STA4003_HW2

Yuzhou Peng

2023-10-19

```
library(fpp3)
```

```
## Warning: 程辑包'fpp3'是用R版本4.2.3 来建造的
```

```
## ── Attaching packages ─────────────────────────────────────────────────────────────────────────────── fpp3 0.
5 ──
```

```
## ✓ tibble      3.1.8     ✓ tsibble     1.1.3
## ✓ dplyr       1.1.0     ✓ tsibbledata 0.4.1
## ✓ tidyr       1.3.0     ✓ feasts      0.3.1
## ✓ lubridate   1.9.2     ✓ fable       0.3.3
## ✓ ggplot2     3.4.1     ✓ fabletools  0.3.3
```

```
## Warning: 程辑包'tsibble'是用R版本4.2.3 来建造的
```

```
## Warning: 程辑包'tsibbledata'是用R版本4.2.3 来建造的
```

```
## Warning: 程辑包'feasts'是用R版本4.2.3 来建造的
```

```
## Warning: 程辑包'fabletools'是用R版本4.2.3 来建造的
```

```
## Warning: 程辑包'fable'是用R版本4.2.3 来建造的
```

```
## ── Conflicts ─────────────────────────────────────────────────────────────────────────────────── fpp3_confl
icts ──
## ✖ lubridate::date()    masks base::date()
## ✖ dplyr::filter()      masks stats::filter()
## ✖ tsibble::intersect() masks base::intersect()
## ✖ tsibble::interval()  masks lubridate::interval()
## ✖ dplyr::lag()         masks stats::lag()
## ✖ tsibble::setdiff()   masks base::setdiff()
## ✖ tsibble::union()     masks base::union()
```

```
library(dplyr)
library(feasts)
library(fable)
library(broom)
```

```
## Warning: 程辑包'broom'是用R版本4.2.3 来建造的
```

4.6.1

Write a function to compute the mean and standard deviation of a time series, and apply it to the *PBS* data. Plot the series with the highest mean, and the series with the lowest standard deviation.

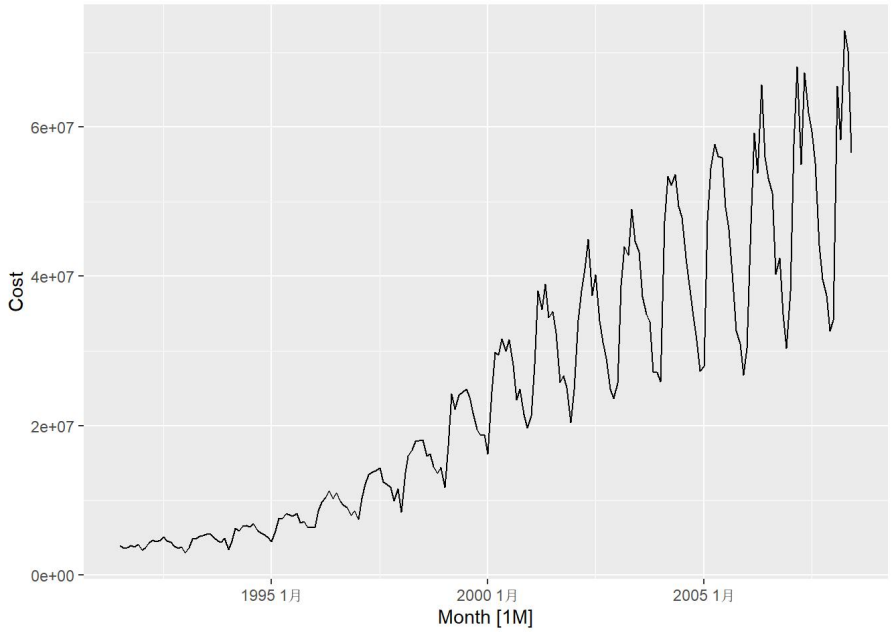
```
data("PBS")
mean_std <- function(x){
  c(
    Mean = mean(x, na.rm = T),
    Std = sd(x, na.rm = T)
  )
}

pbs_feature <- PBS %>%
  features(Cost, mean_std)

pbs_feature
```

```
## # A tibble: 336 × 6
##    Concession   Type         ATC1  ATC2        Mean       Std
##    <chr>        <chr>        <chr> <chr>      <dbl>     <dbl>
##  1 Concessional Co-payments A     A01       67673.    14763.
##  2 Concessional Co-payments A     A02    16455044.  7498596.
##  3 Concessional Co-payments A     A03      476221.   370696.
##  4 Concessional Co-payments A     A04      463392.   154020.
##  5 Concessional Co-payments A     A05      147604.    74190.
##  6 Concessional Co-payments A     A06      417889.   163040.
##  7 Concessional Co-payments A     A07      917795.   338081.
##  8 Concessional Co-payments A     A09      343881.    89815.
##  9 Concessional Co-payments A     A10     5680010.  3180294.
## 10 Concessional Co-payments A     A11      651863.   387439.
## # … with 326 more rows
```

```
pbs_feature %>%
  filter(Mean == max(Mean))
```

```
## # A tibble: 1 × 6
##   Concession   Type         ATC1  ATC2        Mean       Std
##   <chr>        <chr>        <chr> <chr>      <dbl>     <dbl>
## 1 Concessional Co-payments C     C10    24845501. 18384824.
```

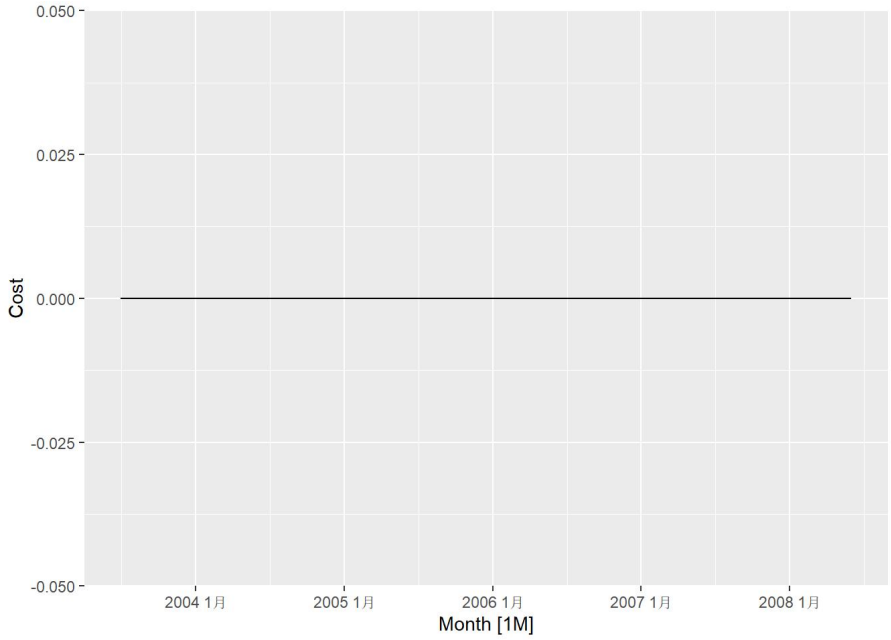
```
PBS %>%
  filter(Concession == "Concessional",
         Type == "Co-payments",
         ATC1 == "C",
         ATC2 == "C10") %>%
  autoplot(Cost)
```



