AUS_Labour_Market_Tech_Sector_Analysis_by_Dohyeon_Lee.R

dohye

2022-01-17

```r
# Description: Analysis of the current state of the Australian tech labour market
#               and predicting the future trend.
# Part A: Import Data
# Part B: Data Preparation and Wrangling
# Part C: Initial Analysis
# Part D: Data Visualization
# Part E: Exploratory Data Analysis, Data-Driven Modeling
# Part F: Evaluate Results
# Part G: Forecast trend

###################### Part A ######################
# install packages if it is not installed
list.of.packages <- c("tidyverse", "stringr", "reshape2", "Metrics", "scales", "lubridate",
                      "RSocrata", "zoo", "xts", "httr", "xlsx", "readxl", "sjmisc", "dplyr",
                      "rpart", "rattle", "randomForest", "gsubfn", "e1071")
new.packages <- list.of.packages[!(list.of.packages %in% installed.packages()[,"Package"])]
if(length(new.packages)) install.packages(new.packages)

# Load packages installed
lapply(list.of.packages, library, character.only = TRUE)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.3     v dplyr   1.0.7
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   2.0.1     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()

##
## Attaching package: 'reshape2'

## The following object is masked from 'package:tidyr':
##
##     smiths
```

```
##
## Attaching package: 'scales'

## The following object is masked from 'package:purrr':
##
##     discard

## The following object is masked from 'package:readr':
##
##     col_factor

##
## Attaching package: 'lubridate'

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

## Warning: package 'RSocrata' was built under R version 4.1.2

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric

##
## Attaching package: 'xts'

## The following objects are masked from 'package:dplyr':
##
##     first, last

## Warning: package 'xlsx' was built under R version 4.1.2

## java.home option:

## JAVA_HOME environment variable: C:\Program Files (x86)\Java\jdk1.6.0_23

## Warning in fun(libname, pkgname): Java home setting is INVALID, it will be ignored.
## Please do NOT set it unless you want to override system settings.

## Warning: package 'sjmisc' was built under R version 4.1.2

##
## Attaching package: 'sjmisc'

## The following object is masked from 'package:purrr':
##
##     is_empty
```

```
## The following object is masked from 'package:tidyr':
##
##     replace_na

## The following object is masked from 'package:tibble':
##
##     add_case

## Loading required package: bitops

## Rattle: A free graphical interface for data science with R.
## Version 5.4.0 Copyright (c) 2006-2020 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.

## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:rattle':
##
##     importance

## The following object is masked from 'package:dplyr':
##
##     combine

## The following object is masked from 'package:ggplot2':
##
##     margin

## Warning: package 'gsubfn' was built under R version 4.1.2

## Loading required package: proto

## Warning: package 'proto' was built under R version 4.1.2

## [[1]]
##  [1] "forcats"   "stringr"   "dplyr"     "purrr"     "readr"     "tidyr"
##  [7] "tibble"    "ggplot2"   "tidyverse" "stats"     "graphics"  "grDevices"
## [13] "utils"     "datasets"  "methods"   "base"
##
## [[2]]
##  [1] "forcats"   "stringr"   "dplyr"     "purrr"     "readr"     "tidyr"
##  [7] "tibble"    "ggplot2"   "tidyverse" "stats"     "graphics"  "grDevices"
## [13] "utils"     "datasets"  "methods"   "base"
##
## [[3]]
##  [1] "reshape2"  "forcats"   "stringr"   "dplyr"     "purrr"     "readr"
```

```
##  [7] "tidyr"     "tibble"    "ggplot2"   "tidyverse" "stats"     "graphics"
## [13] "grDevices" "utils"     "datasets"  "methods"   "base"
##
## [[4]]
##  [1] "Metrics"   "reshape2"  "forcats"   "stringr"   "dplyr"     "purrr"
##  [7] "readr"     "tidyr"     "tibble"    "ggplot2"   "tidyverse" "stats"
## [13] "graphics"  "grDevices" "utils"     "datasets"  "methods"   "base"
##
## [[5]]
##  [1] "scales"    "Metrics"   "reshape2"  "forcats"   "stringr"   "dplyr"
##  [7] "purrr"     "readr"     "tidyr"     "tibble"    "ggplot2"   "tidyverse"
## [13] "stats"     "graphics"  "grDevices" "utils"     "datasets"  "methods"
## [19] "base"
##
## [[6]]
##  [1] "lubridate" "scales"    "Metrics"   "reshape2"  "forcats"   "stringr"
##  [7] "dplyr"     "purrr"     "readr"     "tidyr"     "tibble"    "ggplot2"
## [13] "tidyverse" "stats"     "graphics"  "grDevices" "utils"     "datasets"
## [19] "methods"   "base"
##
## [[7]]
##  [1] "RSocrata"  "lubridate" "scales"    "Metrics"   "reshape2"  "forcats"
##  [7] "stringr"   "dplyr"     "purrr"     "readr"     "tidyr"     "tibble"
## [13] "ggplot2"   "tidyverse" "stats"     "graphics"  "grDevices" "utils"
## [19] "datasets"  "methods"   "base"
##
## [[8]]
##  [1] "zoo"       "RSocrata"  "lubridate" "scales"    "Metrics"   "reshape2"
##  [7] "forcats"   "stringr"   "dplyr"     "purrr"     "readr"     "tidyr"
## [13] "tibble"    "ggplot2"   "tidyverse" "stats"     "graphics"  "grDevices"
## [19] "utils"     "datasets"  "methods"   "base"
##
## [[9]]
##  [1] "xts"       "zoo"       "RSocrata"  "lubridate" "scales"    "Metrics"
##  [7] "reshape2"  "forcats"   "stringr"   "dplyr"     "purrr"     "readr"
## [13] "tidyr"     "tibble"    "ggplot2"   "tidyverse" "stats"     "graphics"
## [19] "grDevices" "utils"     "datasets"  "methods"   "base"
##
## [[10]]
##  [1] "httr"      "xts"       "zoo"       "RSocrata"  "lubridate" "scales"
##  [7] "Metrics"   "reshape2"  "forcats"   "stringr"   "dplyr"     "purrr"
## [13] "readr"     "tidyr"     "tibble"    "ggplot2"   "tidyverse" "stats"
## [19] "graphics"  "grDevices" "utils"     "datasets"  "methods"   "base"
##
## [[11]]
##  [1] "xlsx"      "httr"      "xts"       "zoo"       "RSocrata"  "lubridate"
##  [7] "scales"    "Metrics"   "reshape2"  "forcats"   "stringr"   "dplyr"
## [13] "purrr"     "readr"     "tidyr"     "tibble"    "ggplot2"   "tidyverse"
## [19] "stats"     "graphics"  "grDevices" "utils"     "datasets"  "methods"
## [25] "base"
##
## [[12]]
##  [1] "readxl"    "xlsx"      "httr"      "xts"       "zoo"       "RSocrata"
##  [7] "lubridate" "scales"    "Metrics"   "reshape2"  "forcats"   "stringr"
```

```
## [13] "dplyr"     "purrr"     "readr"     "tidyr"     "tibble"    "ggplot2"
## [19] "tidyverse" "stats"     "graphics"  "grDevices" "utils"     "datasets"
## [25] "methods"   "base"
##
## [[13]]
##  [1] "sjmisc"    "readxl"    "xlsx"      "httr"      "xts"       "zoo"
##  [7] "RSocrata"  "lubridate" "scales"    "Metrics"   "reshape2"  "forcats"
## [13] "stringr"   "dplyr"     "purrr"     "readr"     "tidyr"     "tibble"
## [19] "ggplot2"   "tidyverse" "stats"     "graphics"  "grDevices" "utils"
## [25] "datasets"  "methods"   "base"
##
## [[14]]
##  [1] "sjmisc"    "readxl"    "xlsx"      "httr"      "xts"       "zoo"
##  [7] "RSocrata"  "lubridate" "scales"    "Metrics"   "reshape2"  "forcats"
## [13] "stringr"   "dplyr"     "purrr"     "readr"     "tidyr"     "tibble"
## [19] "ggplot2"   "tidyverse" "stats"     "graphics"  "grDevices" "utils"
## [25] "datasets"  "methods"   "base"
##
## [[15]]
##  [1] "rpart"     "sjmisc"    "readxl"    "xlsx"      "httr"      "xts"
##  [7] "zoo"       "RSocrata"  "lubridate" "scales"    "Metrics"   "reshape2"
## [13] "forcats"   "stringr"   "dplyr"     "purrr"     "readr"     "tidyr"
## [19] "tibble"    "ggplot2"   "tidyverse" "stats"     "graphics"  "grDevices"
