

VYSOKÁ ŠKOLA EKONOMICKÁ



**Predikce hodnoty nemovistostí
ve smyslu modelu společnosti Zillow
pomocí jazyka R**

==== 4EK416 Pokročilá ekonometrie ====

==== seminární práce ====

Lubomír Štěpánek

prosinec 2017

(2017) Lubomír Štěpánek, CC BY-NC-ND 3.0 (CZ)



Dílo lze dále svobodně šířit, ovšem s uvedením původního autora a s uvedením původní licence. Dílo není možné šířit komerčně ani s ním jakkoliv jinak nakládat pro účely komerčního zisku. Dílo nesmí být jakkoliv upravováno. Autor neručí za správnost informací uvedených kdekoliv v předložené práci, přesto vynaložil nezanedbatelné úsilí, aby byla uvedená fakta správná a aktuální, a práci sepsal podle svého nejlepšího vědomí a svých „nejlepších“ znalostí problematiky.

Obsah

1	Úvod	1
2	Metodologie a statistické zpracování dat	2
2.1	Předzpracování dat	2
2.2	Průzkumová analýza dat	3
2.3	Model smíšených efektů	3
3	Výsledky	5
3.1	Průzkumová analýza dat	5
3.2	Model smíšených efektů	6
4	Diskuze	10
5	Závěr	11
6	Appendix	12
6.1	<code>--main.R--</code>	12
6.2	<code>initialization.R</code>	13
6.3	<code>helper_functions.R</code>	14
6.4	<code>data_loading.R</code>	15
6.5	<code>data_processing.R</code>	18
6.6	<code>data_saving_and_reloading.R</code>	24
6.7	<code>data_postprocessing.R</code>	28
6.8	<code>exploratory_data_analysis.R</code>	31
6.9	<code>hierarchical_model.R</code>	38
7	Reference	44

Abstrakt

Schopnost správně odhadnout tržní cenu nemovitosti je důležitá pro účely trhu i investic jednotlivců, zejména jde-li o nemovitost, která dosud nebyla nikdy naceněna nebo prodána. Na americkém trhu se objektivním a data-driven odhadováním tržních cen nemovitostí zabývá společnost Zillow a své odhady nazývá *Zestimate*'s. Možnost odhad co možná nejvíce je žádoucí a proto společnost Zillow ve spolupráci s portálem kaggle.com nabídla soutěž otevřenou široké komunitě datových analytiků s cílem nalézt efektivnější mechanismus odhadování cen nemovitostí. Tato práce představuje jeden z pokusů o model odhadující chybu mezi originálním odhadem *Zestimate* a skutečnou cenou.

Dostupné datasety byly nejdříve předzpracovány, tj, numerické proměnné byly standardizovány na interval $\langle 0, 1 \rangle$, časové proměnné byly zjednodušeny z přesnosti na den pouze na přesnost na měsíc. Datasety byly rozděleny na trénovací (70 % původních dat) a testovací (zbylých 30 % původních dat). Vzhledem k velkému počtu sledovaných nemovitostí v krátkém sledovaném časovém období je na data nahlíženo jako na krátký panel a byly použity metody v rámci panelové regrese. Na trénovacím datasetu byl vytvořen model smíšených efektů, který zahrnul některé vstupní proměnné jako fixní efekty a proměnnou identifikující danou nemovitost považuje za náhodný efekt, čímž dává každé nemovitosti individuální hladinu závisle proměnné a očištěje od vlivu opakovaných záznamů vícekrát prodaných nemovitostí.

Získaný model není příliš vhodný pro účely statistické inference; realizace chybové složky nesledují dokonalé normální rozdělení; jejich rozdělení je spíše špičatější, rovněž porušují homoskedasticitu. Pro účely predikce však model může relativně dobře sloužit; predikční přesnost byla měřena pomocí střední absolutní chyby (MAE), ta byla vyčíslena predikcí hodnot závisle proměnné nad testovacím datasetem a výsledná chyba $MAE \doteq 0,07$ je srovnatelná s ostatními modely účastníckimi se soutěže kaggle.com.

Model se smíšenými efekty se zdá být jedním z možných algoritmů pro odhad tržních cen nemovitostí s relativně dobrými predikčními schopnostmi. Modely, které slibují potenciálně lepší predikční přesnost (na úkor interpretace) jsou pak někteří zástupci strojového učení (*machine learning*), které nevyžadují prakticky žádné předpoklady o vstupních datech (jsou neparametrické) – ať už jde o *random forest regresi* anebo *neuronové sítě*.

1 Úvod

Tato práce předkládá pokus o vytvoření (nového) modelu odhadu ceny nemovitostí v podobném smyslu, v jakém je již jedenáct let odhaduje pro americký trh společnost Zillow [1]. Současně jde o řešení seminárního zadání v rámci předmětu 4EK416 Pokročilá ekonometrie, vyučovaném na Vysoké škole ekonomické v Praze. V neposlední řadě práce předkládá i možné řešení veřejně dostupné soutěže portálu kaggle.com, aktuálně jednoho z nejznámějších webů v oblasti *data science*.

Protože nemovitost je mnohdy nejdražší komodita, do které běžný spotřebitel investuje, je třeba mít k dispozici vhodný a věrohodný ukazatel reálné ceny nemovitosti. Komplexní odhad ceny nemovitosti, takzvaný *Zestimate*, který používá společnost Zillow, byl vytvořen kvůli co možná nejlepšímu vystihnutí ceny dané nemovitosti na trhu, a to i kvůli rankingu těch nemovitostí, které ještě nebyly nikdy naceněny či prodány, eventuálně u nich taková informace není z různých důvodů dostupná, [1].

Odhady *Zestimate* jsou založeny na zhruba 7,5 milionech (!) statistických modelů (s nějakou apriorní parametrizací) a modelů z oblasti strojového učení, jež analyzují stovky proměnných pro každou nemovitost (přesné algoritmy samozřejmě nejsou kvůli ochraně know-how dostupné). V průběhu času společnost Zillow zmenšila střední chybu mediánu odchylky odhadnuté ceny nemovitosti z asi 14 % na současných 5 %, čímž se Zillow's etabloval jako nejdůvěryhodnější data-driven hráč na trhu amerických nemovitostí, zároveň jde o ukázku vhodně a efektivně aplikovaného přístupu strojového učení, [1].

Přesto je zde prostor pro možné vylepšení predikční přesnosti, resp. snížení průměrné chyby odhadu ceny nemovitosti. I z toho důvodu portál kaggle.com ve spolupráci se společností Zillow hostí veřejně otevřenou datovou soutěž *Zillow Prize: Zillow's Home Value Prediction (Zestimate)*, [1], kde je možné na testovacích datasetech porovnat predikční přesnost vlastního modelu.

K dispozici je několik rozsáhlých datasetů, které jednak popisují pomocí velkého množství proměnných vlastnosti jednotlivých evidovaných nemovitostí, jednak nabízí údaje o velké řadě provedených prodejů daných nemovitostí, kdy kromě časového určení takového prodeje v letech 2016–2017 je k dispozici i proměnná logerror, která je definována jako

$$\text{logerror} = \log z - \log s,$$

kde z je *Zestimate*, tedy předpokládaná tržní cena nemovitosti odhadnutá pomocí algoritmů společnosti Zillow, a s je skutečná tržní cena, za kterou byla daná nemovitost v evidovaném čase prodána. V optimálním případě, kdy by byla společnost Zillow schopná odhadovat tržní ceny zcela přesně, je pro každý prodej každé nemovitosti zřejmě $\text{logerror} = 0$.

Protože množství evidovaných prodejů nemovitostí v datasetech je zhruba řádu 10^5 , zatímco počet časových okamžiků (zde tedy prodejů jednotlivých nemovitostí) je maximálně v řádu jednotek (s tím, že velká většina nemovitostí byla prodána za sledované období jen jednou, anebo vůbec), budeme na data nadále nahlížet jako na velmi krátký panel a použitá metodologie bude pramenit z rodiny panelové regrese [2], ovšem vzhledem k „délce“ panelu půjde spíše o běžnou regresi bez prvků analýzy časové řady. Nebudeme tedy brát v potaz TSRM (*time-series regression modelling*) [3], naopak v rámci panelové regrese nad krátkým modelem zvážíme efekty každé nemovitosti zvlášť, vydáme se tedy cestou *modelu smíšených efektů* (fixních a náhodných).

2 Metodologie a statistické zpracování dat

Celá úloha byla řešena v prostředí R, které je určeno pro statistické výpočty a následné grafické náhledy [4]. Datové soubory byly vzhledem ke své velikosti nahrány do prostředí R pomocí balíčku `data.table`, který díky funkcím `fread()` a `fwrite()` umožňuje uživatelsky příjemně rychlé nahrávání i souborů dokonce větších než je velikost dostupné RAM paměti jednotky, na které je analýza prováděna.

Jednotlivé výpočty v rámci průzkumové analýzy dat a v rámci modelování smíšených efektů byly provedeny pomocí nadstavbových R-kových balíčků `corrgram`, `lme4` a `lmerTest`.

Veškerou analýzu lze v prostředí R spustit pomocí hlavního skriptu `__main.R__`.

2.1 Předzpracování dat

Časově nejnáročnější fází byla předzpracování rozsáhlých dat. Použité datasety `properties_2016.csv` a `properties_2017.csv` popisující vlastnosti nemovitostí mají oba velikost > 600 MB, proto bylo nutné použít k jejich nahrání balíček `data.table` a funkci `fread()`. Oba zahrnují právě 2 985 217 pozorování a 58 proměnných. Primárním klíčem je vždy proměnná `parcelid`.

Datasety `train_2016_v2.csv` (rozměrů 90275×3) a `train_2017.csv` (rozměrů 77613×3) obsahují primární klíč `parcelid`, dále vypočtenou hodnotu `logerror` a nakonec `transactiondate`, informaci o času prodeje nemovitosti s přesností na den.

