Assignment 4

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:\Program Files\R\R-3.6.2")

library("tidyverse")

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

## v ggplot2 3.3.0     v purrr   0.3.3
## v tibble  3.0.0     v dplyr   0.8.3
## v tidyr   1.0.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0

## Warning: package 'ggplot2' was built under R version 3.6.3

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

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

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

library("tidymodels")

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

## -- Attaching packages -------------------------------------- tidymodels
0.1.0 --

## v broom     0.5.5      v rsample   0.0.6
## v dials     0.0.5      v tune      0.1.0
## v infer     0.5.1      v workflows 0.1.1
## v parsnip   0.0.5      v yardstick 0.0.6
## v recipes   0.1.10

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

## Warning: package 'scales' was built under R version 3.6.3
## Warning: package 'recipes' was built under R version 3.6.3

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

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

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

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

## -- 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()

library("plotly")

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

##
## 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")

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

library("caret")

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

## Loading required package: lattice

##
## Attaching package: 'caret'
## The following objects are masked from 'package:yardstick':
##
##     precision, recall, sensitivity, specificity

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

```r
library("lubridate")
```

```
##
## Attaching package: 'lubridate'

## The following object is masked from 'package:base':
##
##     date
```

```r
library("plyr")
```

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

## -------------------------------------------------------------------------
----

## You have loaded plyr after dplyr - this is likely to cause problems.
## If you need functions from both plyr and dplyr, please load plyr first,
then dplyr:
## library(plyr); library(dplyr)

## -------------------------------------------------------------------------
----

##
## Attaching package: 'plyr'

## The following object is masked from 'package:lubridate':
##
##     here

## The following objects are masked from 'package:plotly':
##
##     arrange, mutate, rename, summarise

## The following objects are masked from 'package:dplyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize

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

```
library("dplyr")
library("fpp3")
```

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

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

## v tsibble     0.8.6     v feasts      0.1.3
## v tsibbledata 0.1.0     v fable       0.1.2

## 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 plyr::arrange()          masks plotly::arrange(), dplyr::arrange()
## x plyr::compact()          masks purrr::compact()
## x plyr::count()            masks dplyr::count()
## x lubridate::date()        masks base::date()
## x scales::discard()        masks purrr::discard()
## x plyr::failwith()         masks dplyr::failwith()
## x plotly::filter()         masks dplyr::filter(), stats::filter()
## x fabletools::generate()   masks infer::generate()
## x plyr::here()             masks lubridate::here()
## x tsibble::id()            masks plyr::id(), dplyr::id()
## x tsibble::interval()      masks lubridate::interval()
## x dplyr::lag()             masks stats::lag()
## x caret::lift()            masks purrr::lift()
## x fabletools::MAE()        masks caret::MAE()
## x dials::margin()          masks ggplot2::margin()
## x plyr::mutate()           masks plotly::mutate(), dplyr::mutate()
## x tsibble::new_interval()  masks lubridate::new_interval()
## x fabletools::null_model() masks parsnip::null_model()
## x plyr::rename()           masks plotly::rename(), dplyr::rename()
## x fabletools::RMSE()       masks caret::RMSE()
## x plyr::summarise()        masks plotly::summarise(), dplyr::summarise()
## x plyr::summarize()        masks dplyr::summarize()
```

```
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 </>
```

Part 1

Question 1

```
# 1.a

tsLCOrg <- read_csv("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()
## )

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	▁▇▁▁▁
avgEmpLength	2	1	5.57	1.45	1.00	5.03	5.99	6.36	10.00	▁▃▇▂▁
avgAnnualInc	0	1	69476.03	18173.26	2000.00	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	▇▁▁▁▁

```
# 1.b

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

```
# 1.c

summary(tsLCOrg)
```

```
##       date               state              avgLoans       totalLoans
##  Min.   :2007-06-01   Length:4943        Min.   :  500   Min.   :      500
##  1st Qu.:2010-06-01   Class :character   1st Qu.:10583   1st Qu.:   115688
##  Median :2012-11-01   Mode  :character   Median :13704   Median :   927825
##  Mean   :2012-09-13                      Mean   :12607   Mean   :  4234850
##  3rd Qu.:2015-02-01                      3rd Qu.:14954   3rd Qu.:  4303438
##  Max.   :2017-03-01                      Max.   :29975   Max.   :126477500
##
##     avgTerm        avgIntRate       avgGrade      avgEmpLength
##  Min.   :36.00   Min.   : 6.03   Min.   :1.000   Min.   : 1.000
##  1st Qu.:36.00   1st Qu.:12.17   1st Qu.:2.571   1st Qu.: 5.026
##  Median :42.13   Median :12.96   Median :2.750   Median : 5.989
##  Mean   :41.38   Mean   :12.92   Mean   :2.769   Mean   : 5.569
##  3rd Qu.:43.90   3rd Qu.:13.89   3rd Qu.:2.923   3rd Qu.: 6.362
##  Max.   :60.00   Max.   :23.63   Max.   :7.000   Max.   :10.000
##                                                  NA's   :2
##   avgAnnualInc    avgVerifStatus    avgHomeOwner       avgOpenAcc
##  Min.   :  2000   Min.   :0.0000   Min.   :0.00000   Min.   : 1.000
##  1st Qu.: 62276   1st Qu.:0.5333   1st Qu.:0.04000   1st Qu.: 9.446
##  Median : 69841   Median :0.6667   Median :0.08898   Median :10.910
##  Mean   : 69476   Mean   :0.5768   Mean   :0.09383   Mean   :10.505
##  3rd Qu.: 76669   3rd Qu.:0.7244   3rd Qu.:0.12500   3rd Qu.:11.715
##  Max.   :556880   Max.   :1.0000   Max.   :1.00000   Max.   :25.000
##                                                      NA's   :9
##   avgRevolBal      avgRevolUtil    avgTotalAcc     countOfLoans
##  Min.   :     0   Min.   : 0.00   Min.   : 1.00   Min.   :   1
##  1st Qu.: 12984   1st Qu.:49.06   1st Qu.:22.20   1st Qu.:  11
##  Median : 15465   Median :53.97   Median :24.61   Median :  67
##  Mean   : 15796   Mean   :52.59   Mean   :23.89   Mean   : 287
##  3rd Qu.: 17677   3rd Qu.:57.94   3rd Qu.:26.29   3rd Qu.: 290
##  Max.   :404868   Max.   :99.40   Max.   :61.00   Max.   :8081
##                   NA's   :12      NA's   :9
```

```
# 1.d

nyei_df <- read_csv("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()
## )
```

```
nyei_df$date <- mdy(nyei_df$date)
nyei_df <- as_tsibble(nyei_df, index = date, key = state)
nyei_df
```

```
## # A tsibble: 118 x 6 [1D]
## # Key:       state [1]
##    date       state NYCPI NYUnemployment NYCondoPriceIdx NYSnapBenefits
##    <date>     <chr> <dbl>          <dbl>           <dbl>          <dbl>
##  1 2007-06-01 NY     660.            4.5            228.        1801707
##  2 2007-07-01 NY     661.            4.6            228.        1792916
##  3 2007-08-01 NY     660.            4.7            227.        1816805
##  4 2007-09-01 NY     660.            4.7            226.        1823494
##  5 2007-10-01 NY     661.            4.8            226.        1825759
##  6 2007-11-01 NY     663.            4.8            227.        1830858
##  7 2007-12-01 NY     663.            4.8            227.        1849851
##  8 2008-01-01 NY     665.            4.8            227.        1932022
##  9 2008-02-01 NY     668.            4.9            229.        1927903
## 10 2008-03-01 NY     674.            4.9            231.        1950582
## # ... with 108 more rows
```

```
# 1.e.i

pop_df <- read_csv("statePop.csv")
```

```
## Parsed with column specification:
## cols(
##   state = col_character(),
##   `Total population` = col_double()
## )
```

```
tsLCOrg <- inner_join(tsLCOrg, pop_df, by = "state")
tsLCOrg
```

```
## # A tsibble: 4,943 x 17 [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
```

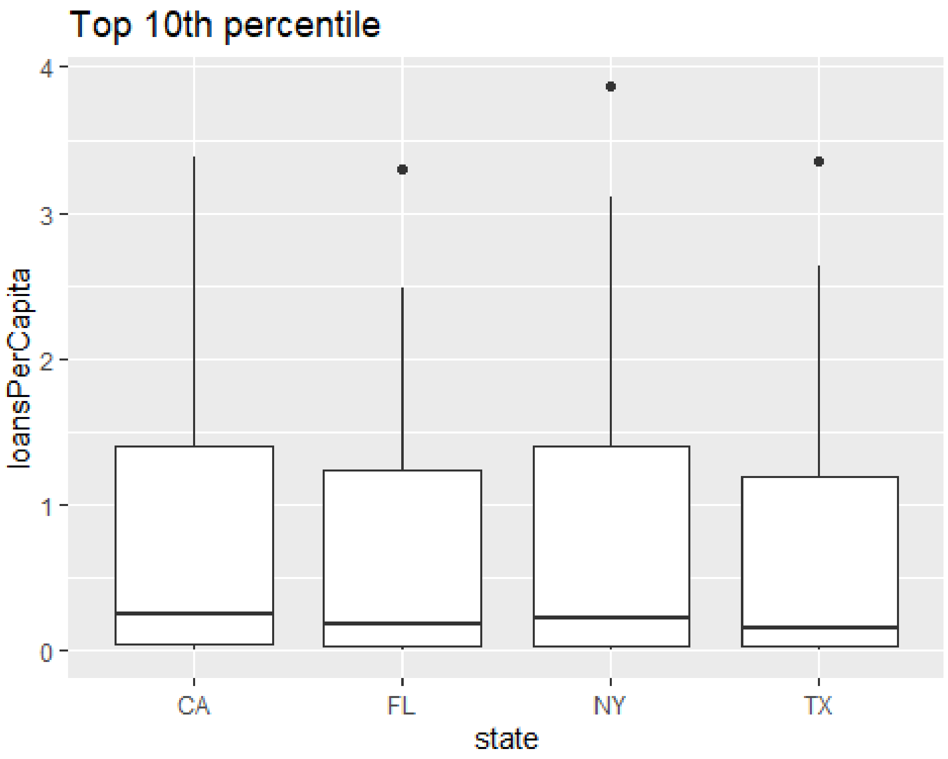
```
7
##  6 2009-03-01 AK      14662.      29325      36      13         3
7
##  7 2009-04-01 AK      20000       20000      36      11.9       2
5
##  8 2009-05-01 AK      16000       16000      36      12.2       2
2
##  9 2009-07-01 AK       1000        1000      36      11.9       2
10
## 10 2009-11-01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

```
# 1.e.ii

tsLCOrg$loansPerCapita <- tsLCOrg$totalLoans/tsLCOrg$`Total population`
tsLCOrg
```

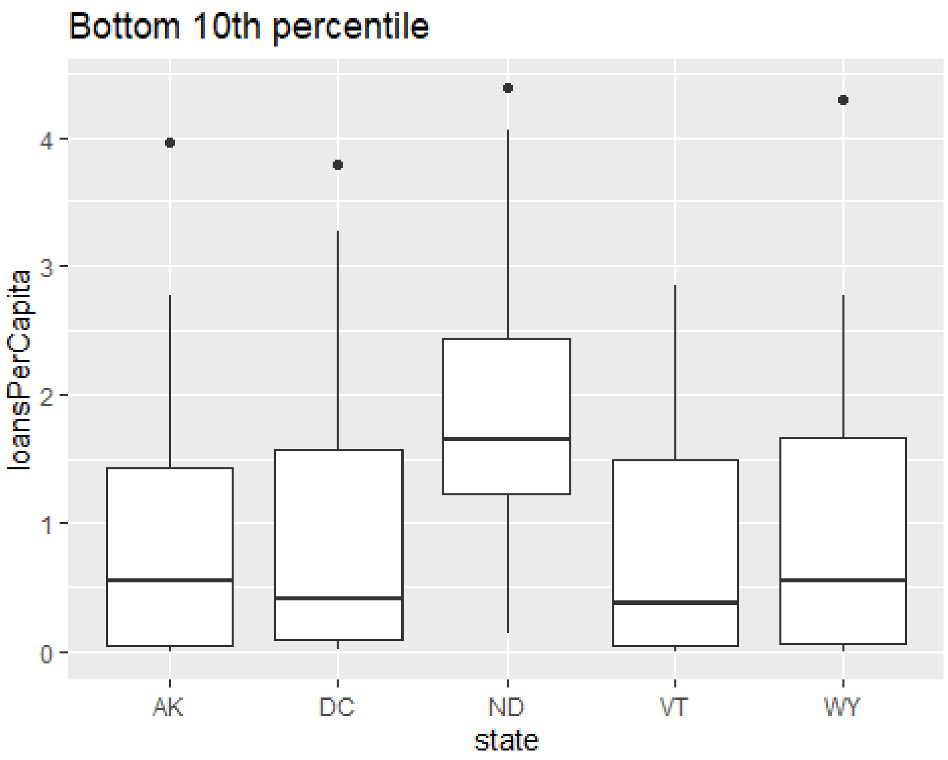
```
## # A tsibble: 4,943 x 18 [1D]
## # Key:       state [51]
##    date       state avgLoans totalLoans avgTerm avgIntRate avgGrade
avgEmpLength
##    <date>     <chr>    <dbl>      <dbl>   <dbl>      <dbl>    <dbl>
<dbl>
##  1 2008-01-01 AK       5600        5600      36      18.0       7
5
##  2 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 10 more variables: avgAnnualInc <dbl>,
## #   avgVerifStatus <dbl>, avgHomeOwner <dbl>, avgOpenAcc <dbl>,
## #   avgRevolBal <dbl>, avgRevolUtil <dbl>, avgTotalAcc <dbl>,
## #   countOfLoans <dbl>, `Total population` <dbl>, loansPerCapita <dbl>

# 1.e.iii

tsLC <- left_join(tsLCOrg, nyei_df) %>%
  as_tsibble(index = date, key = state)

## Joining, by = c("date", "state")

tsLC

## # A tsibble: 4,943 x 22 [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 14 more variables: avgAnnualInc <dbl>,
## #   avgVerifStatus <dbl>, avgHomeOwner <dbl>, avgOpenAcc <dbl>,
## #   avgRevolBal <dbl>, avgRevolUtil <dbl>, avgTotalAcc <dbl>,
## #   countOfLoans <dbl>, `Total population` <dbl>, loansPerCapita <dbl>,
## #   NYCPI <dbl>, NYUnemployment <dbl>, NYCondoPriceIdx <dbl>,
## #   NYSnapBenefits <dbl>
```

Question 2

```
# 2.a

top_10 <- filter(tsLC, tsLC$`Total population` > quantile(tsLC$`Total
population`, probs = 0.90))
bottom_10 <- filter(tsLC, tsLC$`Total population` < quantile(tsLC$`Total
population`, probs = 0.10))

boxLoansPerCapita_top10 <- top_10 %>%
  ggplot(aes(x = state, y = loansPerCapita)) +
  geom_boxplot() + ggtitle("Top 10th percentile")

boxLoansPerCapita_bottom10 <- bottom_10 %>%
  ggplot(aes(x = state, y = loansPerCapita)) +
  geom_boxplot() + ggtitle("Bottom 10th percentile")

boxLoansPerCapita_top10
```



```
boxLoansPerCapita_bottom10
```



```
# 2.b

tsLC_NY <- tsLC %>%
  filter(state == "NY")

tsLC_NY$avgOpenAcc[which(is.na(tsLC_NY$avgOpenAcc))] <-
mean(tsLC_NY$avgOpenAcc, na.rm = TRUE)
tsLC_NY$avgRevolUtil[which(is.na(tsLC_NY$avgRevolUtil))] <-
mean(tsLC_NY$avgRevolUtil, na.rm = TRUE)
tsLC_NY$avgTotalAcc[which(is.na(tsLC_NY$avgTotalAcc))] <-
mean(tsLC_NY$avgTotalAcc, na.rm = TRUE)

tsLC_NY

## # 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>, `Total population` <dbl>, loansPerCapita <dbl>,
## #   NYCPI <dbl>, NYUnemployment <dbl>, NYCondoPriceIdx <dbl>,
## #   NYSnapBenefits <dbl>
```

```
tsLC_CO <- tsLC %>%
  filter(state == "CO")

tsLC_CO$avgOpenAcc[which(is.na(tsLC_CO$avgOpenAcc))] <-
mean(tsLC_CO$avgOpenAcc, na.rm = TRUE)
tsLC_CO$avgRevolUtil[which(is.na(tsLC_CO$avgRevolUtil))] <-
mean(tsLC_CO$avgRevolUtil, na.rm = TRUE)
tsLC_CO$avgTotalAcc[which(is.na(tsLC_CO$avgTotalAcc))] <-
mean(tsLC_CO$avgTotalAcc, na.rm = TRUE)

tsLC_CO
```

