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.

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

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

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

```
## v ggplot2 3.2.1     v purrr   0.3.3
## v tibble  2.1.3     v dplyr   0.8.5
## v tidyr   1.0.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
```

```
## Warning: package 'ggplot2' was built under R version 3.6.1
```

```
## Warning: package 'tibble' was built under R version 3.6.2
```

```
## Warning: package 'tidyr' was built under R version 3.6.2
```

```
## Warning: package 'readr' was built under R version 3.6.2
```

```
## Warning: package 'purrr' was built under R version 3.6.2
```

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

```
## Warning: package 'forcats' was built under R version 3.6.2
```

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

```
library("dplyr")
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 'lubridate' was built under R version 3.6.2

## 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 lubridate::date()       masks base::date()
## x dplyr::filter()         masks stats::filter()
## x tsibble::id()           masks dplyr::id()
## x tsibble::interval()     masks lubridate::interval()
## x dplyr::lag()            masks stats::lag()
## x tsibble::new_interval() masks lubridate::new_interval()

library("tidymodels")

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

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

## v broom     0.5.4     v rsample   0.0.5
## v dials     0.0.4     v tune      0.1.0
## v infer     0.5.1     v workflows 0.1.1
## v parsnip   0.0.5     v yardstick 0.0.4
## v recipes   0.1.9

## Warning: package 'dials' was built under R version 3.6.2

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

## Warning: package 'infer' was built under R version 3.6.2

## Warning: package 'parsnip' was built under R version 3.6.2

## Warning: package 'recipes' was built under R version 3.6.2

## Warning: package 'rsample' was built under R version 3.6.2

## 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 3.6.2

## -- Conflicts ------------------------------------------------------------
------------------------------ tidymodels_conflicts() --
## x yardstick::accuracy() masks fabletools::accuracy()
## x scales::discard()     masks purrr::discard()
## x dplyr::filter()       masks stats::filter()
## x recipes::fixed()      masks stringr::fixed()
## x infer::generate()     masks fabletools::generate()
## x tsibble::id()         masks dplyr::id()
## x dplyr::lag()          masks stats::lag()
## x dials::margin()       masks ggplot2::margin()
## x parsnip::null_model() masks fabletools::null_model()
## 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")

## Warning: package 'skimr' was built under R version 3.6.2

library("lubridate")
library("caret")

## Warning: package 'caret' was built under R version 3.6.2

## Loading required package: lattice

##
## Attaching package: 'caret'

## The following objects are masked from 'package:yardstick':
##
##     precision, recall
```

```
## The following objects are masked from 'package:fabletools':
##
##     MAE, RMSE

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

PART I]

Question 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()
## )

#tsLCOrg
```

Question 1) (b)

```
tsLCOrg <- tsLCOrg %>%
  as_tsibble(index = date, key= state) #index is the timestamp variable,key d
efines subject to be measured over time

uniqueAbbrs <- unique(tsLCOrg$state)
```

Question 1) (c)

```
str(tsLCOrg)

## Classes 'tbl_ts', 'tbl_df', 'tbl' and 'data.frame':  4943 obs. of  16 vari
ables:
##  $ date          : Date, format: "2008-01-01" "2008-03-01" ...
##  $ state         : chr  "AK" "AK" "AK" "AK" ...
```

```
##  $ avgLoans      : num  5600 11700 7500 25000 15000 ...
##  $ totalLoans    : num  5600 23400 7500 25000 30000 ...
##  $ avgTerm       : num  36 36 36 36 36 36 36 36 36 36 ...
##  $ avgIntRate    : num  18 11.8 13.9 15.2 12.5 ...
##  $ avgGrade      : num  7 3 4 5 2.5 3 2 2 2 1 ...
##  $ avgEmpLength  : num  5 3.5 3 1 7 7 5 2 10 7 ...
##  $ avgAnnualInc  : num  45000 41400 165000 150000 120000 ...
##  $ avgVerifStatus: num  0 0 0 1 1 0.5 1 0 0 0 ...
##  $ avgHomeOwner  : num  1 0.5 0 0 0 0 0 0 0 0 ...
##  $ avgOpenAcc    : num  11 6.5 15 5 13 10 12 7 5 6 ...
##  $ avgRevolBal   : num  10147 7772 14289 0 91183 ...
##  $ avgRevolUtil  : num  89.8 48.8 34.3 0 25.1 ...
##  $ avgTotalAcc   : num  17 9.5 29 12 27.5 37.5 30 23 18 14 ...
##  $ countOfLoans  : num  1 2 1 1 2 2 1 1 1 1 ...
##  - attr(*, "spec")=
##   .. 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()
##   .. )
##  - attr(*, "key")=Classes 'tbl_df', 'tbl' and 'data.frame':  51 obs. of  2
variables:
##   ..$ state: chr  "AK" "AL" "AR" "AZ" ...
##   ..$ .rows:List of 51
##   .. ..$ : int  1 2 3 4 5 6 7 8 9 10 ...
##   .. ..$ : int  97 98 99 100 101 102 103 104 105 106 ...
##   .. ..$ : int  205 206 207 208 209 210 211 212 213 214 ...
##   .. ..$ : int  309 310 311 312 313 314 315 316 317 318 ...
##   .. ..$ : int  425 426 427 428 429 430 431 432 433 434 ...
##   .. ..$ : int  539 540 541 542 543 544 545 546 547 548 ...
##   .. ..$ : int  656 657 658 659 660 661 662 663 664 665 ...
##   .. ..$ : int  770 771 772 773 774 775 776 777 778 779 ...
##   .. ..$ : int  877 878 879 880 881 882 883 884 885 886 ...
##   .. ..$ : int  983 984 985 986 987 988 989 990 991 992 ...
##   .. ..$ : int  1101 1102 1103 1104 1105 1106 1107 1108 1109 1110 ...
##   .. ..$ : int  1219 1220 1221 1222 1223 1224 1225 1226 1227 1228 ...
##   .. ..$ : int  1321 1322 1323 1324 1325 1326 1327 1328 1329 1330
```

```
##   .. ..$ : int  1331 1332 1333 1334 1335 1336 1337 1338 1339 1340 ...
##   .. ..$ : int  1350 1351 1352 1353 1354 1355 1356 1357 1358 1359 ...
##   .. ..$ : int  1463 1464 1465 1466 1467 1468 1469 1470 1471 1472 ...
##   .. ..$ : int  1521 1522 1523 1524 1525 1526 1527 1528 1529 1530 ...
##   .. ..$ : int  1625 1626 1627 1628 1629 1630 1631 1632 1633 1634 ...
##   .. ..$ : int  1731 1732 1733 1734 1735 1736 1737 1738 1739 1740 ...
##   .. ..$ : int  1842 1843 1844 1845 1846 1847 1848 1849 1850 1851 ...
##   .. ..$ : int  1959 1960 1961 1962 1963 1964 1965 1966 1967 1968 ...
##   .. ..$ : int  2076 2077 2078 2079 2080 2081 2082 2083 2084 2085 ...
##   .. ..$ : int  2099 2100 2101 2102 2103 2104 2105 2106 2107 2108 ...
##   .. ..$ : int  2210 2211 2212 2213 2214 2215 2216 2217 2218 2219 ...
##   .. ..$ : int  2321 2322 2323 2324 2325 2326 2327 2328 2329 2330 ...
##   .. ..$ : int  2438 2439 2440 2441 2442 2443 2444 2445 2446 2447 ...
##   .. ..$ : int  2492 2493 2494 2495 2496 2497 2498 2499 2500 2501 ...
##   .. ..$ : int  2591 2592 2593 2594 2595 2596 2597 2598 2599 2600 ...
##   .. ..$ : int  2683 2684 2685 2686 2687 2688 2689 2690 2691 2692 ...
##   .. ..$ : int  2704 2705 2706 2707 2708 2709 2710 2711 2712 2713 ...
##   .. ..$ : int  2735 2736 2737 2738 2739 2740 2741 2742 2743 2744 ...
##   .. ..$ : int  2841 2842 2843 2844 2845 2846 2847 2848 2849 2850 ...
##   .. ..$ : int  2959 2960 2961 2962 2963 2964 2965 2966 2967 2968 ...
##   .. ..$ : int  3067 3068 3069 3070 3071 3072 3073 3074 3075 3076 ...
##   .. ..$ : int  3177 3178 3179 3180 3181 3182 3183 3184 3185 3186 ...
##   .. ..$ : int  3295 3296 3297 3298 3299 3300 3301 3302 3303 3304 ...
##   .. ..$ : int  3407 3408 3409 3410 3411 3412 3413 3414 3415 3416 ...
##   .. ..$ : int  3513 3514 3515 3516 3517 3518 3519 3520 3521 3522 ...
##   .. ..$ : int  3621 3622 3623 3624 3625 3626 3627 3628 3629 3630 ...
##   .. ..$ : int  3731 3732 3733 3734 3735 3736 3737 3738 3739 3740 ...
##   .. ..$ : int  3836 3837 3838 3839 3840 3841 3842 3843 3844 3845 ...
##   .. ..$ : int  3949 3950 3951 3952 3953 3954 3955 3956 3957 3958 ...
##   .. ..$ : int  4041 4042 4043 4044 4045 4046 4047 4048 4049 4050 ...
##   .. ..$ : int  4102 4103 4104 4105 4106 4107 4108 4109 4110 4111 ...
##   .. ..$ : int  4217 4218 4219 4220 4221 4222 4223 4224 4225 4226 ...
##   .. ..$ : int  4327 4328 4329 4330 4331 4332 4333 4334 4335 4336 ...
##   .. ..$ : int  4444 4445 4446 4447 4448 4449 4450 4451 4452 4453 ...
##   .. ..$ : int  4531 4532 4533 4534 4535 4536 4537 4538 4539 4540 ...
##   .. ..$ : int  4644 4645 4646 4647 4648 4649 4650 4651 4652 4653 ...
##   .. ..$ : int  4758 4759 4760 4761 4762 4763 4764 4765 4766 4767 ...
##   .. ..$ : int  4848 4849 4850 4851 4852 4853 4854 4855 4856 4857 ...
##   ..- attr(*, ".drop")= logi TRUE
##  - attr(*, "index")= chr "date"
##   ..- attr(*, "ordered")= logi TRUE
##  - attr(*, "index2")= chr "date"
##  - attr(*, "interval")=List of 12
##   ..$ year       : num 0
##   ..$ quarter    : num 0
##   ..$ month      : num 0
##   ..$ week       : num 0
##   ..$ day        : num 1
##   ..$ hour       : num 0
##   ..$ minute     : num 0
```

```
##   ..$ second     : num 0
##   ..$ millisecond: num 0
##   ..$ microsecond: num 0
##   ..$ nanosecond : num 0
##   ..$ unit       : num 0
##   ..- attr(*, "class")= chr "interval"
```

Question 1) (d)

```
nyEcon <- 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()
## )
```

```
#nyEcon
```

```
nyEcon$date <- mdy(nyEcon$date)
```

```
str(nyEcon)
```

```
## Classes 'spec_tbl_df', 'tbl_df', 'tbl' and 'data.frame': 118 obs. of  6 va
riables:
##  $ date           : Date, format: "2007-06-01" "2007-07-01" ...
##  $ state          : chr  "NY" "NY" "NY" "NY" ...
##  $ NYCPI          : num  660 661 660 660 661 ...
##  $ NYUnemployment : num  4.5 4.6 4.7 4.7 4.8 4.8 4.8 4.8 4.9 4.9 ...
##  $ NYCondoPriceIdx: num  228 228 227 226 226 ...
##  $ NYSnapBenefits : num  1801707 1792916 1816805 1823494 1825759 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   date = col_character(),
##   ..   state = col_character(),
##   ..   NYCPI = col_double(),
##   ..   NYUnemployment = col_double(),
##   ..   NYCondoPriceIdx = col_double(),
##   ..   NYSnapBenefits = col_double()
##   .. )
```

```
#nyEcon
```

```
#nyEcon$date <-  as.Date(nyEcon$date)
```

Question 1) (e)

```r
pop <- read_csv("populationData.csv")
```

```
## Parsed with column specification:
## cols(
##   GEO_ID = col_character(),
##   NAME = col_character(),
##   P001001 = col_character()
## )
```

```r
pop <- pop[-1,] %>%
  rename(Total_Population = P001001)

pop$GEO_ID <- NULL
pop$NAME
```

```
##  [1] "Alabama"              "Alaska"               "Arizona"
##  [4] "Arkansas"             "California"           "Louisiana"
##  [7] "Kentucky"             "Colorado"             "Connecticut"
## [10] "Delaware"             "District of Columbia" "Florida"
## [13] "Georgia"              "Hawaii"               "Idaho"
## [16] "Illinois"             "Indiana"              "Iowa"
## [19] "Kansas"               "Maine"                "Maryland"
## [22] "Massachusetts"        "Michigan"             "Minnesota"
## [25] "Mississippi"          "Missouri"             "Montana"
## [28] "Nebraska"             "Nevada"               "New Hampshire"
## [31] "New Jersey"           "New Mexico"           "New York"
## [34] "North Carolina"       "North Dakota"         "Ohio"
## [37] "Oklahoma"             "Oregon"               "Pennsylvania"
## [40] "Rhode Island"         "South Carolina"       "South Dakota"
## [43] "Tennessee"            "Texas"                "Utah"
## [46] "Vermont"              "Virginia"             "Washington"
## [49] "West Virginia"        "Wisconsin"            "Wyoming"
## [52] "Puerto Rico"
```

```r
stateData <- data.frame("NAME"= state.name, "state"= state.abb)
#rbind(stateData, c("District of Columbia","DC"))
stateData <- stateData %>%
  add_row(NAME ="District of Columbia" , state = "DC")

#stateData

tempPop <- merge(stateData, pop,by= "NAME")
#tempPop

tsLCOrg <-  merge(tempPop,tsLCOrg, by= "state")
unique(tsLCOrg$NAME)
```

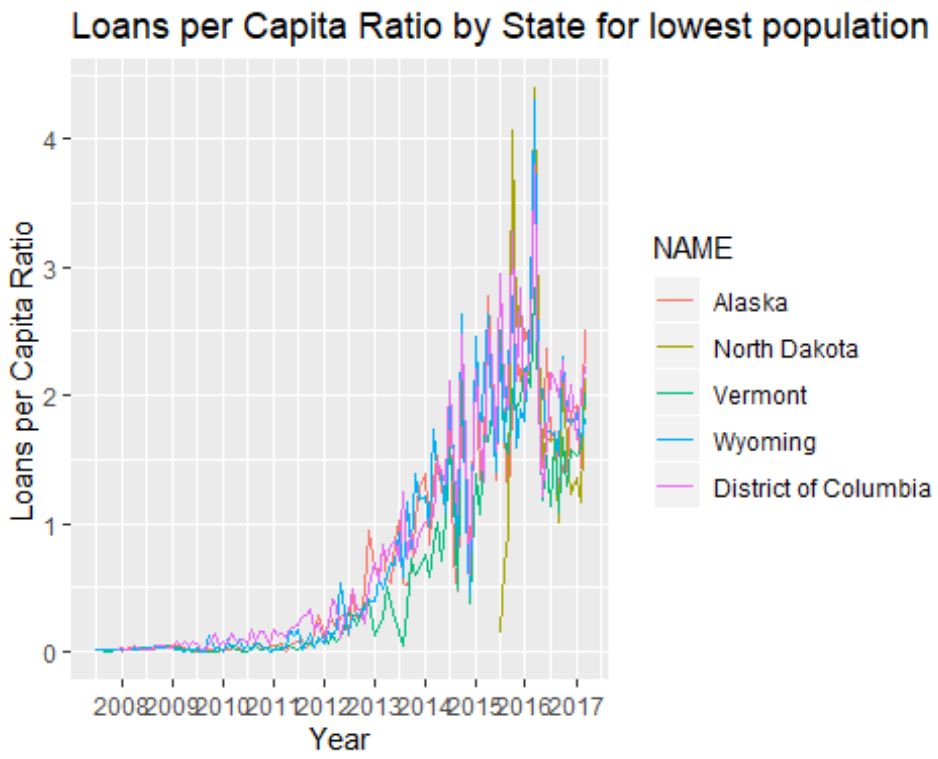
```
##  [1] Alaska              Alabama             Arkansas
##  [4] Arizona             California          Colorado
##  [7] Connecticut         District of Columbia Delaware
## [10] Florida             Georgia             Hawaii
```

```
## [13] Iowa                 Idaho                Illinois
## [16] Indiana              Kansas               Kentucky
## [19] Louisiana            Massachusetts        Maryland
## [22] Maine                Michigan             Minnesota
## [25] Missouri             Mississippi          Montana
## [28] North Carolina       North Dakota         Nebraska
## [31] New Hampshire        New Jersey           New Mexico
## [34] Nevada               New York             Ohio
## [37] Oklahoma             Oregon               Pennsylvania
## [40] Rhode Island         South Carolina       South Dakota
## [43] Tennessee            Texas                Utah
## [46] Virginia             Vermont              Washington
## [49] Wisconsin            West Virginia        Wyoming
## 51 Levels: Alabama Alaska Arizona Arkansas California Colorado ... Distric
t of Columbia

str(tsLCOrg)

## 'data.frame':    4943 obs. of  18 variables:
##  $ state           : Factor w/ 51 levels "AK","AL","AR",..: 1 1 1 1 1 1 1
1 1 1 ...
##  $ NAME            : Factor w/ 51 levels "Alabama","Alaska",..: 2 2 2 2 2
2 2 2 2 2 ...
##  $ Total_Population: chr  "710231" "710231" "710231" "710231" ...
##  $ date            : Date, format: "2017-03-01" "2016-03-01" ...
##  $ avgLoans        : num  17486 17032 15911 16978 17027 ...
##  $ totalLoans      : num  1783550 2827375 1622950 1018650 1685650 ...
##  $ avgTerm         : num  41.9 42.9 42.4 42 43 ...
##  $ avgIntRate      : num  13.5 12.7 13.5 12.9 12.6 ...
##  $ avgGrade        : num  2.7 2.75 2.88 2.67 2.59 ...
##  $ avgEmpLength    : num  5.66 6.22 5.24 6.17 5.98 ...
##  $ avgAnnualInc    : num  81700 74125 72486 68248 85184 ...
##  $ avgVerifStatus  : num  0.716 0.687 0.735 0.717 0.768 ...
##  $ avgHomeOwner    : num  0.1176 0.1446 0.0784 0.1333 0.101 ...
##  $ avgOpenAcc      : num  10.7 10.8 10.6 10.6 10.8 ...
##  $ avgRevolBal     : num  23200 17445 16993 19557 18034 ...
##  $ avgRevolUtil    : num  60.8 55.8 57.3 57.4 56.2 ...
##  $ avgTotalAcc     : num  22.4 22.6 21.1 23.4 24.3 ...
##  $ countOfLoans    : num  102 166 102 60 99 68 77 58 60 54 ...

tsLCOrg <-  tsLCOrg %>%
  mutate(loansPerCapita = totalLoans/as.numeric(Total_Population))

tsLCOrg$NAME.x <- NULL
tsLCOrg$NAME.y <- NULL
tsLCOrg$Total_Population.x <- NULL
tsLCOrg$Total_Population.y <- NULL
#tsLCOrg

#nyEcon
```

