STA 141A - Fall 2024 - Homework 2

Instructor: Dr. Akira Horiguchi

Student name: Andrew Jowe; Student ID: 919586453

Due date: F Oct 11, 2024 at 07:59 PM (PT)

The assignment has to be done in an R Markdown document. The assignment has to be submitted electronically on Canvas by the due date above by uploading two files:

1. a .rmd or .qmd source file in CANVAS;
2. a .pdf file in GRADESCOPE (if you can knit/compile your .rmd to a .html file only, please save the created .html file as a .pdf file (by opening the .html file -> print -> save to .pdf)).

Email submissions will not be accepted.

Each answer has to be based on `R` code that shows how the result was obtained. The code has to answer the question or solve the task. For example, if you are asked to find the largest entry of a vector, the code has to return the largest element of the vector. If the code just prints all values of the vector, and you determine the largest element by hand, this will not be accepted as an answer. No points will be given for answers that are not based on `R`. This homework already contains chunks for your solution (you can also create additional chunks for each solution if needed, but it must be clear to which tasks your chunks belong).

There are many possible ways to write `R` code that is needed to answer the questions or do the tasks, but for some of the questions or tasks you might have to use something that has not been discussed during the lectures or the discussion sessions. You will have to come up with a solution on your own. Try to understand what you need to do to complete the task or to answer the question, feel free to search the Internet for possible solutions, and discuss possible solutions with other students. It is perfectly fine to ask what kind of an approach or a function other students use. However, you are not allowed to share your code or your answers with other students. Everyone has to write the code, do the tasks and answer the questions on their own.

During the discussion sessions, you may be asked to present and share your solutions.

Good luck!

1. Data exploration and manipulation (6 points)

The task is to explore the US census population estimates by county for 2022 from the package `usmap` (load the data frame from `countypop.RData`). The data frame has `3142` rows and `4` variables: `fips` is the 5-digit FIPS code corresponding to the county; `abbr` is the 2-letter state abbreviation; `county` is the full county name; `pop_2022` is the 2022 population estimate (in number of people) for the corresponding county. Each row of the data frame represents a different county or county equivalent. For the sake of simplicity, 'county' stands also for a county equivalent, and District of Columbia for a 'state'.

Without creating new functions, and without using `for` loops, answer the following questions (using `dplyr` is allowed).

a) (1 point) Remove all the rows that contain at least one `NA`.

```r
library(usmap)
library(dplyr)

cleaned_data <- countypop %>% filter(complete.cases(.))
knitr::kable(head(cleaned_data))
```

fips	abbr	county	pop_2022
01001	AL	Autauga County	59759
01003	AL	Baldwin County	246435
01005	AL	Barbour County	24706
01007	AL	Bibb County	22005
01009	AL	Blount County	59512
01011	AL	Bullock County	10202

b) (1 point) How many unique county names are there?

```r
unique_counties <- cleaned_data %>% summarise(unique_count = n_distinct(county))
knitr::kable(unique_counties)
```

unique_count
1960

c) (2 points) In order to answer the following question, you can combine the functions `lapply()`, `split()`, `order()`, and `tail()` (or `head()`): What is the largest county in terms of population of each of the states?

```r
largest_counties <- cleaned_data %>%
  split(.$abbr) %>%
  lapply(function(state_data) {
    state_data[order(state_data$pop_2022, decreasing = TRUE), ] %>%
      head(1)
  }) %>%
  bind_rows()

knitr::kable(largest_counties)
```

