Guide on reproducing BLS graphic

Xianbin Xu

8/24/2022

Introduction

This R Markdown file replicates various graphics from Bureau of Labor Statistics data. It contains specific instructions and replicable codes.

Running Environment

This RMD (R Markdown) file had been knitted into a PDF file for easier reading. PDF file and R Markdown file(end with .rmd) are all included in the directory under the same name. To reproduce the code, open the R Markdown in RStudio, then run the whole markdown file.

To get a LaTeX file of the document, add "keep_tex: true" below the pdf_output: line, in the header section of R Markdown. Then, knit the whole markdown file.

Make sure the file is located in /Graphic_Reproduction_Folder, with directory BLS_Data in the same directory containing /Graphic_Reproduction_Folder.

Make sure the following files are in place at BLS_Data folder:

Employment_By_State.csv. It can be recreated by running BLS_Download_State_Level.R

The following packages are used:

```r
library("dplyr") # For data processing
library("tidyr")
```

use install.packages() to install if you haven't installed already. The formatR package was used to creat tidy code chunks for easier reading. You will NOT need it if you just do the codes in R Studio.

Creating a function for jobs-per-capita scatterplot

Read the file.

```r
Employment_Data = read.csv("./../BLS_Data/Employment_By_State.csv")
```

Here, the ../ get us to the directory containing both Graphic_Reproduction_Folder and BLS_Data.

The following function shall be used to calculate job-to-population ratio, averaged over years, for each state. Washington DC was included, with abbreviation DC, but Puerto Rico was not, due to it not being a state but always being an outlier. The formula is:

$$\text{job ratio growth} = \frac{\sum_{\text{begin year of the period}}^{\text{end year of the period}} \text{employment by year}}{\sum_{\text{begin year of the period}}^{\text{End Year of the period}} \text{population by year}}$$

By calculating ave_job_ratio for two periods, we can find job ratio growth:

job_ratio_growth = Ave_job_ratio in period 2 / Ave_job_ratio in period 1

This is considered the growth of job per capita, where job ratio growth < 1 means a shrink in job per capita and > 1 means an increase.

By designating two occupations, we can create a scatterplot on the job_ratio_growth for two occupations, where X and Y axis each gives refers to an occupation.

I used function instead of codes because it will give an easier time trying to recreate same plot under different variables. The variables used are as the following:

> First_Occupation and Second_Occupation refers to names of occupations on X and Y axis respectively.
>
> Input_Data refers to data frame to be used, in our case just use Employment_Data
>
> Begin_Year_1 and End_Year_1 refers to beginning and ending year for period 1. Beginning and Ending years are included when averaging
>
> Plot = 1 if you want to get the plot. Default is 1.
>
> give_provessed_data = 1 if you want to get processed data that was directly used in plot function. Default is 0 (Don't return)

Specific technical details are included in code chunk.

