Ekonomická fakulta
Faculty of Economics
Jihočeská univerzita v Českých Budějovicích
University of South Bohemia in České Budějovice

Jihočeská univerzita v Českých Budějovicích
Ekonomická fakulta
Katedra aplikované matematiky a informatiky

Bakalářská práce

Výpočet hodnoty v riziku v R

Vypracoval: František Žiška
Vedoucí práce: doc. RNDr. Tomáš Mrkvička, Ph.D
České Budějovice 2021

JIHOČESKÁ UNIVERZITA V ČESKÝCH BUDĚJOVICÍCH

Ekonomická fakulta

Akademický rok: 2019/2020

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**František ŽIŠKA**
Osobní číslo:	**E18321**
Studijní program:	**B6209 Systémové inženýrství a informatika**
Studijní obor:	**Ekonomická informatika**
Téma práce:	**Výpočet hodnoty v riziku v R**
Zadávající katedra:	**Katedra aplikované matematiky a informatiky**

Zásady pro vypracování

Cílem práce je vytvoření pravděpodobnostního modelu pro výpočet hodnoty v riziku a vytvoření R programu jenž je schopný hodnotu v riziku počítat.

Metodický postup:

1. Studium literatury – 1. semestr.
2. Vytvoření kódu – 2. semestr.
3. Zpracování výsledků – 3. semestr.
4. Závěr a doporučení.

Rozsah pracovní zprávy:	**40 – 50 stran**
Rozsah grafických prací:	
Forma zpracování bakalářské práce:	**tištěná**

Seznam doporučené literatury:

1. Alemany, R., Bolance, C. & Guillen, M. (2012). Nonparametric estimation of Value-at-Risk. *Working Papers XREAP2012-19, Xarxa de Referencia en Economia Aplicada (XREAP)*. Revised Oct 2012.
2. Cipra, T. (2015). *Praktický průvodce finanční a pojistnou matematikou.* Praha: Ekopress, s.r.o.
3. Zima, P. & Brown, L. R. (1996). *Schaum's Outline of Theory and Problems of Mathematics of Finance.* New York: McGraw-Hill Companies, Inc.
4. Další časopisecká a knižní literatura dle zaměření práce.

Vedoucí bakalářské práce: **doc. RNDr. Tomáš Mrkvička, Ph.D.**
Katedra aplikované matematiky a informatiky

Datum zadání bakalářské práce: **17. ledna 2020**
Termín odevzdání bakalářské práce: **16. dubna 2021**

JIHOČESKÁ UNIVERZITA
V ČESKÝCH BUDĚJOVICÍCH
EKONOMICKÁ FAKULTA
Studentská 13 (26)
370 05 České Budějovice

doc. Dr. Ing. Dagmar Škodová Parmová
děkanka

doc. RNDr. Tomáš Mrkvička, Ph.D.
vedoucí katedry

V Českých Budějovicích dne 4. března 2020

Prohlášení

Prohlašuji, že svou bakalářskou práci „Výpočet v riziku v R“ jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to - v nezkrácené podobě - elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

V Českých Budějovicích dne

_____

Jméno Příjmení

Poděkování

Chtěl bych poděkovat především svému vedoucímu práce doc. RNDr. Tomáši Mrkvičkovi, Ph.D za jeho čas, rady a trpělivost při konzultacích a při vypracování této bakalářské práce.

Obsah

1 Úvod

Investice a riziko s nimi spojené je již dlouho diskutované téma. Při současných krizích ve světě, ať už se jedná o politické, finanční či pandemické, riziko spojené s investicemi výrazně kolísá. A proto je výpočet risku investic v současné době obzvlášť důležitý. (Duda & Schmidt, 2009)

Přestože hodnota v riziku je sám o sobě jeden termín, tak pod něj spadá spousta metod a řešení výpočtu VaR. Měření rizika se dá dosáhnout mnoha metodami, z kterých každá má své výhody a nevýhody. Diskuse o nejlepší metodě pro výpočet hodnoty v riziku jsou rozsáhlé a nedá se jednoznačně určit, která metoda či kombinace metod je nejlepší. Hodnota v riziku není výhodná pouze pro statistiky, finanční poradce či pracovníky v risk managementu, ale také pro manažery různých firem a pro lidi, co chtějí sami investovat, aby mohli provést správná rozhodnutí při investování. Rozsah této práce zahrnuje nástroj risk managementu – hodnotu v riziku (VaR).

Hlavní myšlenkou metody VaR je zaznamenat nejvyšší ztrátu investice za určité období v dané konfidenční hladině. Když použijeme konfidenční hladinu 95%, která se často používá, a budeme odhadovat neznámý parametr konfidenčním intervalem, tak ze 100 dat bude zhruba 95 dat daný parametr obsahovat a přibližně 5 nikoli. Nejvýznamnější výhodou hodnoty v riziku je, že dokáže zhodnotit risk jediným číslem.

Cílem práce je sestrojit kód v programovacím jazyce R, který dokáže po zadání potřebných informací vypočíst hodnotu v riziku. Hodnota v riziku se dá spočítat mnoha způsoby, výstupem této práce jsou celkově čtyři kódy, které dokáží spočítat hodnotu v riziku dvěmi metodami, a to upravenou metodou Normálního VaR a Historickým VaR. U každé z těchto metod je rozdělení, pokud se počítá s investicí jako pohledávkou či závazkem.

Přínosem práce je zdrojový kód v R, který může být využit pro výpočet hodnoty v riziku a použit v budoucích on-line kalkulátorech rizika či pro samostatný program, který dokáže po zadání potřebných informací ukázat riziko dané investice.

2 Hodnota v riziku

Široce využívaný výpočet risku je hodnota v riziku na úrovni α. Tento výpočet je definován jako:

$$VaR_\alpha(X) = inf\{x, F_X(x) \geq \alpha\} = F_X^{-1}(\alpha) \tag{1}$$

kde X je náhodná proměnná s funkcí pravděpodobnostní míry (pdf) f_x a kumulovanou funkcí (cdf) F_x. (2012 Alemany R., Bolance C., Guillen M.)

Existuje mnoho metod pro výpočet hodnoty v riziku, které se dají rozdělit do tří hlavních přístupů, mezi které patří parametrické modely, neparametrické modely a semiparametrické modely.[1] Nejprve máme parametrické přístupy, které měří riziko pomocí křivek pravděpodobností (normální distribuce či založené na teorii extrémních hodnot EVT) a následným odvozením hodnoty v riziku z dané křivky. Jako druhou skupinu přístupů máme neparametrické modely, které využívají empirické rozdělení výnosů z historických vzorků dat. A jako poslední přístupy máme semiparametrické, které využívají jak flexibilní využití křivek parametrických metod, ale také neparametrických přístupů. V této práci budu využívat dva ze tří přístupů, a to parametrické a dále neparametrické metody.

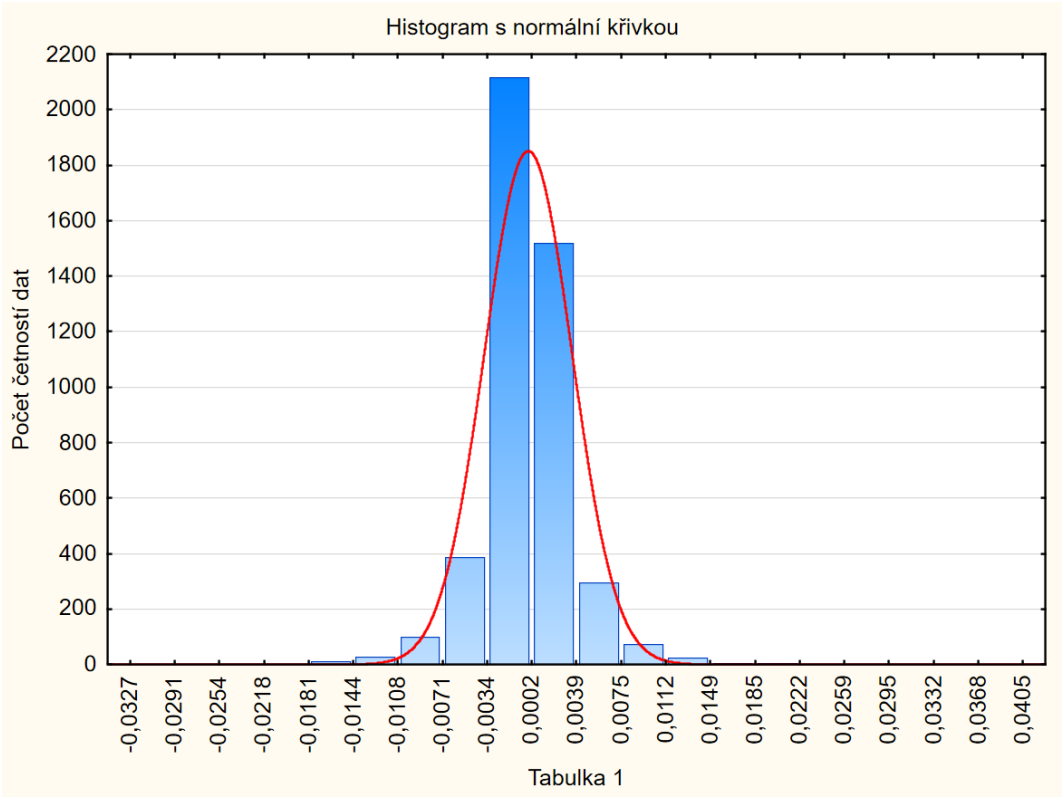
První přístup pro výpočet hodnoty v riziku je parametrický model. Parametrický model předpokládá libovolné rozdělení, v této práci se budeme zaměřovat na rozdělení normální. Pod podmínkou, že hodnota v riziku portfolia je v konfidenční hladině $1-\alpha\%$, tak hodnota v riziku se spočítá jako:

$$VaR_\alpha = \mu + \sigma_t G^{-1}(\alpha), \tag{2}$$

kde $G^{-1}(\alpha)$ je α kvantil standardní normální distribuce a σ_t je podmíněná směrodatná odchylka výnosů portfolia. (Abad et al., 2014)

[1] Čerpáno z: Mrkvička, T., Krásnická, M., Friebel, L., Volek, T., & Rolínek, L. (2018). Backtesting evaluation of Value-at-Risk methods for exchange rates in time horizon up to one year, strana 2

Obrázek 1: Histogram s normální křivkou



Zdroj: Autor práce

Jak je možno vidět na obrázku 1, tak jako vstup zde bylo použito mnoho dat s kursovními změnami EUR/CZK v čase a největší četnost dat je okolo prostředku normální křivky. Hodnota v riziku zkoumá hodnoty, které jsou na levém kraji dat tohoto histogramu. Když si tyto hodnoty spojíme například s denními změnami kursu, tak můžeme vidět, že nejvíce se kurs změní o málo, ale v některých případech vzroste či spadne o mnohem víc - a to jsou ty hodnoty, které nás v této práci budou zajímat.

2.1 Ukázkový praktický příklad hodnoty v riziku

Na tomto ukázkovém praktickém příkladu si jednoduše ukážeme, jak přesně funguje VaR a co nám počítá.

Příklad: *VaR = 10 tisíc Kč | Počet dní = 1 |* α *= 95%*

Když máme VaR s hodnotou 10 tisíc a na hladině $\alpha = 95\%$ tak můžeme říct, že v 95 dnech ze 100 dnů nebudeme mít vyšší ztrátu než 10 tisíc Kč. Dále můžeme říct, že maximální ztráta nám nastane alespoň v 5 dnech ze 100 (nebo v jednom dni z 20).

