Spis treści

Wprowadzenie
Informacje wstępne
Wczytanie/oczyszczenie/transformacja . . .
Analiza opisowa
Przekształcanie szeregów
Dekompozycja . . .
Modele ARIMA
Prognozowanie
Metody sztucznych . . .
Metody niekonwencjonalne

1. Wprowadzenie

1.1. Literatura

Adam Zagdański i Artur Suchowałko: *Analiza i prognozowanie szeregów czasowych*, PWN 2016 `http://quantup.pl/o-nas/ksiazka-szeregi-czasowe/`

Maria Cieślak: *Prognozowanie gospodarcze. Metody i zastosowania*, PWN 2001–2005

Aleksander Zeliaś, Barbara Pawełek, Stanisław Wanat; *Prognozowanie ekonomiczne. Teoria, przykłady, zadania*, PWN 2004

1.2. Oprogramowanie

R/Rstudio do pobrania z:
`https://cran.r-project.org/bin/windows/base/`
`https://www.rstudio.com/products/rstudio/download/`

Oprogramowanie wspomagające (system kontroli wersji git):
`https://git-scm.com/downloads`

1.3. Namiary na materiały/kontakt

`https://github.com/hrpunio/PPE`

`tprzechlewski@acm.org`

1.4. Zaliczenie

Projekt zaliczeniowy wykonany w formie pisemnej składający się z dwóch części: 1) **raportu z przeprowadzonego badania** oraz 2) **eseju** dotyczący niestandardowej/nieomawianej na zajęciach metody prognozowania lub innego ciekawego aspektu problematyki prognozowania (szczegóły do ustalenia).

Autorzy projektu powinni **samodzielnie określić ciekawy problem prognostyczny** (co będzie dodatkowo punktowane), jeżeli będą do tego niezdolni to takowy zostanie zaproponowany przez prowadzącego

Zaliczenie projektu odbędzie się na **ostatnich zajęciach**.

2. Informacje wstępne

2.1. Przegląd metod prognozowania

Metoda prognozowania to sposób przetworzenia danych o przyszłości oraz sposób przejścia od danych przetworzonych do prognozy [C.37]:
faza diagnozowania → faza określenia przyszłości

Diagnozowanie to budowa modelu formalnego lub myślowego

Reguła prognozy to sposób przejścia od danych przetworzonych do prognozy: podstawowa, podstawowa z poprawką, największego prawdopodobieństwa i minimalnej straty [C.39]

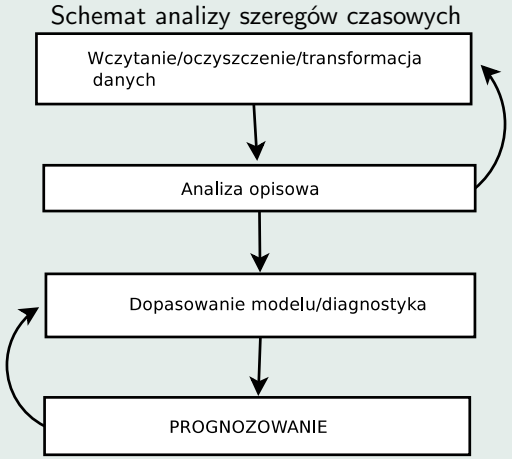
Podstawowa. Prognozą jest stan zmiennej prognozowanej w należącym do przyszłości momencie/okresie, otrzymany z modelu przy przyjęciu założenia, że model będzie aktualny w chwili, na którą określa się prognozę.

Prognozę otrzymuje się wskutek ekstrapolacji modelu poza próbę.
Użyteczna w prognozowaniu zjawisk **o dużej inercji**.

2.2. Etapy prognozowania

Wg [C.59]: sformułowanie zadania prognostycznego (obiekt/zjawisko/zmienne, którego stan podlega prognozowaniu, cel prognozowania, wymagania odnośnie dopuszczalności i horyzontu) → podanie przesłanek prognostycznych → wybór metody → wyznaczenie prognozy → ocena dopuszczalności → weryfikacja.

