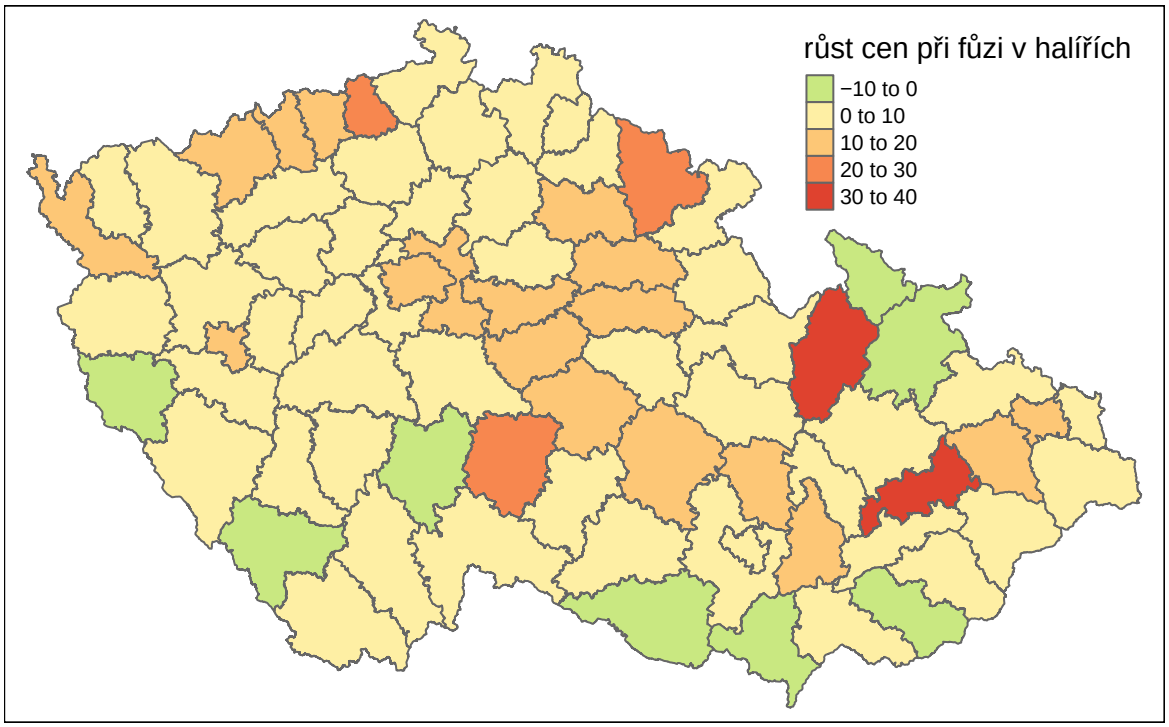
Dokumentace systému MergerSim pro podporu rozhodování při posuzování fúzí na trzích s homogenním produktem a prostorovou diferenciací



Software byl vytvořen týmem Masarykovy univerzity s podporou grantu TAČR TL02000122 *Systém pro podporu rozhodování při posuzování fúzí na trzích s homogenním produktem a prostorovou diferenciací*.

Obsah

1 Úvod

Tento text popisuje instalaci a použití softwaru **MergerSim** pro podporu rozhodování při posuzování fúzí na trzích s homogenním produktem a prostorovou diferenciací. Tvorba tohoto softwaru byla podpořena grantem TAČR TL02000122 *Systém pro podporu rozhodování při posuzování fúzí na trzích s homogenním produktem a prostorovou diferenciací*. Předpokládaným uživatelem softwaru je zejména Úřad pro ochranu hospodářské soutěže (ÚOHS), který na vzniku tohoto softwaru spolupracoval.

Software **MergerSim** je implementován ve statistickém výpočetním systému R. Skládá se z jednotlivých skriptů, které implementují obecně použitelné funkce, zavádějí potřebné korekce dat a ukazují použití systému. Toto otevřené řešení bylo zvoleno z několika důvodů: 1) použitý statistický software je zdarma k dispozici pro všechny hlavní operační systémy, 2) implementuje metody metody pro statistickou a ekonometrickou analýzu prostorových dat, a 3) jednotlivé implementované funkce jsou uživateli přímo k dispozici, takže může prostudovat jejich kód a případně si je upravit podle svých potřeb.

Všechny funkce, u kterých se předpokládá využití koncovým uživatelem, jsou detailně dokumentované v jednotlivých skriptech. Před definicí funkce je vždy uvedeno, co funkce dělá, jaké očekává vstupy a jaký vrací výstup. Dále jsou zde popsány podstatné detaily fungování dané funkce. Dokumentace také ukazuje praktické využití jednotlivých funkcí. Tento text se proto nezaměřuje na jednotlivé funkce, nýbrž na celkový pohled a přístup k použití tohoto software. Ukazuje

- jak software nainstalovat,
- jak připravit data pro ekonometrickou analýzu,
- jak odhadnout ekonometrický model konkurence na daném trhu,
- jak simulovat dopady fúze velkých značek na tomto trhu a
- jak vizualizovat původní data i výsledky simulací.

Práce se softwarem **MergerSim** je v tomto dokumentu ilustrovaná na příkladu analýzy vlivu několika velkých fúzí na trhu benzínem Natural 95 (E5) v České republice, kdy firma Mol koupila v roce 2014 všechny čerpací stanice Lukoil a v roce 2015 všechny čerpací stanice Agip a kdy Benzina koupila v roce 2016 vybrané čerpací stanice OMV včetně některých čerpacích stanic fungujících pod značkou Avanti. Zdrojem dat o cenách pohonných hmot, které jsou zde použity, je Pumpdroid.[1]

V tomto dokumentu budeme jednotlivé provozovny, která prodávají vybraný produkt na určité adrese (v příkladech čerpací stanice), nazývat “stanice”. Předpokládáme, že každá stanice patří jedné značce. Jedna firma může vlastnit více značek, které si pak nekonkurují, protože firma řídí cenotvorbu na jednotlivých stanicích centrálně.

Poznámka: Protože je software **MergerSim** implementován ve výpočetním systému R, předpokládá tento dokument od uživatele pracovní znalost jazyka R včetně balíků ze skupiny **tidyverse**. Zároveň předpokládá odpovídající znalosti statistiky a ekonometrie.

[1] Pumpdroid je aplikace, která sbírá data o cenách pohonných hmot od svých uživatelů. K dispozici je jak pro Android, tak (pod jménem iPumpuj) pro iPhone. Detaily viz http://www.pumpdroid.com/. Za poskytnutí těchto dat děkujeme panu Marcelu Matulovi z Pumpdroidu.

2 Systémové nároky a instalace softwaru MergerSim

Tato kapitola popisuje, jaké hardwarové a softwarové vybavení je pro analýzu potřeba, jak software **MergerSim** nainstalovat a jakou adresářovou strukturu software předpokládá.

2.1 Potřebná konfigurace hardwaru a softwaru

Výpočty jsou poměrně hardwarově náročné. Proto se předpokládá následující konfigurace hardwaru a softwaru:

- procesor třídy i5 nebo vyšší (větší počet jader je výhodou),
- minimálně 32 GB operační paměti,
- operační systém Linux nebo Windows 10 (software pravděpodobně funguje i na nových verzích macOS, nebyl však na tomto systému testován),
- R verze 4.0.3 nebo novější s aktualizovanými balíky a
- Sun Java verze 14.0.2 nebo novější.

2.2 Instalace softwaru MergerSim

Software **MergerSim** je předáván v komprimovaném souboru typu `.zip`. Jeho instalace spočítá v rozbalení tohoto souboru do vhodného adresáře `MERGERSIM`. Absolutní cesta z kořene systému k tomuto adresáři může obsahovat pouze písmena anglické abecedy, číslice a podtržítka, tj. nesmí obsahovat mezery, písmena s diakritikou apod.

Rozbalením komprimovaného souboru se v tomto adresáři vytvoří následující podadresáře:

- `MERGERSIM/R` ... adresář se skripty, které definují uživatelské funkce, ukazují použití softwaru a tvorbu korekcí dat,
- `MERGERSIM/rawdata` ... adresář pro umístění původních dat o cenách, lokacích atd.,
- `MERGERSIM/data` ... adresář pro umístění dat vytvořených tímto softwarem,
- `MERGERSIM/map_shapes` ... adresář pro umístění mapových podkladů,
- `MERGERSIM/roadrunner` ... adresář pro umístění programu `RoadRunner.jar` a
- `MERGERSIM/doc` ... adresář pro umístění této dokumentace.

Jednotlivé funkční skripty načtete do R standardně pomocí funkce `source()`. Jejich funkčnost vyžaduje, aby byla před jejím použitím vytvořena proměnná `CODEDIR` typu *character*, která obsahuje cestu z pracovního adresáře do adresáře `MERGERSIM/R`.

2.3 Získání a umístění potřebných mapových podkladů

Plná funkcionalita softwaru vyžaduje přítomnost mapových podkladů Arc ČR a OpenStreetMap.

Arc ČR: Ze stránky http://download.arcdata.cz/data/ stáhněte instalační soubor pro Windows a nainstalujte jej. Vzniklé podadresáře `AdministrativniCleneni_v13.gdb` a `ArcCR500_v33.gdb` přesuňte do adresáře `MERGERSIM/map_shapes`.

- `AdministrativniCleneni_v13.gdb` představuje nezbytný mapový poklad. Používá se pro ořez dat a výpočtů do obrysu České republiky, vizualizaci dat a připojení údajů o umístnění stanice do municipality, okresu, kraje a ČR. Celý tento adresář uložte do adresáře `MERGERSIM/map_shapes`.
- `ArcCR500_v33.gdb` je volitelný mapový poklad. Používá se pro připojení údajů o typech silnic, na kterých stanice leží. Celý tento adresář uložte do adresáře `MERGERSIM/map_shapes`.

OpenStreetMap: Ze stránky https://download.geofabrik.de/europe/czech-republic.html stáhněte soubory `czech-republic-latest.osm.pbf` a `czech-republic-latest-free.shp.zip`.

- `czech-republic-latest.osm.pbf` představuje nezbytný mapový poklad, který se používá pro zjišťování dojezdových vzdáleností mezi stanicemi. Uložte jej přímo do adresáře `MERGERSIM/map_shapes`.
- `czech-republic-latest-free.shp.zip` je volitelný mapový poklad, který se používá pro připojení údajů o typech silnic, na kterých stanice leží. Soubor rozzipujte a výsledek uložte do adresáře `MERGERSIM/map_shapes/OpenStreetMaps`.

2.4 Přehled funkčních, korekčních a demonstračních skriptů

Podadresář `R` obsahuje tři typy skriptů. Zaprvé, obsahuje obecné skripty pro přípravu a analýzu dat:

- `preparatory_functions_general.R` implementuje funkce pro obecnou přípravu dat pro ekonometrické modelování,
- `econometric_functions.R` implementuje funkce pro ekonometrické modelování konkurence a simulaci fúzí jak na průřezových, tak panelových datech,
- `graphing_functions.R` implementuje funkce pro grafickou vizualizaci původních dat i výsledků simulací fúze, a to jak v podobě statických grafů ve formátech PNG a PDF, tak interaktivních grafů určených pro práci ve webovém prohlížeči a
- `utilities.R` definuje drobné užitkové funkce.

Zadruhé, podadresář `R` obsahuje demonstrační skripty, které ukazují použití softwaru:

- `prepare_pumpdroid_data.R` ukazuje kompletní přípravu dat pro ekonometrickou analýzu. Používají se data o cenách pohonných hmot ze zdroje Pumpdroid. Skript proto ukazuje nejdříve převod dat ze struktury poskytnuté Pumpdroidem do standardního tvaru, a pak tvorbu agregovaných dat a přidání různých statistik konkurence apod.
- `example_crosssectional_estimation_and_merger_simulations.R` ukazuje odhad modelu konkurence na trhu pohonných hmot na průřezových datech, simulaci fúze a vizualizaci výsledků této simulace.
- `example_panel_estimation_and_merger_simulations.R` ukazuje odhad modelu konkurence na trhu pohonných hmot na panelových datech se zahrnutím fixních efektů na čas a jednotlivé stanice, ex-post simulaci fúze a vizualizaci výsledků této simulace.
- `example_panel_estimation_and_merger_simulations_time_fixed_effects_only.R` ukazuje odhad modelu konkurence na trhu pohonných hmot na panelových datech se zahrnutím fixních efektů na čas, jednotlivé značky, velikost municipality a přítomnost dálnice, simulaci hypotetické fúze a simulaci důsledků uvalení závazku prodeje několika stanic.
- `example_plotting.R` ukazuje vizualizaci původních cenových dat.

