R Notebook

The following is your first chunk to start with. Remember, you can add chunks using the menu above (Insert -> R) or using the keyboard shortcut Ctrl+Alt+I. A good practice is to use different code chunks to answer different questions. You can delete this comment if you like.

Other useful keyboard shortcuts include Alt- for the assignment operator, and Ctrl+Shift+M for the pipe operator. You can delete these reminders if you don't want them in your report.

```
setwd("C:/Users/munis/Desktop") #Don't forget to set your working directory before you start!

library("tidyverse")

## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.2.1     v purrr   0.3.3
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   1.0.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0

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

library("tidymodels")

## Registered S3 method overwritten by 'xts':
##   method     from
##   as.zoo.xts zoo

## -- Attaching packages --------------------------------------- tidymodels 0.0.3 --

## v broom     0.5.3     v recipes   0.1.9
## v dials     0.0.4     v rsample   0.0.5
## v infer     0.5.1     v yardstick 0.0.4
## v parsnip   0.0.5

## -- Conflicts ------------------------------------------
tidymodels_conflicts() --
## x scales::discard()   masks purrr::discard()
## x dplyr::filter()     masks stats::filter()
## x recipes::fixed()    masks stringr::fixed()
## x dplyr::lag()        masks stats::lag()
```

```
## x dials::margin()     masks ggplot2::margin()
## x yardstick::spec()   masks readr::spec()
## x recipes::step()     masks stats::step()
## x recipes::yj_trans() masks scales::yj_trans()
```

```
library("plotly")
```

```
##
## Attaching package: 'plotly'
```

```
## The following object is masked from 'package:ggplot2':
##
##     last_plot
```

```
## The following object is masked from 'package:stats':
##
##     filter
```

```
## The following object is masked from 'package:graphics':
##
##     layout
```

```
library("skimr")
```

```
library("tidyverse")
library("fpp3")
```

```
## Warning: package 'fpp3' was built under R version 3.6.3
```

```
## -- Attaching packages ---------------------------------------------- fpp3
0.2 --
```

```
## v lubridate   1.7.4     v feasts      0.1.3
## v tsibble     0.8.6     v fable       0.1.2
## v tsibbledata 0.1.0
```

```
## Warning: package 'tsibble' was built under R version 3.6.3
```

```
## Warning: package 'tsibbledata' was built under R version 3.6.3
```

```
## Warning: package 'feasts' was built under R version 3.6.3
```

```
## Warning: package 'fabletools' was built under R version 3.6.3
```

```
## Warning: package 'fable' was built under R version 3.6.3
```

```
## -- Conflicts ------------------------------------------------
fpp3_conflicts --
## x fabletools::accuracy()   masks yardstick::accuracy()
## x lubridate::date()        masks base::date()
## x scales::discard()        masks purrr::discard()
## x plotly::filter()         masks dplyr::filter(), stats::filter()
```

```
## x fabletools::generate()   masks infer::generate()
## x tsibble::id()            masks dplyr::id()
## x tsibble::interval()      masks lubridate::interval()
## x dplyr::lag()             masks stats::lag()
## x dials::margin()          masks ggplot2::margin()
## x tsibble::new_interval()  masks lubridate::new_interval()
## x fabletools::null_model() masks parsnip::null_model()

library("plotly")
library("skimr")
library("lubridate")
```

Q1)A)

```
tsLCOrg <-
  read_csv("C:/Users/munis/Desktop/lendingClub.csv")

## Parsed with column specification:
## cols(
##   date = col_date(format = ""),
##   state = col_character(),
##   avgLoans = col_double(),
##   totalLoans = col_double(),
##   avgTerm = col_double(),
##   avgIntRate = col_double(),
##   avgGrade = col_double(),
##   avgEmpLength = col_double(),
##   avgAnnualInc = col_double(),
##   avgVerifStatus = col_double(),
##   avgHomeOwner = col_double(),
##   avgOpenAcc = col_double(),
##   avgRevolBal = col_double(),
##   avgRevolUtil = col_double(),
##   avgTotalAcc = col_double(),
##   countOfLoans = col_double()
## )

tsLCOrg

## # A tibble: 4,943 x 16
##    date       state avgLoans totalLoans avgTerm avgIntRate avgGrade
avgEmpLength
##    <date>     <chr>    <dbl>      <dbl>   <dbl>      <dbl>    <dbl>
<dbl>
##  1 2008-01-01 AK       5600        5600      36      18.0       7
5
##  2 2008-03-01 AK      11700       23400      36      11.8       3
3.5
##  3 2008-06-01 AK       7500        7500      36      13.9       4
3
##  4 2008-12-01 AK      25000       25000      36      15.2       5
```

```
1
##  5 2009-01-01 AK      15000       30000      36      12.5       2.5
7
##  6 2009-03-01 AK      14662.      29325      36      13         3
7
##  7 2009-04-01 AK      20000       20000      36      11.9       2
5
##  8 2009-05-01 AK      16000       16000      36      12.2       2
2
##  9 2009-07-01 AK       1000        1000      36      11.9       2
10
## 10 2009-11-01 AK      11000       11000      36       8.94      1
7
## # ... with 4,933 more rows, and 8 more variables: avgAnnualInc <dbl>,
## #   avgVerifStatus <dbl>, avgHomeOwner <dbl>, avgOpenAcc <dbl>,
## #   avgRevolBal <dbl>, avgRevolUtil <dbl>, avgTotalAcc <dbl>,
## #   countOfLoans <dbl>
```

Q1)C)

```
head(tsLCOrg)
```

```
## # A tibble: 6 x 16
##   date       state avgLoans totalLoans avgTerm avgIntRate avgGrade
avgEmpLength
##   <date>     <chr>    <dbl>      <dbl>   <dbl>      <dbl>    <dbl>
<dbl>
## 1 2008-01-01 AK       5600        5600      36       18.0      7
5
## 2 2008-03-01 AK      11700       23400      36       11.8      3
3.5
## 3 2008-06-01 AK       7500        7500      36       13.9      4
3
## 4 2008-12-01 AK      25000       25000      36       15.2      5
1
## 5 2009-01-01 AK      15000       30000      36       12.5      2.5
7
## 6 2009-03-01 AK      14662.      29325      36       13        3
7
## # ... with 8 more variables: avgAnnualInc <dbl>, avgVerifStatus <dbl>,
## #   avgHomeOwner <dbl>, avgOpenAcc <dbl>, avgRevolBal <dbl>,
## #   avgRevolUtil <dbl>, avgTotalAcc <dbl>, countOfLoans <dbl>
```

```
nrow(tsLCOrg)
```

```
## [1] 4943
```

```
skim(tsLCOrg)
```

Data summary

Name	tsLCOrg
Number of rows	4943
Number of columns	16

_____

Column type frequency:

character	1
Date	1
numeric	14

_____

Group variables	None

Variable type: character

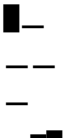
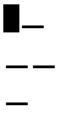
skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
state	0	1	2	2	0	51	0

Variable type: Date

skim_variable	n_missing	complete_rate	min	max	median	n_unique
date	0	1	2007-06-01	2017-03-01	2012-11-01	118

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
avgLoans	0	1	12606.97	3426.15	500.00	10583.33	13704.14	14954.03	29975.00	▁▃▇▁▁
totalLoans	0	1	4234850.33	9097259.44	500.00	115687.50	927825.00	4303437.50	126477500.00	▇▁▁▁▁
avgTerm	0	1	41.38	3.79	36.00	36.00	42.13	43.90	60.00	▇▁▁▁▁
avgIntRate	0	1	12.92	1.45	6.03	12.17	12.96	13.89	23.63	▁▇▁▁▁
avgGrade	0	1	2.77	0.56	1.00	2.57	2.75	2.92	7.00	▁▇▁▁▁
avgEmp	2	1	5.57	1.45	1.00	5.03	5.99	6.36	10.00	▁▃▇

Length										
avgAnnualInc	0	1	69476.03	18173.26	200.00	62275.50	69840.54	76669.39	556879.50	
avgVerifStatus	0	1	0.58	0.25	0.00	0.53	0.67	0.72	1.00	
avgHomeOwner	0	1	0.09	0.10	0.00	0.04	0.09	0.12	1.00	
avgOpenAcc	9	1	10.51	1.95	1.00	9.45	10.91	11.71	25.00	
avgRevolBal	0	1	15796.32	10076.87	0.00	12984.53	15465.00	17676.67	404867.50	
avgRevolUtil	12	1	52.59	11.02	0.00	49.06	53.97	57.94	99.40	
avgTotalAcc	9	1	23.89	4.71	1.00	22.20	24.61	26.29	61.00	
countOfLoans	0	1	287.00	601.47	1.00	11.00	67.00	290.00	8081.00	

Q1)B)

```
library(dplyr)
library(lubridate)
library(tsibble)

tsLCOrg <- as_tsibble(tsLCOrg, index=date, key = state)
```

Q1)D)

```
nyEcon <-
  read_csv("C:/Users/munis/Desktop/nyEcon.csv")

## Parsed with column specification:
## cols(
##   date = col_character(),
##   state = col_character(),
##   NYCPI = col_double(),
##   NYUnemployment = col_double(),
##   NYCondoPriceIdx = col_double(),
##   NYSnapBenefits = col_double()
## )

nrow(tsLCOrg)
```

```
## [1] 4943

tsLCOrg%>%group_by(state)%>%tally()

## # A tsibble: 4,943 x 3 [1D]
## # Key:       state [51]
##    state date           n
##    <chr> <date>     <int>
##  1 AK    2008-01-01     1
##  2 AK    2008-03-01     1
##  3 AK    2008-06-01     1
##  4 AK    2008-12-01     1
##  5 AK    2009-01-01     1
##  6 AK    2009-03-01     1
##  7 AK    2009-04-01     1
##  8 AK    2009-05-01     1
##  9 AK    2009-07-01     1
## 10 AK    2009-11-01     1
## # ... with 4,933 more rows
```

Q1)E)

```
Census <-
  read_csv("C:/Users/munis/Desktop/2010 Census.csv")

## Warning: Missing column names filled in: 'X2' [2], 'X3' [3], 'X4' [4],
'X5' [5],
## 'X6' [6], 'X7' [7], 'X8' [8], 'X9' [9], 'X10' [10], 'X11' [11], 'X12'
[12],
## 'X13' [13]

## Parsed with column specification:
## cols(
##   `table with row headers in column A and column headers in rows 3 through
4. (leading dots indicate sub-parts)` = col_character(),
##   X2 = col_character(),
##   X3 = col_character(),
##   X4 = col_character(),
##   X5 = col_number(),
##   X6 = col_number(),
##   X7 = col_number(),
##   X8 = col_number(),
##   X9 = col_number(),
##   X10 = col_number(),
##   X11 = col_number(),
##   X12 = col_number(),
##   X13 = col_number()
## )

Census
```

```
## # A tibble: 66 x 13
##    `table with row~ X2    X3    X4         X5      X6      X7      X8
X9
##    <chr>            <chr> <chr> <chr>   <dbl>   <dbl>   <dbl>   <dbl>
<dbl>
##  1 Table 1. Annual~ <NA>  <NA>  <NA>  NA      NA      NA      NA      NA
##  2 Geographic Area  Apri~ <NA>  Popu~ NA      NA      NA      NA      NA
##  3 <NA>             Cens~ Esti~ 2010   2.01e3  2.01e3  2.01e3  2.01e3
2.02e3
##  4 United States    30,8~ 30,8~ 30,9~  3.12e8  3.14e8  3.16e8  3.18e8
3.21e8
##  5 Northeast        5,53~ 5,53~ 5,53~  5.56e7  5.58e7  5.59e7  5.60e7
5.60e7
##  6 Midwest          6,69~ 6,69~ 6,69~  6.72e7  6.73e7  6.76e7  6.77e7
6.79e7
##  7 South            11,4~ 11,4~ 11,4~  1.16e8  1.17e8  1.18e8  1.20e8
1.21e8
##  8 West             7,19~ 7,19~ 7,21~  7.28e7  7.35e7  7.42e7  7.49e7
7.57e7
##  9 .Alabama         47,7~ 47,8~ 47,8~  4.80e6  4.82e6  4.83e6  4.84e6
4.85e6
## 10 .Alaska          7,10~ 7,10~ 7,13~  7.22e5  7.30e5  7.37e5  7.36e5
7.37e5
## # ... with 56 more rows, and 4 more variables: X10 <dbl>, X11 <dbl>, X12
<dbl>,
## #   X13 <dbl>
```

```
Census<-Census%>% select(1,4)
Census<-Census[-c(1,2,3,4,5,6,7,8),]
#head(Census)

Census$X4<-gsub("\\,","",Census$X4)
Census$X4<-as.numeric(Census$X4)
Census[1][Census[1] == '.Alabama'] <- 'AL'

Census[1][Census[1] == '.Alaska'] <- 'AK'
Census[1][Census[1] == '.Arizona'] <- 'AZ'
Census[1][Census[1] == '.Arkansas'] <- 'AR'
Census[1][Census[1] == '.California'] <- 'CA'
Census[1][Census[1] == '.Colorado'] <- 'CO'
Census[1][Census[1] == '.Connecticut'] <- 'CT'
Census[1][Census[1] == '.Delaware'] <- 'DE'



Census[1][Census[1] == '.District of Columbia']<- 'DC'
Census[1][Census[1] == '.Florida']<- 'FL'
Census[1][Census[1] == '.Georgia']<- 'GA'
Census[1][Census[1] == '.Hawaii']<- 'HI'
Census[1][Census[1] == '.Idaho']<- 'ID'
Census[1][Census[1] == '.Illinois']<- 'IL'
```

```
Census[1][Census[1] == '.Indiana']<- 'IN'
Census[1][Census[1] == '.Iowa']<- 'IA'
Census[1][Census[1] == '.Kansas']<- 'KS'
Census[1][Census[1] == '.Kentucky']<- 'KY'
Census[1][Census[1] == '.Louisiana']<- 'LA'
Census[1][Census[1] == '.Maine']<- 'ME'
Census[1][Census[1] == '.Maryland']<- 'MD'
Census[1][Census[1] == '.Massachusetts']<- 'MA'
Census[1][Census[1] == '.Michigan']<- 'MI'
Census[1][Census[1] == '.Minnesota']<- 'MN'
Census[1][Census[1] == '.Mississippi']<- 'MS'
Census[1][Census[1] == '.Missouri']<- 'MO'
Census[1][Census[1] == '.Montana']<- 'MT'
Census[1][Census[1] == '.Nebraska']<- 'NE'
Census[1][Census[1] == '.Nevada']<- 'NV'
Census[1][Census[1] == '.New Hampshire']<- 'NH'
Census[1][Census[1] == '.New Jersey']<- 'NJ'
Census[1][Census[1] == '.New Mexico']<- 'NM'
Census[1][Census[1] == '.New York']<- 'NY'
Census[1][Census[1] == '.North Carolina']<- 'NC'
Census[1][Census[1] == '.North Dakota']<- 'ND'
Census[1][Census[1] == '.Ohio']<- 'OH'
Census[1][Census[1] == '.Oklahoma']<- 'OK'
Census[1][Census[1] == '.Oregon']<- 'OR'
Census[1][Census[1] == '.Pennsylvania']<- 'PA'
Census[1][Census[1] == '.Rhode Island']<- 'RI'
Census[1][Census[1] == '.South Carolina']<- 'SC'
Census[1][Census[1] == '.South Dakota']<- 'SD'
Census[1][Census[1] == '.Tennessee']<- 'TN'
Census[1][Census[1] == '.Texas']<- 'TX'
Census[1][Census[1] == '.Utah']<- 'UT'
Census[1][Census[1] == '.Vermont']<- 'VT'
Census[1][Census[1] == '.Virginia']<- 'VA'
Census[1][Census[1] == '.Washington']<- 'WA'
Census[1][Census[1] == '.West Virginia']<- 'WV'
Census[1][Census[1] == '.Wisconsin']<- 'WI'
Census[1][Census[1] == '.Wyoming']<- 'WY'
```

```
tsLCOrg
```

```
## # A tsibble: 4,943 x 16 [1D]
## # Key:       state [51]
##    date       state avgLoans totalLoans avgTerm avgIntRate avgGrade
avgEmpLength
```

```
##    <date>     <chr>    <dbl>      <dbl>   <dbl>      <dbl>    <dbl>
<dbl>
##  1 2008-01-01 AK       5600        5600      36      18.0       7
5
##  2 2008-03-01 AK      11700       23400      36      11.8       3
3.5
##  3 2008-06-01 AK       7500        7500      36      13.9       4
3
##  4 2008-12-01 AK      25000       25000      36      15.2       5
1
##  5 2009-01-01 AK      15000       30000      36      12.5       2.5
7
##  6 2009-03-01 AK      14662.      29325      36      13         3
7
##  7 2009-04-01 AK      20000       20000      36      11.9       2
5
##  8 2009-05-01 AK      16000       16000      36      12.2       2
2
##  9 2009-07-01 AK       1000        1000      36      11.9       2
10
## 10 2009-11-01 AK      11000       11000      36       8.94      1
7
## # ... with 4,933 more rows, and 8 more variables: avgAnnualInc <dbl>,
## #   avgVerifStatus <dbl>, avgHomeOwner <dbl>, avgOpenAcc <dbl>,
## #   avgRevolBal <dbl>, avgRevolUtil <dbl>, avgTotalAcc <dbl>,
## #   countOfLoans <dbl>
```

```
colnames(Census)
```

```
## [1] "table with row headers in column A and column headers in rows 3
through 4. (leading dots indicate sub-parts)"
## [2] "X4"
```

```
  colnames(Census)[1]<-"state"
colnames(Census)[2]<-"totalPop"
```

```
Census
```

```
## # A tibble: 58 x 2
##    state totalPop
##    <chr>    <dbl>
##  1 AL     4785437
##  2 AK      713910
##  3 AZ     6407172
##  4 AR     2921964
##  5 CA    37319502
##  6 CO     5047349
##  7 CT     3579114
```

```
##  8 DE      899593
##  9 DC      605226
## 10 FL    18845537
## # ... with 48 more rows
```

```
#Q1Ei)


tsLCOrg <- merge(tsLCOrg,Census,by="state")
```

