hw4

Problem 1

Check out this file (which can also be grabbed from here).:

```r
# ../ reads the file from the folder above the current one
annot <- read.table("PZ.annot.txt",
                     header = FALSE,
                     sep = "\t",
                     stringsAsFactors = FALSE)

# print rows 10 through 20 to see structure (but not entries so long that it wraps the output)
print(annot[10:20, ])
```

```
##                      V1         V2                                      V3
## 10 PZ7180000020052_APQ GO:0055114         isocitrate dehydrogenase (nad+)
## 11 PZ7180000020052_APQ GO:0006099         isocitrate dehydrogenase (nad+)
## 12 PZ7180000020052_APQ GO:0004449         isocitrate dehydrogenase (nad+)
## 13 PZ7180000020052_APQ GO:0005739         isocitrate dehydrogenase (nad+)
## 14        PZ547337_APR GO:0009408                      heat shock protein
## 15 PZ7180000033253_APS GO:0043565 bhlhzip transcription factor max bigmax
## 16 PZ7180000033253_APS GO:0003700 bhlhzip transcription factor max bigmax
## 17 PZ7180000033253_APS GO:0005634 bhlhzip transcription factor max bigmax
## 18 PZ7180000033253_APS GO:0046982 bhlhzip transcription factor max bigmax
## 19 PZ7180000033253_APS GO:0006355 bhlhzip transcription factor max bigmax
## 20 PZ7180000033254_APS GO:0003713 bhlhzip transcription factor max bigmax
```

While most of the sequence IDs have an underscore suffix, not all do. In the next code block, start by extracting a `suffix_only` data frame containing just those rows where the sequence ID contains an underscore. Similarly, extract a `no_suffix` data frame for rows where sequence IDs do *not* contain an underscore. As a hint, the number of rows in `suffix_only` should be 16685.

```r
## your code here



## uncomment to test
# print(head(suffix_only))
# print(head(no_suffix))
# print(nrow(suffix_only))   #  should be 16685
# print(nrow(no_suffix))
```

Next, add to the suffix_only data frame columns for `base_id` and `suffix`, where base IDs are the parts before the underscore and suffices are the parts after the underscore (e.g., base_id is "PZ7180000023260" and suffix is "APN" for the ID "PZ7180000023260_APN").

```r
## your code here



## uncomment to test
# print(head(suffix_only))    # should include two new colums for base_id and suffix
```

Problem 2

The line `s <- rep(c("4", "1", "0", "3", "2"), 20)` generates a character vector of 100 `"0"`s, `"1"`s, `"2"`s, `"3"`s, and `"4"`s. Suppose that `"0"` means "Strongly Disagree," `"1"` means "Disagree," `"2"` means "Neutral," `"3"` means "Agree," and `"4"` means "Strongly Agree." Convert `s` into an ordered factor with levels `Strongly Disagree < Disagree < Neutral < Agree < Strongly Agree`.

```r
s <- rep(c("4", "1", "0", "3", "2"), 20)
## your code here


## uncomment to test
# print(head(s))
```

The output should look like this (with the long first line potentially wrapped differently):

```
[1] Strongly Agree    Disagree          Strongly Disagree Agree             Neutral
[6] Strongly Agree
Levels: Strongly Disagree < Disagree < Neutral < Agree < Strongly Agree
```

Problem 3

The first two lines here produce different output, but the second two lines produce the same output. Why?

```r
# different output
print(c("A", "C", "G", "D", "A", "G", "B", "F", "E") %in% c("A", "B", "C"))
```

```
## [1]  TRUE  TRUE FALSE FALSE  TRUE FALSE  TRUE FALSE FALSE
```

```r
print(c("A", "C", "G", "D", "A", "G", "B", "F", "E") == c("A", "B", "C"))
```

```
## [1]  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

```r
# same output
print(c("A", "C", "G", "D", "A", "G", "B", "F", "E", "B") %in% "D")
```

```
##  [1] FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE
```

```r
print(c("A", "C", "G", "D", "A", "G", "B", "F", "E", "B") == "D")
```

```
##  [1] FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE
```

Your answer here.

Problem 4

Like vectors, data frames (both rows and columns) can be selected by index number (*numeric vector*), logical vector, or name (*character vector*). Suppose that `current_grade` is a character vector generated by `current_grade <- sample(c("A", "B", "C", "D", "E"), size = 100, replace = TRUE)`, and `gpa` is a numeric vector, as in `gpa <- runif(100, min = 0.0, max = 4.0)`. Further, we add these as columns to a data frame, `grades <- data.frame(current_grade, gpa, stringsAsFactors = FALSE)`.

```r
current_grade <- sample(c("A", "B", "C", "D", "E"), size = 100, replace = TRUE)
gpa <- runif(100, min = 0.0, max = 4.0)

grades <- data.frame(current_grade, gpa, stringsAsFactors = FALSE)
print(head(grades))
```

```
##   current_grade       gpa
## 1             D 3.4084785
## 2             A 0.5731325
## 3             D 2.3889389
## 4             A 1.4069344
## 5             B 0.6850495
## 6             E 3.5388699
```

We are interested in pulling out all rows that have “A”, “B”, or “C” in the `current_grade` column.

```
passing <- grades[grades$current_grade %in% c("A", "B", "C"), ]   # line 1
passing <- grades[grades$current_grade == c("A", "B", "C"), ]     # line 2
passing <- grades[c("A", "B", "C"), ]                             # line 3
```

Describe what each of the three lines above actually does:

Answer for line 1

Answer for line 2

Answer for line 3

Which line (or lines) properly pull out the correct rows?

Answer

Next, which of these lines differ from their counterpart above, and why? (You may want to check the help page for `subset`.)

```
passing <- subset(grades, current_grade %in% c("A", "B", "C"))  # line 1
passing <- subset(grades, current_grade == c("A", "B", "C"))    # line 2
passing <- subset(grades, c("A", "B", "C"))                     # line 3
```

Answer