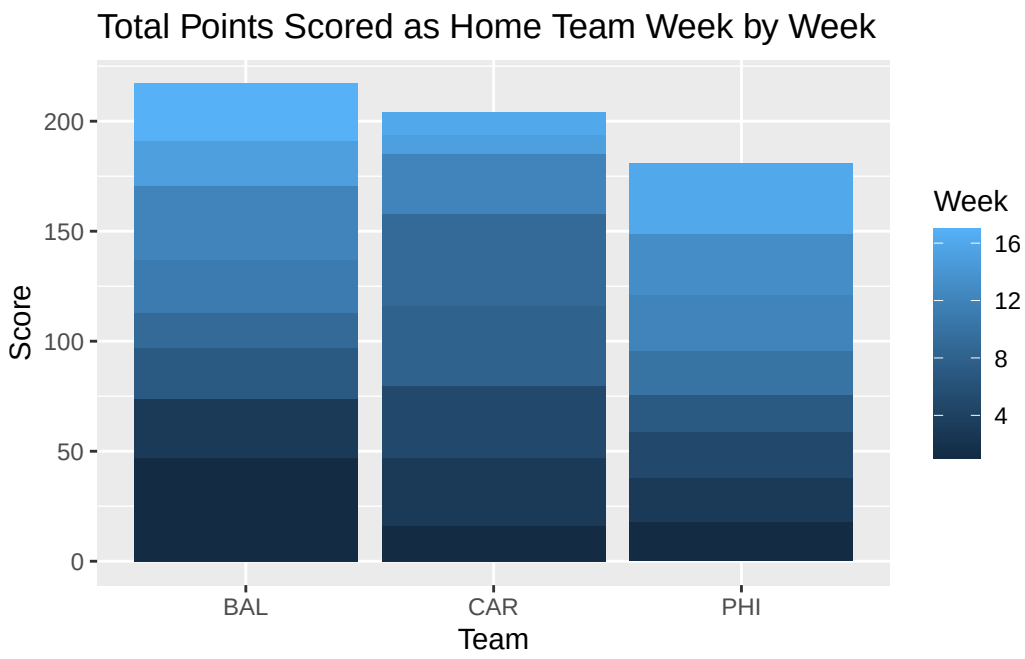
None of my other categorical values lean themselves towards bar-plots as they really don't carry statistical meaning (each of the 32 teams appears 8 times in home team and 8 times in away team, each of the 17 weeks appears either 16 times or slightly less)

6 Visualizing Relationship Between a Catergorical and Continuous Feature

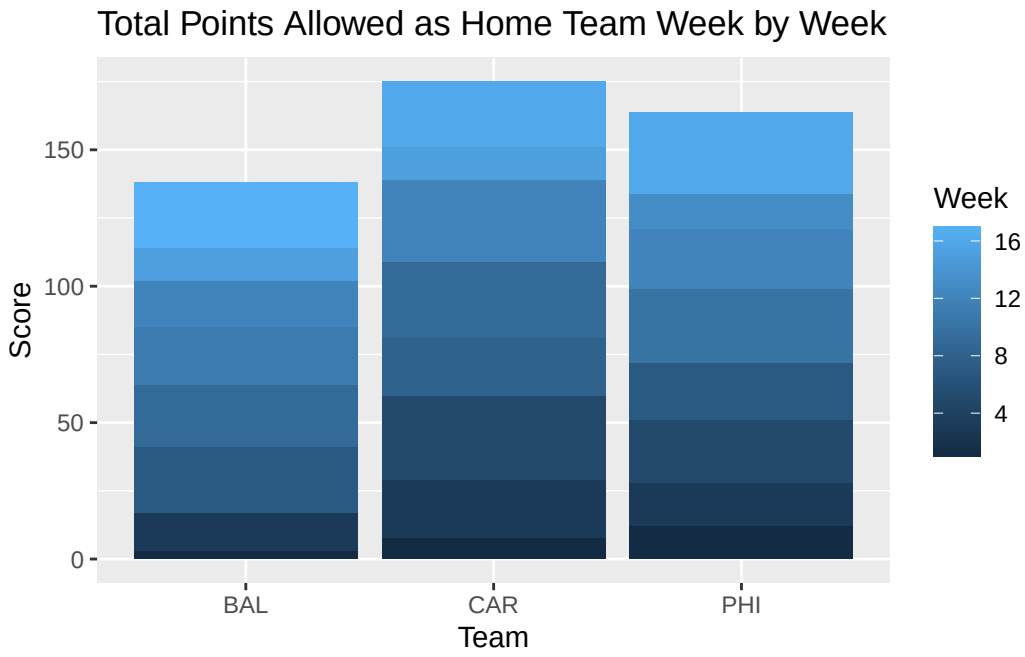
For a subset of the categorical and continuous features, perform **stacked bar-plot visualizations**. Comment on what you observed.