Mniej formalnie/bardziej praktycznie [ZS.23]



3. Wczytanie/oczyszczenie/transformacja danych

Git/github/praca grupowa. Koncepcja *reproducible research*

4. Analiza opisowa

4.1. Wykresy

Sposób działania funkcji `plot()` zależy od typu obiektu

```
library(TSAFBook)
```

```
# install.packages("TSAFBook")
# package 'TSAFBook' is not available (for R version 3.2.2)
# Pobierz ze strony http://quantup.pl/o-nas/ksiazka-szeregi-czasowe/
# Wykonaj (pierwszy parametr to nazwa pliku ze ścieżką):
# install.packages("C:\\Users\\E5410\\Downloads\\TSAFBook_0.1.tar.gz",
#   repos = NULL, type="source")

data(AirPass)

par(mfrow = c(2,1))
#?par
# szereg typu ts
plot(AirPass, main="Szereg AirPass")

# File -> Save As. / pdf("mygraph.pdf")
# zwykły wektor
plot(as.vector(AirPass), main="Szereg AirPass", type="1")
```

4.2. Funkcja `xyplot`

Funkcja `xyplot()` z pakietu `lattice`

modyfikacja *aspect-ratio* /podział (długiego szeregu) na panele:

```
library ("lattice")
xyplot (AirPass)

xyplot(AirPass, aspect = 1/4)
```

```
xyplot(AirPass, strip=T, cut = list(number=3, overlap=0,5))
```

Wykresy sezonowe – monthplot

```
library ("TSAFBook")
library("expsmooth")
data(AirPass)
data(usgdp)

par(mfrow = c(2,1))
monthplot(AirPass, main="Szereg AirPass")

monthplot(usgdp, main="Szereg USGDP")
```

Wykresy sezonowe – seasonplot

```
library (forecast)
par(mfrow = c(2,1))

seasonplot(AirPass, col=rainbow"12'', year.labels=T, pch=19)

seasonplot(usgdp, col=rainbow"12'', year.labels=T, pch=19)
```

4.3. Autokorelacja

Czy i w jakim stopniu wcześniejsze obserwacje mają wpływ na aktualną wartość szeregu = jak silna jest autokorelacja

Metody graficzne wykrywania autokorelacji: lag plots, funkcja autokorelacji (ACF), funkcja cząstowej autokorelacji (PACF)

lag plot to wykres rozrzutu (scatter plot) dla par (X_t, X_{t-h})

```
b.szum <- as.ts (rnorm(100)) # dane losowe
lag.plot(bialy.szum, lags = 4, do.lines=F, main="Biały szum wykres")

lag.plot(AirPass, lags = 12, do.lines=F, main="AirPass wykres")
```

Autokorelacja rzędu k jest funkcją, która argumentowi naturalnemu k przypisuje wartość współczynnika korelacji Pearsona pomiędzy szeregiem czasowym, a tym samym szeregiem cofniętym o k jednostek czasu.

Funkcja autokorelacji (ACF)

$$ACF(h) = \rho(k) = \gamma(k)/\gamma(0) \quad (1)$$
$$\gamma(h) = cov(X_t, X_{t+h}) \quad (2)$$

$ACF(0) = 1; ACF(h) \in [-1, 1]$

Interpretacja wykresu ACF (korelogramu):

dodatnie i powoli znikające wartości ACF wskazują na trend

powoli i cyklicznie znikające wartości ACF wskazują na sezonowość

Funkcja autokorelacji cząstkowej (PACF)

$$\alpha(1) = \mathrm{Corr}(X_2, X_1) = \rho(1) \quad \text{oraz} \quad (3)$$
$$\alpha(h) = \mathrm{Corr}(X_{h+1} - P_{\overline{SP}\{1,X_2,\ldots x_h\}} X_{h+1}, X_1 - P_{\overline{SP}\{1,X_2,\ldots x_h\}} X_{h+1}) \quad (4)$$
$$\text{dla } h \geqslant 2 \quad (5)$$

gdzie

$P_{\overline{sp}\{1,x_2,\ldots x_h\}}$ oznacza rzut ortogonalny na podprzestrzeń liniową wyznaczoną przez $1, X_2, \ldots, X_h$

5. Dekompozycja szeregów czasowych

Dekompozycja: wyodrębnienie **składowej systematycznej** oraz **składowej przypadkowej**.

