Ekonometrija (II) su R.

Tomas Dzedulionis,
III kursas, Ekonominė analizė
Vilniaus universitetas
Ekonomikos ir verslo administravimo fakultetas

2020m.

Turinys

1 Įvadas **3**

2 Klasikinis laiko eilutės išskaidymas **3**

2.1 Laiko eilutės samprata . . . 3

2.2 Duomenų importas ir vertimas laiko eilute . . . 3

2.3 Laiko eilutės išskaidymas . . . 5

2.3.1 Multiplikatyviu būdu . . . 5

2.3.2 Adityviu būdu . . . 6

2.3.3 Desezonizavimas . . . 7

2.4 Aprašomosios statistikos . . . 7

3 Eksponentinis glodinimas **8**

3.1 Paprastas eksponentinis glodinimas (SES) . . . 8

3.2 Dvigubas eksponentinis glodinimas (Holt glodinimas) . . . 9

3.3 Holt-Winters be sezoniškumo komponentės. . . . 11

3.4 Holt_Winters su adityviu sezoniškumu. 13
3.5 Holt_Winters su multiplikatyviu sezoniškumu. 14
3.6 Hodrick-Prescott filtras 16

4 ARIMA **17**

4.1 Stacionarumo tikrinimas 17
4.1.1 Grafinė analizė 17
4.1.2 Vienetinės šaknies testai 19
4.2 Stacionarizavimas 20
4.3 ARIMA modelio sudarymas 23
4.4 Prognozavimas 25

5 VAR **26**

5.1 Pirmoji grafinė analizė 27
5.2 Stacionarizavimas 29
5.3 VAR Modelio sudarymas 31
5.4 Prognozė 34
5.5 Granger priežastingumo testai 34
5.6 Reakcija į impulsus 35

6 Panelinių duomenų modeliai **37**

6.1 Grafinė analizė 38
6.2 Stacionarumo tikrinimas [1] 42
6.3 Duomenų stacionarizavimas. 43
6.4 Stacionarumo tikrinimas [2] 43
6.5 Grafinė analizė [2] 44
6.6 Pastovių konstantų modelis. 47
6.7 Fiksuotų efektų modelis 49
6.8 Atsitiktinių efektų modelis 51
6.9 Galutinis modelis 52

1 Įvadas

Ši mokomoji medžiaga skirta ekonomikos studentams, kurie studijuoja ekonometriją II ir nori plėsti R programavimo žinias bei pasitelkti **R** programą mokomojo dalyko darbams atlikti. Medžiagoje apžvelgiamos šios temos:

- Klasikinis laiko eilutės išskaidymas
- Laiko eilutės eksponentinis glodinimas ir filtrai
- ARIMA modeliai
- VAR modeliai
- Panelinių duomenų modeliai.

Mokomoji medžiaga/konspektas rašytas mokymosi tikslais.

2 Klasikinis laiko eilutės išskaidymas

2.1 Laiko eilutės samprata

Laiko eilutė *(laiko seka)* –reiškinio periodiškų stebėjimų visuma, kurių duomenys tai periodo metu fiksuoti stebėjimų dydžiai arba stebimų dydžių suma.
Laiko eilutės gali būti suformuotos iš įvairaus dažnumo, tačiau vienodo periodiškumo duomenų: valandinių, kasdienių, savaitinių, mėnesinių, metinių ir pan.
Norint duomenims suteikti laiko eilutės formą naudojame komandą `ts(...)`, o skliaustų viduje nurodome duomenis `data=...`, pradžios reikšmę (datą) `start=...`, pabaigą `end=...` ir stebejimų dažnį `frequency=...` (pavyzdžiui ketvirtiniai duomenys `frequency=4`).

2.2 Duomenų importas ir vertimas laiko eilute

Duomenims naudosime paskaitoje suteiktus NETO darbo užmokesčio ūkio šakose duomenis (2008K1-2020K2), jie įkelti į Google Drive platformą lengvesniam pasiekiamumui iš skirtingų kompiuterių.

Importavimas ir duomenų valymas

```
# Nurodome nuorodą į duomenis
url <- "https://drive.google.com/uc?export=download&id=1IGFYeF3E58_g9Ak5sd5BGkt1jgDBfpUq"
# Importuojame duomenis
df <- read.delim(url, header=TRUE, sep=";", na.strings=c("", "NA"), encoding = "UTF-8")
# Pasileidžiame paketą duomenų valymui
if(!require("tidyverse")) install.packages("tidyverse"); library("tidyverse")
# Valome duomenis (išmetame X stulpelį, panaikiname tuščias reikšmes (NA), pervadiname stulpelį)
df <- df%>% select(-"X") %>% na.omit() %>% rename(Metai="X.U.FEFF.")
# Renkamės nagrinėjamą sritį, mūsų atveju Finansinė ir draudimo veikla ("K")
df <- select(df, 1,"K")
# Verčiame skaičius iš character vektoriaus į numeric vektorių
df$K <- as.numeric(sub(",", ".", df$K, fixed=T))
```

```r
# Paverčiame duomenis į laiko eilutės formatą
data <- ts(df$K, frequency = 4, start = c(2008, 1))
```

```r
data
```

```
##         Qtr1   Qtr2   Qtr3   Qtr4
## 2008  880.4  899.5  913.0  909.1
## 2009 1009.1  867.3  860.8  836.5
## 2010  843.3  840.6  832.2  825.7
## 2011  865.0  870.3  864.1  872.4
## 2012  934.9  909.5  874.1  883.0
## 2013  962.1  921.3  911.7  895.6
## 2014 1009.7  950.2  939.2  953.5
## 2015 1045.1 1005.6 1016.1 1012.9
## 2016 1132.3 1095.4 1064.1 1083.4
## 2017 1199.9 1175.8 1176.2 1161.1
## 2018 1297.8 1292.5 1220.2 1241.8
## 2019 1444.7 1430.9 1383.0 1396.7
## 2020 1541.9 1529.5
```

2.3 Laiko eilutės išskaidymas

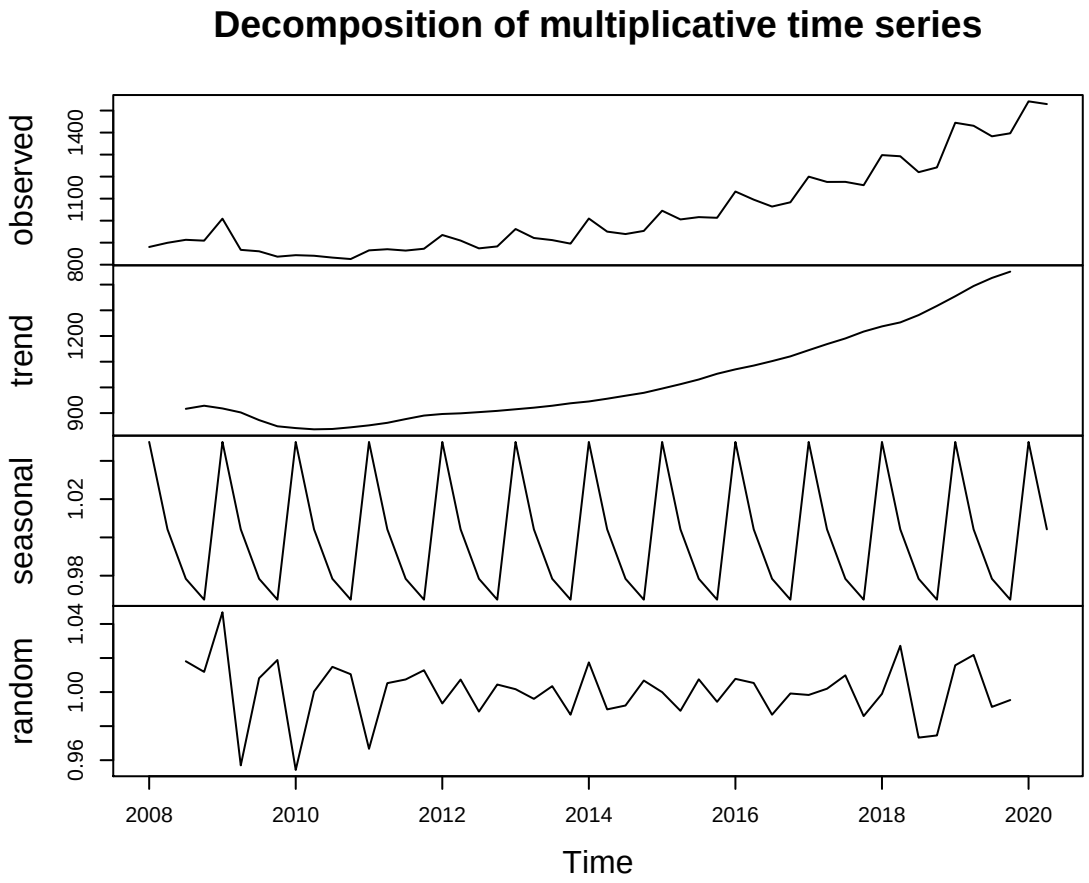
2.3.1 Multiplikatyviu būdu

Skaidome laiko eilutę multiplikatyviu būdu.

```
data_multi <- decompose(data, type="multiplicative")
## Sezoniškumo indeksai slėpsis šiame objekte "data_multi"
```

Sezoniškumo indeksus galime išsitraukti panaudoję komandą
`data_multi$seasonal`

```
plot(data_multi)
```



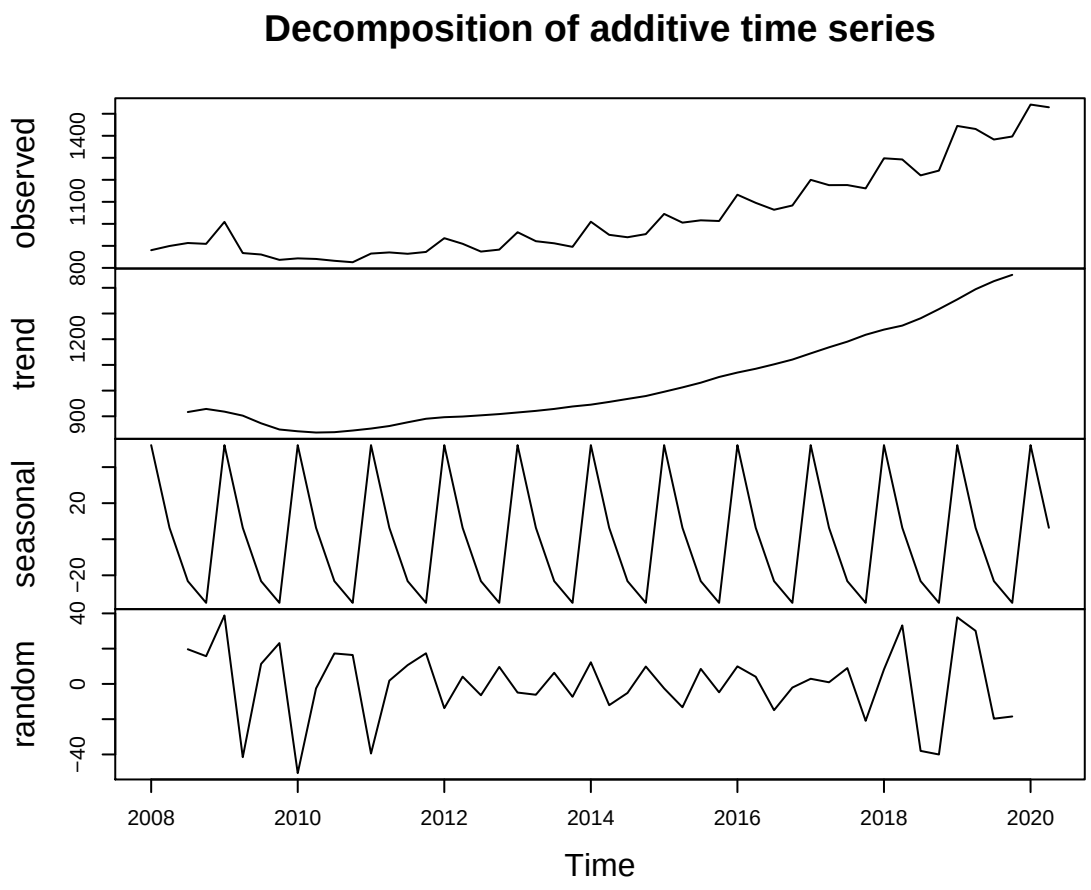
2.3.2 Adityviu būdu

Skaidome laiko eilutę adityviu būdu.

```
data_adityv <- decompose(data, type="additive")
## Sezoniškumo indeksai slėpsis šiame objekte "data_adityv"
```

Sezoniškumo indeksus galime išsitraukti panaudoję komandą
`data_adityv$seasonal`

```
plot(data_adityv)
```



2.3.3 Desezonizavimas

Norint desezonizuoti laiko eilutę, reikia iš jos atimti sezoniškumo indeksus.
Pavyzdys su adityviu išskaidymu:

```
data - data_adityv$seasonal
##           Qtr1      Qtr2      Qtr3      Qtr4
## 2008  828.2194  893.1558  936.2442  944.3806
## 2009  956.9194  860.9558  884.0442  871.7806
## 2010  791.1194  834.2558  855.4442  860.9806
## 2011  812.8194  863.9558  887.3442  907.6806
## 2012  882.7194  903.1558  897.3442  918.2806
## 2013  909.9194  914.9558  934.9442  930.8806
## 2014  957.5194  943.8558  962.4442  988.7806
## 2015  992.9194  999.2558 1039.3442 1048.1806
## 2016 1080.1194 1089.0558 1087.3442 1118.6806
## 2017 1147.7194 1169.4558 1199.4442 1196.3806
## 2018 1245.6194 1286.1558 1243.4442 1277.0806
## 2019 1392.5194 1424.5558 1406.2442 1431.9806
## 2020 1489.7194 1523.1558
```

2.4 Aprašomosios statistikos

Aprašomosios statistikos gali būti išgaunomas keliais būdais:

1. Naudojant komandą `summary()`

```
summary(data)
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   825.7   881.0   957.8  1041.6  1172.1  1541.9
```

2. Naudojant paketą `psych`

```
if(!require("psych")) install.packages("psych"); library("psych")
describe(data)
##    vars  n    mean     sd median trimmed    mad   min    max range skew
## X1    1 50 1041.62 201.18  957.8 1013.98 138.25 825.7 1541.9 716.2 0.98
##    kurtosis    se
## X1    -0.19 28.45
```

3 Eksponentinis glodinimas

Įvairiems glodinimo būdams naudosime paketą `fpp2`.