Lets analyze three teams home performances throughout the season, first we can compare their total home points scored and allowed week by week

```
df <- gs_18[gs_18$HomeTeam %in% c('CAR','PHI','BAL'), ]
df <- df %>% arrange(Week)
ggplot(df, aes(fill=Week, y=H.Score, x=HomeTeam)) +
  geom_bar(position='stack', stat='identity') +
  labs(x='Team',y='Score',title='Total Points Scored as Home Team Week by Week')
```

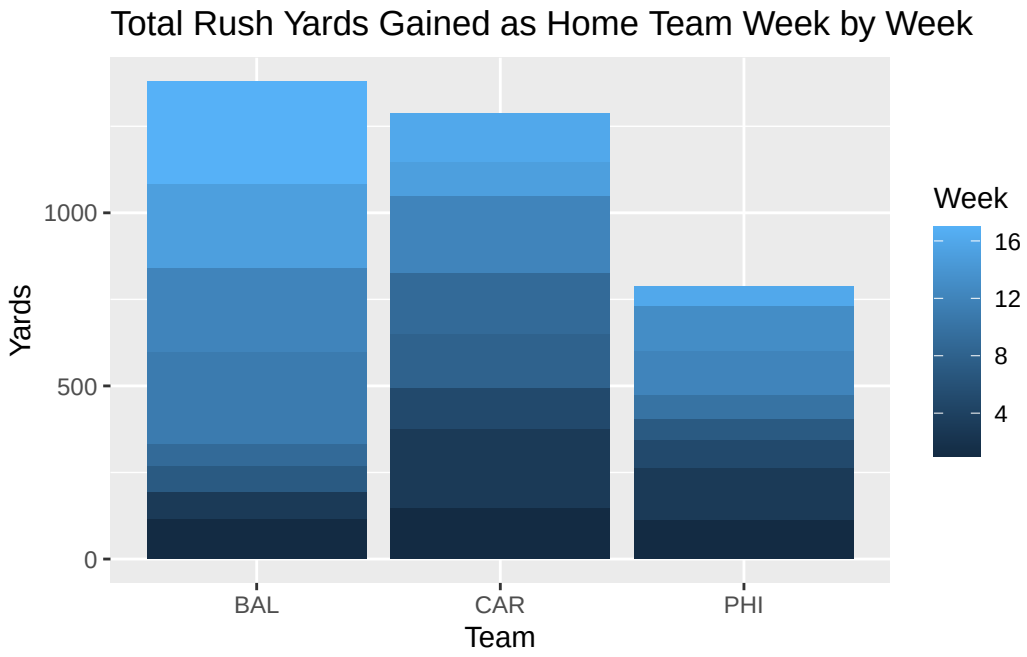



```
ggplot(df, aes(fill=Week, y=A.Score, x=HomeTeam)) +
  geom_bar(position='stack', stat='identity') +
  labs(x='Team',y='Score',title='Total Points Allowed as Home Team Week by Week')
```

