Univerzita Palackého v Olomouci
Přírodovědecká fakulta
Katedra geoinformatiky

ROZŠÍŘENÍ MOŽNOSTÍ BALÍČKU UNCERTAINTY INTERPOLATION

Magisterská práce

Bc. Tomáš BURIAN

Vedoucí práce: doc. Mgr. Jiří Dvorský, Ph.D

Konzultant: Mgr. Jan Caha, Ph.D

Olomouc 2015
Geoinformatika

ANOTACE

Tato diplomová práce se zabývá úpravou a funkčním rozšířením balíčku Uncertainty Interpolation pro software R, a to především o možnou parametrizaci funkcí a převedením formátu ze starších datových struktur S3 do verze S4. Cílem samotné práce je poté zpřístupnit praktické možnosti případnému uživateli a poukázat na vliv nejistoty na data (ale i na postupy všeobecně) za pomoci fuzzy teorie. Tohoto cíle je dosaženo prostřednictvím vlastních funkcí, které nabízí rozšíření zvolených dat o modely nejistoty a mimo jiné poskytují i základní metody pro interpolace. Veškeré funkční algoritmy a procesy byly vytvořeny a testovány v rámci softwaru The R Project for Statistical Computing. Výsledný balíček, jakožto nástroj pro R, pak obsahuje několik funkcí určených pro definice tříd objektů datových struktur S4, vstupní data, modely nejistoty, interpolační procesy, tvorbu gridu, transformace dat, odhad variogramu a vizualizace. Výsledek práce tedy tvoří nová verze původního balíčku UncerIn, který je řádně okomentovaný a připravený k volnému využití.

KLÍČOVÁ SLOVA

diplomová práce, nejistota, interpolace, S4, R balíček

Počet stran práce: 51
Počet příloh: 1

ANOTATION

The diploma thesis deals with modifications and functional extensions of the Uncertainty Interpolation program package for the R software, particularly with the possible parametrization of its features and format conversion of the old S3 structures to S4 version. The goal of the thesis is to enable a user to utilize all functions of the program and to point out the influence of the uncertainty on data using the fuzzy theory. This goal is achieved through the features that allow extending the selected data with several uncertainty models; in addition to that, these features offer some basic interpolation methods as well. All of the operational algorithms and processes were created and tested in The R Project for Statistical Computing software. The final package as an R tool includes a number of features designated for definitions of object classes of the S4 data structures, input data, uncertainty model, interpolation processes, grid-making, data transformation, variogram approximation, and visualization. The result of the thesis is therefore a new version of the original UncerIn package that is annotated properly and ready-to-use.

KEYWORDS

diploma thesis, uncertainty, interpolation, S4, R package

Number of pages: 51
Number of appendixes: 1

Prohlašuji, že

- diplomovou práci včetně příloh, jsem vypracoval samostatně a uvedl jsem všechny použité podklady a literaturu. Ve své diplomové práci jsem využil software The R Project for Statistical Computing a jeho doplňky, vše dostupné pod licencí GNU GPL.

- jsem si vědom, že na moji diplomovou práci se plně vztahuje zákon č.121/2000 Sb. - autorský zákon, zejména § 35 – využití díla v rámci občanských a náboženských obřadů, v rámci školních představení a využití díla školního a § 60 – školní dílo,

- beru na vědomí, že Univerzita Palackého v Olomouci (dále UP Olomouc) má právo nevýdělečně, ke své vnitřní potřebě, diplomovou práci užívat (§ 35 odst. 3),

- souhlasím, aby jeden výtisk diplomové práce byl uložen v Knihovně UP k prezenčnímu nahlédnutí,

- souhlasím, že údaje o mé diplomové práci budou zveřejněny ve Studijním informačním systému UP,

- v případě zájmu UP Olomouc uzavřu licenční smlouvu s oprávněním užít výsledky a výstupy mé diplomové práce v rozsahu § 12 odst. 4 autorského zákona,

- použít výsledky a výstupy mé diplomové práce nebo poskytnout licenci k jejímu využití mohu jen se souhlasem UP Olomouc, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly UP Olomouc na vytvoření díla vynaloženy (až do jejich skutečné výše).

V Olomouci dne Bc. Tomáš BURIAN

Děkuji vedoucímu práce doc. Mgr. Jiřímu Dvorskému, Ph.D za podněty, připomínky a čas při vypracování práce. Dále děkuji Janu Cahovi za výbornou spolupráci a obětovaný čas.

Dále děkuji své rodině za poskytnuté prostředky a prostředí.

Obsah

ÚVOD

V roce 2013 byla obhájena bakalářská práce na téma Rozšíření interpolačních nástrojů v R project o modely nejistot (Burian, 2013), přičemž cílem bylo prozkoumat praktické možnosti softwaru R v oblasti propojení modelů nejistot a interpolací. Tato hypotéza byla velmi kladně potvrzena a tím se i otevřely nové možnosti pro další výzkum. Z těchto důvodů se připustilo rozšíření celého tématu, které zjevně expanzi nabízí, a to především přepisem do datové struktury formátu *S4* a parametrizaci využitých interpolačních funkcí. Jednotlivé procesy je zapotřebí dopracovat do ideální fáze, kdy výsledný balíček pro R bude pracovat podle režie uživatele a nebude jej příliš omezovat. Nejistotu je totiž nutné vnímat jako jev s možným dynamickým průběhem, a proto je i na místě, aby dostupné funkce nebyly omezené (bez možnosti manipulace s parametry).

Problematiku programování v R vystihuje velmi dobře prof. Edzer Pebesma z německého institutu pro geoinformatiku v Münsteru, který také značně přispěl svými poznatky do vývoje této práce. Navíc, je-li to zapotřebí, je kdykoli k dispozici a z jeho příspěvků je denně čerpáno. Na druhé straně stojí například pan dr. Albert Gáspár, z katedry kartografie a geoinformatiky univerzity v Budapešti, který se zajímá spíše o vliv modelů nejistoty a průběhu výpočtů na výsledné 3D modely. Samotnou propagací nejistoty se pak zabývá pan doktor Jan Caha, který ve svých publikacích poukazuje nakolik je propagace a modelování nejistoty důležité a potřebné pro aplikace v GIS.

Právě propojení myšlenek a záměrů výše zmíněných odborníků je klíčovým cílem této diplomové práce. Celá problematika se čím dál více dostává do všeobecného podvědomí a je důležité ji neustále propagovat nejen v teoretickém, ale i praktickém úhlu pohledu. Novodobého trendu potřeby získat co možno nejkvalitnějších výsledků není možné dosáhnout bez kritického pohledu na svět, proto je na místě, aby procesy šetření a propagace nejistoty pokračovaly i nadále.

V celkovém výčtu je na zahraničním poli tato tematika hodnocena (z vlastní zkušenosti) velmi pozitivně, čímž byl i význam této diplomové práce obhájen a potvrzen. Pro úplnost sumarizace se pak na domácí půdě, tedy v České republice, tímto směrem zabývá Jan Caha, s nímž byla založena aktivní spolupráce a figuruje také jako konzultant celé práce.

1 CÍLE PRÁCE

Cílem diplomové práce je rozšíření funkčnosti balíčku Uncertainty Interpolation pro software R zejména o parametrizaci interpolačních procesů. V textové části pak budou popsány nejdůležitější parametry jednotlivých interpolačních metod a jejich vliv na interpolaci s daty obsahující nejistotu. Mimo to bude v teoretické části také nastíněna problematika datových struktur *S4* v rámci R software.

Smyslem praktické části je úprava a funkční rozšíření balíčku Uncertainty Interpolation pro software R (Burian, 2013). První kroky práce zasáhnou syntaxi původního kódu, který bude sofistikovaně upraven a náležitě okomentován. Zároveň bude vytvořena anglická verze balíku tak, aby bylo možné práci publikovat na uložišti CRAN (celosvětový archiv balíčků pro R software). Dále bude proveden přepis starších funkcí (datové struktury formátu *S3*) do podoby datových struktur o formátu *S4*. V návaznosti na vývoj funkčnosti se vyhodnotí vliv parametrů interpolačních metod a na základě tohoto průzkumu pak bude provedena parametrizace zvolených parametrů interpolačních funkcí. Výsledný balíček bude volně dostupný. Veškeré poznatky budou řádně zaznamenané v textové části, která bude zpracována podle zásad šablony dostupné na webových stránkách katedry v typografickém softwaru Tex. Bude zde popsána problematika parametrizace interpolačních funkcí a datové struktury *S4*.

Kompletní diplomová práce bude odevzdána v řádném termínu. Bude vytvořena i její digitální verze odevzdaná na CD a bude o ni vytvořena webová stránka v souladu s pravidly dostupnými na stránkách katedry.

2 POUŽITÉ METODY A POSTUPY ZPRACOVÁNÍ

2.1 Použitá data

Stejně jako tomu bylo u bakalářské práce, na kterou tato práce diplomová myšlenkově navazuje, ani zde nejsou jakákoli reálná data potřebná. Cílem je vyvinutí funkčního rozšíření celého aparátu a pro ověření správnosti procesů nejsou data nutností. Tvrzení potvrzuje i prohlášení, že vyhovujících datových sad není mnoho nebo je obtížné jich dosáhnout (Burian, 2013).

Možná elementární data pro vstup pak tedy může zajišťovat sada funkcí uvnitř balíčku. Přičemž byl tento proces upraven tak, aby pro případného uživatele byla tato cesta co možno nejjednodušší. K dosažení funkčních dat tak postačí pouze označit pár vstupních veličin. Nabízí se zde proto varianta využití volně dostupných datasetů (například *meuse*). Touto cestou je tak i zpřístupněna cesta exemplárního užití balíčku pro běžného uživatele.

2.2 Použité programy

2.2.1 The R Project for Statistical Computing

Klíčový nástroj pro tvorbu diplomové práce bude představovat program The R Project for Statistical Computing (ve zkratce pouze R). Jedná se o volně šiřitelný software, který se uplatňuje především na poli statistických výpočtů a vizualizací (R Core Team, 2014). Podrobný rozbor produktu popisuje ve své práci například Burian (2013), nebo Hornik (2014).

Obrázek 1: Grafické logo programu The R Project for Statistical Computing (převzato z: R Core Team (2014))

2.2.2 RStudio

RStudio je integrované vývojové prostředí (IDE) pro výše zmíněný software. Uživateli přináší především příjemnější (user-friendly) přístup k jádrovému řešení a na internetu je k dostání ve dvou variantách. První možností je bezplatná open source edice, druhou

pak placená komerční licence za bezmála tisíc dolarů ročně. Rozdíl mezi nimi tvoří především uživatelská online podpora, kde placená verze nabízí dokonce garanci osmi hodinové pohotovosti během pracovního období. Samozřejmostí jsou pak také jiné licenční podmínky, přičemž komerční varianta disponuje svou vlastní definicí licence (oproti bezplatné licenci AGPL v3). Oba zmíněné typy integrovaného vývojového prostředí však disponují téměř shodnými možnostmi pro uživatele v rámci R a pro vývoj diplomové práce plně postačuje open source edice (Rst, 2014).

Obrázek 2: Grafické logo integrovaného vývojového prostředí RStudio (převzato z: Rst (2014))

2.2.3 RTools

Nástroj RTools představuje funkční extenzi pro tvorbu, správu a vývoj balíčků v rámci R, která se aktuálně nachází ve verzi 3.3.0.1958 (R Core Team, 2014). Do prostředí RStudia přidává novou kartu Build pro obsluhu při manipulaci s vyvíjenými balíčky. Nedílnou součást však tvoří balíček roxygen2, který se zasluhuje především o tvorbu dokumentace výsledného balíčku (Hadley Wickham, 2015).

2.3 Postup zpracování

Počátek práce byl zahájen potřebnou četbou odborné literatury a studiem pokročilého programování v jazyku R, a to především proto, že datové struktury formátu *S4* se diametrálně liší od svého předchůdce generace *S3*. V rámci vývoje balíčku byla dokonce podniknuta výzkumná stáž v německém městě Münster (duben 2014). Tamní institut geoinformatiky patří, v oblasti programování a využívání softwaru The R Project for Statistical Computing, k těm nejlepším na světě. Klíčovou roli zde přitom sehrává prof. Edzer Pebesma (viz obr. 3), který jakožto vedoucí zdejší katedry a uznávaný odborník v oboru, představuje i tvůrce světoznámých balíčků pro R software (například balíček gstat). Právě na tomto místě byly položeny praktické základy tvorby datových struktur *S4* a vytvořen znatelný pokrok v diplomové práci.

Obrázek 3: Exemplární fotografie z přednášky o programování v R, na snímku prof. Edzer Pebesma

Získání dostatečných znalostí v oblasti problematiky *S4* umožnilo přistoupit k samotnému řešení praktické části diplomové práce. Nejdůležitějším krokem byl přepis starších funkcí a jejich funkční rozšíření. Postupný vývoj balíčku byl přibližně stejný jako u bakalářské práce, přičemž na počátku byl řešen stěžejní problém vstupu dat a definice tříd pro data. Funkce pro definici primárních dat byla znatelně pozměněna a pro její chod nyní postačuje pouze označení veličin (údajů) pro souřadnice x, y a zkoumanou proměnnou z, která posléze vstupuje do procesů. Stanovení jednotlivých tříd objektů *S4* pak zajišťuje formát a správnost dat mezi jednotlivými funkcemi balíčku. Celkem bylo vytvořeno pět tříd objektů:

1. Points: třída pro vstupní data bez jakékoli úpravy,
2. UncertainPoints: data obohacená o určitý model nejistoty,
3. UncertainInterpolation: výsledná data po průběhu interpolace,
4. FuzzyInterpolation: výsledná data po průběhu fuzzy interpolace,
5. Variogram: pro výstup z funkce pro odhad variogramu.

Parametrizace a funkční rozšíření balíčku provázelo průběh práce celou dobu. Ať už to byla variabilita vstupních dat, nebo možnost výběru argumentů funkce. Pozornosti neunikl ani kód pro generování gridu, jakožto vstupní veličina do interpolací. Koncovému uživateli se nyní dostává možnosti vytvořit až šesti různých gridů, přičemž důležitým mezníkem je zde výběr, zda si přeje či nepřeje mít u výsledného gridu

definované souřadnice. Samotné interpolace (idw, spline, kriging) pak byly parametrizovány podle možností klíčových funkcí zajišťující vlastní výpočet. Veškeré argumenty byly rozepsány a byly jim stanoveny základní (výchozí) hodnoty pro ukázkový chod balíčku. Záleží tedy pouze na koncovém uživateli, jakým způsobem si bude přát tyto hodnoty přepisovat na vstupu funkce a modifikovat tak průběh i výsledek. Stejně tak bylo přistoupeno i k rozšíření pro výpočet variogramu. Jediný rozdíl zde tvoří výstupní formát dat, který tvoří standardní *S4* objekt, avšak uvnitř se skrývají další tři podobné objekty definující hodnoty variogramu. Výjimku pak tvoří skupina funkcí pro tvorbu modelů nejistot. Tyto funkce byly již dříve parametrizovány a nyní postačil pouze jejich přepis do nové podoby uvnitř balíčku.

