Univerzita Pardubice
Fakulta ekonomicko-správní

Časové řady v jazyce R

Bc. Tomáš Hrnčíř

Diplomová práce
2018

Univerzita Pardubice
Fakulta ekonomicko-správní
Akademický rok: 2017/2018

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	Bc. Tomáš Hrnčíř
Osobní číslo:	E15652
Studijní program:	N6209 Systémové inženýrství a informatika
Studijní obor:	Pojistné inženýrství Management finančních rizik
Název tématu:	Časové řady v jazyce R
Zadávající katedra:	Ústav matematiky a kvantitativních metod

Zásady pro vypracování:

Cílem diplomové práce je ukázat možnosti využití volně dostupného programovacího jazyku R pro analýzu časových řad, namísto komerčních softwarů jako STATISTICA, EViews, atd. R je programovací jazyk a prostředí určeném pro statistickou analýzu dat a jejich grafické zobrazení. Jde o implementaci programovacího jazyka s volnou licencí. Praktická část se zaměří hlavně na finanční a ekonomické časové řady.

Osnova:
- Časové řady.
- Program R.
- Modelace časových řad v R.
- Porovnání výsledku s výstupy z jiných statistických programů.

Rozsah grafických prací: –

Rozsah pracovní zprávy: cca 50 stran

Forma zpracování diplomové práce: tištěná/elektronická

Seznam odborné literatury:

ARLT, Josef a Markéta ARLTOVÁ. Finanční časové řady: [vlastnosti, metody modelování, příklady a aplikace]. Praha: Grada, 2003. ISBN 80-247-0330-0.
CIPRA, Tomáš : Analýza časových řad s aplikacemi v ekonomii, Alpha:Státní nakladatelství technické literatury, Praha, 1986
Data mining with SPSS modeler: theory, exercises and solutions. ISBN 978-3-319-28707-2.
MAREK, Luboš. Statistika v SPSS: časové řady. Praha: Vysoká škola ekonomická, 1995. ISBN 80-7079-642-1.
NGAI HANG CHAN. Time Series Applications to Finance. Hoboken: John Wiley, 2002. ISBN 9780471461647.

Vedoucí diplomové práce: RNDr. Ján Gogola, Ph.D.
Ústav matematiky a kvantitativních metod

Datum zadání diplomové práce: 1. září 2017

Termín odevzdání diplomové práce: 30. dubna 2018

doc. Ing. Romana Provazníková, Ph.D.
děkanka

L.S.

doc. RNDr. Bohdan Linda
vedoucí ústavu

V Pardubicích dne 1. září 2017

PROHLÁŠENÍ

Prohlašuji, že jsem tuto práci vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako Školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 9/2012, bude práce zveřejněna v Univerzitní knihovně a prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 27. 4. 2018 Bc. Tomáš Hrnčíř

PODĚKOVÁNÍ:

Tímto bych rád poděkoval svému vedoucímu práce RNDr. Jánu Gogolovi, Ph.D. za jeho odbornou pomoc, cenné rady a poskytnuté materiály, které mi pomohly při zpracování diplomové práce. V neposlední řadě bych chtěl poděkovat rodině a přátelům, kteří mě vždy podporovali.

ANOTACE

V této práci bude představen programovací jazyk R a jeho možnost využití při analýze časových dat. Na reálných datech bude použita dekompoziční a Box - Jenkinsovonova metoda, včetně predikce budoucích pozorování. Postup a výsledky obou metod v jazyce R budou porovnány s ostatními komerčně využívanými programy Excel, STATISTICA a Gretl.

KLÍČOVÁ SLOVA

Jazyk R, analýza časových řad, dekompozice časových řad, Box – Jenkinsonova metoda

TITLE

Time series in R language

ANNOTATION

This thesis will introduce programmer language R and its possibility to analyse time series. The decomposition method and the Box- Jenkins method will be used on the basis of real data including prediction of future observations. The procedure and results of both methods in language R will be compared with other commercial programs – Excel, STATISTICA and Gretl.

KEYWORDS

R Language, time series analysis, decomposition of time series, Box – Jenkins method.

OBSAH

ÚVOD **13**

1. ČASOVÉ ŘADY **15**

1.1. DĚLENÍ ČASOVÝCH ŘAD 15
1.2. POROVNATELNOST HODNOT ČASOVÉ ŘADY 16
1.3. ZÁKLADNÍ CHARAKTERISTIKY ČASOVÝCH ŘAD 16
1.3.1. Popisné statistiky *16*
1.3.2. Míry dynamiky *17*
1.4. PŘEDPOKLADY FINANČNÍCH ČASOVÝCH ŘAD 18
1.4.1. Leptokurtické rozdělení *18*
1.4.2. Shlukování volatility *18*
1.4.3. Pákový efekt *18*

2. ZÁKLADNÍ PŘÍSTUPY K ANALÝZE ČASOVÝCH ŘAD **19**

2.1. DEKOMPOZICE ČASOVÉ ŘADY 19
2.1.1. Modelování trendové složky pomocí trendových funkcí *20*
2.1.2. Modelování trendové složky pomocí klouzavých průměrů *20*
2.1.3. Míry přesnosti prognóz *21*
2.1.4. Modelace sezonní složky *22*
2.2. BOX-JENKINSONOVA METODA 23
2.2.1. Stochastický proces *23*
2.2.2. Stacionarita *23*
2.2.3. Autokorelační funkce (ACF) *24*
2.2.4. Parciální autokorelace (PACF) *25*
2.2.5. Proces bílého šumu *25*
2.2.6. Lineární proces *26*
2.2.7. Lineární modely stacionárních časových řad *26*
2.2.8. Modely nestacionárních časových řad *31*
2.2.9. Konstrukce modelu pomocí Box-Jenkinsonovy metody *32*

3. PROGRAMOVACÍ JAZYK R **34**

3.1. ZÁKLADNÍ ROZHRANÍ R 35
3.2. PROMĚNNÉ 38
3.2.1. Přiřazení proměnných *38*
3.3. DATOVÉ TYPY 40
3.4. VEKTOR 41
3.4.1. Faktory *42*
3.5. MATICE 42
3.6. POKROČILÉ DATOVÉ STRUKTURY 43
3.6.1. Datové tabulky *43*
3.6.2. Seznam *45*
3.7. GRAFICKÉ MOŽNOSTI V R 46
3.7.1. High-level grafika *46*
3.7.2. Low-level grafika *50*
3.7.3. Funkce par() *50*

4. DEKOMPOZICE ČASOVÉ ŘADY V R **53**

4.1. TRENDOVÁ SLOŽKA 53
4.1.1. Lineární trendová funkce *55*
4.1.2. Kvadratická trendová funkce *56*
4.2. SEZONNÍ SLOŽKA 59
4.3. REZIDUÁLNÍ (NÁHODNÁ) SLOŽKA 60
4.4. PREDIKCE DO BUDOUCNA 61
4.1. FUNKCE *DECOMPOSE* 63

5. BOX-JENKINSOVONOVA METODA V R 63

5.1. MODELACE DAT 64
5.2. PREDIKCE BUDOUCÍCH HODNOT 70
5.3. FUNKCE *AUTO.ARIMA()* 71

6. ČASOVÉ ŘADY V MICROSOFT EXCEL 73

6.1. DEKOMPOZICE V PROGRAMU EXCEL 73
6.1.1. Modelace trendové složky v Excelu 73
6.1.2. Modelace sezonní složky 76
6.1.3. Výpočet reziduální složky 76
6.1.4. Predikce budoucích hodnot 77
6.2. BOX – JENKINSONOVA METODA V EXCELU 78
6.2.1. Výstavba modelu ARIMA 78

7. ČASOVÉ ŘADY V PROGRAMU GRETL 80

7.1. DEKOMPOZIČNÍ METODA V GRETL 80
7.1.1. Predikce budoucích hodnot 82
7.2. BOX – JENKINSONOVA METODA V PROGRAMU GRETL 83
7.2.1. Predikce budoucích hodnot 85

8. MODELACE ČASOVÝCH ŘAD V PROGRAMU STATISTICA 87

8.1. DEKOMPOZIČNÍ METODA V PROGRAMU STATISTICA 87
8.2. BOX – JENKINSONOVA METODA V PROGRAMU STATISTICA 89

ZÁVĚR 91

POUŽITÁ LITERATURA 95

SEZNAM TABULEK –

Tabulka 1:Popis ACF a PACF pro výběr model AR,MA, ARMA 33
Tabulka 2: Jednotlivé barvy v programu R 51

SEZNAM OBRÁZKŮ

Obrázek 1: Korelogram Autokorelační funkce 25
Obrázek 2: ACF a PACF AR (1) 28
Obrázek 3: ACF a PACF MA (1) 30
Obrázek 4: Příkazový řádek v jazyce R 35
Obrázek 5: Základní rozhraní programu R 36
Obrázek 6: Vyvolána funkce *Help* 37
Obrázek 7: Přiřazení proměnných 39
Obrázek 8: Přiřazení vektorů 42
Obrázek 9: Základní graf návštěvnosti 48
Obrázek 10: Jednotlivé možnosti tvorby grafů 48
Obrázek 11:Sloupcový graf návštěvnosti 49
Obrázek 12: Jednotlivé znaky pro vykreslení bodů a typy čar 52
Obrázek 13:Graf časové řady výroby australského piva 53
Obrázek 14: Porovnání sezonně očištěných dat s původními 54
Obrázek 15:Lineární trendová funkce a výsledná rezidua 56
Obrázek 16:Porovnání lineárního a kvadratického modelu 58
Obrázek 17: Graf reziduí odhadů trendové funkce 58
Obrázek 18: Graf sezonní složky 59
Obrázek 19: Průměrná sezonní složka 60
Obrázek 20: Reziduální složka 60
Obrázek 21: Predikce budoucího vývoje 61
Obrázek 22: Predikce trendové složky 62
Obrázek 23: Predikce budoucího vývoje 62
Obrázek 24: Graf funkce decompose 63
Obrázek 25: Graf indexu NASDAQ Composite 64
Obrázek 26: Logaritmická tranforamce NASDAQ 65
Obrázek 27: ACF a PACF logarimu NASDQ indexu 66
Obrázek 28: Graf první diference logaritmických hodnot NASDAQ 66
Obrázek 29:ACF a PACF logaritmických hodnot NASDAQ 67
Obrázek 30: ACF a PACF modelu ARIMA(2,0,1) 68
Obrázek 31: ACF a PACF modelu ARIMA(1,0,1) 69
Obrázek 32: Predikce budoucích hodnot na základě modelu ARIMA(1,0,1) 70
Obrázek 33: ACF a PACF modelu ARIMA(1,1,2) 72
Obrázek 34: Graf výrovy piva v austrálii 73
Obrázek 35: Trendová složka 74
Obrázek 36: Výsledek odhadu lineárního trendu 74
Obrázek 37: Výsledek odhadu kvadratického trendu 75
Obrázek 38: Porovnání jednotlivých trendů 75
Obrázek 39: Přidání spojnice trendu do grafu 76
Obrázek 40: Modelace sezonní složky 76
Obrázek 41:Predikce budoucích hodnot 77
Obrázek 42: Porovnání skutečných hodnot a modelu 77
Obrázek 43: XLMiner Platform 78
Obrázek 44: ACF a PACF v XLMiner Platform 78
Obrázek 45: ACF a PACF LN(NASDAQ) 79

Obrázek 46: Predikce budoucích hodnot 79
Obrázek 47: Graf časové řady výroby piv v Gretl 80
Obrázek 48: Odhad trendu v programu Gretl 81
Obrázek 49: Odhad kvadratického trendu v Gretl 81
Obrázek 50: Porovnání trendových funkcí 82
Obrázek 51: Predikce budoucích hodnot v Gretl 83
Obrázek 52: Výstup funkce Korelogramv Gretl 83
Obrázek 53: ACF a PACF modelu ARIMA(2,0,2) 84
Obrázek 54: Odhad modelu ARIMA (2,0,2) 84
Obrázek 55: Odhad modelu ARIMA(1,0,1) 85
Obrázek 56: ACF a PACF modelu ARIMA(1,0,1) 85
Obrázek 57: Predikce budoucích hodnot na základě modelu ARIMA (1,0,1) 86
Obrázek 58: Dekompozice v STATISTICA 87
Obrázek 59: Výstup z regrese 88
Obrázek 60: Predikce buduoucích hodnot v STATISTICA 88
Obrázek 61: ACF a PACF NASDAQ 89
Obrázek 62: ACF a PACF modelu ARIMA (1,0,1) 89
Obrázek 63: Predikce buoucích hodnot na základě modelu ARIMA (1,0,1) 90

SEZNAM ZKRATEK

ACF	Autokorelační funkce
AR	Autoregresní proces
B-J	Box – Jenkinsonova metoda
MA	Moving average
NASDAQ	National Association of Securities Dealers Automated Quotations
PACF	Parciální autokorelační funkce

ÚVOD

Časové řady nás obklopují ze všech stran a většina ekonomických rozhodování je prováděna na základě jejich analýzy. Proto je nutné porozumět jednotlivým zákonitostem, které je ovlivňují, dříve než uděláme rozhodnutí. Tyto vztahy dále můžeme využívat i při odhadu vývoje budoucích pozorování. K analýze časových řad je možné využít velké množství programů, z nichž všechny nejsou poskytovány zdarma.

V posledních letech začíná vzrůstat popularita jazyka R, který je tzv. freeware, tedy software, který je možné zdarma stahovat z internetu. Tento jazyk je určen zejména pro statistickou analýzu a následnou grafickou prezentaci. Vzrůstající popularita způsobila, že komerčně využívané softwary jako například STATISTICA nebo SPSS umožňují integraci tohoto jazyka do svého softwaru.

Cílem práce je seznámit uživatele nejprve se základním ovládáním jazyka R a následným využitím při analýze časových řad. Následně porovnat průběh analýzy časových řad s komerčními statistickými programy jako Microsoft Excel, Statistica a Gretl.

V této práci budou nejdříve popsány teoretické zákonitosti analýzy časových řad, jejich dělení a základní charakteristiky. Protože pro finanční časovou řadu platí jiné předpoklady než pro nefinanční časovou řadu, budou tyto vlastnosti vysvětleny. Poté objasníme standartní metody analýzy časových řad. Dekompoziční metoda i Box - Jenkinsonova metodologie budou následně použity v praktické části při modelaci časových řad.

Další část práce je zaměřena na základní principy - práce v jazyce R. Bude vysvětleno základní rozhraní programu a práce s jednotlivými proměnnými a datovými typy, které je možné v programu používat. Nebo podstatnou částí programu je práce s knihovnami, kde vysvětlíme jejich instalaci a využití. Jedna z největších předností programu R je zpracování grafů. Zaměříme se nejen na tvorbu grafů tzv. High-level grafiku,ale i na tak i na úpravu již vytvořených grafů tzv. Low – level grafiku.

V praktické části se nejprve podrobněji zaměříme na dekompoziční metodu v jazyce R. Modelace touto metodou bude probíhat na datech, která jsou součástí databáze jazyka R, takže každý čtenář je schopnen vyzkoušet si také modelaci sám. Nejprve bude ukázána postupná dekompoziční metoda a nakonec, jak je možné v jazyce R dělat automatickou dekompozici obdobně jako v komerčně využívaném SPSS Modeleru. Pro modelaci pomocí Box – Jenkinsonovi metody budou použita reálná skutečná data indexu NASDAQ.

V poslední části práce bude na totožných datech provedena modelace pomocí známějších softwarů MS Excel, Statistica a Gretl. Při těchto modelacích ukážeme rozdílné požadavky na jednotlivé vstupní hodnoty a jiný postup při tvorbě analýz. Na závěr také budou vysvětleny jednotlivé výhody a nevýhody při využití těchto softwarů.

1. ČASOVÉ ŘADY

Za ekonomickou časovou řadu považujeme řadu hodnot jistého věcného a prostorově vymezeného ekonomického ukazatele, která je uspořádána v čase směrem od minulosti do přítomnosti. S časovými řadami se setkáváme v ekonomii, fyzice, mechanice nebo v meteorologii.

Ekonomické časové řady jsou charakteristické:

- Trendem
- Sezónností
- Podmíněnou heteroskedasticitou
- Nelinearitou
- Společnými vlastnostmi více časových řad

Tyto výše zmíněné vlastnosti se většinou neobjevují najednou, ale jejich přítomnost je závislá na typu časové řady např. podmíněná heteroskedasticita se objevuje u vysokofrekvenčních řad. [4]

1.1. Dělení časových řad

Jednotlivé časové řady můžeme rozlišovat podle časového omezení na **intervalové** nebo **okamžikové**. Intervalové časové řady jsou řadami ukazatelů, jejichž hodnota závisí na celém sledovaném intervalu. Typickým příkladem může být objem výroby, výnosy či náklady nebo spotřeba energie atd. Okamžikové časové řady jsou řadami ukazatelů, jejichž hodnoty se vztahují ke konkrétnímu časovému okamžiku, na který nemá vliv délka pozorování ani předchozí hodnoty. Příkladem této řady může být měření venkovní teploty v konkrétní hodinu, počet studentů k začátku školního roku nebo návštěvnost sportovních událostí.

Časové řady je možno také charakterizovat podle délky intervalu na **krátkodobé** a **dlouhodobé** časové řady. Dlouhodobé časové řady jsou charakteristické ročním nebo delším časovým úsekem intervalem pozorování. Krátkodobé časové řady mají interval pozorování menší než jeden rok, pokud časový interval pozorování je kratší, než jeden týden jsou tyto řady nazývány vysokofrekvenční.

Dále je možné rozlišovat podle sledovaných ukazatelů na **absolutní** a **odvozené**. Absolutní časové řady neboli primární pracují s přímo zjišťovanými ukazateli. Naopak odvozené pracují

zejména s kumulací nebo poměrem primárních ukazatelů např. průměrná doba čekání u lékaře nebo kumulované náklady. [2]

1.2. Porovnatelnost hodnot časové řady

Abychom mohli porovnávat více časových řad, je důležité, aby jednotlivé řady splňovali následující podmínky:

a) **Homogenita časové řady** – její hodnoty za delší časové období jsou porovnatelné z prostorového, časového a věcného hlediska.

b) **Prostorové vymezení** - uskutečňuje se vymezením hranic regionu, v rámci kterého se zjišťují hodnoty ukazatele

c) **Věcné vymezení** - vyžaduje, aby ukazatel byl jednoznačně definovaný spolu s měrnou jednotkou. Problém toho vymezení je zejména u dlouhých časových řad, na které působí vliv nových technologii nebo inovací výroby

d) **Časové vymezení** – hodnoty proměnné nebo ekonomického ukazatele je třeba zjišťovat za stejně dlouhé období (nejčastěji problém při měsíčních intervalech, kdy každý měsíc má jiný počet dnů). Řešením je očistit časovou řadu od kalendářních variací podle vzorce:

$$y_t{}^{(0)} = y_t \frac{\bar{d}_t}{d_t}, \tag{1.1}$$

$\bar{d}_t$ = průměrný počet kalendářních dnů v dílčím období

d_t = počet kalendářních dnů v příslušném období

1.3. Základní charakteristiky časových řad

Základní charakteristiky časových řad, jsou takové charakteristiky, které vyžadují vždy stejnou délku jednotlivých pozorování.

1.3.1. Popisné statistiky

Časové řady se velmi často sledují např. hodinově, denně, měsíčně nebo ročně atd. Pro lepší interpretaci a modelaci se velmi často využívá tzv. **agregace hodnot časové řady**, což můžeme interpretovat jako vytváření z hodinových hodnot denní, z měsíčních roční apod., kdy rozlišujeme intervalovou časovou řadu a okamžitou.

a) Intervalová časová řada – součet nebo prostý aritmetický průměr

$$\bar{y} = \frac{\sum_{t=1}^{T} y_t}{T} \tag{1.2}$$

b) Okamžiková časová řada - chronologický průměr

- **Prostý –** pokud vzdálenost mezi jednotlivými okamžiky sledování je konstantní

$$\bar{y} = \frac{\frac{y_1+y_2}{2}+\frac{y_2+y_3}{2}+\cdots+\frac{y_{T-1}+y_T}{2}}{T-1} = \frac{\frac{1}{2}y_1+\sum_{t=1}^{T-1} y_t+\frac{1}{2}y_T}{T-1} \tag{1.3}$$

- **Vážený –** pokud není konstantní vzdálenost mezi jednotlivými okamžiky pozorování

$$\bar{y} = \frac{\frac{y_1+y_2}{2}d_1+\frac{y_2+y_3}{2}d_2+\cdots+\frac{y_{T-1}+y_T}{2}d_{T-1}}{d_1+d_2+\cdots+d_{T-1}}, \tag{1.4}$$

kde d_t, $t = 1,\ldots,T\text{-}1$, jsou délky jednotlivých časových intervalů. [1],[2]

1.3.2. Míry dynamiky

Míry dynamiky nám umožnují charakterizovat základní rysy chování časových řad a přizpůsobit kritéria pro jejich modelaci

a) Absolutní diference (první diference) – vyjadřuje, jak se změní hodnota v čase *t* ve srovnání s *t* - 1

$$\Delta y_t = y_t - y_{t-1} \;\; t = 2,3,\ldots T. \tag{1.5}$$

Diferencováním první diference lze získat druhou diferenci

$$\Delta^2 y_t = \Delta y_t - \Delta y_{t-1}, t = 3,4,\ldots,T \tag{1.6}$$

analogicky lze postupovat při tvorbě třetí a vyšší diference.

a) Průměrný absolutní přírůstek – aritmetický průměr absolutní diference za celou časovou řadu

$$\bar{\Delta} = \frac{\sum_{t=2}^{T} \Delta y_t}{T-1} = \frac{y_T - y_1}{T-1} \tag{1.7}$$

b) Koeficient růstu (tempo růstu) **–** po vynásobení stem udává, o kolik procent se změnila hodnota v čase t ve srovnání s hodnotou v čase *t* – 1

$$k_t = \frac{y_t}{y_{t-1}}, t = 2,3,\ldots,T. \text{ [2]} \tag{1.8}$$

1.4. Předpoklady finančních časových řad

Hlavní znakem finančních časových řad je velmi vysoká časová frekvence jednotlivých hodnot, které jsou nejčastěji zaznamenávány v denní frekvenci. Vliv na dynamiku časových řad mají systematické a značný vliv mají i nesystematické faktory, které mají vysokou variabilitu a proměnlivost.

1.4.1. Leptokurtické rozdělení

Velmi často se v praxi využívá logaritmování měr zisku r_t finančních aktiv jako jsou například akcie nebo cenových indexů

$$r_t = \ln \frac{P_t}{P_{t-1}} \tag{1.9}$$

kde P_t je cena příslušného finančního aktiva v čase t, a P_{t-1} je cena v čase t-1. Nejdůležitější faktor pro logaritmování je, že ceny žádného aktiva nemohou být záporné a z toho důvodu se předpokládá logaritmicko – normální rozdělení jejich logaritmů. V praxi se ovšem často ukáže, že časové řady logaritmů výnosů r_t mají rozdělené špičatější v porovnání s normálním a četnost výskytů extrémně vysokých hodnot je vyšší než za předpokladu normality.

1.4.2. Shlukování volatility

Volatilita finančních časových řad se v průběhu času mění, zejména díky nejistotě na trhu. Období s velkou volatilitou jsou střídána s obdobími velmi malou volatilitou. Logaritmus výnosů má poté rozdělení s rozptylem, který se mění v čase a můžeme ho označit za podmíněně heteroskedastický. Nepodmíněné rozdělení logaritmů výnosů je potom směsicí rozdělení, kdy ta s malým podmíněným rozptylem koncertují hodnoty v blízkosti střední hodnoty a ostatní s velkým podmíněným rozptylem naopak posouvají výnosy do konců rozdělení, díky čemuž je výsledkem nepodmíněné špičaté rozdělení s „tlustými“ konci.

1.4.3. Pákový efekt

Pákový efekt znamená, že negativní šoky na trhu působí na volatilitu například akcie firmy a na trhu nastane pozitivní šok, cena akcie začne narůstat a my jakožto investoři jsme spokojeni s touto změnou. V opačném případě, kdy nastane negativní šok a cena akcie začne klesat, investoři začnou přehodnocovat svou pozici a mohou začít prodávat a naopak nový investoři mohou nakupovat cenu akcie, a je způsobena větší volatilita. Důležité je také zveřejňování důležitých informací, jako například výplata dividend, i zde platí větší vliv na volatilitu špatných zpráv než pozitivních. [1]

2. ZÁKLADNÍ PŘÍSTUPY K ANALÝZE ČASOVÝCH ŘAD

Výběr správné metody závisí na celé řadě faktorů, které je třeba zohlednit při rozhodování jako například účel analýzy, typ sledované řady, dostupnost výpočetní techniky a softwaru a v neposlední řadě na zkušenosti analytika. V této kapitole budou představeny dva nejvíce používané typy: dekompoziční metoda a Box-Jenkinsonova metoda.

2.1. Dekompozice časové řady

Klasická analýza časových řad předpokládá, že časovou řadu y_t pro $t = 1,2, \ldots, T$ je možné rozložit na čtyři složky: trendovou, cyklickou, sezónní a nesystematickou složku (reziduální). Výhoda této metody spočívá zejména v přesnosti a možnosti dají se lehce používat v dlouhém období, nicméně jsme velmi omezeni v modelaci a odhadu náhodné složky. Časová řada nemusí obsahovat všechny výše uvedené složky najednou, proto provádíme dekompozici a snažíme se identifikovat pravidelné chování. Dekompoziční metoda pracuje pouze se systematickými složkami (trendovou, sezónní a cyklickou složkou) a zpravidla při tom využívá regresní analýzy. [1]

Trendová složka (T_t) představuje dlouhodobé změny v průměrném chování řasové řady, respektive obecnou tendenci vývoje zkoumaného jevu za dlouhé období, ve kterém na něho působí řada faktorů ve stejném směru (ekonomické podmínky, technologie, demografické podmínky). Trend může nabývat různých podob, může být rostoucí, klesající, strmý nebo mírný, a v průběhu času se může měnit a díky tomu ho můžeme považovat za cyklus. Například pokud sledujeme na výdaje na platy zaměstnanců v libovolné odvětví, velký vliv tuto položku má stoupající hodinová mzda všech zaměstnanců.

Cyklická složka (C_t) představuje kolísání okolo trendu, ve kterých se střídají období růstu a poklesu. Jednotlivé cykly se vytvářejí za období delší než jeden rok a mají nepravidelný charakter. U ekonomických časových řad souvisí cyklická složka se střídáním hospodářských cyklů, ale můžou na ně působit i neekonomické faktory jako například inovace, či demografické změny ve společnosti.

Sezonní složka (S_t) vyjadřuje periodické změny v chování časové řady okolo trendu v rámci kalendářního roku a opakují se každoročně ve stejných obdobích. Typickým příkladem je vliv střídání ročního období. Nejčastěji se projevuje u čtvrtletních časových řad a měsíčních.

Nesystematická (reziduální) složka (ε_t) představuje náhodné pohyby v průběhu časové řady, které nejsou systematické jako u výše zmíněných složek a proto není možné rozpoznat

její charakter. Zahrnuje také chyby v měření, zaokrouhlování a ostatní chyby ve statistickém zpracování dat.

Nejčastěji je při analýze předpokládán **aditivní model** popisu chování řady, ve kterém se jednotlivé složky sčítají a platí:

$$y_y = T_t + S_t + C_t + \varepsilon_t \tag{2.1}$$

Tento model se používá, pokud variabilita časových hodnot je přibližně konstantní v čase. Po této dekompozici jsou jednotlivé složky ve stejných jednotkách jako původní časová řada.

Druhý model je **multiplikativní**, kde na rozdíl od předchozího modelu je předpoklad, že jednotlivé složky jsou vyjádřeny součinem:

$$y_y = Tt \cdot St \cdot Ct \cdot \varepsilon_t \tag{2.2}$$

Po multiplikativní dekompozici je trendová složka časové řady ve stejných měrných jednotkách jako původní časová řada, ale zbylé složky (sezonní, cyklická a náhodná) jsou v relativním vyjádření. Využití této dekompozice se používá v případě, kdy variabilita časové řady se mění v čase. [2],[5]

2.1.1. Modelování trendové složky pomocí trendových funkcí

Trendová funkce pro modelování trendu se používá, pokud časová řada odpovídá určité časové funkci např. lineární, kvadratické, exponenciální atd. Pro použití této techniky vycházíme z toho, že časová řada je uspořádána v čase a ve stejně dlouhých intervalech. Jednotlivé parametry se odhadují metodou nejmenších čtverců. Při exponenciální nebo S-křivce se využívá logaritmování k dosažení linearizace. Pro posouzení kvality vybraného modelu se používají tzv. míry přesnosti popsané v kapitole 2.1.3.

2.1.2. Modelování trendové složky pomocí klouzavých průměrů

Populárnější metodou pro modelaci trendové složky je metoda klouzavých průměrů. Tato metoda se využívá zejména, je-li vývoj řady nerovnoměrný v důsledku vlivu nesystematické složky či má extrémní hodnoty. Jak název napovídá podstata je, že posloupnost jednotlivých hodnot pozorování nahradíme řadou průměrů vypočítaných z těchto hodnot. Při výpočtu průměru postupujeme (kloužeme) vždy o jedno pozorování dopředu, tedy nejstarší pozorování vypouštíme a nahrazujeme ho novým pozorováním. Velmi důležitá je volba počtu pozorování, která záleží na zkušenosti analytika, který řadu zpracovává.

Klouzavé průměry můžeme také rozdělovat na jednoduché, vážené a centrované. Jednoduché klouzavé průměry počítají, že všechna pozorování pro nás mají stejnou váhu.

Vážené klouzavé průměry používáme v případě, že jednoduché nám nepřineslo očekávaný výsledek. Jak název napovídá každému pozorování je přiřazena určitá váha. Centrované klouzavé průměry se využívají, pokud časová řada obsahuje sezónní složku. Využívají se nejčastěji čtvrtletní nebo měsíční centrované průměry.