Zatřetí, podadresář `R` obsahuje i příklady tvorby korekčních tabulek pro vyčištění dat ze zdroje Pumpdroid:

- `handmade_corrections_pumpdroid.R` ukazuje, jak vytvořit tabulky pro korekci chyb, které v datech Pumpdroidu vznikly na straně jeho uživatelů a
- `handmade_corrections_general.R` ukazuje, jak vytvořit tabulku společného vlastnictví čerpacích stanic.

3 Očekávaná struktura vstupních dat

Tato kapitole popisuje, v jaké struktuře musí být nachystána vstupní data než se začnou připravovat pro ekonometrickou analýzu. (Jak připravit data pro ekonometrickou analýzu popisuje kapitola 5. Kapitola 4 ukazuje, jak do standardní vstupní struktury převést data poskytnutá Pumpdroidem.) V systému R musejí mít vstupní data podobu několika tabulek třídy *tibble*. Tři tabulky jsou nezbytné: tabulka cen, tabulka lokací a tabulka vlastnictví. Další tabulky (tabulky vlastností a společného vlastnictví) jsou volitelné.

3.1 Tabulka cen

Tabulka cen (třídy *tibble*) musí obsahovat právě následující sloupce:

- `date` (třída *Date*) ... datum, kdy byla cena pozorována,
- `id` (třída *integer*) ... identifikace stanice, kde byla cena pozorována,
- `fuel` (třída *character*) ... typ produktu, např. paliva, a
- `price` (třída *double*) ... cena daného produktu daný den na dané stanici v Kč.

Příklad:

date	id	fuel	price
2011-12-28	432	LPG	17.5
2011-12-30	169	LPG	18.8
2011-12-30	379	LPG	18.4
2011-12-30	695	LPG	18.4
2011-12-30	1524	LPG	17.8

Pro každý den, stanici a typ produktu se předpokládá pouze jedna cena. Pokud bylo v daný den pozorován větší počet cen, je potřeba, aby byla tato pozorování nějak agregovaná. Naopak není potřeba, aby byla pozorování doplněna pro všechny dny v daném rozsahu. Chybějící hodnoty budou interpolovány.

3.2 Tabulka lokací stanic

Tabulka lokací stanic (třídy *tibble*) musí obsahovat právě následující sloupce:

- `id` (třída *integer*) ... identifikace stanice pro spárování s tabulkou cen,
- `lat` (třída *double*) ... latitude stanice v decimálním formátu,
- `lon` (třída *double*) ... longitude stanice v decimálním formátu,
- `gps` (třída *character*) ... (volitelné) latitude a longitude stanice v decimálním formátu oddělené čárkou a
- `address` (třída *character*, sloupec je nepovinný) ... adresa stanice nahlášená poskytovatelem dat; systém ji nepoužívá – má využití jen pro lidskou kontrolu dat.

Příklad:

id	lat	lon	gps	address
1	48.75656	16.87633	48.756561,16.876332	tř. 1. máje, Břeclav
2	50.03534	15.20796	50.035341,15.207961	II/328 Hřbitovní, Kolín
3	50.03234	15.17825	50.032341,15.178248	I/38 Pražská, Kolín
4	50.06464	14.49412	50.064644,14.494115	V korytech, Praha-Strašnice
5	50.07665	14.57955	50.07665,14.579555	Štěrboholská, Praha (směr Černý most)

Předpokládá se, že adresy se nemění, tj. po celou dobu se používá jedna lokace; pokud je nahlášeno několik různých lokací nebo adres, je potřeba vybrat vždy právě jednu lokaci pro každou stanici.

3.3 Tabulka vlastnictví stanic

Tabulka vlastnictví stanic (třídy *tibble*) má následující strukturu:

- `date` (třída *Date*) ... datum, kdy bylo poprvé pozorováno jméno stanice, tj. datum prvního pozorování nebo datum změny,
- `id` (třída *integer*) ... identifikace stanice pro spárování s tabulkou cen a
- `name` (třída *character*) ... jméno značky stanice.

Příklad:

date	id	name
2012-05-12	1	agip
2015-08-01	1	mol
2012-05-12	2	agip
2015-08-01	2	mol
2012-05-12	3	agip

Není potřeba, aby tabulka obsahovala jeden řádek pro každou stanici a každý den – stačí jen data začátku pozorování a změn.

Jména je potřeba standardizovat (např. MOL, Mol a mol vždy převést na stejné jméno, např. mol). Značky, které patří jednomu majiteli (svého času např. PapOil a Mol) budou mít svá skutečná jména (např. papoil a mol). To je potřeba proto, že mohou mít různě nastavené cenové úrovně. To, že si nekonkurují, bude vyřešeno separátní tabulkou, viz oddíl 3.5.

3.4 Tabulky vlastností stanic

Je možné přidat i nepovinné tabulky vlastností stanic (u čerpacích stanic např. certifikát SGS, otevírací hodiny, obchod, ...). Každá taková tabulka (třídy *tibble*) musí obsahovat právě následující sloupce:

- `date` (třídy *Date*) ... datum, kdy byla poprvé pozorována daná vlastnost na dané stanici, tj. datum prvního pozorování nebo datum změny,
- `id` (třída *integer*) ... identifikace čerpací stanice pro spárování s tabulkou cen a
- jeden sloupec vlastností, např. certifikace SGS, otevřeno non-stop apod.

Příklad:

date	id	sgs
2012-05-12	1	FALSE
2012-05-12	2	FALSE
2012-05-12	3	FALSE
2012-05-12	4	FALSE
2012-05-12	5	FALSE

Není potřeba, aby tabulka obsahovala jeden řádek pro každou čerpací stanici a každý den – stačí jen data začátku pozorování a změn.

Tabulka není povinná. Pokud bude poskytnuta, bude její obsah připojen k datasetu a může být využit při analýze (ekonometrii a simulaci fúze). Jméno třetího sloupce (zde sgs) se použije k pojmenování dané vlastnosti v datech připravených pro ekonometrickou analýzu.

3.5 Tabulka společného vlastnictví

Někdy je několik značek vlastněno stejnou společností. To znamená, že si tyto stanice 1) nekonkurují, ale 2) v ekonometrickém modelu musejí mít různé úrovňové konstanty, protože různé značky mohou mít různé cenové úrovně. Příkladem jsou např. značky PapOil a Mol, které obě patřily Molu, ale Mol byl premium brand, zatímco PapOil byl lowcost brand.

Společné vlastnictví je indikováno v tabulce třídy *tibble*, která má čtyři sloupce:

- `from_date` (třída *Date*) ... od kdy patří daná firma do skupiny,
- `to_date` (třída *Date*) ... do kdy patří daná firma do skupiny,
- `group_name` (třída *character*) ... jméno skupiny a
- `name` (třída *character*) ... jméno značky.

Pokud je jméno jedné značky stejné jako jméno celé skupiny, není potřeba pro tuto značku vytvářet v tabulce extra řádek. Tak např. pro spojení PapOil a Mol stačí v korekční tabulce jeden řádek.

Příklad:

from_date	to_date	group_name	name
2012-01-01	2100-12-31	mol	papoil
2012-01-01	2100-12-31	mol	slovnaft
2012-01-01	2100-12-31	benzina	orlen
2012-01-01	2100-12-31	benzina	star
2003-04-01	2100-12-31	omv	avanti

Příklad takové tabulky je implementován ve skriptu `handmade_corrections_general.R`.

4 Převod dat poskytnutých Pumpdroidem do standardního vstupního formátu

Tato kapitola popisuje, jak data poskytnutá Pumpdroidem převést do standardního formátu popsaného v kapitole 3. Do této dokumentace je tento popis zařazen z několika důvodů: 1) hlavní cílový odběratel softwaru (ÚOHS) může tato data využít, 2) software proto implementuje mnoho funkcí pro obecnou přípravu dat poskytnutých Pumpdroidem a 3) tento postup může být inspirací pro přípravu dat z jiných zdrojů do potřebného formátu. Pokud však uživatel neplánuje použít data o cenách pohonných hmot od Pumpdroidu ani nepotřebuje ilustraci přípravy dat, může tuto kapitolu přeskočit.

4.1 Popis zdrojových dat poskytnutých Pumpdroidem

Pumpdroid poskytuje dva typy souborů (zde jsou komprimované pomocí algoritmu gzip):

- `pumpdroid-XXXX-export.csv.gz` jsou soubory s cenami (`XXXX` je daný rok) a
- `export-pumps.YYYY-MM-DD.csv.gz` jsou soubory s informacemi o čerpacích stanicích; `YYYY` označuje rok, `MM` měsíc a `DD` den, ke kterému jsou informace o čerpacích stanicích platné.

Každý typ souboru je třeba uložit do vlastního adresáře, zde v odpovídajícím pořadí do adresářů `pumpdroid_price_data` a `pumpdroid_station_data`.

Údaje o cenách jsou uložené v textových souborech ve zvláštním formátu podobném CSV s položkami oddělenými středníky. Data tvoří vždy čtveřice údajů:

- id čerpací stanice (celé číslo),
- datum pozorování ve formátu `YYYY-MM-DD`,
- cena v CZK ve formátu `KK.H` a
- typ paliva (např. NATURAL95).

Na jednom řádku může být větší počet těchto čtveřic. Pro některé kombinace id, datumu a typu paliva někdy data obsahují několik různých cen.

Soubory s informacemi o jednotlivých čerpacích stanicích jsou skutečné CSV oddělené středníky. Data tvoří tyto sloupce:

- id čerpací stanice (celé číslo),
- jméno (značka) čerpací stanice,
- člověkem zadaná adresa,
- SGS certifikát (0 . . . nemá, 1 . . . má ho),
- gps souřadnice v desetinném formátu, oddělená čárkou (např. 48.756561,16.876332) a
- otevírací doba (“NS” znamená non-stop, ale mohou tam být i zadané hodiny pro jednotlivé dny).

Informace o jednotlivých čerpacích stanicích jsou do jisté míry problematické.

1. Datum, ke kterému pozorování patří, není součástí tabulky, ale jména souboru. Toto datum odpovídá dni, kdy byla ohlášena nějaká změna. Podle počtu souborů je export proveden v průměru zhruba jednou za 14 dnů, ale rozestupy jsou velmi nepravidelné: někdy 4 dny, jindy 2 měsíce apod.

2. Názvy čerpacích stanic jsou zadané nekonzistentně a musejí se dočistit. Navíc mohou být zadané se zpožděním oproti skutečné změně.
3. GPS lokace nejsou úplně přesné a v čase se někdy mění (snad se zlepšují).
4. Uživatelsky zadané adresy jsou chaotické a použitelné maximálně pro hrubou kontrolu, jestli se data načetla správně.

S daty poskytnutými Pumpdroidem jsou i další dva problémy: 1) Některé čerpací stanice ve skutečnosti neexistují – např. proto, že jedna čerpací stanice má víc id, někdy naráz, někdy postupně (stanice možná na čas zanikla). 2) Data obsahují (zvlášť v pozdějších letech) i čerpací stanice mimo ČR.

4.2 Převod dat poskytnutých Pumpdroidem do standardního formátu

Skript `preparatory_functions_pumpdroid.R` implementuje funkce potřebné pro převod dat z formátu poskytnutého Pumpdroidem do standardního formátu, který předpokládá tento software. Příklad použití ukazuje první část skriptu `prepare_pumpdroid_data.R`. Před použitím tohoto skriptu je vhodné vytvořit korekční tabulky, které umožní data vyčistit a opravit problémy popsané v předchozím oddíle.

Z dat poskytnutých Pumpdroidem vytvoříme 5 tabulek, které odpovídají popisu v kapitole 3:

- tabulku cen (`price_data`),
- tabulku lokací čerpacích stanic (`station_location`),
- tabulku jmen (značek) čerpacích stanic (`station_names`) a
- tabulky vlastností čerpacích stanic: zda má čerpací stanice certifikát SGS (`station_sgs`) a zda má čerpací stanice otevřeno nonstop (`station_nonostop`).

4.2.1 Tabulka cen

Postup tvorby tabulky cen je následující: Nejdříve načteme všechny soubory s cenami naráz pomocí funkce `read_pumpdroid_price_data()`. Přitom se pro každou kombinaci čerpací stanice, dne a paliva zachová pouze jedna cena.

