Assignment 3

Nguyen Phuc Thai

2023-04-26

1 Some observation of the data before loading

There are severals empty boxes in the csv file, which I will recognised as missing values (NAs) when I importthe data

```
# the Working directory where I save the dataset
setwd("G:/My Drive/Adelaide uni/Stats7022/Assignment 1")
```

2 Loading data and Libraries

```
pacman::p_load(tidyverse,skimr,forcats,purrr,inspectdf)
jobs<-read.csv('23-02-13_jobs.csv',na.strings = '')
data(diamonds, package = "ggplot2")
```

3 Breaking the data into pieces

```
skim_without_charts(jobs)
```

Table 1: Data summary

Name	jobs
Number of rows	22000
Number of columns	14
Column type frequency:	
character	14

Table 1: Data summary

Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
country	0	1.00	24	24	0	1	0
country_code	0	1.00	2	2	0	1	0
date_added	21878	0.01	8	10	0	78	0
has_expired	0	1.00	2	2	0	1	0
job_board	0	1.00	16	16	0	1	0
job_description	0	1.00	16	20582	0	18744	0
job_title	0	1.00	1	461	0	18759	0
job_type	1628	0.93	6	45	0	39	0
location	0	1.00	2	18836	0	8423	0
organization	6867	0.69	3	132	0	738	0
page_url	0	1.00	55	229	0	22000	0
salary	18554	0.16	2	288	0	1737	0
sector	5194	0.76	5	3505	0	163	0
uniq_id	0	1.00	32	32	0	22000	0

The table above shows that the dataset contains 22,000 observations of job posts, showing the country that the job is based, date the job is added, the website of the post, whether the job has expired, job board, job description, job title, tob type, location, organization, url of the post, id of the post, salary and sector.

As these are all job posts, there is no need to split the data before cleaning. Furthermore, since no machine learning is applied at this stage, and the work at this point is just cleaning, a train-test split is unnecessary

4 Convert data into tibble

The data imported from csv, and is in the dataframe format, so converting it to tibble is not needed for cleaning, but is done as follow.

```
jobs<-as_tibble(jobs)
```

5 Looking at each columns

The cleaning will be done in the order of how much cleaning each column needed, not by the order they appear in the dataset

5.1 Columns that can be removed

Since every job listed in the dataset are US-based and posted on monsterboard.com, the column *country, country_code* and *job_board* can be removed as every observations has the same value for these columns, as can be seen in table 1

Every job post has it unique id and url hence the column *uniq_id* and *page_url* can be removed

Most of the observations has missing date added (less than 1% complete rate in table 1), so it can be removed for analysis

These are removed as follow

```
jobs<-jobs%>%
  select(-c(country,country_code,has_expired,job_board,uniq_id,page_url,date_added))
```

5.2 Job description and job title column

Every jobs has its own description and title given by the company, but they should not be removed it can be used for useful analysis. But as both are text-based data, it requires natural language processing methods, or some text-based processing. Consequently, this data is kept untouched in this cleaning process due to unknown purpose of use for the dataset. An example of a description is:

5.3 Job type

Initially, the dataset contains a large number of job types, 39, containing various description of part-time, full-time and other types of jobs. Furthermore, a few job types was mistyped into the column location and organization, which are shown below:

```
mistyped<-jobs%>%
       select(job_type,location,organization)%>%
       filter(organization=='Full Time Employee'|location=='Full Time Employee')
```

Table 3: mistyped job types

job_type	location	organization
NA	Full Time Employee	Full Time Employee
NA	Full Time Employee	Full Time Employee
NA	Full Time Employee	Full Time Employee
NA	Full Time Employee	Full Time Employee
NA	Full Time Employee	Full Time Employee
NA	Full Time Employee	Full Time Employee
NA	Full Time Employee	Full Time Employee
NA	Full Time Employee	Full Time Employee
NA	Full Time Employee	Full Time Employee

The observations in table 3 will be moved to the correct columns, and the mistyped row will be replaced as missing value in the location and organization columns as follow

```
jobs[jobs$location=='Full Time Employee',]$job_type='Full Time Employee'
jobs[jobs$location=='Full Time Employee',]$location=NA
jobs[jobs$organization=='Full Time Employee'&
           !is.na(jobs$organization),]$organization=NA
```

