Individualus Darbas

Vytautas Kraujalis

2021-12-05

Table of Contents

# 1 Reikalingi paketai

library(ggplot2)  
library(dplyr)  
library(ggfortify)  
library(lubridate)  
library(ggthemes)  
library(RColorBrewer)  
library(scales)  
library(forecast)  
library(zoo)  
library(tibble)  
library(tidyr)  
library(ggrepel)

Funkcija laiko eilutės prognozavimo atvaizdavimui, funkciją apsirašiau kitam modeliui, todėl paliksiu ją anglų kalba.

# Header ----  
# Details: Function data inpuits are defined as follows:  
# fx.dat = forecast data object with 3 predictive intervals  
# year = year from which the time series will be subseted  
# freq.place = day / week / month / year from which the time series will be subseted  
# PI = logical if predictive intervals are drawn  
# shade.cols = character string; 3 shade colors for predictive intervals  
# line.cols = character strine; 3 line colors for training, fitted and forecast data  
# date.breaks = character string; such as "3 months", "1 year"  
# data.format = x-axis date format  
# main.title = character string; main plot title  
# sub.title = character string; plot sub title  
# caption = character string; caption  
# x.title = character string; x-axis title  
# y.title = character string; y-axis title  
#  
##-------------------------------------------------------------------------------------------##  
  
# theme for forecast data objects  
theme.fxdat <- theme\_gdocs() +  
 theme(plot.title = element\_text(size = 20),  
 plot.subtitle = element\_text(size = 14),  
 plot.caption = element\_text(size = 9, hjust = 0, vjust = 0, colour = "grey50"),  
 axis.title.y = element\_text(face = "bold", color = "gray30", size = 14),  
 axis.title.x = element\_text(face = "bold", color = "gray30", size = 14, vjust = -1),  
 panel.background = element\_rect(fill = "grey95", colour = "grey75"),  
 panel.border = element\_rect(colour = "grey75"),  
 panel.grid.major.y = element\_line(colour = "white"),  
 panel.grid.minor.y = element\_line(colour = "white", linetype = "dotted"),  
 panel.grid.major.x = element\_line(colour = "white"),  
 panel.grid.minor.x = element\_line(colour = "white", linetype = "dotted"),  
 strip.background = element\_rect(size = 1, fill = "white", colour = "grey75"),  
 strip.text.y = element\_text(face = "bold"),  
 axis.line = element\_line(colour = "grey75"),  
 axis.text = element\_text(size = 14),  
 legend.text = element\_text(size = 14),  
 legend.title = element\_text(size = 14))  
  
# function for using ggplot2 for forecast objects  
plot\_fx <- function(fx.dat,  
 year = NA,  
 freq.place = NA,  
 PI = TRUE,  
 line.cols = NA,  
 shade.cols = NA,  
 date.breaks = NA,  
 date.format = "%Y-%b",  
 main.title = NA,  
 sub.title = NA,  
 caption = NA,  
 x.title = NA,  
 y.title = NA){  
  
 # manage package libraries  
 pkgs <- c("dplyr",  
 "ggplot2",  
 "RColorBrewer",  
 "scales",  
 "forecast",  
 "zoo")  
 attached <- search()  
 attached\_pkgs <- attached[grepl("package", attached)]  
 need\_to\_attach <- pkgs[which(!pkgs %in% gsub("package:", "", attached\_pkgs))]  
 if (length(need\_to\_attach) > 0) {  
 for (i in 1:length(need\_to\_attach)) {  
 require(need\_to\_attach[i], character.only = TRUE)  
 }  
 }  
  
 # data input testing and formatting  
 if (class(fx.dat) != "forecast") {  
 stop("forecast data object required", call. = FALSE)  
 }  
 if (is.na(line.cols[1])) {  
 line.cols = c("black", "darkcyan", "goldenrod1")  
 }  
 if (length(line.cols) != 3) {  
 stop("length of line.cols not equal to 3", call. = FALSE)  
 }  
 if (PI == TRUE) {  
 pi.levels <- fx.dat$level  
 n.levels <- length(pi.levels)  
 if (is.na(shade.cols)) {  
 shade.cols = brewer.pal(n.levels, "PuBuGn")  
 }  
 if (n.levels != length(shade.cols)) {  
 stop("length of shade.cols not equal to number of predictive intervals",  
 call. = FALSE)  
 }  
 }  
 if (is.na(date.breaks)) {  
 print("date.breaks to set to '6 months' absent user input")  
 date.breaks <- "6 months"  
 }  
  
 # define dataframe with training (x), forecast (y) and interval (pi) data  
   
   
 original.ts <- window(fx.dat$x, c(year, freq.place))  
 fitted.ts <- window(fx.dat$fitted, c(year, freq.place))  
 forecast.ts.mean <- fx.dat$mean  
 original.dates <- format(date\_decimal(as.numeric(time(original.ts, offset = 0.5))), format = "%Y-%m-%d")  
 forecast.dates <- format(date\_decimal(as.numeric(time(forecast.ts.mean, offset = 0.5))), format = "%Y-%m-%d")  
   
 len.x <- length(original.ts)  
 len.y <- length(forecast.ts.mean)  
   
