Data Wrangling Case Studies

Jae Kum (Jackie) Kim

September 19, 2018

Introduction

This document has case studies I used to self-study, and I've made some few twitches from Linkedin Learning website. I give all credits to **Mr. Mike Chapple** for providing a great tutorial for data wrangling case studies.

First Case: Coal Consumption

Goal

The goal of the first case study is to calculate and analyze coal consumptions with respect to regions overtime.

Import Data and Summary

Import **tidyverse** package and import coal consumption (skipping first two rows) data from youcanlearnit (http://594442.youcanlearnit.net/coal.csv):

```
# Import tidyverse
library(tidyverse)

# Import coal consumption data file.
coal <- read_csv("http://594442.youcanlearnit.net/coal.csv", skip = 2)
coal <- as.tibble(coal)
```

For a more visible theme, I have imported **ggthemes**:

```
library(ggthemes)
```

Notice that the first column is not defined. Rename the first column into **region**:

```
# Rename the first column name
colnames(coal)[1] <- 'region'

# Check the summary of the data
glimpse(coal)
```

```
## Observations: 232
## Variables: 31
## $ region <chr> "North America", "Bermuda", "Canada", "Greenland", "Mex...
## $ `1980` <chr> "16.45179", "0", "0.96156", "0.00005", "0.10239", "0", ...
## $ `1981` <chr> "16.98772", "0", "0.99047", "0.00005", "0.10562", "0", ...
## $ `1982` <chr> "16.47546", "0", "1.05584", "0.00003", "0.11967", "0", ...
## $ `1983` <chr> "17.12407", "0", "1.11653", "0.00003", "0.12869", "0", ...
## $ `1984` <chr> "18.4267", "0", "1.23682", "0.00003", "0.13071", "0", "...
## $ `1985` <chr> "18.81819", "0", "1.20679", "0", "0.14646", "0", "17.46...
## $ `1986` <chr> "18.52559", "0", "1.12583", "0", "0.15609", "0", "17.24...
## $ `1987` <chr> "19.43781", "0", "1.25072", "0", "0.17001", "0", "18.01...
## $ `1988` <chr> "20.40363", "0", "1.35809", "0", "0.15967", "0", "18.88...
## $ `1989` <chr> "20.62571", "0", "1.35196", "0", "0.17359", "0", "19.10...
## $ `1990` <chr> "20.5602", "0", "1.21338", "0", "0.1694", "0", "19.1774...
## $ `1991` <chr> "20.4251", "0", "1.26457", "0", "0.15916", "0", "19.001...
## $ `1992` <chr> "20.64672", "0", "1.32379", "0", "0.16584", "0", "19.15...
## $ `1993` <chr> "21.28219", "0", "1.22875", "0", "0.19118", "0", "19.86...
## $ `1994` <chr> "21.39631", "0", "1.24492", "0", "0.1836", "0", "19.967...
## $ `1995` <chr> "21.64225", "0", "1.28479", "0", "0.20768", "0", "20.14...
## $ `1996` <chr> "22.57572", "0", "1.30032", "0", "0.25067", "0", "21.02...
## $ `1997` <chr> "23.20491", "0", "1.44933", "0", "0.26373", "0", "21.49...
## $ `1998` <chr> "23.5002", "0", "1.50985", "0", "0.26753", "0", "21.722...
## $ `1999` <chr> "23.4747", "0", "1.505", "0", "0.28947", "0", "21.68023...
## $ `2000` <chr> "24.55583", "0", "1.61651", "0", "0.29444", "0", "22.64...
## $ `2001` <chr> "23.62705", "0", "1.35444", "0", "0.32908", "0", "21.94...
## $ `2002` <chr> "23.69876", "0", "1.36876", "0", "0.36525", "0", "21.96...
## $ `2003` <chr> "24.17788", "0", "1.38766", "0", "0.41878", "0", "22.37...
## $ `2004` <chr> "24.36024", "0", "1.43684", "0", "0.31944", "0", "22.60...
## $ `2005` <chr> "24.6876", "0", "1.44948", "0", "0.39739", "0", "22.840...
## $ `2006` <chr> "24.32174", "0", "1.42135", "0", "0.39244", "0", "22.50...
## $ `2007` <chr> "24.54746", "0", "1.38369", "0", "0.38911", "0", "22.77...
## $ `2008` <chr> "24.11993", "0", "1.37388", "0", "0.32008", "0", "22.42...
## $ `2009` <chr> "21.14803", "0", "1.14314", "0", "0.3365", "0", "19.668...
```