Q1)E0ii)

```
tsLCOrg<-
tsLCOrg%>%group_by(state)%>%mutate(loansPerCapita=totalLoans/totalPop)

#total <- merge(tsLCOrg,Census,by="state"

nyEcon$date <- as.Date(nyEcon$date, "%m/%d/%y")

tsLC1<- merge(tsLCOrg,nyEcon,by=c("date","state"))

tsLC1
```

```
##           date state  avgLoans totalLoans  avgTerm avgIntRate avgGrade
## 1   2007-06-01    NY  3381.250      13525 36.00000    8.77750 1.750000
## 2   2007-07-01    NY  8611.111      77500 36.00000   11.04889 3.222222
## 3   2007-08-01    NY  7357.692      95650 36.00000   11.15077 3.307692
## 4   2007-09-01    NY  8388.636      92275 36.00000   11.25273 3.454545
## 5   2007-10-01    NY  8804.167     105650 36.00000   12.85417 4.166667
## 6   2007-11-01    NY  7634.375     122150 36.00000   11.53875 3.312500
## 7   2007-12-01    NY 12745.139     458825 36.00000   12.02167 3.694444
## 8   2008-01-01    NY  7807.609     179575 36.00000   11.94478 3.347826
## 9   2008-02-01    NY 12590.476     264400 36.00000   11.90762 3.142857
## 10  2008-03-01    NY 10498.837     451450 36.00000   11.76837 3.139535
## 11  2008-04-01    NY 11834.375     189350 36.00000   12.23937 3.312500
## 12  2008-05-01    NY  5155.000      25775 36.00000   13.74400 4.000000
## 13  2008-06-01    NY  6042.857      42300 36.00000   12.74000 3.714286
## 14  2008-07-01    NY  6745.833      80950 36.00000   11.07750 2.583333
## 15  2008-08-01    NY  6211.111      55900 36.00000   12.68111 3.666667
## 16  2008-09-01    NY  6262.500      50100 36.00000   10.36250 2.125000
## 17  2008-10-01    NY  5016.667      30100 36.00000   12.38000 3.166667
## 18  2008-11-01    NY  9163.333     137450 36.00000   11.87000 2.666667
## 19  2008-12-01    NY 11415.385     148400 36.00000   13.08462 3.153846
## 20  2009-01-01    NY 10618.182     233600 36.00000   12.16000 2.590909
## 21  2009-02-01    NY 10256.000     256400 36.00000   12.78640 2.920000
## 22  2009-03-01    NY 11423.485     376975 36.00000   13.51788 3.363636
## 23  2009-04-01    NY  8189.130     188350 36.00000   12.64348 2.826087
## 24  2009-05-01    NY  9974.000     249350 36.00000   12.10960 2.400000
## 25  2009-06-01    NY  9179.545     403900 36.00000   12.65432 2.727273
## 26  2009-07-01    NY 10018.333     300550 36.00000   12.62333 2.766667
```

```
## 27  2009-08-01    NY 10754.891     494725 36.00000   12.74848 2.739130
## 28  2009-09-01    NY  9407.843     479800 36.00000   13.18235 2.901961
## 29  2009-10-01    NY  9495.673     493775 36.00000   12.76000 2.692308
## 30  2009-11-01    NY  9814.167     588850 36.00000   12.19817 2.500000
## 31  2009-12-01    NY 10956.750    1095675 36.00000   12.79850 2.670000
## 32  2010-01-01    NY 11579.167     833700 36.00000   12.07083 2.416667
## 33  2010-02-01    NY 11582.394     822350 36.00000   11.57408 2.464789
## 34  2010-03-01    NY 10469.940     879475 36.00000   12.66024 2.880952
## 35  2010-04-01    NY 11348.077    1180200 36.00000   12.12327 2.673077
## 36  2010-05-01    NY 11752.778     951975 39.85185   12.17407 2.641975
## 37  2010-06-01    NY  9385.882     797800 42.49412   12.21612 2.588235
## 38  2010-07-01    NY  9710.550    1058450 44.36697   12.24477 2.660550
## 39  2010-08-01    NY 10698.162    1454950 43.58824   12.64191 2.764706
## 40  2010-09-01    NY 10692.578    1368650 42.93750   12.52758 2.757812
## 41  2010-10-01    NY 11240.714    1573700 46.45714   11.86064 2.707143
## 42  2010-11-01    NY 11520.755    1221200 45.96226   11.29594 2.745283
## 43  2010-12-01    NY 10490.962    1363825 43.38462   10.88962 2.538462
## 44  2011-01-01    NY 11091.146    1597125 42.33333   11.31583 2.583333
## 45  2011-02-01    NY 11000.000    1650000 44.48000   11.48740 2.606667
## 46  2011-03-01    NY 11218.321    1469600 44.42748   11.30573 2.526718
## 47  2011-04-01    NY 11336.653    1337725 44.74576   11.83890 2.720339
## 48  2011-05-01    NY 11259.375    1711425 47.84211   12.90211 2.835526
## 49  2011-06-01    NY 11405.000    1995875 43.95429   12.30417 2.588571
## 50  2011-07-01    NY 11605.669    1996175 43.81395   12.73215 2.784884
## 51  2011-08-01    NY 12607.051    1966700 46.15385   12.86083 2.794872
## 52  2011-09-01    NY 11767.995    2435975 42.14493   11.60106 2.352657
## 53  2011-10-01    NY 12257.587    2463775 42.92537   11.91776 2.318408
## 54  2011-11-01    NY 11578.302    2454600 42.22642   12.31198 2.396226
## 55  2011-12-01    NY 14231.517    3002850 45.21327   13.50100 2.767773
## 56  2012-01-01    NY 13275.261    3810000 40.85017   12.79254 2.571429
## 57  2012-02-01    NY 14020.543    3617300 40.37209   12.87655 2.565891
## 58  2012-03-01    NY 13939.196    3902975 40.54286   12.62225 2.328571
## 59  2012-04-01    NY 13452.975    4748900 40.21530   12.82479 2.436261
## 60  2012-05-01    NY 13536.532    4020350 40.52525   12.88202 2.464646
## 61  2012-06-01    NY 13237.588    5625975 40.96941   14.17245 2.795294
## 62  2012-07-01    NY 12850.637    6052650 40.02548   14.08556 2.696391
## 63  2012-08-01    NY 12948.647    7419575 39.64398   13.77435 2.574171
## 64  2012-09-01    NY 12631.020    6871275 39.57353   13.69415 2.560662
## 65  2012-10-01    NY 13423.852    7597900 40.53710   14.16574 2.708481
## 66  2012-11-01    NY 14046.518    7262050 40.68859   14.65545 2.843327
## 67  2012-12-01    NY 15600.514    8346275 41.47290   14.34166 2.738318
## 68  2013-01-01    NY 15519.805    8737650 40.85968   14.35119 2.706927
## 69  2013-02-01    NY 15606.876    9816725 40.31161   14.02580 2.626391
## 70  2013-03-01    NY 14829.154    9549975 40.43478   14.29806 2.723602
## 71  2013-04-01    NY 15091.843   12858250 41.57746   14.87629 2.884977
## 72  2013-05-01    NY 13686.726   10826200 41.46144   15.42661 3.069532
## 73  2013-06-01    NY 13731.534   11767925 41.18086   15.36757 3.038506
## 74  2013-07-01    NY 13900.686   13177850 41.24051   14.75597 2.965190
## 75  2013-08-01    NY 13897.074   15912150 42.33013   15.03644 3.043668
## 76  2013-09-01    NY 14067.564   16979550 42.62138   15.13979 3.030655
```

```
## 77  2013-10-01    NY 14205.553   16052275 42.71150   15.08259 2.925664
## 78  2013-11-01    NY 14282.866   16296750 41.74233   14.31404 2.759860
## 79  2013-12-01    NY 14684.767   18267850 41.71061   14.33932 2.786174
## 80  2014-01-01    NY 14575.434   18481650 41.79180   14.38997 2.829653
## 81  2014-02-01    NY 15021.957   19498500 42.69337   14.63555 2.942989
## 82  2014-03-01    NY 14919.815   18560250 42.84887   14.67084 2.984727
## 83  2014-04-01    NY 14605.944   21806675 43.00871   14.53123 2.943737
## 84  2014-05-01    NY 14488.012   24745525 43.29274   14.11339 2.972482
## 85  2014-06-01    NY 14684.292   23230550 43.47914   13.93224 3.005057
## 86  2014-07-01    NY 14320.656   32966150 42.79757   13.86633 2.993050
## 87  2014-08-01    NY 14902.990   26288875 42.93878   14.00210 3.036281
## 88  2014-09-01    NY 14757.978   13134600 42.63371   13.50511 2.875281
## 89  2014-10-01    NY 14635.149   48091100 42.97505   13.38456 2.845100
## 90  2014-11-01    NY 14806.833   32071600 43.93352   13.05176 2.813943
## 91  2014-12-01    NY 15571.502   14356925 43.65293   12.71026 2.754881
## 92  2015-01-01    NY 14944.191   40259650 43.42984   13.13482 2.848924
## 93  2015-02-01    NY 15396.320   27405450 43.51011   12.73904 2.774719
## 94  2015-03-01    NY 15098.861   31481125 43.68921   12.65954 2.779376
## 95  2015-04-01    NY 15047.001   42778625 43.43721   12.79454 2.816743
## 96  2015-05-01    NY 14863.917   37248975 43.32642   12.78409 2.824421
## 97  2015-06-01    NY 15027.126   34111575 43.89780   12.82343 2.853744
## 98  2015-07-01    NY 14939.054   54901025 43.37959   12.71554 2.817959
## 99  2015-08-01    NY 15054.399   44320150 43.30435   12.69290 2.818274
## 100 2015-09-01    NY 15598.453   37311500 43.82609   12.59610 2.786371
## 101 2015-10-01    NY 15032.134   60263825 43.10002   12.24461 2.684460
## 102 2015-11-01    NY 14913.936   45055000 43.26117   12.10927 2.658722
## 103 2015-12-01    NY 14819.214   53719650 43.21655   12.39905 2.744276
## 104 2016-01-01    NY 15949.942   41007300 42.52509   12.38817 2.693504
## 105 2016-02-01    NY 15472.233   46973700 42.10277   12.38330 2.632740
## 106 2016-03-01    NY 15414.928   74993625 42.12703   12.65648 2.701336
## 107 2016-04-01    NY 14741.327   44828375 40.96416   12.45013 2.638606
## 108 2016-05-01    NY 14664.317   32290825 41.24251   12.67389 2.673025
## 109 2016-06-01    NY 14591.173   41365975 41.70582   12.65541 2.587302
## 110 2016-07-01    NY 13967.108   40043700 42.16952   13.81774 2.811999
## 111 2016-08-01    NY 14211.694   46969650 41.59879   13.87241 2.831165
## 112 2016-09-01    NY 13749.980   34594950 41.01749   14.21786 2.920906
## 113 2016-10-01    NY 13922.201   36309100 41.16258   13.56032 2.749233
## 114 2016-11-01    NY 14370.424   38785775 41.44202   13.70628 2.768433
## 115 2016-12-01    NY 13584.823   40211075 41.15676   14.04329 2.837838
## 116 2017-01-01    NY 15001.597   38974150 41.32102   13.72000 2.770208
## 117 2017-02-01    NY 14993.932   33916275 41.82493   13.56776 2.735632
## 118 2017-03-01    NY 14946.814   47500975 41.63373   13.52344 2.730648
##     avgEmpLength avgAnnualInc avgVerifStatus avgHomeOwner avgOpenAcc
## 1       2.250000    113333.33     0.00000000   0.00000000        NaN
## 2       2.777778    101875.00     0.00000000   0.00000000  11.714286
## 3       1.230769     69363.64     0.00000000   0.07692308   8.200000
## 4       3.000000     55806.00     0.00000000   0.18181818   6.000000
## 5       2.333333     90516.67     0.00000000   0.00000000   8.166667
## 6       2.562500     75750.00     0.00000000   0.12500000   7.750000
## 7       4.277778     71087.28     0.00000000   0.05555556   8.805556
```

```
## 8       3.521739     55731.64     0.00000000   0.04347826  10.086957
## 9       4.571429     72590.00     0.00000000   0.23809524  12.095238
## 10      3.325581     58959.42     0.02325581   0.04651163   8.372093
## 11      5.062500     73751.69     0.12500000   0.18750000  11.937500
## 12      5.400000    151500.00     0.20000000   0.00000000  10.400000
## 13      4.000000     69642.86     0.42857143   0.14285714   9.857143
## 14      4.083333     56595.58     0.16666667   0.00000000   6.666667
## 15      3.777778     55840.00     0.11111111   0.00000000   8.666667
## 16      5.750000    106500.00     0.25000000   0.12500000   7.000000
## 17      6.000000     86342.50     0.66666667   0.16666667  11.666667
## 18      5.733333     95148.80     0.20000000   0.20000000   8.800000
## 19      5.384615     79096.00     0.76923077   0.00000000   9.230769
## 20      5.272727     86706.23     0.50000000   0.13636364   9.590909
## 21      4.960000     69973.12     0.80000000   0.04000000   9.400000
## 22      4.272727     63359.15     0.63636364   0.12121212  10.424242
## 23      4.521739     82680.91     0.26086957   0.08695652   8.869565
## 24      4.080000     66723.36     0.52000000   0.08000000   8.440000
## 25      3.840909     78059.61     0.54545455   0.09090909   9.522727
## 26      3.233333     63078.63     0.60000000   0.03333333   9.100000
## 27      4.478261     76460.17     0.56521739   0.17391304  10.478261
## 28      4.137255     69999.43     0.45098039   0.03921569   9.882353
## 29      4.653846     63282.70     0.25000000   0.15384615   7.788462
## 30      3.783333     68176.49     0.20000000   0.08333333   9.816667
## 31      4.170000     85488.93     0.16000000   0.22000000   8.990000
## 32      4.750000     77813.90     0.05555556   0.16666667   8.805556
## 33      5.000000     67746.29     0.11267606   0.14084507   9.098592
## 34      4.817073     77900.48     0.14285714   0.10714286   9.559524
## 35      4.320000     82840.63     0.41346154   0.08653846   9.057692
## 36      5.291139     62661.71     0.45679012   0.03703704   9.135802
## 37      4.216867     62532.41     0.64705882   0.09411765   8.870588
## 38      4.953704     63805.59     0.65137615   0.07339450   9.495413
## 39      5.204545     73350.97     0.69852941   0.11764706   9.389706
## 40      5.032787     76160.09     0.71093750   0.06250000   9.687500
## 41      5.326087     80375.95     0.66428571   0.09285714   9.914286
## 42      4.598039     69035.15     0.61320755   0.10377358   9.226415
## 43      4.443548     66669.67     0.67692308   0.06153846   9.492308
## 44      4.985075     68890.06     0.67361111   0.08333333   9.131944
## 45      5.441379     69722.14     0.72000000   0.08000000   9.833333
## 46      5.426357     67510.32     0.64885496   0.08396947   9.473282
## 47      5.025862     69446.21     0.70338983   0.05932203   8.805085
## 48      4.802721     68107.94     0.71710526   0.09210526   9.250000
## 49      5.251462     68883.26     0.72000000   0.13714286   9.234286
## 50      5.678788     68447.51     0.65697674   0.08139535   9.389535
## 51      5.300654     71607.83     0.69230769   0.06410256   9.262821
## 52      5.785000     83991.24     0.69082126   0.07246377   9.483092
## 53      5.712821     79525.26     0.72139303   0.07462687   9.701493
## 54      5.272727     70282.81     0.71698113   0.06603774  10.198113
## 55      5.558252     70303.17     0.72511848   0.05687204   9.625592
## 56      5.711191     69976.75     0.61324042   0.06968641   9.560976
## 57      5.219608     70780.03     0.72480620   0.07751938   9.992248
```

```
## 58      5.593985     74083.95     0.72142857   0.07857143  10.046429
## 59      5.429825     71889.66     0.67705382   0.08781870  10.294618
## 60      5.867133     72980.38     0.77441077   0.09090909  10.505051
## 61      5.815085     73062.91     0.62352941   0.07529412  10.722353
## 62      5.570485     66854.39     0.53927813   0.08067941  11.363057
## 63      5.717082     70199.88     0.52181501   0.09947644  11.148342
## 64      5.616412     73749.07     0.53308824   0.07904412  10.437500
## 65      5.688555     68731.63     0.57773852   0.08833922  10.551237
## 66      6.012500     75092.69     0.59381044   0.10251451  11.127660
## 67      6.405039     91651.72     0.68411215   0.09345794  11.555140
## 68      5.789279     82929.71     0.68206039   0.06749556  11.646536
## 69      6.208197     80990.40     0.66454690   0.07631161  11.790143
## 70      6.098361     73549.39     0.59937888   0.08540373  11.736025
## 71      6.147420     80297.65     0.67018779   0.09624413  11.661972
## 72      6.204272     71880.45     0.67130215   0.09102402  11.036662
## 73      6.366300     71103.51     0.62427071   0.07467911  11.348891
## 74      6.266447     74783.08     0.66244726   0.09388186  11.391350
## 75      6.283871     76939.87     0.69344978   0.11004367  11.178166
## 76      6.409794     74293.63     0.64788732   0.10439105  11.419221
## 77      6.441948     74578.92     0.77787611   0.11061947  11.469027
## 78      6.282288     74100.06     0.74846626   0.10429448  11.665206
## 79      6.080034     76753.61     0.70900322   0.10691318  11.451768
## 80      6.339450     77391.43     0.72239748   0.10962145  11.835174
## 81      6.311881     84130.31     0.74036980   0.11864407  12.156394
## 82      6.441821     79476.23     0.77411576   0.09244373  11.612540
## 83      6.343528     77584.24     0.80843938   0.12324180  12.027461
## 84      6.320173     77493.33     0.71779859   0.09074941  11.902225
## 85      6.103974     77410.01     0.68204804   0.10366625  11.965234
## 86      6.150474     76441.83     0.64682884   0.08818419  12.080365
## 87      6.073258     77636.68     0.67233560   0.12018141  12.206349
## 88      6.123399     78326.40     0.65842697   0.11573034  12.262921
## 89      6.371741     77116.88     0.72215460   0.11472915  12.088862
## 90      6.238938     76824.16     0.72345337   0.12557710  11.915051
## 91      6.095890     81044.64     0.70715835   0.11496746  11.643167
## 92      6.061393     79814.40     0.75129918   0.11284336  12.331849
## 93      5.935966     78576.16     0.74044944   0.11235955  12.228090
## 94      6.168864     78694.16     0.70023981   0.11079137  12.206235
## 95      6.138097     79872.33     0.70840661   0.10306015  12.358072
## 96      6.129777     78525.79     0.74022346   0.10135674  12.060255
## 97      6.181481     77711.86     0.69911894   0.11718062  12.281057
## 98      6.233094     80144.81     0.70176871   0.10802721  12.244626
## 99      6.021525     79967.22     0.70855978   0.12058424  11.916780
## 100     6.048889     83385.10     0.74707358   0.13085284  12.165970
## 101     6.332016     82705.43     0.73908705   0.11673734  12.297830
## 102     6.249112     83964.77     0.76266137   0.11850381  11.977160
## 103     6.017788     82481.70     0.73627586   0.12551724  12.200276
## 104     6.199917     88192.37     0.65616492   0.12796577  12.343446
## 105     6.281502     82799.57     0.68280632   0.12615283  12.401186
## 106     6.198451     82127.77     0.66228160   0.12785200  12.304625
## 107     6.340569     80722.66     0.66885893   0.14107202  12.221966
```

```
## 108     6.249383     81131.30     0.70163488   0.12761126  12.166667
## 109     6.017505     82095.31     0.77319224   0.13015873  12.020811
## 110     6.139698     79664.28     0.74084409   0.11370771  11.914545
## 111     6.228498     82002.36     0.72798790   0.12465961  11.885930
## 112     6.071399     81549.30     0.73569157   0.13791733  11.798490
## 113     6.263723     82219.08     0.64800613   0.12423313  12.016871
## 114     6.218775     85300.13     0.67432382   0.10781771  11.930715
## 115     5.994900     84172.41     0.69864865   0.12432432  11.840203
## 116     6.146109     88561.90     0.67474981   0.12779061  12.091224
## 117     6.125717     83215.52     0.68700265   0.11538462  11.984527
## 118     6.174882     87312.97     0.66928886   0.12523600  12.001573
##     avgRevolBal avgRevolUtil avgTotalAcc countOfLoans totalPop
loansPerCapita
## 1         0.000          NaN         NaN            4 19399878
0.0006971693
## 2      8465.889     42.10000    19.85714            9 19399878
0.0039948705
## 3     10373.769     56.90000    13.20000           13 19399878
0.0049304434
## 4      9422.909     42.52727    11.72727           11 19399878
0.0047564732
## 5     10339.417     52.72500    18.00000           12 19399878
0.0054459105
## 6     10513.438     33.11875    13.37500           16 19399878
0.0062964313
## 7     13055.444     48.67222    17.69444           36 19399878
0.0236509219
## 8     12201.652     52.91739    17.34783           23 19399878
0.0092565015
## 9     15721.667     43.99048    21.19048           21 19399878
0.0136289517
## 10    11838.907     41.85581    15.65116           43 19399878
0.0232707649
## 11    21109.688     56.30625    23.43750           16 19399878
0.0097603707
## 12    17386.000     54.22000    20.60000            5 19399878
0.0013286166
## 13    21113.143     53.77143    20.28571            7 19399878
0.0021804261
## 14     6842.250     44.72500    13.83333           12 19399878
0.0041727067
## 15    16376.444     48.62222    16.44444            9 19399878
0.0028814614
## 16    13823.000     50.02500    14.37500            8 19399878
0.0025824905
## 17    17572.667     35.23333    24.33333            6 19399878
0.0015515561
## 18    32722.600     53.62667    20.66667           15 19399878
0.0070850961
## 19    22691.692     51.08462    22.07692           13 19399878
```

```
0.0076495326
## 20    20379.955     47.68636    19.72727          22 19399878
0.0120413128
## 21     8682.480     31.28000    23.16000          25 19399878
0.0132165780
## 22    14770.000     39.07879    18.84848          33 19399878
0.0194318232
## 23    53439.870     51.80000    19.30435          23 19399878
0.0097088239
## 24    15308.880     47.07200    19.28000          25 19399878
0.0128531736
## 25    21931.909     51.30000    20.15909          44 19399878
0.0208197186
## 26    12197.700     47.95000    18.73333          30 19399878
0.0154923655
## 27    19232.326     47.23783    24.15217          46 19399878
0.0255014490
## 28    16114.000     50.11373    22.49020          51 19399878
0.0247321143
## 29    14178.173     45.46346    17.88462          52 19399878
0.0254524796
## 30    12854.017     45.56983    20.20000          60 19399878
0.0303532837
## 31    19346.600     46.51354    21.37000         100 19399878
0.0564784480
## 32    14968.319     40.98194    20.59722          72 19399878
0.0429744971
## 33    14619.099     49.56901    22.52113          71 19399878
0.0423894418
## 34    16013.071     52.56012    21.53571          84 19399878
0.0453340480
## 35    17322.481     50.35962    21.59615         104 19399878
0.0608354341
## 36    12125.469     52.00617    22.76543          81 19399878
0.0490711849
## 37     9574.871     49.76471    19.22353          85 19399878
0.0411239700
## 38    11851.477     48.65780    20.70642         109 19399878
0.0545596215
## 39    11539.338     48.05809    20.86029         136 19399878
0.0749978943
## 40    13000.625     49.55781    21.00000         128 19399878
0.0705494127
## 41    12786.143     48.96329    22.15000         140 19399878
0.0811190668
## 42    15495.151     48.21075    21.82075         106 19399878
0.0629488495
## 43    11906.969     50.99808    21.63846         130 19399878
0.0703006998
## 44    12765.312     49.08125    20.88889         144 19399878
```

```
0.0823265487
## 45    12799.027     50.91600    22.30000          150 19399878
0.0850520813
## 46    13436.641     47.92519    21.04580          131 19399878
0.0757530537
## 47    12541.703     49.75508    20.10169          118 19399878
0.0689553305
## 48    12029.145     46.46447    22.20395          152 19399878
0.0882183383
## 49    12155.531     48.77771    21.03429          175 19399878
0.1028808016
## 50    11360.279     50.38837    20.46512          172 19399878
0.1028962656
## 51    12100.891     48.19705    21.12821          156 19399878
0.1013769262
## 52    13866.681     48.45942    21.99517          207 19399878
0.1255665113
## 53    13580.428     47.53144    22.02488          201 19399878
0.1269995100
## 54    15020.500     50.90236    22.55189          212 19399878
0.1265265689
## 55    14686.488     63.93791    20.64455          211 19399878
0.1547870559
## 56    14249.289     59.35923    20.19512          287 19399878
0.1963929876
## 57    14289.267     55.61977    20.94961          258 19399878
0.1864599355
## 58    14496.443     55.92857    21.21786          280 19399878
0.2011855435
## 59    13932.045     59.20822    21.53824          353 19399878
0.2447901992
## 60    14119.539     52.64040    22.46801          297 19399878
0.2072358393
## 61    14168.734     56.81698    21.95765          425 19399878
0.2900005351
## 62    13675.866     57.26115    23.22930          471 19399878
0.3119942301
## 63    14444.325     54.42710    23.34904          573 19399878
0.3824547247
## 64    13753.873     55.81878    21.50551          544 19399878
0.3541916604
## 65    14440.516     56.34841    21.43110          566 19399878
0.3916467928
## 66    15253.890     57.88508    23.58027          517 19399878
0.3743348283
## 67    20171.654     57.93832    25.40000          535 19399878
0.4302230664
## 68    17222.961     60.05560    24.33748          563 19399878
0.4503971623
## 69    20820.717     62.48569    23.81717          629 19399878
```

```
0.5060199348
## 70    18974.888     57.88865    24.02484          644 19399878
0.4922698483
## 71    17336.384     57.55323    24.53873          852 19399878
0.6628005599
## 72    16339.336     56.87367    22.83818          791 19399878
0.5580550558
## 73    15353.219     56.64959    23.18086          857 19399878
0.6065978869
## 74    16545.752     57.12099    23.10865          948 19399878
0.6792748903
## 75    16395.990     55.50919    23.25066         1145 19399878
0.8202190756
## 76    15594.172     56.10630    23.58824         1207 19399878
0.8752400402
## 77    16744.030     56.93871    24.56726         1130 19399878
0.8274420592
## 78    15576.682     55.38668    24.81858         1141 19399878
0.8400439425
## 79    15813.636     55.62993    24.43891         1244 19399878
0.9416476743
## 80    15868.187     54.67319    25.45505         1268 19399878
0.9526683621
## 81    16295.035     56.72173    25.49769         1298 19399878
1.0050836402
## 82    16647.052     56.46549    24.45740         1244 19399878
0.9567199340
## 83    16579.731     54.87123    24.90154         1493 19399878
1.1240624812
## 84    15159.180     52.52115    24.65281         1708 19399878
1.2755505473
## 85    15488.070     54.19450    24.58470         1582 19399878
1.1974585613
## 86    16142.533     54.17109    24.82146         2302 19399878
1.6992967688
## 87    17027.800     53.61822    24.78288         1764 19399878
1.3551051713
## 88    17293.090     53.17876    25.06404          890 19399878
0.6770454948
## 89    17255.642     54.89933    24.94583         3286 19399878
2.4789382696
## 90    18313.807     54.98263    24.62327         2166 19399878
1.6531856541
## 91    17904.992     54.08525    24.06074          922 19399878
0.7400523343
## 92    18532.916     54.55393    24.94358         2694 19399878
2.0752527413
## 93    18670.729     54.49640    24.37416         1780 19399878
1.4126609456
## 94    18638.162     55.95617    24.65707         2085 19399878
```

```
1.6227486070
## 95    17874.471     54.18077    24.73233         2843 19399878
2.2050976300
## 96    17192.701     53.62905    24.13567         2506 19399878
1.9200623324
## 97    16835.044     52.04476    24.20793         2270 19399878
1.7583396658
## 98    16922.011     52.77621    24.21469         3675 19399878
2.8299675390
## 99    17970.317     53.35524    23.57439         2944 19399878
2.2845581812
## 100   19877.807     52.82146    24.05602         2392 19399878
1.9232852908
## 101   19550.813     53.24194    24.53355         4009 19399878
3.1064022671
## 102   18172.541     51.90567    23.90434         3021 19399878
2.3224372854
## 103   18437.890     52.49545    23.92883         3625 19399878
2.7690715375
## 104   20359.024     52.62373    24.59238         2571 19399878
2.1137916434
## 105   20111.837     52.57752    24.03854         3036 19399878
2.4213399693
## 106   18785.838     51.15755    24.00863         4865 19399878
3.8656750831
## 107   17674.574     49.76405    23.67379         3041 19399878
2.3107555109
## 108   16920.634     49.45329    23.24523         2202 19399878
1.6644859828
## 109   16152.144     48.43469    23.23739         2835 19399878
2.1322801618
## 110   15943.415     49.00862    22.81793         2867 19399878
2.0641212280
## 111   15998.030     50.87904    22.57216         3305 19399878
2.4211312051
## 112   15598.421     48.83113    22.82830         2516 19399878
1.7832560597
## 113   15946.261     46.91704    23.49233         2608 19399878
1.8716148627
## 114   17336.186     50.95269    22.77510         2699 19399878
1.9992793254
## 115   16737.804     49.10753    22.43818         2960 19399878
2.0727488596
## 116   17746.160     48.95094    23.29946         2598 19399878
2.0089894380
## 117   17081.053     49.55925    22.94164         2262 19399878
1.7482725922
## 118   17678.786     49.43762    22.88106         3178 19399878
2.4485192639
##       NYCPI NYUnemployment NYCondoPriceIdx NYSnapBenefits
```

```
## 1   659.861          4.5          228.29        1801707
## 2   660.931          4.6          228.16        1792916
## 3   660.060          4.7          227.16        1816805
## 4   660.006          4.7          226.14        1823494
## 5   660.713          4.8          225.96        1825759
## 6   663.464          4.8          226.88        1830858
## 7   663.150          4.8          226.76        1849851
## 8   664.520          4.8          227.19        1932022
## 9   667.848          4.9          229.21        1927903
## 10  673.924          4.9          231.26        1950582
## 11  675.948          5.0          228.59        1968193
## 12  682.680          5.1          226.87        1986156
## 13  689.702          5.2          225.55        2004511
## 14  694.595          5.4          224.47        2030668
## 15  695.396          5.5          223.54        2051611
## 16  694.064          5.7          221.65        2077774
## 17  689.190          6.0          219.96        2114221
## 18  677.900          6.3          218.30        2137106
## 19  673.604          6.7          215.51        2174325
## 20  674.732          7.1          213.60        2211935
## 21  678.378          7.5          211.56        2246664
## 22  679.546          7.8          208.41        2295103
## 23  681.035          8.1          204.02        2339118
## 24  682.171          8.3          201.63        2384027
## 25  685.631          8.4          199.02        2427841
## 26  686.869          8.5          197.13        2478604
## 27  688.841          8.7          196.38        2508884
## 28  689.668          8.8          196.04        2555081
## 29  689.123          8.8          197.12        2599938
## 30  690.272          8.9          197.42        2623264
## 31  689.261          8.9          198.99        2673143
## 32  690.828          8.9          199.47        2699586
## 33  690.517          8.8          198.91        2712437
## 34  694.099          8.8          199.69        2754632
## 35  695.337          8.7          200.61        2775875
## 36  696.916          8.6          202.38        2799734
## 37  696.168          8.5          199.68        2824845
## 38  697.123          8.5          198.98        2860394
## 39  698.342          8.5          198.00        2874189
## 40  698.099          8.5          199.70        2895995
## 41  699.532          8.5          198.58        2918849
## 42  699.473          8.4          199.31        2934493
## 43  699.225          8.4          195.94        2969868
## 44  701.436          8.3          194.84        2971876
## 45  704.884          8.2          195.52        2975444
## 46  710.044          8.1          195.90        3013945
## 47  712.565          8.1          197.43        3017404
## 48  717.146          8.1          195.10        3019981
## 49  718.394          8.2          197.07        3035825
## 50  720.299          8.3          197.17        3043751
```

```
## 51  722.882             8.3          198.61        3040684
## 52  724.331             8.4          197.92        3057767
## 53  722.862             8.5          197.09        3060107
## 54  720.740             8.5          195.88        3046972
## 55  717.820             8.6          194.64        3068575
## 56  720.754             8.6          193.72        3059120
## 57  723.540             8.6          192.37        3059262
## 58  728.171             8.7          194.06        3081831
## 59  729.507             8.7          194.18        3063238
## 60  730.381             8.7          198.36        3082995
## 61  729.670             8.7          201.56        3095534
## 62  728.545             8.6          203.58        3094677
## 63  732.751             8.5          204.55        3109436
## 64  735.879             8.4          203.99        3101190
## 65  735.080             8.3          205.85        3110070
## 66  735.102             8.2          207.49        3152122
## 67  732.992             8.2          209.80        3186236
## 68  736.613             8.1          210.80        3158541
## 69  740.736             8.1          212.47        3153979
## 70  741.764             8.0          213.11        3182976
## 71  739.965             7.9          214.64        3181218
## 72  740.840             7.8          215.84        3183287
## 73  742.694             7.8          215.44        3186788
## 74  743.894             7.7          216.61        3194470
## 75  744.855             7.7          218.43        3186530
## 76  747.300             7.6          224.95        3169363
## 77  743.150             7.4          225.88        3170323
## 78  744.042             7.3          229.03        3156551
## 79  743.771             7.1          226.34        3158376
## 80  750.456             7.0          230.80        3148532
## 81  748.789             6.8          232.78        3114414
## 82  751.541             6.7          234.22        3109524
## 83  751.579             6.6          234.49        3103477
## 84  755.164             6.5          235.72        3101888
## 85  755.527             6.4          236.64        3094747
## 86  755.953             6.2          238.52        3088298
## 87  754.731             6.1          238.52        3075125
## 88  754.728             6.0          239.56        3066686
## 89  753.070             6.0          238.73        3068825
## 90  749.838             5.9          240.16        3057644
## 91  746.075             5.8          241.62        3075720
## 92  746.929             5.7          244.12        3055942
## 93  749.427             5.7          245.12        3045194
## 94  750.602             5.6          245.93        3050058
## 95  751.506             5.5          248.14        3039251
## 96  754.705             5.4          250.48        3027230
## 97  755.996             5.3          252.93        3028373
## 98  755.091             5.2          254.15        3017604
## 99  755.517             5.0          255.54        3001608
## 100 757.080             5.0          256.57        3001849
```

```
## 101 756.003            4.9          256.58        2996649
## 102 754.540            4.9          257.21        2982398
## 103 751.453            4.9          256.84        2990471
## 104 752.612            4.9          258.53        2975036
## 105 754.153            4.8          259.80        2972012
## 106 755.983            4.8          260.62        2972806
## 107 759.194            4.8          257.23        2961955
## 108 761.198            4.8          259.21        2965167
## 109 762.832            4.9          260.65        2953595
## 110 762.383            4.9          261.88        2941315
## 111 763.651            4.9          262.28        2957116
## 112 764.929            5.0          261.12        2950208
## 113 765.320            4.9          262.01        2938258
## 114 766.664            4.9          262.45        2940107
## 115 767.295            4.8          264.57        2949168
## 116 771.621            4.7          265.52        2944348
## 117 773.774            4.7          266.55        2922436
## 118 773.542            4.7          267.15        2927021

#dates2 <- as.Date(nyEcon$date, "%Y-%d-%m")

#nyEcon$date <- as.Date(nyEcon$date, "%d/%m/%y")
#types(nyEcon$date)
#dates5

#tsLCOrg

tsLC2 <- as_tsibble(tsLC1, index=date, key = state)

tsLC2

## # A tsibble: 118 x 22 [1D]
## # Key:       state [1]
##    date       state avgLoans totalLoans avgTerm avgIntRate avgGrade
avgEmpLength
##    <date>     <chr>    <dbl>      <dbl>   <dbl>      <dbl>    <dbl>
<dbl>
##  1 2007-06-01 NY       3381.      13525      36       8.78     1.75
2.25
##  2 2007-07-01 NY       8611.      77500      36      11.0      3.22
2.78
##  3 2007-08-01 NY       7358.      95650      36      11.2      3.31
1.23
##  4 2007-09-01 NY       8389.      92275      36      11.3      3.45
3
##  5 2007-10-01 NY       8804.     105650      36      12.9      4.17
2.33
##  6 2007-11-01 NY       7634.     122150      36      11.5      3.31
2.56
##  7 2007-12-01 NY      12745.     458825      36      12.0      3.69
```

```
4.28
##  8 2008-01-01 NY       7808.     179575      36      11.9      3.35
3.52
##  9 2008-02-01 NY      12590.     264400      36      11.9      3.14
4.57
## 10 2008-03-01 NY      10499.     451450      36      11.8      3.14
3.33
## # ... with 108 more rows, and 14 more variables: avgAnnualInc <dbl>,
## #   avgVerifStatus <dbl>, avgHomeOwner <dbl>, avgOpenAcc <dbl>,
## #   avgRevolBal <dbl>, avgRevolUtil <dbl>, avgTotalAcc <dbl>,
## #   countOfLoans <dbl>, totalPop <dbl>, loansPerCapita <dbl>, NYCPI <dbl>,
## #   NYUnemployment <dbl>, NYCondoPriceIdx <dbl>, NYSnapBenefits <dbl>
```

```
tsLC2%>%group_by(state)%>%tally()
```

```
## # A tsibble: 118 x 3 [1D]
## # Key:       state [1]
##    state date           n
##    <chr> <date>     <int>
##  1 NY    2007-06-01     1
##  2 NY    2007-07-01     1
##  3 NY    2007-08-01     1
##  4 NY    2007-09-01     1
##  5 NY    2007-10-01     1
##  6 NY    2007-11-01     1
##  7 NY    2007-12-01     1
##  8 NY    2008-01-01     1
##  9 NY    2008-02-01     1
## 10 NY    2008-03-01     1
## # ... with 108 more rows
```