## [25] "utils"     "datasets"  "methods"   "base"
##
## [[16]]
##  [1] "rattle"    "bitops"    "rpart"     "sjmisc"    "readxl"    "xlsx"
##  [7] "httr"      "xts"       "zoo"       "RSocrata"  "lubridate" "scales"
## [13] "Metrics"   "reshape2"  "forcats"   "stringr"   "dplyr"     "purrr"
## [19] "readr"     "tidyr"     "tibble"    "ggplot2"   "tidyverse" "stats"
## [25] "graphics"  "grDevices" "utils"     "datasets"  "methods"   "base"
##
## [[17]]
##  [1] "randomForest" "rattle"       "bitops"       "rpart"        "sjmisc"
##  [6] "readxl"       "xlsx"         "httr"         "xts"          "zoo"
## [11] "RSocrata"     "lubridate"    "scales"       "Metrics"      "reshape2"
## [16] "forcats"      "stringr"      "dplyr"        "purrr"        "readr"
## [21] "tidyr"        "tibble"       "ggplot2"      "tidyverse"    "stats"
## [26] "graphics"     "grDevices"    "utils"        "datasets"     "methods"
## [31] "base"
##
## [[18]]
##  [1] "gsubfn"       "proto"        "randomForest" "rattle"       "bitops"
##  [6] "rpart"        "sjmisc"       "readxl"       "xlsx"         "httr"
## [11] "xts"          "zoo"          "RSocrata"     "lubridate"    "scales"
## [16] "Metrics"      "reshape2"     "forcats"      "stringr"      "dplyr"
## [21] "purrr"        "readr"        "tidyr"        "tibble"       "ggplot2"
## [26] "tidyverse"    "stats"        "graphics"     "grDevices"    "utils"
## [31] "datasets"     "methods"      "base"
##
## [[19]]
##  [1] "e1071"        "gsubfn"       "proto"        "randomForest" "rattle"
##  [6] "bitops"       "rpart"        "sjmisc"       "readxl"       "xlsx"
## [11] "httr"         "xts"          "zoo"          "RSocrata"     "lubridate"
```

```
## [16] "scales"       "Metrics"      "reshape2"     "forcats"      "stringr"
## [21] "dplyr"        "purrr"        "readr"        "tidyr"        "tibble"
## [26] "ggplot2"      "tidyverse"    "stats"        "graphics"     "grDevices"
## [31] "utils"        "datasets"     "methods"      "base"
```

```
# Import list of time series file names
data_ts_list = list.files(path='data/', pattern="EQ", full.names=TRUE)

# Load and ttrim names of data frames
data_name_list <- NULL #initialise list name for list of data frames
data_name_list_trimed <- NULL
for (i in 1:length(data_ts_list)){ # extract titles of each data frames and trim the strings into more
  data_name_list[i] <- gsub( ".*Employed (.+) job .*", "\\1",
                             as.character(colnames(
                               read_excel(
                                 data_ts_list[i], sheet = 3,range = cell_rows(2:2)))
                               )
                             )
  data_name_list_trimed[i] <- gsub(" ", "_", paste0("Employed ", data_name_list[i], " job"))
}

# Import time series files and store as tibble
for (i in 1:length(data_ts_list)){
  data_imported <- as_tibble(read_excel(data_ts_list[i],
                                        skip = 3,
                                        sheet = 3
                                        ), trimws("both"))
  names(data_imported)[1] <- "Date"
  names(data_imported)[grepl('v', names(data_imported))] <- "Occupation"
  names(data_imported) <- gsub(" ", "_", names(data_imported))
  data_imported$Occupation <- gsub("[[:digit:]]", "", data_imported$Occupation)
  assign(paste(data_name_list_trimed[i]), data_imported) # rename each data frame imported to start wit
}

##################### Part B #####################
# Merge data frames into a master data frame
master_df <- get(data_name_list_trimed[1]) #initialise empty tibble data
for (i in 2:length(data_name_list_trimed)){ #merge every tibble data imported; data
  master_df <- merge(master_df, get(data_name_list_trimed[2]),
                     by = c("Date", "Occupation"), all.x = TRUE)
}
```

```
## Warning in merge.data.frame(master_df, get(data_name_list_trimed[2]), by =
## c("Date", : column names 'Sex.x', 'Employed_full-time_('000).x', 'Employed_part-
## time_('000).x', 'Number_of_hours_actually_worked_in_all_jobs_(employed_full-
## time)_('000_Hours).x',
## 'Number_of_hours_actually_worked_in_all_jobs_(employed_part-
## time)_('000_Hours).x', 'Sex.y', 'Employed_full-time_('000).y', 'Employed_part-
## time_('000).y', 'Number_of_hours_actually_worked_in_all_jobs_(employed_full-
## time)_('000_Hours).y',
## 'Number_of_hours_actually_worked_in_all_jobs_(employed_part-
## time)_('000_Hours).y' are duplicated in the result
```

```
## Warning in merge.data.frame(master_df, get(data_name_list_trimed[2]), by =
```

```
## c("Date", : column names 'Sex.x', 'Employed_full-time_('000).x', 'Employed_part-
## time_('000).x', 'Number_of_hours_actually_worked_in_all_jobs_(employed_full-
## time)_('000_Hours).x',
## 'Number_of_hours_actually_worked_in_all_jobs_(employed_part-
## time)_('000_Hours).x', 'Sex.y', 'State_and_territory_(STT):_ASGS_(2011).x',
## 'Employed_full-time_('000).y', 'Employed_part-time_('000).y',
## 'Number_of_hours_actually_worked_in_all_jobs_(employed_full-
## time)_('000_Hours).y',
## 'Number_of_hours_actually_worked_in_all_jobs_(employed_part-
## time)_('000_Hours).y', 'State_and_territory_(STT):_ASGS_(2011).y' are duplicated
## in the result

## Warning in merge.data.frame(master_df, get(data_name_list_trimed[2]), by =
## c("Date", : column names 'Sex.x', 'Employed_full-time_('000).x', 'Employed_part-
## time_('000).x', 'Number_of_hours_actually_worked_in_all_jobs_(employed_full-
## time)_('000_Hours).x',
## 'Number_of_hours_actually_worked_in_all_jobs_(employed_part-
## time)_('000_Hours).x', 'Sex.y', 'State_and_territory_(STT):_ASGS_(2011).x',
## 'Employed_full-time_('000).y', 'Employed_part-time_('000).y',
## 'Number_of_hours_actually_worked_in_all_jobs_(employed_full-
## time)_('000_Hours).y',
## 'Number_of_hours_actually_worked_in_all_jobs_(employed_part-
## time)_('000_Hours).y', 'Sex.x', 'State_and_territory_(STT):_ASGS_(2011).y',
## 'Employed_full-time_('000).x', 'Employed_part-time_('000).x',
## 'Number_of_hours_actually_worked_in_all_jobs_(employed_full-
## time)_('000_Hours).x',
## 'Number_of_hours_actually_worked_in_all_jobs_(employed_part-
## time)_('000_Hours).x', 'Sex.y', 'Employed_full-time_('000).y', 'Employed_part-
## time_('000).y', 'Number_of_hours_actually_worked_in_all_jobs_(employed_full-
## time)_('000_Hours).y',
## 'Number_of_hours_actually_worked_in_all_jobs_(employed_part-
## time)_('000_Hours).y' are duplicated in the result

## Warning in merge.data.frame(master_df, get(data_name_list_trimed[2]), by =
## c("Date", : column names 'Sex.x', 'Employed_full-time_('000).x', 'Employed_part-
## time_('000).x', 'Number_of_hours_actually_worked_in_all_jobs_(employed_full-
## time)_('000_Hours).x',
## 'Number_of_hours_actually_worked_in_all_jobs_(employed_part-
## time)_('000_Hours).x', 'Sex.y', 'State_and_territory_(STT):_ASGS_(2011).x',
## 'Employed_full-time_('000).y', 'Employed_part-time_('000).y',
## 'Number_of_hours_actually_worked_in_all_jobs_(employed_full-
## time)_('000_Hours).y',
## 'Number_of_hours_actually_worked_in_all_jobs_(employed_part-
## time)_('000_Hours).y', 'Sex.x', 'State_and_territory_(STT):_ASGS_(2011).y',
## 'Employed_full-time_('000).x', 'Employed_part-time_('000).x',
## 'Number_of_hours_actually_worked_in_all_jobs_(employed_full-
## time)_('000_Hours).x',
## 'Number_of_hours_actually_worked_in_all_jobs_(employed_part-
## time)_('000_Hours).x', 'Sex.y', 'State_and_territory_(STT):_ASGS_(2011).x',
## 'Employed_full-time_('000).y', 'Employed_part-time_('000).y',
## 'Number_of_hours_actually_worked_in_all_jobs_(employed_full-
## time)_('000_Hours).y',
## 'Number_of_hours_actually_worked_in_all_jobs_(employed_part-
## time)_('000_Hours).y', 'State_and_territory_(STT):_ASGS_(2011).y' are duplicated
```