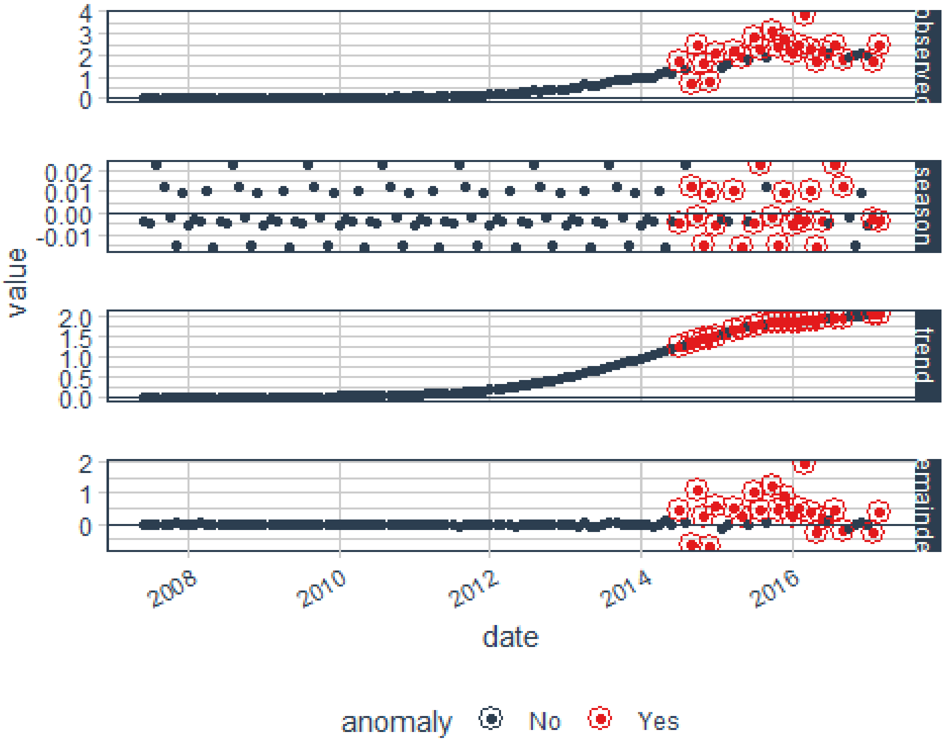
```
## # A tsibble: 117 x 22 [1D]
## # Key:       state [1]
##    date       state avgLoans totalLoans avgTerm avgIntRate avgGrade
avgEmpLength
##    <date>     <chr>    <dbl>      <dbl>   <dbl>      <dbl>    <dbl>
<dbl>
##  1 2007-06-01 CO        2600       2600      36       8.38     1
3
##  2 2007-07-01 CO        5833.     17500      36       9.02     2
3.33
##  3 2007-09-01 CO       13150      13150      36      17.5      7
7
##  4 2007-10-01 CO        5000      10000      36      11.7      3.5
1
##  5 2007-11-01 CO        5792.     17375      36      13.9      5
1.67
##  6 2007-12-01 CO        9511.     85600      36      13.2      4.44
```

```
2.22
##  7 2008-01-01 CO       7888.     102550      36      12.0      3.54
2.69
##  8 2008-02-01 CO       7358.      73575      36      10.9      2.7
3
##  9 2008-03-01 CO       8162.      65300      36      11.7      3.12
7.12
## 10 2008-04-01 CO       6050       24200      36      10.5      2.5
4.75
## # ... with 107 more rows, and 14 more variables: avgAnnualInc <dbl>,
## #   avgVerifStatus <dbl>, avgHomeOwner <dbl>, avgOpenAcc <dbl>,
## #   avgRevolBal <dbl>, avgRevolUtil <dbl>, avgTotalAcc <dbl>,
## #   countOfLoans <dbl>, `Total population` <dbl>, loansPerCapita <dbl>,
## #   NYCPI <dbl>, NYUnemployment <dbl>, NYCondoPriceIdx <dbl>,
## #   NYSnapBenefits <dbl>

tsLC_MA <- tsLC %>%
  filter(state == "MA")

tsLC_MA$avgOpenAcc[which(is.na(tsLC_MA$avgOpenAcc))] <-
mean(tsLC_MA$avgOpenAcc, na.rm = TRUE)
tsLC_MA$avgRevolUtil[which(is.na(tsLC_MA$avgRevolUtil))] <-
mean(tsLC_MA$avgRevolUtil, na.rm = TRUE)
tsLC_MA$avgTotalAcc[which(is.na(tsLC_MA$avgTotalAcc))] <-
mean(tsLC_MA$avgTotalAcc, na.rm = TRUE)

tsLC_MA

## # A tsibble: 117 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 MA       3194.      12775      36       11.3     3.25
1
##  2 2007-07-01 MA       3790       37900      36       10.1     2.5
3.2
##  3 2007-08-01 MA       6004.      36025      36       10.6     2.83
1
##  4 2007-09-01 MA       7750       46500      36       11.1     3
5.67
##  5 2007-10-01 MA       7046.      49325      36       10.9     2.86
1.43
##  6 2007-11-01 MA       8680       86800      36       11.1     3
2.6
##  7 2007-12-01 MA       7783.      70050      36       11.0     3.11
2.89
##  8 2008-01-01 MA       8036.      56250      36       10.9     2.86
4.43
```

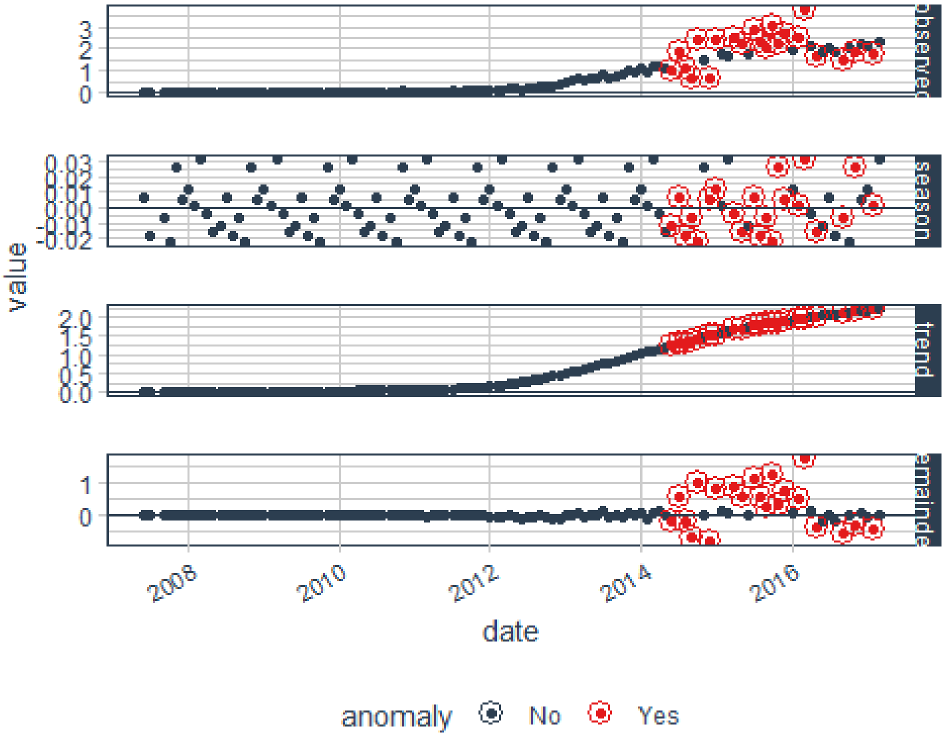
```
##  9 2008-02-01 MA       8184.      65475      36        13.2      4
5.62
## 10 2008-03-01 MA       9750      107250      36        11.7      3
4.27
## # ... with 107 more rows, and 14 more variables: avgAnnualInc <dbl>,
## #   avgVerifStatus <dbl>, avgHomeOwner <dbl>, avgOpenAcc <dbl>,
## #   avgRevolBal <dbl>, avgRevolUtil <dbl>, avgTotalAcc <dbl>,
## #   countOfLoans <dbl>, `Total population` <dbl>, loansPerCapita <dbl>,
## #   NYCPI <dbl>, NYUnemployment <dbl>, NYCondoPriceIdx <dbl>,
## #   NYSnapBenefits <dbl>
```

```
tsLC_NY %>%
  time_decompose(loansPerCapita, method = "stl", frequency = "auto", trend =
"auto") %>%
  anomalize(remainder, method = "iqr", alpha = 0.05, max_anoms = 0.2) %>%
  plot_anomaly_decomposition()
```

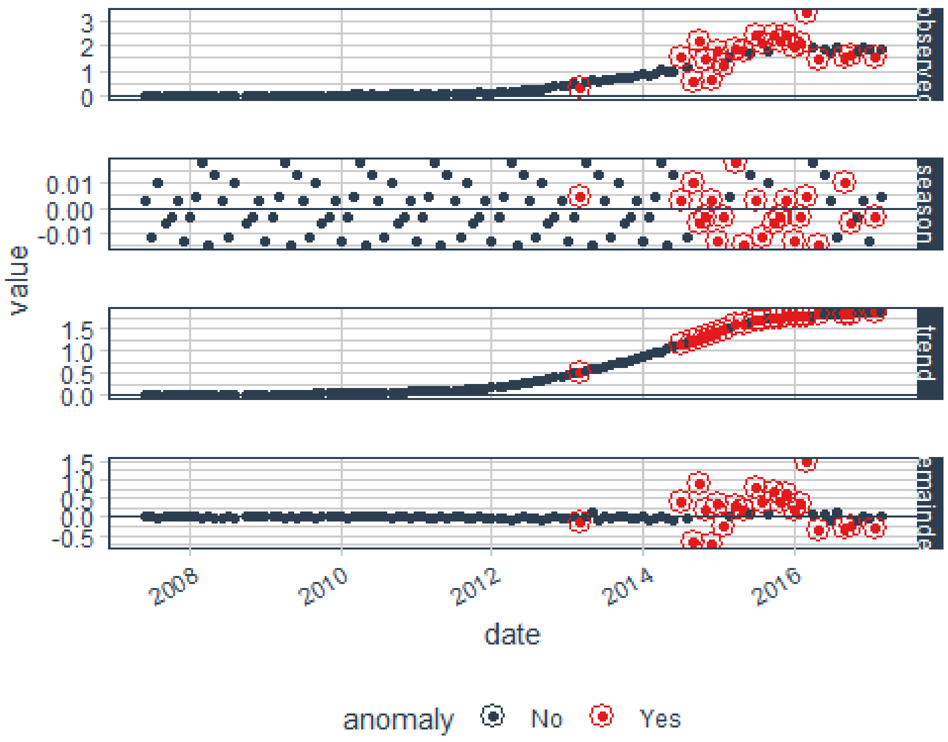
```
## Converting from tbl_ts to tbl_time.
## Auto-index message: index = date

## frequency = 12 months

## trend = 31 months

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



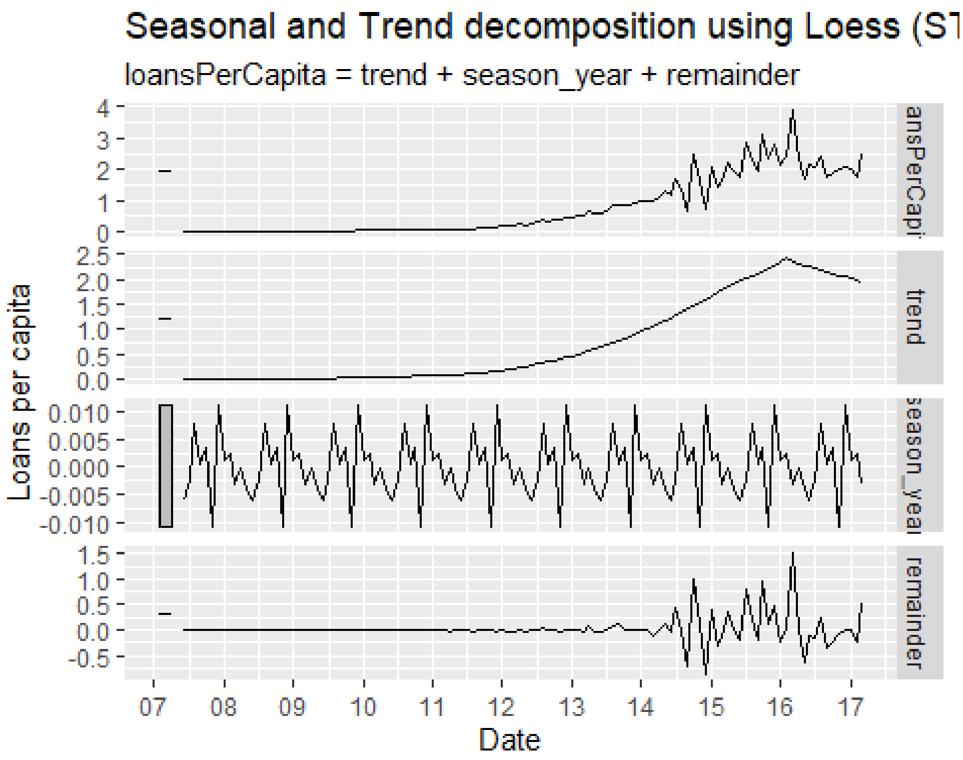
```
tsLC_CO %>%
  time_decompose(loansPerCapita, method = "stl", frequency = "auto", trend =
"auto") %>%
  anomalize(remainder, method = "iqr", alpha = 0.05, max_anoms = 0.2) %>%
  plot_anomaly_decomposition()

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

## frequency = 12 months

## trend = 30 months
```



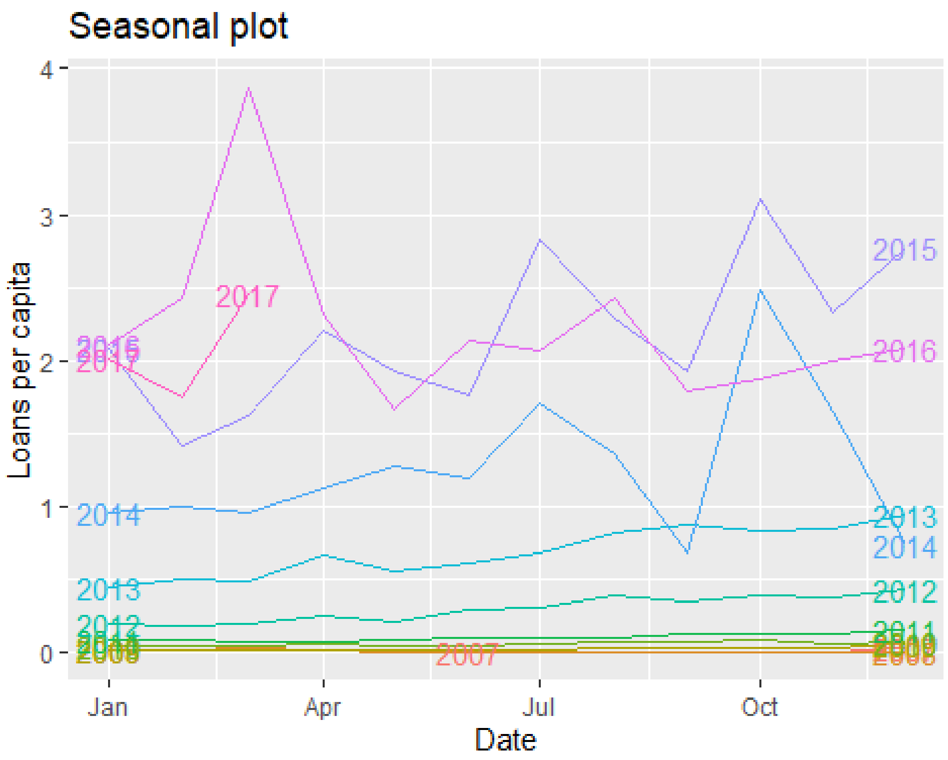
```
tsLC_MA %>%
  time_decompose(loansPerCapita, method = "stl", frequency = "auto", trend =
"auto") %>%
  anomalize(remainder, method = "iqr", alpha = 0.05, max_anoms = 0.2) %>%
  plot_anomaly_decomposition()

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

## frequency = 12 months

## trend = 30 months
```



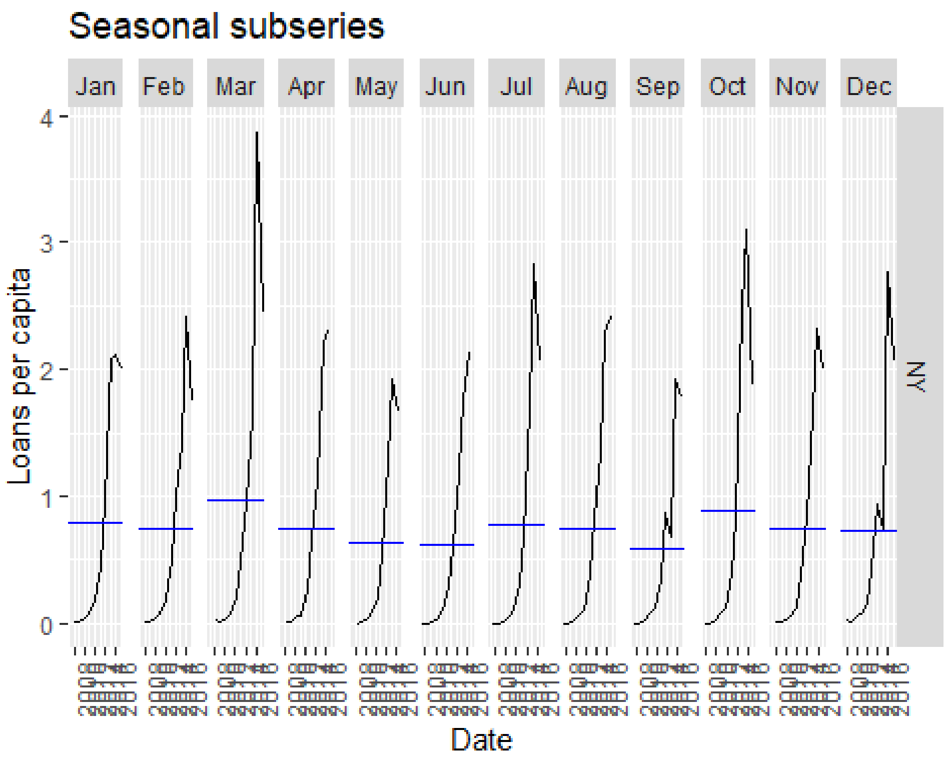
```
# 2.c

tsLC_NY$date<- yearmonth(tsLC_NY$date)

tsLC_NY_Decomposed <- tsLC_NY %>%
  select(date, loansPerCapita) %>%
  model(STL(loansPerCapita ~ trend() + season(window = "periodic"), robust =
TRUE)) %>%
  components() %>%
  autoplot() +
  xlab("Date") + ylab("Loans per capita") +
  ggtitle("Seasonal and Trend decomposition using Loess (STL decomposition)")
+
  scale_x_date(date_breaks = "years" , date_labels = "%y")

ggplotly(tsLC_NY_Decomposed)

tsLC_NY_Decomposed
```



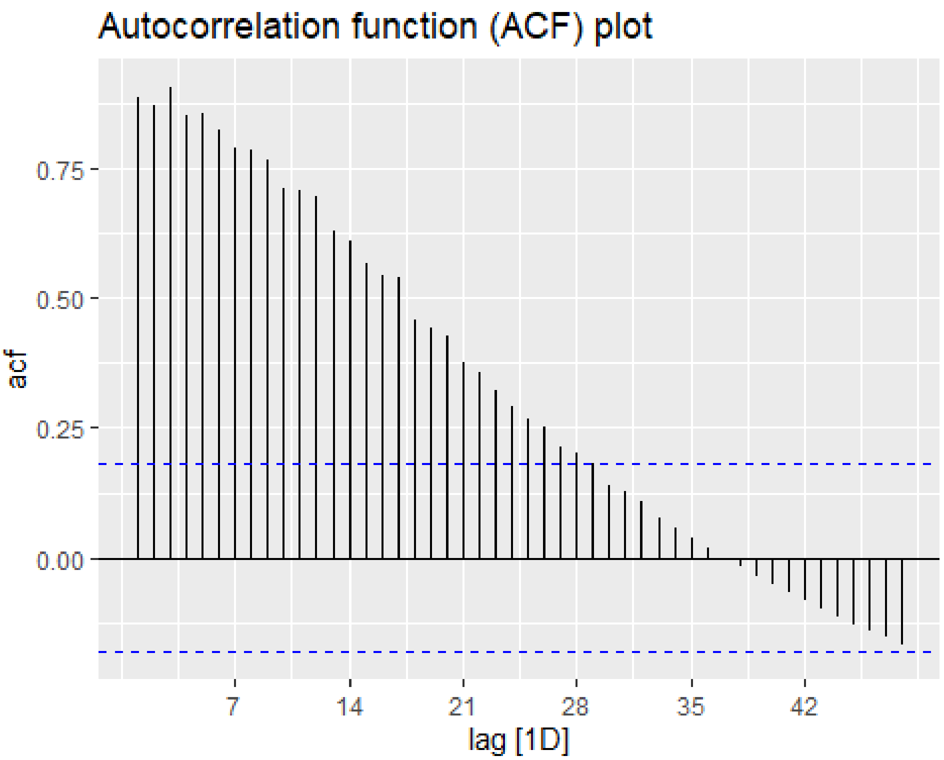
```r
# 2.d

tsLC_NY_seasonalPlot <- tsLC_NY %>%
  gg_season(loansPerCapita, labels = "both") +
  xlab("Date") + ylab("Loans per capita") +
  ggtitle("Seasonal plot")