Q2A)

```
tsLCOrg
```

```
## # A tibble: 4,943 x 18
## # Groups:   state [51]
##    state date       avgLoans totalLoans avgTerm avgIntRate avgGrade
avgEmpLength
##    <chr> <date>        <dbl>      <dbl>   <dbl>      <dbl>    <dbl>
<dbl>
##  1 AK    2008-01-01    5600        5600      36      18.0       7
5
##  2 AK    2008-03-01   11700       23400      36      11.8       3
3.5
##  3 AK    2008-06-01    7500        7500      36      13.9       4
3
##  4 AK    2008-12-01   25000       25000      36      15.2       5
1
##  5 AK    2009-01-01   15000       30000      36      12.5       2.5
7
```

```
##  6 AK    2009-03-01   14662.      29325      36      13         3
7
##  7 AK    2009-04-01   20000       20000      36      11.9       2
5
##  8 AK    2009-05-01   16000       16000      36      12.2       2
2
##  9 AK    2009-07-01    1000        1000      36      11.9       2
10
## 10 AK    2009-11-01   11000       11000      36       8.94      1
7
## # ... with 4,933 more rows, and 10 more variables: avgAnnualInc <dbl>,
## #   avgVerifStatus <dbl>, avgHomeOwner <dbl>, avgOpenAcc <dbl>,
## #   avgRevolBal <dbl>, avgRevolUtil <dbl>, avgTotalAcc <dbl>,
## #   countOfLoans <dbl>, totalPop <dbl>, loansPerCapita <dbl>
```

```
low10<-quantile(tsLCOrg$totalPop, probs = c(0.10))
high10<-quantile(tsLCOrg$totalPop, probs = c(0.90))
low10
```

```
##    10%
## 816166
```

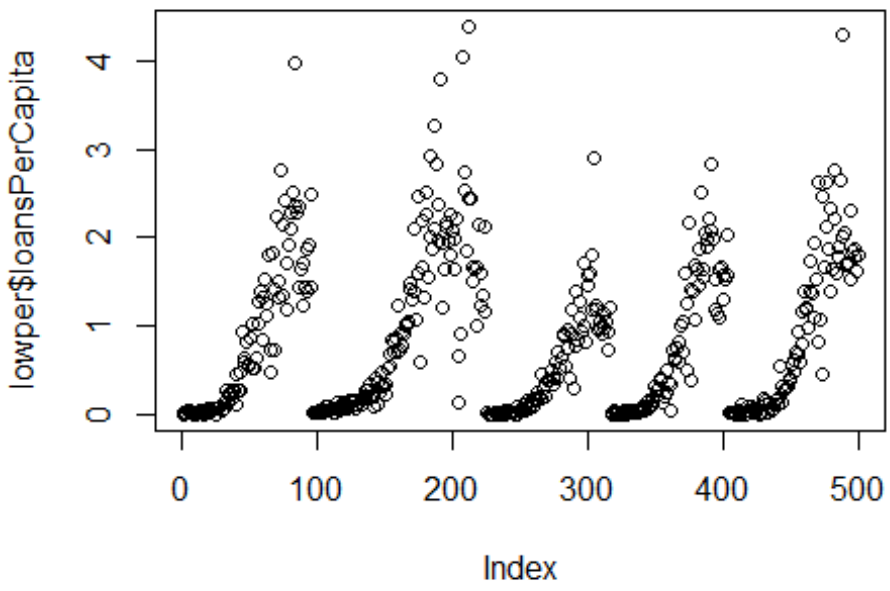
```
high10
```

```
##      90%
## 12840503
```

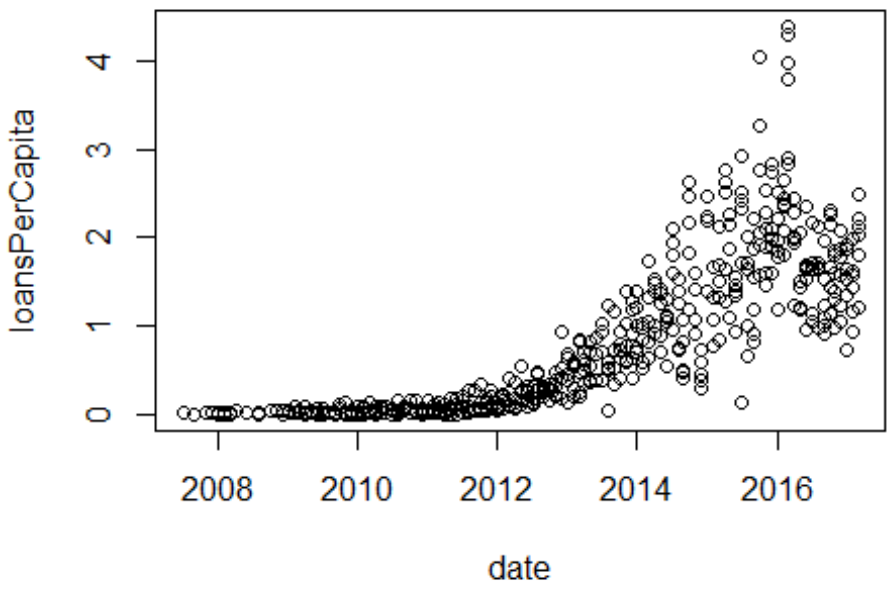
```
lowper<-tsLCOrg%>%filter(totalPop<=low10)

highper<-tsLCOrg%>%filter(totalPop>=high10)

#Graph1 <-
 #  lowper%>%
#  ggplot(aes(x=state, y=totalPop)) +geom_line(color="red")+geom_point()
month <- as_tibble(yearmonth(seq(as.Date("2007-06-01"), as.Date("2017-03-
01"), by = "1 month"))) %>%
  rename(month=value)
```

```
## Warning: Calling `as_tibble()` on a vector is discouraged, because the
behavior is likely to change in the future. Use `tibble::enframe(name =
NULL)` instead.
## This warning is displayed once per session.
```

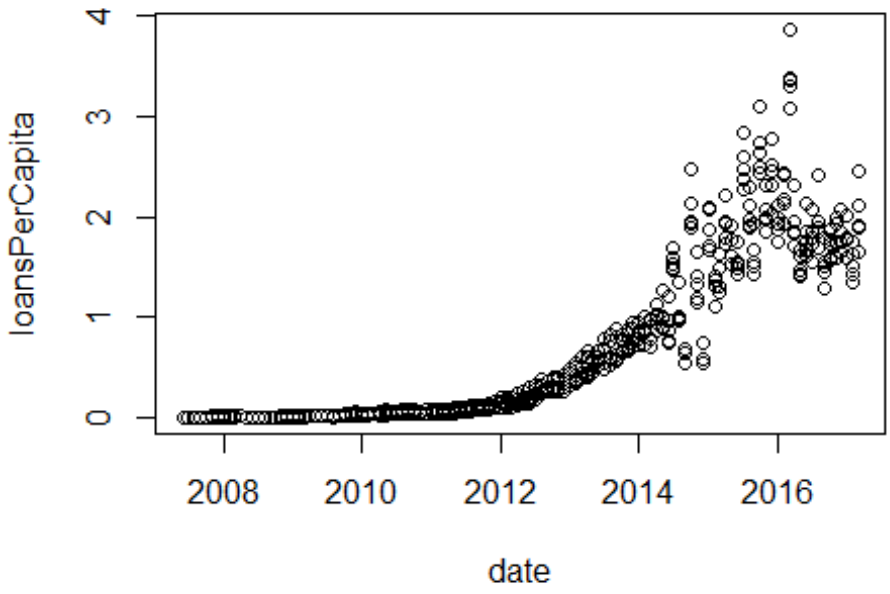
```
plot(lowper$loansPerCapita)
```




```
lowper1<-lowper%>%select(2,18)

## Adding missing grouping variables: `state`

lowper1<-lowper1[-c(1)]
#Graph1
plot(lowper1%>%select(date,loansPerCapita))
```




```
highper1<-highper%>%select(2,18)

## Adding missing grouping variables: `state`

highper1<-highper1[-c(1)]


plot(highper1%>%select(date,loansPerCapita))
```