2.2 Popis metod výpočtu hodnoty v riziku

Nejčastější a nejjednodušší metoda pro výpočet hodnoty v riziku je metoda aproximace hodnoty v riziku pomocí normální aproximace. Normální rozdělení je určeno pouze dvěma parametry, a to střední hodnotou a směrodatnou odchylkou. Tyto parametry se ale vyvíjejí v čase, takže je třeba si stanovit nedávná data. Výhodou metody aproximace hodnoty v riziku pomocí normální aproximace je její jednoduchost a snadné použití. Když vezmeme v potaz obecné nevýhody parametrických modelů, tak hlavní nevýhodou je neschopnost modelovat těžké chvosty.[2]

Druhý přístup výpočtu hodnoty v riziku je semiparametrický model. Modely, které využívají normální distribuce těžko zaznamenávají těžké chvosty. Mezi parametrické modely patří také metoda GARCH (generalized autoregressive conditional heteroskedasticity), která řeší heteroskedasticitu dat.[3] GARCH je statistický model používaný pro data zasazená v čase a popisuje rozptyl aktuální odlišnosti od středové hodnoty jako funkci skutečné velikosti odlišností od středové hodnoty v předchozích časových obdobích. GARCH modely jsou využívány většinou při modelování finančních časových řad, díky kterým můžeme vidět volatilitu v čase u dané investice. Samotný GARCH model je ale nedostačující, proto se používá metoda GARCH+EVT (Extreme Value Theory)[4].

Metoda GARCH+EVT využívá EVT, která je aplikována na řadu residuí vytvořených z GARCH modelu. Dále je rozdělení reziduí upraveno za použití EVT a následně se daná residua implementují zpět do GARCH predikce budoucích hodnot. Velkou výhodou GARCH+EVT metody je vypořádání se s „černými labutěmi“ (těžkými chvosty) a s podmínkou, že rozptyl je závislý na parametru (heteroskedasticita). V této práci se touto metodou zabývat dále nebudu.

Třetí skupinou výpočtu hodnoty v riziku je neparametrický přístup. Neparametrické přístupy zásadně měří hodnotu v riziku pomocí odpovídajícího percentilu výnosu historic-

[2]Čerpáno z: Parametric value-at-risk. (c2014). Https://breakingdownfinance.com/. Retrieved April 13, 2021, from https://breakingdownfinance.com/finance-topics/risk-management/market-risk/parametric-value-at-risk/

[3]Čerpáno z: Kenton, W., James, M. (Ed.). (1999). GARCH Process. Investopedia. Retrieved April 13, 2021, from https://www.investopedia.com/terms/g/generalalizedautogregressiveconditionalheteroskedasticity.asp

[4]Čerpáno z: Mrkvička, T., Krásnická, M., Friebel, L., Volek, T., & Rolínek, L. (2018). Backtesting evaluation of Value-at-Risk methods for exchange rates in time horizon up to one year.

kého rozdělení. Mezi neparametrické přístupy patří metoda Historický VaR či Monte Carlo. Když budeme investovat do cizí měny, tak tato metoda zahrnuje do kursovního rizika také uměle vytvořené změny kursu v čase, které nejsou moc časté, ale jsou velmi výrazné. Tyto změny jsou způsobeny často nečekaným zásahem zvenčí, například politické nebo finanční krize, zásahy centrální banky nebo jiné události, které ovlivňují stabilitu měny. V parametrických výpočtech se tyto události a výjimky špatně předpovídají, ale díky neparametrickému historickému VaR můžeme tyto události určit a vzít v potaz při výpočtu, a to na základě dřívějších takových událostí. Ale vzhledem k vzácnosti těchto jevů je třeba dbát na dlouhou historii vývoje investice pro určení těchto změn. Historický VaR dobře aproximuje těžké chvosty, ale na druhou stranu špatně aproximuje chvosty lehké. Hlavní nevýhodou neparametrických přístupů je jejich neschopnost jít mimo rozsah již pozorovaných dat. Další nevýhodou je, že vyžadují dlouhou časovou řadu a také nejsou schopny adaptace na heteroscedasticitu[5].

3 Programovací jazyk R

V této práci vytvářím skript v programovacím jazyce R. Tento programovací jazyk má stejnojmenný software a je široce využíván zejména pro statistické výpočty. R je často používán lidmi v oboru statistiky pro vývoj statistického softwaru či softwaru pro analýzu dat.[6] Přestože oficiální R program má pouze interface příkazového řádku, tak existuje několik balíčků a softwarů, které přidávají také grafické rozhraní pro uživatele. Příkladem softwaru je například RStudio, které využívám v této práci.

3.1 Historie R

R pochází z programovacího jazyka S. Programovací jazyk R byl poprvé implementován v 90. letech dvěma členy fakulty na Aucklandské Univerzitě, Robert Gentleman a Ross Ihaka. Jazyk R se v průběhu 90. let vyvíjel a zlepšoval až nakonec v únoru roku 2000 byla vydaná první verze R 1.0.0.[7]

Software R se do dnešního dne vyvíjí a dnes existuje již více než 10 tisíc balíčků pro tento software.

[5] Čerpáno z: Mrkvička, T. (2019). kursovní riziko.

[6] Čerpáno z: What is R?. (2021). R. Retrieved March 16, 2021, from https://www.r-project.org/about.html

[7] Čerpáno z: What is R?. (c2021). MRAN. Retrieved March 15, 2021, from https://mran.microsoft.com/documents/what-is-r

3.2 Výhody R

Jak jsem již zmínil, tak programovací jazyk R je široce využíván pro statistické účely. Výhodou R oproti jiným programovacím jazykům je velké množství balíčků či add-onů, které usnadňují či vylepšují práci v softwaru. Další výhodou je poměrně velká oblíbenost mezi uživateli, protože podle TIOBE indexu, který měří popularitu programovacích jazyků, R je 16. nejpopulárnější programovací jazyk k datu 04.04.2021.[8] To znamená, že je na internetu velký počet typů a rad pro programovaní v R a také díky tomu roste počet balíčků pro stejnojmenný software. Výhodou R je taktéž propojitelnost s jinými, více populárnějšími jazyky, jako je C, C++, Java, .NET či Python. Také je v základní verzi možné vytvářet grafy, ale je také možné využít mnoho jiných balíčků pro vytváření grafů, které jsou k dispozici.

Tabulka 1: Tabulka vývoje programovacích jazyků v letech 2006 až 2021

Programovací jazyk	2006	2011	2016	2021
C	2	2	2	1
C++	3	3	3	4
Java	1	1	1	2
Python	8	7	5	3
R	-	33	17	9

Zdroj: Autor práce

V tabulce 1 jsou data o vývoji popularity pěti programovacých jazyků od roku 2006 až po rok 2021. Hodnoty udávají místo na žebříčku popularity, takže hodnota 1 je nejpopulárnější programovací jazyk pro daný rok.

[8] Čerpáno z: TIOBE Index. (c2020). TIOBE. Retrieved April 13, 2021, from https://www.tiobe.com/tiobe-index/

4 Metodika pro výpočet VaR

V této sekci jsou popsané různé postupy výpočtu hodnoty v riziku. Nejprve se zaměříme na metodu aproximace VaR pomocí normální aproximace, dále se podíváme na její upravenou variantu a nakonec zde popíšeme neparametrickou metodu výpočtu VaR.

V podkapitolách budeme obecně pracovat s následujícími hodnotami:

- α = povolená chyba modelu
- c = čas expozice v pracovních dnech
- v = velikost expozice

Je potřeba brát v potaz, že VaR udává maximální ztrátu na kursovním riziku při realizaci platby v době splatnosti, která je vypočtena s povolenou chybou (1-α)%. Jinak řečeno, v (1-α)% případů bude ztráta menší a v α% případů bude ztráta větší.

4.1 Normální VaR

Tato metoda pracuje na principu aproximace rozdělení kursovních změn normálním rozdělením a je to jedna z nejjednodušších metod pro odhad kursovního rizika.

Pro výpočet normální distribuce potřebujeme dva parametry: střední hodnotu μ a směrodatnou odchylku σ. Takže tato metoda zahrnuje pouze odhad těchto dvou parametrů. Problém nastává tehdy, když používáme zastaralá data, protože tyto hodnoty se v čase mění, takže musíme používat nejnovější data.

Pro výpočet této metody budeme potřebovat tyto hodnoty:

- α = povolenou chybu modelu,
- c = čas expozice v pracovních dnech,
- v = velikost expozice,
- μ = střední hodnotu,
- σ = směrodatnou odchylku,
- A_0 = určitý kurs.

Z těchto proměnných musíme stanovit výpočet pouze u střední hodnoty a směrodatné odchylky, ostatní parametry jsou fixní a záleží na konkrétních případech. Nejprve potřebujeme kursovní změny, z kterých můžeme pomocí rozdílu logaritmované řady standardizovat rozptyl. Obecný vzorec pro výpočet je následující:

$$B_n = ln(A_i) - ln(A_{c+i}), i = 0, ..., n-1 \tag{3}$$

A_i jsou kursy v čase i, kde i je den v minulosti. Pro výpočet, přesněji odhad, hodnoty μ využijeme průměr z dat vypočtených v B_n.

$$\mu = mean(B_0, B_n) \tag{4}$$

Dále potřebujeme parametr σ (směrodatná odchylka), která se vypočítá následovně:

$$\sigma = stdev(B_0, B_m) \tag{5}$$

Také bude zapotřebí vzorec pro výpočet normální distribuce cdf: [9]:

$$\phi(x) = P(Z \leq x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} exp\left\{-\frac{u^2}{2}\right\} du \tag{6}$$

Z tohoto uděláme inverzní funkci a dále ji budeme nazívat $Norm.inv$. Nakonec můžeme všechny tyto parametry dosadit do vzorce, a to buď do VaR pro pohledávku či VaR pro závazek:

VaR pro pohledávku[10]:

$$VaR = -v * (e^{ln(A_0)+(\mu - Norm.inv(1-\alpha,0,1)*\sigma)} - A_0) \tag{7}$$

VaR pro závazek[10]:

$$VaR = v * (e^{ln(A_0)+(\mu + Norm.inv(1-\alpha,0,1)*\sigma)} - A_0) \tag{8}$$

4.2 Normální VaR - upravená verze

Normální rozdělení je určeno dvěma parametry: střední hodnotou a směrodatnou odchylkou. Nejprve musíme zavést kursovní změny, z kterých budeme parametry počítat. Tyto

[9] Čerpáno z: NORMINV Function: Normal (Gaussian) Distribution. (2021). Probability Course. Retrieved April 14, 2021, from https://www.probabilitycourse.com/chapter4/4_2_3_normal.php

[10] Čerpáno z: Mrkvička, T. (2019). kursovní riziko.

změny můžeme vypočíst jako rozdíly logaritmované řady. kursovní změny $A_i, i = 0, ..., n-1$ tedy transformujeme jako rozdíly logaritmů za účelem standardizování rozptylu.

$$B_i = ln(A_i) - ln(A_{i+1}), i = 0, ..., n-1 \tag{9}$$

A_0 je současný směnný kurs a A_i jsou kursy v čase *i*, kde *i* odpovídá dni v minulosti. Celkově zde pracujeme s *n* dat, která jsou vypočítaná z kursů A_0 až A_n. Jak již bylo zmíněno, normální rozdělení je určeno střední hodnotou $m(B_0, ..., B_{n-1})$ a směrodatnou odchylkou $s(B_0, ..., B_{m-1})$. Pro odhad střední hodnoty je lepší použít více dat, aby zohledňoval delší vývoj a velké skoky. Dále je dobré brát v úvahu, že pro směrodatnou odchylku je kvůli heteroskedasticitě dat potřeba zvážit, kolik dat je zapotřebí využít pro výpočet směrodatné odchylky. Kvůli výše zmíněné heteroskedasticitě budeme proto brát v úvahu maximálně poslední 3 měsíce logaritmických rozdílů kursovní měny A_i.Po vypočtení rozdílů logaritmů, střední hodnoty a směrodatné odchylky jsme dostali parametry jednodenního přírůstku. Ukazuje se, že řada B nemá žádné krátkodobé ani dlouhodobé korelace. Dále vidíme, že směrodatná odchylka *SD* (řady B s c denní expozicí) odpovídá $\sqrt{c} * SD$(SD je řady B). A proto můžeme logaritmickou řadu *B* prohlásit za „bílý šum“ a použít model náhodné procházky[11].