Speciální případem metody klouzavých průměrů je exponenciální vyrovnávání, kdy váha starších pozorování exponenciálně klesá. Tedy říkáme, že čím aktuálnější pozorování tím je pro nás důležitější. [4],[6]

2.1.3. Míry přesnosti prognóz

Výsledkem extrapolace časové řady jsou odhady zpravidla určené na základě odhadu parametrů určitého matematického modelu, jehož přesnost by měla být potvrzena. V případě přesného kvalitního modelu je předpoklad, že prognózy se od skutečnosti nebudou příliš mnoho lišit. Za delší časový úsek se přesnost odhadů hodnotí pomocí různých průměrných charakteristik, které mají smysl, pokud máme minimálně 5 období. Předpokládejme, že je známo n odhadů $\hat{y}_t$ a n skutečných hodnot y_t časové řady pro $t = 1, 2, \ldots, n$. Chybou odhadu nazveme rozdíl $e_t = y_t - \hat{y}_t$.

a) **Průměrná chyba ME** (mean error) - vyjadřuje průměrnou odchylku odhadu hodnot a skutečných hodnot. Pokud je kladná, model systematicky podhodnocuje skutečnost, pokud je záporná model nadhodnocuje skutečnost. Je vyjádřena ve stejných jednotkách jako časová řada.

$$ME = \frac{1}{n}\sum_{t=1}^{n}(y_t - \hat{y}_t) = \frac{1}{n}\sum_{t=1}^{n} e_t. \tag{2.3}$$

b) **Průměrná absolutní chyba MAE** (mean absolute error) – Vyjadřuje průměrnou absolutní chybu skutečných hodnot od odhadnutých hodnot a je ve stejných měrných jednotkách jako časová řada.

$$MAE = \frac{1}{n}\sum_{t=1}^{n}|y_t - \hat{y}_t| = \frac{1}{n}\sum_{t=1}^{n}|e_t|. \tag{2.4}$$

c) **Průměrná čtvercová chyba – rozptyl chyb MSE** (mean square error) – Používá se při porovnání rozptylu chyb získaných matematickými modely, s cílem vybrat model s nejnižší hodnotou MSE, který je nejpřesnější. Je velmi citlivá na velké odchylky. Při použití druhé odmocniny dostáváme směrodatnou odchylku chyb prognóz, která se značí RMSE (root mean square error).

$$MSE = \frac{1}{n}\sum_{t=1}^{n}(y_t - \hat{y}_t)^2 = \frac{1}{n}\sum_{t=1}^{n} {e_t}^2. \tag{2.5}$$

d) **Průměrná procentuální chyba MPE** (mean percentage error) – stejně jako průměrná chyba vyjadřuje průměrnou odchylku mezi skutečnou hodnotou a naším odhadem v procentech. Také pokud je kladná, náš model systematicky podhodnocuje skutečnost a platí to samozřejmě i naopak.

$$MPE = \frac{1}{n}\sum_{t=1}^{n}\frac{(y_t-\hat{y}_t)}{y_t}\cdot 100\% = \frac{1}{n}\sum_{t=1}^{n}\frac{e_t}{y_t}\cdot 100\%. \quad (2.6)$$

e) **Průměrná absolutní procentuální chyba MAPE** (mean absolute percentage error) – vyjadřuje v procentech průměrnou velikost chyb skutečných hodnot od našich odhadů.

$$MAPE = \frac{1}{n}\sum_{t=1}^{n}\frac{|y_t-\hat{y}_t|}{y_t}\cdot 100\% = \frac{1}{n}\sum_{t=1}^{n}\frac{|e_t|}{y_t}\cdot 100\% \quad (2.7)$$

f) **Koeficient determinace R^2 -** velmi jednoduše říká, jakou část z celkové variability ve výsledné proměnné jsme pomocí odhadovaného modelu dokázali vysvětlit. Může nabývat pouze hodnoty ⟨0,1⟩. Čím blíže hodnotě 1, tím model lépe vystihuje trend časové řady a naopak.

$$R^2 = 1 - \frac{\sum_{t=1}^{T}(y_t-\hat{y}_t)^2}{\sum_{t=1}^{T}(y_t-\bar{y})^2} \quad (2.8)$$

Nedostatkem je závislost na počtu parametrů modelu - Tento nedostatek je odstraněn v modifikovaném indexu determinace, který je definován jako

$$R_M^2 = R^2 - \frac{(1-R^2)*(1-R)}{n-l}, \quad (2.9)$$

kde *l* je počet parametrů modelu trendové funkce a *n* je počet pozorování. [2],[5],[6]

2.1.4. Modelace sezonní složky

Jak bylo výše uvedeno, sezónní složka popisuje periodické změny časové řady, které se odehrávají za periodu kratší než je jeden kalendářní rok a pravidelně se opakují. Její eliminace se často označuje jako tzv. sezónní očištění. V ekonomických modelech se nejčastěji setkáme s měsíční nebo čtvrtletní periodou. Při očišťování se používají tzv. sezonní indexy, které vyjadřuj srovnání průměrné výše příslušné proměnné za jednotlivá období oproti průměru za celou časovou řadu. [10]

2.2. Box-Jenkinsonova metoda

Box-Jenkinsonova metodologie považuje za základní prvek náhodnou (reziduální) složku. Tato složka může být tvořena korelovanými (závislými) náhodnými veličinami. Přistupuje k analýze časových řad na základě speciálních stochastických modelů a díky tomu je schopna uspokojivě modelovat časové řady obecných průběhů, které by byli pomocí dekompozičního přístupu nerealizovatelné. Vyznačuje se nejen stochastickým modelováním trendu a sezónností ale především důrazem na autokorelační analýzu, který tvoří jádro této metody.

Výhody

- Stochastické modely typu ARMA jsou velmi flexibilní, díky tomu jsou použitelné i pro velmi obecné časové průběhy
- Softwarová podoba metodologie je dnes běžně dostupná ve většině statistických a ekonometrických programů
- V současnosti nejlepší metoda pro analýzu časově závislých pozorování

Nevýhody

- Vyžaduje minimálně délku časové řady 40 – 50 pozorování, což u finančních časových řad většinou nebývá problém
- Používání této metody je mnohem náročnější než použití dekompozičních metod
- Výsledné modely, zejména ty s větším počtem parametrů se obtížněji interpretují a zejména laici mají problém pochopit tyto modely [5]

2.2.1. Stochastický proces

Za stochastický proces považujeme v čase uspořádanou řadu náhodných hodnot $\{X(s,t), s \in S, t \in T\}$, kde S je výběrový prostor a T je indexní řada. Časovou řadu lze chápat jako realizaci stochastického procesu. V dalších kapitolách budeme předpokládat, že indexní řada je řadou celých čísel a předpokládané stochastické procesy mají nespojitý charakter. Jejich realizace, tedy časové řady budeme značit jako X_t. [5]

2.2.2. Stacionarita

Stochastický proces je označován za stacionární, pokud jsou charakteristiky jeho náhodných veličin v čase neměnné.

Striktní stacionarita – pravděpodobnost chování příslušného stochastického procesu je invariantní v čase

Slabá stacionarita - Vzhledem k problematickému ověřování striktní stacionarity, byl zaveden pojem slabá stacionarita, která má konstantní střední hodnotu a rozptyl

$$\mu_t = E(y_t) \tag{2.10}$$

$$\sigma_t^2 = D(y_t) = E(y_t - \mu_t)^2 \tag{2.11}$$

V této diplomové práci budeme pracovat pouze se slabou stacionaritou, která bude dále nazývána jen stacionarita. Závislost mezi dvěma pozorováními závisí pouze na jejich vzájemné časové vzdálenosti, na jejich hodnotu tedy nemá vliv skutečné umístění v časové řadě. Za stacionární stochastický proces tedy můžeme označit proces s náhodnými veličinami, které mají konstantní střední hodnotu a konstantní rozptyl.

V rámci Box – Jenkinsonovy metodologie je možné pracovat pouze se stacionárními časovými řadami. Pro převod z nestacionární na stacionární používáme různé transformace, nejčastěji pak diferencování. [1],[5]

2.2.3. Autokorelační funkce (ACF)

V případě stacionárního stochastického procesu $\{X_t\}$ lze vyjádřit autokovarienční funkci mezi veličinami X_t a X_{t-k} jako

$$\gamma_k(t,\ t-k) = cov(y_t,\ y_{t-k}) = E(y_t - \mu_t)(y_{t-k} - \mu_{t-k}) \tag{2.12}$$

a autokorelační funkci jako

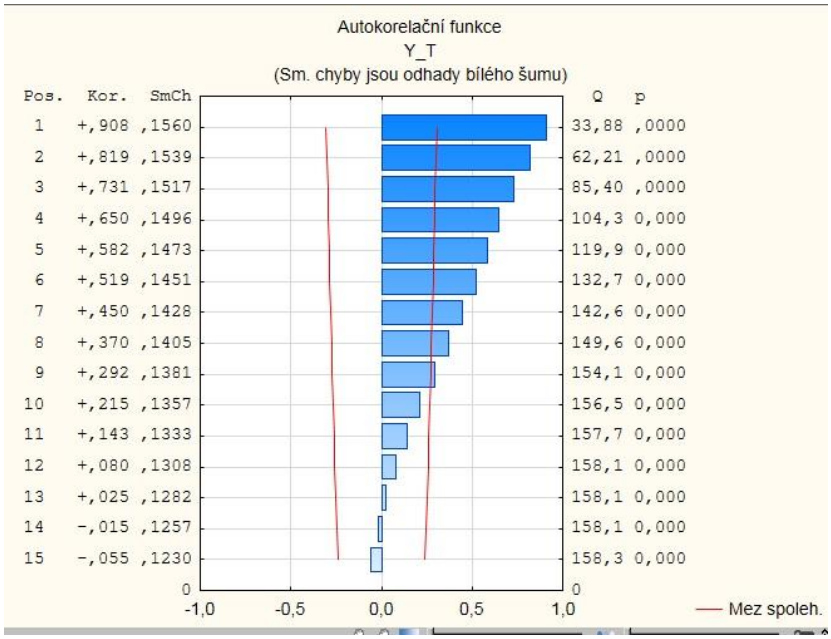
$$\rho_k(t,\ t-k) = \frac{\gamma(t,\ t-k)}{\sqrt{\sigma_t^2}\sqrt{\sigma_{t-k}^2}} = \frac{\gamma_k}{\gamma_0} \tag{2.13}$$

<u>Vlastnosti autokorelační funkce:</u>

- $\rho_0 = 1\ pro\ k = 0$
- $\gamma_k \leq \gamma_0, \rho_k \leq 1, pro\ k > 0$
- $\gamma_k \leq \gamma_0;\ \rho_k \leq 1, pro\ k > 0$
- $\gamma_k = \gamma - k\ a\ \rho_k = \rho - k$ pro všechna k, funkce je symetrická kolem $k = 0$.

Graf autokorelační funkce se nazývá ***korelogram***, viz obrázek 1. [1]

Obrázek 1: Korelogram Autokorelační funkce



Zdroj: Vlastní zpracování dat v softwaru Statistica 12

2.2.4. Parciální autokorelace (PACF)

Korelace mezi dvěma náhodnými veličinami je velmi často způsobena, že hodnoty obou veličin korelují s hodnotami třetí. Parciální autokorelace vyjadřuje pouze korelaci veličin X_t a X_{t-k} očištěnou o vliv veličin ležících mezi nimi. Parciální autokorelaci se zpožděním k vyjadřuje regresní koeficient $\emptyset_{kk}$ v autoregresi k-tého řádu

$$X_t = \emptyset_{k1}X_{t-1} + \emptyset_{k2}X_{t-2} + \cdots + \emptyset_{kk}X_{t-k} + e_t, \tag{2.14}$$

kde e_t je nekorelovaná s veličinami $X_{t-j} \geq 1$. [1]

2.2.5. Proces bílého šumu

Definice bílého šumu citována z [1] : „*Jestliže je stochastický proces* $\{a_t\}$ *řadou nekorelovaných náhodných veličin jednoho pravděpodobnostního rozdělení s konstantní střední hodnotou* $E(a_t) = \mu a$ *(obvykle nulovou), konstantním rozptylem* $D(a_t) = \sigma_a^2$ a $\gamma_k = C(a_t, a_{t-k}) = 0$, $pro\ všechna\ k \neq 0$.“ Z této definice vyplývá, že proces bílého šumu $\{a_t\}$ je stacionární s autokorelační funkcí

$$\rho_k = \begin{cases} 1 & k = 0 \\ 0 & k \neq 0 \end{cases} \tag{2.15}$$

a parciální autokorelační funkcí

$$\emptyset_{kk} = \begin{cases} 1 & k = 0 \\ 0 & k \neq 0 \end{cases}. \tag{2.16}$$

Základní vlastností je, že ACF a PACF bílého šumu jsou identicky nulové. V praxi se tento proces prakticky nevyskytuje, je důležitým prvkem při výstavbě modelů časových řad.

2.2.6. Lineární proces

Každý stacionární proces, který neobsahuje deterministickou složku (složka, která je na základě minulých dat perfektně předpovězena – střední hodnota, resp. konstanta, nějaká perioda, polynomická či exponenciální nebo logaritmická funkce časové proměnné *t*), může být vyjádřen jako kombinace nekorelovaných stejně rozdělených náhodných veličin. Toto tvrzení dokázal poprvé Herman Wold v roce 1938, po něm se tato lineární kombinace označuje jako Woldova reprezentace nebo také častěji zkráceně jen Lineární proces. Lineární proces je definován jako nekonečná řada:

$$y_t - \mu = \varepsilon_t + \psi_1\varepsilon_{t-1} + \psi_2\varepsilon_{t-2} + \cdots = (1 + \psi_1 B + \psi_2 B^2 + \cdots)\varepsilon_t = \sum_{j=0}^{\infty}\psi_j\,\varepsilon_{t-j}, \quad (2.17)$$

Kde ε_t je bílý šum s nulovou střední hodnotou a konstantním rozptylem a nulovou ACF a PACF. *B* je operátor zpětného časového posunu, který je definovaná jako *Byt = yt-1* . Lineární proces existuje pouze v případě, že nekonečná řada veličin konverguje podle kvadratického středu. Tato podmínka současně také zaručuje stacionaritu lineárního procesu a nulovou střední hodnotu. V Box – Jenkinsově metodologii mají praktický význam speciální lineární modely, které mají, konečný počet nenulových parametrů. Jednotlivé parametry modelu volíme tak aby konečné modely splňovali podmínky stacionarity a invertability. V praxi se můžeme i setkat i s nestacionárními modely ARIMA, které nesplňují stacionaritu, ale ty budou přiblíženy v další kapitole. [1,6]

2.2.7. Lineární modely stacionárních časových řad

V této kapitole budou vysvětleny vlastnosti jednotlivých procesu AR a MA a následného použití obou ARMA, které jsou základní procesy Box – Jenkinsonovy metodologie.

2.2.7.1. Autoregresní proces řádu p (AR (p))

Autoregresní proces AR(p) je vyjádřen ve tvaru:

$$y_t = \phi_1 y_{t-1} + \cdots + \phi_p y_{t-p} + \varepsilon_t, \text{ tj. } y_t - \phi_1 y_{t-1} + \cdots + \phi_p y_{t-p} = \varepsilon_t \quad (2.18)$$

Nebo pomocí operátoru zpětného posunu *B*, je možné tento proces vyjádřit jako

$$\phi_p(B) y_t = \varepsilon_t, \quad (2.19)$$

kde $\phi_p(B) = \left(1 - \phi_1 B - \cdots - \phi_p B^p\right)$ je autoregresní operátor. Tento proces vyniká při tzv. useknutí invertovaného tvaru lineárního procesu v bodě, který odpovídá zpoždění *p*.

Pro podmínku stacionarity, musí kořeny polynomické rovnice $\phi_p(B) = 0$ ležet vně jednotkového kruhu v komplexní rovině. Střední hodnota tohoto procesu je rovna nule a rozptyl je vyjádřen vztahem

$$\sigma_y{}^2 = \frac{\sigma^2}{1-\phi_1\rho_1-\cdots-\phi_p\rho_p} \tag{2.19}$$

Autokorelační musí splňovat následující diferenční rovnici

$$\rho_k = \phi_1\rho_{k-1} + \cdots + \phi_p\rho_{k-p} \quad \text{pro } k > 0. \text{ [1]} \tag{2.20}$$

2.2.7.2. Autoregresní proces řádu jedna [AR(1)]

Autoregresní model prvního řádu je možné zapsat ve tvaru:

$$y_t = \phi_1 y_{t-1} + \varepsilon_t, \tag{2.21}$$

nebo pomocí operátoru zpětného posunu jako

$$(1 - \phi_1 B)y_t = \varepsilon_t. \tag{2.22}$$

Jestliže platí $|\phi_1| < 1$, poté lze tento vztah vyjádřit jako formu stacionárního procesu lineárního procesu tj.

$$y_t = \frac{\varepsilon_t}{(1-\phi_1 B)} = \left(1 + \phi_1 B + \phi_1{}^2 B^2 + \cdots\right)\varepsilon_t = \varepsilon_t + \phi_1\varepsilon_{t-1} + \phi_1{}^2\varepsilon_{t-2} + \cdots \tag{2.23}$$

Střední hodnota je $E(y_t) = 0$ a je tedy konstantní v čase, což je v souladu s podmínkou stacionarity. Pro rozptyl platí

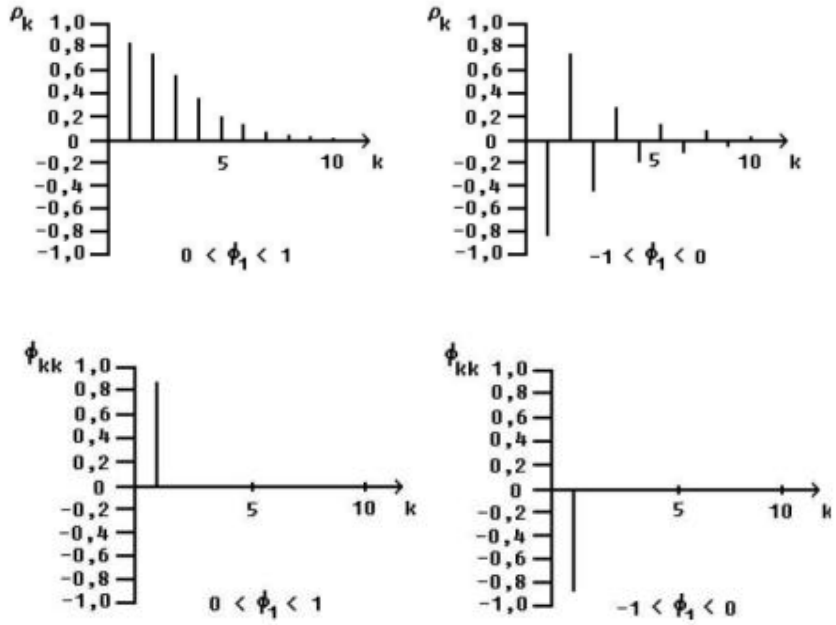
$$\gamma_0 = D(y_t) = \frac{\sigma^2}{1-\phi_1{}^2}, \tag{2.24}$$

který je rovněž konstantní v čase. Parciální autokorelační funkce procesu AR (1) je vyjádřena touto formou:

$$\phi_{kk} = \begin{cases} \rho_1 = \phi_1, & k = 1 \\ 0, & k = 2 \end{cases} \tag{2.25}$$

Na obrázku 2 jsou zobrazeny tvary autokorelační funkce procesu AR (1). [1]

Obrázek 2: ACF a PACF AR (1)



Zdroj:[3]

2.2.7.3. Autoregresní proces řádu dva [AR(2)]

Autoregresní proces druhé řádu můžeme zapsat ve tvaru:

$$y_t = \phi_1 y_{t-1} + \phi_2 y_{t-2} + \varepsilon_t, \tag{2.26}$$

nebo také jako

$$(1 - \phi_1 B - \phi_2 B^2) y_t = \varepsilon_t. \tag{2.27}$$

Proces AR(2) je stacionární, pokud kořeny pokud kořeny rovnice polynomické rovnice $(1 - \phi_1 B - \phi_2 B^2) = 0$ leží uvnitř jednotkového kruhu v komplexní rovině. [1]

2.2.7.4. Proces klouzavých průměrů řádu *q* [MA(*q*)]

Proces klouzavých průměrů q-tého řádu má tvar

$$y_t = \varepsilon_t + \theta_1 \varepsilon_{t-1} + \cdots + \theta_q \varepsilon_{t-q}, \tag{2.28}$$

Případně lze vyjádřit pomocí operátoru zpětného posunu B jako proces

$$y_t = \theta_q(B)\varepsilon_t, \tag{2.29}$$

kde $\theta_q(B) = 1 + \theta_1 B + \cdots + \theta_q B^q$ je operátor klouzavých součtů. Modely MA vychází přímo z lineárního procesu, ale vždy mají konečný počet vah $\psi(\theta)$, díky tomu má vždy nulovou střední hodnotu a rozptyl je vyjádřen

$$\sigma_y{}^2 = \left(1 + \theta_1{}^2 + \ldots + \theta_p{}^2\right)\sigma^2 \tag{2.30}$$

Autokorelační funkce má tvar

$$\rho_k = \begin{cases} \frac{\theta_k + \theta_1\theta_{k+1} + \cdots + \theta_{q-k}\theta_q}{1 + {\theta_1}^2 + \cdots + {\theta_q}^2} & \text{pro } k = 1, \ldots, q \\ 0 & \text{pro } k > q \end{cases}. \tag{2.31}$$

Autokorelační funkce má body useknutí rovný řádu modelu q, naopak parciální autokorelační funkce nemá pod useknutí ale je omezena lineární kombinací geometrických klesajících posloupností a sinusoid s geometricky klesajícími amplitudami. Jestliže je možné modely MA vyjádřit ve formě konvergující reprezentace AR(∞), pak se tento proces dá označit jako invertibilní, proto je nutné, aby kořeny polynomu $\theta_q(B)$ ležet vně jednotkového kruhu v komplexní rovině. [3]

2.2.7.5. Proces klouzavých průměrů řádu jedna [MA(1)]

Proces klouzavých průměrů (moving average) prvního řádu má tvar

$$y_t = \varepsilon_t + \theta_1\varepsilon_{t-1}, \tag{2.32}$$

respektive

$$y_t = (1 + \theta_1 B)\varepsilon_t.$$

Aby tento proces splňoval invertibilní podmínku, musí platit, že $|\theta_1| < 1$. Autokorelační funkce tohoto procesu je

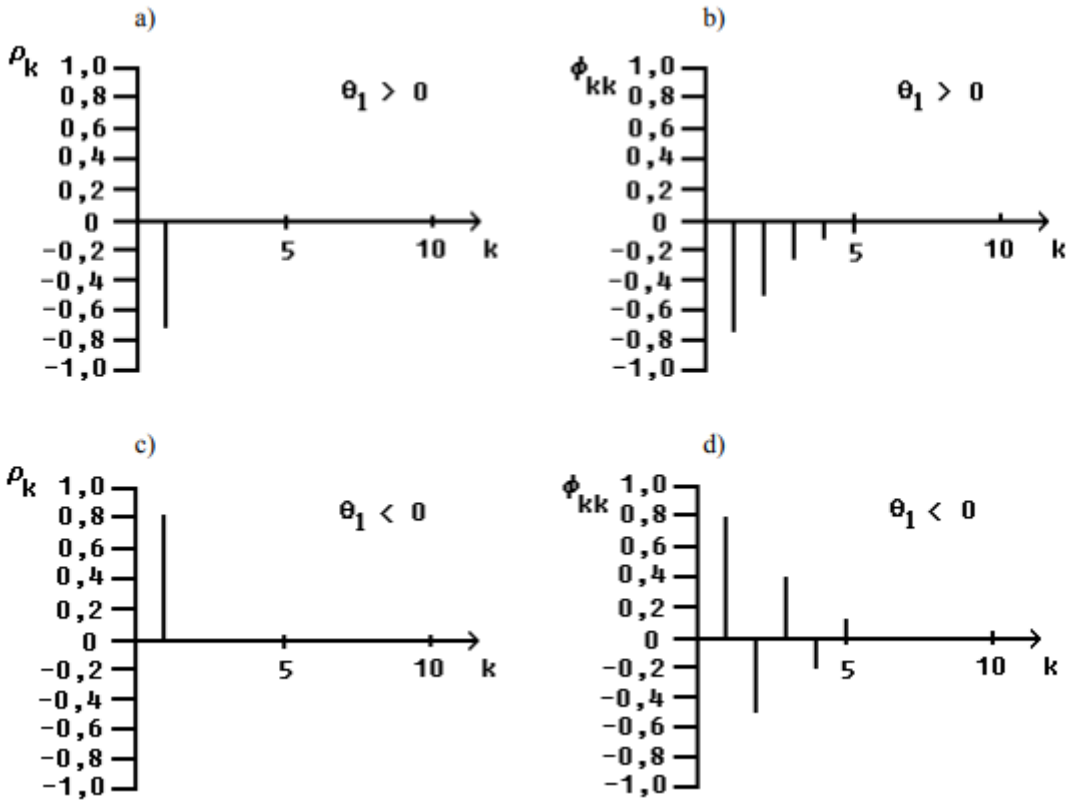
$$\rho_1 = \frac{\theta_1}{1 + {\theta_1}^2}, \quad \rho_k = 0, \quad k = 2,3, \ldots \tag{2.33}$$

a parciální autokorelační funkce má tvar

$$\phi_{kk} = \frac{(-1)^{k-1}{\theta_1}^k(1 - {\theta_1}^2)}{1 - {\theta_1}^{2(k+1)}}, \quad k = 1,\ 2,\ 3 \ldots \tag{2.34}.$$

Na obrázku 3 jsou zobrazeny charakteristické tvary parciální a autokorelační funkce procesu MA(1). [3]

Obrázek 3: ACF a PACF MA (1)



Zdroj: [3]

2.2.7.6. Proces klouzavých průměrů řádů dva [MA(2)]

Proces klouzavých průměrů druhého řádu neboli MA2 je možné zapsat tvarem

$$y_t = \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2}, \tag{2.35}$$

Obdobně jako u modelu MA1 je nutné, aby byl proces invertibilní, tedy kořeny rovnice

$$(1 + \theta_1 B + \theta_2 B^2) = 0 \tag{2.36}$$

Ležely uvnitř jednotkového kruhu v komplexní rovině čísel. Autokorelační funkce má tvar

$$\rho_k = \begin{Bmatrix} \frac{\theta_1(1+\theta_2)}{1+{\theta_1}^2+{\theta_2}^2}, & k = 1, \\ \frac{\theta_2}{1+{\theta_1}^2+{\theta_2}^2}, & k = 2, \\ 0, & k > 2 \end{Bmatrix}. \tag{2.37}$$ [3]

2.2.7.7. Smíšené procesy ARMA *(p, q)*

Smíšený proces řádu p a q je možné vyjádřit rovnicí

$$y_t = \phi_1 y_{t-1} + \cdots + \phi_p y_{t-p} + \varepsilon_t + \theta_1 a_{t-1} + \cdots + \theta_q \varepsilon_{t-q}, \tag{2.38}$$

nebo

$$\phi_p(B) y_t = \theta_q(B) \varepsilon_t, \tag{2.39}$$

$$\text{kde } \phi_p(B) = \left(1 - \phi_1 B - \cdots - \phi_p B^p\right) \text{ a } \theta_q(B) = 1 - \theta_1 B - \cdots - \theta_q B^q. \tag{2.40}$$

Podmínka stacionarity tohoto smíšeného procesu je stejná jako podmínka u modelu AR (p) a podmínka invertibility je zase stejná jako u procesu MA (q). Střední hodnoto smíšeného procesu je nulová. [3]

2.2.7.8. Smíšený proces ARMA (1,1)

Smíšený proces ARMA (1,1) je nejjednodušší ze třídy smíšených procesů a lze ho vyjádřit rovnicí

$$y_t = \phi_1 y_{t-1} + \varepsilon_t - \theta_1 \varepsilon_{t-1} \tag{2.41}$$

nebo také

$$(1 - \phi_1 B) y_t = (1 - \theta_1 B) \varepsilon_t. \tag{2.42}$$

Tento proces bude stacionární, pokud platí, že $|\phi_1| < 1$ a invertibilní když $|\theta_1| < 1$. Proces je samozřejmě stacionární, je-li $|\phi_1| < 1$ a invertibilní, když $|\theta_1| < 1$. Jestliže $\phi_1 = 0$, proces ARMA (1, 1) se redukuje na proces MA (1) a jestliže $\theta_1 = 0$, redukuje se na proces AR (1). [3]

2.2.8. Modely nestacionárních časových řad

V předchozí kapitole jsem se zabýval stacionárními procesy, při nejen ekonomické praxi se velmi často můžeme setkat i s časovými řadami, které jsou tvořeny nestacionární stochastickými procesy. Nestacionarita procesů může být způsobena měnící se střední hodnotou v čase nebo měnícím se rozptylem. [3]

2.2.8.1. Proces náhodné procházky

Proces náhodné procházky, nebo také je možné setkat se v literatuře s pojmem Random Walk, je zvláštní případ modelu AR (1), kde $\emptyset_1 = 1$. Je možné ho vyjádřit rovnicí

$$y_t = y_{t-1} + \varepsilon_t, \tag{2.43}$$

nebo pomocí operátoru zpětného posunutí

$$(1 - B)y_t = \varepsilon_t. \tag{2.44}$$

Jestliže budeme předpokládat začátek procesu v čase $t = 0$, pak je možné zapsat proces jako

$$y_t = y_0 + \varepsilon_t + \varepsilon_{t-1} + \varepsilon_{t-2} + \cdots = \sum_{i=1}^{t} \varepsilon_{t-i}. \tag{2.45}$$

Tento proces je tvořen kumulováním náhodných veličin, které tvoří bílý šum. Jelikož první diference tohoto procesu je proces bílého šumu, je možné ho nazývat integrovaným procesem prvního řádu náhodné procházky. [3]

2.2.8.2. Smíšený proces ARIMA

Jestliže je možné po transformaci pomocí diference *d*-tého řádu, proces vyjádřit ve formě stacionárního a invertibilního modelu ARMA (*p, q),* označujeme původní proces jako ARIMA (*p,d,q)* a zapisujeme ve tvaru

$$\phi_p(B)(1 - B)^d y_t = \theta_q(B)\varepsilon_t, \tag{2.46}$$

kde $\phi_p(B) = 1 - \phi_1 B - \cdots - \phi_p B^p$ a $\theta_q(B) = 1 - \theta_1 B - \cdots - \theta_q B^q$. (2.47)

Je tedy možno říct, že v takovém modelu ARIMA se nejprve provede stacionarizace pomocí vhodné diference modelové řady a vzniklá stacionární řada se modeluje pomocí smíšeného modelu ARMA, jehož vlastnosti jsou podobné jako vlastnosti náhodné procházky. [3]

2.2.9. Konstrukce modelu pomocí Box-Jenkinsonovy metody

Konstrukce modelu pomocí B-J metody se řídí několika kroky: identifikace modelu, odhad parametrů modelu a nakonec verifikace modelu.