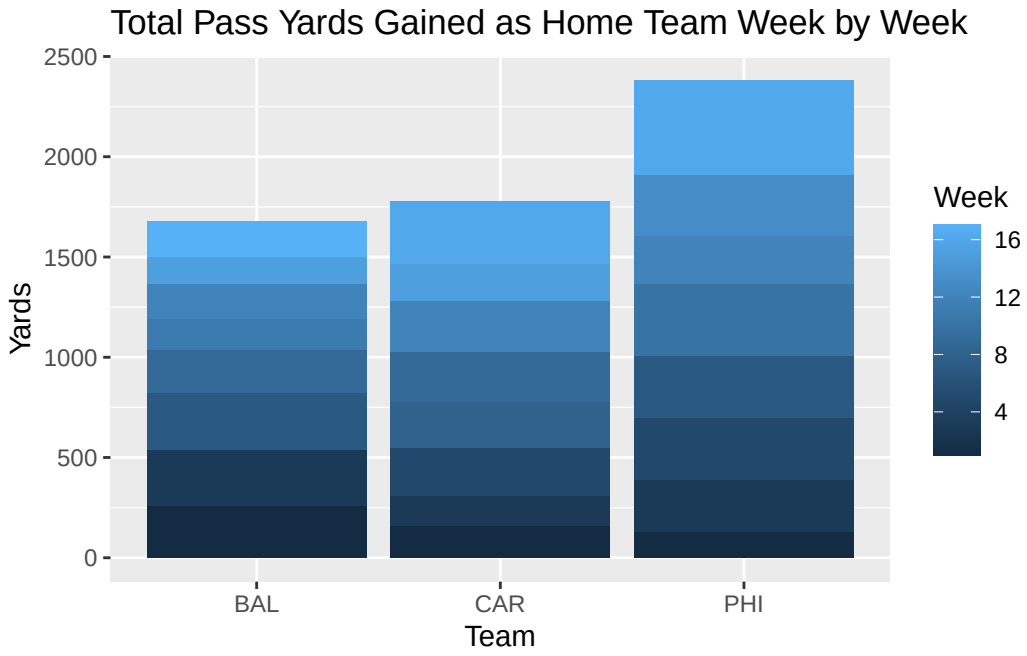


Baltimore has the advantage both offensively and defensively, usually a team with a good running attack helps their defense, lets look into each teams offensive stats and see if this holds true.

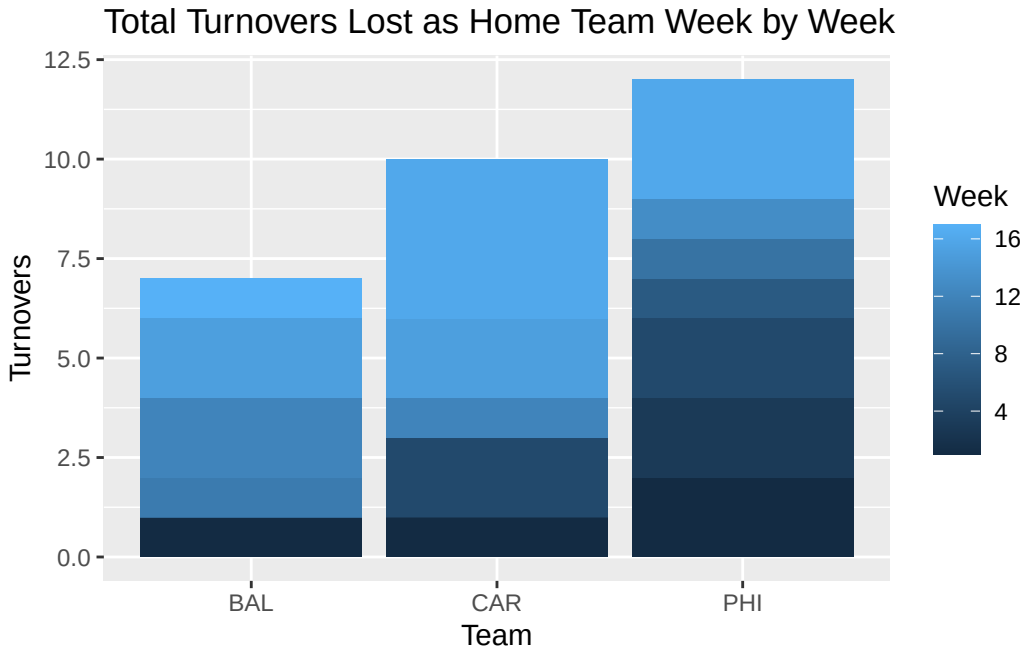
```
ggplot(df, aes(fill=Week, y=H.RushYards, x=HomeTeam)) +
  geom_bar(position='stack', stat='identity') +
  labs(x='Team',y='Yards',title='Total Rush Yards Gained as Home Team Week by Week')
```




```
ggplot(df, aes(fill=Week, y=H.PassYards, x=HomeTeam)) +
  geom_bar(position='stack', stat='identity') +
  labs(x='Team',y='Yards',title='Total Pass Yards Gained as Home Team Week by Week')
```