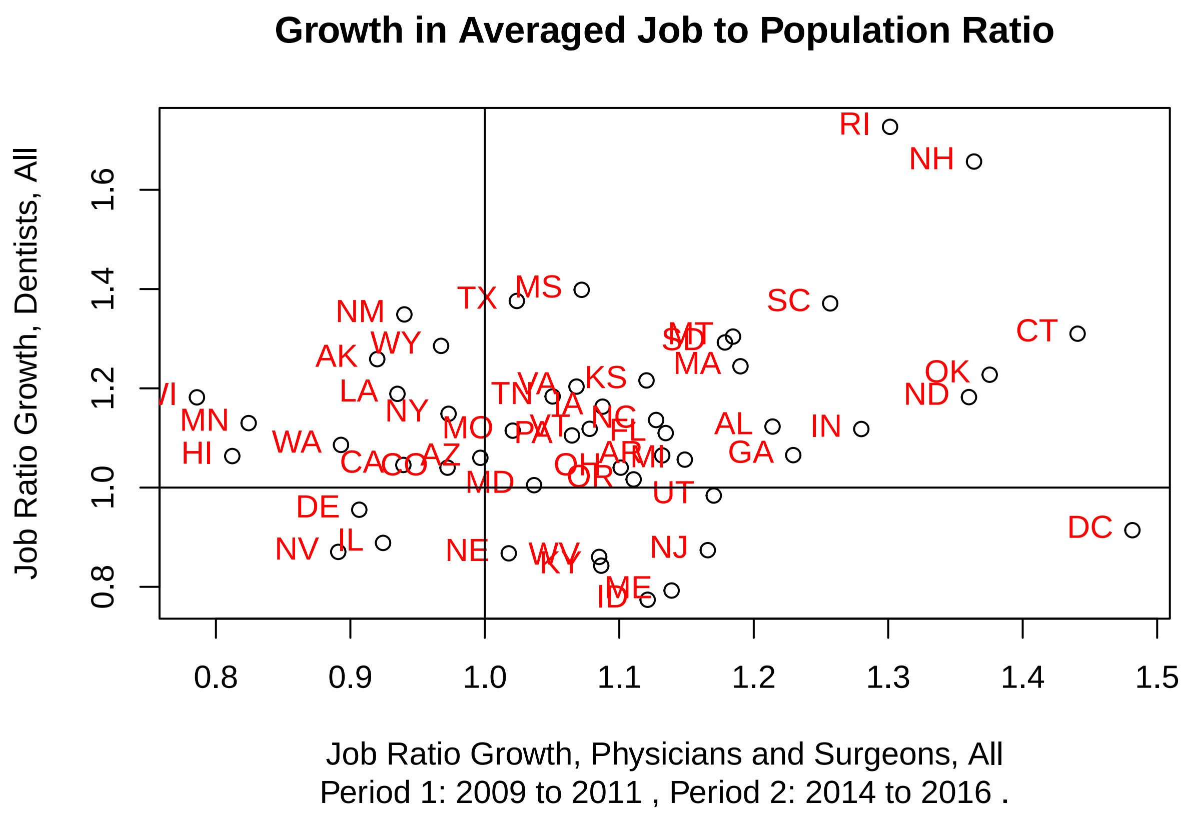
```
Ratio_Extractor = function(First_Occupation, Second_Occupation, Input_Data,
                           Begin_Year_1 = 2009, End_Year_1 = 2011,
                           Begin_Year_2 = 2014, End_Year_2 = 2016,
                           plot = 1, give_processed_data = 0){


  temp = Input_Data
  temp = temp[temp$job_type == First_Occupation |
               temp$job_type == Second_Occupation, ]

  ### This two steps includes only two occupations which we are plotting

  temp = temp[temp$st != "PR", ]
  ### Drop Puerto Rico.


  # Categorize by job type
  temp = temp %>% mutate(year_period = case_when(
    year >= Begin_Year_1 & year <= End_Year_1 ~ 1, #First Period
    year >= Begin_Year_2 & year <= End_Year_2 ~ 2, #Second Period
    TRUE ~ -1

    ### The year is a variable in input_data (here included as "temp").
    ###It refers to the year of the data entry.
    ### If it is between beginning_year_1 and end_year_1,
    ### include it as 1, meaning it being in period 1.
    ### Similarly we get result for period 2.

    ### If you need to have overlap between two periods, this code unfortunately
    ### cannot do it.Instead you will need two variables for them.
  )) %>% filter(year_period != -1) -> temp

  # Having 0 or NA in any of the tot_employment means the raw data was NOT available.
  # Thus, we keep only those with good data.
  temp = temp[!(is.na(temp$Est_Population) | temp$tot_employment == 0), ]

  # Keep only needed variables. st variable stores abbreviation of states' name
  temp = temp[, c("st", "job_type", "occ_code", "tot_employment",
                  "year", "Est_Population", "year_period")]

  # Group by st(state abbreviations), job, and year_period
  temp = temp %>% group_by(st, job_type, year_period) %>%
    summarize(ave_ratio = sum(tot_employment) / sum(Est_Population)) %>%
    # for each state-job-year period, use ave_job_ratio formula
    # to find the averaged job per capita.
    # There shouldn't be an NA problem since they were all dropped above
    unique()


  temp = temp %>% pivot_wider(names_from = year_period, values_from = ave_ratio,
                              names_prefix = "year_period_") %>%

    ## The Pivot_Wider part shall expand the dataset.
    ## The year_period value would become two new variables,
    ## year_period_1 and year_period_2,
    ## and have job_ratio of the respective period beneath it.
    ## As such, we shall have state-job ratio for two periods.

    select(st, job_type, year_period_1, year_period_2) %>%
    mutate(job_ratio_growth = year_period_2 / year_period_1) %>%