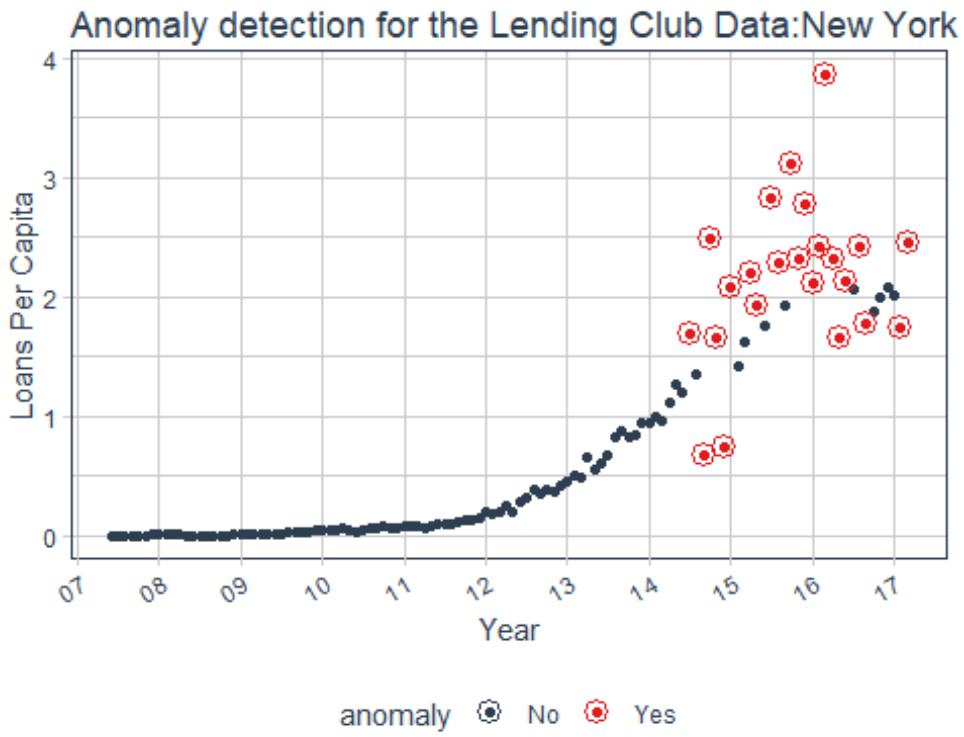
Q2)B)

```
library(tibbletime)

## Warning: package 'tibbletime' was built under R version 3.6.3

##
## Attaching package: 'tibbletime'

## The following object is masked from 'package:stats':
##
##     filter

NY<-tsLC2 %>%select(1,18)

library(anomalize)

## Warning: package 'anomalize' was built under R version 3.6.3

## == Use anomalize to improve your Forecasts by 50%!
==============================
## Business Science offers a 1-hour course - Lab #18: Time Series Anomaly
Detection!
## </> Learn more at: https://university.business-science.io/p/learning-labs-
pro </>

anomalyNY <-
  NY%>%
```

```
  time_decompose(loansPerCapita, method = "stl") %>%
  anomalize(remainder, method = "iqr") %>%
  plot_anomalies() +
  labs(title = "Anomaly detection for the Lending Club Data:New York") +
  xlab("Year") + ylab("Loans Per Capita") +
  scale_x_date(date_breaks = "years" , date_labels = "%y")
```

```
## Converting from tbl_ts to tbl_time.
## Auto-index message: index = date

## frequency = 12 months

## trend = 31 months

## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
#tsLC2
#23
anomalyNY
```




```
anomalyCO <-
  tsLCOrg%>%
  select(2,18)%>%
  filter(state=="CO")%>%
  time_decompose(loansPerCapita, method = "stl") %>%
  anomalize(remainder, method = "iqr") %>%
```

```
  plot_anomalies() +
  labs(title = "Anomaly detection for the Lending Club Data:Colorado") +
  xlab("Year") + ylab("Loans Per Capita") +
  scale_x_date(date_breaks = "years" , date_labels = "%y")
```

```
## Adding missing grouping variables: `state`
```

```
#tsLC2
#21
tsLCOrg%>%filter(state=="NY")
```

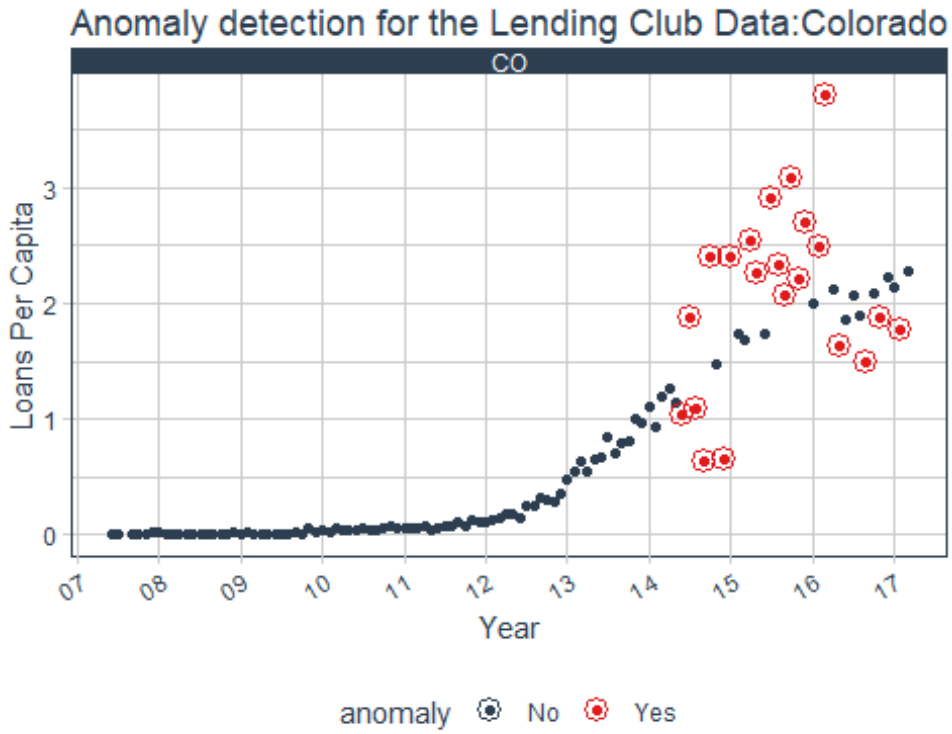
```
## # A tibble: 118 x 18
## # Groups:   state [1]
##    state date       avgLoans totalLoans avgTerm avgIntRate avgGrade
avgEmpLength
##    <chr> <date>        <dbl>      <dbl>   <dbl>      <dbl>    <dbl>
<dbl>
##  1 NY    2007-06-01    3381.      13525      36       8.78     1.75
2.25
##  2 NY    2007-07-01    8611.      77500      36      11.0      3.22
2.78
##  3 NY    2007-08-01    7358.      95650      36      11.2      3.31
1.23
##  4 NY    2007-09-01    8389.      92275      36      11.3      3.45
3
##  5 NY    2007-10-01    8804.     105650      36      12.9      4.17
2.33
##  6 NY    2007-11-01    7634.     122150      36      11.5      3.31
2.56
##  7 NY    2007-12-01   12745.     458825      36      12.0      3.69
4.28
##  8 NY    2008-01-01    7808.     179575      36      11.9      3.35
3.52
##  9 NY    2008-02-01   12590.     264400      36      11.9      3.14
4.57
## 10 NY    2008-03-01   10499.     451450      36      11.8      3.14
3.33
## # ... with 108 more rows, and 10 more variables: avgAnnualInc <dbl>,
## #   avgVerifStatus <dbl>, avgHomeOwner <dbl>, avgOpenAcc <dbl>,
## #   avgRevolBal <dbl>, avgRevolUtil <dbl>, avgTotalAcc <dbl>,
## #   countOfLoans <dbl>, totalPop <dbl>, loansPerCapita <dbl>
```

```
tsLCOrg%>%filter(state=="CO")
```

```
## # A tibble: 117 x 18
## # Groups:   state [1]
##    state date       avgLoans totalLoans avgTerm avgIntRate avgGrade
avgEmpLength
##    <chr> <date>        <dbl>      <dbl>   <dbl>      <dbl>    <dbl>
<dbl>
##  1 CO    2007-06-01    2600        2600      36       8.38     1
```

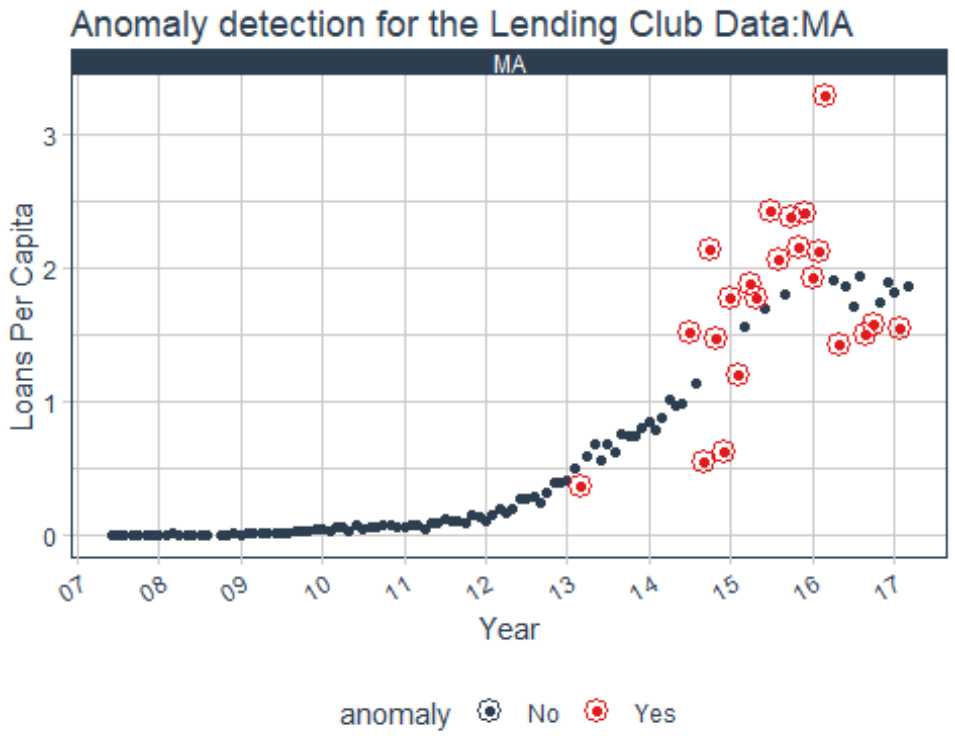
```
3
##  2 CO    2007-07-01    5833.      17500      36       9.02     2
3.33
##  3 CO    2007-09-01   13150       13150      36      17.5      7
7
##  4 CO    2007-10-01    5000       10000      36      11.7      3.5
1
##  5 CO    2007-11-01    5792.      17375      36      13.9      5
1.67
##  6 CO    2007-12-01    9511.      85600      36      13.2      4.44
2.22
##  7 CO    2008-01-01    7888.     102550      36      12.0      3.54
2.69
##  8 CO    2008-02-01    7358.      73575      36      10.9      2.7
3
##  9 CO    2008-03-01    8162.      65300      36      11.7      3.12
7.12
## 10 CO    2008-04-01    6050       24200      36      10.5      2.5
4.75
## # ... with 107 more rows, and 10 more variables: avgAnnualInc <dbl>,
## #   avgVerifStatus <dbl>, avgHomeOwner <dbl>, avgOpenAcc <dbl>,
## #   avgRevolBal <dbl>, avgRevolUtil <dbl>, avgTotalAcc <dbl>,
## #   countOfLoans <dbl>, totalPop <dbl>, loansPerCapita <dbl>

tsLCOrg%>%filter  (state=="MA")#%>%select(totalPop)

## # A tibble: 117 x 18
## # Groups:   state [1]
##    state date       avgLoans totalLoans avgTerm avgIntRate avgGrade
avgEmpLength
##    <chr> <date>        <dbl>      <dbl>   <dbl>      <dbl>    <dbl>
<dbl>
##  1 MA    2007-06-01    3194.      12775      36       11.3     3.25
1
##  2 MA    2007-07-01    3790       37900      36       10.1     2.5
3.2
##  3 MA    2007-08-01    6004.      36025      36       10.6     2.83
1
##  4 MA    2007-09-01    7750       46500      36       11.1     3
5.67
##  5 MA    2007-10-01    7046.      49325      36       10.9     2.86
1.43
##  6 MA    2007-11-01    8680       86800      36       11.1     3
2.6
##  7 MA    2007-12-01    7783.      70050      36       11.0     3.11
2.89
##  8 MA    2008-01-01    8036.      56250      36       10.9     2.86
4.43
##  9 MA    2008-02-01    8184.      65475      36       13.2     4
5.62
```

```
## 10 MA    2008-03-01    9750      107250      36       11.7     3
4.27
## # ... with 107 more rows, and 10 more variables: avgAnnualInc <dbl>,
## #   avgVerifStatus <dbl>, avgHomeOwner <dbl>, avgOpenAcc <dbl>,
## #   avgRevolBal <dbl>, avgRevolUtil <dbl>, avgTotalAcc <dbl>,
## #   countOfLoans <dbl>, totalPop <dbl>, loansPerCapita <dbl>

#CO 5047349
#NY 19399878
#MA 6566307

anomalyCO
```




```
anomalyMA <-
  tsLCOrg%>%
  select(2,18)%>%
  filter(state=="MA")%>%
  time_decompose(loansPerCapita, method = "stl") %>%
  anomalize(remainder, method = "iqr") %>%
  plot_anomalies() +
  labs(title = "Anomaly detection for the Lending Club Data:MA") +
  xlab("Year") + ylab("Loans Per Capita") +
  scale_x_date(date_breaks = "years" , date_labels = "%y")

## Adding missing grouping variables: `state`
```

```
#tsLC2
#21
anomalyMA
```

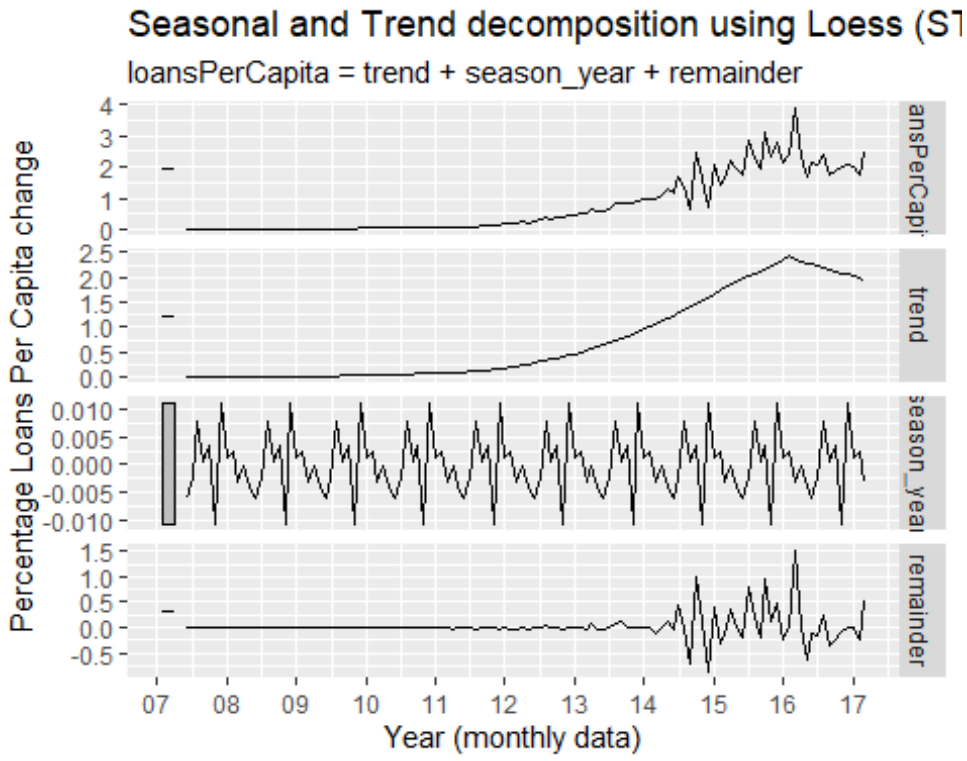



```
tsLCOrg

## # A tibble: 4,943 x 18
## # Groups:   state [51]
##    state date       avgLoans totalLoans avgTerm avgIntRate avgGrade
avgEmpLength
##    <chr> <date>        <dbl>      <dbl>   <dbl>      <dbl>    <dbl>
<dbl>
##  1 AK    2008-01-01    5600        5600      36       18.0        7
5
##  2 AK    2008-03-01   11700       23400      36       11.8        3
3.5
##  3 AK    2008-06-01    7500        7500      36       13.9        4
3
##  4 AK    2008-12-01   25000       25000      36       15.2        5
1
##  5 AK    2009-01-01   15000       30000      36       12.5        2.5
7
##  6 AK    2009-03-01   14662.      29325      36       13          3
7
##  7 AK    2009-04-01   20000       20000      36       11.9        2
5
##  8 AK    2009-05-01   16000       16000      36       12.2        2
```

```
2
##  9 AK    2009-07-01    1000        1000      36      11.9       2
10
## 10 AK    2009-11-01   11000       11000      36       8.94      1
7
## # ... with 4,933 more rows, and 10 more variables: avgAnnualInc <dbl>,
## #   avgVerifStatus <dbl>, avgHomeOwner <dbl>, avgOpenAcc <dbl>,
## #   avgRevolBal <dbl>, avgRevolUtil <dbl>, avgTotalAcc <dbl>,
## #   countOfLoans <dbl>, totalPop <dbl>, loansPerCapita <dbl>
```

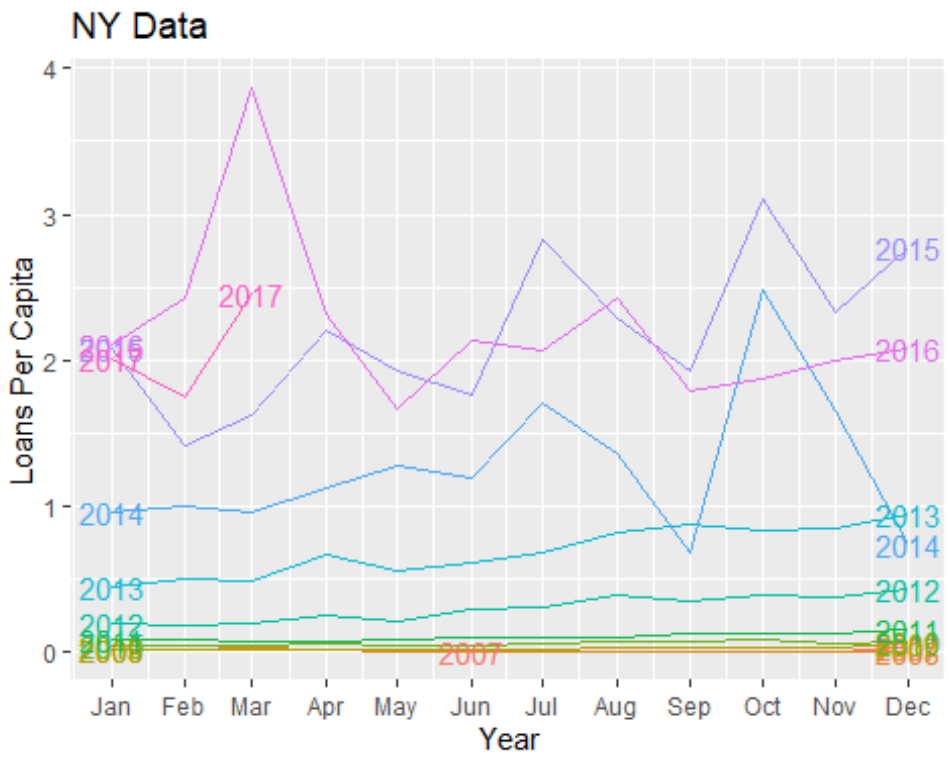
Q2)C)

```
month <- as_tibble(yearmonth(seq(as.Date("2007-06-01"), as.Date("2017-03-
01"), by = "1 month"))) %>%
  rename(month=value)



NY<-NY%>% bind_cols(month) %>%
  #select(month, loansPerCapita = value) %>%
  as_tsibble(index = month)
NY<-NY%>%select(c(2,3))

#tsLCOrg<-tsLCOrg%>% bind_cols(month) %>%
  #select(month, loansPerCapita = value) %>%
#  as_tsibble(index = month)

plotNYDecomposed <-
  NY %>%
  model(STL(loansPerCapita ~ trend() + season(window='periodic'), robust =
TRUE)) %>%
  components() %>%
  autoplot() +
  xlab("Year (monthly data)") + ylab(" Percentage Loans Per Capita change") +
  ggtitle("Seasonal and Trend decomposition using Loess (STL) for NY data") +
  scale_x_date(date_breaks = "years" , date_labels = "%y")
ggplotly(plotNYDecomposed)

plotNYDecomposed
```

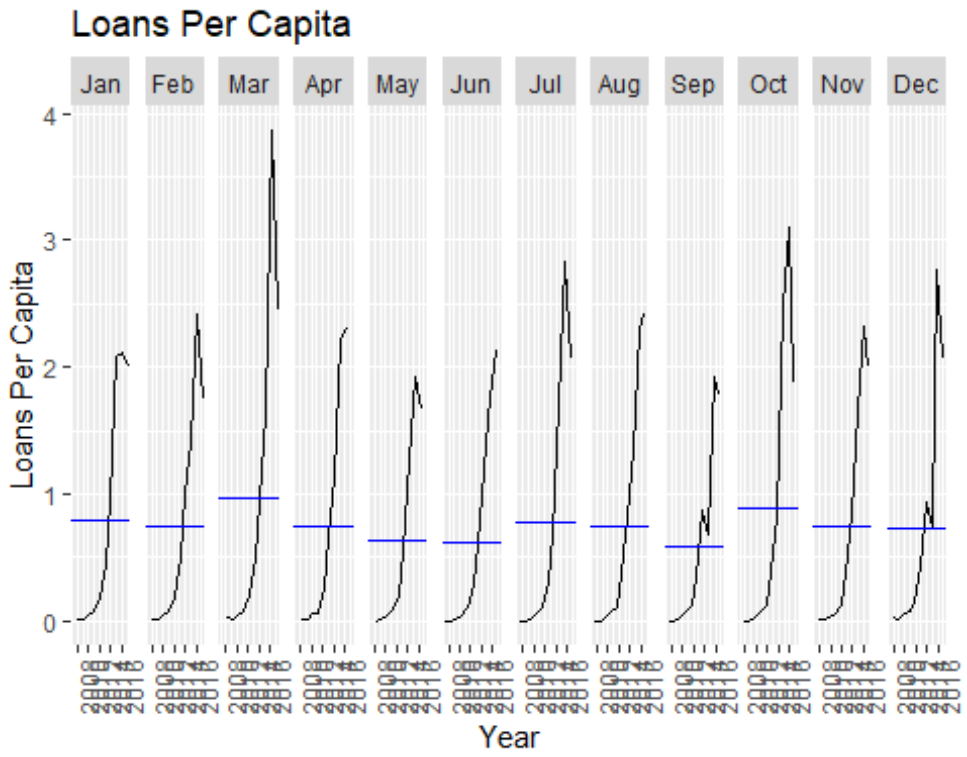


Q2)D)

```
plotSeason <-
  NY %>%
  gg_season(loansPerCapita, labels = "both") +
  xlab("Year") + ylab("Loans Per Capita") +
  ggtitle("NY Data")
plotSeason
```