„Hypotéza náhodné procházky tvrdí, že změny v ceně akcií mezi jednotlivými obchodními dny jsou zcela náhodné, a tedy nepředpověditelné. Tato hypotéza, kterou poprvé zformuloval Bachelier v roce 1900, zůstává jednou z klíčových otázek analýzy akciových kursů. Ceny akcií na mnoha stabilních akciových trzích (především velkých) tuto hypotézu sice potvrzují, některé trhy však na základě provedených testů vykazují od této hypotézy odchylky.”[12]

$$ln(B_{-c+1}) - ln(B_0) = suma_{B_i} = 1^c\, B_i,\ kde\ B_i \sim N(P_1, SD_1) \tag{10}$$

Tudíž můžeme říct, že:

$$ln(B_{-c+1}) - ln(B_0) \sim N(c * P_1, \sqrt{c} * SD_1) \tag{11}$$

[11] Čerpáno z: Mrkvička, T. (2019). kursovní riziko.

[12] Citováno z: Cipra, T. (2005). Praktický průvodce finanční a pojistnou matematikou (II.). Ekopress. - strana 117

Dále je zapotřebí vzorec pro výpočet normální distribuce cdf, který je následovný[13]:

$$\phi(x) = P(Z \leq x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} exp\left\{-\frac{u^2}{2}\right\} du \tag{12}$$

Z této funkce uděláme inverzní funkci $-\phi(x)$.

Nakonec můžeme vytvořit výsledné vzorce pro výpočet VaR, které jsou:

VaR pro pohledávku[14]:

$$VaR = v * (A_0 - e^{ln(A_0)+Norm.inv(\alpha,c*P_1,\sqrt{c}*SD_1)}) \tag{13}$$

VaR pro závazek[14]:

$$VaR = -v * (A_0 - e^{ln(A_0)+Norm.inv(1-\alpha,c*P_1,\sqrt{c}*SD_1)}) \tag{14}$$

V této práci budu programovat tuto upravenou metodu pomocí R softwaru.

4.3 Neparametrická metoda - Historický VaR

Další metodou výpočtu hodnoty v riziku je Historický VaR. Tato metoda předpokládá, že vývoj investice v minulosti je dobrý indikátor pro kalkulaci hodnoty v riziku v blízké budoucnosti. Tato metoda si umí dobře poradit s těžkými chvosty, ale má problémy s chvosty lehkými. (Mentel, 2013 přeloženo)

Nejprve je zapotřebí upravit data na logaritmované diference se vzdáleností dat c. Pro upravení vstupních dat tedy použijeme tento obecný vzorec:

$$B_i = ln(A_i) - ln(A_{i+c}), i = 0, ..., n-1 \tag{15}$$

, kde A_i jsou kursy v čase i. Hodnota c udává vzdálenost dat, jinak řečeno pro investory hodnota c znamená počet dní do splatnosti investice.

Dále tato upravená data použijeme k nalezení maximální či minimální hodnoty změny investice. Tato hodnota je zapotřebí k výpočtu maximální ztráty za určené období u dané investice.

[13]Čerpáno z: NORMINV Function: Normal (Gaussian) Distribution. (2021). Probability Course. Retrieved April 14, 2021, from https://www.probabilitycourse.com/chapter4/4_2_3_normal.php

[14]Čerpáno z: Mrkvička, T. (2019). kursovní riziko.

Pro výpočet maximální ztráty na kursovém riziku u pohledávky použijeme vzorec[14]:

$$Hist_{max} = -v * (e^{ln(A_0)+B_{min}} - A_0) \tag{16}$$

, kde v znamená velikost vkladu v původní měně investice, A_0 je aktuální kurs investice a B_{min} je minimální hodnota z upravených historických dat dané investice.

Vzorec pro výpočet maximální ztráty na kursovém riziku u závazku vypadá následovně[15]:

$$Hist_{max} = v * (e^{ln(A_0)+B_{max}} - A_0) \tag{17}$$

Tento vzorec pro výpočet hodnoty v riziku u závazku používá stejné proměnné, až na B_{max}, která nabírá hodnoty největší změny v souboru upravených dat na změny kursu se vzdáleností c.

U výpočtu Historického VaR je také dobré určit kvantil případů, kdy bude ztráta vyšší než výsledná hodnota. Například v 5% případů je ztráta vyšší než 100Kč u dané investice. Nejdříve je zapotřebí si určit kvantil z upravených dat na hladině α, kde α je procento případů, pro které hledáme danou ztrátu. Pro pohledávku se kvantil vypočítá s α% a u závazku se kvantil vypočítá s $(1 - \alpha)$%.

Výpočet pro pohledávku vypadá následovně[15]:

$$Hist_{\alpha} = -v * (e^{ln(A_0)+B_{\alpha}} - A_0) \tag{18}$$

Hodnota B_{α} je kvantil z upravených dat na procentuální hladině α.

Pro výpočet případů, kdy bude ztráta vyšší než výsledná hodnota u závazku použijeme vzorec[15]:

$$Hist_{\alpha} = v * (e^{ln(A_0)+B_{1-\alpha}} - A_0) \tag{19}$$

,kde proměnná $B_{1-\alpha}$ nabírá hodnoty kvantilu upravených dat na procentuální hladině $(1-\alpha)$.

V této práci je nakódovaná verze této metody Historického VaR v softwaru R.

[15]Čerpáno z: Mrkvička, T. (2019). kursovní riziko.

5 Řešení programování VaR v R

Cílem této práce je naprogramovat skript v programovacím jazyce R, který bude moct vypočítat hodnotu v riziku pro pohledávky a závazky pomocí dvou různých metod, a to upravenou metodou Normální VaR a neparametrickou metodou pro výpočet VaR.

Před samotným programovaním je nutné si určit různé funkce, proměnné, vstupy a výstupy. Dále tyto funkce ošetřit, aby bylo více možností, jak si uživatel může vypočítat hodnotu v riziku. Poté se můžeme pustit do samotného programování a testování.

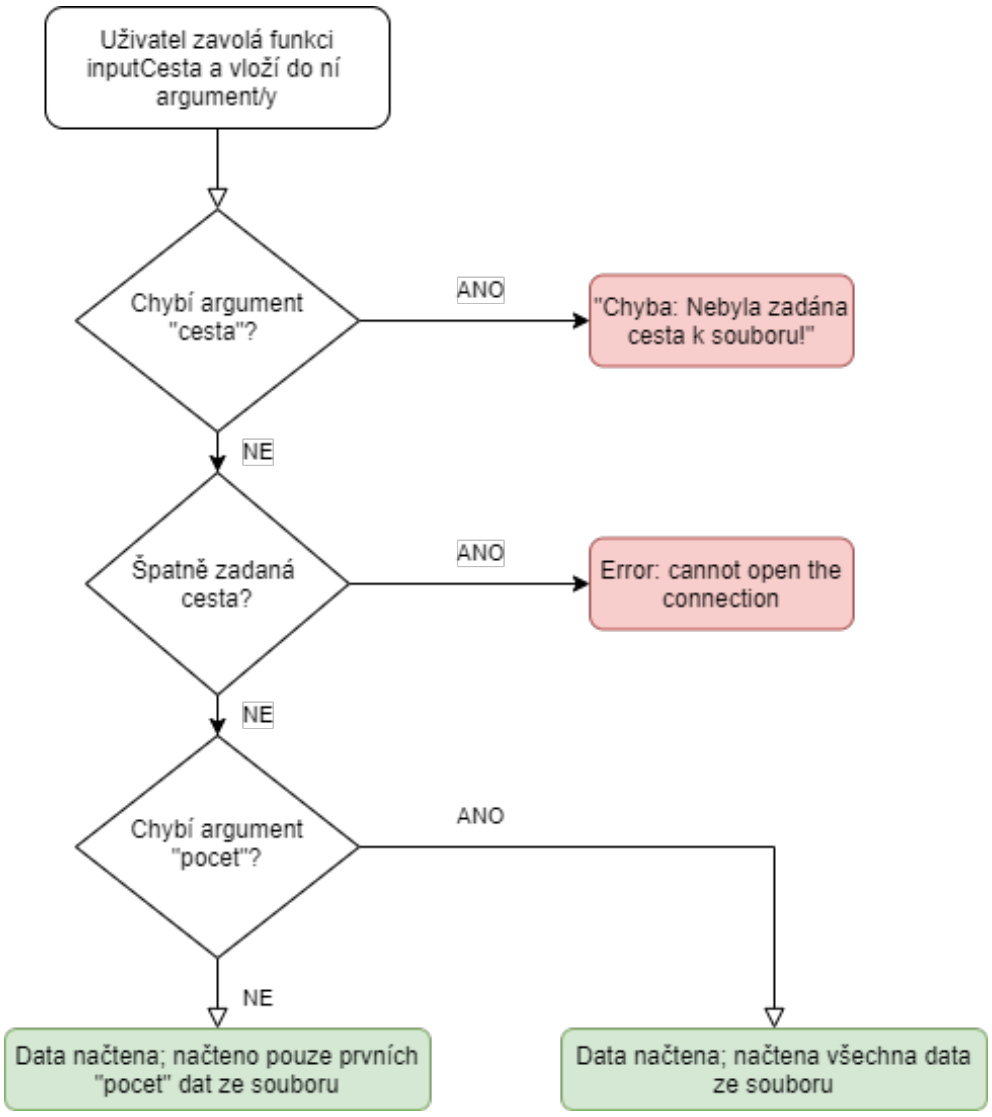
5.1 Funkce a řešení pro obě metody

5.1.1 Řešení vstupu

Jako první funkci, která musela být vyřešena, bylo načítání vstupního souboru do programu. Vstupní soubor ve formátu .csv se načte po zadání cesty do adresáře a jménem souboru. Tato cesta se může změnit, pokud si to uživatel přeje.

Na obrázku 2 můžeme vidět úvodní flowchart dané funkce. Tento graf byl sestaven před samotným kódem a pomohl ujasnit, jaké výstupy tato funkce musí mít a jak ošetřit různé chyby vstupu.

Obrázek 2: Flowchart funkce vstupu



Zdroj: Autor práce

Funkce inputCesta slouží k načtení historických dat o denních změnách investice ze souboru .csv bez hlavičky a má dva argumenty, z toho jeden je povinný a druhý nepovinný. Povinný argument je „cesta", kde je zapotřebí zadat správnou cestu k souboru v textovém formátu - tzn. pomocí uvozovek na začátku a na konci. Dále existuje další argument v této funkci jménem „pocet". Tento argument slouží k tomu, abychom nenačítali příliš mnoho dat, tak můžeme omezit tento počet pomocí tohoto druhého argumentu.

Výstupem této funkce pro uživatele je pouze oznámení, že data byla načtena správně a dále vypíše, kolik dat tato funkce načetla. Na obrázku 2 můžeme vidět flowchart dané funkce a všechny její možné výstupy.

Zdrojový kód 1 je první verze načtení vstupu. Tento vstupní kód stačí pro soubory, které jsou bez hlavičky a mají čárkové separátory.

Na druhém řádku zdrojového kódu 1 vidíme dva argumenty - *cesta* a *pocet*.

Argument *cesta* je povinný argument funkce a očekává datový typ *string*. Zde vložíme úplnou cestu k souboru, ze kterého chceme načíst data do programu a ohraničíme jí uvozovkami (například "C:/Users/FrantišekŽiška/OneDrive/Plocha/Bakalářka/data_210307.csv")

Druhý argument v této funkci je nepovinný argument *pocet*, který je datového typu *integer*, což znamená, že očekává číslo. Zadané číslo udává, kolik hodnot si uživatel přeje načíst.