    ### The mutate step finds the growth in the job ratio.

    select(st, job_type, job_ratio_growth) %>%

    # Keep only the three columns we need

    pivot_wider(names_from = job_type, values_from = job_ratio_growth,
                names_prefix = "Job Ratio Growth, ")

  ### Finally, we shall pivot again so that job_growth_ratios for both
```

```r
  ### jobs are two variables.
  ### The plotting function does not discern different values from a variable
  ### but requires two variables for X and Y axis respectively.

  temp = as.data.frame(temp)
  ### The temp here is of "group" class. It should be converted back to
  ## data frame to be used for plotting


  temp = temp[, c("st", paste("Job Ratio Growth, ", First_Occupation, sep = ""),
                  paste("Job Ratio Growth, ", Second_Occupation, sep = ""))]

  # We shall rename the variables under the occupations' names.
  ### In this way, we can tell the X-axis from the Y-axis.
  ### Note that the naming does NOT use underline but spaces.
  ### This is useful for plotting and plotting only.
  ### So we don't bother changing the X and Y axis again.

  if(plot == 1){
    ### This chunk only runs when plot is 1, which does the plotting
    plot(x = temp[, 2], y = temp[, 3],
         xlab = colnames(temp)[2],
         ylab = colnames(temp)[3])
    title(main = "Growth in Averaged Job to Population Ratio",
          #### sub is used to get sb-titles, by the bottom of the graph
          sub = paste("Period 1:", Begin_Year_1, "to", End_Year_1,
                      ", Period 2:", Begin_Year_2, "to", End_Year_2, "."))
    text(temp[, 2], temp[, 3], temp[, 1], pos = 2, col = "red")

    ### Here is the reason for this line. the text function works as:
    ### text(x-coordinate, y-coordinate, text to be used for each data point,
    ### position, color). Since we already used 2nd and 3rd variable
    ### as x and y axis respectively, and first variable at temp[, 1]
    ### is state abbreviation, we can thus plot in this way.

    abline(h = 1)
    abline(v = 1)
    ### We finally add horizontal and vertical line at y = 1 and x = 1.
    ### In this way we can tell increase/decrease
    ### in ratio of the jobs
  }
  if(give_processed_data == 1){
    #### If you need data output, here it is.
    ### The columns were renamed with underlines so
    ### it shall work better as a data frame.
    colnames(temp) = c("st",
                       paste("job_ratio_growth_", First_Occupation, sep = ""),
                       paste("job_ratio_growth_", Second_Occupation, sep = ""))
    return(temp)
  }
}
```