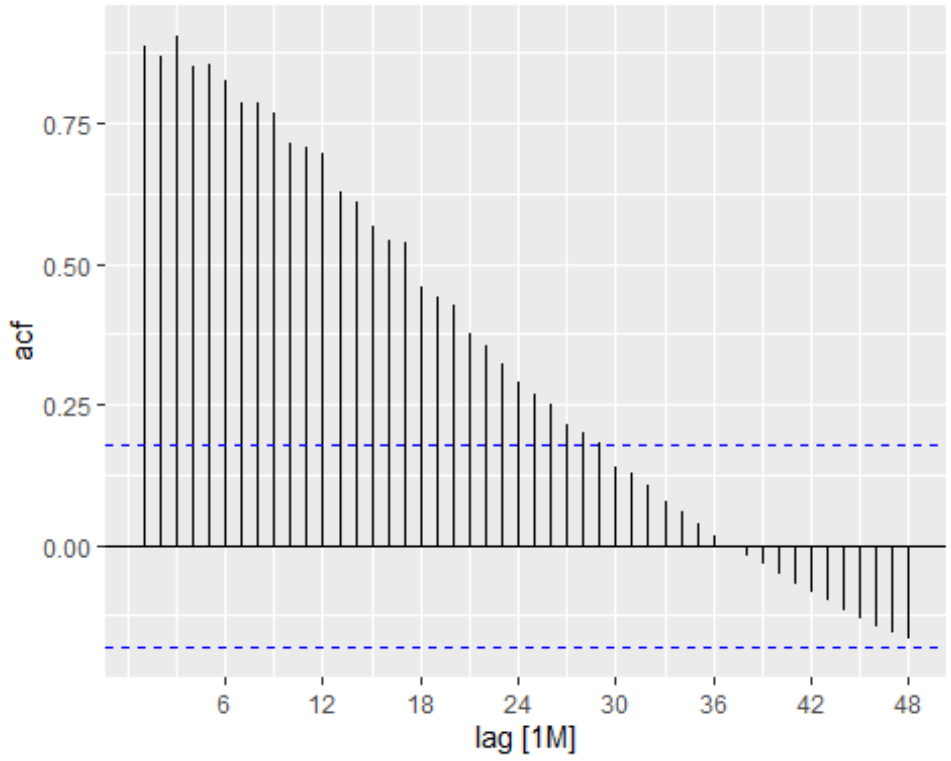



```
#ggplotly(plotSeason)

plotSubseries <-
  NY %>%
  gg_subseries(loansPerCapita, labels = "both") +
  xlab("Year") + ylab("Loans Per Capita") +
  ggtitle("Loans Per Capita")

## Warning: Ignoring unknown parameters: labels

plotSubseries
```

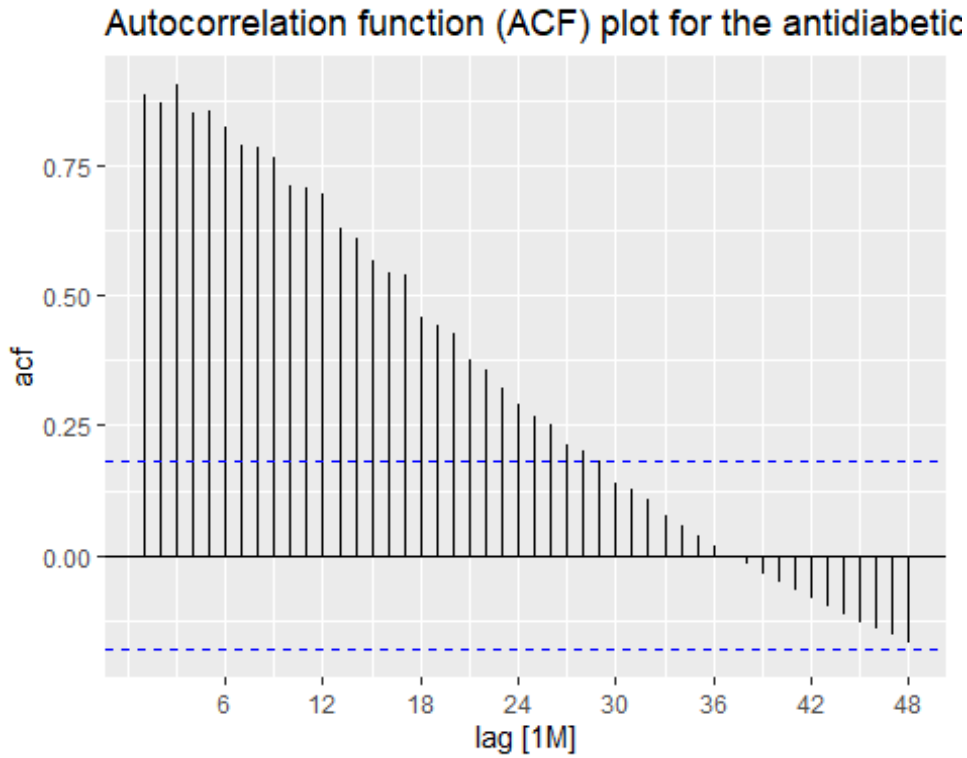


Q2)E)

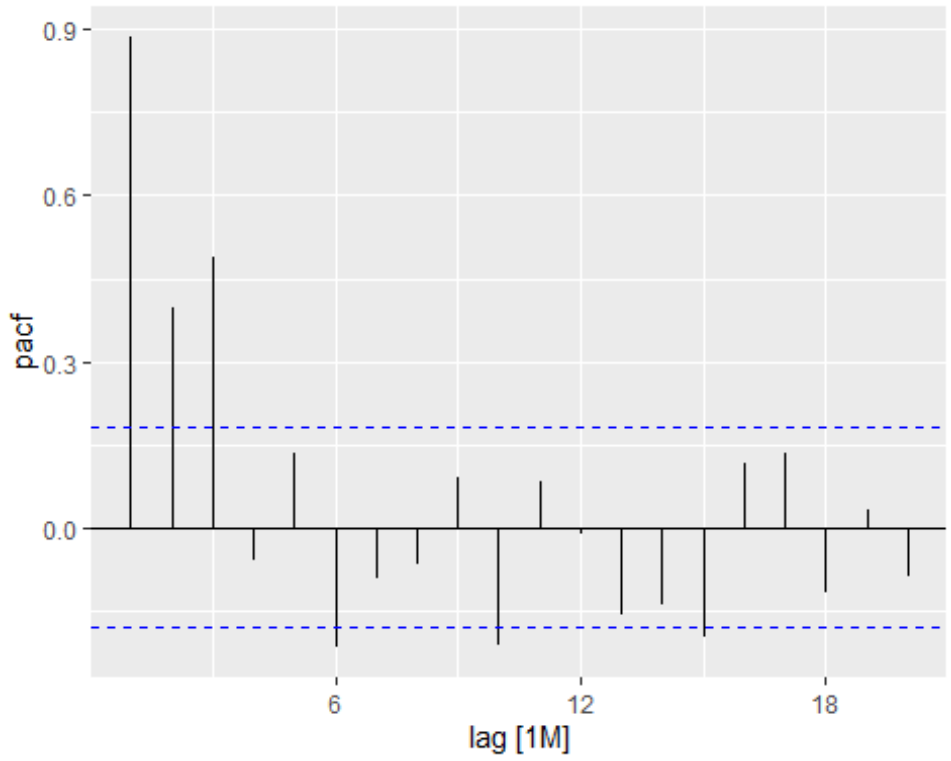
```
NY %>% ACF(loansPerCapita, lag_max = 48) %>% autoplot()
```



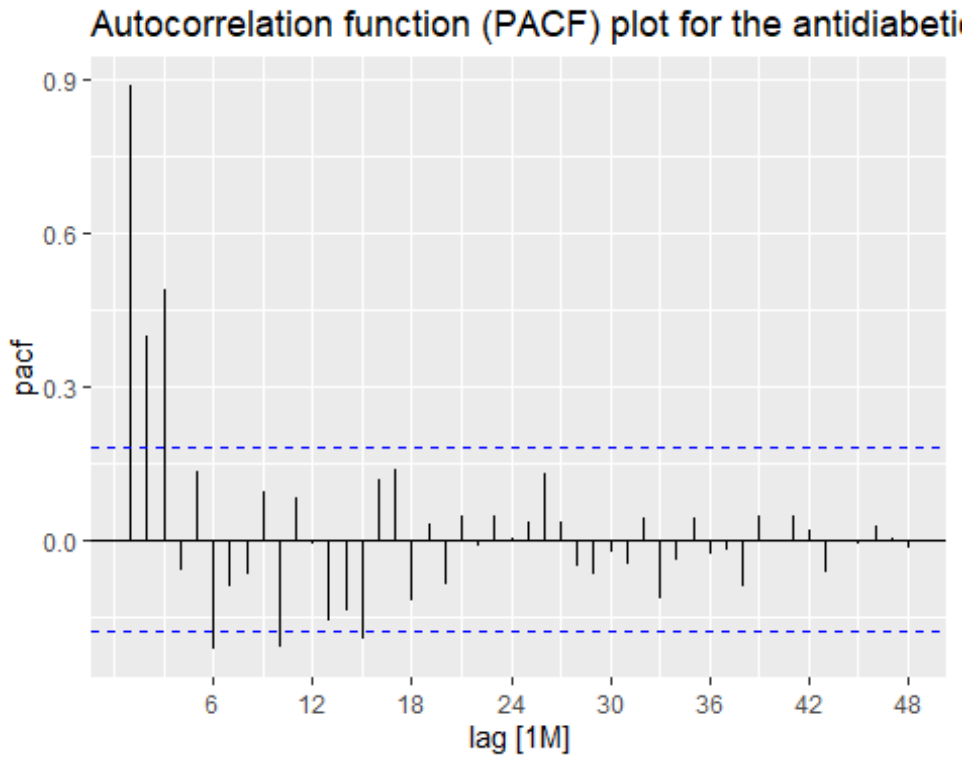
```
plotACF <-NY %>%
  ACF(loansPerCapita, lag_max = 48)  %>%
  autoplot() +  ggtitle("Autocorrelation function (ACF) plot for the
antidiabetic drug sales data")
plotACF
```




```
NY %>% PACF(loansPerCapita) %>% autoplot()
```




```
plotPACF <-NY %>%
  PACF(loansPerCapita, lag_max = 48)  %>%
  autoplot() +  ggtitle("Autocorrelation function (PACF) plot for the
antidiabetic drug sales data")
plotPACF
```

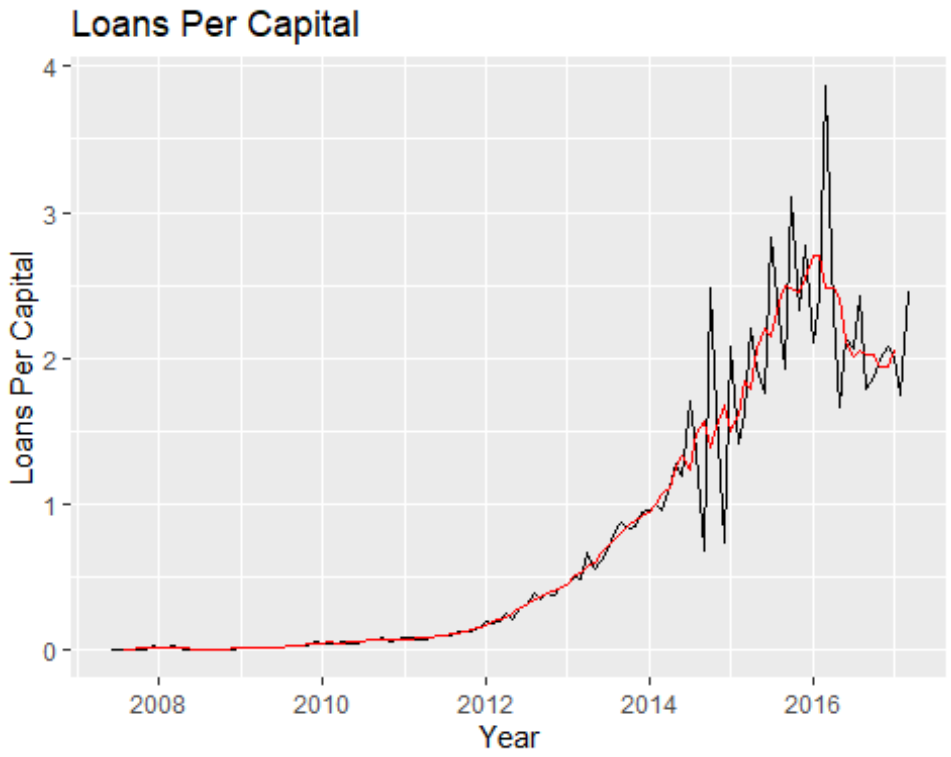


Q2)F)

```
plotAccLag <-
  NY %>%
  gg_lag(loansPerCapita, geom='point',lags = 1:25) +
  xlab(NULL) + ylab(NULL) +
  ggtitle("Lag plots for the accidental deaths data")
ggplotly(plotAccLag)
```

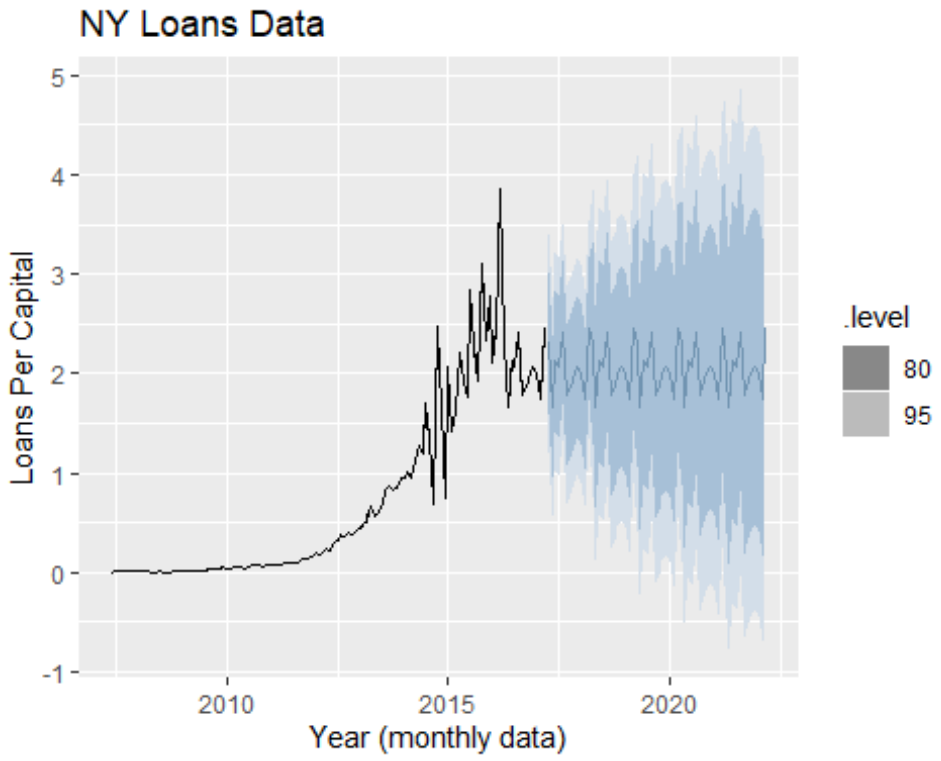
Q2)G)

```
NY5 <- NY%>%
  mutate(
    `5-MA` = slide_dbl(loansPerCapita, mean, .size = 5, .align = "center")
  )

NY5 %>%
  autoplot(loansPerCapita) +
  autolayer(NY5, `5-MA`, color='red') +
  xlab("Year") + ylab("Loans Per Capital") +
  ggtitle("Loans Per Capital") +
  guides(colour=guide_legend(title="series"))

## Warning: Removed 4 rows containing missing values (geom_path).
```

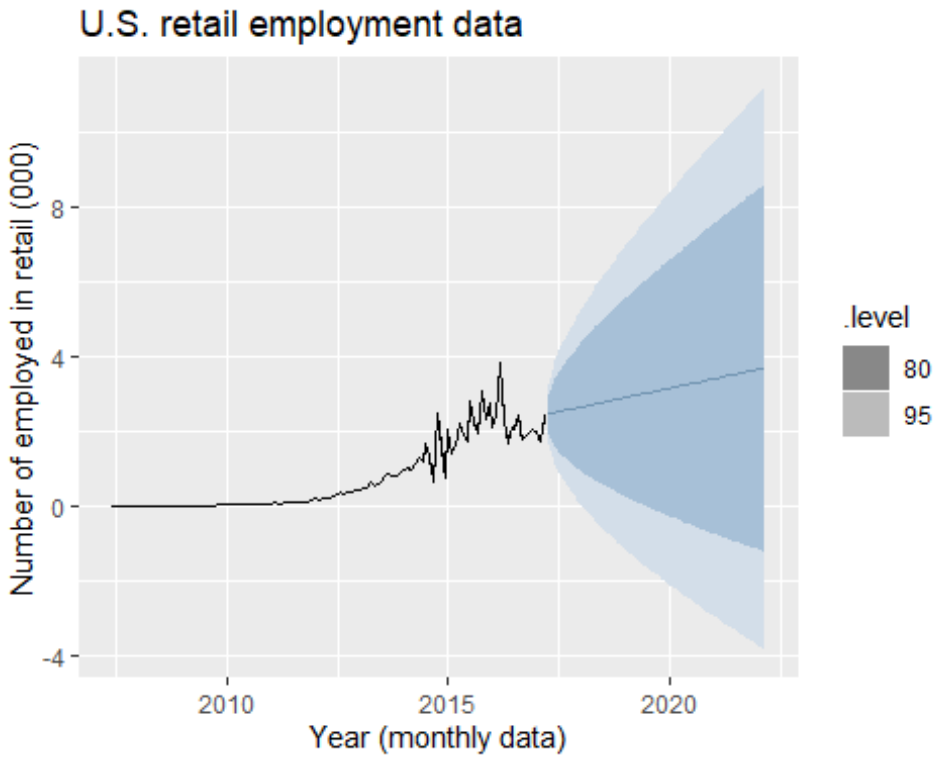


Q3a)

```
plotNYSNaive <-
  NY %>%
  model(SNAIVE(loansPerCapita)) %>%
  forecast(h = "5 years") %>%
  autoplot(NY, colour = "#769ECB") + #level = NULL,
  geom_line(linetype = 'dashed', colour = '#000000') +
  xlab("Year (monthly data)") + ylab("Loans Per Capital") +
  ggtitle("NY Loans Data")
plotNYSNaive
```




```
plotNYDrift <-
  NY %>%
  model(RW(loansPerCapita ~ drift())) %>%
  forecast(h = "5 years") %>%
  autoplot(NY, colour = "#769ECB") + #level = NULL,
  geom_line(linetype = 'dashed', colour = '#000000') +
  xlab("Year (monthly data)") + ylab("Number of employed in retail (000)") +
  ggtitle("U.S. retail employment data")
plotNYDrift
```



#Q3)b)

```
tsLCNY<-tsLC2%>%filter(state=="NY")


tsLCNY<-tsLC2%>% bind_cols(month) %>%
    as_tsibble(index = month)

fitNY <-
  tsLCNY %>%
  model(TSLM(loansPerCapita ~
trend()+season()+avgLoans+NYCPI+NYUnemployment+avgAnnualInc))
report(fitNY)

## Series: loansPerCapita
## Model: TSLM
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.81392 -0.12802 -0.03126  0.09771  1.44289
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)     1.487e+01  3.073e+00   4.838 4.72e-06 ***
## trend()         4.076e-02  4.873e-03   8.364 3.49e-13 ***
## season()year2  -4.733e-02  1.398e-01  -0.339    0.736
## season()year3   2.074e-01  1.403e-01   1.478    0.143
```

```
## season()year4   1.324e-01  1.440e-01   0.919    0.360
## season()year5   4.690e-02  1.468e-01   0.320    0.750
## season()year6   8.802e-02  1.468e-01   0.600    0.550
## season()year7   2.298e-01  1.451e-01   1.584    0.116
## season()year8   1.825e-01  1.446e-01   1.263    0.210
## season()year9  -4.391e-03  1.442e-01  -0.030    0.976
## season()year10  2.355e-01  1.425e-01   1.653    0.101
## season()year11  1.784e-02  1.404e-01   0.127    0.899
## season()year12 -8.589e-02  1.401e-01  -0.613    0.541
## avgLoans        7.856e-06  2.436e-05   0.322    0.748
## NYCPI          -2.163e-02  4.754e-03  -4.551 1.49e-05 ***
## NYUnemployment -1.782e-01  2.270e-02  -7.853 4.48e-12 ***
## avgAnnualInc    1.419e-06  2.822e-06   0.503    0.616
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3114 on 101 degrees of freedom
## Multiple R-squared: 0.9012,  Adjusted R-squared: 0.8856
## F-statistic:  57.6 on 16 and 101 DF, p-value: < 2.22e-16
```

```
fitNY1 <-
  tsLCNY %>%
  model(TSLM(loansPerCapita ~ avgLoans+NYCPI+NYUnemployment+avgAnnualInc))
report(fitNY1)
```

```
## Series: loansPerCapita
## Model: TSLM
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.9332 -0.2363 -0.1146  0.2222  1.8740
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)    -9.081e+00  1.337e+00  -6.794 5.37e-10 ***
## avgLoans        5.840e-05  2.511e-05   2.326   0.0218 *
## NYCPI           1.505e-02  2.153e-03   6.988 2.05e-10 ***
## NYUnemployment -2.671e-01  2.527e-02 -10.573  < 2e-16 ***
## avgAnnualInc    9.630e-07  3.442e-06   0.280   0.7802
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3936 on 113 degrees of freedom
## Multiple R-squared: 0.8235,  Adjusted R-squared: 0.8172
## F-statistic: 131.8 on 4 and 113 DF, p-value: < 2.22e-16
```

```
#+countOfLoans+avgAnnualInc

#fitNYAll <-
#  tsLCNY %>%
```

```
#  model(TSLM(loansPerCapita ~
avgLoans+totalLoans+avgTerm+avgIntRate+avgGrade+avgEmpLength+avgAnnualInc+avg
VerifStatus+avgHomeOwner+avgOpenAcc+avgRevolBal+avgRevolUtil+avgTotalAcc+coun
tOfLoans+totalPop +NYCPI+NYUnemployment+NYCondoPriceIdx+NYSnapBenefits
#))
#report(fitNYAll)

#str(tsLC2)

#avgLoans+totalLoans+avgTerm+avgIntRate+avgGrade+avgEmpLength+avgAnnualInc+av
gVerifStatus+avgHomeOwner+avgOpenAcc+avgRevolBal+avgRevolUtil+avgTotalAcc+cou
ntOfLoans+totalPop+NYCPI+NYUnemployment+NYCondoPriceIdx+NYSnapBenefits
```