Convert Class Type

It seems that value of year and coal consumption are listed as characters. Use **gather** function to simplify data and convert character to integer/numeric:

```
# Use gather()
coal_long <- gather(coal, 'year', 'coal_consumption', -region)
glimpse(coal_long)
```

```
## Observations: 6,960
## Variables: 3
## $ region           <chr> "North America", "Bermuda", "Canada", "Greenl...
## $ year             <chr> "1980", "1980", "1980", "1980", "1980", "1980...
## $ coal_consumption <chr> "16.45179", "0", "0.96156", "0.00005", "0.102...
```

```
# Convert year to date and coal value to numeric
coal_long$year <- as.integer(coal_long$year)
coal_long$coal_consumption <- as.numeric(coal_long$coal_consumption)
summary(coal_long)
```

```
##     region               year      coal_consumption
##  Length:6960        Min.   :1980   Min.   : -0.0002
##  Class :character   1st Qu.:1987   1st Qu.:  0.0000
##  Mode  :character   Median :1994   Median :  0.0002
##                     Mean   :1994   Mean   :  1.3256
##                     3rd Qu.:2002   3rd Qu.:  0.0773
##                     Max.   :2009   Max.   :138.8298
##                                    NA's   :517
```

Segmenting

We discover that region column is mixed of countries and regions. Find all non-country data and categorize them separately:

```
# Locate regions and create columns of regions
head(unique(coal_long$region), n = 10)
```

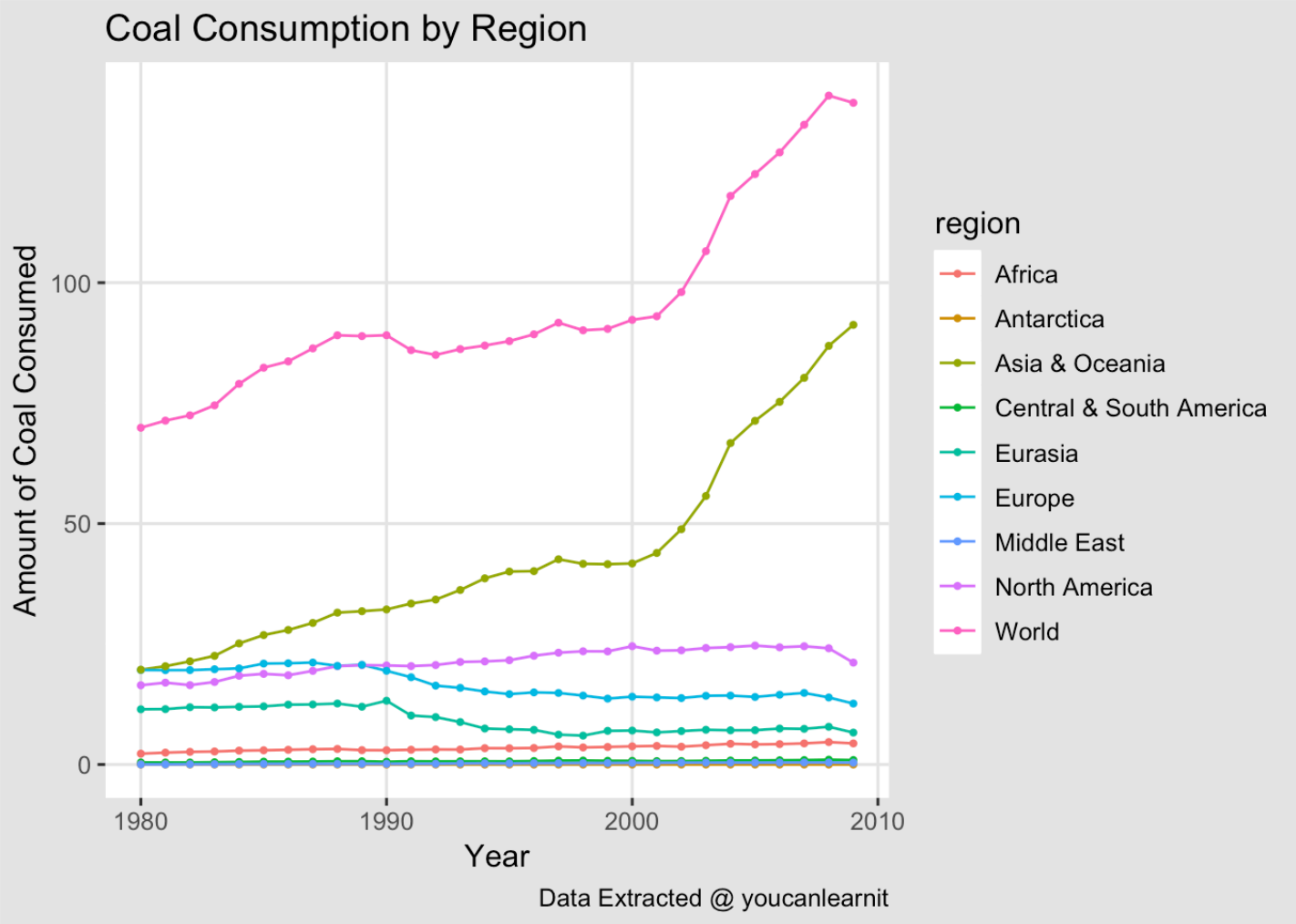
```
##  [1] "North America"            "Bermuda"
##  [3] "Canada"                   "Greenland"
##  [5] "Mexico"                   "Saint Pierre and Miquelon"
##  [7] "United States"            "Central & South America"
##  [9] "Antarctica"               "Antigua and Barbuda"
```

```
noncountries <- c("North America", "Central & South America", "Antarctica",
                  "Europe", "Eurasia", "Middle East", "Africa", "Asia & Oceania",
"World")

# Match and filter data with noncountries and countries
matches <- which(!is.na(match(coal_long$region, noncountries)))

coal_country <- coal_long[-matches,]
coal_region <- coal_long[matches,]
```