Potom z takto vzniklé tabulky odstraníme neexistující čerpací stanice a spojíme stanice, které jsou zadané se dvěma id. To provede funkce `clean_pumpdroid_price_data()`. K tomuto kroku je třeba nejdříve nachystat korekční tabulky, které popisují, které čerpací stanice vyhodit a které spojit.

```
price_data <- PRICE_DATA_FOLDER %>%
    read_pumpdroid_price_data() %>%
    clean_pumpdroid_price_data(unite = pumpdroid_stations_to_be_united,
                               remove = pumpdroid_stations_to_be_removed)
```

Příklady těchto korekčních tabulek najdete ve skriptu `handmade_corrections_pumpdroid.R`. Korekční tabulka `pumpdroid_stations_to_be_removed` je ve skutečnosti atomický vektor třídy *integer*, který obsahuje id stanic (v příkladu čerpacích stanic), které se mají z dat vyřadit, protože ve skutečnosti neexistují.

Naproti tomu tabulka `pumpdroid_stations_to_be_united` je skutečná tabulka třídy *tibble*, která obsahuje šest sloupců:

- `target_id` (třída *integer*) ... id čerpací stanice, pod kterou se sjednotí stanice s id `source_id`; její hodnoty jsou implicitní, tj. použijí se, pokud je nenahradí připojovaná stanice,
- `source_id` (třída *integer*) ... id čerpací stanice, která má být sjednocena s `target_id` – její hodnoty v daném časovém okně nahradí nebo doplní doplní hodnoty `target_id` a tato stanice se pak vyřadí z dat,
- `feature` (třída *character*) ... jaká vlastnost má být nahrazena; může mít hodnoty `"price"`, `"name"`, `"location"`, `"sgs"`, `"nonstop"`, jejich kombinace (např. `"location, sgs, nonstop"`) nebo `"all"`, což indikuje všechny vlastnosti v datech,
- `from` a `to` (třída *character*) ... datum od kdy do kdy ve formátu "YYYY-MM-DD"; `from` může mít i hodnotu `"start"` a `to` hodnotu `"end"`, které indikují začátek a konec dat, a
- `keep` (třída *logical*) ... pokud je `TRUE`, pak se v intervalu nechají i hodnoty z `target_id` (tj. `source_id` je doplní) a jen se následně odstraní duplicity; pokud `FALSE` (což je typičtější hodnota), pak se hodnoty z `target_id` v daném časovém okně odstraní a nahradí hodnotami ze `source_id`.

4.2.2 Tabulka lokací

Postup pro tvorbu tabulky lokací čerpacích stanic je obdobný: Nejprve se načtou všechny informace o čerpacích stanicích pomocí funkce `read_pumpdroid_station_data()`. (Tento krok není nezbytný, ale výrazně zrychlí následující kroky.) Z takto vzniklé tabulky vyfiltruje funkce `read_pumpdroid_station_locations()` data o poloze čerpacích stanic.

Následně je opět třeba z takto vzniklé tabulky odstranit neexistující čerpací stanice a spojit stanice, které jsou zadané se dvěma id. Oba úkoly zařídí funkce `clean_pumpdroid_station_locations()`. K tomu se použijí stejné korekční tabulky jako pro čištění tabulky cen, viz oddíl 4.2.1.

V takto získané tabulce mohou být duplicitní lokace (vzniklé např. špatným zaměřením GPS, kterou později někdo opravil). Pomocí funkce `pumpdroid_select_station_locations()` se nakonec vybere pro každou čerpací stanice jedna poloha, např. ta, která je v datech zadána naposledy (a je tedy snad nejpřesnější).

```
station_data <- read_pumpdroid_station_data(STATION_DATA_FOLDER)
station_locations <- station_data %>%
    read_pumpdroid_station_locations() %>%
    clean_pumpdroid_station_locations(
        unite = pumpdroid_stations_to_be_united,
        remove = pumpdroid_stations_to_be_removed
    ) %>%
    pumpdroid_select_station_locations("last")
```

4.2.3 Tabulka značek (jmen)

Postup pro tvorbu tabulky jmen čerpacích stanic je podobný, ale musí provést víc operací, protože názvy značek jsou často uživateli Pumpdroidu zadávány nekonzistentně.

Nejdříve vezmeme data vytvořená funkcí `read_pumpdroid_station_data()` a z této tabulky vyfiltrujeme pomocí funkce `read_pumpdroid_station_names()` informace o názvech jednotlivých čerpacích stanic. Funkce zároveň tato jména strojově vyčistí:

- jména se převedou na malá písmena,
- odstraní se divné znaky (apostrofy, uvozovky, čárky, ...),
- odstraní se různé varianty `s.r.o.`, `a.s.`, `.cz`,
- odstraní se varianty `čs` a `čerpací stanice phm` a
- odstraní se divné názvy silnic (jako `i/7`).

Zároveň se pomocí korekční tabulky `pumpdroid_station_names_to_be_unified` sjednotí nekonzistentně zadané názvy stanic (např. benzina a bensina). Příklad takové tabulky je implementován ve skriptu `handmade_corrections_pumpdroid.R`. Tabulka je ve skutečnosti seznam atomických řetězců třídy *character*. Každé jméno stanice obsažené v kterémkoli vektoru korekčního seznamu se nahradí prvním prvkem svého vektoru. Pokud tedy seznam začíná např. takto

```
pumpdroid_station_names_to_be_unified <- list(
    c("ab oil", "ab - oil"),
    c("agip", "agip gas", "eni oil", "eni", "agip (eni)", "?eni"),
    ...
```

pak se “ab - oil” nahradí jménem “ab oil”, “agip gas” a “eni” jménem “agip” atd. Nezadaná jména a jména čerpacích stanic, která ve skutečnosti nejsou jména (jako `čerpací stanice`) se nahradí příznakem chybějící hodnoty (`NA`).

Z takto vzniklé tabulky odstraníme neexistující čerpací stanice a spojíme stanice, které jsou zadané se dvěma id. Oboje provede funkce `clean_pumpdroid_station_names()`. K tomu se použijí stejné korekční tabulky jako pro čištění tabulky cen, viz oddíl 4.2.1. Funkce zároveň odstraní redundantní data tak, že se zachová jen první den, kdy bylo hlášeno pro danou čerpací stanici nějaké jméno.

Pokud víme, že jsou některá jména čerpacích stanic zadaná špatně, můžeme je “na tvrdo” opravit pomocí funkce `correct_pumpdroid_station_names()`. Funkce potřebuje zadat korekční tabulku `pumpdroid_station_names_correction`, která říká, na jaké hodnoty se má v určitém rozsahu dnů opravit jméno stanic se zadaným id. Příklad takové tabulky je implementován ve skriptu `handmade_corrections_pumpdroid.R`. Tabulka je třídy *tibble* a má následující sloupce:

- `id` (třída *integer*) ... id stanice, jejíž jméno se má opravit,
- `from` a `to` (třída *character*) ... datum počátku a konce období, kdy se mají data opravit, ve formátu “YYYY-MM-DD”; `from` může mít i hodnotu `"start"` a `to` hodnotu `"end"` a
- `name` (třída *character*) ... správná značka stanice, která přepíše značku přítomnou v datech.

Uživatelé Pumpdroidu typicky změní značku až tehdy, když je čerpací stanice přemalována do nových barev. Po velkých fúzích však může trval rok nebo i víc, než jsou všechny převzaté stanice přemalovány. Pokud chceme vlastnictví opravit přesněji a pokud máme seznam všech stanic, které byly převzaty, můžeme to udělat v předchozím kroku. Pokud tento seznam nemáme, můžeme převzaté stanice změnit odhadem. K tomu potřebujeme korekční tabulku, která obsahuje čtyři informace: datum převzetí, datum, kdy jsou už všechny převzaté stanice správně zadané v datech Pumpdroid, název přebírané značky a název značky, která převzala přebíranou značku (příklad takové tabulky najdete pod jménem `takeovers_brute_force_correction_table` ve skriptu `handmade_corrections_pumpdroid.R`). Zbytek zařídí funkce `correct_brandnames_by_brute_force()`. Funkce najde čerpací stanice, které splňují naráz tři podmínky: 1) v době fúze patřily přebírané značce, 2) v době, kdy už jsou data zadaná správně, patřily přebírající značce a 3) mezi tím patřily jen těmto dvěma značkám. Pak těmto stanicím vnutí jméno přebírající značky, a to ode dne fúze. (Tato funkce je odvážná, a její výsledky je tedy vhodné ručně zkontrolovat.)

```
station_names <- station_data %>%
    read_pumpdroid_station_names(names_correction =
                                     pumpdroid_station_names_to_be_unified) %>%
    clean_pumpdroid_station_names(unite = pumpdroid_stations_to_be_united,
                                  remove = pumpdroid_stations_to_be_removed) %>%
    correct_pumpdroid_station_names(pumpdroid_station_names_correction) %>%
    correct_brandnames_by_brute_force(pumpdroid_takeovers_correction)
```

Strojová normalizace jmen čerpacích stanic využívá určitých konstant zadaných ve funkcích `normalize_pumpdroid_names()` a `remove_pumpdroid_generic_names()`, které jsou implementované ve skriptu `handmade_corrections_pumpdroid.R`. Při každé aktualizaci dat je tudíž třeba ručně projít jména čerpacích stanic a následně je aktualizovat a doplnit tyto funkce, stejně jako korekční tabulku `pumpdroid_station_names_to_be_unified`.

4.2.4 Tabulky vlastností

Tabulky vlastností (v případě Pumpdroidu to, zda má čerpací stanice certifikát SGS a zda má otevřeno nonstop) opět vytvoříme podobně: Vezmeme data vytvořená funkcí `read_pumpdroid_station_data()` a z této tabulky vyfiltrujeme informace o daných vlastnostech pomocí funkcí `read_pumpdroid_station_sgs()` a `read_pumpdroid_station_nonstop()`.

Následně z takto vzniklých tabulek odstraníme neexistující čerpací stanice a spojíme stanice, které jsou zadané se dvěma id. Oboje provede funkce `clean_pumpdroid_station_features()`. Používají se stejné korekční tabulky jako pro čištění tabulky cen, viz oddíl 4.2.1.

```
station_sgs <- read_pumpdroid_station_sgs(station_data) %>%
    clean_pumpdroid_station_features(unite = pumpdroid_stations_to_be_united,
                                     remove = pumpdroid_stations_to_be_removed)
station_nonostop <- read_pumpdroid_station_nonstop(station_data) %>%
    clean_pumpdroid_station_features(unite = pumpdroid_stations_to_be_united,
                                     remove = pumpdroid_stations_to_be_removed)
```

5 Obecná příprava dat pro ekonometrickou analýzu

Tato kapitola popisuje, jak připravit data ze standardního vstupního formátu popsaného v kapitole 3 pro ekonometrickou analýzu. To zahrnuje zejména přidání geografických informací o poloze stanic v municipalitách, okresech, na daném typu silnice apod., odstranění stanic, které nejsou v ČR nebo neobsahují cenová data, tvorbu agregátních dat za vybrané časové úseky, přidání statistik konkurence a případně i přidání fixních efektů na velikost municipality či značku stanice nebo její změnu v důsledku fúze.

Funkce potřebné k obecné přípravě dat pro ekonometrické modelování jsou implementovány ve skriptu `preparatory_functions_general.R`. Příklad přípravy dat pro ekonometrickou analýzu ze standardizovaných vstupů ukazuje druhá část skriptu `prepare_pumpdroid_data.R`.

Některé výpočty popsané v této kapitole mohou běžet na více jádrech. To je možné zapnout výrazem

```
plan(multisession)
```

Dodatečný parametr `workers` umožňuje omezit počet použitých jader (implicitně se používají všechna dostupná jádra).

5.1 Úpravy tabulky lokací

Do tabulky lokací stanic přidáme katastrální informace a informace o silnici, na které daná stanice leží.

5.1.1 Přidání katastrálních informací

Do tabulky lokací je potřeba přidat informace o obci, okresu, kraji a zemi (`country`), ve kterých se stanice nachází. To udělá funkce `add_arccr_municipalities()`, která ke svému běhu potřebuje lokační tabulku a adresář, kde je umístěna databáze administrativního členění Arc ČR 500 "AdministrativniCleneni_v13.gdb".

```
station_locations <- station_locations %>%
    add_arccr_municipalities(ARCCR_ADMIN_FOLDER)
```

Funkce `add_arccr_municipalities()` přidá do tabulky lokací následující sloupce:

- `obec` je standardizovaný název obce, např. "Brno",
- `okres` je standardizovaný název okresu, např. "Brno-město",
- `kraj` je standardizovaný název kraje, např. "Hlavní město Praha" a
- `country` nabývá hodnot "Česká republika" nebo "cizina".

