ASG - projekt

2023-02-10

Pakiety

```
library(dplyr)
library(forecast)
library(tempdisagg)
```

Tworzenie szeregu

Najpierw wczytujemy dane:

```
dane <-read.csv("dane23.csv")
```

Widzimy, że w naszych danych jest kolumna X, która jest nam zbędna gdyż r samemu indeksuje wiersze. Zatem możemy się jej pozbyć:

```
dane <- dplyr::select(dane, -X)
```

Można również zobaczyć że w oryginalnym pliku dla wiersza nr1 brakuje wpisanej wartości, którą R sam z siebie zamienia na 0. Nie wiemy czy to intecjonalne, dlatego dla bezpieczeństwa usuniemy tą kolumnę:

```
dane <- dane %>% slice(-1)
```

Tak przygotowane dane możemy teraz zamienić w szereg czasowy:

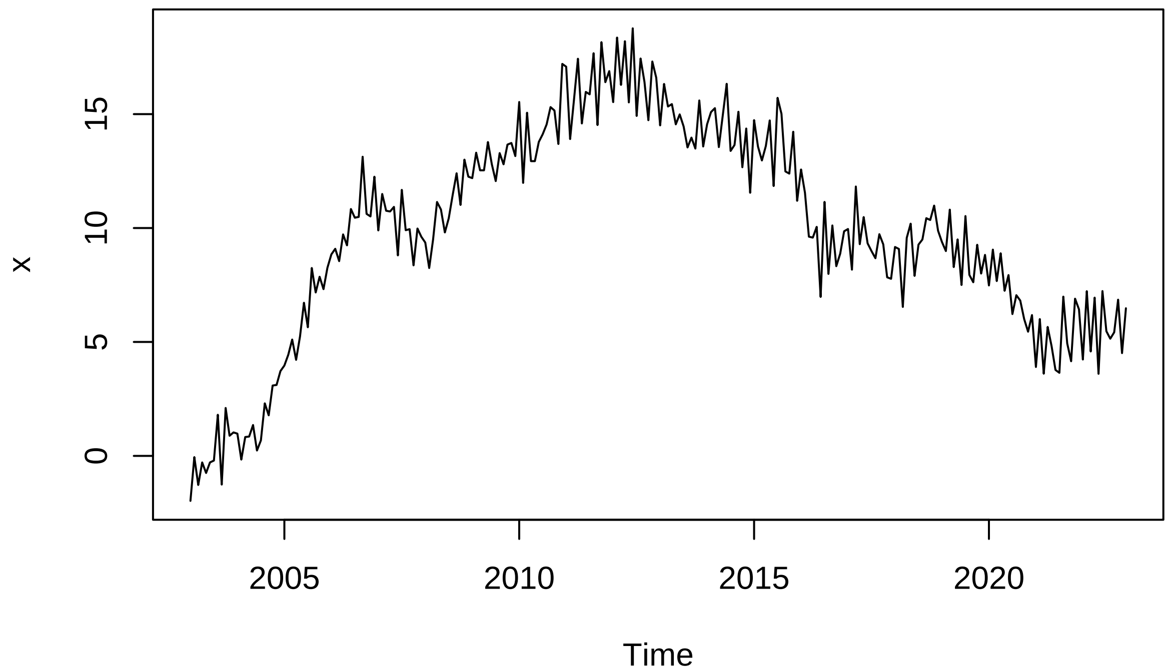
```
szereg <- ts(dane, end = c(2022,12), frequency = 12)
head(szereg)
```

```
##              Jan         Feb         Mar         Apr         May         Jun
## 2003 -1.97428316 -0.05441458 -1.27607199 -0.28721393 -0.75053320 -0.28582558
```

Sprawdzenie stacjonarności

Najpierw wyświetlmy wykres szeregu

```
plot(szereg)
```



Z wykresu widzimy, że szereg nieznacznie zależy od czasu, czyli nie jest stacjonarny.

Możemy upewnić się testem statystycznym z regresją liniową:

```
summary(lm(szereg~time(szereg)))
```

```
##
## Call:
## lm(formula = szereg ~ time(szereg))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.0024  -3.3569   0.0922   3.6862   8.9717
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -153.79866  104.44179  -1.473    0.142
## time(szereg)    0.08129    0.05188   1.567    0.118
##
## Residual standard error: 4.641 on 238 degrees of freedom
## Multiple R-squared:  0.01021,    Adjusted R-squared:  0.00605
## F-statistic: 2.455 on 1 and 238 DF,  p-value: 0.1185
```

Z testu widzimy że na poziomie istotności 0.95 szereg nie zależy od czasu, ale tuż poza naszym poziomem istotności już tak. Zatem trzeba się posłużyć innymi narzędziami.