Visualization

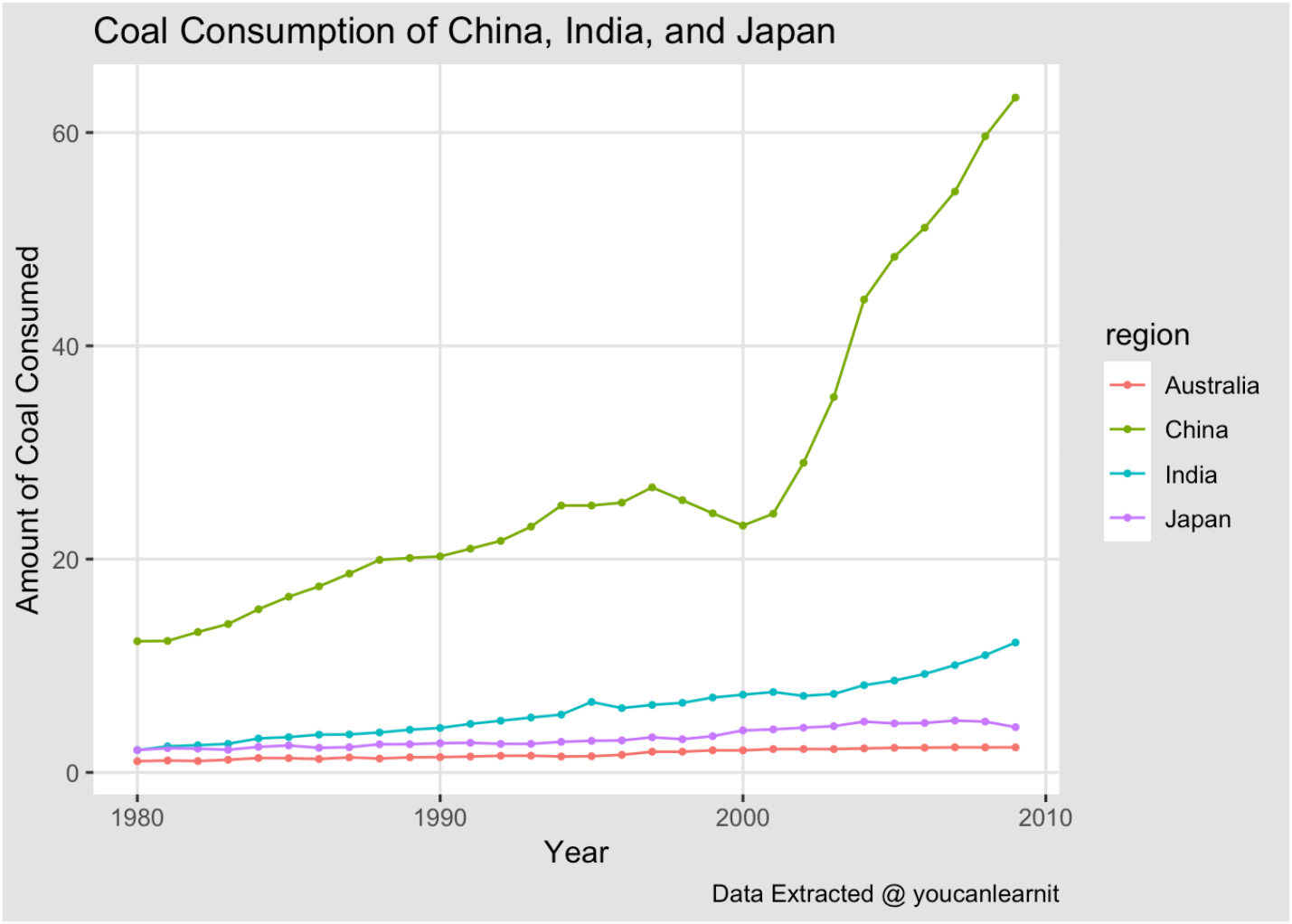
We have successfully grouped data into two different categories. We are going to use **scatter plot** for visualizing coal consumption of each region with respect to year elapsed.

