Assignment 2 Question 1

Nguyen Phuc Thai

2023-04-14

Importing the cleaned data and libraries

```
# the Working directory where I save the dataset
setwd("G:/My Drive/Adelaide uni/Stats7022/Assignment 2")


# library and the dataset
pacman::p_load(tidyverse,skimr,forcats,purrr,inspectdf,kableExtra)
jobs<-readRDS('jobs_clean.rds')
```

Preprocessing

Calculating annual salary

As many salary information given in the salary columns are pay range in both weekly and annual payment, they will be extracted as follow:

```
# store in a new dataset processed salary information
jobs1=jobs %>%
  select(salary)%>%
  mutate(salary=str_replace_all(salary,',',''))%>%
  ## remove all comma in numbers so the number can be converted to numeric data
  mutate(pay_range = map(str_extract_all(salary, '\\d+([.,]\\d+)?')
                         , as.numeric))
## extract all numbers, pay range for each job post will now be store
# as a vector of 2 values, exact pay is vector of length one, so the
#column pay_range is effectively a list



pay_range=jobs1$pay_range  # store pay range in a separate list

idx <- (sapply(pay_range, length))
#storing vector length in a vector

pay_range[idx==0] <- NA
# using vector of length to identify jobs without payment information
```

After extracting numeric payment from the salary columns, the hourly pay and annual salary for jobs with payment information disclosed will be extracted as follow

```
## initialising vectors that contains value to be used for calculations
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))
```

After that a single annual payment is calculated for all jobs with payment information (for jobs with annual pay range, the mean is used, same with hourly pay, and all hourly pay is multiplied by 1800 to get the annual payment)

```
## putting all payment information to a new dataframe
Salary=data.frame(min_hourly_wage=min_hourly_wage,
                  max_hourly_wage=max_hourly_wage,
                  min_yearly_salary=min_yearly_salary,
                  max_yearly_salary=max_yearly_salary)



## calculating a single annual salary (for jobs that already satisfy
#this, this function will just give the mean of a single value)
Salary$yearly_wage=rowMeans(Salary[,c("min_yearly_salary","max_yearly_salary")],
                            na.rm = T)
```

```
## doing the same thing for hourly pay, but not multiply by 1800
Salary$hourly_salary=rowMeans(Salary[,c('min_hourly_wage','max_hourly_wage')],
                              na.rm=T)

## replace NaN by NA in the data frame
Salary <- Salary %>% mutate_all(~ifelse(is.nan(.), NA, .))

## calculating yearly payment for jobs with only hourly pay disclosed
Salary$yearly_wage[is.na(Salary$yearly_wage)]=Salary$hourly_salary[is.na(Salary$yearly_wage)]*1800
```

Then we can add this column to the jobs dataset

```
jobs$anual_salary=Salary$yearly_wage
```

some non missing value from this column now looks like this

```
head(jobs%>%
  select(anual_salary)%>%
  filter(!is.na(anual_salary)))
```

```
## # A tibble: 6 x 1
##   anual_salary
##          <dbl>
## 1        19800
## 2        87500
## 3        66000
## 4        85000
## 5       128700
## 6        87500
```

Extracting the State from the jobs dataset

Some cleaning before extracting

Some location data was misplaced in the organization columns as follow

```
## 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 1: 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.

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.

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

Additionally, 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 2: mistyped job types

job_type	location	organization
Other	Full Time Employee	Full Time Employee
Other	Full Time Employee	Full Time Employee
Other	Full Time Employee	Full Time Employee
Other	Full Time Employee	Full Time Employee
Other	Full Time Employee	Full Time Employee
Other	Full Time Employee	Full Time Employee
Other	Full Time Employee	Full Time Employee
Other	Full Time Employee	Full Time Employee
Other	Full Time Employee	Full Time Employee

The observations in table 2 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
```

And some job description was also found in this column

```
print(jobs[str_length(jobs$location)>100 & !is.na(jobs$location),]$location[1])
```

```
## [1] "DePuy Synthes Companies is a member of Johnson & Johnson's Family of Companies, and is recruiti
```

So any location data with more than 100 character will be removed

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

Some location also have the post code, while others do not. Postcode will also be removed from the data

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

Processing the location data

Since the two-letter abbreviation of the state always come after the comma, and is the last characters of the location, it will be extracted as follow:

```
## extracting character after the comma
state <- sub('.*,\\s*','\\1', jobs$location)
## trim whitespace from the extracted string
state=trimws(state)
## anything that is longer than 2 characters will be removed
state[str_length(state)>2 & !is.na(state)]=NA


## Adding state to the dataset
jobs$state=state
```

Some values from this column

```
head(jobs%>%
  select(state)%>%
  filter(!is.na(state)))
```

```
## # A tibble: 6 x 1
##   state
##   <chr>
## 1 WI
## 2 WI
## 3 CA
## 4 PA
## 5 VA
## 6 TX
```

Simplifying sector column

The column will be lumped so that only the top 20 sectors are kept

```
## lumping the low-count sectors into others
jobs$sector=fct_lump(jobs$sector,20)
```

A summary of this variable:

```
knitr::kable(summary(jobs$sector), caption = "sector column")%>%
  kableExtra::kable_styling(latex_options = "HOLD_position")
```

Table 3: sector column

	x
Accounting/Finance/Insurance	742
Administrative/Clerical	271
Customer Support/Client Care	328
Engineering	153
Entry Level	1172
Experienced (Non-Manager)	4594
Food Services/Hospitality	633
Human Resources	205
Installation/Maintenance/Repair	574
IT/Software Development	861
Legal	237
Logistics/Transportation	371
Manager (Manager/Supervisor of Staff)	900
Manufacturing/Production/Operations	544
Marketing/Product	311
Medical/Health	1254
Project/Program Management	790
Quality Assurance/Safety	323
Sales/Retail/Business Development	938
Security/Protective Services	311
Other	1294
NA's	5194

Aside from the 20 most common sectors, we have levels for missing values and all other uncommon sectors.

Save the data

```
filenamepath <- glue::glue("{lubridate::today()}-jobs.csv")
## the name will be
filenamepath
```

```
## 2023-04-14-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)
```