Bylo vždy nutné propojit pro každý z roků dataset `properties_*.csv` a `train_*.csv` pomocí operace analogické SQL příkazu `LEFT OUTER JOIN`. Nakonec byly automatizovaně kódovány datové typy všech proměnných, a to pomocí vlastní uživatelské funkce `isPotentiallyNumeric()`.

Numerické proměnné byly vzhledem k navzájem relativně odlišné variabilitě standardizovány na interval hodnot $\langle 0, 1 \rangle$. Pro každou numerickou proměnnou $\mathbf{X} = (x_1, x_2, \dots, x_n)^T$ byla provedena standardizace na proměnnou $\mathbf{X}' = (x'_1, x'_2, \dots, x'_n)^T$ tak, že $x'_i = \frac{x_i - \min\{\mathbf{X}\}}{\max\{\mathbf{X}\} - \min\{\mathbf{X}\}}$ pro $\forall i \in \{1, 2, \dots, n\}$.

Časová proměnná `transactiondate` (s přesností na konkrétní den) byla převedena do dvou proměnných `transaction_year` a `transaction_month` značící rok a měsíc prodeje nemovitosti; původní proměnná `transactiondate` nebyla v další analýze již použita.

Předchozí úlohy jsou asynchronní, jejich výsledek je možné uložit pomocí funkce `fwrite()` a při další seanci jen loadovat pomocí a `fread()`.

Předzpracování dat řeší skripty `initialization.R`, `helper_functions.R`, `data_loading.R`, `data_processing.R`, `data_saving_and_reloading.R` a `data_postprocessing.R`.

2.2 Průzkumová analýza dat

Přestože je primárním cílem modelu (viz dále) predikce nových hodnot, nikoliv inference¹, [5], je vhodné se zabývat multikolinearitou vysvětlujících proměnných i z technického úhlu pohledu – model se silnou kolinearitou může být numericky obtížně vyčíslitelný (snižuje se hodnost designové matice modelu, resp. může být problém nalézt inverzní matici k designové matici, jak uvádí [6]).

Míra korelace mezi numerickými regresory byla posouzena korelační maticí rozměru $p \times p$, kde p je počet numerických regresorů, s Pearsonovými koeficienty tak, že Pearsonův koeficient korelace $r_{\mathbf{x}, \mathbf{y}}$ mezi numerickými regresory s vektory hodnot $\mathbf{x} = (x_1, x_2, \dots, x_n)^T$ a $\mathbf{y} = (y_1, y_2, \dots, y_n)^T$ je roven dle [7]

$$r_{\mathbf{x}, \mathbf{y}} = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2} \sqrt{\sum_{i=1}^n (y_i - \bar{y})^2}},$$

kde n je počet pozorování a \bar{x} , resp. \bar{y} je průměrná hodnota nad vektorem \mathbf{x} , resp. \mathbf{y} .

Naopak míra asociace mezi dvěma kategorickými regresory byla posouzena pomocí Cramérora V , kde dle [8] je

$$V = \sqrt{\frac{\chi^2}{n \cdot \min(k-1, r-1)}},$$

kde χ^2 statistika je spočítána nad kontingenční tabulkou o rozměrech $k \times r$ sestavenou pro dané dva kategorické regresory (tedy první regresor nabývá k různých hodnot, druhý r různých hodnot), n je počet pozorování.

Pro lepší posouzení, které skupiny regresorů spolu silně korelují (a tedy z takové skupiny je vhodné do modelu obsadit vždy jen jeden z těchto regresorů), je však vhodné použít grafický výstup typu *koreogram*.

Průzkumovou analýzu dat provádí skript `exploratory_data_analysis.R`.

2.3 Model smíšených efektů

Model smíšených efektů byl sestaven nad náhodně vybranými 70 % vstupního datasetu, ten je poté značen `train_set`. Zbylých 30 % hodnot vstupního datasetu (značených jako `test_set`) posloužilo pro ověření predikční přesnosti modelu.

Model smíšených efektů (*mixed effects model*, obecněji též *hierarchical model*, *nested model* či *multilevel model*) kombinuje fixní a náhodné efekty, [9]. Díky tomu se obzvlášt hodí pro data obsahující opakování měření (v našem případě se jedná o opakováný prodej některých nemovitostí v rámci sledovaného období). Současně jsou mnohem robustnější vzhledem k chybějícím hodnotám či klasickým předpokladům kladeným na chybovou složku než např. analýza rozptylu opakovaných měření.

¹Je známo, že pro účely *predikce* není multikolinearita regresorů v regresním modelu tak velkým problémem jako pro účely *inference*. Důkaz se obvykle staví na myšlence, že při predikci (a hlavně při klasifikaci) nás zajímá zejména predikovaná vyrovnaná (střední) hodnota pro vektor nového pozorování, nikoliv velikost chyby odhadů. Právě velikost variability odhadů je vlivem multikolinearity významně zkreslena. Důsledkem je mimo jiné i to, že např. celá obrovská oblast *machine learningu* (strojového učení) se otázkou multikolinearity či jiné závislosti prediktorů příliš nezabývá (nebo jen do té míry, aby byly odhady numericky vyčíslitelné).

Formálně můžeme model dle [10] vyjádřit jako

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\mathbf{u} + \boldsymbol{\varepsilon}, \quad (1)$$

kde \mathbf{y} je vektor závisle proměnné (v našem případě `logerror`) tak, že platí $\mathbf{E}(\mathbf{y}) = \mathbf{X}\boldsymbol{\beta}$; \mathbf{X} je designová matice fixních efektů (sem patří prakticky všechny uvažované regresory), $\boldsymbol{\beta}$ je matice koeficientů fixních efektů, \mathbf{u} je vektor koeficientů náhodných efektů, též někdy nepřesně nazývaný *grouping vector* tak, že platí $\mathbf{E}(\mathbf{u}) = \mathbf{0}$ a $\text{var}(\mathbf{u}) = \mathbf{G}$; \mathbf{Z} je designová matice náhodných efektů (v našem případě sem patří primární klíč `parcelid` identifikující nemovitosti v datasetu s možnými opakoványmi prodeji některých nemovitostí), $\boldsymbol{\varepsilon}$ je vektor náhodných chyb tak, že platí $\mathbf{E}(\boldsymbol{\varepsilon}) = \mathbf{0}$ a $\text{var}(\boldsymbol{\varepsilon}) = \mathbf{R}$.

Za předpokladů $\mathbf{u} \sim \mathcal{N}(\mathbf{0}, \mathbf{G})$, $\boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{R})$ a $\text{cov}(\mathbf{u}, \boldsymbol{\varepsilon}) = \mathbf{0}$ odvodil Henderson [10] řešení rovnice (1) jako řešení následující rovnosti

$$\begin{pmatrix} \mathbf{X}^T \mathbf{R}^{-1} \mathbf{X} & \mathbf{X}^T \mathbf{R}^{-1} \mathbf{Z} \\ \mathbf{Z}^T \mathbf{R}^{-1} \mathbf{X} & \mathbf{Z}^T \mathbf{R}^{-1} \mathbf{Z} + \mathbf{G}^{-1} \end{pmatrix} \begin{pmatrix} \hat{\boldsymbol{\beta}} \\ \hat{\mathbf{u}} \end{pmatrix} = \begin{pmatrix} \mathbf{X}^T \mathbf{R}^{-1} \mathbf{y} \\ \mathbf{Z}^T \mathbf{R}^{-1} \mathbf{y} \end{pmatrix}.$$

Odhady $\hat{\boldsymbol{\beta}}$ a $\hat{\mathbf{u}}$ jsou nejlepší lineární nezkreslené (i jako prediktory), tedy BLUE (best linear unbiased estimates) i BLUP (best linear unbiased predictors), [10].

V R je možné získat model se smíšenými efekty pomocí funkce `lmer()` balíčku `lme4`. Pomocí funkce `step()` balíčku `lmerTest` je pak možné získat ve smyslu *backward selekce* i „nejlepší“ model co do minimalizace Akaikeho informačního kritéria (AIC), [11].

Diagnostiku vhodnosti modelu provedeme graficky; zkонтrolujeme především předpoklady o chybové složce modelu ($\mathbf{E}(\mathbf{u}) = \mathbf{0}$ a $\text{cov}(\mathbf{X}, \boldsymbol{\varepsilon}) = \mathbf{0}$). Protože však pro účely predikce (nikoliv však inference!) není porušení těchto předpokladů tak zásadní, [5], spokojíme se v podstatě s jakýmkoliv solidně predikujícím modelem.

Predikční přesnost modelu budeme hodnotit stejně jako rankovací mechanismus soutěže kaggle [1]; spočítáme tedy jak na datasetu `train_set`, tak datasetu `test_set` vždy střední absolutní chybu (MAE) ve smyslu

$$\text{MAE} = \frac{\sum_{i=1}^n |y_i - \hat{y}_i|}{n} = \frac{\sum_{i=1}^n |e_i|}{n},$$

kde y_i jsou postupně všechny hodnoty závisle proměnné (`logerror`), \hat{y}_i jsou postupně všechny vyrovnané hodnoty závisle proměnné, $e_i = y_i - \hat{y}_i$ jsou konkrétní realizace chybové složky a n je celkový počet pozorování.