tsLC_NY_seasonalPlot
```



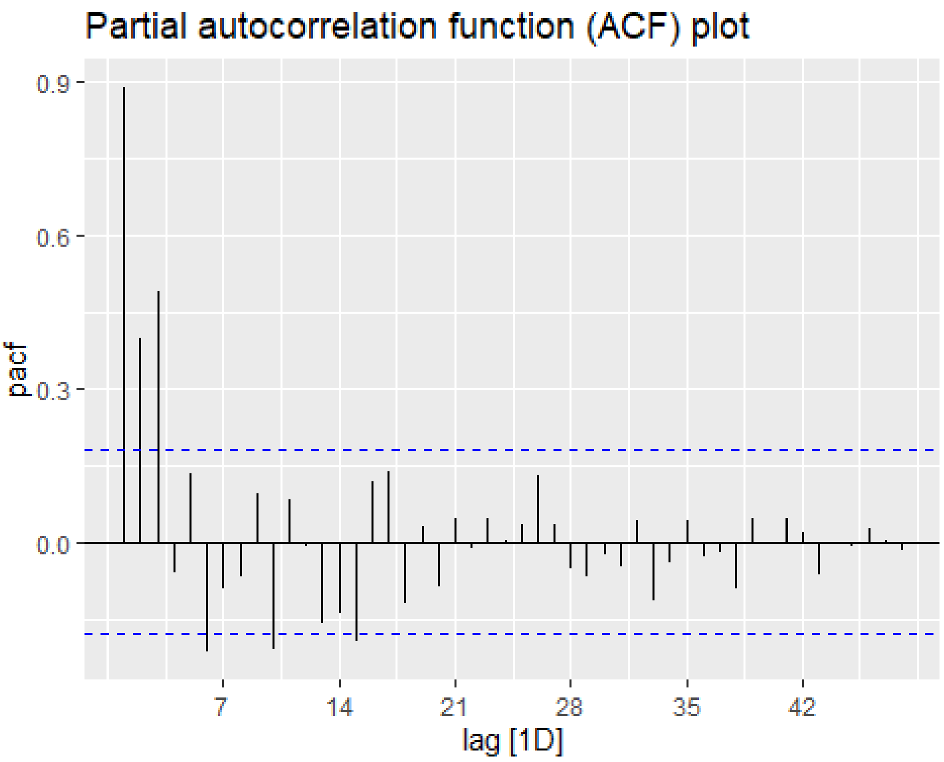
```
tsLC_NY_seasonalSubseries <- tsLC_NY %>%
  gg_subseries(loansPerCapita) +
  ylab("Loans per capita") +
  xlab("Date") +
  ggtitle("Seasonal subseries")

tsLC_NY_seasonalSubseries
```



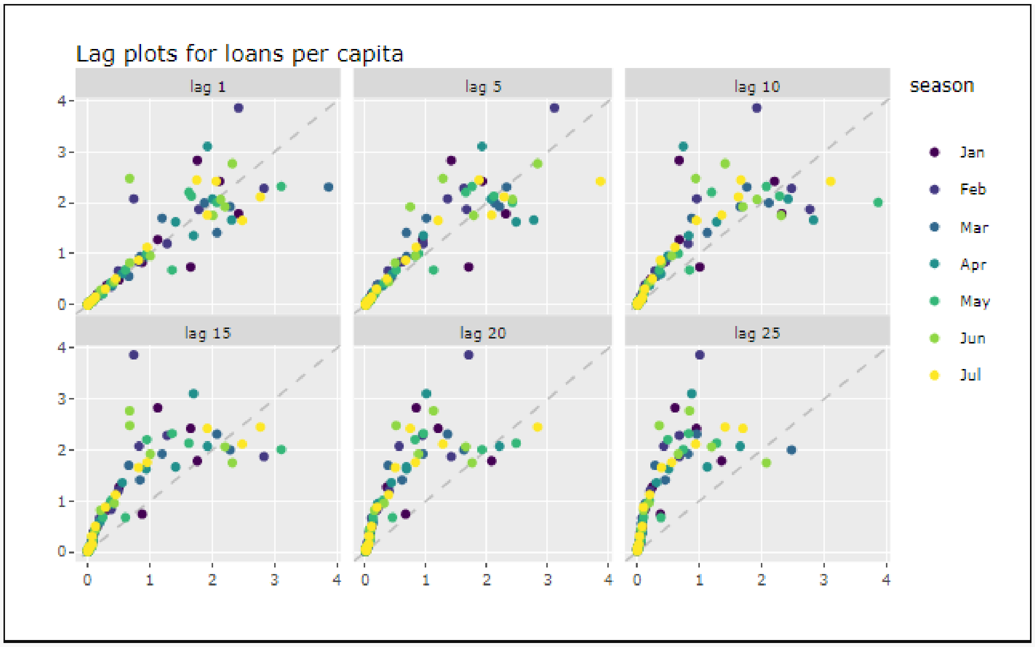
```
# 2.e

tsLC_NY_ACF <- tsLC_NY %>%
  ACF(loansPerCapita, lag_max = 48) %>%
  autoplot() + ggtitle("Autocorrelation function (ACF) plot")

tsLC_NY_ACF
```



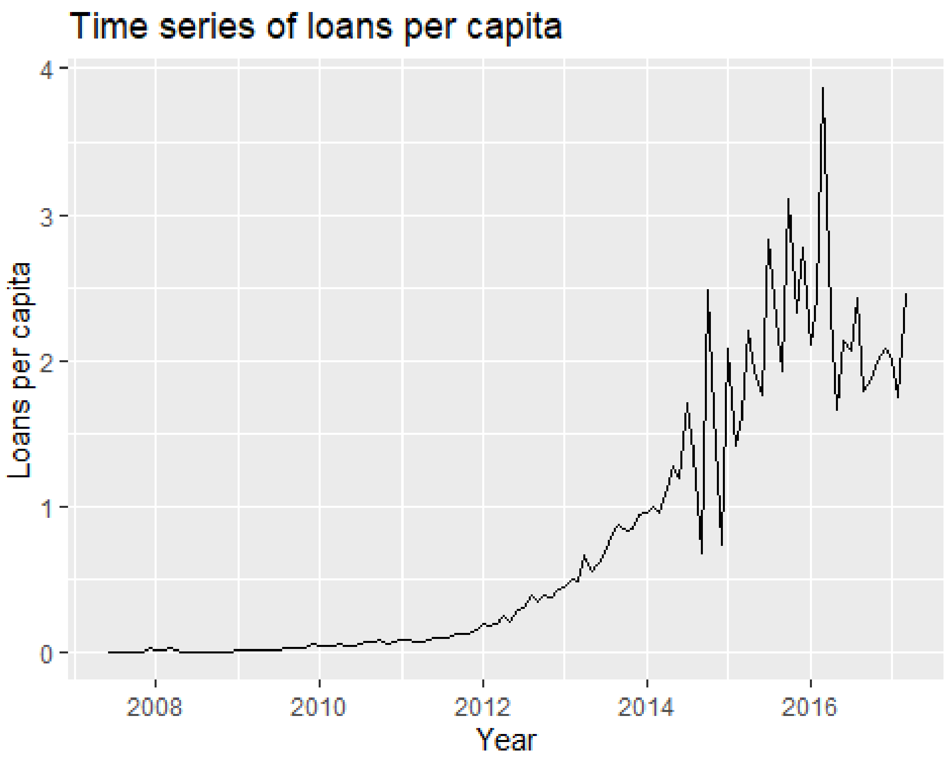
```
tsLC_NY_PACF <- tsLC_NY %>%
  PACF(loansPerCapita, lag_max = 48) %>%
  autoplot() + ggtitle("Partial autocorrelation function (ACF) plot")

tsLC_NY_PACF
```



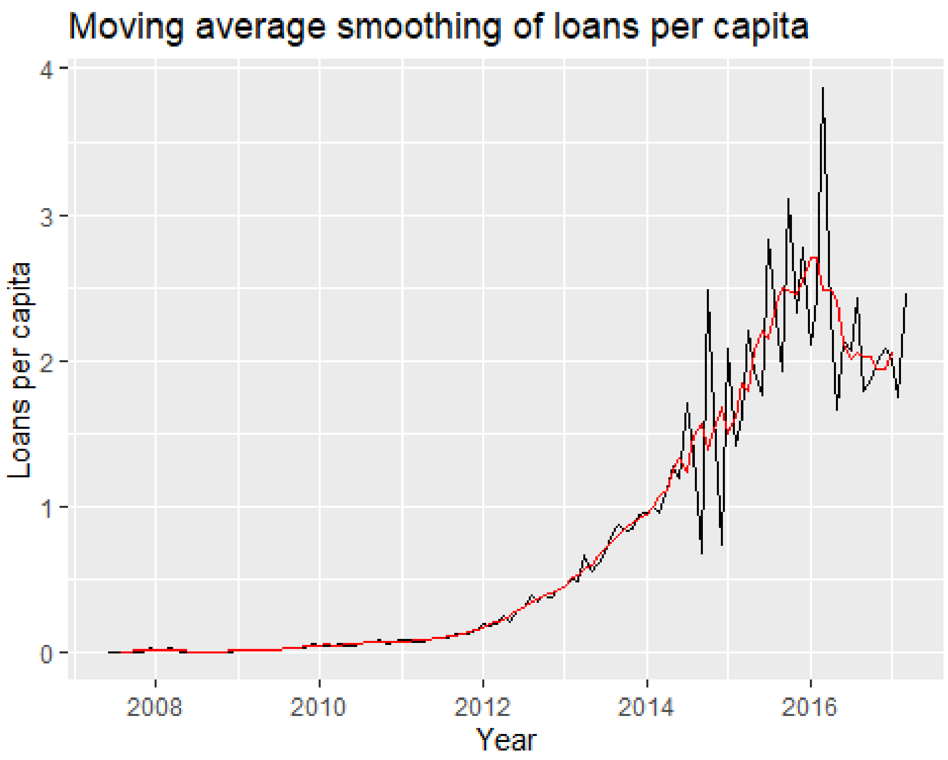
```
# 2.f

tsLC_NY_Lag <- tsLC_NY %>%
  gg_lag(loansPerCapita, lags = c(1, 5, 10, 15, 20, 25), geom = "point") +
  xlab(NULL) + ylab(NULL) +
  ggtitle("Lag plots for loans per capita")

ggplotly(tsLC_NY_Lag)
```



```
# 2.g

library("forecast")

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

##
## Attaching package: 'forecast'

## The following objects are masked from 'package:fabletools':
##
##     GeomForecast, StatForecast

## The following object is masked from 'package:yardstick':
##
##     accuracy

tsLC_NY2 <- tsLC_NY %>%
  select("date", "loansPerCapita")

autoplot(tsLC_NY2) +
xlab("Year") + ylab("Loans per capita") +
ggtitle("Time series of loans per capita")

## Plot variable not specified, automatically selected `.vars =
loansPerCapita`
```

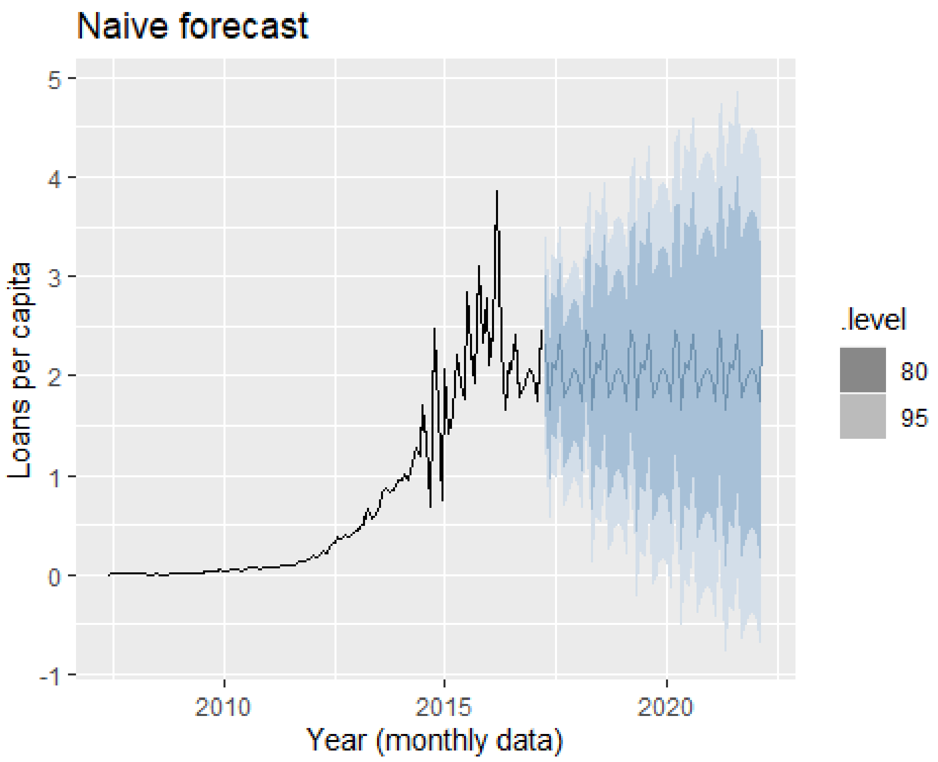


```
tsLC_NY_ma <- tsLC_NY2 %>%
  mutate(`5-MA` = slide_dbl(loansPerCapita, mean, .size = 5, .align =
"center"))

tsLC_NY_ma %>%
  autoplot(loansPerCapita) +
  autolayer(tsLC_NY_ma, `5-MA`, color='red') +
  xlab("Year") + ylab("Loans per capita") +
  ggtitle("Moving average smoothing of loans per capita") +
  guides(colour = guide_legend(title = "series"))

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

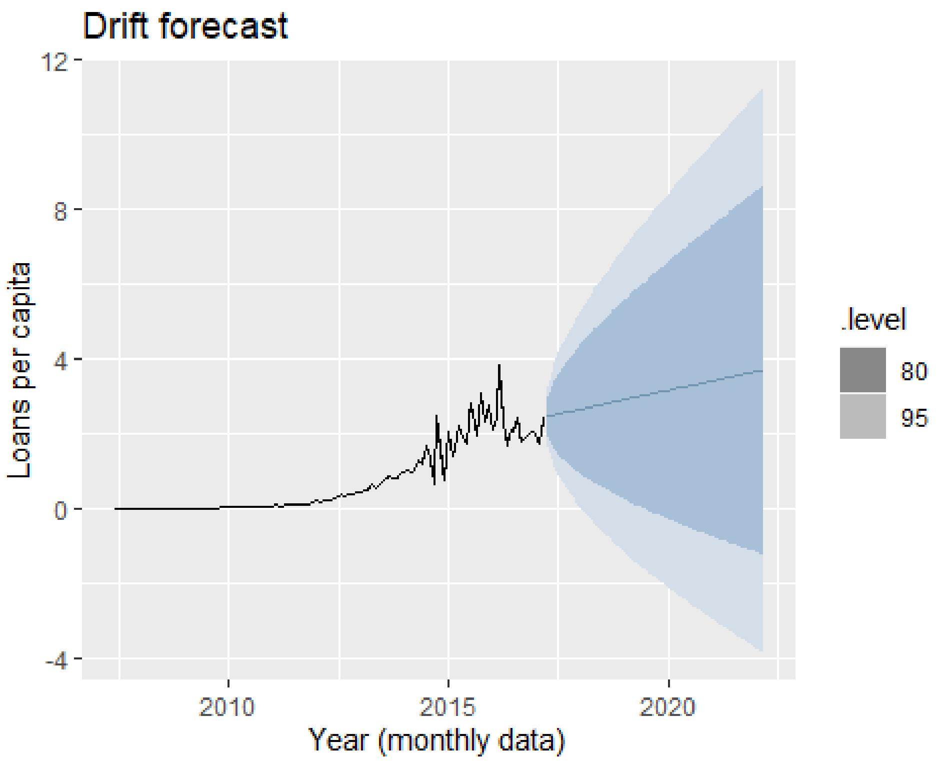


# Question 3

```
# 3.a

tsLC_NY_Naive <- tsLC_NY2 %>%
  model(SNAIVE(loansPerCapita)) %>%
  forecast(h = "5 years") %>%
  autoplot(tsLC_NY2, colour = "#769ECB") +
  geom_line(linetype = 'dashed', colour = '#000000') +
  xlab("Year (monthly data)") + ylab("Loans per capita") +
  ggtitle("Naive forecast")

tsLC_NY_Naive
```

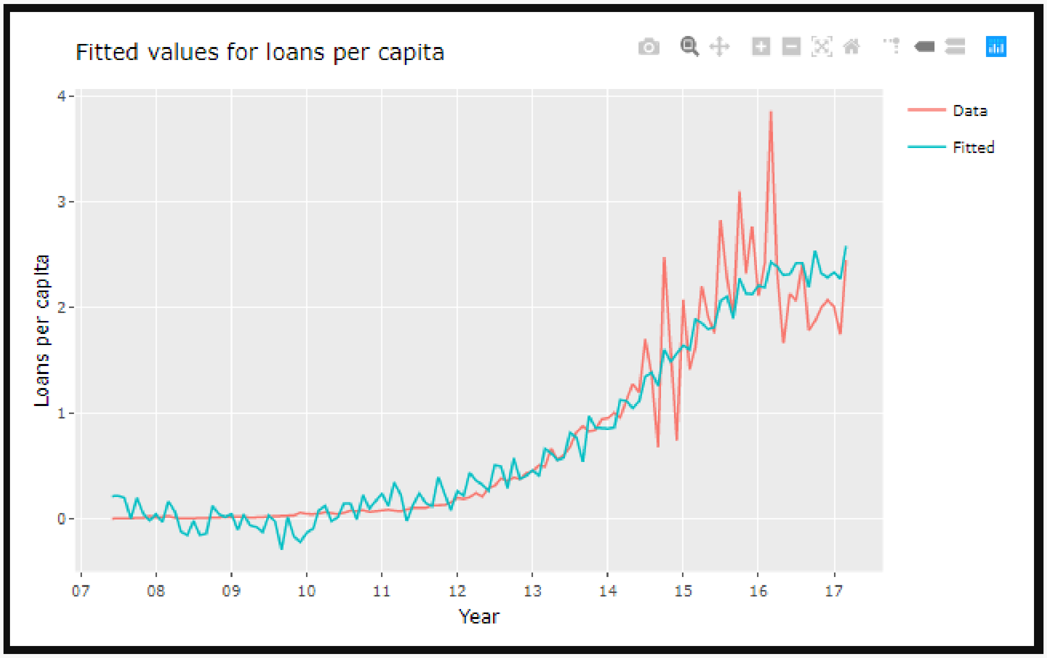


```
tsLC_NY_Drift <- tsLC_NY2 %>%
  model(RW(loansPerCapita ~ drift())) %>%
  forecast(h = "5 years") %>%
  autoplot(tsLC_NY2, colour = "#769ECB") +
  geom_line(linetype = 'dashed', colour = '#000000') +
  xlab("Year (monthly data)") + ylab("Loans per capita") +
  ggtitle("Drift forecast")

tsLC_NY_Drift
```

