Lab 06 Stringr solutions

Question 1

Using the string functions we've learned so far, modify the string below to produce the string "United_States".

```
s <- c("  UNITED  STATES  ")

s_clean <- s %>%
  str_squish() %>%
  str_to_title() %>%
  str_replace_all(" ", "_")

s_clean
```

```
## [1] "United_States"
```

Question 2

Return to the dataset we introduced at the beginning of the lab.

```
dat <- read.csv("https://remiller1450.github.io/data/char_dom.csv")
dat
```

```
##    ID messy_x messy_y
## 1   1     100      50
## 2   2      90      40
## 3   3      85      55
## 4   4      90      45
## 5   5     110     55*
## 6   6 Missing      60
## 7   7     115      40
## 8   8    <NA>      35
## 9   9     105      40
## 10 10     100      50
```

Using the tools we have introduced thus far, correct this dataset so that the missing values in `messy_x` are recorded as `NA` and the annotated values in `messy_y` are modified so that it can be correctly converted to numeric.

```
dat_clean <- dat %>%
  mutate(
    messy_x = as.numeric(messy_x),
    messy_y = str_replace_all(messy_y, "\\*", ""),
    messy_y = as.numeric(messy_y)
  )
```

```
## Warning: There was 1 warning in `mutate()`.
## i In argument: `messy_x = as.numeric(messy_x)`.
```

```
## Caused by warning:
## ! NAs introduced by coercion
```

```
dat_clean
```

```
##    ID messy_x messy_y
## 1   1     100      50
## 2   2      90      40
## 3   3      85      55
## 4   4      90      45
## 5   5     110      55
## 6   6      NA      60
## 7   7     115      40
## 8   8      NA      35
## 9   9     105      40
## 10 10     100      50
```

Question 3

Use `str_extract()` to extract the first letter of each string in this vector. Then combine those letters with `str_c()` to create a single character string.

```
s <- c("howdy", "otter", "tragedy", "danger", "octopus", "grapes")

s %>%
  str_extract("^.") %>%
  str_c(collapse = "")
```

```
## [1] "hotdog"
```

Question 4

A common problem in data processing involves the inclusion of "free response" questions in which respondents are able to enter data without restriction. In the following vector, respondents were asked to give their phone numbers, but no standardization was enforced when doing so:

```
phone_strings <- c("Home: (507)-645-5489",
                   "Cell: 219.917.9871",
                   "My work phone is 507-202-2332",
                   "I don't have a phone")
```

Using functions from this lab, extract the phone numbers from the character vector and modify the strings so that all of the numbers are of the form `XXX-XXX-XXXX`. If a number is not included, it can be left as an empty string (`""`).

Hints:

- A pattern like `"\\d{3}[-.]"` will identify any three digits followed by a hyphen or period
- Regular expressions can be compounded
- `str_replace()` can get rid of pesky things we don't need

A bunch of ways to do this. Here is one

```
# Remove parentheses
s <- str_replace_all(phone_strings, "[()]", "")

# Extract all 3 digit punct 3 digit punct 4 digit
s <- str_extract(s, "\\d{3}[-.]\\d{3}[-.]\\d{4}")

# Swap periods for hyphen
s <- str_replace_all(s, "\\.", "-")

s
```

```
## [1] "507-645-5489" "219-917-9871" "507-202-2332" NA
```

Quesiton 5

The following code chunk takes the rownames from the `USArrests` dataset and uses them to create a data.frame with a single column containing all of the US states

```
states <- data.frame(states = rownames(USArrests))
head(states)
```

```
##       states
## 1    Alabama
## 2     Alaska
## 3    Arizona
## 4   Arkansas
## 5 California
## 6   Colorado
```

Using the `dplyr` package along with `stringr` functions introduced in this lab, create a column that *counts* how many vowels are included in each state name, and then provide a ratio of the number of vowels to the number of total letters in each state name. Arrange the data set so that states with the highest ratio of vowels to letters are on top. Hint: don't forget to deal with uppercase/lowercase letters

```
# Which state has most vowels? which has highest ratio of vowels? lowest?
states %>% mutate(nvowel = str_count(str_to_lower(states), "[aeiou]"),
                  nlen = str_length(states),
                  ratio = nvowel/nlen) %>%
  arrange(desc(ratio)) %>% head(n = 10)
```

```
##       states nvowel nlen   ratio
## 1       Iowa      3    4 0.75000
## 2       Ohio      3    4 0.75000
## 3     Hawaii      4    6 0.66667
## 4  Louisiana      6    9 0.66667
## 5      Idaho      3    5 0.60000
## 6      Maine      3    5 0.60000
## 7    Alabama      4    7 0.57143
## 8    Arizona      4    7 0.57143
## 9    Georgia      4    7 0.57143
## 10   Indiana      4    7 0.57143
```

Question 6

Consider the following character vector and the `str_subset()` function that is intending to return all of the strings with exactly 2 digits. What does it actually return? Why do you think this is? Use `str_view()` to

see what patterns are being matched.

```
s <- c("1one", "22two", "333three", "4444four")

str_subset(s, pattern = "\\d{2}")
```

```
## [1] "22two"    "333three" "4444four"
```

Now modify the regular expression so that it only pulls out `"22two"`.

```
str_view(s, "\\d{2}")
```

```
## [2] | <22>two
## [3] | <33>3three
## [4] | <44><44>four
```

```
str_subset(s, "^\\d{2}[:alpha:]+$")
```

```
## [1] "22two"
```