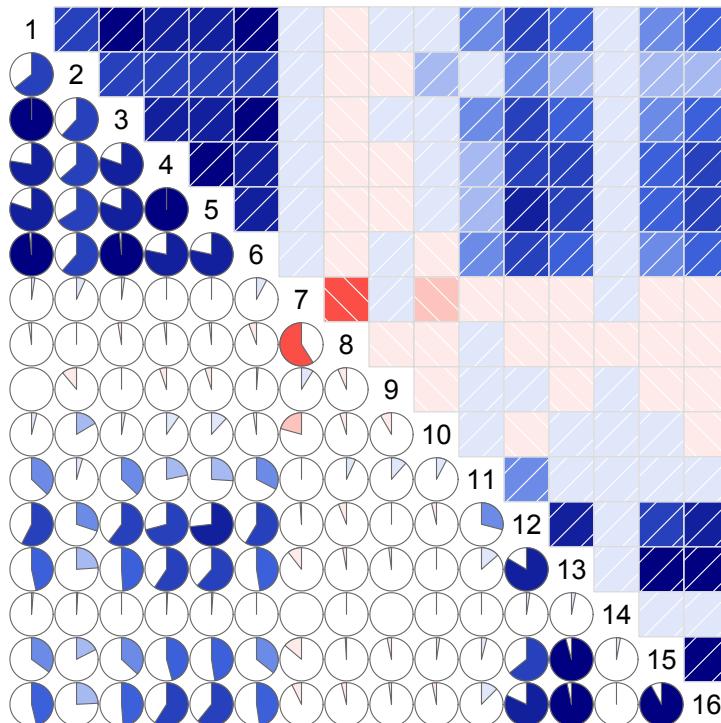
Model smíšených efektů a jeho diagnostiku provádí skript `hierarchical_model.R`. Syntaxe modelů smíšených efektů v R je poněkud triková a základní paradigma má tvar `závisle_proměnná ~ (primární_proměnné | grupovací_proměnné)`, kde `primární_proměnné` jsou obvykle fixní efekty a `grupovací_proměnné` jsou obvykle náhodné efekty. Další poučení o modelech smíšených efektů v R lze hledat v [12].

3 Výsledky

3.1 Průzkumová analýza dat

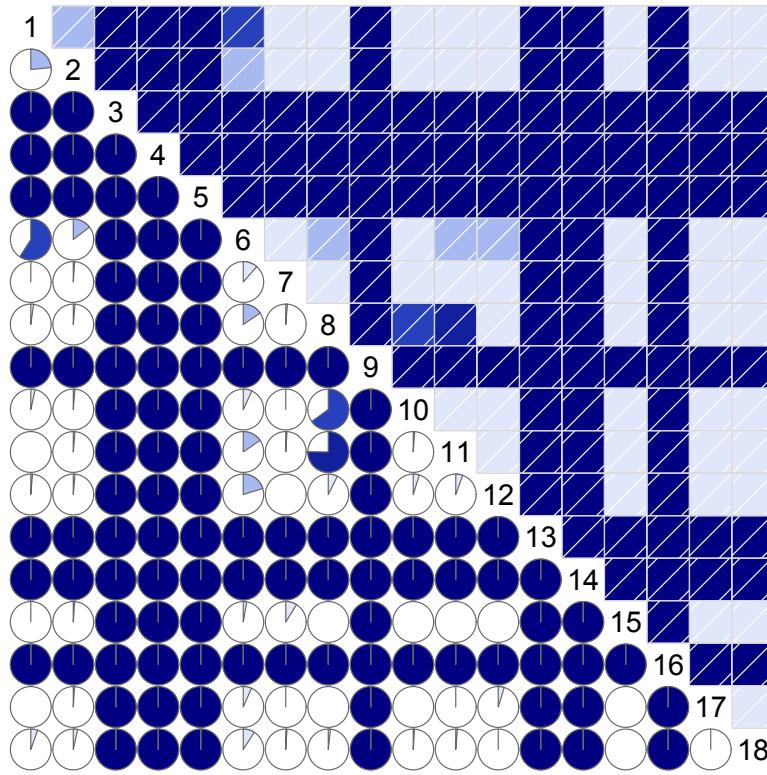
Vzhledem k velkému množství chybějících hodnot (**NA**) prakticky u všech regresorů, byly do modelu zahrnuty pouze ty, které měly jako chybějící nejvýše 25 % svých hodnot. Tím pro účely modelu zůstaly k dispozici pouze ty proměnné, nad kterými byly sestaveny korelační matice.

Náhledy na obě korelační matice, tj. na korelační maticí numerických prediktorů a kategorických prediktorů, u kterých je maximálně 25 % jejich hodnot kódováno jako chybějící, jsou na obrázcích 1 a 2.



Obrázek 1: Koreogram sestavený nad maticí Pearsonových korelačních koeficientů mezi numerickými proměnnými. Modré odstíny značí pozitivní korelaci, červené odstíny negativní korelaci. Sytost je přímo úměrná síle korelace. Koláčové diagramy vyjadřují velikost korelačního koeficientu jako na ciferníku hodin (tj. např. 3:00 odpovídá korelačnímu koeficientu 0,25). Číselná označení numerických proměnných jsou 1 = `bathroomcnt`, 2 = `bedroomcnt`, 3 = `calculatedbathnbr`, 4 = `calculatedfinishedsquarefeet`, 5 = `finishedsquarefeet12`, 6 = `fullbathcnt`, 7 = `latitude`, 8 = `longitude`, 9 = `lotsizesquarefeet`, 10 = `roomcnt`, 11 = `yearbuilt`, 12 = `structuretaxvaluedollarcnt`, 13 = `taxvaluedollarcnt`, 14 = `assessmentyear`, 15 = `landtaxvaluedollarcnt`, 16 = `taxamount`.

Na koreogramu 1 snadno nahlédneme, že spolu silně korelují numerické regresory s indexy 2, 3, 4, 5, 6 a dále pak 13, 15, 16. Z první skupiny byl pro účely modelu se smíšenými efekty ponechán jen regresor s indexem 2 (`bedroomcnt`), z druhé jen s indexem 13 (`taxvaluedollarcnt`).



Obrázek 2: Koreogram sestavený nad maticí Cramérových V mezi kategorickými proměnnými. Modré odstíny značí pozitivní korelace, červené odstíny negativní korelace. Sytost je přímo úměrná síle korelace. Koláčové diagramy vyjadřují velikost korelačního koeficientu jako na ciferníku hodin (tj. např. 3:00 odpovídá korelačnímu koeficientu 0,25). Číselná označení kategorických proměnných jsou 1 = `transaction_year`, 2 = `transaction_month`, 3 = `airconditioningtypeid`, 4 = `architecturalstyletypeid`, 5 = `buildingclasstypeid`, 6 = `buildingqualitytypeid`, 7 = `decktypeid`, 8 = `hashottuborspa`, 9 = `heatingorsystemtypeid`, 10 = `pooltypeid10`, 11 = `pooltypeid2`, 12 = `pooltypeid7`, 13 = `propertycountylandusecode`, 14 = `propertylandusetypeid`, 15 = `storytypeid`, 16 = `typeconstructiontypeid`, 17 = `fireplacedflag`, 18 = `taxdelinquencyflag`.

V případě kategorických proměnných dle koreogramu 2 vidíme celkem zřejmě, že proměnné s indexy 3, 4, 5, 9, 13, 14 a 16 spolu prakticky funkčně (absolutně) korelují. Ze této skupiny ponechme pro účely modelu smíšených efektů jen proměnnou s indexem 3 (`airconditioningtypeid`).

Ostatní numerické a kategorické proměnné, které nebyly dosavadním postupem vyloučeny pro účely modelu se smíšenými efekty, rovněž považujme za možné regresory.

3.2 Model smíšených efektů

Model smíšených efektů. Pro správnou interpretaci modelu smíšených efektů dle rovnice (1) je třeba designovou matici \mathbf{X} fixních efektů vnímat tak, že spojuje numerické i kategorické proměnné (ve sloupcích) – pak je ale nutné na kategorické proměnné pohlížet tak, že jejich jednotlivé

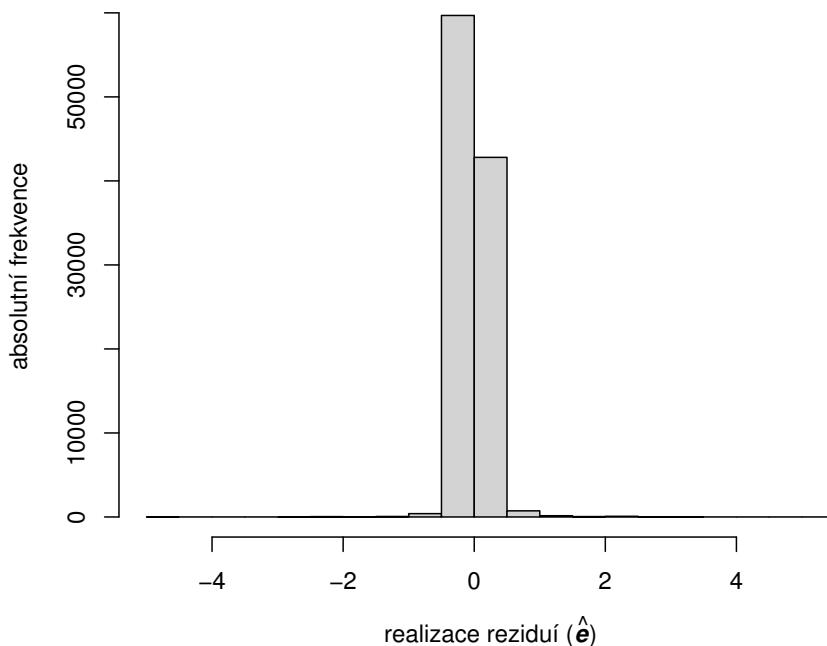
hodnoty jsou kódovány ve smyslu *dummy kódování* jako samostatné nové (dummy) proměnné vždy s příslušným koeficientem $\beta_{i;d}$, kde i je index daného kategorického regresoru a d je index některé jeho hodnoty, pro kterou je v dummy kódování daná identifikátorová dummy rovna jedné. Pak rovnice (1) plně vystihuje model s fixními numerickými i kategorickými regresory (a není třeba \mathbf{X} ani $\boldsymbol{\beta}$ rozdělovat zvlášť pro numerické a kategorické proměnné).