Při řešení Historického VaR z jiného vstupního souboru bylo ale zjištěno, že tato funkce postrádá argumenty, které by mohly rozšířit možnosti funkce načítání souboru.

Zdrojový kód 1: Kód v R funkce inputCesta 1. verze

```
#Funkce na načtení historických dat o změnách investice v čase ze souboru csv bez headeru (cesta=cesta k souboru, optional: pocet=kolik dat chceme načíst/pokud prázdné, tak načte všechna data ze souboru)
inputCesta <- function(cesta, pocet){
  if (missing(cesta)){
    return("Chyba: Nebyla zadána cesta k souboru!")
  }
  else if (missing(pocet)){
    readIt <- read.csv(cesta, colClasses=c('numeric'), header = FALSE)
    assign("input", readIt, envir = .GlobalEnv)
    my_list <- list("Úspěšně načteny data z: " = cesta, "Počet načtených dat: " = nrow(readIt))
    return(my_list)

  }
  else{
    readIt <- read.csv(cesta, colClasses=c('numeric'), header = FALSE, nrows=pocet)
    assign("input", readIt, envir = .GlobalEnv)
    my_list <- list("Úspěšně načteny data z: " = cesta, "Počet načtených
    dat: " = nrow(readIt))
    return(my_list)
  }
}
```

Zdroj: Autor práce

Zdrojový kód 2 je upravená verze zdrojového kódu 1. Tato načítací funkce umožňuje načíst soubory jak s hlavičkou, tak i bez hlavičky a také určit separátory dat v .csv souboru. Na druhém řádku zdrojového kódu 2 můžeme vidět, že oproti staré verzi zde přibyly nové vstupní argumenty - *hasHeader* a *separator*.

Argument *hasHeader* je povinný argument a je datového typu *boolean*, takže zde můžeme zadat buď hodnotu *TRUE*, pokud nahraný soubor má hlavičku s popisky dat a nebo *FALSE*, pokud nahraný soubor hlavičku s popisky dat nemá.

Argument *separator* je povinný argument a očekává datový typ *character*, takže když zadáváme do tohoto argumentu separátor dat, tak jej zadáme mezi uvozovkami (například ";"). Pro budoucí výpočty a práci s nahranými daty se bude pracovat s proměnnou *input*, která je na řádku 8 a 14 (viz. Zdrojový kód 2) zvidilněna i mimo tuto konkrétní funkci a jsou do ní uložena data z nahraného souboru.

Zdrojový kód 2: Kód v R funkce inputCesta 2. verze

```
1  #Funkce na načtení historických dat z .csv souboru (cesta = cesta k
   ↪  souboru, hasHeader = pokud má soubor hlavičku, separator =
   ↪  separátor dat použit v souboru, optional: pocet = počet řádků,
   ↪  které chceme načíst)
2  inputSoubor <- function(cesta, hasHeader, seperator, pocet){
3    if (missing(cesta)){
4      return("Chyba: Nebyla zadána cesta k souboru!")
5    }
6    else if (missing(pocet)){
7      readIt <- read.csv(cesta, header = hasHeader, fill=TRUE, sep =
       ↪  seperator)
8      assign("input", readIt, envir = .GlobalEnv)
9      my_list <- list("Úspěšně načteny data z: " = cesta,
       ↪  "Počet načtených dat: " = nrow(readIt))
10     return(my_list)
11   }
12   else{
13     readIt <- read.csv(cesta, header = hasHeader, fill=TRUE, sep =
       ↪  seperator, nrows=pocet)
14     assign("input", readIt, envir = .GlobalEnv)
15     my_list <- list("Úspěšně načteny data z: " = cesta,
       ↪  "Počet načtených dat: " = nrow(readIt))
16     return(my_list)
17   }
18 }
```

Zdroj: Autor práce

Příklad zavolání funkce inputSoubor a její výstup můžeme vidět v kódu 3.

Zdrojový kód 3: Zavolání inputSoubor a její výstup

```
1  inputSoubor(
2  "C:/Users/FrantišekŽiška/OneDrive/Plocha/Bakalářka/data_210307.csv",
   ↪  TRUE, ";", 5000)
3
4  Výstup:
5  `Úspěšně načteny data z: `
6  [1]
7  "C:/Users/FrantišekŽiška/OneDrive/Plocha/Bakalářka/data_210307.csv"
8  `Počet načtených dat: `
9  [1] 5000
```

Zdroj: Autor práce

5.1.2 Řešení výběru sloupce pro výpočet VaR

Pokud uživatel nahraje soubor, kde je více než jeden sloupec, tak je zapotřebí, aby tento program také uměl vybrat ten sloupec, který si uživatel přeje použít pro výpočet hodnoty v riziku. Přesně pro řešení tohoto problému existuje v programu funkce *vybratSloupec*.

Tato funkce má dva argumenty (viz. řádek 2 zdrojového kódu 4), z toho jeden povinný a druhý nepovinný. První argument *vyberData* očekává číslo sloupce v .csv souboru, ze kterého má čerpat data o vývoji kursu. Druhý argument *vyberCas* je nepovinný a očekává číslo sloupce z původního .csv souboru, kde se nacházejí datumy v korelaci s daty o vývoji investice.

Na řádku 3 je podmínka, která zkontroluje, pokud uživatel zadal sloupec s datumy. Pokud ne, tak proběhne kód na řádcích 4-7 a pokud ano, tak proběhne kód na řádcích 10-14. Na řádku 4 se z pomocné proměnné *input* nahrají všechna data ze sloupce *vyberData*, který zadal uživatel, do souboru dat *grafX*. Dále proběhne funkce *na.omit()* na řádce 5, která vymaže z *grafX* všechny data, který nabírají hodnotu *NA* nebo jsou prázdné. Jde o formu ošetření vstupu, aby mohly budoucí výpočty prováděné s daty z *grafX* fungovat správně a bez chybových hlášení. Dále pomocí příkazu *names()* funkce přejmenuje první sloupec *grafX* na *Data*. A nakonec, stejně jako u proměnné *input*, tak zviditelníme *grafX* i mimo tuto funkci, aby s ní mohly pracovat i jiné funkce. Stejný postup je u druhé strany podmínky, akorát se do *grafX* ukládají dva sloupce dat.

Zdrojový kód 4: Řešení výběru požadovaného sloupce z .csv souboru

```
# funkce pro výběr sloupce v .csv souboru(argumenty: číslo sloupce
↪  s daty, nepovinné: číslo sloupce s datumy)
vybratSloupec <- function(vyberData, vyberCas){
  if (missing(vyberCas)) {
    grafX<- data.frame(input[1:nrow(input),vyberData])
    grafX <- na.omit(grafX)
    names(grafX)[1] <- "Data"
    assign("grafX", grafX, envir = .GlobalEnv)
  }
  else {
    grafX<- data.frame(input[1:nrow(input),vyberCas],
    ↪  input[1:nrow(input),vyberData])
    grafX <- na.omit(grafX)
    names(grafX)[1] <- "Datum"
    names(grafX)[2] <- "Data"
    assign("grafX", grafX, envir = .GlobalEnv)
  }
}
```

Zdroj: Autor práce

5.1.3 Řešení vykreslení histogramu a spojnicového grafu

Funkce na vykreslení grafu se liší podle toho, která data chceme vykreslit. Pro surová vstupní data, neboli samotné kursy, použijeme spojnicový graf. Pro zpracovaná data použitelná k výpočtu je ideální histogram s normální křivkou, který udává četnost nahraných dat. Nejdříve se podíváme na řešení histogramu.

Ve zdrojovém kódu 5 se nejdříve vytvoří proměnná *std*, do které se vloží směrodatná odchylka *logSloupec*. V souboru dat *logSloupec* se nachází upravená data o změnách kursu investice v čase. Dále následuje vytvoření histogramu ze sloupce *Data* ze souboru dat *logSloupec* (viz. řádek 4 zdrojový kód 5) a poté se na tento graf vytvoří normální křivka (viz. řádek 5 a 6 ve zdrojovém kódu 5).

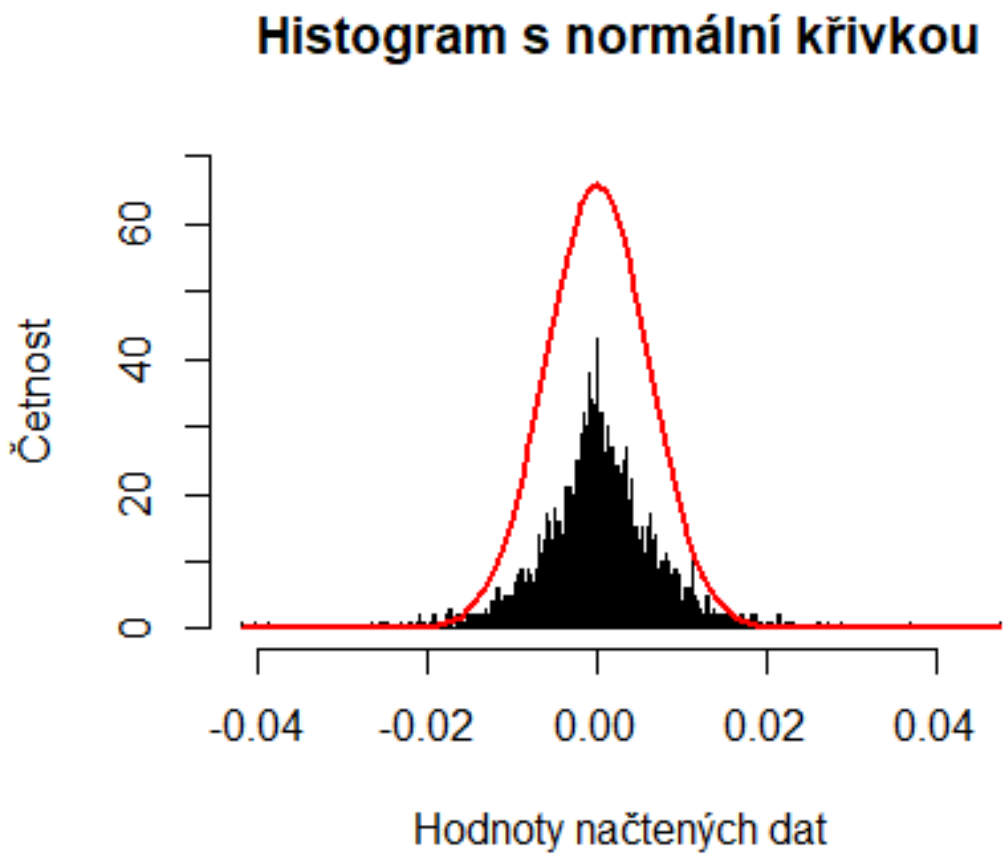
Na obrázku 3 můžeme vidět výstup ze zdrojového kódu 5 s počtem nahraných dat 5676 a jde o data změn kursu EUR/USD. Z tohoto grafu můžeme vidět, že prostředek normální křivky je vyšší než četnost dat v grafu a na krajích jsou data, která jsou vyšší než daná křivka. To znamená, že zde existují těžké chvosty.

Zdrojový kód 5: Řešení vytvoření histogramu s normální křivkou

```r
#funkce pro přehledový histogram s normální křivkou dat z logSloupec
vytvorGrafZmen <- function(){
  std <- sqrt(var(logSloupec$Data))
  hist(logSloupec$Data, ylim = c(0,70), breaks=nrow(grafX)/2, col="blue",xlab="Hodnoty načtených dat", ylab="Četnost", main="Histogram s normální křivkou")
  curve(dnorm(x, mean=mean(logSloupec$Data), sd=std),
        col="red", lwd=2, add=TRUE, yaxt="n")
}
```

Zdroj: Autor práce

Obrázek 3: Histogram s normální křivkou pomocí R



Zdroj: Autor práce

Vzhledem k formě vstupních dat, které jsou samotné kursy, tak pro vyobrazení grafu vstupních dat je nejlepší využít jednoduchý vývojový spojnicový graf. Tento druh grafu nejlépe vykreslí vstupní data.

