UNIVERZITA PARDUBICE

FAKULTA ELEKTROTECHNIKY A INFORMATIKY

BAKALÁŘSKÁ PRÁCE

2025 Mikhail Demchenko

Univerzita Pardubice
Fakulta Elektrotechniky a Informatiky

Robustní Odhady

Bakalářská práce

2025 Mikhail Demchenko

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: 2024/2025

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Mikhail Demchenko**
Osobní číslo:	**I21135**
Studijní program:	**B0688A140009 Informační technologie**
Téma práce:	**Robustní odhady**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování

V matematické statistice se často hledá odhad neznámého vektorového parametru a bývá také k dispozici varianční matice tohoto odhadu.Tato bakalářská práce se bude komplexně zabývat aplikací robustních odhadů ve statistických výpočtech se zvláštním zaměřením na robustní algoritmy lineární regrese. Studie bude zkoumat různé robustní metody odhadu, včetně mediánu, Huberových M-odhadů a trimovaného průměru, aby bylo možné zpracovat odlehlé hodnoty a zkreslená data. Kromě toho bude analyzovat výkon robustních lineárních regresních technik, jako je Theil-Sen a nejmenší střední hodnota čtverců (LMS), při zlepšování spolehlivosti a přesnosti statistických analýz. Snahou je prostřednitvím robustní statistické analýzy respektovat reálné datové scénáře.
Cílem práce je vytvořit aplikaci, která bude realizovat výpočet různých typů robustních odhadů. V práci budou popsány veškeré aplikované metody a bude vysvětleno uživatelské prostředí aplikace.

Rozsah pracovní zprávy: **45**
Rozsah grafických prací: **5**
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

WILCOX, Rand R. *Introduction to robust estimation and hypothesis testing*. 4th edition. Amsterdam: Elsevier, [2017]. ISBN 978-0-12-804733-0.
MARONNA, Ricardo A.; MARTIN, R. Douglas a YOHAI, Víctor J. *Robust statistics: theory and methods*. *Wiley series in probability and statistics*. Chichester: John Wiley, c2006. ISBN 0-470-01092-4.
ANTOCH, Jaromír a VORLÍČKOVÁ, Dana. *Vybrané metody statistické analýzy dat*. Praha: Academia, 1992. ISBN 80-200-0204-9.

Vedoucí bakalářské práce: **Mgr. Jaroslav Marek, Ph.D.**
Katedra automatizace a matematiky

Datum zadání bakalářské práce: **15. prosince 2024**
Termín odevzdání bakalářské práce: **16. května 2025**

L.S.

prof. Ing. Petr Doležel, Ph.D. v.r.
děkan

Ing. Jan Panuš, Ph.D. v.r.
vedoucí katedry

V Pardubicích dne 28. února 2025

Prohlašuji:

Práci s názvem Robustní odhady jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

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

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 7/2019 Pravidla pro odevzdávání, zveřejňování a formální úpravu závěrečných prací, ve znění pozdějších dodatků, bude práce zveřejněna prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 9. 5. 2025

Mikhail Demchenko v.r.

PODĚKOVÁNÍ

Rád bych poděkoval Mgr. Jaroslavu Markovi, Ph.D. za možnost pracovat na této závěrečné práci, která pro mě byla velmi zajímavá. Poskytl mi cenné rady, odpovídal na mé dotazy a nabídl relevantní materiály k prostudování. Také chci poděkovat svým rodičům za to, že mi poskytli tuto příležitost a podporovali mě během studií. V neposlední řadě děkuji svým přátelům, kteří mě podporovali v těžkých chvílích.

ANOTACE

Cílem této práce je implementovat aplikaci, která uživateli umožní řešit některé statistické problémy prostřednictvím robustních metod. Aplikace bude zahrnovat různé metody výpočtu mediánu a regresní přímky (Theil-Sen, metoda nejmenší střední hodnoty čtverců (LMS), trimovaný průměr, Huberova metoda). Aplikace bude zahrnovat možnost zadávání a importu dat, exportu dat a ukládání grafických výstupů.

KLÍČOVÁ SLOVA

C#, Avalonia UI, MVVM, robustní metody, robustní odhady, medián, regresní přímka, M-odhad, Huberův M-odhad, Theil-Sen odhad, LMS, desktopová aplikace

TITLE

Robust estimates

ANNOTATION

The aim of this work is to implement an application that allows the user to solve some statistical problems through robust methods. The application will include various methods of calculating the median and regression line (Theil-Sen, Least Mean Squares (LMS), trimmed mean, Huber's method). The application will include the ability to enter and import data, export data, and store graphical outputs.

KEYWORDS

C#, Avalonia UI, MVVM, robust estimators, robust methods, median, regression line, M-estimate, Huber’s M-estimate, Theil-sen estimate, LMS, desktop application.

OBSAH

SEZNAM ILUSTRACÍ

SEZNAM ZKRATEK A ZNAČEK

LMS – Least Median of Squares

OLS – Ordinary Least Squares

MAD – Median Absolute Deviation

OOP – Object Oriented Programming

UI – User Interface

MVVM – Model-View-ViewModel

XAML – Extensible Application Markup Language

CIL – Common Intermediate Language

JIT – Just-in-Time

MSIL – Microsoft Intermediate Language

TERMINOLOGIE

Influence function (vlivová funkce): funkce popisující, jak malé změny v datech (například přidání odlehlé hodnoty) ovlivní konečný odhad, tedy ukazuje citlivost metody na extrémy.

Outlier (odlehlá hodnota): jednotlivé měření, které se výrazně odlišuje od ostatních hodnot a může výrazně zkreslit výsledky klasických statistických metod.

Breakdown point: maximální podíl chybných nebo odlehlých dat, který metoda dokáže tolerovat, aniž by došlo k jejímu zhroucení (extrémnímu vychýlení výsledku); u mediánu a LMS je to až 50 %.

Aritmetický průměr (arithmetic mean): součet všech hodnot dělený jejich počtem; citlivý na outliery, protože každá hodnota má stejnou váhu.

Medián (median): prostřední hodnota v uspořádaném vzorku; odolný vůči extrémům

Trimovaný průměr (trimmed mean): průměr zbývajících hodnot po odstranění α % krajních pozorování na každém chvostu distribuce náhodné veličiny (typicky $\alpha = 5–25$ %).

Modus (mode): nejpravděpodobnější hodnota, která odpovídá argumentu lokálního maxima hustoty náhodné veličiny, existují i vícedílní rozdělení

M-estimát (M-estimate): zobecnění metody nejmenších čtverců, kde se minimalizuje součet robustní ztrátové funkce namísto kvadratické; například Huberova funkce.

Huberův M-odhad (Huber's M-estimator): robustní odhad, který kombinuje kvadratickou ztrátovou funkci pro malá rezidua a lineární pro velká, čímž omezuje vliv outlierů; tuning constant $c \approx 1{,}345$.

Theil–Senův odhad (Theil–Sen estimator): koeficient regresní přímky určený jako medián všech složených směrnic mezi dvojicemi bodů, velmi odolný, ale výpočetně náročný (složitost $O(n^2)$).

LMS (Least Median of Squares): metoda minimalizující medián čtverců reziduí místo jejich součtu, s breakdown point až 50 %, má vyšší výpočetní náročnost.

Reziduum (residual): rozdíl mezi pozorovanou hodnotou y_i a predikovanou hodnotou $\hat{y}_i$ modelu, klíčový pro výpočet regresních odhadů.

Koeficient determinace (R^2): podíl vysvětlené variability závislé proměnné modelem; hodnoty v intervalu [0,1], kde 1 znamená úplné vysvětlení rozptylu dat.

Střední kvadratická chyba (MSE – mean squared error): průměr čtverců rozdílů mezi pozorovanými a predikovanými hodnotami; běžná metrika kvality přizpůsobení modelu.

ÚVOD

Při statistické analýze dat se často setkáváme s jevy, které mohou narušovat platnost klasických (konvenčních) statistických metod. Jedná se zejména o výskyt odlehlých pozorování (outliers), silně asymetrická rozdělení, případně různé typy chyb v datech. Konvenční přístupy, jako je obyčejný aritmetický průměr pro bodový odhad střední hodnoty nebo metoda nejmenších čtverců pro odhad regresní přímky analýzy mohou být na takové výkyvy velmi citlivé. Proto se v posledních desetiletích dostávají do popředí **robustní metody**, jejichž cílem je poskytnout spolehlivější (odolnější, robustnější) výsledky i v případě, že data neodpovídají ideálním předpokladům.

Tato bakalářská práce se zabývá teoretickým přehledem a praktickou implementací vybraných robustních odhadů, které slouží především k omezení vlivu extrémních či chybných měření na výsledky analýzy. Konkrétně se zaměřuje na robustní odhady mediánu (M-odhad, Huberův M-odhad, Theil-Senův odhad) a robustní odhady v problému lineární regrese (nejmenší střední hodnota čtverců – LMS) a demonstruje výhody jejich využití oproti klasickým metodám. Praktickou součástí práce je návrh a realizace aplikace, která umožňuje uživateli výpočet vybraných robustních odhadů a případně jejich vizualizaci. V teoretické části budou popsány a srovnány principy robustních postupů dle dostupné odborné literatury, zejména podle práce Randa Wilcoxa [1]. V praktické části pak bude představeno konkrétní softwarové řešení, jeho uživatelské rozdemostrován výpočet na zvolených příkladech.

Zvolená problematika je relevantní jak pro akademické prostředí (kde bývají robustní metody často tématem výzkumu), tak i pro reálnou praxi v průmyslu, ekonomii či společenských vědách. Při analýze velkých a různorodých datasetů se totiž odlehlé hodnoty a nestandardní rozdělení dat vyskytují poměrně běžně. Pevně věřím, že tato práce nabídne ucelený pohled na výhody i omezení robustních přístupů a zároveň představí funkční software, jenž by mohl být využit jako nástroj pro zpracování dat robustními metodami.

1 TEORETICKÁ ČÁST

V této kapitole je podrobně rozebrána problematika robustních odhadů se zaměřením na jejich využití pro výpočet odhadu mediánu a odhadu regresní přímky. Nejprve jsou nastíněny klíčové vlastnosti klasických odhadů a jejich omezení v případech, kdy data obsahují odlehlé hodnoty. Následuje přehled základních principů robustní statistiky a popis vybraných robustních metod.

1.1 Klasické statistické odhady a jejich citlivost na odlehlé hodnoty

Tradiční statistické metody, mezi něž patří zejména aritmetický průměr pro bodový odhad střední hodnoty a ze složitějších problémů metoda nejmenších čtverců v regresní analýze (Ordinary Least Squares, OLS) vycházejí z předpokladu, že data pocházejí z relativně homogenního rozdělení bez výrazných odchylek. Pokud se ve vzorku vyskytne několik extrémních či chybných měření (outliers), může to vést k významnému zkreslení výsledků. Aritmetický průměr pak může poskytnout vychýlený odhad střední hodnoty. Metoda nejmenších čtverců je obzvláště náchylná k vychýlení, když i relativně malý počet extrémních pozorování může dramaticky ovlivnit odhad neznámých parametrů regresní přímky.

Citlivost konvenčních přístupů na odlehlé hodnoty je často spojována s tzv. **influence function** (vlivovou funkcí), která ukazuje, jak se výsledný odhad změní při malé změně v pozorovaných datech. Vysoká citlivost značí, že metodu lze snadno vychýlit i několika extrémními body, což je v mnoha reálných situacích nežádoucí.

1.2 Princip robustní statistiky

Robustní metody jsou navrženy tak, aby odhad zůstal relativně stabilní (nevychýlený) i při přítomnosti odlehlých pozorování nebo při porušení předpokladů o distribuci dat. Základním cílem robustní statistiky je snížit **vliv extrémních bodů** na výsledné odhady neznámých parametrů a poskytnout věrohodnější závěry o struktuře dat. Typicky se toho dosahuje nahrazením klasických kritérií (např. minimalizace součtu čtverců) odolnějšími přístupy (např. minimalizace upravené ztrátové funkce), využitím odhadů mediánu namísto průměru, případně ořezáním extrémních 5 % či 10 % hodnot (trimování) před výpočtem základních statistik.

Podle Wilcoxa [1] lze robustní přístupy definovat několika různými způsoby, přičemž jedním z oblíbených kritérií je tzv. **breakdown point** – největší podíl chybných nebo odlehlých dat, který metoda zvládne tolerovat, aniž by došlo k naprostému zhroucení odhadu (např. extrémní

vychýlení výsledku). Dalším důležitým pojmem je **odolnost vůči porušení normality**, kde robustní metody často poskytují lepší výsledky než klasické testy a odhady založené na předpokladu Gaussianu.

Rousseeuw a Leroy [2] dále poukazují na to, že robustní metody mohou účinně redukovat vliv odlehlých hodnot i v případech, kdy je podíl těchto odlehlých pozorování relativně vysoký.