V tabulce 1 je sumář modelu se smíšenými efekty. Z tabulky je tedy možné vyčíst všechny uvažované fixní efekty. Jako náhodný efekt byla použita proměnná *parcelid* jednoznačně identifikující každou nemovitost. Její zahrnutí mezi náhodné efekty vede k odhadnutí individuální hladiny závisle proměnné, čímž ošetří situaci, kdy se v rámci trénovacího datasetu opakuje prodej dané nemovitosti (byť se v některých jiných proměnných může lišit – zejména v čase prodeje a zaznamenané *logerror*). Protože předmětem našeho zájmu je predikce, jednotlivé odhady regresních koeficientů nebudeme přímo interpretovat, přestože je zde interpretace jednoduchá – při změně i -tého regresoru o jednotku se změní závislá proměnná *logerror* o $\hat{\beta}_i$.

Shodou okolností se jedná i o model s nejnižším možným AIC, jak bylo ověřeno použitím funkce `step()` balíčku `lmerTest`.

Diagnostika modelu. Na obrázcích 3, 4 a 5 vidíme postupně histogram reziduů, kvantil-kvantil diagram reziduů a diagram reziduů proti vyrovnaným hodnotám. Rezidua a vyrovnané hodnoty pochází z modelu smíšených efektů. Rozložení realizací reziduů je sice symetrické, ale není normální, je podstatně špičatější. Rozložení realizací reziduů proti vyrovnaným hodnotám nevykazuje nápadný trend, ale je zde náznak centrálního nakupení kolem střední hodnoty vyrovnaných hodnot.

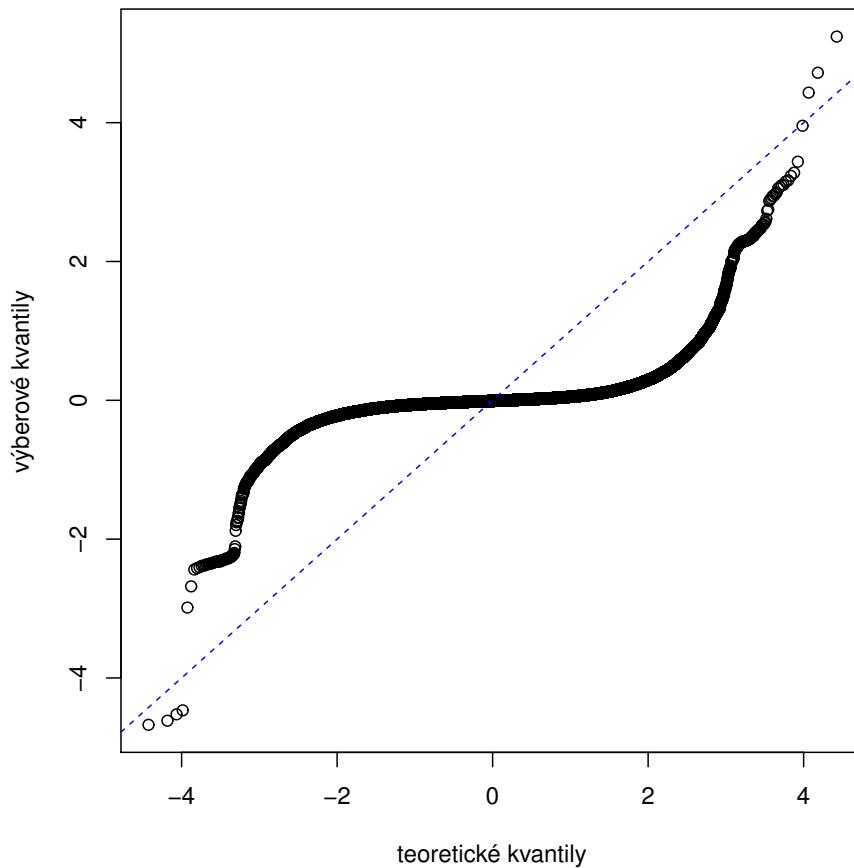
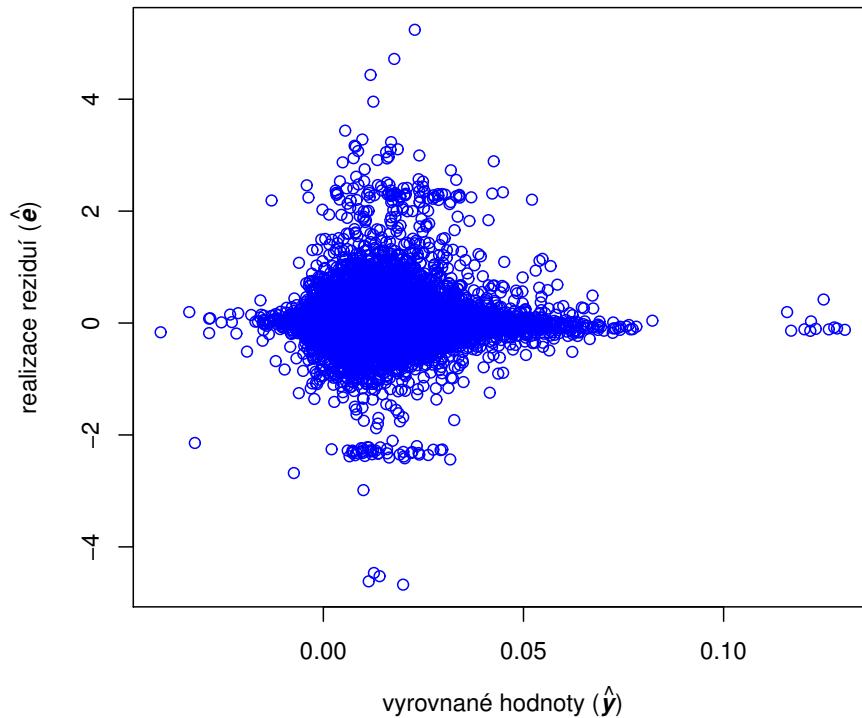
Predikční přesnost. Pro trénovací dataset `train_set` (70 % původních dat), na kterém byl model vytvořen, má spočítaná střední absolutní chyba hodnotu $MAE \doteq 0,07098$ a pro testovací dataset `test_set` (zbylých 30 % původních dat) má spočítaná střední absolutní chyba hodnotu $MAE \doteq 0,07038$.



Obrázek 3: Histogram realizací reziduů \hat{e} .

	odhad	střední chyba	df	t-hodnota	p-hodnota
$\hat{\beta}_0$	0,004	0,006	104002	0,679	0,497
$\hat{\beta}_1$ (bathroomcnt)	0,076	0,017	104002	4,414	< 0,001
$\hat{\beta}_2$ (bedroomcnt)	0,019	0,010	104002	1,883	0,060
$\hat{\beta}_3$ (latitude)	-0,002	0,004	104002	-0,537	0,591
$\hat{\beta}_4$ (longitude)	0,005	0,004	104002	1,467	0,142
$\hat{\beta}_5$ (lotsizesquarefeet)	0,111	0,030	104002	3,692	< 0,001
$\hat{\beta}_6$ (roomcnt)	-0,004	0,005	104002	-0,886	0,376
$\hat{\beta}_7$ (yearbuilt)	0,006	0,005	104002	1,278	0,201
$\hat{\beta}_8$ (structuretaxvaluedollarcnt)	0,064	0,048	104002	1,332	0,183
$\hat{\beta}_9$ (taxvaluedollarcnt)	-0,114	0,045	104002	-2,531	0,011
$\hat{\beta}_{10;2}$ (transaction_year = 2017)	0,009	0,001	104002	6,301	< 0,001
$\hat{\beta}_{11;2}$ (transaction_month = 2)	-0,002	0,003	104002	-0,600	0,549
$\hat{\beta}_{11;3}$ (transaction_month = 3)	-0,006	0,002	104002	-2,434	0,015
$\hat{\beta}_{11;4}$ (transaction_month = 4)	-0,010	0,002	104002	-4,072	< 0,001
$\hat{\beta}_{11;5}$ (transaction_month = 5)	-0,012	0,002	104002	-5,254	< 0,001
$\hat{\beta}_{11;6}$ (transaction_month = 6)	-0,010	0,002	104002	-4,188	< 0,001
$\hat{\beta}_{11;7}$ (transaction_month = 7)	-0,007	0,002	104002	-2,982	0,003
$\hat{\beta}_{11;8}$ (transaction_month = 8)	-0,005	0,002	104002	-1,979	0,048
$\hat{\beta}_{11;9}$ (transaction_month = 9)	-0,000	0,003	104002	-0,018	0,985
$\hat{\beta}_{11;10}$ (transaction_month = 10)	-0,002	0,004	104002	-0,698	0,485
$\hat{\beta}_{11;11}$ (transaction_month = 11)	-0,003	0,005	104002	-0,637	0,524
$\hat{\beta}_{11;12}$ (transaction_month = 12)	0,002	0,005	104002	0,305	0,760
$\hat{\beta}_{12;2}$ (aircondition = Central)	-0,002	0,002	104002	-1,040	0,298
$\hat{\beta}_{12;3}$ (aircondition = None)	-0,004	0,011	104002	-0,413	0,680
$\hat{\beta}_{12;4}$ (aircondition = Refrigeration)	-0,013	0,037	104002	-0,347	0,729
$\hat{\beta}_{12;5}$ (aircondition = Wall Unit)	0,005	0,030	104002	0,183	0,855
$\hat{\beta}_{12;6}$ (aircondition = Yes)	-0,001	0,004	104002	-0,317	0,752
$\hat{\beta}_{13;2}$ (buildingquality = 1)	-0,002	0,005	104002	-0,413	0,679
$\hat{\beta}_{13;3}$ (buildingquality = 2)	0,029	0,056	104002	0,521	0,602
$\hat{\beta}_{13;4}$ (buildingquality = 3)	0,032	0,011	104002	2,774	0,006
$\hat{\beta}_{13;5}$ (buildingquality = 4)	-0,003	0,003	104002	-1,118	0,264
$\hat{\beta}_{13;6}$ (buildingquality = 5)	0,002	0,005	104002	0,346	0,729
$\hat{\beta}_{13;7}$ (buildingquality = 6)	-0,009	0,003	104002	-3,192	0,001
$\hat{\beta}_{13;8}$ (buildingquality = 7)	-0,006	0,003	104002	-2,244	0,025
$\hat{\beta}_{13;9}$ (buildingquality = 8)	-0,014	0,003	104002	-4,678	< 0,001
$\hat{\beta}_{13;10}$ (buildingquality = 9)	-0,011	0,005	104002	-2,284	0,022
$\hat{\beta}_{13;11}$ (buildingquality = 10)	-0,005	0,006	104002	-0,836	0,403
$\hat{\beta}_{13;12}$ (buildingquality = 11)	-0,009	0,007	104002	-1,251	0,211
$\hat{\beta}_{13;13}$ (buildingquality = 12)	-0,023	0,014	104002	-1,676	0,094
$\hat{\beta}_{14;2}$ (hashottuborspa = true)	-0,013	0,003	104002	-3,633	< 0,001

