STATS 506 Problem Set #1

Haiming Li

Wine Data

a. Data loading.

```
wine_df <- read.csv('./wine/wine.data', header=FALSE)
names(wine_df) <- c('class', 'alcohol', 'malic_acid', 'ash',
                    'alcalinity_of_ash', 'magnesium', 'total_phenols',
                    'flavanoids', 'nonflavanoid_phenols', 'proanthocyanins',
                    'color_intensity', 'hue', 'od280_od315_ratio', 'proline')
```

b. Class count seems to be consistent.

```
table(wine_df$class)
```

```
 1  2  3
59 71 48
```

c. Here are the answers.

1. Correlation is 0.5463642.

```
cor(wine_df$alcohol, wine_df$color_intensity)
```

```
[1] 0.5463642
```

2. It seems class 1 have the highest correlation and class 2 have the lowest.

```
splited_df <- split(wine_df, wine_df$class)
sapply(splited_df, function(df) cor(df$alcohol, df$color_intensity))
```

```
        1         2         3 
0.4082913 0.2697891 0.3503777 
```

3. The alcohol content of the wine with the highest color intensity is 14.34.

```r
wine_df$alcohol[which.max(wine_df$color_intensity)]
```

```
[1] 14.34
```

4. 8.43% of wines had a higher content of proanthocyanins compare to ash.

```r
mean(wine_df$proanthocyanins > wine_df$ash)
```

```
[1] 0.08426966
```

d. Here's the table. Notice that here I used a previously defined variable that splits the dataframe based on class.

```r
res <- data.frame(
  overall=colMeans(wine_df[,-1]),
  class1=colMeans(splited_df$"1"[,-1]),
  class2=colMeans(splited_df$"2"[,-1]),
  class3=colMeans(splited_df$"3"[,-1])
)
data.frame(t(res))
```

```
         alcohol malic_acid      ash alcalinity_of_ash magnesium total_phenols
overall 13.00062   2.336348 2.366517          19.49494  99.74157      2.295112
class1  13.74475   2.010678 2.455593          17.03729 106.33898      2.840169
class2  12.27873   1.932676 2.244789          20.23803  94.54930      2.258873
class3  13.15375   3.333750 2.437083          21.41667  99.31250      1.678750
        flavanoids nonflavanoid_phenols proanthocyanins color_intensity
overall  2.0292697            0.3618539        1.590899        5.058090
class1   2.9823729            0.2900000        1.899322        5.528305
class2   2.0808451            0.3636620        1.630282        3.086620
class3   0.7814583            0.4475000        1.153542        7.396250
              hue od280_od315_ratio   proline
overall 0.9574494          2.611685  746.8933
class1  1.0620339          3.157797 1115.7119
class2  1.0562817          2.785352  519.5070
class3  0.6827083          1.683542  629.8958
```

e. Here we can see all p-values are less than 0.05, thus we can reject the null hypothesis. The phenols level appears to be different across classes.

```r
# class 1 v.s. class 2
t.test(splited_df$"1"$total_phenols, splited_df$"2"$total_phenols)
```

```

	Welch Two Sample t-test

data:  splited_df$"1"$total_phenols and splited_df$"2"$total_phenols
t = 7.4206, df = 119.14, p-value = 1.889e-11
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 0.4261870 0.7364055
sample estimates:
mean of x mean of y 
 2.840169  2.258873 
```

```r
# class 1 v.s. class 3
t.test(splited_df$"1"$total_phenols, splited_df$"3"$total_phenols)
```

```

	Welch Two Sample t-test

data:  splited_df$"1"$total_phenols and splited_df$"3"$total_phenols
t = 17.12, df = 98.356, p-value < 2.2e-16
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 1.026801 1.296038
sample estimates:
mean of x mean of y 
 2.840169  1.678750 
```

```r
# class 2 v.s. class 3
t.test(splited_df$"2"$total_phenols, splited_df$"3"$total_phenols)
```

```

	Welch Two Sample t-test

data:  splited_df$"2"$total_phenols and splited_df$"3"$total_phenols
```

```
t = 7.0125, df = 116.91, p-value = 1.622e-10
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 0.4162855 0.7439610
sample estimates:
mean of x mean of y
 2.258873  1.678750
```