 sub.title <- paste0("Faktiniai duomenys rodomi nuo ", format(as.Date(min(original.dates)), format = "%Y-%m"), " - ", format(as.Date(max(original.dates)), format = "%Y-%m"), " ir prognozė nuo ", format(as.Date(min(forecast.dates)), format = "%Y-%m"), " - ", format(as.Date(max(forecast.dates)), format = "%Y-%m"))  
  
 df <- data.frame(  
 date = as.Date(c(original.dates, forecast.dates)),  
 x = c(original.ts, rep(NA, len.y)),  
 fitted = c(fitted.ts, rep(NA, len.y)),  
 forecast = c(rep(NA, len.x), forecast.ts.mean),  
 lo.80 = c(rep(NA, len.x), fx.dat$lower[, 1]),  
 up.80 = c(rep(NA, len.x), fx.dat$upper[, 1]),  
 lo.95 = c(rep(NA, len.x), fx.dat$lower[, 2]),  
 up.95 = c(rep(NA, len.x), fx.dat$upper[, 2]),  
 lo.99 = c(rep(NA, len.x), fx.dat$lower[, 3]),  
 up.99 = c(rep(NA, len.x), fx.dat$upper[, 3])  
 )  
  
 # plot training, fitted and forecast data  
 ggplot(df, aes(date, x)) +  
 geom\_line(aes(colour = "Faktiniai")) +  
 geom\_line(data = df, aes(date, fitted, colour = "Modelio faktiniai"), size = 0.75) +  
 geom\_ribbon(data = df, aes(date, ymin = lo.99, ymax = up.99, fill = "99%")) +  
 geom\_ribbon(data = df, aes(date, ymin = lo.95, ymax = up.95, fill = "95%")) +  
 geom\_ribbon(data = df, aes(date, ymin = lo.80, ymax = up.80, fill = "80%")) +  
 geom\_line(data = df, aes(date, forecast, colour = "Prognozė"), size = 0.75) +  
 geom\_point(data = df, aes(date, forecast, colour = "Prognozė"), size = 1) +  
 geom\_point(size = 1) +  
 scale\_x\_date(breaks = seq(df$date[1], df$date[length(df$date)],  
 by = date.breaks),  
 date\_labels = date.format) +  
 scale\_colour\_manual(name = "Modelis",  
 values = c("Faktiniai" = line.cols[1],  
 "Modelio faktiniai" = line.cols[2],  
 "Prognozė" = line.cols[3]),  
 breaks = c("Faktiniai", "Modelio faktiniai", "Prognozė")) +  
 scale\_fill\_manual(name = "Prognozės PI",  
 values = c("99%" = shade.cols[1], "95%" = shade.cols[2],  
 "80%" = shade.cols[3])) +  
 guides(colour = guide\_legend(order = 1), fill = guide\_legend(order = 2)) +  
 labs(title = main.title,  
 subtitle = sub.title,  
 caption = caption,  
 x = x.title,  
 y = y.title) +  
 theme.fxdat  
}

Naudosime JAV saldainių gaminimo duomenų imtį.

data <- read.csv("candy\_production.csv") %>%   
 rename(data = observation\_date, saldainiai = IPG3113N)

# 2 Užd. 1

Atlikite pirminę duomenų apžvalgą, pašalinkite išskirtis, atsižvelkite į praleistas reikšmes, ir pan.

nrow(data)

## [1] 548

Duomenų imtis sudaryta iš 548 stebėjimų.

sum(is.na(data$saldainiai))

## [1] 0

Duomenų imtyje neturime nei vienos praleistos reikšmės

data %>%   
 group\_by(metai = year(as.Date(data))) %>%   
 summarise(n = n())

## # A tibble: 46 x 2  
## metai n  
## <dbl> <int>  
## 1 1972 12  
## 2 1973 12  
## 3 1974 12  
## 4 1975 12  
## 5 1976 12  
## 6 1977 12  
## 7 1978 12  
## 8 1979 12  
## 9 1980 12  
## 10 1981 12  
## # ... with 36 more rows

Matom, jog turime duomenis nuo 1972 m. iki 2017 m. Visi metai turi po 12 stebėjimų (mėnesiniai duomenys), tik paskutiniai 2017 metai turi 8 stebėjimus.

png(file = "Uzd1\_boxplot.png", width = 1200, height = 850)  
boxplot(data$saldainiai)  
dev.off()

## png   
## 2

Stačiakampė diagrama neparodo, jog mūsų duomenyse būtų kokių nors tai išskirčių.

summary(data)

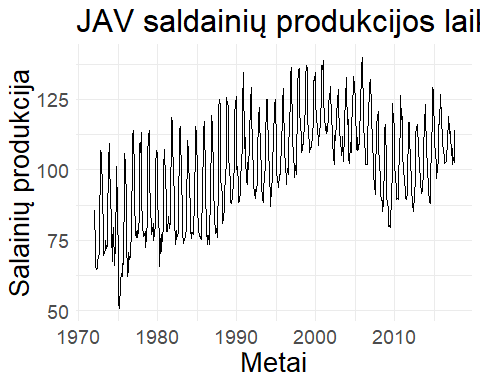
## data saldainiai   
## Length:548 Min. : 50.67   
## Class :character 1st Qu.: 87.86   
## Mode :character Median :102.28   
## Mean :100.66   
## 3rd Qu.:114.69   
## Max. :139.92

Mažiausia reikšmė - 50.67, didžiausia - 139.92 Vidurkis ir mediana yra panašūs, atitinkamai 100.66 ir 102.28.