2.2.9.1. Identifikace modelu

Při konstrukci modelu v Box –Jenkinsnově metodologie postupujeme nejdříve, tak že je nutné vybrat správný model (AR, MA či ARMA). Tato část jaký typ zvolit patří k jednomu z nejtěžších úloh při výstavbě modelu. Nejprve je nutné ověřit stacionaritu, a pokud není splněna tak stacionarizovat pomocí diference. Vlastní analýza modelu je založena na odhadu autokorelační a parciální autokorelační funkce. Pokud se jedná o finanční časovou řadu, je nejprve nutné použít logaritmickou transformaci a až poté přistoupit k diferencování. Pokud nejsme schopni na první pohled odhalit trend řady, je nutné udělat odhad ACF a PACF původní řady, a na jejich základě potvrdit potřebu stacionarizace. V takovém případě budou hodnoty ACF blízké jedné a budou pomalu klesat naopak hodnoty PACF budou v prvním zpoždění blízké jedné a ostatní budou velmi malé.[3,8]

2.2.9.2. Odhad parametrů modelu

K určení identifikace modelů AR a MA (tj. nalezení hodnot p a q) po provedení stacionarizace (pokud je nutná) se opět používá ACF a PACF. Tato identifikace je založena na principu podobnosti výběrové a teoretické ACF a PACF. Popis tvarů ACF a PACF pro modely AR a MA je uveden v Tabulce 1.

Tabulka 1:Popis ACF a PACF pro výběr model AR, MA, ARMA

Model	**ACF**	**PACF**
AR (p)	Exponenciální a/nebo exponenciálně sinusoidní pokles	$\emptyset_{kk} = 0\ pro\ k > p$
MA (q)	$\rho_k = 0\ pro\ k > q$	Omezená exponenciální a/nebo exponenciálně sinusoidním poklesem
ARMA (*p*,*q*)	Od zpoždění (q-p) pro $q>p$ exponenciální nebo exponenciálně sinusoidní pokles	Od zpoždění (p-q) pro $p>q$ omezená exponenciálním nebo exponenciálně sinusoidním poklesem

Zdroj: Vlastní zpracování na základě [3],[13]

2.2.9.3. Verifikace

Cílem verifikace modelu je potvrzení jeho správnosti a kvality. Nejprve je nutné ověřit významnost námi odhadnutých parametrů. Pro verifikaci modelu se používá metoda založena na Portmanově testu autokorelace nebo metoda, kdy testujeme rezidua z modelu na přítomnost autokorelace. Jsou-li zjištěny závažné odchylky modelu od původních řad, opakujeme postup a zkusíme najít kvalitnější model. [13]

3. PROGRAMOVACÍ JAZYK R

Programovací jazyk R a prostředí R bylo vytvořeno speciálně pro statistické výpočty a grafiku, jako projekt skupiny GNU. Tento programovací jazyk je podobný již dříve vytvořenému jazyku S, existují rozdíly mezi těmi to dvěma jazyky, ale naprostá většina kódů funguje na obou platformách. R poskytuje velmi širokou škálu statistických a grafických technik. Ze statistických patří mezi nejpoužívanější lineární a nelineární modelování, klasické statistické testy, v případě potřeby je možné využít funkce klasifikace nebo shlukování a mnoho dalších. V této práci se zaměřím zejména na využití při analýze časových řad. Jazyk S je velmi často používán při výzkumech ve statistické metodologii, v komerčním využití se ale preferuji jazyk R, který poskytuje velmi podobné funkce ale je tzv. freeware. To znamená, že program je možné bezplatně stahovat ze stránek vydavatele na https://www.r-project.org/. Program poskytuje modifikace jak pro uživatele operačního systému Microsoft od Windows XP až po nejnovější Windows 10, uživatele Apple mohou využít možnosti verze pro Mac-OS.

Jednou z předností R je velmi snadné vytváření grafických analýz a grafů včetně matematických symbolů a vzorců, pokud je to potřeba. Velká pozornost je věnována možnostem úprav v grafice jednotlivých modelů a výstupů, která je ovšem velmi dobře ovladatelná. Je také možné dodatečně stáhnout velké množství tzv. knihoven, které se skládají z tzv. balíčků. Díky tomu lze využít například jiné grafické rozhraní, které je nám bližší nebo využít již naprogramovaných postupů, které byli k naší problematice vytvořeny jiným uživatelem. Pro pokročilejší využití například v genetice, bioinformatice, geografii nebo sociologii je nutné pomocí knihoven doplnit základní funkce programu. Všechny knihovny je možné stahovat zcela automaticky. Například pro import dat ze souboru Microsoft Excel je nutné mít staženou knihovnu, která se nenachází v základní verzi, program se Vás zeptá, zda souhlasíte se stažením knihovny k tomuto úkonu ze serveru. Po potvrzení se program sám připojí na portál knihoven na webu https://cran.r-project.org/ a stáhne knihovnu, instalace probíhá také automaticky a je možné nainstalovanou knihovnu ihned začít používat.

Velkou výhodou programu je rovněž schopnost pracovat s více druhy záznamů, respektive program umí importovat například i textové soubory, které se doporučují nahrávat ve formě textového bloku, s nimiž jsou ovšem složitější výpočty a jejich analýza. Pro všechny druhy nahrávání dat existuje zvláštní knihovna, kterou je nutné stáhnout, jak bylo popsáno výše postup stažení, je velice jednoduchý a rychlý. Běžný uživatel bude zřejmě nejčastěji využívat nahrávání dat ve formě xls., tedy ve formě dat v programu Microsoft Excel. Platí obecné pravidlo, že v souboru by měli figurovat pouze data, která jsou důležitá pro analýzu veškeré další dodatečné poznámky nebo mezi výpočty by měli být před importováním odmazány. Od

roku 2014 byla vytvořena knihovna, která umožnuje nahrávat data z programu SPSS Modeler od firmy IBM, kde v tomto programu je již také přidána funkce pro využívání programovací jazyka R. Data jsou možné taky nahrát ze softwaru Statistika nebo SAS, který se velmi často využívá v bankách nebo pojišťovnách, kde se pomocí něho analyzují nebo modelují časové řady, zejména v oblasti risk managementu a řízení Cash Flow.

Mezi nevýhody tohoto rozhraní pro běžného uživatele jednoznačně patří nutnost ovládání pomocí příkazového řádku. Pro znalce programovacího jazyka VBA nebo SQL to nemusí být velká nevýhoda, nicméně v porovnání třeba s Microsoft Excel je tvorba grafu nebo základních výpočtů komplikovanější. Díky možnosti stažení si knihovny, může prostředí fungovat podobně jako příkazový řádek v již zmíněném Microsoft Excel, kdy vypíšeme funkci, a program nám formou nápovědy radí, co by mělo být vyplněné v jednotlivých argumentu viz obrázek 4. [18]

Obrázek 4: Příkazový řádek v jazyce R

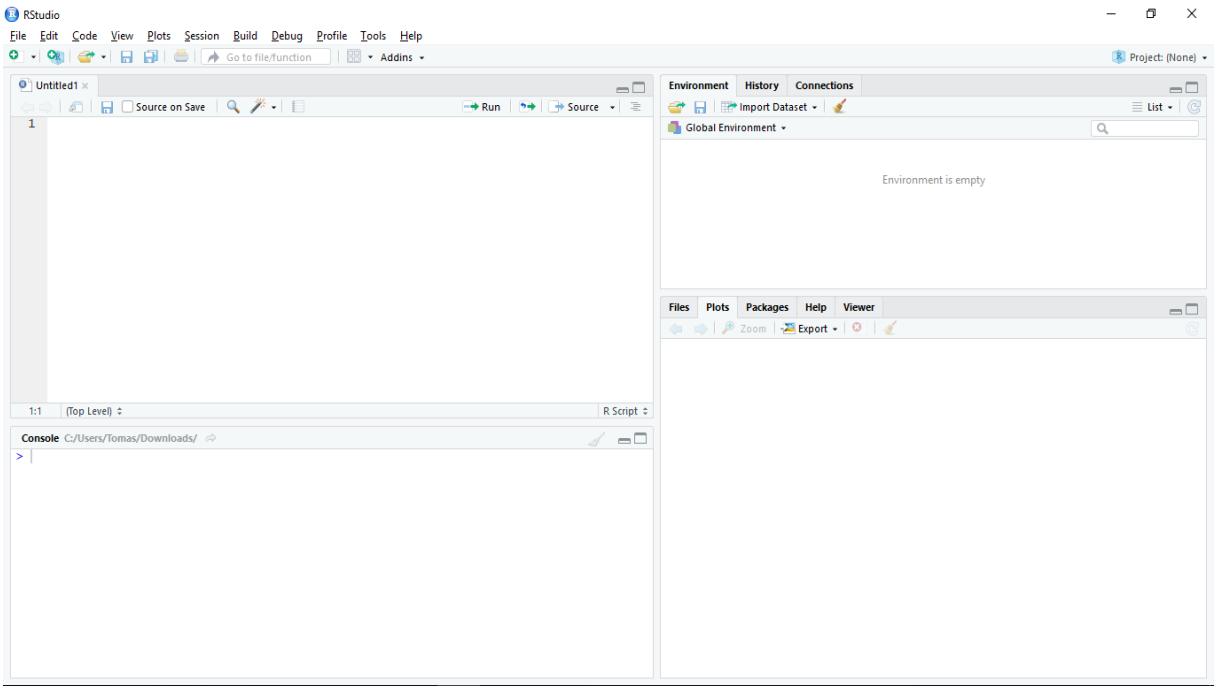
Zdroj: Vlastní zpracování z R

Dalším problémem může být, pokud pracujeme s velkými databáze řádově o stovkách tisíc řádků, že si s tím program neumí poradit a je nutné využít jiný software popřípadě rozdělit databázi na menší části, což může být velmi časově náročné a někdy neproveditelné.

3.1. Základní rozhraní R

Na obrázku 5 je vidět základní rozhraní programu po instalaci bez dodatečných knihoven nebo jakýchkoliv změn v zobrazení. Defaultně je program v angličtině, dá se změnit i na ostatní jazyka, nicméně čeština v nabídce schází. Na internetu se dá dohledat český jazyk, autor ale upozorňuje, že kompatibilita se všemi funkce není zaručena, zejména pokud jsou dosahovány další knihovny.

Obrázek 5: Základní rozhraní programu R

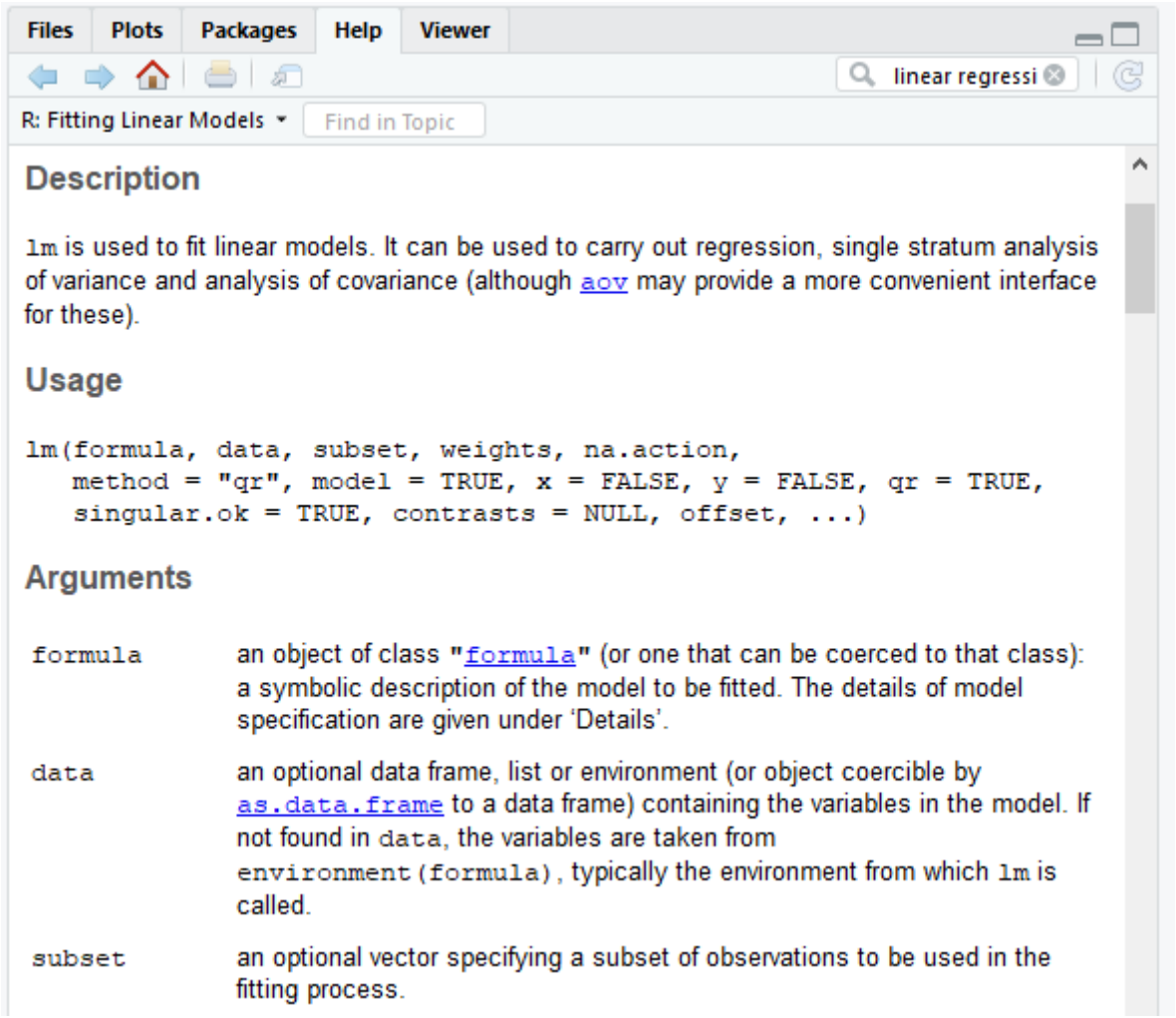


Zdroj: Vlastní zpracování v R

Vlevo nahoře obrázku 5 je umístěna lišta záložek, která je velmi podobná většině programů. Zde se nachází zřejmě nejvíce používaná záložka File, kde si vytváříme nový projekt, otevíráme pomocí příkazu „Open“ starší projekt, na kterém jsme již pracovali, také se zde nachází možnosti pro uložení projektu a postup. Další záložka „Edit“ poskytuje celkem obvyklé možnosti jako vložit, vyjmout nebo kopírovat, popřípadě vrátit o krok zpět. Další velmi důležitá záložka je Tools, ve které máme přístup do knihovny dalších rozšíření, které můžeme potřebovat popřípadě k aktualizaci již nainstalovaných knihoven.

V poslední záložce „Help“ máme přístup jak k nápovědě k programu samotnému tak k nápovědě ohledně funkcí které chceme použít. Další možnost jak vyvolat nápovědu je pokud do příkazového řádku napíšeme *help (), kde* v závorce je název funkce, ke které požadujeme nápovědu. Tato varianta může být někdy velmi komplikovaná, neboť ne všechny názvy funkcí jsou uživateli známé. Například lineární regrese, kterou budeme používat k modelaci časových řad, se vyvolává pomocí příkazu *lm()*. Pokud neznáme přímo název funkce, nejjednodušší cestou jak zjistit požadovanou radu k funkci je použít v na obrázku vpravo dole záložku *Help,* a vyhledávacího pole napsat požadovanou funkci a program sám vyhledá nápovědu a nabídne řešení, jak je vidět na obrázku 6. Také ukáže popis funkce, případně příklad využití, velmi užitečné je zejména u složitějších funkcí, které mohou obsahovat několik argumentů, jejich popis co který obsahuje včetně vysvětlení, jaké hodnoty jsou přípustné pro něho.

Obrázek 6: Vyvolána funkce *Help*



Zdroj: Vlastní zpracování v R

Další záložky v okně vpravo dole na obrázku 5, jsou Files, kde se nastavuje výchozí složku, kam se budou nastavovat ukládat vytvořené projekty a ve které mohou být například zdrojové soubory, poznámky a jiné. Druhá záložky *Plots*, je záložky pro úpravu a exportování grafů, které jsme vytvořily v rámci našeho kódu v příkazovém řádku. Graf v tomto okně je možné upravovat, například zvětšovat, nebo pokud jsme vytvořily větší počet grafů, zobrazovat jednotlivé grafy, kde je možné je třeba porovnávat mezi sebou nebo mazat ty, se kterými nejsme spokojeni, nebo nejsou důležité pro další postup. Zřejmě nejdůležitější je prostřední záložka *Packages*, kde se nachází seznam knihoven. Jsou zde zobrazeny nainstalované knihovny uživatelem a systémově nainstalované knihovny, to vše včetně popisu, co knihovna dělá. Vidíme zde také verzi jednotlivé knihovny, kterou můžeme tlačítkem zkontrolovat, jestli máme aktuální verzi, pokud ne můžeme ji jedním kliknutím tlačítka aktualizovat.

Na obrázku 5(ten první), vlevo nahoře pole je podobné například Matlabu, kam vypisujeme program, jednotlivé proměnné, funkce atd. Pokud použijeme symbol # na příkazovém řádku, způsobí to ignoraci zbytku řádku a zvýrazní to jako komentář pro uživatele. Ale v tomto poli není program spouštěn, zde je pouze kód, který se musí spouštět buď ručně anebo překopírovat do pole vlevo dole, kde po zmáčknutí klávesy *Enter* se automaticky spustí všechny námi vypsané procedury včetně jejich výsledků. Pokud jsme nadefinovali špatně proměnnou nebo funkci vypíše program chybu, a upřesnění kde nastala komplikace při výpočtu.

Pole z obrázku 5, které je umístěno vpravo nahoře a obsahuje tři záložky. První *Enviroment* obsahuje tlačítko pro import databáze. Program umí nahrávat data, která jsou v textové podobě, pokud je stažená knihovna, která to umožnuje. Další možnosti jsou import ze souboru Microsoft Excel, kde zvládá soubory s koncovkou csv. i klasické xls. Poslední možností je importovat data ze statistických softwarů SAS, SPSS a Stastitica. Další záložka *History* obsahuje minulé projekty, které byli již modelované. Můžeme tyto informace použít pro inspiraci při tvorbě dalších projektů, nebo prostě využít část přechozí práce do současného projektu. Poslední záložka je Connections, kterou využíváme v případě, že máme více propojených souborů a projektů, v této práci se tato možnost nebude využívat. [12],[18]

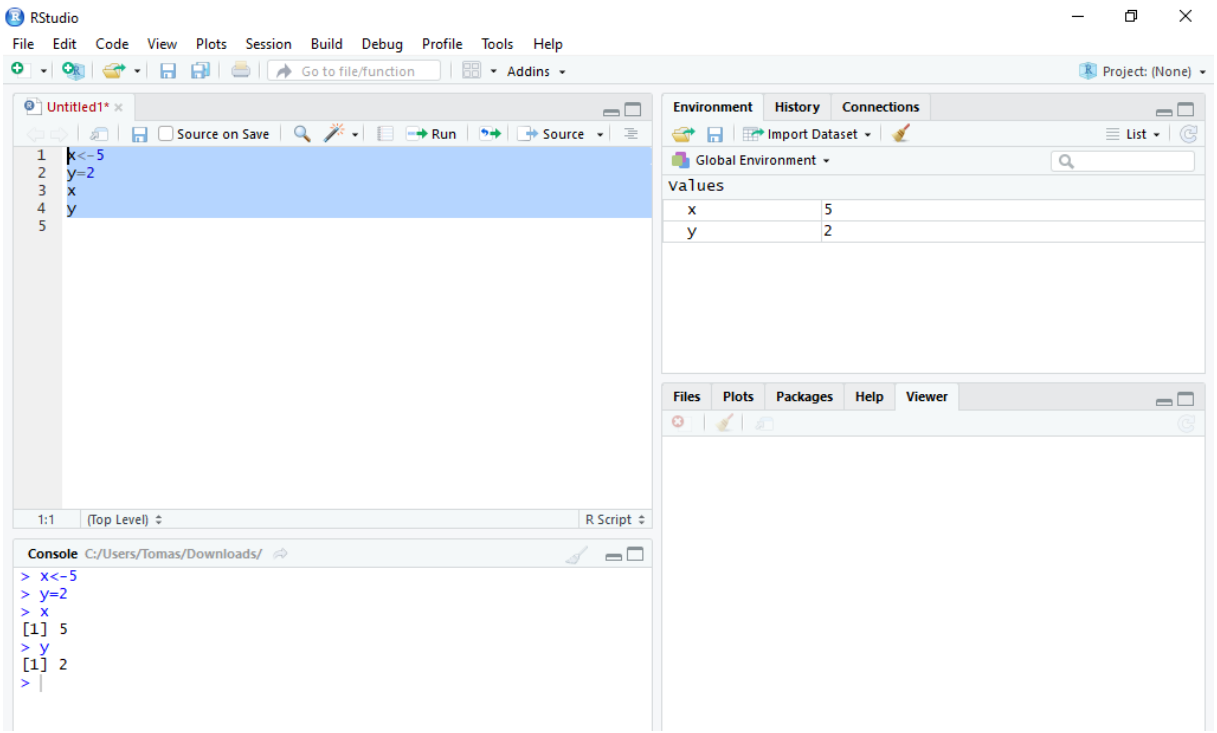
3.2. Proměnné

Proměnné jsou velmi důležitou součástí každého statistického programu, ale ne vždy nabízí takovou flexibilitu jako právě v programu R. Na rozdíl od staticky zadaných programů, jako je například jazyk C++, R nevyžaduje proměnné typy, které mají být pevně deklarovány. Proměnná může být určena jako libovolný dostupný datový typ, jak bude popsáno v další kapitole. Může také obsahovat jakýkoliv objekt, jako je například funkce, výsledek provedené analýzy nebo dokonce i graf. Proměnná může být na začátku procesu vedena jako číslo, později je možné s touto proměnou počítat a nakonec ji změnit v kódu například na graf, který z toho vznikl a později ji znovu změnit na číselnou hodnotu. [11]

3.2.1. Přiřazení proměnných

Existuje řada způsobů, jak přiřadit hodnotu proměnné a nezávisí to na typu přiřazené hodnoty. Platné operátory přiřazeny jsou <- a =,přičemž ten první je preferován většinou literatur. Například pokud chceme uložit proměnné *x* hodnotu 5 a proměnné *y* přiradíme hodnotu 2.

Obrázek 7: Přiřazení proměnných



Zdroj: Vlastní zpracování v R

Na obrázku 7 je vidět, jsou vidět oba způsoby přiřazení proměnným. Následně je vidět vpravo nahoře, že se uložily obě proměnné do paměti programu. Poté kdykoliv vypíšeme název proměnné do Console a program vypíše její hodnotu.

Pro přiřazení proměnné je také možné využít více hodnot současně jako například, hodnota *a* i *b* mohou být stejné a mít stejnou hodnotu. Další způsob jak přiradit proměnou zejména pro složitější je použít funkci *assign(),* která k námi zvolenému názvu přiřadí hodnotu

```
> a <- b <-5
> a
[1] 5
> b
[1] 5
> assign("cas t",5)
>
> `cas t`
[1] 5
```

Názvy proměnných mohou obsahovat libovolnou kombinaci alfanumerických znaků včetně tečky(.) a podtržítka (_), nemohou ovšem těmito znaky začínat. Program také nemá problém s háčky a čárky, což se občas stává s programy, které neobsahují český jazyk, ale uživatel definuje názvy proměnných v tomto jazyce. Nejběžnější formou přiřazení se v komunitě uživatelé R využívá levá šipka(<-), což nemusí být úplně uživatelsky přívětivé pro začínající uživatele. Je to zejména z důvodu, že proměnná poukazuje na její hodnotu. Tam je také vidět rozdílnost s jazykem SQL, kde znak rovnosti (=) je test pro rovnost. Obecně se považuje za

nejlepší postup k použití skutečných jmen namísto zkratek či jen písmen, obvykle se používají podstatná jména. To poskytuje lepší orientaci dalším uživatelů, kteří se nepodíleli na tvorbě kódu, nebo i v případě dlouhého kódu poskytuje lepší orientaci. Program rozlišuje velká a malá písmena, což může činit problémy zejména uživatelů jazyka SQL a Visual Basic, kde není nutné tyto věci rozlišovat.

V případě potřeby je možné také nahrané proměnné mazat. To se využívá buď z důvodu větší přehlednosti, kdy proměnné které už nebudeme používat, vymažeme pomocí příkazu *rm*. Tím se uvolní paměť pro operační systém, a umožnuje rychlejší práci při velkém počtu nahraných dat. [15]

3.3. Datové typy

V programu R existuje mnoho datových typů pro uchování dat. Jako čtyři hlavní můžeme označit numerické, znakové (řetězec), datum a logické (TRUE /FALSE). V případě, že si uživatel není jist, který datový typ použít funkci *class*, která sama ověří, jaký datový typ neznámá obsahuje a vypíše to uživateli. Funkce nemusí klasifovat pouze uložené proměnné, ale i přímo hodnoty například

```
> class(125)
[1] "numeric".
```

Numerické hodnoty jsou podle očekáváná nejpoužívanějším typem. Číselně uložená proměnná je automaticky považována za číselnou. Pokud proměnná je celé číslo, považujeme ji za proměnou *integer*. V programu R se desetinná čísla zapisují pomocí místo desetinné čárky jako tečka např. 1.25. V případě potřeby je také možné využívat komplexní čísla, která musí být ve tvaru *1+2i*.

Logické hodnoty reprezentují data, která mohou pouze nabývat hodnoty TRUE nebo FALSE. Numericky je možné vyjádřit TRUE jako pravdu číslem 1 a FALSE jako nepravdu číslem 0. Logické výrazy jsou výsledkem některých analýz, používají se jako argumenty některých funkcí popřípadě pro ověření dat např.

```
> a=10
> b=25
> a>b
[1] FALSE
```

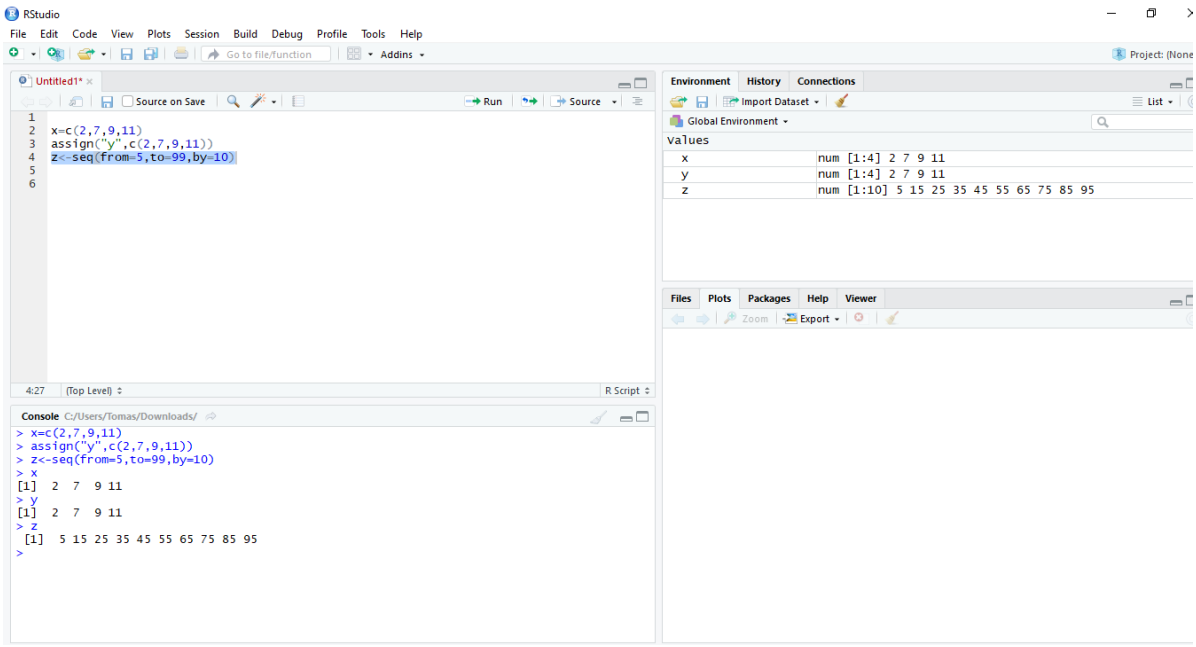
Řetězec používáme pro textové hodnoty, pro popisky os, název grafu atd. musí vždy být zadán do jednoduchých apostrofů (’) nebo dvojitých uvozovek (”). [11]

3.4. Vektor

Nejjednodušší datová struktura je vektor. Vektory jsou posloupnost hodnost s jistou informační hodnotou – mohou obsahovat počet nezaměstnaných, hodnotu HDP, teplotu v místnosti atd. Vektory jsou jedno-dimenzové struktury, které se skládají z posloupnosti prvků, které všechny musí být stejného datového typu (numerické, logické atd.) Některé prvky nemusejí být známi, v těchto případech je umístěna speciální hodnota *NA*(„Not Available“).

Na obrázku 8 je vytvořen vektor *x* o hodnotách 2, 7, 9 a 11 nejprve pomocí příkazu *x* <-(2,7,9,11) a následně analogicky vektor *y* se stejnými hodnotami ale za pomocí příkazu assign. Obdobně je možné zadávat textové hodnoty, jako zde roční období. Funkce *c()* je zobrazována jako řádkový vektor ale je s ní nakládáno jako s vektorem sloupcovým. V případě potřeby generování aritmetické posloupnosti využijeme funkci ***seq()***. První dva argumenty této funkce jsou *from* a *to*. Tedy určení začátku a konce posloupnosti. Další argument *by*, znamená, jak velký bude krok, tedy rozdíl mezi hodnotou *n* a *n-1*. V programu R je vytvořena časová posloupnost, která začíná hodnotou 5, krokem 10 a maximální hodnota je 99. V tomto případě nebude poslední hodnota 99, jelikož bychom nesplňovali diferenci mezi hodnotami 10, proto poslední vypsaná hodnota na obrázku 8 je 95. [11]

Obrázek 8: Přiřazení vektorů



Zdroj: Vlastní zpracování v R

3.4.1. Faktory

Faktory jsou speciálním případem vektorů, které se skládají z nominálních nebo ordinálních dat. Jedná se o datovou strukturu, která umožňuje přiřadit smysluplné názvy jednotlivým kategoriím, díky informaci „*Levels*“ – jedná se o konečnou množinu hodnot, kterých kategorická proměnná může nabývat, kde hodnoty „*NA*“ nejsou zahrnuty a vypsané hodnoty jsou řazeny od nejmenší po největší nebo abecedně.

```
> factor(c("jaro","leto", NA,"podzim","zima"))
[1] jaro   leto   <NA>   podzim zima
Levels: jaro leto podzim zima
```

V případě pokud chceme jednotlivé položky seřadit, použijeme ještě funkci „*labels*“, nejdříve přiřadíme jakou důležitost pomocí čísla a následně pomocí popisku doplníme název. Například typickým pro toto je vzdělání, kdy nemůžeme numericky vyjádřit o kolik je které lepší, ale můžeme seřadit od nejnižšího po nejvyšší.

```
  > factor(c(4,2,3,1),labels = c("Ing.","maturita","Bc.","zakladni"))
[1] zakladni maturita Bc.      Ing.
Levels: Ing. maturita Bc. zakladni
```

3.5.Matice

Matice mají velmi široké využití nejen v matematice, ale i v technických a ekonomických vědách. Jejich uplatnění lze nalézt například v řešení lineárních rovnic. Matice je na rozdíl od

vektoru 2-dimenzionální datová struktura, která je tvořena řádky a sloupci. Stejně jako u vektorů je nutné, aby matice obsahovala pouze stejný datový typ (numerický, logický, textový).