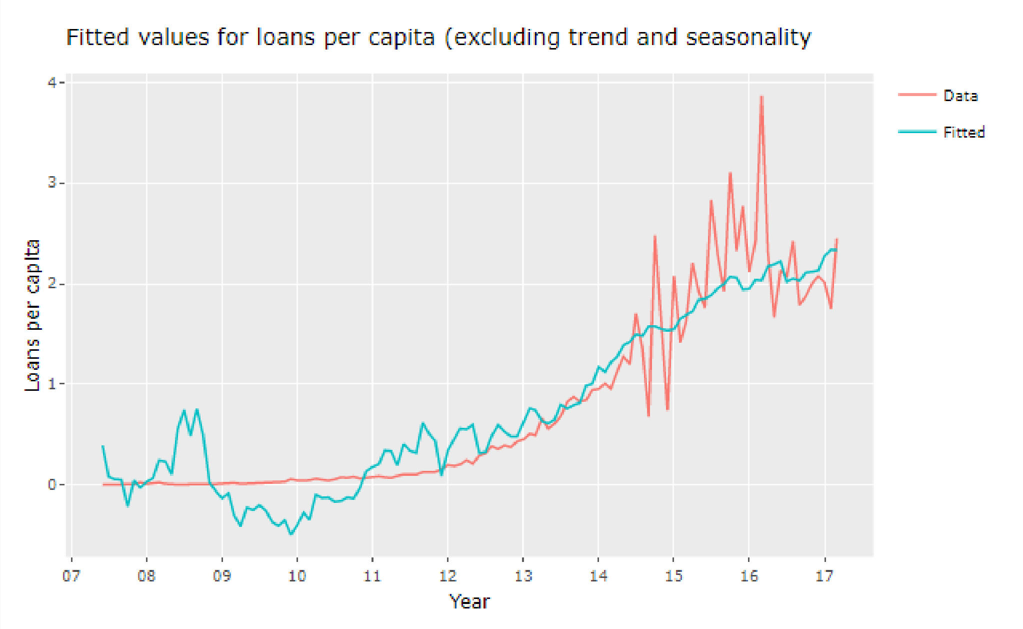


```
# 3.b

fit_tsLC_NY <- tsLC_NY %>%
  model(TSLM(loansPerCapita ~ trend() + season() + avgTerm + avgIntRate +
avgAnnualInc + avgVerifStatus + NYCPI + NYUnemployment))

report(fit_tsLC_NY)

## Series: loansPerCapita
## Model: TSLM
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.82573 -0.13762 -0.01711  0.09748  1.43227
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)       1.335e+01  3.256e+00   4.101 8.44e-05 ***
## trend()           3.885e-02  4.951e-03   7.847 5.10e-12 ***
## season()year2    -5.769e-02  1.386e-01  -0.416  0.67812
## season()year3     1.990e-01  1.390e-01   1.431  0.15555
## season()year4     1.234e-01  1.428e-01   0.864  0.38942
## season()year5     5.527e-02  1.450e-01   0.381  0.70398
## season()year6     6.689e-02  1.415e-01   0.473  0.63754
## season()year7     2.228e-01  1.413e-01   1.577  0.11794
## season()year8     1.940e-01  1.415e-01   1.371  0.17345
## season()year9    -2.792e-02  1.407e-01  -0.198  0.84316
## season()year10  2.421e-01  1.395e-01   1.735  0.08588 .
## season()year11  2.076e-02  1.384e-01   0.150  0.88111
## season()year12 -5.088e-02  1.397e-01  -0.364  0.71653
## avgTerm        -2.557e-02  1.562e-02  -1.637  0.10487
## avgIntRate     -6.550e-02  3.766e-02  -1.739  0.08506 .
## avgAnnualInc    6.767e-07  2.740e-06   0.247  0.80542
## avgVerifStatus  1.612e-01  2.324e-01   0.693  0.48964
## NYCPI          -1.679e-02  5.353e-03  -3.136  0.00225 **
## NYUnemployment -1.666e-01  2.579e-02  -6.459 3.97e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3084 on 99 degrees of freedom
## Multiple R-squared: 0.9051,  Adjusted R-squared: 0.8878
## F-statistic: 52.46 on 18 and 99 DF, p-value: < 2.22e-16
```

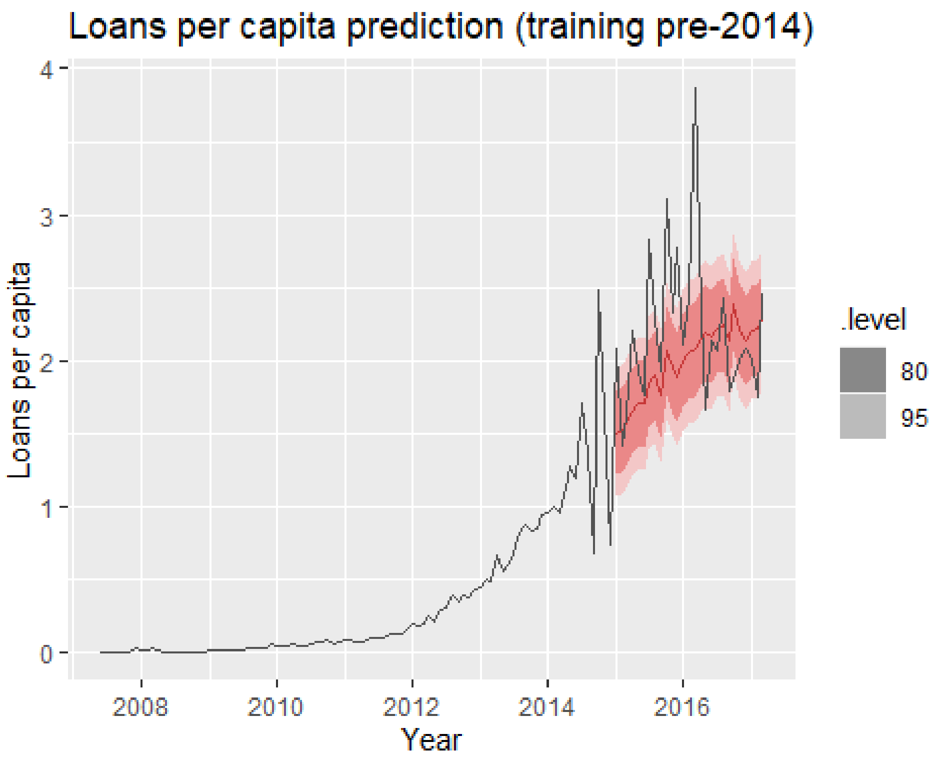
```
# 3.c

tsLC_NY_Fitted <- augment(fit_tsLC_NY) %>%
  ggplot(aes(x = date)) +
  geom_line(aes(y = loansPerCapita, colour = "Data")) +
  geom_line(aes(y = .fitted, colour = "Fitted")) +
  xlab("Year") + ylab("Loans per capita") +
  ggtitle("Fitted values for loans per capita") +
  scale_x_date(date_breaks = "years" , date_labels = "%y") +
  guides(colour = guide_legend(title = NULL))

ggplotly(tsLC_NY_Fitted)
```



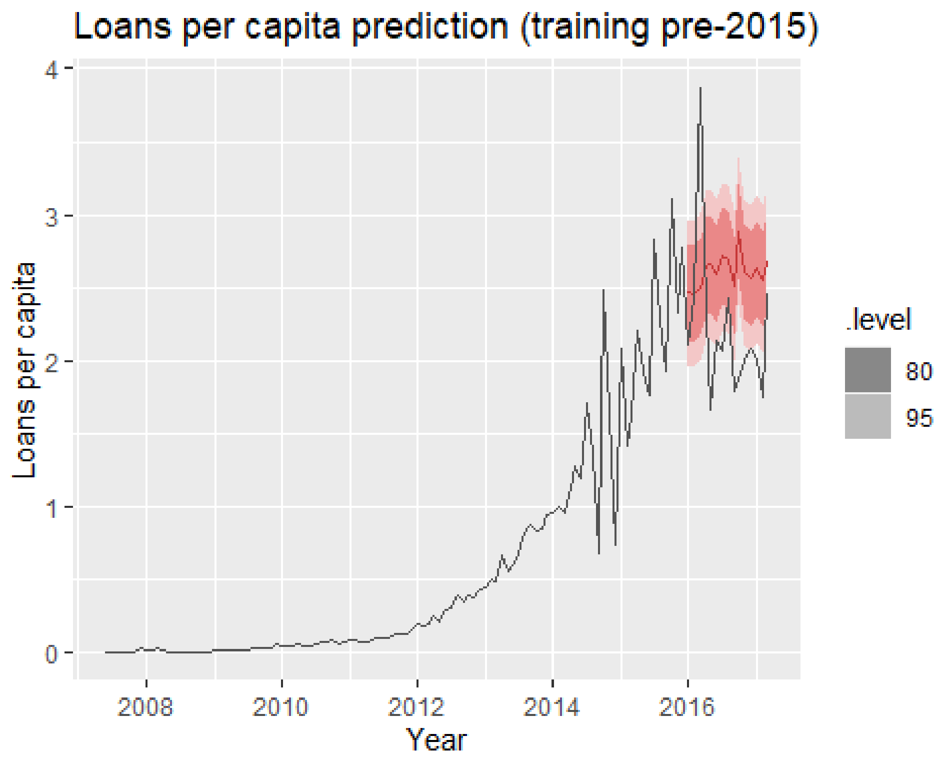
```r
fit_tsLC_NY1 <- tsLC_NY %>%
  model(TSLM(loansPerCapita ~ avgTerm + avgIntRate + avgAnnualInc +
avgVerifStatus + NYCPI + NYUnemployment))

report(fit_tsLC_NY1)

## Series: loansPerCapita
## Model: TSLM
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.89858 -0.22235 -0.05252  0.19183  1.83780
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)    -1.057e+01  1.408e+00  -7.505 1.65e-11 ***
## avgTerm        -3.193e-02  1.914e-02  -1.668  0.09809 .
## avgIntRate     -1.415e-01  4.412e-02  -3.208  0.00175 **
## avgAnnualInc   -9.565e-07  3.333e-06  -0.287  0.77467
## avgVerifStatus  4.609e-01  2.820e-01   1.635  0.10498
## NYCPI           2.205e-02  2.388e-03   9.235 2.07e-15 ***
## NYUnemployment -2.428e-01  2.981e-02  -8.145 6.19e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3857 on 111 degrees of freedom
## Multiple R-squared: 0.8336,  Adjusted R-squared: 0.8246
## F-statistic: 92.68 on 6 and 111 DF, p-value: < 2.22e-16

tsLC_NY_Fitted1 <- augment(fit_tsLC_NY1) %>%
  ggplot(aes(x = date)) +
  geom_line(aes(y = loansPerCapita, colour = "Data")) +
  geom_line(aes(y = .fitted, colour = "Fitted")) +
  xlab("Year") + ylab("Loans per capita") +
  ggtitle("Fitted values for loans per capita (excluding trend and
seasonality") +
  scale_x_date(date_breaks = "years" , date_labels = "%y") +
  guides(colour = guide_legend(title = NULL))

ggplotly(tsLC_NY_Fitted1)
```



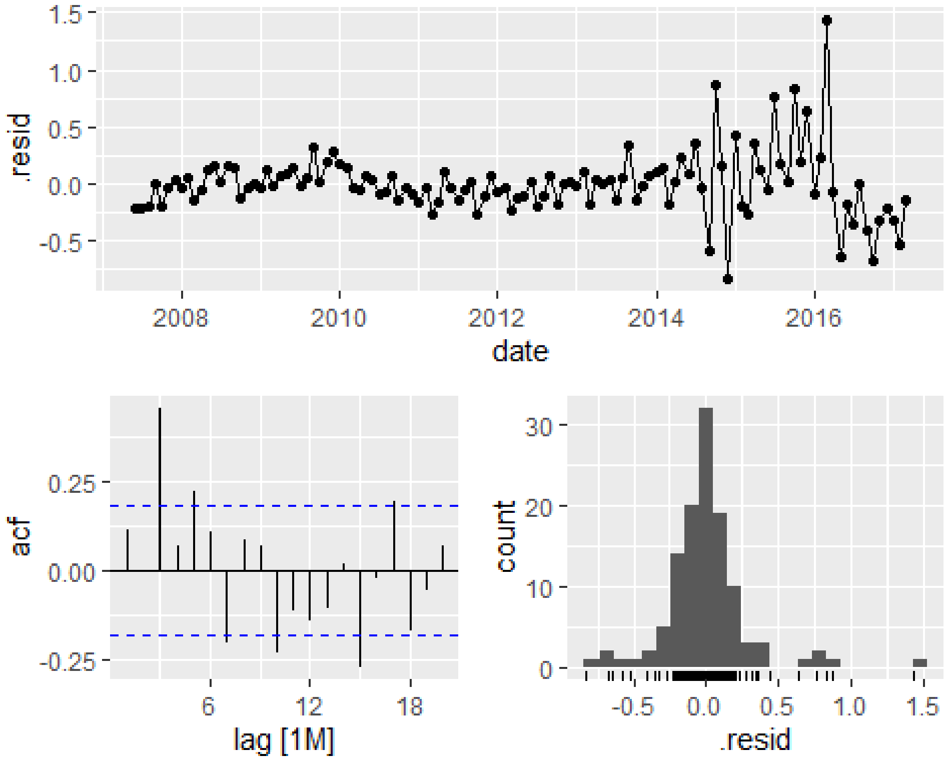
```
# 3.d

tsLC_NY_Predicted <- tsLC_NY %>%
  filter(date < "2015-01-01") %>%
  model(TSLM(loansPerCapita ~ trend() + season() + avgTerm + avgIntRate +
avgAnnualInc + avgVerifStatus + NYCPI + NYUnemployment)) %>%
  forecast(new_data = tsLC_NY %>% filter(date >= "2015-01-01")) %>%
  autoplot(tsLC_NY, colour = "#960A0A") +
  geom_line(colour = '#535353') +
  xlab("Year") + ylab("Loans per capita") +
  ggtitle("Loans per capita prediction (training pre-2014)")

tsLC_NY_Predicted
```



```
tsLC_NY_Predicted1 <- tsLC_NY %>%
  filter(date < "2016-01-01") %>%
  model(TSLM(loansPerCapita ~ trend() + season() + avgTerm + avgIntRate +
avgAnnualInc + avgVerifStatus + NYCPI + NYUnemployment)) %>%
  forecast(new_data = tsLC_NY %>% filter(date >= "2016-01-01")) %>%
  autoplot(tsLC_NY, colour = "#960A0A") +
  geom_line(colour = '#535353') +
  xlab("Year") + ylab("Loans per capita") +
  ggtitle("Loans per capita prediction (training pre-2015)")

tsLC_NY_Predicted1
```



```
# 3.e

fit_tsLC_NY %>% gg_tsresiduals()
```



```r
# 3.f

fit_tsLC_NY_ARIMA <- tsLC_NY %>%
  model(fitArima = ARIMA(loansPerCapita ~ PDQ(0,0,0) + avgTerm + avgIntRate +
avgAnnualInc + avgVerifStatus + NYCPI + NYUnemployment, stepwise = FALSE,
approximation = FALSE))

report(fit_tsLC_NY_ARIMA)

## Series: loansPerCapita
## Model: LM w/ ARIMA(2,0,3) errors
##
## Coefficients:
##          ar1     ar2      ma1      ma2    ma3  avgTerm  avgIntRate
##       0.6791  0.3062  -0.4465  -0.5304  0.6587   0.0047      0.0126
## s.e.  0.1170  0.1164   0.0963   0.0649  0.0639   0.0173      0.0330
##       avgAnnualInc  avgVerifStatus   NYCPI  NYUnemployment
##                  0         -0.1048  0.0018         -0.1264
## s.e.             0          0.1619  0.0012          0.0878
##
## sigma^2 estimated as 0.06785:  log likelihood=-5.38
## AIC=34.76   AICc=37.73   BIC=68

# 3.g

tsLC_NY %>%
  features(loansPerCapita, unitroot_kpss)

## # A tibble: 1 x 3
##   state kpss_stat kpss_pvalue
##   <chr>     <dbl>       <dbl>
## 1 NY         2.09        0.01

tsLC_NY %>%
  features(loansPerCapita, unitroot_ndiffs)

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

tsLC_NY %>%
  features(difference(loansPerCapita), unitroot_kpss)

## # A tibble: 1 x 3
##   state kpss_stat kpss_pvalue
##   <chr>     <dbl>       <dbl>
## 1 NY        0.129         0.1

fit_tsLC_NY_ARIMA1 <- tsLC_NY %>%
  model(fitArima = ARIMA(loansPerCapita ~ PDQ(0,0,0) + pdq(2,1,3) + avgTerm +
avgIntRate + avgAnnualInc + avgVerifStatus + NYCPI + NYUnemployment, stepwise
= FALSE, approximation = FALSE))

report(fit_tsLC_NY_ARIMA1)
```

```
## Series: loansPerCapita
## Model: LM w/ ARIMA(2,1,3) errors
##
## Coefficients:
##           ar1      ar2      ma1      ma2     ma3  avgTerm  avgIntRate
##       -0.4875  -0.2298  -0.3113  -0.5041  0.5506   0.0031      0.0029
## s.e.   0.1425   0.1319   0.1286   0.0758  0.1011   0.0160      0.0310
##       avgAnnualInc  avgVerifStatus    NYCPI  NYUnemployment  intercept
##                  0         -0.0979  -0.0042         -0.1542     0.0226
## s.e.             0          0.1551   0.0013          0.0753     0.0100
##
## sigma^2 estimated as 0.06496:  log likelihood=-1.04
## AIC=28.08   AICc=31.62   BIC=63.99
```