Tabulka 1: Odhadové regresních koeficientů pro model smíšených efektů. Symbol *df* značí počet stupňů volnosti. Názvy některých regresorů zkráceny kvůli velikosti tabulky.

Obrázek 4: Kvantil-kvantil diagram realizací reziduí \hat{e} .Obrázek 5: Diagram realizací reziduí \hat{e} proti vyrovnaným hodnotám \hat{y} .

4 Diskuze

Z metodologického úhlu pohledu nejde o příliš složitý model. Oproti obyčejné mnohorozměrné regresi bere v potaz možnost opakovaných záznamů pro nemovitosti, které byly ve sledovaném období prodány vícekrát než jednou.

Modelování celého procesu jako časové řady s prvky regrese se nezdá příliš vhodné, protože jde ve skutečnosti o velké množství ne nutně nezávislých časových řad pro jednotlivé nemovitosti, ovšem krok těchto řad není jednotný a mnohdy ani známý (máme obvykle jen jedno pozorování pro danou nemovitost v daném časovém období).

Vybudovaný model se smíšenými efekty není stran statistické inference úplně udržitelný, ale protože nám jde především o predikci, je model i přes porušení předpokladů kladených na realizace reziduů akceptovatelný. Realizace reziduů daného modelu nesledují normální rozdělení, přesto je jejich rozdělení alespoň symetrické, ale poněkud špičatější než normální. Rovněž nesplňují předpoklad homoskedasticity. Dosažená střední absolutní chyba je prakticky stejná nad trénovacími i testovacími daty; protože není nad testovacími výrazně větší, lze očekávat, že predikce závisle proměnné **logerror** skutečně vrací smysluplné výsledky.

Porovnáním s hodnotami středních absolutních chyb, které získávají ostatní účastníci soutěže na kaggle.com, lze konstatovat, že vytvořený model si zdaleka nestojí špatně. Hodnoty středních absolutních chyb vedoucích soutěžících se pohybují kolem hodnot 0,07, jsou však počítány na zcela nových datech – je tedy nutné kriticky očekávat, že vytvořený model smíšených efektů si na takových nových datech povede nejspíše poněkud hůře a konečná střední absolutní chyba bude větší než uvedený odhad.

Lze předpokládat, že model, který bude v rámci možností co nejvíce minimalizovat střední absolutní chybu mezi skutečnou a odhadnutou závisle proměnnou **logerror** (a soutěž na kaggle.com dost možná vyhraje), bude založen na strojovém učení. Vzhledem k uváděné přesnosti predikce a množství vyzkoušených modelů společnosti Zillow se nabízí, že jejich predikční model využívá *neuronové sítě s velkým počtem skrytých vrstev a rekurentní axonální propagaci*, nebo alespoň rozsáhlou *random forest regresi*. Výhodou těchto modelů je vysoká přesnost predikce plynoucí z jejich komplexnosti, nevýhodou pak prakticky nulová možnost smysluplné interpretace architektury modelu.

Zde je vhodné si říct i to, že tak komplexní modely se nevytváří v jedné desktopové seanci, ale dnes obvykle pomocí gridového nebo lépe cloudového řešení a pomocí jiných jazyků než je R (to je díky tomu, že jde o interpretovaný jazyk (tedy původně napsaný v *low-level* jazyce C++, ale „zpomalený“ první komplikací a interpreterem umožňujícím používat jednodušší, R-kovou syntaxi), relativně „pomalé“ a musí ukládat veškeré objekty do RAM) – obvykle se používá knihovna Keras jazyka Python [13], vývojářské prostředí TensorFlow vytvořené společností Google [14] anebo top framework současnosti pro machine learning – open-source nástroj CAFFE (Convolutional Architecture for Fast Feature Embedding) vyvinutý univerzitou v Berkley [15].

5 Závěr

Odhad tržní ceny nemovitosti je důležitým nástrojem pro její objektivní ohodnocení na trhu, zvláště pokud nebyl nemovitost dosud naceněna, nikdy nebyla předmětem obchodní transakce, anebo je její cena záměrně neuvedena.

Na americkém trhu nemovitostí je používán pro odhad tržních cen nemovitosti tzv. Zillow's estimate (*Zestimate*), který vychází z velkého řady sledovaných vlastností nemovitostí a obrovského množství dat a prozkoušených modelů.

Portál kaggle.com ve spolupráci se společností Zillow přesto otevřel veřejnou soutěž, kde je možné pokusit se o vylepšení odhadu cen nemovitostí.

Data, která portál kaggle.com pro tyto účely nabízí, jsou rozsáhlá a vyžadují předzpracování. Numerické proměnné byly standardizovány převodem na intervaly $\langle 0, 1 \rangle$. Pro účely vytvoření modelu byly vyloučeny ty proměnné, které spolu silně korelují. Korelace byly vypočteny pomocí Pearsonova korelačního koeficientu a Cramérova V . U časové proměnné udávající s přesnosti na konkrétní den, kdy byla nemovitost prodána, byla zvýšena granularita pouze na rok a měsíc.

Na takto upravených datasetech byl poté vytvořen model se smíšenými efekty, který bere v potaz některé z evidovaných vlastností nemovitostí jako fixní efekty, a ošetřuje odhad závisle proměnné, tj. rozdílu mezi logaritmem odhadu tržní ceny nemovitosti a logaritmem její skutečné hodnoty, o náhodný efekt daný individuální hladinou tohoto rozdílu pro každou nemovitost.

Vytvořený model není příliš použitelný pro účely statistické inference; avšak pro účely predikce vykazuje solidní výsledky. Predikční přesnost měřená střední absolutní chybou mezi odhadnutou a skutečnou hodnotou závisle proměnné je srovnatelná s ostatními modely, které se soutěže účastní.

6 Appendix

Zde je uveden kód v jazyce R, ve kterém byly zpracovávány veškeré výpočty a rovněž generovány diagramy.

6.1 __main.R__

```
#####
#####
#####
#####
#####

## nutné spouštět v R 64-bit !!! ----

#####
#####

## nastavuje pracovní složku ----

while(!grepl("__seminarni_prace__$", getwd())){

    setwd(choose.dir())

}

mother_working_directory <- getwd()

## ----

#####
#####

## spouštím sekvenci skriptů ----

for(my_script in c(

    "initialization",                      # spouštím inicializaci
    "helper_functions",                     # definuji pomocné funkce
    "data_loading",                         # loaduju data
    "data_processing",                      # upravuji data, vytvářím některé výstupy
    "data_saving_and_reloading",            # ukládám zpracovaná data
    "data_postprocessing",                  # upravuji data do finální podoby
    "exploratory_data_analysis",           # průzkumová analýza dat
    "hierarchical_model"                   # vytvářím hierarchický model

)) {

    setwd(mother_working_directory)

    source(

```

```

    paste(my_script, ".R", sep = ""),
    echo = TRUE,
    encoding = "UTF-8",
    max.deparse.length = Inf
  )
}

## -----
#####
#####
```

6.2 initialization.R

```

#####
#####
#####

## instaluju a inicializuju balíčky ----

for(my_package in c(
  "openxlsx",
  "xtable",
  "data.table",
  "corrgram",
  ##"forecast",
  ##"astsa",
  ##"vars",
  "lme4",
  "lmerTest"
)) {

  if(!(my_package %in% rownames(installed.packages()))){

    install.packages(
      my_package,
      dependencies = TRUE,
      repos = "http://cran.us.r-project.org"
    )
  }

  library(my_package, character.only = TRUE)
}
```

```
}

## ----

#####
## nastavuji handling se zipováním v R ----

Sys.setenv(R_ZIPCMD = "C:/Rtools/bin/zip")

## ----

#####
## zakládám podsložku "výstupy" ----

for(my_directory in c("vystupy")){
  if(!file.exists(my_directory)){
    dir.create(file.path(
      mother_working_directory,
      my_directory
    ))
  }
}

## ----

#####
```

6.3 helper_functions.R

```
#####
## definuji pomocné funkce ----
```

```

isPotentiallyNumeric <- function(
  x,
  original.na.mark = ""

){

  # '''
  # Vrací TRUE tehdy a jen tehdy, pokud po koerci vektoru textových
  # hodnot "x" na datový typ "numeric" neuznáknou nové chybějící
  # hodnoty (NA).
  # Argument "original.na.mark" vyjadřuje, jak jsou značeny
  # chybějící hodnoty v původním vektoru "x".
  # '''

  return(
    !any(
      is.na(
        suppressWarnings(
          as.numeric(
            x[
              x != original.na.mark
            ]
          )
        )
      )
    )
  )
}

## -----
#####
#####
```