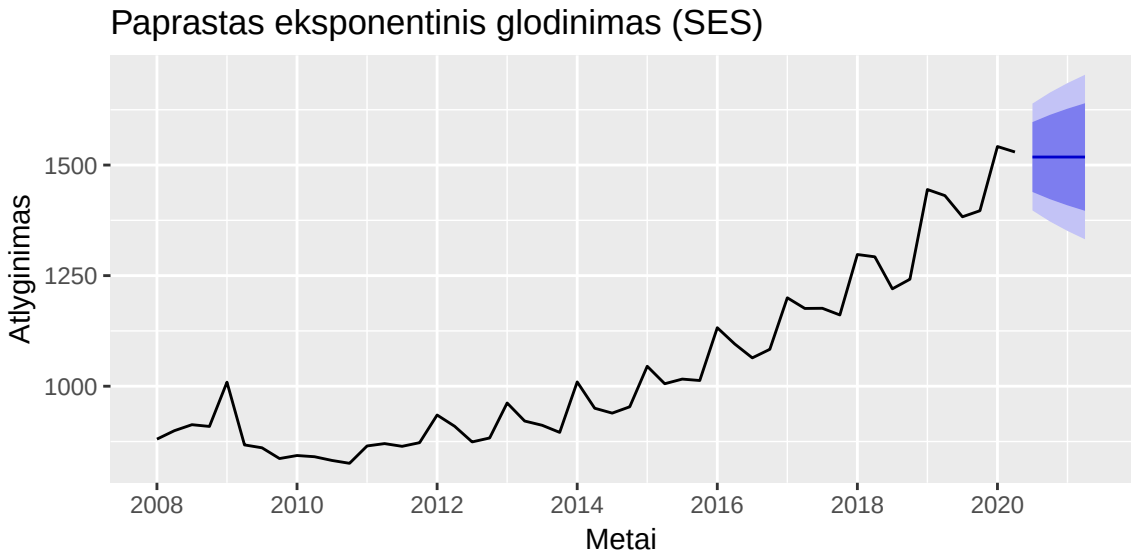
```
if(!require("fpp2")) install.packages("fpp2"); library("fpp2")
```

3.1 Paprastas eksponentinis glodinimas (SES)

Naudosime komandą `ses()`. Optimalų alpha koeficientą parinks pati programa.

```
# alpha=NULL reiškia, jog komanda parinks optimalų alpha koeficientą
# h=4 nurodo suglodinti kitiems 4-iems periodams
ses.data <- ses(data, alpha = NULL, h=4)
```

```
autoplot(ses.data, xlab="Metai", ylab="Atlyginimas")+
  ggtitle("Paprastas eksponentinis glodinimas (SES)")+
  scale_x_continuous(breaks = seq(2008,2020,2))
```



Modelio paklaidas išgausime komanda `accuracy()`

```
accuracy(ses.data)
##                     ME     RMSE      MAE       MPE     MAPE      MASE        ACF1
## Training set 18.6336 60.38712 39.06815 1.396452 3.554291 0.5937212 -0.04636163
```

Modelio apibendrinta informacija išgaunama komanda `summary()`. Matome, kad komanda parinko alpha koeficientą lygų 0.6758.

```
summary(ses.data)
##
## Forecast method: Simple exponential smoothing
##
## Model Information:
## Simple exponential smoothing
##
## Call:
##  ses(y = data, h = 4, alpha = NULL)
##
##   Smoothing parameters:
##     alpha = 0.6758
##
##   Initial states:
##     l = 888.4658
##
##   sigma:  61.6323
##
##      AIC     AICc      BIC
## 611.6787 612.2005 617.4148
##
## Error measures:
##                    ME     RMSE      MAE      MPE     MAPE      MASE        ACF1
## Training set 18.6336 60.38712 39.06815 1.396452 3.554291 0.5937212 -0.04636163
##
## Forecasts:
##         Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 2020 Q3       1518.138 1439.153 1597.123 1397.341 1638.935
## 2020 Q4       1518.138 1422.806 1613.470 1372.340 1663.936
## 2021 Q1       1518.138 1408.877 1627.398 1351.038 1685.237
## 2021 Q2       1518.138 1396.534 1639.742 1332.161 1704.115
```

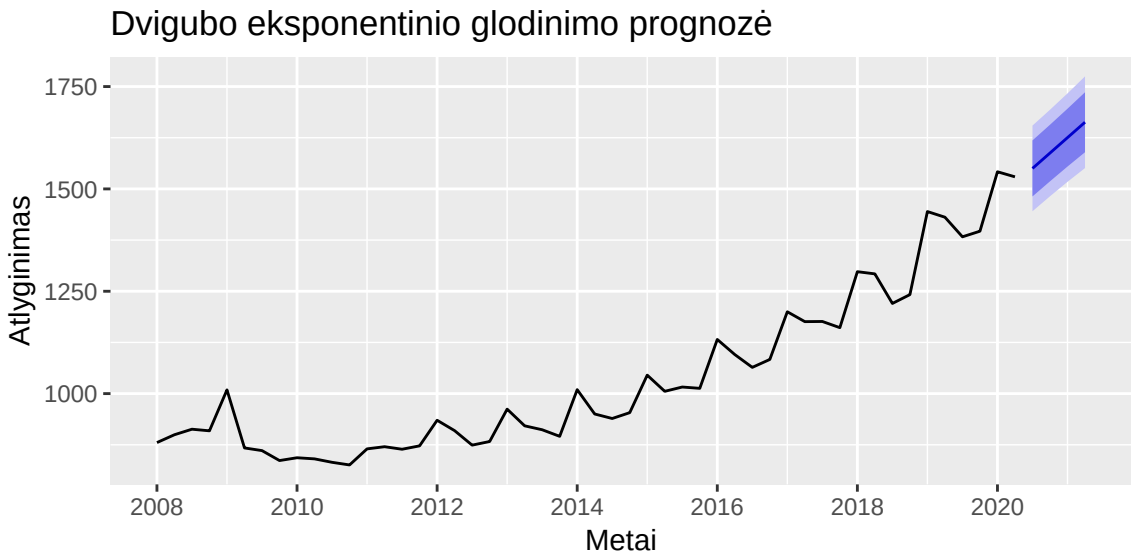
Paprastasis eksponentinis glodinimas netinkamas šiems duomenims, kadangi duomenys turi stipriai išreikštą trendą ir sezoniškumą.

3.2 Dvigubas eksponentinis glodinimas (Holt glodinimas)

Šiam glodinimo būdui naudojama komanda `holt()`. Optimalius alpha ir beta koeficientus parinks pati programa.

```
holt.data <- holt(data, h = 4, alpha=NULL, beta = NULL)
```

```
autoplot(holt.data, xlab="Metai", ylab="Atlyginimas")+
  ggtitle("Dvigubo eksponentinio glodinimo prognozė")+
  scale_x_continuous(breaks = seq(2008,2020,2))
```



Paklaidos

```
accuracy(holt.data)
##                     ME     RMSE      MAE      MPE     MAPE      MASE      ACF1
## Training set 15.52436 51.19839 39.53951 1.358421 3.701622 0.6008845 0.1088822
```

Apibendrinta informacija. Programa parinko alpha=0.0707, beta=0.0707.

```
summary(holt.data) ### Alfa parinkta 0.9167, Beta 0.0001
##
## Forecast method: Holt's method
##
## Model Information:
## Holt's method
##
## Call:
##  holt(y = data, h = 4, alpha = NULL, beta = NULL)
##
##   Smoothing parameters:
##     alpha = 0.0707
##     beta  = 0.0707
##
##   Initial states:
##     l = 931.6853
##     b = -17.2868
```

```
##
##   sigma:  53.378
##
##      AIC     AICc      BIC
## 599.1720 600.5356 608.7321
##
## Error measures:
##                     ME     RMSE      MAE      MPE     MAPE      MASE      ACF1
## Training set 15.52436 51.19839 39.53951 1.358421 3.701622 0.6008845 0.1088822
##
## Forecasts:
##         Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 2020 Q3       1550.059 1481.652 1618.465 1445.440 1654.678
## 2020 Q4       1587.676 1518.588 1656.764 1482.015 1693.336
## 2021 Q1       1625.293 1554.697 1695.889 1517.325 1733.260
## 2021 Q2       1662.910 1589.709 1736.111 1550.958 1774.862
```

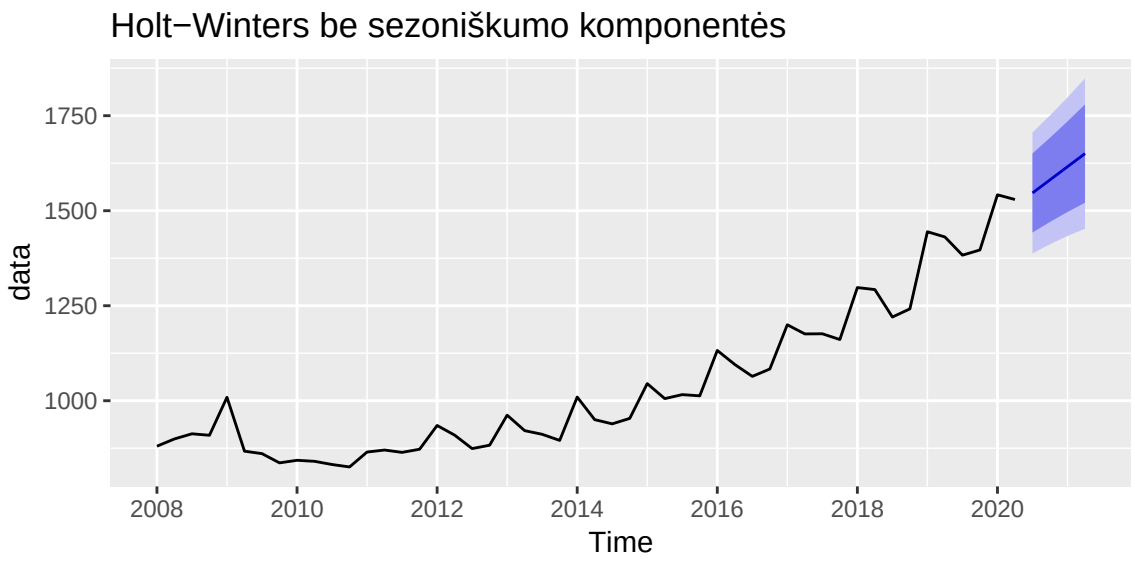
Glodinimo būdas netinka šiems duomenims, kadangi duomenys turi sezoniškumą.

3.3 Holt-Winters be sezoniškumo komponentės.

Naudosime komandą `ets()`. `model=...` parametras: Paklaidos - "Z" (programa nustato multiplikatyvios ar adityvios), Trendas - "Z" (programa nustato multiplikatyvi ar adityvi), Sezoniškumas - "N".

```
data.hw <- ets(data, model = "ZZN")
```

```
autoplot(forecast(data.hw,4, xlab="Metai", ylab="Atlyginimas"))+
  ggtitle("Holt-Winters be sezoniškumo komponentės")+
  scale_x_continuous(breaks = seq(2008,2020,2))
```



```r
summary(forecast(data.hw,4))
##
## Forecast method: ETS(M,A,N)
##
## Model Information:
## ETS(M,A,N)
##
## Call:
##  ets(y = data, model = "ZZN")
##
##   Smoothing parameters:
##     alpha = 0.2349
##     beta  = 0.0614
##
##   Initial states:
##     l = 918.7409
##     b = -7.3707
##
##   sigma:  0.0525
##
##      AIC     AICc      BIC
## 598.6590 600.0226 608.2191
##
## Error measures:
##                    ME     RMSE     MAE      MPE     MAPE      MASE       ACF1
## Training set 13.6687 51.99545 40.8965 1.115989 3.845188 0.6215069 0.02718329
##
## Forecasts:
##         Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 2020 Q3       1546.696 1442.663 1650.729 1387.591 1705.801
## 2020 Q4       1581.261 1470.515 1692.007 1411.889 1750.632
## 2021 Q1       1615.826 1496.679 1734.973 1433.606 1798.045
## 2021 Q2       1650.390 1521.087 1779.694 1452.638 1848.143
accuracy(data.hw)
##                    ME     RMSE     MAE      MPE     MAPE      MASE       ACF1
## Training set 13.6687 51.99545 40.8965 1.115989 3.845188 0.6215069 0.02718329
```

3.4 Holt_Winters su adityviu sezoniškumu.

Komanda ta pati, tiesiog keičiame `model=` parametrus. Sezoniškumas - "A".

```
data.hwadditive <- ets(data, model = "ZZA")
```

```
autoplot(forecast(data.hwadditive,4, xlab="Metai", ylab="Atlyginimas"))+
  ggtitle("Holt Winters su adityviu sezoniškumu")+
  scale_x_continuous(breaks = seq(2008,2020,2))
```


```
summary(forecast(data.hwadditive,4))
##
## Forecast method: ETS(A,A,A)
##
## Model Information:
## ETS(A,A,A)
##
## Call:
##  ets(y = data, model = "ZZA")
##
##   Smoothing parameters:
##     alpha = 0.5699
##     beta  = 0.0714
##     gamma = 0.2943
##
##   Initial states:
##     l = 877.3358
##     b = 5.3022
##     s = -15.3133 -22.1398 11.3864 26.0667
##
```

```
##   sigma:  37.1284
##
##      AIC     AICc      BIC
## 566.3217 570.8217 583.5299
##
## Error measures:
##                     ME     RMSE      MAE       MPE     MAPE      MASE       ACF1
## Training set 5.716956 34.02873 24.97444 0.4110585 2.469831 0.3795382 0.02991265
##
## Forecasts:
##         Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 2020 Q3       1479.960 1432.378 1527.542 1407.190 1552.730
## 2020 Q4       1501.717 1445.192 1558.243 1415.270 1588.165
## 2021 Q1       1654.941 1589.025 1720.856 1554.131 1755.750
## 2021 Q2       1630.009 1554.269 1705.749 1514.175 1745.843
accuracy(data.hwadditive)
##                     ME     RMSE      MAE       MPE     MAPE      MASE       ACF1
## Training set 5.716956 34.02873 24.97444 0.4110585 2.469831 0.3795382 0.02991265
```

3.5 Holt_Winters su multiplikatyviu sezoniškumu.

Komanda ta pati, tiesiog keičiame `model=` parametrus. Sezoniškumas - "M".

```
data.hwmultiplicative <- ets(data, model = "ZZM")
```

```
autoplot(forecast(data.hwmultiplicative,4, xlab="Metai", ylab="Atlyginimas"))+
  ggtitle("Holt Winters su multiplikatyviu sezoniškumu")+
  scale_x_continuous(breaks = seq(2008,2020,2))
```

```
summary(forecast(data.hwmultiplicative,4))
##
## Forecast method: ETS(M,A,M)
##
## Model Information:
## ETS(M,A,M)
##
## Call:
##  ets(y = data, model = "ZZM")
##
##   Smoothing parameters:
##     alpha = 0.6448
##     beta  = 0.126
##     gamma = 1e-04
##
##   Initial states:
##     l = 925.2988
##     b = 5.1542
##     s = 0.9701 0.9786 1.006 1.0453
##
##   sigma:  0.0352
##
##      AIC     AICc      BIC
## 563.1448 567.6448 580.3530
##
## Error measures:
##                     ME     RMSE      MAE       MPE     MAPE      MASE
## Training set 4.102487 32.54301 23.87374 0.3004123 2.385879 0.3628108
##                      ACF1
## Training set 0.007533274
##
## Forecasts:
##         Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 2020 Q3       1511.762 1443.495 1580.030 1407.356 1616.168
## 2020 Q4       1528.582 1442.041 1615.123 1396.229 1660.935
## 2021 Q1       1679.188 1563.755 1794.621 1502.648 1855.727
## 2021 Q2       1646.918 1512.789 1781.046 1441.786 1852.050
accuracy(data.hwmultiplicative)
##                     ME     RMSE      MAE       MPE     MAPE      MASE
## Training set 4.102487 32.54301 23.87374 0.3004123 2.385879 0.3628108
##                      ACF1
## Training set 0.007533274
```

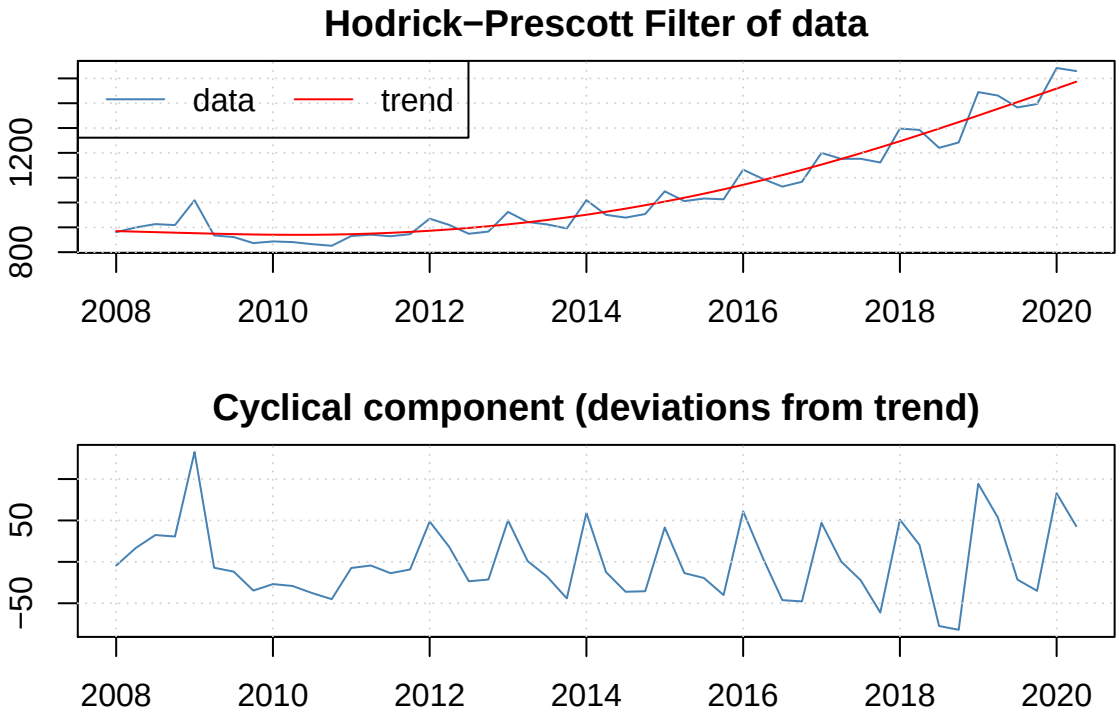
3.6 Hodrick-Prescott filtras

Naudosime paketą `mFilter` ir komandą `hpfilter()`

```
if(!require("mFilter")) install.packages("mFilter"); library("mFilter")
```

```
hp <- hpfilter(data, type="lambda", freq=1600)
```

```
plot(hp)
```