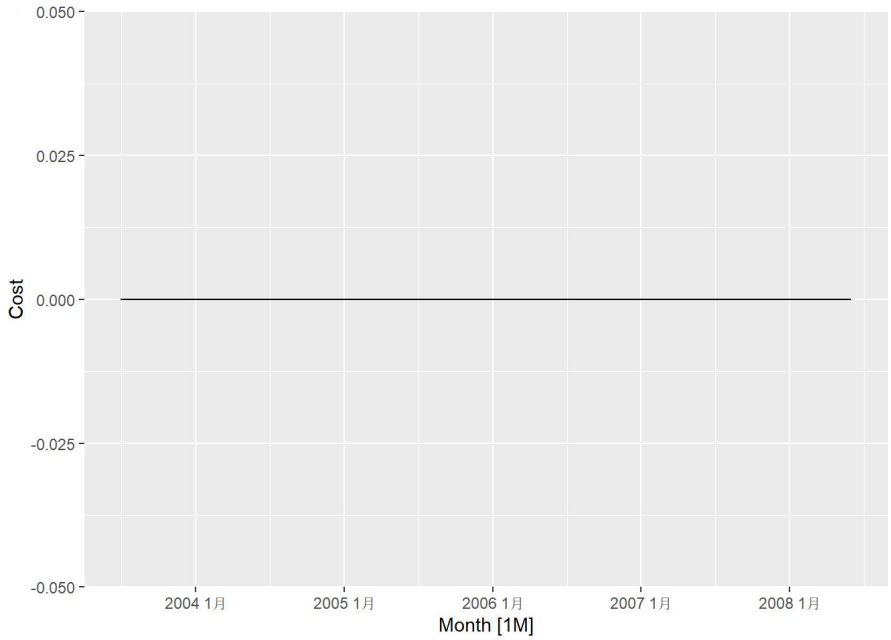

```
pbs_feature %>%
  filter(Std == min(Std))
```

```
## # A tibble: 2 × 6
##   Concession Type         ATC1  ATC2   Mean   Std
##   <chr>      <chr>        <chr> <chr> <dbl> <dbl>
## 1 General    Co-payments R     R         0     0
## 2 General    Co-payments S     S         0     0
```

```
PBS %>%
  filter(
    Concession == "General",
    Type == "Co-payments",
    ATC1 == "R",
    ATC2 == "R"
  ) %>%
  autoplot(Cost)
```




```
PBS %>%
  filter(
    Concession == "General",
    Type == "Co-payments",
    ATC1 == "S",
    ATC2 == "S"
  ) %>%
  autoplot(Cost)
```



4.6.3

Use a feature-based approach to look for outlying series in the PBS data. What is unusual about the series you identify as “outliers”.

```
PBS_feast <- PBS %>%
  features(Cost, feature_set(pkgs = "feasts"))
```

```
## Warning in optimise(lambda_coef_var, c(lower, upper), x = x, .period =
## max(.period, : NA/Inf被换成最大的正值

## Warning in optimise(lambda_coef_var, c(lower, upper), x = x, .period =
## max(.period, : NA/Inf被换成最大的正值

## Warning in optimise(lambda_coef_var, c(lower, upper), x = x, .period =
## max(.period, : NA/Inf被换成最大的正值

## Warning in optimise(lambda_coef_var, c(lower, upper), x = x, .period =
## max(.period, : NA/Inf被换成最大的正值
```

```
## Error in ar.burg.default(x, aic = aic, order.max = order.max, na.action = na.action,  :
##   zero-variance series
## Error in ar.burg.default(x, aic = aic, order.max = order.max, na.action = na.action,  :
##   zero-variance series
```

```
## Warning: `n_flat_spots()` was deprecated in feasts 0.1.5.
## ℹ Please use `longest_flat_spot()` instead.
## ℹ The deprecated feature was likely used in the fabletools package.
##   Please report the issue at <  ]8;;https://github.com/tidyverts/fabletools/issues  https://github.com/tidyverts/fabletools/issues  ]8;;  >.
```

```
## New names:
## • `` -> `...26`
```

```
## Warning: 336 errors (1 unique) encountered for feature 6
## [336] The `urca` package must be installed to use this functionality. It can be installed with install.packages("urca")
```

```
## Warning: 336 errors (1 unique) encountered for feature 7
## [336] The `urca` package must be installed to use this functionality. It can be installed with install.packages("urca")
```

```
## Warning: 336 errors (1 unique) encountered for feature 8
## [336] The `urca` package must be installed to use this functionality. It can be installed with install.packages("urca")
```

```
## Warning: 336 errors (1 unique) encountered for feature 12
## [336] loadNamespace()里算'slider'时.onLoad失败了，详细内容：
##   调用: fun(libname, pkgname)
##   错误: 程序包'obj_is_vector'不提供'vctrs'这样的函数
```

```
## Warning: 336 errors (1 unique) encountered for feature 13
## [336] loadNamespace()里算'slider'时.onLoad失败了，详细内容：
##   调用: fun(libname, pkgname)
##   错误: 程序包'obj_is_vector'不提供'vctrs'这样的函数
```

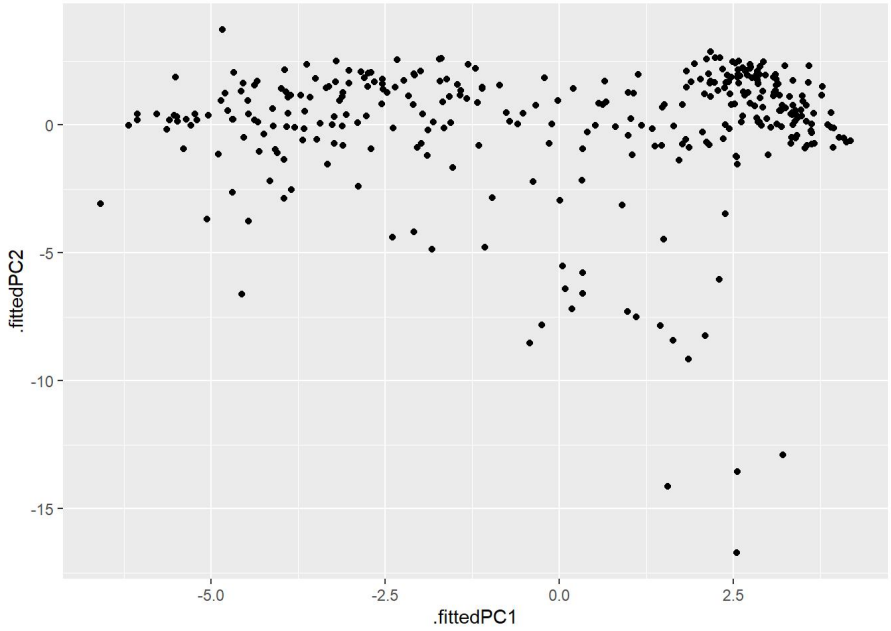
```
## Warning: 336 errors (1 unique) encountered for feature 14
## [336] loadNamespace()里算'slider'时.onLoad失败了，详细内容：
##   调用: fun(libname, pkgname)
##   错误: 程序包'obj_is_vector'不提供'vctrs'这样的函数
```

```
## Warning: 336 errors (1 unique) encountered for feature 15
## [336] loadNamespace()里算'slider'时.onLoad失败了，详细内容：
##   调用: fun(libname, pkgname)
##   错误: 程序包'obj_is_vector'不提供'vctrs'这样的函数
```

```
## Warning: 336 errors (1 unique) encountered for feature 16
## [336] loadNamespace()里算'slider'时.onLoad失败了，详细内容：
##   调用: fun(libname, pkgname)
##   错误: 程序包'obj_is_vector'不提供'vctrs'这样的函数
```

```
## Warning: 336 errors (1 unique) encountered for feature 20
## [336] The `fracdiff` package must be installed to use this functionality. It can be installed with install.packages("fracdiff")
```