In this way, we should have a function that plots the job_ratio_growth of one occupation against another. Here's an example use:

```r
Ratio_Extractor(First_Occupation = "Physicians and Surgeons, All",
                Second_Occupation = "Dentists, All",
                Input_Data = Employment_Data)
```

```
## ‘summarise()‘ has grouped output by ’st’, ’job_type’. You can override using
## the ‘.groups‘ argument.
```



For example, the function above plot job growth rate of dentists over job growth rate of physicians and surgeons. The first and second occupations can take from the following:

```r
unique(factor(Employment_Data$job_type))
```

```
## [1] Dental Hygienists
## [2] Dentists, All
## [3] Nurses
## [4] Physicians and Surgeons, All
## [5] Physicians and Surgeons, Family Practitioner
## [6] Dentists, General
## 6 Levels: Dental Hygienists Dentists, All Dentists, General ... Physicians and Surgeons, Family Prac
```

Here are some notes:

> Use the exact name, as characters, for function input.

> Dentists, All are calculated from summing up ALL dentist subcategories (dentists, general; dental surgeons; etc.)
>
> Dentists, General refers to denetists who does not specialize in surgeon or other field. They are NOT found in original BLS data beofre 2003, instead only Dentists_All were found.
>
> Similarly, Physicians and Surgeons, All were sum of all doctors, phycisians, surgeons, etc.
>
> See https://www.bls.gov/oes/current/oes_stru.htm, 29-1060, 29-1020, 29-2021, 29-1141 for more specific information.

The begin_year_1, end_year_1, begin_year_2, and end_year_2 parameter is by default 2009, 2011, 2014, and 2016 respectively, thus we compare period between 2009 and 2011 with period between 2014 and 2016, beginning and ending included. You can change it for other years.

plot is by default 1 so you get output plot. Set it to anything else for no plot. give_processed_data was by default 0, set it to 1 to get processed dataframe used for plotting.

Plot with BLS codes for occupations

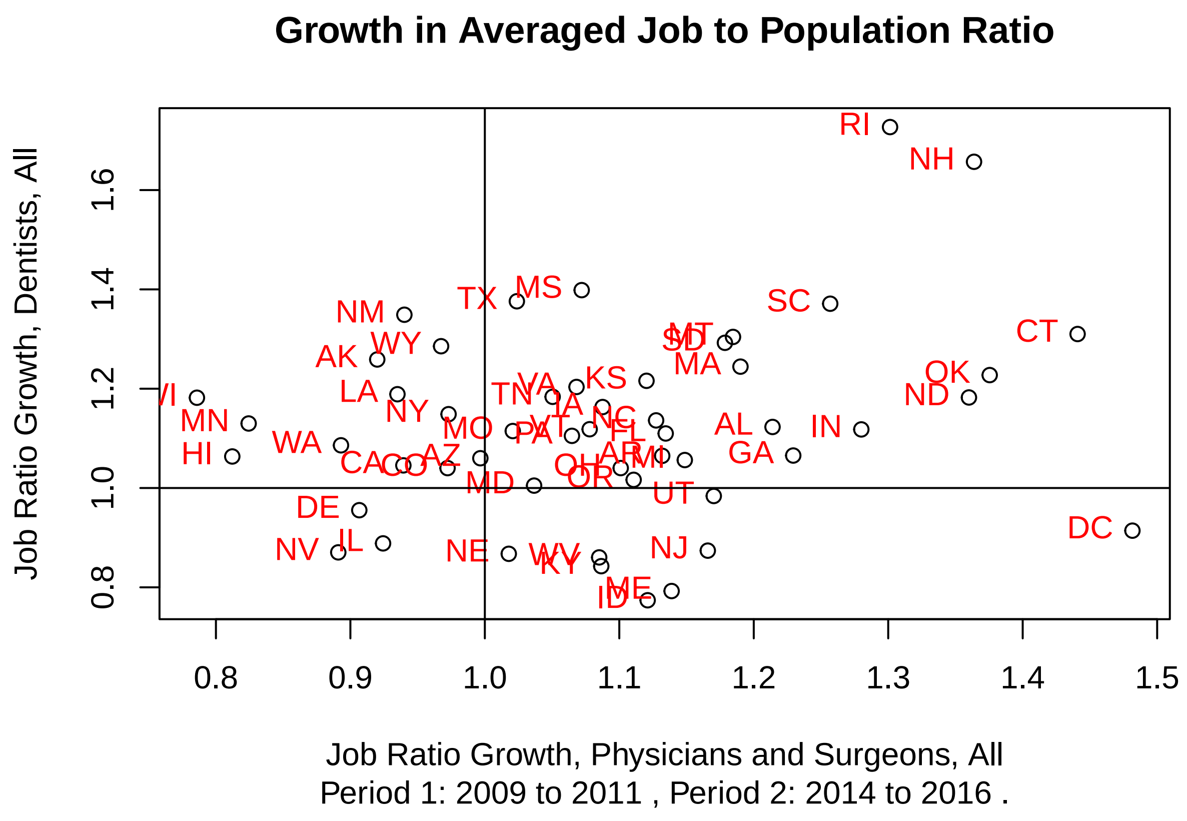
Alternatively, the following function takes number instead of occupation names. It serves generally the same use:

```
Ratio_Extractor_by_Code = function(First_Occupation_Code, Second_Occupation_Code, Input_Data,
                                   Begin_Year_1 = 2009, End_Year_1 = 2011,
                                   Begin_Year_2 = 2014, End_Year_2 = 2016,
                                   plot = 1, give_processed_data = 0){
  # This function basically does the same thing as above (Ratio_Extractor),
  # But it uses occupation code instead of occupation titles
  return(
    Ratio_Extractor(
      First_Occupation = Input_Data[Input_Data$occ_code == First_Occupation_Code, ][1, ]$job_type,

      #### This line shall be read as the following: For the lines in input_data,
      ### With same occ_code as the First Occupation Code input of the function,
      ### Take their first line, and the entry in "job_type" variable
      ### This shall give occupation name by character to be used in
      ### Ratio_Extractor function as written above.

      Second_Occupation = Input_Data[Input_Data$occ_code == Second_Occupation_Code, ][1, ]$job_type,

      Input_Data = Input_Data,
      Begin_Year_1 = Begin_Year_1, End_Year_1 = End_Year_1,
      Begin_Year_2 = Begin_Year_2, End_Year_2 = End_Year_2,
      plot = plot, give_processed_data = give_processed_data

      ### All other parameters were passed without changes.
    )
  )
}
```