fips	abbr	county	pop_2022
02020	AK	Anchorage Municipality	287145
01073	AL	Jefferson County	665409
05119	AR	Pulaski County	399145
04013	AZ	Maricopa County	4551524
06037	CA	Los Angeles County	9721138
08041	CO	El Paso County	740567
09110	CT	Capitol Planning Region	981447
11001	DC	District of Columbia	671803
10003	DE	New Castle County	575494
12086	FL	Miami-Dade County	2673837
13121	GA	Fulton County	1074634
15003	HI	Honolulu County	995638
19153	IA	Polk County	501089
16001	ID	Ada County	518907
17031	IL	Cook County	5109292
18097	IN	Marion County	969466
20091	KS	Johnson County	619195
21111	KY	Jefferson County	773399
22033	LA	East Baton Rouge Parish	450544
25017	MA	Middlesex County	1617105
24031	MD	Montgomery County	1052521
23005	ME	Cumberland County	307451
26163	MI	Wayne County	1757043
27053	MN	Hennepin County	1260121
29189	MO	St. Louis County	990414
28049	MS	Hinds County	217730
30111	MT	Yellowstone County	169852
37183	NC	Wake County	1175021
38017	ND	Cass County	192734
31055	NE	Douglas County	586327
33011	NH	Hillsborough County	426594
34003	NJ	Bergen County	952997
35001	NM	Bernalillo County	672508
32003	NV	Clark County	2322985
36047	NY	Kings County	2590516
39049	OH	Franklin County	1321820
40109	OK	Oklahoma County	802559
41051	OR	Multnomah County	795083
42101	PA	Philadelphia County	1567258
72127	PR	San Juan Municipio	334776
44007	RI	Providence County	657288
45045	SC	Greenville County	547950
46099	SD	Minnehaha County	203971
47157	TN	Shelby County	916371
48201	TX	Harris County	4780913
49035	UT	Salt Lake County	1186257
51059	VA	Fairfax County	1138331
50007	VT	Chittenden County	169301
53033	WA	King County	2266789
55079	WI	Milwaukee County	918661
54039	WV	Kanawha County	175515
56021	WY	Laramie County	100723

d) (2 points) What is the average population of the 100 largest counties in the US?

```
avg_pop_100_largest <- cleaned_data %>%
  arrange(desc(pop_2022)) %>%
  slice_head(n = 100) %>%
  summarise(average_population = mean(pop_2022))

knitr::kable(avg_pop_100_largest)
```

average_population
1405817

2. Conditional and repetitive execution (8 points + 1 Bonus point)

a) (2 points) Define `x` as a random number between 1 and 100 (without replacement). By using `if`, `else if` and `else`, return the string " 'x' is very small!" if `x` is smaller than 10, return " 'x' is very large!" if `x` is larger than 90, return " 'x' is either small or large!" if `x` is at least 10 and at most 25, or at least 75 and at most 90, and return " 'x' is medium sized!" otherwise.

```
set.seed(123)
x = sample(1:100, 1)
if (x < 10) {
  print("'x' is very small!")
} else if (x > 90) {
  print("'x' is very large!")
} else if (x >= 10 && x <= 25 || x >= 75 && x <= 90) {
  print("'x' is either small or large!")
} else {
  print("'x' is medium sized!")
}
```

```
## [1] "'x' is medium sized!"
```

b) (2 points) By using a `for` loop calculate $\frac{1}{10}\sum_{i=3}^{12} 2^i$.

```
res = 0
for (i in 3:12) {
  res = res + 2 ** i
}
res = res / 10
res
```

```
## [1] 818.4
```

c) (2 points) By using `for` loops calculate $\frac{1}{1000}\sum_{i=2}^{9}\sum_{j=1}^{i} 4^{2i-3j}$.

```
res = 0
for (i in 2:9) {
  for (j in 1:i) {
    res = res + 4 ** (2 * i - 3 * j)
  }
}
res = res / 1000
res
```

```
## [1] 1163504
```

d) (2 points) Find the bug: The following `for` loop creates a vector that contains the sum of the first `n` numbers. In particular, if you set `n=10`, the `for` loop should return a vector of size `10` containing the values 1, (1+2), (1+2+3), . . . , (1+2+3+4+5+6+7+8+9+10). Explain why this `for` loop does not create the desired vector, and write the correct code.

Sums does not return a vector but rather a number because it is assigning the sum from 1 to i to the sums variable itself, not the actual element in the vector.

```
n=10
sums=numeric(n)
for(i in 1:n){
  sums[i]=sum(1:i)
}

sums
```

```
##  [1]  1  3  6 10 15 21 28 36 45 55
```

e) (1 Bonus point) Explain the following code in your own words:

```
n=10 # set n to 10
x=1:(2*n) # create a list from 1 to 20
# while the first element is less than n
while(x[1] < n){
    x=x[-1] # remove the first element from the list
}
x # print the list = a list from 10 to 20
```

3. Examining the data distribution (6 points)

Go to UCI Machine learning repository and download the data on the white wine quality. This page contains also the background information on the data. In our analysis, we will only consider the following variables: `pH` (pH level) and `quality` (wine quality with values between 0 and 10).

