DSA2101 mid-term AY20/21 Sem 1 Review

Question 1 (5 marks)

The website http://www.weather.gov.sg/home contains a forecast of weather conditions in Singapore over the next 24 hours. This information can be found here in the top right-hand corner of that website (see the red rectangle below). Write a function called `get_forecast` with *no arguments*, that will extract this forecast from that website. It would return a dataframe with 9 columns and 1 row:

Comments

Overall, this question was well done. Everyone knew how to get an appropriate css path, and then format the scraped data to get the dataframe. Some of the common mistakes were that the degrees symbol was not removed, or that the number of columns in the dataframe was incorrect.

```
##        fcast_datetime      description min_temp max_temp min_humid max_humid
## 1 2020-11-02 12:01:29 Thundery showers       24       34        55        95
##   wind_dir min_wind max_wind
## 1       NW       10       20
```

Question 2 (10 marks)

The dataset `wine_reviews_midterm_2021.xlsx` contains wine reviews from a magazine. This is a modified version of the dataset from (https://www.kaggle.com/zynicide/wine-reviews). If you are interested, you may get the column names from there.

1. Read the data into R. Did you observe any warnings? Investigate why they are there and fix the issues **within R**. Save your cleaned dataset as `wine_mag_clean`.
2. Most entries in the `title` column contain the vintage (year of the wine), year of the winery, and the region from which the wine came (in parentheses). Extract the region and vintage items and store them in two new columns, called `region` and `vintage`. You may ignore the currently existing `region_1` and `region_2` columns. *Hint: Vintage should not be earlier than the 20th century.*
3. Convert the following columns to lower case:
    1. `title`
    2. `variety`
    3. `winery`
    4. `description`
4. Convert the `price` and `points` columns to numeric.
5. Explore the dataset, and answer one question you find interesting about this data. Include the code you used, and summarise (in words) what you tried.

Comments (1)

Most of you realised what was wrong with the dataset - some of the rows had been shifted, resulting in the data frame jutting out for 100+ rows. The description column had spilled over into the designation column

for some of these rows. However, some of you chose to delete those rows instead of fixing them. This is not correct.

Another problem with this dataset is that there are duplicate rows. If you remove the id column, you will find 10K-odd exact duplicates. These should be removed too.

```
library(tidyverse)
library(stringr)
library(readxl)

wine_mag <- read_excel("../data/wine_reviews_midterm_2021.xlsx")

filter(wine_mag, str_detect(points, "[^0-9]")) -> need_to_fix
filter(wine_mag, str_detect(points, "[^0-9]", TRUE)) -> no_need_to_fix
filter(wine_mag, is.na(points)) -> na_points       # also need to fix

filter(need_to_fix, is.na(price)) -> fix1  # shift by two
filter(need_to_fix, !is.na(price)) -> fix2 # shift by one
no_need_to_fix <- mutate(no_need_to_fix, points = as.numeric(points))

fix1 <- mutate(fix1, description = paste(description, designation, points, sep=" "))
fix1[, 4:14] <- fix1[, 6:16]
fix1 <- mutate(fix1, designation = as.character(designation), price = as.numeric(price),
               points=as.numeric(points))

fix2 <- mutate(fix2, description = paste(description, designation, sep=" "))
fix2[, 4:14] <- fix2[, 5:15]
fix2 <- mutate(fix2, designation = as.character(designation), price = as.numeric(price),
               points=as.numeric(points))

na_points <- mutate(na_points, description = paste(description, designation, sep=" "))
na_points[, 4:14] <- na_points[, 5:15]
na_points <- mutate(na_points, designation = as.character(designation),
                    price = as.numeric(price), points=as.numeric(points))

wine_mag_clean <- bind_rows(no_need_to_fix, fix1, fix2, na_points) %>%
  select(1:14)
dup_ids <- which(duplicated(select(wine_mag_clean, -id)))
wine_mag_clean2 <- wine_mag_clean[-dup_ids,]
```

Comments (2)

Extracting the region was well done, but some of the parenthesis contain "+" and one of the title contains two pairs of parentheses (ids 16831 and 99508). I manually edited these.