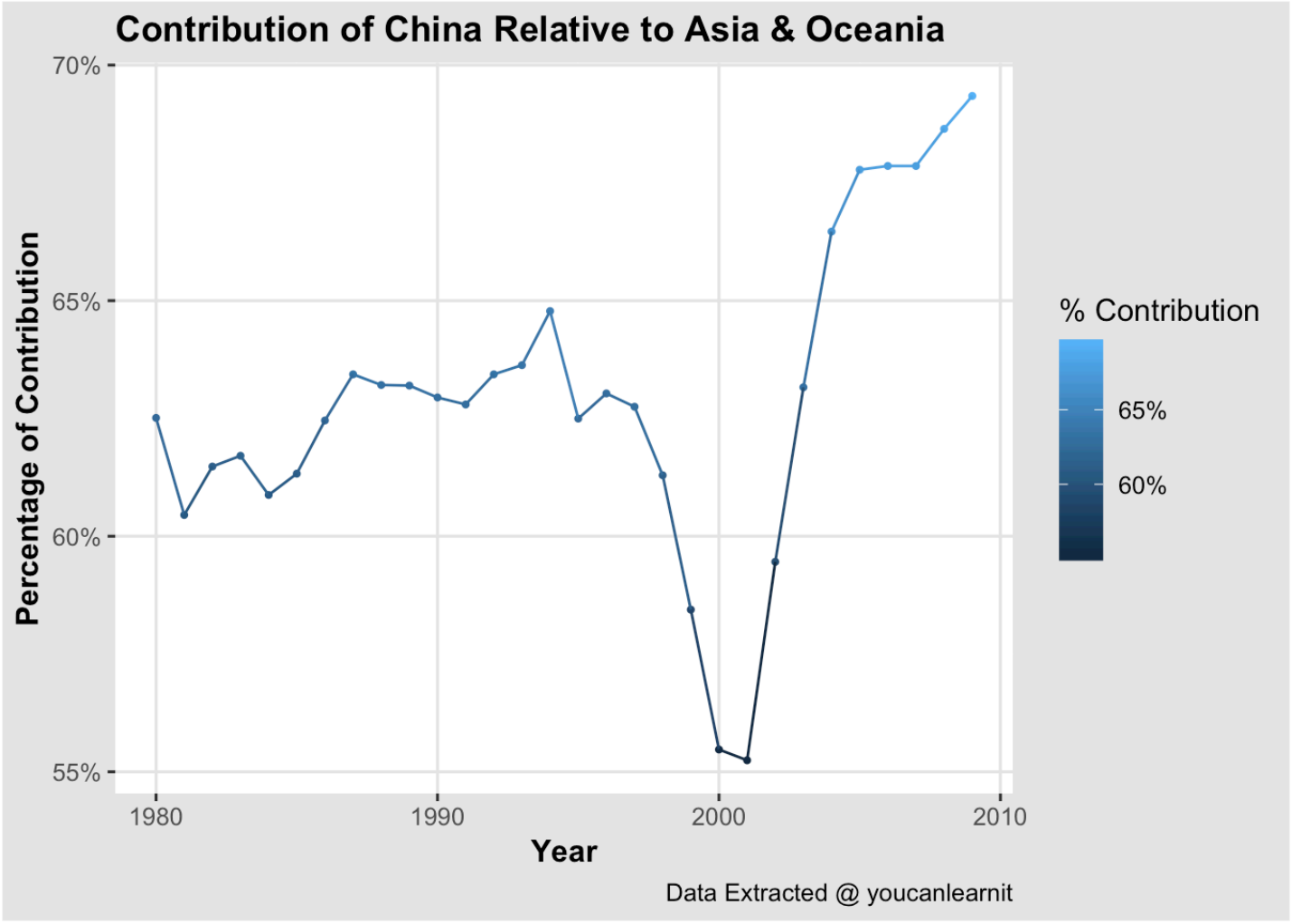
```
ggplot(coal_region, aes(x = year, y = coal_consumption)) +
  geom_line(aes(color = region)) +
  geom_point(aes(color = region), size = 0.8) +
  labs(title = "Coal Consumption by Region",
       caption = "Data Extracted @ youcanlearnit",
       x = "Year",
       y = "Amount of Coal Consumed") +
  theme_igray()
```