# 3 Užd. 2

## 3.1 Laiko eilutė

saldainiai.ts <- ts(data$saldainiai, start = c(1972, 1), frequency = 12)  
saldainiai.ts %>%   
 autoplot() +  
 theme\_minimal() +  
 labs(title = "JAV saldainių produkcijos laiko eilutė",  
 x = "Metai",  
 y = "Salainių produkcija") +  
 theme(title = element\_text(size = 20),  
 axis.text = element\_text(size = 14))



ggsave(filename = "Uzd2\_LaikoEilute.png", width = 14, height = 7, units = "in", bg = "white")

Duomenys akivaizdžiai turi sezoniškumą ir trendą

## 3.2 Autokoreliacinė funkcija

png(file = "Uzd2\_Autokoreliacine.png", width = 1200, height = 850)  
Acf(saldainiai.ts)  
dev.off()

## png   
## 2

Autokoreliacijos grafikas parodo, jog mūsų laiko eilutė nėra stacionari, kadangi ties visomis lag reikšmėmis viršijame 95% pasikliautinuosius intervalus, kurie nurodo, jog duomenys skiriasi nuo baltojo triukšmo.

## 3.3 Spektrinio tankio funkcija

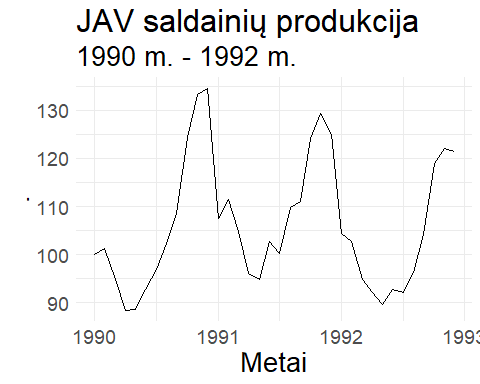
png(file = "Uzd2\_SpektrinioTankio.png", width = 1200, height = 850)  
spectrum(saldainiai.ts)  
dev.off()

## png   
## 2

Kadangi spektriniame grafike matome “pikus” ties 1, 2, 3 ir t.t. dažniais, galime daryti išvadą, jog mūsų duomenyse yra metinis sezoniškumas.

Pasižiūrime, ar mūsų spėjimas yra teisingas:

saldainiai.ts %>%   
 window(start = c(1990, 1), end = c(1992, 12)) %>%   
 autoplot() +  
 theme\_minimal() +  
 labs(title = "JAV saldainių produkcija",   
 subtitle = "1990 m. - 1992 m.",  
 x = "Metai") +  
 theme(title = element\_text(size = 20),  
 axis.text = element\_text(size = 14))

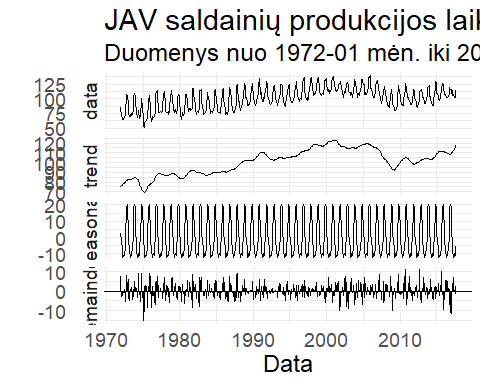


ggsave(filename = "Uzd2\_LaikoEilutesSezoniskumas.png", width = 14, height = 7, units = "in", bg = "white")

Akivaizdžiai matome, jog mūsų spėjimas galimai yra tiesa, matosi, jog metai iš metų kartojasi ta pati tendencija.

# 4 Užd. 3

saldainiai.ts %>%  
 stl(s.window = "periodic") %>%  
 autoplot(range.bars = F) +   
 labs(x = "Data",   
 title = "JAV saldainių produkcijos laiko eilutės dekompozicija",  
 subtitle = "Duomenys nuo 1972-01 mėn. iki 2017-08 mėn.") +  
 theme\_minimal() +  
 theme(text = element\_text(size = 18))



ggsave(filename = "Uzd3\_LaikoEilutesDekompozicija.png", width = 14, height = 7, units = "in", bg = "white")

Kaip jau darėmė išvadą prieš tai, duomenyse matomas akivaizdus metinis sezoniškumas. Tą parodo ir dekompozicijos sezoniškumo grafikas.

# 5 Užd. 4

Prognozuosime 24 laiko momentus (24 mėnesius).

