STATS 102A Homework 3

Winter 2022

Takao Oba
205615894

February 12, 2022

Sourcing the Functions

```
source("205615894_stats102a_hw3.R")
```

1 Relational Data

Requirements

Consider the entity-relationship diagram given in Homework3.pdf and corresponding CSV files. Please use the functions from the `dplyr` package to answer the questions below.

```
library(readr)
stock <- read_csv("stock.csv")
```

```
## Rows: 14 Columns: 3

## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (2): food_item, shop
## dbl (1): price (US dollars per lb)

##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
recipe <- read_csv("recipe.csv")
```

```
## Rows: 3 Columns: 2

## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (2): name, Inventor
```

```
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
food_item <- read_csv("food_item.csv")
```

```
## Rows: 7 Columns: 3

## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (2): item, type
## dbl (1): calories

##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
ingredient <- read_csv("ingredient.csv")
```

```
## Rows: 10 Columns: 3

## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (2): recipe, food_item
## dbl (1): weight (oz)

##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(tidyr)

glimpse(stock)
```

```
## Rows: 14
## Columns: 3
## $ food_item                   <chr> "Pasta", "Meatballs", "Tomato Sauce", "Che~
## $ shop                        <chr> "Food warehouse", "Food warehouse", "Food ~
## $ `price (US dollars per lb)` <dbl> 1.5, 5.0, 1.2, 6.5, 6.0, 2.5, 1.0, 1.6, 5.~
```

```
glimpse(recipe)
```

```
## Rows: 3
## Columns: 2
## $ name     <chr> "Spaghetti and Meatballs", "Cheese Soup", "Burger"
## $ Inventor <chr> "Le cook", "Re", "Cowboys"
```

```
glimpse(food_item)
```

```
## Rows: 7
## Columns: 3
## $ item     <chr> "Pasta", "Meatballs", "Tomato Sauce", "Onions", "Cheese", "Br~
## $ type     <chr> "Wheat product", "Meat", "Sauce", "Vegetables", "Diary", "Whe~
## $ calories <dbl> 20, 40, 5, 1, 30, 25, 45
```

```
glimpse(ingredient)
```

```
## Rows: 10
## Columns: 3
## $ recipe        <chr> "Spaghetti and Meatballs", "Spaghetti and Meatballs", "S~
## $ food_item     <chr> "Pasta", "Meatballs", "Tomato Sauce", "Onions", "Bread",~
## $ `weight (oz)` <dbl> 50, 10, 5, 1, 10, 20, 2, 4, 15, 20
```

Note:

- The efficiency and accuracy of your code account for 40% of this assignment.
- Don't use any control statements.

Answers

a. Write R code to return the food items required to cook the recipe "Spaghetti and Meatballs".

```
ingredient %>% filter(recipe == "Spaghetti and Meatballs")
```

```
## # A tibble: 4 x 3
##   recipe                  food_item    `weight (oz)`
##   <chr>                   <chr>                <dbl>
## 1 Spaghetti and Meatballs Pasta                   50
## 2 Spaghetti and Meatballs Meatballs               10
## 3 Spaghetti and Meatballs Tomato Sauce             5
## 4 Spaghetti and Meatballs Onions                   1
```

The food items required to cook the recipe "Spaghetti and meatballs" are Pasta, Meatballs, Tomato Sauce, and Onions.

b. Write R code to return all recipes that contain both "Bread" and "Onions".

```r
ingredient %>%
  spread(key = "food_item", value = "weight (oz)") %>%
  filter((!is.na(Bread)) & (!is.na(Onions)))
```

```
## # A tibble: 2 x 8
##   recipe      Bread Cheese `Ground Beef` Meatballs Onions Pasta `Tomato Sauce`
##   <chr>       <dbl>  <dbl>         <dbl>     <dbl>  <dbl> <dbl>          <dbl>
## 1 Burger         10     NA            20        NA      2    NA             NA
## 2 Cheese Soup    20     15            NA        NA      4    NA             NA
```

The recipes that contain both “Bread” and “Onions” are Burger and Cheese Soup.