Further Analysis on the Graph

Asia & Oceania region was a major contributor of rapid rise in coal consumption with respect to World Average. I was interested in finding out which countries from Asia & Oceania are contributing in rapid rise of coal consumption starting year 2000.


Coal Consumption of China, India, and Japan
Amount of Coal Consumed
60
40
20
0
1980
1990
2000
2010
Year
region
Australia
China
India
Japan
Data Extracted @ youcanlearnit


Contribution of China Relative to Asia & Oceania
Percentage of Contribution
70%
65%
60%
55%
1980
1990
2000
2010
Year
% Contribution
65%
60%
Data Extracted @ youcanlearnit

According to two graphs, China began to consume coal rapidly at year 2000, becoming a major contributor in rapid increase of overall Asia & Oceania's overall coal consumption.

Second Case: Social Security Disability

Goal

The goal of this case study is to analyze how much of a **percentage** of people used Internet to apply for Social Security Disability overtime.

Again, import data from SSD_youcanlearnit (%22http://594442.youcanlearnit.net/ssadisability.csv%22) and attach required packages:

```
# Load the tidyverse, lubridate, and stringr
library(tidyverse)
library(lubridate)
library(stringr)

# Read in the coal dataset
ssa <- read_csv("http://594442.youcanlearnit.net/ssadisability.csv")

# Take a look at how this was imported
head(glimpse(ssa), n = 10)
```

```
## Observations: 10
## Variables: 25
## $ Fiscal_Year     <chr> "FY08", "FY09", "FY10", "FY11", "FY12", "FY13"...
## $ Oct_Total       <dbl> 176407, 244781, 286598, 299033, 227456, 224624...
## $ Oct_Internet    <dbl> 15082, 32578, 65533, 92856, 86811, 92542, 9840...
## $ Nov_Total       <dbl> 204287, 181161, 213297, 209553, 200140, 249910...
## $ Nov_Internet    <dbl> 17301, 25620, 50098, 63424, 71175, 107053, 117...
## $ Dec_Total       <dbl> 151687, 176107, 198733, 215239, 254766, 188183...
## $ Dec_Internet    <dbl> 14321, 27174, 44512, 62877, 91424, 79719, 8337...
## $ Jan_Total       <dbl> 162966, 249062, 265665, 264286, 221146, 199588...
## $ Jan_Internet    <dbl> 18391, 57908, 68843, 84944, 85848, 93703, 1253...
## $ Feb_Total       <dbl> 228623, 221368, 225319, 223625, 228519, 219604...
## $ Feb_Internet    <dbl> 26034, 50408, 58465, 71314, 83576, 101878, 108...
## $ Mar_Total       <dbl> 190716, 235360, 243266, 246630, 299267, 285923...
## $ Mar_Internet    <dbl> 21064, 53592, 62198, 77916, 112104, 129415, 11...
## $ Apr_Total       <dbl> 194403, 234304, 298065, 300359, 233685, 224804...
## $ Apr_Internet    <dbl> 22372, 53675, 76573, 94722, 88330, 101619, 112...
## $ May_Total       <dbl> 226549, 281343, 239409, 241673, 239503, 269955...
## $ May_Internet    <dbl> 26337, 65822, 65780, 77603, 93826, 123440, 134...
## $ June_Total      <dbl> 193094, 237329, 231964, 233351, 284136, 223238...
## $ June_Internet   <dbl> 22551, 54285, 67163, 79925, 113613, 104146, 11...
## $ July_Total      <dbl> 181552, 285172, 300442, 292949, 221745, 204072...
## $ July_Internet   <dbl> 22728, 66565, 92957, 105276, 91323, 98326, 106...
## $ August_Total    <dbl> 245429, 240611, 248284, 237555, 298458, 281828...
## $ August_Internet <dbl> 30580, 54915, 75535, 86514, 119795, 135423, 13...
## $ Sept_Internet   <dbl> 24141, 52687, 73403, 103564, 93375, 104270, 10...
## $ Sept_Total      <dbl> 186750, 228692, 238965, 280913, 230648, 214004...
```