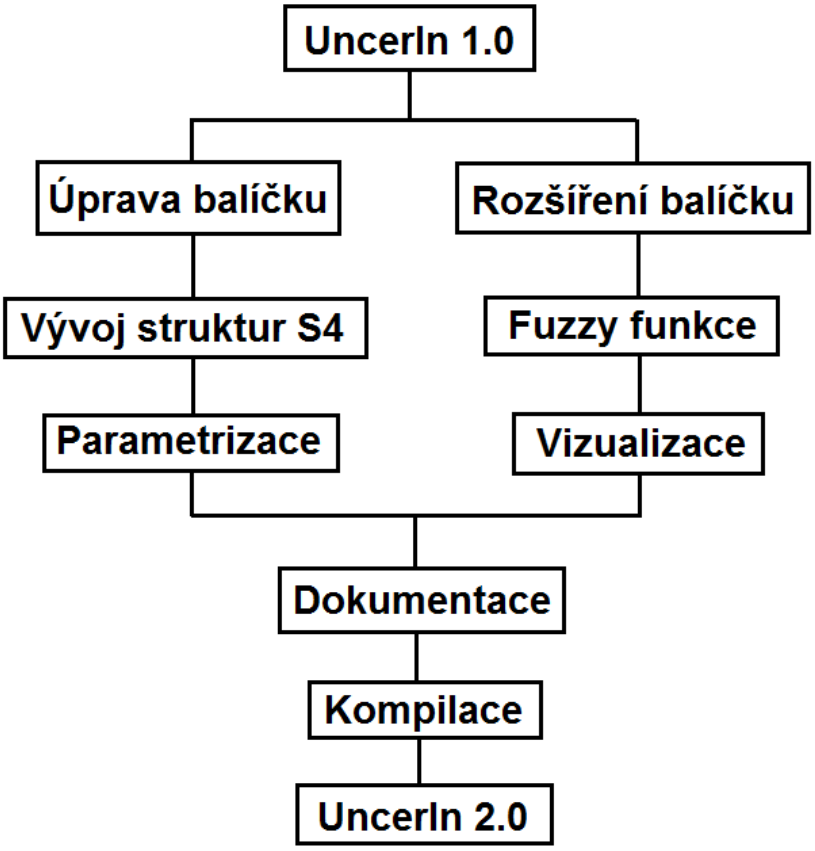
Zajímavostí balíčku pak můžou být transformační funkce *as.data.frame* a *as.UncertainPoints*, které zajišťují převod dat do požadovaných formátů. V prvním případě se jedná o skrytý proces uvnitř interpolací tak, aby nebyl finální kód příliš složitý a také aby byl zajištěn hladký průchod výpočtů bez nutnosti zásahu do datových struktur ze strany uživatele. Druhá zmíněná funkce přináší nástroj pro konverzi mezi třídami *UncertainInterpolation* a *UncertainPoints*. Jedná se opět pouze o zvýšení komfortu uživatele v případě, kdy by byl žádoucí zpětný převod třídy objektů.

Závěr vývojového procesu byl věnován tvorbě funkcí pro vizualizace dat. Rozšířila se tak funkční část balíčku a uživatel si může zobrazit jednoduché výstupy. Úplný konec a definitivní podobu výsledku završila řádná dokumentace funkcí a poté i jejich kompilace do výsledného balíčku pro software R. Kompletní nový balíček UncerIn 2.0 (Uncertainty Interpolation) se tak nově skládá z následujících okruhů funkcí pro:

- definice tříd objektů datových struktur *S4*,
- vstupní data,
- modely nejistoty,
- interpolační procesy,
- tvorbu gridu,
- transformace dat,
- odhad variogramu,
- vizualizace.

2.3.1 Schéma postupu práce

Pro zvýšení představivosti a názornosti postupu práce, bylo vytvořeno i jednoduché schéma průběhu (viz obr. 4). Na počátku tedy figuroval původní balíček UncerIn 1.0 a jednotlivými kroky úprav či rozšíření byl postupně přetvořen do podoby verze 2.0.



Obrázek 4: Jednoduché schéma postupu práce

3 TEORIE NEJISTOTY

Pochopení a především porozumění nejistoty je velmi důležité. Svoboda (2014) uvádí amerického matematika Norberta Wienera jako autora citátu „Bez výjimky každá informace dochází k nám nějak zdeformovaná a zubožená.“ Přidáme-li k tomuto tvrzení myšlenku, že informace všeobecně jsou v dnešním světě základem a jejich interpretace je klíčová, získáváme základní indicie pro řešení fenoménu nazývaného jako nejistota. Tato problematika bývá často přehlížena, a to i přestože nás neustále obklopuje (Burian, 2013). V návaznosti přichází Caha (2014) s informací, že nejistota byla ve vědeckém světě přítomna vždy, a právě proto by přehlížena být neměla. V otázce digitálního světa se pak můžeme s nejistotou setkat prakticky kdekoli, protože všechna digitální data v sobě nesou určitou chybu (Fisher a Tate, 2006), mnoho z nich také není absolutně přesná a čím detailněji budeme reálný svět pozorovat, tím více nejasné se nám bude jevit jeho řešení. Všeobecně pak lze tvrdit, že se s nejistotou setkáme vždy v případech, kdy se do dané problematiky zahledíme dostatečně hluboko (Caha, 2015; Zadeh, 1973).

Výbornou ukázku přítomnosti nejistoty v našem světě popisuje Caha (2015) v jeho knize, kde demonstruje tento jev na definici světoznámého Ludolfova čísla, neboli π. Je všeobecně známo, že hodnota π nebyla nikdy přesně určena a tudíž ji nelze považovat za absolutně přesnou. Konstanta byla podle Arndt a Haenel (2006) definována jako rozpětí dané vztahem mezi dvěma zlomky:

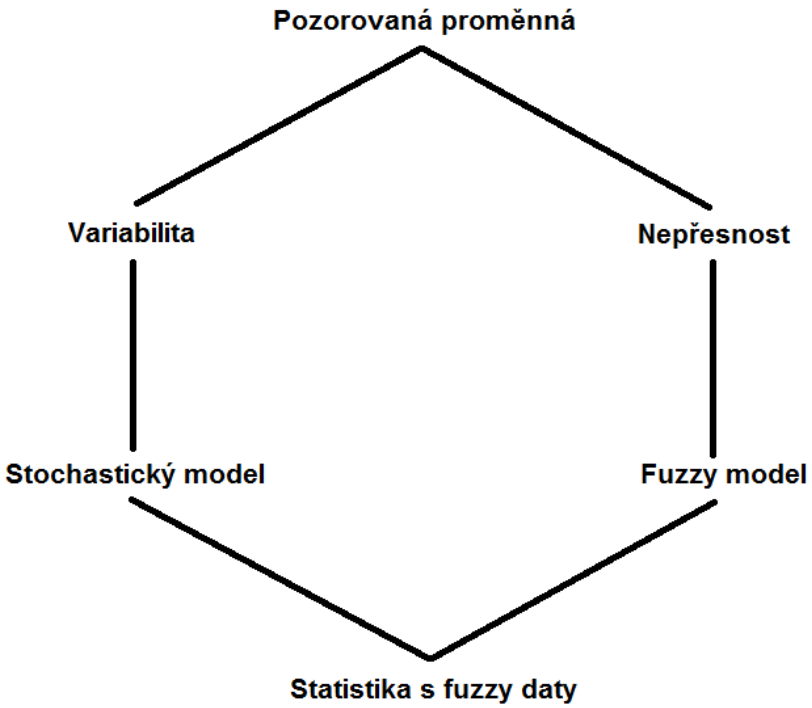
$$\frac{223}{71} < \pi < \frac{22}{7} \,. \tag{1}$$

Nejistota tedy ve vstupních hodnotách, nebo také v datech obecně, byla očividně zakomponována již dávno. Důležitý je především přístup k dané problematice a kritické myšlení, přičemž je vždy nutné mít na paměti právě přítomnost nejistoty. Nedílnou součástí by pak mohlo být i modelování situace s ohledem na vliv nejistoty v datech a na propagaci nejistoty skrze další analýzy, která se poprvé na scéně geoinformatiky objevila zhruba na konci 80. let 20. století (Caha, 2015).

3.1 Skladba nejistoty

Nejistota se sestává z několika více či méně známých dílčích součástí. Existuje několik různých zdrojů, které popisují právě různý počet těchto složek. Caha (2015) poznamenává, že některé zdroje uvádí až osm komponent. Publikační autoři Viertl (2011), Celikyilmaz a Turksen (2009) a Klir a Yuan (1995) se však shodují na faktu, že existují především dva hlavní typy nejistoty založené na variabilitě a nepřesnosti (Caha, 2015). Viertl (2011) pak vyjadřuje tuto problematiku pomocí jednoduchého

grafického obrazce (viz obr. 5). Jednotlivé složky a modely předlohy popisuje ve své práci Burian (2013).

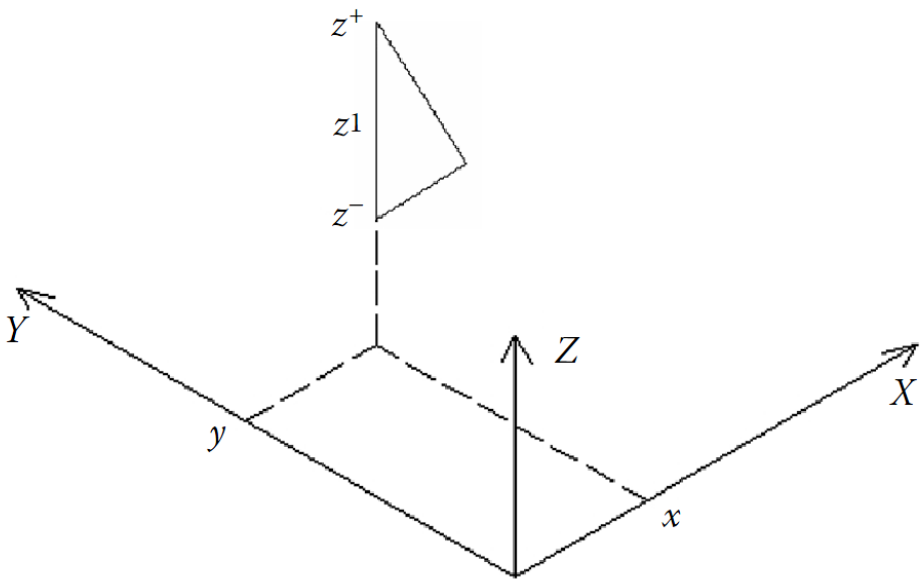


Obrázek 5: Graficky vyjádřené komponenty nejistoty (upraveno dle: Viertl (2011))

Viertl (2011) tedy zavedl dva elementární přístupy k řešení propagace nejistoty, a to cestou statistického zkoumání nebo skrze fuzzy teorii. Základním kamenem však bude vždy zkoumaná proměnná, u které se připouští právě dvě důležité vlastnosti tvořící základy nejistoty - variabilita a nepřesnost (Caha, 2015). Samotná variabilita vstupních dat pak nahrává statistice a jejím metodám, přičemž tato cesta byla již dříve více či méně prozkoumána a dnes je využívána poměrně často. Velmi známou metodou je zde například metoda Monte Carlo, kterou ve své publikaci uvádí Helton a Oberkampf (2004). Na druhé straně nepřesnost dává prostor fuzzy teorii, která je oproti statistickému řešení relativně novým přístupem. Není proto divu, že v porovnání se statistickým přístupem je řešena podstatně méně, a i proto je na ni zaměřena tato diplomová práce. Přičemž fuzzy teorie a metody ji blízké jsou vhodné pro řešení nejistoty založené na nepřesnosti (Zadeh, 1995; Caha, 2015).

Nepřesnost dat může být způsobena mnoha faktory. Příkladem zde může být špatně zvolená metoda sběru dat, vada měřící aparatury, či selhání lidského faktoru a podobně (Fisher a Tate, 2006). Jestliže tedy připustíme fakt, že vstupní data jsou nepřesná, pak můžeme převést tato data na fuzzy hodnoty, jakožto nejjednodušší fuzzy objekty (Burian, 2013). Veškerá absolutní data budou poté převedena na fuzzy intervaly (viz obr. 6) a jejich matematická reprezentace bude vyjádřena jako interval

$A = [z^-, z^0, z^+]$ (Waelder, 2007), který tvoří základní předpoklad pro tvorbu fuzzy povrchu a právě touto myšlenkou se budeme dále zabývat (viz níže).



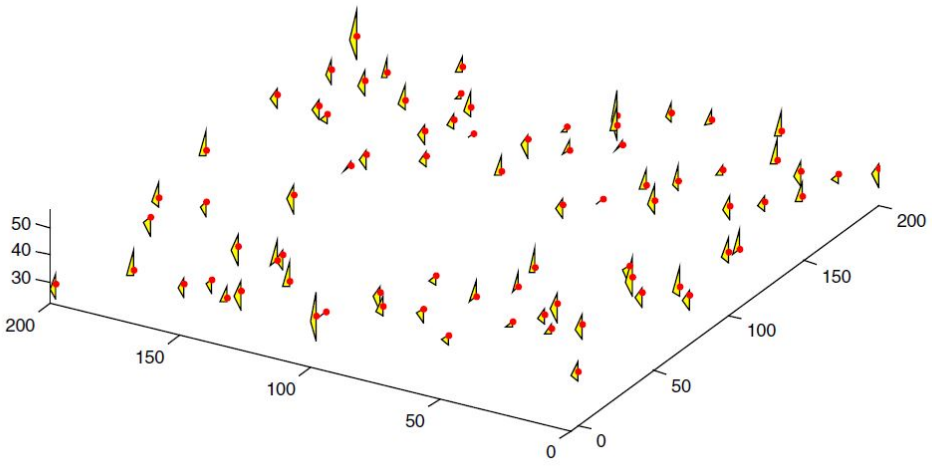
Obrázek 6: Grafické vyjádření fuzzy intervalu z původní nepřesně zaměřené hodnoty (převzato z: Santos (2008))

3.2 Fuzzy povrch

Případné řešení vyjádření nejistoty je tedy mimo jiné přístupné skrze fuzzy povrchy (Caha, 2014). Samotný fuzzy povrch může být definován jako běžná plocha, která však ke svému vyjádření, namísto absolutních hodnot, využívá fuzzy hodnoty. Každý takový povrch pak zahrnuje nejistotu, jelikož jakožto každý bod celku uvažuje určitý rozsah hodnot, nebo-li fuzzy interval (Caha et al., 2012). V celkovém souhrnu pak existují dva základní přístupy ke tvorbě fuzzy povrchů (Caha et al., 2014):

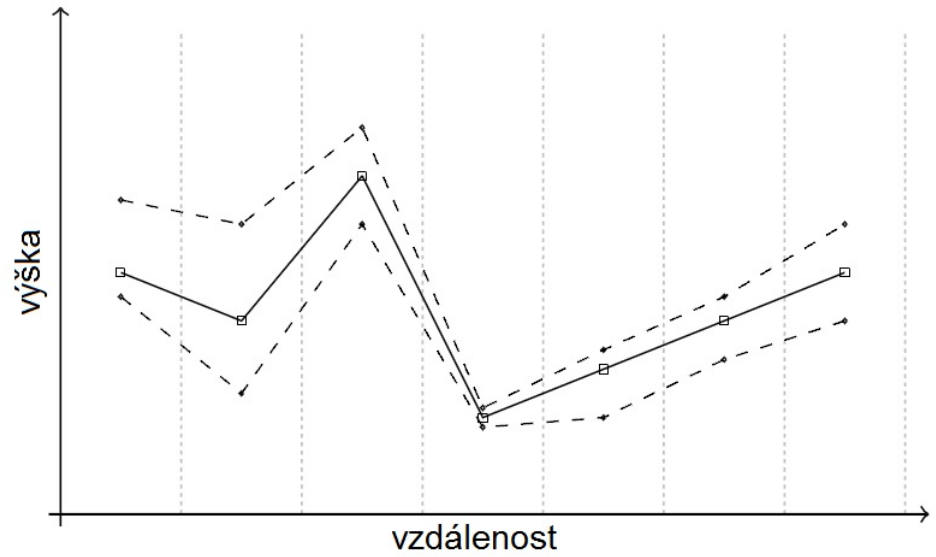
1. metoda vycházející z nejistých dat (specifikované jako fuzzy čísla), rozšiřující interpolační procesy na základě principu o rozšíření,
2. alternativní, mnohem častější metoda používající ostrá data, která specifikuje parametry pro interpolace jako fuzzy čísla, kde výsledkem jsou poté také fuzzy čísla.