Základní výraz pro vytvoření matice je příkaz „*matrix ()*", kde do argumentu nejdříve vkládáme název matice, poté počet řádků a jako poslední počet sloupců matice. Matice může být tvořena předem zadanými vektory, náhodnými čísly vygenerovanými funkcí „*runif()*" nebo posloupností čísel vytvořenou pomocí příkazu „*seq()*". V případě, že vektor, který má být součástí matice je kratší než počet prvků matice, je uplatněno pravidlo opakování složek vektoru tak dlouho, dokud jeho délka nedosáhne počtu prvků matice.

```
> y<-(c(runif(3,min = 5,max = 20)))
> matrix(y,4,5)
         [,1]     [,2]     [,3]     [,4]     [,5]
[1,] 12.72599 16.94560 18.04845 12.72599 16.94560
[2,] 16.94560 18.04845 12.72599 16.94560 18.04845
[3,] 18.04845 12.72599 16.94560 18.04845 12.72599
[4,] 12.72599 16.94560 18.04845 12.72599 16.94560
```

V některých případech je nutné využít nejprve transponování matice, které se provádí pomocí příkazu „*t ()*". *Další* funkce podobně jako například v programu MATLAB, je možné využít funkce *eigen()*, která vypíše vlastní čísla námi zadané matice. V případě lineárních rovnic je možné rovněž využít funkci „*det()*", pokud chceme znát determinant matice a mnoho jiných. [11]

3.6. Pokročilé datové struktury

Někdy zdrojová data vyžadují komplexnější uložení a zobrazení dat než je vektor nebo matice, naštěstí program R poskytuje řadu datových struktur, která zvládá tyto potřeby náročnějších uživatelů. Nejběžnější jsou datové tabulky, které se vytváří pomocí funkce „*data.frames()*", popřípadě zejména programátoři využívají seznamy pomocí funkce *list.* Tyto funkce budou více jednotlivě rozebrány v následujících kapitolách.

3.6.1. Datové tabulky

Jednou z nejužitečnějších možností podle knihy „*R for Everyone*", je schopnost tvořit datové tabulky pomocí funkce „*data.frames()*". V argumentu funkce zadáváme sloupcové vektory, které chceme, aby byly v datové tabulce obsaženy. Datovou tabulku si můžeme představit jako pracovní list v Microsoft Excel, přičemž také obsahuje řádky a sloupce. Ze statistického hlediska považujeme každý sloupce jako proměnnou a každý řádek jako pozorování. Program R organizuje tabulku jako samostatný sloupcový vektor, díky tomu je

možné, aby každý vektor měl jinou datovou strukturu, zároveň je ale nutné aby všechny vektory měly stejnou délku.

Existuje mnoho způsobů, jak sestavit datovou tabulku, nejjednodušší je zřejmě využít již výše zmíněnou funkci „*data.frames*“. Využijeme z předchozí **kapitoly 4.4.1** již nadefinovaného vektoru „*rocniobdobi*“. Dále nadefinujeme další dva vektory, *v* představuje vlhkost, a *t,* který bude představovat teplotu. Můžeme si je představit, jako že každý první den jednotlivého ročního období jsme změřily teplotu a vlhkost vzduchu a zapsali si je. Přičemž vlhkost je uváděna v procentech a teplota ve stupních Celsia. Je také nutné vytvořenou tabulku pojmenovat, po použití funkce „*data.frames*“, je vytvořena tabulka ale pouze načtena do paměti proměnných, proto je nutné ji vybraným názvem opět „zavolat“ a zobrazit.

```
> t<-(c(5,23,11,-5))
> v<-(c(33,25,41,7))
> tabulka<-data.frame(rocniobdobi,t,v)
> tabulka
  rocniobdobi  t  v
1        jaro  5 33
2        leto 23 25
3      podzim 11 41
4        zima -5  7
```

Pro větší přehlednost je lepší vždy vyplňovat proměnné celými názvy nikoliv pouze zkratkami. V programu R je možné přejmenovávat v tabulce do paměti nahrané proměnné. V tomto případě změníme zkratky na celé označení, pro větší přehlednost.

```
> merenidat<-data.frame(rocniobdobi,Teplota=t,Vlhkost=v)
> merenidat
  rocniobdobi Teplota Vlhkost
1        jaro       5       33
2        leto      23       25
3      podzim      11       41
4        zima      -5        7
```

Datové tabulky jsou velmi komplexní objekty s mnoha atributy, z nichž jsou nejčastěji používané počet řádků a sloupců. Při menších databázích je počet evidentní, ale při databázích o několika tisících řádcích a několika desítkami sloupců tento počet nemusí být evidentní. Proto se používají funkce „*nrow()*“ a „*ncol()*“, popřípadě pokud nás zajímají oba atributy, můžeme využít funkci „*dim()*“, která nám vypíše oba atributy v pořadí, kdy první je počet řádků a druhý je počet sloupců v tabulce.

```
> nrow(merenidat)
[1] 4
> ncol(merenidat)
[1] 3
> dim(merenidat)
[1] 4 3
```

Defaultně se vždy jako první sloupec vypisuje číslo řádku, což ne vždy je nutné a žádoucí. Například v naší tabulce s naměřenými hodnotami není nutné mít první sloupec s čísly řádku, ale bylo by lepší, kdyby místo čísla řádku byl rovnou roční období. K tomuto přejmenování slouží funkce „*rownames()*“. Po přejmenování řádků už tedy není nutné, aby v tabulce bylo dvakrát roční období. Proto vytvoříme novou tabulku, kde přejmenujeme řádky na roční období a poté vytvoříme tabulku pouze s teplotou a vlhkostí.

```
> merenidat2<-data.frame(teplota,vlhkost)
> merenidat2
  teplota vlhkost
1       5       33
2      23       25
3      11       41
4      -5        7
> rownames(merenidat2) <-c("jaro","leto","podzim","zima")
> merenidat2
       teplota vlhkost
jaro         5       33
leto        23       25
podzim      11       41
zima        -5        7
```

Další funkce užitečná zejména při velkých databází je funkce „*head()*“, která vypíše pouze námi zadaný počet řádků nikoliv všechny. Například námi vytvořené tabulce vypíšeme pouze první dvě hodnoty. [11,16]

```
> head(merenidat2,n=2)
     teplota vlhkost
jaro       5       33
leto      23       25
```

3.6.2. Seznam

Seznam je nejobecnější datová struktura, která se je v programu R. Seskupuje několik různých objektů do jednoho objektu o větším rozsahu. Jedná se o datovou strukturu, která se skládá z posloupností objektů, které jsou nazývány složkami. Každá složka může obsahovat objekt jakéhokoliv datového typu. Může tedy obsahovat vektory různých datových typů a délek, matice, pole, datové tabulky, funkce nebo jiné seznamy. Z tohoto důvodu jsou nejvhodnější jako výstupy jednotlivých procesů a analýz. Důležité je také si uvědomit, že datová tabulka je speciálním typem seznamu. Jedná se v podstatě o seznam, jehož složky jsou vektory o stejné délce a odpovídající pozici vyjadřující stejné případy. [11]

3.7. Grafické možnosti v R

Grafika jazyka R patří k nejvíce propracované části, nabízí velké množství funkcí pro tvorbu grafů včetně popisků a funkcím k přidávání nových částí grafů či změně již existujícího vzhledu grafu. R rozlišuje tři základní skupiny při tvorbě grafů:

- **High-level** – vytváří zcela nový kompletní graf
- **Low-level** – přidává do již existujícího grafu další informace
- **Interaktivní grafika –** umožnuje pomocí myši přidávat data do již dříve vytvořeného grafu [16]

3.7.1. High-level grafika

Všechny grafy vytvořené v programu R jsou nejprve vtvořeny pomocí high-level funkce, která vytváří nový graf, včetně vygenerování os, popisků a nadpisů. High-level funkce vždy vytvoří nový graf, pokud již byl nějaký graf vytvořen, přepíše jej. Mezi hlavní funkce při využívané při tvorbě high-level grafů patří:

Axes- implicitní hodnota TRUE, nastavení na hodnotu FALSE potlačuje vykreslování os

log - nastavením na hodnoty log="x", log="y", log="xy" budou mít vybrané osy logaritmické měřítko

main - textový řetězec pro název grafu, je umístěn nad grafem

sub - textový řetězec, je umístěn pod grafem a psán menším fontem

type - typ výstupního grafu

type="p" - vykresluje samostatné body (implicitní nastavení)

type="l" - linie

type="b" - přerušované linie s body

type="c" - přerušované linie bez bodů

type="o" - body navzájem spojené liniemi

type="h" - vertikální linie

type="s" - schodovitý graf s první linií horizontální

type="S" - schodovitý graf s první linií vertikální

type="n" - žádné vykreslování dat, vykresleny jsou pouze osy, rozsah

souřadného systému závisí na datech

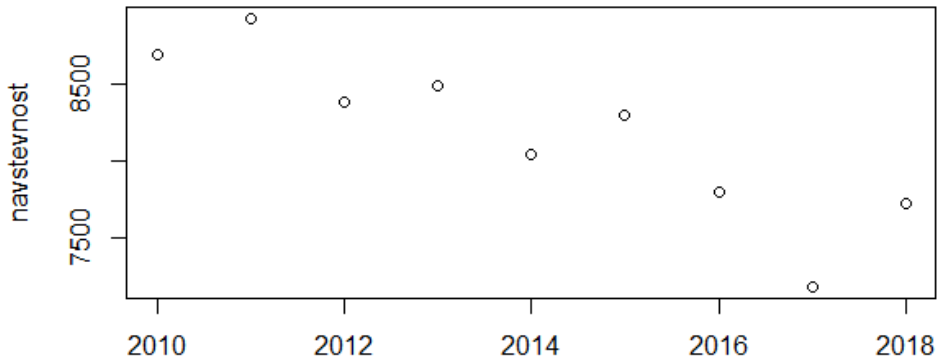
xlab, ylab - textové řetězce pro názvy os x a y, tyto argumenty mění implicitně zadané názvy os při volání high-level funkcí

Mezi nejpoužívanější typy patří tzv. plot grafy. Rozlišujeme grafy dva druhy „*plot(x)*“ a „*plot(x,y)*“. Rozdíl mezi těmito dvěma příkazy je v podstatě dat. Funkce *plot(x)* vytváří jednoduchý graf s indexy na ose *x* a hodnotami na ose *y*. Naopak funkce *plot(x,y)* vykreslí bodový graf hodnot *y* na pozicích *x*. Lze také nahradit argumenty funkce buď seznamem, obsahující dvě složky *x* a *y*, popřípadě maticí. Vzhled obou grafů jde upravit pomocí velmi široké škály volitelných argumentů, které budou vysvětleny více v kapitole 4.8.3.

Pro příklad vykreslíme graf návštěvnosti hokejového týmu HC Dynamo Pardubice. Z jejich webových stránek http://www.hcdynamo.cz, jsme převzali data průměrné návštěvnosti v základní části od roku 2010. Tyto hodnoty následně vykreslíme v některých námi výše vysvětlených možnostech grafu. Data nahrajeme pomocí funkce *data.frame()* do datové tabulky a následně vykreslíme základní *plot()*. [15]

```
> rok<-c(2010,2011,2012,2013,2014,2015,2016,2017,2018)
> navstevnost<-c(8682,8922,8380,8490,8037,8298,7792,7182,7720)
> prumernanavstevnsot<-data.frame(row.names = rok,navstevnost)
> prumernanavstevnsot
     navstevnost
2010        8682
2011        8922
2012        8380
2013        8490
2014        8037
2015        8298
2016        7792
2017        7182
2018        7720
> plot(rok,navstevnost)
```

Obrázek 9: Základní graf návštěvnosti

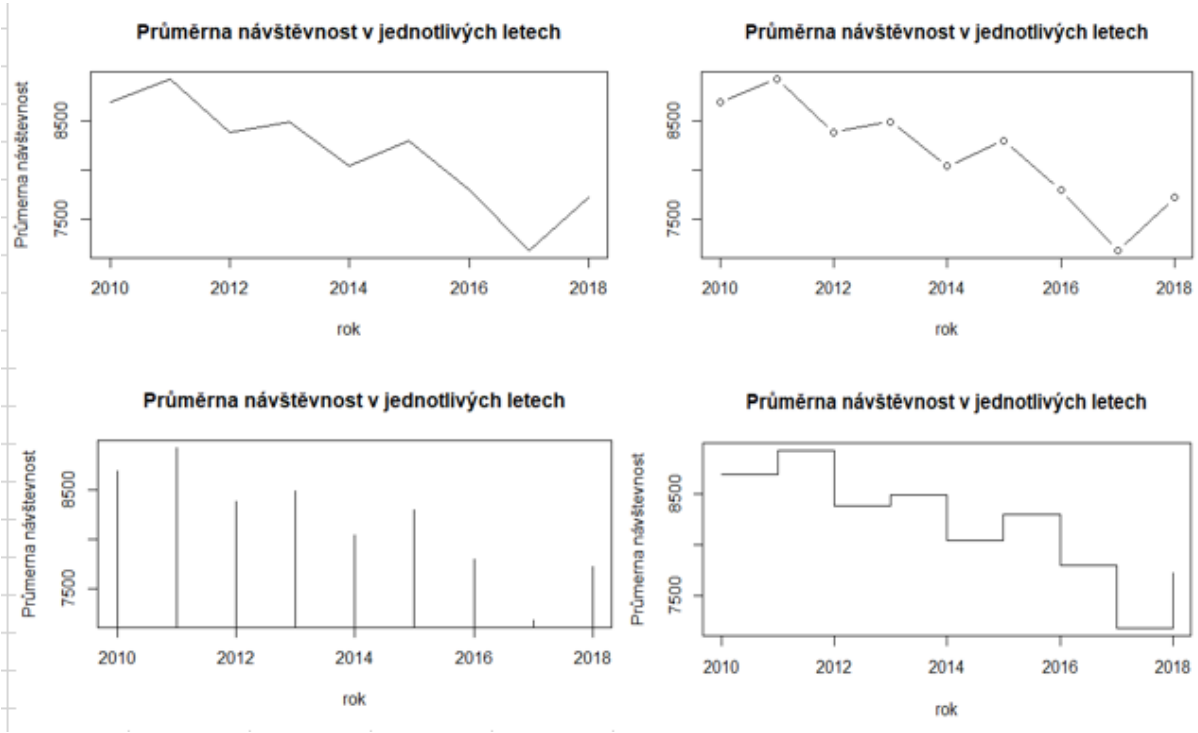


Zdroj: Vlastní zpracování v R z dat [9]

Na obrázku 10, je vidět, že jednotlivé osy jsou pojmenovaní podle popisků v datové tabulce. Základním vykreslením grafu je pouze bodový, který je možné nahradit jinou modifikací, spojnicového grafu. Zároveň je vidět i popisek grafu.

```
> plot(rok,navstevnost, type = "l",main="Průměrna návštěvnost v jednotlivýc
h letech",ylab = "Průmerna návštevnost")
> plot(rok,navstevnost, type = "b",main="Průměrna návštěvnost v jednotlivýc
h letech",ylab = "Průmerna návštevnost")
> plot(rok,navstevnost, type = "h",main="Průměrna návštěvnost v jednotlivýc
h letech",ylab = "Průmerna návštevnost")
> plot(rok,navstevnost, type = "s",main="Průměrna návštěvnost v jednotlivýc
h letech",ylab = "Průmerna návštevnost")
```

Obrázek 10: Jednotlivé možnosti tvorby grafů



Zdroj: Vlastní zpracování v R z dat [9]

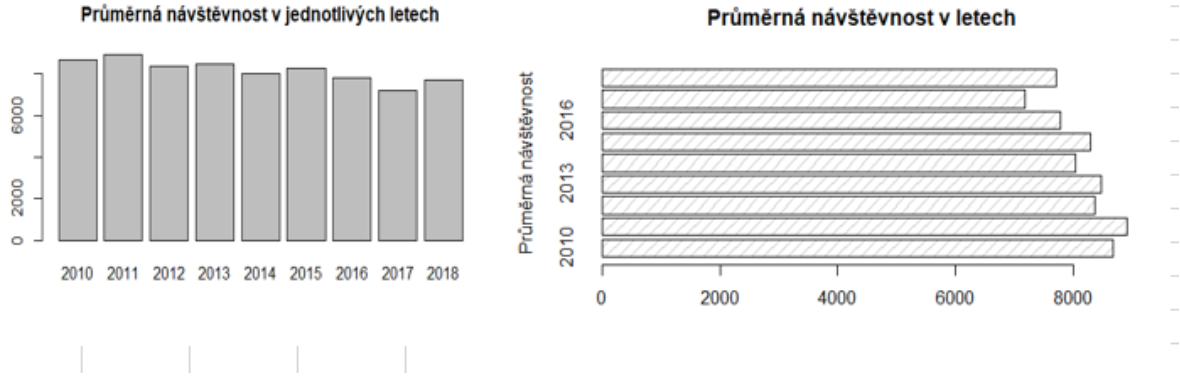
V případě menšího počtu dat jako v tomto případě je přehlednější data vykreslit pomocí sloupcového grafu. Pro tento se používá funkce *barplot()*. V programu R, jak bylo výše zmíněno je možná spousta modifikací pomocí příkazů např.:

- **width** – udává šířku sloupce
- **space** - udává velikost, která bude vynechána mezi jednotlivými sloupci
- **names.arg** – vektor názvů ke každému sloupci nebo skupině
- **legend.text** – vektor textových řetězců uvádějící názvy v legendě
- **horiz** – pokud nastavíme hodnotu na TRUE, vykreslí sloupce horizontálně
- **densinity** – nastavuje hustotu šrafování
- **angle** – úhel pro sklon šrafování

Na obrázku 11 je vidět vykreslené možnosti grafu, základní a poté rozšířenými popisky a šrafováním.

```
> barplot(navstevnost,names.arg = rok,
main = "Průměrná návštěvnost v jednotlivých letech")
> barplot(navstevnost,names.arg = rok,
ylab = "Průměrná návštěvnost",main = "Průměrná návštěvnost v letech",
angle = 45,density = 12,horiz = TRUE)
```

Obrázek 11:Sloupcový graf návštěvnosti



Zdroj: Vlastní zpracování v R na základě dat [9]

Jako další možnosti je možné využít funkci *pie()*, která vytvoří koláčový graf z nezáporného vektoru čísel. Ve statistickém zpracování dat je možní využívat funkci *boxplot()*, která vytvoří krabicový graf, v němž se dá upravovat, zda konce grafu, budou jeho minima a maxima nebo kvantily podle našeho výběru. V případě, že potřebujeme vykreslit více datových řad do jednoho grafu, nejjednoduší je využít funkci *matplot()*. Pro možnost vykreslení 3-D grafů slouží funkce *persp()*. [16]

3.7.2. Low-level grafika

V některých případech se stává, že grafické funkce popsané v kapitole 4.8.1 nevykreslují přesný typ grafu, který bychom preferovali. V těchto případech se využívají tzv. low-level funkce, pomocí nichž se vytváří celý graf samostatně po částech přidáváním dalších informací do již vytvořeného grafu.

Nejvíce využívané funkce pro tvorbu low-level grafiky jsou:

- ***Points()*** **–** vytvoří do grafu body daných tvarů a barev na souřadnice *x* a *y*
- ***Lines()*** **–** vykreslí lomenou čáru mezi souřadnicemi *x* a *y*
- ***Segmens()*** **–** vykreslí úsečky mezi body o souřadnicích x_0 a y_0 x_1 a x_1
- ***Abline()*** **–** vykreslí přímku se směrnicí *b* a průsečníkem s osou *y*
- ***Legend()*** **–** na souřadnice *x* a *y* vytvoří legendu (vektor textových řetězců)
- **Title() –** vypíše název grafu, podtitulek a popisky os

Všechny výše popsané funkce se ještě více dají více upravit pomocí dalších příkazů vysvětlených v následující kapitole. [16]

3.7.3. Funkce *par()*

Pro změny jednotlivých parametrů aktuálního grafu složí funkce *par ()*. Vyvolání upravení nastavení pomocí funkce *par () nastálo* dokud není grafické okno zavřeno. S otevřením nového grafického okna jsou všechna předchozí nastavení ignorována.

- ***Adj()*** **–** zarovnání textu pro název grafu a popisky
  - 0 – zarovná vlevo
  - 1 – zarovná vpravo
  - 0,5 – zarovnat horizontálně na střed
- ***Ann()*** **–** v případě hodnoty TRUE vypisuje názvy os a nadpisy v případě hodnoty FALSE potlačuje anotace
- **Col() –** barva bodů, spojnic nebo obrysů. Vždy se spojuje s výrazem, čeho chceme změnit barvu např. popisky *col.axis(),* pro titulek a podtitulek *col.lab().* Jednotlivé odstíny je možné nastavovat pomocí čísla nebo textového řetězce viz. Tabulka 2.
- **Bg() –** nastavuje pozadí grafu viz. Tabulka 2

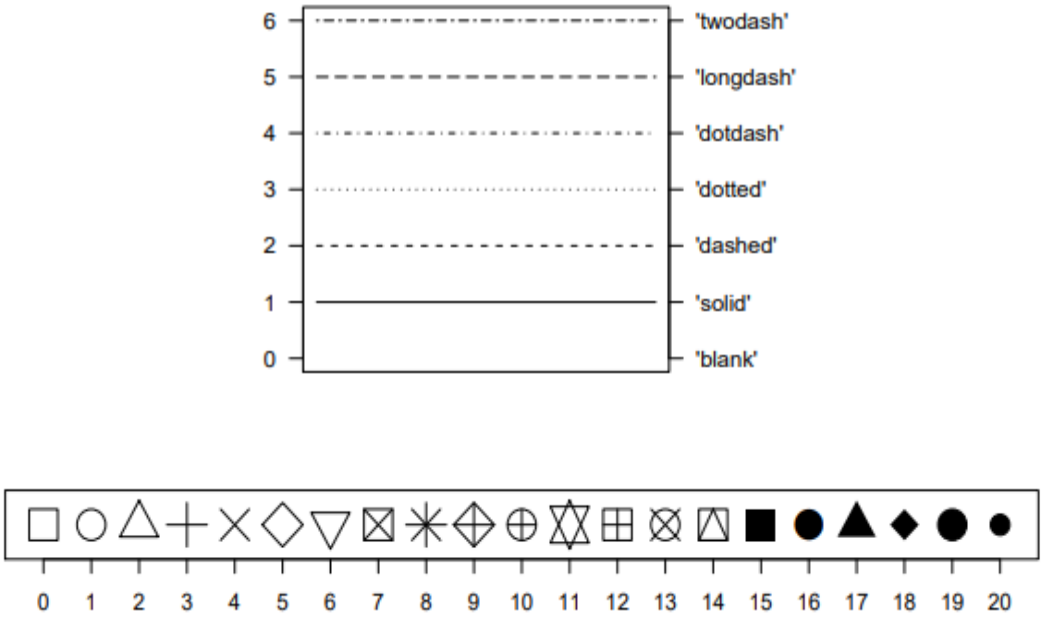
Tabulka 2: Jednotlivé barvy v programu R

Barva	*Textový řetězec*	*Numerická hodnota*
Bílá	**„white“**	**0**
Černá	**„black“**	**1**
Červená	**„red“**	**2**
Zelená	**„green“**	**3**
Modrá	**„blue“**	**4**
Modrozelená	**„cyan“**	**5**
Fialová	**„magenta“**	**6**
Žlutá	**„yellow“**	**7**
šedá	**„gray“**	**8**

Zdroj: Vlastní zpracování v R z dat [16]

- **Las() –** numerická hodnota pro otočení popisků os
  - **0 –** ***x*** vodorovně a *y* svisle
  - **1 –** *x* a *y* vodorovně
  - **2 –** *x* svisle a *y* vodorovně
  - **3 –** *x* i *y* svisle
- **Lty() –** numerická hodnota udává typ čáry viz. Obrázek 12
- **Lwd() –** udává tloušťku čáry
- **Pch() –** numerická hodnota udává znak pro vykreslení bodů. Jednotlivé znaky jsou uvedeny na obrázku 12. Znaky je možné ještě vyplnit volitelnou barvou pomocí funkce *bg()* a změnit barvu hranice symbolu pomocí funkce *col ()*. [14],[15]

Obrázek 12: Jednotlivé znaky pro vykreslení bodů a typy čar



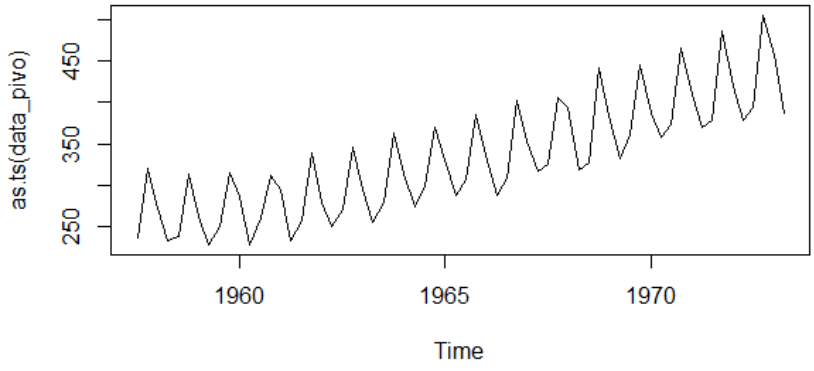
Zdroj: [16]

4. DEKOMPOZICE ČASOVÉ ŘADY V R

V této kapitole bude vysvětleno, postup modelace časové řady dekompoziční metodou v programu R. Nejprve dekompozice řady na jednotlivé složky a poté modelace trendové složky pomocí lineární regrese. Data pro modelaci jsem vybral výrobu australského piva, kde jsou k dispozici čtvrtletní data a uvádějí výrobu v megalitrech v Austrálii. Tato data jsou volně dostupná v databázi programu R, ovšem nejprve je nutné naistalovat knihovnu „fpp“, která je nutná pro analýzu časových řad. Data začínají rokem 1958, končí rokem 1973, stáří dat pro představení modelace dat není důležité. Tyto data jsou vybrána pro dostupnost, kdy každý čtenář této práce může zkusit tento postup na totožných datech.

Jak bylo vysvětleno v kapitole 2, existují dvě možnosti chování časové řady: aditivní a multiplikativní. Pokud se rozptyl nemění v čase, jedná se o aditivní model a naopak pokud ano jedná se o model multiplikativní. Rozhodnutí je možné udělat na základě grafické analýzy. Na obrázku 13 je vidět graf časové řady. Na první pohled vidíme, že řada obsahuje trendovou složku, sezonní a nesystematickou složku, kterou obsahuje každá řada. Naopak vidíme, že zde schází cyklická složka, jelikož všechny periody jsou kratší než jeden rok, proto ji nebudeme dále ve výpočtech zahrnovat. Zároveň se je zřejmé, že přes nárůst hodnot zůstává rozptyl konstantní a budeme používat aditivní dekompozici.

Obrázek 13:Graf časové řady výroby australského piva



Zdroj: Vlastní zpracování v R

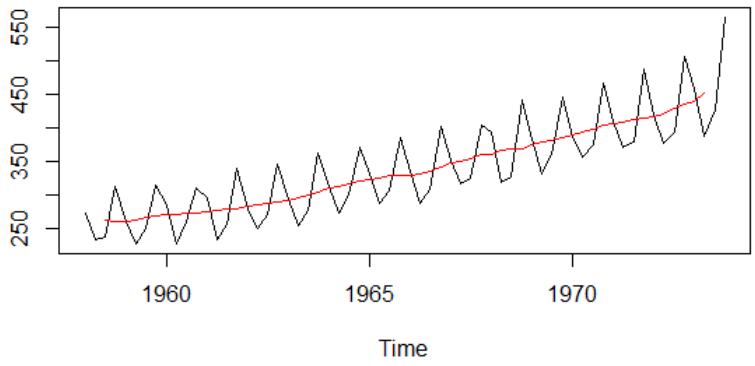
4.1. Trendová složka

Jako první začneme rozpoznáním trendové složky časové řady. Pro sezonní očištění dat využijeme centrovaný klouzavý průměr, pro jehož použití je důležité znát délku periody. V našem případě, kde se jedná o čtvrtletní data, je perioda 4. Pro využití klouzavého je nejprve nutné mít nainstalovanou knihovnu „*forecast*“. Poté vytvoříme novou proměnou a pojmenujme

ji jako *„trend_pivo“*. Funkce pro klouzavé průměry se jmenuje *ma (), do* které se zadává proměnná, na kterou aplikujeme klouzavé průměry. Druhý argument je *„order“,* který určuje z kolika hodnot je průměr tvořen a poslední je *„centre“,* kde nastavuje hodnotu *T* a *F (TRUE a FALSE),* podle toho zda se jedná o centrovaný průměr. Na obrázku14 je vidět přidána funkce trendu vytvořeného pomocí klouzavých průměrů.

```
> install.packages("forecast")
> library(forecast)
> trend_pivo=ma(data_pivo,order=4,centre = T)
> plot(as.ts(trend_pivo))
> lines(trend_pivo,col="red")
```

Obrázek 14: Porovnání sezonně očištěných dat s původními



Zdroj: Vlastní zpracování v R

Na obrázku 14 je vidět trendová funkce. Vybrat správný model trendové funkce obecně patří k velmi obtížným úkolům analytika, kde je důležité odhad, zkušenost a taky občas v neposlední řadě i trošku štěstí při volbě správného modelu. Většinou není možné řadu, jednoznačně identifikovat a je proto nutné zkusit vždy více modelů a poté je porovnat, který lépe vystihuje variabilitu souboru. Jednotlivé modely porovnáváme pomocí přesnosti prognóz, které jsou vysvětleny v kapitole 2.1.3, z nichž nejvíce využívat je koeficient determinace. Na první pohled, je zřejmé, že by to mohla být lineární trendová funkce. Na začátku a na konci nemá tvar přímky, ale lehce připomíná parabolu, takže by mohl být i kvadratický trend. Pomocí lineární regrese uděláme odhad obou trendů a podle hodnoty koeficientu determinace, určíme lepší model. Pro zjednodušená zavedeme vektor *t*, který bude obsahovat čísla 1-64, která budou značit pořadí pozorované hodnoty.