For the vintage, it was important to realise that some titles contain the year the winery was established and the year of the wine. Some attempt to deal with this should be present in the code (egs 54720 and 72369). Below, I assumed that the larger of the two is the vintage, and if there is only one, that is the vintage and not the winery's age.

```
tmp <- mutate(wine_mag_clean2,
       region=str_extract(title, "(?=\\().*(?<=\\))"), .after=country) %>%
  select(1:5)
# problems with ("B") Sauvignon Blanc (Marlborough), (+) Malbec (Mendoza)
tmp <- mutate(tmp, region = str_sub(region, 2, -2))
```

```
tmp$region[tmp$id == 16831] <- "Marlborough"
tmp$region[tmp$id == 99508] <- "Mendoza"
wine_mag_clean2$region <- tmp$region

#mutate(wine_mag_clean, vintage = str_extract(title, "[12]?[0-9]{3}"),
#       .after=country) %>% select(1:4, title) %>% View

possible_vintage <- str_extract_all(wine_mag_clean2$title, "[0-9]{4}") %>%
  lapply(FUN=as.integer)
possible_vintage2 <- vapply(possible_vintage,
                            function(x) {
                              if(length(x) >= 1) {
                                x <- x[x >= 1900 & x <= 2017]
                                if(length(x) > 1)
                                  x <- max(x)
                              }
                              if(length(x) == 0)
                                return(NA)
                              x
                            }, FUN.VALUE = 2L,USE.NAMES = FALSE)
wine_mag_clean2$vintage <- possible_vintage2
#mutate(wine_mag_clean, vintage = str_extract(title, "[0-9]{4}"),
#       .after=country) %>% select(1:4, title) %>% View
```

Comments (3)

For these two questions, I was looking for the use of `across`, `mutate_at` or even just `lapply`.

```
wine_mag_clean <- mutate(wine_mag_clean,
       across(c("title", "variety", "winery", "description"), str_to_lower))
```

Comments (4)

Some discussion of a question and the code to check, followed by some summary would have got you the marks.

A **rough breakdown** of marks was $4 + 3 + 1 + 1 + 3$. Remember that I also look for efficient code, so the breakdown is not hard and fast. These are some of the id numbers that I checked:

```
filter(wine_mag_clean, id %in% c(36709, 71003, 76463, 92861, 19904, 45236,
                                 16831, 99508, 54720, 72369)) %>% View
select(wine_mag_clean, points, price, vintage) %>% summary()
```

Question 4 (10 marks)

The file `transactions.rds` contains transactions from sales through an online gift shop. The sales are stored in an S3 object of class `transactions`:

There were 38 categories of items sold at the store. Their names can be found in the `itemLabels` component.

The object contains information on 18270 invoices. The invoice numbers are stored in the `transactionID` component of the object.

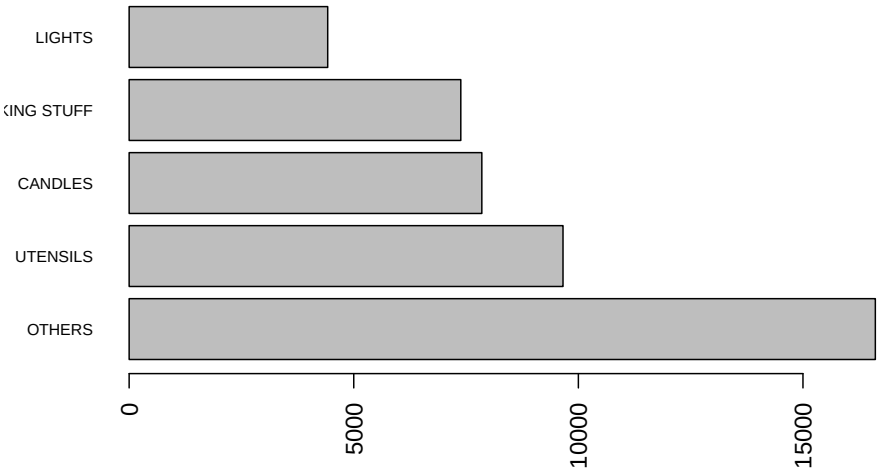
The items corresponding to each invoice are stored in a a 38 by 18270 matrix of 1's and 0's. A one in row (i, j) of this matrix would indicate category i item was present in invoice j, where $1 \leq i \leq 38$ and $1 \leq j \leq 18270$. However, to save space, a sparse representation of this matrix is used - only the non-zero elements are stored. Component `p` is used to indicate which range of elements in `i` correspond to each of the 18270 invoices. For instance, it shows that 4 - 0 = 4 categories were present in the first invoice: categories 5, 16, 21 and 35. Similarly, 5 - 4 = 1 category was present in the second invoice: category 13.