Rozdíl mezi obyčejným a fuzzy povrchem se tedy jednoduše skrývá v definici zkoumané proměnné z, která je standardně doplněna o známé souřadnice x, y. Proměnná z zde však vystupuje jako fuzzy číslo $A = [z^-, z^0, z^+]$ (Waelder, 2007; Burian, 2013). Toto vymezení patřičným intervalem pak vystihuje možné původní hodnoty z v dané oblasti (viz obr. 7).



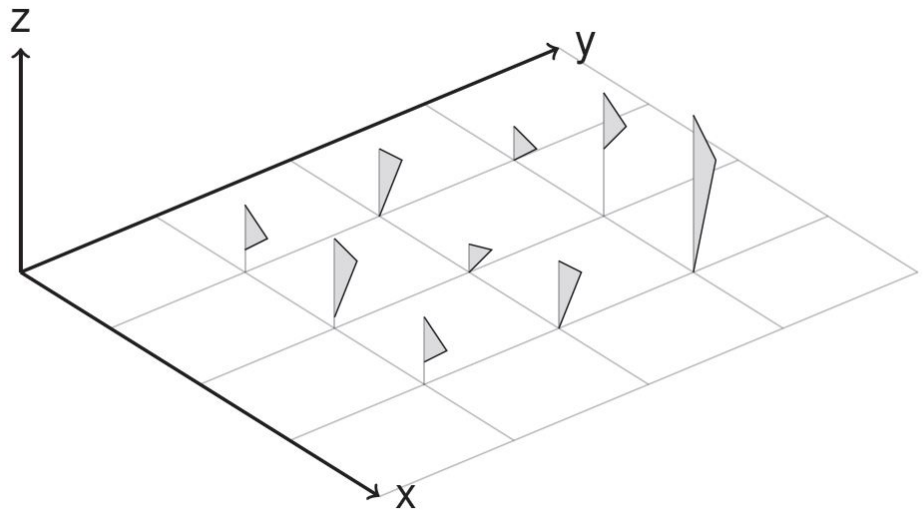
Obrázek 7: Fuzzy data, tečky představují původní hodnoty zkoumané proměnné (převzato z: Santos et al. (2002))

Při výpočtech tedy pro každý bod uvažujeme 3 různé hodnoty proměnné. Reprezentace se skládá z dolní hranice z^-, střední hodnoty z^0, která obvykle odpovídá nefuzzifikovanému řešení (původní hodnotě bez nejistoty) a horní hranice z^+ (Caha, 2015). Z tohoto důvodu lze také očekávat jiný přístup ke kalkulacím a formě výsledků. V současnosti neexistuje algoritmus pro zpracování vizualizace dat takové definice, proto v případě fuzzy povrchů hovoříme o celkem třech odlišných površích (adekvátní pro hodnoty z^-, z^0 a z^+). Ukázku možného profilu (viz obr. 8) fuzzy povrchu publikoval Caha (2014).



Obrázek 8: Profil fuzzy povrchu. Plná linie vyjadřuje z^0 a čárkované linie představují z^-, z^+. (upraveno dle: Caha (2014))

Teoreticky nejblíže, v oblasti vizualizace fuzzy povrchu, se dostal ve své práci Santos et al. (2002), který dokázal dostat fuzzy hodnoty do 3D projekce. Nicméně toto řešení je přípustné pouze pro povrch menšího rozsahu (viz obr. 9). Postupné zobrazování většího areálu rapidně snižuje čitelnost výsledku (Caha, 2015).



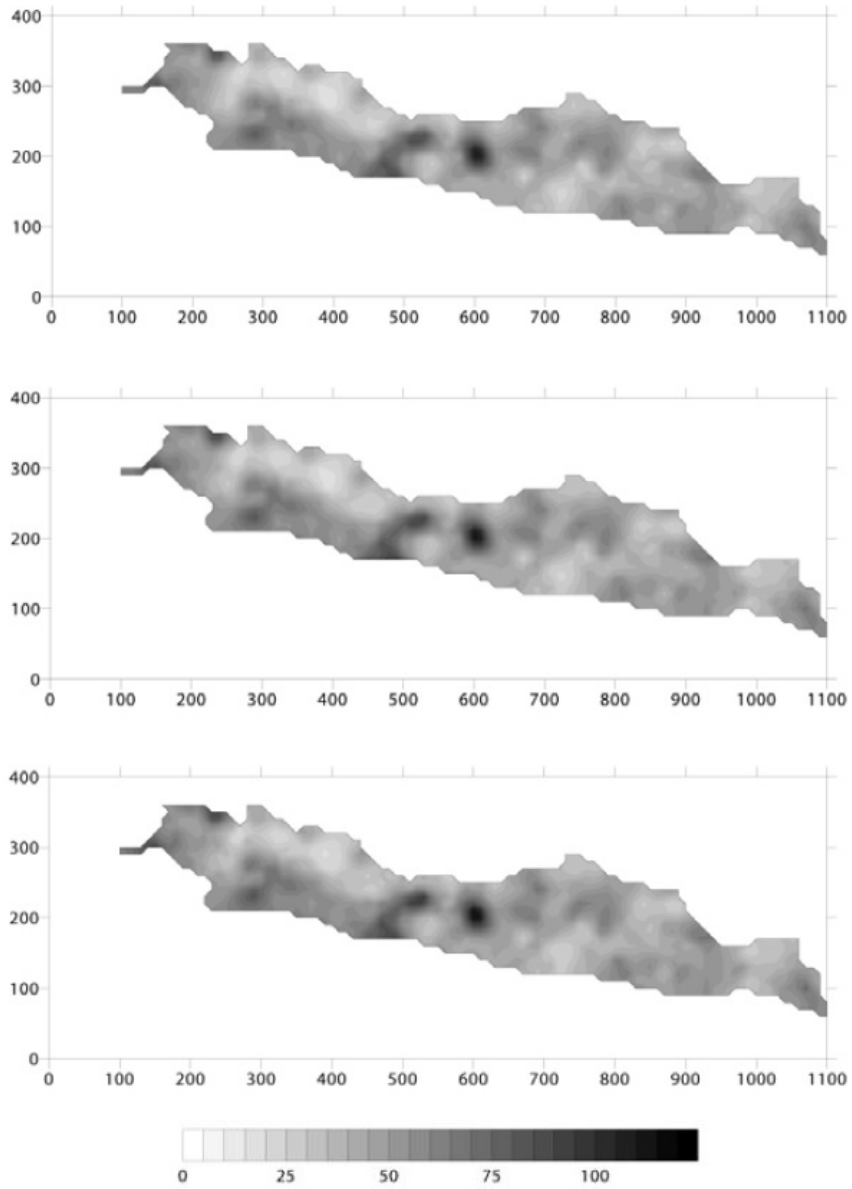
Obrázek 9: Ukázka fuzzy povrchu.
(převzato z: Santos et al. (2002))

3.3 Využití v praxi

Vondráková a Caha (2014) ve své publikaci uvádí, že stále ještě neproběhl řádný výzkum, jak v rámci fuzzy povrchů a jejich vizualizace vyjádřit kompletní informace o možných hodnotách a nejistotě. Caha (2015) pak nabízí, jakožto zdroj informací, například náhled do profilu fuzzy povrchu.

Několik autorů však našlo cestu, jak uplatnit fuzzy povrchy nebo fuzzy teorii obecně, a dokázali využít těchto poznatků minimálně ke zvýšení objektivity jejich práce. Příkladem může být studie zabývající se výškovou změnou britské pobřežní oblasti, která je klíčová v rámci protipovodňových opatření Velké Británie. Fisher a Caha (2014) odhalili ve svém výzkumu rozdíly za pomoci fuzzy povrchů vytvořených z LIDARových dat oblasti. Možné využití teorie o nejistotě v datech potvrzuje také výzkum o možnostech rozšíření interpolační metody kriging, která se ukázala ve výsledku jako přesnější s využitím fuzzy výpočtů (Masoomi et al., 2011). Nejlepší a nejhorší možný obraz skutečnosti představuje také ve své práci Piotrowski et al. (1996b), který se zabýval odhadem hydrogeologických parametrů pro modelování podzemní vody a jenž také dosáhl výsledků právě díky využití fuzzy logiky. Zajímavý pohled na problematiku přináší rovněž Spinella et al. (2008) v příspěvku o

interpolacích dat v rámci životního prostředí, kde byly fuzzy výsledky porovnávány se statickými modely. Poznatky o nejistotě byly využity i v Indonésii na ostrově Jáva pro modelování emisí metanu (Stein a Verma, 2005). Díky znalosti tématiky byla také navržena i tvorba fuzzy povrchu založeném na nejistotě v rámci variogramu a systému optimalizace (Caha et al., 2014). Dále, pro úplnost celku, je možné zmínit tematické publikace například od Caha et al. (2012), Piotrowski et al. (1996a) nebo Loquin a Dubois (2010).



Obrázek 10: Ukázka fuzzy krigingu obsahu metanu s použitím fuzzy variogramu (ostrov Jáva) - maximální hodnoty (nahoře), středové hodnoty odpovídající klasickému variogramu(uprostřed), minimální hodnoty (dole).

(převzato z: Stein a Verma (2005))

4 DATOVÁ STRUKTURA S4

Jedním z cílů diplomové práce je také převod původních funkcí do novější verze struktury *S4*, která se postupně dostává do popředí a nahrazuje svého předchůdce. Toto tvrzení však neplatí globálně a řada tradičních vývojářů pro R software se stále snaží držet verze *S3*. Nicméně, výhledově lze očekávat postupný odchod starší varianty a její nahrazení novou generací. Aby byla problematika programovacího jazyka S a jeho datových struktur řádně pochopena, je nutné si nejprve shrnout několik základních faktů. První důležitý mezník, je podle Genolini (2008) implementace jazyka, která čitá dvou směrů:

1. S-plus, jakožto komerční záležitost,
2. R, jakožto nekomerční záležitost.

Pohybujeme se tedy v nekomerční sféře, na poli jazyka R, v jedné ze dvou implementací jazyka S. Obsáhlý a velmi vydařený materiál k tomuto jazyku poskytuje například literatura od Johna M. Chamberse: Programming with data (Chambers, 1998). V návaznosti na předchozí tvrzení je nutné také poznamenat, že R nabízí hned několik objektově orientovaných systémů – *S3, S4* a *R5, R6* (Wickham, 2015). Hodnota číslice pak udává tzv. generaci neboli stupeň vývoje daného systému. V této práci se budeme věnovat datové struktuře *S4*, která bude níže také porovnána s předchozí verzí *S3*.

4.1 Srovnání S3/S4

První, ale také nejvíce markatní, rozdíl mezi oběma přístupy je jejich (vývojové) stáří. Přičemž je logicky na první pohled jasné, že technika *S3* je starší a že *S4* ji postupně nahrazuje. Avšak dle Genolini (2008) informačně signifikantní rozdíl přináší vývoj funkcí, který je charakteristický pro strukturu *S4*. Oproti starší verzi reflektuje programování v jazyku S více jako objektové. V rámci staršího přístupu (*S3*) je možné změnit jakýkoli objekt do objektu konkrétní třídy pouhým nastavěním atributu. Nové řešení *S4* je však mnohem přísnější a vyžaduje definici konkrétní třídy. Navíc otevírá pouze jedinou cestu, jak nového objektu dosáhnout, a to prostřednictvím konstrukční funkce *new*. Můžeme tedy stanovit dvě hlavní odlišnosti (Wickham, 2015):

1. definice formální třídy: oproti *S3* definuje *S4* formálně reprezentaci a dědictví pro každou třídu,

2. mnohonásobné volání funkcí: struktury *S4* umožňují definice obecných funkcí, které mohou být odkazovány do metody založené na třídě o libovolném počtu argumentů.

4.2 Charakteristika S4

Jak již bylo řečeno, datové struktury *S4* jsou založeny na formálním objektově orientovaném programování. Naprostý základ pro jejich tvorbu v softwaru R je instalace a načtení balíku methods (R Core Team, 2014) v rámci prostředí R. Díky tomuto kroku získáváme nástroje pro definice tříd objektů a tím i zakládání objektů *S4*. Kostru datové struktury pak lze rozdělit na tři elementární komponenty (Aboyoun, 2010):

1. třídy,
2. obecné funkce,
3. metody.

4.2.1 Definice třídy

Třída určuje reprezentaci, tedy podobu a obsah, daného objektu. Z praktického hlediska je možné definovat v podstatě jakýkoli formát dat uvnitř objektu, nebo jej alespoň nahradit formátem podobným. Důraz by měl být však kladem na základní typy dat *character, numeric* a podobně. Pro založení samotné třídy slouží klíčová funkce *setClass*. Definice tříd jsou pak uloženy jako metadata v jednotlivých balíčcích (lze využít i principu o dědičnosti objektů) a budou udrženy v paměti do doby, nežli to nebude dáno jinak, nebo nežli bude prostředí uzavřeno. Objektové třídy jsou tedy tvořeny za pomoci definice o jejich obsahu, která je specifikována několika argumenty (R Core Team, 2014; Gentleman, 2003):

- Class = tvořen textovým řetězcem, označuje název třídy,
- representation = jmenný seznam tzv. slotů v obsahu třídy, neboli datová reprezentace včetně definice datového typu daného vstupu do objektu,
- prototype = je argument, který poskytuje základní (předpřipravená) data pro jednotlivé sloty (důležité při zakládání objektu bez importu vlastních dat, nový objekt bude naplněn právě podle tohoto argumentu),
- contains = jmenný seznam tříd, která tato nově vytvořená třída bude obsahovat,

- validity = lze volně přeložit jako kontrola správnosti dat uvnitř objektu,
- where = prostředí, kde se mají metadata uložit nebo odstranit,
- access, version = kontrolní argumenty pro kompatibilitu s S-Plus (není často využíváno),
- sealed = logický operátor pro zákaz tvorby třídy o stejném názvu,
- package = volitelné, pro definici R balíčku do kterého daná třída spadá,
- S3methods = od verze R 3.0.0 se nepoužívá.