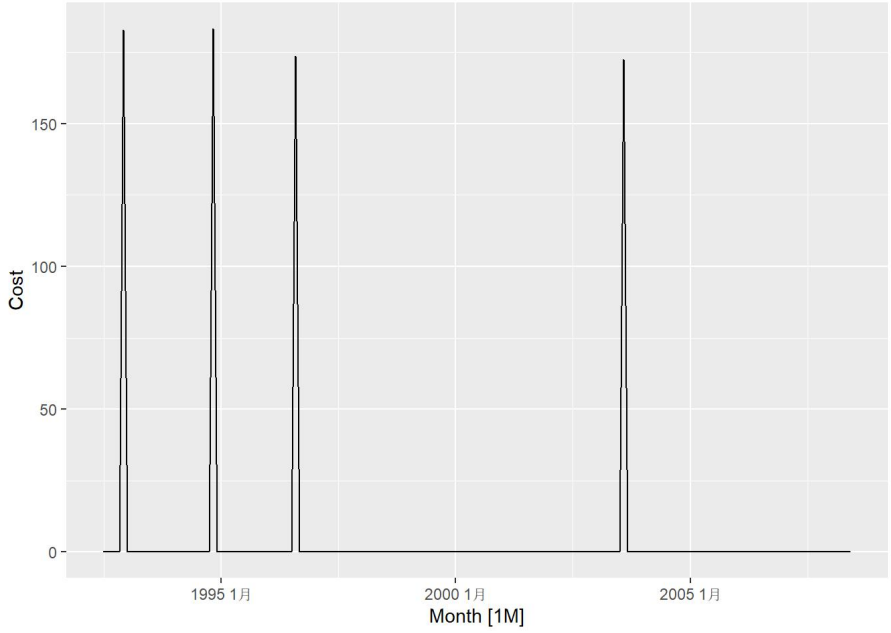
```
PBS_feast <- PBS_feast[,-30]

PBS_feast <- PBS_feast %>%
  na.omit()

PBS_pc <- PBS_feast %>%
  select(-Concession, -Type, -ATC1, -ATC2) %>%
  prcomp(scale = T) %>%
  augment(PBS_feast)

p1 <- PBS_pc %>%
   unite("serie", Concession:ATC2, sep = "-", remove = FALSE) %>%
  ggplot(aes(x = .fittedPC1, y = .fittedPC2, label = serie)) +
  geom_point()

p1
```



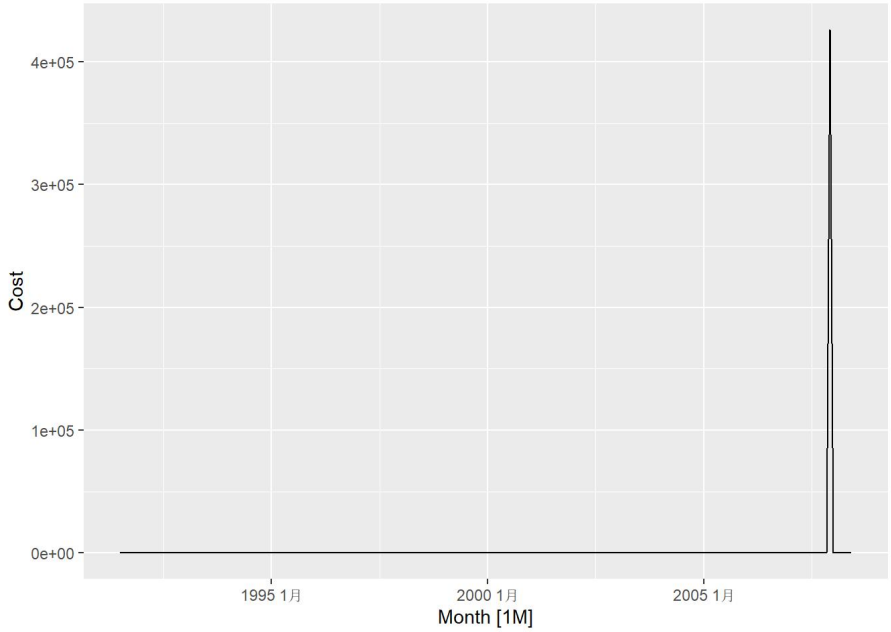

```
outliers <- PBS_pc %>%
  filter(.fittedPC2 < -10)
outliers %>%
  select(ATC1, ATC2, Type, Concession)
```

```
## # A tibble: 4 × 4
##   ATC1  ATC2  Type        Concession
##   <chr> <chr> <chr>       <chr>
## 1 J     J06   Safety net  Concessional
## 2 C     C05   Co-payments General
## 3 S     S02   Co-payments General
## 4 S     S03   Co-payments General
```

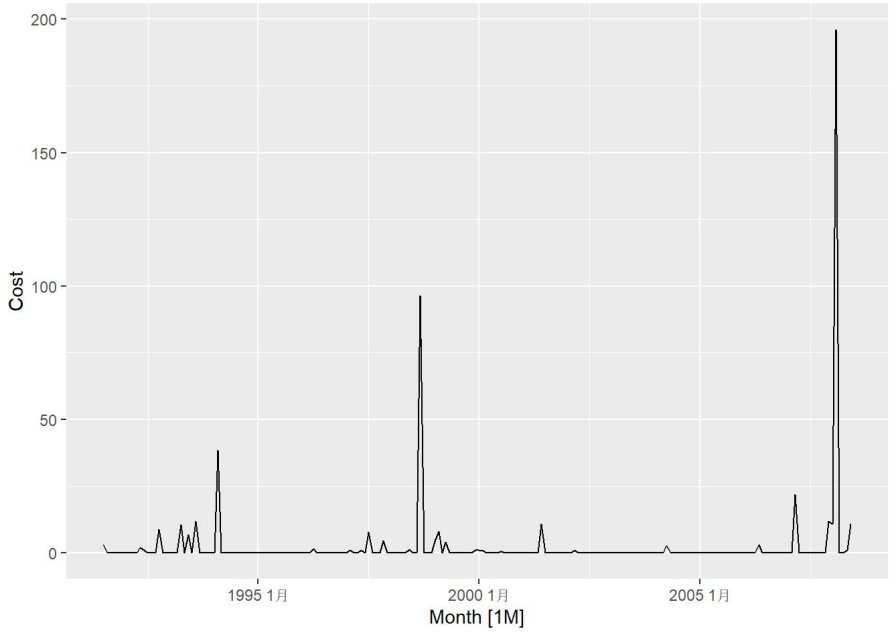
```
PBS %>%
  filter(
    Concession == "Concessional",
    Type == "Safety net",
    ATC1 == "J",
    ATC2 == "J06"
  ) %>%
  autoplot(Cost)
```