```
## in the result

## Warning in merge.data.frame(master_df, get(data_name_list_trimed[2]), by =
## c("Date", : column names 'Sex.x', 'Employed_full-time_('000).x', 'Employed_part-
## time_('000).x', 'Number_of_hours_actually_worked_in_all_jobs_(employed_full-
## time)_('000_Hours).x',
## 'Number_of_hours_actually_worked_in_all_jobs_(employed_part-
## time)_('000_Hours).x', 'Sex.y', 'State_and_territory_(STT):_ASGS_(2011).x',
## 'Employed_full-time_('000).y', 'Employed_part-time_('000).y',
## 'Number_of_hours_actually_worked_in_all_jobs_(employed_full-
## time)_('000_Hours).y',
## 'Number_of_hours_actually_worked_in_all_jobs_(employed_part-
## time)_('000_Hours).y', 'Sex.x', 'State_and_territory_(STT):_ASGS_(2011).y',
## 'Employed_full-time_('000).x', 'Employed_part-time_('000).x',
## 'Number_of_hours_actually_worked_in_all_jobs_(employed_full-
## time)_('000_Hours).x',
## 'Number_of_hours_actually_worked_in_all_jobs_(employed_part-
## time)_('000_Hours).x', 'Sex.y', 'State_and_territory_(STT):_ASGS_(2011).x',
## 'Employed_full-time_('000).y', 'Employed_part-time_('000).y',
## 'Number_of_hours_actually_worked_in_all_jobs_(employed_full-
## time)_('000_Hours).y',
## 'Number_of_hours_actually_worked_in_all_jobs_(employed_part-
## time)_('000_Hours).y', 'Sex.x', 'State_and_territory_(STT):_ASGS_(2011).y',
## 'Employed_full-time_('000).x', 'Employed_part-time_('000).x',
## 'Number_of_hours_actually_worked_in_all_jobs_(employed_full-
## time)_('000_Hours).x',
## 'Number_of_hours_actually_worked_in_all_jobs_(employed_part-
## time)_('000_Hours).x', 'Sex.y', 'Employed_full-time_('000).y', 'Employed_part-
## time_('000).y', 'Number_of_hours_actually_worked_in_all_jobs_(employed_full-
## time)_('000_Hours).y',
## 'Number_of_hours_actually_worked_in_all_jobs_(employed_part-
## time)_('000_Hours).y' are duplicated in the result

## Warning in merge.data.frame(master_df, get(data_name_list_trimed[2]), by =
## c("Date", : column names 'Sex.x', 'Employed_full-time_('000).x', 'Employed_part-
## time_('000).x', 'Number_of_hours_actually_worked_in_all_jobs_(employed_full-
## time)_('000_Hours).x',
## 'Number_of_hours_actually_worked_in_all_jobs_(employed_part-
## time)_('000_Hours).x', 'Sex.y', 'State_and_territory_(STT):_ASGS_(2011).x',
## 'Employed_full-time_('000).y', 'Employed_part-time_('000).y',
## 'Number_of_hours_actually_worked_in_all_jobs_(employed_full-
## time)_('000_Hours).y',
## 'Number_of_hours_actually_worked_in_all_jobs_(employed_part-
## time)_('000_Hours).y', 'Sex.x', 'State_and_territory_(STT):_ASGS_(2011).y',
## 'Employed_full-time_('000).x', 'Employed_part-time_('000).x',
## 'Number_of_hours_actually_worked_in_all_jobs_(employed_full-
## time)_('000_Hours).x',
## 'Number_of_hours_actually_worked_in_all_jobs_(employed_part-
## time)_('000_Hours).x', 'Sex.y', 'State_and_territory_(STT):_ASGS_(2011).x',
## 'Employed_full-time_('000).y', 'Employed_part-time_('000).y',
## 'Number_of_hours_actually_worked_in_all_jobs_(employed_full-
## time)_('000_Hours).y',
## 'Number_of_hours_actually_worked_in_all_jobs_(employed_part-
## time)_('000_Hours).y', 'Sex.x', 'State_and_territory_(STT):_ASGS_(2011).y',
```

```
## 'Employed_full-time_('000).x', 'Employed_part-time_('000).x',
## 'Number_of_hours_actually_worked_in_all_jobs_(employed_full-
## time)_('000_Hours).x',
## 'Number_of_hours_actually_worked_in_all_jobs_(employed_part-
## time)_('000_Hours).x', 'Sex.y', 'State_and_territory_(STT):_ASGS_(2011).x',
## 'Employed_full-time_('000).y', 'Employed_part-time_('000).y',
## 'Number_of_hours_actually_worked_in_all_jobs_(employed_full-
## time)_('000_Hours).y',
## 'Number_of_hours_actually_worked_in_all_jobs_(employed_part-
## time)_('000_Hours).y', 'State_and_territory_(STT):_ASGS_(2011).y' are duplicated
## in the result

## Warning in merge.data.frame(master_df, get(data_name_list_trimed[2]), by =
## c("Date", : column names 'Sex.x', 'Employed_full-time_('000).x', 'Employed_part-
## time_('000).x', 'Number_of_hours_actually_worked_in_all_jobs_(employed_full-
## time)_('000_Hours).x',
## 'Number_of_hours_actually_worked_in_all_jobs_(employed_part-
## time)_('000_Hours).x', 'Sex.y', 'State_and_territory_(STT):_ASGS_(2011).x',
## 'Employed_full-time_('000).y', 'Employed_part-time_('000).y',
## 'Number_of_hours_actually_worked_in_all_jobs_(employed_full-
## time)_('000_Hours).y',
## 'Number_of_hours_actually_worked_in_all_jobs_(employed_part-
## time)_('000_Hours).y', 'Sex.x', 'State_and_territory_(STT):_ASGS_(2011).y',
## 'Employed_full-time_('000).x', 'Employed_part-time_('000).x',
## 'Number_of_hours_actually_worked_in_all_jobs_(employed_full-
## time)_('000_Hours).x',
## 'Number_of_hours_actually_worked_in_all_jobs_(employed_part-
## time)_('000_Hours).x', 'Sex.y', 'State_and_territory_(STT):_ASGS_(2011).x',
## 'Employed_full-time_('000).y', 'Employed_part-time_('000).y',
## 'Number_of_hours_actually_worked_in_all_jobs_(employed_full-
## time)_('000_Hours).y',
## 'Number_of_hours_actually_worked_in_all_jobs_(employed_part-
## time)_('000_Hours).y', 'Sex.x', 'State_and_territory_(STT):_ASGS_(2011).y',
## 'Employed_full-time_('000).x', 'Employed_part-time_('000).x',
## 'Number_of_hours_actually_worked_in_all_jobs_(employed_full-
## time)_('000_Hours).x',
## 'Number_of_hours_actually_worked_in_all_jobs_(employed_part-
## time)_('000_Hours).x', 'Sex.y', 'State_and_territory_(STT):_ASGS_(2011).x',
## 'Employed_full-time_('000).y', 'Employed_part-time_('000).y',
## 'Number_of_hours_actually_worked_in_all_jobs_(employed_full-
## time)_('000_Hours).y',
## 'Number_of_hours_actually_worked_in_all_jobs_(employed_part-
## time)_('000_Hours).y', 'Sex.x', 'State_and_territory_(STT):_ASGS_(2011).y',
## 'Employed_full-time_('000).x', 'Employed_part-time_('000).x',
## 'Number_of_hours_actually_worked_in_all_jobs_(employed_full-
## time)_('000_Hours).x',
## 'Number_of_hours_actually_worked_in_all_jobs_(employed_part-
## time)_('000_Hours).x', 'Sex.y', 'Employed_full-time_('000).y', 'Employed_part-
## time_('000).y', 'Number_of_hours_actually_worked_in_all_jobs_(employed_full-
## time)_('000_Hours).y',
## 'Number_of_hours_actually_worked_in_all_jobs_(employed_part-
## time)_('000_Hours).y' are duplicated in the result

## Warning in merge.data.frame(master_df, get(data_name_list_trimed[2]), by =
```

```