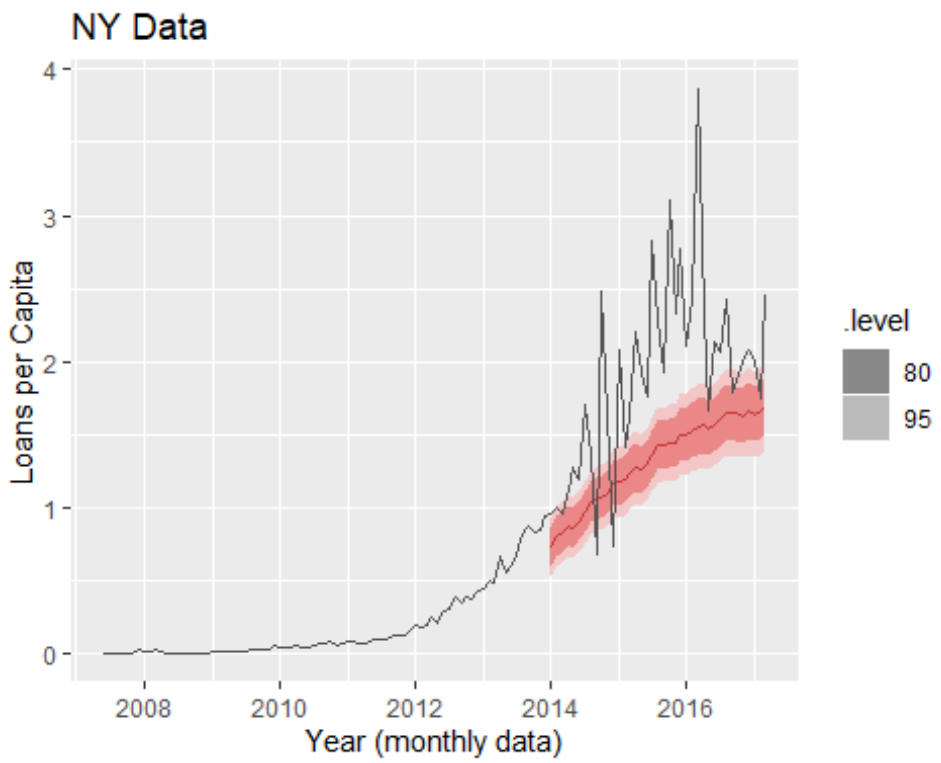
Q3)c) ## Regression plot with fitted values

```
fitNYFitted <-
  augment(fitNY1) %>%
  ggplot(aes(x = month)) +
  geom_line(aes(y = loansPerCapita, colour = "Data")) +
  geom_line(aes(y = .fitted, colour = "Fitted")) +
  xlab("Year") + ylab("Loans Per Capita") +
  ggtitle("NY Data") +
  scale_x_date(date_breaks = "years" , date_labels = "%y") +
  guides(colour=guide_legend(title=NULL))
ggplotly(fitNYFitted)

fitNYFitted1 <-
  augment(fitNY) %>%
  ggplot(aes(x = month)) +
  geom_line(aes(y = loansPerCapita, colour = "Data")) +
  geom_line(aes(y = .fitted, colour = "Fitted")) +
  xlab("Year") + ylab("Loans Per Capita") +
  ggtitle("New york Data") +
  scale_x_date(date_breaks = "years" , date_labels = "%y") +
  guides(colour=guide_legend(title=NULL))
ggplotly(fitNYFitted1)
```

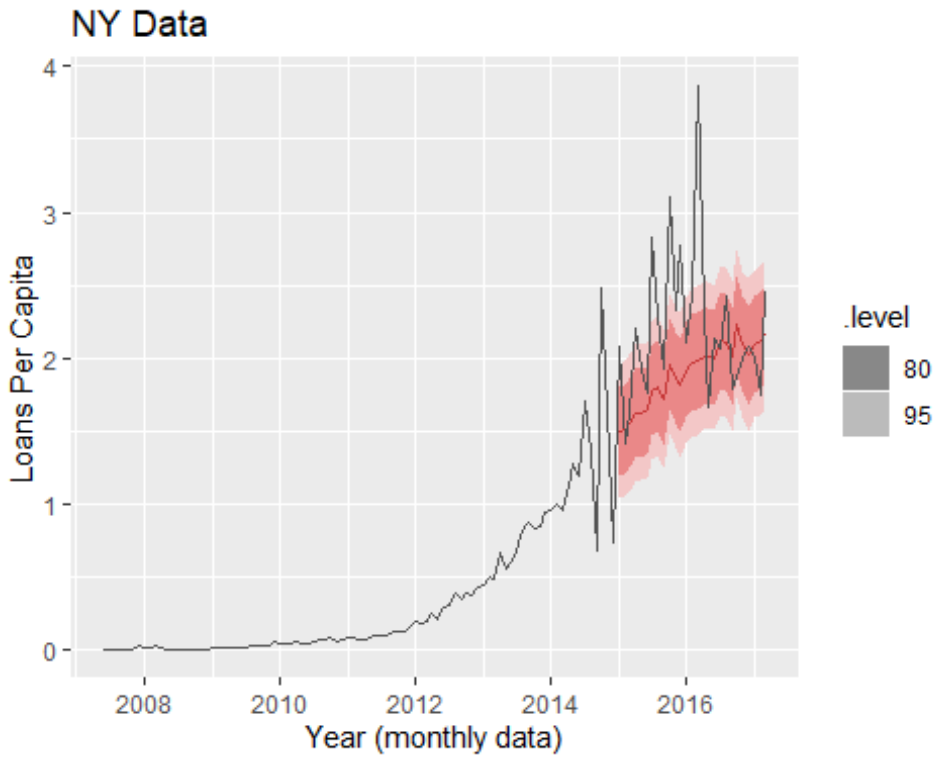
#Q3d)

```
#data before 2014
plotNYPredicted <-
  tsLCNY%>%
  filter(month < '2014-01-01') %>%
  model(TSLM(loansPerCapita ~ trend() +
season()+avgLoans+NYCPI+NYUnemployment+avgAnnualInc)) %>%
  forecast(new_data = tsLCNY %>% filter(month >= '2014-01-01')) %>%
  autoplot(tsLCNY, colour = "#960A0A") +
  geom_line(colour = '#535353') +
  xlab("Year (monthly data)") + ylab("Loans per Capita") +
  ggtitle("NY Data")
```

```
plotNYPredicted
```

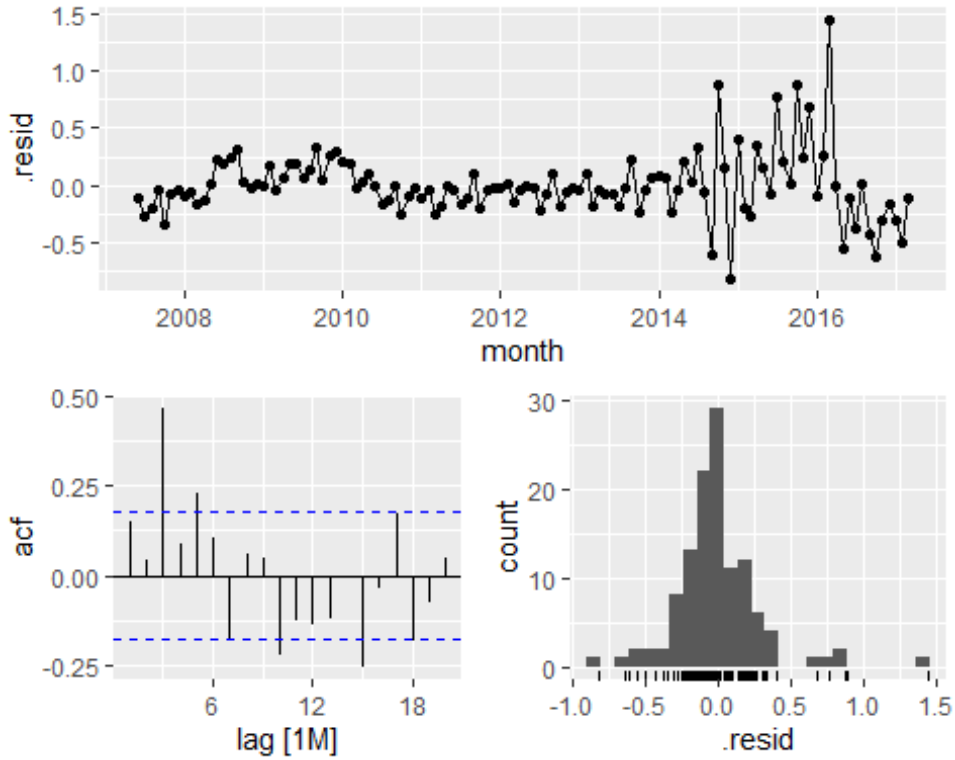



```
#Data before 2015
plotEconPredicted <-
  tsLCNY %>%
  filter(month < '2015-01-01') %>%
  model(TSLM(loansPerCapita ~ trend() +
season()+avgLoans+NYCPI+NYUnemployment+avgAnnualInc)) %>%
  forecast(new_data = tsLCNY %>% filter(month >= '2015-01-01')) %>%
  autoplot(tsLCNY, colour = "#960A0A") +
  geom_line(colour = '#535353') +
  xlab("Year (monthly data)") + ylab("Loans Per Capita") +
  ggtitle("NY Data")
plotEconPredicted
```



Q3)e)

```
fitNY %>% gg_tsresiduals()
```



Q3)f)

```
fitNYARIMAOthersGridAgain <-
  tsLCNY %>%
  model(fitArima = ARIMA(loansPerCapita
~avgLoans+NYCPI+NYUnemployment+avgAnnualInc, stepwise = FALSE, approximation
= FALSE)) #pdq(2,1,4) +

## Warning in sqrt(diag(best$var.coef)): NaNs produced

report(fitNYARIMAOthersGridAgain)

## Series: loansPerCapita
## Model: LM w/ ARIMA(1,0,4)(0,0,1)[12] errors
##
## Coefficients:
##          ar1      ma1      ma2     ma3      ma4    sma1  avgLoans  NYCPI
##       0.9827  -0.7309  -0.3432  0.8124  -0.2675  0.1955         0  2e-03
## s.e.  0.0131   0.0786   0.0886  0.0500   0.0754  0.1077       NaN  3e-04
##       NYUnemployment  avgAnnualInc
##              -0.1345             0
## s.e.             NaN           NaN
##
## sigma^2 estimated as 0.06458:  log likelihood=-3.04
## AIC=28.07   AICc=30.56   BIC=58.55
```

Q3)G)

```
tsLCNY %>%
  features(loansPerCapita, unitroot_ndiffs)

## # A tibble: 1 x 2
##   state ndiffs
##   <chr>  <int>
## 1 NY         1

#+avgAnnualInc

tsLCNY %>%
  features(difference(loansPerCapita), unitroot_ndiffs)

## # A tibble: 1 x 2
##   state ndiffs
##   <chr>  <int>
## 1 NY         0

tsLCNY %>%
  features(difference(loansPerCapita), unitroot_nsdiffs)

## # A tibble: 1 x 2
##   state nsdiffs
```

```
##   <chr>   <int>
## 1 NY          0
```

Q3)H)

```
fitArimadegree1<-
  tsLCNY %>%
  model(fitArima = ARIMA(loansPerCapita
~avgLoans+NYCPI+NYUnemployment+avgAnnualInc+pdq(d=1), stepwise = FALSE,
approximation = FALSE)) #pdq(2,1,4) +

## Warning in sqrt(diag(best$var.coef)): NaNs produced

report(fitArimadegree1)

## Series: loansPerCapita
## Model: LM w/ ARIMA(0,1,4) errors
##
## Coefficients:
##           ma1      ma2     ma3      ma4  avgLoans    NYCPI  NYUnemployment
##       -0.7915  -0.3301  0.8332  -0.3115         0  -0.0043         -0.1552
## s.e.   0.0894   0.0878  0.0753   0.0959         0   0.0006          0.0737
##       avgAnnualInc  intercept
##                  0     0.0222
## s.e.           NaN     0.0002
##
## sigma^2 estimated as 0.06381:  log likelihood=-1.61
## AIC=23.21   AICc=25.29   BIC=50.84
```

Q4)A)

```
set.seed(333)
tsNYTrain <- tsLCNY %>% filter(month < '2016-03-01')
tsNYTest <- tsLCNY %>% filter(month >= '2016-03-01')

tsNYFitAll <-
  tsNYTrain %>%
  model(
    model1 = TSLM(loansPerCapita ~ trend() + season()),
    model2=TSLM(loansPerCapita ~
trend()+season()+avgLoans+avgAnnualInc+NYCPI+NYUnemployment),
    model3= ARIMA(loansPerCapita,stepwise = FALSE, approximation = FALSE),
    model4=ARIMA(loansPerCapita ~avgLoans+avgAnnualInc+NYCPI+NYUnemployment,
stepwise = FALSE, approximation = FALSE))

## Warning in sqrt(diag(best$var.coef)): NaNs produced

tsNYPredictAll <-
  tsNYFitAll %>%
  forecast(new_data = tsNYTest)

accuracy(tsNYPredictAll, tsNYTest)
```

```
## # A tibble: 4 x 10
##   .model state .type     ME  RMSE   MAE   MPE  MAPE  MASE   ACF1
##   <chr>  <chr> <chr>  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  <dbl>
## 1 model1 NY    Test   0.340 0.702 0.406  11.4  15.1   NaN 0.137
## 2 model2 NY    Test  -0.379 0.687 0.598 -22.7  28.3   NaN 0.0838
## 3 model3 NY    Test  -0.790 0.972 0.906 -41.8  45.0   NaN 0.291
## 4 model4 NY    Test  -0.492 0.724 0.649 -27.7  31.8   NaN 0.0575
```

Q4)B)

```
set.seed(333)
tsNYTrain2 <- tsLCNY %>% filter(month < '2016-04-01')
tsNYTest2 <- tsLCNY %>% filter(month >= '2016-04-01')

tsNYFitAll2 <-
  tsNYTrain2 %>%
  model(
    model1 = TSLM(loansPerCapita ~ trend() + season()),
    model2=TSLM(loansPerCapita ~
trend()+season()+avgLoans+NYCPI+avgAnnualInc+NYUnemployment),
    model3= ARIMA(loansPerCapita,stepwise = FALSE, approximation = FALSE),
    model4=ARIMA(loansPerCapita ~NYCPI+NYUnemployment+avgLoans+avgAnnualInc,
stepwise = FALSE, approximation = FALSE))

## Warning in sqrt(diag(best$var.coef)): NaNs produced

    #  model4=



   # model2TimeTrend = TSLM(Consumption ~ trend()),
    #model3OtherFeatures = TSLM(Consumption ~ Income + Savings +
Unemployment),
    #model4TimeTrendAndOthers = TSLM(Consumption ~ trend() + Income + Savings
+ Unemployment),
    #model5TimeTrendAndOthersKnot = TSLM(Consumption ~ trend(knots = c(1980))
+ Income + Savings + Unemployment)


tsNYPredictAll2 <-
  tsNYFitAll2 %>%
  forecast(new_data = tsNYTest2)

accuracy(tsNYPredictAll2, tsNYTest2)

## # A tibble: 4 x 10
##   .model state .type      ME  RMSE   MAE    MPE  MAPE  MASE   ACF1
##   <chr>  <chr> <chr>   <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl>  <dbl>
## 1 model1 NY    Test   0.0943 0.268 0.207   3.43  9.73   NaN -0.185
## 2 model2 NY    Test  -0.654  0.695 0.654 -33.7  33.7    NaN -0.398
```

```
## 3 model3 NY    Test  -1.42   1.53  1.43  -71.6  72.1    NaN  0.167
## 4 model4 NY    Test  -0.402  0.509 0.444 -20.9  22.6    NaN -0.302
```

####Part2

Predicting/forecasting the U.S. retail sales

Q1)A)

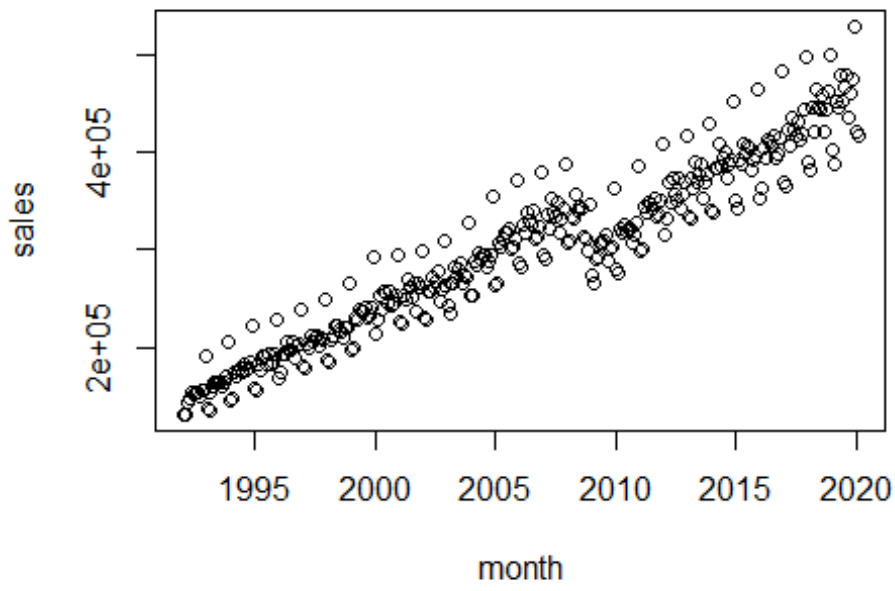
```
tsRetail<-read_csv("C:/Users/munis/Desktop/retailSales.csv")

## Parsed with column specification:
## cols(
##   date = col_character(),
##   sales = col_double()
## )
```

Q1)B)

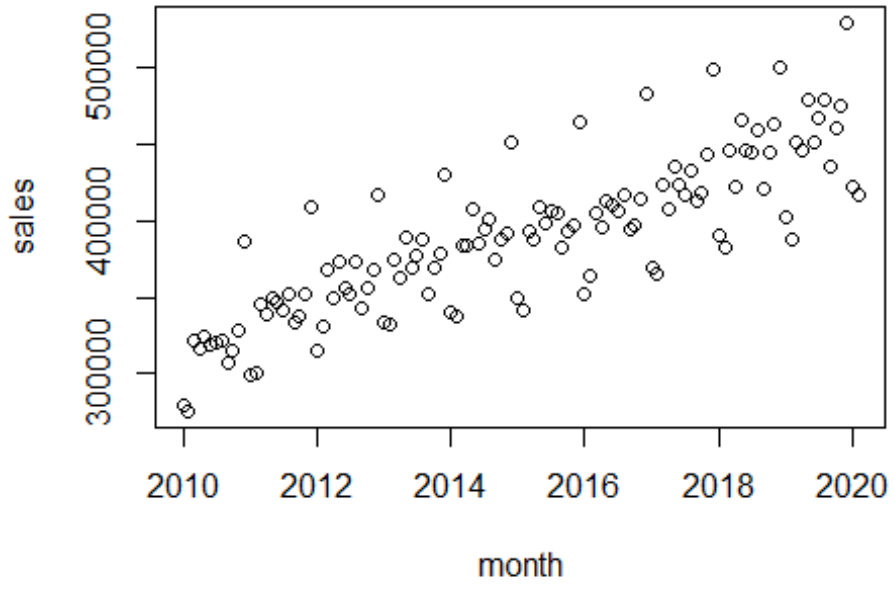
```
month <- as_tibble(yearmonth(seq(as.Date("1992-01-01"), as.Date("2020-02-
01"), by = "1 month"))) %>%
  rename(month=value)
tsRetail <- read_csv("C:/Users/munis/Desktop/retailSales.csv") %>%
  bind_cols(month) %>%
  select(month, sales = sales) %>%
  as_tsibble(index = month)

## Parsed with column specification:
## cols(
##   date = col_character(),
##   sales = col_double()
## )
```