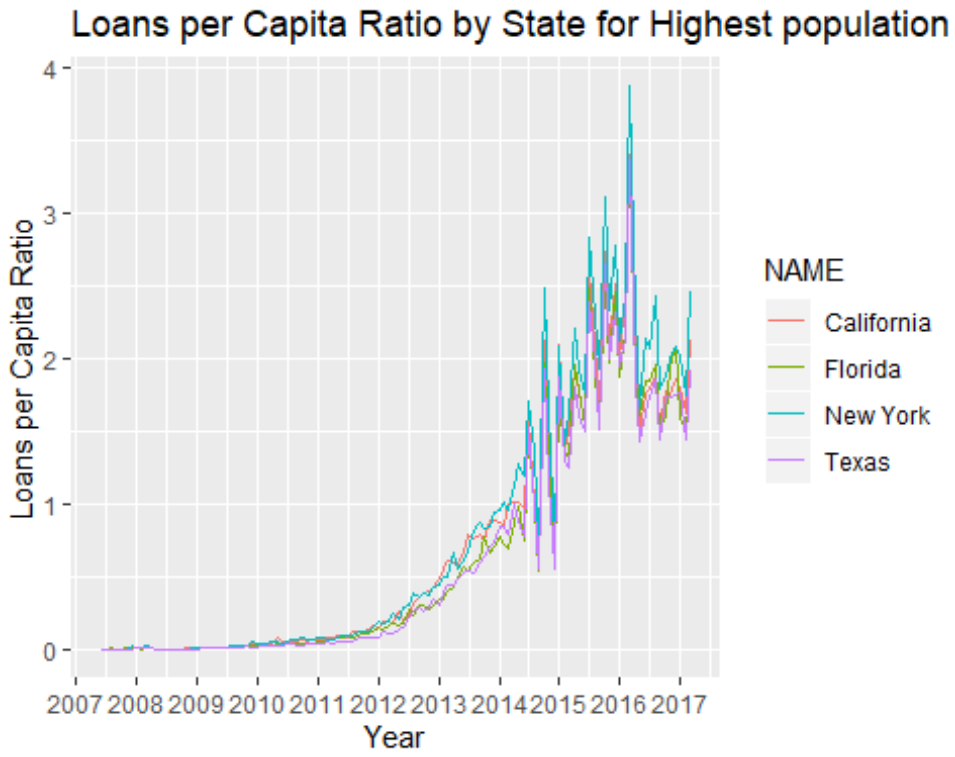
```
tsLCNy <- merge(tsLCOrg,nyEcon,by= c("date","state"))
#tsLCNy

#tsLCOrg
```

Question 2) (a)

```
tsLCOrg$Total_Population <- as.numeric(tsLCOrg$Total_Population)
tsLCOrg$date <-yearmonth(tsLCOrg$date)

quantile(tsLCOrg$Total_Population, 0.9)

##      90%
## 12830632

quantile(tsLCOrg$Total_Population, 0.1)

##    10%
## 814180

#tsLCOrg

maxPop <-tsLCOrg %>%
  filter(quantile(Total_Population, 0.9) < Total_Population) %>%
  select(NAME,date, loansPerCapita)%>%
  as_tsibble(index = date, key= NAME)

minPop <-tsLCOrg %>%
  filter(quantile(Total_Population, 0.1) > Total_Population)%>%
  select(NAME,date, loansPerCapita) %>%
  as_tsibble(index = date, key= NAME)

#maxPop

plotMinPop <-
  minPop %>%
  autoplot()+
  xlab("Year") + ylab("Loans per Capita Ratio") +
  ggtitle("Loans per Capita Ratio by State for lowest population") +
  scale_x_date(date_breaks = "years" , date_labels = "%Y")

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

plotMinPop
```



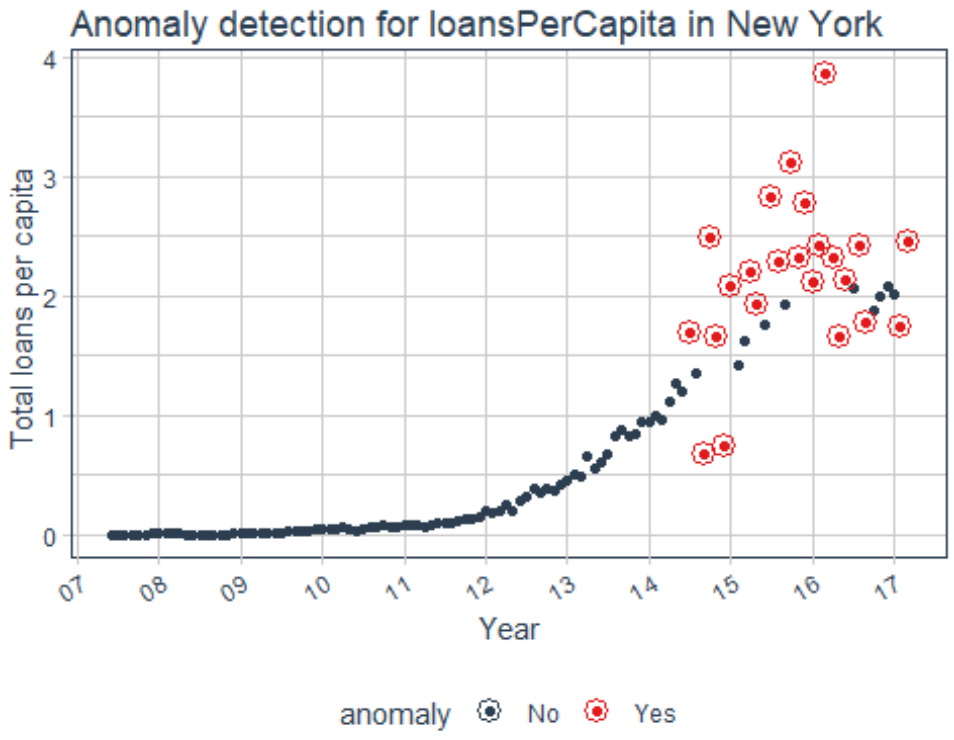

```
plotMaxPop <-
  maxPop %>%
  autoplot()+
  xlab("Year") + ylab("Loans per Capita Ratio") +
  ggtitle("Loans per Capita Ratio by State for Highest population") +
  scale_x_date(date_breaks = "years" , date_labels = "%Y")

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

plotMaxPop
```



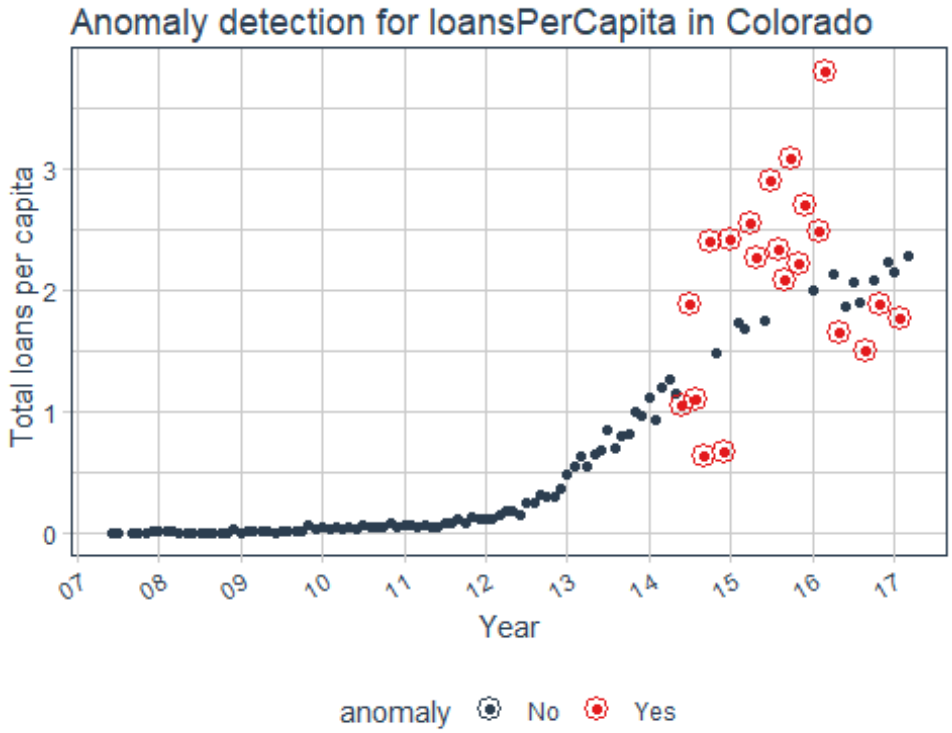
Question 2) (b)

```
#Anomaly Plot



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 Det
ection!
## </> Learn more at: https://university.business-science.io/p/learning-labs-
pro </>

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
```

```r
tsLCOrg2 <-  tsLCOrg
tsLCOrg2$date <- as.Date(tsLCOrg$date)

tsLCOrg2 <- tsLCOrg2 %>%
as_tsibble(index= date, key= state)

anomalyPlotNY <-
  tsLCOrg2 %>% as_tbl_time(index = date) %>%
  filter(state=="NY") %>%
  time_decompose(loansPerCapita, method = "stl") %>%
  anomalize(remainder, method = "iqr") %>%
  plot_anomalies() +
  labs(title = "Anomaly detection for loansPerCapita in New York") +
  xlab("Year") + ylab("Total loans per capita ") +
  scale_x_date(date_breaks = "years" , date_labels = "%y")

## frequency = 12 months

## trend = 31 months

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

anomalyPlotNY
```

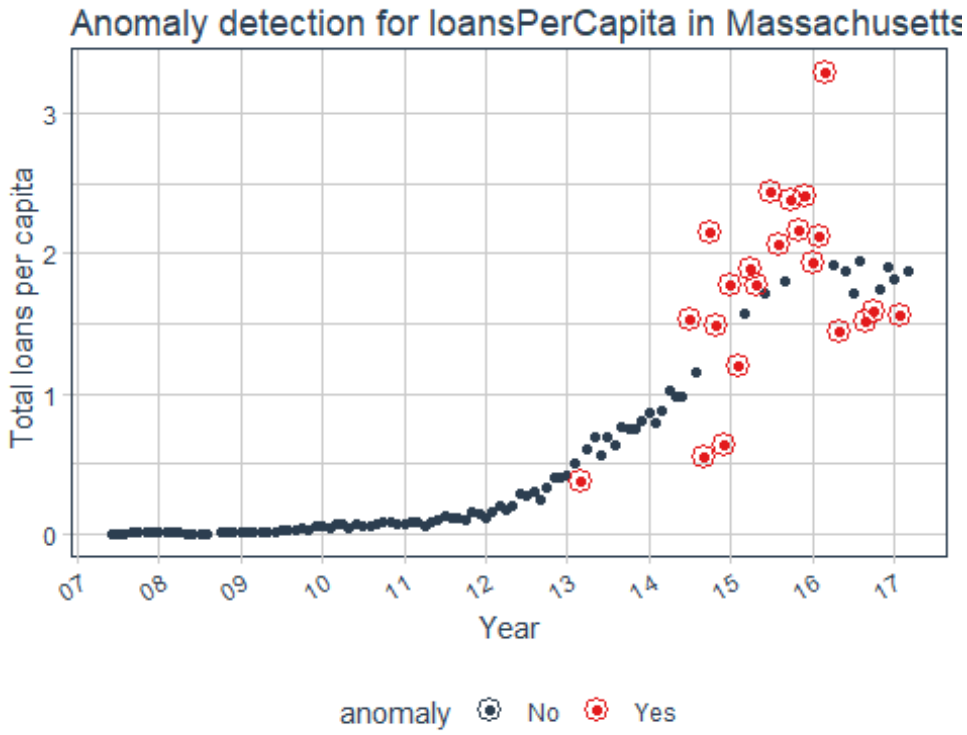


```
anomalyPlotCO <-
  tsLCOrg2 %>% as_tbl_time(index = date) %>%
  filter(state=="CO") %>%
  time_decompose(loansPerCapita, method = "stl") %>%
  anomalize(remainder, method = "iqr") %>%
  plot_anomalies() +
  labs(title = "Anomaly detection for loansPerCapita in Colorado") +
  xlab("Year") + ylab("Total loans per capita ") +
  scale_x_date(date_breaks = "years" , date_labels = "%y")

## frequency = 12 months

## trend = 30 months

anomalyPlotCO
```



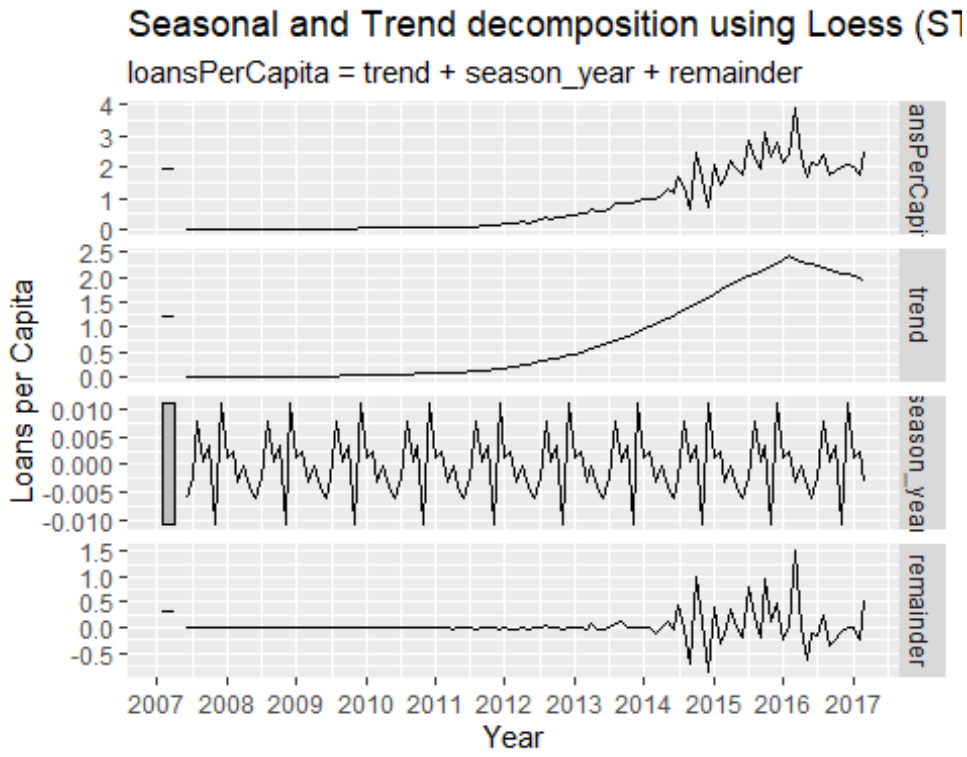

```
anomalyPlotMA <-
  tsLCOrg2 %>% as_tbl_time(index = date) %>%
  filter(state=="MA") %>%
  time_decompose(loansPerCapita, method = "stl") %>%
  anomalize(remainder, method = "iqr") %>%
  plot_anomalies() +
  labs(title = "Anomaly detection for loansPerCapita in Massachusetts ") +
  xlab("Year") + ylab("Total loans per capita ") +
  scale_x_date(date_breaks = "years" , date_labels = "%y")

## frequency = 12 months
## trend = 30 months
```

```
anomalyPlotMA
```



Question 2) (c)

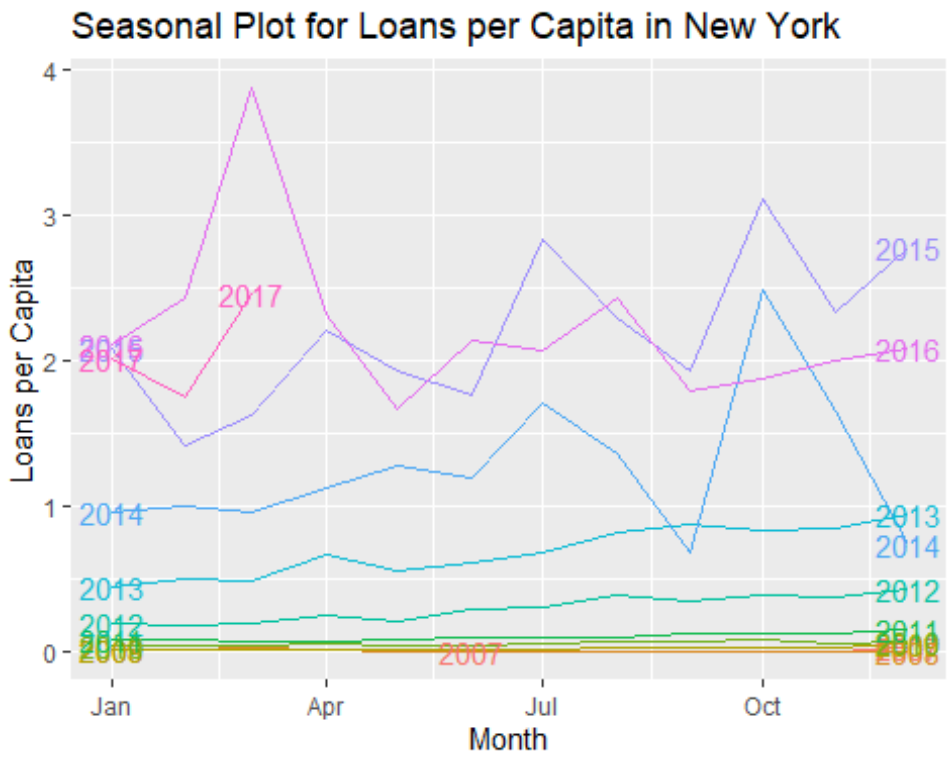
```
#tsLCNy

tsLCNy <- tsLCNy %>%
  as_tsibble(index = date, key= state)

tsLCNy$date <-  yearmonth(tsLCNy$date)

#tsLCNy

plotStlNy <-

  tsLCNy %>%
  model(STL(loansPerCapita ~ trend() + season(window='periodic'), robust = TR
UE)) %>%  #removed window= 10
  components() %>%
  autoplot() +
  xlab("Year") + ylab("Loans per Capita") +
  ggtitle("Seasonal and Trend decomposition using Loess (STL decomposition)")
+
  scale_x_date(date_breaks = "years" , date_labels = "%Y")

plotStlNy
```



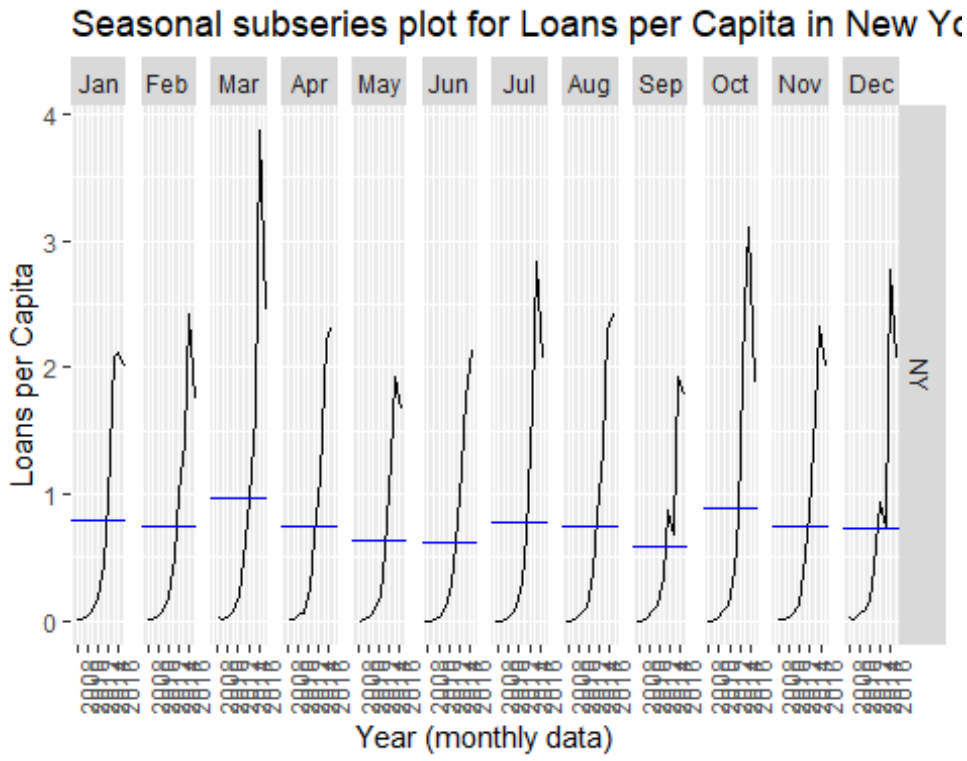
#March 2016 is when the trend reverses

Question 2) (d)