Sloupec `country` umožní odfiltrovat stanice, které nejsou v ČR. Sloupec `obec` je užitečný např. ke kontrole o velikost obce, viz oddíl 5.6.1. Pozor: Stanice je umístěna do *katastru obce*, tj. do dané obce jsou umístěny všechny stanice v jejím katastru, i když jsou na silnici za značkou konce obce. Pro vymezení obcí podle značek začátku a konce obce není k dispozici žádný mapový poklad.

5.1.2 Přidání informací o silnicích

Dále je užitečné přidat do tabulky lokací informace o silnici, na které daná stanice leží. K tomu sloučí funkce `add_arccr_roads()` a `add_osm_roads()`, které čerpají informace z mapových podkladů Arc ČR a OpenStreetMap. Obě funkce mají dva povinné parametry: tabulku lokací a cestu k adresářům `ArcCR500_v33.gdb`, resp. `OpenStreetMaps`. Funkce mají i další volitelné parametry, viz jejich dokumentace.

```
station_locations <- station_locations %>%
    add_arccr_roads(ARCCR_GEO_FOLDER) %>%
    add_osm_roads(OSM_FOLDER)
```

Funkce `add_arccr_roads()` přidá do tabulky lokací následující sloupce:

- `arc_road_class` ... číslo třídy silnice v podkladu Arc ČR,
- `arc_road_int_class` ... mezinárodní kódy silnic v podkladu Arc ČR,
- `arc_road_number` ... český kód silnice v podkladu Arc ČR,
- `arc_road_lanes` ... počet pruhů silnice v podkladu Arc ČR,
- `arc_road_distance` ... vzdálenost stanice od silnice v metrech a
- `arc_highway` ... logická proměnná, která má `TRUE` pro stanice, které podle Arc ČR leží na dálnici.

Funkce `add_osm_roads()` přidá do tabulky lokací následující sloupce:

- `osm_road_class` ... typ dané silnice v kódování OSM map,
- `osm_oneway` ... zda je silnice jednosměrná v kódování OSM map,
- `osm_maxspeed` ... rychlostní limit silnice podle OSM map,
- `osm_road_dist` ... vzdálenost stanice od silnice v metrech a
- `osm_highway` ... logická proměnná, která má `TRUE` pro stanice, které podle OSM map leží na dálnici.

Postup umístění stanic na silniční síť je následující:

1. Vyberou se silnice, na kterých může stanice ležet (u Arc ČR všechny silnice, u OpenStreetMap jen vybrané silnice, např. ne servisní silnice, což jsou často nájezdy a výjezdy ze stanice na silnici).
2. Najde se nejbližší silnice dané třídy, která je maximálně 150 m daleko od umístění stanice.
3. Přidá se příznak dálnic stanicím, které mají přidělenou silnici dané třídy (u Arc ČR 1 a 2, tj. dálnice a silnice pro motorová vozidla; u OSM map je to složitější, viz kód); v implicitním nastavení přidává výpočet podle OSM map i některé stanice v Praze, Brně a Ostravě, které jsou na vnitřním okruhu, který spojuje dálnice, které do města přicházejí

Pozor: Umístění stanic na silniční síť není vždy zcela spolehlivé, a to ze tří důvodů: 1) mapy (zvlášť Arc ČR) nemají silnice zakreslené zcela přesně (Arc ČR ani neobsahují všechny silnice, např. ulice ve městě), 2) stanice nemusejí být vždy zcela přesně zaměřené a 3) jedna stanice může fakticky ležet nebo být snadno dostupná z více silnic.

Příznak pro umístění stanice na dálnici podle Arc ČR (`arc_highway`) se může lišit do příznaku umístění na dálnice podle map OSM (`osm_highway`). Výběr dálnic je možné udělat jako sjednocení obou množin, použít jen OSM dálnice, nebo sjednocení obou množin s přihlédnutím k rychlosti na

cestě. Nejlepší je natáhnout lokační tabulku do GISu a upravit ji ručně, aspoň pro stanice ležící na dálnici.

Proces přidávání silnic je náročný na procesor i paměť. Pomůže, pokud se prvně odstraní nepotřebné čerpací stanice (ty, které jsou mimo ČR, které neexistují nebo neprodávají dané palivo), viz oddíl 5.2.

5.2 Odstranění čerpacích stanic

Pokud data obsahují stanice mimo území České republiky nebo čerpací stanice, které neobsahují žádná cenová pozorování (jako je tomu např. u dat poskytnutých Pumpdroidem), je třeba tato pozorování odstranit.

K odstranění stanic mimo území ČR slouží funkce `remove_foreign_gas_stations()`. Má dva parametry: tabulku, ze které se stanice mimo území ČR odstraňují a tabulku lokací stanic. Ta už musí obsahovat sloupec `country` přidaný funkcí `add_arccr_municipalities()`, viz oddíl 5.1.1.

```
price_data <- remove_foreign_gas_stations(price_data, station_locations)
station_locations <- remove_foreign_gas_stations(station_locations,
                                                 station_locations)
station_names <- remove_foreign_gas_stations(station_names, station_locations)
station_sgs <- remove_foreign_gas_stations(station_sgs, station_locations)
station_nonostop <- remove_foreign_gas_stations(station_nonostop,
                                                station_locations)
```

K odstranění stanic, které neobsahují žádná cenová pozorování, slouží funkce `remove_gas_stations_with_no_price_data()`. Ta má opět dva parametry: tabulku, ze které se stanice mimo území ČR odstraňují a tabulku cen.

```
station_locations <- remove_gas_stations_with_no_price_data(station_locations,
                                                            price_data)
station_names <- remove_gas_stations_with_no_price_data(station_names,
                                                        price_data)
station_sgs <- remove_gas_stations_with_no_price_data(station_sgs, price_data)
station_nonostop <- remove_gas_stations_with_no_price_data(station_nonostop,
                                                           price_data)
```

5.3 Příprava tabulek vzdáleností mezi stanicemi

Některé metriky konkurence a některé typy odhadů prostorových ekonometrických modelů vyžadují zadat vzdálenosti mezi čerpacími stanicemi. Software **MergerSim** umožňuje spočítat vzdálenosti dvou typů: vzdálenosti vzdušnou čarou a typické dojezdové vzdálenosti po silniční síti.

5.3.1 Vzdálenosti vzdušnou čarou

Vzdálenosti vzdušnou čarou spočítá funkce `get_beeline_distances()`. Jejím jediným vstupem je lokační tabulka:

```
station_beeline_distances <- get_beeline_distances(station_locations)
```

Protože vzdálenost vzdušnou čarou je symetrická, tj. vzdálenost z A do B je vždy stejná jako vzdálenost z B do A, je výsledkem tabulka, která každou dvojici stanic obsahuje pouze jednou (obsahuje vzdálenosti ze stanice s `id` $= x$ do stanice s `id` $= y$, kde $x < y$). Struktura této tabulky je následující:

from_id	to_id	beeline_distance
1	2	186773.3
1	3	187928.8
1	4	225902.6
1	5	222061.4
1	6	228006.5

kde

- `from_id` a `to_id` (třída *integer*) ... id dvou stanic a
- `beeline_distance` (třída *double*) ... vzdálenost mezi těmito stanicemi vzdušnou čarou v metrech.

Pozor: Pokud je stanic větší počet (v řádu tisíců), může výpočet i s použitím většího počtu jader trvat několik desítek minut, a to přesto, že vlastní hledání vzdáleností vzdušnou čarou je poměrně rychlá operace.

5.3.2 Dojezdové vzdálenosti

Dojezdové vzdálenosti mezi stanicemi najde funkce `get_driving_distances()`, která k tomu používá drobný prográmek `RoadRunner.jar`. Tento prográmek (vytvořený jako součást tohoto softwaru) vyžaduje ke svému běhu Javu, viz oddíl 2.1. Jedná se o wrapper nad navigačním softwarem GraphHopper.[2] Ke svému běhu potřebuje OSM mapy v komprimovaném formátu `czech-republic-latest.osm.pbf`, viz oddíl 2.3.

Hledání dojezdových vzdáleností je výpočetně náročná záležitost. Proto funkce umožňuje omezit hledání dojezdových vzdáleností na ty páry stanic, jejichž vzdálenost vzdušnou čarou nepřesahuje určitou vzdálenost. K tomu slouží parametr `max_dist`, který tuto vzdálenost zadává v kilometrech. Pokud už jsou spočítané vzdálenosti vzdušnou čarou, je možné je zadat; jinak je funkce sama spočítá.

[2] GraphHopper je navigační software, který umožňuje hledat nejkratší a nejrychlejší dojezdové vzdálenosti na mapách OSM. Zde je použita open source verze tohoto softwaru, která umožňuje off-line navigaci. Detaily viz https://www.graphhopper.com/.

```
station_driving_distances <- get_driving_distances(
    station_locations,
    station_beeline_distances,
    max_dist = 40,
    map_path = NAVIGATION_MAP,
    roadrunner_folder = ROADRUNNER_FOLDER
)
```

Funkce umožňuje hledat buď délku nejkratší nebo nejrychlejší trasy. Implicitně se hledá délka nejrychlejší trasy, a to ze dvou důvodů: 1) nejkratší trasy často vedou problematickými úseky, takže většina řidičů v navigacích volí spíše nejrychlejší cestu – nejrychlejší cesta tedy zřejmě lépe popisuje vzdálenosti mezi dvěma stanicemi a 2) OSM mapy v minulosti obsahovaly na některých silnicích chybné údaje, které způsobily, že nejkratší vzdálenost nemohla být nalezena. V posledních vydáních OSM map se sice zdá, že je tato chyba už opravena, může se však kdykoli vrátit.

Protože dojezdová vzdálenost není symetrická, tj. vzdálenost z A do B může být kratší nebo delší než vzdálenost z B do A, obsahuje výsledná tabulka vzdálenosti oběma směry. Její struktura je následující:

from_id	to_id	driving_distance	driving_time
1	67	21632.087	20.80548
1	139	23444.357	22.76983
1	149	32518.843	22.16587
1	160	22704.108	22.17017
1	213	9949.188	9.85230

kde

- `from_id` (třída *integer*) ... id výchozí stanice,
- `to_id` (třída *integer*) ... id cílové stanice,
- `driving_distance` (třída *double*) ... zvolená dojezdová vzdálenost ze stanice `from_id` do stanice `to_id` v metrech a
- `driving_time` (třída *double*) ... očekávaná doba jízdy ze stanice `from_id` do stanice `to_id` v minutách.

Pozor: Očekávaná dojezdová doba je spočítaná z nejvyšších povolených rychlostí jízdy uvedených v OSM mapách a neuvažuje skutečný provoz. Nemusí tak odpovídat skutečné době jízdy, která se navíc může velmi lišit podle dne v týdnu a denní doby.

5.4 Tvorba argegovaných dat

Před vlastní ekonometrickou analýzou je potřeba data ze všech tabulek spojit, doplnit chybějící hodnoty a agregovat. K tomu slouží funkce `prepare_average_fuel_data()`. Tato funkce provede několik operací:

1. vybere data o prodeji jednoho vybraného typu produktu (zde paliva),
2. spojí cenovou tabulku, tabulku lokací, tabulku značek a případné tabulky vlastností,

3. doplní chybějící údaje poslední známou hodnotou,
4. agreguje výsledné hodnoty za určitá období a
5. odhadne, zda daná stanice v daném období skutečně existovala.