```
## # A tibble: 10 x 25
##    Fiscal_Year Oct_Total Oct_Internet Nov_Total Nov_Internet Dec_Total
##    <chr>           <dbl>        <dbl>     <dbl>        <dbl>     <dbl>
##  1 FY08           176407        15082    204287        17301    151687
##  2 FY09           244781        32578    181161        25620    176107
##  3 FY10           286598        65533    213297        50098    198733
##  4 FY11           299033        92856    209553        63424    215239
##  5 FY12           227456        86811    200140        71175    254766
##  6 FY13           224624        92542    249910       107053    188183
##  7 FY14           206471        98400    237621       117934    175607
##  8 FY15           254294       133740    178697        96718    171692
##  9 FY16           244599       125971    174105        90199    169912
## 10 FY17           173396        90325    161320        85626    200515
## # ... with 19 more variables: Dec_Internet <dbl>, Jan_Total <dbl>,
## #   Jan_Internet <dbl>, Feb_Total <dbl>, Feb_Internet <dbl>,
## #   Mar_Total <dbl>, Mar_Internet <dbl>, Apr_Total <dbl>,
## #   Apr_Internet <dbl>, May_Total <dbl>, May_Internet <dbl>,
## #   June_Total <dbl>, June_Internet <dbl>, July_Total <dbl>,
## #   July_Internet <dbl>, August_Total <dbl>, August_Internet <dbl>,
## #   Sept_Internet <dbl>, Sept_Total <dbl>
```

Simplify Data

There are a lot of similar data in rows repeating themselves. Simplify the data with **gather** function:

```
# Use gather()
ssa_long <- gather(ssa, month, application, -Fiscal_Year)
head(ssa_long)
```

```
## # A tibble: 6 x 3
##   Fiscal_Year month     application
##   <chr>       <chr>           <dbl>
## 1 FY08        Oct_Total      176407
## 2 FY09        Oct_Total      244781
## 3 FY10        Oct_Total      286598
## 4 FY11        Oct_Total      299033
## 5 FY12        Oct_Total      227456
## 6 FY13        Oct_Total      224624
```

Now we want to separate the month and application method (ex. *Total*). Further filter data with **separate** function:

```
ssa_long <- separate(ssa_long, month, c("month","application_method"), sep = "_")
head(ssa_long)
```

```
## # A tibble: 6 x 4
##   Fiscal_Year month application_method application
##   <chr>       <chr> <chr>                    <dbl>
## 1 FY08        Oct   Total                   176407
## 2 FY09        Oct   Total                   244781
## 3 FY10        Oct   Total                   286598
## 4 FY11        Oct   Total                   299033
## 5 FY12        Oct   Total                   227456
## 6 FY13        Oct   Total                   224624
```