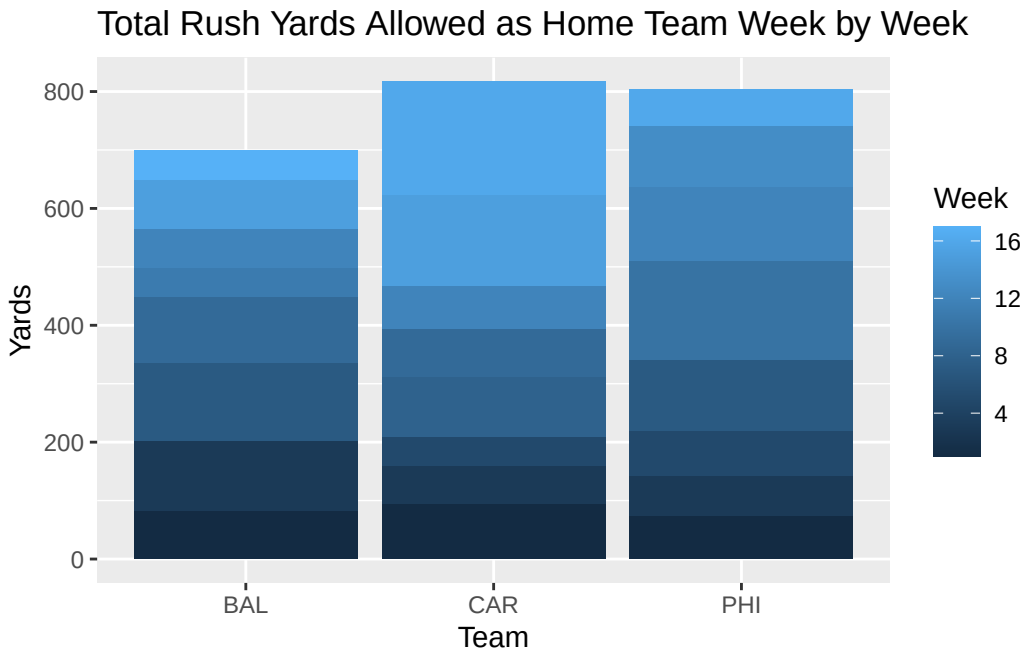


```
ggplot(df, aes(fill=Week, y=H.Turnover, x=HomeTeam)) +
  geom_bar(position='stack', stat='identity') +
  labs(x='Team',y='Turnovers',title='Total Turnovers Lost as Home Team Week by Week')
```

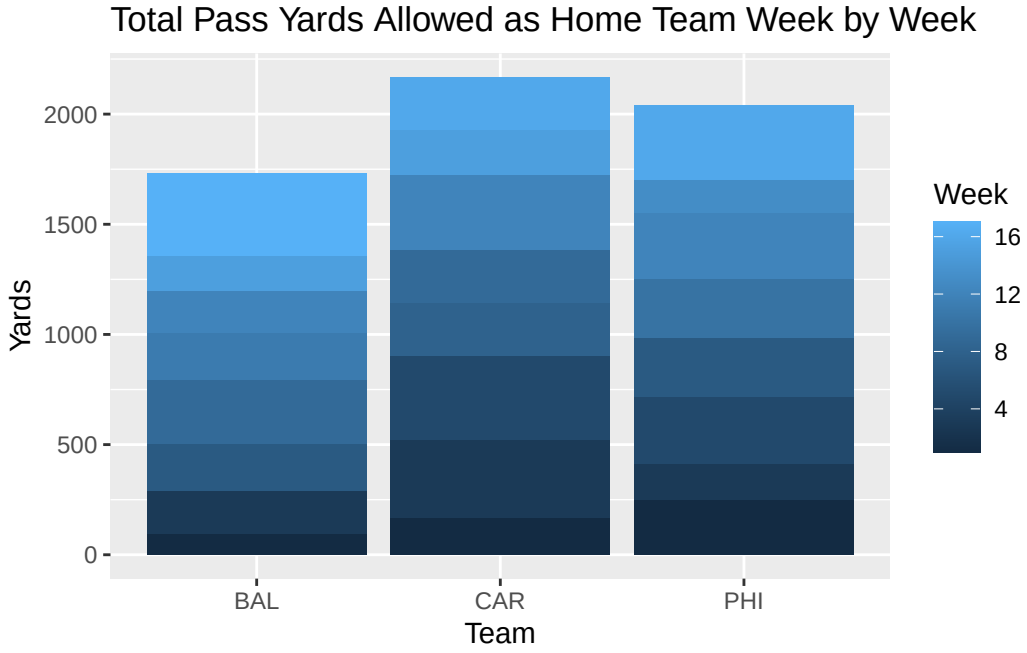


Interestingly our hypothesis was somewhat true as Baltimore did have the best rush attack as we predicted yet Carolina, who had an awful defense, has a similarly good run attack. To best explain this difference we can simply look at the amount of turnovers each team lost and see that minimizing turnovers may have big impacs on defensive performance. Next lets look deeper at each teams defensive stats