Składowa systematyczna: **trend** (długoterminowa ogólna tendencja rozwojowa); **cykliczność** (regularne wzorce wzrostó/spadków); **sezonowość** (krótkie, powtarzające się cykle, związanie najczęściej z porami roku/dnia);

5.1. Rodzaje dekompozycji

Ogólna formuła

$$Y_t = f(s_t, m_t, Z_t) \tag{6}$$

gdzie: s_t – składowa sezonowa, m_t – trend; Z_t – zakłócenie losowe

Dekompozycja **addytywna** (amplituda wahań stała):

$$Y_t = s_t + m_t + Z_t \tag{7}$$

Dekompozycja **multiplikatywna** (amplituda wahań rośnie/maleje):

$$Y_t = s_t \cdot m_t \cdot Z_t \tag{8}$$

Zamiast dekompozycji multiplikatywnej można zastosować najpierw transformację Boxa-Coxa a następnie model dekompozycji addytywnej.

5.2. Parametryczne/nieparametryczne metody dekompozycji

Parametryczne – oparte o określony model formalny (np. liniowa f. trendu).
Nieparametryczne – średnia ruchoma, ważona śrenia ruchoma

5.3. Średnia ruchoma

Symetryczna średnia ruchoma [ZS.146]:

$$m(t) = 1/(2q+1) \sum_{j=-q}^{q} Y_{t-j} \tag{9}$$

Do obliczania średniej ruchomej służy funkcja `filer` z pakietu albo `ma` z pakietu `forecast stats`:

```
library ("forecast")
library("expsmooth")
##> install.packages("expsmooth")
##Installing package into ‘C:/Users/E5410/Documents/R/win-library/3.1’

data("dji")

## summary(dij)
dji.close <- dji[, "Close"]

m3 <- ma (dji.close, order=3)
## albo
m3 <- filter (dji.close, sides=2, filter=rep(1/3, 3))
```

```
## ?rep

plot(m3)

ts.plot(m3,dji.close)
##?ts.plot

## jeszcze więcej wygłaszania
m11 <- ma(dji.close, order = 11)
m21 <- ma (dji.close, order=21)

ts.plot(m3,dji.close, m21)

## albo standardowe plot:

plot(m3, main = paste (" Metoda symetrycznej ruchomej sredniej"),
  col = "blue", lty = 2)
lines(m11, col = "red", lty = 2)
lines(m21, col = " green ", lty = 2)
lines(dji.Close , col = " black ", lty = 1)
legend("bottomright",
legend = c(" wyjsciowy szereg ", "ruchoma srednia (q=1)",
  "ruchoma srednia (q=5)","ruchoma srednia (q =10)"),
col = c("black", "blue", "red", "green"), lty = c(1, 2, 2, 2))
```

Podstawowy problem: jaki wybrać rząd wygładzania – za mały, zbyt duża zmienność; za duży, zbyt duże wygładzenie.