c. Write R code to return the recipes and the average calories that contain the food items are of type “Wheat product” or type “Meat”.

```r
# head(food_item)
# head(ingredient)


food_item %>%
  left_join(ingredient, by = c("item" = "food_item")) %>%
  filter(type == "Wheat product" | type == "Meat") %>%
  group_by(recipe) %>%
  mutate(avg = mean(calories * `weight (oz)`)) %>%
  select(recipe, avg) %>%
  distinct()
```

```
## # A tibble: 3 x 2
## # Groups:   recipe [3]
##   recipe                     avg
##   <chr>                    <dbl>
## 1 Spaghetti and Meatballs    700
## 2 Burger                     575
## 3 Cheese Soup                500
```

The recipe that contain the food items of type “Wheat product” or type “Meat” are Spaghetti and Meatballs, Burger, and Cheese Soup and their average calories are 700, 575, and 500 calories respectively.

d. Write R code to return the food items and their prices for all items of type “Wheat product” sold at “Food warehouse”.

```r
# head(stock)
# head(food_item)

food_item %>%
  left_join(stock, by = c("item"= "food_item")) %>%
  filter(type == "Wheat product" & shop == "Food warehouse") %>%
  select(item, `price (US dollars per lb)`)
```

```
## # A tibble: 2 x 2
##   item  `price (US dollars per lb)`
##   <chr>                          <dbl>
## 1 Pasta                            1.5
## 2 Bread                            2.5
```

The food items with type "Wheat product" and also sold at "Food warehouse" are Pasta and Bread and their prices are $1.5 per lb and $2.5 per lb respectively.

e. Write R code to return the average price of food items per type. For example, `(Diary, Cheese, 6.75)`.

```r
# head(stock)
# head(food_item)

food_item %>%
  left_join(stock, by = c("item" = "food_item")) %>% group_by(type) %>%
  mutate(`average price` = mean(`price (US dollars per lb)`)) %>%
  select(type, item, `average price`) %>%
  distinct()
```

```
## # A tibble: 7 x 3
## # Groups:   type [5]
##   type          item          `average price`
##   <chr>         <chr>                   <dbl>
## 1 Wheat product Pasta                    2.1
## 2 Meat          Meatballs                5.58
## 3 Sauce         Tomato Sauce             1.4
## 4 Vegetables    Onions                   1
## 5 Diary         Cheese                   6.75
## 6 Wheat product Bread                    2.1
## 7 Meat          Ground Beef              5.58
```

This shows the average price of food items per type. In row 5, we can see `(Diary, Cheese, 6.75)`.

2 Regex Golf

You are going to produce several examples of regex and trying to get the **lowest** score you can using the scoring function `regex_golf()` provided below. The scoring works as follows, you will get:

1. 10 points for every word incorrectly matched or not matched.
2. 1 point for every character in your regex expression.

```r
library(stringr)
regex_golf <- function(x, y, regex) {
  xmatch <- str_extract_all(x, regex) == x
  matched_x <- x[xmatch]
  unmatched_x <- x[!xmatch]
  ymatch <- str_extract_all(y, regex) == y
  matched_y <- y[ymatch]
  unmatched_y <- y[!ymatch]
  penalty <- 10 * sum(!xmatch, ymatch)
  score <- nchar(regex) + penalty
  list(score = score, matched_x = matched_x,
  unmatched_x = unmatched_x, matched_y = matched_y,
  unmatched_y = unmatched_y)
}
```

Requirements

Write regex patterns to play Regex Golf on the following lists. For each pattern, store the pattern as a variable named in the following way `pat_x` where the x is replaced by the number of the exercise you are doing. You should save these patterns in your .R file.

Note:

- In your output file, for each problem, report your score, along with displaying the strings in x you failed to match and the strings in y you did match.
- All of the strings you will need to match or not are in the files `wordlists.RDS` and `wordlists.RData` on Bruin Learn (use whichever you prefer).
- **Please choose 9 of the following 12 problems to do**. You are NOT expected to do all of these perfectly. In fact, on some of the more challenging ones, it is possible you will struggle to find a pattern which selects every string in the x list. THIS IS OKAY.
- These are openly available problems which do have solutions available online. DO NOT search these out. That is not the point of this exercise. Rather, the point is for you to play with regular expressions to get a better feel for them and to flex your creative thinking muscles.
- Any solution you submit you will be expected to be able to explain. If you do not understand your solution well enough to explain it, do not submit it. This question accounts for 60% of this assignment.