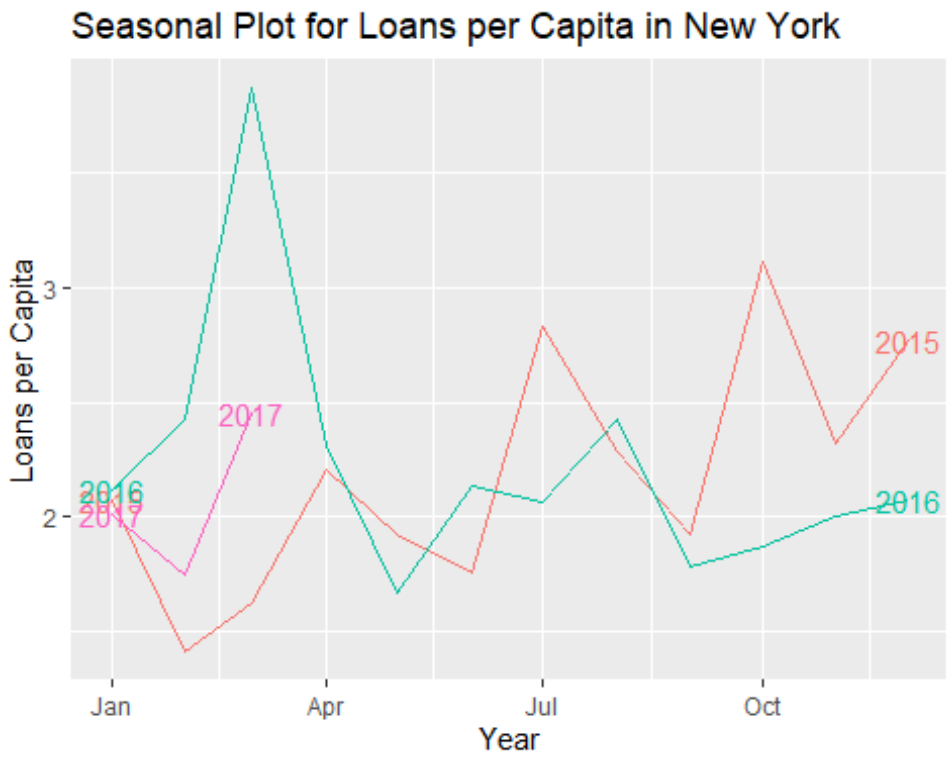
```
#Seasonal plot

plotNySeason <-
  tsLCNy %>%
  gg_season(loansPerCapita, labels = "both") +
  xlab("Month") + ylab("Loans per Capita") +
  ggtitle("Seasonal Plot for Loans per Capita in New York ")

plotNySeason
```



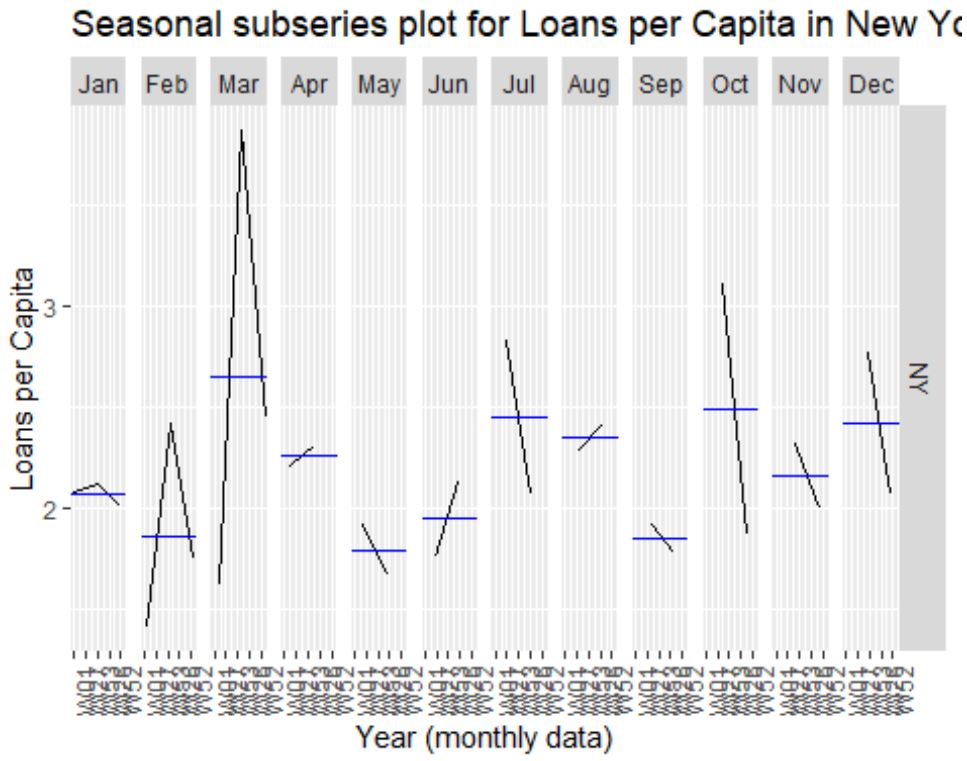

```
#Seasonal subseries

plotNySeasonSub <-
  tsLCNy %>%
  gg_subseries(loansPerCapita) +
  ylab("Loans per Capita") +
  xlab("Year (monthly data)") +
  ggtitle("Seasonal subseries plot for Loans per Capita in New York  ")

plotNySeasonSub
```




```
tsLCNyLast3 <-  tsLCNy %>%
  filter(year(tsLCNy$date)>= 2015)

#Seasonal plot

plotNy3Season <-
  tsLCNyLast3 %>%
  gg_season(loansPerCapita, labels = "both") +
  xlab("Year") + ylab("Loans per Capita") +
  ggtitle("Seasonal Plot for Loans per Capita in New York ")

plotNy3Season
```

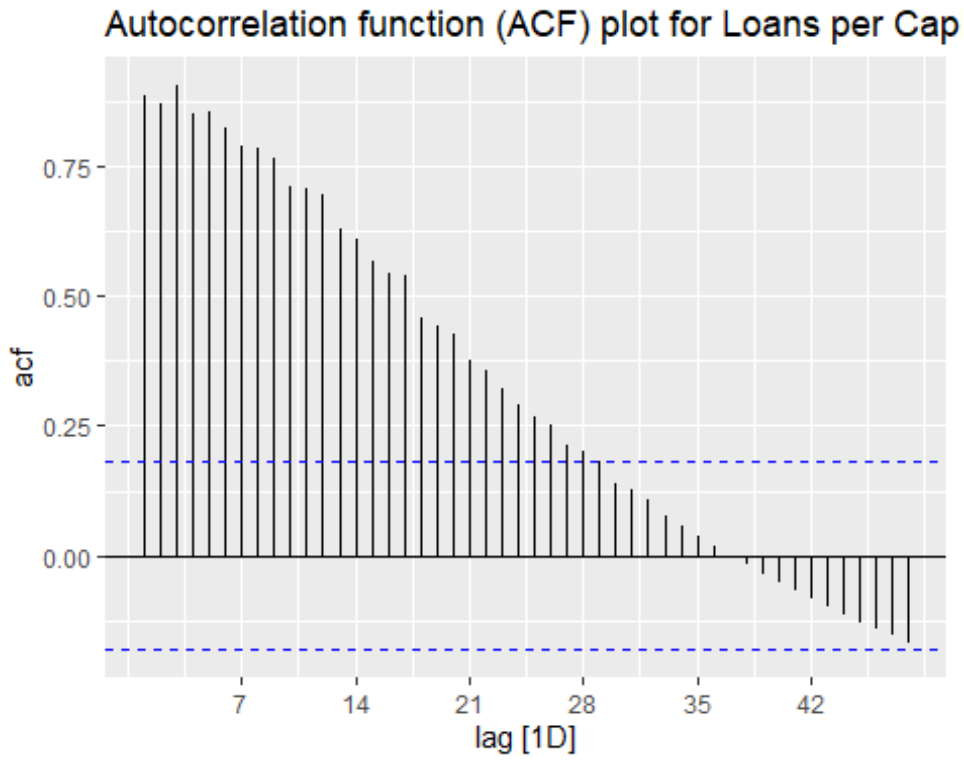



```
#Seasonal subsery

plotNy3SeasonSub <-
  tsLCNyLast3 %>%
  gg_subseries(loansPerCapita) +
  ylab("Loans per Capita") +
  xlab("Year (monthly data)") +
  ggtitle("Seasonal subseries plot for Loans per Capita in New York ")

plotNy3SeasonSub
```

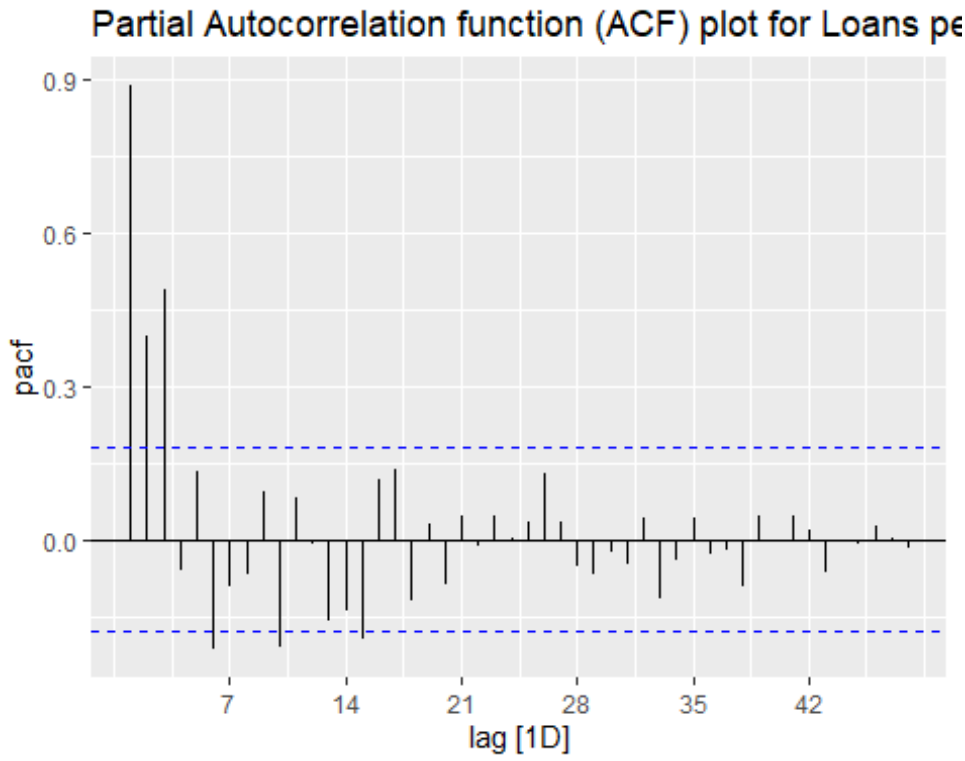


Question 2) (e)

```
#Autocorrelation Function

plotNyACF <-
  tsLCNy %>%
  ACF(loansPerCapita, lag_max = 48) %>%
  autoplot() +
  ggtitle("Autocorrelation function (ACF) plot for Loans per Capita in New Yo
rk ")

plotNyACF
```

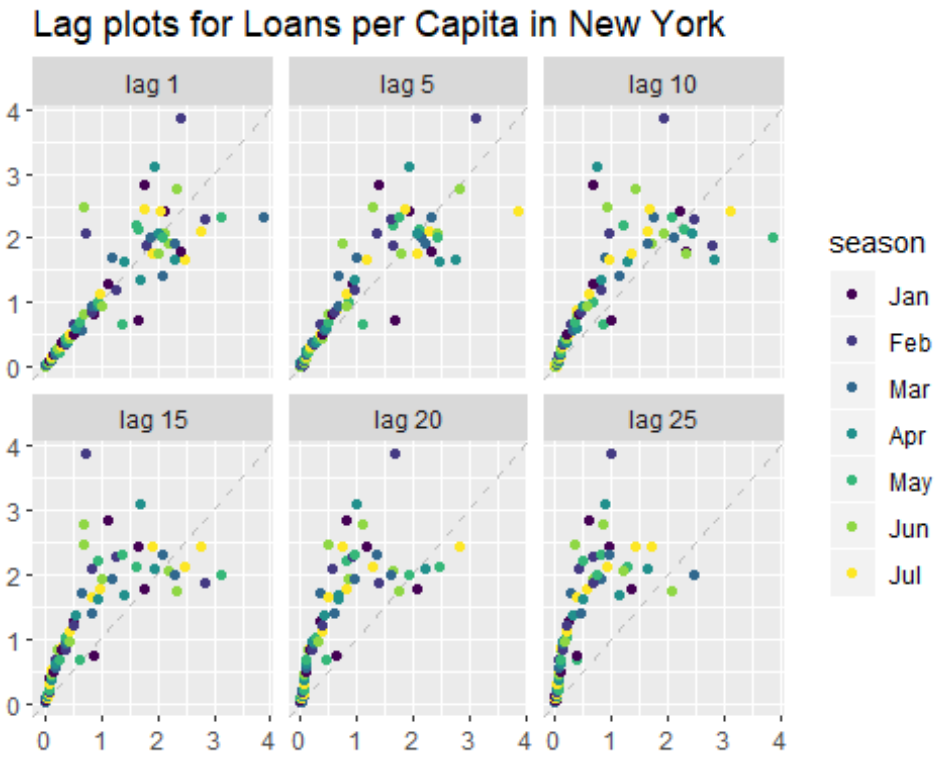



```
#Partial Autocorrelation Function

plotNyPACF <-
  tsLCNy %>%
  PACF(loansPerCapita, lag_max = 48) %>%
  autoplot() +
  ggtitle("Partial Autocorrelation function (ACF) plot for Loans per Capita i
n New York ")

plotNyPACF
```



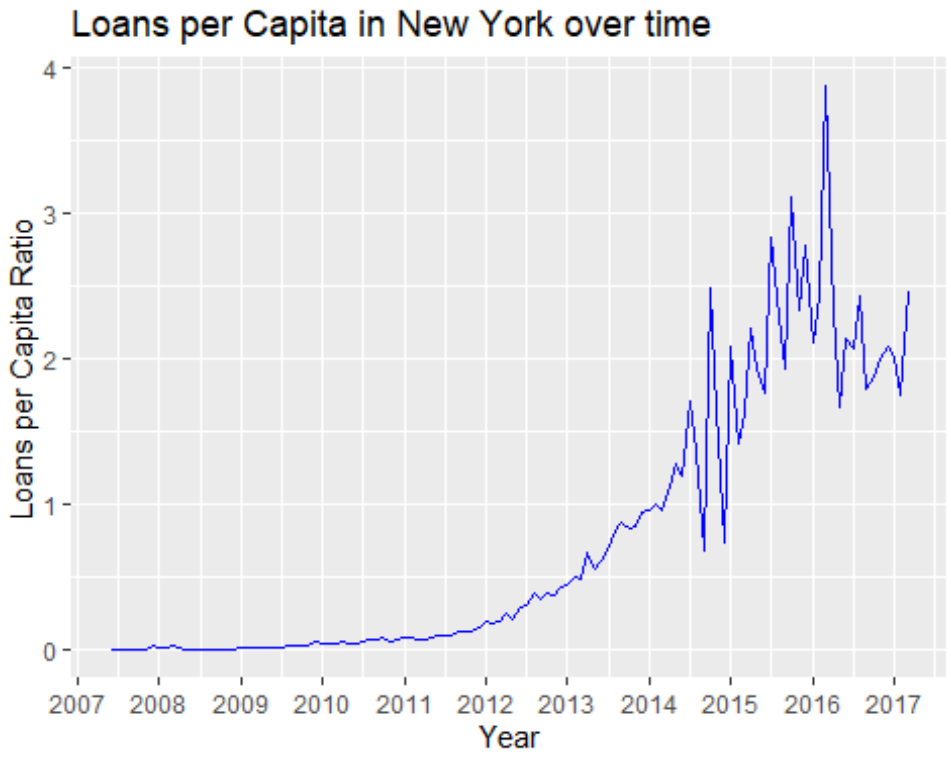
Question 2) (f)

```
#Lag plots for Lag 1,5,10,15,20,25

plotNyACF2 <-
  tsLCNy %>%
  gg_lag(loansPerCapita, lags = c(1,5,10,15,20,25), geom='point') +
  xlab(NULL) + ylab(NULL) +
  ggtitle("Lag plots for Loans per Capita in New York")

plotNyACF2
```

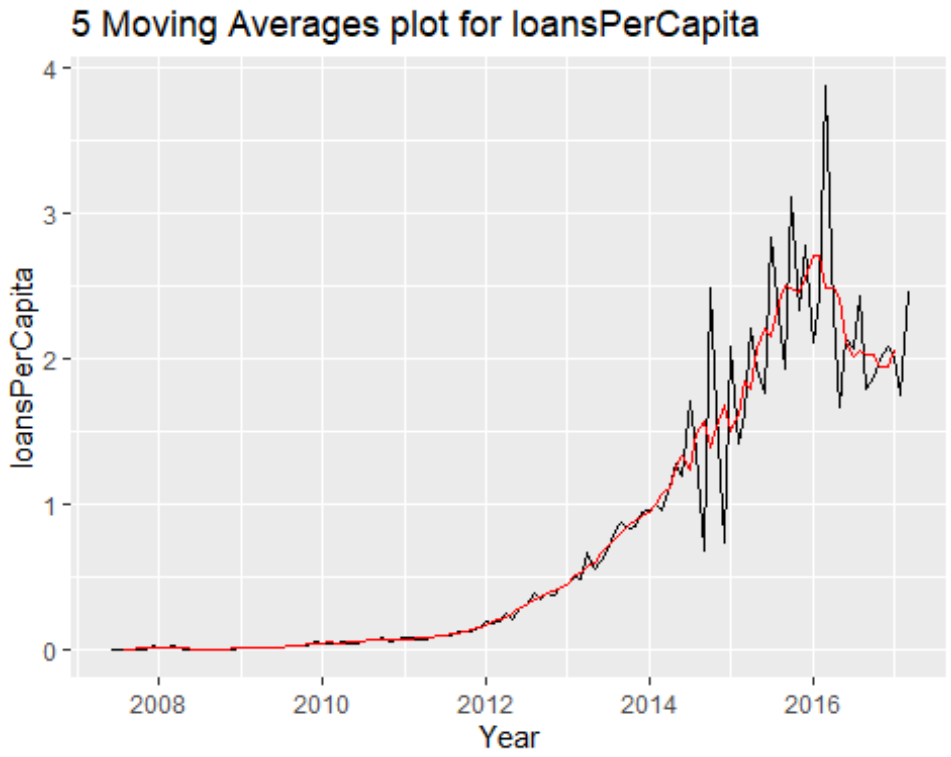


Question 2) (g)

```
plotNy <-
  tsLCNy %>%
  autoplot(.vars= loansPerCapita, colour= "blue")+
  xlab("Year") + ylab("Loans per Capita Ratio") +
  ggtitle("Loans per Capita in New York over time") +
  scale_x_date(date_breaks = "years" , date_labels = "%Y")

plotNy
```