To replicate the same plot as above, the following input were used:

```
Ratio_Extractor_by_Code(First_Occupation_Code = 1060,
                        Second_Occupation_Code = 1020,
                        Input_Data = Employment_Data)
```

```
## ‘summarise()‘ has grouped output by ’st’, ’job_type’. You can override using
## the ‘.groups‘ argument.
```



Since 29-1060 were BLS code for physicians and surgeons, and 1020 for all dentists. Here is a table for job types along their codes:

```
unique(Employment_Data[, c("job_type", "occ_code")])
```

```
##                                        job_type occ_code
## 1                              Dental Hygienists     2021
## 2                                  Dentists, All     1020
## 3                                         Nurses     1141
## 4                 Physicians and Surgeons, All       1060
## 187  Physicians and Surgeons, Family Practitioner     1062
## 1448                           Dentists, General     1021
```

The second row, occ_code, are codes used as input for Ratio_Extractor_by_Code. To search for corresponding occupation and their profile, description etc. on BLS website, add 29- before the code.

Plot job ratio against wage

The following function make a scatterplot for datapoints representing 50 U.S. States and District Columbia. The X-axis is job-per-1,000-population, or ave_job_ratio, or averaged jobs per 1,000 people over a period of years, for one occupation. The Y-axis is average annual wage for one occupation, calculated as mean annual income of an occpuation averaged over years. Not inflation adjusted.

These two occupations NEED NOT to be the same. Furthermore, their beginning and ending years for averaging can also be self-defined and need also not be the same.

Here is the list of input variable explained.

> Occupation_for_Ratio: Name of the occupation for job/population ratio, on X-axis
>
> Occupation_for_Wage: Name of the occupation for average annual income, on Y-axis. By default same as Occupation_for_Ratio but can be changed
>
> Input_Data: just use employment_data
>
> Begin_Year_Ratio, End_Year_Ratio: Defines year period to calculate averaged job/population ratio. Default 2003 and 2018 respectively
>
> Begin_Year_Wage, End_Year_Wage: Defines year period to calculate averaged Annual Income. Default 2003 and 2018 respectively
>
> plot: 1 if you need scatterplot output, default 1
>
> give_processed_data: 1 if you need processed data frame used for plotting to be returned. Default 0.