Then the *job_type* column will be collapsed down to 3 levels, full time, part time and other, along with missing values

```
jobs<-jobs%>%
  mutate(job_type=case_when(str_detect(job_type,'Full Time')~'Full Time',
                            str_detect(job_type,'Part Time')~'Part Time',
                            is.na(job_type)~ NA_character_,
                            T~ 'Other'
         ))
```

5.4 Location and organization

These two columns will be cleaned together, as the problem with both are, some values that should be in the location columns were typed into the organization columns, and vice versa. A few of them can be observed below:

```
## extracting unique value in both, aside from NA
unique_location=unique(jobs$location)
unique_location=unique_location[!is.na(unique_location)]
unique_organzation=unique(jobs$organization)
unique_organzation=unique_organzation[!is.na(unique_organzation)]


mistype2<-head(jobs%>%
       select(location,organization)%>%
       filter(organization %in% unique_location))
```

Table 4: mistyped location-organization

location	organization
Sr. Process Engineer, Manufacturing	Chicago, IL
RF System Technician, Field Service	Oklahoma City, OK
Bi-Lingual Editorial Strategist	San Francisco, CA
Quality Engineer	Durham, NC
Business Analyst	Houston, TX
Packaging Engineer	Chicago, IL 60661

To fix this issue, if a row is found to have value organization values that can also be found in all unique location values, the value organization and location will be swapped for that row. The swap is done based on the organization row because during checking location values that can be found in organization columns, the location values checked are actual location.

The number of observation in location value that can be found in organization columns before the swap: 4088

The number of observation in organization column that can be found in location columns before the swap: 218

```
jobs[jobs$organization %in% unique_location, c("organization", "location")] <-
  jobs[jobs$organization %in% unique_location, c( "location",'organization')]

## Redo unique values after swapping
unique_location=unique(jobs$location)
unique_location=unique_location[!is.na(unique_location)]
unique_organzation=unique(jobs$organization)
unique_organzation=unique_organzation[!is.na(unique_organzation)]
```

This swap is imperfect, as if there is a location that only appear once in the dataset, and it appears in the organization columns, the swap will not be made, and also vice versa. So further checking may be required. However:

The number of observation in location value that can be found in organization columns before the swap: 22

The number of observation in organization column that can be found in location columns before the swap: 21

The number of possible misplaced value has reduced significantly. Additionally, looking at table 3, the format of the location value are not consistent, with some location contains post code and some not. Therefore, to make it consistent, postcodes will be removed. Additionally, some job description was appeared to be entered into both columns, and will be removed. To removed them, all values containing more than 100 characters in the location columns, and those with more than 150 characters for the organization columns are replaced as missing values.

```
jobs$location<-str_replace_all(jobs$location, "[:digit:]", "")

jobs[str_length(jobs$location)>100 & !is.na(jobs$location),]$location=NA

jobs[str_length(jobs$organization)>150 & !is.na(jobs$organization),]$location=NA
```

5.5 Salary

Some non-missing salary values are shown as follow:

```
salary<-head(jobs%>%
       select(salary)%>%
       filter(!is.na(salary)))
```

Table 5: Some salary values

salary
9.00 - 13.00 $ /hour
80,000.00 - 95,000.00 $ /year
60,000.00 - 72,000.00 $ /year
Excellent Pay and Incentives
70,000.00 - 100,000.00 $ /year
62.00 - 81.00 $ /hour

As shown above, these values are inconsistent, with some hourly pay and yearly salary, and some are shown as range, some are not. Therefore, this columns will be used to create new columns, namely: columns containing minimum hourly pay, columns containing maximum hourly pay, columns containing minimum yearly pay, columns containing maximum yearly pay. These are created as they provide more insight regarding the pay range of the job, which could be a very crucial factor of a job post. In the current format, salary is just text-based information, while it should be numeric data.