Programový kód 1: Funkce *setClass* a její argumenty

```
setClass(Class, representation, prototype, contains = character(),
         validity, access, where, version, sealed, package,
         S3methods = FALSE, slots)
```

Je evidentní, že ne všechny výše zmíněné argumenty jsou povinné, některé z důvodu vývoje dokonce již vyřazené. Pro jednoduchou a praktickou ukázku postačí tedy pouze první tři zmíněné. Pakliže chceme demonstrovat názorný příklad, musíme nejprve promyslet skladbu. Příkladem budiž definice třídy Diplomanti, určující skupinu jedinců s atributy Jméno a Rok zadání diplomové práce. Poté pro vytvoření objektu zavedené třídy použijeme konstrukční funkci *new* (Wickham, 2015). Dosáhneme tak nového objektu *S4* s daným obsahem (viz níže). V případě absence vstupních informací dojde k založení na základě argumentu prototype, kdy se jednotlivé sloty naplní základní hodnotou *NA*.

Programový kód 2: Příklad definice třídy a vytvoření objektu *S4*

```
setClass(Class = "Diplomanti",
         representation = representation(
                  jmeno = "character",
                  rok = "numeric"
                ),
          prototype = prototype(
            jmeno = NA_character_,
            rok = NA_real_
          ))

Seznam <- new("Diplomanti", jmeno = "Burian", rok = 2013)
```

Výsledkem je nový objekt Seznam třídy Diplomanti s atributy jmeno a rok. Data lze jednoduše zobrazit zavoláním názvu objektu. Případná selekce jednotlivých dat

(slotů) se provádí pomocí výběrového znaku @, což je další odlišností mezi datovými strukturami *S4* a *S3* (dříve se užívalo znaku *$*).

Programový kód 3: Výsledný objekt Seznam a výběr slotu jmeno

```
> Seznam
An object of class "Diplomanti"
Slot "jmeno":
[1] "Burian"

Slot "rok":
[1] 2013

> Seznam@jmeno
[1] "Burian"
```

Zajímavou možností, při definici nové třídy, je také kontrola vstupních dat. Tento proces zajišťuje argument validity a jeho definice není rozhodně složitá (Black, 2014). Reálná praxe navíc dokazuje, že kontrolní prvky v kódu jsou více než žádoucí. Zaveďme si proto kontrolní sekci pro slot rok. Budeme předpokládat, že dříve než v roce 1000 n.l. bylo zadání práce logicky nemožné. Argument pak tvoří jednoduchá rozhodovací funkce založená na cyklu *if-else* a vrací logické hodnoty *true/false*. V případě zavedení validity jednoduše postačí vložit tuto funkci do argumentu v těle definice třídy (*setClass*), nebo použít funkci *setValidity* (Morgan a Gentleman, 2008).

Programový kód 4: Argument validity

```
validity = function(object)
{
  if(object@rok < 1000) {
    return("Zadejte správný letopočet.")
  }
  return(TRUE)
}
```

4.2.2 Obecné funkce

Obecná funkce *setGeneric* je normální R funkce, která pracuje s metodami. Nejdůležitější je konstatovat, že se jedná o všeobecnou definici funkce, která disponuje možností mnohonásobného volání. Typicky tedy pouze zapouzdřuje obecný koncept výpočtů (například průměr, graf, predikce). Důležité je ale také mít na paměti, že sama o sobě žádné výpočty neprovádí (Peng, 2003). Primárně definuje název a klíčové argumenty

dané metody. Můžeme vytvořit metodu pro již existující obecnou funkci, nebo napsat svou vlastní, a to i včetně připojených metod. Mezi její hlavní argumenty patří (Morgan a Gentleman, 2008; R Core Team, 2014):

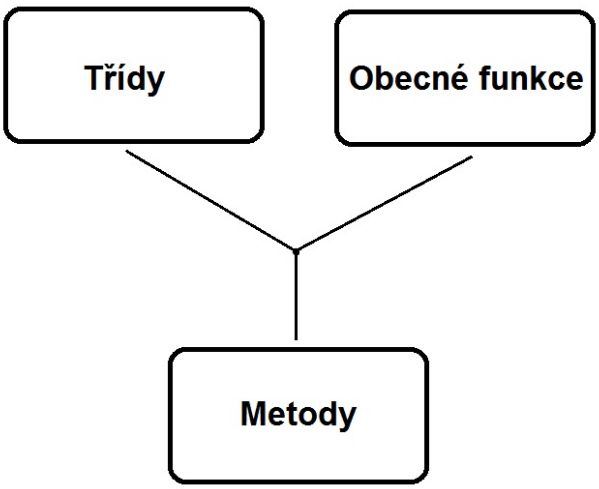
- name = název dané funkce pod kterým se bude volat,
- def = definice vstupních argumentů funkce,
- signature = výpis textových řetězců, které se mohou objevit na pozici signatury metod a tím se přiřadit právě k této obecné funkci,
- ostatní argumenty nejsou příliš často používané.

Programový kód 5: Plná definice obecné funkce

```
setGeneric(name, def = , group = list(), valueClass = character(),
           where = , package = , signature = , useAsDefault = ,
           genericFunction = , simpleInheritanceOnly = )
```

4.2.3 Metody

Právě metody, které obecné funkce rozšiřují, udávají pravidla a postupy pro výpočty. Zjednodušeně lze tvrdit, že se jedná o implementaci obecné funkce pro objekt určité třídy. Tento vztah lze jednoduše prezentovat i v grafické formě (viz obr. 11).



Obrázek 11: Graficky vyjádřený vztah tříd, obecných funkcí a metod (upraveno dle: Peng (2003))

Metodami se tedy určuje způsob, jakým stylem bude nakládáno s funkcí obecnou a navíc může odkazovat na určité třídy objektů, se kterými bude pracovat. Největší

výhodou metod je právě možnost odkazování se na dané třídy objektů, čímž se jednoznačně určí vstupní formát dat a tím i odpadá povinnost formální kontroly. Předepsaná definice jak pak opět uložena ve formě metadat u příslušné obecné funkce. Samotná skladba funkce *setMethod*, kterou lze definovat jednotlivé metody disponuje relativně podobnými argumenty jako u předešlých případů (R Core Team, 2014):

- f = název obecné funkce pro kterou je metoda přiřazena,
- signature = výpis názvů tříd vstupních objektů,
- definition = definice funkce, kterou metoda vykonává,
- where = prostředí, do kterého se definice metody ukládá (většinou se nepoužívá),
- valueClass = již zastaralý a nepoužívaný argument (stejně jako u obecných funkcí),
- sealed = logický operátor pro zákaz změny definice, nebo odstranění metody.

Programový kód 6: Funkce *setMethod* a její argumenty

```
setMethod(f, signature = character(), definition,
          where = topenv(parent.frame()),
          valueClass = NULL, sealed = FALSE)
```

Pro dokončení praktické ukázky, a zároveň i úplnost kapitoly, je žádoucí dokončit exemplární příklad. Vytvoříme si proto obecnou funkci s názvem "trvani"a napíšeme pro ni jednoduchou metodu, která bude pracovat s třídou Diplomanti (viz Programový kód 2). Smyslem úlohy bude zjistit, kolik let uplynulo diplomantovi od zadání diplomové práce, v úvahu přitom bude brán aktuální rok 2015. Jedná se tedy o základní matematickou úlohu, ve které se odečte od aktuálního roku ročník zadání práce.

Nejprve bude definována funkce obecná s názvem "trvani". Jejím úkolem je určit, že se bude pracovat s jedním objektem typu Diplomanti a blíže nespecifikovaným počtem argumentů uvnitř metody. Poté přijde na řadu klíčová metoda pro výpočet, zde musí být na prvním místě uveden název obecné funkce pod kterou spadá. Dále předepsaná signatura říká, aby se ke vstupnímu objektu přistupovalo jako k objektu třídy Diplomanti (jasná definice typu vstupu). Nakonec se připojuje formát funkce s argumenty object a aktualni_rok. V pořadí druhý argument samozřejmě vstupuje do výpočtu, jenž je doplněn o rok zadání práce z objektu. Závěr je ukončen vytvořením nového objektu s pozměněným atributem "rok", jenž bude nyní reprezentovat počet let uplynulých od doby zadání diplomové práce (viz níže).

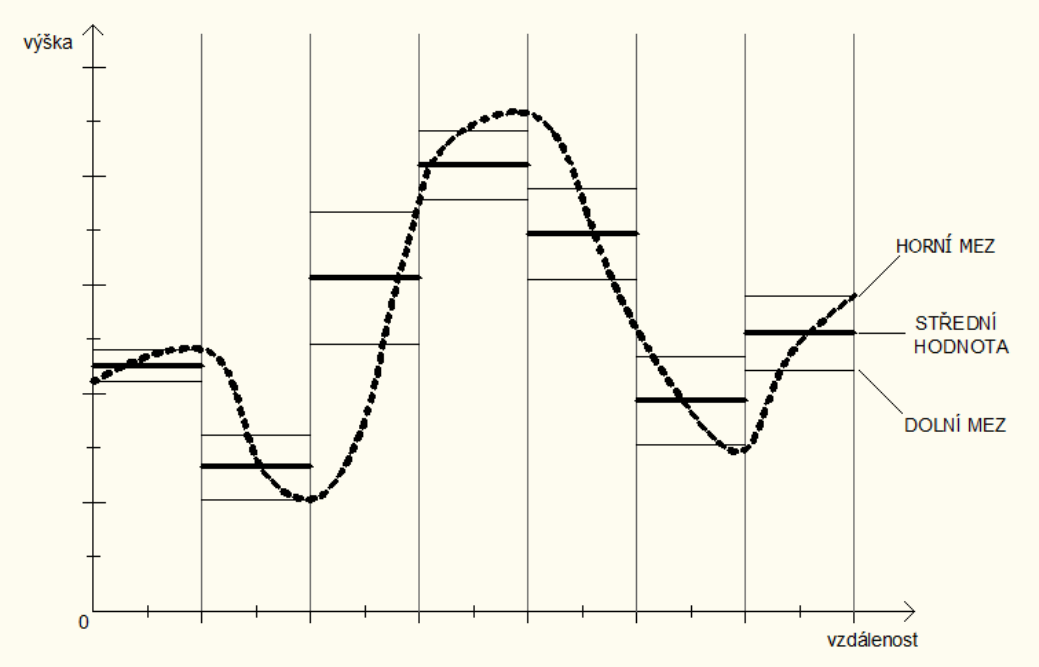
Programový kód 7: Výpočet počtu let trvání diplomové práce od roku 2015

```
setGeneric("trvani",
           function(object,  ...)
           standardGeneric("trvani")
)

setMethod("trvani",
          signature(object = "Diplomanti"),
          definition = function(object, aktualni_rok = 2015)
          {
            rozdil = aktualni_rok - object@rok

            new("Diplomanti", jmeno = object@jmeno, rok = rozdil)
          })
```

5 INTERPOLACE

Pojem interpolace všeobecně vyjadřuje proces odhadu neznámých hodnot v určité problematice na základě známých dat. Jeho průběh se také často označuje jako predikce neznámých hodnot (Křikavová, 2009). Vstup do interpolace pak tvoří dvě základní veličiny, již zmíněná data a zvolený grid. Vstupní grid poté figuruje v celém procesu pouze jako mřížka určená pro uspořádání interpolovaných hodnot (Burian, 2013). V případě této diplomové práce jsou však vstupní data obohacena o model nejistoty a tudíž vystupují jako fuzzy čísla (viz kapitola 3). Vstupem do interpolací budou tedy fuzzy intervaly $\widetilde{z}_i = [z_i^-, z_i^0, z_i^+], i = 1, ..., n$, přičemž veškeré naměřené hodnoty budou reprezentovat právě jednu souřadnicovou pozici $(x_i, y_i), i = 1, ..., n$ (Lodwick et al., 2008). Podle Waelder (2007) jsou však všechna měření rozmístěna nepravidelně, proto nemusí odpovídat datům v definovaném gridu. Pokud vezmeme v úvahu exemplární grid s příkladnou definicí $\{X_j, Y_k\}, j = 1, ..., N, k = 1, ..., M$, pak hodnoty interpolace $\widetilde{Z}_{jk} = [Z_{jk}^-, Z_{jk}^0, Z_{jk}^+]$ budou také fuzzy čísla ovlivněná interpolační metodou (Santos, 2008; Burian, 2013).

Průběh interpolace pak tedy probíhá celkem ve třech iteracích, kde jednotlivé vstupy tvoří právě tři zkoumané proměnné vstupních fuzzy čísel. Do kalkulací postupně vstupují hodnoty pro dolní mez z^-, střední hodnotu z^0 a horní mez z^+ (Burian, 2013).



Obrázek 12: Exemplární průběh interpolace - vyjádřen přerušovanou linií (převzato z: Burian (2013))

Pro vyšší srozumitelnost byl celý proces znázorněn i graficky (viz obr. 12). V rámci diplomové práce pak byly brány v potaz celkem tři základní interpolační metody:

1. IDW,
2. spline,
3. kriging.

Výše zmíněné metody byly implementovány do původního balíčku Uncertainty interpolation pro software R (Burian, 2013). V předchozí práci však prakticky nebyl brán ohled na funkční parametrizaci procesů, nebo jakýkoli jiný hlubší výzkum problematiky. Jednotlivé interpolace byly zavedeny do balíčku v podobě základních funkcí, které poskytují potřebné algoritmy. Úkolem této (navazující) diplomové práce je pak prozkoumat možnosti pro parametrizace a případné rozšíření těchto již dříve použitých funkcí. Funkce proto byly parametricky rozepsány podle jejich možností v softwaru R a zároveň jim byly i nastaveny výchozí hodnoty pro elementární funkčnost celého balíčku.