Zdrojový kód 6 je řešení k vytvoření spojnicového grafu. Na řádku 2 se vytváří funkce *vytvorGrafKurs*, který nepotřebuje od uživatele žádné argumenty. Tyto argumenty nepotřebuje, protože si data vezme ze souboru dat *grafX* díky funci *inputSoubor*. Nahrává se zde pouze jeden sloupec s daty o vývoji kursu od nejstarších dat až po ty nejnovější.

Na řádku 3-10 probíhá samotné vytváření spojnicového grafu. Pomocí vestavěné funkce *plot()* můžeme pomocí několika argumentů vytvořit spojnicový graf. Nejdříve do prvního argumentu dáme data z *grafX* a to pouze data ze sloupce s konkrétními daty o vývoji měny. Dále určíme název grafu, název osy x, název osy y, typ grafu a nakonec barvu spojnicového grafu.

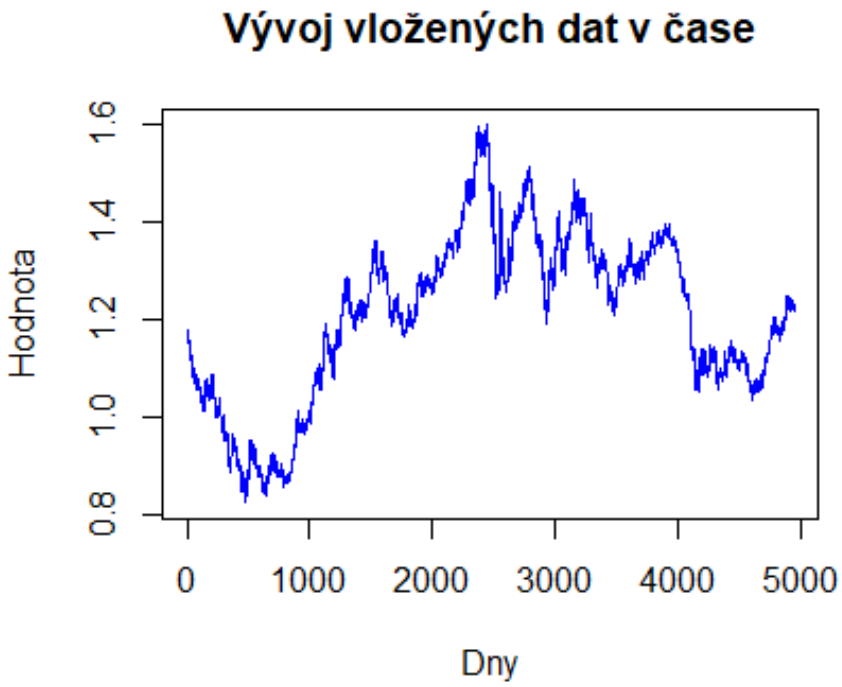
Když vložíme vstupní soubor o vývoji kursu EUR/USD s velikostí dat 5000, tak výsledný graf můžeme vidět na obrázku 4. Graf na ose x zobrazuje počet dní (0 nejstarší a 5000 nejaktuálnější data) a na ose y jakých hodnot data nabírají.

Zdrojový kód 6: Řešení spojnicového grafu u Historického VaR

```
1 #funkce na vytvoření grafu vývoje vložených dat v čase
2 vytvorGrafKurs <- function(){
3   plot(grafX$Data,
4        main="Vývoj vložených dat v čase",
5        xlab ="Dny",
6        ylab="Hodnota",
7        type="l",
8        col="blue")
9 }
```

Zdroj: Autor práce

Obrázek 4: Spojnicový graf Historického VaR



Zdroj: Autor práce

5.1.4 Řešení vyčištění paměti po ukončení programu

Program R si ukládá hodnoty do proměnných do tzv. *Environment*, kde tyto informace zůstávájí uloženy i po zavření programu. Kdyby nebyl tento *Environment* vyčištěn, tak mohou nastat problémy při výpočtech hodnoty v riziku. Proto na konci každého programu u každé metody je tento krátký kód (viz. zdrojový kód 7).

Funkce *rm()* na řádce 2 vyčistí *Environment* od všech uložených proměnných, aby v dalších relacích programu nezpůsobovaly problémy.

Funkce *gc()* slouží k uvolnění paměti, co tento program využíval a také vypíše údaje o využití paměti v systému.

Poslední funkce *dev.off()* slouží k vyčištění grafů zanechaných v mezipaměti.

Zdrojový kód 7: Řešení vyčištění paměti po ukončení programu

```
#vyčištění Environment (uložených proměnných)
rm(list = ls(all.names = TRUE))
#uvolnění paměti + ukázání paměti
gc()
#vyčištění grafů
dev.off(dev.list()["RStudioGD"])
```

Zdroj: Autor práce

5.2 Normální VaR - upravená metoda

Jako první bude řešení upravené metody Normálního VaR. Cílem této metody je číselný výstup, který bude udávat, jaká je maximální ztráta na hladině spolehlivosti α jak u pohledávky, tak u závazku. Metodika výpočtu již byla popsána v minulé kapitole, takže tato kapitola je zaměřena na programování a ukázky kódu. Vstup historických dat o změnách investice, vykreslení grafů, načtení požadovaného sloupce a uvolnění paměti bylo již vyřešeno v minulé sekci. Je třeba vzít v potaz, že výpočet VaR pomocí Normálního VaR potřebuje upravit vstupní data do formy, se kterou může správně aproximovat.

5.2.1 Řešení úpravy vstupních dat

Úprava vstupních dat kursů je zásadní pro správný výpočet hodnoty v riziku. Program očekává, že vstupní data budou v samotných kursech a ne v logaritmovaných rozdílech řady změn kursů v čase. Tato transformace dat slouží k standardizování rozptylu.

Zdrojový kód 8 je aktuální řešení úpravy vstupních dat na požadovanou formu této metody. Na řádce 2 je vytvořená funkce bez argumentů. Uvnitř této funkce se vytvoří nový *data.frame* pojmenovaný *workSloupec* a rovnou do něj uloží data z proměnné *grafX*, která obsahuje data o vývoji samotného kursu. Funkce ošetří prázdná pole *workSloupec* a vytvoří prázdný *data.frame logSloupec*. Ten bude sloužit později jako výstup této funkce. Na řádce 7 začíná cyklus *for()*, díky kterému funkce provede operaci na řádce 8 tolikrát, kolikrát je *workSloupec* dlouhý a o jednu hodnotu kratší, protože poslední hodnota v sloupci nikdy nemůže mít následující hodnotu k odečtení. Nakonec proběhne pojmenování sloupce s nově uloženými daty v *logSloupec* a zpřístupní tento *data.frame* i pro ostatní funkce.

Zdrojový kód 8: Řešení úpravy dat na logaritmické rozdíly

```r
# funkce pro úpravu dat na rozdíly logaritmického rozdělení
upravaDat <- function (){
  workSloupec <- data.frame(grafX$Data)
  workSloupec <- na.omit(workSloupec)
  logSloupec <- data.frame()

  for (i in 1:(nrow(workSloupec)-1)) {
    logSloupec[i,1] =  log(workSloupec[i,1]) -
    ↪  log(workSloupec[(i+1),1])
  }
  names(logSloupec) <- "Data"
  assign("logSloupec", logSloupec, envir = .GlobalEnv)
}
```

Zdroj: Autor práce

Výstup *logSloupec* bude vstupem pro vytvoření grafu změn v kursech (*vytvorGrafZmen*) a v konečné metodě pro výpočet Normálního VaR (*pohledVar* a *zavazekVar*).

5.2.2 Řešení funkce pro výpočet pohledávky a závazku

Pro vytvoření řešení pro výpočet Normálního VaR je nejdříve zapotřebí určit dva nezbytné parametry - střední hodnotu a směrodatnou odchylku. Střední hodnota i směrodatná odchylka se dají vypočíst pomocí built-it funkcí v programu R, a to díky *mean()* pro střední hodnotu (v tomto případě průměr) a *sd()* pro směrodatnou odchylku.

Ve Zdrojovém kódu 9 uložíme střední hodnotu a směrodatnou odchylku z načtených dat *input* do proměnných *odhadMu* a *stDev*.

Zdrojový kód 9: Určení střední hodnoty a směrodatné odchylky

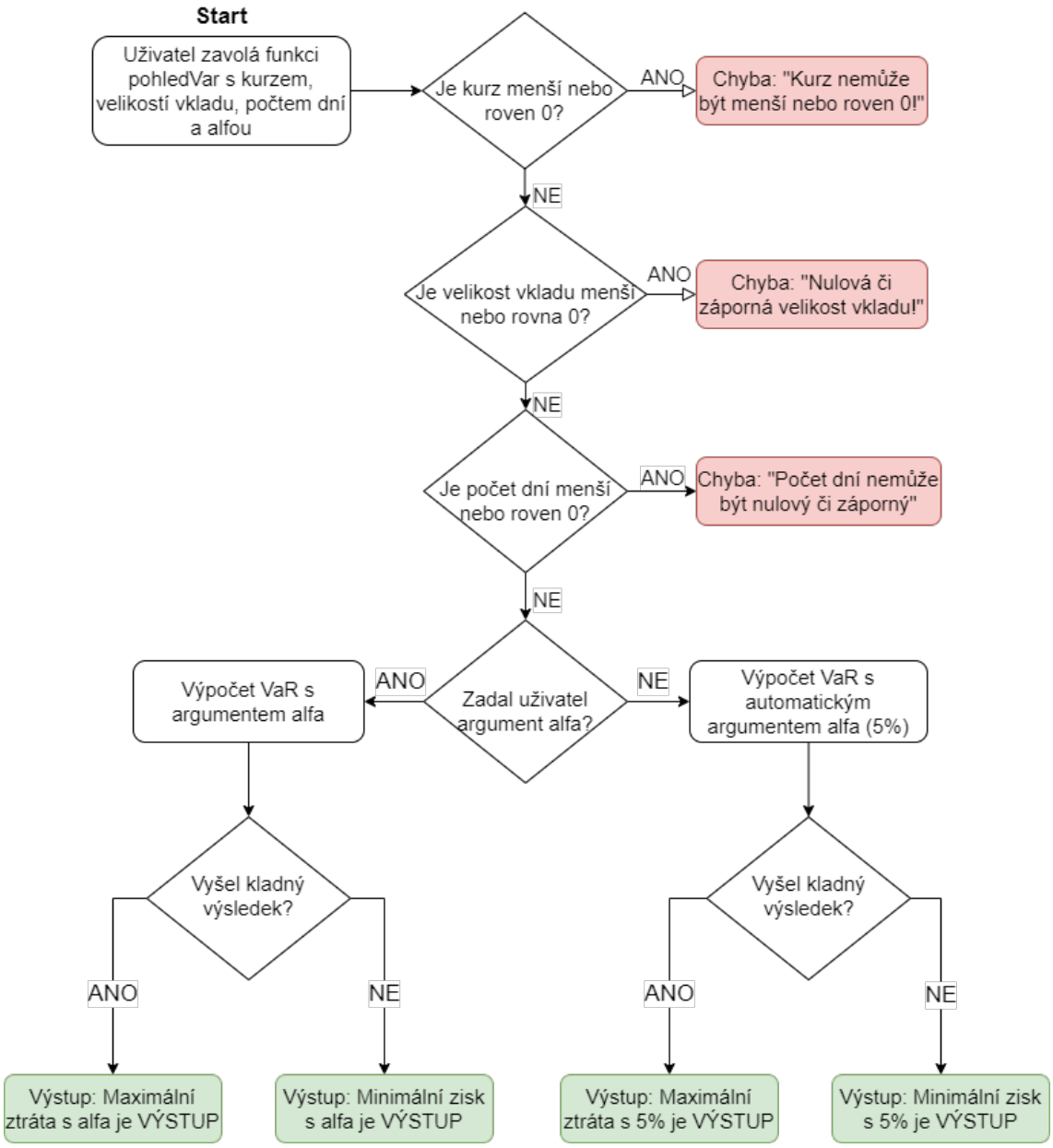
```r
#výpočet a uložení průměru ze souboru input do odhadMu
odhadMu <- mean(input$V1, na.rm = FALSE)

#výpočet a uložení směrodatné odchylky ze souboru input do stDev
stDev <- sd(input$V1, na.rm = FALSE)
```

Zdroj: Autor práce

Je nutné podotknout, že výpočet střední hodnoty i směrodatné odchylky jsou stejné a jsou zapotřebí jak pro pohledávku, tak i pro závazek. Poté, co si uložíme tyto dva potřebné parametry, kód může pokračovat a přejít k samotné funkci výpočtu. Zde se výpočty budou lišit, jeden pro pohledávku a druhý pro závazek.