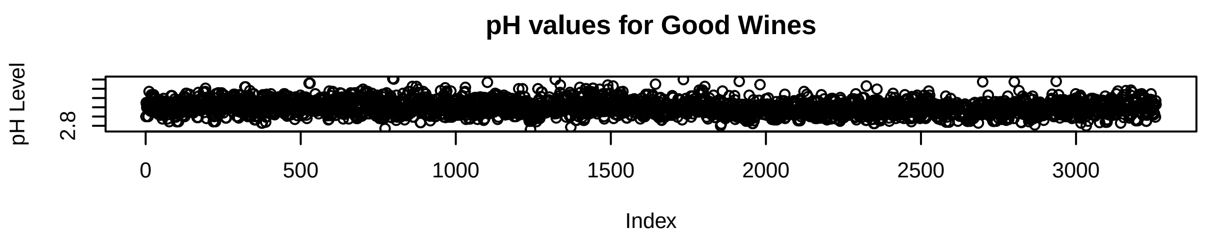
a) (2 points) Read the data into `R` and add a new binary variable with value 1 if the quality is greater than 5 (this is considered as `good` wine), and 0 otherwise (this is considered as `bad` wine).

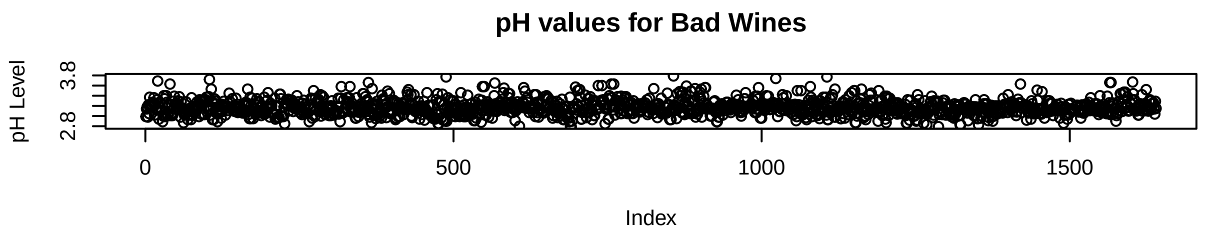
```
library(dplyr)

wine_data <- read.csv("winequality-white.csv", sep = ";")
wine_data <- wine_data %>%
  mutate(good_wine = ifelse(quality > 5, 1, 0))

str(wine_data)
```

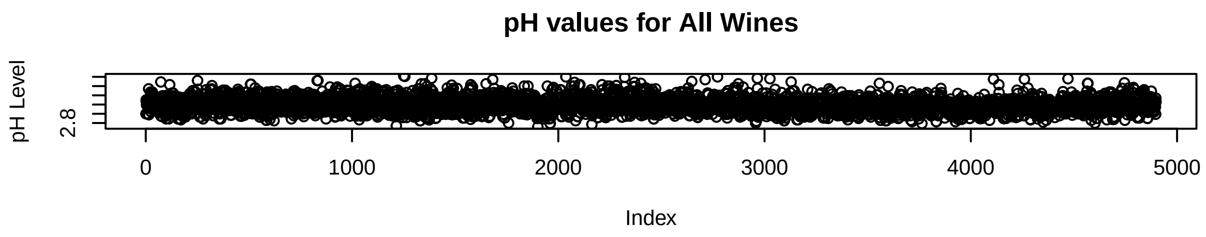
```
## 'data.frame':    4898 obs. of  13 variables:
##  $ fixed.acidity       : num  7 6.3 8.1 7.2 7.2 8.1 6.2 7 6.3 8.1 ...
##  $ volatile.acidity    : num  0.27 0.3 0.28 0.23 0.23 0.28 0.32 0.27 0.3 0.22 ...
##  $ citric.acid         : num  0.36 0.34 0.4 0.32 0.32 0.4 0.16 0.36 0.34 0.43 ...
##  $ residual.sugar      : num  20.7 1.6 6.9 8.5 8.5 6.9 7 20.7 1.6 1.5 ...
##  $ chlorides           : num  0.045 0.049 0.05 0.058 0.058 0.05 0.045 0.045 0.049 0.044 ...
##  $ free.sulfur.dioxide : num  45 14 30 47 47 30 30 45 14 28 ...
##  $ total.sulfur.dioxide: num  170 132 97 186 186 97 136 170 132 129 ...
##  $ density             : num  1.001 0.994 0.995 0.996 0.996 ...
##  $ pH                  : num  3 3.3 3.26 3.19 3.19 3.26 3.18 3 3.3 3.22 ...
##  $ sulphates           : num  0.45 0.49 0.44 0.4 0.4 0.44 0.47 0.45 0.49 0.45 ...
##  $ alcohol             : num  8.8 9.5 10.1 9.9 9.9 10.1 9.6 8.8 9.5 11 ...
##  $ quality             : int  6 6 6 6 6 6 6 6 6 6 ...
##  $ good_wine           : num  1 1 1 1 1 1 1 1 1 1 ...
```