## c("Date", : column names 'Sex.x', 'Employed_full-time_('000).x', 'Employed_part-
## time_('000).x', 'Number_of_hours_actually_worked_in_all_jobs_(employed_full-
## time)_('000_Hours).x',
## 'Number_of_hours_actually_worked_in_all_jobs_(employed_part-
## time)_('000_Hours).x', 'Sex.y', 'State_and_territory_(STT):_ASGS_(2011).x',
## 'Employed_full-time_('000).y', 'Employed_part-time_('000).y',
## 'Number_of_hours_actually_worked_in_all_jobs_(employed_full-
## time)_('000_Hours).y',
## 'Number_of_hours_actually_worked_in_all_jobs_(employed_part-
## time)_('000_Hours).y', 'Sex.x', 'State_and_territory_(STT):_ASGS_(2011).y',
## 'Employed_full-time_('000).x', 'Employed_part-time_('000).x',
## 'Number_of_hours_actually_worked_in_all_jobs_(employed_full-
## time)_('000_Hours).x',
## 'Number_of_hours_actually_worked_in_all_jobs_(employed_part-
## time)_('000_Hours).x', 'Sex.y', 'State_and_territory_(STT):_ASGS_(2011).x',
## 'Employed_full-time_('000).y', 'Employed_part-time_('000).y',
## 'Number_of_hours_actually_worked_in_all_jobs_(employed_full-
## time)_('000_Hours).y',
## 'Number_of_hours_actually_worked_in_all_jobs_(employed_part-
## time)_('000_Hours).y', 'Sex.x', 'State_and_territory_(STT):_ASGS_(2011).y',
## 'Employed_full-time_('000).x', 'Employed_part-time_('000).x',
## 'Number_of_hours_actually_worked_in_all_jobs_(employed_full-
## time)_('000_Hours).x',
## 'Number_of_hours_actually_worked_in_all_jobs_(employed_part-
## time)_('000_Hours).x', 'Sex.y', 'State_and_territory_(STT):_ASGS_(2011).x',
## 'Employed_full-time_('000).y', 'Employed_part-time_('000).y',
## 'Number_of_hours_actually_worked_in_all_jobs_(employed_full-
## time)_('000_Hours).y',
## 'Number_of_hours_actually_worked_in_all_jobs_(employed_part-
## time)_('000_Hours).y', 'Sex.x', 'State_and_territory_(STT):_ASGS_(2011).y',
## 'Employed_full-time_('000).x', 'Employed_part-time_('000).x',
## 'Number_of_hours_actually_worked_in_all_jobs_(employed_full-
## time)_('000_Hours).x',
## 'Number_of_hours_actually_worked_in_all_jobs_(employed_part-
## time)_('000_Hours).x', 'Sex.y', 'State_and_territory_(STT):_ASGS_(2011).x',
## 'Employed_full-time_('000).y', 'Employed_part-time_('000).y',
## 'Number_of_hours_actually_worked_in_all_jobs_(employed_full-
## time)_('000_Hours).y',
## 'Number_of_hours_actually_worked_in_all_jobs_(employed_part-
## time)_('000_Hours).y', 'State_and_territory_(STT):_ASGS_(2011).y' are duplicated
## in the result
```

```r
# Remove columns where all values are NA
master_df <- Filter(function(x)!all(is.na(x)), master_df)

# Remove columns where NA values are more than 90% of records
master_df <- master_df[, which(colMeans(!is.na(master_df)) >= 0.1)] # filter the columns with more than

# Replace the remaining NAs with zero, if exist
nums <- unlist(lapply(master_df, is.numeric)) # select only numeric columns
for(i in 1:ncol(master_df[, nums])){
  return_numeric <- as.numeric(as.vector(as.numeric(unlist(master_df[, nums][, i])))) # converts import
  master_df[, nums][is.na(master_df[, nums][,i]), i]<- 0 # replace missing values with Zero
  }
```

```r
# Remove unnecessary characters from col names
names(master_df) <- sub("\\.x", "", names(master_df))

# Encode date column into Date format
master_df$Date <- as.Date(master_df$Date, format = "%Y-%m-%d")

# Change column names in a data frame
# format: "new column name to be assigned" = "previous column name"
master_df <- master_df %>%
  rename(
    Area = `Greater_capital_city_and_rest_of_state_(GCCSA):_ASGS_(2011)`,
    Employed_full_time = `Employed_full-time_('000)`,
    Employed_part_time = `Employed_part-time_('000)`,
    Number_of_hours_worked_full_time = `Number_of_hours_actually_worked_in_all_jobs_(employed_full-time)
    Number_of_hours_worked_part_time = `Number_of_hours_actually_worked_in_all_jobs_(employed_part-time)
    )

# Derive a column from the master data frame to indicate relevance to tech sector
tech_value <- c("ICT", "Info", "Tech", "Engin", "IT", "Sci", "Math", "Mechanic", "Network") # Keywords
master_df$Tech <- grepl(paste(tech_value, collapse="|"),master_df$Occupation, ignore.case = TRUE)

# Print occupation types specified as being relevant to tech sector
occupation_tech <- unique(grep(paste(tech_value, collapse="|"),
                               master_df$Occupation, value=TRUE, ignore.case = TRUE))

# Add column into master data frame indicating combined number of jobs and worked hour
master_df <- master_df %>%
  group_by(Date) %>%
  mutate(Employed_total = Employed_full_time + Employed_part_time,
         Number_of_hours_worked_avg = (Number_of_hours_worked_full_time + Number_of_hours_worked_part_t

# check for problems
assertthat::assert_that(nrow(problems(master_df)) == 0, # assert that there is NO problems
                        msg="There is still problem/s, which you need to fix first")
```

```
## [1] TRUE
```

```r
# print data frame dimensions
cat("data dimensions are: ", dim(master_df))
```

```
## data dimensions are:  78637 11
```

```r
# Export the completed records
write_csv(master_df, "data/master_df_all.csv")

# Derive a data frame only containing occupation types relevant to tech sector
tech_df <- master_df %>% filter(Tech == TRUE)

# Export the completed records
write_csv(tech_df, "data/master_df_tech.csv")

##################### Part C #####################
```

```r
# Generate data frame arranged by area and occupation
mdata <- melt(master_df, id=c("Date", "Occupation", "Sex", "Area", "Tech"))
df_by_occupation_area <- dcast(mdata, Date + Occupation + Area + Tech ~ variable, mean)

# Generate data frame arranged by occupaiton and gender
df_by_occupation_gender <- dcast(mdata, Date + Occupation + Tech + Sex ~ variable, mean)

# Generate data frame arranged by occupation
df_by_occupation <- dcast(mdata, Date + Occupation + Tech ~ variable, mean)

# Generate newly arranged data frame only containing occupation types relevant to tech sector
mdata_tech <- master_df %>%
  filter(Tech == TRUE) %>%
  # mutate(Employed_total_tech = Employed_full_time + Employed_part_time,
  #        Number_of_hours_worked_avg_tech = (Number_of_hours_worked_full_time + Number_of_hours_worked_
  melt(id=c("Date", "Occupation", "Sex", "Area", "Tech"))
df_tech <- dcast(mdata_tech, Date + Sex + Area ~ variable, mean)

# Generate tech sector data frame arranged by gender
df_tech_by_area <- dcast(mdata_tech, Date + Sex ~ variable, mean)

# Generate tech sector data frame arranged by area
df_tech_by_gender <- dcast(mdata_tech, Date + Area ~ variable, mean)

# Generate tech sector data total
df_tech_total <- dcast(mdata_tech, Date ~ variable, mean)

# Generate newly rarranged data frame with occupation types other than tech sector
mdata_others <- master_df %>%
  filter(Tech == FALSE) %>%
  # mutate(Employed_total_others = Employed_full_time + Employed_part_time,
  #        Number_of_hours_worked_avg_others = (Number_of_hours_worked_full_time + Number_of_hours_work
  melt(id=c("Date", "Occupation", "Sex", "Area", "Tech"))
df_others <- dcast(mdata_others, Date + Sex + Area ~ variable, mean)

# Generate data frame arranged by gender
df_others_by_area <- dcast(mdata_others, Date + Sex ~ variable, mean)

# Generate data frame arranged by area
df_others_by_gender <- dcast(mdata_others, Date + Area ~ variable, mean)

# Generate data frame total
df_others_total <- dcast(mdata_others, Date ~ variable, mean)

# Print the date with the highest volume tech sector jobs
Highest_job_date_tech <- df_tech_total[which(df_tech_total$Employed_total == max(df_tech_total$Employed_
Highest_job_date_tech$Date
```

```
## [1] "2021-05-01"
```

```r
# Print the date with the highest volume in jobs excluding tech sector
Highest_job_date_others <- df_others_total[which(df_others_total$Employed_total == max(df_others_total$E
Highest_job_date_others$Date
```

```
## [1] "2021-11-01"
```

```
# Print the date with highest worked hours in tech sector
Highest_wokred_hour_tech <- df_tech_total[which(df_tech_total$Number_of_hours_worked_avg == max(df_tech_
Highest_wokred_hour_tech$Date
```

```
## [1] "2021-02-01"
```

```
# Print the date with highest worked hours in jobs excluding tech sector
Highest_wokred_hour_others <- df_others_total[which(df_others_total$Number_of_hours_worked_avg == max(d
Highest_wokred_hour_others$Date
```

```
## [1] "2021-11-01"
```

```
# Generate data frame comparing job volume development between tech sector and others
df_volume_job_tech_others <- data.frame(Date = df_tech_total$Date,