## 5.1 4.1

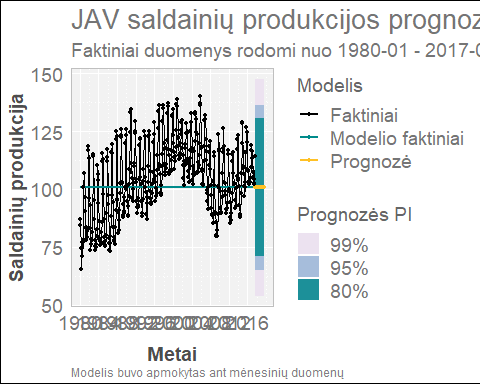
### 5.1.1 Vidurkio metodas

mean\_ts <- meanf(saldainiai.ts, h = 24, level = c(90, 95, 99))

Gauname vidurkio modelį su vidurkiu: 100.6625241

Pasižiūrim, kaip atrodo prognozė:

plot\_fx(  
 mean\_ts,  
 year = 1980,  
 freq.place = 1,  
 date.breaks = "4 years",  
 date.format = "%Y",  
 x.title = "Metai",  
 y.title = "Saldainių produkcija",  
 main.title = "JAV saldainių produkcijos prognozė Vidurkio metodu",  
 caption = "Modelis buvo apmokytas ant mėnesinių duomenų"  
)



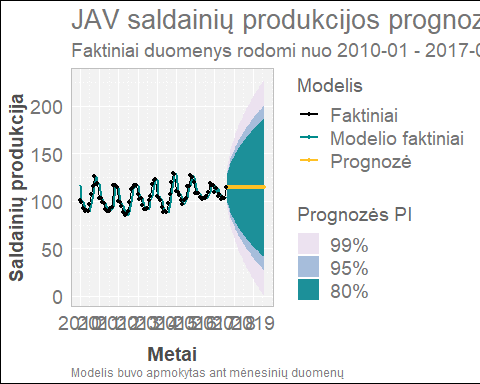
ggsave(filename = "Uzd41\_VidurkioMetodas.png", width = 14, height = 7, units = "in", bg = "white")

Kaip matom, modelio prognozė daug ko nesako, kadangi yra paimamas visos laiko eilutės vidurkis. Taip pat, pasikliautinieji intervalai apima labai didelį plotį.

### 5.1.2 Naivusis metodas

naive\_ts <- naive(saldainiai.ts, h = 24, level = c(90, 95, 99))

plot\_fx(  
 naive\_ts,  
 year = 2010,  
 freq.place = 1,  
 date.breaks = "years",  
 date.format = "%Y",  
 x.title = "Metai",  
 y.title = "Saldainių produkcija",  
 main.title = "JAV saldainių produkcijos prognozė Naiviuoju metodu",  
 caption = "Modelis buvo apmokytas ant mėnesinių duomenų"  
)



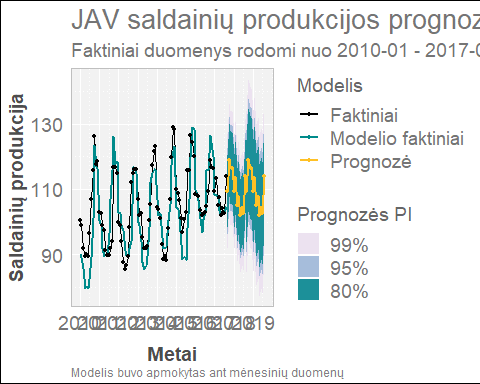
ggsave(filename = "Uzd41\_NaivusisMetodas.png", width = 14, height = 7, units = "in", bg = "white")

Naiviojo metodo prognozė yra tiesiog paskutinė laiko eilutės reikšmė, todėl tolimame horizonte ši prognozė nebus tikslinga, tą parodo ir platėjantis pasikliautinasis intervlas.

### 5.1.3 Sezoninis naivusis metodas

snaive\_ts <- snaive(saldainiai.ts, h = 24, level = c(90, 95, 99))

plot\_fx(  
 snaive\_ts,  
 year = 2010,  
 freq.place = 1,  
 date.breaks = "years",  
 date.format = "%Y",  
 x.title = "Metai",  
 y.title = "Saldainių produkcija",  
 main.title = "JAV saldainių produkcijos prognozė Sezoniniu Naiviuoju metodu",  
 caption = "Modelis buvo apmokytas ant mėnesinių duomenų"  
)



ggsave(filename = "Uzd41\_SezoninisNaivusisMetodas.png", width = 14, height = 7, units = "in", bg = "white")

Sezoninio naiviojo metodo prognozė galima laikyti daug tikslingesne, tačiau vėl gi, yra atkartojamas paskutinis “sezonas”.

## 5.2 4.2

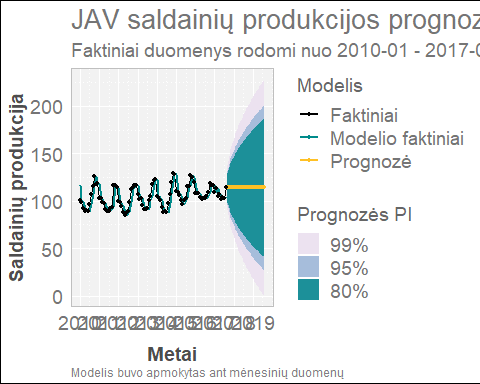
### 5.2.1 Paprastasis eksponentinis glodinimas

ses\_ts <- ses(saldainiai.ts, h = 24, level = c(90, 95, 99))

Paprastojo eksponentinio glodinimo metodo alpha parametro reikšmė: 0.9999, kadangi alpha parametras yra arti 1, todėl didžiausias svoris yra suteikiamas paskutinėms laiko eilutės reikšmėms ir metodas turėtų sutapti su naiviuoju metodu.