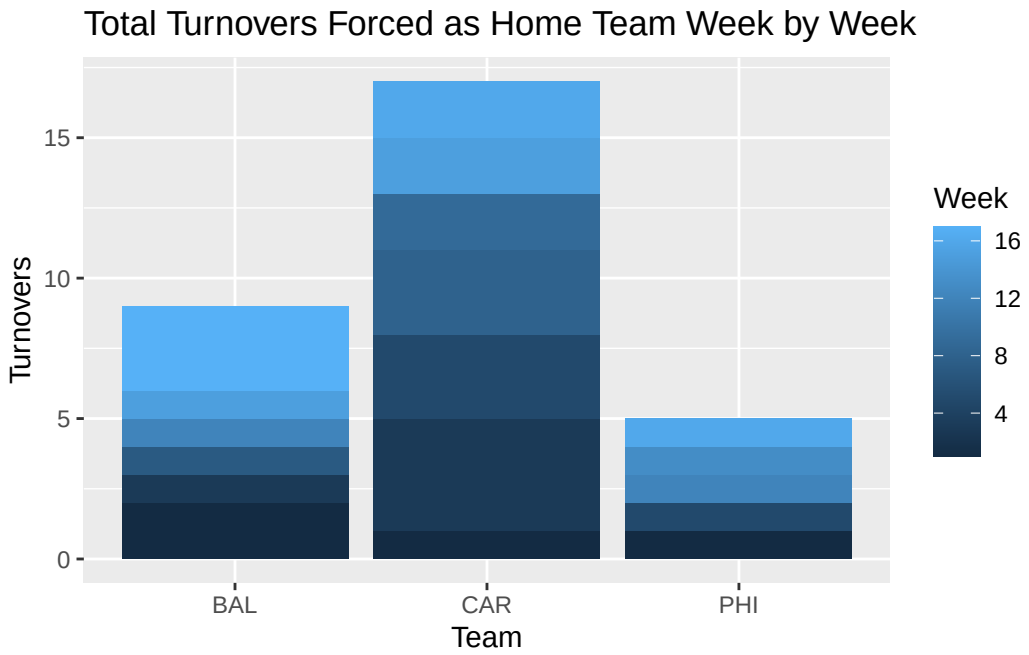
```
ggplot(df, aes(fill=Week, y=A.RushYards, x=HomeTeam)) +
  geom_bar(position='stack', stat='identity') +
  labs(x='Team',y='Yards',title='Total Rush Yards Allowed as Home Team Week by Week')
```




```
ggplot(df, aes(fill=Week, y=A.PassYards, x=HomeTeam)) +
  geom_bar(position='stack', stat='identity') +
  labs(x='Team',y='Yards',title='Total Pass Yards Allowed as Home Team Week by Week')
```



```
ggplot(df, aes(fill=Week, y=A.Turnover, x=HomeTeam)) +
  geom_bar(position='stack', stat='identity') +
  labs(x='Team',y='Turnovers',title='Total Turnovers Forced as Home Team Week by Week')
```

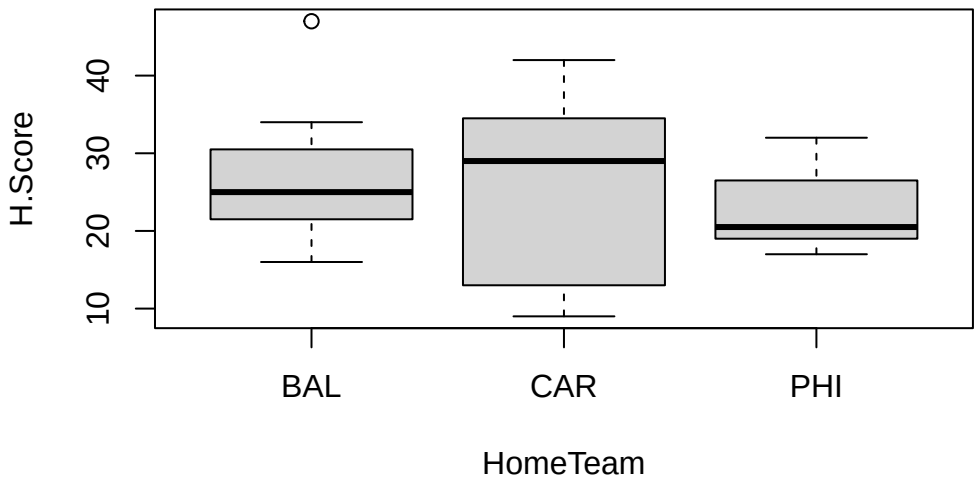


These follow what would expect except for the Turnovers. Carolina seems to have had a major outlier year in terms of forcing turnovers, its odd, and a bit depressing if you're a Carolina fan, that these huge turnover numbers did not translate better in total defensive and offensive statistics