6.4 data_loading.R

```

#####
#####
#####

## nastavuji složku se vstupy ----

setwd(paste(mother_working_directory, "vstupy", sep = "/"))
```

```
## loaduji data -----
##### nejdříve nahrávám všechny .csv, pak .xlsx soubory -----
for(my_suffix in c(".csv", ".xlsx")){
  for(
    my_filename in dir(){
      grep(
        paste("\\" , my_suffix, "$", sep = ""),
        dir()
      ) &
      !grep(
        "_processed",
        dir()
      )
    }
  )
}

if(
  my_suffix == ".csv"
){

  ## nahrávám všechny .csv soubory ----

  if(
    !(
      paste(
        gsub(
          paste("\\" , my_suffix, "$", sep = ""),
          "",
          my_filename
        ),
        "_processed",
        my_suffix,
        sep = ""
      ) %in% dir()
    )
  ){
    assign(
      gsub(
        paste("\\" , my_suffix, "$", sep = ""),
        "",
        my_filename
      ),
      read.table(

```

```
        file = my_filename,
        header = TRUE,
        sep = ",",
        dec = ".",
        row.names = NULL,
        comment.char = "",
        check.names = FALSE,
        colClasses = "character"

    )

}

}

}else{

## nahrávám všechny listy všech .xlsx souborů; předpokládám,
## že jména všech listů jsou navzájem různá ----

for(
    my_sheetname in getSheetNames(my_filename)
){

    my_data <- read.xlsx(

       xlsxFile = my_filename,
        sheet = my_sheetname,
        colNames = TRUE,
        rowNames = FALSE,
        check.names = FALSE

    )

    for(i in 1:dim(my_data)[2]){

        my_data[, i] <- as.character(my_data[, i])

    }

    assign(
        my_sheetname,
        my_data
    )

}

}
```

```

}

## logovací hlášky -----
flush.console()

print(
  paste(
    "Právě nahrán ",
    if(
      grepl("\\.csv$", my_filename)
    ){
      "soubor"
    }else{
      "list"
    },
    " ",
    my_filename,
    ".",
    sep = ""
  )
)

}

## ----

setwd(mother_working_directory)

## ----

#####
#####
```

6.5 data_processing.R

```

#####
#####
```

```
## nastavuji složku se vstupy -----
setwd(paste(mother_working_directory, "vstupy", sep = "/"))

## ----

##### následující procedura proběhne pouze na nativních datech, tj.
##### bez přípony "_processed" ----

if(
  !all(
    c(
      "properties_2016_processed.csv",
      "properties_2017_processed.csv"
    ) %in% dir()
  )
){

  ##### nejdříve určuji datové typy v datasetech "properties_2016"
  ##### a "properties_2017"; jde-li o kategorickou proměnnou, kóduju ji
  ##### ve smyslu self-labelingu podle codebooku "zillow_data_dictionary" ----

  if(
    all(colnames(properties_2016) == colnames(properties_2017))
  ){

    ## pro každý z datasetů "properties_2016" a "properties_2017" měním
    ## datové typy na nejvhodnější možné ----

    for(
      my_dataset_name in paste(
        "properties_",
        c("2016", "2017"),
        sep = ""
      )
    ){

      my_data <- get(my_dataset_name)

      ## pro každou proměnné v datasetu se ptám, zda je dle codebooku
      ## "zillow_data_dictionary" dozajista kategorická ----

      for(my_variable in colnames(my_data)){

        if(
          my_variable %in% tolower(ls()[grep("ID$", ls())])
        ){

```

```

## pokud kategorická určitě je, překódovávám její
## kategorie podle codebooku ve smyslu self-labelingu ----

my_codebook <- get(
  ls()[grep("ID$", ls())][
    tolower(
      ls()[grep("ID$", ls())]
    ) == my_variable
  ]
)

for(my_level in my_codebook[, 1]){

  my_data[
    which(my_data[, my_variable] == my_level),
    my_variable
  ] <- my_codebook[
    my_codebook[, 1] == my_level,
    2
  ]

}

## nakonec ji ukládám jako datový typ faktor ----

my_data[, my_variable] <- as.factor(
  as.character(
    my_data[, my_variable]
  )
)

}else{

  ## pokud není podle codebooku apriorně kategorická,
  ## testuji, zda nemůže být určitě numerická,
  ## v případě falešné pozitivivy, tedy absence nově
  ## vzniklých chybějících hodnot po koerci
  ## na reálná čísla, by taková proměnná jistě byla
  ## v codebooku -- to už je ale nyní vyloučeno, je tedy
  ## numerická ----

  if(
    isPotentiallyNumeric(
      my_data[, my_variable]
    ) &
    !((
      grep("id[0-9]*$", my_variable)
    ) &

```

```
!(  
  grep(   
    "(block|city|county|hodd|zip)$",  
    my_variable  
  )  
)  
{  
  
  ## proměnnou ukládám jako "numeric" -----  
  
  my_data[, my_variable] <- as.numeric(  
    as.character(  
      my_data[, my_variable]  
    )  
  )  
  
}  
  
}else{  
  
  ## ve zbývajícím případě ji musím ošetřit jako  
  ## kategorickou proměnnou, ale bez self-labeling  
  ## kódování kategorií -----  
  
  my_data[, my_variable] <- as.factor(  
    as.character(  
      my_data[, my_variable]  
    )  
  )  
  
}  
  
}  
  
}  
  
## logovací hlášky -----  
  
flush.console()  
  
print(  
  paste(  
    my_dataset_name,  
    ": proměnná '",  
    my_variable,  
    "' přetypována na '",  
    class(my_data[, my_variable]),  
    "'.",  
    sep = "  
  )  
)
```

```
}

## ukládám nově upravený dataset zpátky pod jméno původního
## datasetu -----
assign(
  my_dataset_name,
  my_data
)

}

#### manuálně opravuji nesprávnou automatickou koerci -----
for(
  my_dataset_name in paste("properties_", c("2016", "2017"), sep = ""))
{

  my_data <- get(my_dataset_name)

  ## následující proměnné jsou kategorické -----
  for(my_variable in c(
    "fips"
  )){

    my_data[, my_variable] <- as.factor(
      as.character(
        my_data[, my_variable]
      )
    )

    ## logovací hlášky -----
    flush.console()

    print(
      paste(
        my_dataset_name,
        ": proměnná ''",
        sep = ""
      )
    )
  }
}
```

```
        my_variable,
        "' přetypována na '",
        class(my_data[, my_variable]),
        "' .",
        sep = ""
    )
}

## ukládám nově upravený dataset zpátky pod jméno původního
## datasetu -----
assign(
    my_dataset_name,
    my_data
)
}

## -----
setwd(mother_working_directory)

## -----
#####-----

## upravuji datasety "train_2016_v2" a "train_2017" -----
for(my_filename in c("train_2016_v2", "train_2017")){
    ## inicializuje dataset -----
    my_data <- get(my_filename)

    ## přetypovávám jednotlivé proměnné na správné datové typy -----
    my_data[, "parcelid"] <- as.factor(my_data[, "parcelid"])

    my_data[, "logerror"] <- as.numeric(my_data[, "logerror"])

    my_data[, "transactiondate"] <- as.Date(
```

```

    my_data[, "transactiondate"],
    format = "%Y-%m-%d"
)

## ukládám zpětně dataset pod původní název ----

assign(
  my_filename,
  my_data
)

## logovací hlášky ----

flush.console()

print(
  paste(
    "Právě zpracován dataset ''",
    my_filename,
    "'.'",
    sep = ""
  )
)

}

## ----

#####
#####
```

6.6 data_saving_and_reloading.R

```

#####
#####
#####

## nastavuji složku se vstupy ----

setwd(paste(mother_working_directory, "vstupy", sep = "/"))

## ----
```

```
##### pokud není v pracovní složce alespoň jeden ze souborů
##### "properties_2016_processed" či "properties_2017_processed", jsou
##### zpracovaná data uložena pod těmito jmény, aby byla napříště
##### pouze nahrána bez nutného processingu (asynchronní úloha) -----
##### pokud jsou oba soubory v pracovní složce přítomny, jsou nahrány včetně
##### datových typů všech jejich proměnných (bez nutného processingu) ----

if(
  !all(
    c(
      "properties_2016_processed.csv",
      "properties_2017_processed.csv"
    ) %in% dir()
  )
){}

for(
  my_dataset_name in paste(
    "properties_",
    c("2016", "2017"),
    sep = ""
  )
){

  ## ukládám datové typy proměnných v každém souboru -----
  my_data <- get(my_dataset_name)

  my_classes <- NULL

  for(i in 1:dim(my_data)[2]){
    my_classes <- c(
      my_classes,
      class(my_data[, i])
    )
  }

  writeLines(
    text = my_classes,
    con = paste(
      my_dataset_name,
      "_column_classes",
      ".txt",
      sep = ""
    )
  )
}
```

```
        sep = ""
    )
)

## nyní ukládám vždy celý dataset pomocí rychlého uložení -----
fwrite(
    x = my_data,
    file = paste(
        my_dataset_name,
        "_processed",
        ".csv",
        sep = ""
    ),
    sep = ";"
)

## logovací hlášky -----
flush.console()

print(
    paste(
        "Právě uložen dataset ''",
        my_dataset_name,
        "_processed",
        ".csv",
        "'.'",
        sep = ""
    )
)
}

}else{
    ## pokud ale oba soubory se zpracovanými daty již v pracovní složce
    ## existují, konečně je nahtávám pomocí rychlého nahrání (a bez nutného
    ## processingu)
    ## předtím je samozřejmě nutné nahrát soubory s datovými typy
    ## proměnných -----
    for(
```

```
my_dataset_name in paste(
  "properties_",
  c("2016", "2017"),
  sep = ""
)
){

## nejdříve nahrávám soubory s datovými typy proměnných datasetů ----

my_column_classes <- readLines(

  con = paste(
    my_dataset_name,
    "_column_classes",
    ".txt",
    sep = ""
  ),
  encoding = "UTF-8"
)

## nyní nahrávám oba datasety pomocí rychlého čtení ----

assign(
  my_dataset_name,
  data.frame(
    fread(
      input = paste(
        my_dataset_name,
        "_processed",
        ".csv",
        sep = ""
      ),
      sep = ";",
      header = TRUE,
      check.names = FALSE,
      encoding = "UTF-8",
      colClasses = my_column_classes
    )
  )
)

## logovací hlášky ----

flush.console()
```

```

    print(
      paste(
        "Právě načten dataset ",
        my_dataset_name,
        "_processed",
        ".csv",
        "'.",
        sep = ""
      )
    )

}

## ----

setwd(mother_working_directory)

## ----

#####
#####
```