4.1.1. Lineární trendová funkce

Začneme jednoduchou lineární regresí, která pracuje se dvěma veličinami neboli proměnnými: závislou proměnou, kterou v našem případě bude číslo pozorování, a nezávislou proměnnou, kterou je výroba piva. Obecný tvar lineární regrese je:

$$y_t = \beta_0 + \beta_1 * t. \tag{4.1}$$

Regrese se vyvolává pomocí příkazu *lm()*. Jako první se vždy uvádí nezávislá proměnná. Poté pomocí znaku ~ oddělíme nezávislou proměnnou a vypíšeme závislou proměnou. Těch může víc více a oddělujeme mezi sebou znakem +. Pojmenujeme výstup jako „modelace_lin_trend“, ta výsledná funkce bude zapsaná jako:

```
> modelace_lin_trend<-lm(data_pivo~t)
> modelace_lin_trend

Call:
lm(formula = data_pivo ~ t)

Coefficients:
(Intercept)            t
    232.771        3.278
```

Následným zavoláním vidíme odhad jednotlivých členů lineární rovnice, z něhož vyplývá, že konstanta je rovna 232,77 a hodnota směrnice je 3,28. Tyto hodnoty můžeme dosadit do rovnice a dostáváme

$$y_t = 232{,}771 + 3{,}278 * t + \varepsilon_t. \tag{4.2}$$

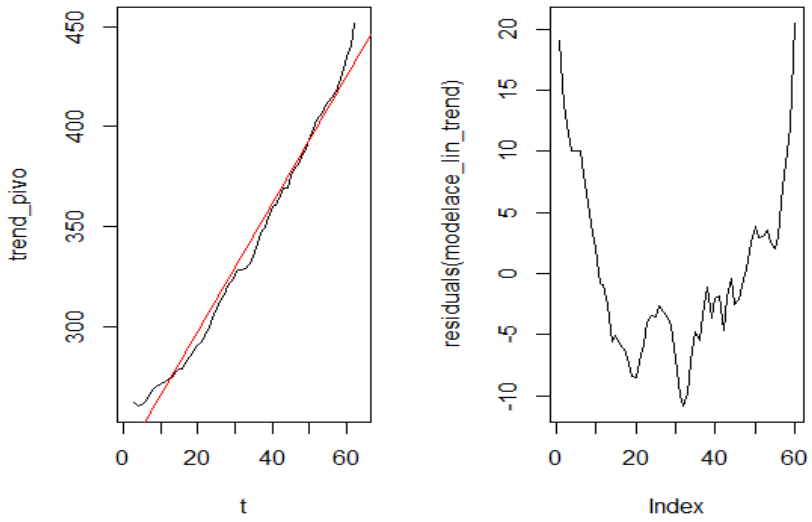
Pro další vlastnosti vytvořené regrese vyvoláme pomocí funkce „*summary()*“.

```
> summary(modelace_lin_trend)

Call:
lm(formula = trend_pivo ~ t)

Residuals:
    Min      1Q  Median      3Q     Max
-10.860  -4.861  -1.951   3.096  20.523

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 233.74433    1.94839  119.97   <2e-16 ***
t             3.18723    0.05291   60.24   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 7.097 on 58 degrees of freedom
  (4 observations deleted due to missingness)
Multiple R-squared:  0.9843,   Adjusted R-squared:  0.984
F-statistic:  3629 on 1 and 58 DF,  p-value: < 2.2e-16
```

Z těchto statistik například vidíme, rozložení reziduí, včetně jejich minima a maxima a kvantilů. Dále vidíme rozšířené statistiky k odhadu parametrů, zda jsou statisticky významné podle p-hodnoty nebo jejich standartní chybu. Velmi důležitou statistikou je koeficient determinace, podle kterého budeme posuzovat kvalitu modelu. V tomto případě koeficient determinace je 98,43% a modifikovaný index má hodnotu 98,4%. Tento model tedy vysvětluje více než 98% variability původních dat, což je velmi uspokojující výsledek modelace. Do grafu očištěných dat od sezonnosti jsme vložily náš model a na obrázku 15 vidíme, jak náš odhad na začátku a na konci pozorování podhodnocuje skutečnost. Toto by mohl odstranit kvadratický trend. Toto je také vidět na grafu reziduí, která jsou větší na začátku a na konci pozorování.

```
> split.screen(c(2,1))
[1] 1 2
> screen(1)
> plot(t,trend_pivo,type = "l")
> abline(233.744,3.18723,col="red")
> screen(2)
> plot(residuals(modelace_lin_trend),type="l")
```

Obrázek 15:Lineární trendová funkce a výsledná rezidua



Zdroj: Vlastní zpracování v R

4.1.2. Kvadratická trendová funkce

Jak bylo zmíněno v předchozí kapitole, koncové hodnoty lineární trendová funkce odhadla neuspokojivě pro nás. Proto totožnými daty proložíme kvadratickou trendovou funkci, jejíž základní rovnice má tvar

$$y_t = \beta_0 + \beta_1 * t + \beta_2 * t^2. \qquad (4.3)$$

Nejprve je nutné vytvořit novou proměnou t^2,kterou pojmenujeme t_2. Vytvoříme ji pomocí

```
> t_2<-t^2
```

Následný postup je stejný jako v předchozí kapitole, pouze teď máme dvě závislé proměnné, které vstupují do modelu.

```
> modelace_kvadr_trend<-lm(trend_pivo~t+t_2)
> modelace_kvadr_trend

Call:
lm(formula = trend_pivo ~ t + t_2)

Coefficients:
(Intercept)            t          t_2
  251.75845      1.63908      0.02382
```

Vidíme, že hodnoty jednotlivých členů a můžeme je dosadit do původní rovnice, která bude vypadat

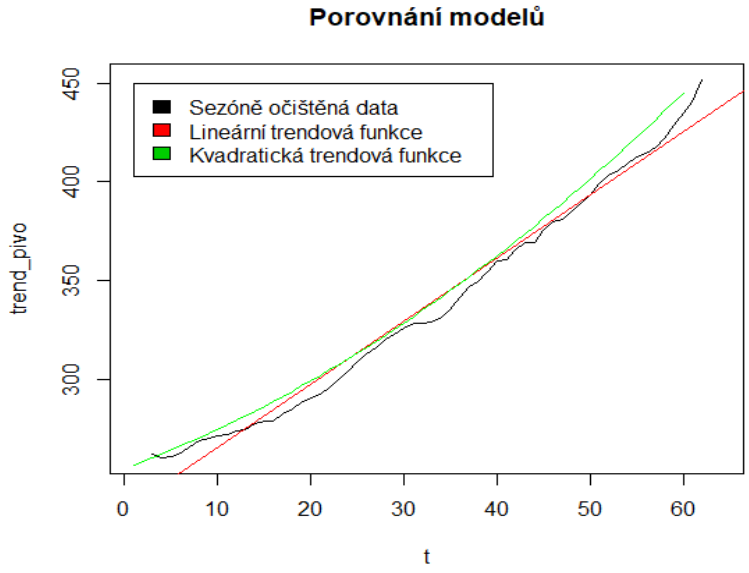
$$y_t = 251{,}758 + 1{,}639 * t + 0{,}024 * t^2. \tag{4.4}$$

```
> summary(modelace_kvadr_trend)

Call:
lm(formula = trend_pivo ~ t + t_2)

Residuals:
    Min      1Q  Median      3Q     Max
-5.3226 -2.5496  0.1036  2.0013  6.9387

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 2.518e+02  1.316e+00  191.38   <2e-16 ***
t           1.639e+00  9.280e-02   17.66   <2e-16 ***
t_2         2.382e-02  1.389e-03   17.15   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.885 on 57 degrees of freedom
  (4 observations deleted due to missingness)
Multiple R-squared:  0.9974,   Adjusted R-squared:  0.9974
F-statistic: 1.113e+04 on 2 and 57 DF,  p-value: < 2.2e-16
```

Po vyvolání funkce „*summary ()“,* vidíme, že všechny proměnné jsou statisticky významné. Také vidíme, že koeficient determinace je ještě vyšší než u předchozí modelace a to 99,74% a modifikovaný index determinace nabývá totožné hodnoty. Kvadratický trend vysvětluje skoro dokonale variabilitu dat. Vytvoříme graf, kde budou skutečná data, lineární a kvadratická odhadovaná funkce. Můžeme také porovnat jednotlivá rezidua obou modelů. Toto je vidět na obrázku 16.

```
> plot(t,trend_pivo,type = "l",main = "Porovnání modelů")
> lines(predict(modelace_lin_trend,col="red"))
> lines(predict(modelace_kvadr_trend),col="green")
> legend(1,450,legend = c("Sezóně očištěná data","Lineární trendová funkce"
,"Kvadratická trendová funkce"),col = c("black","red","green"),fill =1:3)
```

Obrázek 16:Porovnání lineárního a kvadratického modelu

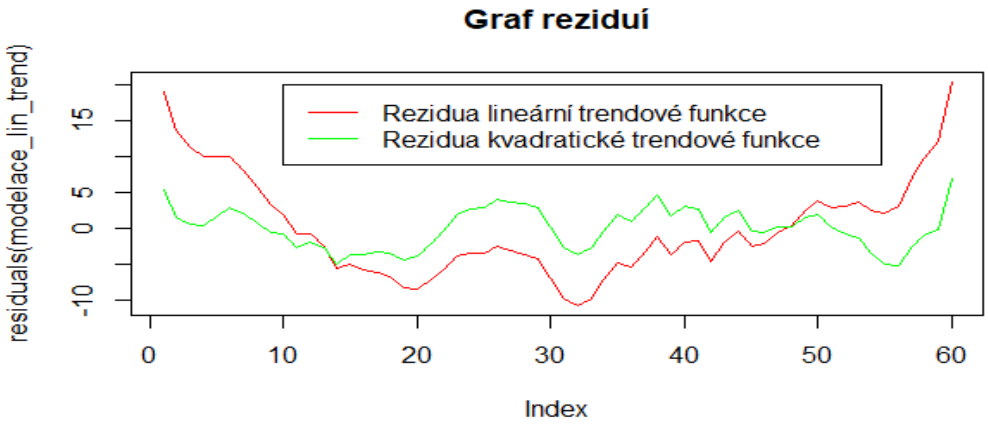


Zdroj: Vlastní zpracování v R

Z grafu reziduí obou modelů na obrázku 17 můžeme vidět, že kvadratická lineární trendová funkce nenabývá tak vysokých hodnot zejména na začátku a na konci pozorování. Lineární funkce by nadále ještě zvětšovala svá rezidua, jak je dobře vidět z obou obrázků.

```
> plot(residuals(modelace_lin_trend),type="l",col="red",main = "Rezidua")
> lines(residuals(modelace_kvadr_trend),col="green")
> legend(15,20,legend = c("Lineární trendová funkce","Kvadratická trendová
funkce"),col = c("red","green"),lty = 1:1)
```

Obrázek 17: Graf reziduí odhadů trendové funkce



Zdroj: Vlastní zpracování v R

4.2.Sezonní složka

V předchozí kapitole jsme se zabývali trendovou složkou časové řady, která byla sezonně očištěná. V této kapitole se budeme zabývat sezónností, kterou můžeme vyjádřit tak, že od původních dat odečteme trendovou složku a vliv reziduální složky nebudeme uvažovat.

$$y_t = T_t + S_t => S_t = y_t - T_t. \tag{4.5}$$

Na obrázku 18 je vidět graf sezonní složky.

```
> sezonnost_pivo<-data_pivo - trend_pivo
> plot(as.ts(sezonnost_pivo),main = "Sezonní složka")
```

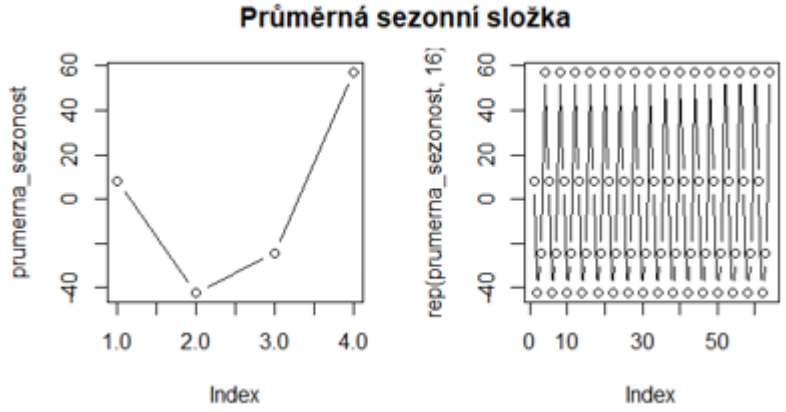
Obrázek 18: Graf sezonní složky

Zdroj: Vlastní zpracování v R

Pro dekompozici časové řady je nutné vypočítat průměrnou sezonnost. Časovou řadu rozdělíme podle pozorování na období tzv. jedné sezonnosti. Pro program R je toto důležité pro vytvoření matice, ze které budeme dělat průměrné hodnoty. Následně provedeme transformaci, jejíž důvod je aby každý sloupec obsahoval pouze stejného období (stejný den, měsíc rok nebo v našem případě čtvrtletí). Nakonec vypočítáme průměrnou hodnotu každého sloupce. Počet řádků volíme podle délky periody jedné sezonnosti, tedy 4 na čtvrtletních datech.

```
> prumerna_sezonost
[1]   7.916667 -42.233333 -24.516667  57.383333

> split.screen(c(1,2))
[1] 1 2
> screen(1)
> plot(prumerna_sezonost,type="b")
> screen(2)
> plot(as.ts(rep(prumerna_sezonost,16))
> plot(rep(prumerna_sezonost,16),type = "b")
```

Obrázek 19: Průměrná sezonní složka



Zdroj: Vlastní zpracování v R

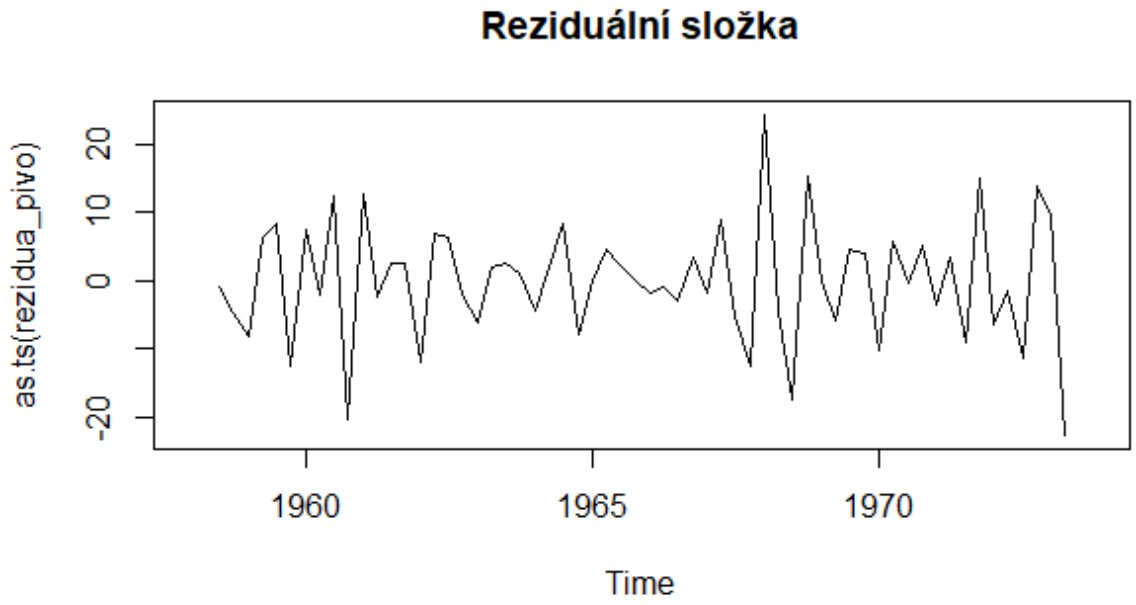
4.3. Reziduální (náhodná) složka

Poslední složkou, kterou jsme se zatím nezabývali, je reziduální (náhodná) složka. V dekompoziční metodě se říkáme, že po odstraněný trendové, sezonní a cyklické složky je zůstatek reziduální složka, která je tvořena náhodnými vlivy a dále ji nijak nemodulujeme.

$$\varepsilon_t = y_t - T_t - S_t \tag{4.6}$$

```
> rezidua_pivo<-data_pivo - trend_pivo - sezonnost_pivo
> plot(as.ts(rezidua_pivo),main="Reziduální složka")
```

Obrázek 20: Reziduální složka



Zdroj: Vlastní zpracování v R

4.4.Predikce do budoucna

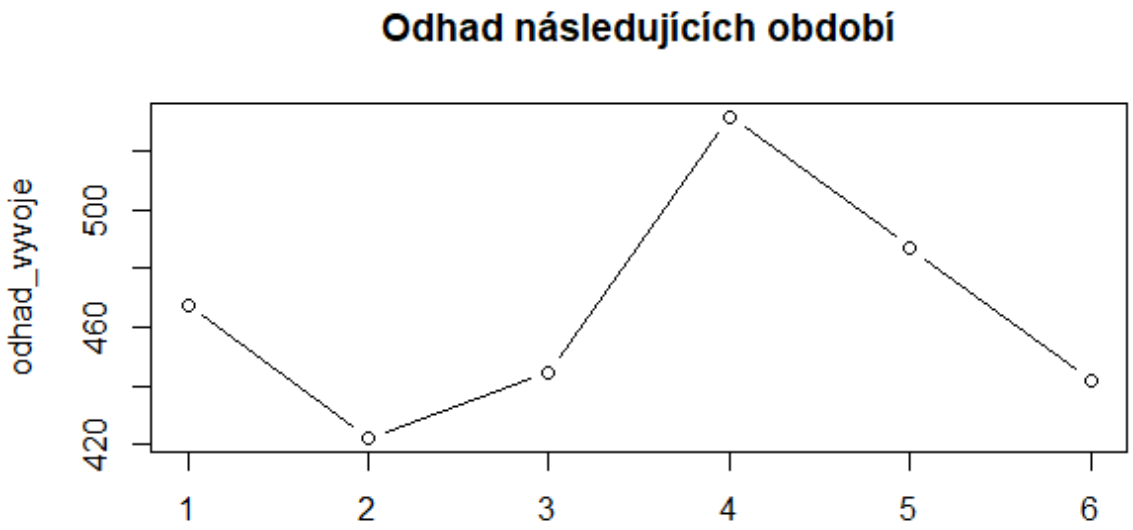
Důvodem pro dekompozici určitě časové řady může být pochopení odchylek od trendu. Například při prodeji zmrzliny, kdy v zimě klesá prodej, nemusí znamenat špatnou prodejní strategii firmy. Druhým důvodem může být snaha predikovat budoucí vývoj, díky čemuž můžeme uzpůsobit budoucí očekávání, firemní strategie atd. Odhad budoucí hodnot může uživatel udělat pomocí pouhého *t*, které dosadí do předchozí rovnice časové řady. Kdy známe jednotlivé parametry trendové funkce, které byly již dříve odhadnuty a průměrné sezonní hodnoty pro jednotlivé čtvrtletí. Tedy pro pozorování t = 65 by odhad vypadal následovně (uvažujeme kvadratický trendové funkce, z důvodu vyššího koeficientu determinace)

$$y_{65} = 251{,}758 + 1{,}639 * 65 + 0{,}024 * 65^2 + 7{,}917 = 467{,}61 \qquad (4.7)$$

Pro odhad následujících 5 období vytvoříme vektor *b*, který bude reprezentovat pozorování číslo 65-70. Tato čísla dosadíme předchozí rovnice a výpočtem dostaneme odhad budoucích období.

```
v1<-b*1.639+b*b*0.024
> v1+251.78+prumerna_sezonost
[1] 467.6317 422.2647 444.8123 531.5913 487.0517 441.8767
> plot(as.ts(odhad_vyvoje,type = "b")
```

Obrázek 21: Predikce budoucího vývoje



Zdroj: Vlastní zpracování v R

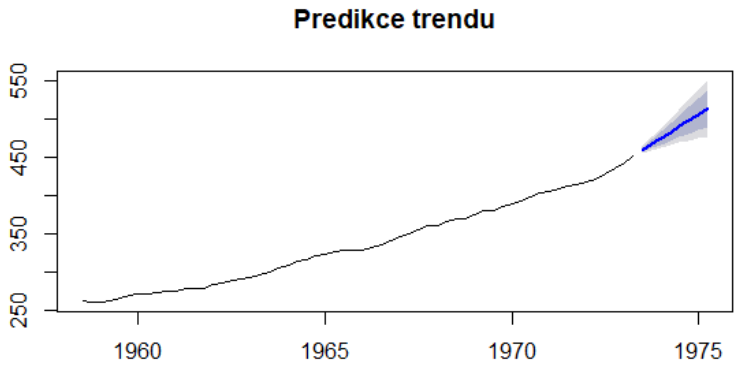
Případně v programu R po doinstalování knihovna "*forecast*" je možné využít funkci *"predict"*, která tento odhad vývoje je schopna sama odhadovat, z předchozích dat. Tato funkce umí vytvořit bodovou předpověď tak i interval spolehlivost. Rozdíl oproti předchozím číslům

je v modelaci trendové složky, kdy tato funkce dělá její odhad na základě klouzavých průměrů, naopak náš odhad vychází z modelace trendové složky trendovými funkcemi.

```
> predict(data_pivo)
        Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
1974 Q1       474.9743 456.4008 493.5478 446.5685 503.3800
1974 Q2       411.3428 394.8954 427.7903 386.1886 436.4971
1974 Q3       441.9349 423.7181 460.1517 414.0747 469.7951
1974 Q4       566.7689 542.4915 591.0463 529.6399 603.8979
1975 Q1       500.6573 478.2104 523.1043 466.3277 534.9870
1975 Q2       433.2885 412.8321 453.7450 402.0031 464.5739
```

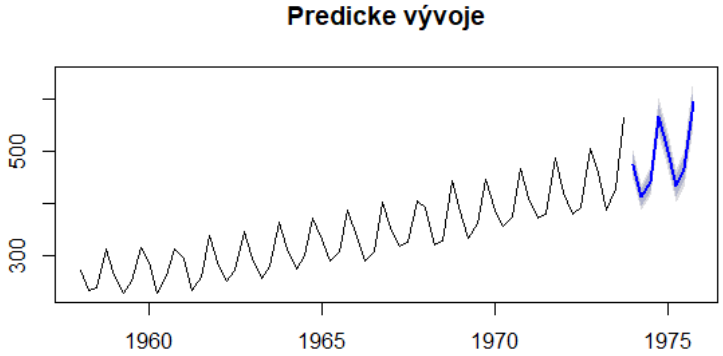
Na obrázku 22 je vidět, jak program pomocí vějířových grafů modeluje budoucí vývoj trendové složky a poté i vývoj celé analyzované časové řady

Obrázek 22: Predikce trendové složky



Zdroj: Vlastní zpracování v R

Obrázek 23: Predikce budoucího vývoje



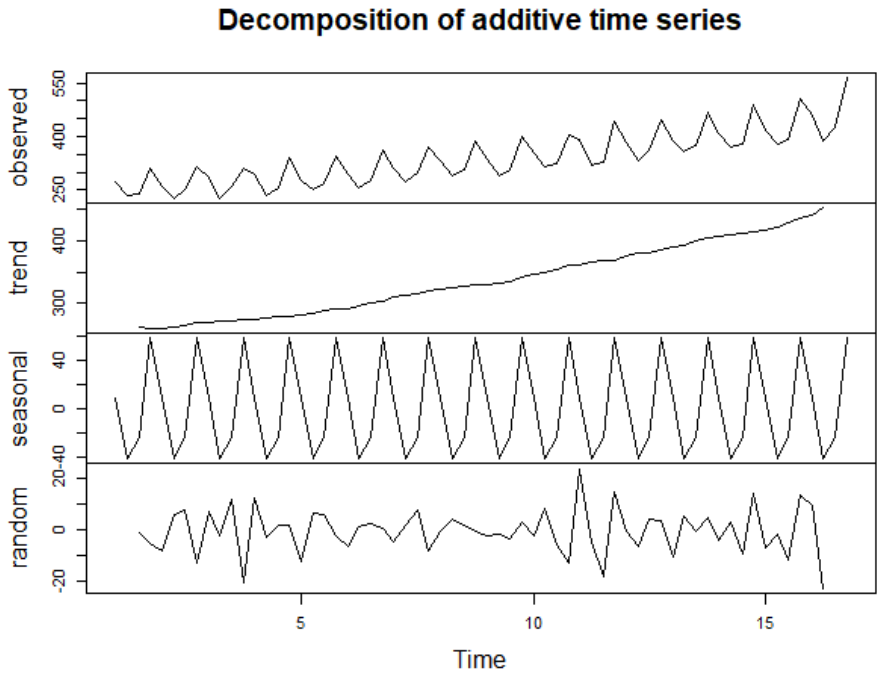
Zdroj: Vlastní zpracování v R

4.1. Funkce *decompose*

V předchozích kapitolách byl vysvětlen postup postupné dekompozice po jednotlivých složkách. V programu R je funkce *decompose*, která všechny předchozí kroky zvládá sama bez uživatelovy asistence. Respektive sama pomocí centrovaného klouzavého průměru sezoně očistí řadu, udělá průměr sezonní složky a vypočítá na závěr residuální hodnotu. Před jejím použitím je nejprve nutné uložit periodu modelované časové řady, kterou program neumí sám odhadnout. Tato délka je nutná pro sezonní očištění. Druhý argument je druh dekompozice, respektive zda se jedná o *„aditivní“* nebo *„multiplikativní“*. Poté již program sám provede automaticky všechny kroky. Na obrázku 24 je vidět, že grafy jednotlivých pozorování jsou totožné jako ta data z předchozích kapitol.

```
> casova_rada<-ts(data_pivo,frequency = 4)
> dekompozice_pivo=decompose(casova_rada,"additive")
> plot(dekompozice_pivo)
```

Obrázek 24: Graf funkce decompose



Zdroj: Vlastní zpracování v R

5. BOX-JENKINSOVONOVA METODA V R

V této kapitole bude vysvětlena modelace finančních časových řad pomocí Box-Jenkinsonovi metodologie. Cílem této části bude na reálných finančních datech sestrojit vhodný model a ten použít na odhad budoucích hodnot. V této práci se budou použity základní modely Box-Jenkinsonovi metodologie. V dnešní době je možné aplikovat na data rozšířené

modelace, například pomocí modelu ARCH je možné nejprve ověřit, zda se rozptyl hodnot vyvíjí v čase a dále ho modelovat. Pokud se rozptyl v čase vyvíjí, je možné modelovat ho pomocí GARCH nebo EGARCH modelů ale těmito se nebudeme v této práci zabývat.

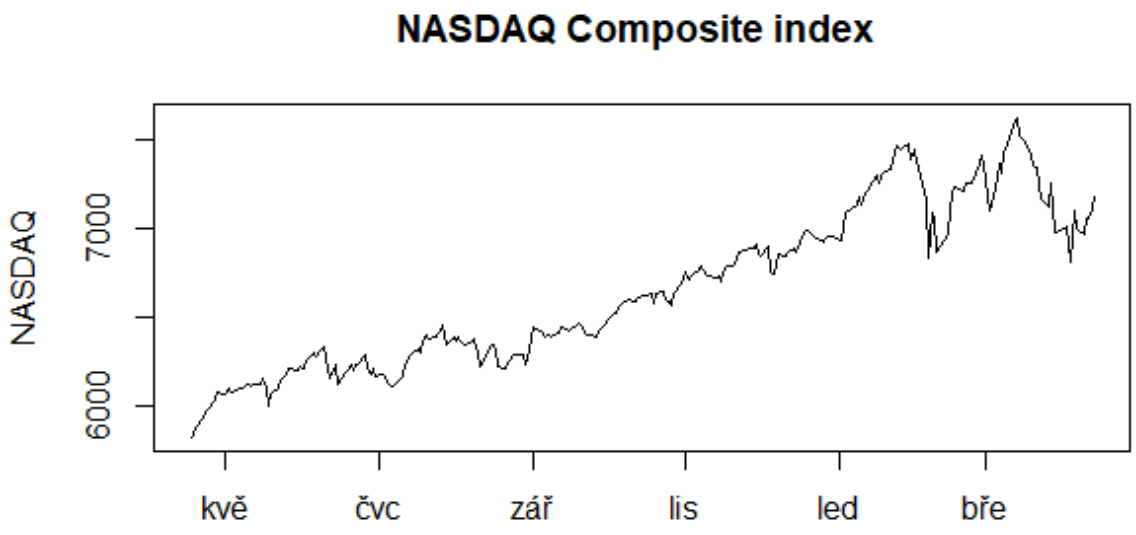
Jako data jsem si zvolil týdenní index NASDAQ Composite od 16. 4. 2017 – 13. 04. 2018 volně dostupný na https://finance.yahoo.com. Jako hodnotu beru vždy páteční konečnou hodnotu v okamžiku zavření burzy NASDAQ. Tento index reprezentuje index akciových trhů běžných akcií a podobných cenných papírů, které jsou kótovány na akciovém trhu NASDAQ. Zároveň spolu s indexem Dow Jones Average a S&P 500 je jedním z nejvíce sledovaných indexů na amerických burzách. Index NASDAQ Composite tvoří společnosti zabývající se informačními technologiemi a biotechnologií. Celkem obsahuje více než 3 000 společností, se kterými se obchoduje na burce NASDAQ. Mezi největší a nejznámější společnosti, které jsou v tomto indexu obsazeny, patří například Apple, Google, Microsoft nebo Amazon.