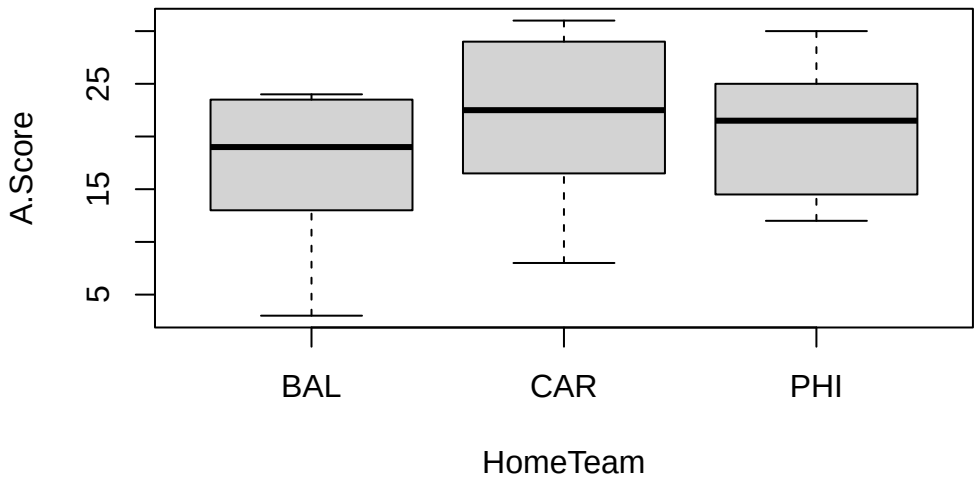
7 Boxplot Visualizations

For a subset of the categorical and continuous features, perform **boxplot visualizations**. Comment on what you observed.

```
boxplot(H.Score~HomeTeam, data=df)
```




```
boxplot(A.Score~HomeTeam, data=df)
```



8 Covariance Matrix

For the continuous features, construct the **covariance matrix**. Comment on what you observed.

```
cov(con_18)
```

```
                 Total   H.RushAtt H.RushYards  H.PassYards  H.Turnover
Total        24.2252451   -3.307230   -20.24963   158.044118  -0.6723039
H.RushAtt    -3.3072304   62.625919   338.08683  -151.391667  -1.3476716
H.RushYards -20.2496324  338.086826  2977.27102 -1381.192647  -8.7562500
H.PassYards 158.0441176 -151.391667 -1381.19265  6743.752941   9.9578431
H.Turnover   -0.6723039   -1.347672    -8.75625     9.957843   1.3409314
H.Score      17.9344363   34.049203   224.79688   330.045588  -4.1893382
A.RushAtt    -6.2991422  -34.772733  -171.46158   -63.676961   1.4128676
A.RushYards  -5.3537377 -152.912714  -750.32601   -68.345343   2.5488358
A.PassYards 152.9261642  -82.949724  -112.14782  1560.162500 -19.6134191
A.Turnover    0.3482843    3.272426    13.69816    -8.383333  -0.1365196
A.Score      11.1057598  -31.982047  -138.92286   225.217157   2.5432598
                H.Score   A.RushAtt A.RushYards A.PassYards A.Turnover
Total         17.934436   -6.299142   -5.353738   152.92616  0.3482843
H.RushAtt     34.049203  -34.772733 -152.912714   -82.94972  3.2724265
H.RushYards  224.796875 -171.461581 -750.326011  -112.14782 13.6981618
H.PassYards  330.045588  -63.676961  -68.345343  1560.16250 -8.3833333
H.Turnover    -4.189338    1.412868    2.548836   -19.61342 -0.1365196
H.Score      112.985233  -33.485233 -134.909651   270.14844  4.5150735
A.RushAtt    -33.485233   55.428370  269.602788  -140.84746 -2.4680147
A.RushYards -134.909651  269.602788 2390.906235  -831.72330 -8.7827819
A.PassYards  270.148438 -140.847457 -831.723300  6789.26187 19.5863358
A.Turnover     4.515074   -2.468015   -8.782782    19.58634  1.5977941
A.Score        2.499571   27.588664  173.897641   357.52387 -2.2905637
                A.Score
Total         11.105760
H.RushAtt    -31.982047
H.RushYards -138.922855
H.PassYards  225.217157
H.Turnover     2.543260