6.7 data_postprocessing.R

```

#####
#####
```

```

## meruje vždy dataset "properties_2016" s "train_2016_v2" a dále
## "properties_2017" s "train_2017" ----

for(my_year in c("2016", "2017")){
  assign(
    paste(
      "merged_data_",
      my_year,
      sep = ""
    ),
    merge(
      x = get(

```

```
paste(
    "train_",
    my_year,
    if(my_year == "2016"){"_v2"},
    sep = ""
)
),
y = get(
    paste(
        "properties_",
        my_year,
        sep = ""
    )
),
by = "parcelid",
all.x = TRUE      # left outer join,
# tj. budou zachována všechna "parcelid" z "x",
# ale ne nutně všechna "parcelid" z "y"
)
)

## logovací hlášky -----
flush.console()

print(
    paste(
        "Právě joinovány datasety ''",
        paste(
            "train_",
            my_year,
            if(my_year == "2016"){"_v2"},
            sep = ""
        ),
        "' a ''",
        paste(
            "properties_",
            my_year,
            sep = ""
        ),
        "'.'",
        sep = ""
    )
)
}

}
```

```
## -----  
#####  
## vytvářím jeden velký dataset trénovacích dat -----  
  
train_data <- rbind(  
  
  merged_data_2016,  
  merged_data_2017  
  
)  
  
## -----  
#####  
## přidávám ještě proměnnou "transaction_year" a "transaction_month"  
## pro každé pozorování datasetu "train_data" -----  
  
train_data <- data.frame(  
  
  train_data[  
    ,  
    1:(which(colnames(train_data) == "transactiondate") - 1)  
  ],  
  "transaction_year" = format(  
    train_data[  
      ,  
      which(colnames(train_data) == "transactiondate")  
    ],  
    format = "%Y"  
  ),  
  "transaction_month" = factor(  
    format(  
      train_data[  
        ,  
        which(colnames(train_data) == "transactiondate")  
      ],  
      format = "%b"  
    ),  
    levels = as.character(c(1:12))  
  ),  
  train_data[  
    ,  
    (which(colnames(train_data) == "transactiondate") + 1):  
    dim(train_data)[2]  
  ]
```

```
)  
  
## -----  
  
#####  
#####  
#####
```

6.8 exploratory_data_analysis.R

```
#####
## nejdříve rozděluji data na trénovací a testovací množinu ----

#### do trénovací množiny zahrnu 70 % dat ----

train_set_portion <- 0.7

set.seed(2017)

#### vytvářím množinu indexů pozorování, která budou zahrnuta do trénovací
#### množiny ----

train_set_indices <- sample(
  c(1:dim(train_data)[1]),
  floor(dim(train_data)[1] * train_set_portion),
  replace = FALSE
)

#### vytvářím trénovací a testovací množinu ----

train_set <- train_data[
  train_set_indices
,
]

test_set <- train_data[
  setdiff(
    c(
      1:dim(train_data)[1]
    ),
  )]
```

```
    train_set_indices
)
,
]

## ----

#####
## definuji regresory zájmu ----

#### nejdříve numerické regresory zájmu ----

numeric_regressors_of_interest <- setdiff(
  colnames(train_data)[
    unname(
      unlist(
        lapply(
          1:dim(train_data)[2],
          function(i) class(train_data[, i])
        )
      )
    )
  ] == "numeric"
),
c(
  "logerror"
)
)

#### nyní kategorické regresory zájmu ----

factor_regressors_of_interest <- colnames(train_data)[
  unname(
    unlist(
      lapply(
        1:dim(train_data)[2],
        function(i) class(train_data[, i])
      )
    )
  ) == "factor"
][
  !grepl(
    "(parcelid|error|fips|desc|block|city|county|hood|zip)$",
    colnames(train_data)[
      unname(

```

```

        unlist(
          lapply(
            1:dim(train_data)[2],
            function(i) class(train_data[, i])
          )
        )
      ) == "factor"
    ]
)
]

## -----
#####
## vytipovávám numerické proměnné s největším podílem chybějících hodnot;
## ty bude nutné vyloučit před vytvořením modelu -----
which_to OMIT <- colnames(train_set)[
  apply(
    train_set,
    2,
    function(x) sum(is.na(x)) / length(x) * 100 # vrací, kolik procent
                                                 # dané proměnné tvoří
                                                 # chybějící hodnoty (NA)
  ) > 25
] # je-li procento chybějících hodnot dané proměnné větší než 25,
# danou proměnnou do modelu nezahrnu (a uložím ji do vektoru
# "which_to OMIT")

numeric_regressors_of_interest <- setdiff(
  numeric_regressors_of_interest,
  which_to OMIT
)

## -----
#####
## redukuji počet numerických regresorů na základě korelační struktury -----
#### počítám korelační matici -----
my_correlations <- suppressWarnings(

```

```
cor(
  x = train_set[, numeric_regressors_of_interest],
  method = "pearson",
  use = "pairwise.complete.obs"
)
)

#### kvůli vykreslení korelogramu nahrazuji nespočitatelné korelace (kvůli
#### chybějícím hodnotám, NA) nulami -----
my_correlations[is.na(my_correlations)] <- 0

#### ukládám koreogram -----
setwd(paste(mother_working_directory, "vystupy", sep = "/"))

cairo_ps(
  file = "koreogram_numeric.eps",
  width = 8,
  height = 8,
  pointsize = 14
)

par(mar = c(0.1, 0.1, 0.1, 0.1))

corrgram(
  x = my_correlations,
  order = FALSE,
  lower.panel = panel.pie,
  labels = 1:dim(my_correlations)[1]
)

dev.off()

setwd(mother_working_directory)

#### dle koreogramu vyřazuji z numerických regresorů ty, které silně
#### korelují s některými jinými regresory (kolinearita) -----
numeric_regressors_of_interest <- numeric_regressors_of_interest[
  -c(3, 4, 5, 6, 14, 15, 16)
  #-c(3, 4, 14)
]
```

```

##### nakonec ještě přeškálovávám numerické regresory na interval (0, 1) ----

train_set[
  ,
  numeric_regressors_of_interest
] <- apply(
  train_set[
    ,
    numeric_regressors_of_interest
],
  2,
  function(x){
    (
      x - min(x, na.rm = TRUE)
    ) / (
      max(x, na.rm = TRUE) - min(x, na.rm = TRUE)
    )
  }
)

test_set[
  ,
  numeric_regressors_of_interest
] <- apply(
  test_set[
    ,
    numeric_regressors_of_interest
],
  2,
  function(x){
    (
      x - min(x, na.rm = TRUE)
    ) / (
      max(x, na.rm = TRUE) - min(x, na.rm = TRUE)
    )
  }
)

## -----
#####
## obdobně pro kategorické regresory
## zkoumám vždy chí-kvadrát statistiku (a p-hodnotu) mezi vsemi možnými
## dvojicemi kategorických regresorů a z dvojic, které spolu významně
## souvisí (zamítnutí H_0 o nezávislosti) vyberu jen jeden regresor ----

#### do tabulky "factor_association" ukládám Cramerova V -----

```

```
factor_association <- matrix(  
  rep(0, length(factor_regressors_of_interest) ^ 2),  
  nrow = length(factor_regressors_of_interest)  
)  
  
for(i in 1:length(factor_regressors_of_interest)){  
  
  for(j in 1:length(factor_regressors_of_interest)){  
  
    my_table <- table(  
      train_set[  
        ,  
        factor_regressors_of_interest[i]  
      ],  
      train_set[  
        ,  
        factor_regressors_of_interest[j]  
      ]  
    )  
  
    if(is.nan(suppressWarnings(chisq.test(my_table)$statistic))){  
  
      factor_association[i, j] <- 1.0  
  
    }else{  
  
      factor_association[i, j] <- sqrt(  
        suppressWarnings(  
          chisq.test(my_table)$statistic  
        ) / sum(my_table)  
      )  
  
    }  
  
## logovací hlášky -----  
  
flush.console()  
  
print(  
  paste(  
    "Proces hotov z ",  
    format(  
      round(  
        (i - 1) / length(factor_regressors_of_interest) +  
        j / length(factor_regressors_of_interest) ^ 2  
      ) * 100,  
      digits = 2  
    ),  
  )
```

```
        nsmall = 2
    ),
    " %.",
    sep = ""
)
)

}

#### při malých velikostech tabulek může být Cramerovo V větší než 1.0,
#### což ošetřuje -----
setwd(paste(mother_working_directory, "vystupy", sep = "/"))

cairo_ps(
  file = "koreogram_factor.eps",
  width = 8,
  height = 8,
  pointsize = 14
)

par(mar = c(0.1, 0.1, 0.1, 0.1))

factor_association[factor_association > 1.0] <- 1.0

corrgram(
  x = factor_association,
  order = FALSE,
  lower.panel = panel.pie,
  labels = 1:dim(factor_association)[1]
)

dev.off()

setwd(mother_working_directory)

#### zcela spolu asocují faktory s indexy 3, 4, 5, 9, 13, 14, 16;
#### ponechávám z nich jen ten s indexem 3 -----
factor_regressors_of_interest <- factor_regressors_of_interest[
  -c(4, 5, 9, 13, 14, 16)
]

#### u kategorických proměnných je třeba vyřadit i ty faktory, které mají
```

```
#### některé hodnoty prakticky nezastoupeny; pro ně by se model neměl
#### na čem "naučit" odhadovat logerror ----

#### expertně tedy vyřazuji ----

factor_regressors_of_interest <- setdiff(
  factor_regressors_of_interest,
  c(
    "taxdelinquencyflag",
    "fireplaceflag",
    "storytypeid",
    "pooltypeid10",
    "pooltypeid2",
    "pooltypeid7",
    "decktypeid"
  )
)

## ----

#####
#####
```