Specific technical details are in comments in code below:

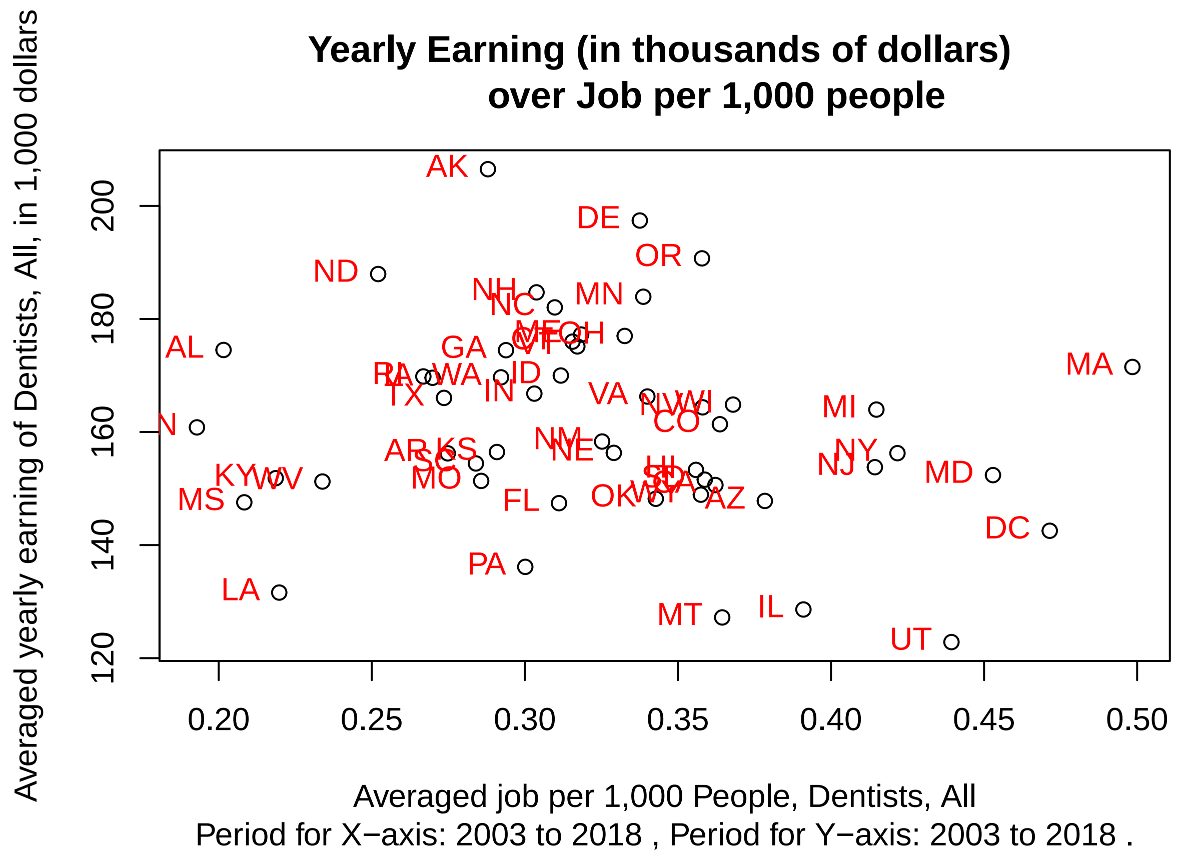
```r
Ratio_Wage_Extractor = function(Occupation_for_Ratio,
                                Occupation_for_Wage = Occupation_for_Ratio,
                                Input_Data,
                           Begin_Year_Ratio = 2003, End_Year_Ratio = 2018,
                           Begin_Year_Wage = 2003, End_Year_Wage = 2018,
                           plot = 1, give_processed_data = 0){

  temp = Input_Data
  temp = temp[temp$st != "PR", ]
  ## Input data and drop Puerto Rico

  data_for_ratio = temp %>% filter(job_type == Occupation_for_Ratio) %>%

    #### This filters the entries with correct occupation
    #### to calculate job/population ratio, A.K.A. ave_job_ratio

    filter(year >= Begin_Year_Ratio & year <= End_Year_Ratio) %>%

    # Filter year range

    filter(!is.na(Est_Population)) %>% filter(tot_employment > 0)
    ## Drop useless entries. Same idea as previous function

  #Group them and sum 'em up
  data_for_ratio = data_for_ratio %>% group_by(st) %>%
    summarize(ave_job_ratio = sum(tot_employment) / sum(Est_Population)) %>%
    mutate(ave_job_ratio = ave_job_ratio * 1000) %>%
    unique()

  ### The above function groups the data by state and find job ratio
  ### Through sum(tot_employment) / sum(Est_Population).
  ### This might involve replicate calculations for all years. However,
  ### This is a small dataset so the calculation time is ignorable.

  #Compute mean wage.
  data_for_wage = temp %>% filter(job_type == Occupation_for_Wage) %>%

    ## Similarly, filter only those occupation used
    ### to calculate wages.

    filter(year >= Begin_Year_Wage & year <= End_Year_Wage) %>%

    ## Year range

    filter(!is.na(Est_Population)) %>% filter(tot_employment > 0)

  data_for_wage = data_for_wage %>% group_by(st) %>%
    summarize(ave_annual_wage = mean(annual_mean_income, na.rm = TRUE)) %>%
    mutate(ave_annual_wage = ave_annual_wage / 1000) %>%
    unique()

  ## Again, by each state make average on mean annual income

  temp = merge(data_for_ratio, data_for_wage, by = "st")

  if(plot == 1){
    plot(x = temp$ave_job_ratio, y = temp$ave_annual_wage,
         xlab = paste("Averaged job per 1,000 People, ",
                      Occupation_for_Ratio, sep = ""),
         ylab = paste("Averaged yearly earning of ",
                      Occupation_for_Wage, ", in 1,000 dollars", sep = ""))
    # Fix Scale)
    title(main = "Yearly Earning (in thousands of dollars)
          over Job per 1,000 people",
          sub = paste("Period for X-axis:", Begin_Year_Ratio,
                      "to", End_Year_Ratio,
                      ", Period for Y-axis:", Begin_Year_Wage,
                      "to", End_Year_Wage, "."))
    text(temp[, 2], temp[, 3], temp[, 1], pos = 2, col = "red")
  }
  if(give_processed_data == 1){
    return(temp)
  }
}
```