5.1.Modelace dat

Než přikročíme k modelaci, je nutné nejprve nahrát data do programu R. Data se stáhnout z YAHOO Finance ve formě souboru Microsoft Excel s koncovkou csv. Pro nahrání není potřeba dodatečně stahovat knihovnu, využijeme knihovnu již dříve staženou pro import souborů xls., jelikož program R nedělá rozdíly mezi jednotlivými soubory Microsoft Excel. Z nahraných dat vytvoříme novou proměnou *index_nasdaq*, ze které následně vytvoříme graf hodnot zobrazený na obrázku 25.

```
> index_nasdaq<-NASDAQ$NASDAQ
> plot(index_nasdaq,type="l",main = "NASDAQ Composite index ")
```

Obrázek 25: Graf indexu NASDAQ Composite

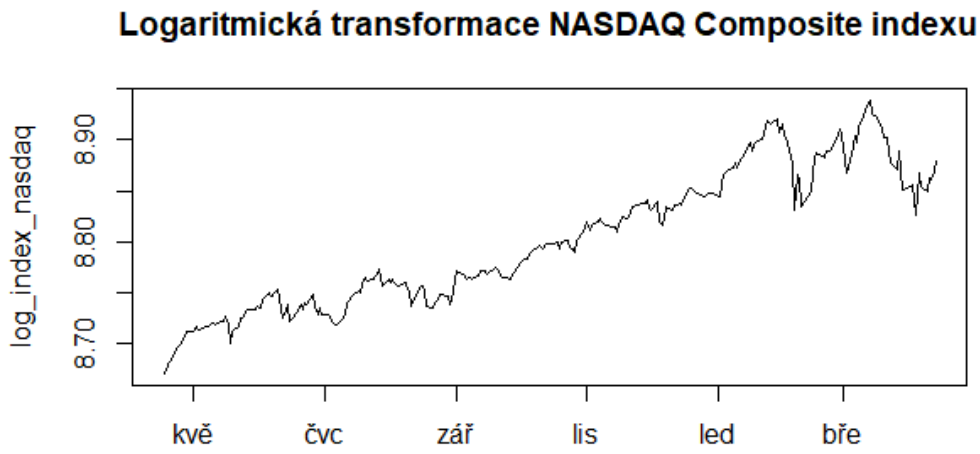


Zdroj: Vlastní zpracování v R na základě dat z [19]

Z Obrázku 26 je možné odhadnout, že bude zřejmě nestacionární, což ověříme následně pomocí ACF a PACF. Jelikož se jedná o denní finanční řadu, nejprve ji stabilizujeme pomocí logaritmické transformace a tím vytvoříme novou proměnou *log_index_nasdaq*, ze které vytvoříme graf zobrazený na obrázku 26.

```
> log_index_nasdaq<-ln(index_nasdaq)
> plot(NASDAQ$t,log_index_nasdaq,type = "l",main = "Logaritmická transformace NASDAQ Composite indexu")
```

Obrázek 26: Logaritmická tranforamce NASDAQ

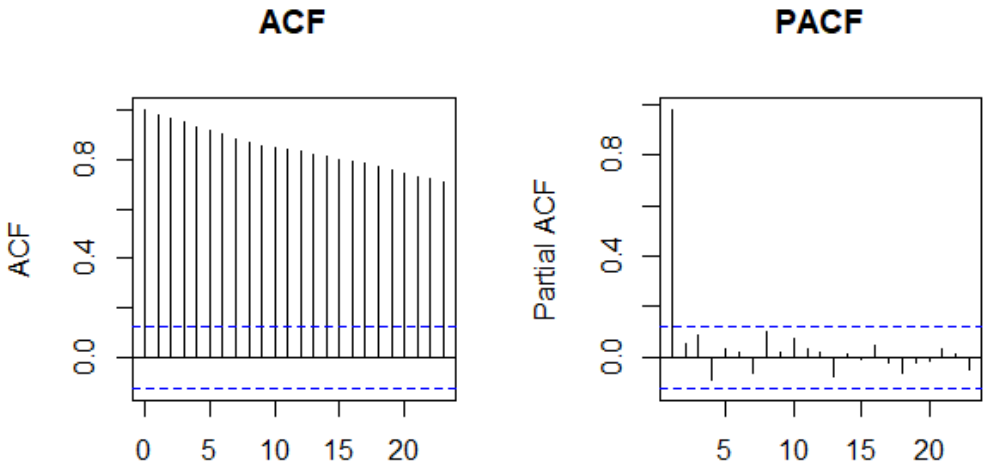


Zdroj: Vlastní zpracování v R na základě dat z [19]

Z obrázku 27, na kterém je graf autokorelace a parciální autokorelace vidíme, že se potvrdila předchozí domněnka o nestacionární časové řadě. Hodnot ACF klesají lineárním trendem, naopak první hodnota PACF je blízká jedné a ostatní jsou velmi malé.

```
> split.screen(c(1,2))
[1] 1 2
> screen(1)
> acf(log_index_nasdaq,main="ACF")
> screen(2)
> pacf(log_index_nasdaq,main="PACF")
```

Obrázek 27: ACF a PACF logaritmu NASDQ indexu

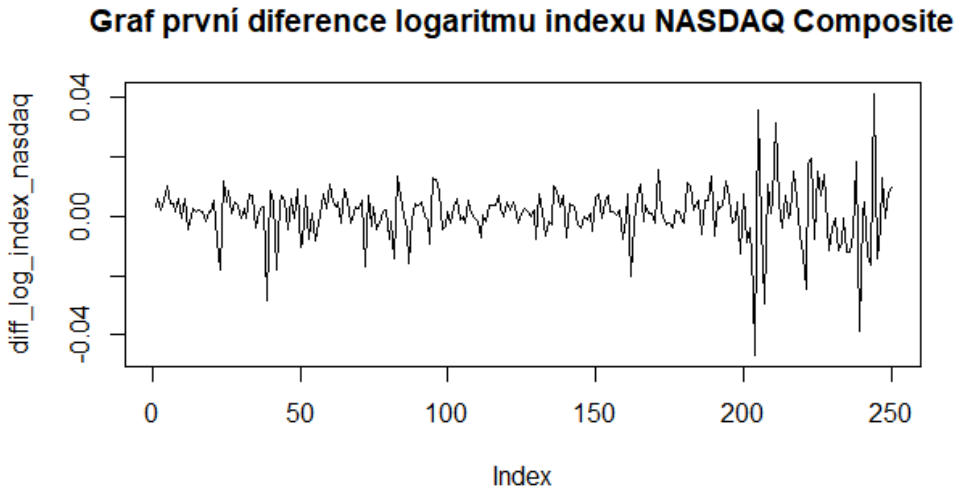


Zdroj: Vlastní zpracování v R na základě dat z [19]

Řadu budeme stacionarizovat pomocí první diference. A následně vytvoříme z první diference a zobrazíme její graf. K vytvoření první diference použijeme funkci *diff()* a dále pracujeme s nově vytvořenou proměnnou pomocí první diference.

```
> diff_log_index_nasdaq<-diff(log_index_nasdaq)
> plot(diff_log_index_nasdaq,type = "l",main = "Graf první diference logari
tmu indexu NASDAQ Composite")
```

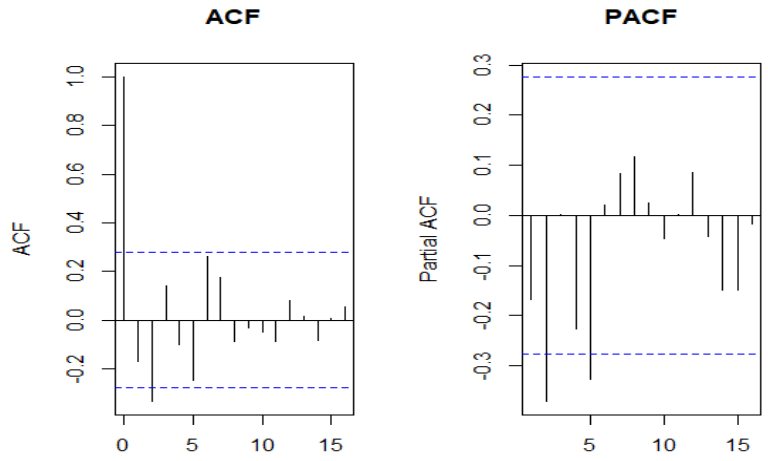
Obrázek 28: Graf první diference logaritmických hodnot NASDAQ



Zdroj: Vlastní zpracování v R na základě dat z [19]

Z obrázku 28 je patrné že se již nejedná o nestacionární časovou řadu a můžeme přistoupit k odhadu modelu. Z první diference logaritmických hodnot vytvoříme ACF a PACF zobrazeny na Obrázku 29. Z něho je vidět, že obě funkce mají sinusoidní tvar a bude se jednat o smíšený model ARMA.

Obrázek 29:ACF a PACF logaritmických hodnot NASDAQ



Zdroj: Vlastní zpracování v R na základě dat z [19]

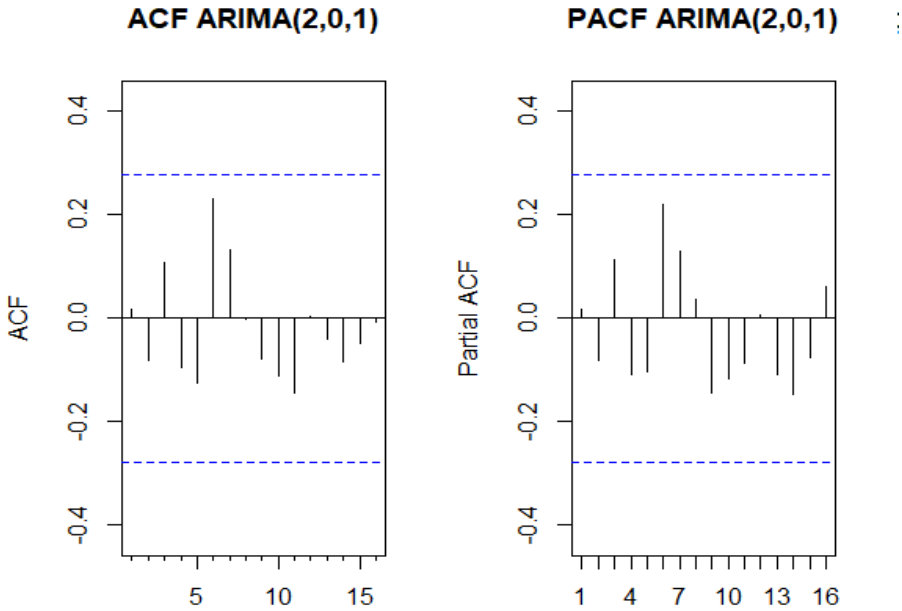
V grafu autokorelační funkce vidíme zejména první hodnotu velmi odchýlenou a v grafu parciální autokorelace je druhá hodnota velmi odchýlená. Proto zkusíme model ARIMA(2,0,1) a z jeho reziduí vytvoříme graf autokorelace a parciální autokorelace. Funkce ARIMA v programu R, má velké množství argumentů, kde můžeme sami nastavovat zpoždění, nastavovat jakou metodou chceme odhadovat parametry , zda počítat s konstantou nebo bez atd. My budeme používat standartní nastavení a do argumentu budeme zadávat proměnnou, na které chceme provést odhad modelu. Dále o jaký model se jedná, musíme zadávat jako vektor hodnot. Pokud by data obsahovala sezonnost můžeme buď doinstalovat knihovnu *"Sarima"*, která obsahuje ve které můžeme upravit sezonost modelu nebo přímo ve funkci ARIMA při vyplnění argumentu *seasonal(order=c())*, rovnou nastavit sezonní AR a MA hodnotu.

Pro otestování zda jsou všechny parametry statisticky významné je nutné nainstalovat knihovnu *„lmtest"* ,která obsahuje funkci *coeftest()*. Tato funkce provádí výpočet *p*-hodnoty na jejímž základě rozhodujeme o statistické významnosti odhadnutých hodnot. Také vždy pro zjednodušení zadávání modelu v budoucnu, vytvoříme novou proměnou, která bude mít název jako model, který odhadujeme.

```
> ARIMA2_0_1<-arima(diff_log_index_nasdaq,order = c(2,0,1))
> screen(1)
> Acf(residuals(ARIMA2_0_1) ,main="ACF")
> screen(2)
> Pacf(residuals(ARIMA2_0_1,main="PACF"))
```

```
Call:
arima(x = diff_log_index_nasdaq, order = c(2, 0, 1))

Coefficients:
         ar1      ar2      ma1  intercept
      0.3756  -0.2619  -0.7293     0.0039
s.e.  0.2181   0.1673   0.1979     0.0009

sigma^2 estimated as 0.0003216:  log likelihood = 129.76,  aic = -249.52
```

Obrázek 30: ACF a PACF modelu ARIMA(2,0,1)



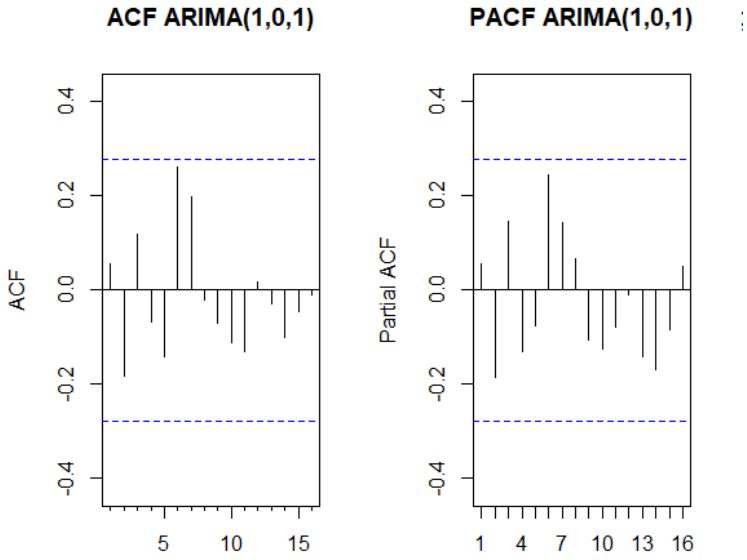
Zdroj: Vlastní zpracování v R na základě dat z [19]

```
> coeftest(ARIMA2_0_1)

z test of coefficients:

             Estimate  Std. Error z value  Pr(>|z|)
ar1        0.37555645  0.21813468  1.7217 0.0851289 .
ar2       -0.26191881  0.16727344 -1.5658 0.1173926
ma1       -0.72934577  0.19787285 -3.6859 0.0002279 ***
intercept  0.00390268  0.00085174  4.5820 4.606e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Z obrázku 30 vidíme autokorelaci a parciální autokorelaci vytvořenou z reziduálních hodnot funkce ARIMA (2,0,1), která dopadla úspěšně. Tedy žádná hodnota není statisticky významná a můžeme tento model podle tohoto kritéria prohlásit za vhodný model. Pomocí funkce *coeftest()*, jsme zjistili, že obě hodnoty autoregresní funkce považujeme za statisticky nevýznamné. Proto ubere jeden řád autoregresní funkce a opět zkusíme odhadnout parametry modely ale ARIMA (1,0,1). Stejně jako v předchozím případě odhadneme parametry, poté jejich významnost a nakonec vytvoříme graf autokorelace reziduí.

```
> ARIMA1_0_1<-arima(diff_log_index_nasdaq,order = c(1,0,1))
>
> screen(1)
> Acf(residuals(ARIMA1_0_1),main="ACF ARIMA(1,0,1)")
> screen(2)
> Pacf(residuals(ARIMA1_0_1),main="PACF ARIMA(1,0,1)")
> ARIMA1_0_1

Call:
arima(x = diff_log_index_nasdaq, order = c(1, 0, 1))

Coefficients:
         ar1      ma1  intercept
      0.5193  -1.0000     0.0042
s.e.  0.1279   0.0642     0.0004

sigma^2 estimated as 0.0003121:  log likelihood = 129.45,  aic = -250.89
> coeftest(ARIMA1_0_1)

z test of coefficients:

             Estimate  Std. Error  z value  Pr(>|z|)
ar1        0.51925006  0.12786749   4.0608  4.89e-05 ***
ma1       -0.99999711  0.06421037 -15.5738 < 2.2e-16 ***
intercept  0.00420125  0.00035843  11.7214 < 2.2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Obrázek 31: ACF a PACF modelu ARIMA(1,0,1)



Zdroj: Vlastní zpracování v R na základě dat z [19]

Z obrázku 31 vidíme, že žádná hodnota není statisticky významná a leží uvnitř tolerančních mezí. Zároveň konstanta, autoregresní funkce i model klouzavých průměrů jsou statisticky významné. Můžeme považovat tento model jako uspokojivý. Po dosazení parametrů modelu ARIMA (1,0,1) na hodnoty první diference zlogaritmovaných hodnot dostáváme rovnici

$$y_t = 0{,}0042 + 0{,}5193 * y_{t-1} - 0{,}9999 * \varepsilon_{t-1} - \varepsilon_t. \qquad (4.8)$$

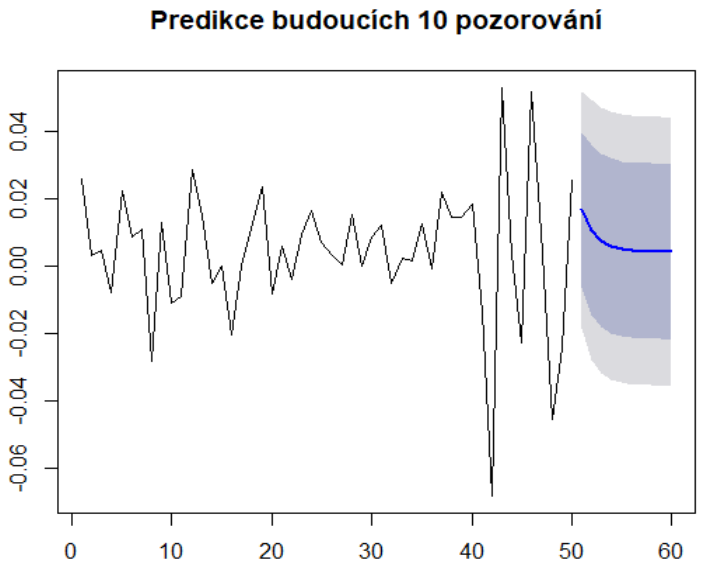
5.2. Predikce budoucích hodnot

Po nalezení uspokojivého modelu, můžeme přistoupit k predikci budoucích hodnot. K tomuto účelu slouží funkce *forecast()*, do které jako argument zadáme proměnnou, na jejímž základě chceme predikovat budoucí data a počet hodnot, které chceme odhadnout. V našem případě děláme predikci na základě modelu ARIMA (1,0,1) a budeme predikovat 10 hodnot.

```
> forecast(ARIMA1_0_1,n=10)
   Point Forecast        Lo 80      Hi 80       Lo 95      Hi 95
51    0.016623923 -0.006228152 0.03947600 -0.01832531 0.05157315
52    0.010651723 -0.014520515 0.03582396 -0.02784589 0.04914934
53    0.007550658 -0.018211405 0.03331272 -0.03184902 0.04695033
54    0.005940430 -0.019978364 0.03185922 -0.03369894 0.04557980
55    0.005104318 -0.020856572 0.03106521 -0.03459944 0.04480807
56    0.004670168 -0.021302061 0.03064240 -0.03505093 0.04439126
57    0.004444735 -0.021530550 0.03042002 -0.03528103 0.04417050
58    0.004327679 -0.021648430 0.03030379 -0.03539935 0.04405471
59    0.004266898 -0.021709433 0.03024323 -0.03546047 0.04399427
60    0.004235337 -0.021741054 0.03021173 -0.03549212 0.04396280
```

Funkce *forecast()*, nabízí jak bodový odhad následujících hodnot anebo intervalový odhad, který je rozlišen na 80% interval a 95 % interval spolehlivosti. Na obrázku 32 je vidět graf predikovaných hodnot, který ukazuje jak bodový odhad i taktéž oba intervalové odhady.

Obrázek 32: Predikce budoucích hodnot na základě modelu ARIMA(1,0,1)



Zdroj: Vlastní zpracování v R na základě dat z [19]

5.3.Funkce *auto.arima()*

Program R po nainstalování knihoven *forecast* a *tseries,* obsahuje funkci *auto.arima().* Jak již název napovídá, jedná se o funkci, která automaticky vybírá nejlepší model ARIMA, na základě zadaných dat. Do argumentu funkce povinně zadáváme, na kterých datech chceme modelaci provést. Mezi nepovinné argumenty patří, maximální hodnota AR a MA, která je přípustná pro nás, zda se jedná o sezonní data či ne, zda chceme odhadovat s konstantou nebo ne, zda provést Box-Cox transformaci atd. Nejlepší odhad je určen podle nejmenší hodnoty AIC (Akaike information criterion).

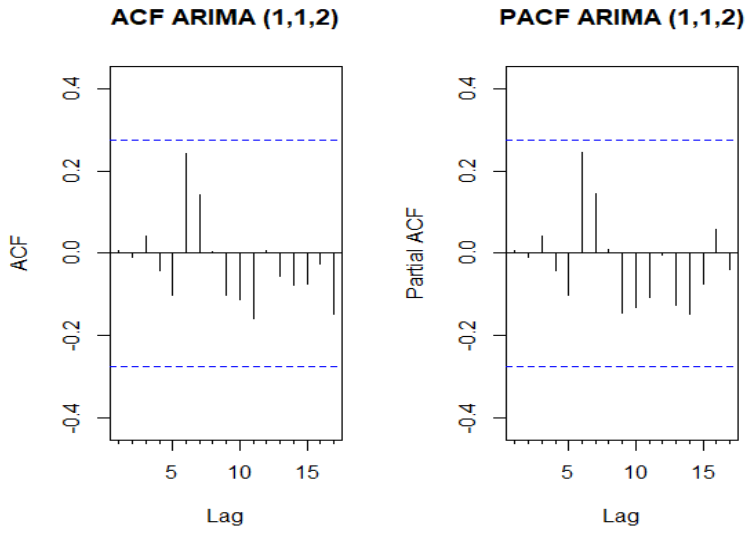
V našem případě nebudeme nastavovat žádná omezující parametry a provedeme modelaci na původních datech (zlogaritmovaných). Program sám zhodnotí, zda se jedná o stacionární nebo nestacionární řadu a v případě potřeby vytvoří diferenci.

```
> library(tseries)
> library(forecast)
> auto.arima(log_index_nasdaq)
Series: log_index_nasdaq
ARIMA(1,1,2) with drift

Coefficients:
          ar1      ma1      ma2  drift
      -0.0062  -0.3148  -0.4676  0.004
s.e.   0.5685   0.5746   0.1729  0.001

sigma^2 estimated as 0.0003425:  log likelihood=130.28
AIC=-250.55   AICc=-249.19   BIC=-240.99
```