Wyswietlmy jeszcze wykres szeregu wraz z funkcjami autokorelacji i częściowej autokorelacji:

```r
tsdisplay(szereg)
```

Możemy wywnioskować, że nasz szereg nie ma sezonowości, ale moze mieć trend. By się upewnić użyje funkcji ndiffs i nsdiffs, które zwrócą mi ile razy należy zróznicować szereg by uzyskać szereg stacjonarny:

```r
ndiffs(szereg)
```

```
## [1] 2
```

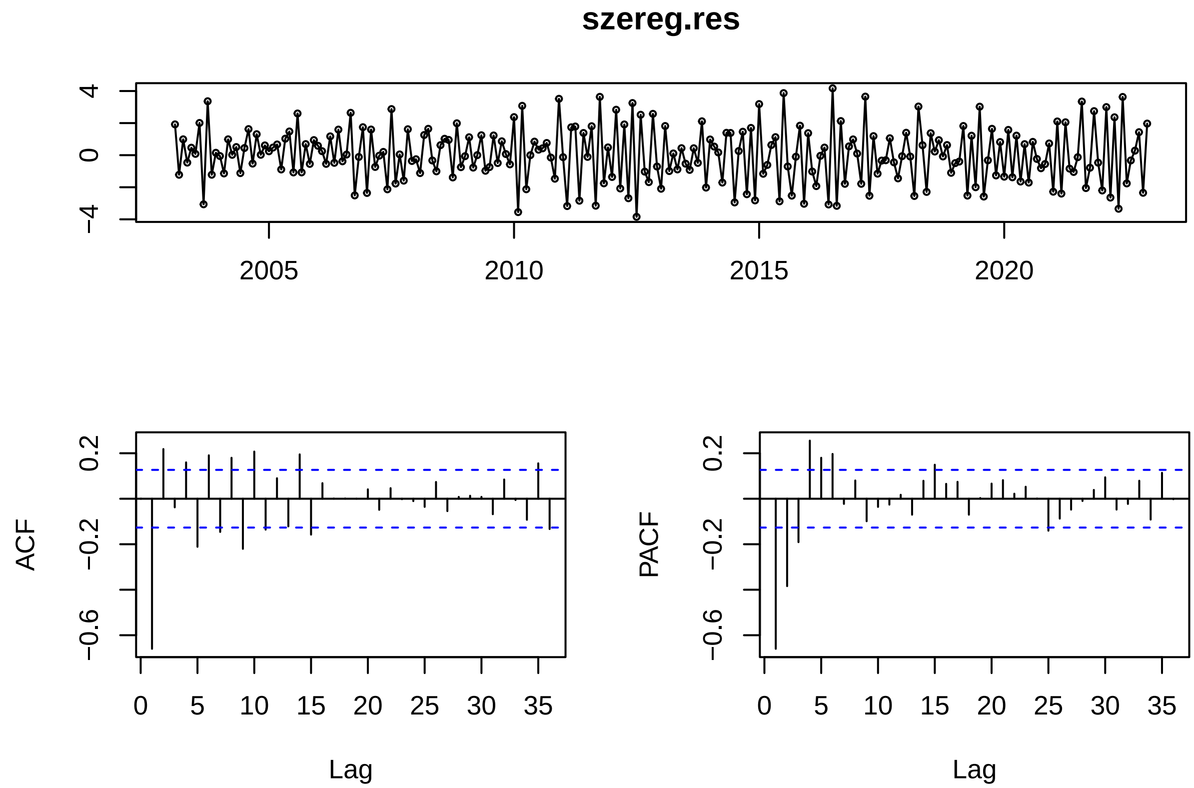
```r
nsdiffs(szereg)
```

```
## [1] 0
```

Funkcja ndiffs zwróciła wartość 1. Oznacza to że nasz szereg ma trend, którego możemy się pozbyć raz róznicując nasz szereg. Druga funkcja nsdiffs dała wartość 0, co oznacza że nasz szereg nie ma sezonowości i nie musimy się jej pozbywać różnicowaniem.