1. Write a `plot` method for this class of objects, that displays the count of the `topN` item categories. Your method should call the `barplot` function from base R; it should not use ggplot. The signature of the argument should be:

```
plot.transactions <- function(x, topN=10, ...) {
  item_count <- sort(table(x$data$i), decreasing = TRUE)[1:topN]
  ids <- as.integer(names(item_count))
  counts <- as.vector(item_count)
  barplot(counts, names.arg = x$itemLabels$labels[ids+1], ... )
}
`[.transactions` <- function(x, j) {
  total_num_invoices <- length(x$transactionID$transactionID)
  invoice_nos <- x$transactionID$transactionID[j]

  start_ids <- x$data$p[-total_num_invoices][j] + 1
  end_ids <- x$data$p[-1][j]
  item_list <- mapply(function(a,b) x$data$i[a:b], start_ids, end_ids, SIMPLIFY = FALSE,
                      USE.NAMES = FALSE)
  item_vec <- unlist(item_list)

  #browser()
  item_counts <- vapply(item_list, length, FUN.VALUE = 1L)
  invoice_vec <- rep(invoice_nos, times=item_counts)

  data.frame(invoice=invoice_vec, item=x$itemLabels$labels[item_vec+1])
}
trans_data <- readRDS("../data/transactions.rds")
```

(*Read up on the ... argument on your own.*) Here is an example of the call and output:

```
plot(trans_data, topN=5, horiz=TRUE, las=2, cex.names=0.7)
```



2. The indexing operator that we use in R is just another function. Take a look:

```
X <- 6:10
X[c(1,4,2)]
```

```
## [1] 6 9 7
```

```
`[`(X, c(1,4,2))
```

```
## [1] 6 9 7
```

Write an indexing method for objects of class `transactions`, that accesses invoice numbers. It would work this way:

```
trans_data[1:2]
```

```
##   invoice        item
## 1  536365     CANDLES
## 2  536365      LIGHTS
## 3  536365      OTHERS
## 4  536365    UTENSILS
## 5  536366 HAND WARMER
```

```
trans_data[c(2, 4)]
```

```
##   invoice        item
## 1  536366 HAND WARMER
## 2  536368      OTHERS
```

Comments

For this question, you would have lost marks if you did not use the `...` argument, or if you set the default `topN` wrongly. If you used `for` loops you would also have lost marks. Those who named the function `plot` would have lost marks because that is not an S3 function.

The second question showed a lot of incorrect indexing. Here are some examples:

```
# Observed code:
item_index <- x$data$i
items <- x$itemLabels$labels
item_list <- sapply(item_index, function(y)items[y+1])
# Can be replaced with:
x$itemLabels$labels[x$data$i + 1]

# Observed code:
num_items_vec <- sapply(1:18270, function(x) trans_data$data$p[x + 1] - trans_data$data$p[x])
# Can be replaced with:
trans_data$data$p[2:18270] - trans_data$data$p[1:18269]

# Observed code:
labels <- x$itemLabels
get_label <- function(x, la) {
  return(la[x + 1, ])
}
which_cats_in_invoice <- x$data$i
which_cats_in_invoice <- unlist(lapply(which_cats_in_invoice, get_label, labels))
# Can be replaced with:
x$itemLabels$labels[x$data$i + 1]
```

```r
# Observed code
counts <- sapply(0:37, function(x) length(which(items == x)))
# Can be replaced with:
table(items)
```

The final thing I looked for in this question was that it worked with negative indexing.

Summary

Overall, the grading was strict, since this is an open-book, open-internet take-home exam. Please take note of the indexing issue above, this is very important. Finally, I am happy to see that all of you really paid attention to your data!