Dekompozycja (funkcja decompose z pakietu stats):

```
dji.close.decomp.add <- decompose(dji.close, type = "additive")

summary(dji.close.decomp.add)

plot(dji.close.decomp.add)
```

5.4. Średnia ruchoma ważona

Ważona średnia ruchoma [ZS.151]:

$$m(t) = \sum_{j=-q}^{q} w_j \cdot Y_{t-j} \tag{10}$$

gdzie: $w_{-q} + w_{-q+1} + \cdot + w_{q-1} + w_q = 1$ oraz $w_j = w_{-j}$

Przykład: Filtr Spencera [ZS.151], dla którego rząd $q = 7$ i wagi określamy następująco:

$$[w_0, \ldots, w_7] = 1/320 \cdot [74, 67, 46, 21, 3, -5, -6, -3] \tag{11}$$

```
m.spencer <- filter (x=dji.Close,
(1/320) * c(-3, -6, -5, 3, 21, 46, 67, 74, 67, 46, 21, 3, -5, -6, -3),
sides=2)

plot(m3, main="Metoda średniej ruchomej", col = "blue", lty=2)
lines(m11, col="red", lty=2)
lines(m21, col="magenta", lty=2)
lines(m.spencer, col="green", lty=2)
lines(dji.Close, col="black", lty=1)
grid()
```

5.5. Model regresji

Prosta regresja liniowa:

$$Y_t = a + b \cdot t + Z_t, \quad t = 1, 2, \ldots \tag{12}$$

```
library("TSAFBook")
## Pobierz ze strony http://quantup.pl/o-nas/ksiazka-szeregi-czasowe/
## Wykonaj (pierwszy parametr to nazwa pliku ze ścieżką):
## install.packages("C:\\Users\\E5410\\Downloads\\TSAFBook_0.1.tar.gz",
##    repos = NULL, type="source")
## Biblioteka jest instalowana gdzieś tutaj:
## C:/Users/E5410/Documents/R/win-library/3.1'

library("TSAFBook")
data(pkb)

# dekompozycja na podstawie modelu regresji: trend liniowy

pkb.tslm.t <- tslm(pkb ~ trend)
summary(pkb.tslm.t)
tsdisplay(residuals(pkb.tslm.t), main = "reszty losowe")


# dekompozycja na podstawie modelu regresji: trend liniowy + sezonowość
pkb.tslm.s <- tslm(pkb ~ trend + season)
summary(pkb.tslm.s)
tsdisplay(residuals(pkb.tslm.s), main = "reszty losowe")
```

```
# dekompozycja zlogarymowanych danych: trend liniowy + sezonowość
# transformacja Boxa-Coxa (lambda=0)
pkb.log.tslm.l <- tslm(pkb ~ season + trend, lambda = 0)
summary(pkb.log.tslm.l)
tsdisplay(residuals(pkb.log.tslm.l), main = "reszty losowe")

# dekompozycja zlogarymowanych danych: trend kwadratowy + sezonowość
pkb.log.tslm.sq <- tslm(pkb ~ season + trend + I(trend ^ 2), lambda = 0)
summary(pkb.log.tslm.sq)
tsdisplay(residuals(pkb.log.tslm.sq), main = "reszty losowe")

# porównanie modeli dekompozycji
plot(pkb, col = "black", main = " Dane oryginalne ")
lines(fitted(pkb.tslm.s), col = "blue")
lines(fitted(pkb.log.tslm.l), col = "red")
lines(fitted(pkb.log.tslm.sq), col = "green")
legend("bottomright",legend=c("oryginalny szereg",
  "trend liniowy + sezonowość","trend liniowy + sezonowość (dane zlog.)",
  "trend kwadartowy + sezonowość (dane zlog.)"),
 col = c("black","blue","red","green"), lwd=1)
```

6. Modele ARIMA

→ Następne spotkanie

7. Prognozowanie

7.1. Ocena dokładności prognoz

Trafność prognozy to jakość prognoz ustalana **ex-post**. **Dokładność prognozy** to jakość prognoz ustalana **ex-ante**.

Dopuszczalność prognozy: prognoza jest dopuszczalna, gdy jest obdarzona przez odbiorcę stopniem zaufania wystarczającym do tego, aby mogła być wykorzystana w celu, dla którego została ustalona [C.54]

Maksymalny horyzont prognozy to należący do przyszłości najdalszy moment/okres, dla którego prognoza jest dopuszczalna.