Mając tą wiedzę możemy przejść do stworzenia szeregu stacjonarnego. Zróżnicuje zatem nasz oryginalny szereg i zobaczy czy faktycznie będzie on wtedy stacjonarny:

```r
szereg.res<-diff(szereg)
tsdisplay(szereg.res)
```




```r
summary(lm(szereg.res~time(szereg.res)))
```

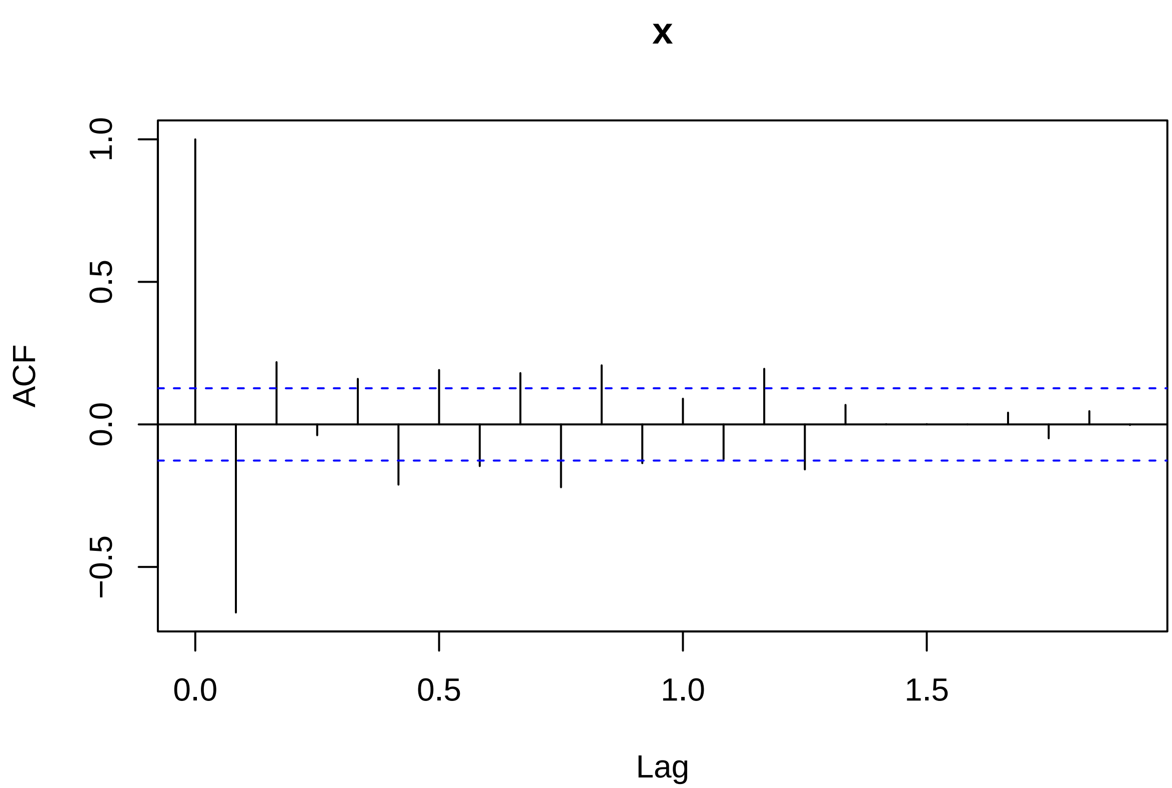
```
##
## Call:
## lm(formula = szereg.res ~ time(szereg.res))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.8886 -1.1612 -0.0798  1.1343  4.2009
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)      38.68887   38.50085   1.005    0.316
## time(szereg.res) -0.01920    0.01913  -1.004    0.316
##
## Residual standard error: 1.7 on 237 degrees of freedom
## Multiple R-squared:  0.004235,   Adjusted R-squared:  3.342e-05
## F-statistic: 1.008 on 1 and 237 DF,  p-value: 0.3164
```

I z wykresu i z testu widać, że raz zróżnicowany szereg jest już stacjonarny.

Funkcje ACF i PACF

Dla stacjonarnego szeregu możemy wyświetlić wykresy autokorelacji ACF i częściowej autokorelacji PACF. Najpierw wyświetlmy ACF:

```
acf(szereg.res)
```



Interesuje nas ostatnia zerowa wartość tej funkcji. Jednak ponieważ są to dane emipryczne, to nasza funkcja nie zawsze będzie dokładnie zerem, tylko jakąś mała, pomijalną liczbą. R ułatwia nam to tworząc przedział, dla którego możemy traktować nasze wartośći ACF jako praktycznie zera, oznaczony niebieskimi liniami. W tym przykłądzie nasza ostatnia niezerowa wartość jest wtedy dla słupka nr 15 (pierwszego słupka nie liczymy bo ACF od 0 to zawsze 1). Czyli możemy z tego wnioskować że nasz szereg możemy opisać modelem MA(15).