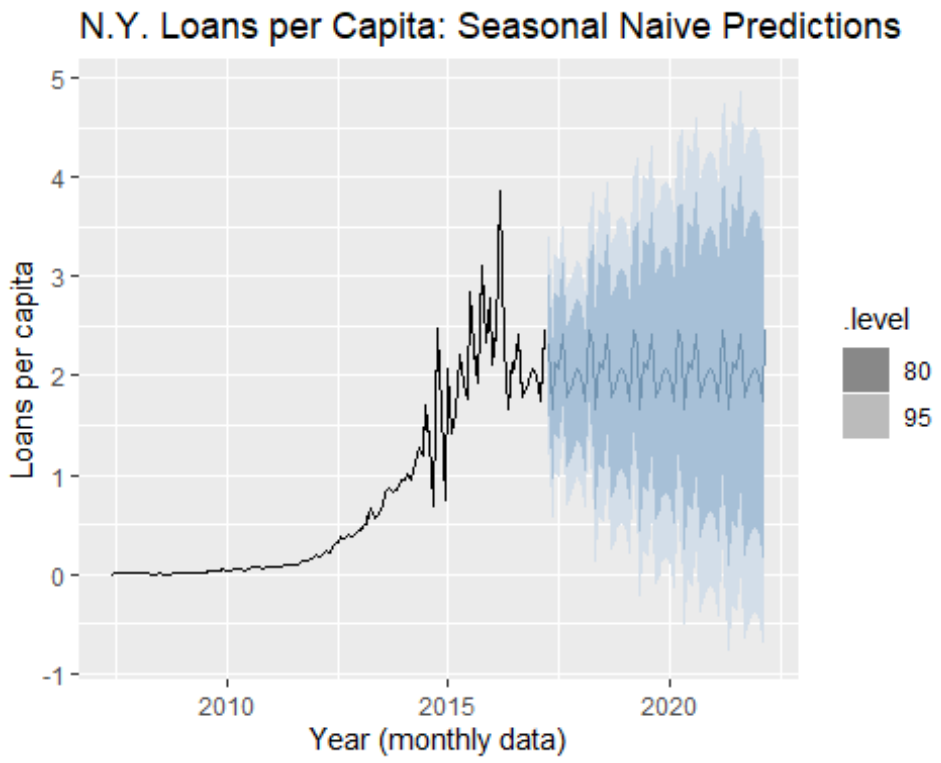



```
#5th order moving average smoothing

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

plotMAvg %>%
  autoplot(loansPerCapita) +
  autolayer(plotMAvg, `5-MA`, color='red') +
  xlab("Year") + ylab("loansPerCapita") +
  ggtitle("5 Moving Averages plot for loansPerCapita") +
  guides(colour=guide_legend(title="series"))

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

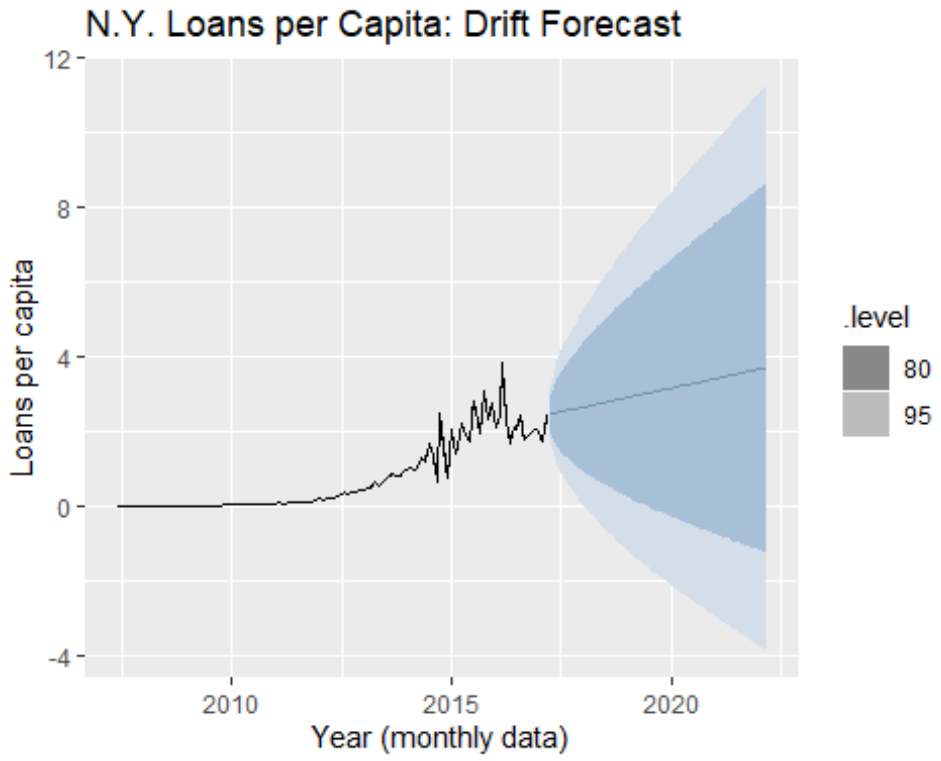


Question 3) (a)

```
#Seasonal Naive

plotNySNaive <-
  tsLCNy %>%
  model(SNAIVE(loansPerCapita)) %>%
  forecast(h = "5 years") %>%
  autoplot(tsLCNy, colour = "#769ECB") + #level = NULL,
  geom_line(linetype = 'dashed', colour = '#000000') +
  xlab("Year (monthly data)") + ylab("Loans per capita") +
  ggtitle("N.Y. Loans per Capita: Seasonal Naive Predictions")

plotNySNaive
```




```
#Drift Forecast

plotNyDrift <-
  tsLCNy %>%
  model(RW(loansPerCapita ~ drift())) %>%
  forecast(h = "5 years") %>%
  autoplot(tsLCNy, colour = "#769ECB") + #level = NULL,
  geom_line(linetype = 'dashed', colour = '#000000') +
  xlab("Year (monthly data)") + ylab("Loans per capita") +
  ggtitle("N.Y. Loans per Capita: Drift Forecast")

plotNyDrift
```



Question 3) (b)

```
#Time series regression using trend and season

fitNy <-
  tsLCNy %>%
  model(TSLM(loansPerCapita ~ trend() + season()))

report(fitNy)

## Series: loansPerCapita
## Model: TSLM
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.6856 -0.3684 -0.0555  0.2601  1.9152
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)    -0.649286   0.171877  -3.778 0.000263 ***
## trend()         0.023300   0.001301  17.908  < 2e-16 ***
## season()year2  -0.074354   0.214643  -0.346 0.729730
## season()year3   0.134285   0.214654   0.626 0.532943
## season()year4   0.019519   0.220555   0.088 0.929649
## season()year5  -0.106107   0.220536  -0.481 0.631421
## season()year6  -0.016274   0.214832  -0.076 0.939760
## season()year7   0.121947   0.214781   0.568 0.571401
```

```
## season()year8   0.069353   0.214737   0.323 0.747364
## season()year9  -0.117326   0.214702  -0.546 0.585910
## season()year10  0.167261   0.214674   0.779 0.437651
## season()year11 -0.005619   0.214654  -0.026 0.979166
## season()year12 -0.044524   0.214643  -0.207 0.836072
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4799 on 105 degrees of freedom
## Multiple R-squared: 0.7566,  Adjusted R-squared: 0.7288
## F-statistic:  27.2 on 12 and 105 DF, p-value: < 2.22e-16
```

```
fitNy
```

```
## # A mable: 1 x 2
## # Key:     state [1]
##   state `TSLM(loansPerCapita ~ trend() + season())`
##   <fct> <model>
## 1 NY    <TSLM>
```

```
head(tsLCNy)
```

```
## # A tsibble: 6 x 23 [1D]
## # Key:       state [1]
##         date state NAME  Total_Population avgLoans totalLoans avgTerm avgI
ntRate
##        <mth> <fct> <fct> <chr>               <dbl>      <dbl>   <dbl>
<dbl>
## 1   2007 Jun NY    New ~ 19378102            3381.      13525      36
8.78
## 2   2007 Jul NY    New ~ 19378102            8611.      77500      36
11.0
## 3   2007 Aug NY    New ~ 19378102            7358.      95650      36
11.2
## 4   2007 Sep NY    New ~ 19378102            8389.      92275      36
11.3
## 5   2007 Oct NY    New ~ 19378102            8804.     105650      36
12.9
## 6   2007 Nov NY    New ~ 19378102            7634.     122150      36
11.5
## # ... with 15 more variables: avgGrade <dbl>, avgEmpLength <dbl>,
## #   avgAnnualInc <dbl>, avgVerifStatus <dbl>, avgHomeOwner <dbl>,
## #   avgOpenAcc <dbl>, avgRevolBal <dbl>, avgRevolUtil <dbl>, avgTotalAcc <
dbl>,
## #   countOfLoans <dbl>, loansPerCapita <dbl>, NYCPI <dbl>,
## #   NYUnemployment <dbl>, NYCondoPriceIdx <dbl>, NYSnapBenefits <dbl>
```

```
str(tsLCNy)
```

```
## Classes 'tbl_ts', 'tbl_df', 'tbl' and 'data.frame':  118 obs. of  23 varia
bles:
```

```
##  $ date            : yearmonth, format: "2007 Jun" "2007 Jul" ...
##  $ state           : Factor w/ 51 levels "AK","AL","AR",..: 34 34 34 34 34
34 34 34 34 34 ...
##  $ NAME            : Factor w/ 51 levels "Alabama","Alaska",..: 32 32 32 3
2 32 32 32 32 32 32 ...
##  $ Total_Population: chr  "19378102" "19378102" "19378102" "19378102" ...
##  $ avgLoans        : num  3381 8611 7358 8389 8804 ...
##  $ totalLoans      : num  13525 77500 95650 92275 105650 ...
##  $ avgTerm         : num  36 36 36 36 36 36 36 36 36 36 ...
##  $ avgIntRate      : num  8.78 11.05 11.15 11.25 12.85 ...
##  $ avgGrade        : num  1.75 3.22 3.31 3.45 4.17 ...
##  $ avgEmpLength    : num  2.25 2.78 1.23 3 2.33 ...
##  $ avgAnnualInc    : num  113333 101875 69364 55806 90517 ...
##  $ avgVerifStatus  : num  0 0 0 0 0 ...
##  $ avgHomeOwner    : num  0 0 0.0769 0.1818 0 ...
##  $ avgOpenAcc      : num  NaN 11.71 8.2 6 8.17 ...
##  $ avgRevolBal     : num  0 8466 10374 9423 10339 ...
##  $ avgRevolUtil    : num  NaN 42.1 56.9 42.5 52.7 ...
##  $ avgTotalAcc     : num  NaN 19.9 13.2 11.7 18 ...
##  $ countOfLoans    : num  4 9 13 11 12 16 36 23 21 43 ...
##  $ loansPerCapita  : num  0.000698 0.003999 0.004936 0.004762 0.005452 ...
##  $ NYCPI           : num  660 661 660 660 661 ...
##  $ NYUnemployment  : num  4.5 4.6 4.7 4.7 4.8 4.8 4.8 4.8 4.9 4.9 ...
##  $ NYCondoPriceIdx : num  228 228 227 226 226 ...
##  $ NYSnapBenefits  : num  1801707 1792916 1816805 1823494 1825759 ...
##  - attr(*, "key")=Classes 'tbl_df', 'tbl' and 'data.frame':  1 obs. of  2
variables:
##   ..$ state: Factor w/ 51 levels "AK","AL","AR",..: 34
##   ..$ .rows:List of 1
##   .. ..$ : int  1 2 3 4 5 6 7 8 9 10 ...
##   ..- attr(*, ".drop")= logi TRUE
##  - attr(*, "index")= chr "date"
##   ..- attr(*, "ordered")= logi TRUE
##  - attr(*, "index2")= chr "date"
##  - attr(*, "interval")=List of 12
##   ..$ year       : num 0
##   ..$ quarter    : num 0
##   ..$ month      : num 0
##   ..$ week       : num 0
##   ..$ day        : num 1
##   ..$ hour       : num 0
##   ..$ minute     : num 0
##   ..$ second     : num 0
##   ..$ millisecond: num 0
##   ..$ microsecond: num 0
##   ..$ nanosecond : num 0
##   ..$ unit       : num 0
##   ..- attr(*, "class")= chr "interval"

head(tsLCNy)
```

```
## # A tsibble: 6 x 23 [1D]
## # Key:       state [1]
##         date state NAME  Total_Population avgLoans totalLoans avgTerm avgI
ntRate
##        <mth> <fct> <fct> <chr>               <dbl>      <dbl>   <dbl>
<dbl>
## 1   2007 Jun NY    New ~ 19378102            3381.      13525      36
8.78
## 2   2007 Jul NY    New ~ 19378102            8611.      77500      36
11.0
## 3   2007 Aug NY    New ~ 19378102            7358.      95650      36
11.2
## 4   2007 Sep NY    New ~ 19378102            8389.      92275      36
11.3
## 5   2007 Oct NY    New ~ 19378102            8804.     105650      36
12.9
## 6   2007 Nov NY    New ~ 19378102            7634.     122150      36
11.5
## # ... with 15 more variables: avgGrade <dbl>, avgEmpLength <dbl>,
## #   avgAnnualInc <dbl>, avgVerifStatus <dbl>, avgHomeOwner <dbl>,
## #   avgOpenAcc <dbl>, avgRevolBal <dbl>, avgRevolUtil <dbl>, avgTotalAcc <
dbl>,
## #   countOfLoans <dbl>, loansPerCapita <dbl>, NYCPI <dbl>,
## #   NYUnemployment <dbl>, NYCondoPriceIdx <dbl>, NYSnapBenefits <dbl>
```

```
tsLCNy$Total_Population <-  as.numeric(tsLCNy$Total_Population)
```

```
names(which(colSums(is.na(tsLCNy))>0))  #columns which have null values
```

```
## [1] "avgOpenAcc"   "avgRevolUtil" "avgTotalAcc"
```

```
#colnames(tsLCNy)

fitNyAllVar <-
  tsLCNy %>%
  model(TSLM(loansPerCapita ~  trend() + season() + avgIntRate  + avgLoans+
avgTerm + avgGrade + avgEmpLength + avgAnnualInc + avgVerifStatus + avgHomeOw
ner + avgRevolBal + countOfLoans + NYCPI + NYUnemployment + NYCondoPriceIdx +
NYSnapBenefits))

report(fitNyAllVar)
```

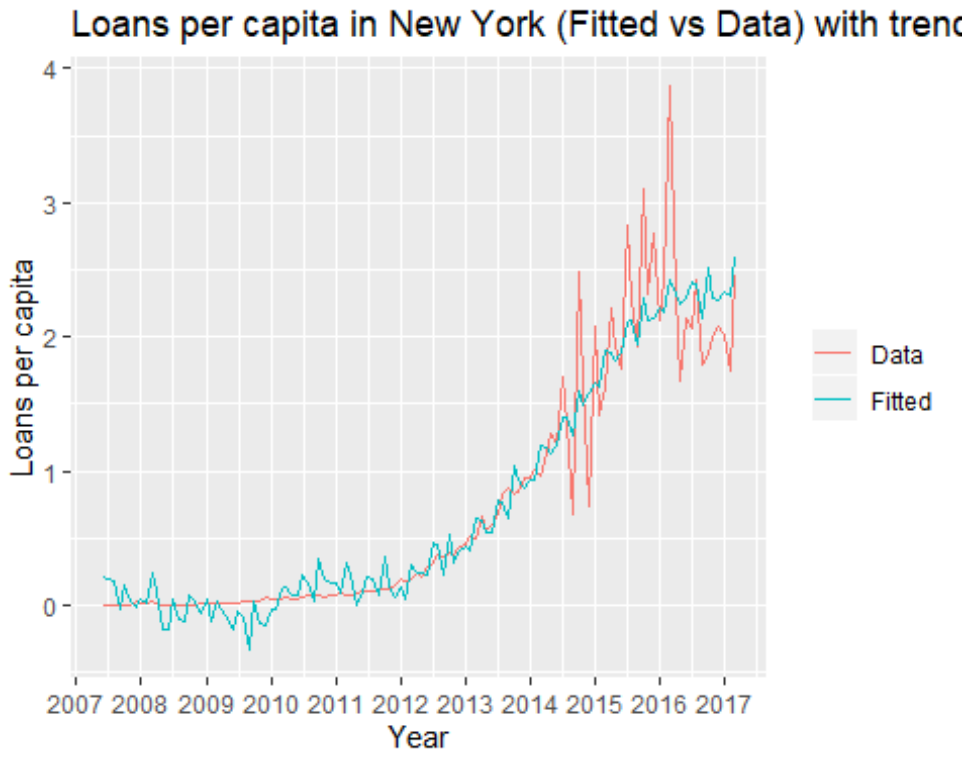
```
## Series: loansPerCapita
## Model: TSLM
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.116941 -0.013873 -0.002317  0.015466  0.106489
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept)     -1.994e+00  7.585e-01  -2.629   0.0101 *
## trend()         -4.680e-03  1.068e-03  -4.381 3.15e-05 ***
## season()year2   -2.198e-03  1.584e-02  -0.139   0.8899
## season()year3   -8.093e-03  1.607e-02  -0.504   0.6157
## season()year4   -3.541e-02  1.664e-02  -2.128   0.0360 *
## season()year5   -2.458e-02  1.694e-02  -1.451   0.1502
## season()year6   -2.404e-02  1.719e-02  -1.399   0.1652
## season()year7   -4.073e-02  1.715e-02  -2.375   0.0197 *
## season()year8   -3.259e-02  1.734e-02  -1.879   0.0634 .
## season()year9   -2.876e-02  1.645e-02  -1.748   0.0839 .
## season()year10  -3.159e-02  1.675e-02  -1.886   0.0625 .
## season()year11  -2.827e-02  1.594e-02  -1.774   0.0795 .
## season()year12  -3.725e-02  1.574e-02  -2.366   0.0201 *
## avgIntRate      -2.663e-02  8.136e-03  -3.273   0.0015 **
## avgLoans         1.293e-05  3.163e-06   4.088 9.37e-05 ***
## avgTerm         -1.271e-03  3.140e-03  -0.405   0.6865
## avgGrade         2.547e-02  2.151e-02   1.184   0.2396
## avgEmpLength    -1.473e-02  8.495e-03  -1.734   0.0864 .
## avgAnnualInc     7.383e-07  3.500e-07   2.109   0.0377 *
## avgVerifStatus   5.503e-02  3.105e-02   1.772   0.0797 .
## avgHomeOwner    -6.320e-02  9.546e-02  -0.662   0.5096
## avgRevolBal      2.271e-06  8.455e-07   2.686   0.0086 **
## countOfLoans     7.836e-04  9.445e-06  82.959  < 2e-16 ***
## NYCPI            1.969e-03  9.155e-04   2.150   0.0342 *
## NYUnemployment   1.482e-02  1.610e-02   0.921   0.3597
## NYCondoPriceIdx  2.682e-03  1.195e-03   2.245   0.0272 *
## NYSnapBenefits   9.710e-08  7.245e-08   1.340   0.1835
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.03458 on 91 degrees of freedom
## Multiple R-squared: 0.9989,  Adjusted R-squared: 0.9986
## F-statistic:  3192 on 26 and 91 DF, p-value: < 2.22e-16

fitNy2 <-
  tsLCNy %>%
  model(TSLM(loansPerCapita ~  trend() + season() + avgIntRate + NYCPI + NYCo
ndoPriceIdx))

report(fitNy2)

## Series: loansPerCapita
## Model: TSLM
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.836552 -0.135070 -0.005781  0.091424  1.444271
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
```

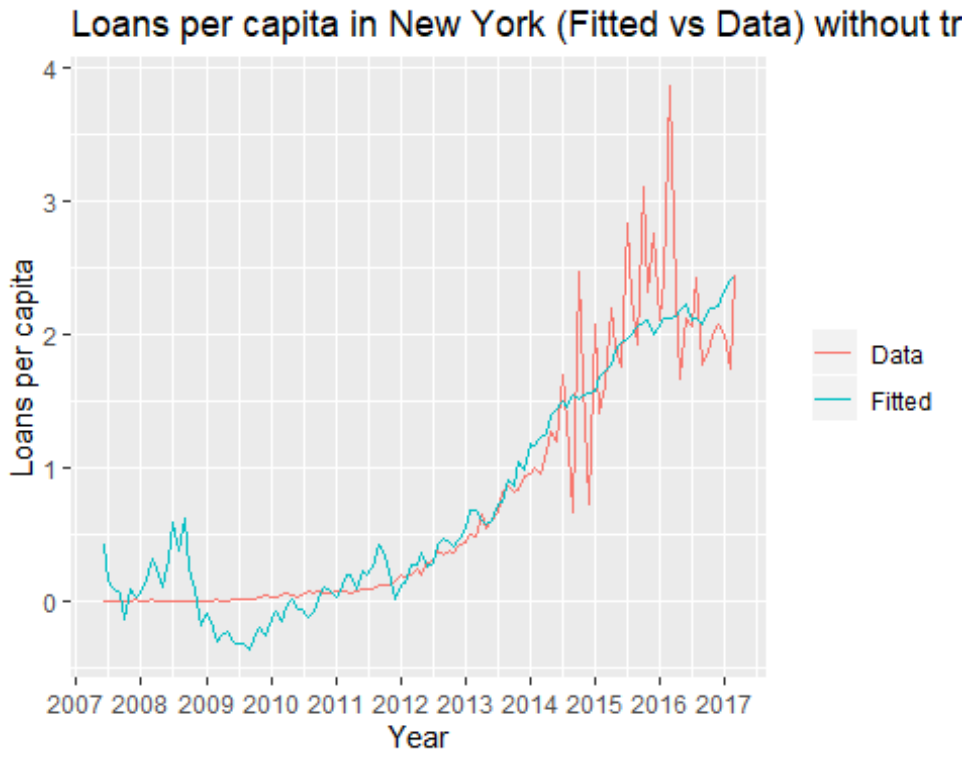
```
## (Intercept)      5.859105   3.423590   1.711   0.0900 .
## trend()          0.029817   0.005130   5.812 7.08e-08 ***
## season()year2   -0.063744   0.133640  -0.477   0.6344
## season()year3    0.177539   0.134228   1.323   0.1889
## season()year4    0.120125   0.138055   0.870   0.3863
## season()year5    0.034024   0.139149   0.245   0.8073
## season()year6    0.040913   0.136302   0.300   0.7647
## season()year7    0.188790   0.136000   1.388   0.1681
## season()year8    0.159958   0.135956   1.177   0.2421
## season()year9   -0.054590   0.136311  -0.400   0.6896
## season()year10   0.223708   0.134324   1.665   0.0989 .
## season()year11  -0.005926   0.133668  -0.044   0.9647
## season()year12  -0.045744   0.134090  -0.341   0.7337
## avgIntRate      -0.076552   0.033088  -2.314   0.0227 *
## NYCPI           -0.013199   0.005182  -2.547   0.0123 *
## NYCondoPriceIdx  0.016023   0.001767   9.070 9.28e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2984 on 102 degrees of freedom
## Multiple R-squared: 0.9086,  Adjusted R-squared: 0.8952
## F-statistic:  67.6 on 15 and 102 DF, p-value: < 2.22e-16
```

Question 3) (c)