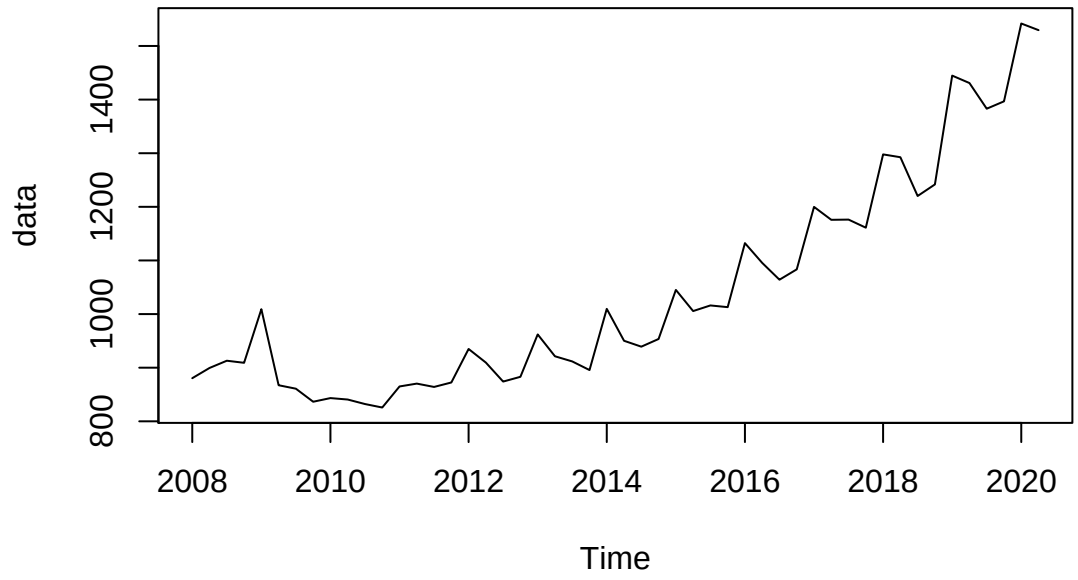
4 ARIMA

4.1 Stacionarumo tikrinimas

4.1.1 Grafinė analizė

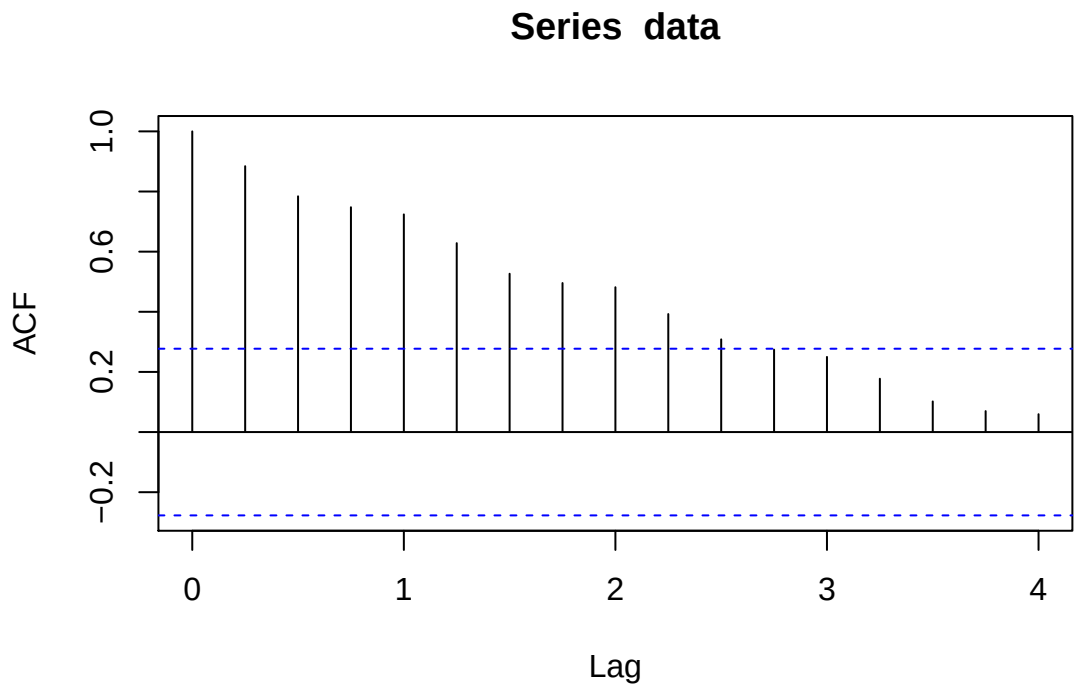
Įvertinsime duomenų stacionarumą žvigtelėję į laiko eilutės grafiką.
Matome, kad duomenys nėra stacionarūs - jie turi aiškiai išreikštą trendą.

```
plot(data)
```

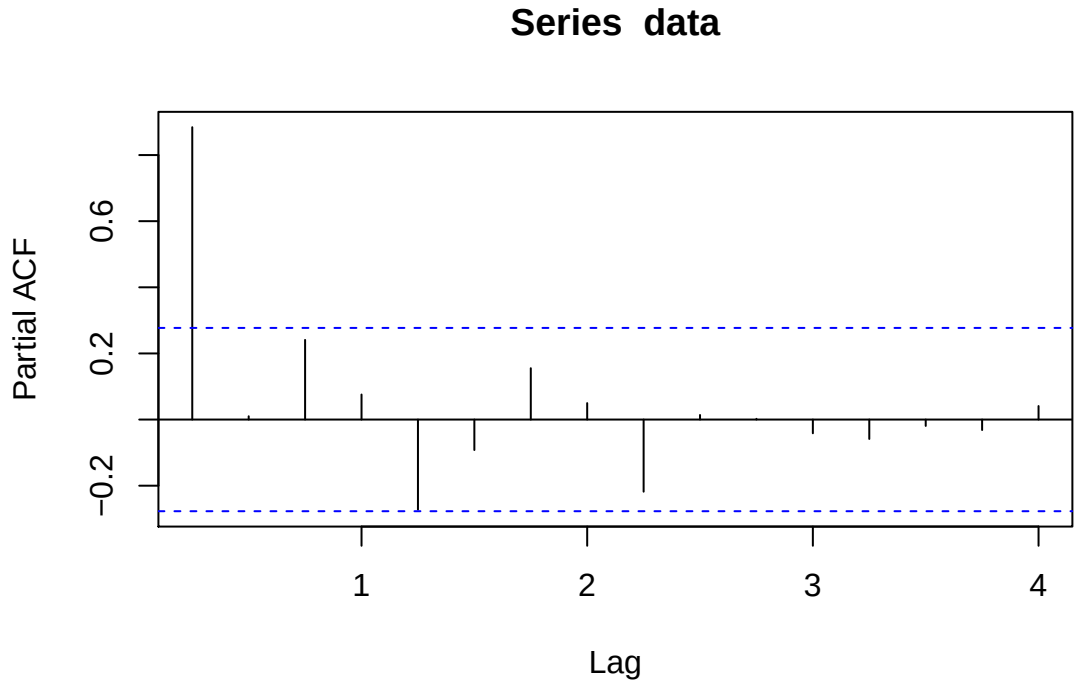


Taip pat galime pažiūrėti į autokoreliacijos ir dalinės autokoreliacijos grafikus.
Matome, jog autokoreliacijos funkcija laipsniškai mažėja, kas indikuoja apie pirmo lago autokoreliacija.
Tą patvirtina dalinės autokoreliacijos grafikas, su reikšminga pirmo laikotarpio autokoreliacija.

```
acf(data)
```




```
pacf(data)
```



4.1.2 Vienetinės šaknies testai

$H0$: Kintamasis nėra stacionarus ir turi vienetinę šaknį
$H1$: Kintamasis yra stacionarus ir neturi vienetinės šaknies.
Naudosime `fUnitRoots` paketą ir komandą `adfTest()`, bei paketą `tseries` ir komandą `adf.test()`.

```
if(!require("fUnitRoots")) install.packages("fUnitRoots"); library("fUnitRoots")
if(!require("tseries")) install.packages("tseries"); library("tseries")
```

Norėdami atlikti **Augmented Dickey-Fuller** testą su `adfTest()` komanda, turime nurodyti pagrindinius parametrus.
Lagų skaičius nurodome `lags=...` parametro pagalba, jei nenurodysime, pagal nutylėjimą bus paimtas 1 lagas. Testo tipą pasirenkame `type=...` parametro pagalba:
1. `type="nc"`, vienetinės šaknies testui be konstantos ir trendo.
2. `type="c"`, vienetinės šaknies testas su konstanta ir be trendo.
3. `type="ct"`, vienetinės šaknies testas su konstanta ir su trendu.
Tuo tarpu `adf.test()` pati parenka lagus bei atlieka testą su konstanta ir trendu. Patikrinkime pradinius duomenis.

```
adfTest(data, type="nc")
##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 1
##   STATISTIC:
##     Dickey-Fuller: 1.9088
##   P VALUE:
##     0.9838
##
## Description:
##  Sat Dec 19 19:58:12 2020 by user: PC
adfTest(data, type="c")
##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 1
##   STATISTIC:
##     Dickey-Fuller: 0.8607
##   P VALUE:
##     0.99
```

```
##
## Description:
##  Sat Dec 19 19:58:12 2020 by user: PC
adfTest(data, type="ct")
##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 1
##   STATISTIC:
##     Dickey-Fuller: -1.6155
##   P VALUE:
##     0.7285
##
## Description:
##  Sat Dec 19 19:58:12 2020 by user: PC
adf.test(data)
##
##  Augmented Dickey-Fuller Test
##
## data:  data
## Dickey-Fuller = -0.27633, Lag order = 3, p-value = 0.9882
## alternative hypothesis: stationary
```

Matome, jog visų testų p reikšmės > 0.05, kas rodo, jog negalime atmesti $H0$, o duomenys nėra stacionarūs.

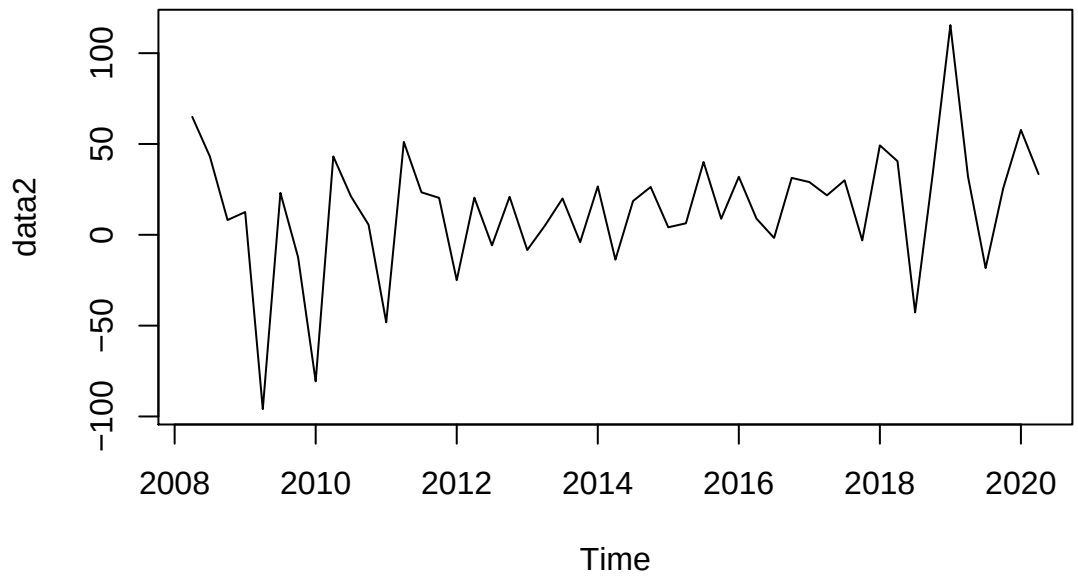
4.2 Stacionarizavimas

Duomenis stacionarizuoti galime dviem būdais - logaritmuojant (`log(...)`) ir/arba diferencijuojat (`diff=(...)`) duomenis. Šiuo atveju duomenis desezonizuosime, tuomet panaikinsime trendą duomenis diferencijuodami ir pereisime prie pirmųjų skirtumų analizės.

```
diff(data-data_adityv$seasonal)
##            Qtr1       Qtr2       Qtr3       Qtr4
## 2008             64.936364  43.088352   8.136458
## 2009  12.538826 -95.963636  23.088352 -12.263542
## 2010 -80.661174  43.136364  21.188352   5.536458
## 2011 -48.161174  51.136364  23.388352  20.336458
## 2012 -24.961174  20.436364  -5.811648  20.936458
## 2013  -8.361174   5.036364  19.988352  -4.063542
## 2014  26.638826 -13.663636  18.588352  26.336458
## 2015   4.138826   6.336364  40.088352   8.836458
```

```
## 2016  31.938826   8.936364  -1.711648  31.336458
## 2017  29.038826  21.736364  29.988352  -3.063542
## 2018  49.238826  40.536364 -42.711648  33.636458
## 2019 115.438826  32.036364 -18.311648  25.736458
## 2020  57.738826  33.436364
data2 <- diff(data-data_adityv$seasonal)
```

```
plot(data2)
```



Iš grafiko matome, jog diferencijuoti duomenys atrodo stacionarūs.
Patikrinkime tai vienetinės šaknies testu.

```
adfTest(data2, type="nc")
##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 1
##   STATISTIC:
##     Dickey-Fuller: -4.9008
##   P VALUE:
##     0.01
##
```

```
## Description:
##  Sat Dec 19 19:58:12 2020 by user: PC
adfTest(data2, type="c")
##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 1
##   STATISTIC:
##     Dickey-Fuller: -5.9329
##   P VALUE:
##     0.01
##
## Description:
##  Sat Dec 19 19:58:12 2020 by user: PC
adfTest(data2, type="ct")
##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 1
##   STATISTIC:
##     Dickey-Fuller: -8.0172
##   P VALUE:
##     0.01
##
## Description:
##  Sat Dec 19 19:58:12 2020 by user: PC
adf.test(data2)
##
##  Augmented Dickey-Fuller Test
##
## data:  data2
## Dickey-Fuller = -6.3211, Lag order = 3, p-value = 0.01
## alternative hypothesis: stationary
```

Matome, jog desezonizavus ir diferencijavus duomenis jie tapo stacionarūs (p reikšmės <0.05, todėl atmetame $H0$).

4.3 ARIMA modelio sudarymas

Arima modelio parinkimas R programoje yra labai paprastas. Viskas vyksta komandos `auto.arima` pagalba, kuri "prasuka" skirtingos eilės arima modelius, kol randa geriausią pagal Akaike informacijos kriterijų.