Dopasujmy taki model:

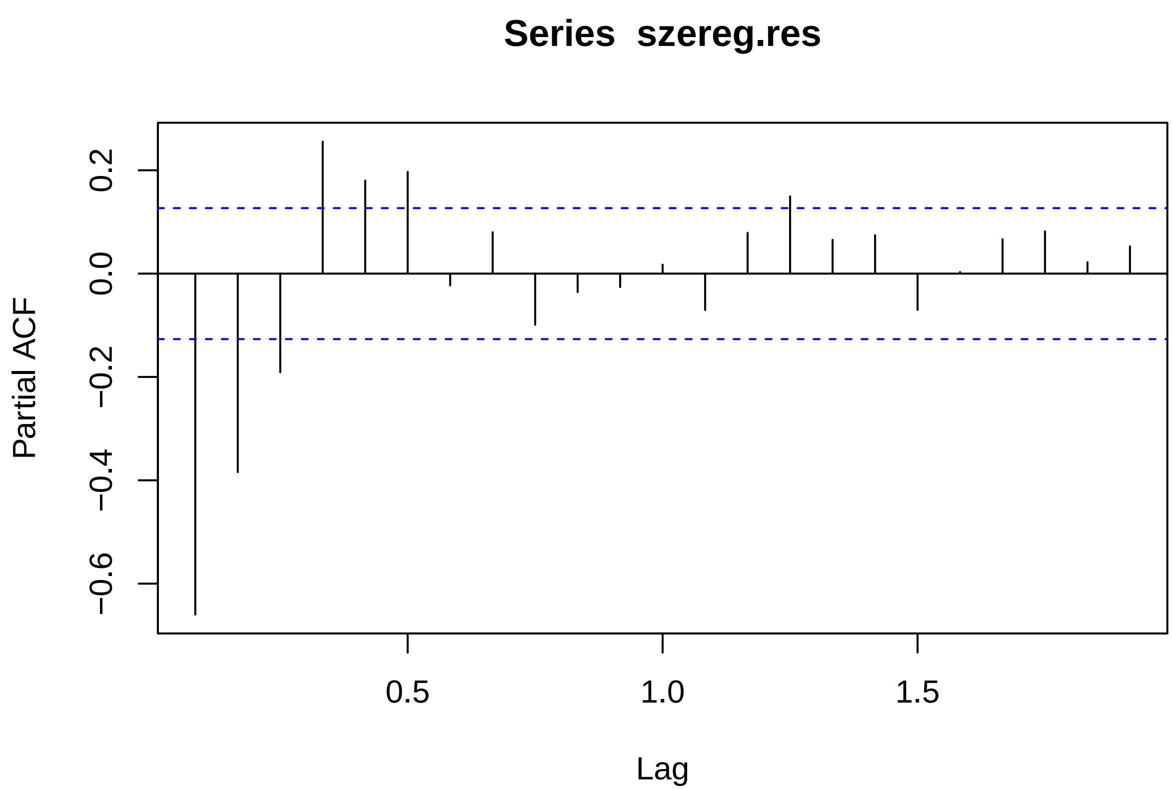
```
model1<-Arima(szereg, order=c(0,1,15))
summary(model1)
```

```
## Series: szereg
## ARIMA(0,1,15)
##
## Coefficients:
##           ma1     ma2     ma3      ma4      ma5     ma6      ma7     ma8
##       -0.9678  0.4558  0.2311  -0.0174  -0.0878  0.2361  -0.1994  0.2741
## s.e.   0.0680  0.0930  0.0940   0.0959   0.0934  0.0937   0.0901  0.0775
```

```
##           ma9    ma10     ma11    ma12     ma13    ma14     ma15
##       -0.3803  0.4647  -0.4213  0.3094  -0.2145  0.1859  -0.1921
## s.e.   0.0995  0.1032   0.0959  0.1031   0.1005  0.1004   0.0797
##
## sigma^2 = 1.098:  log likelihood = -345.73
## AIC=723.46   AICc=725.91   BIC=779.08
##
## Training set error measures:
##                      ME     RMSE       MAE       MPE     MAPE      MASE
## Training set 0.04482199 1.012315 0.8034885 -3.783217 25.47011 0.3854922
##                    ACF1
## Training set -0.0235215
```

Teraz zobaczmy PACF:

```
pacf(szereg.res)
```



Tutaj ostatnia niezerowa wartośc jest również dla słupka nr 15 (tutaj już wliczamy pierwszy słupek). Zatem z tego wykresu możemy przyjąc że nasz szereg można opisać modelem AR(15).