5.1 Grid

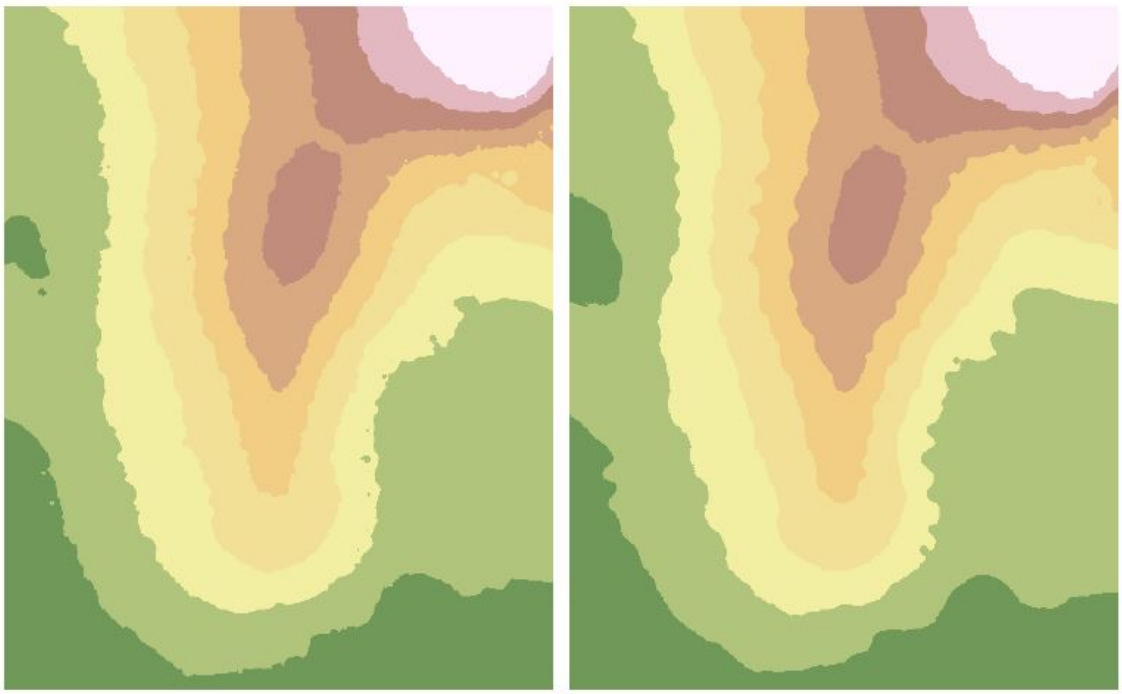
Vstupní grid je jedním z elementárních vstupů do interpolací, který určuje výsledné uspořádání dat. V rámci vyvíjeného balíčku byl proto grid chápán jako důležitý aspekt parametrizace interpolačních procesů, protože právě zvolené interpolační funkce se liší ve svých nárocích na formát gridu. Zatímco metoda spline si vystačí pouze s obecným gridem ve formátu seznamu proměnných o stejném počtu řádků s jedinečnými názvy (*data.frame*), algoritmy pro výpočet metody IDW a kriging vyžadují prostorový (*Spatial*) grid (R Core Team, 2014). Rozdíl mezi nimi je v již přiřazených souřadnicích pro jednotlivé body uvnitř mřížky. V kalkulacích se tento krok pak projeví ve formátu kódu funkce, který je jednodušší a efektivnější. Uživateli navíc odpadá povinnost řešit tuto problematiku, protože je již vyřešena vedlejší funkcí, která zajišťuje tvorbu gridu ve dvou možných variantách:

1. na základě uživatelem definovaným počtem buněk výsledné mřížky ve směru osy x a y,
2. na základě předdefinované velikosti mřížky určené podle vstupních dat, kde si uživatel zvolí pouze velikost hrany pro každou buňku.

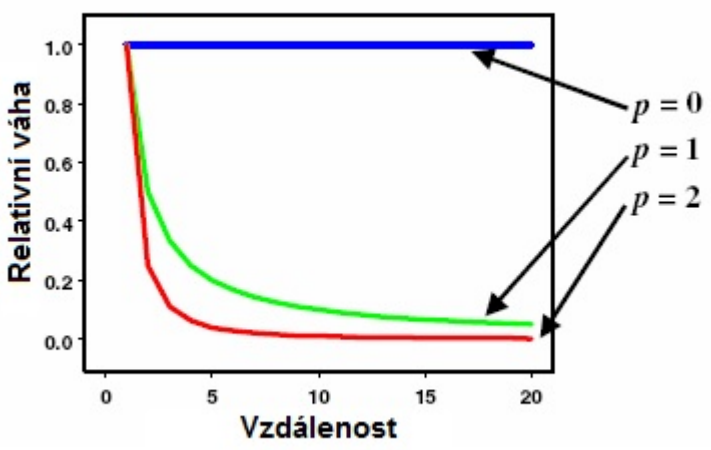
5.2 IDW

Zkratku IDW lze volně přeložit jako metodu inverzních vzdáleností, která je založena na základním geostatistickém principu. Reflektuje jevy, které jsou si v prostoru vzájemně blíže, jako více podobné než jevy, které jsou od sebe v prostoru vzdálenější (Křikavová, 2009).

Uvnitř výpočtů pak figurují celkem dva klíčové parametry, a to Power (neboli síla) a počet vstupních bodů, které budou vstupovat do funkčních kalkulací. Prvním argumentem lze regulovat vliv jednotlivých bodů na interpolované hodnoty v závislosti na jejich vzdálenosti od odhadovaného bodu. Čím vyšší tato hodnota bude, tím vyšší důraz bude kladen na nejbližší body, výsledný povrch pak bude více členitý a tím pádem i méně vyhlazený. Na druhou stranu nižší hodnota argumentu Power pak logicky ukládá větší vliv na vzdálenější body a výsledkem je tím pádem i více vyhlazený povrch. V případě, kdy je argument nulový, pak je výsledný bod pouhým průměrem zahrnutých bodů (Svobodová, 2008). Standardně je volena hodnota síly Power na stupnici číslic od nuly do dvou.



Obrázek 13: Míra vyhlazení povrchu metodou IDW - vlevo Power = 1, vpravo Power = 2 (převzato z: Křikavová (2009))



Obrázek 14: Volba parametru Power (upraveno dle: Křikavová (2009))

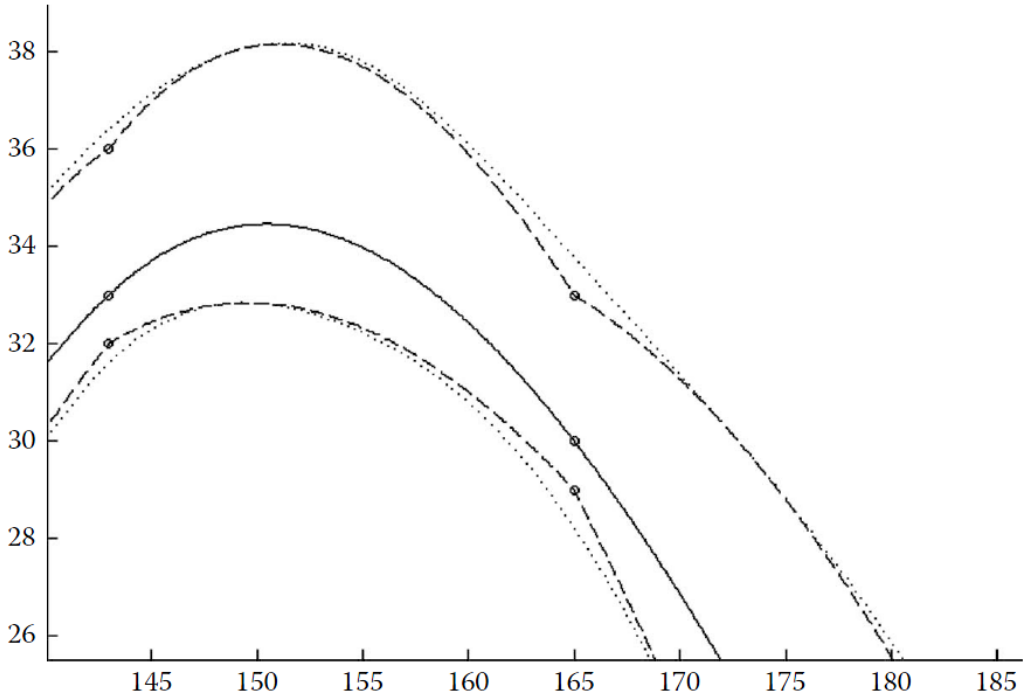
5.3 Spline

Interpolační metoda, jenž je založena na matematicky definovaných křivkách. Tyto křivky pak po částech postupně interpolují jednotlivé úseky povrchu, který musí protínat datové body a také musí mít minimální křivost. Výsledek, díky eliminaci skoků a bariér, poté bývá zpravidla velmi vyhlazený (Svobodová, 2008). Největší uplatnění samotné metody se nachází při modelování pozvolna se měnících jevů, na druhou stranu však není doporučena její aplikace u bodově zhuštěných datových sad se značně rozdílnými hodnotami (Křikavová, 2009). Svobodová (2008) pak mimo jiné uvádí dva základní typy této metody:

1. regulovaný spline,
2. spline s tenzí.

Obě uvedené metody jsou nejvíce ovlivněné parametrem definovaným jako váha. V případě první varianty ovlivňuje váhu třetích derivací povrchu v rámci minimalizace křivosti. Čím vyšší hodnotu váhy uvedeme, tím více vyhlazený pak bude i výsledný povrch, přičemž by se tato hodnota měla rovnat nule, nebo pohybovat v kladných hodnotách. Běžně uváděné váhy pro regulovaný spline jsou 0, 0.01 nebo 0.5. Metoda spline s tenzí pak řeší tuhost povrchu podle charakteru modelované problematiky. Váha zde určuje míru tenze, která ovlivňuje minimalizační vlastnosti pomocí zahrnutí první derivace do samotného procesu. Vyšší hodnoty váhy tenze udávají hrubší výsledný povrch, typickými hodnotami bývají například 0, 1 nebo 10 (Svobodová, 2008). V rámci diplomové práce byl však, pro práci s prostorovými daty, použit kubický spline. Tato metoda dokáže pracovat s nepravidelně rozmístěnými daty a její

vyhlazovací parametr je generován na základě generalizace křížové validace (R Core Team, 2014).



Obrázek 15: Exemplární průchod fuzzy spline (převzato z: Santos (2008))

5.4 Kriging

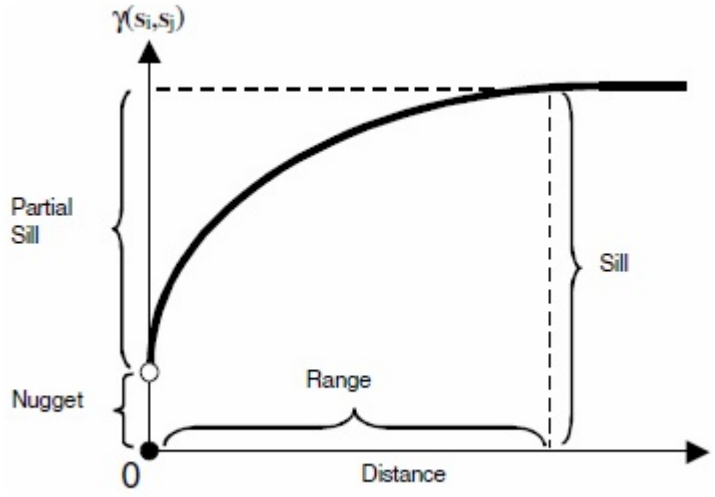
Autorem této metody je francouzský matematik Georges Matheron, který ji vynalezl na základě tezí Daniela Gerharduse Kriega, podle nějž byla i metoda pojmenována. Oproti předchozím zmíněným metodám je postavena na geostatistice, proto je i jejím všeobecným principem prostorová závislost, neboli autokorelace zkoumaných jevů v okolí předpovídané hodnoty. Tuto prostorovou závislost pak definuje variogram, který je stanoven ze vstupních dat a vlastností celkové veličiny. Samotný kriging pak podává nejlepší nestranný lineární odhad interpolované veličiny, přičemž po technické stránce patří metoda k nejsložitějším a její uplatnění je především při interpolacích jevů, které se v prostoru mění s určitou kontinuitou (Goovaerts, 1997; Křikavová, 2009).

Základ tedy tvoří variogram, který lze jednoduše popsat jako matematicky definovanou křivku jenž prokládá vstupní data tak, aby je co možno nejlépe vystihovala. Obecně platí, že pro plošná data by měl existovat dvojrozměrný variogram a pro data v prostoru pak variogram trojrozměrný. Pro praktické zjednodušení definice variogramu bylo následně definováno několik funkcí, které lze efektivně využít pro jeho modelování. Mezi tyto funkce například patří (R Core Team, 2014; Goovaerts, 1997):

- sférické,

- Gaussovy,
- exponenciální,
- kruhové, Besselovy, mocninné a další.

Samotný variogram pak vymezuje několik parametrů (viz obr. 16). Prvním je range, neboli rozsah vzdálenosti mezi body, kde ještě působí autokorelace. Průběh udává nugget, který vymezí počátek a hodnota sill, jenž určí maximální hodnoty bodů, které budou vstupovat do kalkulací. Rozdíl mezi nugget a sill pak definuje partial sill (Cressie, 1991).



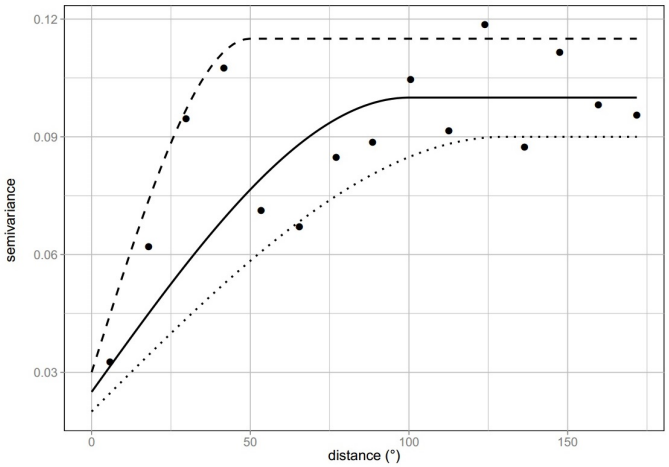
Obrázek 16: Složky variogramu (převzato z: Křikavová (2009))

Po sestavení modelu prostorové autokorelace lze přistoupit ke tvorbě interpolovaného povrchu, který podle Cressie (1991) tvoří tři složky. První je obecný trend, jakožto strukturální komponenta. Dále je to regionalizovaná proměnná vyjadřující kolísání průběhu funkce a třetí složku představuje náhodný šum. Všechny tyto složky jsou řešeny odděleně, přičemž trend odhaduje obecná trendová funkce a prostorově korelované kolísání (regionalizovanou proměnnou) analyzuje variogram. Podle typu trendu pak rozlišujeme tři varianty pro metodu kriging (Goovaerts, 1997):

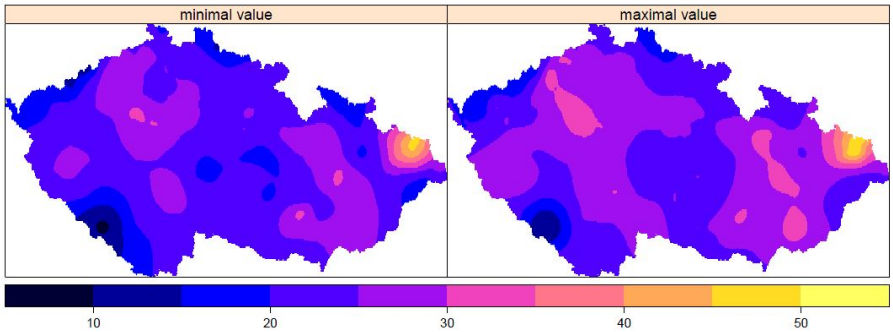
1. simple kriging - známý, konstantní trend,
2. ordinary kriging - neznámý, konstantní trend,
3. universal kriging - trend představuje lineární kombinaci matematických funkcí v závislosti na poloze.