```
#Plot of fitted values from trend + season + variables model

fitNy2Fitted <-
  augment(fitNy2) %>%   #converts model to a dataframe
  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("Loans per capita in New York (Fitted vs Data) with trend and seaso
n") +
  scale_x_date(date_breaks = "years" , date_labels = "%Y") +
  guides(colour=guide_legend(title=NULL))

fitNy2Fitted
```

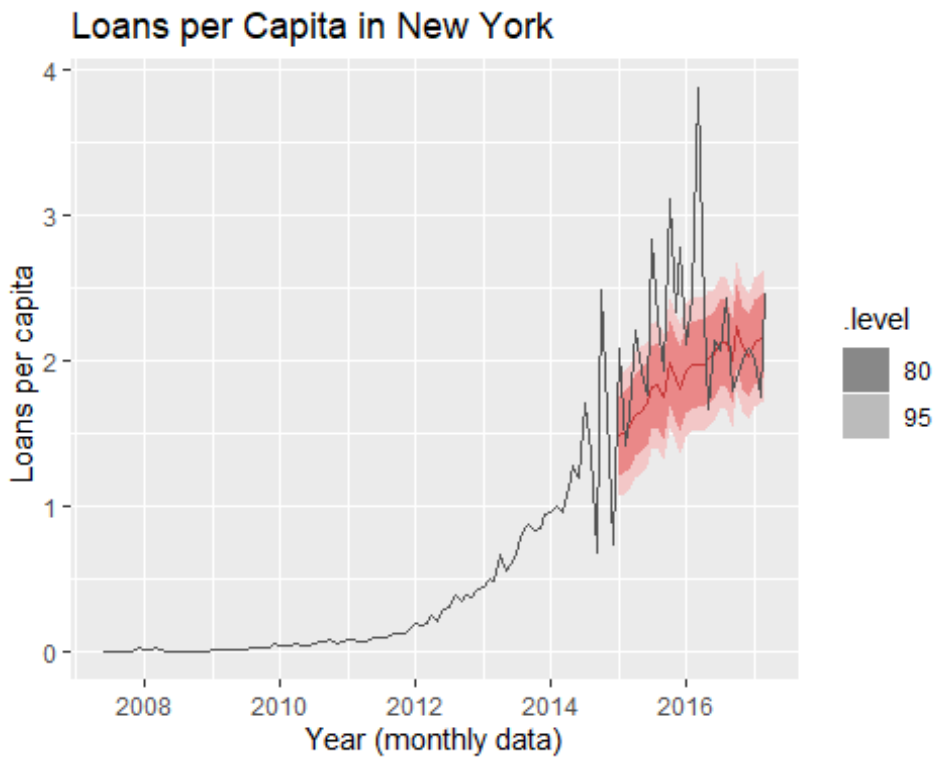



```
fitNy3 <-
  tsLCNy %>%
  model(TSLM(loansPerCapita ~ avgIntRate + NYCPI + NYCondoPriceIdx))

#report(fitNy3)

#Plot of fitted values from trend + season + variables model

fitNy3Fitted <-
  augment(fitNy3) %>%   #converts model to a dataframe
  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("Loans per capita in New York (Fitted vs Data) without trend and se
ason") +
  scale_x_date(date_breaks = "years" , date_labels = "%Y") +
  guides(colour=guide_legend(title=NULL))

fitNy3Fitted
```

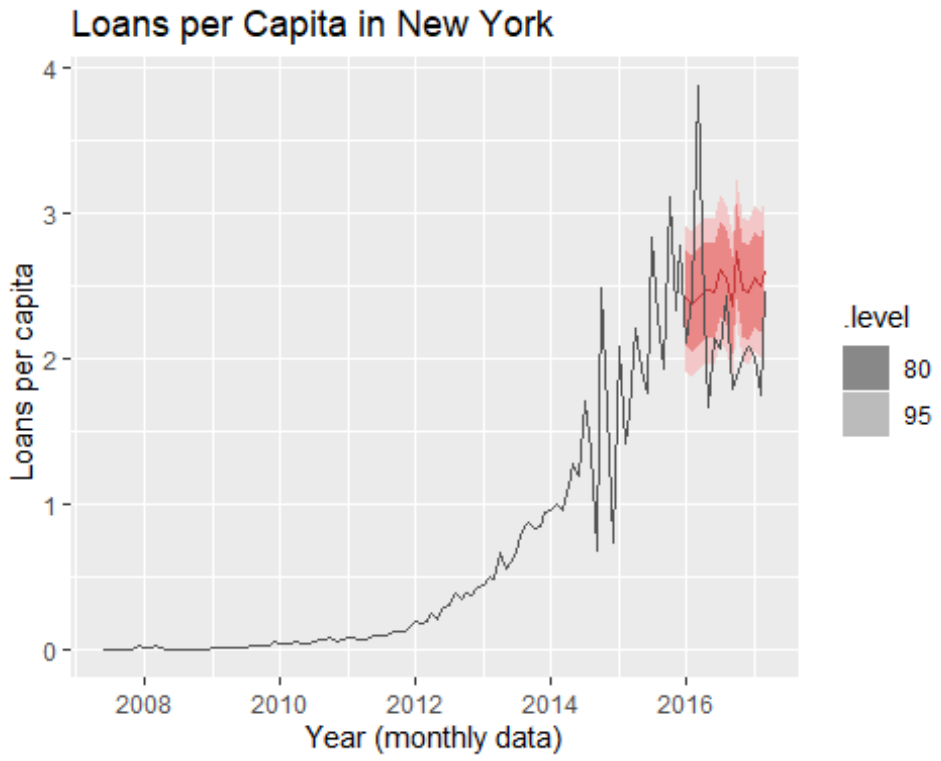


Question 3) (d)

```
#1st split

plotLcNyPredicted1 <-
  tsLCNy %>%
  filter(year(date) <= '2014') %>%
  model(TSLM(loansPerCapita ~ trend() + season() + avgIntRate  + NYCPI + NYCo
ndoPriceIdx)) %>%
  forecast(new_data = tsLCNy %>% filter(year(date) > '2014')) %>%
  autoplot(tsLCNy, colour = "#960A0A") +
  geom_line(colour = '#535353') +
  xlab("Year (monthly data)") + ylab("Loans per capita ") +
  ggtitle("Loans per Capita in New York")

plotLcNyPredicted1
```

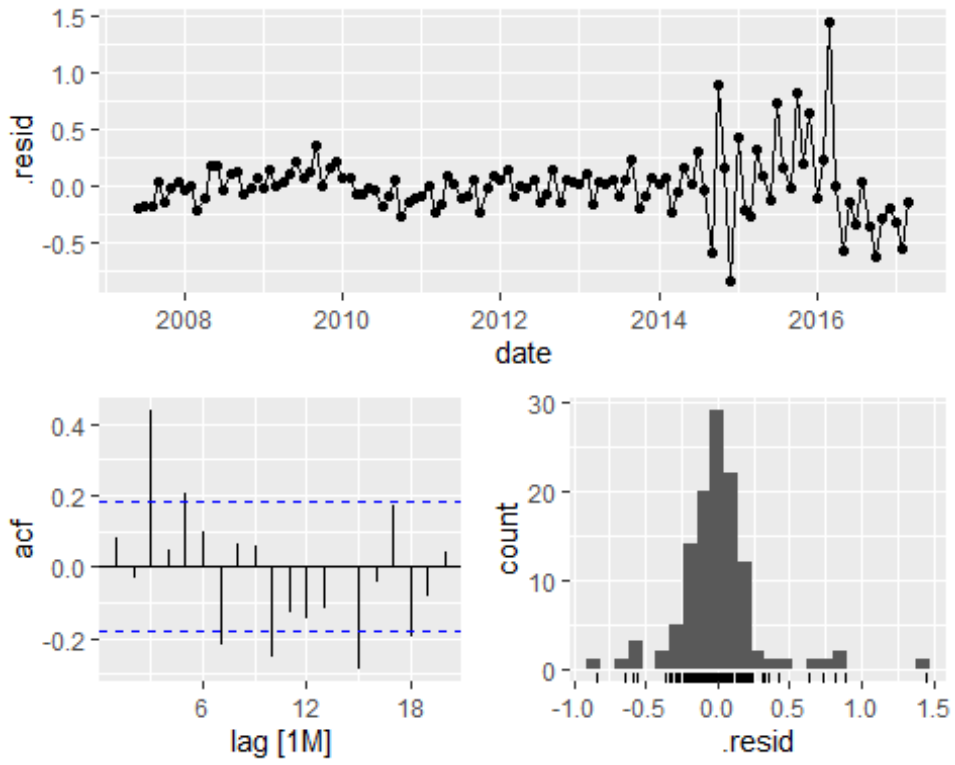



```
#2nd split

plotLcNyPredicted2 <-
  tsLCNy %>%
  filter(year(date) <= '2015') %>%
  model(TSLM(loansPerCapita ~ trend() + season() + avgIntRate  + NYCPI + NYCo
ndoPriceIdx)) %>%
  forecast(new_data = tsLCNy %>% filter(year(date) > '2015')) %>%
  autoplot(tsLCNy, colour = "#960A0A") +
  geom_line(colour = '#535353') +
  xlab("Year (monthly data)") + ylab("Loans per capita ") +
  ggtitle("Loans per Capita in New York")

plotLcNyPredicted2
```



Question 3) (e)

```
fitNy2 %>% gg_tsresiduals()
```



Question 3) (f)

```
#fitEconARIMA <-
  #tsEcon %>%
  #model(fitArima = ARIMA(Consumption ~ PDQ(0,0,0), #pdq(3,0,0) +
              #stepwise = FALSE, approximation = FALSE)
  #)
#report(fitEconARIMA)
```

```
fitLCARIMA <-
  tsLCNy %>%
  model(fitArima = ARIMA(loansPerCapita ~ PDQ(0,0,0) + avgIntRate +NYCPI + NY
CondoPriceIdx, stepwise = FALSE, approximation = FALSE))
report(fitLCARIMA)
```

```
## Series: loansPerCapita
## Model: LM w/ ARIMA(2,0,3) errors
##
## Coefficients:
##          ar1     ar2      ma1      ma2     ma3  avgIntRate    NYCPI
##       0.6683  0.3181  -0.4395  -0.5296  0.6464     -0.0037  -0.0013
## s.e.  0.1168  0.1163   0.0927   0.0618  0.0648      0.0281   0.0022
##       NYCondoPriceIdx
##                0.0086
## s.e.           0.0061
##
## sigma^2 estimated as 0.06635:  log likelihood=-5.68
## AIC=29.35   AICc=31.02   BIC=54.29
```

```
#p-values

2*pt(-abs(-0.0037/0.0281), nrow(tsLCNy)-3) #avgIntRate
```

```
## [1] 0.8954733
```

```
2*pt(-abs(-0.0013/0.0022), nrow(tsLCNy)-3)   #NYCPI
```

```
## [1] 0.5557409
```

```
2*pt(-abs(0.0086/0.0061), nrow(tsLCNy)-3)   #NYCondoPriceIdx
```

```
## [1] 0.1612869
```

Question 3) (g)

```
tsLCNy %>%
  features(loansPerCapita, unitroot_kpss)
```

```
## # A tibble: 1 x 3
##   state kpss_stat kpss_pvalue
```

```
##   <fct>     <dbl>       <dbl>
## 1 NY         2.09        0.01
```

Question 3) (h)

```
tsLCNy %>%
  features(difference(loansPerCapita), unitroot_kpss)

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

Question 3) (g)

```
fitLCARIMA2 <-
  tsLCNy %>%
  model(fitArima = ARIMA(loansPerCapita ~ pdq(,1,) + PDQ(0,0,0) + avgIntRate
+NYCPI + NYCondoPriceIdx, stepwise = FALSE, approximation = FALSE))
report(fitLCARIMA2)

## Series: loansPerCapita
## Model: LM w/ ARIMA(0,1,4) errors
##
## Coefficients:
##           ma1      ma2     ma3      ma4  avgIntRate    NYCPI  NYCondoPrice
Idx
##       -0.8095  -0.2965  0.8164  -0.3097     -0.0099  -0.0034           0.0
095
## s.e.   0.0927   0.0899  0.0762   0.0992      0.0279   0.0044           0.0
066
##       intercept
##          0.0185
## s.e.     0.0103
##
## sigma^2 estimated as 0.06436:  log likelihood=-2.58
## AIC=23.16   AICc=24.84   BIC=48.02
```

Question 4 (a)

```
set.seed(333)
tsNyTrain <- tsLCNy %>% filter(date < '2016-03-01')
tsNyTest <- tsLCNy %>% filter(date >= '2016-03-01')

tsNyFitAll <-
  tsNyTrain %>%
  model(
    model1TimeTrendAndSeason = TSLM(loansPerCapita ~ trend() + season()),
    model2WithFeatures       = TSLM(loansPerCapita ~  trend() + season() + avg
IntRate + avgLoans + avgAnnualInc + avgRevolBal + countOfLoans + NYCPI + NYCo
ndoPriceIdx),
    model3fitArima = ARIMA(loansPerCapita ~ PDQ(0,0,0)),
```

```
    model4fitArima = ARIMA(loansPerCapita ~ PDQ(0,0,0) + avgIntRate +NYCPI +
NYCondoPriceIdx, stepwise = FALSE, approximation = FALSE))



tsNyPredictAll <-
  tsNyFitAll %>%
  forecast(new_data = tsNyTest)

accuracy(tsNyPredictAll, tsNyTest)



## # A tibble: 4 x 9
##   .model                    .type      ME   RMSE    MAE    MPE  MAPE  MASE  A
CF1
##   <chr>                     <chr>   <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl> <d
bl>
## 1 model1TimeTrendAndSeason Test    0.433  0.721 0.463  16.1   17.8   NaN 0.
323
## 2 model2WithFeatures       Test    -0.374  0.659 0.561 -21.8   26.6   NaN 0
.255
## 3 model3fitArima           Test   -0.765  0.950 0.898 -40.3   43.7   NaN 0.
323
## 4 model4fitARIMA           Test   -0.0630 0.5655 0.345  -6.86  15.1   NaN 0
.185
```

Question 4 (b)

```
set.seed(333)
tsNyTrain2 <- tsLCNy %>% filter(date < '2016-04-01')
tsNyTest2 <- tsLCNy %>% filter(date >= '2016-04-01')

tsNyFitAll2 <-
  tsNyTrain2 %>%
  model(
    model1TimeTrendAndSeason2 = TSLM(loansPerCapita ~ trend() + season()),
    model2WithFeatures2       = TSLM(loansPerCapita ~  trend() + season() + av
gIntRate + avgLoans + avgAnnualInc + avgRevolBal + countOfLoans + NYCPI + NYC
ondoPriceIdx),
    model3fitArima2 = ARIMA(loansPerCapita ~ PDQ(0,0,0)),
    model4fitArima2 = ARIMA(loansPerCapita ~ PDQ(0,0,0) + avgIntRate +NYCPI +
NYCondoPriceIdx, stepwise = FALSE, approximation = FALSE
    ))



tsNyPredictAll <-
  tsNyFitAll2 %>%
  forecast(new_data = tsNyTest2)

accuracy(tsNyPredictAll, tsNyTest)
```

```
## # A tibble: 4 x 9
##   .model                    .type     ME  RMSE   MAE   MPE  MAPE  MASE   AC
F1
##   <chr>                     <chr>  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  <db
l>
## 1 model1TimeTrendAndSeason2Test   0.234 0.352 0.272  10.2  12.3   NaN -0.2
45
## 2 model2WithFeatures2      Test  -0.556 0.608 0.556 -28.9  28.9   NaN -0.3
47
## 3 model3fitArima2          Test  -0.877 0.914 0.877 -44.6  44.6   NaN -0.2
12

## 4 model4fitArima2          Test  -0.263 0.369 0.297 -14.5  15.9   NaN -0.3
29
```

PART II]

Question 1) (a)

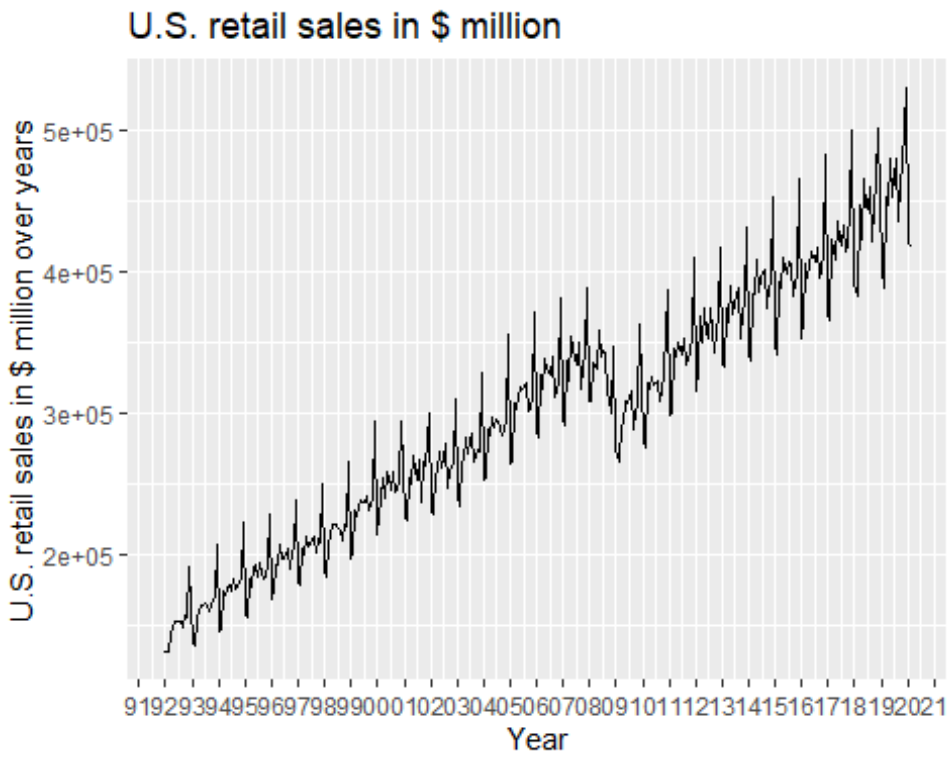
```
tsRetail <- read_csv("retailSales.csv")

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

#tsRetail

tsRetail$date <- mdy(tsRetail$date)
tsRetail2 <- tsRetail
tsRetail$date <- yearmonth(tsRetail$date)
#tsRetail
```

Question 1) (b)

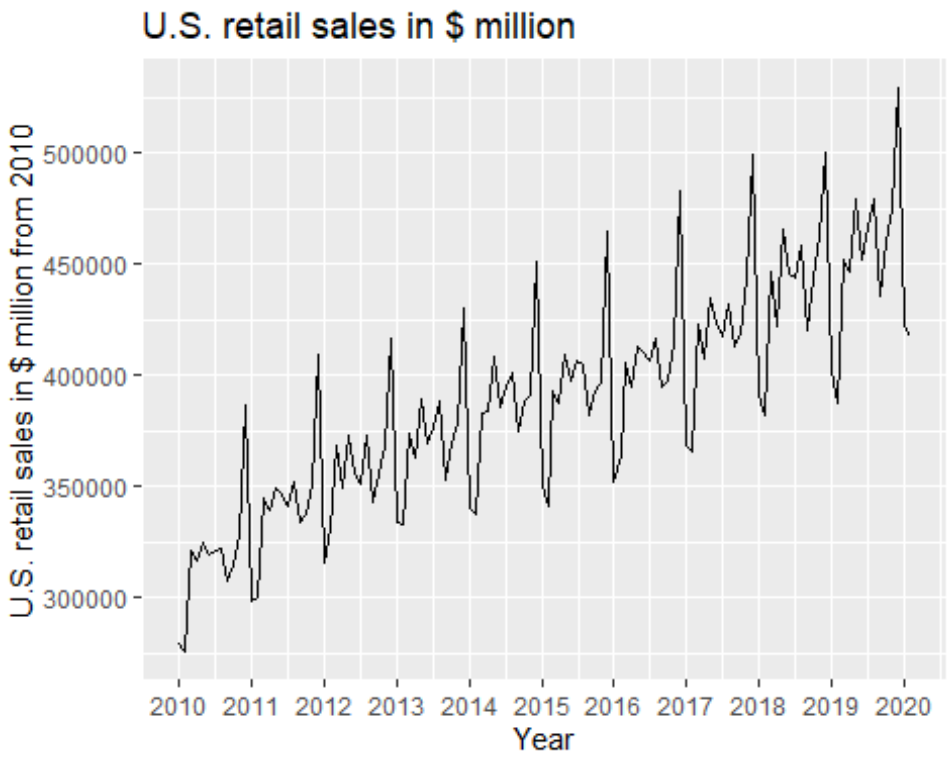
```
tsRetail <- tsRetail %>%
  as_tsibble(index = date)
```