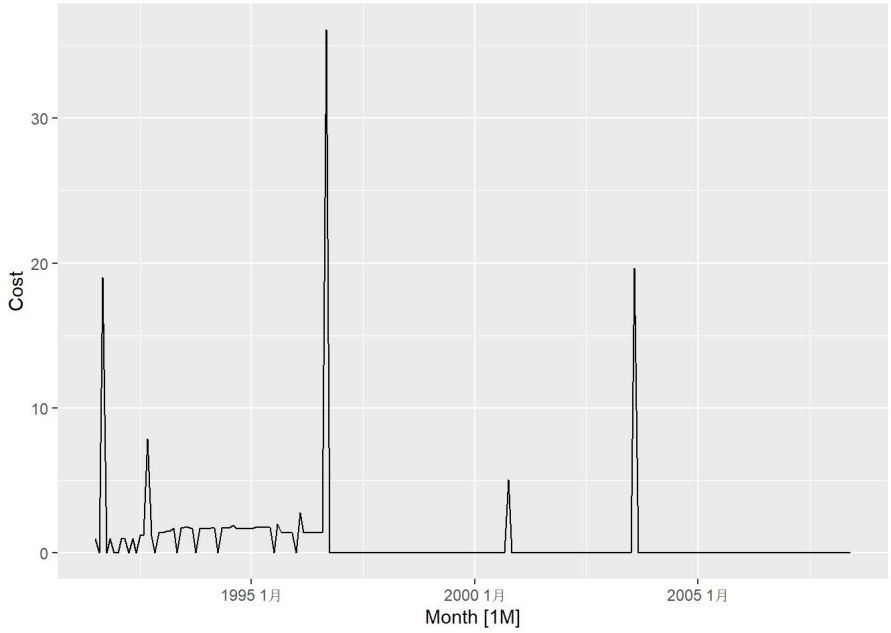
```
PBS %>%
  filter(
    Concession == "General",
    Type == "Co-payments",
    ATC1 == "C",
    ATC2 == "C05"
  ) %>%
  autoplot(Cost)
```




```
PBS %>%
  filter(
    Concession == "General",
    Type == "Co-payments",
    ATC1 == "S",
    ATC2 == "S02"
  ) %>%
  autoplot(Cost)
```




```
PBS %>%
  filter(
    Concession == "General",
    Type == "Co-payments",
    ATC1 == "S",
    ATC2 == "S03"
  ) %>%
  autoplot(Cost)
```



The recorded times payment is extremely low

5.11.6

a. False. Forcasting by bootstrap method generally does not have normally distributed residuals
b. False. It is possible that a model with small residuals is a over-fitting model.
c. False. Measuring accuracy by MAPE has bas performance when the series have a number of meaningful zero values
d. False. More complicated model does not suggest more accuracy. Reasons for bad performance could be wrong selction of variable or assu
e. False. If the test set has only a single observation, or a very small number of observations, then test result is not reliable.

5.11.9

a. Create a training set for household wealth (*hh_budget*) by withholding the last four years as a test set.

```
data("hh_budget")
hh_training <- hh_budget %>%
  filter(Year <= 2012)

hh_testing <- hh_budget %>%
  filter(Year > 2012)
```

b. Fit all the appropriate benchmark methods to the training set and forecast the periods covered by the test set.

```
hh_fit <- hh_training %>%
  model(
    mean = MEAN(Wealth),
    naive = NAIVE(Wealth),
    drift = NAIVE(Wealth ~ drift())
  )


hh_fc <- hh_fit %>%
  forecast(h = 4)
```

c. Compute the accuracy of your forecasts. Which method does best?

```
accuracy(hh_fc, hh_testing)
```

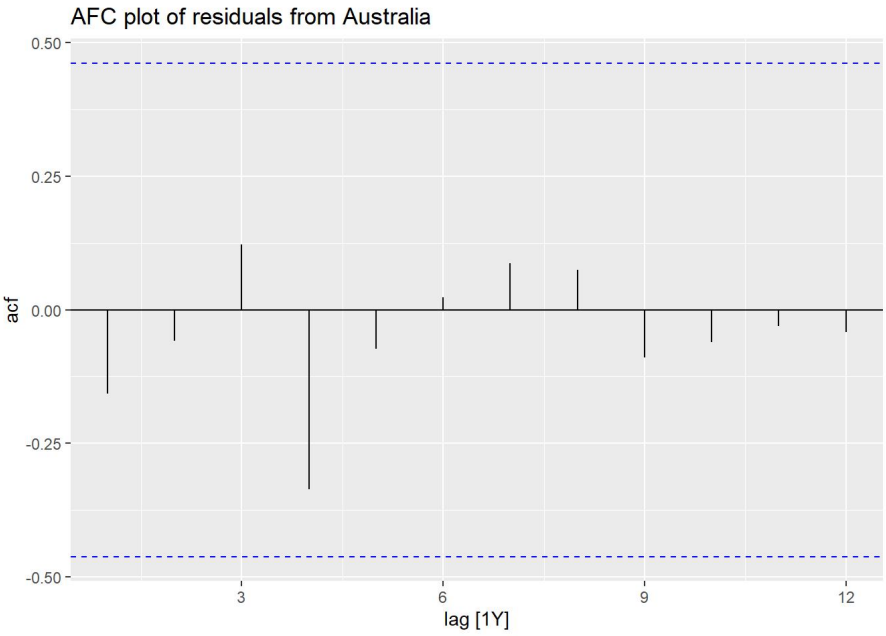
```
## # A tibble: 12 × 11
##    .model Country   .type    ME  RMSE   MAE   MPE  MAPE  MASE RMSSE    ACF1
##    <chr>  <chr>     <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>   <dbl>
##  1 drift  Australia Test   29.1  35.5  29.1  7.23  7.23   NaN   NaN  0.210
##  2 drift  Canada    Test   33.3  37.2  33.3  6.09  6.09   NaN   NaN -0.229
##  3 drift  Japan     Test   14.7  17.9  14.7  2.44  2.44   NaN   NaN -0.229
##  4 drift  USA       Test   75.9  76.2  75.9 12.7  12.7    NaN   NaN -0.561
##  5 mean   Australia Test   35.7  42.3  35.7  8.89  8.89   NaN   NaN  0.216
##  6 mean   Canada    Test   90.4  92.9  90.4 16.7  16.7    NaN   NaN -0.0799
##  7 mean   Japan     Test  100.  101.  100.  16.8  16.8    NaN   NaN -0.534
##  8 mean   USA       Test   82.9  83.3  82.9 13.9  13.9    NaN   NaN -0.423
##  9 naive  Australia Test   34.7  41.5  34.7  8.64  8.64   NaN   NaN  0.216
## 10 naive  Canada    Test   46.2  51.0  46.2  8.46  8.46   NaN   NaN -0.0799
## 11 naive  Japan     Test   36.3  37.8  36.3  6.06  6.06   NaN   NaN -0.534
## 12 naive  USA       Test   82.1  82.5  82.1 13.8  13.8    NaN   NaN -0.423
```

According to the results, drifting method has the best performance.

d.Do the residuals from the best method resemble white noise?