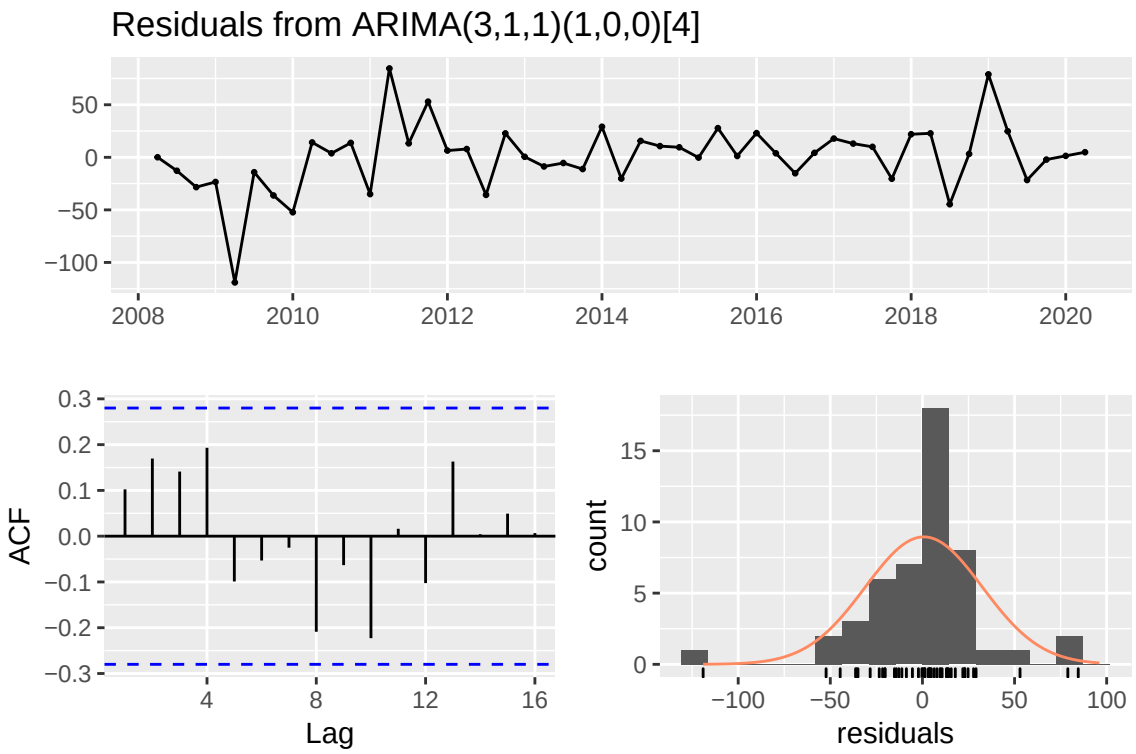
```
arima <- auto.arima(data2, ic="aic")
summary(arima)
## Series: data2
## ARIMA(3,1,1)(1,0,0)[4]
##
## Coefficients:
##           ar1      ar2      ar3      ma1     sar1
##       -0.7818  -0.8971  -0.6520  -0.3225  -0.5754
## s.e.   0.1745   0.1334   0.1625   0.2209   0.1809
##
## sigma^2 estimated as 1127:  log likelihood=-235.86
## AIC=483.73   AICc=485.77   BIC=494.95
##
## Training set error measures:
##                     ME    RMSE      MAE      MPE     MAPE      MASE      ACF1
## Training set 0.7566398 31.4442 21.44832 92.52787 128.0835 0.7228146 0.1023296
```

Matome, jog programa parinko SARIMA (ARIMA(3,1,1)(1,0,0)) modelį.
Dabar reikia atlikti Ljung-Box testą iš paketo `stats`, kad įsitikintume, jog paklaidos yra baltas triukšmas (ar nėra autokoreliacijos).
$H0$: Paklaidos yra baltas triukšmas.
$H1$: Paklaidos nėra baltas triukštmas.

```
if(!require("stats")) install.packages("stats"); library("stats")
```

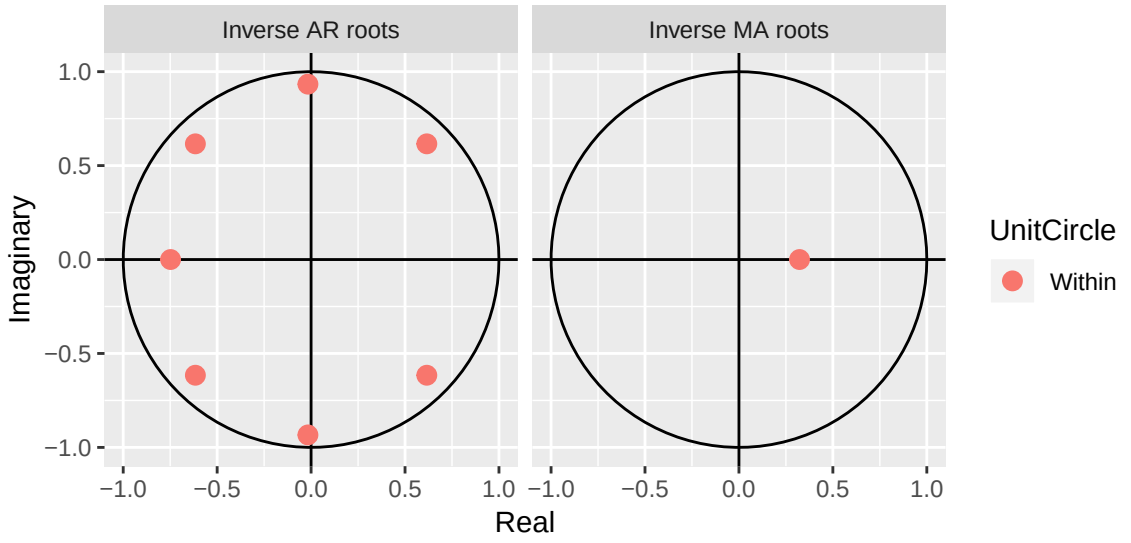
```
Box.test(residuals(arima), type="Ljung-Box", lag=12)
##
##  Box-Ljung test
##
## data:  residuals(arima)
## X-squared = 12.805, df = 12, p-value = 0.3834
```

```
checkresiduals(arima)
```



Matome, jog Ljung-Box testo p reikšmė >0.05, todėl negalime atmesti nulinės hipotezės, o tai reiškia, kad paklaidos yra baltas triukšmas beigi modelis yra tinkamas. Taip pat galime patikrinti atvirkštines modelio šaknis.

```
autoplot(arima)
```

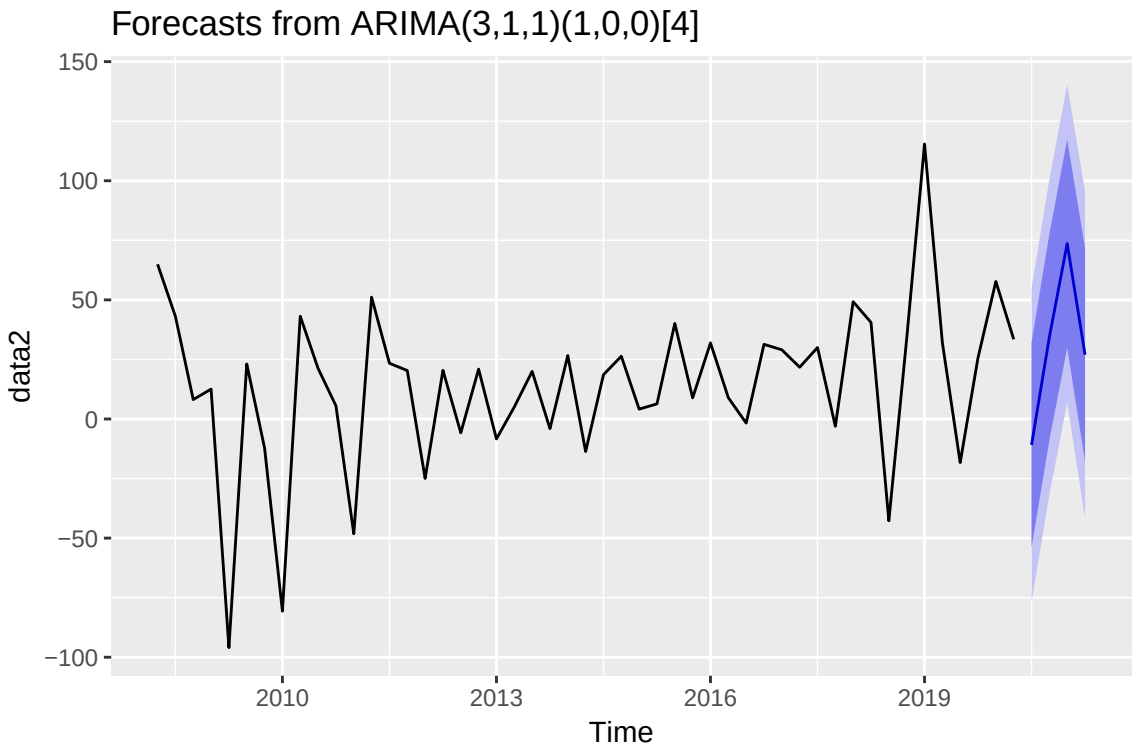


4.4 Prognozavimas

Prognozuosime artimiausiems 4-iems ketvirčiams naudodami komandą `forecast()`.
Būtina atsiminti, kad pateikiamos reikšmės yra atlyginimo augimo skirtumai (kadangi duomenis diferencijavome, kai juos stacionarizavome) ir iš jų atimtos sezoninės reikšmės.

```
prognoze <- forecast(arima, 4)
prognoze
##         Point Forecast      Lo 80     Hi 80      Lo 95     Hi 95
## 2020 Q3      -10.92690 -53.943938  32.09014 -76.715782  54.86198
## 2020 Q4       34.35632  -8.893836  77.60648 -31.789088 100.50173
## 2021 Q1       73.64208  29.986290 117.29788   6.876309 140.40786
## 2021 Q2       26.97061 -17.763809  71.70502 -41.444778  95.38599
```

```
autoplot(forecast(arima,4))
```



Galiausiai reikia pridėti sezonines vertes ir gauti tikruosius prognozinius įverčius.

```
# Kadangi prognozavome paskutinių metų 3,4 ketvirčius ir naujų metų 1 ir 2 ketvirčius
# Reikia sezoniškumo indeksų: Qtr3, Qtr4, Qtr1, Qtr2
# Sudedame prognozuotus įverčius ir pridedame kiekvieno ketvirčio sezoniškumo indeksus
forecast <- as.numeric(prognoze$mean) + as.numeric(diff(data_adityv$seasonal[2:6]))
forecast <- ts(forecast, start=c(2020,3), frequency = 4)
forecast
##           Qtr1      Qtr2      Qtr3      Qtr4
## 2020                     -40.51525  22.31986
## 2021 161.10326 -18.86576
```

5 VAR

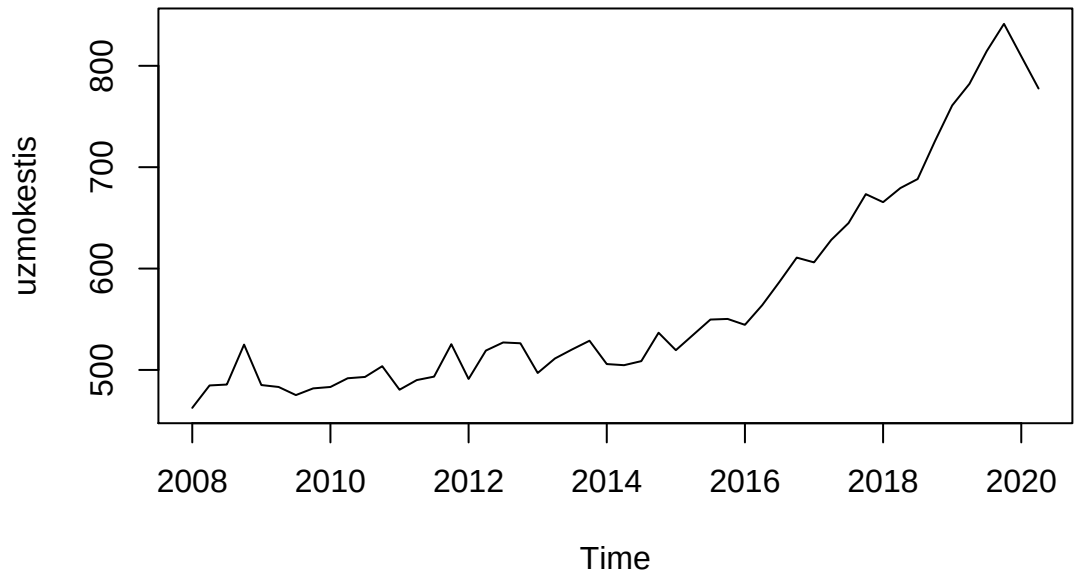
Var modelių sudarymui naudosime paskaitoje pateiktus MMA, TUI, dirbančiųjų skaičiaus ir darbo užmokesčio duomenis. Jie įkelti į Google Drive platformą dėl patogesnio priėjimo.
Įsikeliame duomenis:

```
url <-"https://drive.google.com/uc?export=download&id=1R6DkKwwGC3mcPL8YhRBD1ARnyKdVnsjR"
df <- read.csv(url, header=TRUE, sep=";")
df$Uzmokestis<- as.numeric(sub(",", ".", df$Uzmokestis, fixed=T))
df$Dirbantys<- as.numeric(sub(",", ".", df$Dirbantys, fixed=T))
df$TUI<- as.numeric(sub(",", ".", df$TUI, fixed=T))
df$MMA<- as.numeric(sub(",", ".", df$MMA, fixed=T))
```

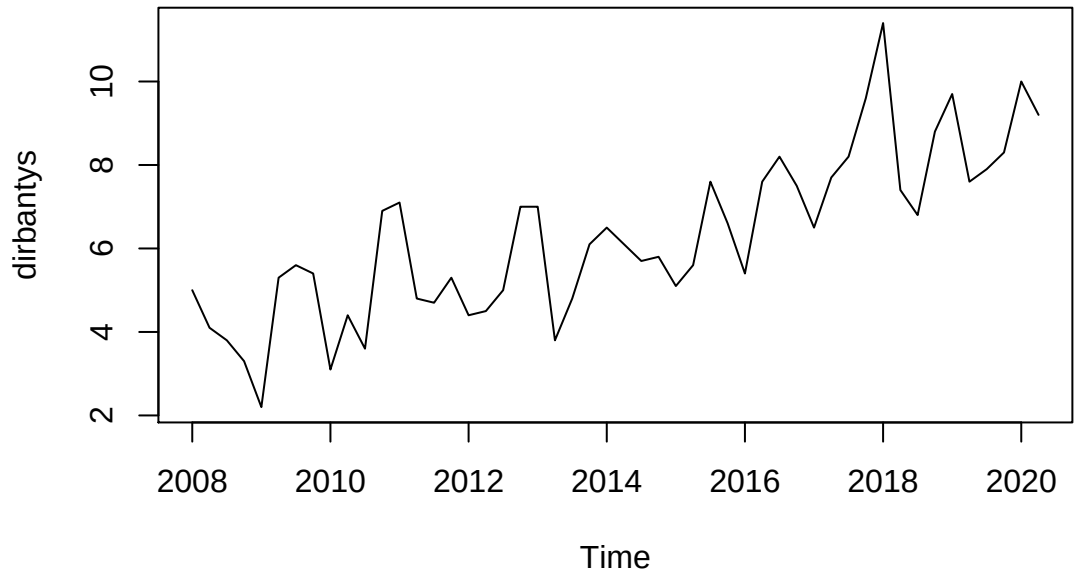
```
uzmokestis <- ts(df$Uzmokestis,
                 start=c(2008,1),
                 end=c(2020,2),
                 frequency = 4)
dirbantys <-ts(df$Dirbantys,
               start=c(2008,1),
               end=c(2020,2),
               frequency = 4)
tui <- ts(df$TUI,
          start=c(2008,1),
          end=c(2020,2),
          frequency = 4)
mma <- ts(df$MMA,
          start=c(2008,1),
          end=c(2020,2),
          frequency = 4)
```

5.1 Pirmoji grafinė analizė

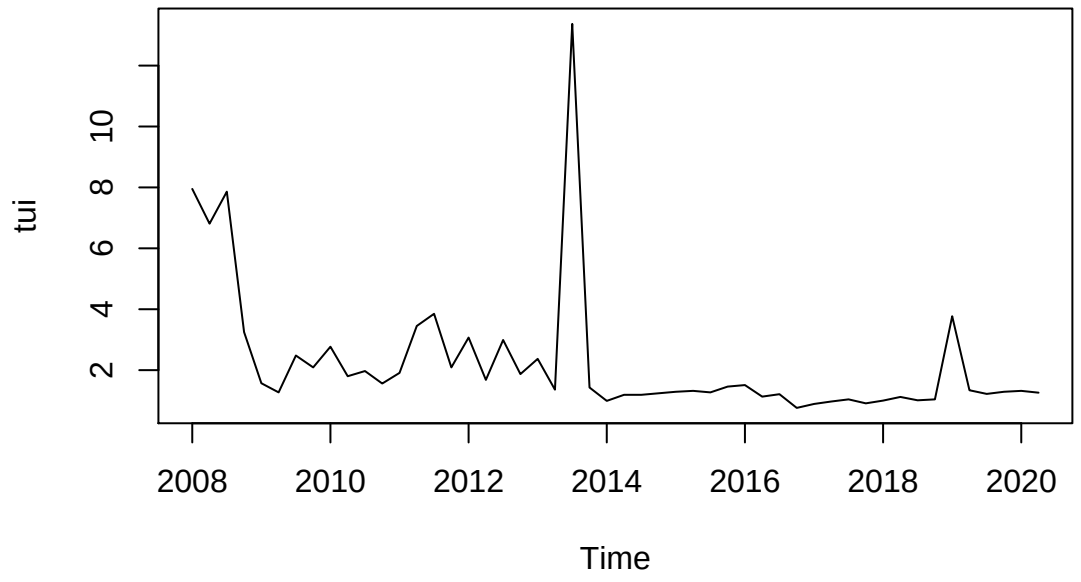
```
plot.ts(uzmokestis)
```



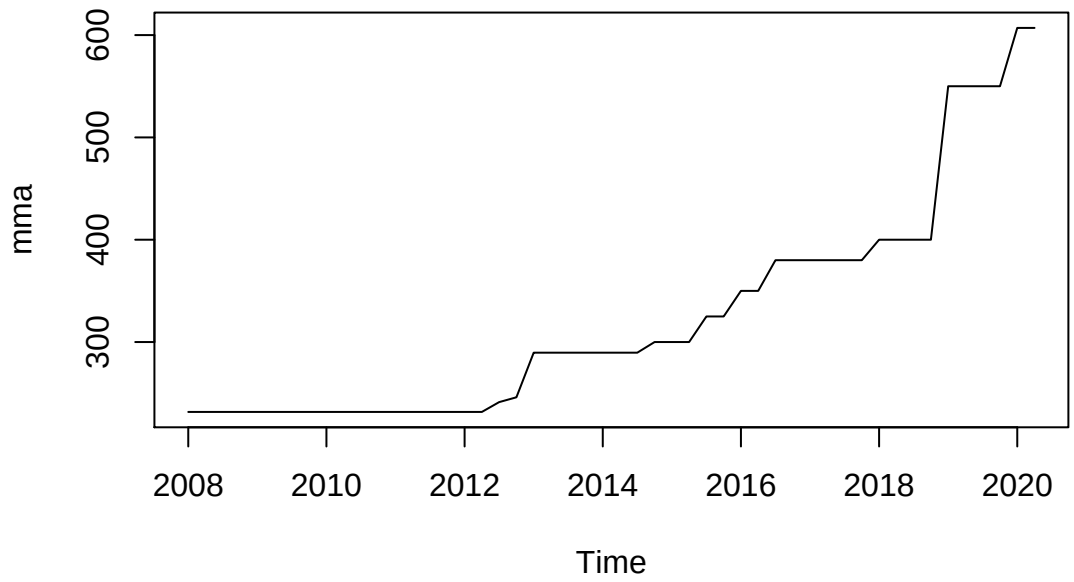
```
plot.ts(dirbantys)
```




```
plot.ts(tui)
```



```r
plot.ts(mma)
```