Question 1) (c)

```
plotRetail <-

  tsRetail %>%
  autoplot()+
  xlab("Year") + ylab("U.S. retail sales in $ million over years") +
  ggtitle("U.S. retail sales in $ million") +
  scale_x_date(date_breaks = "years" , date_labels = "%y")

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

plotRetail
```

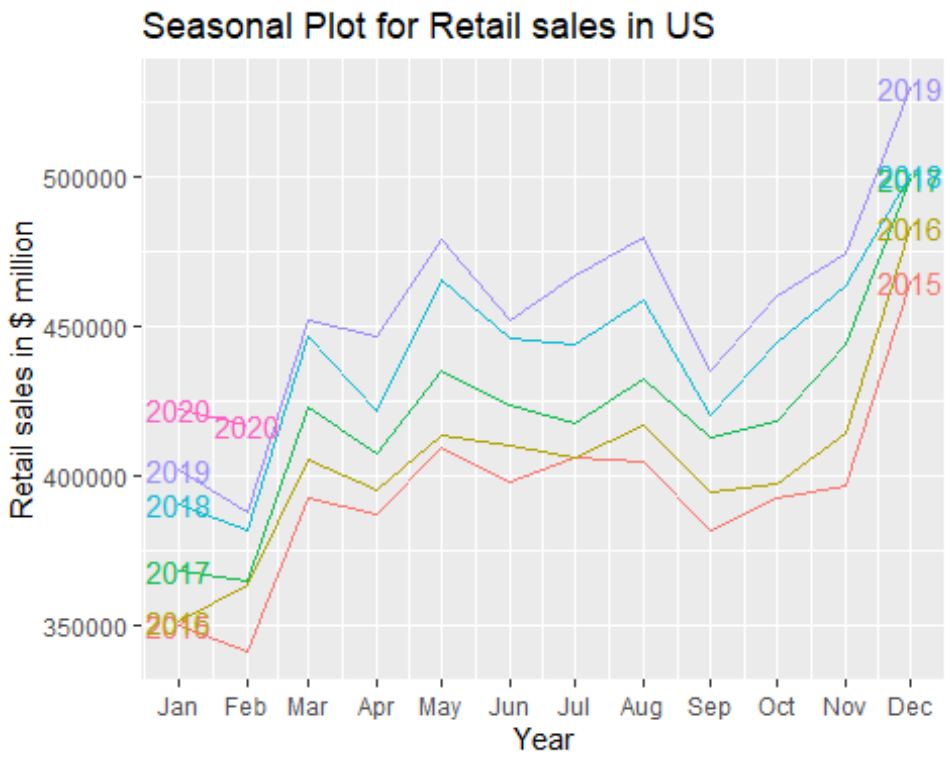



```
#Subsetting

tsRetailSubset <-  tsRetail %>%
  filter(year(date) >= 2010)

#tsRetailSubset


plotRetailSubset <-

  tsRetailSubset %>%
  autoplot()+
  xlab("Year") + ylab("U.S. retail sales in $ million from 2010") +
  ggtitle("U.S. retail sales in $ million") +
  scale_x_date(date_breaks = "years" , date_labels = "%Y")

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

plotRetailSubset
```

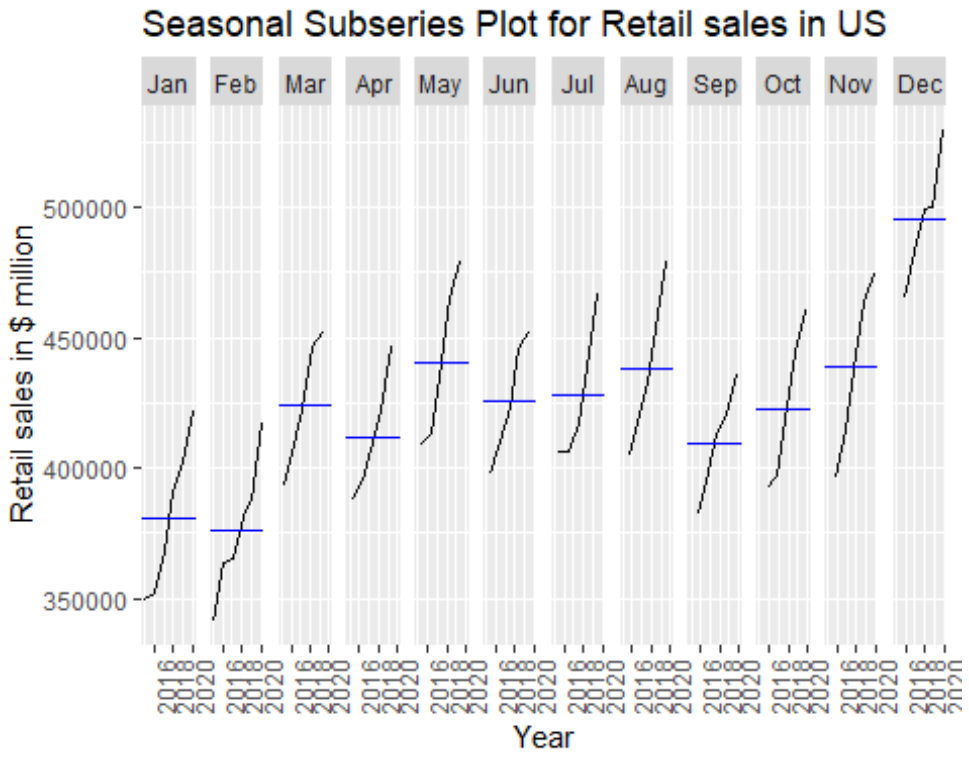


Question 2) (a)

```
#Seasonal plot

plotNySeason2015 <-tsRetail %>%
  filter(year(date) >= 2015) %>%
  gg_season(sales, labels = "both") +
  xlab("Year") + ylab("Retail sales in $ million") +
  ggtitle("Seasonal Plot for Retail sales in US")

plotNySeason2015
```

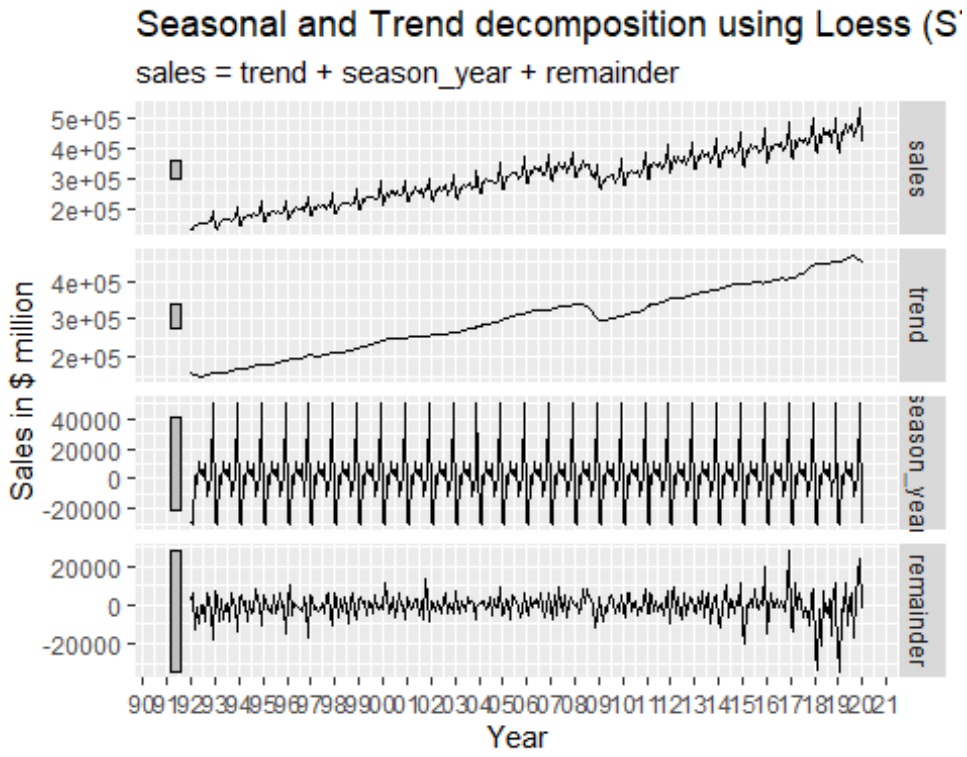



```r
#Seasonal subseries

plotNySeasonSub2015 <-
  #tsRetailSubset2015 %>%
  tsRetail %>%
  filter(year(date) >= 2015) %>%
  gg_subseries(sales) +
  xlab("Year") + ylab("Retail sales in $ million") +
  ggtitle("Seasonal Subseries Plot for Retail sales in US")

plotNySeasonSub2015
```

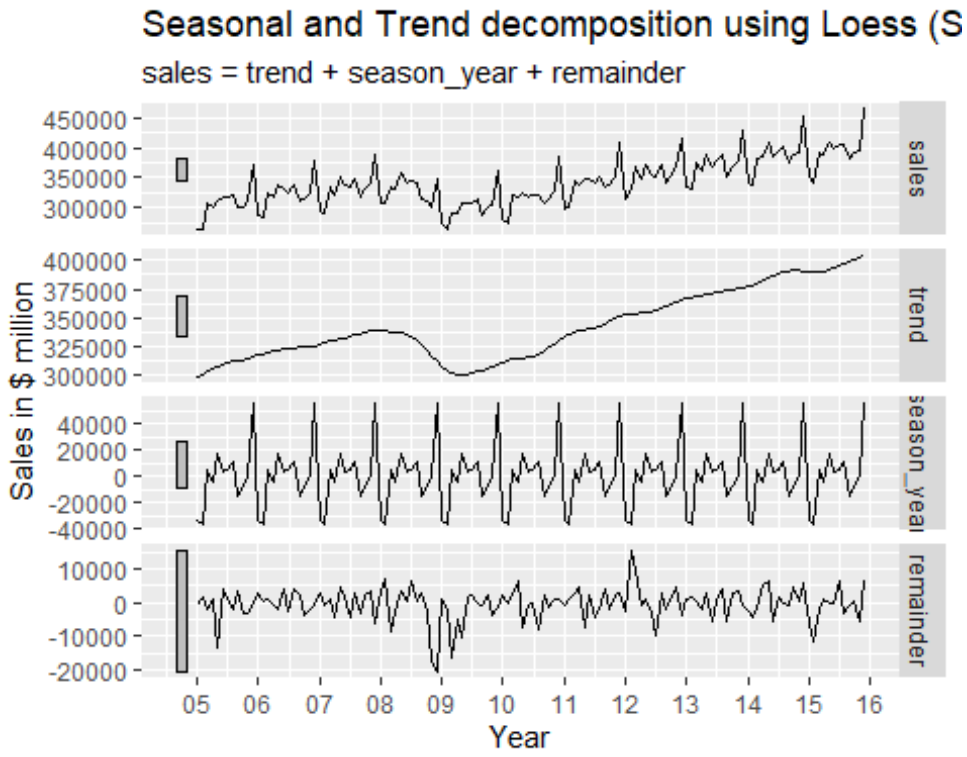


Question 2) (b)

```
#Full data STL decomposition

plotStlRetail <-

  tsRetail %>%
  model(STL(sales ~ trend(window=10) + season(window='periodic'), robust = TR
UE)) %>%
  components() %>%
  autoplot() +
  xlab("Year") + ylab("Sales in $ million") +
  ggtitle("Seasonal and Trend decomposition using Loess (STL decomposition)")
+
  scale_x_date(date_breaks = "years" , date_labels = "%y")

plotStlRetail
```

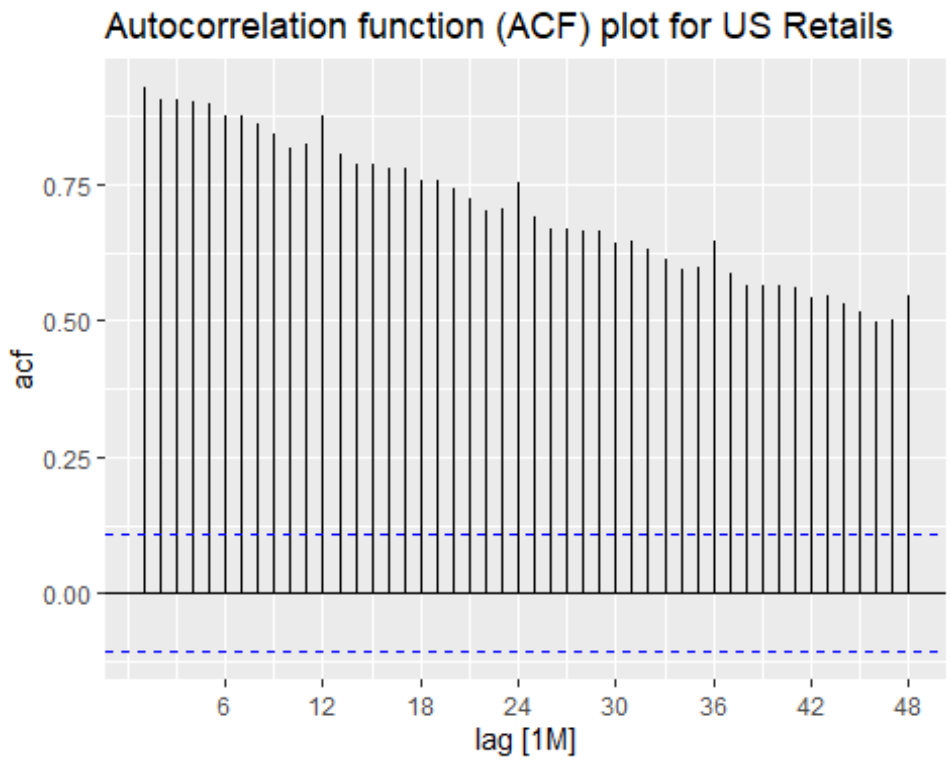



```
#Subset from 2005 t0 2015 STL decomposition

plotStlRetailSub <-

  tsRetail %>%
  filter(year(date) >=2005 & year(date) <= 2015) %>%
  model(STL(sales ~ trend(window=10) + season(window='periodic'), robust = TR
UE)) %>%
  components() %>%
  autoplot() +
  xlab("Year") + ylab("Sales in $ million") +
  ggtitle("Seasonal and Trend decomposition using Loess (STL decomposition) :
Subset data") +
  scale_x_date(date_breaks = "years" , date_labels = "%y")

plotStlRetailSub
```

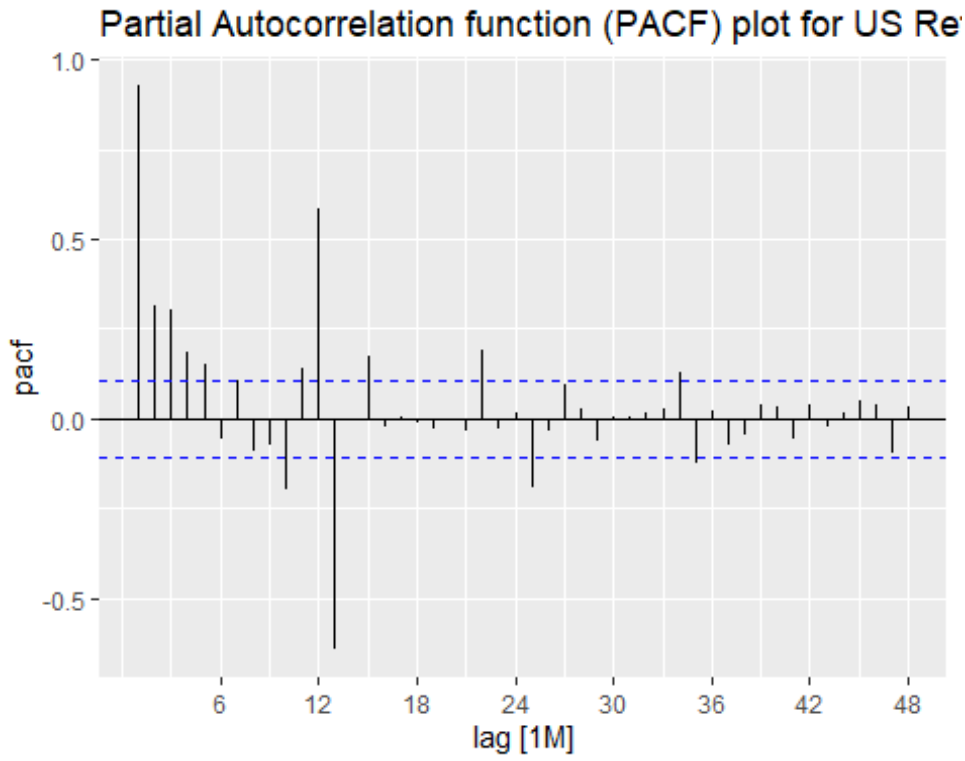


Question 2) (c)

```
#Autocorrelation Function

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

plotRetailACF
```

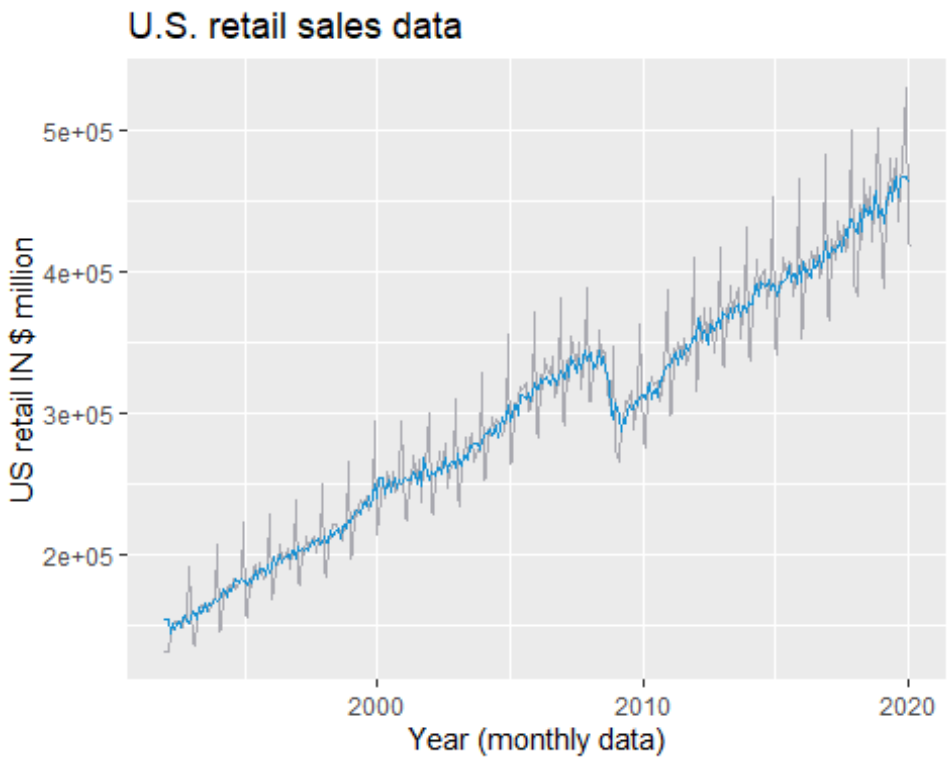



```
#Partial Autocorrelation Function

plotRetailPACF <-
  tsRetail %>%
  PACF(sales, lag_max = 48) %>%
  autoplot() +
  ggtitle("Partial Autocorrelation function (PACF) plot for US Retails")

plotRetailPACF
```