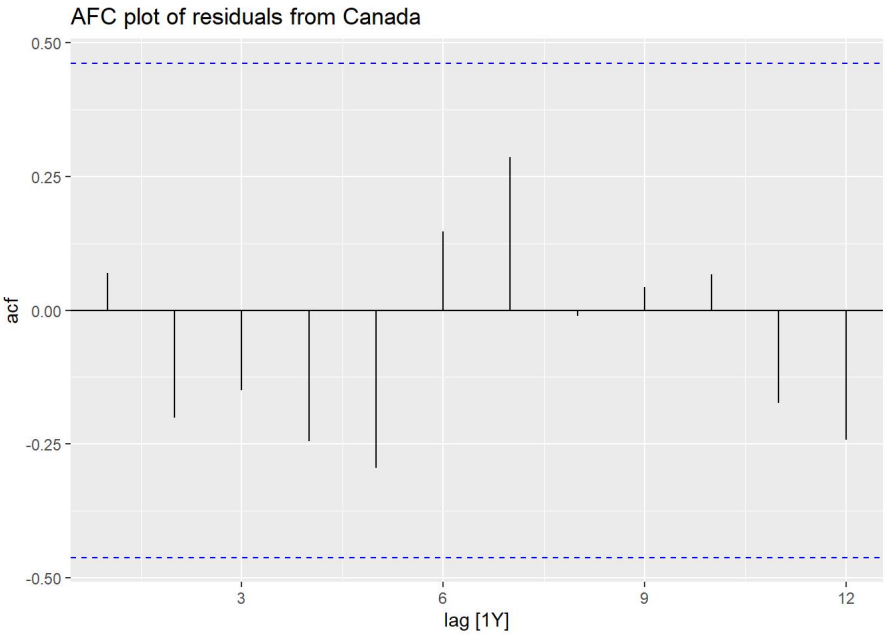
```
aug <- hh_training %>%
  model(NAIVE(Wealth ~ drift())) %>%
  augment()

res_aus <- aug %>%
  filter(Country == "Australia") %>%
  select(.innov)

res_ca <- aug %>%
  filter(Country == "Canada") %>%
  select(.innov)

res_jp <- aug %>%
  filter(Country == "Japan") %>%
  select(.innov)

res_us <- aug %>%
  filter(Country == "USA") %>%
  select(.innov)

res_aus %>% ACF() %>%
  autoplot() +
  labs(title = "AFC plot of residuals from Australia")
```

```
## Response variable not specified, automatically selected `var = .innov`
```



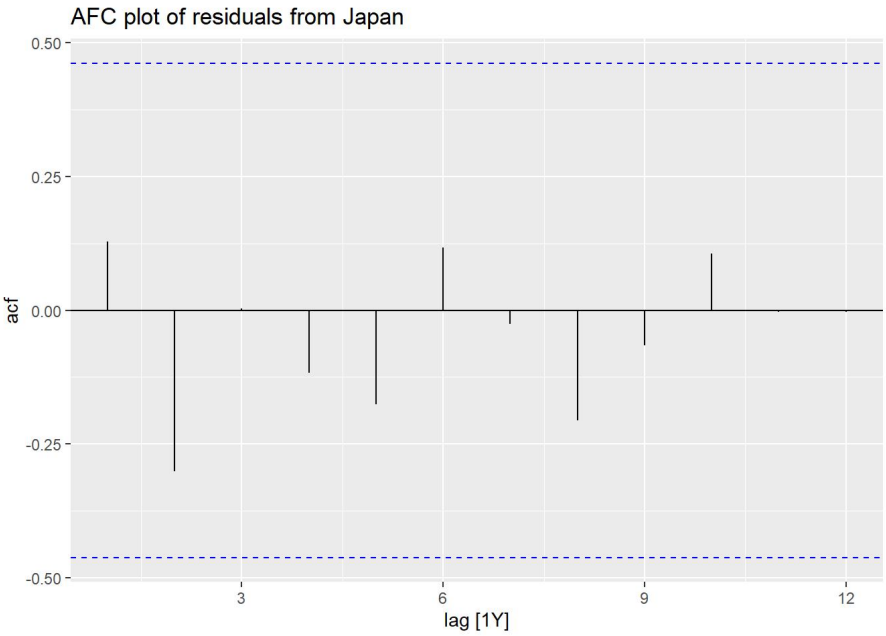

```
res_ca %>% ACF() %>%
  autoplot() +
  labs(title = "AFC plot of residuals from Canada")
```

```
## Response variable not specified, automatically selected `var = .innov`
```



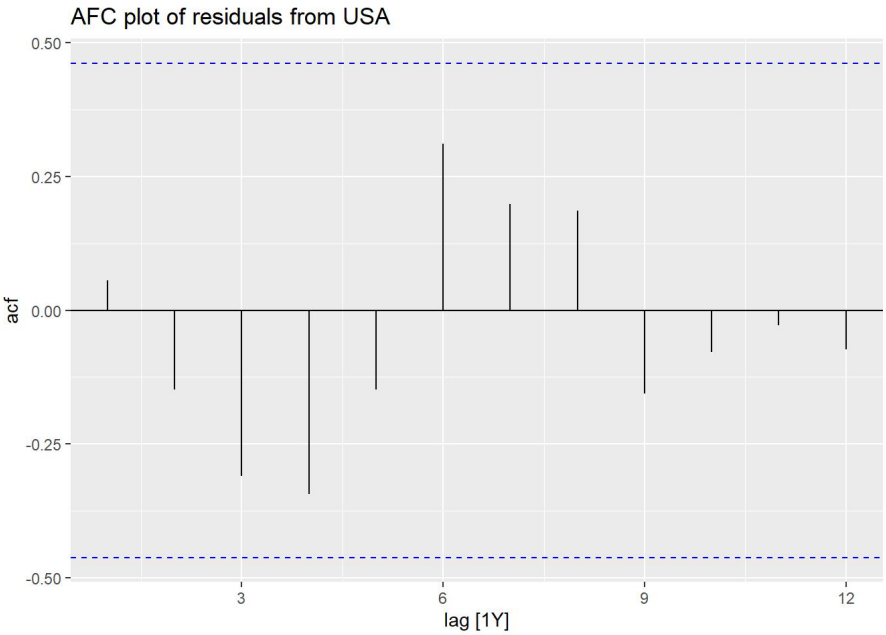

```
res_jp %>% ACF() %>%
  autoplot() +
  labs(title = "AFC plot of residuals from Japan")
```

```
## Response variable not specified, automatically selected `var = .innov`
```




```
res_us %>% ACF() %>%
  autoplot() +
  labs(title = "AFC plot of residuals from USA")
```

```
## Response variable not specified, automatically selected `var = .innov`
```



According to ACF plots of residuals of modeling from the 4 countries, residuals of Australia and Japan behave similarly to white noise.