1.3 Klasické odhady polohy

Odhady polohy slouží k určení „středu“ rozdělení dat. Mezi základní míry polohy patří střední hodnota, medián a modus, jejichž výpočet spadá do kategorie **klasických odhadů** (citlivé na odlehlé hodnoty).

1.3.1 Aritmetický průměr

Definice:

$\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$

Tento statistický odhad není odolný vůči extrémním hodnotám, protože každé x_i přispívá při jeho výpočtu stejnou vahou.

1.3.2 Medián

Nechť $x_{(1)} \leq x_{(2)} \leq \cdots \leq x_{(n)}$: jsou uspořádaná pozorování.

$$\widetilde{x} = \begin{cases} x_{\left(\frac{n+1}{2}\right)}, & n \text{ liche}, \\ \frac{1}{2}\left(x_{(\frac{n}{2})} + x_{(\frac{n}{2}+1)}\right), & n \text{ sude}. \end{cases}$$

Breakdown point: Uvažují se hodnoty až do 50 %.

Pro účinnější výpočet mediánu lze použít *Quickselect* nebo pro hladší odhad kvantilu *Harrell–Davisův odhad*:

$$\hat{x}_{0.5} = \sum_{i=1}^{n} w_{i,n}\, x_{(i)}, w_{i,n} = F_{\text{Beta}}\left(\tfrac{i}{n}; \tfrac{n+1}{2}, \tfrac{n+1}{2}\right) - F_{\text{Beta}}\left(\tfrac{i-1}{n}; \tfrac{n+1}{2}, \tfrac{n+1}{2}\right).$$

Kde:

$F_{Beta}(x, \beta, \alpha)$ je distribuční funkce (CDF) Beta rozdělení se dvěma parametry tvaru α a β.

První argument x je bod, ve kterém CDF vyhodnocujeme.

Druhý parametr α a třetí argument β. určují "tvar" Beta rozdělení.

Rozdíl dvou hodnot CDF na úsecích dává pravděpodobnost, že fiktivní náhodná veličina leží právě v tomto intervalu.

Celý odhad pak je váženým průměr uspořádaných hodnot.

1.3.3 Trimovaný průměr

Odstraní se α-procent z každého konce a spočte se průměr zbývajících hodnot.

$$\overline{x}_{(\alpha)} = \frac{1}{n-2k} \sum_{i=k+1}^{n-k} x_{(i)}, k = \lfloor \alpha\, n \rfloor.$$

Volba (breakdown point) α je typicky 5 % až 25 %.

1.3.4 Modus

Namísto vyřazení krajních hodnot se „ořezávají“ na nejbližší přeživší extrémy.

$$x^*_{(i)} = \begin{cases} x_{(k+1)}, & i \le k, \\ x_{(i)}, & k < i \le n-k, \\ x_{(n-k)}, & i > n-k, \end{cases} \quad \overline{x}_{\mathrm{w}} = \frac{1}{n} \sum_{i=1}^{n} x^*_{(i)}.$$

U tohoto odhadu dochází k mírnějšímu potlačení outlierů než u trimovaného průměru.

1.4 Vybrané robustní metody odhadu polohy

V této sekci jsou podrobněji představeny čtyři vybrané robustní postupy: Huberovy M-odhady, trimovaný průměr (příp. medián), Theil-Senův odhad a metoda nejmenší střední hodnoty čtverců (Least Median of Squares, LMS). Každý z těchto přístupů má své specifické vlastnosti, výhody i nevýhody, které budou dále diskutovány.

1.4.1 Kvalita aproximace

V regresní analýze se dnes dominantně používá metoda nejmenších čtverců, jejíž autorství se připisuje německému matematikovi Johannu (Carli) Friedrichu Gaussovi (1777 – 1855).
Často studovaný problém je odhad neznámých parametrů regresní přímky
$\mathrm{yi} = \beta_0 + \beta_1 \mathrm{x_i}$.

Podrobnosti o regresní analýze a metodě nejmenších čtverců jsou uvedeny v knize Jiřího Anděla [3].

Metoda nejmenších čtverců není jedinou možností. V polovině 18. století metodu založenou na minimalizaci sumy absolutní hodnoty korekcí navrhl Roger Boškovič. Viz Škrabanek [4] Existují i další přístupy, například Lambertova přímka. Viz Stigler [5], Marek [6].

Kvalita aproximace modelu se nejčastěji měří pomocí koeficientu determinace R^2, který udává, jakou část celkového rozptylu závislé proměnné model vysvětluje.

Nechť

y_i, jsou pozorované hodnoty,

$\hat{y}_i$, jsou predikce modelu,

$\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$, je jejich průměr.

Dále definujeme:

$$\mathrm{SST} = \sum_{i=1}^{n}(y_i - \bar{y})^2\,,\ \mathrm{SSE} = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Pak koeficient determinace vyjádříme jako

$$\mathrm{R}^2 = 1 - \frac{\mathrm{SSE}}{\mathrm{SST}}.$$

Hodnota R^2 nabývá rozsahu od 0 do 1, kde $R^2 = 1$ znamená, že model přesně vysvětluje všechna data, a $R^2 = 0$ že nevysvětluje žádný rozptyl.

Pro zohlednění počtu vysvětlujících proměnných se často používá tzv. upravený koeficient determinace

$$R^2_{\mathrm{adj}} = 1 - (1 - R^2)\,\frac{n-1}{n-p-1},$$

kde p je počet nezávislých proměnných v regresním modelu (bez interceptu). Tento ukazatel penalizuje nadbytečné proměnné, které nepřispívají k lepší predikci.

V praxi se kromě R^2 a $\mathrm{R}^2_{\mathrm{adj}}$ může hodnotit kvalita aproximace i pomocí střední kvadratické chyby

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2},$$

či dalších metrik, nicméně koeficient determinace zůstává nejrozšířenějším ukazatelem vysvětlitelnosti rozptylu dat.

1.4.2 Huberovy M-odhady

Huber [7] navrhl tzv. M-odhady jako zobecnění klasické metody nejmenších čtverců.

Základní idea spočívá v náhradě kvadratické ztrátové funkce

$$\rho(u) = \frac{1}{2}u^2$$

novou, tzv. Huberovou funkcí, která je pro malá rezidua (tj. $|u|$ pod určitým prahem c) stejná jako kvadratická, ale pro velká rezidua roste již lineárněji, a tím omezuje vliv extrémních hodnot.

Matematicky lze tento přístup v regresi vyjádřit minimalizací

$$\min_{\beta} \sum_{i=1}^{n} \rho!\left(\frac{r_i}{\sigma}\right)$$

Kde

$$r_i = y_i - x_i^T$$

značí reziduum, přičemž

$$x_i^T \beta \; = \sum_{j=0}^{p} x_{ij} \; \beta_j$$

pro vektor vysvětlujících proměnných
$x_i = (1, x_{i1}, \ldots, x_{ip})^T$,

a vektor parametrů
$\beta = (\beta_0, \beta_1, \ldots, \beta_p)^T$,

kde operátor T označuje transpozici (tedy převrácení řádkového vektoru na sloupcový a naopak).

Konstanta c (tzv. tuning constant) se často volí kolem 1,345, což zajišťuje vysokou účinnost odhadu pro data blízká normálnímu rozdělení.

Parametr σ může představovat odhad směrodatné odchylky či robustní míru rozptýlení (např. MAD).

Výsledkem optimalizace je vektor β, který je méně citlivý na extrémní odchylky než klasická OLS regrese.

1.4.3 Theil-Senův odhad

Theil-Senův odhad [8,9] je robustní metoda určování regresní přímky v modelu $y = \beta_0 + \beta_1 x$.

Základní myšlenka spočívá v tom, že koeficient

β_1

je určen jako **medián** všech dílčích směrnic

$\frac{y_j - y_i}{x_j - x_i}, \quad i < j,\ x_j \neq x_i,\ \text{kde: } i < j.$

Následně se odhaduje průsečík β_0 jako medián z hodnot

$(y_i - \beta_1 x_i)$.

Tímto způsobem je eliminován vliv malého počtu extrémních bodů, které by v OLS metodě mohly silně ovlivnit regresní přímku.

Výhodou Theil-Senova odhadu je poměrně jednoduchá koncepce a vysoká robustnost vůči odlehlým datům.

Nevýhodou může být výpočetní náročnost, jelikož je potřeba pro n bodů vytvářet $\frac{n(n-1)}{2}$ možných dvojic.

Při velkém n se proto používají efektivnější algoritmy (např. randomizované nebo aproximační metody).

Poznámka: Při výpočtu Theil–Senova sklonu se medián nachází nad množinou dílčích směrnic. Pro efektivní výběr „prostřední“ hodnoty lze použít výše zmíněné metody (Quickselect nebo Harrell–Davisův odhad mediánu), což zvyšuje numerickou stabilitu a rychlost při velkých datech.

1.4.4 Metoda nejmenší střední hodnoty čtverců (LMS)

Dalším známým robustním postupem v regresi je tzv. **Least Median of Squares (LMS)**, který navrhli Rousseeuw a Leroy. Oproti klasickým nejmenším čtvercům, které minimalizují součet čtverců reziduí $\sum r_i^2$, metoda LMS minimalizuje **medián** čtverců reziduí: $\min_{\beta} \text{Med}(r_i^2)$.

Díky použití mediánu místo součtu má LMS mnohem vyšší odolnost vůči odlehlým hodnotám – extrémní residuum ovlivní cíl minimalizace pouze v té míře, jak ovlivní pořadí čtverců reziduí. Rousseeuw a Leroy dokonce ukázali, že LMS vykazuje **breakdown point až 50 %**, což z ní činí jednu z nejodolnějších regresních metod. Na druhou stranu bývá LMS algoritmicky náročná a v čisté podobě může fungovat pomaleji než OLS pro větší datové soubory.

1.5 Důvody pro využití robustních metod a jejich omezení

Z předchozích kapitol je zřejmé, že robustní metody umožňují spolehlivější odhady a regresní modely v případě, že data obsahují odlehlé hodnoty nebo nejsou symetricky rozložena. Jejich uplatnění však s sebou nese i jistá **omezení**, a proto je vhodné vždy pečlivě volit konkrétní přístup podle charakteru dat a cílů analýzy. Následující body shrnují klíčová **pozitiva a negativa** robustních postupů s ohledem na praktické využití.

1.5.1 Výhody robustních metod

- **Odolnost vůči odlehlým pozorováním**
  Robustní odhady udržují výsledky stabilní i v situaci, kdy se ve vzorku vyskytuje několik extrémních (chybných) bodů. Tím zabraňují nadměrnému zkreslení, které by u klasických metod (např. metoda nejmenších čtverců) mohlo výrazně ovlivnit celkovou regresi či bodový odhad.

- **Větší flexibilita v reálných podmínkách**
  V mnoha oborech (ekonomie, technika, společenské vědy) se setkáváme s distribucemi, jež nejsou normální a obsahují výrazné asymetrie. Robustní metody jsou v takových případech častokrát spolehlivější a méně vyžadují „čistotu“ dat.

- **Vysoký breakdown point**
  Některé robustní přístupy (např. medián, LMS) mají breakdown point až 50 %, což znamená, že dokážou tolerovat relativně vysoký podíl odlehlých měření, aniž by došlo k úplnému selhání odhadu.

- **Využitelnost v různých úlohách**
  Princip robustnosti lze aplikovat nejen v lineární regresi, ale i v dalších oblastech

statistiky: odhady polohy, míry rozptýlení (MAD), robustní testy hypotéz a pod.

1.5.2 Omezení a slabé stránky

- **Vyšší výpočetní náročnost**
  Metody jako Theil-Sen nebo LMS mohou mít algoritmickou složitost O(n2), O(n^2), O(n2) či vyšší, což je problém zejména pro velké datové soubory. V praxi se využívají aproximační nebo iterativní algoritmy, aby bylo možné zpracovávat rozsáhlé datasety rychleji.

- **Nutnost volby parametrů**
  U některých robustních metod je třeba volit tzv. *tuning constants* (např. konstanta *c* u Huberovy funkce), a jejich špatné nastavení může vést k menší účinnosti odhadu. Trimovaný průměr zase vyžaduje rozhodnutí o velikosti trimování α.

- **Nižší efektivita při ideálních podmínkách**
  Pokud data skutečně pocházejí z čistě normálního rozdělení a neobsahují žádné extrémy, některé robustní odhady (např. medián, LMS) mohou mít vyšší rozptyl než klasická OLS nebo aritmetický průměr. To znamená, že při „dokonale“ normálním rozdělení jsou tradiční metody často mírně účinnější.

- **Komplikovanější interpretace**
  Robustní modely mohou pro uživatele postrádat přehlednost klasických postupů, protože vyžadují detailnější vysvětlení principu robustní ztrátové funkce či ořezávání dat. Pro správnou interpretaci výsledků je nutné vědět, jakým způsobem byly odlehlé body potlačeny a jak se to promítlo do závěrů.