Na obrázku 5 můžeme vidět počáteční návrh pro výpočet Normálního VaR pro pohledávku i pro závazek. Největším problémem bylo vyřešit chyby vstupu, jelikož metodika výpočtu upravené metody Normální VaR již byla popsána v minulé kapitole. Nejdříve uživatel zavolá funkci *pohledVar* a do ní zadá potřebné argumenty. Záleží na uživateli, pokud zadá pouze tři argumenty nebo zadá i čtvrtý argument, který je nepovinný. Poté, co zadá svůj počet argumentů, tak funkce nejprve zkontroluje, pokud je zadaný kurs menší nebo roven nule. Pokud ano, tak funkce vyhodí chybu, protože kurs nemůže být menší nebo roven nule. Dále funkce zkontroluje, pokud je velikost vkladu záporná nebo nula. Pokud ano, tak funkce vrátí chybu, jelikož velikost vkladu nemůže být záporná. Následovně funkce zkontroluje, jestli počet dní je menší nebo roven nule. Pokud tomu tak je, tak vyskočí hláška, že počet dní nemůže být nulový či záporný. Nakonec přijde na řadu to, jestli uživatel zadal nepovinný argument či ne. Pokud ne, tak proběhne jedna strana funkce, která vypočítá výslednou hodnotu v riziku za pomoci předem stanovené hladiny spolehlivosti, která se rovná 5%. Pokud uživatel zadal vlastní hladinu spolehlivosti, tak funkce vypočítá hodnotu v riziku s danou hodnotou namísto své přednastavené. Jako konečná podmínka výstupu je, pokud je výsledná hodnota kladná či záporná. Pokud je kladná, tak výsledkem je maximální ztráta na investici a pokud je záporná, tak výsledkem je minimální zisk na investici. Tato konečná podmínka platí jak pro výpočet s nepovinným argumentem, tak i za použití přednastaveného argumentu.

Obrázek 5: Návrh funkce pro výpočet Normálního VaR



Zdroj: Autor práce

Po stanovení návrhu kódu pro výpočet VaR můžeme přejít k samotnému kódování. Ve Zdrojovém kódu 5 na řádce 3 můžeme vidět vytvoření samotné funkce *pohledVar*, který počítá hodnotu v riziku u pohledávky a také jsou zde vidět jména argumentů dané funkce. Argument *kurs* očekává *integer* hodnotu. Do argumentu *kurs* uživatel zadá dnešní kurs investice, jejíž soubor dat nahrál ve vstupní fázi programu. Argument *velikostVkladu* udává, kolik peněz hodlá uživatel do dané investice investovat. *velikostVkladu* očekává datový typ *integer*. Je nutno vzít v potaz, že hodnota musí být zadána v měně, do které chce in-

vestovat a konečný výsledek bude vyobrazen v původní měně. Například uživatel nahrál do programu soubor dat změn kursu EUR/CZK, takže do *velikostVkladu* bude zadávat velikost vkladu v EUR a výsledek mu vyjde v CZK. Dalším argumentem je *pocetDni*, který očekává *integer* a udává, jak dlouho chce uživatel investici držet. Posledním argumentem je *alpha*, která udává povolenou chybu v modelu VaR. Příkladem může být investice s trváním 100 dní s povolenou chybou modelu 5%. To znamená, že maximální ztráta je v 5% případů či dnech.

Jak již bylo řečeno, tak tato funce ošetřuje různé chyby vstupu. Na řádku 4-6 je ošetřeno to, aby uživatel nezadal záporný či nulový *kurs*. Na řádku 7-9 je podmínka, že pokud je *velikostVkladu* menší nebo rovna nule, tak funkce vyhodí chybovou hlášku. Na řádku 10-12 je zase ošetřen argument *pocetDni*, kdyby uživatel zadal nulový či záporný trvání investice.

Podmínka na řádce 13 zjistí, pokud uživatel zadal nepovinný argument *alpha*. Pokud ho zadal, tak funkce přeskočí na řádek 33 a pracuje až do konce. Pokud ho nezadal, tak funkce pracuje od řádku 14 až po řádek 30.

Pokud uživatel nezadal nepovinný argument *alpha*, tak do proměnné *output*, řádka 21, funkce uloží vypočtenou hodnotu v riziku podle vzorce 13 pro výpočet hodnoty v riziku za pomocí zadaných argumentů, vypočtené střední hodnoty a směrodatné odchylky a přednastavené hodnoty *alpha*. Následovně proběhne podmínka na řádku 22. Tato podmínka zkontroluje proměnnou *output* a určí, pokud je kladná či záporná. Pokud je kladná, tak funkce vrátí hodnotu v riziku uloženou v proměnné *output* na hladině spolehlivosti 95%, bude danou hodnotu brát jako maximální ztrátu a funkce skončí. Pokud je záporná, tak funkce vrátí hodnotu v riziku *output* a udělá z ní kladné číslo na hladině spolehlivosti 95%, bude tuto hodnotu brát jako minimální zisk a funkce skončí.

Pokud uživatel zadal nepovinný argument *alpha*, tak na řádku 40 proběhne výpočet hodnoty v riziku s argumentem *alpha* a za pomoci ostatních zadaných argumentů a také směrodatné odchylky a střední hodnoty. Tento výpočet je pak uložen do proměnné *output*. Následně, stejně jako když uživatel nezadal nepovinný argument, tak proběhne kontrola proměnné *output* jestli je kladná či záporná. Pokud je kladná, tak funkce *pohledVar* vrátí hodnotu *output* a bude s ní nakládat jako s maximální ztrátou na hladině spolehlivosti $(1 - \alpha) * 100$ % a funkce skončí. Pokud je *output* záporný, tak funkce tuto proměnnou

změní na kladnou hodnotu, bude tuto hodnotu brát jako minimální zisk na hladině spolehlivosti $(1-\alpha)*100$ % a funkce skončí.

Zdrojový kód 10: Funkce pro výpočet Normálního VaR u pohledávky

```
#funkce pro výpočet normálního VaR u pohledávky (velikost vkladu je v původní měně kontraktu = EUR/USD = vklad v EURECH!! - výsledek je v druhé měně = USD)
#vzorec: VaR = v*(A_0-exp(ln(A_0)+Norm.inv(alpha,c*P_1,odmocnina(c)*SD_1)))
pohledVar <- function(kurs, velikostVkladu, pocetDni, alpha){
  if (kurs<=0) {
  return("kurs nemůže být menší nebo roven 0!")
  }
  else if (velikostVkladu <= 0){
    return("Chyba: Nulová či záporná velikost vkladu!")
  }
  else if (pocetDni<=0){
    return("Počet dní nemůže být nulový či záporný!")
  }
  else if (missing(alpha))
  {
    #výpočet a uložení průměru do odhadMu
    odhadMu <- mean(logSloupec$Data, na.rm = FALSE)

    #výpočet a uložení směrodatné odchylky do stDev
    stDev <- sd(logSloupec$Data, na.rm = FALSE)

    output <- velikostVkladu * (kurs - exp(log(kurs) + qnorm(0.05, pocetDni * odhadMu, sqrt(pocetDni) * stDev)))
    if (output>=0) {
      my_list <- list("Maximální ztráta na hladině spolehlivosti alpha %" = 95, " je v cílové měně: " = output)
      return(my_list)
    }
    else
    {
      my_list <- list("Minimální zisk na hladině spolehlivosti alpha %" = 95, " je v cílové měně: " = output)
      return(my_list)
    }
  }
  else
  {
    #výpočet a uložení průměru do odhadMu
    odhadMu <- mean(logSloupec$Data, na.rm = FALSE)
```

```
36
37      #výpočet a uložení směrodatné odchylky do stDev
38      stDev <- sd(logSloupec$Data, na.rm = FALSE)
39
40      output <- velikostVkladu * (kurs - exp(log(kurs) + qnorm(alpha,
        ↪  pocetDni * odhadMu, sqrt(pocetDni) * stDev)))
41      if (output>=0) {
42        my_list <-
          ↪  list("Maximální ztráta na hladině spolehlivosti alpha %"
          ↪  = (1-alpha)*100, "je v cílové měně: " = output)
43        return(my_list)
44      }
45      else
46      {
47        my_list <-
          ↪  list("Minimální zisk na hladině spolehlivosti alpha %" =
          ↪  (1-alpha)*100, "je v cílové měně: " = -output)
48        return(my_list)
49      }
50    }
51  }
```

Zdroj: Autor práce

Zdrojový kód 11 je funkce pro výpočet Normálního VaR u závazku a je velice podobný zdrojovému kódu pro výpočet pohledávky, co se týče struktury kódu. Jediný rozdíl je ve výpočtu samotné hodnoty v riziku a ve jménu funkce, jinak je struktura kódu stejná.

Zdrojový kód 11: Funkce pro výpočet Normálního VaR u závazku

```
1   #funkce pro výpočet normálního VaR u závazku (velikost vkladu je v
    ↪  původní měně kontraktu = EUR/USD = vklad v EURECH!! - výsledek
    ↪  je v druhé měně = USD)
2   #vzorec: VaR =
    ↪  v*(A_0-exp(ln(A_0)+Norm.inv(alpha,c*P_1,odmocnina(c)*SD_1)))
    ↪  )))
3   zavazekVar <- function(kurs,velikostVkladu,pocetDni, alpha){
4     if (kurs<=0) {
5       return("kurs nemůže být menší nebo roven 0!")
6     }
7     else if (velikostVkladu <= 0){
8       return("Chyba: Nulová či záporná velikost vkladu!")
9     }
10    else if (pocetDni<=0){
11      return("Počet dní nemůže být nulový či záporný!")
12    }
13    else if (missing(alpha))
```

```
{
  #výpočet a uložení průměru do odhadMu
  odhadMu <- mean(logSloupec$Data, na.rm = FALSE)

  #výpočet a uložení směrodatné odchylky do stDev
  stDev <- sd(logSloupec$Data, na.rm = FALSE)

  output <- -velikostVkladu * (kurs - exp(log(kurs) +
  ↪  qnorm(1-0.05, pocetDni * odhadMu, sqrt(pocetDni) * stDev)))
  if (output>=0) {
    my_list <-
    ↪  list("Maximální ztráta na hladině spolehlivosti alpha %"
    ↪  = 95, " je v cílové měně: " = output)
    return(my_list)
  }
  else
  {
    my_list <-
    ↪  list("Minimální zisk na hladině spolehlivosti alpha %" =
    ↪  95, " je v cílové měně: " = -output)
    return(my_list)
  }
}
else
{
  #výpočet a uložení průměru do odhadMu
  odhadMu <- mean(logSloupec$Data, na.rm = FALSE)

  #výpočet a uložení směrodatné odchylky do stDev
  stDev <- sd(logSloupec$Data, na.rm = FALSE)

  output <- -velikostVkladu * (kurs - exp(log(kurs) +
  ↪  qnorm(1-alpha, pocetDni * odhadMu, sqrt(pocetDni) *
  ↪  stDev)))
  if (output>=0) {
    my_list <-
    ↪  list("Maximální ztráta na hladině spolehlivosti alpha %"
    ↪  = (1-alpha)*100, "je v cílové měně: " = output)
    return(my_list)
  }
  else
  {
    my_list <-
    ↪  list("Minimální zisk na hladině spolehlivosti alpha %" =
    ↪  (1-alpha)*100, "je v cílově měně: " = -output)
    return(my_list)
  }
}
```

```
}
```

Zdroj: Autor práce

Výstupem zdrojového kódu 10 i 11 je číslo, které nám udává, jaká je nejvyšší možná ztráta či minimální zisk v cílové měně pohledávky či zakázky na dané investici za určené období na hladině spolehlivosti *alpha*, která nabývá často hodnoty 95% či 99%.