Wiele metod prognozowania (większość?) **nie pozwalaja** na ustalenie błędu ex-ante/prawdopodobieństwa realizacji prognozy.

7.2. Miary dokładności prognozowania

Błąd prognozy:

$$\mathrm{PE}_t = Y_t - F_t \tag{13}$$

gdzie: PE_t – błąd predykcji (*prediction error*); Y_t – wartość rzeczywista; F_t – wartość prognozowana

Średni błąd kwadratowy (MSE)

$$\mathrm{MSE} = \frac{1}{n}\sum_{t=1}^{n}(Y_t - F_t)^2 \tag{14}$$

Pierwiastek błędu średniokwadratowego (RMSE)

$$\mathrm{RMSE} = \sqrt{\mathrm{MSE}} \tag{15}$$

Inne współczynniki → [ZS.254]

Implementacja w R: funkcja `accuracy` z pakietu `forecast` (przykłady w następnym punkcie)

7.2.1. Przedziały predykcyjne

Przedziałem predykcyjnym jest przedział losowy $\mathrm{PI}(h) = [L, P]$, skonstuowany tak, aby zawierał przyszłą (nieznaną) obserwację Y_{n+h} z określonym prawdopodobieństwem.

Gaussowskie przedziały predykcyjne (Boxa-Jenkinsa). Jeżeli szereg czasowy jest gaussowskim szeregiem czasowym, to błąd predykcji $\mathrm{PE}(h) \sim N(0, \mathrm{PMSE}(h)$.
Stąd końce przedziałów dane są wzorem:

$$\mathrm{PI}_{\mathrm{BJ}}(h) = [F_{n+h} - z_{\frac{1}{2}(1+\gamma)}\sqrt{\mathrm{PMSE}(h)}, F_{n+h} + z_{\frac{1}{2}(1+\gamma)}\sqrt{\mathrm{PMSE}(h)}] \qquad (16)$$

gdzie: F_{n+h} – prognozowana wartość Y_{t+h};
$\sqrt{\mathrm{PMSE}(h)} = \sqrt{E(F_{n+h} - Y_{n+h})^2}$ – pierwiastek średniokwadratowego błędu predykcji (*prediction MSE*); $z_{\frac{1}{2}(1+\gamma)}$ – kwantyl rzędu $\frac{1}{2}(1+\gamma)$ z rozkładu $N(0,1)$.

Implementacja w R: z automatu, należy podać w argumencie `level` poziom ufności, np:

```
library("forecast")
library("TSAFBook")
data(usgdp)

## 95% przedział predykcyjny
usgdp.forecast.rwf <- rwf(usgdp, drift =T, h=20, level=0.95)
summary(usgdp.forecast.rwf)

## co jest w środku obiektu usgdp.forecast.rwf?:
attributes(usgdp.forecast.rwf)
```

```
## drukowanie
cat(sprintf("%f %f\n", usgdp.forecast.rwf$lower, usgdp.forecast.rwf$upper))
## c(usgdp.forecast.rwf$lower, usgdp.forecast.rwf$upper) nie zadziała

plot(usgdp.forecast.rwf)
```

7.2.2. Wykresy wachlarzowe (*fanplots*)

Porównanie przedziałów predykcyjnych dla różnych poziomów ufności

```
usgdp.forecast.rwf.fp <- rwf(usgdp, drift =T, h=20, level=c(0.8, 0.95))
summary(usgdp.forecast.rwf)

## co jest w środku obiektu usgdp.forecast.rwf?:
attributes(usgdp.forecast.rwf)

plot(usgdp.forecast.rwf.fp, main="Przedziały predykcyjne, pu=0,80 i 0,95")
```

7.2.3. Podział na zbiór uczący i testowy

Trik pozwalający na ocenę dokładności prognoz: dzielimy dostępną próbę na dwie części: uczącą (*training set*) oraz testową (*training set*).