Question 4

```
# 4.a

set.seed(333)
tsLC_NY_Train <- tsLC_NY %>% filter(date < "2016-03-01")
tsLC_NY_Test <- tsLC_NY %>% filter(date >= "2016-03-01")

tsLC_NY_FitAll <- tsLC_NY_Train %>%
  model(
  model1TimeTrendAndSeason = TSLM(loansPerCapita ~ trend() + season()),
  model2_fit_tsLC_NY = TSLM(loansPerCapita ~ trend() + season() + avgTerm +
avgIntRate + avgAnnualInc + avgVerifStatus + NYCPI + NYUnemployment),
  model3ArimaGrid = ARIMA(loansPerCapita ~ PDQ(0,0,0), stepwise = FALSE,
approximation = FALSE),
  model4fit_tsLC_NY_ARIMA = ARIMA(loansPerCapita ~ PDQ(0,0,0) + avgTerm +
avgIntRate + avgAnnualInc + avgVerifStatus + NYCPI + NYUnemployment, stepwise
= FALSE, approximation = FALSE))

tsLC_NY_PredictAll <- tsLC_NY_FitAll %>%
  forecast(new_data = tsLC_NY_Test)

accuracy(tsLC_NY_PredictAll, tsLC_NY_Test)
```

```
## # 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 model1TimeTrendAnd~ NY    Test   0.341 0.702 0.406  11.4    15.1   NaN
```

```
0.137
## 2 model2_fit_tsLC_NY  NY    Test  -0.420 0.713 0.635 -24.6    30.2   NaN
0.0696
## 3 model3ArimaGrid     NY    Test  -0.439 0.648 0.595 -24.8    29.3   NaN
0.240
## 4 model4fit_tsLC_NY_~ NY    Test   0.105 0.501 0.338   0.964  14.4   NaN -
0.0693

# 4.b

set.seed(333)
tsLC_NY_Train1 <- tsLC_NY %>% filter(date < "2016-04-01")
tsLC_NY_Test1 <- tsLC_NY %>% filter(date >= "2016-04-01")

tsLC_NY_FitAll1 <- tsLC_NY_Train1 %>%
  model(
  model1TimeTrendAndSeason1 = TSLM(loansPerCapita ~ trend() + season()),
  model2_fit_tsLC_NY1 = TSLM(loansPerCapita ~ trend() + season() + avgTerm +
avgIntRate + avgAnnualInc + avgVerifStatus + NYCPI + NYUnemployment),
  model3ArimaGrid1 = ARIMA(loansPerCapita ~ PDQ(0,0,0), stepwise = FALSE,
approximation = FALSE),
  model4fit_tsLC_NY_ARIMA1 = ARIMA(loansPerCapita ~ PDQ(0,0,0) + avgTerm +
avgIntRate + avgAnnualInc + avgVerifStatus + NYCPI + NYUnemployment, stepwise
= FALSE, approximation = FALSE))

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

tsLC_NY_PredictAll1 <- tsLC_NY_FitAll1 %>%
  forecast(new_data = tsLC_NY_Test1)

accuracy(tsLC_NY_PredictAll1, tsLC_NY_Test1)

## # 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 model1TimeTrendAnd~ NY    Test   0.0944 0.268 0.207   3.43  9.73   NaN -
0.185
## 2 model2_fit_tsLC_NY1 NY    Test  -0.684  0.723 0.684 -35.2  35.2    NaN -
0.332
## 3 model3ArimaGrid1    NY    Test  -1.14   1.27  1.16  -58.4  59.0    NaN
0.0519
## 4 model4fit_tsLC_NY_~ NY    Test  -0.191  0.379 0.314 -10.6  15.8    NaN -
0.0448
```

Part 2

Question 1

```r
# 1.a

tsRetail <- read_csv("retailSales.csv")
```

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

```r
tsRetail$date <- mdy(tsRetail$date)
tsRetail$date<- yearmonth(tsRetail$date)
skim(tsRetail)
```

Data summary

Name	tsRetail
Number of rows	338
Number of columns	2
_______________________	
Column type frequency:	
Date	1
numeric	1
________________________	
Group variables	None

Variable type: Date

skim_variable	n_missing	complete_rate	min	max	median	n_unique
date	0	1	1992 Jan	2020 Feb	2006-01-16	338

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
sales	0	1	298323	89613.27	130683	226865	300140	367685.5	529345	▆▇▇ ▅▁

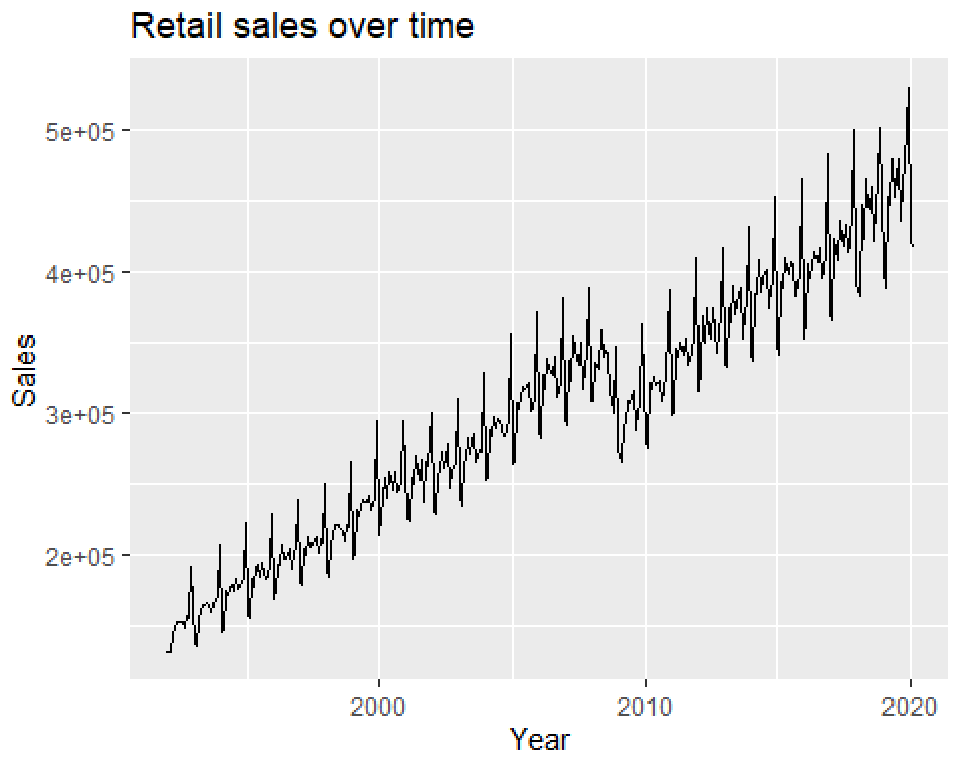
```r
# 1.b
```

```
tsRetail <- as_tsibble(tsRetail, index = date)
tsRetail
```

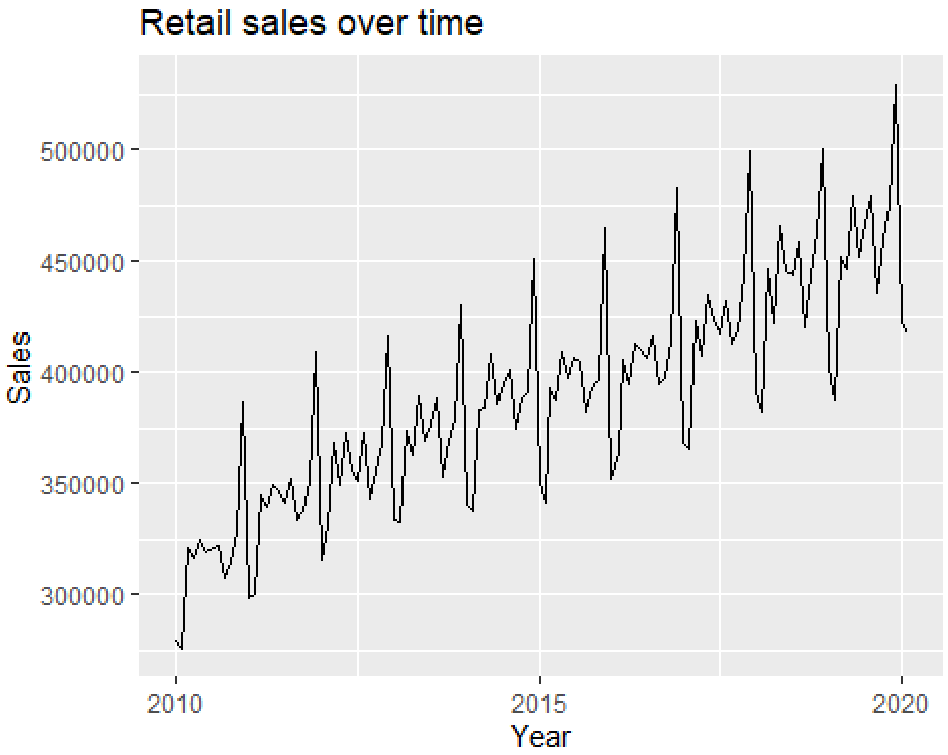
```
## # A tsibble: 338 x 2 [1M]
##         date  sales
##        <mth>  <dbl>
##  1 1992 Jan 130683
##  2 1992 Feb 131244
##  3 1992 Mar 142488
##  4 1992 Apr 147175
##  5 1992 May 152420
##  6 1992 Jun 151849
##  7 1992 Jul 152586
##  8 1992 Aug 152476
##  9 1992 Sep 148158
## 10 1992 Oct 155987
## # ... with 328 more rows
```

```
# 1.c

autoplot(tsRetail) +
xlab("Year") + ylab("Sales") +
ggtitle("Retail sales over time")
```

```
## Plot variable not specified, automatically selected `.vars = sales`
```

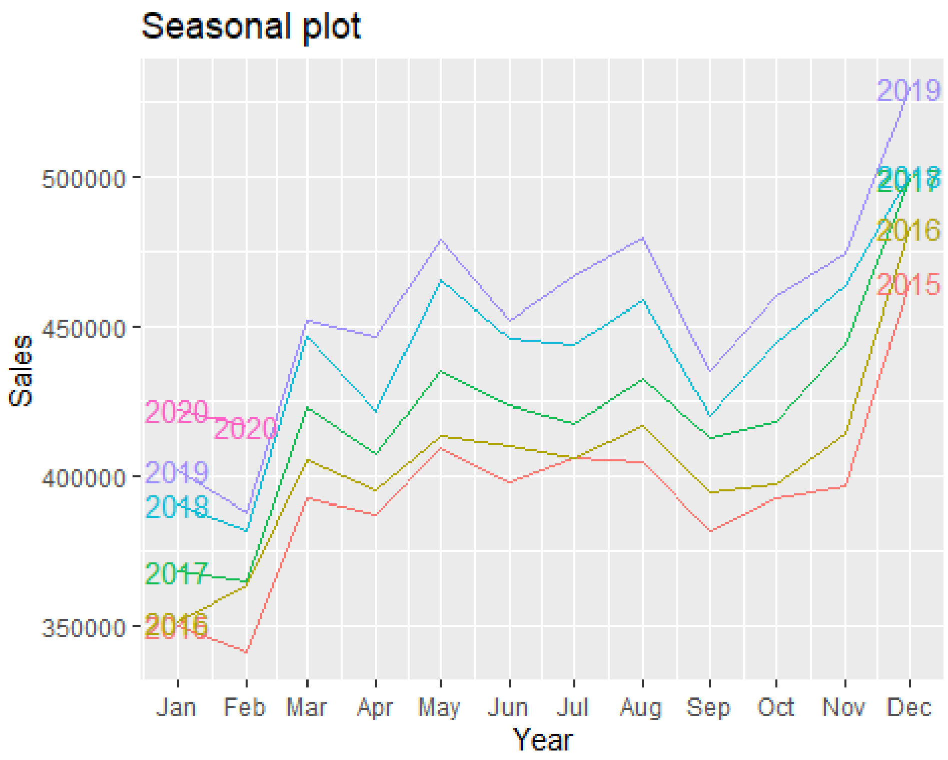


```
tsRetail_subset <- tsRetail %>%
  filter(date > "2009-12-31")

autoplot(tsRetail_subset) +
  xlab("Year") + ylab("Sales") +
  ggtitle("Retail sales over time")

## Plot variable not specified, automatically selected `.vars = sales`
```

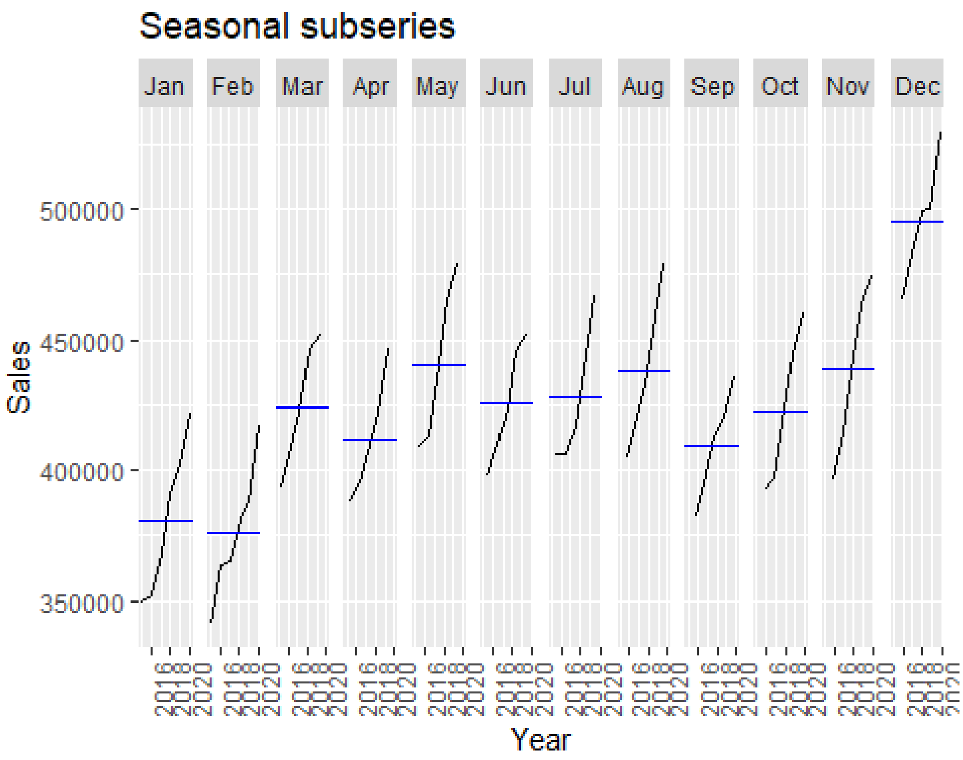


# Question 2

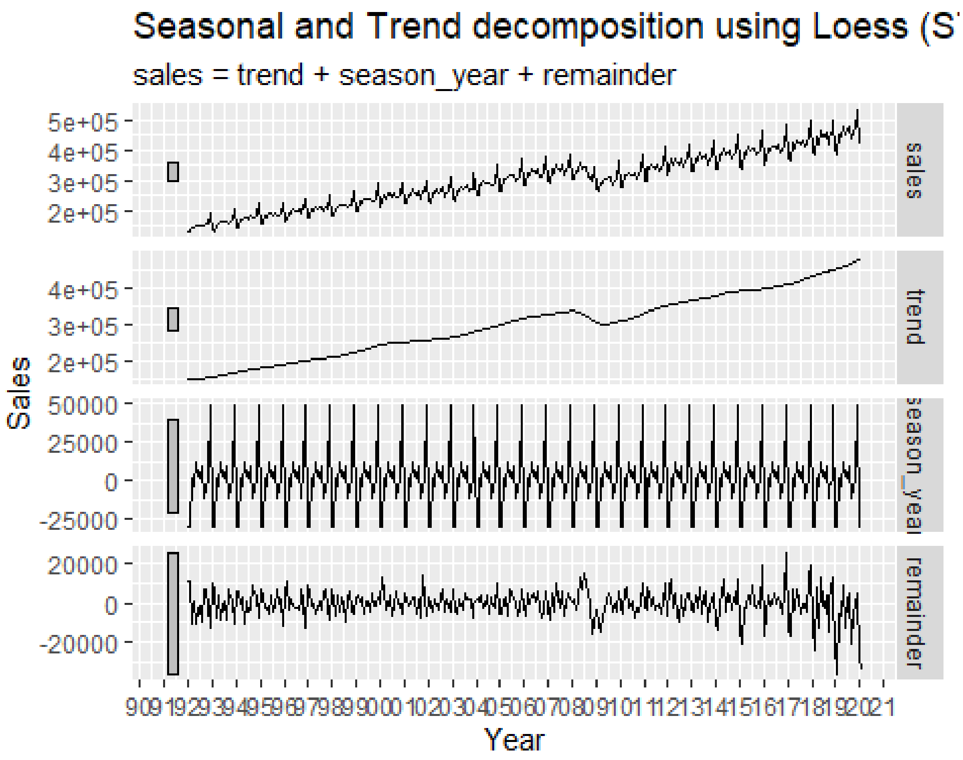
```
# 2.a

tsRetail_seasonalPlot <- tsRetail %>%
  filter(date >= "2015-01-01") %>%
  gg_season(sales, labels = "both") +
  xlab("Year") + ylab("Sales") +
  ggtitle("Seasonal plot")

tsRetail_seasonalPlot
```



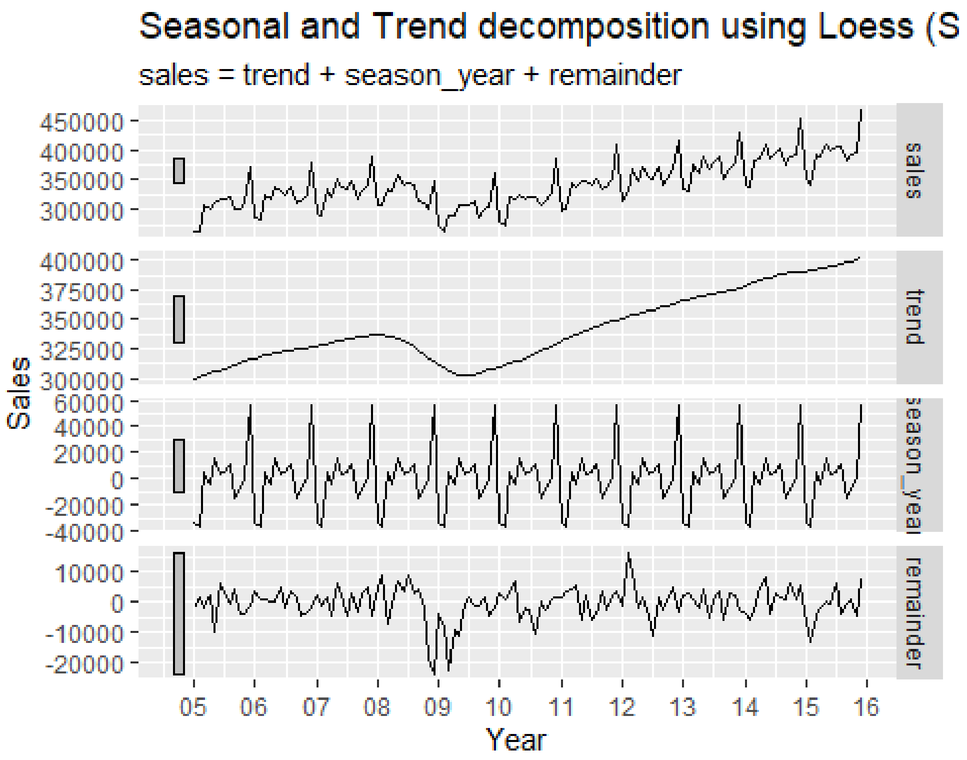
```
tsRetail_seasonalSubseries <- tsRetail %>%
  filter(date >= "2015-01-01") %>%
  gg_subseries(sales) +
  ylab("Sales") +
  xlab("Year") +
  ggtitle("Seasonal subseries")

tsRetail_seasonalSubseries
```