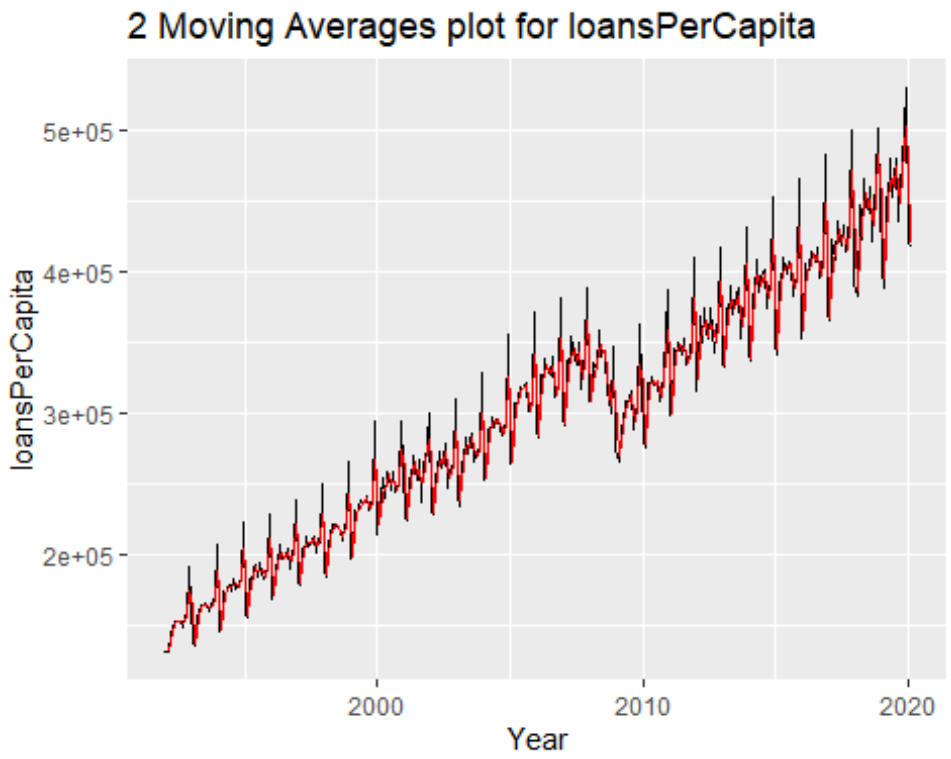
Question 2) (d)

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

plotRetailSeasonallyAdjusted
```

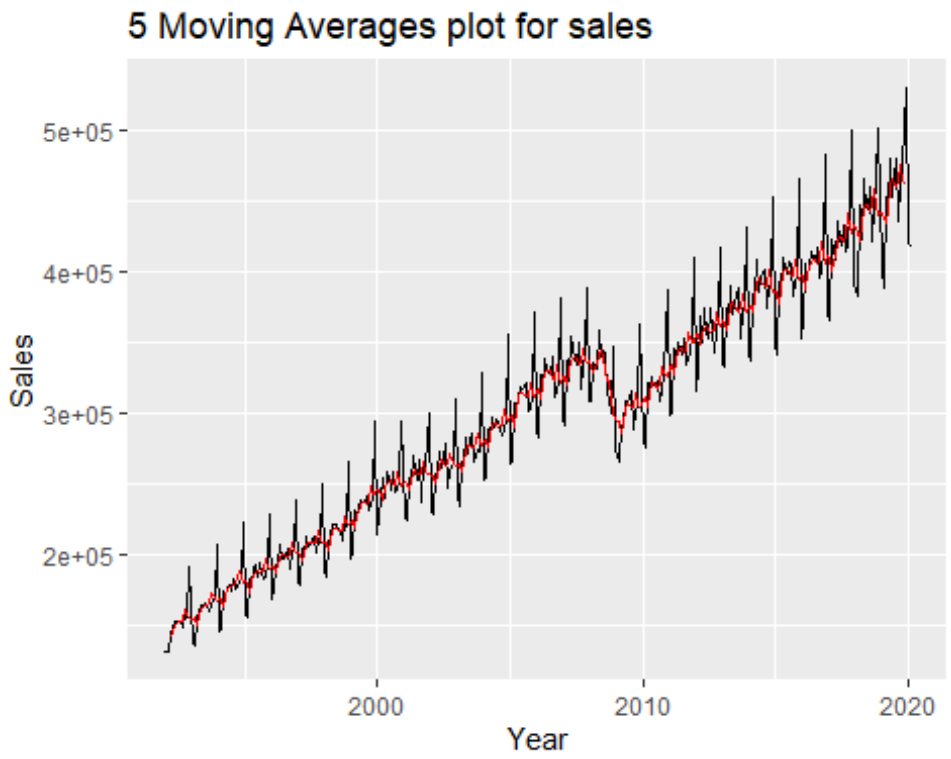


Question 2) (e)

```
#2nd order moving average

plot2MAvg <- tsRetail %>%
 mutate(`2-MA` = slide_dbl(sales, mean, .size = 2, .align = "center-right"))

plot2MAvg <-  tsRetail %>%
  mutate(`2-MA` = slide_dbl(sales, mean, .size = 2, .align = "center-right"))
%>%
  autoplot(sales) +
  autolayer(plot2MAvg, `2-MA`, color='red') +
  xlab("Year") + ylab("loansPerCapita") +
  ggtitle("2 Moving Averages plot for loansPerCapita") +
  guides(colour=guide_legend(title="series"))

plot2MAvg

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




```
plot5MAvg <- tsRetail %>%
  mutate(`5-MA` = slide_dbl(sales, mean, .size = 5, .align = "center"))

plot5MAvg %>%
  autoplot(sales) +
  autolayer(plot5MAvg, `5-MA`, color='red') +
  xlab("Year") + ylab("Sales") +
  ggtitle("5 Moving Averages plot for sales") +
  guides(colour=guide_legend(title="series"))

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



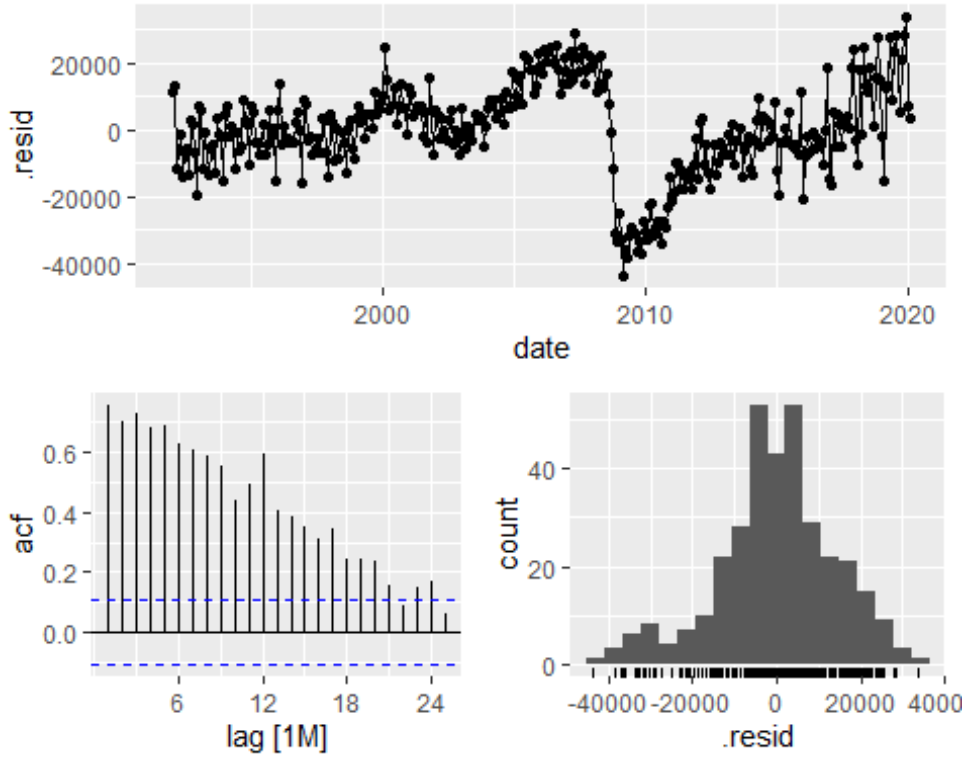
Question 3) (a)

```
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
```

```
#Residual diagnostics
fitRetail %>% gg_tsresiduals()
```



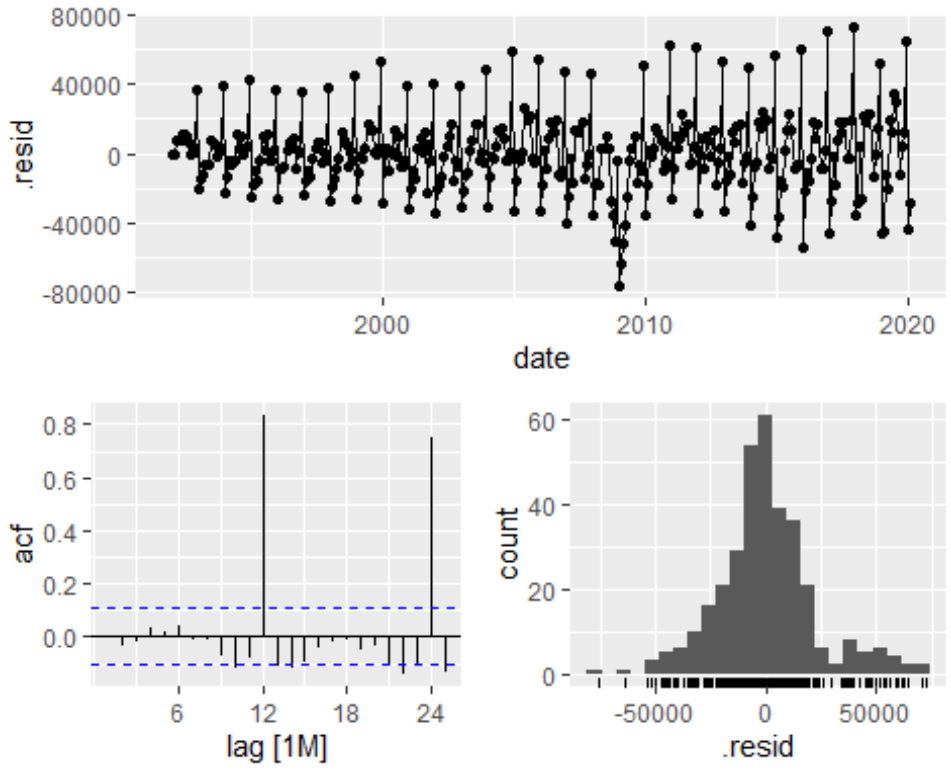
Question 3) (b)

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

report(fitRetailARIMA)
```

```
## 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
```

```
#Result diagnostics
fitRetailARIMA %>% gg_tsresiduals()
```



Question 3) (c)

```
#unit root test
tsRetail %>%
  features(sales, unitroot_ndiffs)
```

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

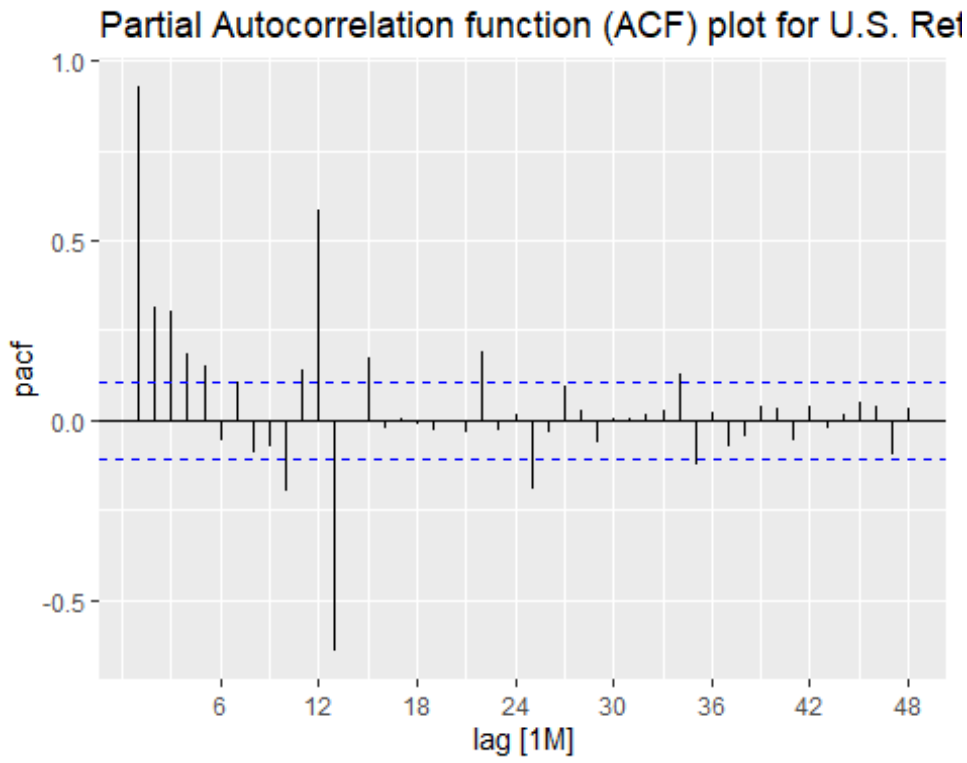
```
tsRetail %>%
  features(sales, unitroot_ndiffs)  #should this be difference(sales)?
```

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

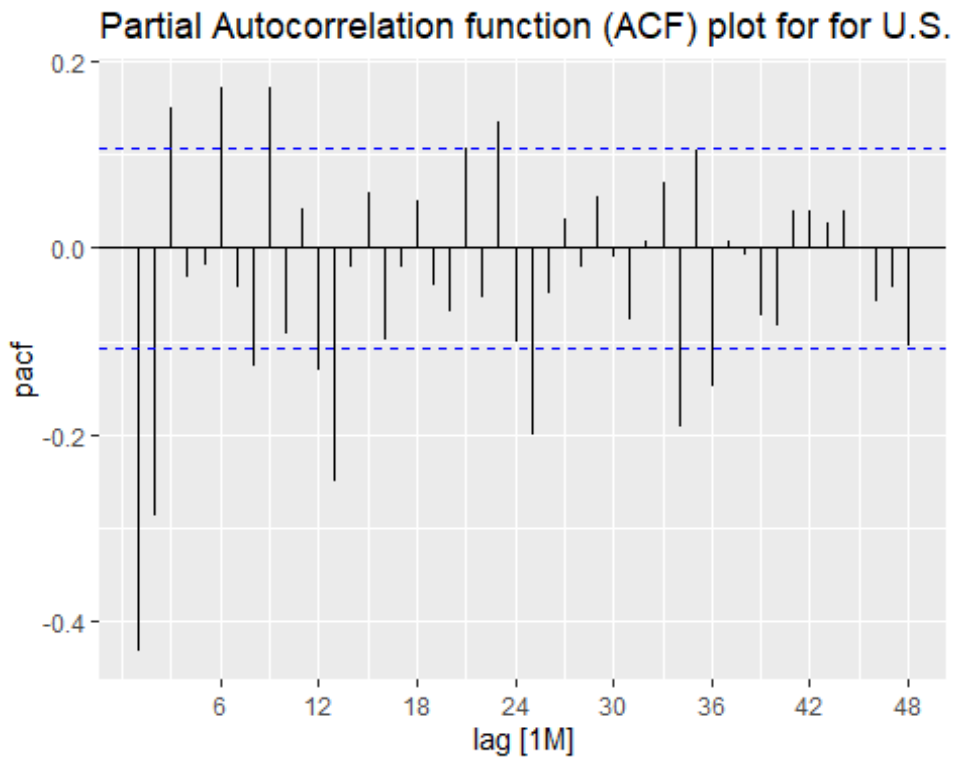
```r
#Suggested differencing and seasonal differencing with KPSS test

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

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

```r
plotRetailPACF <-
  tsRetail %>%
  PACF(sales, lag_max = 48) %>%
  autoplot() +  ggtitle("Partial Autocorrelation function (ACF) plot for U.S.
Retail Sales")

plotRetailPACF
```

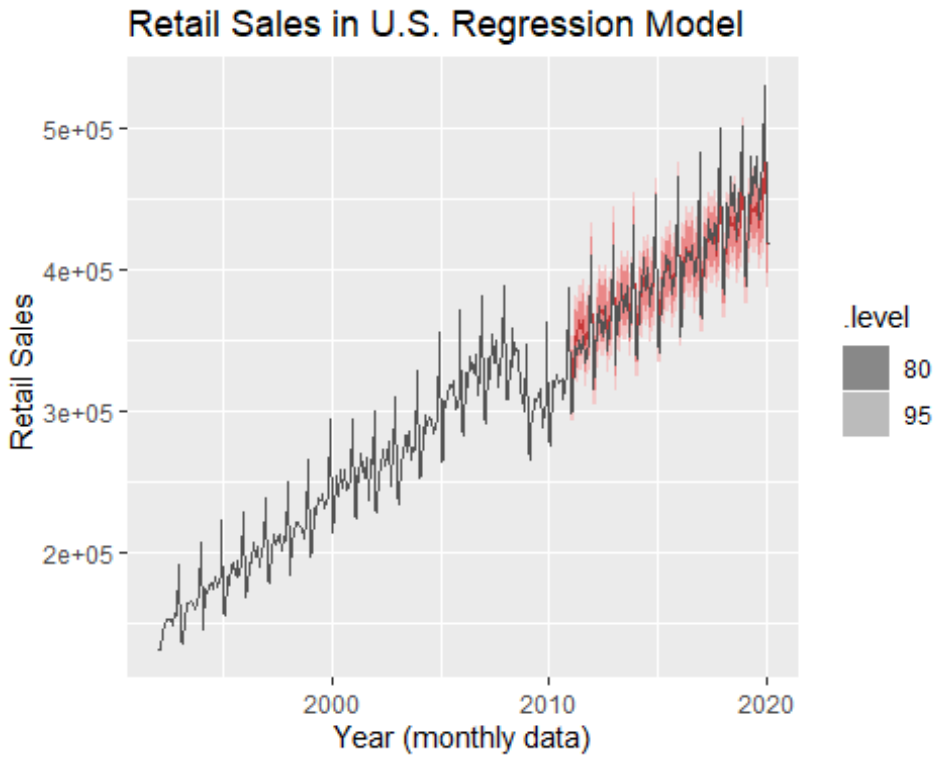



```r
plotRetailSeasonalDiffPACF <-
  tsRetail %>%
  mutate(diffSalesthenDiffSeason = difference(difference(sales), 12)) %>%
  PACF(diffSalesthenDiffSeason, lag_max = 48) %>%
  autoplot() +  ggtitle("Partial Autocorrelation function (ACF) plot for for
U.S. Retail Sales: Differencing")

plotRetailSeasonalDiffPACF
```

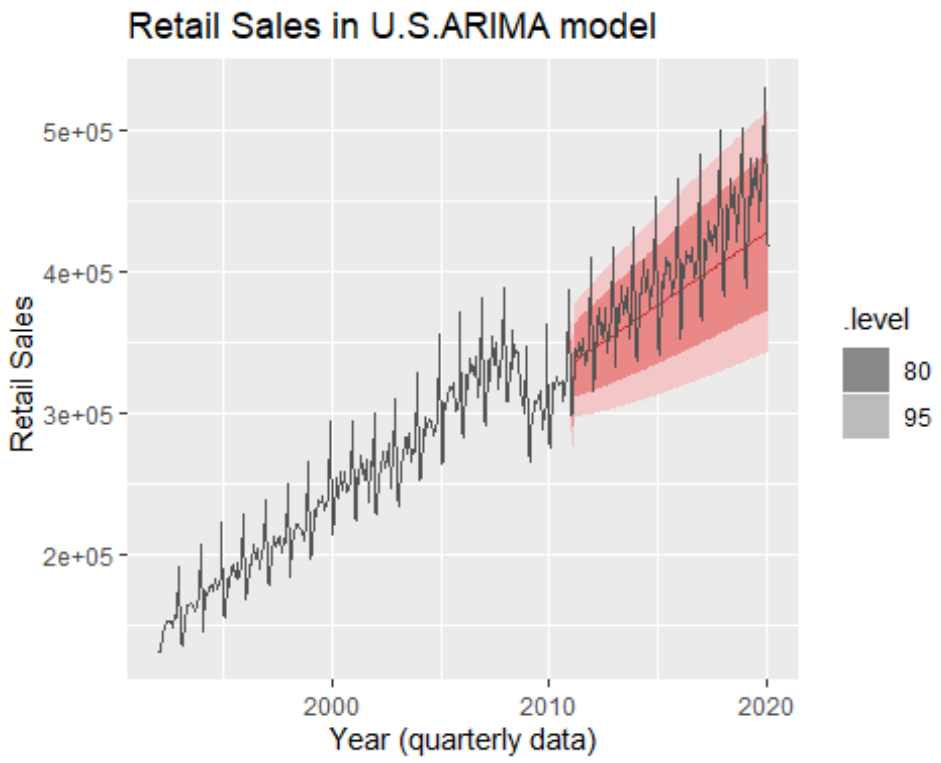


Question 3) (d)