Prvním parametrem funkce je typ produktu (zde benzín E5, v datech poskytnutých Pumpdroidem označený jako "NATURAL95"), dalším je délka období, na které se mají data agregovat (zde 1 měsíc, ale může být 1 týden, kvartál, rok, nebo i 1 den, pokud se data agregovat nemají), následuje délka období pro odhad existence stanice a jednotlivé vstupní tabulky v pořadí: tabulka cen, tabulka lokací, tabulka značek a tabulky vlastností.

```
natural_monthly <- prepare_average_fuel_data("NATURAL95",
                                             by = "1 month",
                                             exist_lag = 30L,
                                             price_data,
                                             station_locations,
                                             station_names,
                                             station_sgs,
                                             station_nonostop,
                                             remove_source = TRUE)
```

Příklad několika prvních řádků i sloupců výsledku:

from_date	to_date	id	fuel	name	price	no_of_true_price_obs	sgs
2013-01-01	2013-01-31	1	NATURAL95	agip	37.05194	22	FALSE
2013-01-01	2013-01-31	2	NATURAL95	agip	35.37742	23	FALSE
2013-01-01	2013-01-31	3	NATURAL95	agip	35.38710	7	FALSE
2013-01-01	2013-01-31	4	NATURAL95	agip	35.91613	21	FALSE
2013-01-01	2013-01-31	5	NATURAL95	agip	37.73226	21	FALSE
2013-01-01	2013-01-31	6	NATURAL95	agip	35.63548	5	FALSE

Výsledná tabulka obsahuje následující sloupce:

- `from_date` a `to_date` (třída *Date*) ... začátek a konec období, pro které data platí,
- `id` (třída *integer*) ... id stanice,
- `fuel` (třída *character*) ... název produktu,
- `name` (třída *character*) ... název značky stanice platný v daném období převzatý z tabulky značek,
- `price` (třída *double*) ... průměrná cena v období vypočtená z tabulky cen,
- `no_of_true_price_obs` (třída *integer*) ... počet skutečných (tj. neinterpolovaných pozorování v období,
- `sgs`, `nonstop` ... hodnoty doplněné z tabulky vlastností (zde dvě logické hodnoty, zda má v daném období čerpací stanice certifikát SGS a zda má otevřeno non-stop) – v případě jiných tabulek vlastností zde budou jiná jména (pokud tabulky vlastností nebudou zadány, budou tyto sloupce chybět),
- `station_exists_X` ... logická hodnota, zda čerpací stanice v daném období pravděpodobně existuje podle zadaného kritéria (zde je `X` rovno 30; detaily jsou popsané níže),
- `lat`, `lon`, `gps`, a `address` ... hodnoty převzaté z tabulky lokací,

- `obec`, `okres`, `kraj` a `country` ... hodnoty převzaté z tabulky lokací, kam byly přidány funkcí `add_arccr_municipalities()`, viz oddíl 5.1.1,
- `arc_road_class`, `arc_road_int_class`, `arc_road_number`, `arc_road_lanes`, `arc_road_distance`, `arc_highway` ... hodnoty převzaté z tabulky lokací, kam byly přidány funkcí `add_arccr_roads()`, viz oddíl 5.1.2,
- `osm_road_class`, `osm_oneway`, `osm_maxspeed`, `osm_road_dist` a `osm_highway` ... hodnoty převzaté z tabulky lokací, kam byly přidány funkcí `add_osm_roads()` viz oddíl 5.1.2.

Existují dva důvody, proč je dobré data pomocí funkce `prepare_average_fuel_data()` agregovat za delší časové období. První je ten, že vstupní data nemusí být kompletní – zejména nemusí obsahovat cenovou informaci pro každý den. Např. data z Pumpdroidu mají údaje o cenách hlášené uživateli této platformy. Ti většinou nezadávají cenu pokaždé, když čerpají dané palivo, ale pouze v případě, že si všimnou, že se cena paliva změnila. Na frekventovaných stanicích se změna ceny může promítnout do dat okamžitě; na méně frekventovaných i s několikadenním zpožděním. Z tohoto důvodu může být užitečné nejen data doplnit poslední známou hodnotou, ale i data zprůměrovat za určité období, aby se jednak snížil šum v datech, jednak data nepředstírala více informací, než je jich skutečně k dispozici. Druhým důvodem je to, že spočítat měřítka konkurence (viz oddíl 5.5) je nutné pro každé období znovu a tento výpočet je poměrně nákladný. Jako příklad používáme data za 6 let, tj. více než 2 000 dnů. Výpočet tolika měřítek konkurence je prohibitivně nákladný i s použitím více jader.

Data se proto nejprve vždy doplní tak, aby obsahovala všechny dny, pak se chybějící hodnoty doplní poslední známou hodnotou. Následně se data agregují za zvolené období (zde 1 měsíc). Sloupec `price` pak obsahuje průměr z doplněných cen. Sloupec `name` obsahuje nejčastěji uváděnou značku v daném období; v případě shodného počtu výskytů první jméno značky. Stejný postup se použije i v případě příznaků vlastností (zde `sgs` a `nonstop`). Údaje z tabulky lokací jsou konstantní pro celé období.

Sloupec `no_of_true_price_obs` udává skutečný počet cenových pozorování v daném období. Logický sloupec `station_exists_X` (kde `X` je nějaké číslo, zde 30) má hodnotu `TRUE` pro ta období, ve kterých stanice pravděpodobně existovala a prodávala daný produkt. Přesný výpočet je takový, že hodnota je `TRUE`, pokud v daném období rozšířeném o `X` dnů před jeho začátkem a `X` dnů po jeho konci existuje v datech nějaké skutečné (tj. ne interpolované) cenové pozorování. Hodnotu `X` je možné zadat pomocí parametru `exist_lag`. Parametr `exist_lag` může být i vektor celých čísel. V tom případě se do dat přidá odpovídající počet sloupců `station_exists_X`.

Délka období se zadává pomocí parametru `by`. Jeho implicitní hodnota je `"1 month"`. Zadání délky `by = "1 day"` zabrání agregaci dat; chybějící hodnoty však budou stále doplněny.

5.5 Přidání statistik lokální konkurence

Jakmile je vytvořena tabulka agregovaných dat (viz oddíl 5.4), je možné do ní přidat statistiky konkurence. Software implementuje tři typy statistik konkurence:

1. dojezdovou vzdálenost k nejbližšímu konkurentovi,
2. počet konkurentů v zadaných dojezdových vzdálenostech a
3. měřítko prostorového klastrování (*spatial clustering measure*, SC) odvozené Pennerstorferem a Weissem (2013).

Statistiky konkurence se počítají pro každé období v tabulce agregovaných hodnot zvlášť.

Všechny funkce pro výpočet statistik konkurence obsahují vždy dva speciální parametry, které proto popíšeme společně. 1) Tyto funkce obsahují nepovinný parametr `existence`, který obsahuje jméno sloupce v agregované tabulce, který indikuje, zda daná stanice v daném období existuje. Statistiky konkurence se počítají pouze z existujících stanic. Pokud není parametr zadán, pak se předpokládá, že všechny stanice existují.

2) Funkce pro výpočet statistik konkurence vždy obsahují nepovinný parametr `same_group`, který umožňuje zadat tabulku, která ukazuje, které značky jsou vlastněné stejnou firmou, a tedy si nekonkurují, viz oddíl 3.5. Pokud není tabulka zadaná, pak se předpokládá, že si vzájemně konkurují každé dvě stanice, které mají různé značky; naopak stanice, které mají stejné značky si nekonkurují. Pokud má aspoň jedna z dvojice značek neznámé jméno (indikované hodnotou `NA`), pak se předpokládá, že si tyto dvě značky konkurují. Pokud je tabulka zadaná, pak se předpokládá, že si značky ve stejné skupině nekonkurují; jinak platí předchozí pravidla.

Poznámka: Hodnocení kauzálního vlivu jednotlivých měřítek konkurence může být obtížné, protože tato měřítka jsou typicky korelovaná. Takto vzniklá multikolinearita však nepředstavuje problém pro tvorbu predikcí, a tedy ani pro simulace fúzí. Naopak, tato měřítka se vzájemně dobře doplňují a měří různé aspekty lokální konkurence.

5.5.1 Dojezdová vzdálenost k nejbližšímu konkurentovi

Dojezdová vzdálenost k nejbližšímu konkurentovi je nejprimitivnější měřítko konkurence. Předpokládá se, že čím je vzdálenost k nejbližšímu konkurentovi větší, tím menší konkurenci stanice čelí, a tedy tím vyšší cenu si (ceteris paribus) účtuje. Dojezdovou vzdálenost k nejbližšímu konkurentovi do agregované tabulky přidá funkce `add_distance_to_closest_competitor()`. Kromě obecných parametrů popsaných výše má dva parametry: doplňovanou tabulku agregátních hodnot a tabulku dojezdových vzdáleností vytvořenou pomocí funkce `get_driving_distances()`.

```
natural_monthly <- add_distance_to_closest_competitor(
    natural_monthly,
    station_driving_distances,
    existence = existence_criterium,
    same_group = same_owners_correction_table
)
```

Funkce `add_distance_to_closest_competitor()` přidá do tabulky dva sloupce:

- `distance_to_closest_competitor` (třída *double*) … vzdálenost k nejbližšímu konkurentovi v kilometrech a
- `distance_to_closest_same_brand` (třída *double*) … vzdálenost k nejbližší stanici stejné značky v kilometrech.

Sloupce mohou mít hodnotu `NA` v případě, že daná stanice v daném období neexistuje nebo se nenašel žádný její soused. Pokud je ve sloupci `distance_to_closest_competitor` hodnota `NA` druhého typu, je třeba přegenerovat tabulku dojezdových vzdáleností s vyšší hodnotou parametru `max_dist`. Naopak hodnoty `NA` ve sloupci `distance_to_closest_same_brand` mohou nastat zcela přirozeně tak, že stanice patří značce s jedinou provozovnou.

5.5.2 Počet konkurentů v zadaných dojezdových vzdálenostech

Sílu lokální konkurence neovlivňuje jen vzdálenost k nejbližšímu konkurentovi, ale i počet konkurentů na lokálním trhu. Lokální trhy bývají definovány různě. Nejobvyklejší je definice lokálního trhu jako území v určité vzdálenosti (vzdušnou čarou nebo lépe dojezdové) od dané stanice. Tento přístup však trpí určitými problémy: 1) použitá vzdálenost je nutně arbitrární, 2) dá se předpokládat, že vliv konkurence klesá se vzdáleností spíše spojitě, než aby byl do určité hranice konstantní a za ní okamžitě klesl na nulu. Proto Kvasnička, Staněk a Krčál (2018)[3] tento přístup zobecnili a počítají počet konkurenčních stanic v dojezdových vzdálenostech 0–X km, X–2X km, 2X–3X km atd. (tj. v jakýchsi zobecněných "mezikružích"). Ukazují, že lokální konkurence roste (a cena tedy ceteris paribus klesá) s počtem konkurentů v jednotlivých vzdálenostech a že vliv konkurentů klesá s jejich vzdáleností. Vliv konkurence je v datech detekovatelný cca do 6 až 9 kilometrů.

Tato měřítka konkurence je možné do tabulky agregátních hodnot přidat pomocí funkce `add_competitors_in_distances()`. Kromě výše zmíněných obecných parametrů vyžaduje funkce zadat doplňovanou tabulku agregátních hodnot, tabulku dojezdových vzdáleností a vektor `distances`, který udává, v jakých vzdálenostech se počítá počet stanic. Zde se používá vektor 0, 3, 6, 9, 12, tj. počítá se počet konkurenčních stanic ve vzdálenostech 0–3 km, 3–6 km, 6–9 km a 9–12 km od každé stanice.

```
natural_monthly <- add_competitors_in_distances(
    natural_monthly,
    station_driving_distances,
    distances = seq(from = 0, to = 12, by = 3),
    existence = existence_criterium,
    same_group = same_owners_correction_table
)
```

Funkce přidá do tabulky agregovaných hodnot sloupce:

- `other_stations_in_driving_distance_X_Y` (třída *double*) … počet konkurenčních stanic v dojezdové vzdálenosti od `X` do `Y` a
- `own_stations_in_driving_distance_X_Y` (třída *double*) … počet stanic se stejným vlastníkem v dojezdové vzdálenosti od `X` do `Y`.

V našem případě tak přibudou sloupce `other_stations_in_driving_distance_0_3`, `other_stations_in_driving_distance_3_6`, `other_stations_in_driving_distance_6_9` a ‘`other_stations_in_driving_distance_9_12` a odpovídající sloupce s počty vlastních stanic.

5.5.3 Měřítko prostorového klastrování stanic (SC)

Pennerstorfer a Weiss (2013)[4] odvodili měřítko lokální konkurence, které stojí na zcela jiném principu než jsou vzdálenosti mezi stanicemi. Vyšli z intuice Salopova modelu. V něm je firma do jisté míry chráněna před konkurencí v případě, kdy jsou jejími sousedy členové stejného řetězce. Pokud jeden ze členů řetězce zvýší svou cenu, někteří z jeho zákazníků odejdou; část z nich však přejde

[3] Kvasnička, Staněk a Krčál: Is the Retail Gasoline Market Local or National?, *Journal of Industry, Competition and Trade*, 18, 2018, s. 47–58.