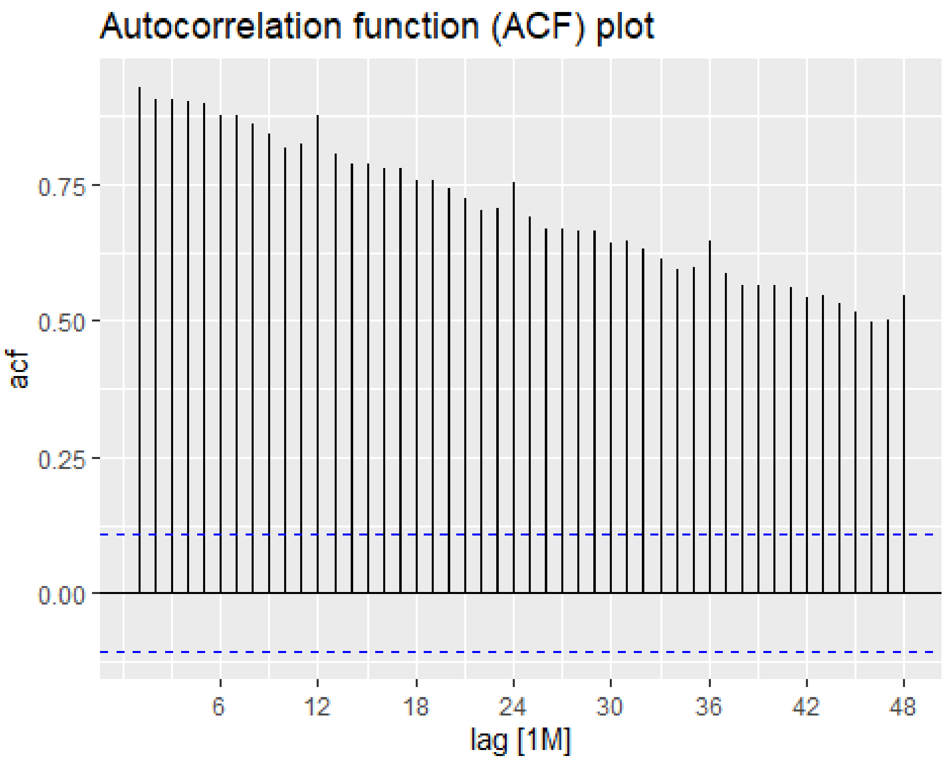
```
# 2.b

tsRetail_Decomposed <- tsRetail %>%
  model(STL(sales ~ trend() + season(window = "periodic"), robust = TRUE))
%>%
  components() %>%
  autoplot() +
  xlab("Year") + ylab("Sales") +
  ggtitle("Seasonal and Trend decomposition using Loess (STL decomposition)")
+
  scale_x_date(date_breaks = "years" , date_labels = "%y")

ggplotly(tsRetail_Decomposed)

tsRetail_Decomposed
```



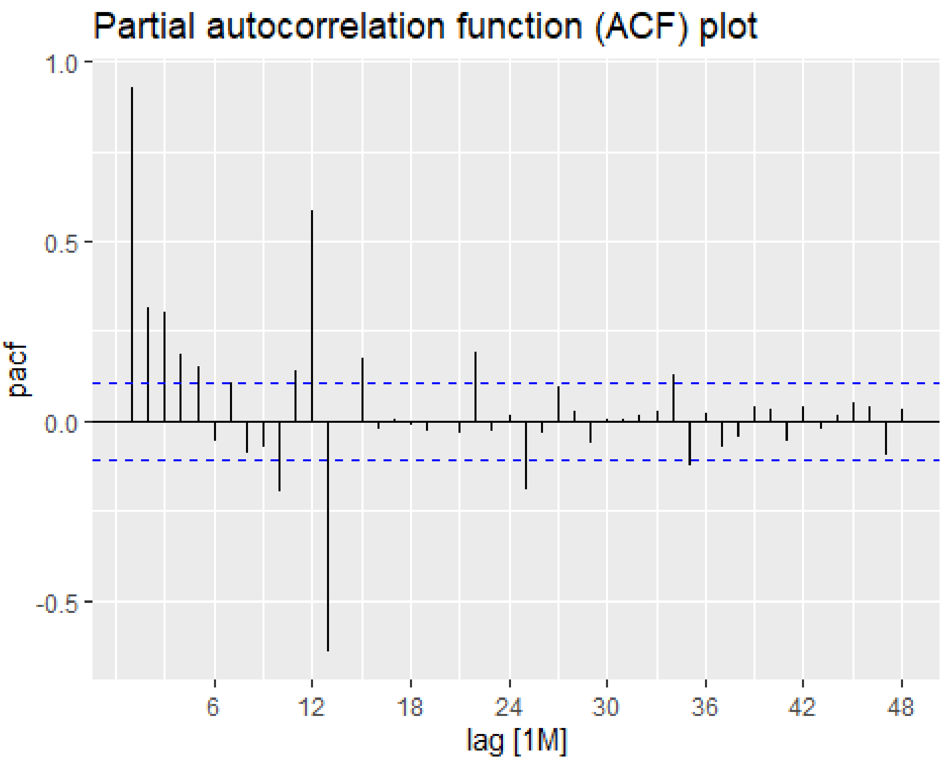
```
tsRetail_Decomposed1 <- tsRetail %>%
  filter(year(date) >= 2005 & year(date) <= 2015) %>%
  model(STL(sales ~ trend() + season(window = "periodic"), robust = TRUE))
%>%
  components() %>%
  autoplot() +
  xlab("Year") + ylab("Sales") +
  ggtitle("Seasonal and Trend decomposition using Loess (STL decomposition)")
+
  scale_x_date(date_breaks = "years" , date_labels = "%y")

ggplotly(tsRetail_Decomposed1)

tsRetail_Decomposed1
```



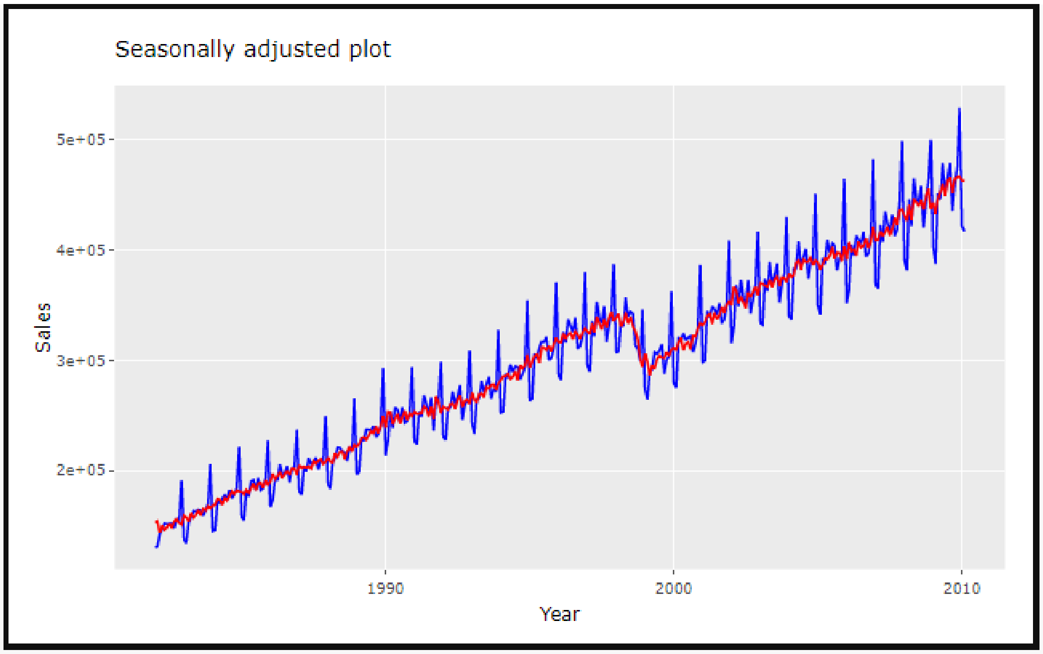
```
# 2.c

tsRetail_ACF <- tsRetail %>%
  ACF(sales, lag_max = 48) %>%
  autoplot() + ggtitle("Autocorrelation function (ACF) plot")

tsRetail_ACF
```



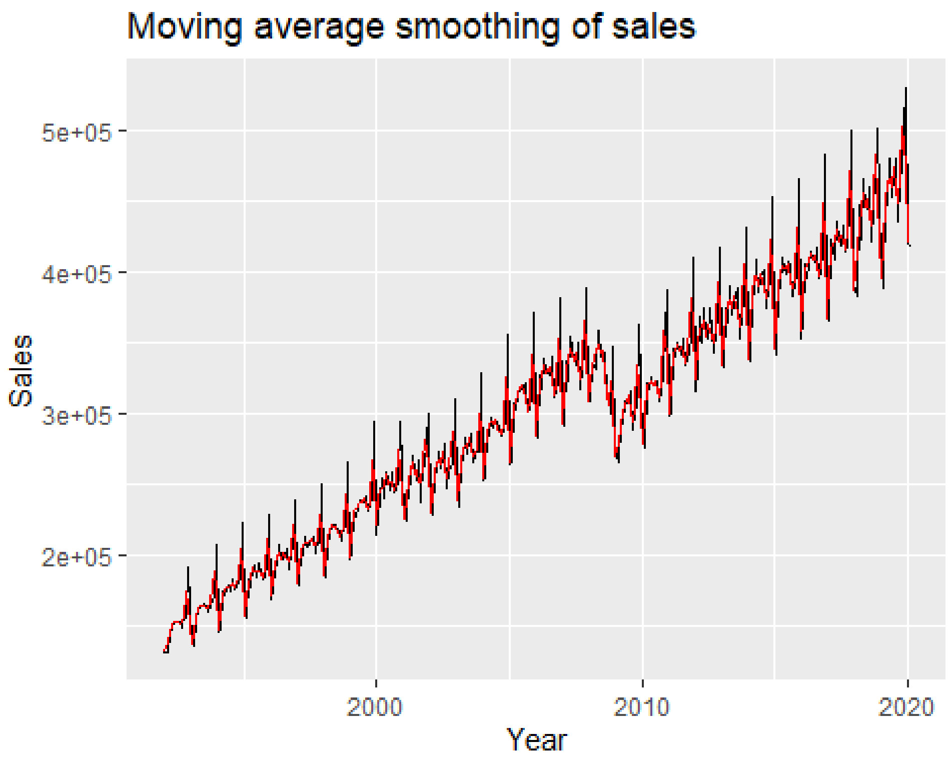
```
tsRetail_PACF <- tsRetail %>%
  PACF(sales, lag_max = 48) %>%
  autoplot() + ggtitle("Partial autocorrelation function (ACF) plot")

tsRetail_PACF
```



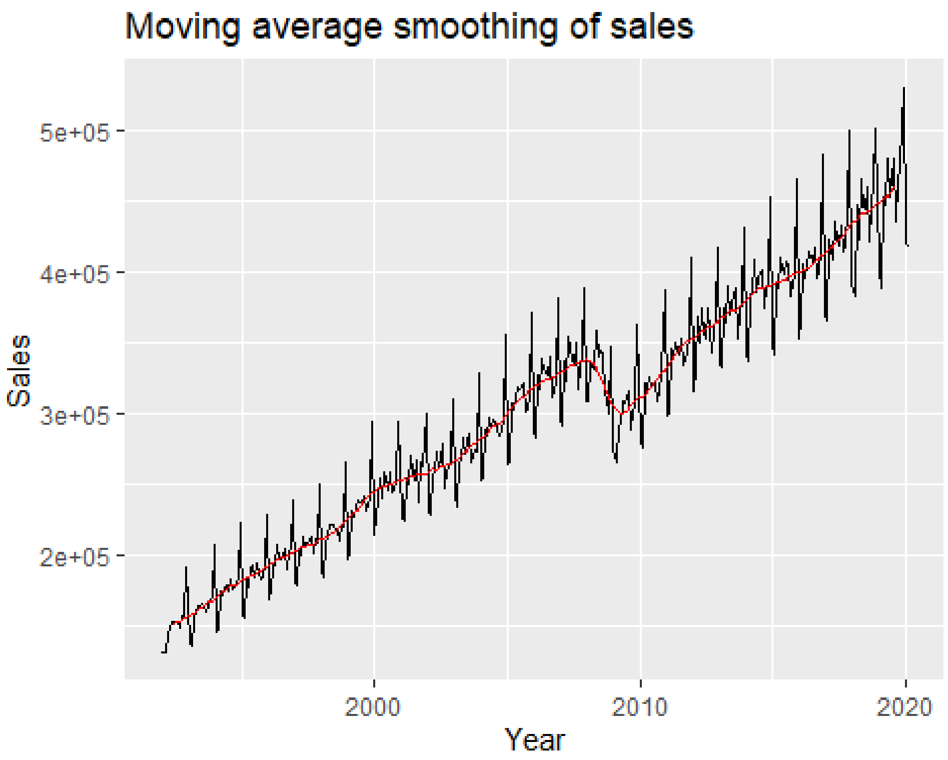
```r
# 2.d

tsRetail_SeasonAdjusted <- tsRetail %>%
  autoplot(sales, color = "blue") +
  autolayer(components(tsRetail %>%
  model(STL(sales))),
  season_adjust, color = "red") +
  xlab("Year") + ylab("Sales") +
  ggtitle("Seasonally adjusted plot")

ggplotly(tsRetail_SeasonAdjusted)
```



```
# 2.e

tsRetail_ma <- tsRetail %>%
  mutate(`2-MA` = slide_dbl(sales, mean, .size = 2, .align = "center-left"))

tsRetail_ma %>%
  autoplot(sales) +
  autolayer(tsRetail_ma, `2-MA`, color='red') +
  xlab("Year") + ylab("Sales") +
  ggtitle("Moving average smoothing of sales") +
  guides(colour = guide_legend(title = "series"))

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



```
tsRetail_ma1 <- tsRetail %>%
  mutate(`12-MA` = slide_dbl(sales, mean, .size = 12, .align = "center-
left"))

tsRetail_ma1 %>%
  autoplot(sales) +
  autolayer(tsRetail_ma1, `12-MA`, color='red') +
  xlab("Year") + ylab("Sales") +
  ggtitle("Moving average smoothing of sales") +
  guides(colour = guide_legend(title = "series"))

## Warning: Removed 11 row(s) containing missing values (geom_path).
```

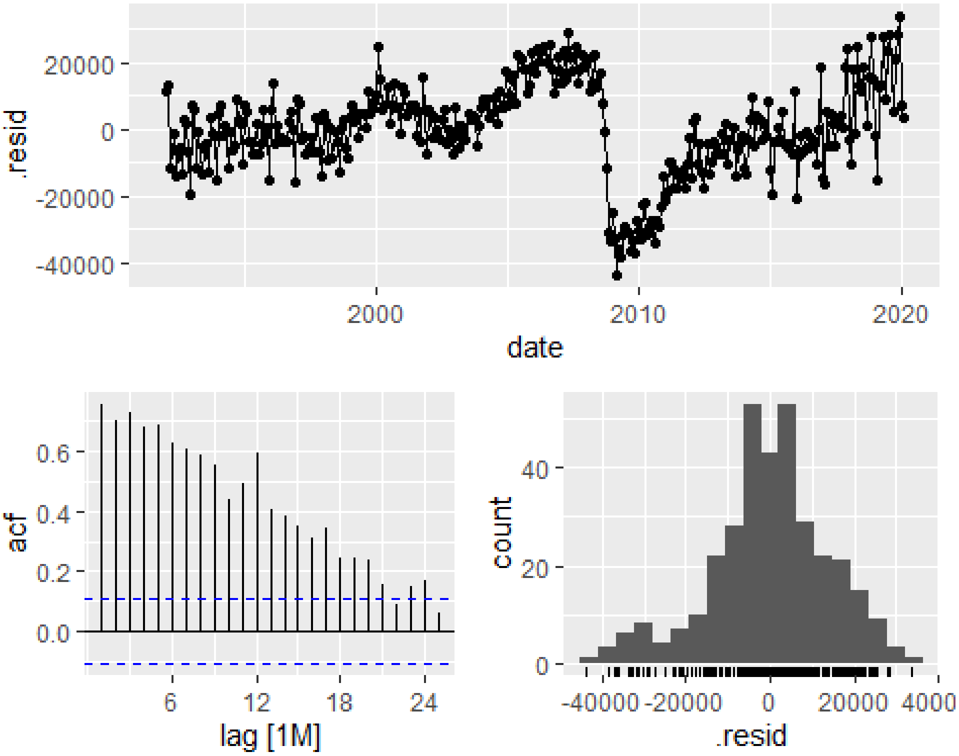


Question 3

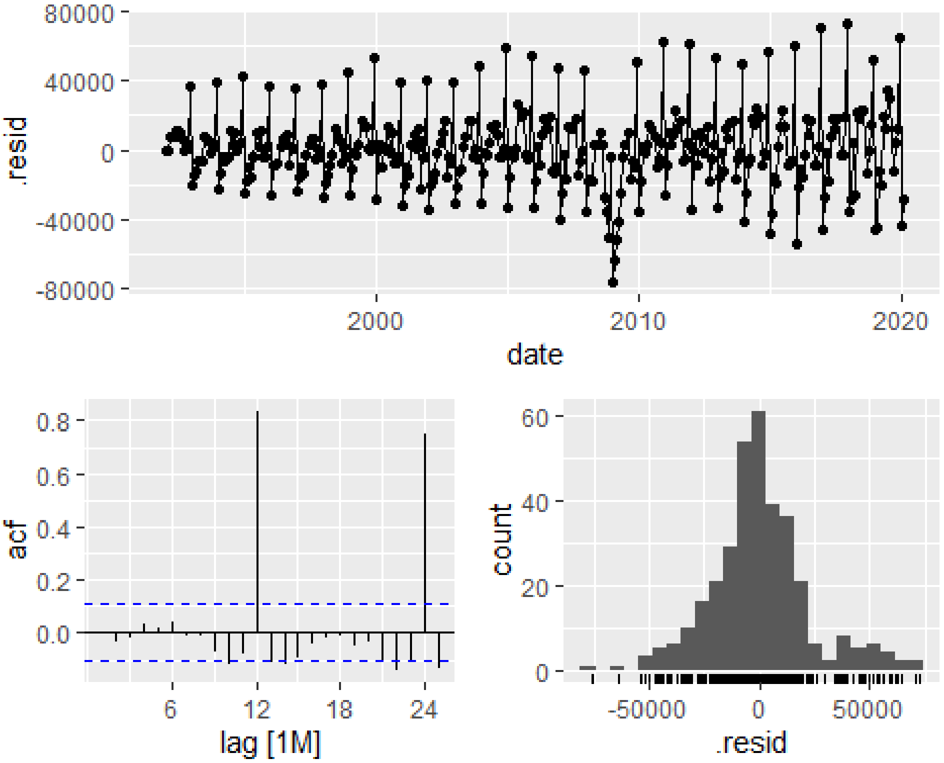
```
# 3.a