H.Score        2.499571
A.RushAtt     27.588664
A.RushYards  173.897641
A.PassYards  357.523866
A.Turnover    -2.290564
```

```
A.Score       98.341115
```

```
print('\n')
```

```
[1] "\n"
```

```
cov(con_19)
```

```
                Total   H.RushAtt  H.RushYards H.PassYards   H.Turnover
Total       15.81029029   -3.477191     5.096883    99.84066   0.29313725
H.RushAtt   -3.47719056   55.488174   295.982567   -96.17099  -2.78823529
H.RushYards  5.09688266  295.982567  2825.538955 -1021.68243 -12.75882353
H.PassYards 99.84066330  -96.170987 -1021.682430  6014.73712  13.29215686
H.Turnover   0.29313725   -2.788235   -12.758824    13.29216   1.85882353
H.Score      5.20344669   31.824449   225.260509   280.01480  -3.78431373
A.RushAtt    0.33789062  -36.178523  -175.413894   -55.93246   2.66470588
A.RushYards -5.93865656 -156.004687  -693.378202  -236.85253   5.16666667
A.PassYards 58.11951593  -64.174142   -91.282659   633.89357 -14.78431373
A.Turnover  -0.08240656    2.558548    11.789691   -19.40915  -0.06078431
A.Score      8.93258272  -35.835233  -151.642371   190.52295   4.94509804
               H.Score    A.RushAtt A.RushYards A.PassYards   A.Turnover
Total         5.203447    0.3378906   -5.938657   58.119516  -0.08240656
H.RushAtt    31.824449  -36.1785233 -156.004687  -64.174142   2.55854779
H.RushYards 225.260509 -175.4138940 -693.378202  -91.282659  11.78969056
H.PassYards 280.014798  -55.9324602 -236.852528  633.893566 -19.40914522
H.Turnover   -3.784314    2.6647059    5.166667  -14.784314  -0.06078431
H.Score      98.231311  -28.4192096  -89.687531  119.714093   2.88354779
A.RushAtt   -28.419210   53.1238817  287.509145  -42.404350  -1.85188419
A.RushYards -89.687531  287.5091452 2730.857583 -482.641483  -9.00589767
A.PassYards 119.714093  -42.4043505 -482.641483 5633.425245  -4.17383578
A.Turnover    2.883548   -1.8518842   -9.005898   -4.173836   1.37768076
A.Score      -6.759743   35.8265625  215.278217  353.970221  -3.92031250
               A.Score
Total         8.932583
H.RushAtt    -35.835233
H.RushYards -151.642371
H.PassYards  190.522947
H.Turnover     4.945098
H.Score       -6.759743
A.RushAtt     35.826563
A.RushYards  215.278217
```

```
A.PassYards  353.970221
A.Turnover    -3.920313
A.Score      108.464645
```