5.4.1 Fuzzy kriging

Pro rozšíření funkčních možností interpolací uvnitř výsledného balíčku byla implementována i alternativní metoda pro dosažení výsledků pomocí fuzzy krigingu. Tato výpočetní varianta je však určená spíše pro zkušenější uživatele. Nutností vstupu je zde udání hodnot pro definovaný počet parametrů krigovací funkce (model, sill, range, nugget). Výpočet modelu pak probíhá zvlášť pro každou kombinaci podle počtu zadaných vstupních parametrů. Problém však může nastat v případě, kdy je na vstupu vloženo příliš mnoho údajů pro každý parametr, logicky pak exponenciálně roste i počet kombinací jednotlivých modelů pro výpočet. Například při deseti údajích pro každý parametr přichází deset tisíc kombinací pro výpočty všech možných modelů, časová náročnost procesu se tak může značně navýšit. Výsledné modely poté vstupují do predikcí krigingu. Z jednotlivých výsledků predikce jsou pak postupně vybírány právě nejvyšší a nejnižší hodnoty v daném zkoumaném bodě. Na konci procesu jsou výsledkem (oproti originální datové sadě) fuzzy čísla ve formátu horní mez/střední (původní) hodnota/dolní mez (Caha et al., 2015).



Obrázek 17: Fuzzy variogram. Dolní a horní hranici představují přerušované linie, střední hodnotu symbolizuje plná linie. (převzato z: Caha et al. (2015))



Obrázek 18: Ukázka výstupu fuzzy krigingu. Vlevo spodní hranice a vpravo horní hranice fuzzy čísel. (převzato z: Caha et al. (2015))

6 BALÍČEK UNCERTAINTY INTERPOLATION

V návaznosti na předchozí kapitoly bylo přistoupeno k úpravě původního balíčku Uncertainty Interpolation. Nejprve byly přepsány datové struktury formátu *S3* do novější verze *S4*. Poté byly nově upravené funkce rozšiřovány a parametrizovány. Na závěr byl celý balíček řádně okomentován a doplněn o funkční dokumentaci. Poslední krok byl záměrně vytvořen kompletně v anglickém jazyce, a to z praktického důvodu pro kontakt se zahraničními uživateli (předchozí verze byla okomentována z části v českém a z části v anglickém jazyce). Výsledkem je tedy nový balíček verze 2.0, který bude v prostředí R vystupovat pod zkratkou UncerIn2.

Tabulka 1: Základní charakteristika verzí balíčků UncerIn

UncerIn 1.0	**UncerIn 2.0**
datové struktury S3	datové struktury S4
14 funkcí	19 funkcí
CZ/ENG dokumentace	ENG dokumentace
omezený vstup dat	variabilita možného vstupu (např. meuse)
2 možnosti pro tvorbu gridu	až 6 možností pro tvorbu gridu
základní možnosti interpolace	parametrizace procesů
neřešen výstup	výstup v definované třídě (jednoduché pro manipulaci i vizualizaci)
bez vizualizace	funkce pro vizualizace dat i variogramu

Z původního balíčku UncerIn 1.0 tedy příliš mnoho nezůstalo, zachována byla pouze funkční myšlenka balíčku a některé klíčové funkce (například pro výpočet interpolací). Nový balíček stále reflektuje myšlenku propagace nejistoty skrze fuzzy modely postavených na vstupních datech, které vstupují do interpolací. Vedle funkcí pro tvorbu modelů nejistoty a výpočet interpolací, nabízí také funkce pro manipulaci s daty, generování gridu, odhad variogramu, či vizualizace. Nově také bylo zavedeno celkem pět objektových tříd formátu *S4*, jejichž účelem je zajistit správnost dat a zjednodušit manipulaci s nimi uvnitř procesů. Výsledky tedy lze snadno vizualizovat a prakticky tak i ověřit funkčnost celého balíčku.

6.1 Třídy S4 objektů

Objekt, jakožto vstupní veličina do jednotlivých funkcí uvnitř balíčku, musí mít předem očekávaný (definovaný) formát. Z tohoto důvodu je velmi výhodné zavádění právě objektových tříd pro jednotlivé vstupy do funkcí. Předem připravená třída zajišťuje správnost i kontrolu dat na vstupu a případně, jestliže je některá podmínka nesplněna, dokáže uživatele upozornit na možné neshody.

Příkladem mohou být funkce tvořící modely nejistoty, které přijímají bodová vstupní data s atributy *x, y, z*, přičemž údaje *x, y* značí přirozeně souřadnice a údaj *z* udává zkoumanou proměnnou. Efektivní řešení v tomto případě představuje definice třídy *Points* pro data o numerických atributech *x, y, z*, která bude definovat vstup do procesu tvorby zmíněných modelů.

Programový kód 8: Ukázka definice datového objektu Points

```
setClass(Class = "Points",
         representation = representation(
           x = "numeric",
           y = "numeric",
           z = "numeric"
         ),
         validity = function(object){
           if(length(object@x) != length(object@y) || length(
              object@x) != length(object@z)){
             return("Length of all parameters (x,y,z) should be
                equal.")
           }
           return(TRUE)
         })
```

Druhou část programového kódu (viz výše) pak tvoří kontrola formátu dat (například sloupce o stejné délce), aby nedocházelo k chybám vzniklým na základě chybějících, nebo dokonce chybných dat. Podobným způsobem se přistupovalo i k ostatním objektovým třídám uvnitř balíčku, postupně tak bylo vytvořeno pět tříd určující případný vstup nebo výstup dané funkce.

Další třída se nazývá *UncertainPoints* a po vzoru z předchozí bakalářské práce zaobaluje data obohacená o model nejistoty. Jejími atributy tedy jsou již zmíněné souřadnice *x, y* a modifikovaná zkoumaná proměnná ve formátu dolní hranice nejistoty *uncertaintyLower*, střední hodnoty *modalValue* a horní hranice *uncertaintyUpper*. Následuje objekt třídy *UncertainInterpolation*, který definuje výstup z interpolačních procesů. Obsahuje stejné atributy jako předchozí třída, avšak souřadnice souboru

dat již definují výstupní grid a jednotlivé hodnoty zkoumané proměnné představují výsledky interpolací v daných bodech. Předposlední třída *FuzzyInterpolation* je specifická především díky procesu fuzzy interpolace, jehož výstup definuje v podobném stylu jako předchozí objekty. *Variogram* je poslední třídou uvnitř balíčku sloužící pro definici tří skrytých tříd (*Vario*), které popisují odhady variogramů pro jednotlivé prahy nejistoty na dolní/horní hranici a střední hodnotu. Uvnitř těchto skrytých objektů jsou pak definované klíčové atributy (model, sill, range a podobně) variogramu, ve výsledku je však zaobaluje třída *Variogram* a navenek poté vystupují pouze jako tři variogramy pro dané hodnoty.

6.2 Vstupní data

Vstupní data definuje objektová třída *Points*, které postačí pro vytvoření odpovídajícího objektu pouhé označení požadovaných veličin pro atributy *x, y, z*. Uživateli se tak otevírá praktická možnost pro testování v podstatě libovolných vyhovujících dat. Pro ověření funkčnosti algoritmů se zde také nabízí možnost vložení dat z volně dostupných datových sad (například meuse).

Programový kód 9: Funkce zajišťující naplnění třídy Points, na vstupu argumenty pro data x/y/z

```
Points <- function(x, y, z)
{
  new("Points", x = x, y = y, z = z)
}
```

6.3 Modely nejistoty

Funkce pro tvorbu modelů nejistoty přijímají na vstupu procesu data ve formátu objektové třídy *Points*. Logicky jsou tak očekávána předdefinovaná data, která nevyžadují žádné další složité úpravy. Po vstupní kontrole je tedy možné ihned přistoupit k samotnému jádru kalkulací a vytvořit tak model nejistoty nad danými daty.

Proces tvorby nejistoty může být jednoduše popsán jako určitá manipulace se zkoumanou proměnnou *z*. Od vybrané proměnné jsou postupně odčítány a přičítány uživatelem definované hodnoty, které později určují samotný model nejistoty. Pro názornou demonstraci tvorby modelu o konstantní nejistotě lze uvést fiktivní situaci, kdy probíhal sběr terénních dat pomocí běžného příručního přístroje GPS. U tohoto stroje předpokládáme (za normálních konstantních podmínek) určitou výškovou nepřesnost vlivem nedokonalosti technologie, například 3 metry. Takto nepřesná data pak mohou být ideálním vstupem do balíčku, kdy na počátku budou obohacena právě o zmíněný

konstantní model nejistoty. Výšková proměnná bude základem pro výpočty, kdy bude k jednotlivým bodům postupně připočítávána/odečítána hodnota nepřesnosti pro vytvoření horní/dolní hranice modelu nejistoty. Po modifikaci zkoumané proměnné jsou pak data včetně souřadnic zabalena do třídy *UncertainPoints*. Na stejném principu bylo postupně vytvořeno celkem šest různých funkcí pro tvorbu šesti různých modelů nejistoty založených na (Burian, 2013):

1. uncertaintyConstant - konstantní nepřesnosti,
2. uncertaintyError - základě prostorově korelované chyby,
3. uncertaintyPercent - procentuální nepřesnosti,
4. uncertaintyRandomDeviate - náhodné odchylce od zkoumané proměnné,
5. uncertaintyRandomNumber - výběru konstantní nepřesnosti z definovaného intervalu hodnot,
6. uncertaintyRandomPercent - výběru procentuální nepřesnosti z definovaného intervalu hodnot.

Programový kód 10: Ukázka výpočetní části funkce pro tvorbu modelu nejistoty založeného na náhodné procentuální nepřesnosti

```
randomPercent = runif(length(data@z), min, max)

percent = (data@z/100) * randomPercent
modify = abs(percent)

uncertaintyLower = data@z - modify
uncertaintyUpper = data@z + modify

new("UncertainPoints", x = data@x, y = data@y, uncertaintyLower
   = uncertaintyLower, modalValue = data@z, uncertaintyUpper =
   uncertaintyUpper)
```

6.4 Grid

Již dříve bylo předesláno, že funkce pro tvorbu gridu byly značně rozvinuty pro potřeby vstupních hodnot do interpolací. Oproti předešlé verzi je tedy nyní možné vytvořit v rámci balíčku až šest variant gridu. Největší zásluhu na tomto faktu nese nově zavedený vstupní parametr gridded, který určuje, zda bude mít výsledný grid

definované vlastní souřadnice. Na počátku každé funkce si tak uživatel vybírá jaký formát výsledku aktuálně preferuje. Volba probíhá skrze jednoduché naplnění zmíněného argumentu gridded, který přijímá logické hodnoty *TRUE/FALSE*. V zásadě pak lze hovořit o třech metodách, které formulují pro uživatele klíčové vlastnosti gridu a navíc se všechny rozdělují do dvou funkčních kategorií podle uživatelem definovaného argumentu gridded. První varianta umožňuje definovat počet buněk mřížky ve směru jednotlivých os x a y. Druhou možností je přenechat kalkulace rozměrů mřížky na funkčním kódu podle souřadnic vstupních dat a definovat pouze velikost každé buňky v mřížce. Poslední přístup byl poté vytvořen čistě pro doplnění možností prací s výsledky interpolací. Jedná se o variantu, kdy je uživateli nabídnuto vytažení souřadnic z výsledné třídy interpolací (*UncertainInterpolation*) a vytvořit z nich zpětně gridovou mřížku.

Programový kód 11: Část metody tvorby gridu určující velikost mřížky na základě souřadnic vstupních dat, o velikosti buňky 10

```
setMethod("Grid.box",
          signature(object = "Points"),
          definition = function(object, gridded = FALSE,
                                cellsize = 10)
          {
            if (identical(gridded, FALSE)){
              x = seq(min(object@x), max(object@x),by = cellsize)
              y = seq(min(object@y), max(object@y),by = cellsize)

              grid = expand.grid (x, y)
              result = grid
              return(result)
            }
             else if (identical(gridded, TRUE)){
               x = seq(min(object@x), max(object@x),by = cellsize)
               y = seq(min(object@y), max(object@y),by = cellsize)

               grid = expand.grid (x, y)
               names(grid) = c("x", "y")
               gridded(grid) = ~ x + y
               result = grid
               return(result) # class SpatialPixels
             }
```

6.5 Interpolace

Interpolační metody uvnitř balíčku byly parametrizovány podle jejich možností v rámci funkcí softwaru R. Zjednodušeně lze tvrdit, že funkční parametry byly rozepsány a nabídnuty uživateli pro jejich definování. Zároveň však byly i nastaveny exemplární hodnoty jednotlivých vstupů tak, aby byl umožněn ukázkový chod výpočetních algoritmů funkcí (pro případ, kdy uživatel nemodifikuje parametry). Práce s daty se tak dostává do určité režie uživatele, který může nastavovat vlastnosti procesů a získat tak kvalitnější výsledek.

Programový kód 12: Příkladná parametrizace interpolační metody spline

```
setMethod("splineUncertain",
          signature(object = "UncertainPoints",
                    grid = "data.frame"),
          definition = function(object, grid, m = NULL, p = NULL,
                                scale.type = "range", lon.lat = FALSE,
                                miles = TRUE, method = "GCV",
                                GCV = TRUE))
```

Jednotlivé výpočetní algoritmy funkce byly tedy rozepsány a zároveň do nich byly zakomponovány definované objekty pro vstup dat. Na vstupu jsou očekávána data ve formátu třídy *UncertainPoints* a definovaný grid v podobě jenž je očekávána danou interpolační metodou. Data pak postupují do interpolačního procesu, který je řízen podle definovaných parametrů. Před interpolací jsou však data ještě upravena, a to skrze skrytou transformační funkci *as.dataframe*, která je také součástí balíčku. Úkolem této transformace je převést vstupní *S4* objekty třídy *UncertainPoints* nebo *Points* do podoby *dataframe* se kterou pracují algoritmy interpolace.

Programový kód 13: Ukázka průběhu interpolace spline pro dolní mez modelu nejistoty (uncertaintyLower)

```
a = as.dataframe(object)
loc = cbind(a$x, a$y)

a1 <- Tps(loc, object@uncertaintyLower, m = m, p = p,
          scale.type = scale.type,
          lon.lat = lon.lat, miles = miles,
          method = method, GCV = GCV)

a11 <- predict(a1, grid)
```

Po průchodu dat interpolacemi jsou jednotlivé predikce v bodech vytaženy a zabaleny do výsledné objektové třídy *UncertainInterpolation*. Tato třída pak může vstupovat do vizualizační funkce balíčku, která vytvoří přehledný obrázek vzniklé situace. Zároveň se také nabízí možnost exportu gridové mřížky pomocí funkce pro tvorbu gridů (viz předchozí podkapitola).