Tutaj też możemy od razu dopasować model:

```
model2<-Arima(szereg, order=c(15,1,0))
summary(model2)
```

```
## Series: szereg
```

```
## ARIMA(15,1,0)
##
## Coefficients:
##           ar1      ar2     ar3     ar4     ar5     ar6     ar7      ar8
##       -1.0174  -0.5161  0.1179  0.5237  0.4486  0.2723  0.0236  -0.0547
## s.e.   0.0638   0.0899  0.0954  0.0958  0.1016  0.1052  0.1056   0.1055
##           ar9     ar10     ar11     ar12    ar13    ar14    ar15
##       -0.2072  -0.1494  -0.1222  -0.0011  0.1067  0.2427  0.1531
## s.e.   0.1062   0.1054   0.1015   0.0960  0.0959  0.0907  0.0652
##
## sigma^2 = 1.146:  log likelihood = -348.92
## AIC=729.83   AICc=732.28   BIC=785.46
##
## Training set error measures:
##                      ME    RMSE       MAE       MPE     MAPE      MASE
## Training set 0.03808218 1.03428 0.8337469 -4.527374 26.84011 0.4000094
##                      ACF1
## Training set -0.009125105
```

Model SARIMA

Do naszego szeregu możemy też dopasować model SARIMA (czyli model ARIMA ale z sezonowością). By to zrobić możemy wykorzystać funkcję auto.arima, która sama dopasuje nam najlepszy model do naszego szeregu, z tym że musimy zmienić wartość parametru seasonal na TRUE by był to model SARIMA:

```
model3<-auto.arima(szereg, seasonal = T)
summary(model3)
```

```
## Series: szereg
## ARIMA(3,2,4)
##
## Coefficients:
##           ar1     ar2      ar3      ma1     ma2     ma3      ma4
##       -0.7380  0.1156  -0.0835  -1.2994  -0.045  0.9798  -0.6106
## s.e.   0.1351  0.1623   0.0873   0.1188   0.151  0.1319   0.0815
##
## sigma^2 = 1.167:  log likelihood = -355.61
## AIC=727.21   AICc=727.84   BIC=754.99
##
## Training set error measures:
##                       ME     RMSE       MAE      MPE     MAPE      MASE
## Training set -0.07497696 1.060013 0.8581921 5.847532 18.37037 0.4117376
##                      ACF1
## Training set -0.003060963
```

Funkcja zwróciła nam model ARIMA(1,1,4).

Diagnostyka Modeli

Mamy dopasowane trzy modele, model1-MA(15), model2-AR(15) i model3-ARIMA(1,1,4). Dla każdego z nich musimy najpierw sprawdzić czy ich reszty są losowe. Możemy to zrobić przy pomocy testu Ljunga-Boxa:

```r
Box.test(residuals(model1), type="Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  residuals(model1)
## X-squared = 0.13445, df = 1, p-value = 0.7139
```

```r
Box.test(residuals(model2), type="Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  residuals(model2)
## X-squared = 0.020235, df = 1, p-value = 0.8869
```

```r
Box.test(residuals(model3), type="Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  residuals(model3)
## X-squared = 0.0022769, df = 1, p-value = 0.9619
```

Dla każdego modelu wartość p jest duża, czyli przyjmujemy hipotezy zerowe, że reszty są losowe w każdym modelu.

Teraz upewnijmy się, że reszty pochodzą z rozkładu normalnego. Posłuży nam do tego test Shapiro-Wilka:

```r
shapiro.test(residuals(model1))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(model1)
## W = 0.99584, p-value = 0.769
```

```r
shapiro.test(residuals(model2))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(model2)
## W = 0.99334, p-value = 0.3615
```

```r
shapiro.test(residuals(model3))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(model3)
## W = 0.98992, p-value = 0.09392
```

Dla dwóch pierwszych modeli wartość p jest bardzo duża, czyli przyjmujemy H0 że reszty pochodzą z rozkładu normalnego. Dla modelu3 wartość p jest bardzo nieprzyjemna do interpretacji (nie jest ani bardzo duża przy przyjąc H0 ani bardzo mała by ją odrzucić) więc dla pewności możemy użyc drugiego testu diagnozującego rozkład reszt, testu Kołmogorowa-Smirnova:

```r
ks.test(residuals(model3), "pnorm")
```

```
##
##  Asymptotic one-sample Kolmogorov-Smirnov test
##
## data:  residuals(model3)
## D = 0.054822, p-value = 0.4664
## alternative hypothesis: two-sided
```

Z tego testu widzimy że rozkład reszt w trzecim modelu również jest normalny.