Abbreviation

First check month if all are abbreviated enough.

```
##  [1] "Oct"    "Nov"    "Dec"    "Jan"    "Feb"    "Mar"    "Apr"
##  [8] "May"    "June"   "July"   "August" "Sept"
```

Notice that June, July, and August are not abbreviated properly. Abbreviate month value. This is where **stringr** package comes in:

```
ssa_long$month <- substr(ssa_long$month, 1, 3)
unique(ssa_long$month)
```

```
##  [1] "Oct" "Nov" "Dec" "Jan" "Feb" "Mar" "Apr" "May" "Jun" "Jul" "Aug"
## [12] "Sep"
```

Fiscal Year

Now, we would like to replace fiscal year into 20's. Overwrite Fiscal Year to 20's:

```
ssa_long$Fiscal_Year <- str_replace(ssa_long$Fiscal_Year, "FY","20")
unique(ssa_long$Fiscal_Year)
```

```
##  [1] "2008" "2009" "2010" "2011" "2012" "2013" "2014" "2015" "2016" "2017"
```

Next, create a new column that has 1st day of the month:

```
ssa_long$Date <- dmy(paste('01', ssa_long$month, ssa_long$Fiscal_Year))

head(unique(ssa_long$Date))
```

```
## [1] "2008-10-01" "2009-10-01" "2010-10-01" "2011-10-01" "2012-10-01"
## [6] "2013-10-01"
```

Note that Fiscal Year was supposed to start at October of previous year. However, the Date column shows a **double counting of October-December in the same year**. We are going to fix this arbitrary:

```
# Find Dates with month >= 10 (October, November, December)
advanced_dates <- which(month(ssa_long$Date) >= 10)

# Subtract the year of these months by 1 to be the previous year
year(ssa_long$Date[advanced_dates]) <- year(ssa_long$Date[advanced_dates]) - 1
```

After fixing data, we realize that we have unnecessary columns. Remove columns that will not be used for analysis:

```
ssa_long$Fiscal_Year <- NULL
ssa_long$month <- NULL
```

Visualization

For visualization, turn the application method to a factor:

```
ssa_long$application_method <- as.factor(ssa_long$application_method)
summary(ssa_long)
```

```
##  application_method  application          Date
##  Internet:120       Min.   : 14321   Min.   :2007-10-01
##  Total   :120       1st Qu.: 91399   1st Qu.:2010-03-24
##                     Median :145344   Median :2012-09-16
##                     Mean   :154322   Mean   :2012-09-15
##                     3rd Qu.:224669   3rd Qu.:2015-03-08
##                     Max.   :300442   Max.   :2017-09-01
##                     NA's   :16
```