5.2 Stacionarizavimas

Patikrinkime stacionarumą.

```r
adf.test(uzmokestis, k=0)
##
##  Augmented Dickey-Fuller Test
##
## data:  uzmokestis
## Dickey-Fuller = -1.3023, Lag order = 0, p-value = 0.8541
## alternative hypothesis: stationary
adf.test(dirbantys, k=0)
##
##  Augmented Dickey-Fuller Test
##
## data:  dirbantys
## Dickey-Fuller = -5.6212, Lag order = 0, p-value = 0.01
## alternative hypothesis: stationary
adf.test(mma,k=0)
##
##  Augmented Dickey-Fuller Test
##
```

```
## data:  mma
## Dickey-Fuller = -1.07, Lag order = 0, p-value = 0.9184
## alternative hypothesis: stationary
adf.test(tui, k=0)
##
##  Augmented Dickey-Fuller Test
##
## data:  tui
## Dickey-Fuller = -6.249, Lag order = 0, p-value = 0.01
## alternative hypothesis: stationary
```

Matome, jog ne visos laiko eilutės stacionarios, todėl diferencijuojame laiko eilutes ir kartojame vienetinės šaknies testą.

```
uzmokestis <- diff(uzmokestis)
mma <- diff(mma)
dirbantys <- diff(dirbantys)
tui <- diff(tui)
adf.test(uzmokestis, k=0)
##
##  Augmented Dickey-Fuller Test
##
## data:  uzmokestis
## Dickey-Fuller = -7.2661, Lag order = 0, p-value = 0.01
## alternative hypothesis: stationary
adf.test(dirbantys, k=0)
##
##  Augmented Dickey-Fuller Test
##
## data:  dirbantys
## Dickey-Fuller = -7.7545, Lag order = 0, p-value = 0.01
## alternative hypothesis: stationary
adf.test(mma,k=0)
##
##  Augmented Dickey-Fuller Test
##
## data:  mma
## Dickey-Fuller = -8.2004, Lag order = 0, p-value = 0.01
## alternative hypothesis: stationary
adf.test(tui, k=0)
##
##  Augmented Dickey-Fuller Test
##
## data:  tui
## Dickey-Fuller = -11.408, Lag order = 0, p-value = 0.01
```

```
## alternative hypothesis: stationary
```

5.3 VAR Modelio sudarymas

Laiko eilutės stacionarios, galime pereiti prie VAR sudarymo.
Visų pirma reikia susidaryti duomenų matrica.

```
endogeniniai <- cbind(uzmokestis, dirbantys)
colnames(endogeniniai) <- c("uzmokestis", "dirbantys")
egzogeniniai <- cbind(mma, tui)
colnames(egzogeniniai) <- c("mma", "tui")
```

Dabar `vars` paketo `VARselect()` komanda, remiantis Akaike informacijos kriterijumi, galėsime išsiaiškinti, kelintos eilės VAR modelį sudaryti.

```
if(!require("vars")) install.packages("vars"); library("vars")
```

```
info <- VARselect(endogeniniai, lag.max = 4, exogen = egzogeniniai)
info$selection
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      4      4      1      4
```

Matome, jog komanda siūlo sudaryti 4-os eilės VAR modelį.
Sudarykime modelį.

```
Model1 <- VAR(endogeniniai, p = 4, ic="AIC", exog= egzogeniniai)
summary(Model1)
##
## VAR Estimation Results:
## ========================
## Endogenous variables: uzmokestis, dirbantys
## Deterministic variables: const
## Sample size: 45
## Log Likelihood: -253.591
## Roots of the characteristic polynomial:
## 0.8566 0.8566 0.8349 0.825 0.825 0.7387 0.6803 0.6803
## Call:
## VAR(y = endogeniniai, p = 4, exogen = egzogeniniai, ic = "AIC")
##
##
## Estimation results for equation uzmokestis:
## ===========================================
## uzmokestis = uzmokestis.l1 + dirbantys.l1 + uzmokestis.l2 + dirbantys.l2 + uzmokestis.l3 + dirbant
##
##                Estimate Std. Error t value Pr(>|t|)
```

```
## uzmokestis.l1  0.08643    0.15209   0.568   0.5736
## dirbantys.l1  -3.30926    2.25093  -1.470   0.1507
## uzmokestis.l2  0.02445    0.16714   0.146   0.8845
## dirbantys.l2  -1.00365    2.27811  -0.441   0.6623
## uzmokestis.l3 -0.15320    0.15758  -0.972   0.3378
## dirbantys.l3   1.80657    2.47152   0.731   0.4698
## uzmokestis.l4  0.35620    0.15561   2.289   0.0284 *
## dirbantys.l4   1.54895    2.34085   0.662   0.5126
## const          3.37642    3.58804   0.941   0.3533
## mma            0.15879    0.13357   1.189   0.2428
## tui           -0.53622    1.18561  -0.452   0.6539
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 17.68 on 34 degrees of freedom
## Multiple R-Squared: 0.3247,  Adjusted R-squared: 0.1261
## F-statistic: 1.635 on 10 and 34 DF,  p-value: 0.1386
##
##
## Estimation results for equation dirbantys:
## ==========================================
## dirbantys = uzmokestis.l1 + dirbantys.l1 + uzmokestis.l2 + dirbantys.l2 + uzmokestis.l3 + dirbanty
##
##                Estimate Std. Error t value Pr(>|t|)
## uzmokestis.l1 -0.002020   0.010695  -0.189  0.85132
## dirbantys.l1  -0.539537   0.158289  -3.409  0.00170 **
## uzmokestis.l2 -0.004810   0.011754  -0.409  0.68491
## dirbantys.l2  -0.740033   0.160200  -4.619 5.32e-05 ***
## uzmokestis.l3 -0.003093   0.011081  -0.279  0.78182
## dirbantys.l3  -0.497343   0.173801  -2.862  0.00716 **
## uzmokestis.l4  0.016277   0.010942   1.487  0.14610
## dirbantys.l4  -0.329433   0.164612  -2.001  0.05340 .
## const          0.257797   0.252316   1.022  0.31413
## mma            0.008514   0.009393   0.906  0.37111
## tui            0.001499   0.083373   0.018  0.98576
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 1.243 on 34 degrees of freedom
## Multiple R-Squared: 0.4824,  Adjusted R-squared: 0.3301
## F-statistic: 3.168 on 10 and 34 DF,  p-value: 0.005721
##
##
```

```
##
## Covariance matrix of residuals:
##            uzmokestis dirbantys
## uzmokestis    312.408    -3.362
## dirbantys      -3.362     1.545
##
## Correlation matrix of residuals:
##            uzmokestis dirbantys
## uzmokestis      1.000    -0.153
## dirbantys      -0.153     1.000
```

Patikrinkime apskaičiuoto modelio AR charakteringo polinomo atvirkštines šaknis ir atlikime Jarque Bera testą paklaidų normalumui nustatyti.

```
roots(Model1, modulus = TRUE)
## [1] 0.8566473 0.8566473 0.8349304 0.8249715 0.8249715 0.7386961 0.6802874
## [8] 0.6802874
normality.test(Model1, multivariate.only = TRUE)
## $JB
##
##  JB-Test (multivariate)
##
## data:  Residuals of VAR object Model1
## Chi-squared = 9.0878, df = 4, p-value = 0.05894
##
##
## $Skewness
##
##  Skewness only (multivariate)
##
## data:  Residuals of VAR object Model1
## Chi-squared = 5.807, df = 2, p-value = 0.05483
##
##
## $Kurtosis
##
##  Kurtosis only (multivariate)
##
## data:  Residuals of VAR object Model1
## Chi-squared = 3.2808, df = 2, p-value = 0.1939
```

Matome, jog nei viena polinomo atvirkštinė šaknis nėra didesnė už 1-ą, o Jarque-Bera testo p reikšmė = 0.05894 vos vos didesnė už 0.05, kas leidžia teigti (su 90proc. pasitikėjimo lygiu), jog paklaidos yra normalios.

5.4 Prognozė

Prognozuosime 4-iems ateinantiems laikotarpiams (4-iems ketvirčiams). Turime sukurti egzogeninių reikšmių prognozuojamiems laikotarpiams matricą. Jei nežinome, kokios kintamųjų reikšmės bus ateinančiuose laikotarpiuose, tuomet įrašome 0. Ir galiausiai atliekame prognozę.

```
# Kuriame egzogeninių kintamųjų matricą ateičiai
mma2<-c(0,0,0,0)
tui2<-c(0,0,0,0)
egzogeniniai2 <- cbind(mma2, tui2)
colnames(egzogeniniai2) <- c("mma", "tui")
# Prognozuojame
predict(Model1, n.ahead = 4, ci = 0.95, dumvar = egzogeniniai2)
## $uzmokestis
##           fcst     lower    upper       CI
## [1,]  9.358263 -25.28424 44.00076 34.64250
## [2,] 23.066001 -12.73129 58.86329 35.79729
## [3,] -1.699268 -37.53499 34.13646 35.83573
## [4,] -9.185270 -46.63958 28.26904 37.45431
##
## $dirbantys
##            fcst     lower    upper       CI
## [1,] -0.2057095 -2.641819 2.230400 2.436109
## [2,]  0.6544504 -2.109414 3.418315 2.763864
## [3,] -0.6193237 -3.581312 2.342665 2.961988
## [4,] -0.1805847 -3.166680 2.805510 2.986095
```

5.5 Granger priežastingumo testai

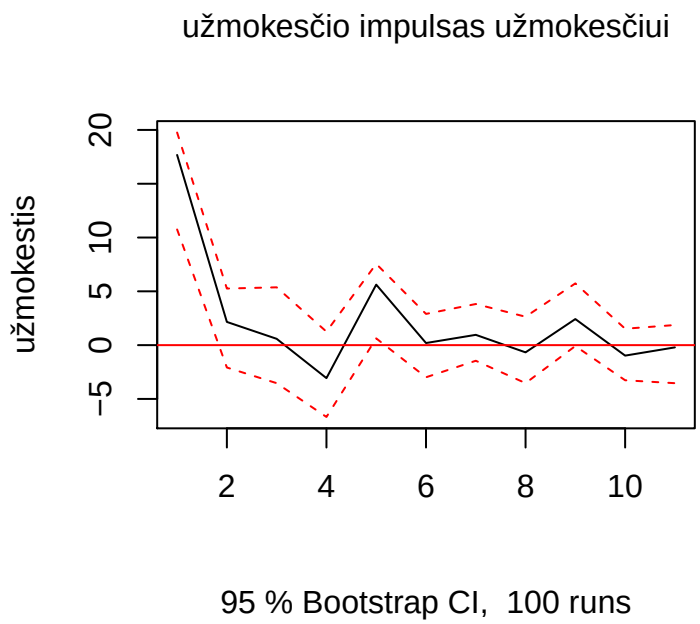
Atliekame Granger priežastingumo testus.

```
Grangeruzmokestis<- causality(Model1, cause = "uzmokestis")
Grangerdirbantys <- causality(Model1, cause = "dirbantys")
Grangeruzmokestis
## $Granger
##
##  Granger causality H0: uzmokestis do not Granger-cause dirbantys
##
## data:  VAR object Model1
## F-Test = 0.63004, df1 = 4, df2 = 68, p-value = 0.6427
##
##
## $Instant
##
##  H0: No instantaneous causality between: uzmokestis and dirbantys
```

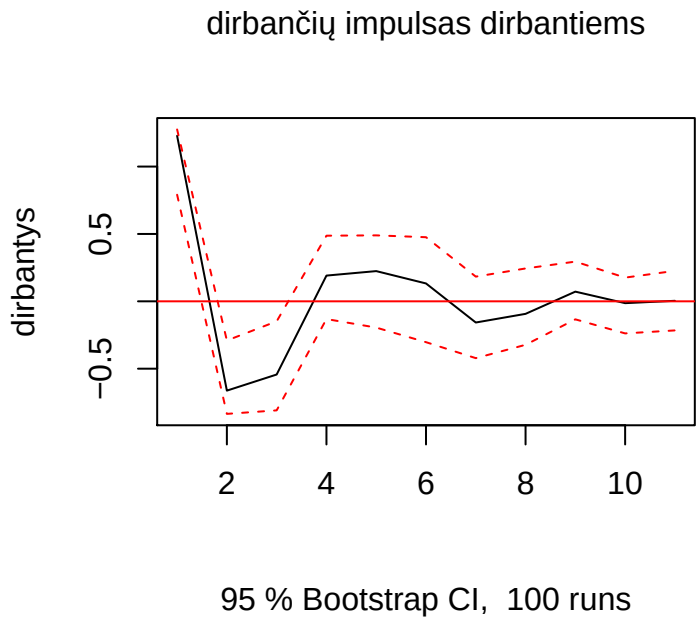
```
##
## data:  VAR object Model1
## Chi-squared = 1.0297, df = 1, p-value = 0.3102
Grangerdirbantys
## $Granger
##
##  Granger causality H0: dirbantys do not Granger-cause uzmokestis
##
## data:  VAR object Model1
## F-Test = 1.7399, df1 = 4, df2 = 68, p-value = 0.1513
##
##
## $Instant
##
##  H0: No instantaneous causality between: dirbantys and uzmokestis
##
## data:  VAR object Model1
## Chi-squared = 1.0297, df = 1, p-value = 0.3102
```

5.6 Reakcija į impulsus

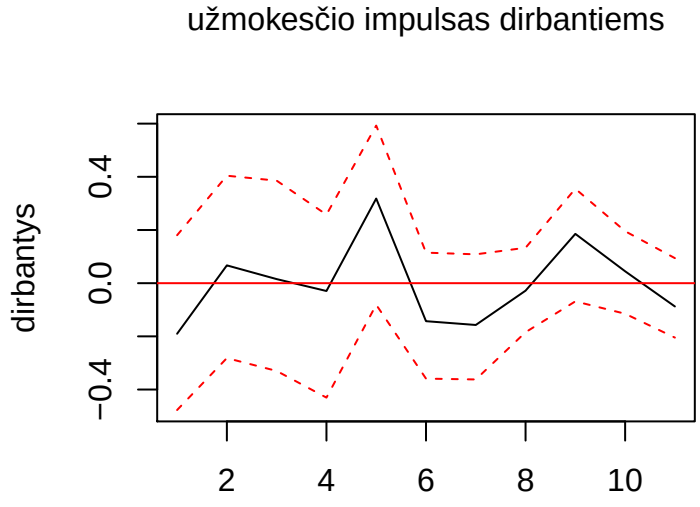
```
irf1 <- irf(Model1, impulse = "uzmokestis", response = "uzmokestis", n.ahead = 10, boot = TRUE)
plot(irf1, ylab = "užmokestis", main = "užmokesčio impulsas užmokesčiui")
```



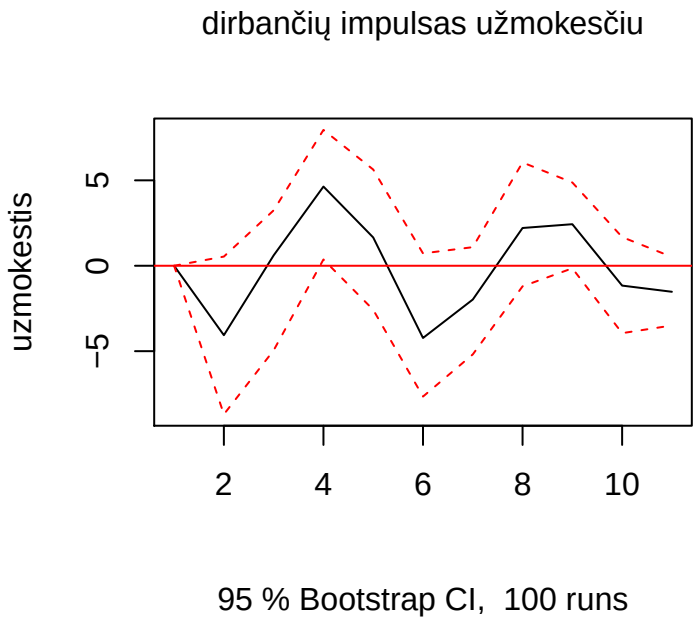
```
irf2 <- irf(Model1, impulse = "dirbantys", response = "dirbantys", n.ahead = 10, boot = TRUE)
plot(irf2, ylab = "dirbantys", main = "dirbančių impulsas dirbantiems")
```




```
irf3 <- irf(Model1, impulse = "uzmokestis", response = "dirbantys", n.ahead = 10, boot = TRUE)
plot(irf3, ylab = "dirbantys", main = "užmokesčio impulsas dirbantiems")
```



```
irf4 <- irf(Model1, impulse = "dirbantys", response = "uzmokestis", n.ahead = 10, boot = TRUE)
plot(irf4, ylab = "uzmokestis", main = "dirbančių impulsas užmokesčiu")
```