                                        Emplyed_tech = df_tech_total$Employed_total,
                                        Emplyed_others = df_others_total$Employed_total
)

# Generate data frame comparing number of hour worked between tech sector and others
df_worked_hour_tech_others <- data.frame(Date = df_tech_total$Date,
                                        Number_of_hours_worked_tech = df_tech_total$Number_of_hours_wor
                                        Number_of_hours_worked_others = df_others_total$Number_of_hours
)

# Generate data frame comparing job volumes in each occupation
mdata_others_by_occupation <- df_by_occupation %>%
  filter(Tech == FALSE) %>%
  group_by(Date) %>%
  select(-Tech, -Employed_full_time, -Employed_part_time, -Number_of_hours_worked_full_time,
         -Number_of_hours_worked_part_time)
mdata_others_volume_by_occupation <- melt(mdata_others_by_occupation, id=c("Date", "Occupation"))
mdata_others_colume_by_occupation <- dcast(mdata_others_volume_by_occupation, Date ~ Occupation, mean)
names(mdata_others_colume_by_occupation) <- gsub(" ", "_", names(mdata_others_colume_by_occupation))
df_job_volume_development_tech_others <- data.frame(Date = df_volume_job_tech_others$Date,
                                                    Employed_tech = df_volume_job_tech_others$Emplyed_t
                                                    )
df_job_volume_development_tech_others <- merge(df_job_volume_development_tech_others,
                                               mdata_others_colume_by_occupation)
names(df_job_volume_development_tech_others) <- gsubfn(".", list(" " = "_", "," = "_"),
                                                       names(df_job_volume_development_tech_others))

##################### Part D #####################
#print summary of data frame generated in the previous step
head(df_volume_job_tech_others)
```

```
##         Date Emplyed_tech Emplyed_others
## 1 1984-11-01     6.766399       13.44001
## 2 1985-02-01     6.747070       13.48750
## 3 1985-05-01     6.906259       13.68875
## 4 1985-08-01     6.992410       13.65062
## 5 1985-11-01     7.203985       14.08972
## 6 1986-02-01     7.345076       14.00195
```

```r
summary(df_volume_job_tech_others)
```

```
##       Date             Emplyed_tech    Emplyed_others
##  Min.   :1984-11-01   Min.   : 6.747   Min.   :13.44
##  1st Qu.:1994-02-01   1st Qu.: 8.174   1st Qu.:16.13
##  Median :2003-05-01   Median :11.069   Median :19.05
##  Mean   :2003-05-02   Mean   :11.893   Mean   :19.49
##  3rd Qu.:2012-08-01   3rd Qu.:14.952   3rd Qu.:22.60
##  Max.   :2021-11-01   Max.   :19.083   Max.   :26.31
```

```r
df_volume_job_tech_others %>% tail
```

```
##           Date Emplyed_tech Emplyed_others
## 144 2020-08-01     17.83269       25.00101
## 145 2020-11-01     18.38723       25.46150
## 146 2021-02-01     19.05385       25.62662
## 147 2021-05-01     19.08332       26.09547
## 148 2021-08-01     18.78730       25.79299
## 149 2021-11-01     18.87154       26.31336
```

```r
df_volume_job_tech_others %>% head
```

```
##         Date Emplyed_tech Emplyed_others
## 1 1984-11-01     6.766399       13.44001
## 2 1985-02-01     6.747070       13.48750
## 3 1985-05-01     6.906259       13.68875
## 4 1985-08-01     6.992410       13.65062
## 5 1985-11-01     7.203985       14.08972
## 6 1986-02-01     7.345076       14.00195
```

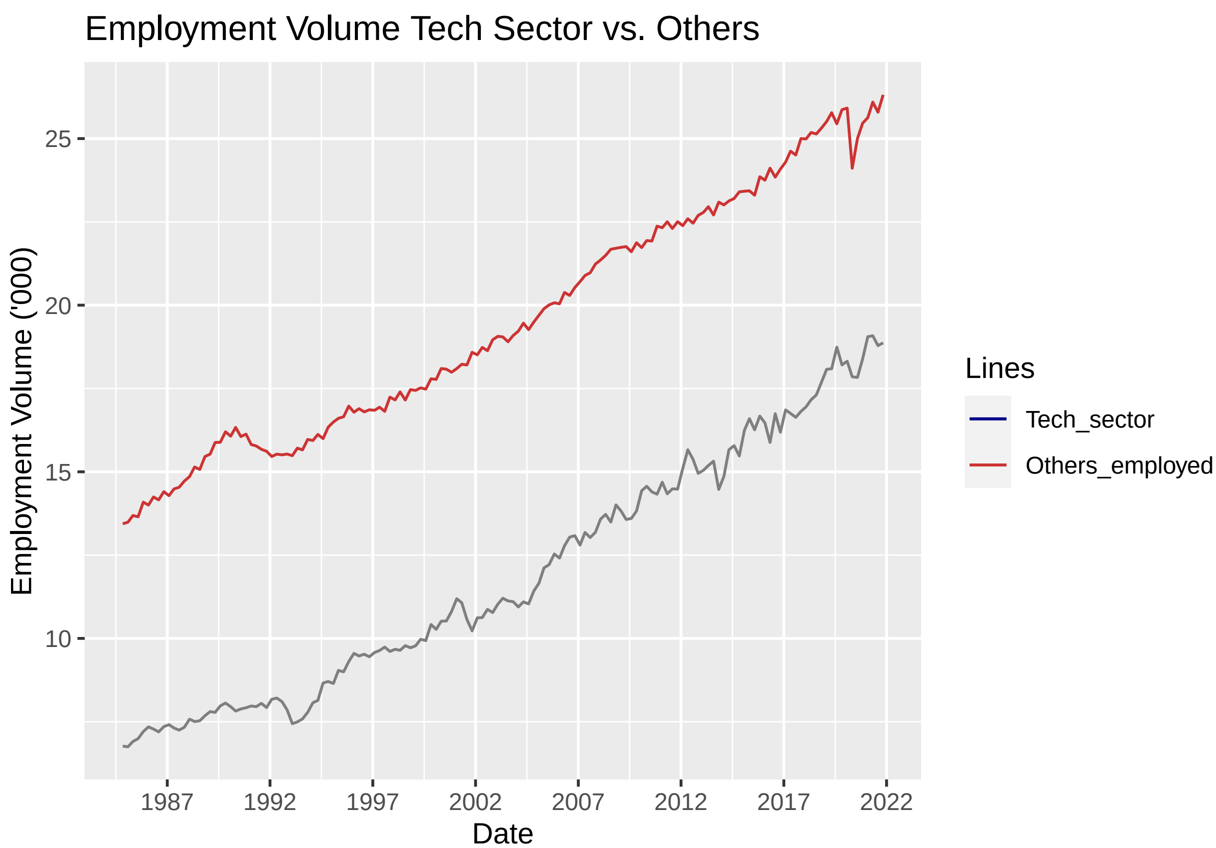
```r
min(df_volume_job_tech_others[, 1], na.rm=T)
```

```
## [1] "1984-11-01"
```

```r
max(df_volume_job_tech_others[, 1], na.rm=T)
```

```
## [1] "2021-11-01"
```

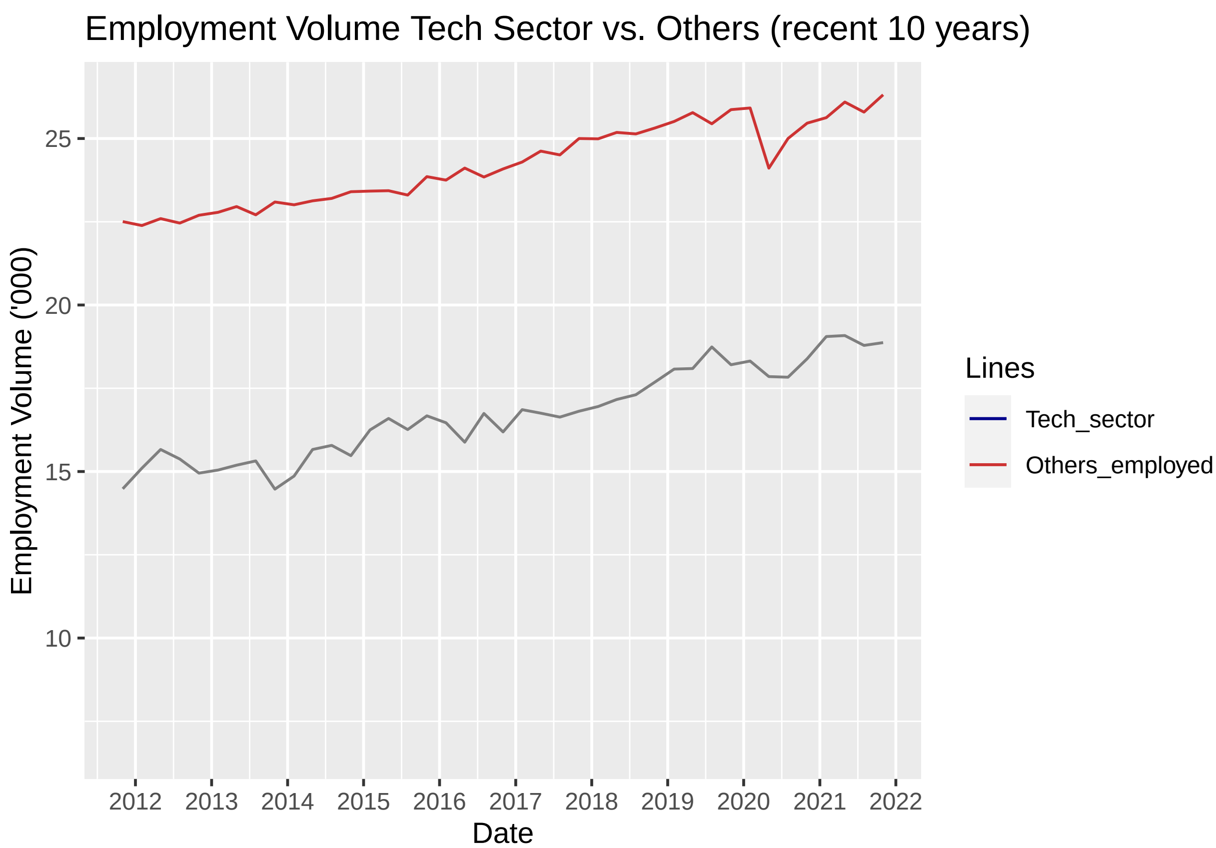
```r
# Generate a graph showing development of volume of jobs in tech sector compared to others
cols <- c("Tech_sector"="darkblue","Others_employed"="brown3") # map the color in order to generate leg
graph_employement_tech_others <-  df_volume_job_tech_others %>%
  ggplot(aes(x = Date)) +
  geom_line(aes(y = Emplyed_tech, color = "Tech_sector_employed")) +
  geom_line(aes(y = Emplyed_others, color = "Others_employed")) +
  scale_colour_manual(name="Lines",values=cols) +
  ggtitle("Employment Volume Tech Sector vs. Others") +
  scale_x_date(date_breaks = "5 year",
               labels=date_format("%Y")) +
  ggtitle("Employment Volume Tech Sector vs. Others") +
  ylab("Employment Volume ('000)")
print(graph_employement_tech_others)
```




```
# limit x axis to recent 10 years
cols <- c("Tech_sector"="darkblue","Others_employed"="brown3") # map the color in order to generate leg
graph_employement_tech_others_xlimited <-  df_volume_job_tech_others %>%
  ggplot(aes(x = Date)) +
  geom_line(aes(y = Emplyed_tech, color = "Tech_sector_employed")) +
  geom_line(aes(y = Emplyed_others, color = "Others_employed")) +
  scale_colour_manual(name="Lines",values=cols) +
  scale_x_date(date_breaks = "1 year",
               labels=date_format("%Y"),
               limits = as.Date(c(max(df_volume_job_tech_others$Date) - years(10),
                                  max(df_volume_job_tech_others$Date)))) +
  ggtitle("Employment Volume Tech Sector vs. Others (recent 10 years)") +
  ylab("Employment Volume ('000)")
print(graph_employement_tech_others_xlimited)
```

```
## Warning: Removed 108 row(s) containing missing values (geom_path).
```

```
## Warning: Removed 108 row(s) containing missing values (geom_path).
```




```r
##################### Part E #####################
# For reproducibility
set.seed(123)

# randomly select 70% of the number of observations
index <- sample(1:nrow(df_job_volume_development_tech_others),
                size = 0.7*nrow(df_job_volume_development_tech_others))

# subset other variables from the train data frame to include only the elements in the index
train <- df_job_volume_development_tech_others[index,]

# subset other variables from the test data frame to include only the elements in the index
test <- df_job_volume_development_tech_others [-index,]

nrow(train)
```

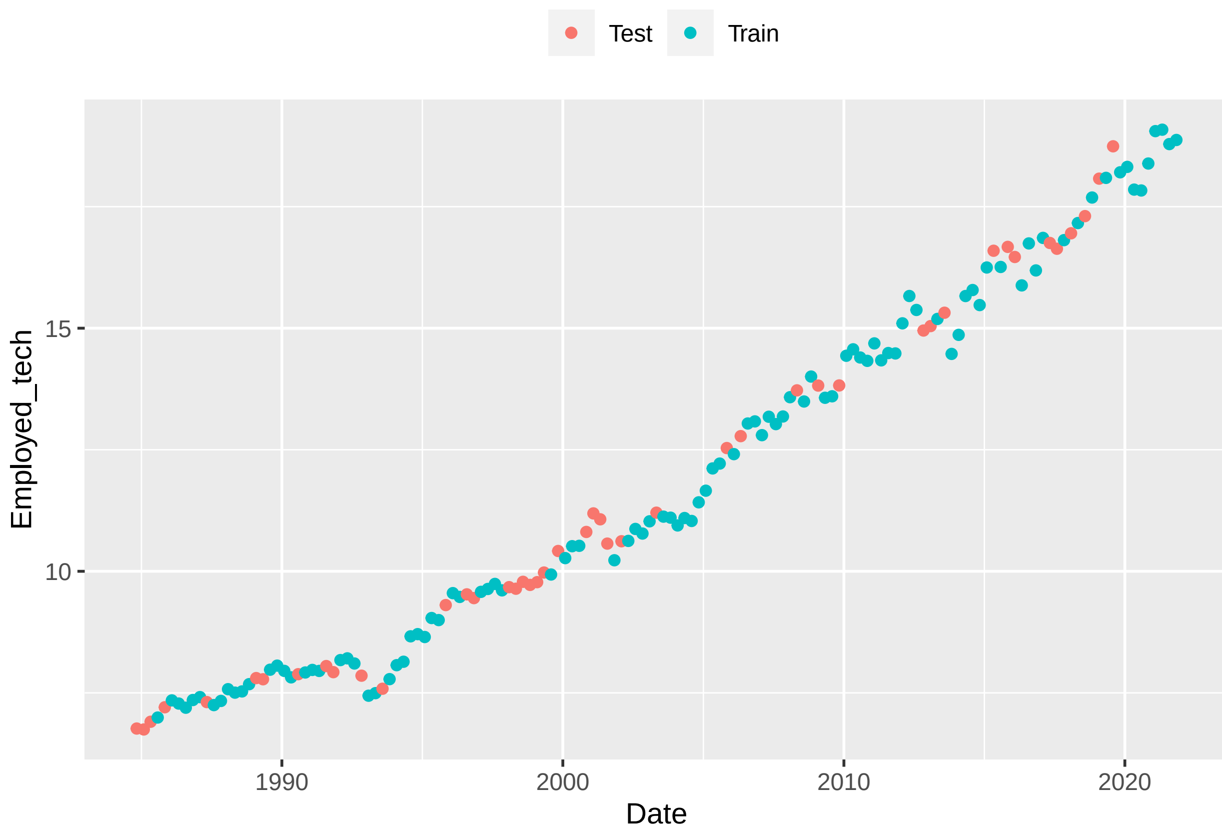
```
## [1] 104
```

```r
nrow(test)
```

```
## [1] 45
```

```r
# Create a data frame with train and test indicator
group <- rep(NA,nrow(train)+nrow(test))
```

```r
group <- ifelse(seq(1,nrow(train)+nrow(test)) %in% index,"Train","Test")
df <- data.frame(Date=df_job_volume_development_tech_others$Date,
                 Employed_tech=df_job_volume_development_tech_others$Employed_tech,group)

# Plot test vs. train data frames
ggplot(df,aes(x = Date,y = Employed_tech, color = group)) + geom_point() +
  scale_color_discrete(name="") + theme(legend.position="top")
```




```r
# Baseline model - predict the mean of the training data
best.guess <- mean(train$Employed_tech)

# Evaluate RMSE and MAE on the testing data
RMSE.baseline <- rmse(test$Employed_tech, best.guess)
RMSE.baseline
```

```
## [1] 3.623765
```

```r
MAE.baseline <- mae(test$Employed_tech, best.guess)
MAE.baseline
```

```
## [1] 3.248703
```

```r
#Multiple linear regression
# Create a multiple (log)linear regression model using the training data
lin.reg <- lm(log(Employed_tech+1) ~ ., data = train)

# Inspect the model
summary(lin.reg)
```

```
##
## Call:
## lm(formula = log(Employed_tech + 1) ~ ., data = train)
##
## Residuals:
##        Min         1Q    Median        3Q       Max
## -0.067223 -0.011122  0.000335  0.014383  0.046588
##
## Coefficients:
##                                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)                             1.452e+00  1.439e-01  10.087 3.03e-16