- **Závislost na kvalitě odhadu míry rozptýlení**
  Některé robustní regresní metody (např. Huberovy M-odhady) předpokládají, že máme spolehlivou robustní míru rozptýlení (např. MAD). Pokud je tento krok proveden nevhodně, odhady koeficientů mohou být zkreslené.

1.5.3 Doporučení pro výběr metody

Volba konkrétního robustního postupu závisí na množství a struktuře dat, očekávané míře přítomnosti odlehlých bodů a požadavcích na rychlost výpočtu. Ve většině případů se doporučuje kombinovat robustní odhady s předběžnou datovou analýzou (tzv. *exploratory data analysis*), díky níž lze identifikovat extrémní měření či atypická rozdělení. Tím se minimalizuje riziko, že vybraný robustní přístup bude nedostatečný nebo naopak příliš agresivní.

1.6 Shrnutí

Teoreticky je zřejmé, že robustní metody dokážou zvládnout situace, kde klasické odhady selhávají. Jejich zavedení však vyžaduje zohlednění možné vyšší výpočetní složitosti a nutnost pečlivého výběru konkrétního robustního postupu dle povahy dat (např. podíl odlehlých hodnot, velikost vzorku). Následující část práce se zaměří na praktickou implementaci vybraných robustních odhadů a porovnání výsledků s tradičními přístupy.

2 PRAKTICKÁ ČÁST

V této kapitole bude podrobně popsána realizace aplikace, jež umožňuje provádět výpočty nad zadanými číselnými sériemi (datasets). Zaměříme se na volbu programovacího jazyka, architekturu aplikace, použitý pattern návrhu (MVVM), klíčové úseky zdrojového kódu a vysvětlíme funkčnost i uživatelské rozhraní vytvořeného softwaru.

2.1 Programovací jazyky

2.1.1 Jazyk C#

C# je moderní programovací jazyk spravovaný platformou .NET, který poprvé představila společnost Microsoft na přelomu 20. a 21. století [10]. Inspiroval se jazyky C++ a Java, a proto přejímá jejich klíčové rysy – objektově orientovaný přístup, silnou typovou kontrolu i bohatou standardní knihovnu. Díky tomu poskytuje vývojářům nástroje, jež usnadňují tvorbu kódu od malých projektů až po rozsáhlé softwarové systémy.

V rámci **procesu kompilace** je zdrojový kód v C# nejprve převeden do mezipřekladového jazyka (MSIL/CIL), jenž je nezávislý na konkrétním operačním systému a procesoru. Za běhu aplikace se potom tento mezipřeklad překládá pomocí technologie **JIT (Just-In-Time)** do nativního strojového kódu přizpůsobeného danému hardware [11]. Touto vrstvou abstrakce se docílí přenositelnosti aplikací a zároveň možnosti optimalizace výkonu na různých platformách.

C# podporuje více programovacích paradigmat, přesto se nejčastěji používá v **objektově orientovaném** stylu. Ten umožňuje třídit logiku do tříd a objektů, zjednodušuje opakované využití kódu a přispívá k lepší čitelnosti rozsáhlých projektů. Vedle OOP nabízí C# podporu pro **asynchronní programování** (klíčová slova async/await) i pro funkcionální konstrukce (např. LINQ). Také klade důraz na bezpečnost, díky správě paměti pomocí **garbage collectoru** eliminuje problémy s manuální alokací a dealokací [12].

Mezi praktické výhody C# patří rozsáhlý ekosystém knihoven a nástrojů, včetně frameworků pro vývoj **desktopových**, **webových** či **mobilních aplikací**. Vzhledem k podpoře multiplatformních řešení (např. .NET Core, .NET 8, Avalonia UI) už C# nemusí být vázán čistě na operační systém Windows. V této práci bylo rozhodnuto použít C# díky přehledné syntaxi, bohaté dokumentaci a snadné integraci s dalšími technologiemi v rámci platformy .NET.

2.1.2 Jazyk XAML

XAML (Extensible Application Markup Language) je deklarativní značkovací jazyk, jehož hlavním smyslem je **definice uživatelského rozhraní** a s ním souvisejících prvků. Vzešel původně z technologie **WPF (Windows Presentation Foundation)**, ale dnes se používá i v jiných multiplatformních řešeních, včetně knihovny **Avalonia UI** [13].

Díky XAML lze rozdělit design aplikace od její logiky. V souborech .xaml se definují jednotlivá okna, panely, tlačítka či formuláře, zatímco kód v C# (tzv. *code-behind*) zpracovává uživatelské akce, řeší komunikaci s datovou vrstvou apod. Tato separace přispívá k lepší udržovatelnosti kódu, protože vývojáři mohou odděleně pracovat na vzhledu a chování aplikace.

XAML výrazně **podporuje vzor MVVM** (Model-View-ViewModel). V praxi to funguje tak, že definice GUI (View) je čistě v .xaml, ViewModel drží data a aplikační logiku a propojení mezi nimi je řízeno mechanismem *data binding*. Tímto způsobem si lze udržet jasnou strukturu kódu, využít testování na úrovni ViewModelu a v případě potřeby snadno měnit či upravovat design.

Jednou z předností XAML je také možnost používat **styly a šablony**. Pomocí nich se dá sjednotit vzhled všech ovládacích prvků v aplikaci nebo flexibilně přizpůsobit rozhraní konkrétním potřebám. Samotné úpravy UI je pak možné provádět buď ruční editací souborů .xaml, nebo vizuálními nástroji (jako například Visual Studio Designer), které ukazují náhled aplikace v reálném čase.

V kontextu této aplikace byl XAML zvolen jako nástroj k vytvoření přehledného a snadno udržovatelného uživatelského rozhraní. Deklarativní popis v XAML eliminuje nutnost psát velké objemy kódu pro práci s UI přímo v C#, a současně zůstává kompatibilní se zásadami multiplatformního vývoje v Avalonia UI.

2.2 Framework Avalonia UI

V této sekci se zaměříme na **framework Avalonia UI**, který zprostředkovává multiplatformní vývoj desktopových aplikací s využitím konceptů známých z WPF. Hlavní výhodou Avalonia je možnost vytvářet aplikace v C# a XAML, jež lze bez větších úprav spustit na Windows, Linux i macOS [14].

2.2.1 Základní charakteristika a architektura

Avalonia UI je open-source projekt inspirovaný WPF, a proto využívá podobné mechanismy, jako jsou *data binding*, *styly*, *šablony* a *resources*. Rozdíl však spočívá v multiplatformním renderingovém jádru, které se stará o vykreslování ovládacích prvků na různých operačních systémech. Samotná architektura Avalonia je navržena modulárně: jádro frameworku funguje jako abstraktní vrstva pro vykreslování, zatímco konkrétní implementace pro daný OS (např. Direct2D pro Windows nebo platforma X11 pro Linux) se řeší pomocí specializovaných modulů.

2.2.2 Podpora MVVM a XAML

Podobně jako WPF sází Avalonia na pattern **MVVM**, což znamená, že rozhraní (View) je deklarováno v *.xaml* a aplikační logika (ViewModel) je implementována v C#. Binding, příkazy (commands) a notifikace (např. INotifyPropertyChanged) pracují téměř shodně jako v klasické WPF aplikaci, díky čemuž mohou vývojáři snadno přecházet mezi oběma frameworky.

2.2.3 Důvody využití v této práci

- **Multiplatformní charakter**: Požadavek spouštět aplikaci na různých operačních systémech (Windows, Linux, macOS) vedl k volbě technologie, jež tuto vlastnost přirozeně podporuje.
- **Deklarativní popis UI**: Pro tvorbu uživatelského rozhraní postačí XAML, takže lze oddělit vzhled a aplikační logiku, což usnadňuje údržbu i testování.
- **Podobnost s WPF**: Pokud má vývojář zkušenost s WPF, dokáže velmi rychle přejít na Avalonia UI, jelikož základní principy (styly, šablony, data binding) zůstávají podobné.
- **Kompatibilita s .NET**: Avalonia pracuje nad platformou .NET (aktuálně .NET 7/8), takže lze bezproblémově využívat knihovny a nástroje z ekosystému C#.

2.2.4 Stručné zhodnocení

Avalonia UI poskytuje stabilní základ pro tvorbu multiplatformních desktopových aplikací, přičemž v porovnání s WPF či UWP není vázána na konkrétní operační systém. V této práci plní roli hlavního nástroje pro grafické rozhraní a je logickou volbou kvůli flexibilitě a

podpoře standardního XAML. Ve spojení s jazykem C# a architekturou MVVM tak umožňuje vytvářet čitelné a přenositelně navržené aplikace.

2.3 Architektura aplikace MVVM

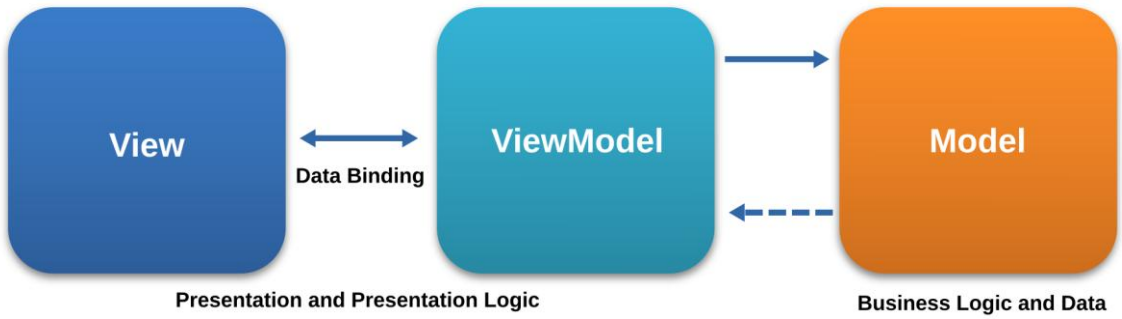
V rámci této kapitoly bude představena architektura **Model–View–ViewModel (MVVM)**, která byla v aplikaci zvolena pro oddělení prezentační logiky od samotných dat a jejich zpracování. Tento vzor se hojně používá v kontextu uživatelských rozhraní založených na XAML, například ve WPF či Avalonia UI [15].

2.3.1 Princip MVVM

Architektura MVVM je variací na klasický pattern MVC (*Model–View–Controller*), avšak přizpůsobený požadavkům deklarativního popisu uživatelského rozhraní. Základní idea MVVM je:

1. **Model (M)**
   Model představuje doménová data a jejich logiku. Může obsahovat třídy pro práci s daty, metody pro načítání či ukládání informací a obecně vše, co souvisí s aplikační logikou. V robustnějších projektech může Model zahrnovat i vrstvy pro komunikaci s databází či webovými API.
2. **View (V)**
   View je zodpovědné za **prezentaci** dat koncovému uživateli. V případě aplikací postavených na XAML se jedná o soubory *.xaml*, v nichž je definován vzhled oken, ovládací prvky (tlačítka, textová pole, tabulky) a layout. View v ideálním případě neobsahuje aplikační logiku – pouze deklaruje, jak se mají data zobrazovat.
3. **ViewModel (VM)**
   ViewModel tvoří jakési „lepidlo“ mezi View a Modelem. Obsahuje vlastnosti, kolekce a příkazy (*commands*), které jsou bindovány k prvkům uživatelského rozhraní. Pokud uživatel změní data v rozhraní (např. vyplní textové pole), ViewModel tuto změnu zachytí a zaktualizuje Model. Naopak, pokud se Model změní (např. dojde k načtení nových dat), ViewModel informuje View, aby zobrazení aktualizovalo.

Díky tomuto uspořádání je uživatelské rozhraní (View) odtrženo od detailů implementace (Model), což usnadňuje testování, údržbu i možnost rychlého redesignu bez nutnosti měnit logiku celé aplikace.



Obrázek 1: Ukázka principu architektury MVVM [17]

2.3.2 Základní mechanismy

V MVVM bývají klíčové následující principy:

- **Data binding**
  Pomocí *data bindingu* se propojují vlastnosti ViewModelu (např. `public string Username { get; set; }`) s prvky rozhraní (např. `<TextBox Text="{Binding Username}"/>`). Kdykoli se hodnota vlastnosti změní, View se automaticky aktualizuje a naopak.
- **Notifications (INotifyPropertyChanged)**
  Aby binding fungoval, musí ViewModel implementovat rozhraní `INotifyPropertyChanged`. Pomocí události `PropertyChanged` dává najevo, že se nějaká vlastnost změnila a View na to může reagovat.
- **Commands**
  Místo klasických *event handlers* (např. `Button_Click`) využívá MVVM příkazy (např. `ICommand`), které obsahují logiku pro zpracování akcí uživatele. Tím se minimalizuje závislost uživatelského rozhraní na konkrétních metodách v code-behind a usnadňuje se testování, jelikož příkaz lze volat i bez spuštěného UI.
- **ViewModel-lokální logika**
  ViewModel často obsahuje i drobné logické operace (např. validaci vstupu), které nejsou přímo doménovou logikou (tu řeší Model), ale přímo se týkají uživatelského rozhraní (např. zobrazení chybové hlášky).