The cleaning is done step by step as follow

```
## Extracting the pay range from the given information in salary column
jobs1=jobs %>%
       select(salary)%>%
       mutate(salary=str_replace_all(salary,',',''))%>%
  mutate(pay_range = map(str_extract_all(salary, '\\d+([.,]\\d+)?'), as.numeric))

#pay range will be a list of vector
pay_range=jobs1$pay_range

## columns with only description of salary with no range or numerical values are extrac
counter <- (sapply(pay_range, length))

pay_range[counter==0] <- NA

## initialising vectors that contains value to be used for new columns
min_hourly_wage=rep(NA,nrow(jobs))
max_hourly_wage=rep(NA,nrow(jobs))
min_yearly_salary=rep(NA,nrow(jobs))
max_yearly_salary=rep(NA,nrow(jobs))
```

```
## To get the min hourly wage, find values in the salary columns that contains "hour"
##and is not NA in the pay range vector
min_hourly_wage[str_detect(jobs$salary,'hour') & !is.na(pay_range)]=
  unlist(map(pay_range[str_detect(jobs$salary,'hour') & !is.na(pay_range)],1))


## To get the max hourly wage, find values in the salary columns that contains "hour"
##and is not NA in the pay range vector and are vector containing more than 1 value
max_hourly_wage[str_detect(jobs$salary,'hour') & !is.na(pay_range)
                & sapply(pay_range,length)>1]=
  unlist(map(pay_range[str_detect(jobs$salary,'hour') & !is.na(pay_range)&
                         sapply(pay_range,length)>1],2))


## To get the min yearly wage, find values in the salary columns that contains "year"
##and is not NA in the pay range vector
min_yearly_salary[str_detect(jobs$salary,'year') & !is.na(pay_range)]=
  unlist(map(pay_range[str_detect(jobs$salary,'year') & !is.na(pay_range)],1))



## To get the max yearly wage, find values in the salary columns that contains "year"
##and is not NA in the pay range vector and are vector containing more than 1 value
max_yearly_salary[str_detect(jobs$salary,'year') & !is.na(pay_range)&
                    sapply(pay_range,length)>1]=
  unlist(map(pay_range[str_detect(jobs$salary,'year')
                       & !is.na(pay_range)& sapply(pay_range,length)>1],2))

## putting the newly created vector into the dataset
jobs$min_hourly_wage=min_hourly_wage
jobs$max_hourly_wage=max_hourly_wage
jobs$min_yearly_salary=min_yearly_salary
jobs$max_yearly_salary=max_yearly_salary
```

5.6 Sector

Finally, on inspection, this columns appear to be relatively error-free, except for a few lengthy, description-like entries that should not belong to this column. They are cleaned as follow:

```
jobs[str_length(jobs$sector)>150 & !is.na(jobs$sector),]$location=NA
```

6 On the missing data and problem summary

```
NAs=inspect_na(jobs)
```

Table 6: Missing data

col_name	cnt	pcnt
max_hourly_wage	20936	95.163636
min_hourly_wage	20856	94.800000
max_yearly_salary	20332	92.418182
min_yearly_salary	20243	92.013636
salary	18554	84.336364
organization	6876	31.254545
sector	5194	23.609091
job_type	1619	7.359091
location	1239	5.631818
job_description	0	0.000000
job_title	0	0.000000

Table 6 shows that the dataset contains a significant amount of of missing data. Some were dealt with (like the *date_added* column) but others are not. Because it is unsure what is the purpose of the cleaned data, columns with large number of missing data are cleaned rather than removed completely. Hence further cleaning are needed to deal with missing values in this dataset before it is used for any purposes.

Consequently, some problem of the data remaining after cleaning are:

- job_description and job_title: both are text based and need some processing to be used.
- job_type: missing value
- location and organization: some misplaced value could still remain, and missing value
- salary and new columns created from it: missing values
- sector: missing values

7 Save the data

```
filenamepath <- glue::glue("{lubridate::today()}-jobs.csv")

## the name will be
filenamepath
```

```
## 2023-04-26-jobs.csv
```

```
##add the working directory before running this,
##I set my working directory at the start so I don't need it
write.csv(jobs,filenamepath)
```