6 Panelinių duomenų modeliai

Panelinių duomenų modeliui nagrinėti pasitelksime Lietuvos statistikos departamento duomenis: nusikalstamumo lygis, išsilavinimo lygis ir BVP vienam gyventojui.
Modelio tikslas - nustatyti ar nusikalstumo lygiui Lietuvos regionuose turį įtakos išsilavinimo, nedarbo lygis ir BVP vienam gyventojui. Jei įtaka yra, kokia ji.
Duomenys yra 2005 - 2019metų iš 10 Lietuvos regionų.
Importuokime duomenis, kurie įkelti į Google Drive platformą dėl patogesnio prieinamumo iš skirtingų kompiuterių. Tuo pačiu persiskaičiuosime išsilavinusių žmonių skaičių iš absoliučių reikšmių į procentus nuo bendro išsilavinusių skaičiaus

```
data <- read.delim("https://drive.google.com/uc?export=download&id=189Gq3W33BjoFkeDmlEcOCLjR4wAtC6-f"
data <- dplyr::filter(data, Administracine.teritorija !="Lietuvos Respublika",
                      Administracine.teritorija !="Sostin?s regionas",
                      Administracine.teritorija != "Vidurio ir vakar? Lietuvos regionas") %>%
        mutate(Vidutinis = Vidutinis/Viso.issilavinusiu* 100,
               Aukstas= Aukstas/Viso.issilavinusiu* 100,
               Zemas = Zemas/Viso.issilavinusiu* 100) %>%
        rename( "teritorija" =Administracine.teritorija)
```

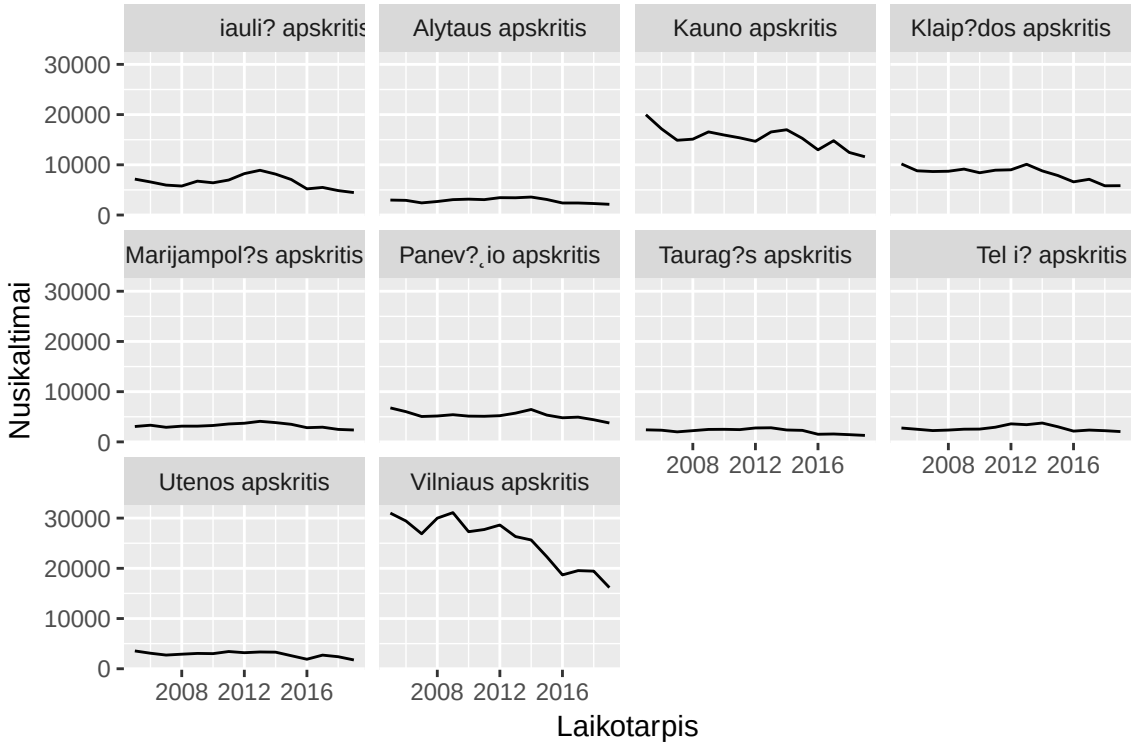
Paverskime duomenis į panelinių duomenų formatą. Tam prireiks `plm` paketo.

```
if(!require("plm")) install.packages("plm"); library("plm")
## Loading required package: plm
##
## Attaching package: 'plm'
## The following object is masked from 'package:timeSeries':
##
##     lag
## The following objects are masked from 'package:dplyr':
##
##     between, lag, lead
# Paverčiame į panelinių duomenų formatą
data.p <- pdata.frame(data,index=c("teritorija", "Laikotarpis"))
# Verčiame praleistas "NA" reikšmes į 0
data.p[is.na(data.p)] <- 0
# Suformatuojame, kad laikotarpis būtų datos formato. Prireiks paketo "lubridate"
if(!require("lubridate")) install.packages("lubridate"); library("lubridate")
## Loading required package: lubridate
##
## Attaching package: 'lubridate'
## The following object is masked from 'package:base':
##
##     date
data.p$Laikotarpis <- year(as.Date(data.p$Laikotarpis, format = "%Y"))
```

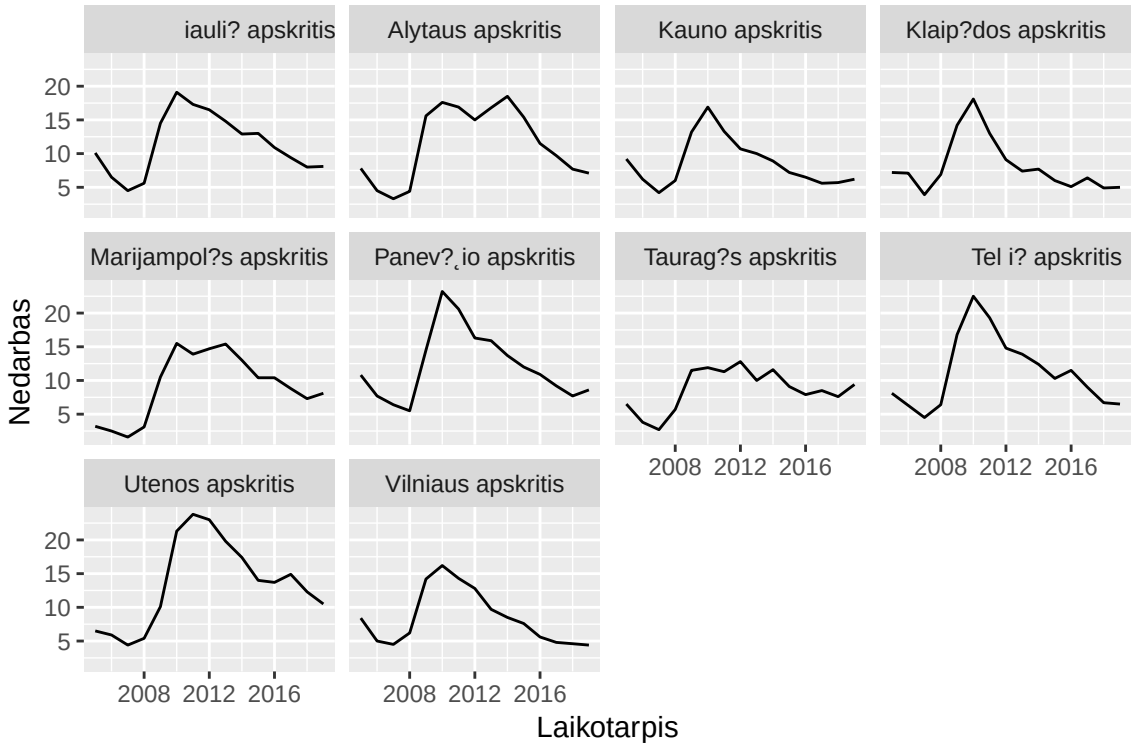
6.1 Grafinė analizė

Pažiūrėkime, kaip elgiasi duomenys. Kokios tendencijos bei ar duomenys yra stacionarūs. Duomenų atvaizdavimui naudosime `ggplot()` paketą.

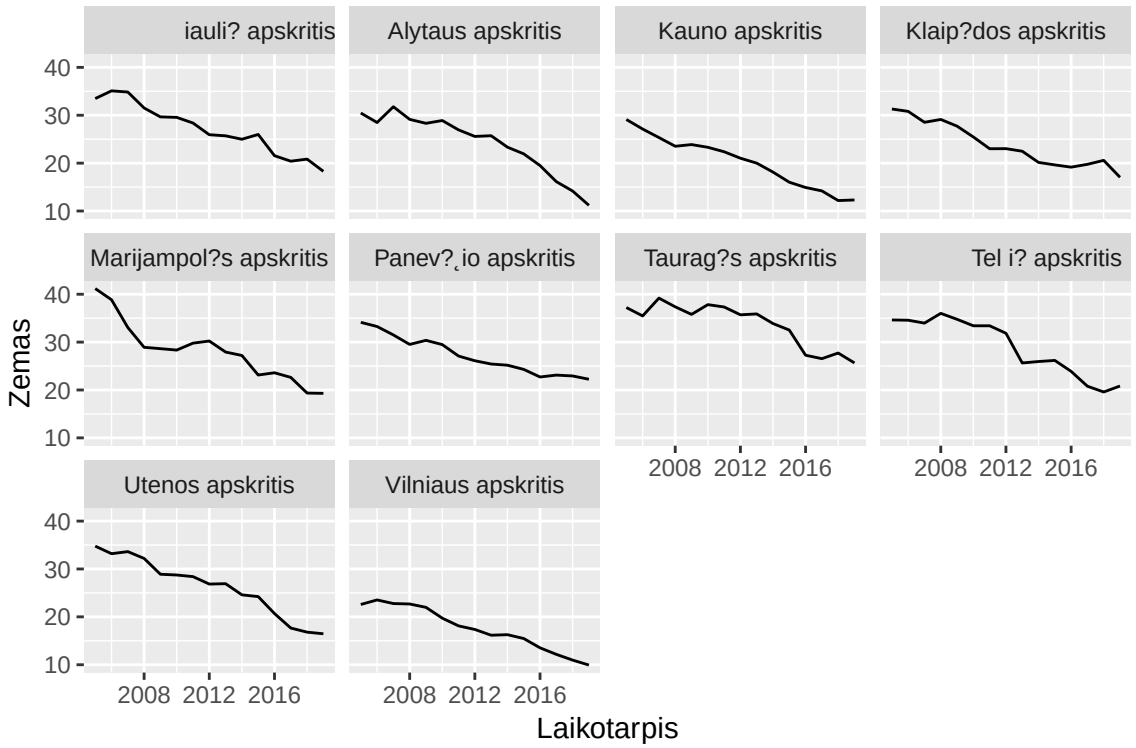
```
ggplot(data.p, aes(Laikotarpis, Nusikaltimai, group=1)) +
        geom_line()+
        facet_wrap(~ teritorija)
```



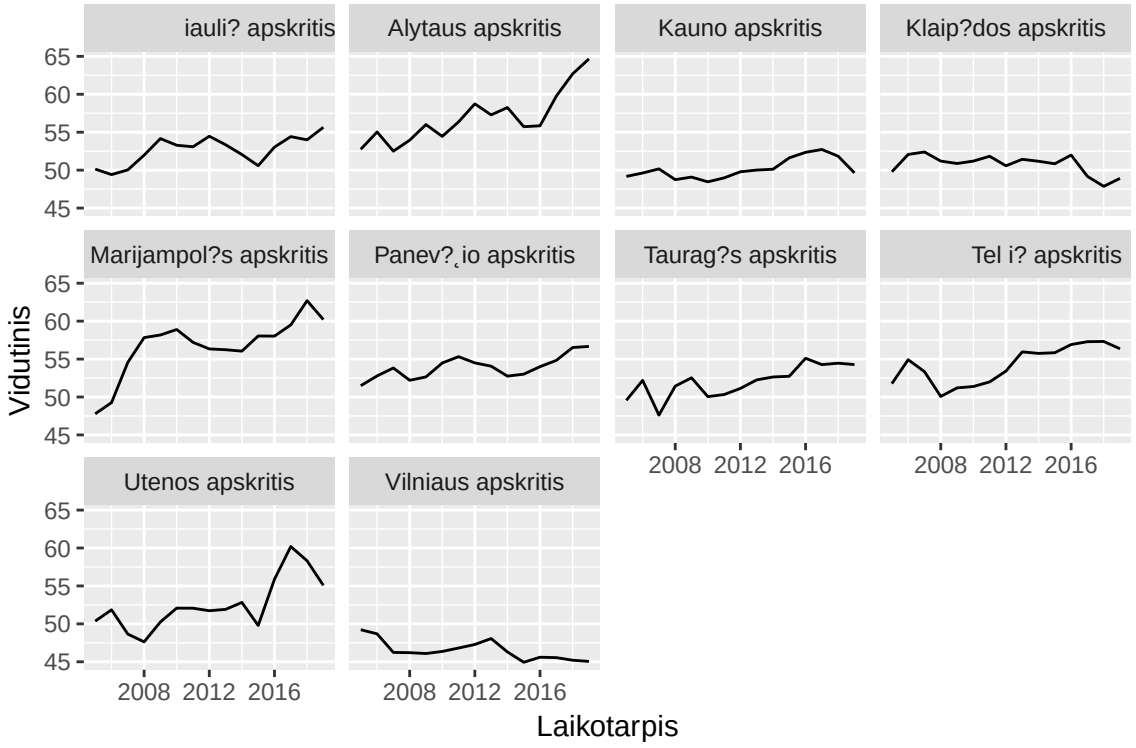

```
ggplot(data.p, aes(Laikotarpis, Nedarbas, group=1)) +
        geom_line()+
        facet_wrap(~ teritorija)
```



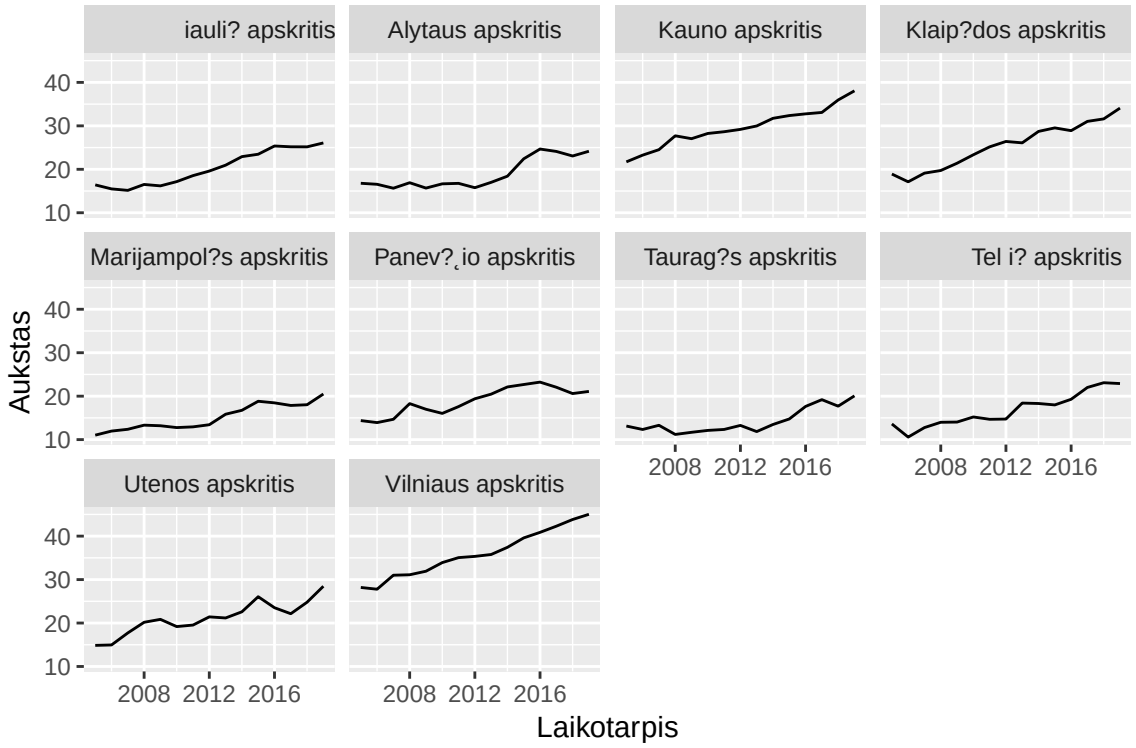
```
ggplot(data.p, aes(Laikotarpis, Zemas, group=1)) +
        geom_line()+
        facet_wrap(~ teritorija)
```




```
ggplot(data.p, aes(Laikotarpis, Vidutinis, group=1)) +
        geom_line()+
        facet_wrap(~ teritorija)
```




```
ggplot(data.p, aes(Laikotarpis, Aukstas, group=1)) +
        geom_line()+
        facet_wrap(~ teritorija)
```