Although the acf values of residuals of Canada and the US are within the threshold, we observe some seasonal patterns.

So whether they are truely white noise require more data

5.11.12

tourism contains quarterly visitor nights (in thousands) from 1998 to 2017 for 76 regions of Australia.

```
data("tourism")
```

a. Extract data from the Gold Coast region using *filter()* and aggregate total overnight trips (sum over *Purpose*) using *summarise()*. Call this new dataset gc_tourism.

```
tourism <- as_tibble(tourism)
gc_tourism <- tourism %>%
  filter(Region == "Gold Coast") %>%
  group_by(Quarter) %>%
  summarise(Trips = sum(Trips)) %>%
  as_tsibble(index = Quarter)
```

b. Using slice() or filter(), create three training sets for this data excluding the last 1, 2 and 3 years. For example, gc_train_1 <- gc_tourism |> slice(1:(n()-4)).

```
gc_train_1 <- gc_tourism %>%
  filter_index("1998 Q1" ~ "2016 Q4")

gc_train_2 <- gc_tourism %>%
  filter_index("1998 Q1" ~ "2015 Q4")

gc_train_3 <- gc_tourism %>%
  filter_index("1998 Q1" ~ "2014 Q4")
```

c. Compute one year of forecasts for each training set using the seasonal naïve (SNAIVE()) method. Call these gc_fc_1, gc_fc_2 and gc_fc_3, respectively.

```
gc_fc_1 <- gc_train_1 %>%
  model(snaive = SNAIVE(Trips)) %>%
  forecast(h = 4)

gc_fc_2 <- gc_train_2 %>%
  model(snaive = SNAIVE(Trips)) %>%
  forecast(h = 4)

gc_fc_3 <- gc_train_3 %>%
  model(snaive = SNAIVE(Trips)) %>%
  forecast(h = 4)
```

d. Use accuracy() to compare the test set forecast accuracy using MAPE. Comment on these.

```
accur_MAPE_1 <- gc_fc_1 %>%
  accuracy(gc_tourism)

accur_MAPE_2 <- gc_fc_2 %>%
  accuracy(gc_train_1)

accur_MAPE_3 <- gc_fc_3 %>%
  accuracy(gc_train_2)

accur_MAPE_1$MAPE
```

```
## [1] 15.06055
```

```
accur_MAPE_2$MAPE
```

```
## [1] 4.320729
```

```
accur_MAPE_3$MAPE
```

```
## [1] 9.067368
```

The the forecast of the second dataset is more accurate than the others. The model gennerally gives accurate results.

7.10.4

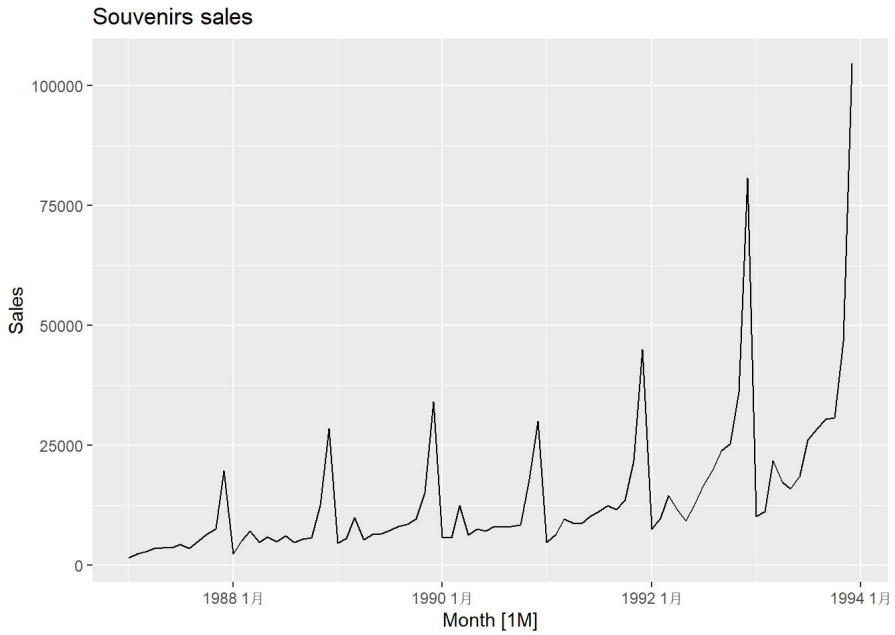
The data set *souvenirs* concerns the monthly sales figures of a shop which opened in January 1987 and sells gifts, souvenirs, and novelties. The shop is situated on the wharf at a beach resort town in Queensland, Australia. The sales volume varies with the seasonal population of tourists. There is a large influx of visitors to the town at Christmas and for the local surfing festival, held every March since 1988. Over time, the shop has expanded its premises, range of products, and staff

```
data("souvenirs")
```

a.Produce a time plot of the data and describe the patterns in the graph. Identify any unusual or unexpected fluctuations in the time series.

```
souvenirs %>%
  autoplot() +
  labs(title = "Souvenirs sales")
```

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



There is unexpected inceasing right after the end of each year. This might be due to the surfing festival.

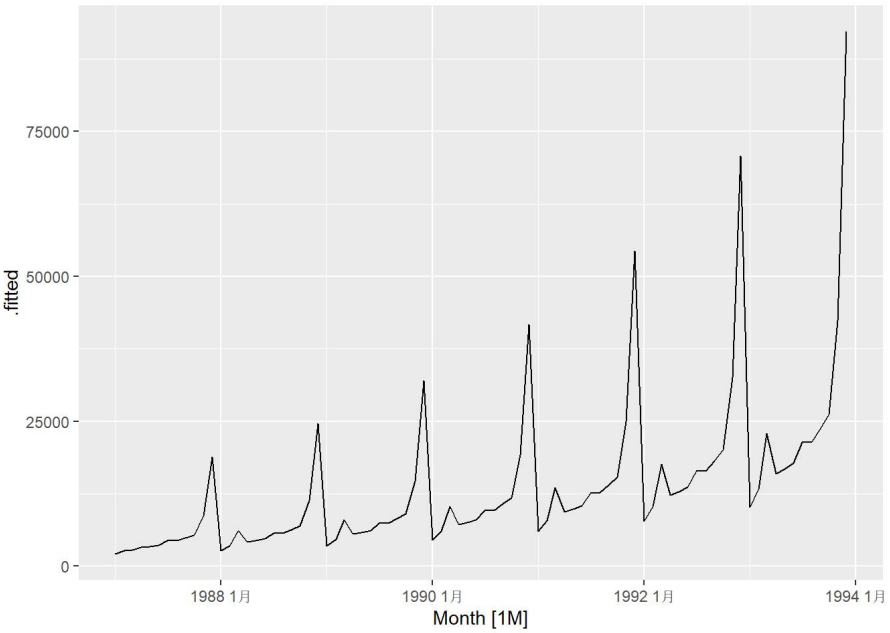
b.Explain why it is necessary to take logarithms of these data before fitting a model.
The data shows a variation that inceases with the level of the series. Logarithmic transformation works well in interpreting this multiplica