Dále byla, v rámci rozšíření kriging interpolace, parametrizována také funkce pro odhad variogramu. Celý proces slouží sice jen jako doplnění interpolací, ale prakticky je tak umožněn uživateli další pohled na daná data (objektu třídy *UncertainPoints*). Nový kód odhadu variogramu s sebou totiž přináší i rychlý vizualizační export variogramu, a to rovnou pro všechny mezníky nejistoty (horní mez, střední hodnota, dolní mez). Samotné výsledky variogramu je také možně vytisknout skrze funkci *show*, která byla definována uvnitř funkce pro vymezení třídy *Variogram*.

Programový kód 14: Definice funkce pro odhad variogramu a příklad vytisknutých hodnot výsledku

```
setMethod("variogram",
          signature(object = "UncertainPoints"),
          definition = function(object, model = c("Sph", "Exp",
                        "Gau", "Ste"), kappa = c(0.05, seq(0.2,
                        2, 0.1), 5, 10), fix.values = c(NA,NA,NA),
                        verbose = FALSE, GLS.model = NA,
                        start_vals = c(NA,NA,NA),
                        miscFitOptions = list())

> show(Variogram)
Miminimal Variogram
model    psill   range   kappa
Nug 1.23606911682792  0 0
Ste 19.7480969619628  15302.6771975328  10
----------------------------------------
Modal Variogram
model    psill   range   kappa
Nug 0.722907070741834 0 0
Ste 0.468759617472766 837.524955495225  10
----------------------------------------
Maximal Variogram
model    psill   range   kappa
Nug 0.659218782868021 0 0
Sph 0.967140410356054 1072.05528829073  0
----------------------------------------
```

Závěrečné, a také nejsložitější, rozšíření metod interpolací tvoří *fuzzy kriging*. Princip tohoto alternativního přístupu k interpolacím byl shrnut v předešlé kapitole, přičemž samotná myšlenka vychází z publikace Caha et al. (2015). Jedná se tedy o metodu pro zkušenější uživatele, která kombinuje uvnitř výpočetního procesu vstupní parametry krigingu. Na vstupu přijímá pouze původní data třídy *Points* a definovaný grid. Samotné parametry i algoritmy výpočtů byly převzaty a implementovány z online zdroje github.[1] Výsledkem procesu je pak speciální třída objektu *S4* nazvaná jako *FuzzyInterpolation*, a to z důvodu odlišení výsledků od ostatních funkčních procesů uvnitř balíčku. Využití této varianty metody (pro kriging interpolaci) však vyžaduje hlubší znalosti v rámci tematiky a samozřejmě také znalost samotné publikace, ze které vychází.

6.6 Vizualizace

Finální část balíčku tvoří nástroje pro vizualizace. Toto dodatečné rozšíření bylo vytvořeno čistě pro praktické účely tak, aby byl uživateli umožněn náhled na data. Jako první byla vytvořena funkce (viz níže) pro vizualizaci výsledků interpolací, tedy třídy *UncertainInterpolation*. Jedná se o aplikaci vizualizační funkce *spplot*, která má za úkol data vykreslit podle zadaných parametrů.

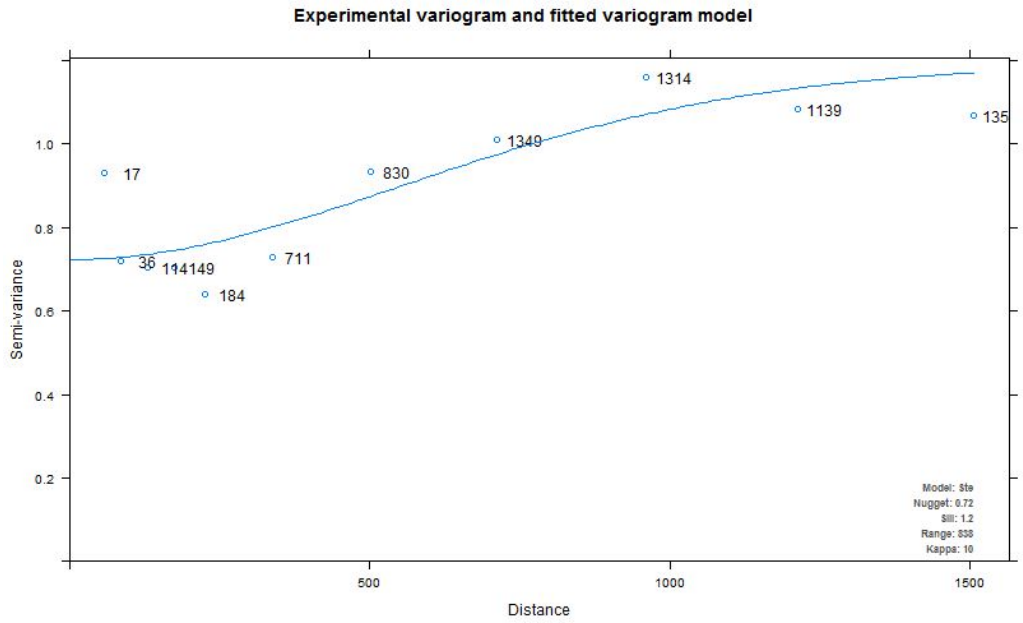
Programový kód 15: Definice a funkční jádro funkce pro vizualizace

```
setMethod("Plot",
          signature(object = "UncertainInterpolation"),
          definition = function(object, attr1 = "uncertaintyLower",
                 attr2 = "modalValue", attr3 = "uncertaintyUpper",
                 cuts = 10, pretty = TRUE)
           {
             a = as.UncertainPoints(object)
             a = as.dataframe(a)
             gridded(a) = ~ x + y

             spplot(a, c(attr1, attr2, attr3), names.attr= c(attr1,
                 attr2, attr3), colorkey=list(space="bottom"),
                 layout=c(3,1), cuts = cuts, pretty = pretty)
           })
```

Případný uživatel si pak jen zvolí vstupní objekt, v němž označí atributy určené pro zobrazení, dále poté i parametry pro počet zobrazených skupin rozdělujících data (cuts) v rámci vizualizace a styl vykreslení (pretty). Definici potřebných souřadnic

[1] Autorem Jan Caha, zdroj: https://github.com/JanCaha/Hais2015-paper

dat, které jsou obsaženy ve vstupním objektu, zajišťuje samotná funkce a tím i poskytuje určitý komfort pro uživatele. Uvnitř algoritmu *spplot* jsou pak nastaveny základní parametry pro pozici legendy a rozložení jednotlivých obrazů vstupních dat. V rámci celého balíčku však byla uskutečněna ještě jedna myšlenka pro zobrazení dat, a to pro odhad variogramu (viz obr. 19). Uvnitř funkčního kódu pro jeho tvorbu jsou zakomponovány základní vizualizační funkce *plot*, které vytvoří obraz variogramu pokaždé je-li funkce zavolána. Uživateli se tak opět dostává rychlého pohledu na data pro přehlednou kontrolu či analýzu výsledků.



Obrázek 19: Ukázka vizualizace variogramu (s výpisem klíčových hodnot v pravém dolním rohu)

7 INSTALACE BALÍČKU V R SOFTWARE

V případě, kdy je potřeba rozšířit funkční aparát prostředí R, je nutné rozšíření nejprve stáhnout (nainstalovat) a poté i načíst do softwaru v podobě balíčků obsahujících dané postupy. Instalace balíčku do prostředí R pak může probíhat ve dvou směrech (Li, 2008):

1. ze zdrojového souboru na disku počítače,
2. z online uložiště (například CRAN).

Celý proces zajišťují celkem dva základní balíčky utils a base, které jsou nachystány uvnitř programového řešení The R Project for Statistical Computing (R Core Team, 2014). Prostředníkem pro instalaci je zde funkce *install.packages*, která obsahuje dva klíčové argumenty, a to pkgs a repos (Burian, 2013). První zmíněný udává název daného balíčku a druhý název cílového uložiště. Pro načtení připraveného balíčku pak slouží funkce *library*, u které postačí vyplnit pouze jediný argument, a to název vstupního balíčku (Hornik, 2014).

7.1 Instalace ze souboru na disku

Tato cesta bývá volena v situaci, kdy daný balíček není k dispozici na žádném z dostupných online uložišť. Cílový soubor je tedy nutné stáhnout či přenést do vlastního počítače na disk a manuálně jej dohledat z prostředí softwaru (Li, 2008).

Jakmile bude balíček fyzicky přítomen na disku počítače uživatele, pak je možné přistoupit k jeho lokalizaci a instalaci do prostředí R. Tento krok zajistí výše zmíněna funkce, přičemž argumenty nabudou následujících hodnot (Burian, 2013; R Core Team, 2014):

- pkgs = file.choose(): definuje výchozí hodnotu pro název balíčku, zde zůstává prázdné kvůli volbě uživatele (uživateli se zobrazí okno pro výběr cílového adresáře s daným balíčkem a tím se i naplní argument),
- repos = NULL: hodnota NULL zde (jakožto uložiště) nastavuje harddisk uživatele.

7.2 Instalace z online uložiště

Častější variantou, pro rozšiřování funkčnosti softwaru The R Project for Statistical Computing, je instalace dostupných balíčků z online uložišť. Nejznámější archiv pro

R software je pak CRAN (Hornik, 2014). Pro získání balíčku právě z tohoto uložiště je nutné následně upravit vstupní argumenty funkce (R Core Team, 2014):

- pkgs = "název balíčku": uvedená hodnota v uvozovkách reprezentuje název daného cílového balíčku (například "gstat") uloženého ve vybraném uložišti (definovaného v následujícím argumentu),
- repos = "URL adresa uložiště": vyplnění argumentu pomocí platné URL (například "http://mirrors.nic.cz/R/") udáváme adresu cílového archivu, ze kterého bude požadovaný balíček stažen do programového prostředí.

7.3 Načtení balíčku

Po stažení a instalaci balíčku do prostředí R je nutné jej ještě načíst. Jedná se pouze o proces začlenění funkčního kódu do struktury softwaru. Samotná funkce *library* si v tomto případě vystačí pouze s jedním argumentem package, který definuje název balíčku připraveného k načtení (Hornik, 2014; R Core Team, 2014).

Programový kód 16: Příklad instalace balíčku gstat z online uložiště a jeho následné načtení do softwaru The R Project for Statistical Computing

```
install.packages(
                pkgs = "gstat",
                repos = "http://mirrors.nic.cz/R/"
                )

library(gstat)
```

8 VÝSLEDKY

Stěžejním výsledkem celé práce je nový R balíček Uncertainty Interpolation 2, který bude vystupovat v prostředí softwaru The R Project for Statistical Computing pod zkratkou UncerIn2. Jedná se tedy o upravenou (rozšířenou) verzi původního balíčku UncerIn (Burian, 2013). Tato nová, ale i původní, verze zmíněného balíčku pro software R bude dostupná na webové adrese (v sekci download) katedry geoinformatiky univerzity Palackého v Olomouci:

http://www.geoinformatics.upol.cz/dprace/magisterske/burian15//.

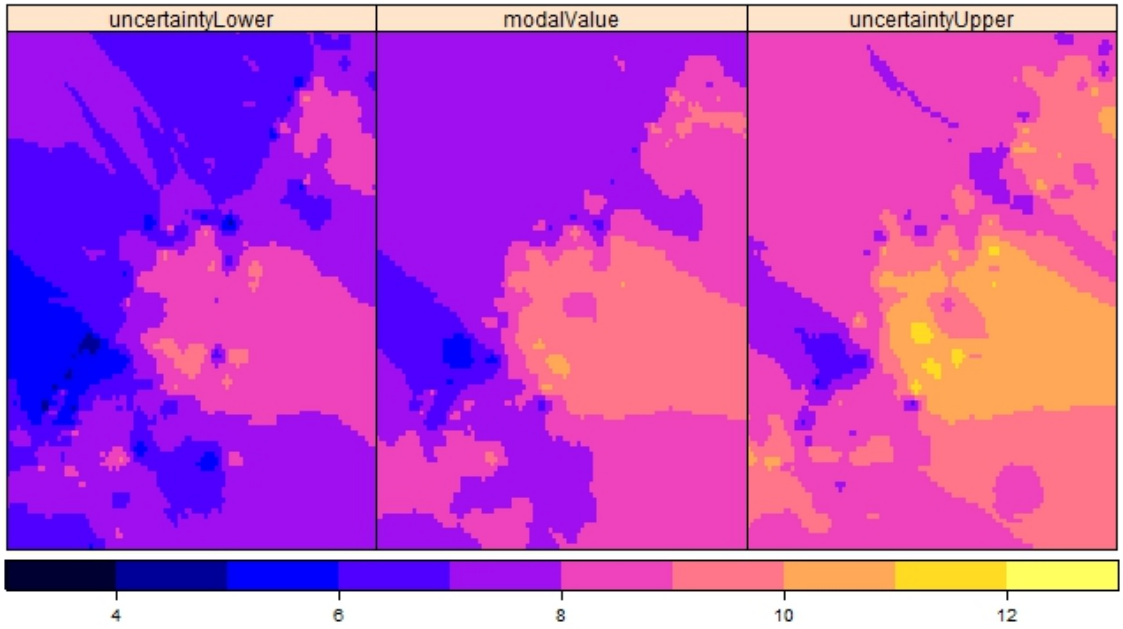
Kompletní balíček byl řádně okomentován a doplněn o ukázkový návod (tutorial), kde je prakticky popsáno jak jej případný uživatel může využít. Všechny funkce uvnitř balíčku navíc disponují nápovědou, která obsahuje popis daného procesu i jednotlivých argumentů. Zároveň jsou vždy i uvedeny exemplární formáty volacích funkcí pro případné použití.

Programový kód 17: Ukázka využití balíčku UncerIn2 nad datasetem meuse

```
# package UncerIn2 TUTORIAL

# dataset meuse import
data(meuse)

# S4 class object Points input data definition
points = Points(x = meuse$x, y = meuse$y, z = meuse$elev)

# building uncertainty model (based on uncertaintyError function)
uncertaintyModel = uncertaintyError(points)

# generate grid for IDW interpolation
gridDef.spat = Grid.def(points, TRUE) # SpatialPixels

# IDW interpolation process
IDW = idwUncertain(uncertaintyModel, gridDef.spat)

# visualization of results
Plot(IDW)
```

Ukázkový kód využití balíčku (viz výše) nabízí otestování funkcí na volně dostupném datasetu. Nejprve jsou zde data nahrána a naplněna vstupní třída dat *Points*. Poté přichází na řadu tvorba modelu nejistoty založeném na prostorově korelované chybě. Následuje vytvoření vyhovujícího gridu a jeho vstup, společně s modifikova-

nými daty, do interpolace IDW. Závěr pak tvoří plotovací funkce, která z výsledných dat vyprodukuje přehledný pohled na situaci (viz obr. 20).