2.3.3 Využití MVVM v této aplikaci

V této práci je architektura MVVM použita z několika důvodů:

1. **Čisté oddělení uživatelského rozhraní a logiky**
   Díky MVVM je možné koncentrovat vizuální definice do XAML (View) a zpracování dat (Model) a stav aplikace (ViewModel) do samostatných tříd. To přispívá k lepší srozumitelnosti a škálovatelnosti kódu.
2. **Snadné testování**
   Protože ViewModel je samostatná třída nezávislá na UI, lze jej testovat pomocí jednotkových (unit) testů, aniž by byla potřeba spouštět grafické rozhraní.
3. **Podpora XAML frameworků**
   Avalonia UI, podobně jako WPF, umožňuje rychlý binding dat na vlastnosti ViewModelu i používání stylů a šablon. Získáváme tak výhodu deklarativního přístupu, kdy se většina interakce mezi uživatelem a aplikačním stavem řeší automaticky přes binding.
4. **Modulární rozšíření**
   Pokud je potřeba do aplikace přidat nový modul (např. další okno, stránku nebo funkci), stačí vytvořit novou View + ViewModel + případně doplnit odpovídající třídy v Modelu. Nezasahujeme přitom do existující struktury mimo nezbytné vazby.

Díky těmto vlastnostem se MVVM stává vhodným řešením pro všechny aplikace, kde je kladen důraz na čistotu kódu, testovatelnost a možnost rychlého rozvoje. Následující kapitoly se již budou zabývat konkrétní implementací této architektury, zejména detailnějším rozborem tříd ViewModel, komunikací s datovou vrstvou a ukázkami bindingu v XAML.

2.4. Pomocná knihovna OxyPlot

Pro grafickou vizualizaci statistických výstupů (například zobrazení datových bodů, regresní křivky či jiných veličin) byla do projektu zařazena knihovna **OxyPlot**. Jedná se o open-source řešení, které umožňuje kreslit různé typy grafů (lineární, bodové, sloupcové atd.) v prostředí .NET [18]. OxyPlot je známý svou jednoduchou konfigurací, relativně nízkou závislostí na dalších balíčcích a dobrou dokumentací.

2.4.1 Důvod použití verze OxyPlot.AvaloniaCore

Vzhledem k tomu, že tato aplikace je vyvíjena v **Avalonia UI** verze vyšší než 0.10, nebylo možné použít balíček OxyPlot.Avalonia, který není s novějšími verzemi Avalonie plně kompatibilní. Proto se využívá knihovna **OxyPlot.AvaloniaCore**, která nabízí podporu pro modernější verze Avalonie.

Instalace probíhá např. prostřednictvím NuGet balíčku: Install-Package OxyPlot.AvaloniaCore nebo úpravou souboru .csproj. Tato knihovna poskytuje speciální ovládací prvek (control) pro vykreslování grafů v XAML a funkce pro konfiguraci série (Series), os (Axes) či legendy.

2.4.2 Integrace s aplikací

Po přidání balíčku do projektu lze v XAML deklarovat ovládací prvek typu PlotView (respektive AvaloniaPlot), jenž se váže na instanci třídy PlotModel z OxyPlotu. Architektura MVVM zůstává zachována – logika pro sestavení grafu (např. Series, Axes, Title) je umístěna ve ViewModelu, zatímco samotné vykreslování obsluhuje OxyPlot.AvaloniaCore v prezentační vrstvě.

Zjednodušený příklad XAML kódu:

```
<UserControl xmlns="https://github.com/avaloniaui"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
xmlns:oxy="clr-namespace:OxyPlot.Avalonia;assembly=OxyPlot.AvaloniaCore">
    <oxy:PlotView Model="{Binding MyPlotModel}" />
</UserControl>
```

kde MyPlotModel je vlastnost ve ViewModelu typu PlotModel.

2.4.3 Ukázka typického použití

V praxi lze OxyPlot využít pro:

- **Zobrazení měřených dat** – např. body ze senzoru;

- **Regresní křivku** – pokud jsme pomocí robustních metod spočítali regresní přímku, můžeme ji vykreslit do stejného grafu jako datové body;
- **Histogram nebo BoxPlot** – pro základní statistický přehled rozdělení dat (pokud je implementováno v rámci OxyPlotu nebo vlastních rozšíření).

V PlotModel lze definovat:

```
var model = new PlotModel { Title = "Robustní regrese" };

var series = new LineSeries

{

   Title = "Regresní křivka",

   ItemsSource = dataPoints, // seznam bodů pro vykreslení

   DataFieldX = "X",

   DataFieldY = "Y"

};

model.Series.Add(series);
```

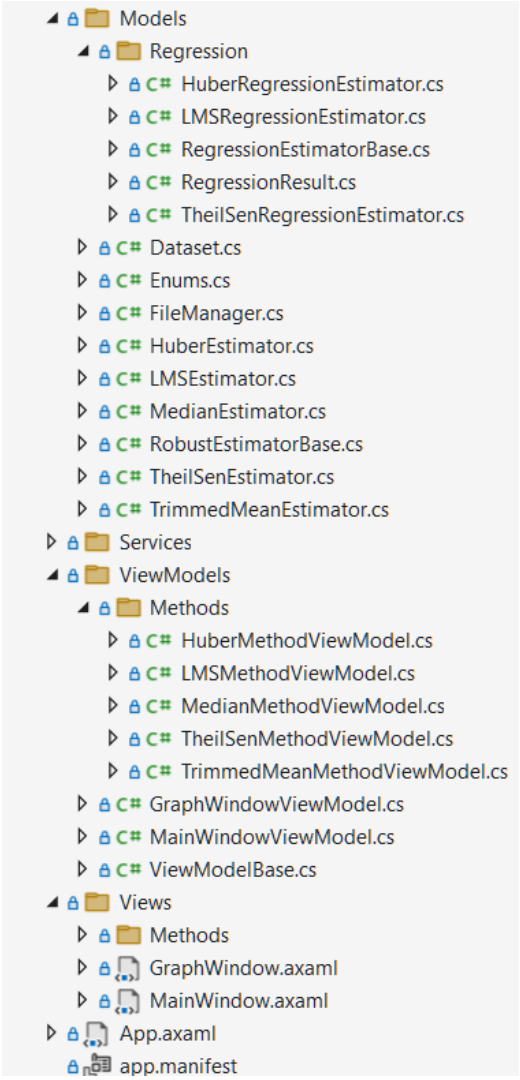
Následně se model přiřadí k vlastnosti MyPlotModel ve ViewModelu. Graf se automaticky vykreslí v UI, jakmile dojde k přiřazení modelu.

2.4.4 Shrnutí

Knihovna **OxyPlot** umožňuje do aplikace snadno integrovat grafické výstupy bez nutnosti psát nízkoúrovňové kreslicí rutiny. Verze **OxyPlot.AvaloniaCore** je přitom nezbytná pro kompatibilitu s novějšími verzemi Avalonie (nad 0.10). Díky tomu lze prezentovat uživatelům aplikace výsledky robustní statistické analýzy v podobě grafů, čímž se usnadňuje interpretace dat a metod (např. porovnání klasické a robustní regrese v jednom zobrazení).

2.5. Aplikace “RobustEstimation”

2.5.1 Souborová struktura aplikace



Models/

- **Estimatorové třídy**: HuberEstimator, LMSEstimator, MedianEstimator, TheilSenEstimator, TrimmedMeanEstimator dědí z RobustEstimatorBase.
- **Regression/**: specializované regresní implementace (HuberRegressionEstimator, LMSRegressionEstimator, TheilSenRegressionEstimator) a společné objekty RegressionResult, RegressionEstimatorBase.
- Dataset.cs drží vstupní kolekci hodnot, Enums.cs definuje použitá výčtová kritéria, FileManager.cs zajišťuje I/O operace nad soubory.

Services/

- IGraphable.cs – rozhraní, které definuje metodu pro získání dat vhodných pro vykreslení (např. IEnumerable<DataPoint> GetPlotData()), využívané ve ViewModelu pro graf.

- IWindowService.cs – rozhraní abstrahující otevírání nových oken (např. void ShowGraphWindow(object viewModel)).
- WindowService.cs – implementace IWindowService, která pomocí Avalonia Window vytváří instance GraphWindow, přiřazuje jim DataContext a volá Show().

ViewModels/

- MainWindowViewModel řídí celkovou logiku: načítání Dataset, výběr a spuštění estimatoru, zobrazení výsledků.
- GraphWindowViewModel připravuje data (OxyPlot.PlotModel) pro vykreslení v samostatném okně.
- **Methods/**: třídy XxxMethodViewModel poskytují parametry pro každou metodu (např. trimming level, tuning constant) a volají odpovídající Estimator.

Views/

- MainWindow.axaml – hlavní UI; volba zdroje dat, metody a tlačítko „Spustit“.
- **Methods/GraphWindow.axaml** – okno s PlotView, kde se vykreslí dataset a odhadnutá křivka/průměr.

App.axaml, **Program.cs**, **ViewLocator.cs** – standardní bootstrap Avalonia aplikace:

- Program.cs nastaví hostitel a spustí hlavní okno,
- ViewLocator mapuje ViewModely na Views pro automatické vytváření instancí při navigaci.

2.5.2 Soubor MainWindow.axaml

```
<Window
  xmlns="https://github.com/avaloniaui"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  xmlns:vm="using:RobustEstimation.ViewModels"
  xmlns:vmm="using:RobustEstimation.ViewModels.Methods"
  xmlns:loc="clr-namespace:RobustEstimation.Properties"
  x:Class="RobustEstimation.Views.MainWindow"
  x:DataType="vm:MainWindowViewModel"
```

```
 Title="Robustní Estimace"
 Width="900" Height="700">

        <Grid>
            <Grid.RowDefinitions>
                <RowDefinition Height="Auto"/>
                <RowDefinition Height="*"/>
                <RowDefinition Height="Auto"/>
            </Grid.RowDefinitions>

            <!-- Top bar: method & language selectors -->
            <StackPanel Grid.Row="0" Orientation="Horizontal" Spacing="10">
                <ComboBox ItemsSource="{Binding Methods}"
                                SelectedItem="{Binding SelectedMethod}">
                </ComboBox>
            </StackPanel>

            <!-- Method UI -->
            <Border
              Grid.Row="1"
              MinHeight="200"
              Margin="10"
              BorderThickness="1"
              BorderBrush="Gray">
                <ScrollViewer VerticalScrollBarVisibility="Auto">
                    <ContentControl Content="{Binding
CurrentMethodViewModel}"/>
                </ScrollViewer>
            </Border>

            <!-- Bottom bar: input, buttons, progress -->
```

```
<StackPanel Grid.Row="2" Orientation="Vertical" Spacing="10">
    <TextBlock Text="Body odhadu" FontSize="14"/>
    <TextBox
     Text="{Binding InputNumbers,
UpdateSourceTrigger=PropertyChanged}"
     Watermark="{Binding InputPlaceholder}"
     AcceptsReturn="True"
     Height="100"/>
    <StackPanel Orientation="Horizontal" Spacing="10">
        <Button
         Content="Načíst soubor..."
         Width="150"
         Command="{Binding LoadFileCommand}"/>
        <Button
         Content="Uložit vysledek..."
         Width="170"
         Command="{Binding SaveFileCommand}"/>
        <Button
         Content="Ukazát graf..."
         Width="150"
         Command="{Binding ShowGraphCommand}"
         IsEnabled="{Binding IsGraphAvailable}"/>
        <ProgressBar
         Minimum="0"
         Maximum="100"
         Value="{Binding Progress}"
         Height="20"/>
        <TextBlock
         Text="{Binding ExecutionTime}"
         FontSize="12"
         Foreground="Gray"/>
```

```
            </StackPanel>
        </StackPanel>
    </Grid>
</Window>
```

MainWindow.axaml definuje hlavní okno aplikace a jeho rozložení v XAML. V hlavičce se deklarují XML namespace pro Avalonia, ViewModely a lokalizaci. Okno je typu <Window> s třídou RobustEstimation.Views.MainWindow a datovým kontextem MainWindowViewModel.

- **Grid** se třemi řádky:

    1. **Horní lišta** (Row 0): <ComboBox> pro výběr metody, zdroj položek Methods a vybraná metoda vázaná na SelectedMethod.

    2. **Centrální oblast** (Row 1): <Border> s ScrollViewer a <ContentControl> vázaným na CurrentMethodViewModel. Zde se dynamicky načítá UI pro zvolenou metodu.

    3. **Dolní lišta** (Row 2): vstupní pole <TextBox> pro zadání bodů, tři tlačítka (LoadFileCommand, SaveFileCommand, ShowGraphCommand), <ProgressBar> vázaná na Progress a <TextBlock> zobrazující ExecutionTime.