Z výsledků vidíme, že byla také provedena první diference, ale funkce určila jako nejlepší model ARIMA (1,1,2). Pro verifikaci modelu necháme vytvořit ACF a PACF reziduí a zda jsou jednotlivé parametry významné.

```
> split.screen(c(1,2))
[1] 1 2
> screen(1)
> Acf(residuals(auto.arima(log_index_nasdaq)),main="ACF ARIMA (1,1,2)")
> screen(2)
> Pacf(residuals(auto.arima(log_index_nasdaq)),main="PACF ARIMA (1,1,2)")
> coeftest(auto.arima(log_index_nasdaq))

z test of coefficients:

             Estimate  Std. Error z value  Pr(>|z|)
ar1       -0.00633622  0.56565843 -0.0112  0.991063
ma1       -0.31468894  0.57149790 -0.5506  0.581881
ma2       -0.46758433  0.17277456 -2.7063  0.006803 **
intercept  0.00404047  0.00095227  4.2430 2.206e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Obrázek 33: ACF a PACF modelu ARIMA(1,1,2)



Zdroj: Vlastní zpracování v R na základě dat [19]

Z těchto výsledků vyplývá , že parametr AR1 a MA1 není statisticky významný. Toto je nevýhoda funkce *auto.arima()*, kdy není možné nastavit například, aby všechny proměnné výsledného modelu byly statisticky významné parametry. Výhoda je, že nemusíme s původními parametry provádět žádné úpravy nebo stacionarizovat. Pouze pokud se jedná o finanční data, zlogaritmujeme je a o zbytek se funkce postará sama.

6. ČASOVÉ ŘADY V MICROSOFT EXCEL

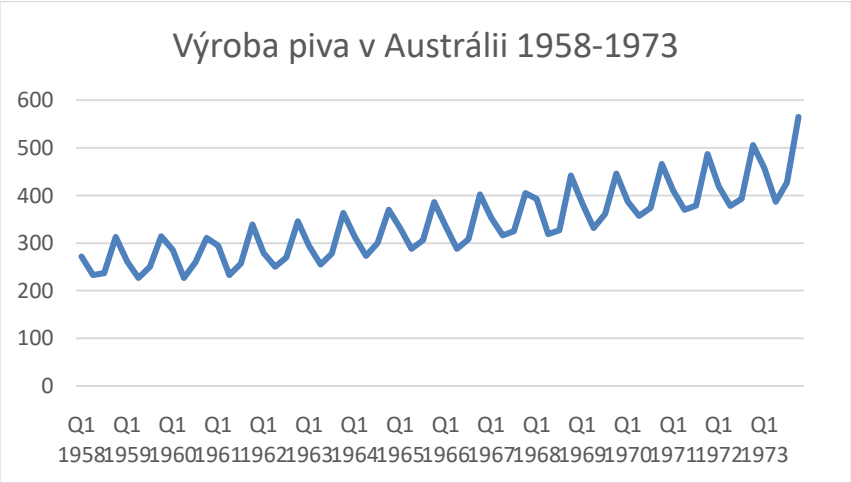
Program Microsoft Excel je nejpoužívanější tabulkový procesor na světě, který dále v této práci budeme nazývat pouze jako Excel. Je součástí balíčku Microsoft Office od firmy Microsoft. Má velmi široké využíti od výpočtů, grafických nástrojů, kontingenčních tabulek až po tvorbu maker pomocí vlastního programovacího jazyka Visual Basic for Applications.

Základní ovládání funkcí v programu Excel probíhá pomocí mřížky buněk uspořádaných v očíslovaných řádcích a sloupcích, které jsou označeny písmeny. Disponuje velmi velkou databází funkcí, jako například statistické, datumové, finanční, logické a mnoho dalších. Pomocí doplňku Power Querry a Power Pivot, je možné spravovat velké databáze o několika statisících řádků. Z dat je také možné tvořit grafy a histogramy, ale je omezen trojrozměrným prostorem. Uživatelsky přívětivé je rovněž propojování tabulek a grafů do ostatních programů z balíčku Microsoft Office jako je například Word nebo PowerPoint.

6.1. Dekompozice v programu Excel

V dekompozici v Excelu využijeme stejná data, která jsme používali v programu R. Tedy výrobu piva v Austrálii od roku 1958 – 1973, po jednotlivých kvartálech. Z hodnot vytvoříme graf zobrazený na obrázku 34. Při aditivní dekompozici budeme uvažovat trendovou, sezonní a reziduální složku.

Obrázek 34: Graf výrovy piva v austrálii



Zdroj: Vlastní zpracování v Excelu

6.1.1. Modelace trendové složky v Excelu

Nejprve se zaměříme na trendovou složku. Abychom ji mohli začít modelovat, musíme nejprve sezoně očistit pomocí centrovaného průměru délky 4. Po očištění na obrázku 35 vidíme

očištěnou trendovou složku, kterou budeme také modelovat pomocí lineárního a kvadratického trendu za pomocí funkce *Regrese*.

Obrázek 35: Trendová složka

Zdroj: Vlastní zpracování v Excelu

Na obrázku 36 vidíme výstup z funkce Regrese. Hodnota absolutního členu je 233,74 a lineárního 3,18, přičemž oba považujeme na základě p-hodnoty za staticky významné. Koeficient determinace má hodnotu 98,42%. Tato čísla jsou totožná jako předchozí modelace v R, jelikož oba softwary využívají pro odhad parametrů metodu nejmenších čtverců.

Obrázek 36: Výsledek odhadu lineárního trendu

Výsledek odhadu Lin. Trendu

Regresní statistika	
Násobné R	0,992103341
Hodnota spolehlivos	0,984269039
Nastavená hodnota s	0,983997816
Chyba stř. hodnoty	7,097337344
Pozorování	60

ANOVA

	Rozdíl	*SS*	*MS*	*F*
Regrese	1	182800,5365	182800,5365	3628,997
Rezidua	58	2921,587447	50,37219737	
Celkem	59	185722,124		

	Koeficienty	*Chyba stř. hodnoty*	*t Stat*	*Hodnota P*
β0	233,7443341	1,948390734	119,9678945	3,33E-71
β1	3,187225618	0,052907778	60,24115429	5,34E-54

Zdroj: Vlastní zpracování v Excelu

Dále zkusíme modelovat trend pomocí kvadratické trendové funkce. Postup je stejný jen přidáme druhou nezávislou proměnou t^2.

Obrázek 37: Výsledek odhadu kvadratického trendu

Výsledek odhadu Kvard. Trendu

Regresní statistika	
Násobné R	0,998722
Hodnota spolehlivosti R	0,997446
Nastavená hodnota spol	0,997356
Chyba stř. hodnoty	2,884711
Pozorování	60

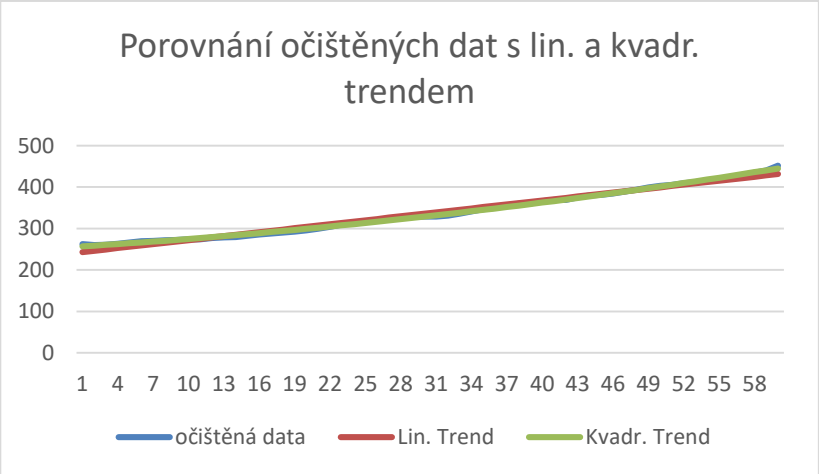
ANOVA	*Rozdíl*	*SS*	*MS*	*F*
Regrese	2	185247,8	92623,9	11130,59
Rezidua	57	474,3288	8,321559	
Celkem	59	185722,1		

	Koeficienty	*a stř. hodi*	*t Stat*	*Hodnota P*
β0	251,7585	1,315518	191,3759	9,51E-82
β1	1,639075	0,092803	17,66196	8,73E-25
β2	0,023818	0,001389	17,14895	3,61E-24

Zdroj: Vlastní zpracování v Excelu

Z Obrázku 37 vidíme vyplývá, že jednotlivé odhady křivky jsou velmi podobné. Toto zjištění jen potvrzuje velmi vysokou hodnotu indexu determinace, z něhož jako přesnější vychází kvadratický trend.

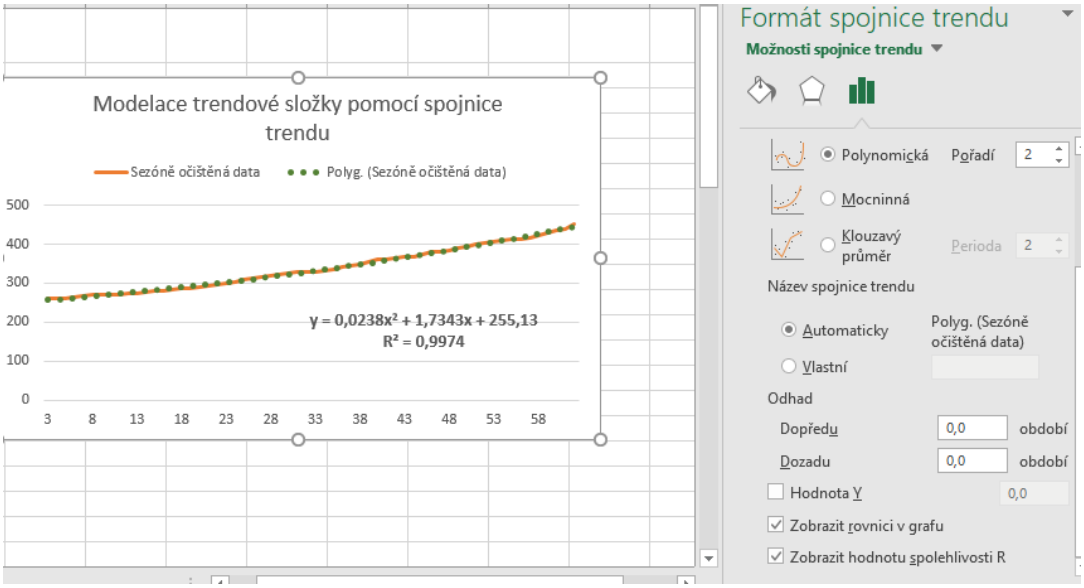
Obrázek 38: Porovnání jednotlivých trendů



Zdroj: Vlastní zpracování v Excelu

V Excelu po vytvoření grafu z hodnot, která chceme modelovat je možné přidáním spojnice trendu automaticky modelovat trendovou složku. Při zobrazení rovnice trendu a koeficientu determinace pojmenovaném jako Hodnota spolehlivosti R, ihned vidíme parametry jednotlivých nastavení a velmi rychle se dostaneme k uspokojivému modelu.

Obrázek 39: Přidání spojnice trendu do grafu



Zdroj: Vlastní zpracování v Excelu

6.1.2. Modelace sezonní složky

Pro modelaci sezonní složky nejprve vytvořit novou proměnou, kterou dostaneme jako rozdíl mezi původní hodnotou a centrovaným průměrem. Z těchto hodnot vytvoříme průměry pro jednotlivé kvartály. Poté vytvoříme jeden hromadný průměr pro průměrné hodnoty jednotlivých kvartálů. Nakonec odečteme od průměrné hodnoty jednotlivých kvartálů celkový průměr a to je odhad sezonní složky. Na obrázku vidíme, výsledky které jsme dopočítaly, a jsou jiné z důvodu, že program R dělá pouze průměr jednotlivých period a dále je neprůměruje a tuto hodnotu dále neodčítá. Tedy končí ve sloupci C na obrázku 40.

Obrázek 40: Modelace sezonní složky

A	B	C	D	E
Kvartál	**Očištěná hodnota**	**Průměr jednotlivých kvartálů**	**Celkový průměr**	**Sezonní složka**
Q1		7,916666667	14,62396	-6,70729
Q2		-42,23333333		-56,8573
Q3	-25,375	3,703125		-10,9208
Q4	52,75	89,109375		74,48542

Zdroj: Vlastní zpracování v Excelu

6.1.3. Výpočet reziduální složky

Jako poslední krok vypočítáme reziduální složku, která se z výše zmíněného důvodu o jiném pohledu na výpočet sezonnosti nebude rovnat programu R. Od původní hodnoty odečteme odhad kvadratického trendu a sezonnost, zbytek považujeme za reziduální složku.

6.1.4. Predikce budoucích hodnot

Jako poslední krok provedeme predikci následujících 6 období. K predikci využijeme kvadratický trend, tedy

$$y_t = 251{,}75 + 1{,}639 * t + 0{,}023 * t^2 \quad (6.1)$$

a příslušnou hodnotu sezonní složky.

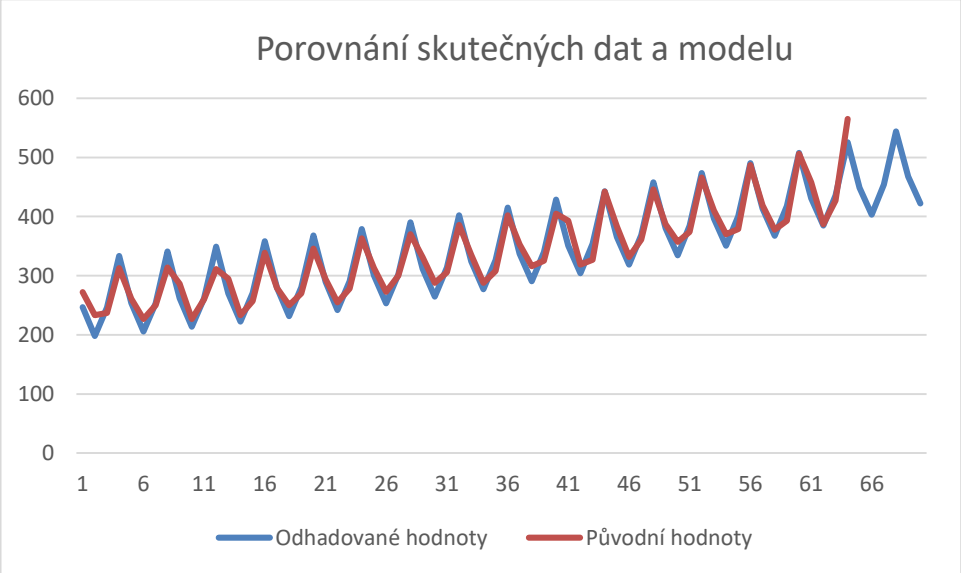
Obrázek 41:Predikce budoucích hodnot

	A	B	C	D	E	F
1	*Kvartál*	*Pozorovná t*	*t^2*	*Odhad kvadr. Trendu*	*odhad sezonní složky*	*predikce*
66	Q1	65	4225	455,46	-6,707291667	448,7527
67	Q2	66	4356	460,112	-56,85729167	403,2547
68	Q3	67	4489	464,81	-10,92083333	453,8892
69	Q4	68	4624	469,554	74,48541667	544,0394
70	Q1	69	4761	474,344	-6,707291667	467,6367
71	Q2	70	4900	479,18	-56,85729167	422,3227

Zdroj: Vlastní zpracování v Excelu

Na obrázku 41 je vidět vypočítané hodnoty následující predikce. Na Obrázku 42 níže je vidět graf odhadovaných hodnot a predikce následujících hodnot

Obrázek 42: Porovnání skutečných hodnot a modelu

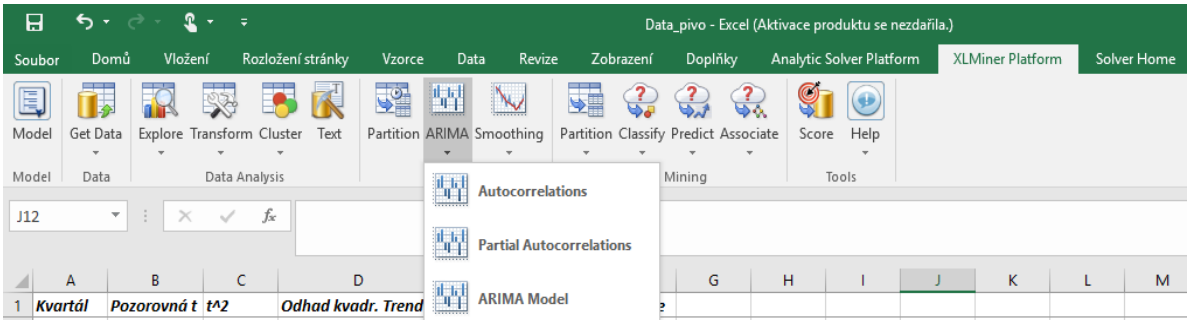


Zdroj: Vlastní zpracování v Excelu

6.2.Box – Jenkinsonova metoda v Excelu

V této kapitole si rozebereme modelaci pomocí Box- Jenkinsonovi metodologie v programu Excel. Pro použití této metody je nutné nainstalovat placený doplněk od firmy Frontline Solvers s názvem XLMiner Platform. Tento doplněk není volně přístupný, ale dá se na 15 dní vyzkoušet ve volné verzi, jinak pokud nevyužijeme této možnosti se jeho cena pohybuje od 250 do 5 000 amerických dolarů, podle toho, jaké funkce chceme mít přístupné. Na obrázku 43 je vidět, jak poté vypadá dodatečná lišta, z níž nám bude stačit funkce ARIMA, při jejímž rozkliknuti vidíme možnosti tvorby grafů autokorelačních funkce tak i parciální včetně možnosti tvorby modelu.

Obrázek 43: XLMiner Platform

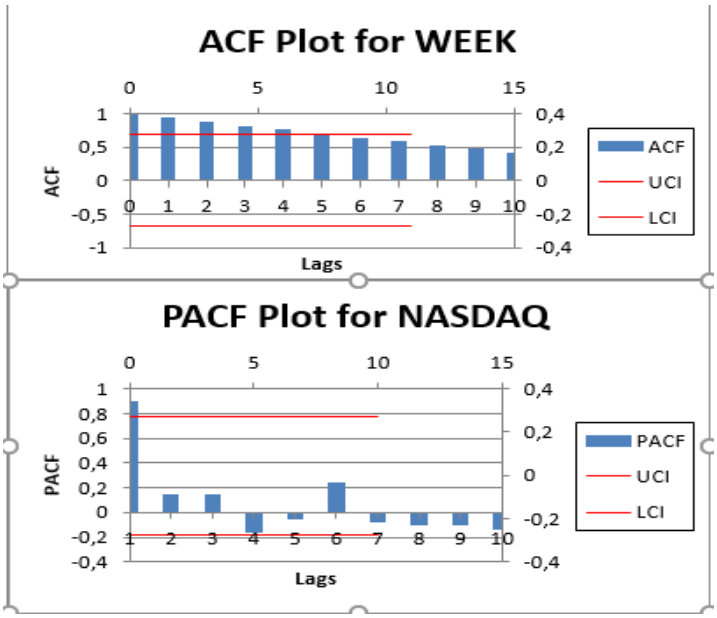


Zdroj: Vlastní zpracování v Excelu

6.2.1. Výstavba modelu ARIMA

Pomocí Box – Jenkinona budeme modelovat stejná data jako v R, tedy týdenní index NASDAQ. Nejprve pro ověření stacionarity dat vytvoříme korelogram ACF a PACF.

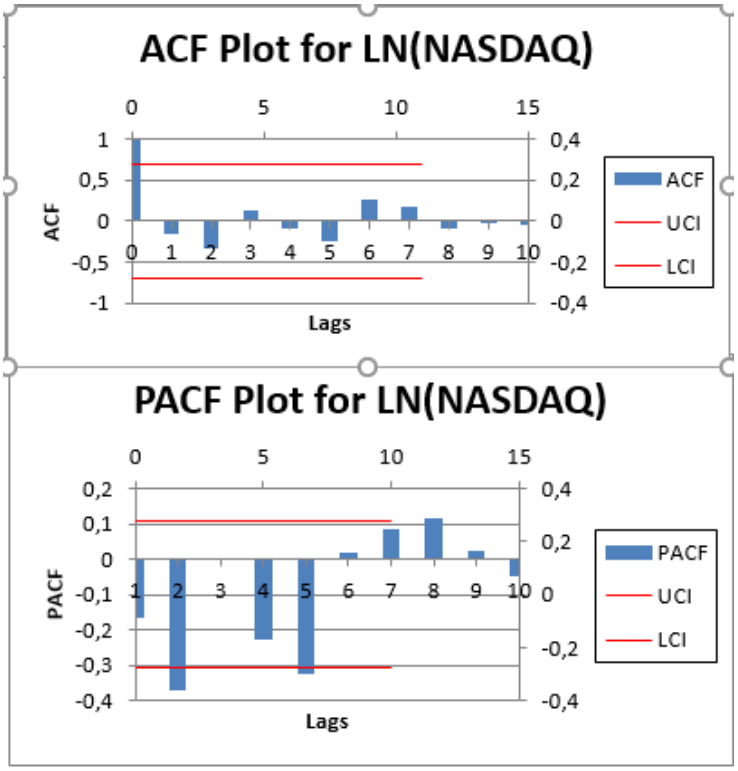
Obrázek 44: ACF a PACF v XLMiner Platform



Zdroj: Vlastní zpracování v Excelu n základě dat [19]

Z obrázku 44 je vidět, že PACF je velmi blízká k hodnotě jedna a graf ACF má sestupný lineární trend, takže řadu budeme muset pomocí stacionarizovat pomocí první diference.

Obrázek 45: ACF a PACF LN(NASDAQ)



Zdroj: Vlastní zpracování v Excelu na základě dat [19]

Na obrázku 45 vidíme, že řada je již stacionární a budeme modelovat model ARIMA (2,0,1). Poté na reziduích vytvoříme ACF a PACF, ze kterého vidíme, že všechny jsou uvnitř mezi spolehlivosti. Můžeme model prohlásit za uspokojivý a na jeho základě udělat odhad následujících 10 hodnot.

Obrázek 46: Predikce budoucích hodnot

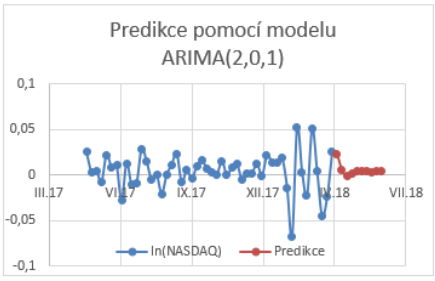
ARIMA Model

ARIMA	Coeff	StErr	p-value
Const. tern	0,003431404	0,002548022	0,178079064
AR1	0,373700285	0,00124951	0
AR2	-0,262317009	0,00316739	0
MA1	-0,727392438	0,021538461	5,1192E-250

Mean	0,003861512
-2LogL	-259,513238
Res. StdDev	0,018178234
#Iterations	200

Forecast

WEEK	Forecast
Forecast 1	0,023733661
Forecast 2	0,00563841
Forecast 3	-0,000687264
Forecast 4	0,001695522
Forecast 5	0,004245302
Forecast 6	0,00457311
Forecast 7	0,004026761



Zdroj: Vlastní zpracování v Excelu na základě dat [19]

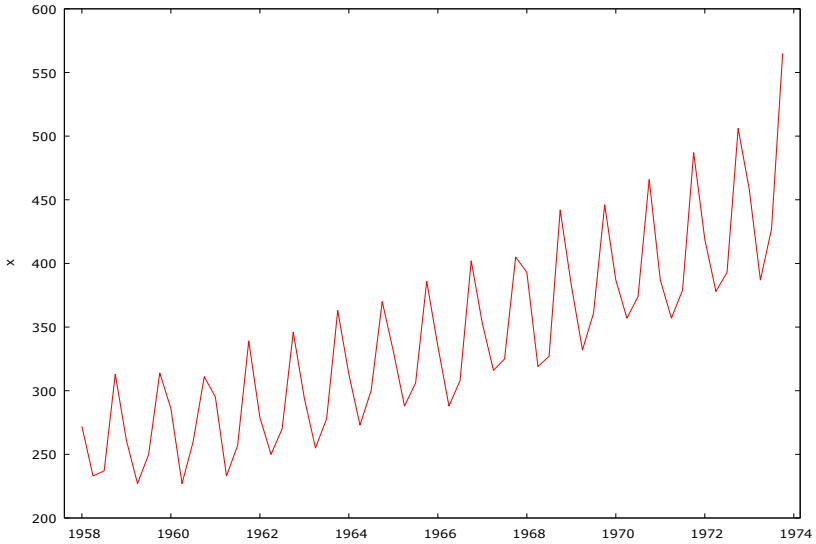
7. ČASOVÉ ŘADY V PROGRAMU GRETL

Název programu Gretl je zkratkou *Gnu,Regression, Econometrics a Time-series Library.* Jedná se o software, který obsahuje jednoduše aplikovatelné softwarové nástroje pro ekonomickou analýzu. Popularitu získal díky své volné dostupnosti, tedy že si program může každý bezplatně stáhnout z webových stránek http://gretl.sourceforge.net/. [8]

7.1. Dekompoziční metoda v Gretl

Nejprve musíme nahrát data australského piva a nastavit data jako časovou čtvrtletní časovou řadu a vykreslíme graf hodnot.

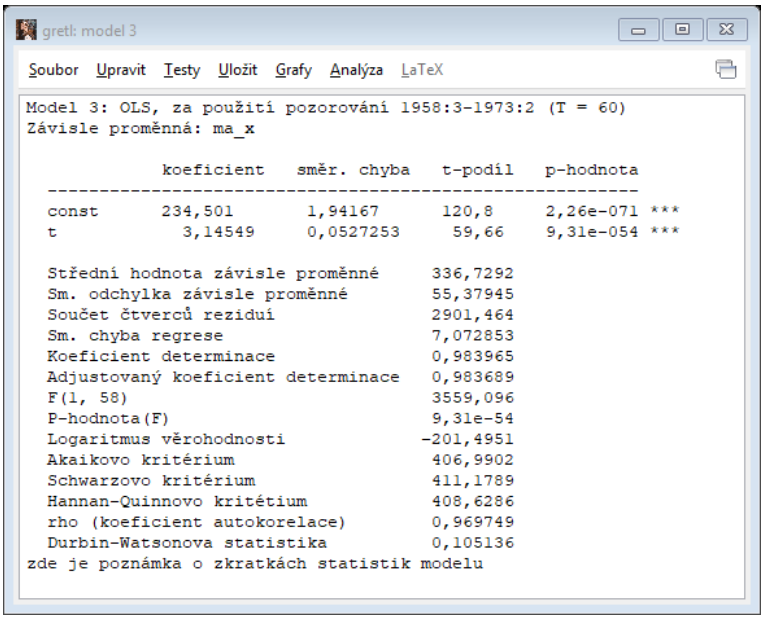
Obrázek 47: Graf časové řady výroby piv v Gretl



Zdroj: Vlastní zpracování v Gretl

Pro sezonní očištění dat, v horní části menu v záložce *„Filtr“* a dále *„Jednoduchý klouzavý průměr“*, v němž nastavíme centrovaný klouzavý průměr délky 4. Také zaškrtneme vytvořit novou proměnnou vyhlazená řada a také uložit do paměti cyklickou komponentu, která je původní hodnota – sezonně očištěná. Na vyhlazených datech centrovaným průměrem uděláme odhad parametrů lineárního trendu. V hlavní nabídce v záložce *„Model“* vybereme *„Ordinary least square“*(Většina funkcí a popisků je v gretlu v češtině ale občas se stane, že ne vše je přeloženo). Na obrázku 48 je vidět výstup, kde vidíme hodnoty absolutního i lineárního členu včetně p-hodnoty a směrodatné odchylky. Dále další statistiky kvality odhadu jako například součet čtverců reziduí nebo koeficient determinace.

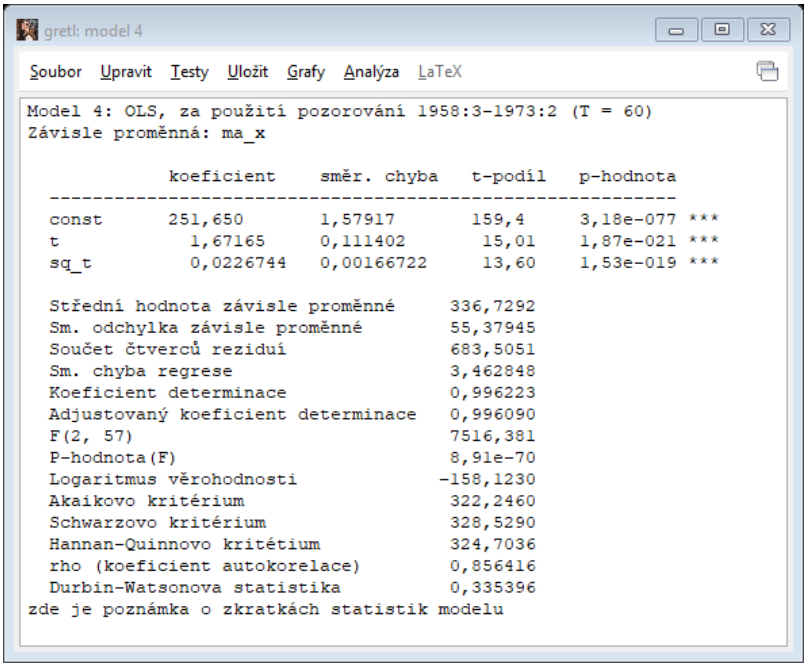
Obrázek 48: Odhad trendu v programu Gretl



Zdroj: Vlastní zpracování v Gretl

Následně vytvoříme novou proměnnou kvadratickou proměnnou a stejnou metodou vytvoříme odhad také pro kvadratický trend. A uložíme novou proměnnou, která bude reprezentovat kvadratický trend.

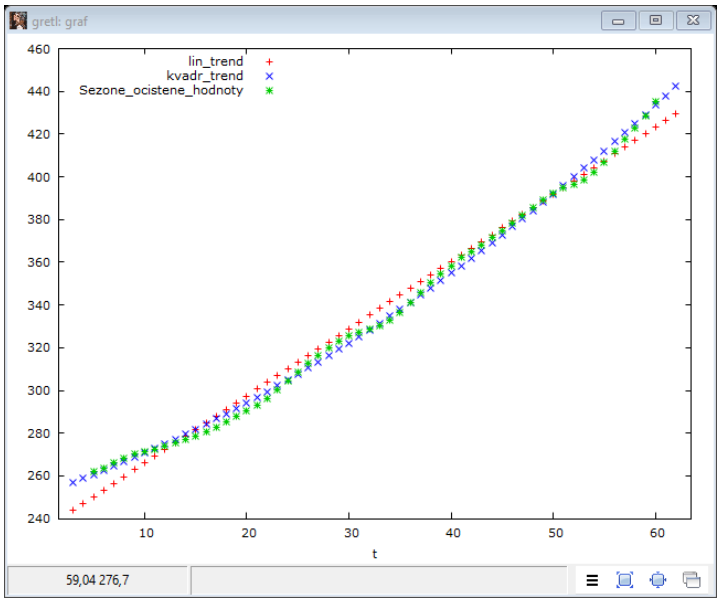
Obrázek 49: Odhad kvadratického trendu v Gretl



Zdroj: Vlastní zpracování v Gretl

Na obrázku 50 je vidět porovnání sezóně očištěných hodnot a modelace lineárního trendu a kvadratického trendu.

Obrázek 50: Porovnání trendových funkcí



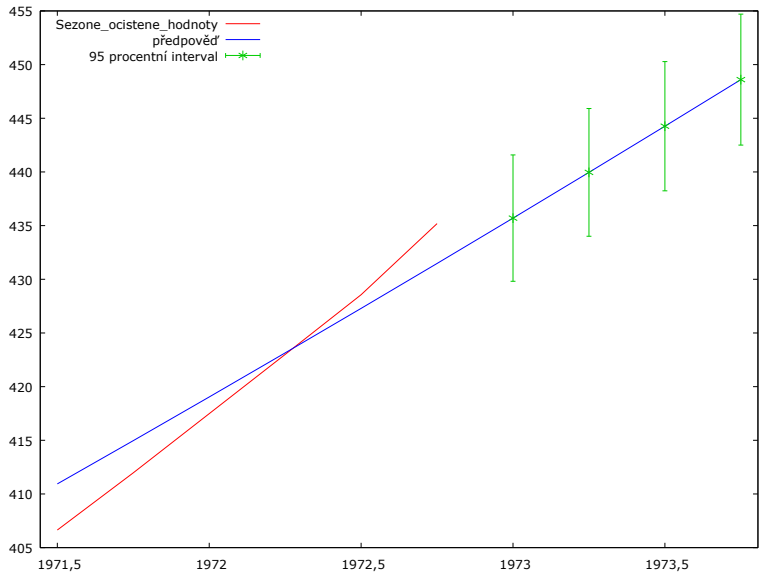
Zdroj: Vlastní zpracování v Gretl

Nevýhodou programu Gretl je nemožnost modelovat sezonní část časové řady. Uživatel má pouze možnost sezonně data očistit, nicméně další práci je nutné využít jiný software.

7.1.1. Predikce budoucích hodnot

Pro predikci budoucích hodnot využijeme přesnější odhad trendové složky tedy kvadratický trend. Program automaticky vykresluje do grafu bodovou i intervalovou předpověď budoucích pozorování.

Obrázek 51: Predikce budoucích hodnot v Gretl

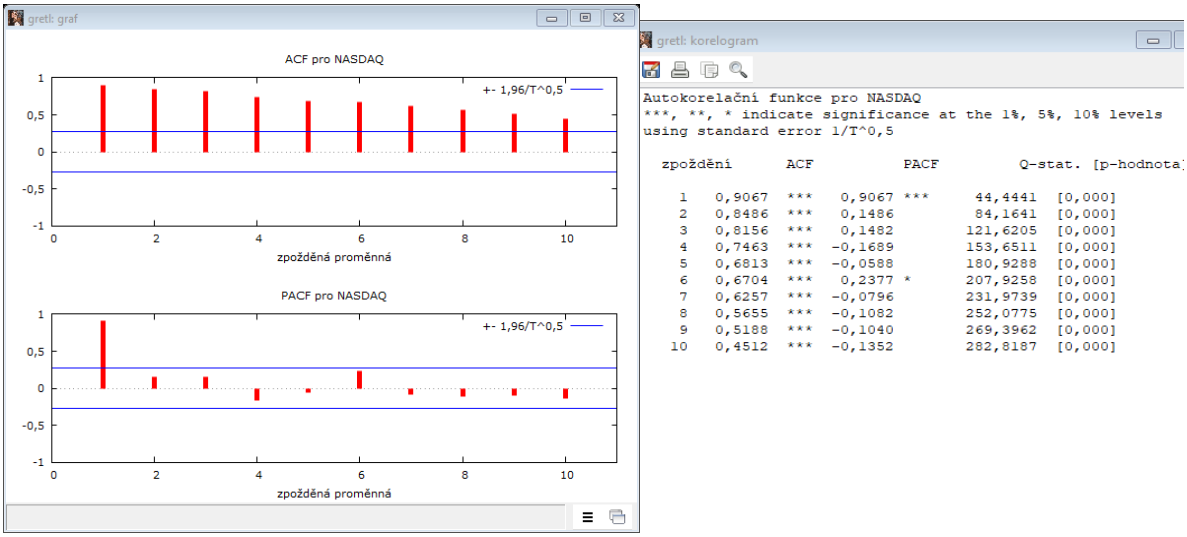


Zdroj: Vlastní zpracování v Gretl

7.2. Box – Jenkinsonova metoda v programu Gretl

Modelace dat pomocí B-J metody je velmi uživatelsky přívětivé a hlavně jednoduché a rychlé. Po nahrání dat NASDAQ Indexu opět označíme data jako týdenní časovou řadu a zlogaritmujeme. Pro ověření stacionarity můžeme ihned použít ACF a PACF, které nalezneme v záložce *„Proměnná“* a dále *„Korelogram“*, který automaticky vykresluje obě funkce. Zároveň spolu s grafy zobrazí také tabulku jednotlivých hodnot.

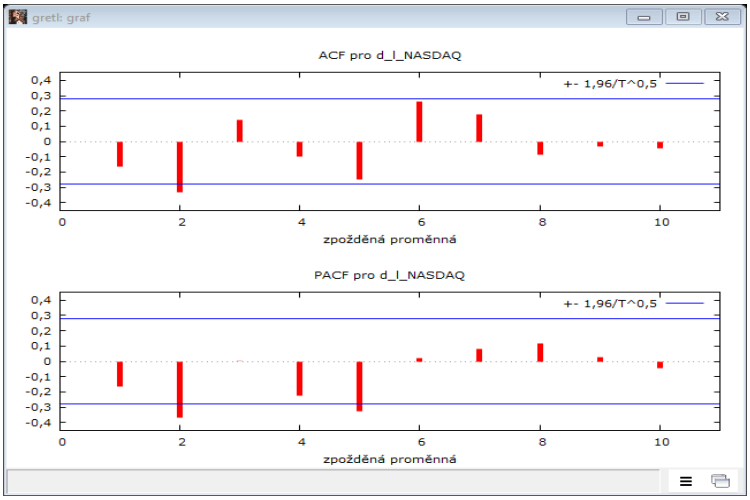
Obrázek 52: Výstup funkce Korelogramy Gretl



Zdroj: Vlastní zpracování v Gretlu na základě dat [19]

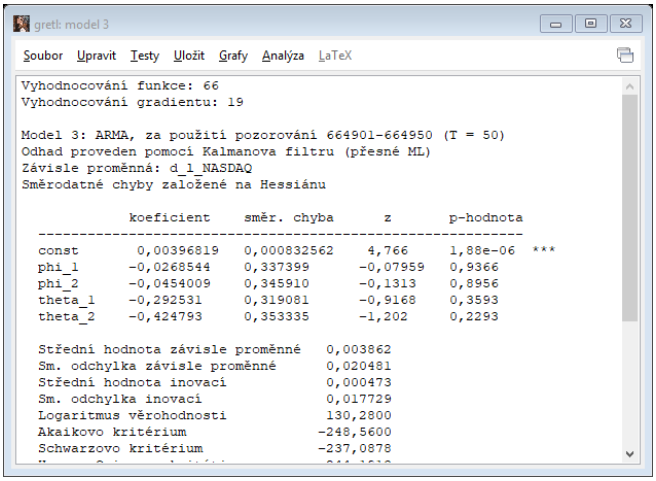
Z Obrázku 52 vidíme, že ACF i PACF prokazují nestacionaritu a bude nutné řadu stacionarizovat pomocí první diference. A opět pomocí vyvoláme ACF a PACF.

Obrázek 53: ACF a PACF modelu ARIMA(2,0,2)



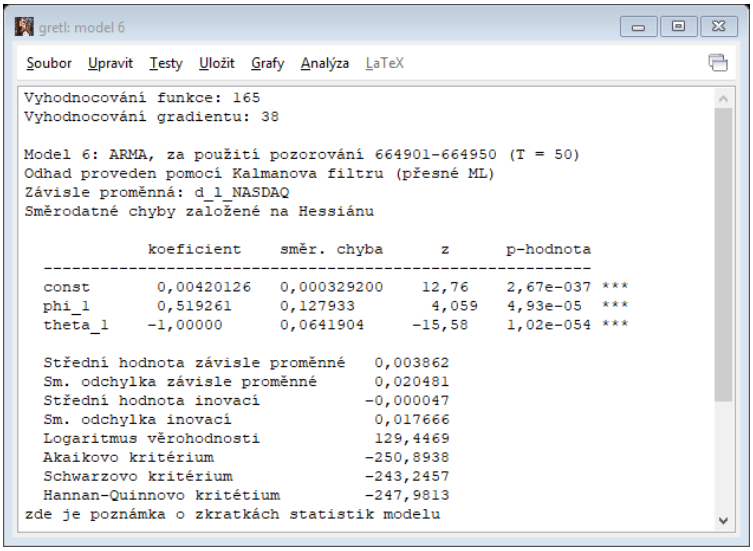
Z obrázku 53 vidíme významné překročení v obou případech v bodě dva, budeme modelovat ARIMA(2,0,2).

Obrázek 54: Odhad modelu ARIMA (2,0,2)



Na obrázku 54 je vidět výsledek modelu ARIMA (2,0,2), ale pouze konstantu považujeme za statisticky významnou. Proto zkusíme model ARIMA (1,0,1), jelikož obě části měli oba koeficienty statisticky nevýznamné.

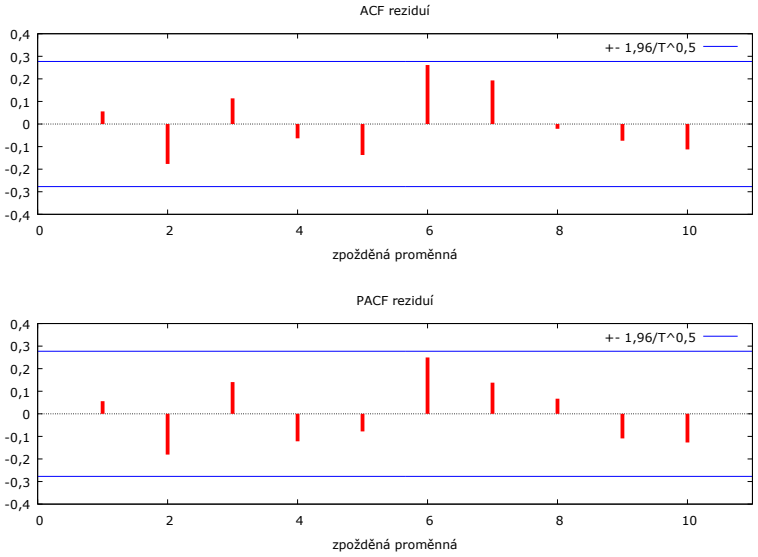
Obrázek 55: Odhad modelu ARIMA(1,0,1)



Zdroj: Vlastní zpracování v Gretlu na základě dat [19]

Z obrázku 55 vidíme, že konstanta i oba parametry modelu jsou již statisticky významné. Také korelogram reziduí na obrázku 56 potvrzuje správnost našeho modelu.

Obrázek 56: ACF a PACF modelu ARIMA(1,0,1)

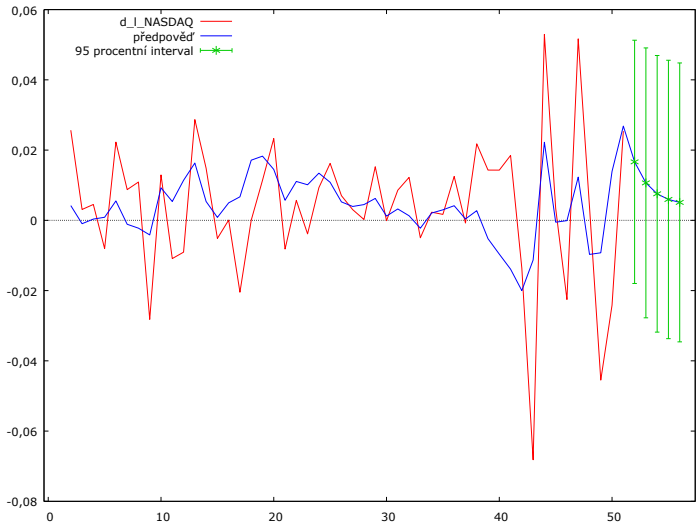


Zdroj: Vlastní zpracování v Gretlu na základě dat [19]

7.2.1. Predikce budoucích hodnot

Podobně jednoduše jako probíhalo modelování ARIMA modelu, je uživatelsky velmi přívětivé modelace budoucích hodnot. Po vytvoření modelu, který uspokojivě modeluje data, ve vyvolaném okně s výsledky, kde jsme například testovali rezidua, je záložka *„Analýza“* a v ní záložka „*Předpovědi“*, kde pouze nastavíme požadovaný počet předpovědí a program již sám zobrazí graf s bodovou i intervalovou předpovědí.

Obrázek 57: Predikce budoucích hodnot na základě modelu ARIMA (1,0,1)



Zdroj: Vlastní zpracování v Gretlu na základě dat [19]

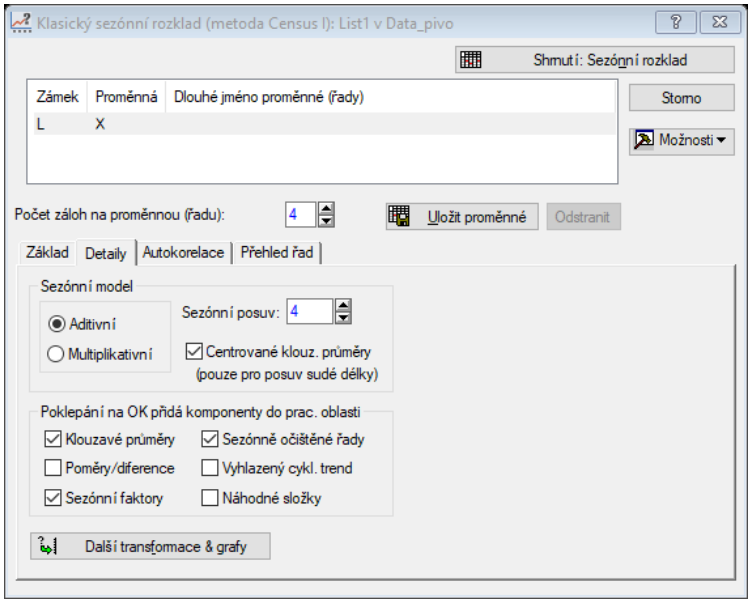
8. MODELACE ČASOVÝCH ŘAD V PROGRAMU STATISTICA

Statistica je pokročilý analytický softwarový balík původně vyvinutý společností StatSoft, kterou v roce 2014 koupila společnost Dell. Software Statistica poskytuje nástroje pro analýzu dat, správu dat, statistické a data miningové metody a vizualizaci dat. Pro zlepšování jednotlivých funkcí je velmi využívána zpětná vazba od uživatelů, která pomáhá neustálému zlepšování software. Uživatelé mohou zdarma vyzkoušet na měsíc využití zdarma, po uplynutí této doby je nutné koupit licenci.

8.1. Dekompoziční metoda v programu Statistica