[4] Pennerstorfer a Weiss: Spatial clustering and market power: Evidence from the retail gasoline market, *Regional Science and Urban Economics*, 43, 2013, s. 661–675.

k jiným členům stejného řetězce. Řetězec si tak může účtovat vyšší ceny, pokud jsou jeho členové geograficky klastrovaní. Pennerstorfer a Weiss vytvořili měřítko prostorového klastrování (*spatial clustering measure*) a empiricky prokázali, že ceny na jednotlivých čerpacích stanicích nezávisí jen na počtu konkurentů na jejich lokálním trhu, ale i na prostorovém klastrování stanic stejného typu.

Jejich měřítko se počítá takto: 1) Celá země se rozdělí na Voronoi polygony okolo jednotlivých stanic. 2) Vytvoří se klastry polygonů stejné značky, nebo obecněji, značek, které si nekonkurují. Každý klastr se skládá ze všech polygonů stejné značky, které mají společné hranice, takže mohou být spojeny čárou, která nemusí projít přes konkurenční polygon. 3) Pro každou stanici i se spočítá počet klastrů M_i, které se dotýkají polygonu stanice i (včetně jejího vlastního klastru). Dále se spočítá počet stanic k_{mi} v každém klastru m_i a počet všech stanic N_i, jejichž polygon má společnou hranici s polygonem stanice i (včetně stanice i). Měřítko prostorového klastrování $\mathrm{SC}_i = \sum_{m_i} k_{m_i}/M_i/N_i$. Čím více jsou stanice stejné značky geograficky klastrované, tím vyšší je SC_i. Pokud je stanice i plně obklopena jinými stanicemi stejné značky, pak je $\mathrm{SC}_i = 1$.

Toto měřítko konkurence přidáme do tabulky agregovaných hodnot pomocí funkce `add_spatial_clustering()`. Ta má kromě obecných parametrů popsaných výše jen dva parametry: doplňovanou tabulku agregovaných hodnot a cestu k adresáři `AdministrativniCleneni_v13.gdb`.

```
natural_monthly <- add_spatial_clustering(
    natural_monthly,
    ARCCR_ADMIN_FOLDER,
    existence = existence_criterium,
    same_group = same_owners_correction_table
)
```

Funkce `add_spatial_clustering()` přidá do tabulky agregovaných hodnot sloupce:

- `sc` (třída *double*) … *spatial clustering measure* SC,
- `border_stations` (třída *logical*) … `TRUE` pokud se polygon stanice dotýká hranice ČR,
- `voronoi_neighbors_id` (seznam vektorů třídy *integer*) … id Voronoi sousedů,
- `voronoi_area` (třída *integer*) … obsah vlastních Voronoi polygonů v km^2,
- `cluster_area` (třída *double*) … obsah vlastních klastrů v km^2,
- `cluster_members_ids` (seznam vektorů třídy *integer*) … id členů vlastního klastru.

5.6 Přidání speciálních efektů

Před ekonometrickým modelováním je možné přidat do tabulky agregovaných hodnot i vybrané fixní efekty. Software usnadňuje přidání tří efektů:

- efektu pro velikost municipality,
- fixního efektu pro velké značky a
- fixního efektu pro fúzi velkých značek.

5.6.1 Přidání efektů pro velikost obce

Při ekonometrickém modelování na průřezových datech může být užitečné kontrolovat pro velikost municipality, v jejímž katastru se daná stanice nachází. Funkce `add_city_effects()` přidá umělé proměnné pro Prahu, Brno a Ostravu a skupiny měst podle velikosti.

```
natural_monthly <- add_city_effects(natural_monthly)
```

Funkce přidá do tabulky následující logické sloupce:

- `praha` ... `TRUE`, pokud je stanice v katastru hlavního města,
- `brno` ... `TRUE`, pokud je stanice v katastru města Brna,
- `ostrava` ... `TRUE`, pokud je stanice v katastru Ostravy,
- `mesto300` ... `TRUE`, pokud je stanice v katastru města s počtem obyvatel mezi 300 tisíci a 1 milionem, dnes totéž co Brno nebo Ostrava,
- `mesto100` ... `TRUE`, pokud je stanice v katastru města se 100 až 300 tisíci obyvateli, dnes totéž co Plzeň, Liberec nebo Olomouc,
- `mesto50` ... `TRUE`, pokud je stanice v katastru města s 50 až 100 tisíci obyvateli, a
- `mesto20` ... `TRUE`, pokud je stanice v katastru města s 20 až 50 tisíci obyvateli.

Samozřejmě není možné použít naráz proměnné `brno`, `ostrava` a `mesto300`, protože by došlo k dokonalé multikolinearitě.

Tvorba těchto proměnných vyžaduje dodatečnou tabulku měst, která je umístěná přímo ve funkci `add_city_effects()`. Jednou za čas je třeba ověřit velikosti obcí a tuto tabulku aktualizovat.

5.6.2 Přidání efektů velkých značek

Jednotlivé značky mohou nastavovat ceny různě, a to i v případě jinak relativně homogenního produktu. Rozdíly v cenové úrovni značek mohou odrážet lepší pověst, služby nebo i mírné odchýlení od homogenity produktu (např. některé čerpací stanice přimíchávají do svého paliva různá aditiva). Při regresi je možné o tuto variabilitu kontrolovat pomocí různých úrovňových konstant pro různé velké značky. K přidání těchto konstant slouží funkce `find_big_brands_in_prepared_data()` a `add_big_brand_name_effects()`.

Funkce `find_big_brands_in_prepared_data()` vezme tabulku agregátních hodnot vyfiltrovanou pro jedno období a vrátí vektor velkých značek, tj. názvy těch značek, které mají aspoň `min_n` stanic; implicitní hodnota parametru `min_n` je 10.

Funkce `add_big_brand_name_effects()` vezme tabulku agregátních hodnot a doplní do ní sloupec `brand_name`, což je faktor s úrovněmi pro jednotlivé velké značky; referenční úroveň má hodnotu `"other"`, která indikuje, že daná stanice nepatří žádné velké značce. Funkci je buď možné zadat vektor jmen velkých značek, nebo minimální počet stanic, které musí značka mít, aby byla považovaná za velkou. Ve druhém případě funkce najde názvy velkých značek pomocí funkce `find_big_brands_in_prepared_data()` (funguje pouze pro data vyfiltrovaná pro jedno období).

```
cs_data <- natural_monthly %>%
    filter(from_date == "2014-01-01") %>%
    add_big_brand_name_effects()
```

5.6.3 Přidání efektů fúzovaných značek

V případě panelové regrese můžeme použít fixní efekty na čas i jednotlivé čerpací stanice. To vylučuje použití efektů na velikost města i efektů na velké značky. I v tomto případě je však potřeba zachytit vliv fúze na změnu značky. K tomu slouží sloupec `brand_change`. Tento sloupec musí být faktor, jehož úrovně musejí být `"none"` (referenční úroveň), která indikuje, že v tomto období neprošla stanice žádnou fúzí, nebo mít jméno dané fúze.

Pokud jsou data dobře vyčištěná (např. pomocí funkce `correct_brandnames_by_brute_force()`, viz oddíl 4.2.3), je možné vytvořit tento vektor automaticky. To je navíc nezbytné pro simulaci fúzí, kde je třeba aktualizovat jednotlivé statistiky automaticky. Ke strojové přípravě sloupce `brand_change` slouží funkce `add_brand_changes_in_takeover()`. Jejím prvním parametrem je tabulka agregovaných hodnot, druhým tabulka převzetí firem, která má stejnou strukturu jako korekční tabulka pro funkci `correct_brandnames_by_brute_force()`, viz oddíl 4.2.3.

Příklad takové tabulky:

takeover_date	terminal_date	source	target
2014-08-01	2018-12-31	lukoil	mol
2015-08-01	2018-12-31	agip	mol
2016-02-01	2018-12-31	omv	benzina
2016-02-01	2018-12-31	avanti	benzina

kde

- `takeover_date` (třída *Date*) je datum převzetí,
- `terminal_date` (třída *Date*) je datum, kdy je převzetí dokončeno
- `source` ... (třída *character*) je jméno převzaté značky a
- `target` ... (třída *character*) je jméno značky, která koupila převzatou značku.

Použití funkce je pak následující:

```
natural_monthly <- add_brand_changes_in_takeover(natural_monthly, takeovers)
```

Pro výše uvedenou korekční tabulku a data z Pumpdroidu bude mít sloupec `brand_change` úrovně `"none"`, `"agip -> mol"`, `"avanti -> benzina"`, `"lukoil -> mol"` a `"omv -> benzina"`.

5.7 Poznámka ke jménům sloupců agegované tabulky

Všechna jména sloupců agregované tabulky popsané v této kapitole jsou závazné a není možné je měnit. Pokud je změníte, nebude simulace fúze popsaná v oddíle 8 fungovat správně!

6 Vizualizace připravených dat

Data je možné vizualizovat mnoha různými způsoby podle potřeb uživatele. Tato kapitola popisuje to, jak je možné vizualizovat agregovaná data přichystaná pro ekonometrickou analýzu na mapových podkladech. Vizualizace reziduí ekonometrických modelů na mapě je popsaná v kapitole 7 v oddíle 7.5; vizualizace důsledků fúze je popsaná v kapitole 8 v oddíle 8.4.

6.1 Obecný přístup k vizualizace dat na mapových podkladech

Veškeré funkce pro vizualizaci dat na mapových podkladech, které jsou součástí softwaru **MergerSim**, jsou implementované ve skriptu `graphing_functions.R`. Tyto funkce jsou postavené nad balíkem **tmap** a vracejí objekt třídy *tmap*. To znamená, že je k dispozici veškerá funkcionalita tohoto balíku. V tomto oddíle se podíváme pouze na tři obecné věci:

- interaktivní mód map,
- ukládání map do souboru a
- předávání vizualizačních parametrů funkcí z balíku **tmap** uživatelským funkcím implementovaným v tomto softwaru.

6.1.1 Interaktivní práce s grafy

Balík **tmap** umožňuje dva typy práce s grafy: tvorbu statických grafů (implicitní nastavení) a tvorbu interaktivních grafů. Mezi těmito dvěma módy přepíná funkce `tmap_mode()`. Po přepnutí do interaktivního módu výrazem

```
tmap_mode("view")
```

se všechny grafy otevřou v prohlížeči RStudia, případně v systémovém prohlížeči webových stránek. Prohlížeč zobrazuje na pozadí mapový poklad (jde volit ze tří možností), umožňuje zoomovat a zobrazovat detailnější informace o zvoleném objektu kliknutím na tento objekt.

Zpět do statického módu je možné se přepnout výrazem

```
tmap_mode("plot")
```

6.1.2 Ukládání grafů do souboru

K ukládání grafů třídy *tmap* do souboru slouží funkce `tmap_save()`. Jejím prvním parametrem je vykreslovaný graf třídy *tmap*, druhým jméno souboru. Další parametry včetně nastavení velikosti obrázku, jeho rozlišení apod. jsou popsány v dokumentaci této funkce.

Funkce umí zapsat obrázek do bitmapových formátů PNG, JPG, BMP a TIFF, vektorových formátů PDF, EPS a WMF (pouze na Windows) a do HTML. Bitmapové a vektorové formáty umožňují zapsat statický obrázek. Uložení grafu do HTML vytvoří off-line webovou stránku s interaktivním grafem, viz oddíl 6.1.1. O volbě typu souboru rozhoduje jeho koncovka.

6.1.3 Předání vizualizačních parametrů *tmap* uživatelským funkcím

Všechny uživatelské funkce implementované v tomto softwaru ve skriptu `graphing_functions.R` umožňují pomocí `...` zadat dodatečné parametry, které jsou předané vlastním funkcím z balíku **tmap**, které vykreslují zpracovaná data. Typické použití je předání palety barev (parametr `palette`) nebo velikosti vykreslovaných teček (parametr `size`), je však možné použít libovolné další parametry použité ve funkcích `tm_dots()` a `tm_polygons()` z balíku **tmap**.