Kombinace DataType="vm:MainWindowViewModel" a bindingů zajišťuje čisté MVVM propojení bez code-behind.

2.5.3 Soubory App.axaml i App.axaml.cs

App.axaml:

```
<Application xmlns="https://github.com/avaloniaui"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
                    xmlns:resm="clr-
namespace:RobustEstimation.Resources;assembly=RobustEstimation"
                    xmlns:loc="clr-
namespace:RobustEstimation.Properties;assembly=RobustEstimation"
        x:Class="RobustEstimation.App"
        xmlns:local="using:RobustEstimation"
        RequestedThemeVariant="Light">
```

```
        <!-- "Default" ThemeVariant follows system theme variant. "Dark" or "Light" are other available options. -->

    <Application.DataTemplates>
        <local:ViewLocator/>
    </Application.DataTemplates>

        <Application.Resources>
                <SolidColorBrush x:Key="BaseBackgroundBrush" Color="#F5F5F5"/>
                <SolidColorBrush x:Key="BaseForegroundBrush" Color="#333333"/>
                <SolidColorBrush x:Key="AccentColorBrush"   Color="#0066CC"/>
                <SolidColorBrush x:Key="ControlBorderBrush" Color="#CCCCCC"/>
        </Application.Resources>

    <Application.Styles>
        <FluentTheme />
                <StyleInclude Source="avares://OxyPlot.AvaloniaCore/Themes/Default.axaml"/>

                <Style Selector="Window">
                        <Setter Property="Background" Value="{StaticResource BaseBackgroundBrush}"/>
                        <Setter Property="Foreground" Value="{StaticResource BaseForegroundBrush}"/>
                </Style>

                <Style Selector="Button">
                        <Setter Property="Background" Value="{StaticResource AccentColorBrush}"/>
                        <Setter Property="Foreground" Value="White"/>
                        <Setter Property="FontSize" Value="14"/>
                        <Setter Property="Padding" Value="8,4"/>
```

```
                <Setter Property="CornerRadius" Value="4"/>
                <Setter Property="BorderThickness" Value="0"/>
                <Setter Property="Margin" Value="4"/>
            </Style>

            <Style Selector="TextBox">
                <Setter Property="Background"      Value="White"/>
                <Setter Property="Foreground"      Value="{StaticResource
BaseForegroundBrush}"/>
                <Setter Property="BorderBrush"     Value="{StaticResource
ControlBorderBrush}"/>
                <Setter Property="BorderThickness" Value="1"/>
                <Setter Property="Padding"         Value="6"/>
                <Setter Property="Margin"          Value="4"/>
            </Style>

            <Style Selector="ComboBox">
                <Setter Property="Background"      Value="White"/>
                <Setter Property="Foreground"      Value="{StaticResource
BaseForegroundBrush}"/>
                <Setter Property="BorderBrush"     Value="{StaticResource
ControlBorderBrush}"/>
                <Setter Property="BorderThickness" Value="1"/>
                <Setter Property="Padding"         Value="6"/>
                <Setter Property="Margin"          Value="4"/>
            </Style>

            <Style Selector="ProgressBar">
                <Setter Property="Background"      Value="#E0E0E0"/>
                <Setter Property="Foreground"      Value="{StaticResource
AccentColorBrush}"/>
                <Setter Property="Height"          Value="8"/>
```

```
                <Setter Property="CornerRadius"    Value="4"/>
                <Setter Property="Margin"          Value="4,2"/>
            </Style>


            <Style Selector="TextBlock">
                <Setter Property="Foreground" Value="{StaticResource
BaseForegroundBrush}"/>
                <Setter Property="FontSize"    Value="14"/>
                <Setter Property="Margin"      Value="2"/>
            </Style>

            <Style Selector="Label">
                <Setter Property="Foreground" Value="{StaticResource
BaseForegroundBrush}"/>
                <Setter Property="FontSize"    Value="14"/>
                <Setter Property="Margin"      Value="2"/>
            </Style>

    </Application.Styles>
</Application>
```

- Definuje globální DataTemplates a ViewLocator, který mapuje ViewModely na Views.
- V <Application.Resources> jsou klíčové barvy (BaseBackgroundBrush, AccentColorBrush apod.).
- V <Application.Styles> se přidává FluentTheme, styly OxyPlot a upravené styly pro Window, Button, TextBox, ComboBox, ProgressBar, TextBlock a Label, které sjednocují vzhled celé aplikace.

App.axaml.cs:

```
using System.Linq;
using Avalonia;
```

```
using Avalonia.Controls.ApplicationLifetimes;
using Avalonia.Data.Core;
using Avalonia.Data.Core.Plugins;
using Avalonia.Markup.Xaml;
using RobustEstimation.ViewModels;
using RobustEstimation.Views;

namespace RobustEstimation
{
    public partial class App : Application
    {
        public override void Initialize()
        {
            AvaloniaXamlLoader.Load(this);
        }

        public override void OnFrameworkInitializationCompleted()
        {
            if (ApplicationLifetime is IClassicDesktopStyleApplicationLifetime desktop)
            {
                // Avoid duplicate validations from both Avalonia and the CommunityToolkit.
                // More info: https://docs.avaloniaui.net/docs/guides/development-guides/data-validation#manage-validationplugins
                DisableAvaloniaDataAnnotationValidation();

                desktop.MainWindow = new MainWindow
                {
                    DataContext = new MainWindowViewModel(new WindowService()),
                };
            }
```

```
            base.OnFrameworkInitializationCompleted();
        }

        private void DisableAvaloniaDataAnnotationValidation()
        {
            // Get an array of plugins to remove
            var dataValidationPluginsToRemove =
BindingPlugins.DataValidators.OfType<DataAnnotationsValidationPlugin>().ToArray();

            // remove each entry found
            foreach (var plugin in dataValidationPluginsToRemove)
            {
                BindingPlugins.DataValidators.Remove(plugin);
            }
        }
    }
}
```

- Initialize() načte XAML definice.
- OnFrameworkInitializationCompleted() při desktopovém startu instancuje MainWindow, přiřadí mu DataContext = new MainWindowViewModel(new WindowService()) a předá ho desktop.MainWindow.
- DisableAvaloniaDataAnnotationValidation() odstraňuje duplicitní validační pluginy, aby se potlačila validační data annotation od Avalonia, což zamezí konfliktním chybovým hlášením.

2.5.4 Soubor Program.cs

```
using System;
using Avalonia;
namespace RobustEstimation;
internal sealed class Program
```

```
{
    // Initialization code. Don't use any Avalonia, third-party APIs or any
    // SynchronizationContext-reliant code before AppMain is called: things aren't initialized
    // yet and stuff might break.
    [STAThread]
    public static void Main(string[] args) => BuildAvaloniaApp()
        .StartWithClassicDesktopLifetime(args);

    // Avalonia configuration, don't remove; also used by visual designer.
    public static AppBuilder BuildAvaloniaApp()
        => AppBuilder.Configure<App>()
            .UsePlatformDetect()
            .WithInterFont()
            .LogToTrace();
}
```

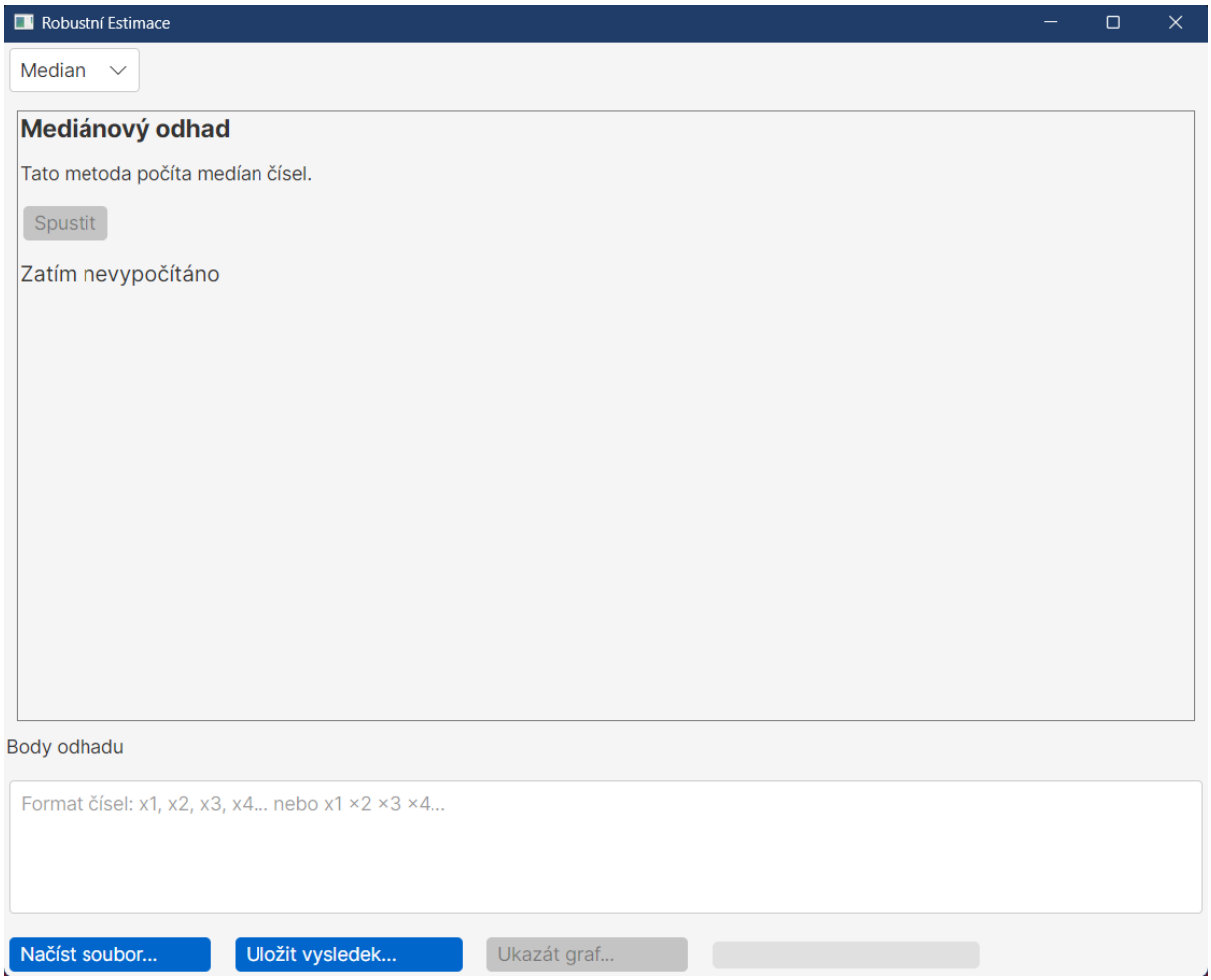
- obsahuje jediné Main() volání Avalonia hostitele
- BuildAvaloniaApp() konfiguruje aplikaci (UsePlatformDetect, WithInterFont, logování)

2.5.5 Soubor Dataset.cs

```
using System.Collections.Generic;
using System.Collections.ObjectModel;

using System.ComponentModel;

public class Dataset : INotifyPropertyChanged
{
    public ObservableCollection<double> Values { get; private set; } = new();

    public ObservableCollection<(double X, double Y)> Points { get; private set; } = new();
```

```
    public void SetValues(IEnumerable<double> vals)
    {
        Points.Clear();
        Values.Clear();
        foreach (var v in vals) Values.Add(v);
        PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(nameof(Values)));
    }

    public void SetPoints(IEnumerable<(double X, double Y)> pts)
    {
        Values.Clear();
        Points.Clear();
        foreach (var p in pts) Points.Add(p);
        PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(nameof(Points)));
    }

    public event PropertyChangedEventHandler PropertyChanged;
}
```