## Date                                    5.408e-05  8.240e-06   6.563 3.80e-09
## Accommodation_and_Food_Services        -9.539e-04  4.336e-04  -2.200   0.0305
## Administrative_and_Support_Services     1.331e-03  5.915e-04   2.250   0.0270
## Agriculture__Forestry_and_Fishing      -9.523e-05  2.754e-04  -0.346   0.7303
## Arts_and_Recreation_Services            1.523e-03  1.167e-03   1.305   0.1955
## Construction                            5.070e-04  2.603e-04   1.947   0.0548
## Education_and_Training                 -3.365e-04  3.740e-04  -0.900   0.3708
## Financial_and_Insurance_Services        1.229e-03  5.110e-04   2.405   0.0183
## Health_Care_and_Social_Assistance       1.631e-04  2.624e-04   0.622   0.5358
## Manufacturing                           3.343e-04  2.392e-04   1.398   0.1658
## Mining                                  2.827e-04  3.120e-04   0.906   0.3675
## Other_Services                          2.475e-04  4.784e-04   0.517   0.6062
## Public_Administration_and_Safety       -5.768e-04  5.074e-04  -1.137   0.2588
## Rental__Hiring_and_Real_Estate_Services -9.633e-04  8.457e-04  -1.139   0.2578
## Retail_Trade                           -1.997e-04  4.013e-04  -0.498   0.6200
## Transport__Postal_and_Warehousing       1.048e-03  4.787e-04   2.190   0.0312
## Wholesale_Trade                        -9.004e-05  3.794e-04  -0.237   0.8130
##
## (Intercept)                            ***
## Date                                   ***
## Accommodation_and_Food_Services        *
## Administrative_and_Support_Services    *
## Agriculture__Forestry_and_Fishing
## Arts_and_Recreation_Services
## Construction                           .
## Education_and_Training
## Financial_and_Insurance_Services       *
## Health_Care_and_Social_Assistance
## Manufacturing
## Mining
## Other_Services
## Public_Administration_and_Safety
## Rental__Hiring_and_Real_Estate_Services
## Retail_Trade
## Transport__Postal_and_Warehousing      *
```

```
## Wholesale_Trade
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.02672 on 86 degrees of freedom
## Multiple R-squared:  0.9927, Adjusted R-squared:  0.9913
## F-statistic: 688.7 on 17 and 86 DF,  p-value: < 2.2e-16
```

```r
# Multiplicative effect of "Mining" variable
exp(lin.reg$coefficients["Mining"])
```

```
##   Mining
## 1.000283
```

```r
# Apply the model to the testing data
test.pred.lin <- exp(predict(lin.reg,test))-1

# Evaluate the accuracy
RMSE.lin.reg <- rmse(test$Employed_tech, test.pred.lin)
RMSE.lin.reg
```

```
## [1] 0.3753509
```

```r
MAE.lin.reg <- mae(test$Employed_tech, test.pred.lin)
MAE.lin.reg
```

```
## [1] 0.2830631
```

```r
#Decision Tree
# rpart function applied to a numeric variable
rt <- rpart(Employed_tech ~ ., data=train)

fancyRpartPlot(rt)

# Predict and evaluate on the test set
test.pred.rtree <- predict(rt,test)

RMSE.rtree <- rmse(test$Employed_tech, test.pred.rtree)
RMSE.rtree
```

```
## [1] 0.7975804
```

```r
MAE.rtree <- mae(test$Employed_tech, test.pred.rtree)
MAE.rtree
```

```
## [1] 0.6478226
```

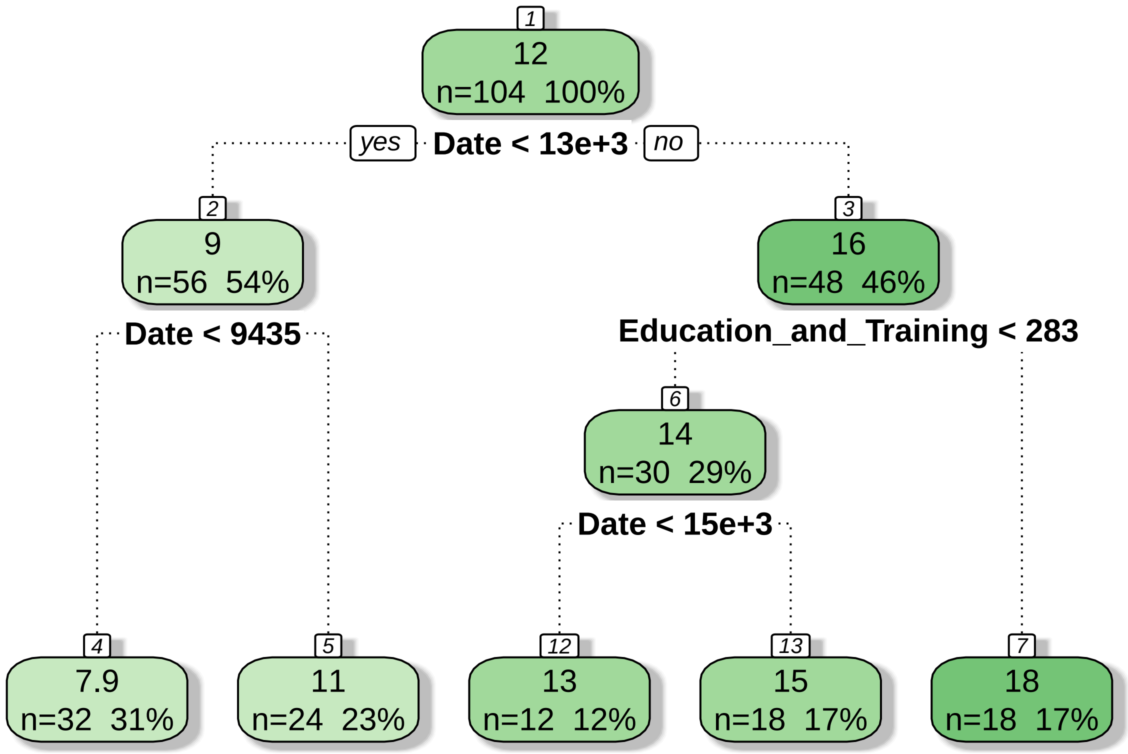
```r
# Cross-validation results (xerror)
printcp(rt)
```

```
##
## Regression tree:
## rpart(formula = Employed_tech ~ ., data = train)
##
## Variables actually used in tree construction:
## [1] Date                   Education_and_Training
##
## Root node error: 1399.5/104 = 13.457
##
## n= 104
##
##         CP nsplit rel error   xerror     xstd
## 1 0.781970      0  1.000000 1.031113 0.092787
## 2 0.093984      1  0.218030 0.238145 0.023953
## 3 0.075003      2  0.124046 0.161939 0.017625
## 4 0.014994      3  0.049043 0.091170 0.019745
## 5 0.010000      4  0.034049 0.083816 0.019797
```

```
# Compute optimal CP
min.xerror <- rt$cptable[which.min(rt$cptable[,"xerror"]),"CP"]
min.xerror
```

```
## [1] 0.01
```

```
# Prune the tree
rt.pruned <- prune(rt,cp = min.xerror)

# Plot pruned tree
fancyRpartPlot(rt.pruned)
```



Rattle 2022−Jan−17 05:08:27 dohye

```
# Evaluate pruned tree on the test set
test.pred.rtree.p <- predict(rt.pruned,test)
RMSE.rtree.pruned <- rmse(test$Employed_tech, test.pred.rtree.p)
RMSE.rtree.pruned
```

```
## [1] 0.7975804
```

```
MAE.rtree.pruned <- mae(test$Employed_tech, test.pred.rtree.p)
MAE.rtree.pruned
```

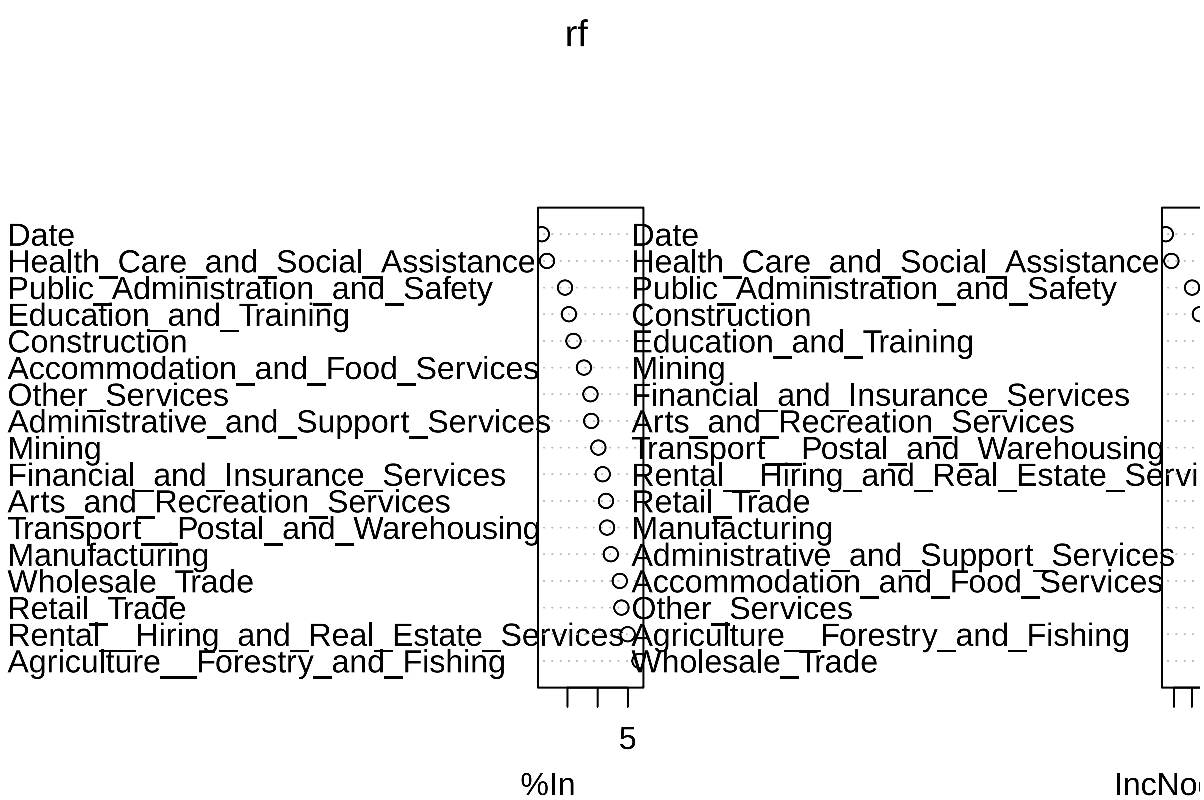
```
## [1] 0.6478226
```

```
# Random forests
set.seed(123)

# Create a model with 1000 trees
rf <- randomForest(Employed_tech ~ ., data = train, importance = TRUE, ntree=1000)

importance(rf)
```

```
##                                             %IncMSE IncNodePurity
## Date                                      19.269466    272.923392
## Accommodation_and_Food_Services           12.274719     11.650497
## Administrative_and_Support_Services       11.053875     16.062905
```

```
## Agriculture__Forestry_and_Fishing        3.040607      3.274280
## Arts_and_Recreation_Services             8.612227     56.883812
## Construction                            14.002774    177.765406
## Education_and_Training                  14.760791    110.250426
## Financial_and_Insurance_Services         9.156275     77.487982
## Health_Care_and_Social_Assistance       18.400583    257.211510
## Manufacturing                            7.808113     16.794758
## Mining                                   9.875452     84.375893
## Other_Services                          11.187192      7.221025
## Public_Administration_and_Safety        15.407323    199.478738
## Rental__Hiring_and_Real_Estate_Services  5.014103     20.224544
## Retail_Trade                             6.047478     19.547567
## Transport__Postal_and_Warehousing        8.427252     52.118162
## Wholesale_Trade                          6.327884      3.040388
```

```r
varImpPlot(rf)