fit_tsRetail <- tsRetail %>%
  model(TSLM(sales ~ trend() + season()))

report(fit_tsRetail)
```

```
## 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
```

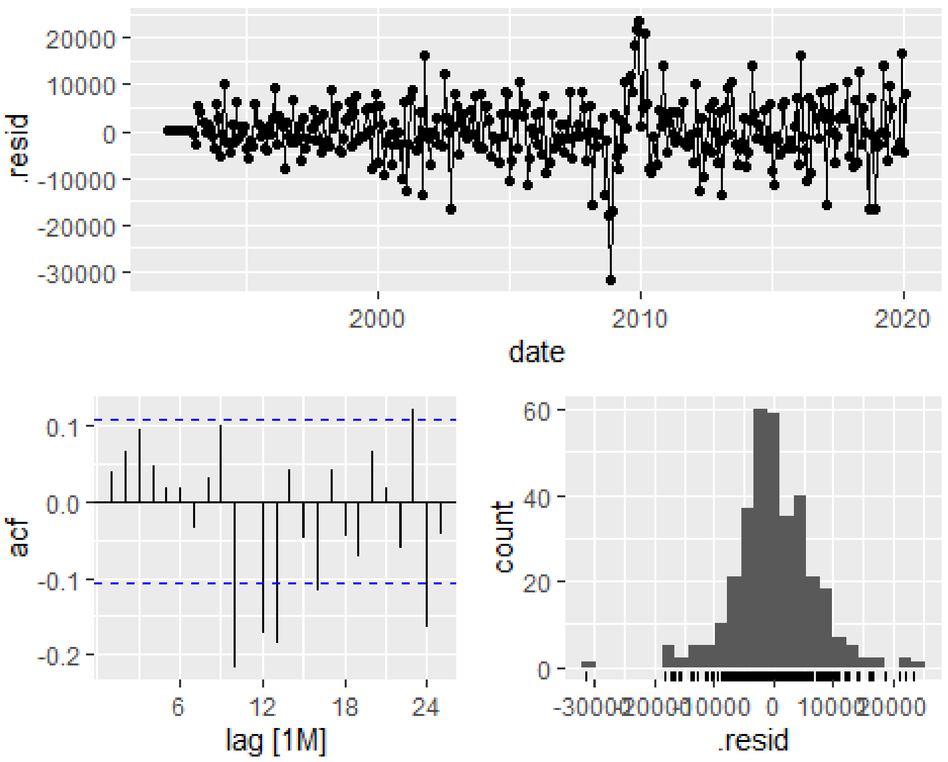
```
fit_tsRetail %>% gg_tsresiduals()
```



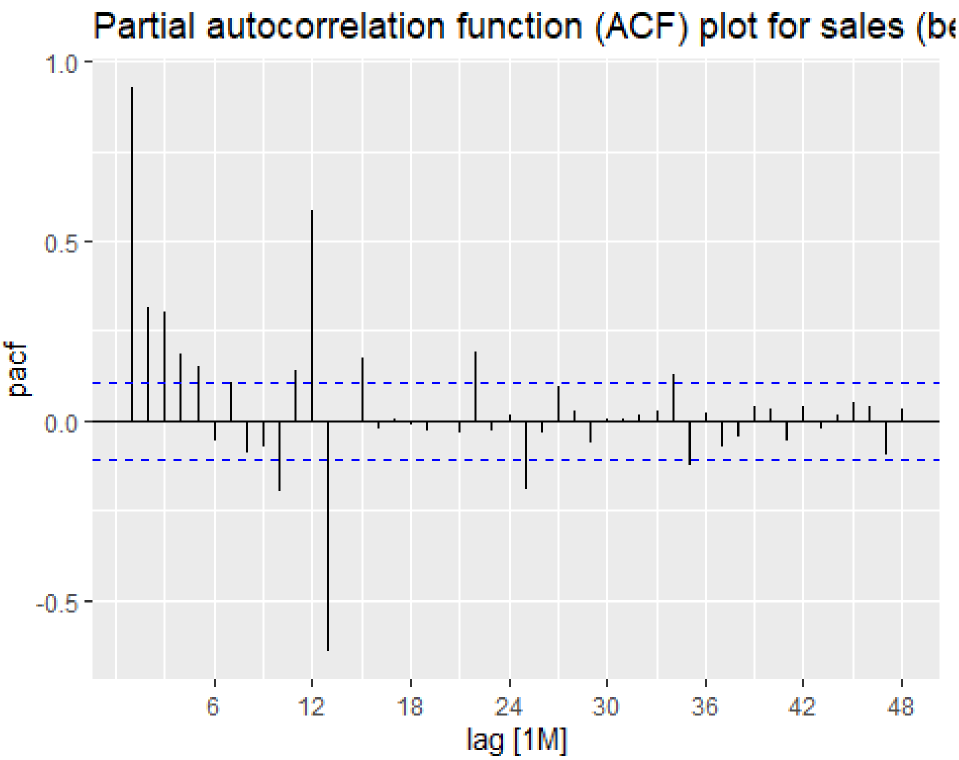
```
# 3.b

fit_tsRetail_ARIMA <- tsRetail %>%
  model(fitArima = ARIMA(sales ~ PDQ(0,0,0), stepwise = FALSE, approximation
= FALSE))

report(fit_tsRetail_ARIMA)

## Series: sales
## Model: ARIMA(4,1,2) w/ drift
##
## Coefficients:
##             ar1      ar2      ar3      ar4      ma1      ma2   constant
##        -0.8347  -0.5704  -0.4584  -0.2791  -0.1269  -0.4631  3010.0579
## s.e.    0.1013   0.0830   0.0830   0.0597   0.0948   0.0780   499.6433
##
## sigma^2 estimated as 498887745:  log likelihood=-3850.47
## AIC=7716.94   AICc=7717.38   BIC=7747.5

fit_tsRetail_ARIMA %>% gg_tsresiduals()
```



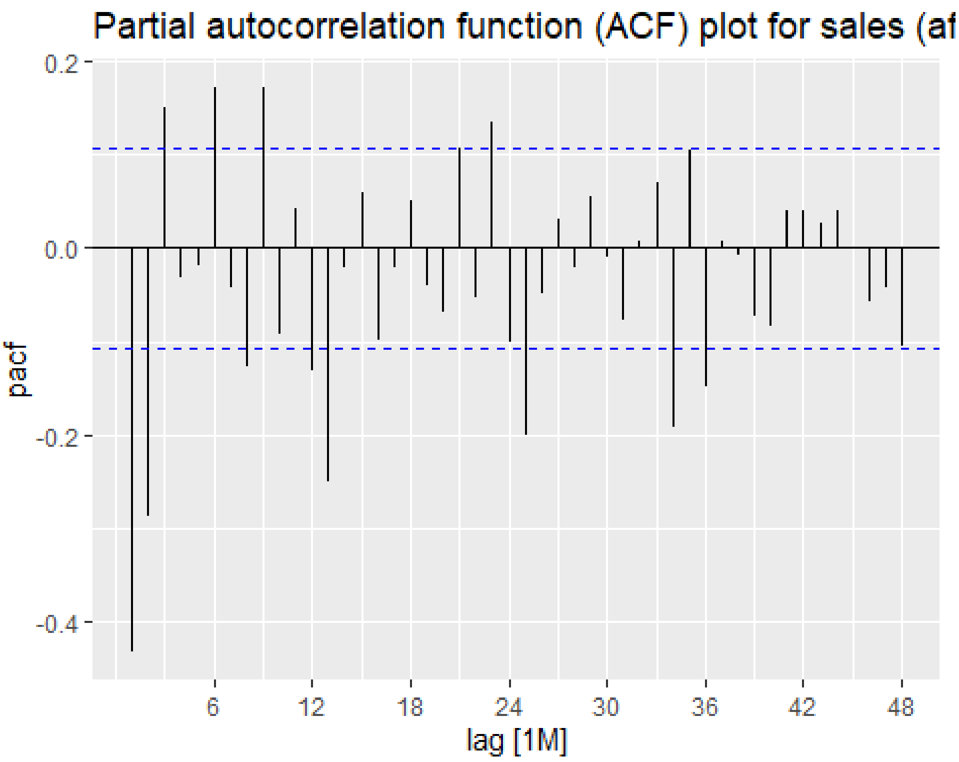
```
fit_tsRetail_ARIMA1 <- tsRetail %>%
  model(fitArima = ARIMA(sales ~ pdq(4,1,2), stepwise = FALSE, approximation
= FALSE))

report(fit_tsRetail_ARIMA1)

## Series: sales
## Model: ARIMA(4,1,2)(0,1,0)[12]
##
## Coefficients:
##            ar1      ar2      ar3      ar4      ma1     ma2
##        -0.4868  -1.0957  -0.4626  -0.3489  -0.0550  0.8788
## s.e.    0.1084   0.0708   0.0672   0.0547   0.1123  0.0646
##
## sigma^2 estimated as 45789269:  log likelihood=-3325
## AIC=6664.01   AICc=6664.36   BIC=6690.5

fit_tsRetail_ARIMA1 %>% gg_tsresiduals()
```



```
# 3.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_PACF <- tsRetail %>%
  PACF(sales, lag_max = 48) %>%
  autoplot() + ggtitle("Partial autocorrelation function (ACF) plot for sales
(before differencing)")

tsRetail_PACF
```



```
tsRetail_DiffPACF <- tsRetail %>%
  mutate(diffSales = difference(difference(sales),12)) %>%
  PACF(diffSales, lag_max = 48) %>%
  autoplot() + ggtitle("Partial autocorrelation function (ACF) plot for sales (after differencing)")

tsRetail_DiffPACF
```

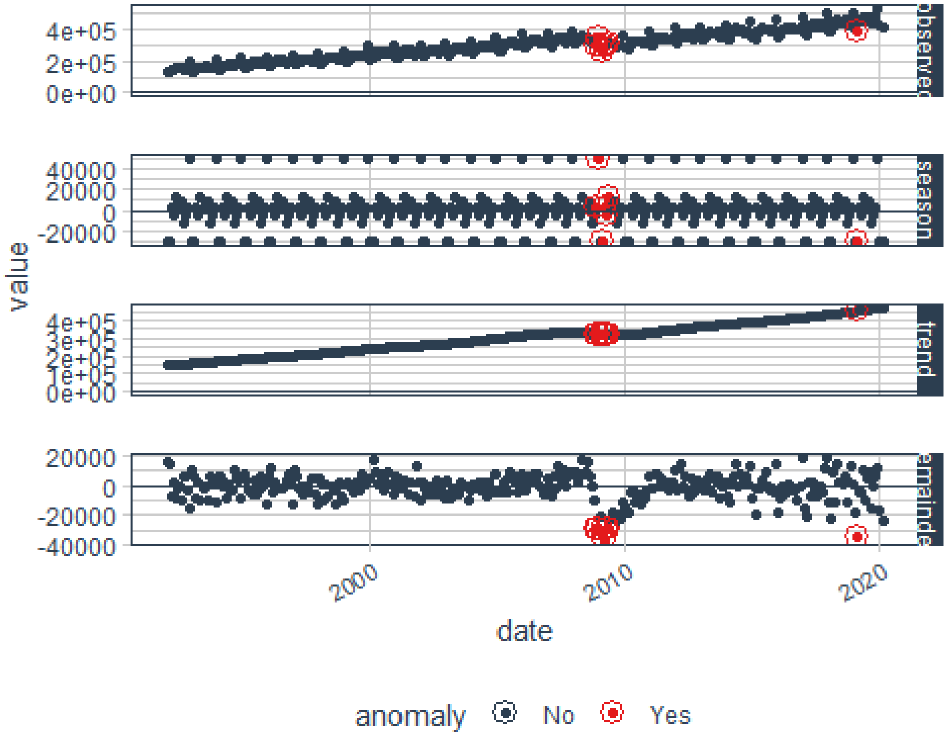


```
# 3.d

set.seed(333)
tsRetail_Train <- tsRetail %>% filter(date < "2011-01-01")
tsRetail_Test <- tsRetail %>% filter(date >= "2011-01-01")

tsRetail_FitAll <- tsRetail_Train %>%
  model(
  model1TimeTrendAndSeason = TSLM(sales ~ trend() + season()),
  model2ArimaGrid = ARIMA(sales ~ PDQ(0,0,0), stepwise = FALSE, approximation
= FALSE))

tsRetail_PredictAll <- tsRetail_FitAll %>%
  forecast(new_data = tsRetail_Test)

accuracy(tsRetail_PredictAll, tsRetail_Test)

## # A tibble: 2 x 9
##   .model                   .type     ME   RMSE    MAE    MPE  MAPE  MASE
ACF1
##   <chr>                    <chr>  <dbl>  <dbl>  <dbl>  <dbl> <dbl> <dbl>
<dbl>
## 1 model1TimeTrendAndSeason Test    969. 14250. 10815. -0.119  2.70   NaN
0.409
## 2 model2ArimaGrid          Test  16511. 32984. 25504.  3.55   6.13   NaN
0.0438
```

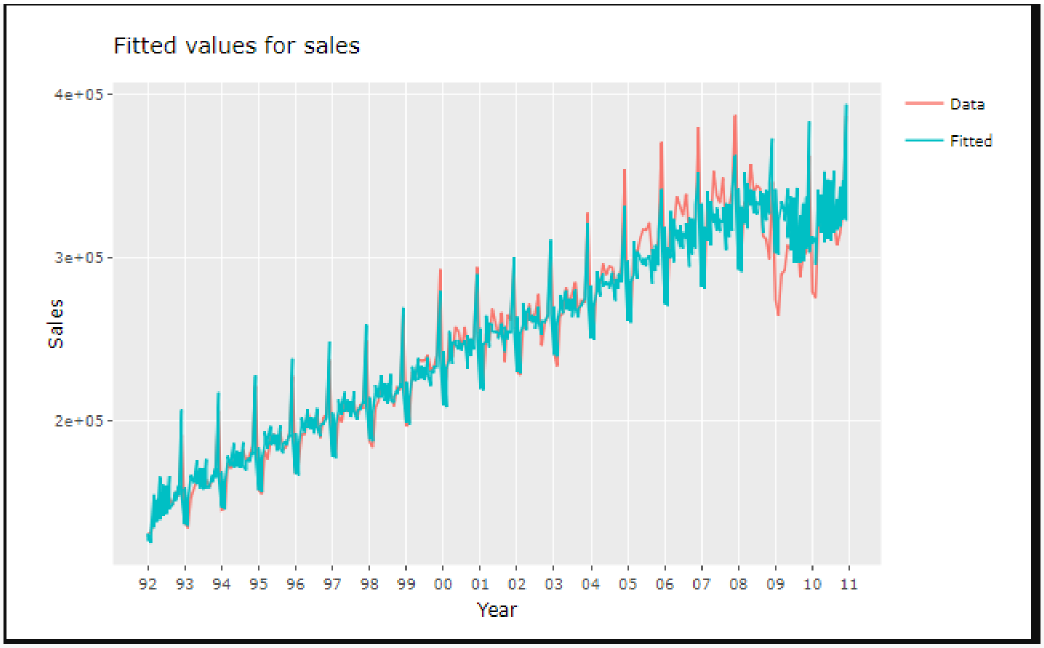
```
# 3.e

set.seed(333)
tsRetail_Train1 <- tsRetail %>% filter(date < "2016-01-01")
tsRetail_Test1 <- tsRetail %>% filter(date >= "2016-01-01")

tsRetail_FitAll1 <- tsRetail_Train1 %>%
  model(
  model1TimeTrendAndSeason1 = TSLM(sales ~ trend() + season()),
  model2ArimaGrid1 = ARIMA(sales ~ PDQ(0,0,0), stepwise = FALSE,
approximation = FALSE))

tsRetail_PredictAll1 <- tsRetail_FitAll1 %>%
  forecast(new_data = tsRetail_Test1)

accuracy(tsRetail_PredictAll1, tsRetail_Test1)

## # A tibble: 2 x 9
##   .model                    .type     ME   RMSE    MAE   MPE  MAPE  MASE
ACF1
##   <chr>                     <chr>  <dbl>  <dbl>  <dbl> <dbl> <dbl> <dbl>
<dbl>
## 1 model1TimeTrendAndSeason1 Test  11405. 18692. 14567.  2.39  3.24   NaN
0.366
## 2 model2ArimaGrid1          Test  -3232. 30570. 23039. -1.32  5.41   NaN
0.0386
```

Question 4

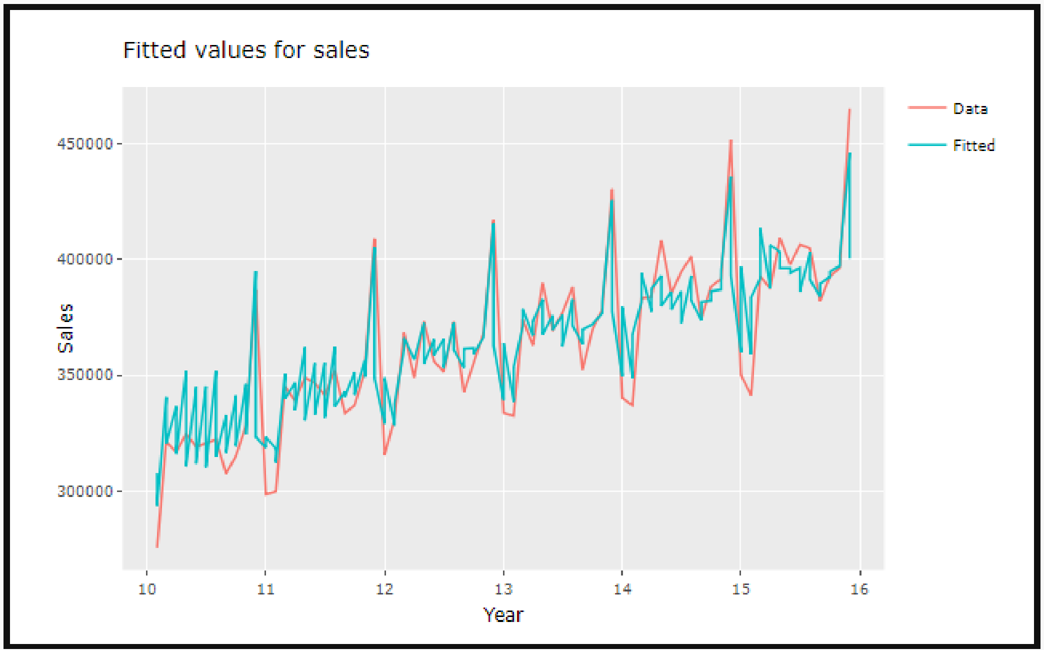
```
# 4.a

tsRetail1 <- read_csv("retailSales.csv")