9 Correlation Matrix

For the continuous features, construct the **correlation matrix**. Comment on what you observed.

```
cor(con_18)
```

```
                Total   H.RushAtt H.RushYards H.PassYards  H.Turnover
Total        1.00000000 -0.08490892 -0.07540044  0.39101500 -0.11795824
H.RushAtt   -0.08490892  1.00000000  0.78296438 -0.23295610 -0.14706311
H.RushYards -0.07540044  0.78296438  1.00000000 -0.30824373 -0.13858164
H.PassYards  0.39101500 -0.23295610 -0.30824373  1.00000000  0.10471555
H.Turnover  -0.11795824 -0.14706311 -0.13858164  0.10471555  1.00000000
H.Score      0.34280157  0.40478001  0.38758793  0.37810486 -0.34035435
A.RushAtt   -0.17190213 -0.59019525 -0.42207690 -0.10415148  0.16388234
A.RushYards -0.02224548 -0.39517118 -0.28122864 -0.01702068  0.04501501
A.PassYards  0.37708254 -0.12721157 -0.02494425  0.23057262 -0.20556007
A.Turnover   0.05598084  0.32713908  0.19860615 -0.08076171 -0.09326772
A.Score      0.22753417 -0.40753192 -0.25674202  0.27655600  0.22147254
                H.Score  A.RushAtt A.RushYards A.PassYards  A.Turnover
Total        0.34280157 -0.1719021 -0.02224548  0.37708254  0.05598084
H.RushAtt    0.40478001 -0.5901953 -0.39517118 -0.12721157  0.32713908
H.RushYards  0.38758793 -0.4220769 -0.28122864 -0.02494425  0.19860615
H.PassYards  0.37810486 -0.1041515 -0.01702068  0.23057262 -0.08076171
H.Turnover  -0.34035435  0.1638823  0.04501501 -0.20556007 -0.09326772
H.Score      1.00000000 -0.4231325 -0.25956785  0.30844685  0.33604201
A.RushAtt   -0.42313254  1.0000000  0.74058865 -0.22959989 -0.26225349
A.RushYards -0.25956785  0.7405887  1.00000000 -0.20643642 -0.14209880
A.PassYards  0.30844685 -0.2295999 -0.20643642  1.00000000  0.18805354
A.Turnover   0.33604201 -0.2622535 -0.14209880  0.18805354  1.00000000
A.Score      0.02371303  0.3736776  0.35862854  0.43754849 -0.18273191
               A.Score
Total        0.22753417
H.RushAtt   -0.40753192
H.RushYards -0.25674202
H.PassYards  0.27655600
H.Turnover   0.22147254
```

```
H.Score      0.02371303
A.RushAtt    0.37367759
A.RushYards  0.35862854
A.PassYards  0.43754849
A.Turnover  -0.18273191
A.Score      1.00000000
```

```
print('/n')
```

```
[1] "/n"
```

```
cor(con_19)
```

```
                Total  H.RushAtt H.RushYards H.PassYards  H.Turnover
Total        1.00000000 -0.1173974  0.02411482  0.32376449  0.05407319
H.RushAtt   -0.11739737  1.0000000  0.74750700 -0.16646983 -0.27454284
H.RushYards  0.02411482  0.7475070  1.00000000 -0.24783175 -0.17605197
H.PassYards  0.32376449 -0.1664698 -0.24783175  1.00000000  0.12570949
H.Turnover   0.05407319 -0.2745428 -0.17605197  0.12570949  1.00000000
H.Score      0.13203717  0.4310580  0.42757218  0.36429037 -0.28005484
A.RushAtt    0.01165900 -0.6663551 -0.45276040 -0.09894884  0.26815463
A.RushYards -0.02858043 -0.4007633 -0.24961461 -0.05844134  0.07251734
A.PassYards  0.19474477 -0.1147819 -0.02287975  0.10889846 -0.14447604
A.Turnover  -0.01765703  0.2926301  0.18896330 -0.21321796 -0.03798378
A.Score      0.21570640 -0.4619191 -0.27392131  0.23588204  0.34826652
               H.Score   A.RushAtt A.RushYards A.PassYards  A.Turnover
Total        0.1320372  0.01165900 -0.02858043  0.19474477 -0.01765703
H.RushAtt    0.4310580 -0.66635509 -0.40076326 -0.11478195  0.29263012
H.RushYards  0.4275722 -0.45276040 -0.24961461 -0.02287975  0.18896330
H.PassYards  0.3642904 -0.09894884 -0.05844134  0.10889846 -0.21321796
H.Turnover  -0.2800548  0.26815463  0.07251734 -0.14447604 -0.03798378
H.Score      1.0000000 -0.39340704 -0.17316398  0.16092890  0.24787224
A.RushAtt   -0.3934070  1.00000000  0.75484405 -0.07751384 -0.21646846
A.RushYards -0.1731640  0.75484405  1.00000000 -0.12305205 -0.14682607
A.PassYards  0.1609289 -0.07751384 -0.12305205  1.00000000 -0.04737779
A.Turnover   0.2478722 -0.21646846 -0.14682607 -0.04737779  1.00000000
A.Score     -0.0654879  0.47197190  0.39555462  0.45283122 -0.32070246
               A.Score
Total        0.2157064
H.RushAtt   -0.4619191
H.RushYards -0.2739213
```

```
H.PassYards  0.2358820
H.Turnover   0.3482665
H.Score     -0.0654879
A.RushAtt    0.4719719
A.RushYards  0.3955546
A.PassYards  0.4528312
A.Turnover  -0.3207025
A.Score      1.0000000
```

10 Range Normalization

List the continuous features that require **range normalization**. What is the rationale for your selection? Perform the range normalization and show the values before and after the normalization.

11 Binning

Do you see the need for converting a subset of the continuous features into categorical features? Select two such continuous features and convert the first into a categorical feature using the **equal-width binning*** and the second using **equal-frequency binning**. Show the feature values after the equal-width and equal-frequency binning.

12 Undersampling

Do you see a need for undersampling? **Undersampling** is used to reduce the instances from the majority class so that the final dataset is balanced. For example, a binary classification problem has a target/outcome variable that takes two values, say, *approved* and *denied*. In the dataset, if 70% of the instances have the *approved* value for the target variable, the dataset is *imbalanced*. Ideally, the dataset should have approximately equal number of instances for each the values the target variable takes. This article illustrates the undersampling.

No

13 Oversampling

Oversampling arises when we have too few instances from a class (called the minority class) relative to other classes. To boost the participation of the minority class in the (training)

dataset, more observations from the minority class are generated usually by replicating the samples from the minority class.

In your dataset, do you see the need for oversampling? If so, which features require oversampling?

No

14 Summary

Summarize the findings you have discovered through the exploratory data analysis. The summary should about a page and should serve as an executive report for non-technical people.