6.2 Stacionarumo tikrinimas [1]

Matome, kad duomenys, išskyrus nedarbą, nėra stacionarūs, kadangi p reikšmės > 0.05.

```
adf.test(data.p$Nedarbas)
##
##  Augmented Dickey-Fuller Test
##
## data:  data.p$Nedarbas
## Dickey-Fuller = -5.7376, Lag order = 5, p-value = 0.01
## alternative hypothesis: stationary
adf.test(data.p$Nusikaltimai)
##
##  Augmented Dickey-Fuller Test
##
## data:  data.p$Nusikaltimai
## Dickey-Fuller = -2.3303, Lag order = 5, p-value = 0.4388
## alternative hypothesis: stationary
adf.test(data.p$Zemas)
##
##  Augmented Dickey-Fuller Test
##
## data:  data.p$Zemas
## Dickey-Fuller = -3.0042, Lag order = 5, p-value = 0.158
## alternative hypothesis: stationary
adf.test(data.p$Aukstas)
##
##  Augmented Dickey-Fuller Test
##
## data:  data.p$Aukstas
## Dickey-Fuller = -1.5047, Lag order = 5, p-value = 0.7828
## alternative hypothesis: stationary
adf.test(data.p$Vidutinis)
##
##  Augmented Dickey-Fuller Test
##
## data:  data.p$Vidutinis
## Dickey-Fuller = -3.2154, Lag order = 5, p-value = 0.08802
## alternative hypothesis: stationary
adf.test(data.p$BVP)
##
##  Augmented Dickey-Fuller Test
##
## data:  data.p$BVP
## Dickey-Fuller = -1.6356, Lag order = 5, p-value = 0.7282
## alternative hypothesis: stationary
```

6.3 Duomenų stacionarizavimas.

Diferencijuojame laiko eilutes, tam, kad gauti pokyčius. Nusikaltimų duomenis logaritmuojame ir differencijuojame, taip gaudami procentinius augimo pokyčius.

```
data.p$Nusikaltimai <- log(data.p$Nusikaltimai)
data.p$Nusikaltimai <- diff(data.p$Nusikaltimai)
data.p$Zemas <- diff(data.p$Zemas)
data.p$Vidutinis <- diff(data.p$Vidutinis)
data.p$Aukstas <- diff(data.p$Aukstas)
data.p$BVP <- diff(data.p$BVP)
data.p[is.na(data.p)] <- 0
```

6.4 Stacionarumo tikrinimas [2]

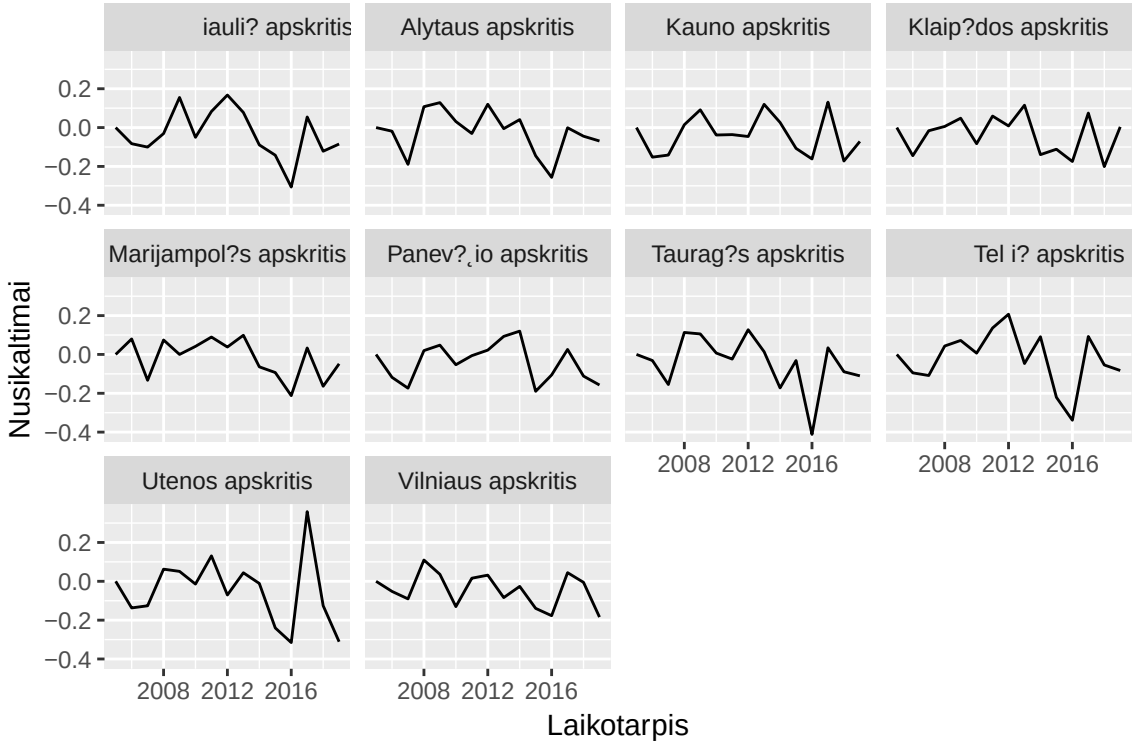
Matome, jog duomenys vienetinės šaknies nebeturi - tapo stacionarūs. p reikšmės < 0.05.

```
adf.test(data.p$Nedarbas)
##
##  Augmented Dickey-Fuller Test
##
## data:  data.p$Nedarbas
## Dickey-Fuller = -5.7376, Lag order = 5, p-value = 0.01
## alternative hypothesis: stationary
adf.test(data.p$Nusikaltimai)
##
##  Augmented Dickey-Fuller Test
##
## data:  data.p$Nusikaltimai
## Dickey-Fuller = -5.4403, Lag order = 5, p-value = 0.01
## alternative hypothesis: stationary
adf.test(data.p$Zemas)
##
##  Augmented Dickey-Fuller Test
##
## data:  data.p$Zemas
## Dickey-Fuller = -6.9539, Lag order = 5, p-value = 0.01
## alternative hypothesis: stationary
adf.test(data.p$Aukstas)
##
##  Augmented Dickey-Fuller Test
##
## data:  data.p$Aukstas
## Dickey-Fuller = -4.9838, Lag order = 5, p-value = 0.01
## alternative hypothesis: stationary
```

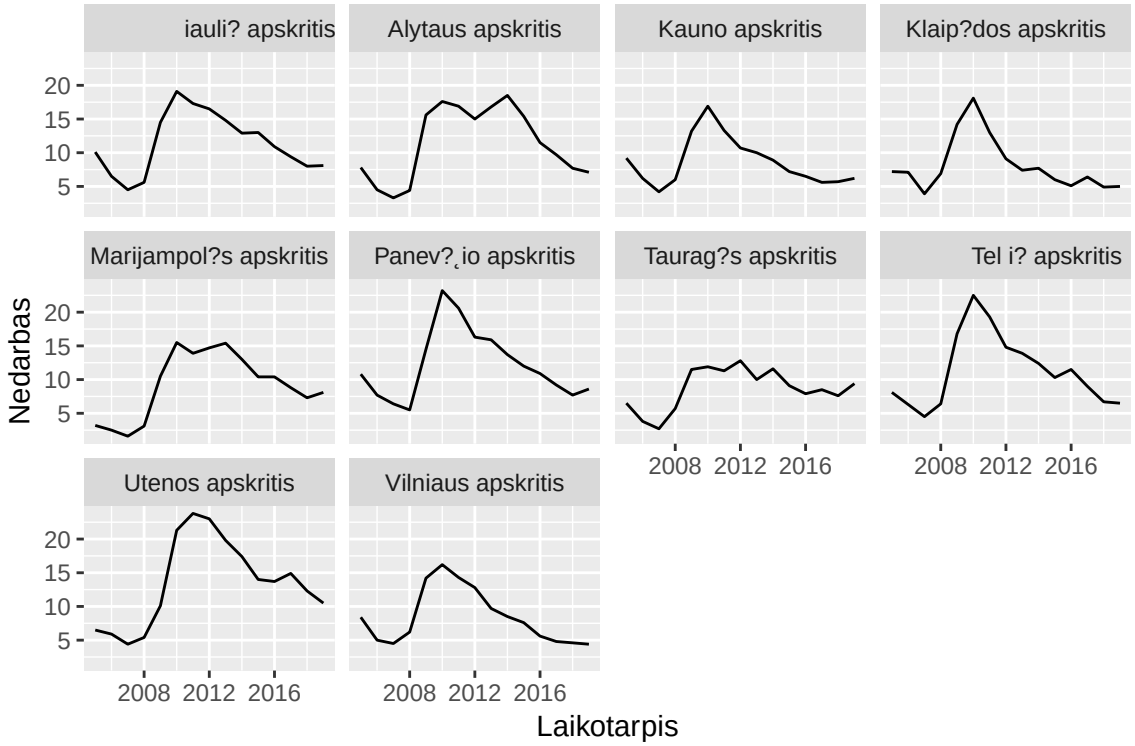
```r
adf.test(data.p$Vidutinis)
##
##  Augmented Dickey-Fuller Test
##
## data:  data.p$Vidutinis
## Dickey-Fuller = -7.7932, Lag order = 5, p-value = 0.01
## alternative hypothesis: stationary
adf.test(data.p$BVP)
##
##  Augmented Dickey-Fuller Test
##
## data:  data.p$BVP
## Dickey-Fuller = -5.4848, Lag order = 5, p-value = 0.01
## alternative hypothesis: stationary
```

6.5 Grafinė analizė [2]

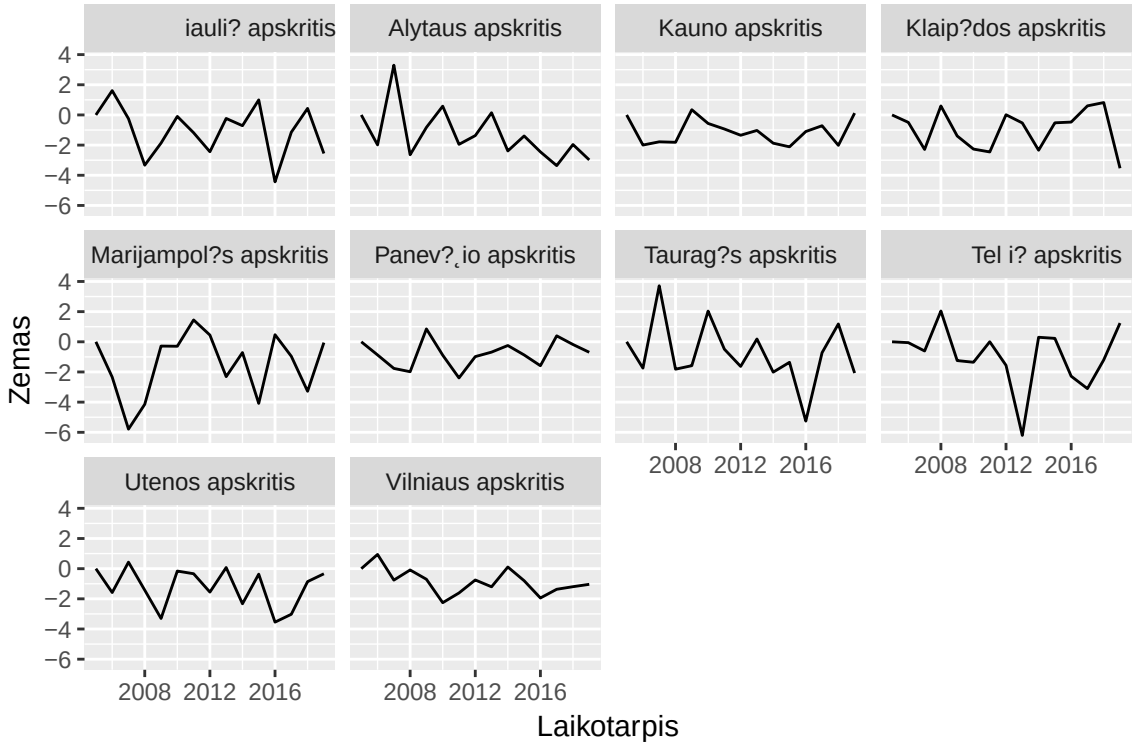
```r
ggplot(data.p, aes(Laikotarpis, Nusikaltimai, group=1)) +
        geom_line()+
        facet_wrap(~ teritorija)
```



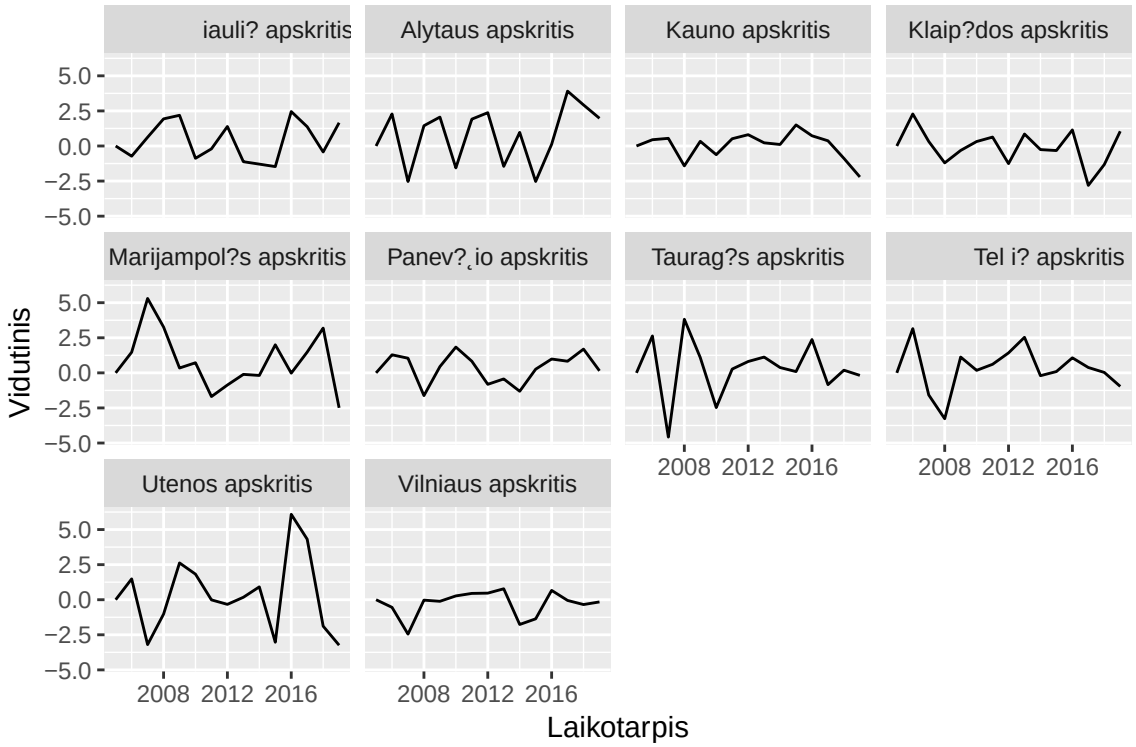
```
ggplot(data.p, aes(Laikotarpis, Nedarbas, group=1)) +
        geom_line()+
        facet_wrap(~ teritorija)
```