c.Fit a regression model to the logarithms of these sales data with a linear trend, seasonal dummies and a “surfing festival” dummy variable.

```
souvenirs_log <- souvenirs %>%
  mutate(log_sales = log(Sales))

d_festival <- rep(0, nrow(souvenirs_log))
d_festival[seq_along(d_festival)%%12 == 3] <- 1
d_festival[3] <- 0

souvenirs_log$Dummy <- d_festival

souvenirs_fit <- souvenirs_log %>%
  model(TSLM(log(Sales) ~ trend() + season() + Dummy))

augment(souvenirs_fit)
```

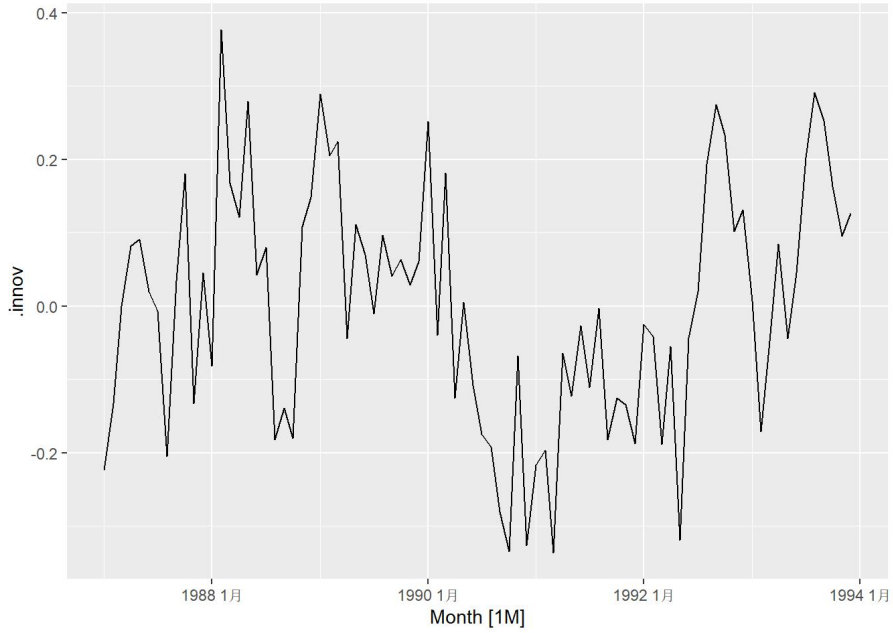
```
## # A tsibble: 84 x 6 [1M]
## # Key:       .model [1]
##    .model                                  Month Sales .fitted .resid    .innov
##    <chr>                                   <mth> <dbl>   <dbl>  <dbl>     <dbl>
##  1 TSLM(log(Sales) ~ trend() + season(… 1987 1月 1665.   2083. -418.  -2.24e- 1
##  2 TSLM(log(Sales) ~ trend() + season(… 1987 2月 2398.   2738. -341.  -1.33e- 1
##  3 TSLM(log(Sales) ~ trend() + season(… 1987 3月 2841.   2841.    0    3.12e-17
##  4 TSLM(log(Sales) ~ trend() + season(… 1987 4月 3547.   3268.  280.   8.21e- 2
##  5 TSLM(log(Sales) ~ trend() + season(… 1987 5月 3753.   3426.  327.   9.10e- 2
##  6 TSLM(log(Sales) ~ trend() + season(… 1987 6月 3715.   3643.   71.6  1.94e- 2
##  7 TSLM(log(Sales) ~ trend() + season(… 1987 7月 4350.   4378.  -28.0 -6.42e- 3
##  8 TSLM(log(Sales) ~ trend() + season(… 1987 8月 3566.   4376. -809.  -2.04e- 1
##  9 TSLM(log(Sales) ~ trend() + season(… 1987 9月 5022.   4852.  170.   3.44e- 2
## 10 TSLM(log(Sales) ~ trend() + season(… 1987 10月 6423.  5363. 1061.   1.80e- 1
## # … with 74 more rows
```

```
augment(souvenirs_fit) %>%
  autoplot(.fitted)
```

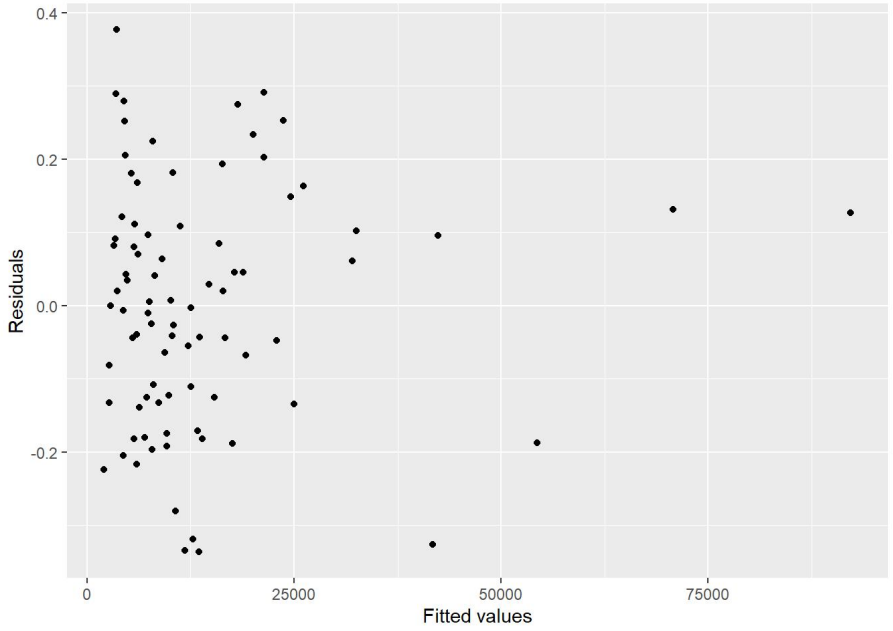


d. Plot the residuals against time and against the fitted values. Do these plots reveal any problems with the model?

```
augment(souvenirs_fit) %>%
  autoplot(.innov)
```