Pradinio suglodintos laiko eilutės vidurkio lygis *l\_0* - 75.9058713

plot\_fx(  
 ses\_ts,  
 year = 2010,  
 freq.place = 1,  
 date.breaks = "years",  
 date.format = "%Y",  
 x.title = "Metai",  
 y.title = "Saldainių produkcija",  
 main.title = "JAV saldainių produkcijos prognozė Paprastuoju Eksponentiniu Glodinimu",  
 caption = "Modelis buvo apmokytas ant mėnesinių duomenų"  
)



ggsave(filename = "Uzd42\_PaprastatisEksponentinisGlodinimas.png", width = 14, height = 7, units = "in", bg = "white")

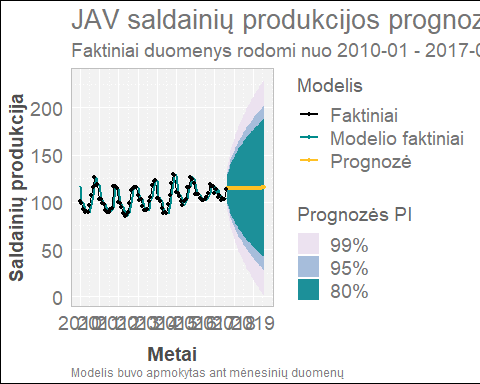
Matome, kad metodas sutampa su naiviuoju metodu.

### 5.2.2 Dvigubas eksponentinis glodinimas

holt\_ts <- holt(saldainiai.ts, h = 24, level = c(90, 95, 99))

Dvigubo eksponentinio glodinimo metodo alpha parametras = 0.9989008, matom, jog alpha yra arti 1, todėl metodas vėl bus panašus į naivųjį metodą. Trendo glodinimo parametras beta = 1.0000412^{-4}. Pradinės sąlygos: *l\_0* = 74.4589216 ir *b\_0* = 0.0486155.

plot\_fx(  
 holt\_ts,  
 year = 2010,  
 freq.place = 1,  
 date.breaks = "years",  
 date.format = "%Y",  
 x.title = "Metai",  
 y.title = "Saldainių produkcija",  
 main.title = "JAV saldainių produkcijos prognozė Dvigubu Eksponentiniu Glodinimu",  
 caption = "Modelis buvo apmokytas ant mėnesinių duomenų"  
)



ggsave(filename = "Uzd42\_DvigubasEksponentinisGoldinimas.png", width = 14, height = 7, units = "in", bg = "white")

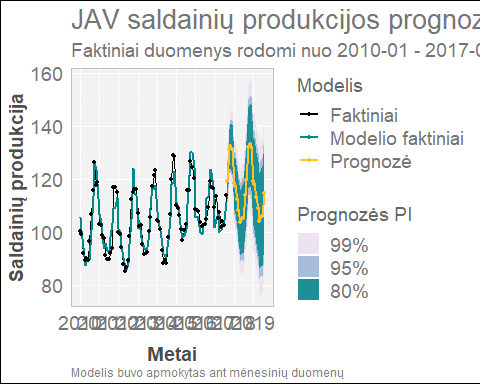
Vėlgi, matom jog metodas yra labai panašus į naivųjį metodą.

### 5.2.3 Holt-Winters sezoninis

holtwinters\_ts <- hw(saldainiai.ts, h = 24, level = c(90, 95, 99))

Holt-Winters sezoninio metodo alpha parametras = 0.557695, matom, jog alpha nėra arti 1, todėl metodas neturėtų būti panašus į naivųjį metodą. Trendo glodinimo parametras beta = 3.3289628^{-4}, parametro reikšmė yra gana maža, todėl ir prognozėje spartaus trendo nepamatysime. Sezoniškumo glodinimo parametras gamma = 0.2416328.

plot\_fx(  
 holtwinters\_ts,  
 year = 2010,  
 freq.place = 1,  
 date.breaks = "years",  
 date.format = "%Y",  
 x.title = "Metai",  
 y.title = "Saldainių produkcija",  
 main.title = "JAV saldainių produkcijos prognozė Holt-Winters Sezoniniu metodu",  
 caption = "Modelis buvo apmokytas ant mėnesinių duomenų"  
)



ggsave(filename = "Uzd42\_HoltWintersSezoninisMetodas.png", width = 14, height = 7, units = "in", bg = "white")

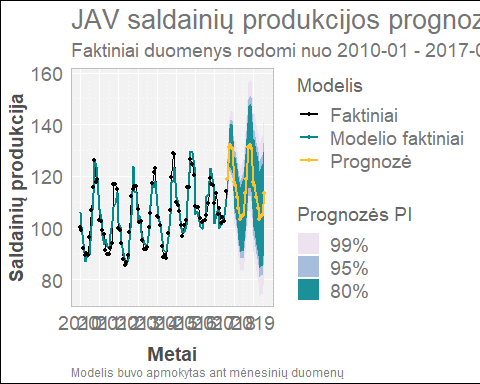
Matome, jog holt-winters sezoninis metodas iš pirmo žvilgsnio visai neblogai prognozuoja (pateikia adekvatesnią prognozę atsižvelgiančią į sezoniškumą).