2.6. Ukázka aplikace RobustEstimation



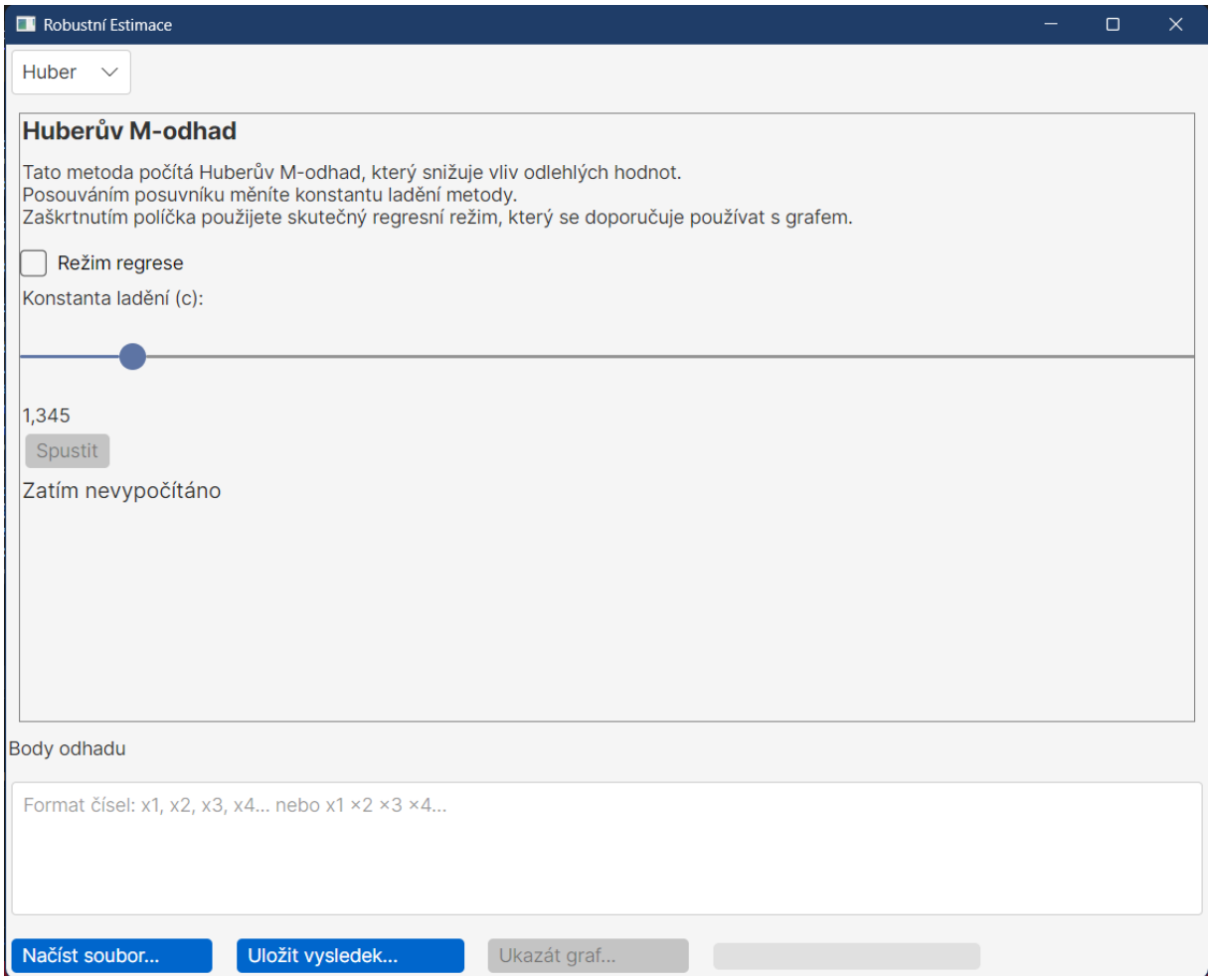
Obrázek 2: Hlavní okno aplikace

Pro názorné představení fungování aplikace uvádíme ukázku hlavního okna při zvolené metodě Medián. Nad celým oknem se v titulku zobrazuje název „Robustní odhady“, který odkazuje na zaměření programu. Ihned pod ním v levé části najdeme rozevírací seznam (ComboBox), v němž je právě vybrána metoda „Medián“. Přepnutím tohoto seznamu uživatel snadno mění mezi všemi podporovanými robustními technikami, jako jsou Theil-Sen, LMS, Trimmed Mean či Huberova metoda.

Pod výběrem metody se v hlavní části okna dynamicky mění obsah oblasti určené pro specifika vybrané techniky. V případě mediánu je zde tučným písmem uveden nadpis „Mediánový odhad“ doplněný krátkým popisem „Tato metoda počítá medián čísel.“ Pod tímto textem je umístěno tlačítko „Spustit“, které zůstává neaktivní, dokud uživatel nezadá vstupní data. Pod ním je textová značka „Zatím nevypočítáno“, jež indikuje, že výpočet dosud neproběhl.

Ve spodní části okna se nachází oblast pro zadání vstupních bodů odhadu. Nejprve je zde popisek „Body odhadu“, pod ním pak velké víceřádkové textové pole s vodotiskem „Formát čísel: x1, x2, x3, x4, … nebo x1 x2 x3 x4 …“, kam uživatel vkládá sekvenci čísel oddělených buď čárkami, nebo znakem násobení. Pod tímto vstupním polem je umístěn řádek s akčními

tlačítky a indikátory. Levé dvě nejsilněji vyvedená tlačítka „Načíst soubor …“ a „Uložit výsledek …“ umožňují práci se soubory – načtení existujícího seznamu čísel z disku a export vypočteného výsledku. Tlačítko „Ukázat graf …“ je v kontextu mediánu deaktivované, neboť u této metody není dostupná grafická vizualizace. Vedle tlačítek si uživatel všimne horizontálního ProgressBaru, jehož průběh by v aktivním výpočtu indikoval postup od nuly do sto procent. A úplně vpravo pak textový element „ExecutionTime“, do něhož se po skončení výpočtu dynamicky zapíše čas, který běh algoritmu zabral.



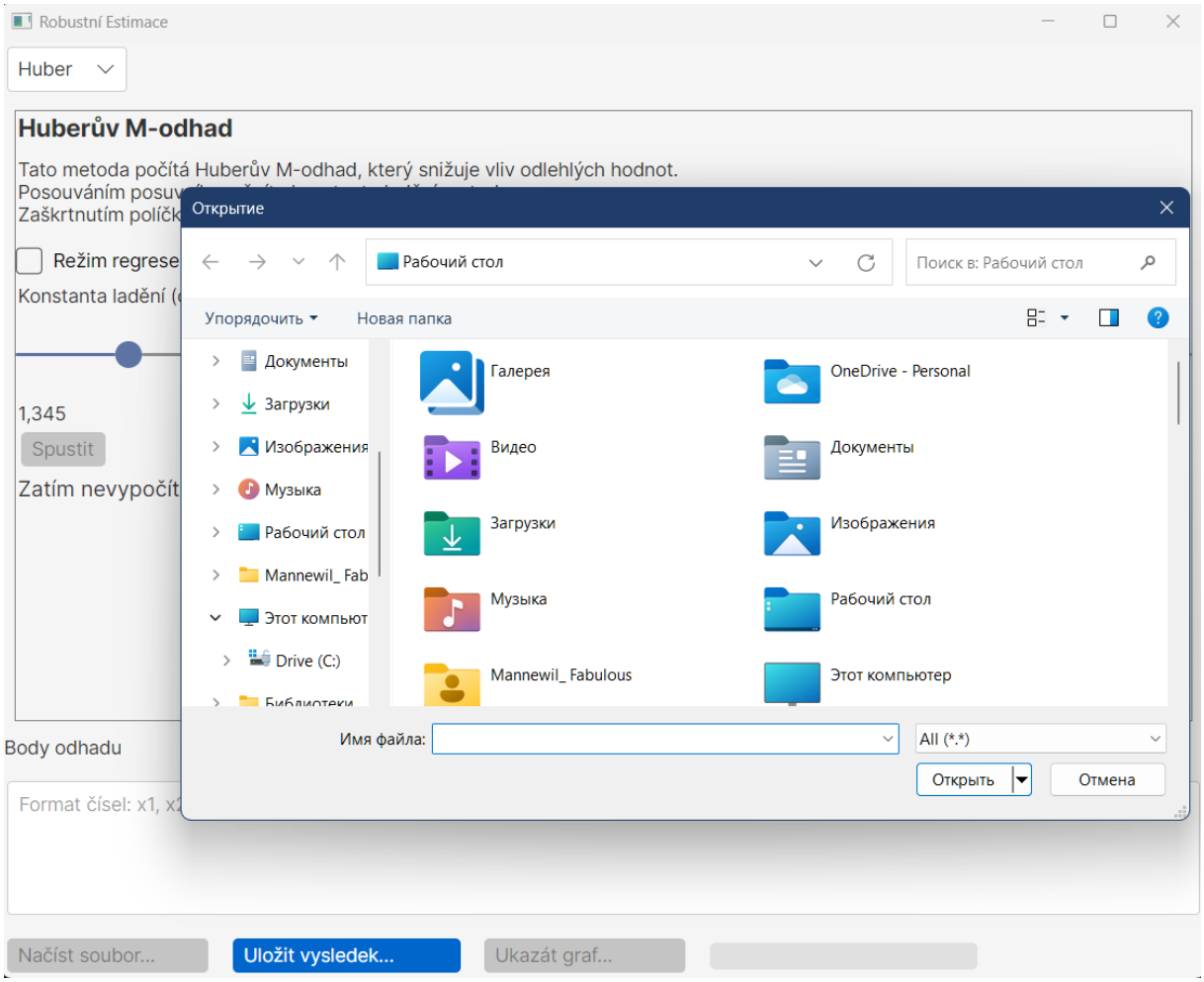
Obrázek 3: Jiná vybraná metoda v aplikaci

Na dalším snímku vidíme, jak se uživatelské rozhraní přizpůsobí volbě režimu Huberova M-odhadu. V rozevíracím seznamu v horní části okna je nyní vybrána položka „Huber“.
V prostřední části se objevuje nadpis „Huberův M-odhad“ doplněný podtitulem „Tato metoda počítá Huberův M-odhad, který snižuje vliv odlehlých hodnot.“ Následuje stručný popis návodu: uživatel jemným posouváním posuvníku mění ladicí konstantu metody a může zaškrtnutím políčka přepnout do skutečného regresního režimu, který se doporučuje kombinovat s grafickou ukázkou.

Pod textem se nachází zatržítko „Režim regrese“, jež po aktivaci přepne výpočet do regresního módu (tedy přepočte odhad na regresní linii místo jednoduché jednorozměrné statistiky). Pod tím uživatel spatří popisek „Konstanta ladění (c):“ následovaný vodorovným

posuvníkem, jehož výchozí hodnota (1,345) je uvedena těsně pod ním. Jakmile ladicí konstanta změní svoji hodnotu, aktualizuje se i numerický štítek.

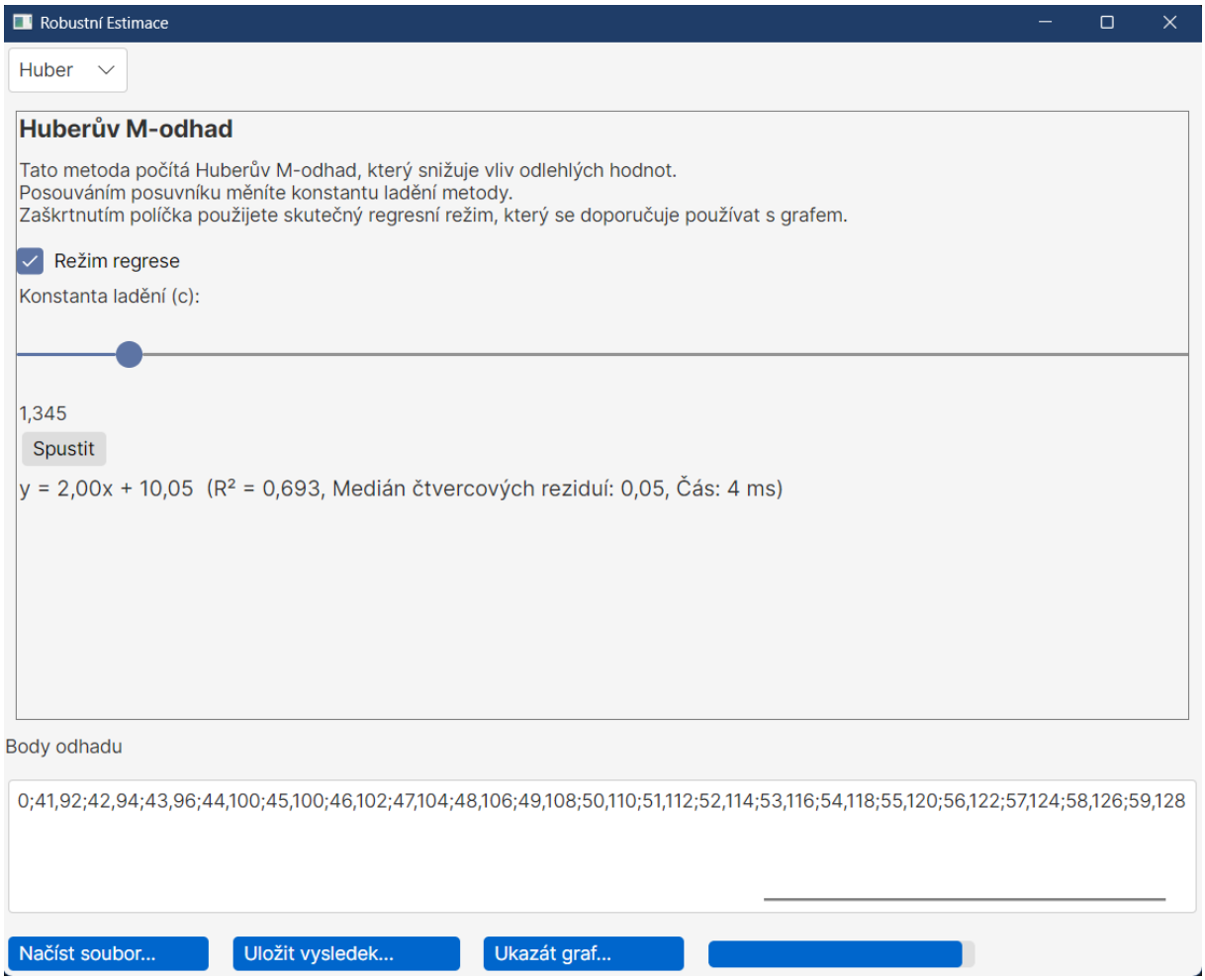
Tlačítko „Spustit“ zůstává do chvíle vložení vstupních dat deaktivované a pod ním stále najdeme text „Zatím nevypočítáno“, který indikuje neprovedený výpočet. Stejně jako v předchozím případě zůstávají ve spodní části okna sekce pro zadání bodů odhadu a akční tlačítka „Načíst soubor …“, „Uložit výsledek …“ a „Ukázat graf …“, přičemž po aktivaci regresního režimu se tlačítko pro grafickou ukázku stává aktivním. ProgressBar a časový odhad běhu algoritmu opět čekají až na zahájení výpočtu.