Příklad výstupu můžeme vidět ve zdrojovém kódu 12. Maximální ztráta na hladině spolehlivosti alpha 95% je v korunách 146.2823.

Zdrojový kód 12: Výstup upravené metody Normálního VaR

```
# zavolání funkce pohledVar(kurs, velikostVkladu, pocetDni),
 ↪  velikost vkladu je v Eurech!
pohledVar(26,400,30)

# Výstup:
`Maximální ztráta na hladině spolehlivosti alpha %`
[1] 95

` je v korunách: `
[1] 146.2823
```

Zdroj: Autor práce

5.3 Historický VaR

Druhou metodou pro výpočet hodnoty v riziku v této práci je Historický VaR. Cílem této metody je stejně jako u předchozí metody číselný výstup, který bude udávat nejvyšší ztrátu pro pohledávku a závazek a pomocí kvantilu z upravených vstupních dat určí výši ztráty v určitých procentech případů. Vstup dat ze souboru pro tuto metodu je již popsán v předešlé sekci. Pro výpočet Historického VaR je zapotřebí souboru dat se samotnými výšemi kursů, rozdíly kursů si tento program umí vypočítat sám z nahraných dat.

5.3.1 Řešení úpravy vstupních dat

Předtím, než program bude pokračovat k samotnému výpočtu hodnoty v riziku, tak si program nejdříve potřebuje upravit data, aby s nimi mohl pracovat dál. Jedná se o zjemnění dat pomocí logaritmického rozdělení a převedení na změny kursů v čase.

Nejdříve si program funkci *pocetDni* na řádku 2 zdrojového kódu 13 s jedním stejnojmenným argumentem, *pocetDni*. Tento argument očekává hodnotu datového typu *integer* a znamená počet dní splatnosti pohledávky či závazku.

Hlavní úskalí problému logaritmického rozdělení spočívalo v přístupu do jednotlivých sloupců a řádků v cyklické funkci. Jak můžeme vidět na řádku 6 zdrojového kódu 13, tak tento cyklus se bude opakovat do té doby, než dosáhne počtu řádků *workSloupec* mínus argument *pocetDni* plus 1. To je zapříčiněno tím, že oproti úpravě dat v upravené metodě Normálního VaR počítáme rozdíly z více než pouze následující jedné hodnoty. To, z kolika hodnot bude funkce rozdíl počítat, záleží na hodnotě argumentu *pocetDni*. Uvnitř cyklu, který bude probíhat po dobu *počet řádků - 1*, se do *workSloupec* postupně přepisují data. Po dokončení cyklu proběhne pojmenování nově vzniklého *data.frame logSloupec* a zveřejnění pro ostatní funkce daného programu.

Zdrojový kód 13: Úprava dat na logaritmické rozdělení

```
1  # funkce pro úpravu dat na logaritmické rozdělení (argument
   ↪  pocetDni = počet dní splatnosti závazku)
2  pocetDni <- function (pocetDni){
3    workSloupec <- data.frame(grafX$Data)
4    workSloupec <- na.omit(workSloupec)
5    logSloupec <- data.frame()
6    for (i in 1:(nrow(workSloupec)-(pocetDni+1))) {
7      logSloupec[i,1] =  log(workSloupec[i,1]) -
       ↪  log(workSloupec[(pocetDni+i),1])
```

```
  }
  names(logSloupec) <- "Data"
  assign("logSloupec", logSloupec, envir = .GlobalEnv)
}
```

Zdroj: Autor práce

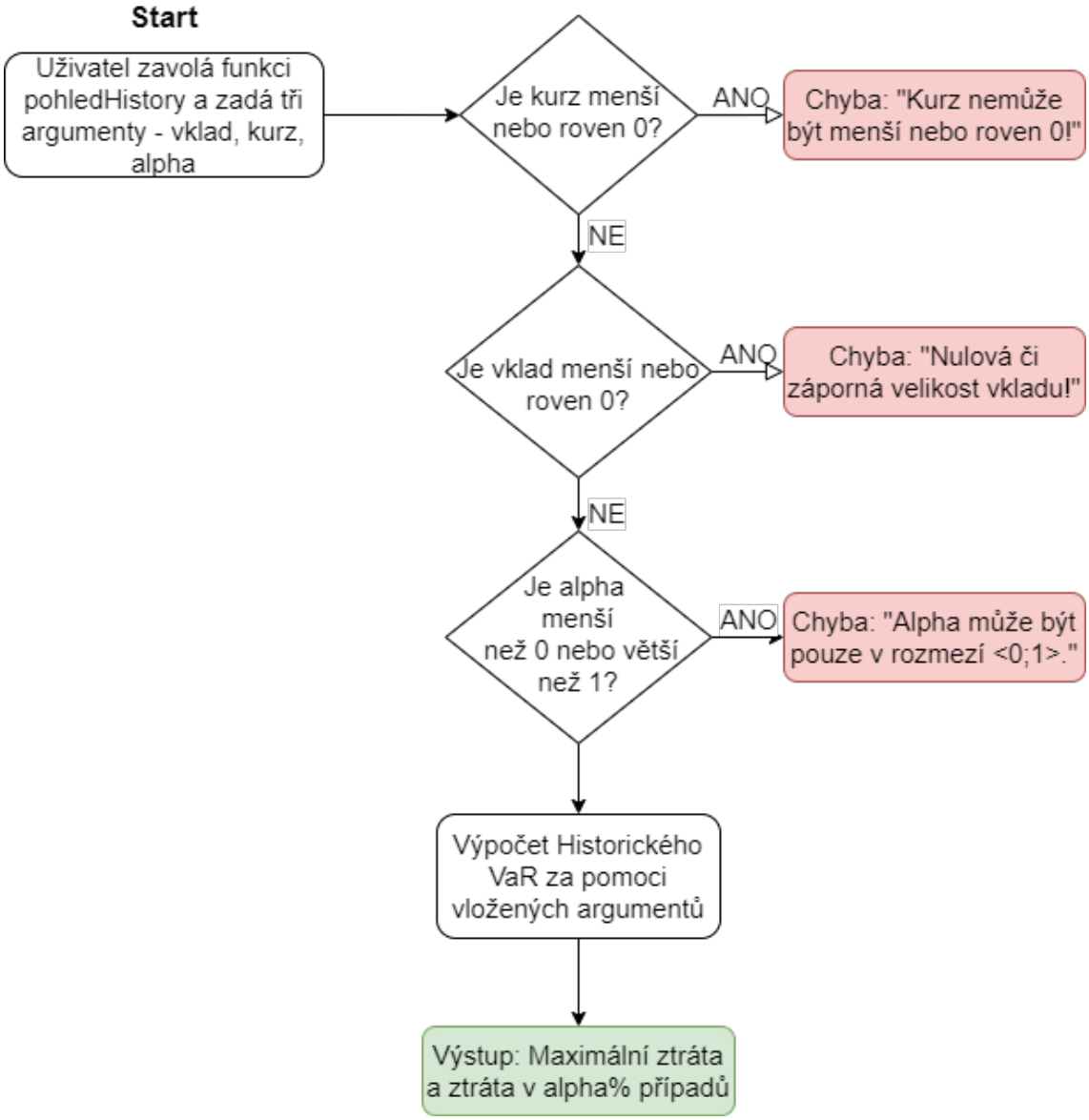
Výstupem této funkce je proměnná *logSloupec*, která bude využita ve funkci pro vykreslení histogramu (*vytvorGrafZmen*) a v samotném výpočtu Historického VaR (*pohledHistory*).

5.3.2 Řešení funkce pro výpočet pohledávky a závazku

Po úpravě dat na logaritmické rozdělení program přejde k samotnému výpočtu hodnoty v riziku. Stejně jako u předchozí metody, Historický VaR pro pohledávku a Historický VaR pro závazek jsou strukturálně velmi podobné, pouze je jiný výpočet samotné hodnoty v riziku a jiná jména funkcí.

Nejdříve jsem si vytvořil návrh stuktury funkce pomocí flowchart diagramu na obrázku 6. Nejdříve si uživatel zavolá funkci *pohledVar* a zadá do ní všechny tři povinné argumenty. Následně funkce zkontroluje, pokud zadal kurs menší nebo roven nule, je-li vklad menší nebo roven nule a pokud je alpha menší než 0 nebo větší než 1. Pokud jedna z těhto podmínek vyjde *TRUE*, tak funkce skončí a vypíše příslušnou chybovou hlášku. Pokud každá z těchto podmínek projde v pořádku a každá z nich vyjde *FALSE*, tak funkce může přejít k samotnému výpočtu hodnoty v riziku za pomoci vložených argumentů. Jako výstupem bude maximální ztráta na investici v cílové měně a ztráta v *alpha* % případů.

Obrázek 6: Návrh funkce pro výpočet Historického VaR



Zdroj: Autor práce

Po vytvoření návrhu jsem se vrhl rovnou na kódování této funkce. Ve zdrojovém kódu 14 na řádce 2 je vytvořena samotná funkce *pohledHistory()* a očekává při svém zavolání 3 povinné argumenty - *vklad, kurs, alpha*. Argument *vklad* očekává číselnou velikost vkladu do pohledávky v původní měně kontraktu. *kurs* je argument, který vyžaduje zadání aktuálního kursu investice, se kterou chce počítat a poslední argument *alpha* očekává číslenou hodnotu mezi nulou a jedničkou a znamená procento případů, pro který chce uživatel zobrazit ztrátu.

Dále probíhají od řádku 3 až po řádek 11 podmínky vstupu, z čehož první podmínkou je, že

kurs musí být vyšší než nula. Pokud není, tak funkce stejně jako v návrhu skončí a vyhodí chybové hlášení, kde nastala chyba. Stejné ošetření vstupu je i pro *vklad*. Na řádce 9 je více podmínek spojené do jedné pomocí *boolean* operátoru „|", který funguje tak, že když alespoň jedna z podmínek ze dvou na každé straně vyjde *TRUE*, tak celá podmínka bude *TRUE*. Když argument *alpha* bude menší než nula nebo bude větší než jedna, tak funkce skončí a vypíše chybovou hlášku. Jelikož *alpha* je procento převedené na reálné číslo, tak může být pouze v rozmezí $< 0; 1 >$.