### 5.2.4 ETS

ets\_ts <- forecast(ets(saldainiai.ts), h = 24, level = c(90, 95, 99))

ETS metodo parinkti tipai: paklaidos - A, trendo - N, sezoniškumo - A. Parametrų reikšmės: alpha = 0.6057677, gamma = 0.2241423, trendo komponentės neturime, todėl ir beta parametro neturime.

plot\_fx(  
 ets\_ts,  
 year = 2010,  
 freq.place = 1,  
 date.breaks = "years",  
 date.format = "%Y",  
 x.title = "Metai",  
 y.title = "Saldainių produkcija",  
 main.title = "JAV saldainių produkcijos prognozė ETS metodu",  
 caption = "Modelis buvo apmokytas ant mėnesinių duomenų"  
)



ggsave(filename = "Uzd42\_ETSMetodas.png", width = 14, height = 7, units = "in", bg = "white")

Iš esmės, parinktas metodas sutampa su holt-winters metodu, kuriame mūsų trendo parametro reikšmė buvo parinkta labai maža.

## 5.3 4.3

### 5.3.1 ARIMA

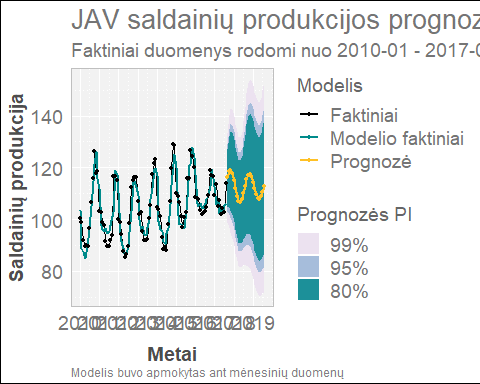
arima <- auto.arima(saldainiai.ts, seasonal = FALSE)  
  
arima\_ts <- forecast(arima, h = 24, level = c(90, 95, 99))

Parinktas arima procesas: ARIMA(3,1,2) su eilėmis:

summary(arima\_ts)$model$coef

## ar1 ar2 ar3 ma1 ma2   
## 1.5387396 -0.6909358 -0.1519496 -1.7721660 0.8862690

plot\_fx(  
 arima\_ts,  
 year = 2010,  
 freq.place = 1,  
 date.breaks = "years",  
 date.format = "%Y",  
 x.title = "Metai",  
 y.title = "Saldainių produkcija",  
 main.title = "JAV saldainių produkcijos prognozė ARIMA metodu",  
 caption = "Modelis buvo apmokytas ant mėnesinių duomenų"  
)



ggsave(filename = "Uzd43\_ArimaMetodas.png", width = 14, height = 7, units = "in", bg = "white")

Matom, jog paprastas arima metodas sugebėjo aptikti sezoniškumą.

### 5.3.2 SARIMA

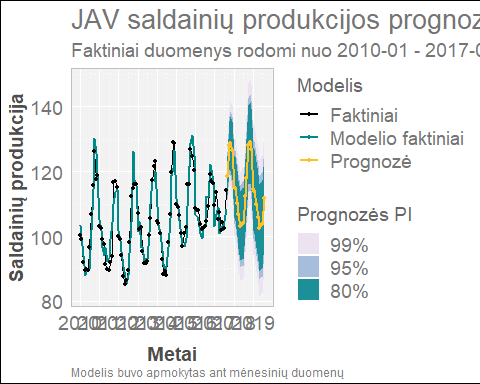
sarima <- auto.arima(saldainiai.ts, seasonal = TRUE)  
  
sarima\_ts <- forecast(sarima, h = 24, level = c(90, 95, 99))

Parinktas sarima procesas: ARIMA(2,0,2)(0,1,2)[12] with drift su eilėmis:

summary(sarima\_ts)$model$coef

## ar1 ar2 ma1 ma2 sma1 sma2   
## 0.008923305 0.827347923 0.680406780 -0.266649289 -0.606346560 -0.115289761   
## drift   
## 0.059450663

plot\_fx(  
 sarima\_ts,  
 year = 2010,  
 freq.place = 1,  
 date.breaks = "years",  
 date.format = "%Y",  
 x.title = "Metai",  
 y.title = "Saldainių produkcija",  
 main.title = "JAV saldainių produkcijos prognozė SARIMA metodu",  
 caption = "Modelis buvo apmokytas ant mėnesinių duomenų"  
)



ggsave(filename = "Uzd43\_SarimaMetodas.png", width = 14, height = 7, units = "in", bg = "white")

Sarima metodas aptiko daug stipresnį sezoniškumą, taip pat pasikliautinieji intervalai yra žymiai siauresnis nei Arima metodo.

# 6 Užd. 5

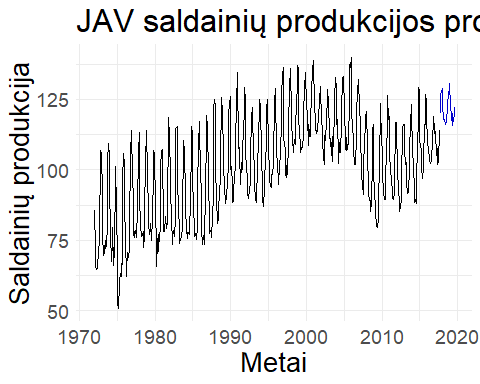
## 6.1 Neuroniniai tinklai

nn <- nnetar(saldainiai.ts)  
  
nn\_ts <- forecast(nn, h = 24)

Sukurtas neuroninis tinklas - NNAR(26,1,14)[12], kur P = 1, vadinasi metodas aptiko sezoniškumą, p = 26 ir k = 14.