Q1)C)

```
plot(tsRetail%>%select(month,sales))
```

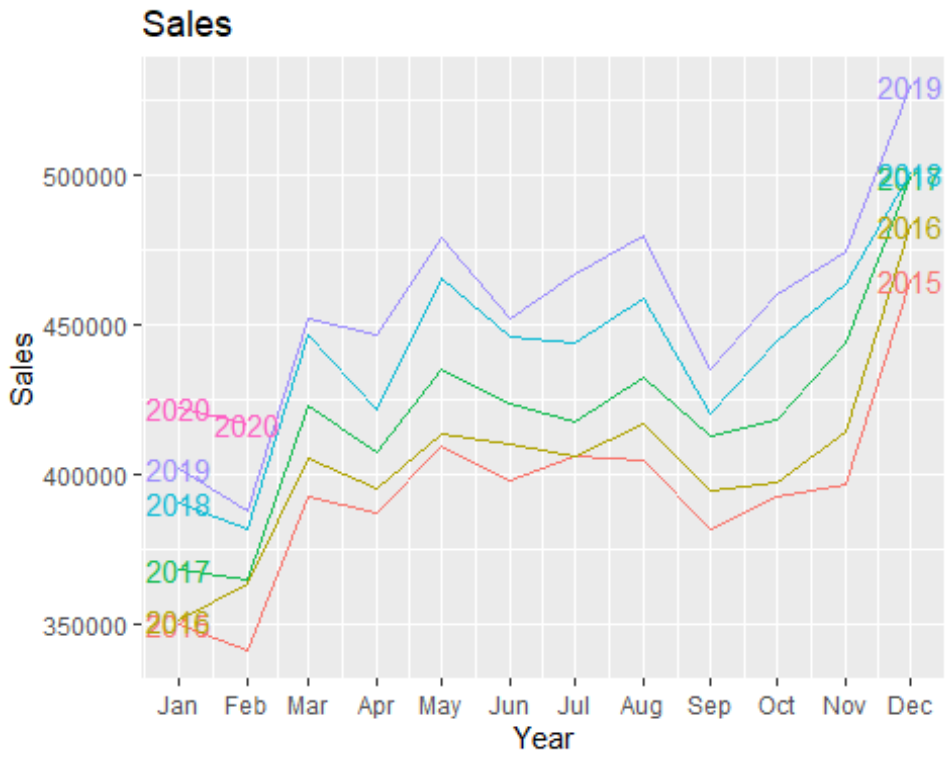



```
Plot2010<-tsRetail%>%
  filter(month >= '2010-01-01')%>%select(month,sales)
plot(Plot2010)
```

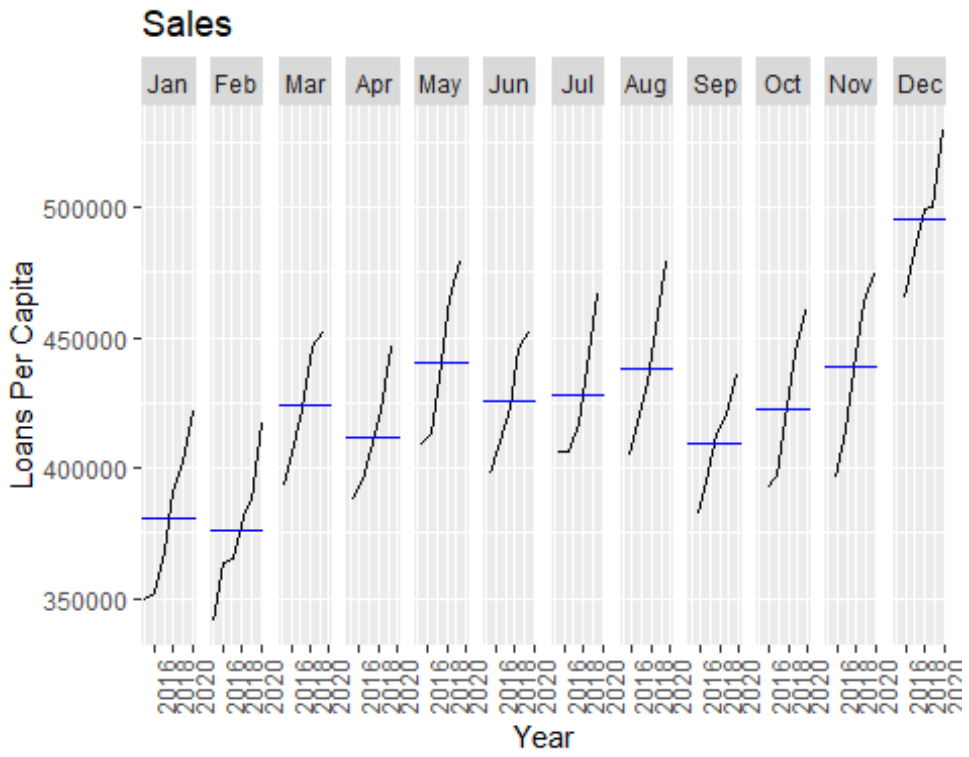


Q2)A)

```
plotSeason <-
 tsRetail %>% filter(month >= '2015-01-01')%>%
  gg_season(sales, labels = "both") +
  xlab("Year") + ylab("Sales") +
  ggtitle("Sales")
plotSeason
```

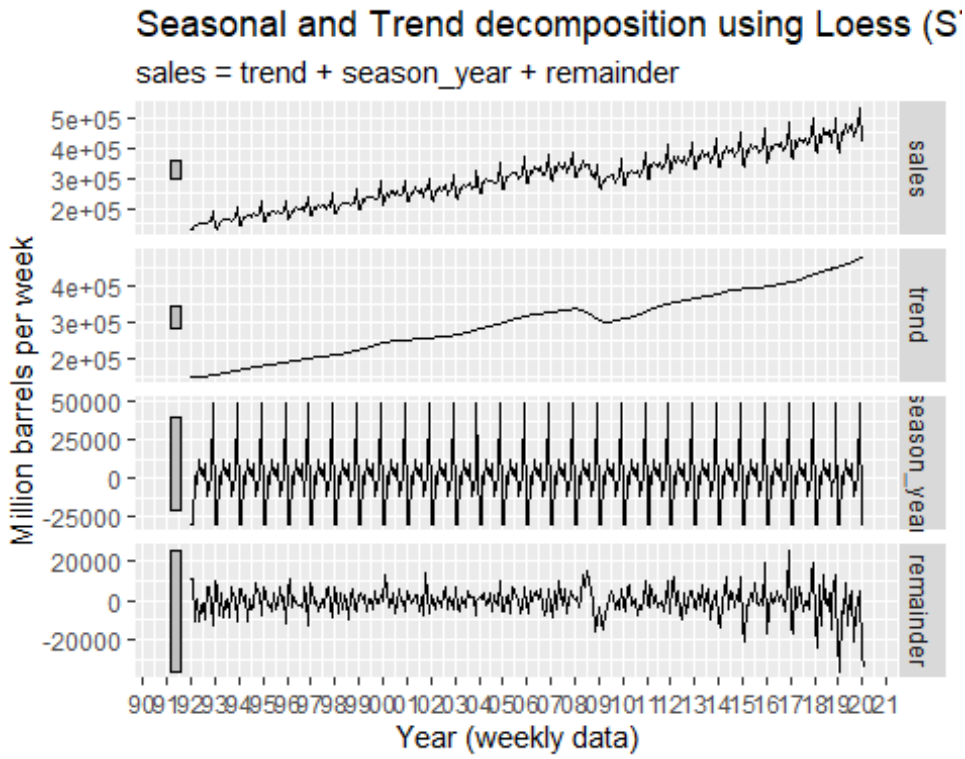



```
plotSubseries <-
  tsRetail %>% filter(month >= '2015-01-01')%>%
  gg_subseries(sales, labels = "both") +
  xlab("Year") + ylab("Loans Per Capita") +
  ggtitle("Sales")

## Warning: Ignoring unknown parameters: labels

plotSubseries
```

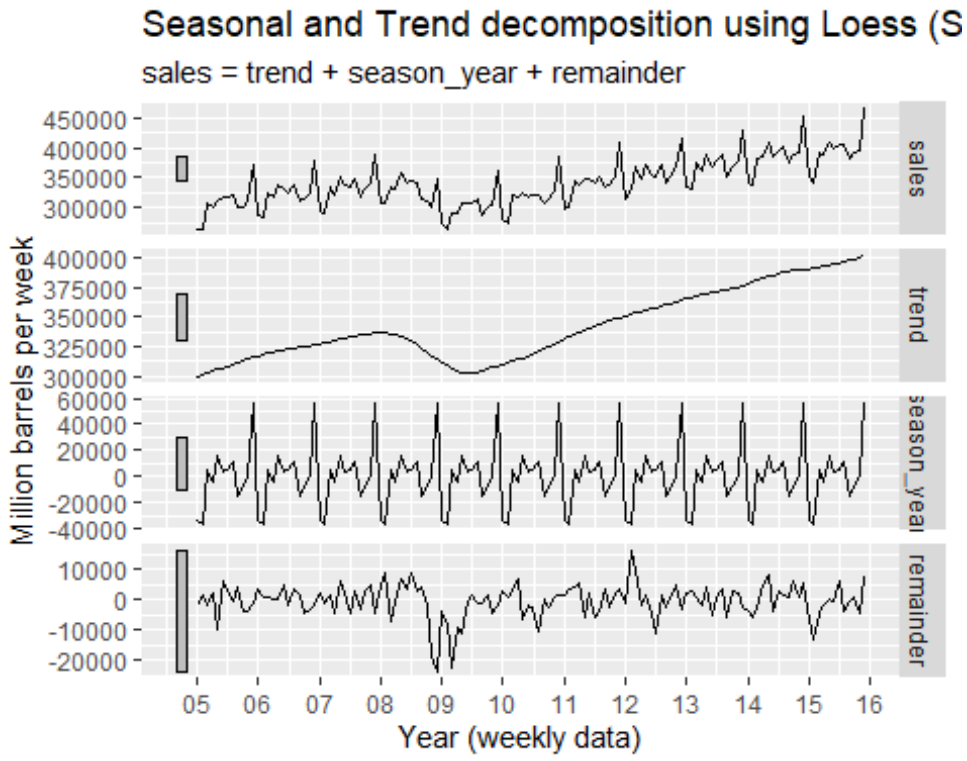


Q2)B)

```
plotRetailDecomposed <-
  tsRetail %>%
  model(STL(sales ~ trend() + season(window='periodic'), robust = TRUE)) %>%
  components() %>%
  autoplot() +
  xlab("Year (weekly data)") + ylab("Million barrels per week") +
  ggtitle("Seasonal and Trend decomposition using Loess (STL) for Retail
data") +
  scale_x_date(date_breaks = "years" , date_labels = "%y")
ggplotly(plotRetailDecomposed)

plotRetailDecomposed
```

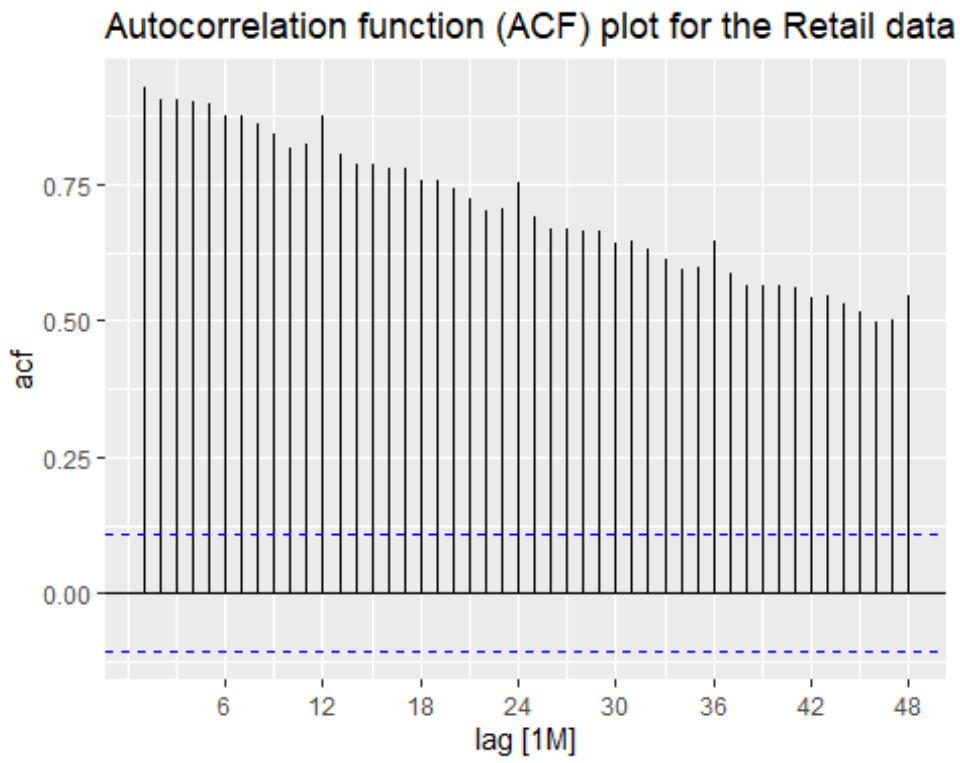



```
plotRetailDecomposed2 <-
  tsRetail %>%
  filter(month>= '2005-01-01' &  month<'2016-01-01')%>%
  model(STL(sales ~ trend() + season(window='periodic'), robust = TRUE)) %>%
  components() %>%
  autoplot() +
  xlab("Year (weekly data)") + ylab("Million barrels per week") +
  ggtitle("Seasonal and Trend decomposition using Loess (STL) for Retail
data") +
  scale_x_date(date_breaks = "years" , date_labels = "%y")

plotRetailDecomposed2
```

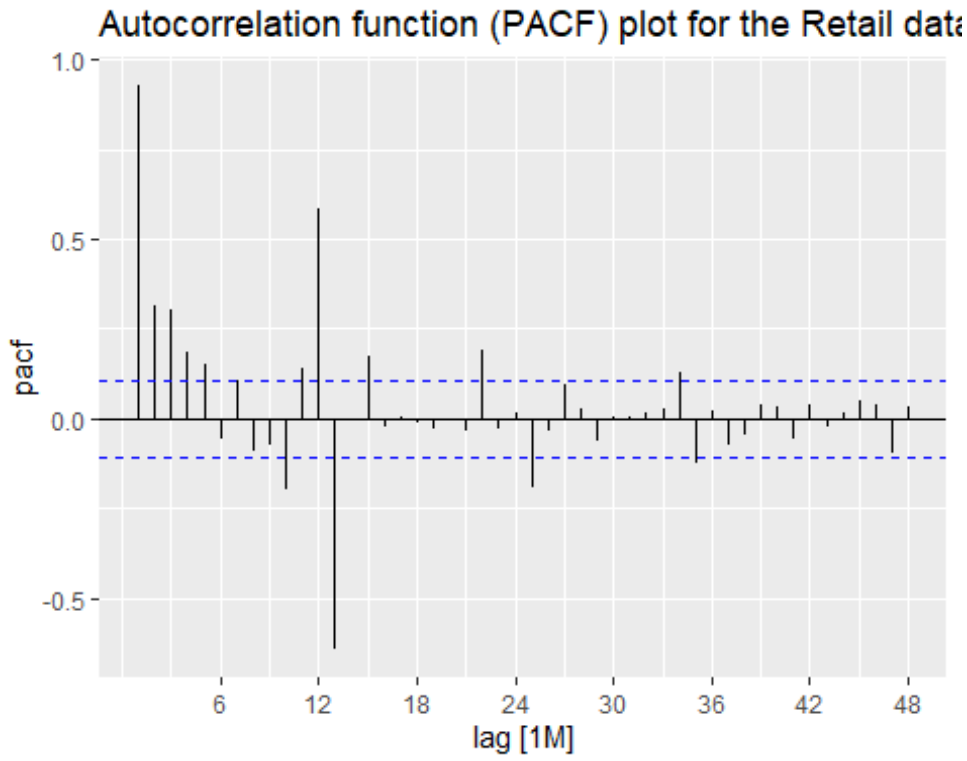


Q2)C)

```
plotRetailACF <-
  tsRetail %>%
  ACF(sales, lag_max = 48) %>%
  autoplot() +  ggtitle("Autocorrelation function (ACF) plot for the Retail
data")
plotRetailACF
```




```
plotRetailPACF <-
  tsRetail %>%
  PACF(sales, lag_max = 48) %>%
  autoplot() +  ggtitle("Autocorrelation function (PACF) plot for the Retail
data")
plotRetailPACF
```

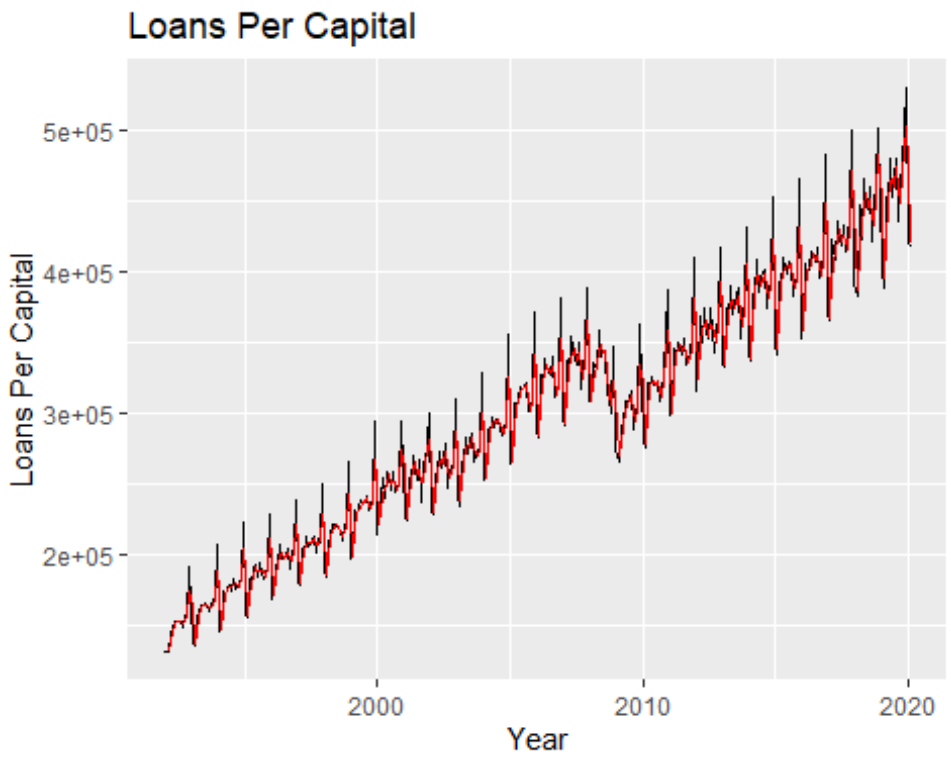


2)D)

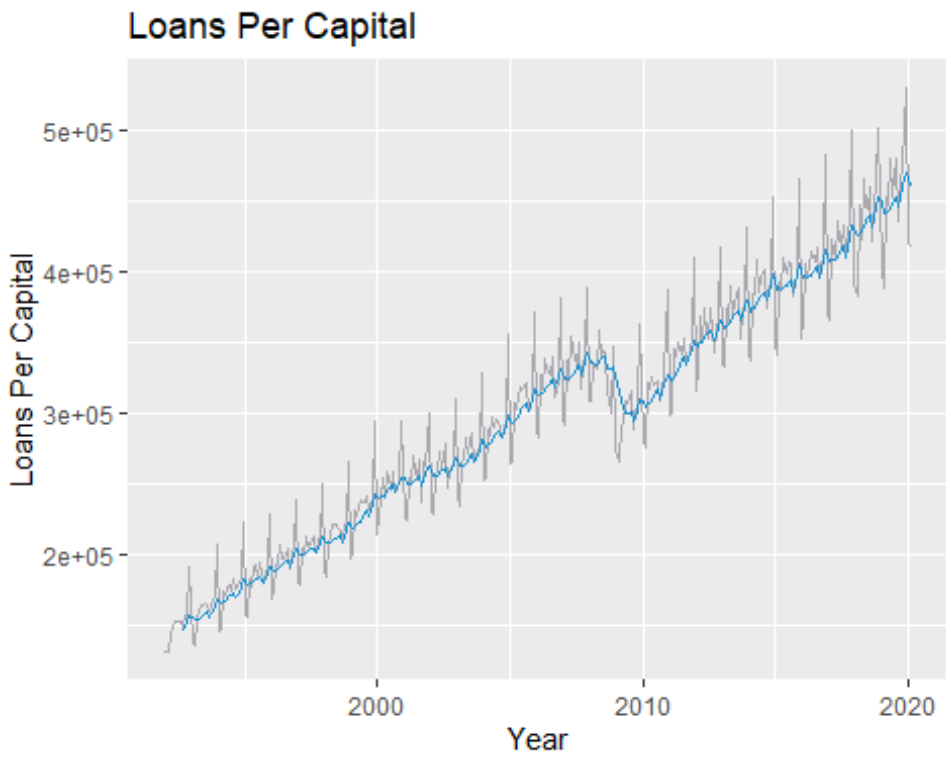
```
plotRetailSeasonallyAdjusted <-
  tsRetail %>%
  autoplot(sales, color='#A9A9B0') +
  autolayer(components(tsRetail%>% model(STL(sales))), season_adjust,
color='#1490D4') +
  xlab("Year (monthly data)") + ylab("Number of salesin retail (000)") +
  ggtitle("U.S. retail sales data")
ggplotly(plotRetailSeasonallyAdjusted)
```

Q2)E)

```
tsOrder <- tsRetail%>%
  mutate(
    `2-MA` = slide_dbl(sales, mean, .size = 2)
  )

tsOrder %>%
  autoplot(sales) +
  autolayer(tsOrder,`2-MA`, color='red') +
  xlab("Year") + ylab("Loans Per Capital") +
  ggtitle("Loans Per Capital") +
  guides(colour=guide_legend(title="series"))

## Warning: Removed 1 rows containing missing values (geom_path).
```




```
tsdate<-read_csv("C:/Users/munis/Desktop/retailSales.csv")

## Parsed with column specification:
## cols(
##   date = col_character(),
##   sales = col_double()
## )

tsOrder <- tsRetail%>%
  mutate(
    `9-MA` = slide_dbl(sales, mean, .size = 9)
  )

tsOrder %>%
  autoplot(sales, color='#A9A9B0') +
  autolayer(tsOrder,`9-MA`, color='#1490D4') +
  xlab("Year") + ylab("Loans Per Capital") +
  ggtitle("Loans Per Capital") +
  guides(colour=guide_legend(title="series"))

## Warning: Removed 8 rows containing missing values (geom_path).
```