Część ucząca jest wykorzystywana w fazie diagnozowania do dopasowania modelu i konstrukcji prognoz. Część testowa jest wykorzystywana do oceny dokładności prognoz.

```
usgdp.train <- window(usgdp, end = c(1999,4))
usgdp.test <- window(usgdp, start = c(2000,1))
```

```
length(usgdp.test) # powinno wypisać 25

### Wyznaczenie prognoz
usgdp.train.f1 <- meanf(usgdp.train, h=25)
usgdp.train.f2 <- naive(usgdp.train, h=25)
usgdp.train.f3 <- rwf(usgdp.train, drift=T, h=25)
usgdp.train.f4 <- rwf(usgdp.train, drift=T, h=25, lambda=0)

### Porównanie prognoz na wykresie:
y.zakres <- c(0.9, 1.1) * range(usgdp, usgdp.test)
par(mfrow = c(4,1))

plot(usgdp.train.f1, ylim=y.zakres)
lines (usgdp.test, col="red", lty=2)

plot(usgdp.train.f2, ylim=y.zakres)
lines (usgdp.test, col="red", lty=2)

plot(usgdp.train.f3, ylim=y.zakres)
lines (usgdp.test, col="red", lty=2)

## zgłasza błąd dla czterech?
plot(usgdp.train.f4, ylim=y.zakres)
lines (usgdp.test, col="red", lty=2)
```

7.2.4. Analiza własności reszt

Jeżeli metoda prognozowania działa dobrze to szereg reszt powinien zachowywać się jak biały szum (brak korelacji).
Nie powinno być widocznych regularności (trend/cykliczność)
histogram, wykres ACF, test losowości Ljung-Boxa

```
reszty.1 <- residuals(usgdp.train.f1)
reszty.2 <- residuals(usgdp.train.f2)
reszty.3 <- residuals(usgdp.train.f3)

par(mfrow = c(3,1))
plot(reszty.1, main = "Reszty f1")
plot(reszty.2, main = "Reszty f2")
plot(reszty.3, main = "Reszty f3")

## ACF i histogramy (normalność)
par(mfrow=c(3,2))
Acf(reszty.1, lag.max=30, main = "Reszty f1")
hist(reszty.1, main = "Reszty f1")

Acf(reszty.2, lag.max=30, main = "Reszty f2")
hist(reszty.2, main = "Reszty f2")

Acf(reszty.3, lag.max=30, main = "Reszty f3")
hist(reszty.3, main = "Reszty f3")

## Test losowości reszt Ljunga-Boxa
Box.test(reszty.1, lag=10, type="Ljung-Box")
```

7.3. Metody naiwne

Prognoza dla chwili t jest równa obserwacji w chwili $t-1$:

$$F_t = Y_{t-1} \tag{17}$$

Wariant: sezonowa metoda naiwna

$$F_t = Y_{t-s} \tag{18}$$

gdzie s – liczba okresów w cyklu sezonowości (dla danych miesięcznych $s = 12$)
Wariant: metoda naiwna z dryfem [ZS.248]:

$$F_{n+h} = Y_n + \frac{h}{n-1}\sum_{t=2}^{n}(Y_t - Y_{t-1}) = Y_n + h(\frac{Y_n - Y_1}{n-1}) \tag{19}$$