Similarly, here is a function that allows plotting with code, using same occupation title to code format as in ratio_extractor function.

```
Ratio_Wage_Extractor_by_Code = function(Occupation_for_Ratio_Code,
                                        Occupation_for_Wage_Code = Occupation_for_Ratio_Code,
                                        Input_Data,
                                        Begin_Year_Ratio = 2003, End_Year_Ratio = 2018,
                                        Begin_Year_Wage = 2003, End_Year_Wage = 2018,
                                        plot = 1, give_processed_data = 0){
  return(
    Ratio_Wage_Extractor(
      Occupation_for_Ratio = Input_Data[Input_Data$occ_code == Occupation_for_Ratio_Code, ][1, ]$job_typ
      Occupation_for_Wage = Input_Data[Input_Data$occ_code == Occupation_for_Wage_Code, ][1, ]$job_type
      Input_Data = Input_Data,
      Begin_Year_Ratio = Begin_Year_Ratio, End_Year_Ratio = End_Year_Ratio,
      Begin_Year_Wage = Begin_Year_Wage, End_Year_Wage = End_Year_Wage,
      plot = plot, give_processed_data = give_processed_data
    )
  )
}
```

Here is an example with dentists_all

```
Ratio_Wage_Extractor_by_Code(Occupation_for_Ratio_Code = 1020, Input_Data = Employment_Data)
```



You can create more plots with these functions should you like.