```




```r
# Compute optimal number of trees
which.min(rf$mse)
```

```
## [1] 44
```

```r
# Calculate the importance of each variable
imp <- as.data.frame(sort(importance(rf)[,1],decreasing = TRUE),optional = T)
names(imp) <- "% Inc MSE"
imp
```

```
##                                        % Inc MSE
## Date                                   19.269466
## Health_Care_and_Social_Assistance      18.400583
## Public_Administration_and_Safety       15.407323
## Education_and_Training                 14.760791
## Construction                           14.002774
## Accommodation_and_Food_Services        12.274719
## Other_Services                         11.187192
## Administrative_and_Support_Services    11.053875
## Mining                                  9.875452
## Financial_and_Insurance_Services        9.156275
## Arts_and_Recreation_Services            8.612227
## Transport__Postal_and_Warehousing       8.427252
## Manufacturing                           7.808113
## Wholesale_Trade                         6.327884
## Retail_Trade                            6.047478
## Rental__Hiring_and_Real_Estate_Services 5.014103
## Agriculture__Forestry_and_Fishing       3.040607
```

```
# Predict and evaluate on the test set
test.pred.forest <- predict(rf,test)
RMSE.forest <- rmse(test$Employed_tech, test.pred.forest)
RMSE.forest
```

```
## [1] 0.3410522
```

```
MAE.forest <- mae(test$Employed_tech, test.pred.forest)
MAE.forest
```

```
## [1] 0.2469949
```

```
# SVM
# trainx <- train[ , purrr::map_lgl(train, is.numeric)] # leave only numeric varialbes
# Train SVM model using train set
svm.model <- svm(Employed_tech ~., scale = T,
                 data=train,kernel="radial",cost=100,gamma=0.1);
# testx <- test[ , purrr::map_lgl(test, is.numeric)]

# Predict and evaluate on the test set
svm.pred <- predict(svm.model, test,decision.values =TRUE);
summary(svm.model)
```

```
##
## Call:
## svm(formula = Employed_tech ~ ., data = train, kernel = "radial",
##     cost = 100, gamma = 0.1, scale = T)
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  radial
##        cost:  100
```

```
##       gamma:  0.1
##     epsilon:  0.1
##
##
## Number of Support Vectors:  33
```

```r
RMSE.svm <- rmse(test$Employed_tech, svm.pred)
MAE.svm <- mae(test$Employed_tech, svm.pred)

##################### Part F #####################
#Evaluate Results
# Create a data frame for error metrics of each method
accuracy <- data.frame(Method = c("Baseline","Linear Regression","Full tree","Pruned tree",
                                  "Random forest", "Support Vector"),
                       RMSE   = c(RMSE.baseline,RMSE.lin.reg,
                                  RMSE.rtree,RMSE.rtree.pruned,RMSE.forest,RMSE.svm),
                       MAE    = c(MAE.baseline,MAE.lin.reg,MAE.rtree,
                                  MAE.rtree.pruned,MAE.forest,MAE.svm))

# Round the values and print the table
accuracy$RMSE <- round(accuracy$RMSE,2)
accuracy$MAE <- round(accuracy$MAE,2)

accuracy
```

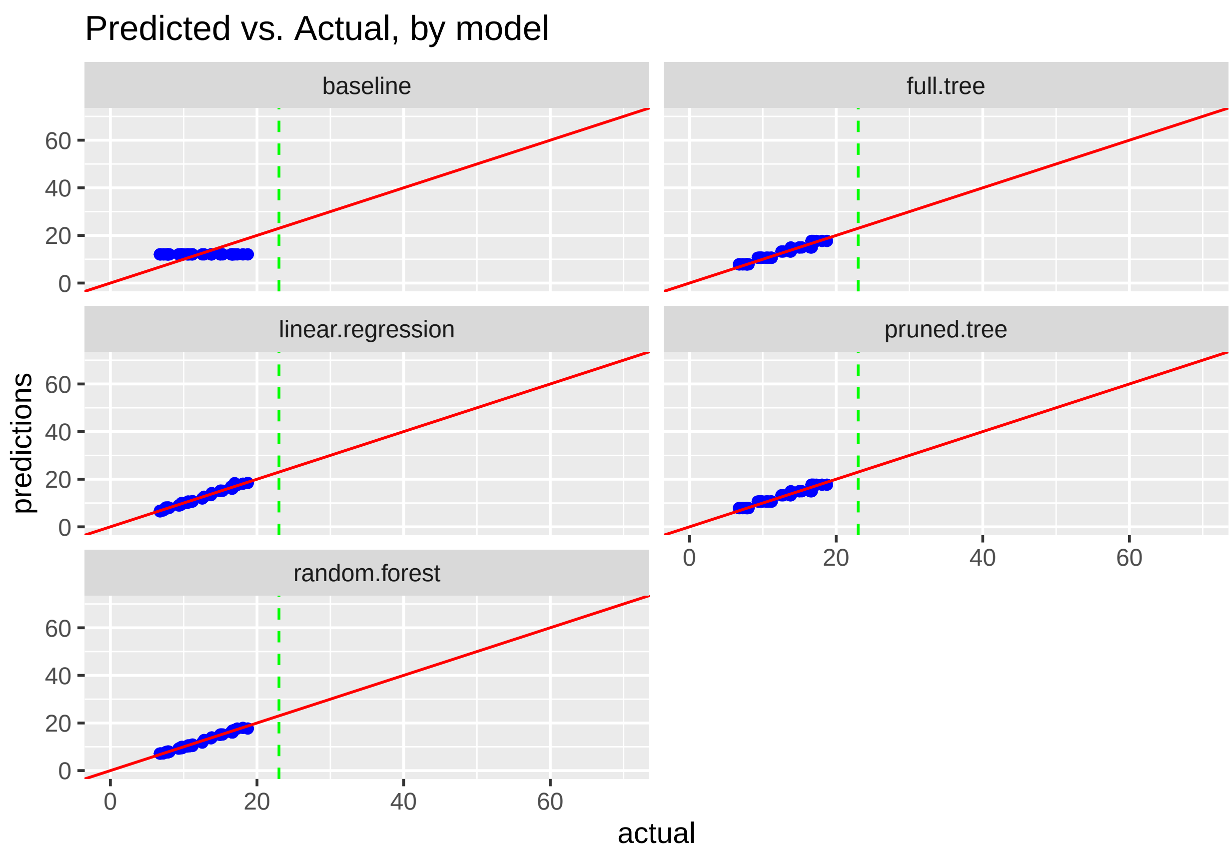
```
##              Method RMSE  MAE
## 1          Baseline 3.62 3.25
## 2 Linear Regression 0.38 0.28
## 3         Full tree 0.80 0.65
## 4       Pruned tree 0.80 0.65
## 5     Random forest 0.34 0.25
## 6    Support Vector 0.48 0.35
```

```r
#Print prediction
# Create a data frame with the predictions for each method
all.predictions <- data.frame(actual = test$Employed_tech,
                              baseline = best.guess,
                              linear.regression = test.pred.lin,
                              full.tree = test.pred.rtree,
                              pruned.tree = test.pred.rtree.p,
                              random.forest = test.pred.forest,
                              support.vector = svm.pred)

# First 10 observations of predictions
head(all.predictions, 10)
```

```
##      actual baseline linear.regression full.tree pruned.tree random.forest
## 1  6.766399 12.05043          6.490045  7.861442    7.861442      7.156777
## 2  6.747070 12.05043          6.708522  7.861442    7.861442      7.170190
## 3  6.906259 12.05043          6.807459  7.861442    7.861442      7.180887
## 5  7.203985 12.05043          6.870216  7.861442    7.861442      7.238689
## 11 7.310302 12.05043          7.280140  7.861442    7.861442      7.326189
## 18 7.803861 12.05043          7.765574  7.861442    7.861442      7.631973
```

```
## 19 7.779840 12.05043          7.976532  7.861442    7.861442      7.898606
## 24 7.884218 12.05043          8.142751  7.861442    7.861442      8.055856
## 28 8.051184 12.05043          7.992188  7.861442    7.861442      7.870927
## 29 7.925592 12.05043          8.006559  7.861442    7.861442      7.945856
##    support.vector
## 1        8.197137
## 2        7.685155
## 3        7.545061
## 5        7.849466
## 11       7.285611
## 18       7.450947
## 19       8.036657
## 24       8.005247
## 28       7.949979
## 29       7.785498
```

```
# Convert the data frame in longer format
all.predictions <- gather(all.predictions,key = model,value = predictions,2:6)

# Plot predicted vs. actual for each model
ggplot(data = all.predictions,aes(x = actual, y = predictions)) +
  geom_point(colour = "blue") +
  geom_abline(intercept = 0, slope = 1, colour = "red") +
  geom_vline(xintercept = 23, colour = "green", linetype = "dashed") +
  facet_wrap(~ model,ncol = 2) +
  coord_cartesian(xlim = c(0,70),ylim = c(0,70)) +
  ggtitle("Predicted vs. Actual, by model")
```

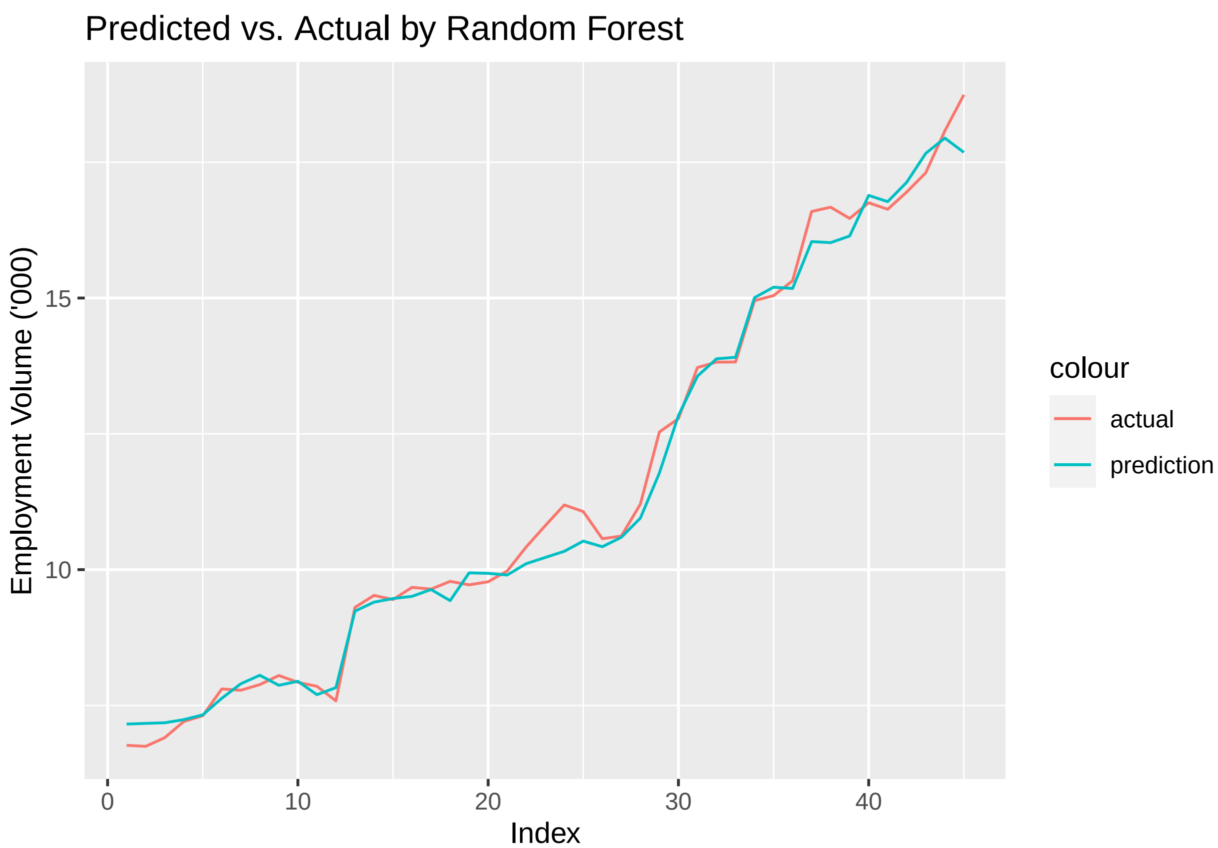



```
# Plot Predicted vs. actual by Random Forest model
random.forest.prediction <- data.frame(actual = test$Employed_tech,
                                       prediction = test.pred.forest)