Kadangi nnetar() funkcija neapskaičiuoja pasikliautinųjų intervalų, naudosime paprastą eilutės atvaizdavimą:

nn\_ts %>%   
 autoplot() +  
 theme\_minimal() +  
 labs(title = "JAV saldainių produkcijos prognozė Neuroninių Tinklų metodu",   
 x = "Metai",  
 y = "Saldainių produkcija") +  
 theme(title = element\_text(size = 20),  
 axis.text = element\_text(size = 14))



ggsave(filename = "Uzd5\_NeuroniniuTinkluMetodas.png", width = 14, height = 7, units = "in", bg = "white")

Matom, jog neuroninių tinklu metodas aptiko tiek sezoniškumą, tiek trendą.

# 7 Užd. 6

Išskaidom laiko eilutę į apmokymo ir testavimo imtis

apmokymo <- window(saldainiai.ts, start = c(1972,1), end = c(2009,12))  
testavimo <- window(saldainiai.ts, start = c(2010,1))

Apmokom visus metodus su apmokymo imtimi

mean\_ts\_apmokymo <- meanf(apmokymo, h = 24, level = c(90, 95, 99))  
naive\_ts\_apmokymo <- naive(apmokymo, h = 24, level = c(90, 95, 99))  
snaive\_ts\_apmokymo <- snaive(apmokymo, h = 24, level = c(90, 95, 99))  
ses\_ts\_apmokymo <- ses(apmokymo, h = 24, level = c(90, 95, 99))  
holt\_ts\_apmokymo <- holt(apmokymo, h = 24, level = c(90, 95, 99))  
holtwinters\_ts\_apmokymo <- hw(apmokymo, h = 24, level = c(90, 95, 99))  
ets\_ts\_apmokymo <- forecast(ets(apmokymo), h = 24, level = c(90, 95, 99))  
  
arima\_apmokymo <- auto.arima(apmokymo, seasonal = FALSE)  
arima\_ts\_apmokymo <- forecast(arima\_apmokymo, h = 24, level = c(90, 95, 99))  
  
sarima\_apmokymo <- auto.arima(apmokymo, seasonal = TRUE)  
sarima\_ts\_apmokymo <- forecast(sarima\_apmokymo, h = 24, level = c(90, 95, 99))  
  
nn\_apmokymo <- nnetar(apmokymo)  
nn\_ts\_apmokymo <- forecast(nn\_apmokymo, h = 24)

Apsirašom apmokytus modelius į vieną list’ą

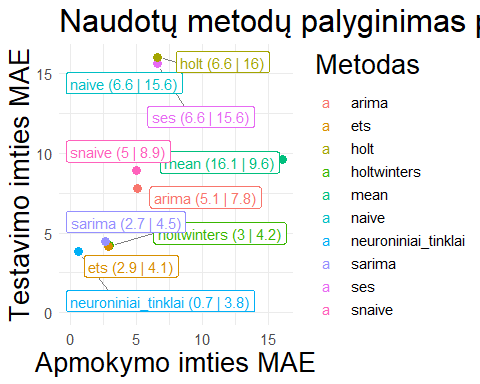
modeliai <- list(  
 mean = mean\_ts\_apmokymo,  
 naive = naive\_ts\_apmokymo,  
 snaive = snaive\_ts\_apmokymo,  
 ses = ses\_ts\_apmokymo,  
 holt = holt\_ts\_apmokymo,  
 holtwinters = holtwinters\_ts\_apmokymo,  
 ets = ets\_ts\_apmokymo,  
 arima = arima\_ts\_apmokymo,  
 sarima = sarima\_ts\_apmokymo,  
 neuroniniai\_tinklai = nn\_ts\_apmokymo  
)

Gaunam tikslumo matus apmokymo ir testavimo imčiai

tikslumai <- lapply(modeliai, accuracy, x = testavimo)  
tikslumai <- do.call(rbind.data.frame, tikslumai) %>%   
 rownames\_to\_column("Metodas") %>%   
 select(Metodas, MAE) %>%   
 separate(col = Metodas, into = c("Metodas", "Imtis"), sep = "\\.") %>%   
 pivot\_wider(names\_from = Imtis, values\_from = MAE) %>%   
 rename(MAE\_apmokymo = `Training set`, MAE\_testavimo = `Test set`)

Palyginam metodus:

tikslumai %>%   
 ggplot(aes(x = MAE\_apmokymo, y = MAE\_testavimo, color = Metodas, label = Metodas)) +  
 geom\_point(size = 3) +  
 geom\_label\_repel(aes(label = paste0(Metodas, " (", round(MAE\_apmokymo, 1), " | ", round(MAE\_testavimo, 1), ")")),  
 box.padding = 0.35,   
 point.padding = 0.5,  
 segment.color = 'grey50') +  
 theme\_minimal() +  
 labs(title = "Naudotų metodų palyginimas pagal MAE naudojant apmokymo ir testavimo imtis",  
 x = "Apmokymo imties MAE",  
 y = "Testavimo imties MAE") +  
 theme(title = element\_text(size = 20),  
 text = element\_text(size = 14)) +  
 expand\_limits(x = 0, y = 0)