b) (2 points (4*0.5)) Plot the `pH` values for `good`, `bad` and all wines separately in a 3x1-matrix. In addition, do another plot where you plot all the `pH` values and distinguish by color whether the wine is good or bad.

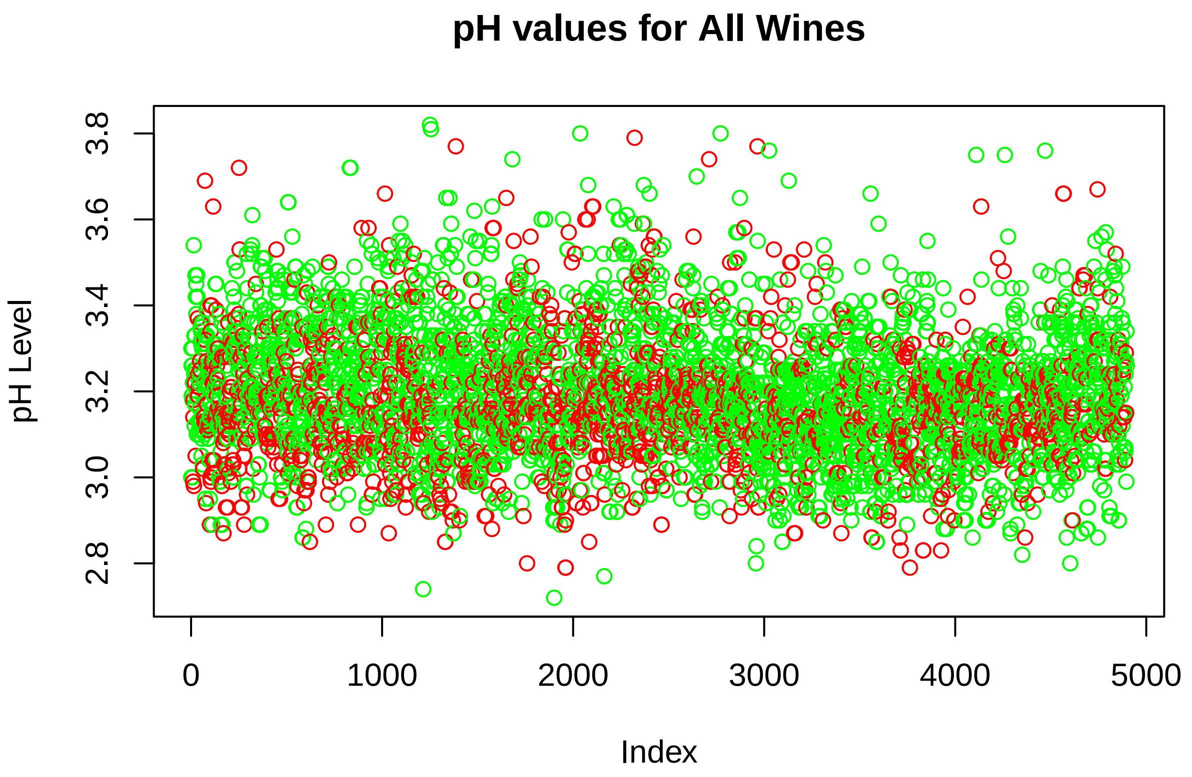
```
library(ggplot2)

par(mfrow = c(3, 1))
plot(wine_data$pH[wine_data$good_wine == 1],
     main = "pH values for Good Wines", xlab = "Index", ylab = "pH Level")
plot(wine_data$pH[wine_data$good_wine == 0],
     main = "pH values for Bad Wines", xlab = "Index", ylab = "pH Level")
plot(wine_data$pH, main = "pH values for All Wines",
     xlab = "Index", ylab = "pH Level")
```