Answers

Please select 9 out of the 12 problems given – make sure to name the regex in the form of `pat_x`, where `x` is the question number.

2.1 Selection 1: Warmup

```
words <- readRDS("wordlists.RDS")
# words$Warmup

regex_golf(words$Warmup$x, words$Warmup$y, pat_1)$score
```

```
## [1] 7
```

```
regex_golf(words$Warmup$x, words$Warmup$y, pat_1)$unmatched_x
```

```
## character(0)
```

```
regex_golf(words$Warmup$x, words$Warmup$y, pat_1)$matched_y
```

```
## character(0)
```

2.2 Selection 2: Anchors

```r
# words$Anchors

regex_golf(words$Anchors$x, words$Anchors$y, pat_2)$score
```

```
## [1] 3
```

```r
regex_golf(words$Anchors$x, words$Anchors$y, pat_2)$unmatched_x
```

```
## character(0)
```

```r
regex_golf(words$Anchors$x, words$Anchors$y, pat_2)$matched_y
```

```
## character(0)
```

2.3 Selection 3: Ranges

```r
# words$Ranges

regex_golf(words$Ranges$x, words$Ranges$y, pat_3)$score
```

```
## [1] 7
```

```r
regex_golf(words$Ranges$x, words$Ranges$y, pat_3)$unmatched_x
```

```
## character(0)
```

```r
regex_golf(words$Ranges$x, words$Ranges$y, pat_3)$matched_y
```

```
## character(0)
```

2.4 Selection 4: Backrefs

```r
# words$Backrefs

regex_golf(words$Backrefs$x, words$Backrefs$y, pat_4)$score
```

```
## [1] 11
```

```r
regex_golf(words$Backrefs$x, words$Backrefs$y, pat_4)$unmatched_x
```

```
## character(0)
```

```
regex_golf(words$Backrefs$x, words$Backrefs$y, pat_4)$matched_y
```

```
## character(0)
```

2.5 Selection 5: Abba

```
# words$Abba
#y has palindrome so we must not match these

regex_golf(words$Abba$x, words$Abba$y, pat_5)$score
```

```
## [1] 19
```

```
regex_golf(words$Abba$x, words$Abba$y, pat_5)$unmatched_x
```

```
## character(0)
```

```
regex_golf(words$Abba$x, words$Abba$y, pat_5)$matched_y
```

```
## character(0)
```

2.6 Selection 6: A man, a plan

```
# words$`A man, a plan`
#first and last letter are same except for sporous in y

regex_golf(words$`A man, a plan`$x, words$`A man, a plan`$y, pat_6)$score
```

```
## [1] 11
```

```
regex_golf(words$`A man, a plan`$x, words$`A man, a plan`$y, pat_6)$unmatched_x
```

```
## character(0)
```

```
regex_golf(words$`A man, a plan`$x, words$`A man, a plan`$y, pat_6)$matched_y
```

```
## character(0)
```

2.7 Selection 7: Four

```
# words$Four
#m(a).....x(a)y(a)z(a)...

regex_golf(words$Four$x, words$Four$y, pat_7)$score
```

```
## [1] 15
```

```
regex_golf(words$Four$x, words$Four$y, pat_7)$unmatched_x
```

```
## character(0)
```

```
regex_golf(words$Four$x, words$Four$y, pat_7)$matched_y
```

```
## character(0)
```

2.8 Selection 8: Order

```
# words$Order
#letter in word in alphabetical order
#five or six letter word, except for oriole in y

regex_golf(words$Order$x, words$Order$y, pat_8)$score
```

```
## [1] 11
```

```
regex_golf(words$Order$x, words$Order$y, pat_8)$unmatched_x
```

```
## character(0)
```

```
regex_golf(words$Order$x, words$Order$y, pat_8)$matched_y
```

```
## character(0)
```

2.9 Selection 9 : Glob

```
# words$Glob

regex_golf(words$Glob$x, words$Glob$y, pat_9)$score
```

```
## [1] 46
```

```
regex_golf(words$Glob$x, words$Glob$y, pat_9)$unmatched_x
```

```
## character(0)
```

```
regex_golf(words$Glob$x, words$Glob$y, pat_9)$matched_y
```

```
## character(0)
```