Prognoza jest równa ostatniej obserwacji do której dodawana jest średnia zmiana.
Implementacja w R: funkcje naive/snaive z pakietu `forecast`:

```
# standardowa metoda naiwna
AirPass.forecast.naive <- naive(x = AirPass, h = 24)
AirPass.forecast.naive
summary(AirPass.forecast.naive)

accuracy.AirPass.forecast.naive <- accuracy(AirPass.forecast.naive)
accuracy.AirPass.forecast.naive
plot(AirPass.forecast.naive)

# sezonowa metoda naiwna
AirPass.forecast.snaive <- snaive(x = AirPass, h = 24)
```

```
AirPass.forecast.snaive
summary(AirPass.forecast.snaive)

accuracy.AirPass.forecast.snaive <- accuracy(AirPass.forecast.snaive)
accuracy.AirPass.forecast.snaive
plot(AirPass.forecast.snaive)

# metoda uwzględniająca dryf
AirPass.forecast.rwf <- rwf(x = AirPass, drift = TRUE, h = 24)
AirPass.forecast.rwf
summary(AirPass.forecast.rwf)

accuracy.AirPass.forecast.rwf <- accuracy(AirPass.forecast.rwf)
accuracy.AirPass.forecast.rwf
plot(AirPass.forecast.rwf)
```

7.4. Oparte o średniej

W przypadku prostej średniej ruchomej jako prognozę dla chwili t przyjmuje się średnią arytmetyczną n wcześniejszych wartości szeregu:

$$F_t = (Y_{t-1} + Y_{t-2} + \cdots + Y_{t-n})/n \tag{20}$$

Implementacja w R: funkcja `meanf` (pakiet `forecast`):

```
# prognoza oparta na średniej
AirPass.forecast.mean <- meanf (x = AirPass, h = 24)
AirPass.forecast.mean
summary(AirPass.forecast.mean)
plot(AirPass.forecast.mean)
```

7.5. dekompozycja klasyczna [ZS8.5]

...

7.6. modele ARIMA [ZS.267]

```
### prognozowanie na podstawie modeli ARIMA(p,d,q)(P,D,Q)[s]

# model 1: ARIMA(0,1,12)(0,1,0)[12]
ARIMA.model1 <- Arima(AirPass, order = c(0, 1, 12), seasonal = list(order = c(0, 1
ARIMA.model1.prognozy <- forecast(ARIMA.model1, h = 24)
ARIMA.model1.prognozy

# model 2: ARIMA(13,1,0)(0,1,0)[12]
ARIMA.model2 <- Arima(AirPass, order = c(13 ,1 ,0), seasonal = list(order = c(0 ,1
ARIMA.model2.prognozy <- forecast(ARIMA.model2, h = 24)
ARIMA.model2.prognozy

# automatycznie wybrany model ARIMA
ARIMA.model.auto <- auto.arima(AirPass)
ARIMA.model.auto.prognozy <- forecast(ARIMA.model.auto, h = 24)
ARIMA.model.auto.prognozy

# porównanie prognoz
par(mfrow = c(3, 1))
plot(ARIMA.model1.prognozy)
plot(ARIMA.model2.prognozy)
```

```
plot(ARIMA.model.auto.prognozy)
par(mfrow = c(1, 1))
```

7.7. Wygładzanie wykładnicze SES

Załóżmy, że znamy $X_1, X_2, \ldots, X_n$. Chcemy wyznaczyć prognozy dla nieznanych X_{n+h}, gdzie $h = 1, 2, \ldots$

Prognoza jednokrokowa SES F_{t+1} jest określana przez następujące równanie rekurencyjne:

$$F_{n+1} = \alpha X_n + (1 - \alpha) F_n \tag{21}$$

gdzie: F_{n+1} – prognoza na okres $n + 1$; F_n prognoza dla aktualnego okresu czasu, X_n – wartość rzeczywista dla aktualnego okresu czasu; α – stała wygładzająca ($\alpha \in (0, 1)$)

Parametr wygładzający jest dobierany przez prognostę. Jeżeli $\alpha < 0,5$ to mniejsza waga będzie przykładana aktualnej wartości X_n, a większa prognozie tej wartości F_n

Jeżeli $h > 1$ prognoza ma postać (flat forecast):

$$F_{n+h} = F_{n+1}, \quad h = 2, 3, \ldots \tag{22}$$

Wniosek: metoda SES nie nadaje się w przypadku szeregów, w których obserwuje się trend/sezonowość.

```
# proste wygładzanie wykładnicze (algorytm SES)
kurs.EUR.PLN.ses <- ses(kurs.EUR.PLN, initial = "simple", h = 24)
summary(kurs.EUR.PLN.ses)
plot(kurs.EUR.PLN.ses)
```