6.2 Vizualizace agregovaných vstupních dat na mapě

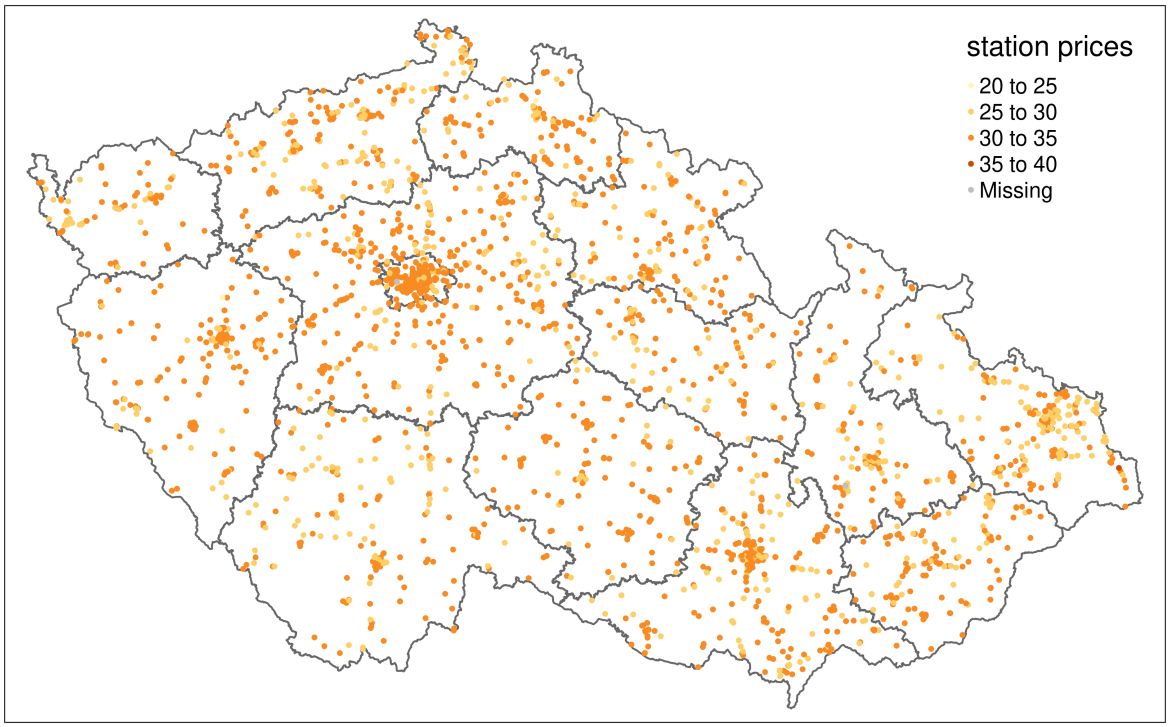
Software **MergerSim** implementuje dva typy vizualizace agregovaných vstupních dat připravených k regresi na mapě: pohled na jednotlivé čerpací stanice formou bodového grafu a pohled na agregátní statistiky spočítané pro jednotlivé regiony ČR. Příklady vizualizace vstupních dat připravených obsahuje skript `example_plotting.R`.

6.2.1 Grafy cen na individuálních stanicích

K vykreslení individuálních cen na jednotlivých stanicích formou bodového grafu slouží funkce `plot_station_prices()`. Jejím prvním parametrem jsou vstupní data vytvořená funkcí `prepare_average_fuel_data()` nebo jejich podmnožina. Další hlavní parametry zahrnují možnost zadat cestu k adresáři, kde jsou umístěné administrativní mapové podklady `AdministrativniCleneni_v13.gdb`, volba regionu, který se má na mapu vykreslit (`"nic"`, `"obce"`, `"okresy"`, `"kraje"` nebo `"cr"`) a jméno sloupce, který indikuje, zda stanice v daném čase existují.

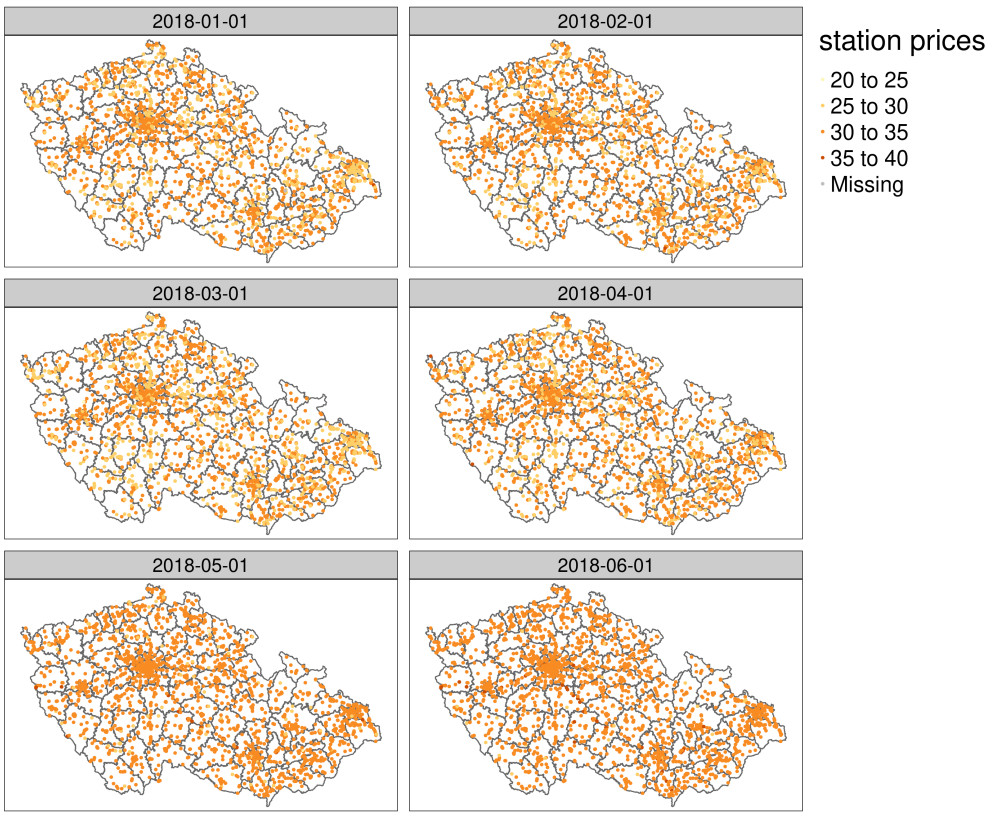
Typicky se předpokládá, že se vykreslí ceny v jednom období, což zajistí předchozí filtrace dat:

```
natural_monthly %>%
    filter(from_date == "2018-01-01") %>%
    plot_station_prices(arccr_folder = PATH_TO_ARCCR,
                        regions = "kraje",
                        existence = "station_exists_30")
```



Pokud je zvoleno více období, je možné zadat statistiku, která se z cen na jednotlivých stanicích spočítá. Implicitně je takovou statistikou průměr. Je však možné spočítat např. medián, směrodatnou odchylku apod. Alternativně je možné vykreslit několik dílčích grafů – každý pro jedno období – do matice:

```
natural_monthly %>%
    filter(from_date >= "2018-01-01", from_date <= "2018-06-01") %>%
    plot_station_prices(arccr_folder = PATH_TO_ARCCR,
                        regions = "okresy",
                        existence = "station_exists_30",
                        facet = TRUE, facet_col = 2)
```



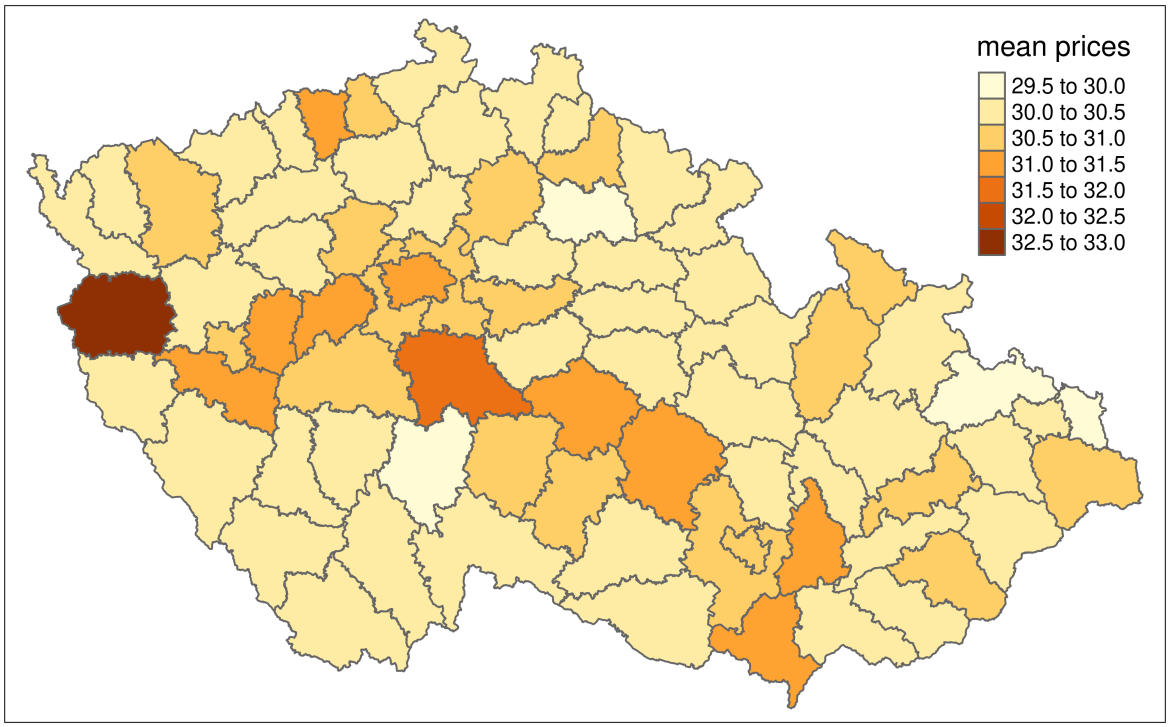
Funkce `plot_station_prices()` má mnoho dalších parametrů popsaných v dokumentaci ve skriptu `graphing_functions.R`. Skript `example_plotting.R` ukazuje mnoho dalších nastavení a použití této funkce.

6.2.2 Grafy průměrných cen v regionech (a dalších statistik)

Průměrné ceny v daných regionech spočítá a zobrací funkce `plot_average_prices()`. Její základní použití je velmi podobné jako v případě funkce `plot_station_prices()`. Nevykresluje však ceny na jednotlivých stanicích, nýbrž jejich průměr v zadaném typu regionů.

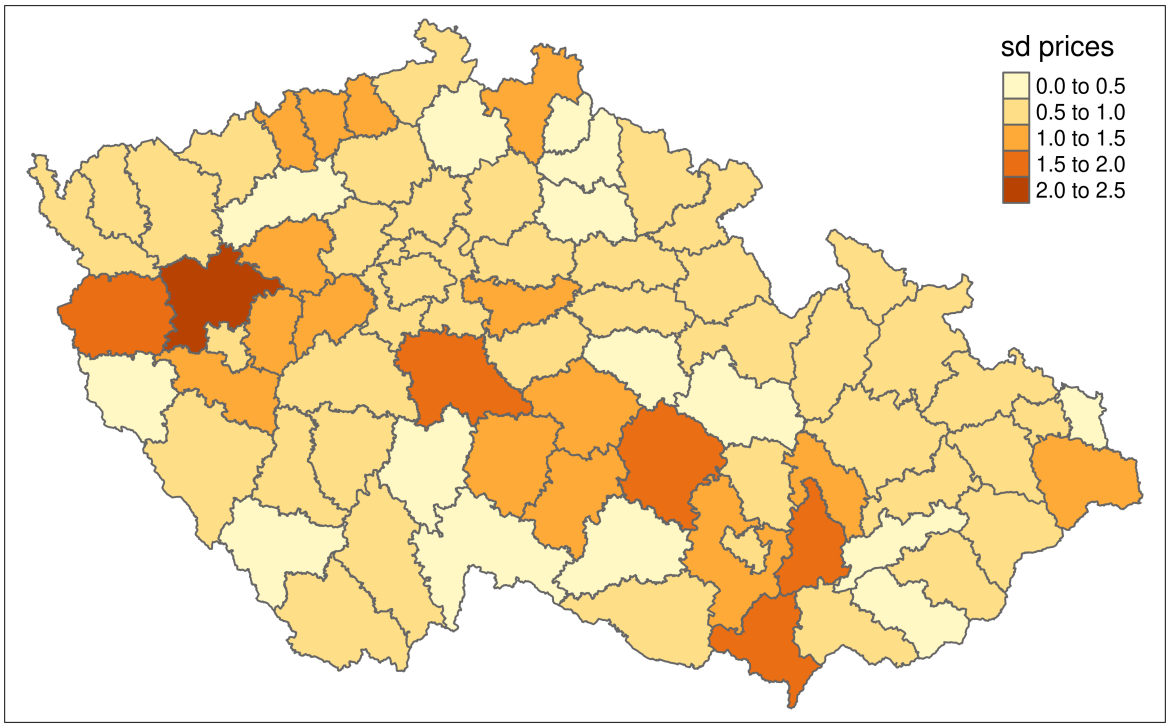
Typický je pohled na průměrné ceny v regionech v jednom období:

```
natural_monthly %>%
    filter(from_date == "2018-01-01") %>%
    plot_average_prices(arccr_folder = PATH_TO_ARCCR,
                        regions = "okresy",
                        existence = "station_exists_30")
```