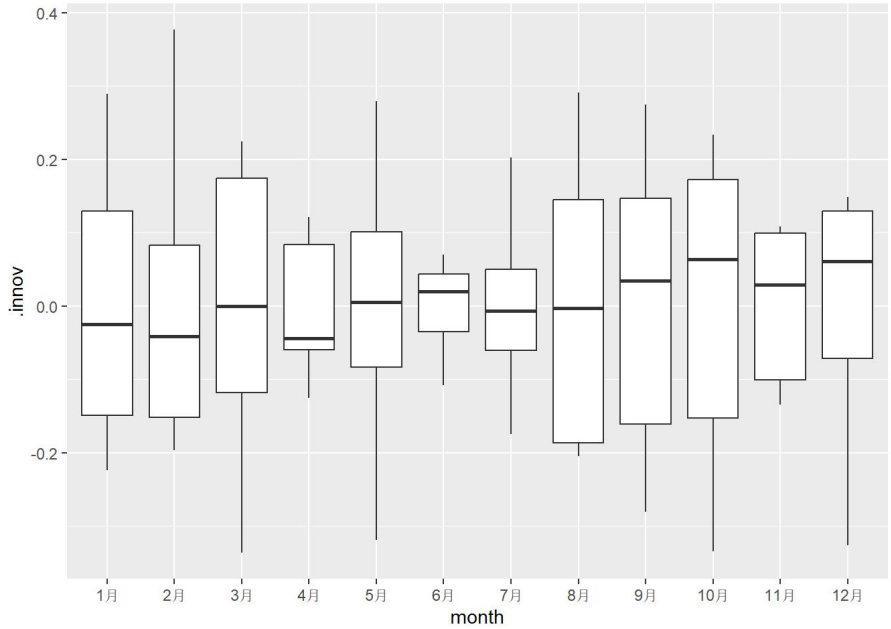

```
augment(souvenirs_fit) %>%
  ggplot(aes(x = .fitted, y = .innov)) +
  labs(y = "Residuals", x = "Fitted values") +
  geom_point()
```



The plot of residuals vs. time indicates that there is still some pattern in the residuals.

e. Do boxplots of the residuals for each month. Does this reveal any problems with the model?

```
augment(souvenirs_fit) %>%
  mutate(month = month(Month, label = T)) %>%
   ggplot(aes(x = month, y = .innov)) +
   geom_boxplot()
```



The boxplot shows largely different variation among different months.

f. What do the values of the coefficients tell you about each variable?

```
report(souvenirs_fit)
```

```
## Series: Sales
## Model: TSLM
## Transformation: log(Sales)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.336727 -0.127571  0.002568  0.109106  0.376714
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)    7.6196670  0.0742471 102.626  < 2e-16 ***
## trend()        0.0220198  0.0008268  26.634  < 2e-16 ***
## season()year2  0.2514168  0.0956790   2.628 0.010555 *
## season()year3  0.2660828  0.1934044   1.376 0.173275
## season()year4  0.3840535  0.0957075   4.013 0.000148 ***
## season()year5  0.4094870  0.0957325   4.277 5.88e-05 ***
## season()year6  0.4488283  0.0957647   4.687 1.33e-05 ***
## season()year7  0.6104545  0.0958039   6.372 1.71e-08 ***
## season()year8  0.5879644  0.0958503   6.134 4.53e-08 ***
## season()year9  0.6693299  0.0959037   6.979 1.36e-09 ***
## season()year10 0.7473919  0.0959643   7.788 4.48e-11 ***
## season()year11 1.2067479  0.0960319  12.566  < 2e-16 ***
## season()year12 1.9622412  0.0961066  20.417  < 2e-16 ***
## Dummy          0.5015151  0.1964273   2.553 0.012856 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.179 on 70 degrees of freedom
## Multiple R-squared: 0.9567,  Adjusted R-squared: 0.9487
## F-statistic:   119 on 13 and 70 DF,  p-value: < 2.22e-16
```

trend() coefficient suggests that with every month sales increases on average by 2.2%.

season()year2 coefficient shows that February log sales are greater than January on average by 25.1%, after allowing for the trend.(All the season() coefficients can be interpreted in the same way.)

Dummy coefficient shows that for months that include the surfing festival, log sales increases on average by 50.1% compared to months without the festival, after allowing for the trend and seasonality.

g.What does the Ljung-Box test tell you about your model?

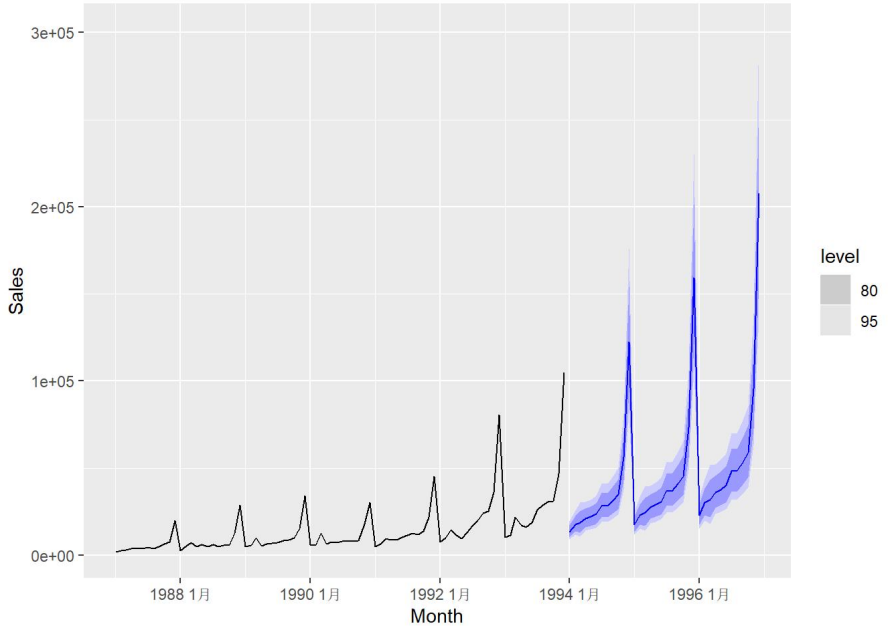
```
augment(souvenirs_fit) %>%
  features(.innov, ljung_box, dof = 14, lag = 24)
```

```
## # A tibble: 1 × 3
##   .model                                    lb_stat lb_pvalue
##   <chr>                                       <dbl>     <dbl>
## 1 TSLM(log(Sales) ~ trend() + season() + Dummy)    112.         0
```

The test indicates that the residuals are still correlated.

h. Regardless of your answers to the above questions, use your regression model to predict the monthly sales for 1994, 1995, and 1996. Produce prediction intervals for each of your forecasts.

```
d_festival_future <- rep(0, 36)
d_festival[seq_along(d_festival)%%12 == 3] <- 1
d_festival[3] <- 0

souvenirs_future <- new_data(souvenirs, n = 36) %>%
  mutate(Dummy = d_festival_future)

souvenirs_fit %>%
  forecast(new_data = souvenirs_future) %>%
  autoplot(souvenirs)
```



i.How could you improve these predictions by modifying the model? The model can be improved by taking into account nonlinearity of the trend.

The model assumed linearity in the trend. However this assumption may not be true according to time plot.
To improve the model, one way can be applying nonlinear methods, instead of linear ones.