7.8. Algorytm Holta

Uogólnienie SES dla przypadku szeregu w którym występuje trend:

$$\text{Poziom: } L_t = \alpha X_t + (1-\alpha)(L_{t-1} + b_{t-1}) \quad (23)$$

$$\text{Trend: } b_t = \beta(L_t - L_{t-1}) + (1-\beta)b_{t-1} \quad (24)$$

gdzie: L_t określa poziom, b_t zaś trend w momencie t; α oraz β to parametry wygładzające $(\alpha, \beta \in (0,1).)$

Równanie prognozy:

$$F_{n+h} = L_n + hb_n \qquad \text{dla} h = 1, 2, \ldots \quad (25)$$

Implementacja w R:

```
# algorytm Holta
usgdp.holt <- holt(usgdp, h = 24)
usgdp.holt.damped <- holt(usgdp, h = 20, damped = TRUE)   # trend tłumiony
usgdp.holt.exp <- holt(usgdp, h = 20, exponential = TRUE) # trend wykładniczy

par(mfrow = c(3, 1))
plot(usgdp.holt)
plot(usgdp.holt.damped)
plot(usgdp.holt.exp)
par(mfrow = c(1, 1))
```

7.9. Algorytm Holta-Wintersa

Uogólnienie algorytmu Holta dla przypadku szeregu, w którym występuje oprócz trendu także sezonowość. Dla addytywnej metody H-W mamy:

$$\text{Poziom: } L_t = \alpha(X_t - S_{t-s}) + (1-\alpha)(L_{t-1} + b_{t-1}) \quad (26)$$

$$\text{Trend: } b_t = \beta(L_t - L_{t-1}) + (1-\beta)b_{t-1} \quad (27)$$

$$\text{Sezonowość: } S_t = \gamma(X_t - L_t) + (1-\gamma)S_{t-s} \quad (28)$$

gdzie: s – liczba okresów w cyklu sezonowości (dla danych miesięcznych $s = 12$); α, β, γ to parametry wygładzające ($\alpha, \beta, \gamma \in (0, 1)$.)

Dla multiplikatywnej metody H-W mamy:

$$\text{Poziom: } L_t = \alpha\frac{X_t}{S_{t-s}} + (1-\alpha)(L_{t-1} + b_{t-1}) \quad (29)$$

$$\text{Trend: } b_t = \beta(L_t - L_{t-1}) + (1-\beta)b_{t-1} \quad (30)$$

$$\text{Sezonowość: } S_t = \gamma\frac{X_t}{L_t} + (1-\gamma)S_{t-s} \quad (31)$$

gdzie: s – liczba okresów w cyklu sezonowości (dla danych miesięcznych $s = 12$); α, β, γ to parametry wygładzające ($\alpha, \beta, \gamma \in (0, 1)$.)

Równanie prognozy (wersja addytywna/multiplikatywna):

$$F_{n+h} = L_n + hb_n + S_{n+h-s} \quad \text{dla} h = 1, 2, \ldots \; F_{n+h} = (L_n + hb_n)S_{n+h-s} \quad \text{dla} h = 1, 2, \ldots \quad (32)$$

Implementacja w R:

```
# algorytm Holta-Wintersa
pkb.hw.add  <- hw(pkb, h = 20, seasonal = "additive")       # wariant addytywny
pkb.hw.mult <- hw(pkb, h = 20, seasonal = "multiplicative") # wariant multiplikaty
par(mfrow = c(2, 1))
plot(pkb.hw.add)
plot(pkb.hw.mult)
par(mfrow = c(1, 1))
```

8. Metody sztucznych sieci neuronowych

→ Następne spotkanie

9. Metody niekonwencjonalne

→ Następne spotkanie