AskAManager.org Data

a. Data loading. (I removed the useless index column)

```r
df <- read.csv("AskAManager.csv")[-1]
```

b. Column cleaning.

```r
names(df) <- c('timestamp', 'age', 'industry', 'job_title', 'job_title_context',
               'salary', 'additional_income', 'income_currency', 'other_currency',
               'income_context', 'country', 'state', 'city', 'yoe_total', 'yoe_in_field',
               'highest_education', 'gender', 'race')
```

c. Filter by income currency.

```r
# make necessary columns as factor
df$income_currency <- as.factor(df$income_currency)
print(paste('Obs before:', dim(df)[1]))
```

```
[1] "Obs before: 28062"
```

```r
df <- subset(df, income_currency == 'USD')
print(paste('Obs after: ', dim(df)[1]))
```

```
[1] "Obs after:  23374"
```

d. Remove impossible working experience. My filtering logic that that total year of experience should be at least as much as year of experience in the field. For people in age range 18 ~ 24, it is impossible to gain total yoe beyond 7 years; for people in age range 25 ~ 34, it is impossible to gain total yoe beyond 16 years; for people in age range 35 ~ 44, it is impossible to gain total yoe beyond 26 years; for people in age range 45 ~ 54, it is impossible to gain total yoe beyond 36 years.

```r
# convert necessary columns to factor
yoe_levels <- c("1 year or less", "2 - 4 years", "5-7 years", "8 - 10 years",
                "11 - 20 years", "21 - 30 years", "31 - 40 years", "41 years or more")
df$age <- as.factor(df$age)
df$yoe_total <- factor(df$yoe_total, yoe_levels, ordered = TRUE)
df$yoe_in_field <- factor(df$yoe_in_field, yoe_levels, ordered = TRUE)

#' Check if a given row of input has valid age & work experience
#' @param  age age in a row of data
#' @param  yoe_total total year of experience in a row of data
#' @param  yoe_in_field year of experience in field in a row of data
#' @return A logical value indicate whether the row of data is valid
isValid <- function(age, yoe_total, yoe_in_field) {
  if (age == "under 18") {
    return(FALSE)
  }
  if (yoe_in_field > yoe_total) {
    return(FALSE)
  }
  if (age == "18-24" & yoe_total > "5-7 years") {
    return(FALSE)
  }
  if (age == "25-34" & yoe_total > "11 - 20 years") {
    return(FALSE)
  }
  if (age == "35-44" & yoe_total > "21 - 30 years") {
    return(FALSE)
  }
  if (age == "45-54" & yoe_total > "31 - 40 years") {
    return(FALSE)
  }
  return(TRUE)
}

res <- mapply(isValid, df$age, df$yoe_total, df$yoe_in_field)
print(paste('Obs before:', dim(df)[1]))
```

```
[1] "Obs before: 23374"
```

```r
df <- df[res,]
print(paste('Obs after: ', dim(df)[1]))
```

```
[1] "Obs after:  23116"
```

e. I will be using the IQR method to identify outliers. (see citation) This method might remove some realistic observations, but it will help stabilize the distribution of salary. This may help with modeling as some models are very sensitive to outliers. As shown by the histogram, the filtered salary is somewhat normally distributed, which can be a desirable property for certain types of model.