Pokud všechny argumenty jsou v pořádku a funkce se dostane až na řádek 12, tak funkce přejde rovnou k výpočtu. Nejdříve je potřeba zjistit nejmenší hodnotu v *data.frame logSloupec* na řádce 14, aby mohla funkce určit nevyšší ztrátu a uloží danou hodnotu do proměnné *maxHist*. Na dalším řádku funkce zjistí kvantil *logSloupec* na úrovni argumentu *alpha*. Tato hodnota se uloží do alphaHist a bude sloužit k zjištění ztráty v *alpha*% počtu případů. Po zjištění těchto dvou hodnot funkce přejde k samotnému výpočtu hodnoty na řádkách 17 a 18 v riziku pro maximální ztrátu a ztrátu v *alpha*% případů. Po vypočtení daných hodnot funkce vypíše dané hodnoty. Na řádku 22, funkce *unname()* zde slouží k odstranění pojmenované proměnné, která byla vytvořena pomocí funkce *quantile()*.

Zdrojový kód 14: Řešení výpočtu VaR u pohledávky pomocí Historické metody

```
1  #funkce na výpočet VaR u pohledávky pomocí historického VaR
2  pohledHistory <- function(vklad, kurs, alpha){
3    if (kurs<=0) {
4      return("kurs nemůže být menší nebo roven 0!")
5    }
6    else if (vklad <= 0){
7      return("Chyba: Nulová či záporná velikost vkladu!")
8    }
9    else if (alpha < 0 | alpha > 1){
10     return("Chyba: Alpha může být pouze v rozmezí <0;1>.")
11   }
12   else
13   {
14     maxHist <- min(logSloupec$Data)
15     alphaHist <- quantile(logSloupec$Data, probs = alpha)
16
17     outputMax <- -vklad * (exp(log(kurs) + maxHist) - kurs)
18     outputAlpha <- -vklad * (exp(log(kurs) + alphaHist) - kurs)
19
20     print("V alpha počtu případů je v cílové měně ztráta
21     vyšší než:")
22     print(unname(outputAlpha))
```

```
    print("Nejvyšší ztráta v cílové měně pro tuto pohledávku je:")
    return(outputMax)
  }
}
```

Zdroj: Autor práce

Jak již bylo zmíněno, tak struktura kódu Historického VaR je u pohledávky i u závazku velice podobná, takže není zapotřebí ji blíže popisovat. Zdrojový kód 15 je finální řešení funkce *zavazekHistory()*. Je potřeba dbát na to, že ve zdrojovém kódu 15 je rozdíl, oproti zdrojovému kódu 14 co se týče kódu a ne jmen, pouze na řádkách 14-18, kde probíhá výpočet samotné hodnoty v riziku.

Zdrojový kód 15: Řešení výpočtu VaR u závazku pomocí Historické metody

```
#funkce na výpočet VaR u závazku pomocí historického VaR()
zavazekHistory <- function(vklad, kurs, alpha){
  if (kurs<=0) {
    return("kurs nemůže být menší nebo roven 0!")
  }
  else if (vklad <= 0){
    return("Chyba: Nulová či záporná velikost vkladu!")
  }
  else if (alpha < 0 | alpha > 1){
    return("Chyba: Alpha může být pouze v rozmezí <0;1>.")
  }
  else
  {
    maxHist <- max(logSloupec$Data)
    alphaHist <- quantile(logSloupec$Data, probs = 1-alpha)

    outputMax <- vklad * (exp(log(kurs) + maxHist) - kurs)
    outputAlpha <- vklad * (exp(log(kurs) + alphaHist) - kurs)

    print("V daném počtu případů je v cílové měně ztráta vyšší než:")
    print(unname(outputAlpha))
    print("Nejvyšší ztráta pro tento závazek je v cílové měně:")
    return(outputMax)
  }
}
```

Zdroj: Autor práce

Výstupem Historického VaR jak u pohledávky, tak i u závazku, jsou dvě čísla. První číslo udává, jaká je ztráta v *alpha*% případů u této investice v cílené měně a druhá hodnota udává maximální ztrátu na investici v cílené měně. Příklad výstupu může být zdrojový kód 16,

kde uživatel zadal vstupní data ze souboru o vývoji kursu EUR/USD a zavolal metodu *pohledHistory()* a zadal do ní svůj vklad 400 euro, aktuální kurs eura vůči americkému dolaru a procento případů 0.05, což je rovno 5%.

Zdrojový kód 16: Příklad výstupu funkce Historického VaR

```
#Výpočet VaR pro pohledávku pomocí Historického VaR (argumenty:
↪  výše vkladu v měně kontraktu, dnešní kurs investice, % případů
↪  ztráty v desetinných číslech = např. 0.05)
pohledHistory(400, 1.19, 0.05)

#Výstup:
[1] "V alpha počtu případů je v cílové měně ztráta vyšší než:"
[1] 25.10567
[1] "Nejvyšší ztráta v cílové měně pro tuto pohledávku je:"
[1] 60.11877
```

Zdroj: Autor práce

Závěr

Cílem této bakalářské práce bylo vytvořit kód v programovacím jazyce R pro výpočet hodnoty v riziku pomocí dvou metod, a to pomocí Normálního VaR a Historického VaR.

V teoretické části jsem nejdříve vysvětlil samotný pojem hodnota v riziku, ukázal obecné metody výpočtu této hodnoty a také stanovil rozdělení metod výpočtu hodnoty v riziku. Tyto tři metody byly parametrické, neparametrické a semiparametrické. Dále jsem vyobrazil histogram s normální křivkou pro vysvětlení hodnoty v riziku. Jako další krok jsem stanovil konkrétní případ pro lepší pochopení daného tématu. V další části jsem popsal jednotlivé rozdělení metod výpočtu hodnoty v riziku a stanovil jejich výhody a nevýhody. Z této části vyplývá, že každá metoda je pouze aproximace hodnoty v riziku a že je velmi těžké vybrat metodu, s kterou se bude hodnota v riziku počítat. V další části jsem popsal historii a samotný programovací jazyk R, v kterém jsem v této bakalářské práci pracoval. Také jsem stanovil jeho výhody oproti ostatním programovacím jazykům a proč jsem si vybral právě R pro práci se statistikou a také jsem vytvořil graf popularity vybraných jazyků pro představu, jak se postupem času vyvíjí popularita programovacího jazyku R. Poté jsem přešel k metodice výpočtů v praktické části, kde jsem detailně popsal všechny proměnné a hodnoty, se kterými budu pracovat a také jsem zde napsal jednotlivé vzorce pro výpočet dané problematiky u každé z metod pro každou operaci, pro přehlednost výpočtů v kódech v praktické části.

V praktické části jsem se nejprve zabýval funkcemi, které používají obě metody výpočtů VaR. Zprvu jsem měl problémy s určením těchto funkcí, ale po konzulataci s vedoucím práce a samotném programování jsem na tyto společné funkce přišel. Po stanovení společných funkcí jsem začal s první metodou výpočtu hodnoty v riziku, a to upraveným Normálním VaR. U obou metod byl problém s úpravou dat na požadovanou formu, kde jsem po konzultaci s vedoucím práce dokázal správně naprogramovat kód, který tyto konverze zvládne. Po vyřešení úpravy vstupu jsem poukázal na návrhový diagram funkce a dále jsem přešel k popisu kódu samotného výpočtu hodnoty v riziku za pomocí upravené metody Normálního VaR. Vložil jsem ukázky kódu pro lepší pochopení při popisu kódu jak pro pohledávku, tak i pro závazek. Jako další jsem popsal vývoj kódu metody Historický VaR, nejdříve úpravu vstupních dat a poté samotnou funkci pro výpočet hodnoty v riziku pro pohledávku a závazek. Při testování kódu jsem nalezl jeden nedostatek vstupu, který

potřeboval vylepšit, proto mám v řešení vstupu dvě verze kódu. Po několika testech jsem kód přepopsal třem uživatelům, kteří ho vyzkoušeli a dali feedback na vylepšení. Všichni uživatelé si kód chválili a hodnotili ho kladně.

V této práci jsem se dozvěděl více do hloubky o riziku investic, samotného výpočtu hodnoty v riziku a získal jsem nové znalosti v programovacím jazyce R. Programování v tomto jazyce byla pro mě nová zkušenost a tuto nově nabytou znalost plánuji využít i do budoucnosti.

Výstupem této práce jsou zdrojové kódy, které by mohly sloužit k výpočtu hodnoty v riziku na webových stránkách či webových kalkulačkách. Také je možné tyto kódy implementovat do aplikací, ve kterých může být jako jedna z možností výpočet hodnoty v riziku u vybrané investice.

Mé plány do budoucna jsou rozšířit tyto zdrojové kódy o další metody výpočtu hodnoty v riziku a také možnou implementaci uživatelského rozhraní pro samostatnou aplikaci, ať už mobilní, desktopovou či webovou.

Seznam použité literatury

[1] Cipra, T. (2005). Praktický průvodce finanční a pojistnou matematikou (II.). Ekopress.

[2] Duda, M., & Schmidt, H. (2009). Evaluation of Various Approaches to Value at Risk [Diplomová práce]. Lund University.

[3] Alemany, R., Bolancé, C., & Guillén, M. Nonparametric estimation of Value-at-Risk, 2. http://xreap.cat/Doc-Import/XREAP2012-19.pdf

[4] Abad, P., Benito, S., & López, C. (2014). A comprehensive review of Value at Risk methodologies. The Spanish Review of Financial Economics, 12(1), 15-32. https://doi.org/10.1016/j.srfe.2013.06.001

[5] Mrkvička, T., Krásnická, M., Friebel, L., Volek, T., & Rolínek, L. (2018) - nepublikováno. Backtesting evaluation of Value-at-Risk methods for exchange rates in time horizon up to one year.

[6] Mentel, G. (2013). Parametric or Non-Parametric Estimation of Value-At-Risk. International Journal of Business and Management, 106. https://doi.org/10.5539/ijbm.v8n11p103

[7] NORMINV Function: Calculates the inverse of the normal cumulative distribution for the specified mean and standard deviation. (c2015–2021). CFI. Retrieved March 14, 2021, from https://corporatefinanceinstitute.com/resources/excel/functions/ norminv-function/

[8] TIOBE Index. (c2020). TIOBE. Retrieved April 13, 2021, from https://www.tiobe.com/tiobe-index/

[9] What is R?. (c2021). MRAN. Retrieved March 15, 2021, from https://mran.microsoft.com/document
is-r

[10] What is R?. (2021). R. Retrieved March 16, 2021, from https://www.r-project.org/about.html

[11] Parametric value-at-risk. (c2014). Https://breakingdownfinance.com/. Retrieved April 13, 2021, from https://breakingdownfinance.com/finance-topics/risk-management/market-risk/parametric-value-at-risk/

[12] Normal (Gaussian) Distribution. (2021). Probability Course. Retrieved April 14, 2021, from https://www.probabilitycourse.com/chapter4/4_2_3_normal.php

Abstrakt

This thesis explores multiple methods of calculating currency risk with the use of value at risk, as well as implementing these methods in code language R. The first part of this thesis is theoretical and explains and proves three methods of assessing parametric and non-parametric models of value at risk and shows how precise they are and what disadvantages these models pose. Apart from that, this part of the thesis explores more possibilities of calculating value at risk. The second part of this thesis focuses on implementing these value at risk calculation methods into code language R, resulting in a computer software, which is able to calculate currency risk with these different value at risk methods using user input of needed values.

Keywords: value at risk, currency risk, code language R, software

Seznam zkratek

ČR	Česká republika
JČU	Jihočeská univerzita
VaR	Value at Risk (hodnota v riziku)
EVT	Extreme Value Theory
GARCH	Generalized Autoregressive Conditional Heteroscedasticity
EUR	Euro
CZK	Česká koruna
USD	Americký dolar

Seznam obrázků

Seznam tabulek

Seznam ukázek zdrojových kódů

Přílohy

Příloha 1: CD se zdrojovými kódy práce, intalační program aplikace pro spuštění těchto kódů, data pro vstup pro dané kódy a celý soubor této balakářské práce