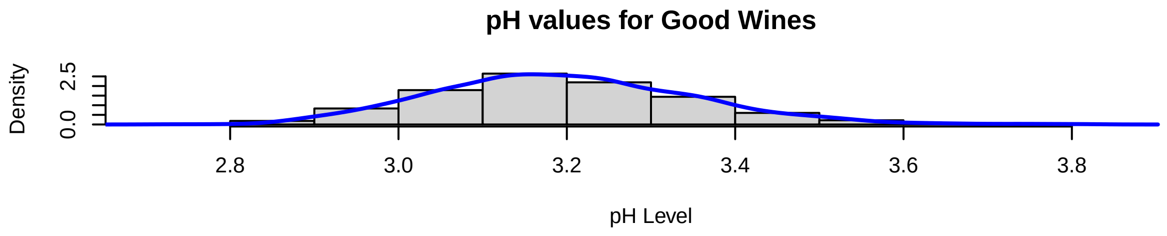


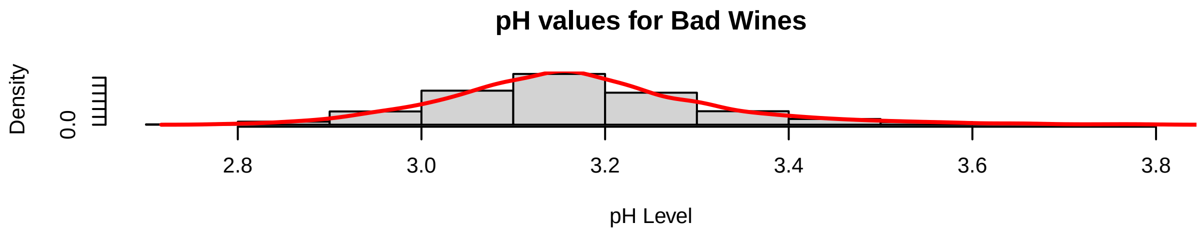
```
par(mfrow = c(1, 1))
plot(wine_data$pH,
     main = "pH values for All Wines",
     xlab = "Index",
     ylab = "pH Level",
     col = ifelse(wine_data$good_wine, "green", "red"))
```

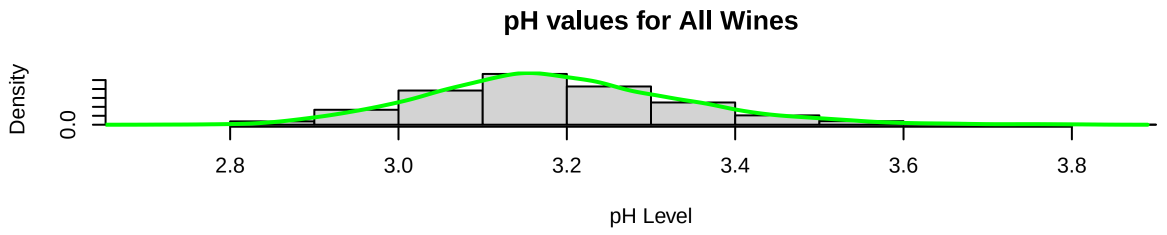


c) (2 points) Plot histograms of `pH` for `good`, `bad` and all wines in a 3x1-plot-matrix, and add the corresponding fitted normal densities to the plots. Do you observe any differences in the distributions?

```r
par(mfrow = c(3, 1))

hist(wine_data$pH[wine_data$good_wine == 1],
     probability = TRUE, main = "pH values for Good Wines", xlab = "pH Level")
lines(density(wine_data$pH[wine_data$good_wine == 1]), col = "blue", lwd = 2)

hist(wine_data$pH[wine_data$good_wine == 0],
     probability = TRUE, main = "pH values for Bad Wines", xlab = "pH Level")
lines(density(wine_data$pH[wine_data$good_wine == 0]), col = "red", lwd = 2)

hist(wine_data$pH, probability = TRUE,
     main = "pH values for All Wines", xlab = "pH Level")
lines(density(wine_data$pH), col = "green", lwd = 2)
```







```r
par(mfrow = c(1, 1))

# There are no differences in the distributions, at least from the human eyes.
```

Appendix - Code

```
knitr::knit_hooks$set(document = function(x) x)
```