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

tsRetail1$date <- mdy(tsRetail1$date)
tsRetail1 <- as_tsibble(tsRetail1, index = date)

tsRetail1 %>%
  time_decompose(sales, method = "stl", frequency = "auto", trend = "auto")
%>%
  anomalize(remainder, method = "gesd", alpha = 0.05, max_anoms = 0.2) %>%
  plot_anomaly_decomposition()

## Converting from tbl_ts to tbl_time.
## Auto-index message: index = date
## frequency = 12 months

## trend = 60 months
```



```
# 4.b

tsRetail_Fitted <- augment(tsRetail_FitAll) %>%
  ggplot(aes(x = date)) +
  geom_line(aes(y = sales, colour = "Data")) +
  geom_line(aes(y = .fitted, colour = "Fitted")) +
  xlab("Year") + ylab("Sales") +
  ggtitle("Fitted values for sales") +
  scale_x_date(date_breaks = "years", date_labels = "%y") +
  guides(colour = guide_legend(title = NULL))

ggplotly(tsRetail_Fitted)
```



```
tsRetail_Fitted1 <- augment(tsRetail_FitAll1) %>%
  filter(date > "2010-01-01") %>%
  ggplot(aes(x = date)) +
  geom_line(aes(y = sales, colour = "Data")) +
  geom_line(aes(y = .fitted, colour = "Fitted")) +
  xlab("Year") + ylab("Sales") +
  ggtitle("Fitted values for sales") +
  scale_x_date(date_breaks = "years", date_labels = "%y") +
  guides(colour = guide_legend(title = NULL))

ggplotly(tsRetail_Fitted1)
```



Bonus questions

```
# 1.

usEcon_df <- read_csv("usEcon.csv")

## Parsed with column specification:
## cols(
##   date = col_character(),
##   income = col_double(),
##   unemployment = col_double(),
##   tenYearTreasury = col_double(),
##   CPI = col_double(),
##   inflation = col_character(),
##   vehicleSales = col_double(),
##   houseSales = col_double()
## )

usEcon_df$date <- mdy(usEcon_df$date)
usEcon_df <- as_tsibble(usEcon_df, index = date)
usEcon_df

## # A tsibble: 338 x 8 [1D]
##    date       income unemployment tenYearTreasury   CPI inflation
vehicleSales
##    <date>      <dbl>        <dbl>           <dbl> <dbl> <chr>
<dbl>
##  1 1992-01-01  5264.          6.6            7.03  138. 2.60%
12.6
##  2 1992-02-01  5304.          6.7            7.34  139. 2.82%
12.9
##  3 1992-03-01  5326.          6.7            7.54  139. 3.19%
12.8
##  4 1992-04-01  5360.          6.7            7.48  140. 3.18%
12.6
##  5 1992-05-01  5396.          6.9            7.39  140. 3.02%
13.1
##  6 1992-06-01  5428.          6.9            7.26  140. 3.09%
13.5
##  7 1992-07-01  5441.          6.9            6.84  140. 3.16%
12.9
##  8 1992-08-01  5470.          6.9            6.59  141. 3.15%
12.9
##  9 1992-09-01  5458.          6.8            6.42  141. 2.99%
13.4
## 10 1992-10-01  5450.          6.7            6.59  142. 3.20%
13.7
## # ... with 328 more rows, and 1 more variable: houseSales <dbl>

tsRetail_usEcon <- left_join(tsRetail, usEcon_df, by = c("date" = "date"),
all = TRUE)
```

```
tsRetail_usEcon <- as_tsibble(tsRetail_usEcon,index = date)
tsRetail_usEcon
```

```
## # A tsibble: 338 x 9 [1M]
##          date  sales income unemployment tenYearTreasury   CPI inflation
##         <mth>  <dbl>  <dbl>        <dbl>           <dbl> <dbl> <chr>
##  1   1992 Jan 130683  5264.          6.6            7.03  138. 2.60%
##  2   1992 Feb 131244  5304.          6.7            7.34  139. 2.82%
##  3   1992 Mar 142488  5326.          6.7            7.54  139. 3.19%
##  4   1992 Apr 147175  5360.          6.7            7.48  140. 3.18%
##  5   1992 May 152420  5396.          6.9            7.39  140. 3.02%
##  6   1992 Jun 151849  5428.          6.9            7.26  140. 3.09%
##  7   1992 Jul 152586  5441.          6.9            6.84  140. 3.16%
##  8   1992 Aug 152476  5470.          6.9            6.59  141. 3.15%
##  9   1992 Sep 148158  5458.          6.8            6.42  141. 2.99%
## 10   1992 Oct 155987  5450.          6.7            6.59  142. 3.20%
## # ... with 328 more rows, and 2 more variables: vehicleSales <dbl>,
## #   houseSales <dbl>
```

```
set.seed(333)
tsRetail_usEcon_Train <- tsRetail_usEcon %>% filter(date < "2011-01-01")
tsRetail_usEcon_Test <- tsRetail_usEcon %>% filter(date >= "2011-01-01")

tsRetail_usEcon_FitAll <- tsRetail_usEcon_Train %>%
  model(tsRetail_usEcon_TimeTrendAndSeason = TSLM(sales ~ trend() + season()
+ income + unemployment + CPI + inflation))

report(tsRetail_usEcon_FitAll)
```

```
## Series: sales
## Model: TSLM
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -9703.27     0.00     0.00    42.55  9703.27
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)     325968.912 81142.167   4.017 0.000198 ***
## trend()            960.070   193.300   4.967 8.34e-06 ***
## season()year2     4464.143  3680.417   1.213 0.230850
## season()year3    31081.566  4612.311   6.739 1.54e-08 ***
## season()year4    24736.451  4517.083   5.476 1.41e-06 ***
## season()year5    36394.343  3706.416   9.819 2.96e-13 ***
## season()year6    32571.192  3544.501   9.189 2.55e-12 ***
## season()year7    29975.029  3991.381   7.510 9.59e-10 ***
## season()year8    35614.417  4072.907   8.744 1.20e-11 ***
## season()year9    16564.736  3419.188   4.845 1.27e-05 ***
## season()year10   26264.309  3936.873   6.671 1.96e-08 ***
## season()year11   24576.020  3586.941   6.852 1.02e-08 ***
## season()year12   67963.943   3839.186  17.703  < 2e-16 ***
## income             21.094      5.074   4.158 0.000126 ***
## unemployment     -4770.319    809.045  -5.896 3.17e-07 ***
## CPI              -2162.103    721.269  -2.998 0.004229 **
## inflation-0.38% -18624.352   9332.929  -1.996 0.051445 .
## inflation-0.74% -10250.233   9407.346  -1.090 0.281114
## inflation-1.28%  -8736.471   9118.101  -0.958 0.342599
## inflation-1.29%  -3134.392   8973.654  -0.349 0.728340
## inflation-1.43%   -351.665   9155.455  -0.038 0.969513
## inflation-1.48%   4786.732   9281.775   0.516 0.608328
## inflation-2.10%   4337.648   9124.091   0.475 0.636568
## inflation0.03%  -12862.918   9292.867  -1.384 0.172453
## inflation0.09%  -14582.010   9356.362  -1.559 0.125419
## inflation0.24%  -20589.131   9248.760  -2.226 0.030540 *
## inflation1.05%   -3834.406   9095.256  -0.422 0.675136
## inflation1.07%   -8617.594   8273.684  -1.042 0.302623
## inflation1.14%    1632.881   7502.924   0.218 0.828601
## inflation1.15%   -6773.863   9267.189  -0.731 0.468221
## inflation1.17%   -8209.860   8580.148  -0.957 0.343250
## inflation1.18%    9879.261   9473.137   1.043 0.302024
## inflation1.24%   -1375.170   9091.804  -0.151 0.880384
## inflation1.31%    1967.814   9258.181   0.213 0.832543
## inflation1.37%   -9124.329  10003.577  -0.912 0.366090
## inflation1.44%   -2976.283   8855.078  -0.336 0.738194
## inflation1.46%    9857.401   9262.265   1.064 0.292327
## inflation1.48%    1582.065   9767.575   0.162 0.871981
## inflation1.49%   -1095.902   8384.606  -0.131 0.896534
## inflation1.50%   17316.032   9147.286   1.893 0.064150 .
## inflation1.51%    1799.756   9379.982   0.192 0.848620
## inflation1.55%    1505.651   8798.513   0.171 0.864816
## inflation1.57%    4505.696   9968.058   0.452 0.653214
## inflation1.61%   -3161.254   8946.429  -0.353 0.725309
## inflation1.62%  -12245.749   9922.584  -1.234 0.222924
## inflation1.64%    9204.038   9708.593   0.948 0.347674
## inflation1.67%   -3434.878  10077.548  -0.341 0.734649
## inflation1.68%   -2773.155   8577.704  -0.323 0.747818
## inflation1.69%    -831.681   8608.022  -0.097 0.923417
## inflation1.70%     731.850  10102.572   0.072 0.942539
## inflation1.73%   -2952.597  10110.687  -0.292 0.771474
## inflation1.74%   12510.972   9668.411   1.294 0.201609
## inflation1.77%    4388.794   9688.025   0.453 0.652500
## inflation1.80%   14777.681   9597.752   1.540 0.129938
## inflation1.83%    3598.713   9959.986   0.361 0.719385
## inflation1.84%    2762.024   9051.441   0.305 0.761521
## inflation1.88%   13291.061   9737.116   1.365 0.178367
## inflation1.90%   13788.226   9685.641   1.424 0.160780
## inflation1.93%    6988.236   9758.155   0.716 0.477235
## inflation1.96%    1203.993   9813.769   0.123 0.902849
## inflation1.97%   16920.868   8852.518   1.911 0.061691 .
## inflation2.02%     816.073   9090.495   0.090 0.928827
## inflation2.03%    5149.893   8903.651   0.578 0.565589
## inflation2.04%    8084.488   8935.798   0.905 0.369947
## inflation2.06%   12715.275   8324.348   1.527 0.132944
## inflation2.08%    6132.752   8356.968   0.734 0.466468
## inflation2.09%    -858.630   9807.058  -0.088 0.930582
## inflation2.11%    8892.493   8176.203   1.088 0.281984
## inflation2.13%   16079.906   9051.677   1.776 0.081742 .
## inflation2.14%   -3417.494   8050.821  -0.424 0.673029
## inflation2.15%    9805.919   9703.846   1.011 0.317112
## inflation2.16%   13243.932   9554.192   1.386 0.171840
## inflation2.20%    9489.998   9583.179   0.990 0.326807
## inflation2.22%   10328.353   9629.824   1.073 0.288628
## inflation2.23%    4055.906   8275.817   0.490 0.626213
## inflation2.24%    8868.403   9430.116   0.940 0.351516
## inflation2.26%    -505.150   9941.857  -0.051 0.959679
## inflation2.28%    2376.973  10010.935   0.237 0.813287
## inflation2.29%    9056.180   8480.480   1.068 0.290701
## inflation2.30%    1864.552   9807.829   0.190 0.849994
## inflation2.31%    8783.733   9340.189   0.940 0.351522
## inflation2.32%   11981.968   9317.388   1.286 0.204374
## inflation2.36%   14729.170   8700.377   1.693 0.096691 .
## inflation2.38%    9122.095   9659.793   0.944 0.349539
## inflation2.42%   -5756.050  10136.991  -0.568 0.572694
## inflation2.49%   11956.359   9726.190   1.229 0.224715
## inflation2.50%    6947.752  10015.327   0.694 0.491073
## inflation2.51%   13422.388   9932.132   1.351 0.182647
## inflation2.52%   15681.697   8665.976   1.810 0.076375 .
## inflation2.53%   22824.382   9514.906   2.399 0.020220 *
## inflation2.54%   14880.004   8077.777   1.842 0.071396 .
## inflation2.56%    -469.200   9324.502  -0.050 0.960069
## inflation2.57%    7908.824  10416.863   0.759 0.451278
## inflation2.60%   16488.368   8734.140   1.888 0.064862 .
## inflation2.61%   15696.440   8334.627   1.883 0.065486 .
## inflation2.62%    8369.861   8786.683   0.953 0.345394
## inflation2.63%    3422.518   8106.833   0.422 0.674706
## inflation2.65%    6689.309   8097.256   0.826 0.412660
## inflation2.67%   14991.286   8978.768   1.670 0.101241
## inflation2.68%   16325.433   8859.997   1.843 0.071320 .
## inflation2.69%   21974.776   8349.776   2.632 0.011266 *
## inflation2.72%    6950.881   7789.198   0.892 0.376467
## inflation2.73%   15212.638   9937.972   1.531 0.132132
## inflation2.74%    -853.429  10120.226  -0.084 0.933132
## inflation2.75%    9739.017   8021.262   1.214 0.230394
## inflation2.76%    6823.852   8191.086   0.833 0.408760
## inflation2.77%    6974.131   8514.691   0.819 0.416633
## inflation2.78%   13428.306   8693.249   1.545 0.128730
## inflation2.80%   14703.387   8550.661   1.720 0.091698 .
## inflation2.81%    8461.231   9129.762   0.927 0.358496
## inflation2.82%   18638.669   9991.613   1.865 0.067994 .
## inflation2.84%    6764.419   9964.407   0.679 0.500358
## inflation2.85%    6233.792  10030.326   0.621 0.537098    
## inflation2.86%    6550.002  10003.221   0.655 0.515605    
## inflation2.88%    8203.539   9968.404   0.823 0.414442    
## inflation2.89%   14265.609   9706.488   1.470 0.147908    
## inflation2.90%    9836.830   8317.383   1.183 0.242528    
## inflation2.92%   -1808.870  10104.223  -0.179 0.858645    
## inflation2.95%    9056.654   9606.823   0.943 0.350351    
## inflation2.96%   22008.882   9635.137   2.284 0.026639 *  
## inflation2.97%    4396.532   9727.744   0.452 0.653254    
## inflation2.98%    -737.131   9580.586  -0.077 0.938978    
## inflation2.99%   16401.674   7664.832   2.140 0.037270 *  
## inflation3.00%   11332.955   8558.968   1.324 0.191490    
## inflation3.01%    2891.842   9648.658   0.300 0.765637    
## inflation3.02%    5347.587   8475.630   0.631 0.530956    
## inflation3.03%    8653.302  10013.516   0.864 0.391626    
## inflation3.04%   15184.170   8805.287   1.724 0.090807 .  
## inflation3.05%   12950.649   8073.660   1.604 0.114999    
## inflation3.07%   -1938.096  10167.018  -0.191 0.849591    
## inflation3.09%    6852.278   8808.136   0.778 0.440265    
## inflation3.15%   11347.078   8531.996   1.330 0.189572    
## inflation3.16%   11831.009   9671.122   1.223 0.226938    
## inflation3.17%   24317.003   9385.255   2.591 0.012510 *  
## inflation3.18%   13141.826  10157.981   1.294 0.201699    
## inflation3.19%    6352.870   7798.034   0.815 0.419121    
## inflation3.20%   17733.559   9163.937   1.935 0.058639 .  
## inflation3.22%    7801.270   8716.770   0.895 0.375090    
## inflation3.23%   15729.251   9980.746   1.576 0.121342    
## inflation3.25%    9519.518   8670.899   1.098 0.277520    
## inflation3.26%   20373.520   8228.470   2.476 0.016717 *  
## inflation3.27%    5865.318   8551.092   0.686 0.495935    
## inflation3.32%    6729.725  10226.130   0.658 0.513498    
## inflation3.36%   19381.840   9993.580   1.939 0.058102 .  
## inflation3.39%    2446.550  10140.971   0.241 0.810345    
## inflation3.41%    1956.276  10000.096   0.196 0.845696    
## inflation3.42%   30537.817   9823.179   3.109 0.003098 ** 
## inflation3.45%    3284.417   8254.952   0.398 0.692418    
## inflation3.46%   19332.423   9700.600   1.993 0.051743 .  
## inflation3.51%   19353.988   9790.852   1.977 0.053599 .  
## inflation3.52%   15575.923   9589.492   1.624 0.110606    
## inflation3.53%   -3870.211  10044.349  -0.385 0.701640    
## inflation3.54%   12755.612   9376.906   1.360 0.179830    
## inflation3.55%   18193.472  10140.307   1.794 0.078833 .  
## inflation3.60%    2328.257   9919.236   0.235 0.815384    
## inflation3.62%   10042.953   9766.409   1.028 0.308750    
## inflation3.64%   22696.919   9742.660   2.330 0.023904 *  
## inflation3.66%   -2179.138   8231.978  -0.265 0.792316    
## inflation3.73%    4279.440   8897.016   0.481 0.632617    
## inflation3.76%    8331.438  10135.897   0.822 0.414995    
## inflation3.82%   28027.556  10101.708   2.775 0.007753 ** 
## inflation3.94%   17217.301  10902.232   1.579 0.120586    
## inflation3.98%   13134.605  10492.767   1.252 0.216476
## inflation3.99%   11784.932   9864.568   1.195 0.237851
## inflation4.03%   10005.137  10371.183   0.965 0.339335
## inflation4.08%   26206.097  10278.463   2.550 0.013897 *
## inflation4.15%   21634.695   9881.580   2.189 0.033263 *
## inflation4.17%   27667.722   9880.155   2.800 0.007238 **
## inflation4.18%   24726.832  10867.111   2.275 0.027203 *
## inflation4.28%   14669.473  10217.647   1.436 0.157315
## inflation4.31%   25847.199  10152.590   2.546 0.014029 *
## inflation4.32%   23728.639   9972.101   2.380 0.021193 *
## inflation4.35%   13989.663   9079.795   1.541 0.129684
## inflation4.69%   24506.234   9679.032   2.532 0.014533 *
## inflation4.94%   14345.587  10768.910   1.332 0.188858
## inflation5.02%   20260.119  11390.099   1.779 0.081361 .
## inflation5.37%   26605.095  11255.492   2.364 0.022018 *
## inflation5.60%   32996.485  11759.386   2.806 0.007130 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5891 on 50 degrees of freedom
## Multiple R-squared: 0.998,   Adjusted R-squared: 0.991
## F-statistic: 142.3 on 177 and 50 DF, p-value: < 2.22e-16
```