Import dat do programu Statistica lze provést pomocí záložky „*Soubor*“ a poté „*Otevřít*“, kde zvolíme odpovídající datové umístění a načteme data. Druhý doporučený pro menší množství dat je označit ve zdrojovém souboru data a zvolit „*Kopírovat*“ a následně v programu Statistica zvolíme vložit, popřípadě využijeme klávesové zkratky Ctrl + C a Ctrl + V. Nejprve vytvoříme novou sezonně očištěnou proměnnou pomocí centrovaného klouzavého průměru délky 4. Na kartě *Statistiky* vybereme *Pokročilé modely* a v nich *Časové řady a predikce.* Nejprve je nutné vybrat proměnnou následně ve vyvolaném okně programu zvolit *Sezonní rozklad.* Na obrázku 58 je vidět nastavení, kde Sezonní model volíme jako aditivní, chceme vytvořit klouzavé průměry, program nabízí rovnou volbu vytvoření sezóně očištěné řady a také výpočet sezonních faktorů.

Obrázek 58: Dekompozice v STATISTICA



Zdroj: Vlastní zpracování v STATISTICA

Pro výpočet lineárního trendu a kvadratického použijeme funkci vícenásobná regrese. Odhady jednotlivých parametrů včetně koeficientu determinace vidíme na obrázku 59 níže.

Obrázek 59: Výstup z regrese

Výsledky regrese se závislou proměnnou : Centr. průměr (List1 v Data_pivo)
R= ,99195012 R2= ,98396504 Upravené R2= ,98368857
F(1,58)=3559,1 p<0,0000 Směrod. chyba odhadu : 7,0729

N=60	b*	Sm.chyba z b*	b	Sm.chyba z b	t(58)	p-hodn.
Abs.člen			254,5007	1,941669	120,7727	0,00
t	0,991950	0,016627	3,1455	0,052725	59,6582	0,00

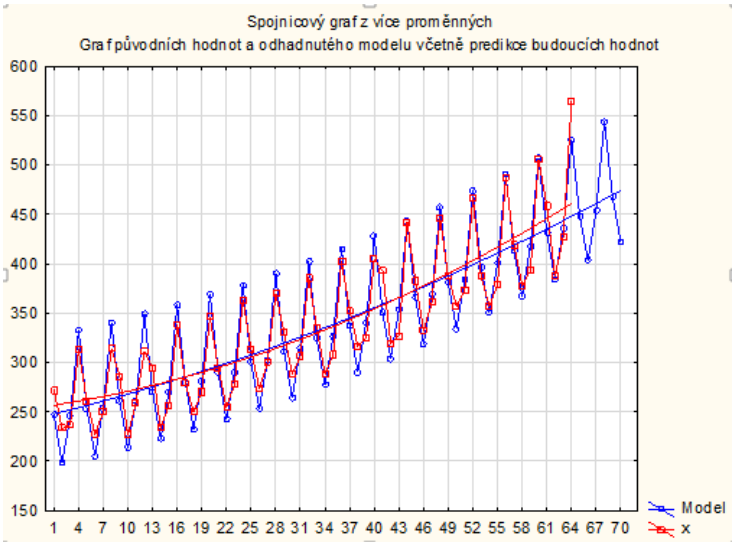
Výsledky regrese se závislou proměnnou : Centr. průměr (List1 v Data_pivo)
R= ,99810952 R2= ,99622260 Upravené R2= ,99609006
F(2,57)=7516,4 p<0,0000 Směrod. chyba odhadu : 3,4628

N=60	b*	Sm.chyba z b*	b	Sm.chyba z b	t(57)	p-hodn.
Abs.člen			251,6501	1,579166	159,3563	0,000000
t	0,527166	0,035131	1,6717	0,111402	15,0057	0,000000
Kvadr_člen	0,477788	0,035131	0,0227	0,001667	13,6001	0,000000

Zdroj: Vlastní zpracování v STATISTICA

Na základě obou modelů můžeme vytvořit predikci budoucích hodnot, bohužel program nenabízí automatickou predikci před dekompozicí, jako například program R. Uživatel nejprve vytvoří predikci vybrané trendové funkce a až poté vytvoří celkovou predikci. Na obrázku 60 níže je vidět graf původních hodnot a vymodelovaných hodnot včetně predikce následujících období.

Obrázek 60: Predikce budoucích hodnot v STATISTICA

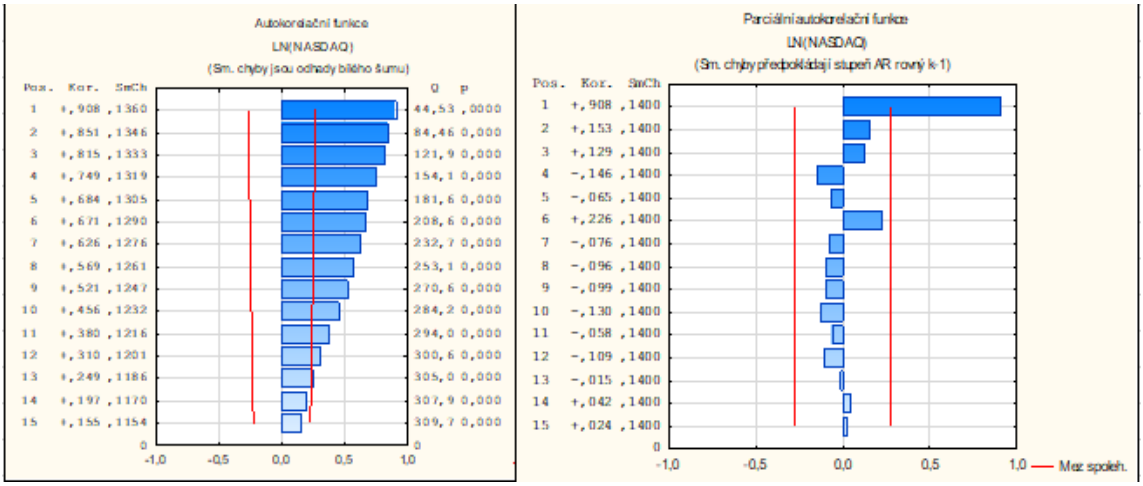


Zdroj: Vlastní zpracování v STATISTICA

8.2.Box – Jenkinsonova metoda v programu Statistica

Pro odhad modelu ARIMA využijeme opět záložku „*Statistica*“ a „*Analýza časových řad*“, kde zvolíme *ARIMA & autokorelační funkce.* Nejprve je nutné ověřit stacionaritu pomocí ACF a PACF, grafy obou funkcí jsou zobrazeny na obrázku 61 níže, z něhož je vidět že bude řadu nutné stacionarizovat pomocí diference.

Obrázek 61: ACF a PACF NASDAQ



Zdroj: Vlastní zpracování v STATISTICA na základě dat [19]

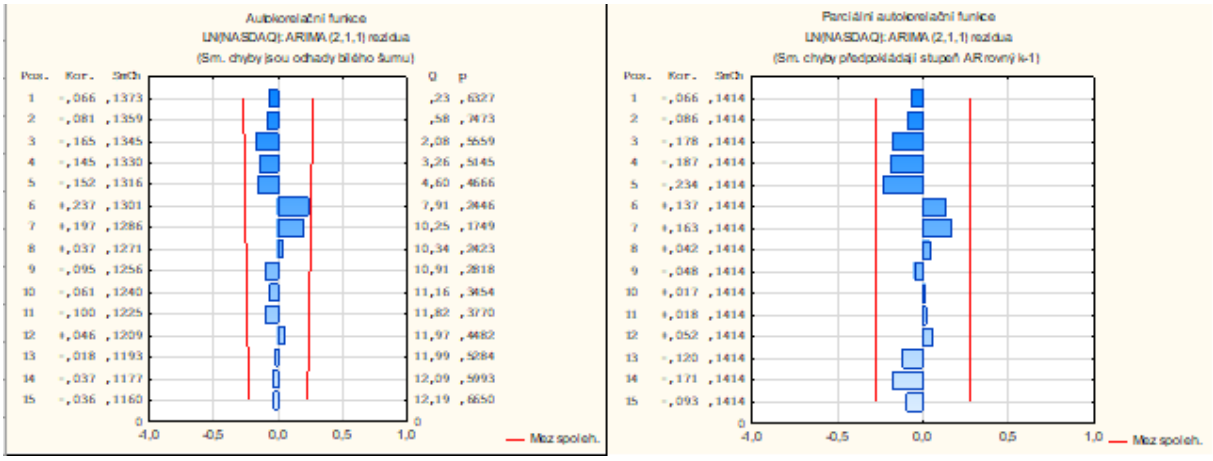
Po stacionarizaci řady provedeme odhad modelu ARIMA (1,0,1) a následně provedeme ACF a PACF na reziduích tohoto modelu, z nichž můžeme prohlásit odhad modelu jako uspokojivý.

Obrázek 62: Odhad paramtrů modelu ARIMA (1,0,1)

Paramet.	Vstup: LN(NASDAQ): D(-1) (NASDAQ) Transformace: D(1) Model:(1,0,1) PČ Rezid. = ,00052 Param.	Asympt. SmCh	p
Konstant	0,004201	0,000329	0,000000
p(1)	0,519261	0,127933	0,000005
q(1)	-0,999999	0,064190	0,000200

Zdroj: Vlastní zpracování v STATISTICA na základě dat [19]

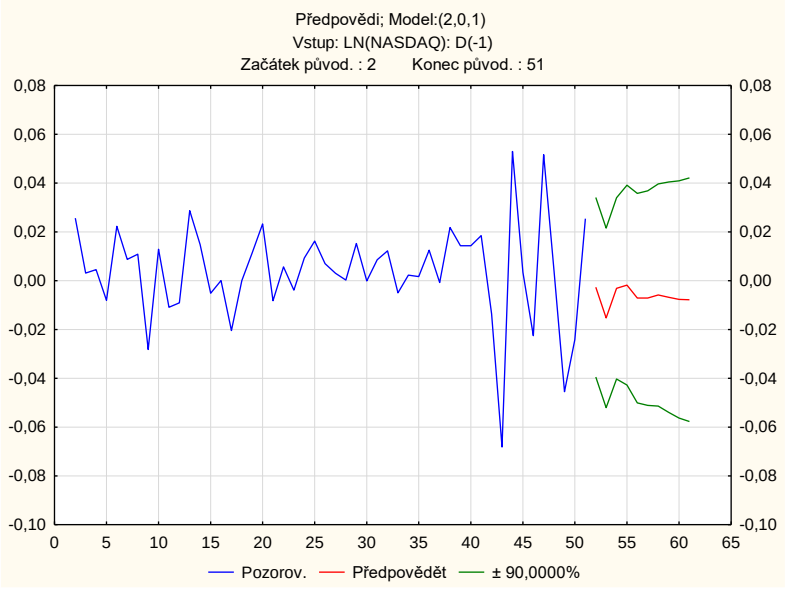
Obrázek 63: ACF a PACF modelu ARIMA (1,0,1)



Zdroj: Vlastní zpracování v STATISTICA na základě dat [19]

Pro predikci následujících období ve výsledném okně s výsledky v záložce *„Detaily“* funkce s názvem *„Předpovědi“*, ve které pouze zvolíme požadovaný počet odhadovaných budoucích hodnot. Na obrázku 63 je graf predikce budoucích hodnot bodově i intervalově.

Obrázek 64: Predikce budoucích hodnot na základě modelu ARIMA (1,0,1)



Zdroj: Vlastní zpracování v STATISTICA na základě dat [19]

ZÁVĚR

Teoretická část práce je zaměřena na analýzu časových řad, jsou uvedeny základní definice a vlastnosti, které jsou v dalších kapitolách použity na skutečných datech, a je předpokládáno, že čtenář porozuměl problematice časových řad, která je využívána v dalších kapitolách.

Hlavní částí této diplomové práce je analýza dat v jazyce R. Nejprve bylo jsme byli seznámeni se základní rozhraní programu, včetně funkce jednotlivých oken a instalace dodatečných knihoven s funkcemi a daty. Tato možnost patří k velké výhodě programu R, kdy základní verze obsahuje pouze základní rozhraní a program tedy není velký a nevyužívá tolik vnitřní paměti počítače. Na druhou stranu je nutné, potřeba instalace jedné knihovny se může protáhnout na pár minut až hodin v momentě, kdy instalace knihovny je podmíněna předchozí instalací jiné knihovny. Jelikož se knihovny stahují pouze z webového portálu https://cran.r-project.org/, v případě problému tohoto serveru je uživatel nucen čekat na jeho zprovoznění.

Pro uživatele bez znalosti SQL nebo VBA jazyka je první setkání s tímto jazykem obtížné. Naštěstí v programu samotném je zabudována funkce „*Help*“, která ve většině případů dokáže uživateli dostatečně poradit. V případě, že i přes použití této funkce uživatel není spokojen s informacemi, může využít spoustu internetových materiálů, které jsou zejména v anglickém jazyce, což v dnešní době internetových překladačů nemusí být problémem. Na těchto fórech si navzájem uživatelé sdělují své návody a tipy a jsou ochotni poradit si navzájem. Při této práci jsem narazil na problém při tvorbě ACF, kdy první hodnota byla permanentně rovna jedné. Napsal jsem tento problém na fórum support.rstudio.com a za pár minut mi jiný uživatel poradil, že problém je ve špatné funkci. Já používal ACF funkci „*acf*()“, což bylo špatně a je nutné využívat funkci se stejným jménem, ale s první písmenem velkým „*Acf*()“.

Pro jednoduché tvorby grafů nebo základní matematické operace je používání jazyka R komplikované a zbytečné zdlouhavé. Například při porovnání s Microsoft Excel uživatel pouze nahraje data, která většinou má v tomto programu, a z nich během pár vteřin vytvoří odpovídající graf. V jazyce R tato tvorba není takto triviální, pro tvorbu složitějších grafů jazyk R nabízí velké množství nastavení, které ocení zejména zkušení analytici. Při grafickém zpracování je možné přidávat jednotlivé další řady, kdy každá řada může mít specifické vlastnosti a vlastní grafické provedení odlišné od ostatních. Podobně tvorba popisů a legend grafu je komplikovanější, je ale možné upravovat od umístění, popisků nebo dokonce rozdělit legendu na několik částí a každou z nich umístit na jiné místo v grafu. Jednotlivé možnosti High i Low grafiky jsou vysvětleny na skutečných datech návštěvnosti HC Dynama Pardubice.

Jazyk R pracuje zejména se složitějšími datovými strukturami jako je vektor, datová tabulka nebo seznam. Tato jednotlivé struktury byly popsány včetně příkladů, jak s nimi pracovat a upravovat je. Každá datová struktura je vhodná pro jiný typ dat a následnou další práci s nimi.

Dekompoziční metoda v R jazyce byla vysvětlena na datech o výrobě piva v Austrálii, která se nachází v databázi programu, a každý čtenář této práce si na nich může vyzkoušet výše popsané postupy a zkontrolovat výsledné hodnoty. Dekompoziční metoda byla nejprve vysvětlena postupně po jednotlivých krocích. Následně byla ukázána jedna z předností - jazyk R, využití funkce „*decompose()*“, do které stačí nahrát pouze data a zadat periodu. Poté funkce automaticky provede dekompozici na trendovou, sezonní a reziduální složku. Toto může být výhoda zejména při nutnosti dekompozice většího množství časových řad.

Box-Jenkinsonova metoda byla představena na reálných ročník datech indexu NASDAQ. Na začátek této metody byla použita postupná tvorba modelu ARIMA. Nejprve byl ověřen předpoklad stacionarity, který nebyl splněn, a proto data byla stacionarizována pomocí první diference. Následně krok po kroku byla provedena výstavba základního modelu včetně ověření statistické významnosti jednotlivých parametrů. Nakonec byla provedena predikce budoucích hodnot. Podobně jako při dekompozici je i zde možné využít již předdefinované funkce „*auto.arima*()“, do které stačí zadat pouze data, na kterých chceme provést výstavbu modelu. Funkce sama podle Aikaike information criterion vyhodnotí nejlepší model a ten zobrazí. Je možné hledání výsledného modelu omezit maximálními hodnotami AR a MA, nebo nastavit, zda chceme modelovat s konstantou, popřípadě zda mají být použity sezonní hodnoty AR a MA.

Jako další program byl využit Excel, ve kterém naopak jednoduché matematické úkony a tvorba grafů trvá pár vteřin. Pomocí něho byly provedeny na stejných datech také modelace časové řady včetně predikce. Obecně je možné říct, že dekompoziční metoda je vcelku rychlá, kdy každý analytik ocení zejména modelace trendu pomocí spojnice trendu. Zde jedním kliknutím myši vidíme odhad rovnice trendu a jeho index determinace. V případě nelineárních modelů je ovšem nutné využít funkci lineární regrese. Výhoda tohoto programu je především jeho dostupnost, jelikož je součástí balíčku MS Office, který je součástí většiny počítačů s operačním systémem Microsoft. Většina uživatelů má alespoň nějaké zkušenosti s tímto softwarem, proto pro ně není komplikací případně nějak upravit data a dále s nimi pracovat. Nevýhodou může být – nemožnost pracovat s modely ARIMA v základní verzi. V této práci je popsáno použití dodatečného doplňku XLMiner, který je na 15 dní zdarma. Tento dodatečný nástroj ovšem velmi brzdí spouštění programu i práci s daty v momentě, kdy používáme pouze základní funkce a grafy. Jako alternativa jde také používat, při tvorbě modelu ARIMA, ovšem

jeho možnosti v porovnání s ostatními programy jsou velmi základní a pro náročnější uživatele mohou být nedostatečné. Zejména B-J je v porovnání s programem R dost komplikovaná a zdlouhavá.

Dalším použitým softwarem je software Gretl, který patří též do skupiny freeware programů. Nevýhodou je, že velmi často přestane odpovídat a ukončí se. Neuložené postupy pak potom není možné nijak obnovit a uživatel o vše, co nebylo uloženo, přijde. Dekompoziční metoda je v porovnání s ostatními softwary komplikovanější a nenabízí tolik možností jako ostatní programy. Naopak modelace pomocí B-J metody v tomto programu se považuje za uživatelsky nejjednodušší a nejrychlejší. Vše je velmi rychlé a přehledné. Sice neobsahuje žádnou automatickou funkci, která by sama vystavěla model, ale pomocí zákonitostí a zkušenosti může analytik velmi rychle nalézt uspokojivý model, na němž následně provede predikci budoucích hodnot.

Posledním použitým programem byl software Statistica. Tento software není volně dostupný a to může odradit některé uživatelé, kteří nevyžadují tak odborný software. Program nabízí velmi příjemné a přehledné uživatelské prostředí. Také ve verzích vydaných po roce 2014 je již možné propojovat jazyk R s tímto programem. Program obsahuje velké množství funkcí, což občas může být kontraproduktivní pro nové uživatele, kterým bude trvat nějakou dobu, než naleznou, co hledají a nastaví všechny parametry. Dekompozice v programu Statistica také probíhá bez větších manuálních operací uživatele, pouze se nastavují jednotlivé vstupní parametry. Tento program je jako jediný ze softwarů použitých v této práci placený a zaměřený přímo na statistické zpracování a nabízí větší množství analytických metod.

Při porovnávání výsledků z jednotlivých softwarů můžeme vidět stejné hodnoty při modelaci trendové složky, jelikož všechny výše uvedené programy používají pro odhad parametrů metodu nejmenších čtverců, dosahují stejných hodnot. Odchylky nalezneme při sezonním očišťování, kdy jazyk R pouze zprůměruje jednotlivé složky, které vzniknou sezonním očištěním dat, a stejné je to v programu Gretl. V softwaru statistika program dále pracuje s těmito očištěnými hodnotami. Tento rozdíl v pohledu na modelaci sezonní složky se poté také projevuje při celkovém odhadu budoucích hodnot. Stejných hodnot dosahujeme při modelace pomocí B-J metody. Při modelaci této metody Statistica a Gretl nabízí uživateli, možnost měnit metodu, na jejímž základě bude modelovat parametry modelu.

Závěrem této práce lze říci, že práce s jazykem R je zpočátku velmi náročná, ale postupem času s přibývajícími zkušenostmi se pro uživatele stává více přívětivá. Využití programu R při modelaci jakékoliv řady nikdy nebude chyba, díky tomu že velmi dobře zvládá obě metody. Obecně není možné rozhodnout, který software je nejlepší pro analýzu časových řad. Záleží na

datech, která chceme modelovat, kterou metodu chceme použít a v neposlední řadě, k jakému účelu analýzu provádíme.

Programovací jazyk R může být zejména dobrou alternativou v momentě, kdy nevíme, zda data bude možné uspokojivě modelovat pomocí dekompoziční metody. Využitím automatických funkcí jsme schopni velmi rychle získat odhad obou metod, a na tomto základě provádět další analýzu nebo porovnat jednotlivé modely navzájem. Pokud jednotlivé výstupy chceme prezentovat v grafické podobě, máme mnoho možností pro úpravu jednotlivých grafů a jejich popisků, ze kterých si vybere i náročnější uživatel.

POUŽITÁ LITERATURA

[1] ARLT, J.; ARLTOVÁ M.: Finanční časové řady: [vlastnosti, metody modelování, příklady a aplikace]. 1. vyd. Praha: Grada, 2003, 220 s. ISBN 80-247-0330-0

[2] ARLT, Josef a Markéta ARLTOVÁ. Ekonomické časové řady. Praha: Professional Publishing, 2009. ISBN 978-80-86946-85-6.

[3] ARLT, Josef, Markéta ARLTOVÁ a Eva RUBLÍKOVÁ. Analýza ekonomických časových řad s příklady. Praha: Vysoká škola ekonomická, Fakulta informatiky a statistiky, 2002. ISBN 80-245-0307-7.

[4] ARLT, Josef. Moderní metody modelování ekonomických časových řad. 1.vyd. Praha: Grada Publishing, 1999, 307 s. ISBN 80-716-9539-4

[5] Cipra, Tomáš: Analýza časových řad s aplikacemi v ekonomii, Alpha: Státní nakladatelství technické literatury, Praha, 1986]

[6] ČAPEK, Jan. Modelování ekonomických a sociálních procesů: pro kombinovanou formu studia. Vyd. 1. Pardubice: Univerzita Pardubice, 2006, 103 s. ISBN 80-719-4838-1

[7] Data mining with SPSS modeler: theory, exercises and solutions. ISBN 978-3-319-28707-2.

[8] GNU Regression, Econometrics and Time-series Library. [online]. 2018 [cit. 2018-04-23]. Dostupné z: http://gretl.sourceforge.net/

[9] HC DYNAMO Pardubice [online]. Pardubice: HOCKEY CLUB DYNAMO PARDUBICE, 2018 [cit. 2018-04-23]. Dostupné z: http://www.hcdynamo.cz/Index.asp

[10] LANDER, Jared P. R for everyone: advanced analytics and graphics. Upper Saddle River: Addison-Wesley, 2014. ISBN 978-0-321-88803-7

[11] MAREK, Luboš. Statistika v SPSS: časové řady. Praha: Vysoká škola ekonomická, 1995. ISBN 80-7079-642-1

[12] MILLS, C., T.: The Ekonometric Modelling of Financial Time Series. 3rd ed. New York: Cambridge University Press, 2008. 445 p. ISBN 978-0-521-71009-1

[13] NGAI HANG CHAN. Time Series Applications to Finance. Hoboken: John Wiley, 2002. ISBN 9780471461647.

[14] PARADIS, Emmanuel. R for Beginners. [online] 2002. 58 s. [cit. květen 2010]. Dostupné z WWW: http://www.karlin.mff.cuni.cz/~kulich/vyuka/Rdoc/r-intro.pdf]

[15] SCOTT, Theresa A. An Introduction to R. [online]. 2004. 52 s. [cit. květen 2010]. Dostupné z WWW: < http://www.karlin.mff.cuni.cz/~kulich/vyuka/Rdoc/RLectureTScott.pdf>

[16] STATSOFT. STATSOFT: STATISTICA Features Overview [online]. 2018 [cit. 2018-04-23]. Dostupné z: http://www.statsoft.com/Products/STATISTICA-Features

[17] The R Project for Statistical Computing. [online][cit. květen 2010]. Dostupné z WWW: https://www.r-project.org/

[18] Yahoo Finance - Business Finance, Stock Market, Quotes, News. Yahoo Finance - Business Finance, Stock Market, Quotes, News[online]. 2018 [cit. 2018-04-23]. Dostupné z: https://finance.yahoo.com/