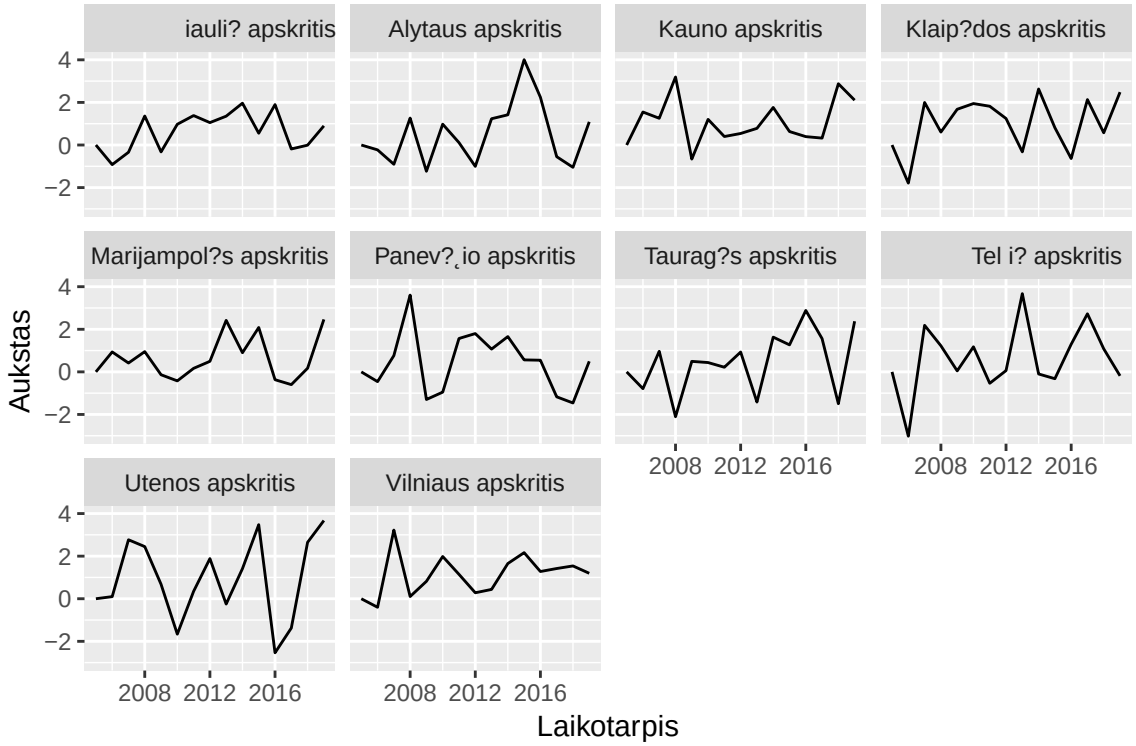

```
ggplot(data.p, aes(Laikotarpis, Zemas, group=1)) +
        geom_line()+
        facet_wrap(~ teritorija)
```




```
ggplot(data.p, aes(Laikotarpis, Vidutinis, group=1)) +
        geom_line()+
        facet_wrap(~ teritorija)
```



```
ggplot(data.p, aes(Laikotarpis, Aukstas, group=1)) +
        geom_line()+
        facet_wrap(~ teritorija)
```



6.6 Pastovių konstantų modelis.

Sudarėme pastovių konstantų modelį su išraiška:

$$log(Nusikaltimai) = \alpha + \beta 1 Nedarbas + \beta 2 Zemas + \beta 3 Vidutinis + \beta 4 Aukstas + \beta 5 BVP$$

```
ols<- lm(Nusikaltimai ~ Nedarbas +  Zemas + Vidutinis + Aukstas + BVP, data=data.p)
summary(ols)
##
## Call:
## lm(formula = Nusikaltimai ~ Nedarbas + Zemas + Vidutinis + Aukstas +
##     BVP, data = data.p)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.34222 -0.07590  0.01479  0.08008  0.31794
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.090486   0.027317  -3.312  0.00117 **
```

```
## Nedarbas     0.006456   0.002044   3.158  0.00193 **
## Zemas       -0.235175   0.187632  -1.253  0.21210
## Vidutinis   -0.237122   0.187678  -1.263  0.20847
## Aukstas     -0.251483   0.186939  -1.345  0.18065
## BVP          0.004091   0.006945   0.589  0.55669
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1115 on 144 degrees of freedom
## Multiple R-squared:  0.1148, Adjusted R-squared:  0.08402
## F-statistic: 3.734 on 5 and 144 DF,  p-value: 0.0033
```

Matome, kad BVP komponentė nėra statistiškai reikšminga, teks jos atsisakyti. Taip pat panaikinome Aukšto išsilavinimo žmonių dalį tam, kad galėtume vertinti kitų kintamųjų poveikį bazinio kintamojo atžvilgiu. Matome, jog visi kintamieji yra statistiškai reikšmingi. Sudarome naują modelį:

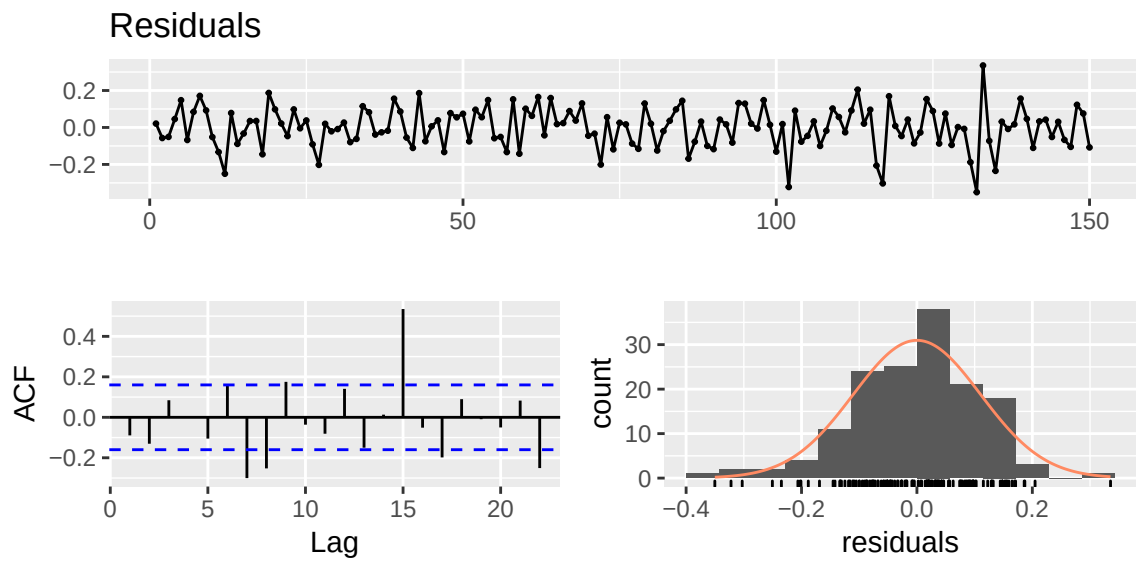
$$log(Nusikaltimai) = \alpha + \beta 1 Nedarbas + \beta 2 Zemas + \beta 3 Vidutinis$$

```
ols<- lm(Nusikaltimai ~ Nedarbas +  Zemas + Vidutinis, data=data.p)
summary(ols)
##
## Call:
## lm(formula = Nusikaltimai ~ Nedarbas + Zemas + Vidutinis, data = data.p)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.35025 -0.07420  0.01686  0.07809  0.33554
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.082158   0.022287  -3.686  0.00032 ***
## Nedarbas     0.006054   0.001855   3.264  0.00137 **
## Zemas        0.017704   0.008050   2.199  0.02943 *
## Vidutinis    0.015913   0.007345   2.167  0.03188 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1116 on 146 degrees of freedom
## Multiple R-squared:  0.102,  Adjusted R-squared:  0.08358
## F-statistic:  5.53 on 3 and 146 DF,  p-value: 0.001266
```

Turime patikrinti ar nėra autokoreliacijos naudodami `durbinWatsonTest()` komandą iš paketo `car`:

```r
if(!require("car")) install.packages("car"); library("car")
```

```r
durbinWatsonTest(ols)
##  lag Autocorrelation D-W Statistic p-value
##    1     -0.08930997      2.171971   0.302
##  Alternative hypothesis: rho != 0
checkresiduals(ols$residuals)
```



Matome, jog Durbin-Watson testas nerodo reikšmingos autokoreliacijos (p>0.05, neatmetam H0). Paklaidų grafike stebime autokoreliaciją 7-tame lage, tačiau toks lagas yra per daug tolimas, kad būtų reikšmingas, kadangi duomenys yra metiniai. Paklaidos pasiskirsčiusios normaliuoju skirstiniu.
Galutinis modelis

$$log(Nusikaltimai) = -0.082605 + 0.006049 * Nedarbas + 0.017063 * Zemas + 0.015339 * Vidutinis$$

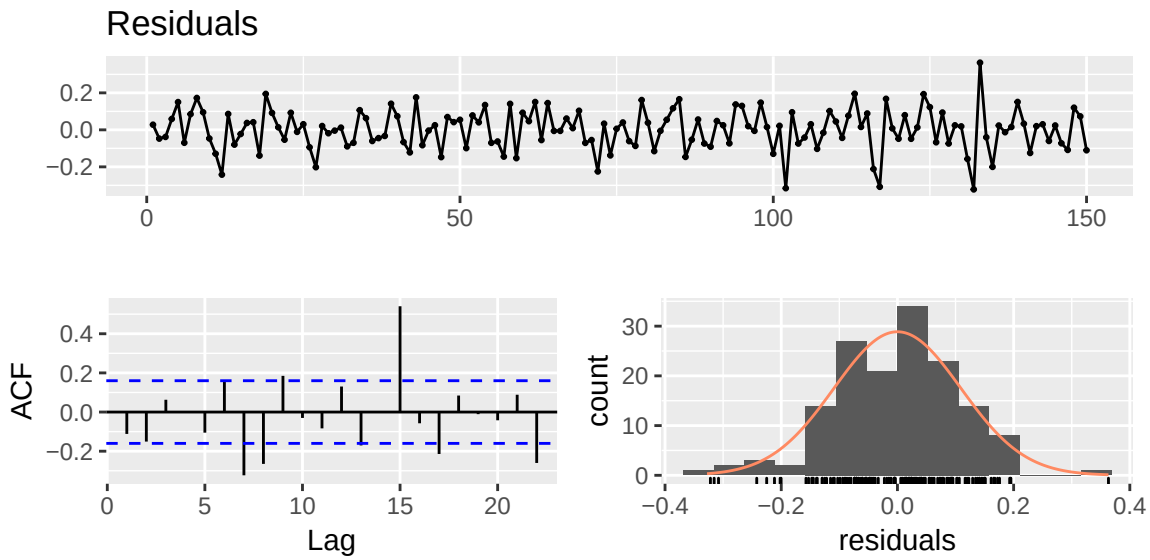
Interpretacija - Koeficientas alpha=-0.082605 rodo, koks būtų nusikalstamumo augimo procentinis pokytis, jei jo neveiktų nei nedarbo lygis, nei žemo ar vidutinio išsilavinimo žmonių skaičius. Koeficientas prie Nedarbo (*0.006049*) rodo, jog nedarbo lygiui išaugus 1proc., nusikalstamumo lygio procentinis pokytis padidėtų 0.6049% (100% * Beta1). Žemo išsilavinimo (*Beta2 = 0.017063*) žmonių padidėjimas, kai aukšto išsilavinimo žmonių skaičius išlieka nepakitęs, iššauktų 1.7063% nusikalstamumo augimo padidėjimą. Tuo tarpu Vidutinio išsilavinimo (*Beta3 = 0.015339*)žmonių skaičiaus padidėjimas 1% nusikalstamumo augimą paspartintų 1.5339%.

6.7 Fiksuotų efektų modelis

$$log(Nusikaltimai) = \alpha 1 + \beta 1 Nedarbas + \alpha 2 + \beta 2 Zemas + \alpha 3 + \beta 3 Vidutinis$$

```r
fixedeff <- plm(data.p$Nusikaltimai~data.p$Nedarbas+data.p$Zemas+data.p$Vidutinis,data=data.p,model="
summary(fixedeff)
## Oneway (individual) effect Within Model
```

```
##
## Call:
## plm(formula = data.p$Nusikaltimai ~ data.p$Nedarbas + data.p$Zemas +
##     data.p$Vidutinis, data = data.p, model = "within")
##
## Balanced Panel: n = 10, T = 15, N = 150
##
## Residuals:
##      Min.   1st Qu.    Median   3rd Qu.      Max.
## -0.321909 -0.069829  0.013526  0.073434  0.363236
##
## Coefficients:
##                    Estimate Std. Error t-value  Pr(>|t|)
## data.p$Nedarbas  0.0071165  0.0020337  3.4992 0.0006298 ***
## data.p$Zemas     0.0185493  0.0083250  2.2281 0.0275016 *
## data.p$Vidutinis 0.0165191  0.0076643  2.1553 0.0328840 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Total Sum of Squares:    2.009
## Residual Sum of Squares: 1.7782
## R-Squared:      0.11486
## Adj. R-Squared: 0.037333
## F-statistic: 5.9261 on 3 and 137 DF, p-value: 0.00078582
## Fiksuotų efektų konstantos kiekvienam regionui
fixef(fixedeff)
##      \212iauli? apskritis      Alytaus apskritis        Kauno apskritis
##               -0.099841              -0.092881              -0.077506
##    Klaip?dos apskritis Marijampol?s apskritis    Panev?\236io apskritis
##               -0.076246              -0.069495              -0.116892
##     Taurag?s apskritis       Tel\232i? apskritis       Utenos apskritis
##               -0.094255              -0.087945              -0.125754
##     Vilniaus apskritis
##               -0.083297
checkresiduals(fixedeff$residuals)
```



Visos komponentės statistiškai reikšmingos, atlikę f testą, patikrinkime, kuris modelis - pastovių konstantų ar fiksuotų efektų - yra geresnis.
Kadangi p>0.05, H0 neatmetame. Pastovios konstantos modelis yra geresnis.

```
pFtest(fixedeff,ols)
##
##  F test for individual effects
##
## data:  data.p$Nusikaltimai ~ data.p$Nedarbas + data.p$Zemas + data.p$Vidutinis
## F = 0.33372, df1 = 9, df2 = 137, p-value = 0.9624
## alternative hypothesis: significant effects
```

6.8 Atsitiktinių efektų modelis

$$log(Nusikaltimai) = \alpha + \beta 1 Nedarbas + \beta 2 Zemas + \beta 3 Vidutinis + (u + v)$$

```
randomeff <- plm(data.p$Nusikaltimai~data.p$Nedarbas+data.p$Zemas+data.p$Vidutinis,data=data.p,model=
summary(randomeff)
## Oneway (individual) effect Random Effect Model
##    (Swamy-Arora's transformation)
##
## Call:
## plm(formula = data.p$Nusikaltimai ~ data.p$Nedarbas + data.p$Zemas +
##     data.p$Vidutinis, data = data.p, model = "random")
##
## Balanced Panel: n = 10, T = 15, N = 150
##
## Effects:
##                    var std.dev share
```

```
## idiosyncratic 0.01298 0.11393     1
## individual    0.00000 0.00000     0
## theta: 0
##
## Residuals:
##      Min.   1st Qu.    Median   3rd Qu.      Max.
## -0.350251 -0.074199  0.016863  0.078086  0.335536
##
## Coefficients:
##                    Estimate Std. Error z-value  Pr(>|z|)
## (Intercept)      -0.0821583  0.0222869 -3.6864 0.0002275 ***
## data.p$Nedarbas   0.0060536  0.0018549  3.2636 0.0010999 **
## data.p$Zemas      0.0177044  0.0080503  2.1992 0.0278614 *
## data.p$Vidutinis  0.0159133  0.0073445  2.1667 0.0302583 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Total Sum of Squares:    2.0237
## Residual Sum of Squares: 1.8172
## R-Squared:      0.10203
## Adj. R-Squared: 0.083583
## Chisq: 16.5897 on 3 DF, p-value: 0.00085822
```

Visas komponentes ir vėl gauname statistiškai reikšmingas. Pasinaudodami Hausman testu tikriname, kuris modelis - atsitiktinių efektų ar fiksuotų efektų - yra geresnis. Gauname p>0.05, todėl H0 neatmetame. Tai reiškia, kad tinkamesnis yra atsitiktinių efektų modelis.

```
phtest(fixedeff,randomeff) ##Neatmetam h0, todel atsitiktiniu dydziu geresnis.
##
##  Hausman Test
##
## data:  data.p$Nusikaltimai ~ data.p$Nedarbas + data.p$Zemas + data.p$Vidutinis
## chisq = 1.7386, df = 3, p-value = 0.6284
## alternative hypothesis: one model is inconsistent
```

6.9 Galutinis modelis

Palyginus atsitiktinių efektų determinacijos koeficientą (0.1) ir pastovios konstantos modelio determinacijos koeficientą (0.1) matome, kad jie yra vienodi. Tai reiškia, kad abu modeliai yra geresni už fiksuotų efektų modelį ir abu paaiškina vienodą dalį duomenų pokyčių. Paprastumo dėlei, kaip galutinį modelį renkamės patobulintą pastovių konstantų modelį:

$$log(Nusikaltimai) = -0.082605 + 0.006049 * Nedarbas + 0.017063 * Zemas + 0.015339 * Vidutinis$$