Obrázek 4: Ukázka dialogu načítaní souboru v aplikaci.

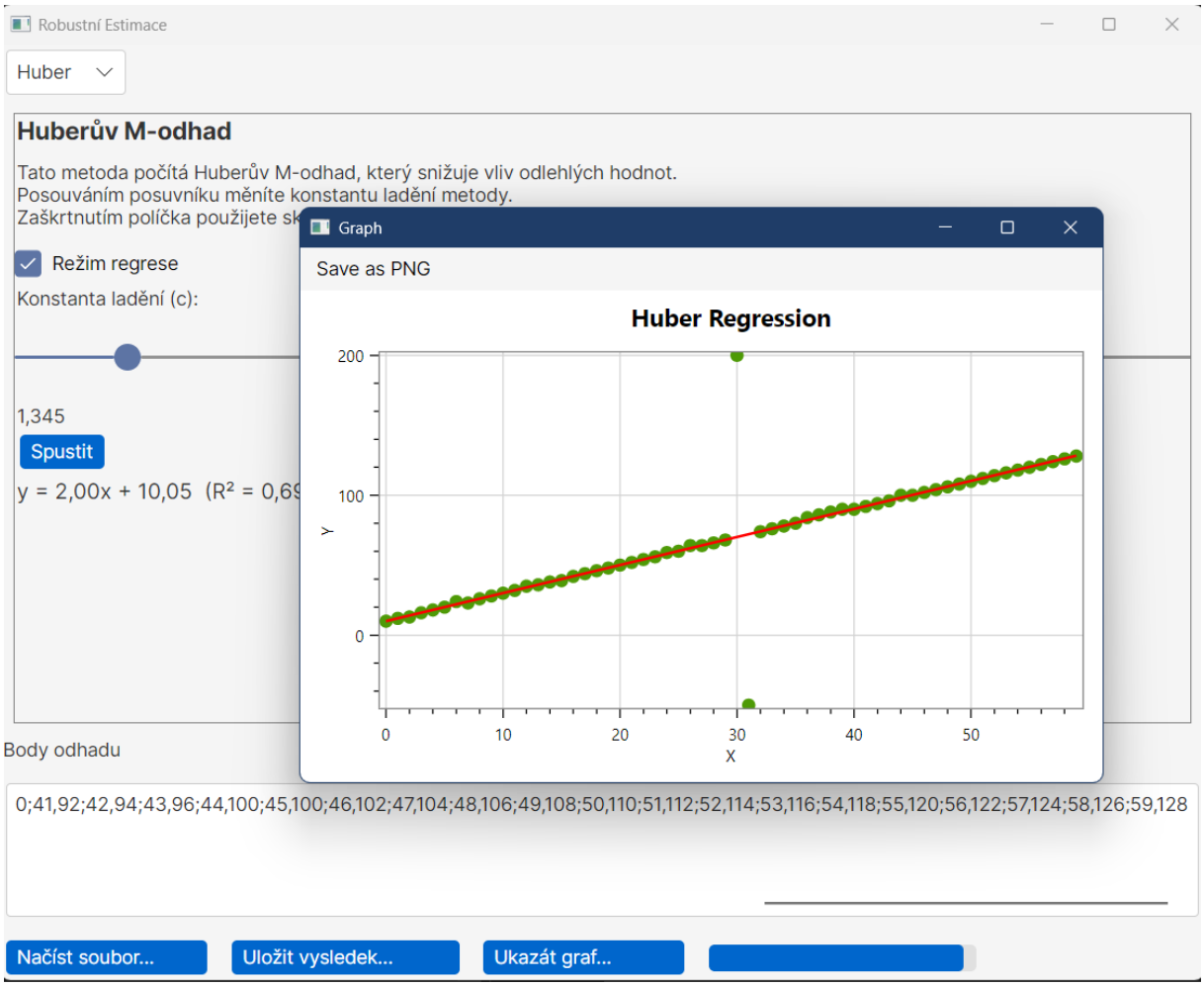
Ukázka načítání vstupních dat probíhá prostřednictvím standardního dialogu pro výběr souboru operačního systému. Po kliknutí na tlačítko „Načíst soubor …“ se nad hlavním oknem aplikace otevře okno „Otevření“, ve kterém může uživatel navigovat v adresářové struktuře (Dokumenty, Stažené soubory, Obrázky apod.) a vybrat libovolný soubor (ve výchozím filtru „All (.)“). V dolní části dialogu je pole „Jméno souboru:“, kam lze zadat název ručně, a tlačítka „Otevřít“ pro potvrzení volby či „Storno“ pro zrušení operace. Jakmile uživatel potvrdí výběr, aplikace zvolený soubor načte a jeho obsah (například číselná data nebo dvojice bodů pro regresi) se zobrazí v textovém poli pro zadání vstupu. Tímto způsobem se do aplikace pohodlně importují data připravená v externích nástrojích.

Poznámka: Je to systémový dialog prostředí dotnetu, což je SaveFileDialog, který bere jazyk operačního systému, a tak zobrazí názvy různých tlačítek, fieldu, a podobné.



Obrázek 5: Ukázka úspěšného výpočtu.

Po úspěšném načtení, či zadání dat a provedení výpočtu aplikace zobrazí kompletní výsledky přímo v hlavním okně. Nadpis a popisek metody zůstávají na svém místě, pod nimi je nyní zaškrtnutý režim „Režim regrese“, který signalizuje, že Huberův M-odhad vypočítává regresní přímku. Hodnota ladicí konstanty „c“ se nastavuje pomocí posuvníku a je zobrazena včetně desetinné čárky (zde „1,345“). Tlačítko „Spustit“ je nyní aktivní a po kliknutí spustí výpočet. Pod ním se objeví text obsahující rovnici přímky ve tvaru „$y = 2{,}00x + 10{,}05$“, dále hodnota koeficientu determinace „$R^2 = 0{,}693$“, medián čtvercových reziduí „0,05“ a čas výpočtu „4 ms“.

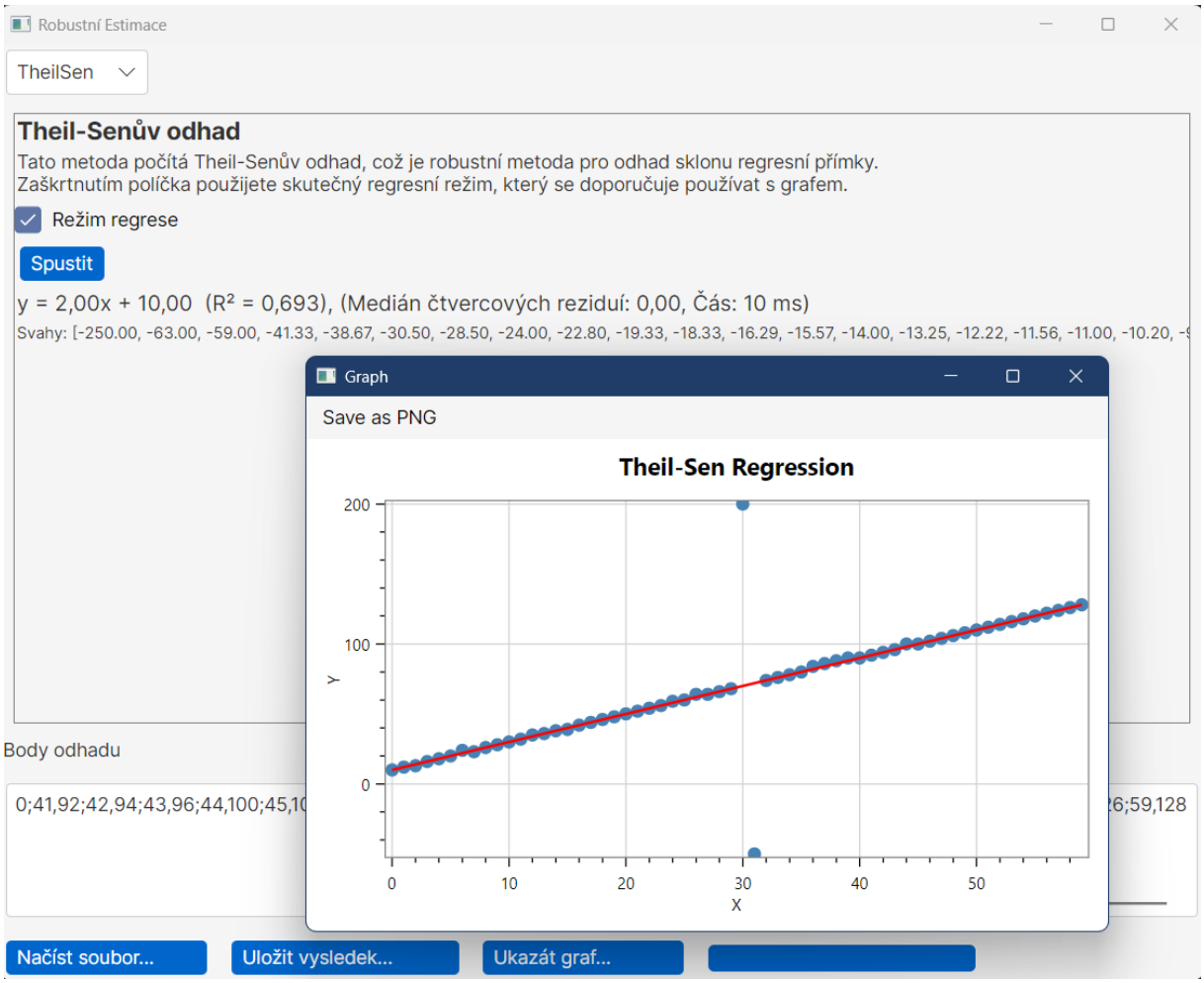


Obrázek 6: Ukázka grafu metody Huberové regresi.

V zobrazení grafu se otevírá samostatné okno pojmenované „Graph“, které poskytuje vizuální reprezentaci výsledků Huberovy regresní analýzy. V horní liště okna je tlačítko „Save as PNG“, umožňující uživateli okamžitě exportovat graf do obrázku bez nutnosti dalších nástrojů. Samotný graf s názvem „Huber Regression“ ukazuje na vodorovné ose „X“ a svislé ose „Y“ vztah mezi původními datovými body a odhadnutou regresní přímkou.

Datové body jsou vykresleny jako drobné zelené značky, které jsou aproximovány, přičemž je na nich jasně vidět, kde se nacházejí případné odlehlé hodnoty. Huberova odhadnutá linie je zvýrazněna červenou čarou vedoucí středem datových bodů, přičemž díky robustnímu způsobu výpočtu odolává vlivu extrémních odchylek. Graf obsahuje jemnou mřížku, která usnadňuje čtení konkrétních souřadnic a porovnání bodů s regresní čarou.

Toto okno slouží jako intuitivní nástroj pro ověření kvality odhadu: uživatel okamžitě vidí, jakým způsobem Huberova metoda vyrovnává vliv odlehlých dat, a může posoudit, zda zvolená konstantní hodnota ladění poskytuje vhodný kompromis mezi citlivostí a odolností. Díky možnosti uložení grafu ve formátu PNG je snadné výsledky archivovat nebo je dále sdílet, včetně názvu grafu a jasně označených os.