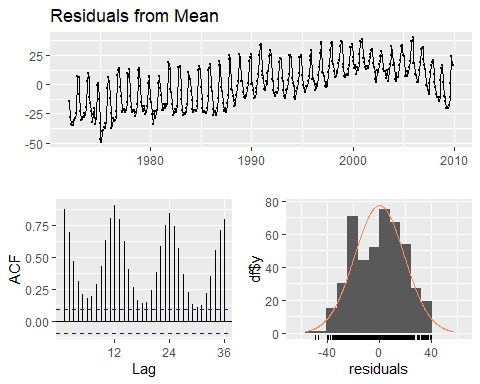
ggsave(filename = "Uzd6\_MetoduPalyginimas.png", width = 14, height = 7, units = "in", bg = "white")

# 8 Užd. 7

h <- data.frame(matrix(NA, 1, 1))  
  
for(i in 1:length(testavimo)){  
 metodas\_ts <- hw(apmokymo, h = i)  
 h[i] <- accuracy(metodas\_ts, x = testavimo)[2,3]  
}  
  
h

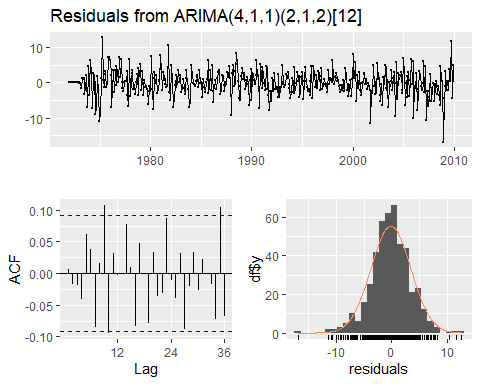
## matrix.NA..1..1. V2 V3 V4 V5 V6 V7  
## 1 4.820918 4.580591 5.145176 4.594597 4.185376 4.029704 3.951384  
## V8 V9 V10 V11 V12 V13 V14 V15  
## 1 4.273994 4.388958 4.324648 4.048654 3.898372 3.79349 3.608151 3.390666  
## V16 V17 V18 V19 V20 V21 V22 V23 V24  
## 1 3.472209 3.359413 3.365219 3.385646 3.65359 4.2909 4.376125 4.298041 4.200127  
## V25 V26 V27 V28 V29 V30 V31 V32  
## 1 4.273322 4.305488 4.327022 4.370158 4.494096 4.57663 4.566294 4.509082  
## V33 V34 V35 V36 V37 V38 V39 V40  
## 1 4.400573 4.518731 4.496044 4.403434 4.302783 4.255942 4.228835 4.165439  
## V41 V42 V43 V44 V45 V46 V47 V48  
## 1 4.11152 4.064505 4.010581 3.943766 3.994397 4.046237 3.990281 4.022592  
## V49 V50 V51 V52 V53 V54 V55 V56  
## 1 3.985912 3.926114 3.873443 3.818588 3.854155 3.875024 3.922362 3.919246  
## V57 V58 V59 V60 V61 V62 V63 V64 V65  
## 1 3.942929 3.950052 4.026535 4.133677 4.111206 4.102358 4.1372 4.176305 4.147  
## V66 V67 V68 V69 V70 V71 V72 V73  
## 1 4.176423 4.231368 4.370281 4.355667 4.32521 4.313875 4.278152 4.232106  
## V74 V75 V76 V77 V78 V79 V80 V81  
## 1 4.20719 4.248631 4.308477 4.345706 4.383089 4.423584 4.394218 4.384575  
## V82 V83 V84 V85 V86 V87 V88 V89  
## 1 4.402666 4.404463 4.381074 4.346962 4.3816 4.379515 4.468499 4.492635  
## V90 V91 V92  
## 1 4.538957 4.564015 4.634408

checkresiduals(mean\_ts\_apmokymo)



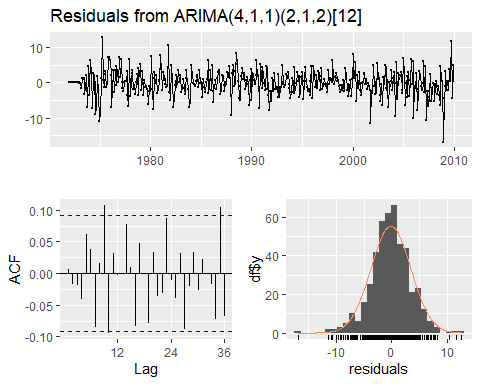
##   
## Ljung-Box test  
##   
## data: Residuals from Mean  
## Q\* = 3290.9, df = 23, p-value < 2.2e-16  
##   
## Model df: 1. Total lags used: 24

checkresiduals(sarima\_ts\_apmokymo)



##   
## Ljung-Box test  
##   
## data: Residuals from ARIMA(4,1,1)(2,1,2)[12]  
## Q\* = 32.667, df = 15, p-value = 0.005218  
##   
## Model df: 9. Total lags used: 24

checkresiduals(sarima\_ts\_apmokymo)



##   
## Ljung-Box test  
##   
## data: Residuals from ARIMA(4,1,1)(2,1,2)[12]  
## Q\* = 32.667, df = 15, p-value = 0.005218  
##   
## Model df: 9. Total lags used: 24

<https://www.kaggle.com/goldens/candy-production-time-series-analysis> <https://www.kaggle.com/grosvenpaul/eda-and-time-series-modeling> <https://otexts.com/fpp2/accuracy.html>