random.forest.prediction
```

```
##       actual prediction
## 1   6.766399   7.156777
## 2   6.747070   7.170190
## 3   6.906259   7.180887
## 5   7.203985   7.238689
## 11  7.310302   7.326189
## 18  7.803861   7.631973
## 19  7.779840   7.898606
## 24  7.884218   8.055856
## 28  8.051184   7.870927
## 29  7.925592   7.945856
## 33  7.852994   7.698590
## 36  7.584705   7.828327
## 45  9.305477   9.236086
## 48  9.526157   9.401297
## 49  9.448940   9.467949
## 54  9.673935   9.508307
## 55  9.641958   9.634266
## 56  9.783893   9.429214
## 57  9.720148   9.940871
## 58  9.776988   9.931530
```

```
## 59   9.974754   9.901261
## 61  10.417863  10.110149
## 65  10.809358  10.224089
## 66  11.191156  10.337577
## 67  11.069500  10.525996
## 68  10.569790  10.421793
## 70  10.617940  10.595697
## 75  11.206245  10.948519
## 85  12.534807  11.779522
## 87  12.780181  12.836996
## 95  13.721785  13.561853
## 98  13.820091  13.881235
## 101 13.822742  13.908749
## 113 14.952276  15.007025
## 114 15.043909  15.199349
## 116 15.318442  15.176080
## 123 16.593138  16.038998
## 125 16.672569  16.019503
## 126 16.464713  16.142294
## 131 16.753101  16.888824
## 132 16.634246  16.774253
## 134 16.952214  17.132296
## 136 17.305651  17.663732
## 138 18.076146  17.944024
## 140 18.741404  17.681609
```

```r
random.forest.prediction$index <- index(random.forest.prediction)
random.forest.prediction %>%
  ggplot(aes(x=index)) +
  geom_line(aes(y = actual, colour="actual")) +
  geom_line(aes(y = prediction, colour= "prediction")) +
  ylab("Employment Volume ('000)") +
  xlab("Index") +
  ggtitle(paste("Predicted vs. Actual by Random Forest"))
```




```
##################### Part G #####################
#set length of forecast (quaters)
duration <- 4

# Create a data frame used for forecasting
df <- df_job_volume_development_tech_others
df$Date <- as.yearmon(df$Date, "%Y-%m")

# Rearrange data frame depending on the prediction length
data_modifi <- df[,-which(names(df) == "Employed_tech")]
data_modifi$Date <- as.yearmon(data_modifi$Date) + (3/12)*duration
data_volume <- data.frame(Date = df$Date,
                          volume_employed = df[,which(names(df) == "Employed_tech")])
data <- merge(data_modifi, data_volume, all = TRUE)

# Drop dates out of range
data <- tail(data, -duration)

# leave only train data range
train <- data[as.yearmon(data$Date)<=(as.yearmon(tail(data$Date,1)) - (3/12)*duration),]

# subset elements in the index
test <- data[as.yearmon(data$Date)>(as.yearmon(tail(data$Date,1)) - (3/12)*duration),]

nrow(train)
```

```
## [1] 145
```

```r
nrow(test)
```

```
## [1] 4
```

```r
# Generate predictions using Random Forest
rf <- randomForest(volume_employed ~., data = train, importance = TRUE, ntree=1000)

# Compute optimal number of trees
ntree <- which.min(rf$mse)

# Regenerate predictions using optimal number of trees
rf <- randomForest(volume_employed ~., data = train, importance = TRUE, ntree=ntree)
test.pred <- predict(rf,test[,-which(names(test) == "volume_employed")])
test.pred
```

```
##      150      151      152      153
## 18.50815 18.63775 18.07068 18.63461
```

```r
# Covert data frame into time series format
train.xts <- xts(train, order.by = as.Date(train$Date))
test.xts <- xts(test, order.by = as.Date(test$Date))

# Create a data frame indicating date range
pred2 <- rbind(train.xts[, "volume_employed"], test.xts[, "volume_employed"])

# Add corresponding date variables to predicted data
volume_employed_pred_xts <- xts(test.pred, order.by = as.Date(test$Date))
names(volume_employed_pred_xts) <- "volume_employed_pred"
volume_employed_pred_xts
```

```
##            volume_employed_pred
## 2022-02-01             18.50815
## 2022-05-01             18.63775
## 2022-08-01             18.07068
## 2022-11-01             18.63461
```

```r
# Create an data frame used for computing slope of change
df_slope <- data.frame(Date = index(volume_employed_pred_xts),
                       pred = volume_employed_pred_xts$volume_employed_pred)

# Compute slope of change
lm(df_slope$volume_employed_pred ~ index(df_slope))
```

```
##
## Call:
## lm(formula = df_slope$volume_employed_pred ~ index(df_slope))
##
## Coefficients:
##     (Intercept)  index(df_slope)
##        18.50971         -0.01877
```

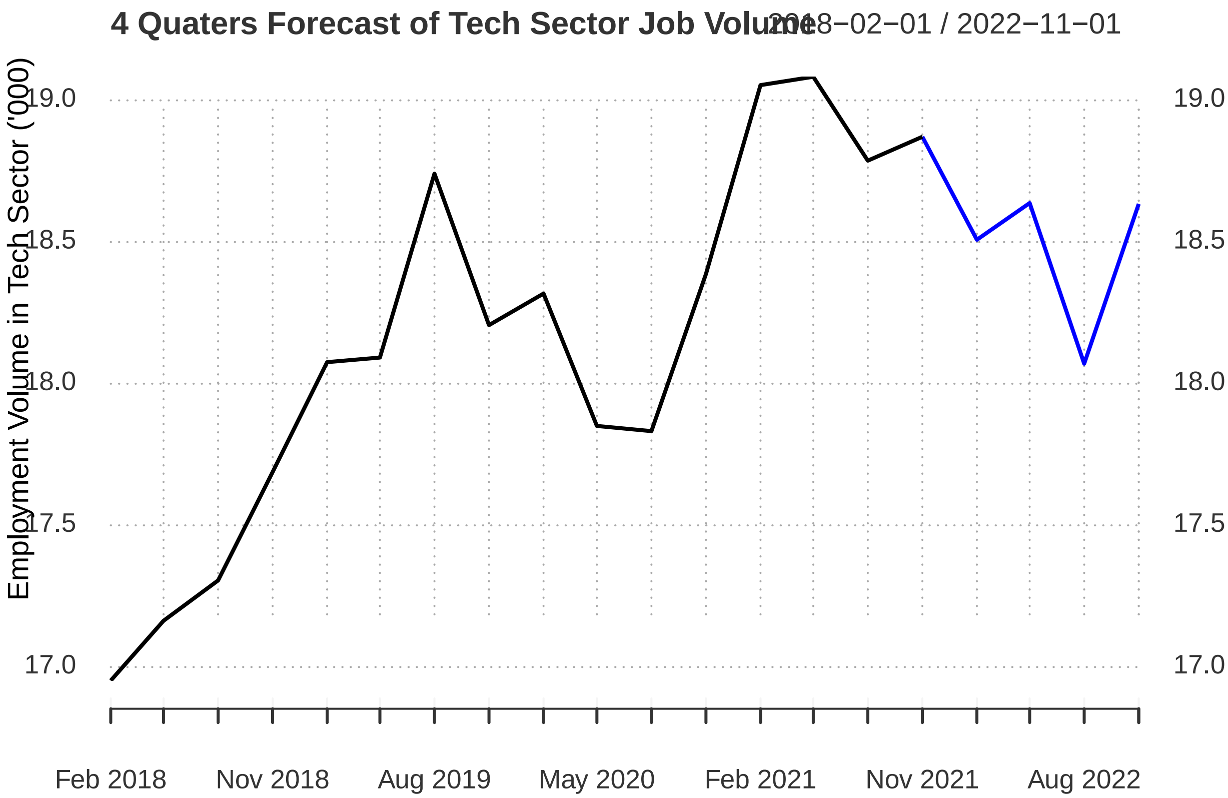
```
# Compute average percentage change
APC <- ((df_slope[nrow(df_slope),2] - df_slope[1,2]) / df_slope[1,2]) * 100
APC
```

```
## [1] 0.6832955
```

```
# Comupte average percentage change per each quarter
APC_quarters <- APC/duration
APC_quarters
```

```
## [1] 0.1708239
```

```
# Merge predicted data with the original data
df_volume_pred_prev <- merge(pred2, volume_employed_pred_xts)
df_volume_pred_prev$volume_employed_pred[nrow(df_volume_pred_prev$volume_employed_pred)-duration] <-
  df_volume_pred_prev$volume_employed[nrow(df_volume_pred_prev$volume_employed_pred)-duration]

# Plot predicted values with the original data
df <- df_volume_pred_prev
graph_pred_prev <- plot(df[as.Date(as.yearmon(index(df)))>(as.Date(tail(as.yearmon(index(df)),1)-(3/12)
                           c("volume_employed", "volume_employed_pred")],
                     major.ticks = "months",
                     grid.ticks.on = "months",
                     grid.ticks.lty = 3,
                     main = paste(duration,  "Quaters Forecast of Tech Sector Job Volume", sep=" "),
                     col = c("black", "blue"),
                     ylab = "Employment Volume in Tech Sector ('000)")
graph_pred_prev
```


4 Quaters Forecast of Tech Sector Job Volume
2018−02−01 / 2022−11−01
Employment Volume in Tech Sector ('000)
19.0
18.5
18.0
17.5
17.0
Feb 2018
Nov 2018
Aug 2019
May 2020
Feb 2021
Nov 2021
Aug 2022