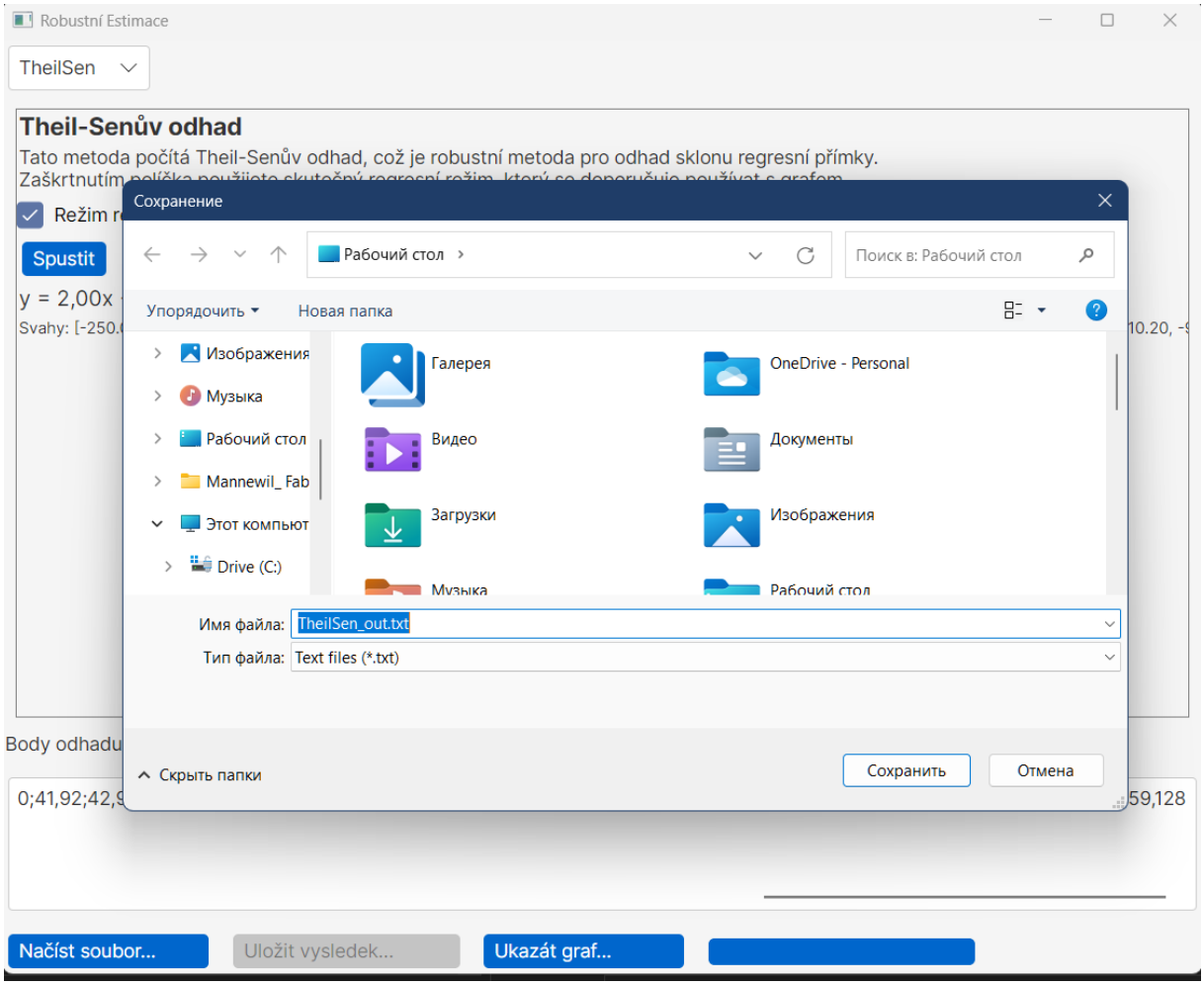


Obrázek 7: Ukázka grafu metody Theil-Senůvého odhadu.

V okamžiku, kdy uživatel přepne výběr metody na Theil-Senův odhad, se v hlavním okně aplikace okamžitě aktualizují popisky i popis funkčnosti. Nadpis “Theil-Senův odhad” doprovází stručný komentář, že jde o robustní metodu pro odhad sklonu regresní přímky, a upozornění, že zaškrtnutím volby “Režim regrese” se aktivuje plný režim výpočtu vhodný k vizualizaci v grafu. Jakmile uživatel zaškrtne pole “Režim regrese” a stiskne modré tlačítko “Spustit”, tlačítko se během výpočtu krátce zablokuje a po dokončení výpočtu se zobrazí v textové části výsledná regrese uvedená ve formátu y = 2,00x + 10,00 (R^2 = 0,693), doplněná o medián čtvercových reziduí, čas potřebný k běhu algoritmu a seznam jednotlivých dílčích směrnic (“Svahy”), kterými Theil-Senův odhad vnitřně pracuje.

Příkaz “Ukázat graf …” nyní odemkne okno s grafickou ukázkou: ve vyskakovacím dialogu “Graph” se zobrazí datové body jako modré tečky, z nichž je patrné, jak robustní metoda ignoruje extrémy, a červená linka reprezentuje průběh odhadnuté regresní přímky. Graf je opatřen osami “X” a “Y”, titulkem “Theil-Sen Regression” a jemnou mřížkou, která

usnadňuje orientaci v datech. Verte že dialog nese tlačítko “Save as PNG”, díky němuž lze celý graf pohodlně uložit ve formátu obrázku pro další dokumentaci či prezentaci.



Obrázek 8: Ukázka dialogu uložení výsledku aplikace.

Otevřením dialogu pro uložení výsledků se zobrazí standardní systémové okno pro výběr cílové složky a názvu souboru. Název je předvyplněn podle aktuálně zvolené metody (v tomto případě “TheilSen_out.txt”), a typ souboru je omezen na běžné textové soubory (*.txt). Uživatel může upravit jméno či cestu, poté klikne na tlačítko “Uložit” a všechen vypočtený obsah – tedy textový řetězec s rovnicí regresní přímky, statistikami (R^2, mediány reziduí, seznam svahů) a seznamem bodů odhadu – se okamžitě zapíše do zvoleného .txt souboru. Tlačítko “Uložit výsledek …” na pozadí zůstane deaktivované až do úspěšné akce, aby nedošlo k opakovanému vyvolání uložení bez nových dat.

Soubor obsahuje původní soubor dat, zvolenou metodu a parametr, pokud je to vhodné (např. konstanta v Huberově metodě). Následuje jedinečný upravený soubor dat pro každou metodu. Poté se zobrazí to, co jsme viděli v programu.

ZÁVĚR

V této práci jsme se zabývali teoretickým i praktickým využitím robustních metod pro odhad parametrů a zpracování dat s odlehlými hodnotami. V teoretické části byly popsány základní principy robustní statistiky, klíčové vlastnosti klasických a robustních odhadů polohy a regresních metod (Theil–Sen, Huberovy M-odhady, trimovaný průměr a LMS). Byla zdůrazněna důležitost parametru breakdown point a odolnosti vůči porušení normality, stejně jako omezení spojená s výpočetní náročností a volbou parametrů.

Praktická část představila návrh a implementaci desktopové aplikace „RobustEstimation“ v C#, využívající framework Avalonia UI a architekturu MVVM. Aplikace umožňuje snadný výběr robustní metody, zadání dat ve formě bodových řad nebo tabulek a vizualizaci výsledků prostřednictvím knihovny OxyPlot. Uživatelé tak mohou porovnat výstupy klasických a robustních přístupů, ověřit vliv odlehlých hodnot a automatizovat výpočet v přehledném grafickém prostředí.

Výsledky testování demonstrovaly, že robustní metody poskytují stabilnější odhady za přítomnosti extrémů na úkor mírně vyšší výpočetní náročnosti. Aplikace byla navržena s ohledem na rozšiřitelnost – implementace nových estimátorů lze snadno přidat rozšířením základní hierarchie tříd. Pro další vývoj lze doporučit integraci podpory vícerozměrných dat, optimalizaci výkonu pro velké soubory a pokročilé nástroje pro předzpracování dat (detekce a vizualizace outlierů).

Domnívám se, že tato práce přispěla jak k pochopení teoretických aspektů robustních odhadů, tak k vytvoření funkčního softwarového řešení, které může sloužit jako výchozí bod pro další výzkum a aplikace v oblasti statistické analýzy dat.

POUŽITÁ LITERATURA

1. WILCOX, Rand R. *Introduction to robust estimation and hypothesis testing*. 4th edition. Amsterdam: Elsevier, [2017]. ISBN 978-0-12-804733-0.

2. ROUSSEEUW, Peter J. *Robust regression and outlier detection*. 4th edition. Chichester: Wiley, 1987. ISBN 04-718-5233-3. Dostupné také z: http://www3.interscience.wiley.com/cgi-bin/bookhome/109871868.

3. ANDĚL, Jiří. *Statistické metody*. 4., upr. vyd. Praha: Matfyzpress, 2007. ISBN 978-80-7378-003-6. Dostupné také z: http://krameriusndk.nkp.cz/search/handle/uuid:bee41960-ad31-11e4-a7a2-005056827e51.

4. STIGLER, Stephen M. *The history of statistics: the measurement of uncertainty before 1900*. Ninth printing. Cambridge: Belknap Press of Harvard University Press, [1986]. ISBN 0-674-40341-x.

5. ŠKRABÁNEK, P., MAREK, J., POZDÍLKOVÁ, A. Boscovich Fuzzy Regression Line *Mathematics*, 2021, vol. 9, no. 6, s. nestránkováno. ISSN: 2227-7390.

6. MAREK, J., HECKENBERGEROVÁ, J. Regresní přímka ve světle dějin *Forum Statisticum Slovacum*, 2014, vol. (XI) 2014, no. 4, s. 119-124. ISSN: 1336-7420.

7. HUBER, Peter J. Robust Estimation of a Location Parameter. In: *The Annals of Mathematical Statistics*. 1964, s. 73-101. ISSN 0003-4851. Dostupné z: https://doi.org/10.1214/aoms/1177703732.

8. THEIL, Henri. A Rank-Invariant Method of Linear and Polynomial Regression Analysis. In: *Henri Theil's Contributions to Economics and Econometrics*. Advanced Studies in Theoretical and Applied Econometrics. Dordrecht: Springer Netherlands, 1992, s. 345-381. ISBN 978-94-010-5124-8. ISSN 0003-4851. Dostupné z: https://doi.org/10.1007/978-94-011-2546-8_20.

9. SEN, Pranab Kumar. Estimates of the Regression Coefficient Based on Kendall's Tau. In: *Journal of the American Statistical Association*. Advanced Studies in Theoretical and Applied Econometrics. Dordrecht: Springer Netherlands, 1992, s. 1379-1389. ISBN 978-94-010-5124-10. ISSN 0162-1459. Dostupné z: https://doi.org/10.1080/01621459.1968.10480934.

11. MICROSOFT. *Introduction to C#*. Online. MICROSOFT. Introduction. 2025. Dostupné z: https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/language-specification/introduction. [cit. 2025-03-12].

12. MICROSOFT. *Managed Execution Process*. Online. MICROSOFT. Managed Execution Process. 2025. Dostupné z: https://learn.microsoft.com/en-us/dotnet/standard/managed-execution-process. [cit. 2025-03-25].

13. MICROSOFT. *Fundamentals of garbage collection*. Online. MICROSOFT. Fundamentals of garbage collection. 2025. Dostupné z: https://learn.microsoft.com/en-us/dotnet/standard/garbage-collection/fundamentals. [cit. 2025-03-25].

14. MICROSOFT. *XAML overview (WPF .NET)*. Online. MICROSOFT. XAML overview (WPF .NET). 2025. Dostupné z: https://learn.microsoft.com/en-us/dotnet/desktop/wpf/xaml/?view=netdesktop-9.0. [cit. 2025-03-25].

15. AVALONIA DEVELOPERS. *Avalonia Documentation*. Online. AVALONIA DEVELOPERS. Avalonia Documentation. 2025. Dostupné z: https://docs.avaloniaui.net/. [cit. 2025-04-10].

16. MICROSOFT. *Model-View-ViewModel (MVVM)*. Online. AVALONIA DEVELOPERS. Model-View-ViewModel (MVVM). 2025. Dostupné z: https://learn.microsoft.com/en-us/dotnet/architecture/maui/mvvm. [cit. 2025-04-10].

17. *Model–view–viewmodel*. Online. In: Wikipedia: the free encyclopedia. San Francisco (CA): Wikimedia Foundation, 2001-, 23-11-2024. Dostupné z: https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93viewmodel. [cit. 2025-04-27].

18. OXYPLOT DEVELOPERS. *Oxyplot Documentation*. Online. OXYPLOT DEVELOPERS. Oxyplot Documentation. 2025. Dostupné z: https://oxyplot.readthedocs.io/en/latest/. [cit. 2025-04-10].

SEZNAM PŘÍLOH

PŘÍLOHA A*:* Zdrojový kód aplikace RobustEstimation

Tato příloha obsahuje .zip archiv s projektem z Visual Studia 2022 jménem RobustEstimation.