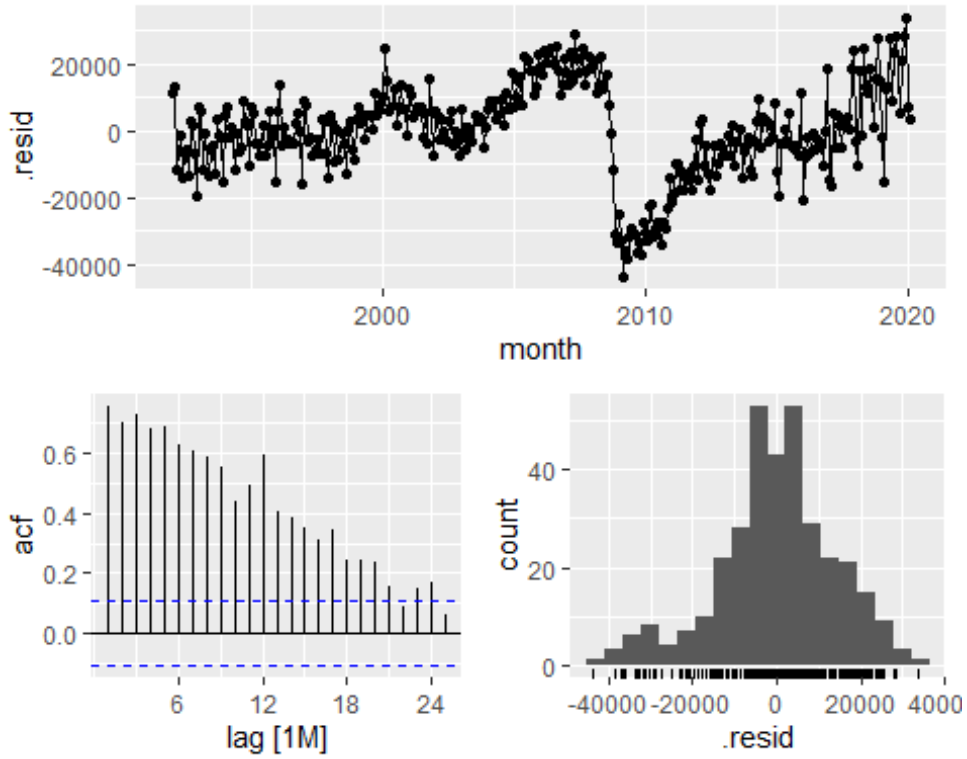
Q3)a)

```
fitRetail <-
  tsRetail %>%
  model(TSLM(sales ~ trend() + season()))
report(fitRetail)
```

```
## Series: sales
## Model: TSLM
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -43506  -6799    329   7662  33529
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)    118607.944   2948.209  40.231  < 2e-16 ***
## trend()           879.249      7.895 111.365  < 2e-16 ***
## season()year2   -2107.214   3717.967  -0.567    0.571
## season()year3   32961.493   3751.141   8.787  < 2e-16 ***
## season()year4   26615.138   3751.083   7.095 8.13e-12 ***
## season()year5   43380.853   3751.041  11.565  < 2e-16 ***
## season()year6   34385.747   3751.017   9.167  < 2e-16 ***
## season()year7   33746.927   3751.008   8.997  < 2e-16 ***
## season()year8   40570.572   3751.017  10.816  < 2e-16 ***
## season()year9   18758.787   3751.041   5.001 9.35e-07 ***
## season()year10  27201.181   3751.083   7.252 3.03e-12 ***
```

```
## season()year11  33160.718   3751.141   8.840  < 2e-16 ***
## season()year12  81780.970   3751.216  21.801  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 14160 on 325 degrees of freedom
## Multiple R-squared: 0.9759,  Adjusted R-squared: 0.975
## F-statistic:  1098 on 12 and 325 DF, p-value: < 2.22e-16
```

```
fitRetail %>% gg_tsresiduals()
```



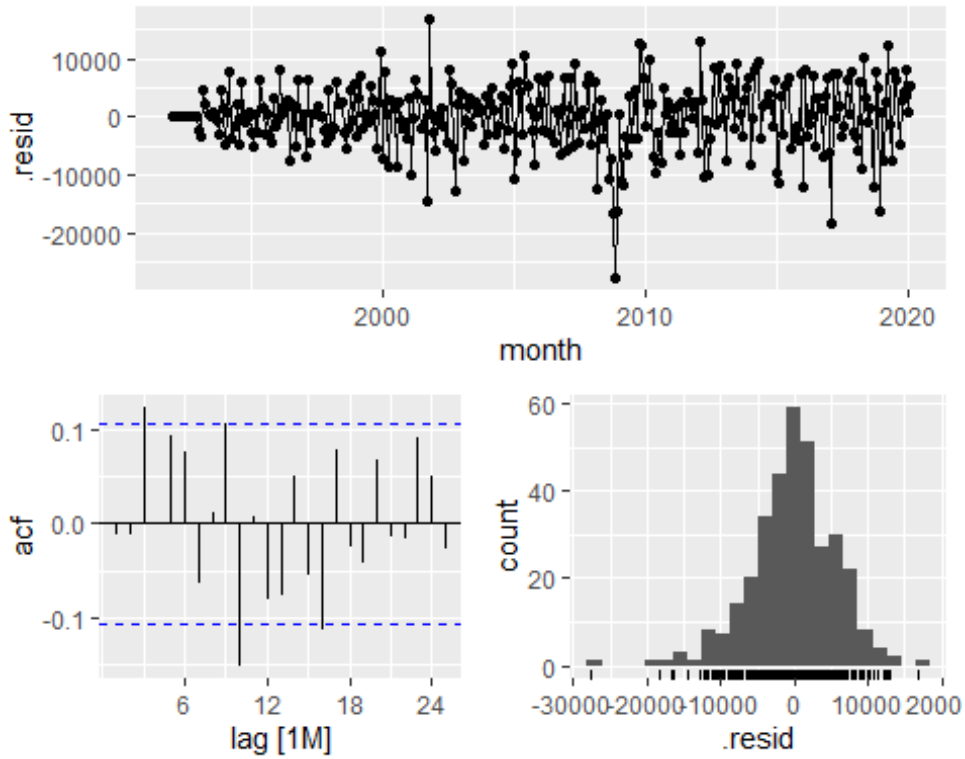
Q3)B)

```
fitRetailARIMA <-
  tsRetail %>%
  model(fitArima = ARIMA(sales,
              stepwise = FALSE, approximation = FALSE))
report(fitRetailARIMA)
```

```
## Series: sales
## Model: ARIMA(1,0,1)(2,1,2)[12] w/ drift
##
## Coefficients:
##          ar1      ma1    sar1     sar2     sma1    sma2  constant
##       0.9548  -0.3826  0.8277  -0.7356  -1.1695  0.6469  468.1625
## s.e.  0.0182   0.0546  0.0583   0.0648   0.0703  0.0672   90.9200
##
```

```
## sigma^2 estimated as 34553166:  log likelihood=-3295.85
## AIC=6607.71   AICc=6608.16   BIC=6638
```

```
fitRetailARIMA %>% gg_tsresiduals()
```



Q3)C)

```
tsRetail %>%
  features(sales, unitroot_ndiffs)
```

```
## # A tibble: 1 x 1
##   ndiffs
##    <int>
## 1      1
```

```
tsRetail %>%
  features(sales, unitroot_nsdiffs)
```

```
## # A tibble: 1 x 1
##   nsdiffs
##     <int>
## 1       1
```

```
tsRetail %>%
  features(difference(sales), unitroot_ndiffs)
```

```
## # A tibble: 1 x 1
##   ndiffs
```

```
##    <int>
## 1      0
```

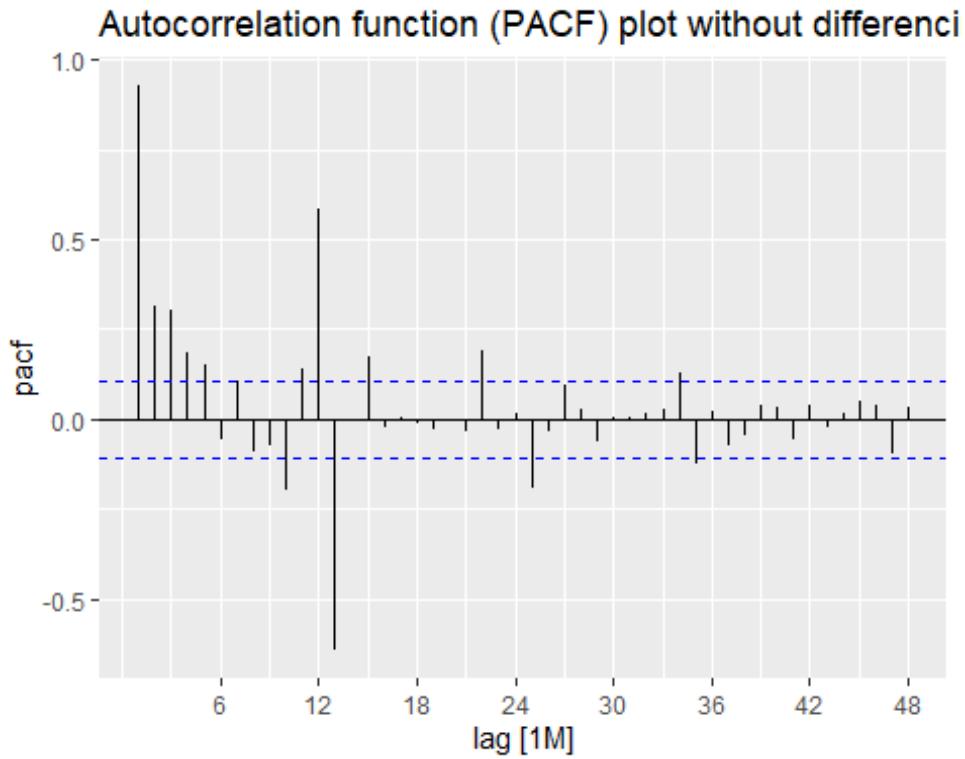
```
tsRetail %>%
  features(difference(sales,12), unitroot_nsdiffs)
```

```
## # A tibble: 1 x 1
##   nsdiffs
##     <int>
## 1       0
```

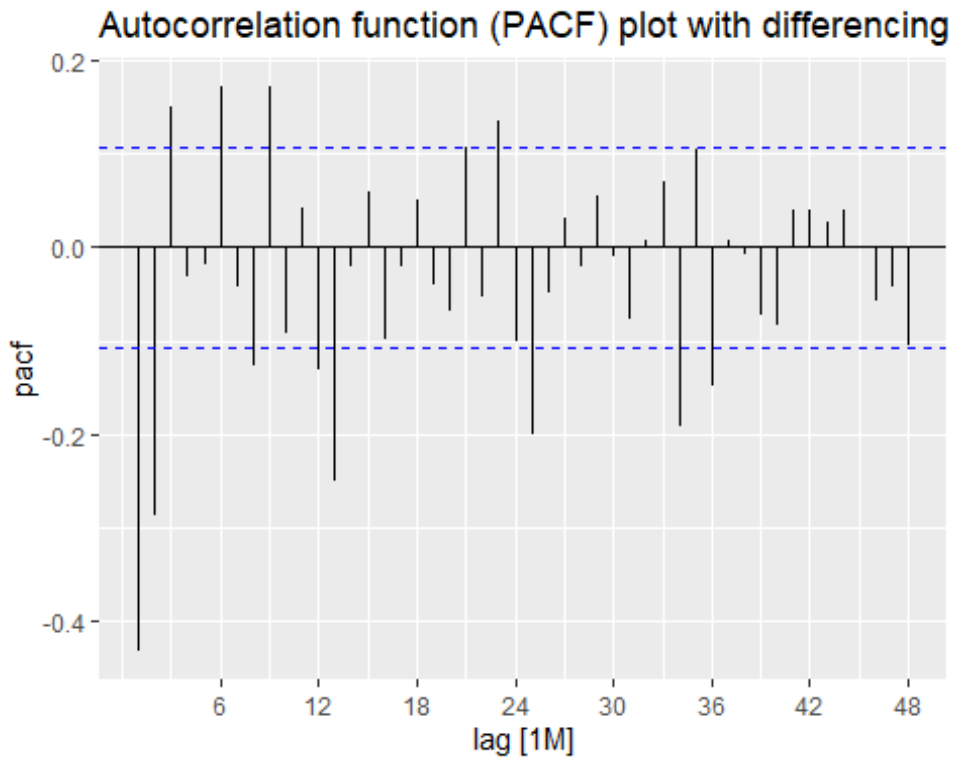
```
tsRetail %>%
  mutate(salesDiff = difference(difference(sales), 12)) %>%
  features(salesDiff, unitroot_kpss)
```

```
## # A tibble: 1 x 2
##   kpss_stat kpss_pvalue
##       <dbl>       <dbl>
## 1    0.0299         0.1
```

```
plotDiff <-
  tsRetail %>%
  mutate(diffsales = sales) %>%
  PACF(diffsales, lag_max = 48) %>%
  autoplot() +  ggtitle("Autocorrelation function (PACF) plot without
differencing")
plotDiff
```



```
plotSeasonalDiffACF <-
  tsRetail %>%
  mutate(diffSales2 = difference(difference(sales,12))) %>%
  PACF(diffSales2, lag_max = 48) %>%
  autoplot() +  ggtitle("Autocorrelation function (PACF) plot with
differencing")
plotSeasonalDiffACF
```



Q3)D)

```
set.seed(333)
tsRetailTrain <- tsRetail %>% filter(month < '2011-01-01')
tsRetailTest <- tsRetail %>% filter(month >= '2011-01-01')



tsFit <-
  tsRetailTrain %>%
  model(
    model1 = TSLM(sales ~ trend() + season()),
    Model2 = ARIMA(sales ,stepwise = FALSE,approximation = FALSE))



tsModel <-
```

```
  tsFit %>%
  forecast(new_data = tsRetailTest)

accuracy(tsModel, tsRetailTest)
```

```
## # A tibble: 2 x 9
##   .model .type     ME   RMSE    MAE    MPE  MAPE  MASE  ACF1
##   <chr>  <chr>  <dbl>  <dbl>  <dbl>  <dbl> <dbl> <dbl> <dbl>
## 1 model1 Test    969. 14250. 10815. -0.119  2.70   NaN 0.409
## 2 Model2 Test  16084. 20998. 16577.  3.76   3.91   NaN 0.604
```

```
model<-  tsRetailTrain %>%
  model(

    Model2 = ARIMA(sales ,stepwise = FALSE,approximation = FALSE))
report(model)
```

```
## Series: sales
## Model: ARIMA(4,0,0)(0,1,2)[12] w/ drift
##
## Coefficients:
##          ar1     ar2     ar3      ar4     sma1     sma2  constant
##       0.5369  0.2366  0.3714  -0.2190  -0.4764  -0.1598  726.1276
## s.e.  0.0672  0.0719  0.0779   0.0701   0.0827   0.0798  148.8050
##
## sigma^2 estimated as 34044160:  log likelihood=-2179.35
## AIC=4374.7   AICc=4375.4   BIC=4401.71
```

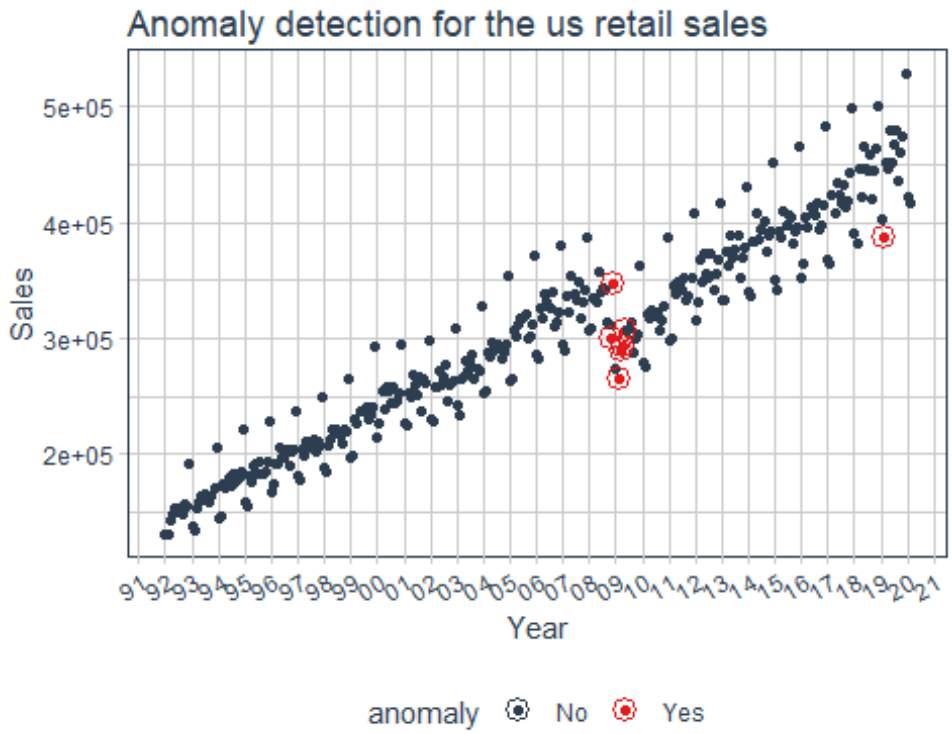
Q3)E)

```
set.seed(333)
tsRetailTrain2 <- tsRetail %>% filter(month < '2016-01-01')
tsRetailTest2 <- tsRetail %>% filter(month >= '2016-01-01')



tsFit2 <-
  tsRetailTrain2 %>%
  model(
    model1 = TSLM(sales ~ trend() + season()),
    Model2 = ARIMA(sales ,stepwise = FALSE,approximation = FALSE))




tsModel2 <-
  tsFit2 %>%
  forecast(new_data = tsRetailTest2)

accuracy(tsModel2, tsRetailTest2)
```

```
## # A tibble: 2 x 9
##   .model .type     ME   RMSE    MAE   MPE  MAPE  MASE  ACF1
##   <chr>  <chr>  <dbl>  <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 model1 Test  11405. 18692. 14567. 2.39   3.24   NaN 0.366
## 2 Model2 Test   4641. 10360.  8432. 0.934  1.93   NaN 0.371
```

Q4)A)

```
library(anomalize)

#tsDia%>%month = date(month)


anomalyDia <-
  tsRetail %>%
  mutate(month = date(month)) %>%
  time_decompose(sales, method = "stl") %>%
  anomalize(remainder, method = "gesd") %>%
  plot_anomalies() +
  labs(title = "Anomaly detection for the us retail sales") +
  xlab("Year") + ylab("Sales") +
  scale_x_date(date_breaks = "years" , date_labels = "%y")

## Converting from tbl_ts to tbl_time.
## Auto-index message: index = month

## frequency = 12 months

## trend = 60 months

anomalyDia
```



Q4)B)

```
tsplot <- tsRetail %>% filter(month >= '2010-01-01')

fitFit3d <-
  tsplot %>%
  ggplot(aes(x = month)) +
  geom_line(aes(y = sales, colour = "Data")) +
  autolayer(tsModel[ which(tsModel$.model=='model1'), ])+
  geom_line(aes(y =sales , colour = "FittedTrendSeason"),data = tsModel[
which(tsModel$.model=='model1'), ]) +

  autolayer(tsModel[ which(tsModel$.model=='Model2'), ])+
  geom_line(aes(y =sales , colour = "FittedArima"),data = tsModel[
which(tsModel$.model=='Model2'), ]) +



  xlab("Year") + ylab("Sales") +
  ggtitle("Sales Data") +
  scale_x_date(date_breaks = "years" , date_labels = "%y") +
  guides(colour=guide_legend(title=NULL))

ggplotly(fitFit3d)
```

```
## Warning in geom2trace.default(dots[[1L]][[1L]], dots[[2L]][[1L]],
dots[[3L]][[1L]]): geom_GeomForecast() has yet to be implemented in plotly.
##   If you'd like to see this geom implemented,
##   Please open an issue with your example code at
##   https://github.com/ropensci/plotly/issues

## Warning in geom2trace.default(dots[[1L]][[1L]], dots[[2L]][[1L]],
dots[[3L]][[1L]]): geom_GeomForecast() has yet to be implemented in plotly.
##   If you'd like to see this geom implemented,
##   Please open an issue with your example code at
##   https://github.com/ropensci/plotly/issues
```

```
fitFit3e <-
  tsplot %>%
  ggplot(aes(x = month)) +
  geom_line(aes(y = sales, colour = "Data")) +
  autolayer(tsModel2[ which(tsModel2$.model=='model1'), ])+
  geom_line(aes(y =sales , colour = "Fittedtrendand Season"),data = tsModel2[
which(tsModel2$.model=='model1'), ])+

  autolayer(tsModel2[ which(tsModel2$.model=='Model2'), ])+
  geom_line(aes(y =sales , colour = "FittedArima"),data = tsModel2[
which(tsModel2$.model=='Model2'), ]) +



  xlab("Year") + ylab("Sales") +
  ggtitle("Sales Data") +
  scale_x_date(date_breaks = "years" , date_labels = "%y") +
  guides(colour=guide_legend(title=NULL))

ggplotly(fitFit3e)
```

```
## Warning in geom2trace.default(dots[[1L]][[1L]], dots[[2L]][[1L]],
dots[[3L]][[1L]]): geom_GeomForecast() has yet to be implemented in plotly.
##   If you'd like to see this geom implemented,
##   Please open an issue with your example code at
##   https://github.com/ropensci/plotly/issues

## Warning in geom2trace.default(dots[[1L]][[1L]], dots[[2L]][[1L]],
dots[[3L]][[1L]]): geom_GeomForecast() has yet to be implemented in plotly.
##   If you'd like to see this geom implemented,
##   Please open an issue with your example code at
##   https://github.com/ropensci/plotly/issues
```