```
set.seed(333)

tsRetailTrain <- tsRetail %>% filter(year(date) < 2011)
tsRetailTest <- tsRetail %>% filter(year(date) >= 2011)

#ten-year forecasting performance of a time series regression with trend and
season

plotRetailPredicted <-
  tsRetailTrain %>%
  model(TSLM(sales ~ trend() + season())) %>%
  forecast(new_data = tsRetailTest) %>%
  autoplot(tsRetail, colour = "#960A0A") +
  geom_line(colour = '#535353') +
  xlab("Year (monthly data)") + ylab("Retail Sales ") +
  ggtitle("Retail Sales in U.S. Regression Model")

plotRetailPredicted
```




```
#ARIMA Model

plotRetailPredictedArima <-
  tsRetailTrain %>%
  model(ARIMA(sales ~ PDQ(0,0,0), stepwise = FALSE, approximation = FALSE)) %
>%
  forecast(new_data = tsRetailTest) %>%
  autoplot(tsRetail, colour = "#960A0A") + #level = NULL,
  geom_line(colour = '#535353') +
  xlab("Year (quarterly data)") + ylab("Retail Sales ") +
  ggtitle("Retail Sales in U.S.ARIMA model")

plotRetailPredictedArima
```



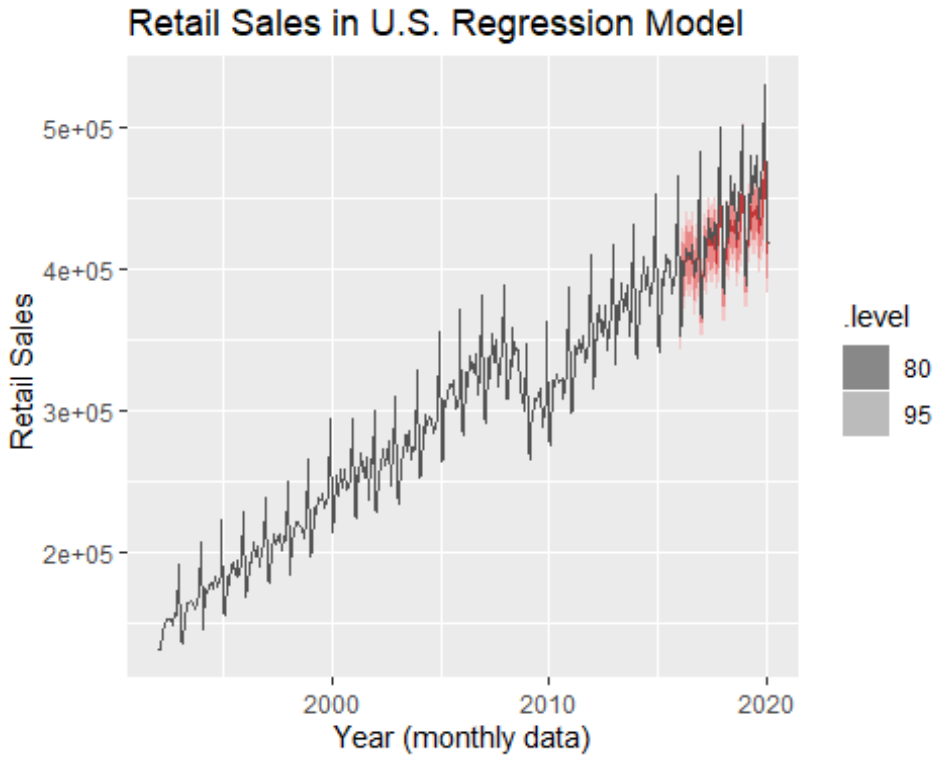

```
tsRetailFitAll2011 <-  tsRetailTrain %>%
  model(
    model1 = TSLM(sales ~ trend() + season()),
    model2 = ARIMA(sales ~ PDQ(0,0,0), stepwise = FALSE, approximation = FALS
E)
  )

tsRetailPredictAll2011 <- tsRetailFitAll2011 %>%
  forecast(new_data= tsRetailTest)


accuracy(tsRetailPredictAll2011, 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  16511. 32984. 25504.  3.55   6.13   NaN 0
.0438
```

Question 3) (e)

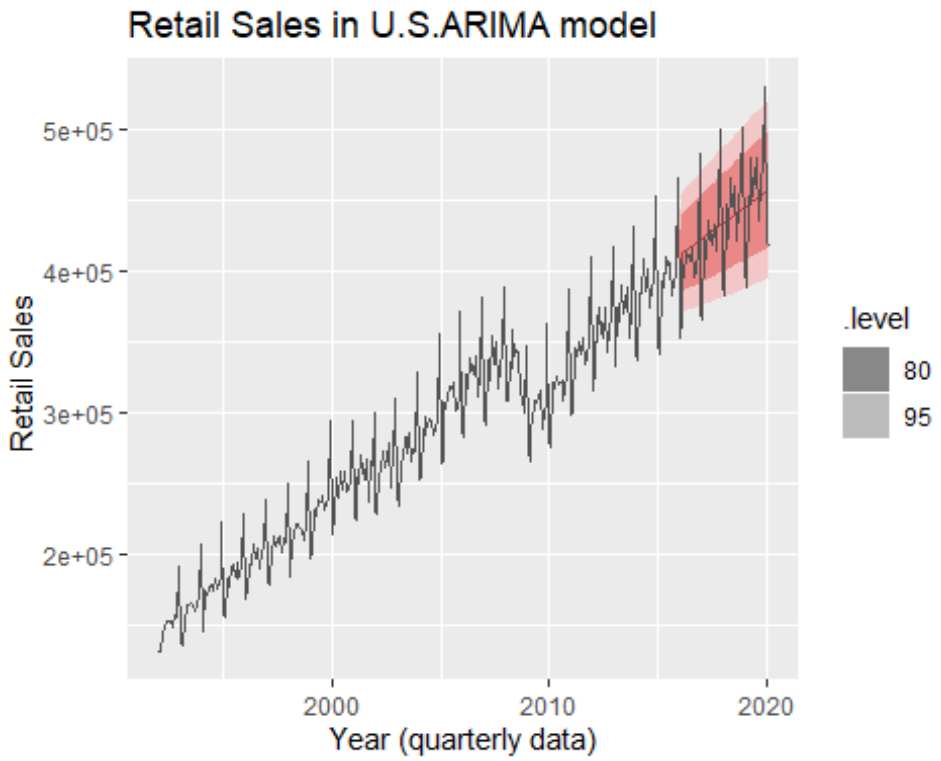
```
set.seed(333)
```

```
tsRetailTrain2 <- tsRetail %>% filter(year(date) < 2016)
tsRetailTest2 <- tsRetail %>% filter(year(date) >= 2016)

#ten-year forecasting performance of a time series regression with trend and
season

plotRetailPredicted2 <-
  tsRetailTrain2 %>%
  model(TSLM(sales ~ trend() + season())) %>%
  forecast(new_data = tsRetailTest2) %>%
  autoplot(tsRetail, colour = "#960A0A") +
  geom_line(colour = '#535353') +
  xlab("Year (monthly data)") + ylab("Retail Sales ") +
  ggtitle("Retail Sales in U.S. Regression Model")

plotRetailPredicted2
```




```
#ARIMA Model

plotRetailPredictedArima2 <-
  tsRetailTrain2 %>%
  model(ARIMA(sales ~ PDQ(0,0,0), stepwise = FALSE, approximation = FALSE)) %
>%
  forecast(new_data = tsRetailTest2) %>%
  autoplot(tsRetail, colour = "#960A0A") + #level = NULL,
  geom_line(colour = '#535353') +
  xlab("Year (quarterly data)") + ylab("Retail Sales ") +
```

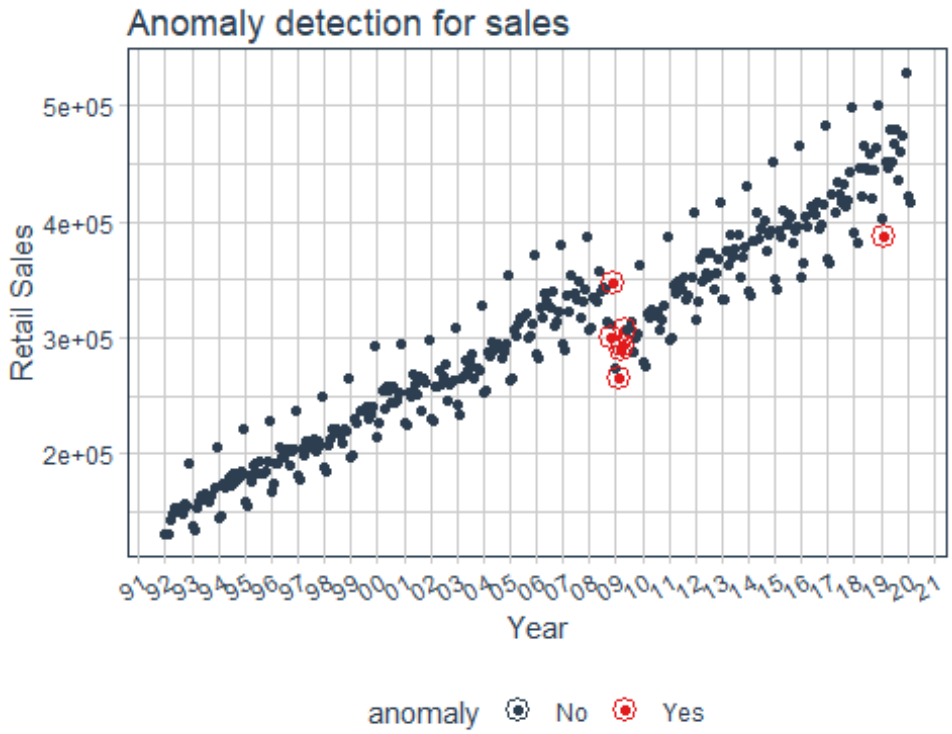
```
  ggtitle("Retail Sales in U.S.ARIMA model")

plotRetailPredictedArima2
```




```
tsRetailFitAll2016 <-  tsRetailTrain2 %>%
  model(
    model1 = TSLM(sales ~ trend() + season()),
    model2 = ARIMA(sales ~ PDQ(0,0,0), stepwise = FALSE, approximation = FALS
E)
  )


tsRetailPredictAll2016 <- tsRetailFitAll2016 %>%
  forecast(new_data= tsRetailTest2)

accuracy(tsRetailPredictAll2016, 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  -3232. 30570. 23039. -1.32  5.41   NaN 0
.0386
```

Question 4) (a)

```
anomalyRetail <-
  tsRetail2 %>% as_tbl_time(index = date) %>%
  time_decompose(sales, method = "stl" ,frequency = "auto", trend = "auto") %
>%
  anomalize(remainder, method = "gesd") %>%
  plot_anomalies() +
  labs(title = "Anomaly detection for sales") +
  xlab("Year") + ylab("Retail Sales") +
  scale_x_date(date_breaks = "years" , date_labels = "%y")

## frequency = 12 months

## trend = 60 months

anomalyRetail
```



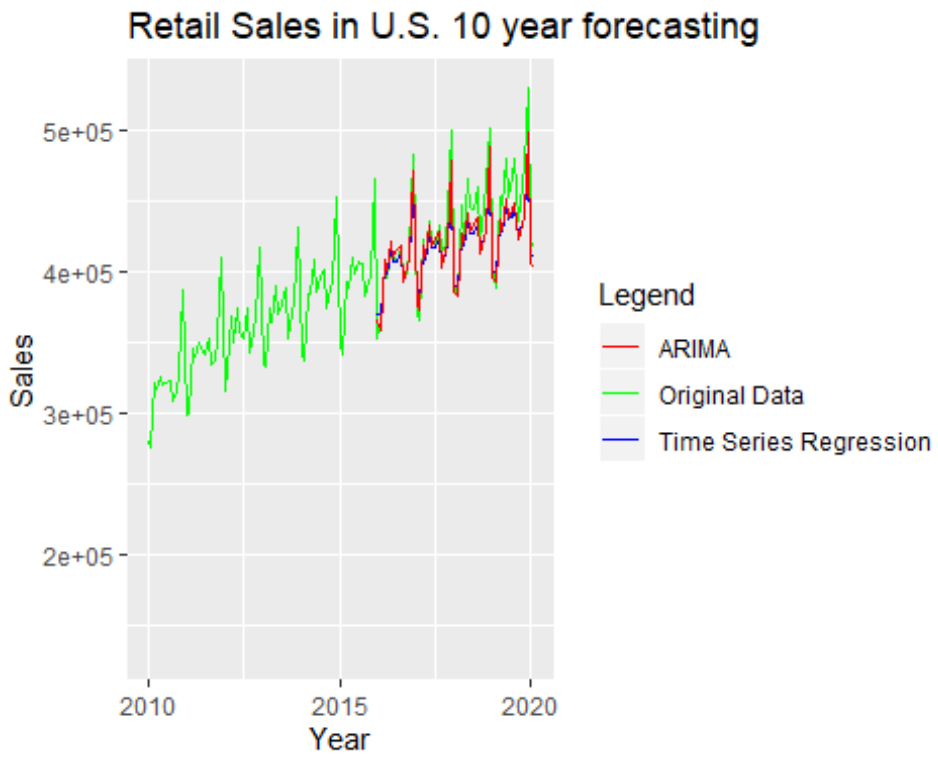

```
listAnomaly <- tsRetail2 %>%
  time_decompose(sales, frequency = "auto", trend = "auto") %>%
  anomalize(remainder,method='gesd') %>%
  time_recompose() %>%
  filter(anomaly == 'Yes')

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

## frequency = 12 months
```

```
## trend = 60 months

listAnomaly

## # A time tibble: 7 x 10
## # Index: date
##   date       observed  season  trend remainder remainder_l1 remainder_l2 a
nomaly
##   <date>        <dbl>   <dbl>  <dbl>     <dbl>        <dbl>        <dbl> <
chr>
## 1 2008-11-01   299238   2377. 3.26e5   -28812.      -26532.       26532. Y
es
## 2 2008-12-01   346513  50103. 3.25e5   -28972.      -26532.       26532. Y
es
## 3 2009-02-01   264465 -30260. 3.25e5   -30267.      -26532.       26532. Y
es
## 4 2009-03-01   290068   2577. 3.25e5   -37399.      -26532.       26532. Y
es
## 5 2009-04-01   292041  -3886. 3.25e5   -28862.      -26532.       26532. Y
es
## 6 2009-05-01   307481  12008. 3.25e5   -29214.      -26532.       26532. Y
es
## 7 2019-02-01   387672 -30260. 4.52e5   -34303.      -26532.       26532. Y
es
## # ... with 2 more variables: recomposed_l1 <dbl>, recomposed_l2 <dbl>
```

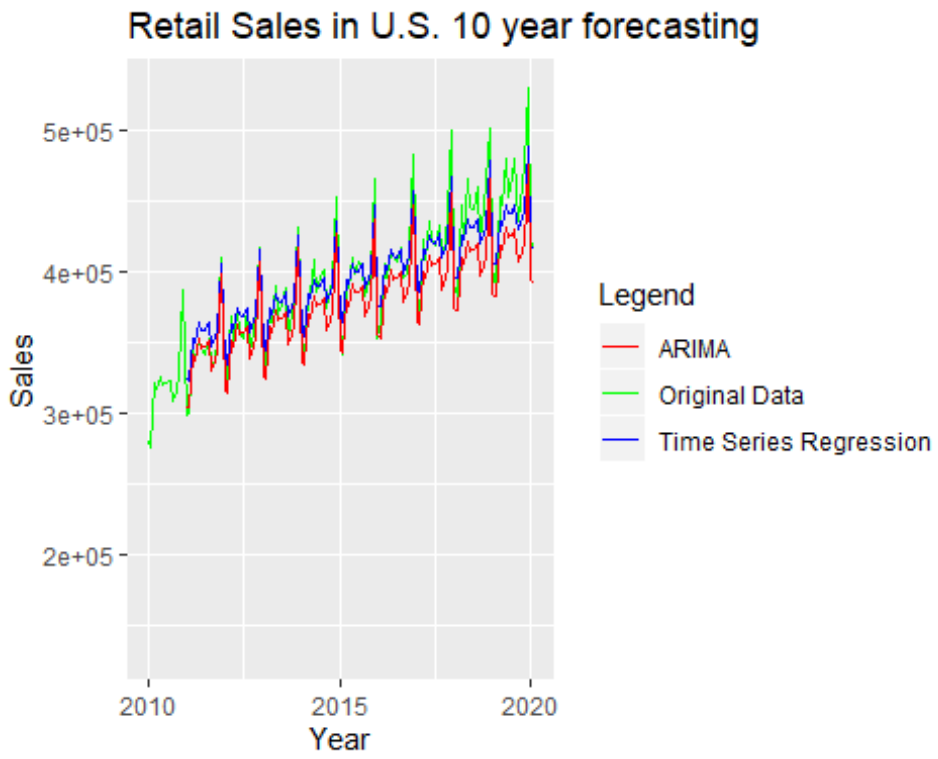
Question 4) (b)

```
tsRetail2016Fit <-
  tsRetailTrain2 %>%
  model(
    model1 = TSLM(sales ~ trend() + season())
  )

tsRetail2016PredictTSLM <-
  tsRetail2016Fit %>%
  forecast(new_data = tsRetailTest2)

tsRetail2016FitArima <-
  tsRetailTrain2 %>%
  model(
    model_new = ARIMA(sales )
  )

tsRetail2016PredictArima <-
  tsRetail2016FitArima %>%
  forecast(new_data = tsRetailTest2)

tsRetail2011Fit <-
  tsRetailTrain %>%
  model(
```

```
    model1 = TSLM(sales ~ trend() + season())
  )

tsRetail2011PredictTSLM <-
  tsRetail2011Fit %>%
  forecast(new_data = tsRetailTest)

tsRetail2011FitArima <-
  tsRetailTrain %>%
  model(
    model_new = ARIMA(sales)
  )

tsRetail2011PredictArima <-
  tsRetail2011FitArima %>%
  forecast(new_data = tsRetailTest)

#tsRetail

plot1 <-
  ggplot() +
  geom_line(data = tsRetail, aes(x = date, y = sales, color = "Original Data"
)) +
  geom_line(data=tsRetail2016PredictTSLM , aes(x = date, y = sales, color = "
Time Series Regression")) +
  geom_line(data= tsRetail2016PredictArima, aes(x = date, y = sales, color =
"ARIMA")) +
  xlim(c(as.Date('2010-01-01'),as.Date('2020-02-01')))+
  ggtitle("Retail Sales in U.S. 10 year forecasting") +
labs(x='Year',y='Sales', color="Legend")+
  scale_color_manual(values = c("red","green","blue"))

plot1

## Warning: Removed 216 rows containing missing values (geom_path).
```




```
plot2 <-
  ggplot() +
  geom_line(data = tsRetail, aes(x = date, y = sales, color = "Original Data"
)) +
  geom_line(data=tsRetail2011PredictTSLM , aes(x = date, y = sales, color = "
Time Series Regression")) +
  geom_line(data= tsRetail2011PredictArima, aes(x = date, y = sales, color =
"ARIMA")) +
  xlim(c(as.Date('2010-01-01'),as.Date('2020-02-01')))+
  ggtitle("Retail Sales in U.S. 10 year forecasting") +
labs(x='Year',y='Sales', color="Legend")+
  scale_color_manual(values = c("red","green","blue"))

plot2

## Warning: Removed 216 rows containing missing values (geom_path).
```


Retail Sales in U.S. 10 year forecasting
5e+05
4e+05
3e+05
2e+05
Sales
2010
2015
2020
Year
Legend
ARIMA
Original Data
Time Series Regression