6.9 hierarchical_model.R

```
#####
## vytvářím hierarchický model -----
my_lmer <- lmer(
  formula = paste(
    "logerror",
    " ~ ",
    paste(
      c(
        numeric_regressors_of_interest,
        factor_regressors_of_interest
      ),
      collapse = " + "
    ),
    .
  ).
```

```
" + (1 | parcelid)",
##" + (1 | transaction_year)",
##" + (1 | transaction_month)",
sep = ""

),
data = train_set

)

## vylepšuji model minimalizací AIC kritéria ----

new_lmer <- step(my_lmer)

## sumář modelu ----

my_summary <- summary(my_lmer)

## ----

#####
## tisknu výstup ----

my_table <- cbind(

  my_summary$coefficients|,
  ##"p-value" = (
  #    1 - unlist(
  #      lapply(
  #        abs(my_summary$coefficients[, "t value"]),
  #        pnorm
  #      )
  #    )
  #) * 2

)

print(
  xtable(
    my_table,
    align = rep("", ncol(my_table) + 1),
    digits = c(0, 3, 3, 0, 3, 3)
  ),
  floating = FALSE,
```

```
tabular.environment = "tabular",
hline.after = NULL,
include.rownames = TRUE,
include.colnames = TRUE
)

## -----
#####
## predikce -----

my_train_prediction <- predict(
  object = my_lmer,#new_lmer$model,
  newdata = train_set,
  allow.new.levels = TRUE,
  type = "response",
  re.form = NULL
)

test_set[
  test_set[, "airconditioningtypeid"] == "Evaporative Cooler",
  "airconditioningtypeid"
] <- NA

my_test_prediction <- predict(
  object = my_lmer,#new_lmer$model,
  newdata = test_set,
  allow.new.levels = TRUE,
  type = "response",
  re.form = NULL
)

## střední absolutní chyba predikce (MAE) -----

mean(abs(my_train_prediction - train_set[, "logerror"])), na.rm = TRUE)
mean(abs(my_test_prediction - test_set[, "logerror"])), na.rm = TRUE)

## -----
#####
```

```
## diagnostika modelu -----  
  
##### diagram reziduí proti vyrovnaným hodnotám -----  
  
setwd(paste(mother_working_directory, "vystupy", sep = "/"))  
  
setEPS()  
postscript(  
  file = "residua_vs_vyrovnané_hodnoty.eps",  
  width = 6,  
  height = 5,  
  pointsize = 12  
)  
  
par(mar = c(4.1, 4.25, 0.1, 0.1))  
  
plot(  
  resid(my_lmer) ~ fitted(my_lmer),  
  xlab = expression(  
    paste("vyrovnané hodnoty ", hat(bolditalic(y)), ") ", sep = ""))  
,  
  ylab = expression(  
    paste("realizace reziduí ", hat(bolditalic(e)), ") ", sep = ""))  
,  
  col = "blue"  
)  
  
dev.off()  
  
setwd(mother_working_directory)  
  
##### histogram reziduí -----  
  
setwd(paste(mother_working_directory, "vystupy", sep = "/"))  
  
setEPS()  
postscript(  
  file = "histogram_rezidui.eps",  
  width = 6,  
  height = 5,  
  pointsize = 12  
)  
  
par(mar = c(4.1, 4.1, 2.1, 0.1))  
  
hist(  
  resid(my_lmer),  
  xlab = expression(
```

```
    paste("realizace reziduí (", hat(bolditalic(e)), ")\"", sep = ""))
),
ylab = "absolutní frekvence",
main = "",
col = "lightgrey"
)

dev.off()

setwd(mother_working_directory)

##### kvantil-kvantil diagram reziduí ----

setwd(paste(mother_working_directory, "vystupy", sep = "/"))

setEPS()
postscript(
  file = "qq_plot_rezidui.eps",
  width = 6,
  height = 6,
  pointsize = 12
)

par(mar = c(4.1, 4.1, 0.1, 0.1))

qqnorm(
  resid(my_lmer),
  xlab = "teoretické kvantily",
  ylab = "výběrové kvantily",
  main = ""
)
abline(
  a = 0,
  b = 1,
  lty = 2,
  col = "blue",
  cex = 0.7
)

dev.off()

setwd(mother_working_directory)

## -----  
#####
#####
```

```
#####
```

7 Reference

- [1] KAGGLE.COM. *Zillow Prize: Zillow's Home Value Prediction (Zestimate)* [online]. 2017 [vid. 2017-12-01]. Dostupné z: <https://www.kaggle.com/c/zillow-prize-1>
- [2] WOOLDRIDGE, Jeffrey M. *Introductory Econometrics: A Modern Approach (Upper Level Economics Titles)*. B.m.: South-Western College Pub, 2012. ISBN 1111531048.
- [3] HYNDMAN, Rob J a George ATHANASOPOULOS. *Forecasting: principles and practice*. B.m.: OTexts, 2013. ISBN 0987507109.
- [4] R CORE TEAM. *R: A Language and Environment for Statistical Computing* [online]. Vienna, Austria: R Foundation for Statistical Computing, 2016. Dostupné z: <https://www.R-project.org/>
- [5] HASTIE, Trevor, Robert TIBSHIRANI a Jerome FRIEDMAN. *The Elements of Statistical Learning* [online]. B.m.: Springer New York, 2009. Dostupné z: doi:10.1007/978-0-387-84858-7
- [6] CHATTERJEE, Samprit a Ali S. HADI. *Regression Analysis by Example (Wiley Series in Probability and Statistics)*. B.m.: Wiley, 2013. ISBN 978-0-470-90584-5.
- [7] Pearson's Correlation Coefficient. In: *Encyclopedia of Public Health* [online]. B.m.: Springer Netherlands, nedatováno, s. 1090–1091. Dostupné z: doi:10.1007/978-1-4020-5614-7_2569
- [8] CRAMÉR, Harald. *Mathematical Methods of Statistics. (PMS-9)*. B.m.: Princeton University Press, 1946. ISBN 0691080046.
- [9] BOLKER, Benjamin M., Mollie E. BROOKS, Connie J. CLARK, Shane W. GEANGE, John R. POULSEN, M. Henry H. STEVENS a Jada-Simone S. WHITE. Generalized linear mixed models: a practical guide for ecology and evolution. *Trends in Ecology & Evolution* [online]. 2009, **24**(3), 127–135. Dostupné z: doi:10.1016/j.tree.2008.10.008
- [10] ROBINSON, G. K. That BLUP is a Good Thing: The Estimation of Random Effects. *Statistical Science* [online]. 1991, **6**(1), 15–32. Dostupné z: doi:10.1214/ss/1177011926
- [11] LIANG, H., H. WU a G. ZOU. A note on conditional AIC for linear mixed-effects models. *Biometrika* [online]. 2008, **95**(3), 773–778. Dostupné z: doi:10.1093/biomet/asn023
- [12] BARR, Dale J., Roger LEVY, Christoph SCHEEPERS a Harry J. TILY. Random effects structure for confirmatory hypothesis testing: Keep it maximal. *Journal of Memory and Language* [online]. 2013, **68**(3), 255–278. Dostupné z: doi:10.1016/j.jml.2012.11.001
- [13] CHOLLET, François a OTHERS. *Keras*. B.m.: <https://github.com/fchollet/keras>; GitHub. 2015
- [14] ABADI, Martín, Ashish AGARWAL, Paul BARHAM, Eugene BREVDO, Zhifeng CHEN, Craig CITRO, Greg S. CORRADO, Andy DAVIS, Jeffrey DEAN, Matthieu DEVIN, Sanjay GHEMAWAT, Ian GOODFELLOW, Andrew HARP, Geoffrey IRVING, Michael ISARD, Yangqing JIA, Rafal JOZEFOWICZ, Lukasz KAISER, Manjunath KUDLUR, Josh LEVENBERG, Dan MANÉ, Rajat MONGA, Sherry MOORE, Derek MURRAY, Chris OLAH, Mike SCHUSTER, Jonathon SHLENS, Benoit STEINER, Ilya SUTSKEVER, Kunal TALWAR, Paul TUCKER, Vincent VANHOUCKE,

Vijay VASUDEVAN, Fernanda VIÉGAS, Oriol VINYALS, Pete WARDEN, Martin WATTENBERG, Martin WICKE, Yuan YU a Xiaoqiang ZHENG. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems* [online]. 2015. Dostupné z: <https://www.tensorflow.org/>. Software available from tensorflow.org

- [15] JIA, Yangqing, Evan SHELHAMER, Jeff DONAHUE, Sergey KARAYEV, Jonathan LONG, Ross GIRSHICK, Sergio GUADARRAMA a Trevor DARRELL. Caffe: Convolutional Architecture for Fast Feature Embedding. *arXiv preprint arXiv:1408.5093*. 2014.