Question 7

Below is a collection of IDs pulled from a data base. A valid ID will consist of a single capital letter, followed by three numbers.

```
ids <- data.frame(ID = c(
  "ID: Q781","no id","ID: B847","ID: Z12","ID: M120",
  "ID: A123", NA, "ID: H064","ID: AA123","ID: R006","ID: D555",
  "ID: K410","ID: G777","ID: C019","ID: L999","NA","ID: E902","ID: J888",
  "ID: P333","ID: F301","ID: N450","ID is R569"
))
```

For this problem, you should:

- Identify which strings contain a valid ID
- Modify the entries so that only the ID code is retained
- Replace invalid or missing IDs with `NA`

```
ids_clean <- ids %>%
  mutate(
    ID = na_if(ID, "NA"),
    has_valid = str_detect(ID, "\\b[A-Z]\\d{3}"),
    ID_extracted = str_extract(ID, "[A-Z]\\d{3}"),
    ID_extracted = if_else(has_valid, ID_extracted, NA_character_)
  )

ids_clean
```

```
##            ID has_valid ID_extracted
## 1    ID: Q781      TRUE         Q781
## 2       no id     FALSE         <NA>
## 3    ID: B847      TRUE         B847
## 4     ID: Z12     FALSE         <NA>
## 5    ID: M120      TRUE         M120
## 6    ID: A123      TRUE         A123
## 7        <NA>        NA         <NA>
## 8    ID: H064      TRUE         H064
```

```
## 9   ID: AA123     FALSE         <NA>
## 10   ID: R006      TRUE         R006
## 11   ID: D555      TRUE         D555
## 12   ID: K410      TRUE         K410
## 13   ID: G777      TRUE         G777
## 14   ID: C019      TRUE         C019
## 15   ID: L999      TRUE         L999
## 16       <NA>        NA         <NA>
## 17   ID: E902      TRUE         E902
## 18   ID: J888      TRUE         J888
## 19   ID: P333      TRUE         P333
## 20   ID: F301      TRUE         F301
## 21   ID: N450      TRUE         N450
## 22 ID is R569      TRUE         R569
```

Question 8

A collaborator has just sent you a data.frame like the one below recording name and demographic data on a list of subjects:

```
df <- data.frame(Subject = c(
  "name: Alice, sex: F, age: 22",
  "name: Bob, sex: Male, age: 9",
  "name: Carol Smith, sex: Female, age: 31",
  "name: David Crockett, sex: M, age: 45",
  "name: Emily Johnson, sex: F, age: 18",
  "name: Frank, sex: Male, age: 27",
  "name: Grace Lee, sex: Female, age: 54",
  "name: Henry, sex: M, age: 6",
  "name: Isabella Martinez, sex: Female, age: 39",
  "name: Jack, sex: M, age: 72"
))
```

Clean this data frame so that there are three columns: `name`, `sex`, and `age`. Then apply the appropriate data summaries to find the average age for each sex.

```
library(tidyr)
df <- separate(df,col = Subject, into = c("Name", "Sex", "Age"), sep = ", ") %>%
  mutate(Age = str_extract(Age, "[:digit:]+") %>% as.numeric(),
         Sex = str_extract(Sex, "[MF]"),
         Name = str_replace(Name, "name: ", ""))
df
```

```
##                 Name Sex Age
## 1              Alice   F  22
## 2                Bob   M   9
## 3        Carol Smith   F  31
## 4     David Crockett   M  45
## 5      Emily Johnson   F  18
## 6              Frank   M  27
## 7          Grace Lee   F  54
## 8              Henry   M   6
## 9  Isabella Martinez   F  39
## 10              Jack   M  72
```

```r
group_by(df, Sex) %>% summarize(mean(Age))
```

```
## # A tibble: 2 x 2
##   Sex   `mean(Age)`
##   <chr>       <dbl>
## 1 F            32.8
## 2 M            31.8
```

Question 9

Below is a list of emails that were collected from an identity harvesting site. A valid email will have the following properties:

- The username will contain only alphanumeric characters, dashes, or underscores
- It will contain an `@` symbol
- The domain will only contain alphanumeric characters
- Domain extensions can only be 3 lowercase letters

```r
emails <- data.frame(email = c(
  "katie35@gmail.com",
  "hilary@grinnell.edu",
  "abbi12@site.org",
  "kendall7@iastate.gov",
  "brianna99@gmail.com",
  "alex@site.org",
  "megan88@grinnell.edu",
  "madison@iastate.gov",
  "jenny4@gmail.com",
  "kelly@site.org",
  "hilaryduff@gmail.com",
  "abby@grinnell.education",
  "kendall@iastate.gov",
  "brianna@gmail.com",
  "alex12@site.org",
  "megan@grinnell.edu",
  "madison_7@gmail.com",
  "jenny!@site.org",
  "noatsymbolgmail.com",
  "kelly3@iastate.g0v"
))
```

For this problem you should:

1. Retain only valid email addresses (you do not need to correct invalid ones)
2. Create a column of usernames and a column of domains
3. Create a table to determine if there are more `.gov`, `.com`, or `.edu` addresses

```r
valid_email <- "^([A-Za-z0-9_-]+)@([A-Za-z0-9]+)\\.([a-z]{3})$"
email_clean <- emails %>% filter(str_detect(email, valid_email)) %>%
  separate(col = email, into = c("username", "domain"), sep = "@") %>%
  separate(col = domain, into = c("domain", "extension"), sep = "\\.")
table(email_clean$extension)
```

```
##
## com edu gov org
##   6   3   3   4
```