Obrázek 20: Výsledný obraz interpolace dat zahrnující model nejistoty: vlevo vizualizace pro dolní mez (uncertaintyLower), uprostřed střední hodnota (modalValue) a vpravo obraz pro horní mez (uncertaintyUpper)

9 DISKUZE

Diplomová práce značně rozvinula funkčnost původního balíčku UncerIn (Burian, 2013), který měl relativně omezené možnosti téměř po všech stránkách využití. Bohužel nebylo možné analyzovat v podstatě jiná data, než která byla pevně definována na vstupu, samotný proces interpolací nepřipouštěl manipulaci s parametry a výsledky nebylo možné řádně vizualizovat. Všechna tato omezení však byla v rámci možností odstraněna a znamenitost případných výsledků této práce dokazuje i zvýšený zájem ze strany oslovených odborníků.

Algoritmy funkcí byly převedeny do nového formátu *S4*, který se na počátku práce objevil jako problém číslo jedna. Problematika přepisu funkcí však byla plně prozkoumána a pochopena, mimo jiné, i s odkazem na podniknutou vědeckou stáž v Německu. Možnosti práce a funkční vlastnosti přístupu *S4* pak umožnili snazší parametrizace většiny funkcí, i proto byly jednotlivé procesy interpolací rozepsány podle možností klíčových funkcí, které obsahují interpolační kalkulace. V potaz tedy nebyly brány pouze hlavní parametry ovlivňující interpolace, ale i většina dostupných argumentů pro funkční výpočty. Přidaný vizualizační nástroj do nového balíčku navíc přináší možnost praktického zobrazení dat (výsledků interpolací a variogramu), a to včetně prahů nejistoty.

Celá problematika nejistoty (a tím pádem i principu balíčku UncerIn2) se v reálném světě dostala během několika posledních let do popředí a je stále na vzestupu. Dnes lze s čistým svědomím tvrdit, že existuje řada odborníků, kteří se nejistotou nebo jejími aplikacemi (fuzzy teorie a podobně), doopravdy zabývají (například Caha, Gagolewski a další). Existuje tedy také řada jiných přístupů, jak pracovat s fenoménem zvaný nejistota. Zajímavý příspěvek představil Caha (2015), který přistoupil k propagaci nejistoty skrze fuzzy povrchy. Podle stejnojmenného autora byla do balíčku UncerIn2 implementována i zvláštní fuzzy varianta pro interpolační metodu kriging. V rámci výsledného balíčku je však samotný proces zatím omezený a představuje pouze nástavbu balíčku, jakožto další možný přístup v rámci softwaru The R Project for Statistical Computing, a to především pro více zkušené uživatele. Tímto směrem by se pak také mohla odvíjet i budoucí možná cesta pro další rozšiřování funkčnosti balíčku.

Kompletní R balíček Uncertainty Interpolation 2 tedy nyní nabízí určité možnosti pro vstup variabilních dat, tvorbu modelů nejistoty, interpolační procesy a dokonce i vizualizace výsledků. V celosvětovém měřítku se však jedná jen o praktický příklad využití metod pro práci s nejistotou.

9.1 Možnosti další práce

Vzhledem k prohlášení a faktu (v případě tématu propagace nejistoty s využitím fuzzy teorie), kdy tato problematika stále není zcela prozkoumána (Vondráková a Caha, 2014), lze také i nadále uvažovat o dalším možném rozšiřování a vylepšování funkčních vlastností balíčku UncerIn2. Možných variant v pokračování vývojové práce je pak nyní celá řada. Jedním ze směrů může být, již zmíněné, plné zakomponování funkcí fuzzy krigingu, kde se nabízí například rozvoj strukturalizace a vizualizace pro fuzzy povrchy. V návaznosti na implementaci externích postupů se také stále více otevírá varianta pro propojení balíčku s jinými externími zdroji (například R balíček FuzzyNumbers). V neposlední řadě je více než žádoucí publikace celého balíčku na online uložišti CRAN a zpřístupnit tak vytvořené nástroje široké veřejnosti.

10 ZÁVĚR

Všeobecným cílem diplomové práce bylo funkční rozšíření balíčku Uncertainty Interpolation (Burian, 2013) pro software R, který byl vytvořen v rámci předchozí bakalářské práce. Tato rozšíření měla představovat především přepis funkcí do objektové datové struktury *S4* a parametrizace funkčních procesů. Na počátku práce proto byla nejprve nastudována problematika programování struktur *S4* v softwaru The R Project for Statistical Computing, poté byly rozebrány jednotlivé funkce a v rámci možností proběhla jejich parametrizace. Veškeré poznatky byly řádně zaznamenány v rámci rešerše a společně pak tvořily základ pro vývoj celého výsledného balíčku.

Původní balíček byl tedy přepsán do nového datového formátu a jednotlivé funkce byly parametrizovány. Zároveň byl balíček rozšířen o alternativní přistup k interpolační metodě kriging (fuzzy kriging) a také o vizualizační funkce, které nabízí zobrazení výsledných dat. Nově jsou také přijímána do balíčku variabilní data, dále je uživateli nabídnuto až šest možností pro tvorbu gridu a přístup k modifikaci průběhu interpolačních funkcí. V celkovém součtu bylo dohromady vytvořeno 19 funkcí, které zajišťují vstup dat, transformace dat, tvorbu modelů nejistoty, generování gridu, interpolační procesy, odhad variogramu a vizualizace.

Stěžejní cíle diplomové práce tak byly splněny. Nad rámec povinností byl navíc původní balíček rozšířen v oblasti interpolací a doplněn o vizualizační nástroje. Výsledkem a důkazem tohoto prohlášení je pak nový balíček Uncertainty Interpolation 2 (ve zkratce UncerIn2), který je volně dostupný a funkční.

LITERATURA

Rstudio, 2014. Dostupné z: http://www.rstudio.com/products/rstudio/.

ABOYOUN, P. S4 system development in bioconductor, May 2010. Dostupné z: http://www.bioconductor.org/help/course-materials/2010/AdvancedR/S4InBioconductor.pdf.

ARNDT, J., HAENEL, C. *Pi Unleashed*. Springer-Verlag, 2006.

BLACK, K. *R Object-oriented Programming*. Packt Publishing - ebooks Account (October 23, 2014), 2014.

BURIAN, T. *Rozšíření interpolačních nástrojů v R project o modely nejistoty*. Univerzita Palackého v Olomouci, 2013.

CAHA, J. *Uncertainty Propagation in Fuzzy Surface Analysis*. Ph.d. thesis, Department of Geoinformatics, Palacký University in Olomouc, 2014.

CAHA, J. *Fuzzy Surface Analyses: First Experiments*. Olomouc : Terra Notitia, Palacký University for Department of Geoinformatics, 1st ed. edition, 2015. ISBN 978-80-244-4055-2.

CAHA, J., TUČEK, P., VONDRÁKOVÁ, A., PACLÍKOVÁ, L. Fuzzy Surface Models based on Kriging Outputs. In *GIS Ostrava 2012 Surface models for geosciences*, s. 25–36, Ostrava, 2012.

CAHA, J., MAREK, L., PÁSZTO, V. Spatial Prediction Using Uncertain Variogram. In BAJAT, B., KILIBARDA, M. (Ed.) *Proceedings of DailyMeteo.org/2014 Conference*, s. 20–24, Belgrade, 2014. Dosije studio doo. Dostupné z: http://dailymeteo.org/sites/default/files/DailyMeteo2014WEB.pdf.

CAHA, J., MAREK, L., DVORSKÝ, J. *(v tisku)* Predicting PM10 concentrations using fuzzy kriging. In *Hybrid Artificial Intelligent Systems*. Springer, 2015.

CELIKYILMAZ, A., TURKSEN, I. B. *Modeling uncertainty with fuzzy logic : with recent theory and applications*. Studies in fuzziness and soft computing. Berlin : Springer, 2009.

CHAMBERS, J. M. *Programming with Data A Guide to the S Language*. Springer-Verlag, 1998.

CRESSIE, N. A. C. *Statistics for Spatial Data*. JohnWiley & Sons, Inc., New York, 1991.

FISHER, P., CAHA, J. On Use of Fuzzy Surfaces to Detect Possible Elevation Change. In STEWART, K., PEBESMA, E., NAVRATIL, G., FOGLIARONI, P., DUCKHAM, M. (Ed.) *Extended Abstract Proceedings of the GIScience 2014*, č. 2007, s. 215–220, Vienna, Austria, 2014. Department of Geodesy and Geoinformation, Vienna University of Technology.

FISHER, P. F., TATE, N. J. Causes and consequences of error in digital elevation models. *Progress in Physical Geography*, 30, 4, s. 467–489, August 2006. ISSN 03091333.

GENOLINI, C. *A (Not So) Short Introduction to S4, Object Oriented Programming in R*. 2008.

GENTLEMAN, R. *S4 Classes in 15 pages, more or less*, February 2003. Dostupné z: `https://www.stat.auckland.ac.nz/S-Workshop/Gentleman/S4Objects.pdf`.

GOOVAERTS, P. *Geostatistics for natural resources evaluation*. Oxford University Press, 1997.

HADLEY WICKHAM, M. E. R. P. D. *In-Source Documentation for R*, April 2015. Dostupné z: `http://cran.r-project.org/web/packages/roxygen2/roxygen2.pdf`.

HELTON, J. C., OBERKAMPF, W. L. Alternative representations of epistemic uncertainty. *Reliability Engineering & System Safety*, 85, 1-3, s. 1–10, 2004.

HORNIK, K. R FAQ, 2014. Dostupné z: `http://CRAN.R-project.org/doc/FAQ/R-FAQ.html`.

KLIR, G. J., YUAN, B. *Fuzzy Sets and Fuzzy Logic: Theory and Applications*. Upper Saddle River : Prentice Hall, 1995.

KŘIKAVOVÁ, L. Interpolace bodových dat v gis. Master's thesis, České vysoké učení technické v Praze, 2009.

LI, L. *A Tutorial on R with Examples*. Department of Mathematics and Statistics, University of Saskatchewan, 106 Wiggins Road, MCLN 219 Saskatoon, SK, S7N 5E6, January 2008. Dostupné z: `http://math.usask.ca/~longhai/doc/others/R-tutorial.pdf`.

LODWICK, W., ANILE, M., SPINELLA, S. Introduction. In LODWICK, W. (Ed.) *Fuzzy surfaces in GIS and geographical analysis : theory, analytical methods, algorithms, and applications*, s. 1–46. Boca Raton : CRC Press, 2008. ISBN 9780849363955.

LOQUIN, K., DUBOIS, D. Kriging with Ill-Known Variogram and Data. In DESHPANDE, A., HUNTER, A. (Ed.) *Scalable Uncertainty Management*, 6379 / *Lecture Notes in Computer Science*, s. 219–235. Springer Berlin / Heidelberg, 2010. Dostupné z: http://dx.doi.org/10.1007/978-3-642-15951-0_23. ISBN 978-3-642-15950-3.

MASOOMI, Z., MESGARI, M. S., MENHAJ, M. B. Modeling uncertainties in sodium spatial dispersion using a computational intelligence-based kriging method. *Computers & Geosciences*, 37, 10, s. 1545–1554, 2011. doi: DOI10.1016/j.cageo.2011.02.002. Dostupné z: <GotoISI>://000295768900002.

MORGAN, M., GENTLEMAN, R. Lecture: S4 classes and methods, February 2008. Dostupné z: http://www.bioconductor.org/help/course-materials/2008/advanced_R/S4_Lecture.pdf.

PENG, R. Biostat 776: Other topics in r, November 2003. Dostupné z: http://www.biostat.jhsph.edu/~rpeng/docs/R-classes-scope.pdf.

PIOTROWSKI, J. a., BARTELS, F., SALSKI, A., SCHMIDT, G. Geostatistical regionalization of glacial aquitard thickness in northwestern Germany, based on fuzzy kriging. *Mathematical Geology*, 28, 4, s. 437–452, May 1996a. ISSN 0882-8121. doi: 10.1007/BF02083655. Dostupné z: http://link.springer.com/10.1007/BF02083655.

PIOTROWSKI, J. A., BARTELS, F., SALSKI, A., SCHMIDT, G. Estimation of hydrogeological parameters for groundwater modelling with fuzzy geostatistics: Closer to nature? In KOVAR, K., HEIJDE, P. (Ed.) *CALIBRATION AND RELIABILITY IN GROUNDWATER MODELLING*, č. 237, s. 511–520, GOLDEN, CO, 1996b. INT ASSOC HYDROLOGICAL SCIENCES.

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2014. Dostupné z: http://www.R-project.org/.

SANTOS, J. Surface modeling. In LODWICK, W. (Ed.) *Fuzzy surfaces in GIS and geographical analysis : theory, analytical methods, algorithms, and applications*, s. 85–104. CRC Press, 2008. ISBN 9780849363955.

SANTOS, J., LODWICK, W., NEUMAIER, A. A New Approach to Incorporate Uncertainty in Terrain Modeling. In EGENHOFER, M., MARK, D. (Ed.) *Geographic Information Science*, 2478 / *Lecture Notes in Computer Science*, s. 291–299. Springer Berlin Heidelberg, 2002. ISBN 978-3-540-44253-0.

SPINELLA, S., OSTOICH, M., ANILE, A. M. River water quality assessment with fuzzy interpolation. *ECOLOGICAL CHEMISTRY AND ENGINEERING S*, 15, 2, 2008.

STEIN, A., VERMA, M. Handling Spatial Data Uncertainty Using a Fuzzy Geostatistical Approach for Modelling Methane Emissions at the Island of Java. In FISHER, P. F. (Ed.) *Developments in Spatial Data Handling*, s. 173–187. Springer Berlin Heidelberg, 2005. doi: 10.1007/3-540-26772-7_14. Dostupné z: http://dx.doi.org/10.1007/3-540-26772-7_14. ISBN 978-3-540-22610-9.

SVOBODA, M. Norbert wiener citáty | citáty slavných osobností, 2014. Dostupné z: http://citaty.net/autori/norbert-wiener/.

SVOBODOVÁ, J. Hodnocení přesnosti digitálních modelů reliéfu. *Geomorphologia Slovaca et Bohemica*, s. 76–81, 2008. Dostupné z: http://www.asg.sav.sk/gfsb/v081/gfsb080109.pdf.

VIERTL, R. *Statistical methods for fuzzy data*. Chichester, West Sussex : Wiley, 2011.

VONDRÁKOVÁ, A., CAHA, J. Visualization of fuzzy surfaces. In *International Multidisciplinary Scientific GeoConference Surveying Geology and Mining Ecology Management, SGEM*, s. 1069–1076, Sofia, 2014. STEF92 Technology Ltd.

WAELDER, O. An application of the fuzzy theory in surface interpolation and surface deformation analysis. *Fuzzy Sets and Systems*, 158, 14, s. 1535–1545, July 2007.

WICKHAM, H. S4 advanced r., the s4 object system, 2015. Dostupné z: https://github.com/hadley/adv-r/blob/master/book/tex/S4.tex.

ZADEH, L. A. Outline of a New Approach to the Analysis of Complex Systems and Decision Processes. *EEE Transactions on Systems, Man and Cybernetics*, 3, 1, s. 16, 1973.

ZADEH, L. A. Discussion: Probability Theory and Fuzzy Logic Are Complementary Rather Than Competitive. *Technometrics*, 37, 3, s. 271–275, 1995.

PŘÍLOHY

Příloha 1 – DVD s daty, webovými stránkami a textem práce.