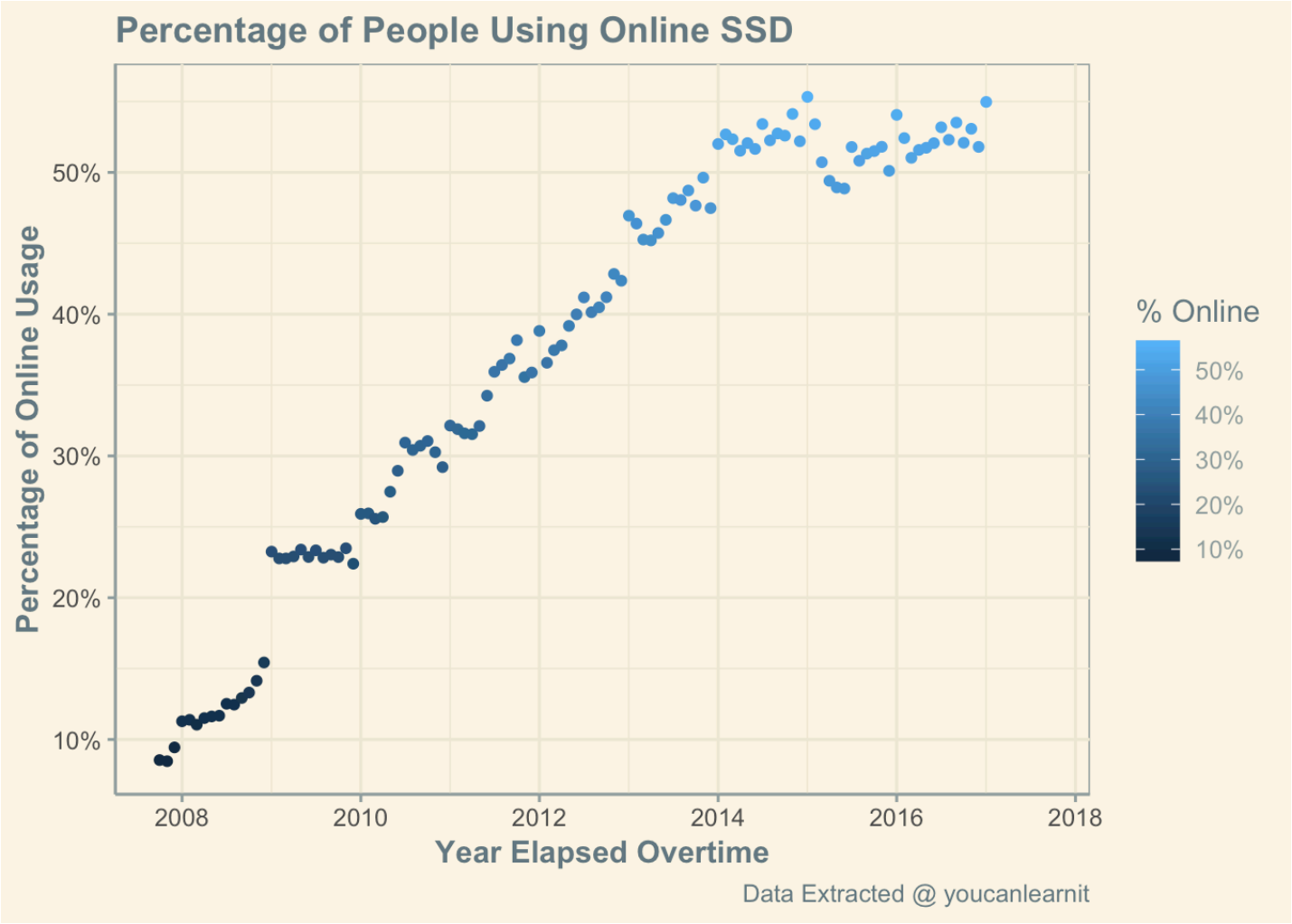
Use **spread** function to separate application method. Our primary goal is to analyze **a number of people using Internet**:

```
ssa <- spread(ssa_long, application_method, application)
head(ssa, n = 20)
```

```
## # A tibble: 20 x 3
##    Date       Internet  Total
##    <date>        <dbl>  <dbl>
##  1 2007-10-01    15082 176407
##  2 2007-11-01    17301 204287
##  3 2007-12-01    14321 151687
##  4 2008-01-01    18391 162966
##  5 2008-02-01    26034 228623
##  6 2008-03-01    21064 190716
##  7 2008-04-01    22372 194403
##  8 2008-05-01    26337 226549
##  9 2008-06-01    22551 193094
## 10 2008-07-01    22728 181552
## 11 2008-08-01    30580 245429
## 12 2008-09-01    24141 186750
## 13 2008-10-01    32578 244781
## 14 2008-11-01    25620 181161
## 15 2008-12-01    27174 176107
## 16 2009-01-01    57908 249062
## 17 2009-02-01    50408 221368
## 18 2009-03-01    53592 235360
## 19 2009-04-01    53675 234304
## 20 2009-05-01    65822 281343
```

Finally, visualize data with scatter plot:

```
# Create Percentage of people using online
ssa$online_percentage <- ssa$Internet / ssa$Total

# Create a plot
ggplot(ssa, aes(x = Date, y = online_percentage)) +
  geom_point(aes(color = online_percentage)) +
  theme_solarized() +
  theme(plot.title = element_text(size = 14, face = "bold"),
        axis.title.x = element_text(size = 12, face = "bold"),
        axis.title.y = element_text(size = 12, face = "bold")) +
  labs(title = "Percentage of People Using Online SSD",
       caption = "Data Extracted @ youcanlearnit",
       x = "Year Elapsed Overtime",
       y = "Percentage of Online Usage") +
  scale_y_continuous(labels = scales::percent) +
  scale_color_gradient(name = "% Online", labels = percent)
```



Conclusion

Since 2008, an annual percentage of people using online for Social Security Disability application have risen consistently.