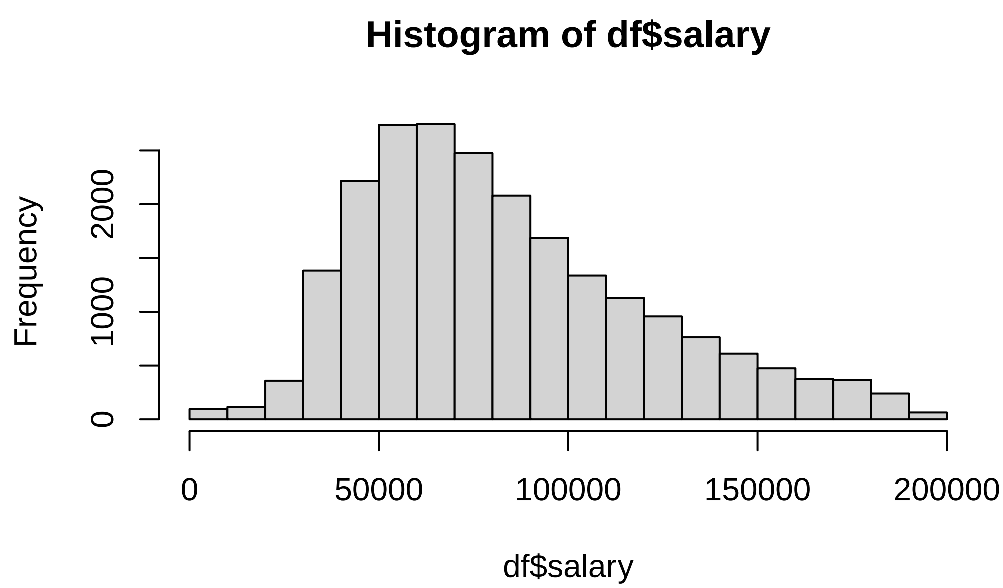
```r
q <- quantile(df$salary, c(0.25, 0.75))
iqr <- IQR(df$salary)
min_val <- q[1] - 1.5 * iqr
max_val <- q[2] + 1.5 * iqr

print(paste('Obs before:', dim(df)[1]))
```

```
[1] "Obs before: 23116"
```

```r
df <- subset(df, salary >= min_val & salary <= max_val)
print(paste('Obs after: ', dim(df)[1]))
```

```
[1] "Obs after:  22207"
```

```r
hist(df$salary)
```



Palindromic Numbers

a. Here's the function.

```r
#' Check if a number is palindromic
#' @param n Positive integer
#' @return A list with two elements:
#'         - isPalindromic: Boolean indicating if the input is palindromic
#'         - reversed: The input with its digits reversed
isPalindromic <- function(n) {
  # check invalid input
  if (!is.numeric(n) | !(all.equal(n, as.integer(n)) == TRUE) | n <= 0) {
    stop('Invalid input: positive integer only')
  }

  # convert to vector of character
  s <- strsplit(as.character(n), '')[[1]]
  # reverse the vector and convert back to number
  reverse_n <- as.numeric(paste(s[length(s):1], collapse = ''))

  return(list(isPalindromic=(all.equal(n, reverse_n) == TRUE), reversed=reverse_n))
}
```

b. Here's the function.

```r
#' Get next palindromic number that's strictly greater than input
#' @param n Positive integer
#' @return Next palindromic number greater than the input
nextPalindrome <- function(n) {
  # check invalid input
  if (!is.numeric(n) | !(all.equal(n, as.integer(n)) == TRUE) | n <= 0) {
    stop('Invalid input: positive integer only')
  }

  next_n <- n + 1
  while (!isPalindromic(next_n)$isPalindromic) {
    next_n <- next_n + 1
  }

  return(next_n)
}
```

c. Here are the results.

```r
# i
nextPalindrome(391)
```

```
[1] 393
```

```r
# ii
nextPalindrome(9928)
```

```
[1] 9999
```

```r
# iii
nextPalindrome(19272719)
```

```
[1] 19277291
```

```r
# iv
nextPalindrome(109)
```

```
[1] 111
```

```r
# v
nextPalindrome(2)
```

```
[1] 3
```

Citaton & Link to GitHub

- Use of ordered factor & mapply() in Q2d
- Outlier detection in Q2e
- GitHub Repo of this Pset