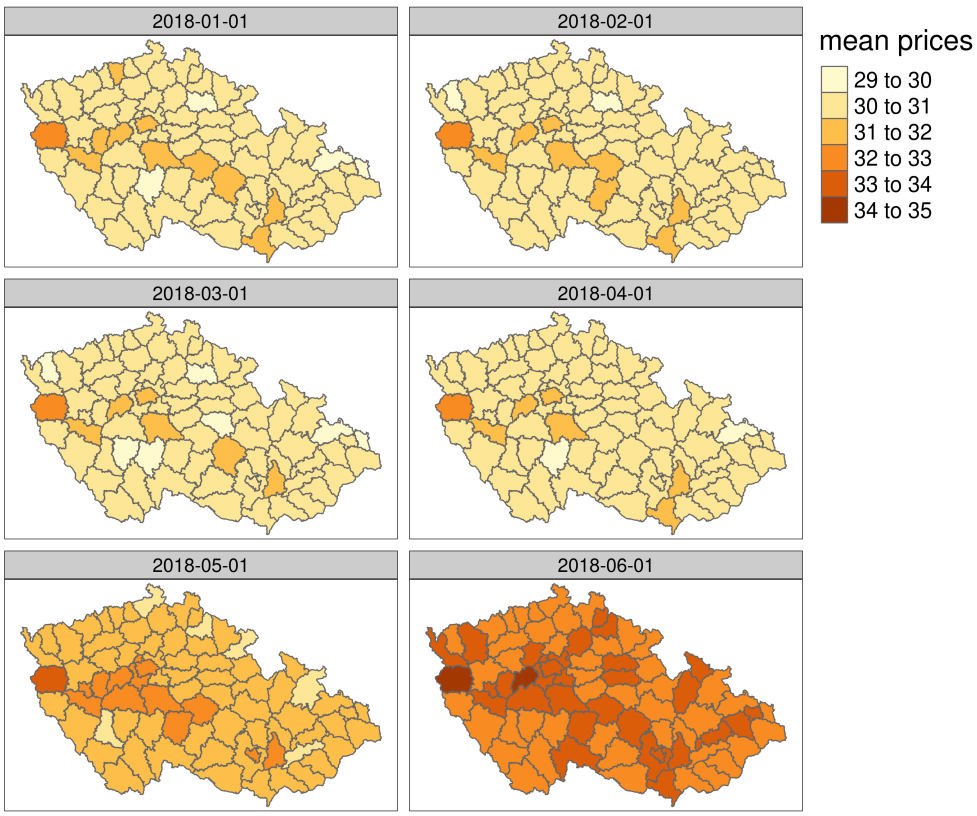
Funkce umí zobrazit i další statistiky, např. směrodatnou odchylku cen v jednotlivých regionech:

```
natural_monthly %>%
    filter(from_date == "2018-01-01") %>%
    plot_average_prices(arccr_folder = PATH_TO_ARCCR,
                        regions = "okresy",
                        existence = "station_exists_30",
                        stat = sd)
```



Stejně jako předchozí funkce umí i funkce `plot_average_prices()` zobrazit data za více období najednou – buď na ně aplikuje stejnou agregační funkci (implicitně průměr), nebo vykreslí pro každé období jeden dílčí graf do matice grafů:

```
natural_monthly %>%
    filter(from_date >= "2018-01-01", from_date <= "2018-06-01") %>%
    plot_average_prices(arccr_folder = PATH_TO_ARCCR,
                        existence = "station_exists_30",
                        facet = TRUE, facet_col = 2)
```



Funkce má mnoho dalších parametrů popsaných v její dokumentaci ve skriptu `graphing_functions.R`. Její různá nastavení a použití jsou ukázaná ve skriptu `example_plotting.R`.

7 Odhad ekonometrických modelů lokální konkurence

Tato kapitola popisuje, jak odhadnout ekonometrický model lokální konkurence, který je základem simulace fúze. Software **MergerSim** podporuje jak použití modelů bez prostorové složky (OLS, modely fixních vlivů), tak i model SAR, který prostorovou složku obsahuje. Modely mohou být odhadnuté jak na průřezových datech (tj. pro jedno období), tak na panelech.

Pro simulaci fúzí umí software **MergerSim** využít následující estimátory:

- OLS na průřezových datech: funkce `lm()` ze základního balíku **stats**,
- SAR na průřezových datech: funkce `lagsarlm()` z balíku **spatialreg**,
- model fixních vlivů na panelových datech: funkce `feols()` z balíku **fixest** a
- SAR s fixními efekty na panelových datech: funkce `spml()` z balíku **splm**.

Zde se tedy budeme zabývat pouze odhady pomocí těchto estimačních funkcí.

7.1 Význam zahrnutí prostorové autokorelace do modelu

Jednotlivé stanice prodávající homogenní produkt typicky vyhlašují vývěskové ceny, a tedy spolu hrají hru podobnou Bertrandovi. Jednotlivé stanice tak při stanovování své ceny berou do úvahy ceny okolních stanic. Tento způsob rozhodování vede k prostorové závislosti cen na jednotlivých stanicích. Dlouhodobá rovnováha tohoto procesu je prostorový autoregresivní data-generující proces SAR, viz LeSage a Pace 2009, 25–26.[5] Význam prostorové autokorelace cen potvrzuje empiricky pro konkrétní případ cen benzínu na čerpacích stanicích v ČR např. i předchozí výzkum, např. Kvasnička, Staněk a Krčál (2018)[6] a článek připravovaný v rámci tohoto projektu.

To znamená, že model SAR může být k zachycení prostorové autokorelace cen mezi jednotlivými stanicemi nezbytný. Pokud jsou ceny prostorově autokorelované a tato autokorelace není explicitně modelovaná, pak jsou 1) odhadnuté parametry vychýlené a nekonzistentní a 2) simulace fúze může výrazně podhodnocovat vliv fúze. (Pokud není uživatel seznámen se základy prostorové ekonometrie, může si je doplnit z některé vhodné publikace, např. výše citované knihy LeSage a Pace (2009).)

7.2 Odhad modelů na průřezových datech

Software umožňuje využít odhady modelu OLS i SAR na průřezových datech, pokud je to z nějakého důvodu potřeba. Kompletní ukázku odhadu a použití ekonometrického modelu na průřezových datech, ukazuje první část skriptu `example_crosssectional_estimation_and_merger_simulations.R`.

[5] LeSage a Pace: *Introduction to Spatial Econometrics*, CRC Press, 2009.

[6] Kvasnička, Staněk a Krčál: Is the Retail Gasoline Market Local or National?, *Journal of Industry, Competition and Trade*, 18, 2018, s. 47–58.

7.2.1 Příprava dat pro odhad

Před vlastním odhadem modelu připravíme data. Pro příklad opět použijeme data o cenách benzínu E5 na jednotlivých čerpacích stanicích. Budeme předpokládat, že máme připravená agregovaná data, jak je popisuje kapitola 5, a že do těchto dat byly přidány všechny tři typy statistik lokální konkurence. Dále předpokládáme, že data pro každou stanici a měsíc obsahují údaj o tom, zda je čerpací stanice otevřená non-stop.

Budeme vysvětlovat cenu benzínu na jednotlivých čerpacích stanicích v závislosti na výše uvedených měřítcích konkurence a údaji o otevření non-stop. Přitom budeme kontrolovat o velikost měst (Prahu, Brno a Ostravu budeme uvažovat jako svébytné kategorie), o jména velkých značek a o to, zda daná čerpací stanice leží na dálnici, nebo ne. Odhadovaná rovnice tedy bude mít tvar:

```
basic_model <- price ~
    sc + distance_to_closest_competitor +
    other_stations_in_driving_distance_0_3 +
    other_stations_in_driving_distance_3_6 +
    osm_highway +
    praha + brno + ostrava + mesto100 + mesto50 + mesto20 +
    brand_name +
    nonstop
```

Nejdříve převedeme ceny zvoleného paliva na halíře (tj. vynásobíme je stem), aby byly odhadnuté parametry a výsledky simulace fúze čitelnější. Pak data připravíme tak, že pro zvolené období najdeme jména velkých značek a do agregátních dat efekty pro tyto velké značky a pro velikost měst. Následně vyfiltrujeme z agregátních dat zvolené období a vyřadíme chybějící pozorování, protože námi použitý balík pro odhad SAR modelu na průřezových datech, **spatialreg**, neumožňuje mít v použitých proměnných chybějící hodnoty.

Efekty velkých měst přidáme pomocí funkce `add_city_effects()`, viz oddíl 5.6.1. Jména velkých značek najdete pomocí funkce `find_big_brands_in_prepared_data()`, efekty velkých značek přidáme pomocí funkce `add_big_brand_name_effects()`, viz oddíl 5.6.2. Neexistující stanice a stanice s nekompletními pozorováními můžeme odstranit pomocí funkce `prepare_for_crosssectional_econometrics()`. Při svém základním použití bere tato funkce na vstupu upravovanou tabulku, jméno sloupce, který indikuje, které stanice v daném čase opravdu existují, a výčet proměnných, které mají být v datech zachovány (a které nesmějí obsahovat chybějící hodnoty). Nejjednodušší způsob, jak předat seznam těchto proměnných, je pomocí odhadované formule.

```
obdobi <- "2014-01-01"
big_brands <- natural_monthly %>%
    filter(from_date == obdobi) %>%
    find_big_brands_in_prepared_data() %>%
    pull(name)
natural_monthly <- natural_monthly %>%
    mutate(price = price * 100) %>%
    add_big_brand_name_effects(big_brands) %>%
    add_city_effects()
```

```
cs_data <- natural_monthly %>%
    filter(from_date == obdobi) %>%
    prepare_for_crosssectional_econometrics(existence_criterium, basic_model)
```

7.2.2 Odhad OLS modelu

Software podporuje odhad OLS modelu na průřezových datech pomocí standardní funkce `lm()`. Model tedy odhadneme pomocí této funkce:

```
estimate_ols <- lm(basic_model, data = cs_data)
```

Výsledek odhadu vypíšeme pomocí funkce `summary()`:

```
summary(estimate_ols)
```

```
##
## Call:
## lm(formula = basic_model, data = cs_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -287.90  -38.63   -2.39   34.48  536.29
##
## Coefficients:
##                                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)                           3557.60401    4.60394 772.730  < 2e-16
## sc                                      46.53040   13.85748   3.358 0.000798
## distance_to_closest_competitor           1.43898    0.54114   2.659 0.007887
## other_stations_in_driving_distance_0_3  -3.64268    0.81664  -4.461 8.57e-06
## other_stations_in_driving_distance_3_6  -1.48520    0.39501  -3.760 0.000174
## osm_highwayTRUE                         85.62073    5.40725  15.834  < 2e-16
## prahaTRUE                               97.51383    8.48662  11.490  < 2e-16
## brnoTRUE                                40.10239   10.23818   3.917 9.23e-05
## ostravaTRUE                             38.92970   10.26644   3.792 0.000153
## mesto100TRUE                           19.48824    8.49264   2.295 0.021838
## mesto50TRUE                             15.56946    5.68805   2.737 0.006243
## mesto20TRUE                              5.97018    4.67782   1.276 0.201987
## brand_namebenzina                       41.51581    4.32692   9.595  < 2e-16
## brand_nameeurooil                        0.02528    5.20082   0.005 0.996123
## brand_nameomv                           79.53269    5.78642  13.745  < 2e-16
## brand_nameshell                         90.97130    6.04198  15.057  < 2e-16
## brand_nameagip                          86.60814    6.77840  12.777  < 2e-16
## brand_namepapoil                         1.78797    6.90937   0.259 0.795833
## brand_namerobin oil                    -15.26323    8.18036  -1.866 0.062190
## brand_namelukoil                       -29.14562   10.25940  -2.841 0.004538
```

```
## brand_namemol                              6.08489   10.69719   0.569 0.569526
## brand_namekm prona                        -9.78888   11.72625  -0.835 0.403926
## brand_nametank ono                      -127.99362   12.03358 -10.636  < 2e-16
## brand_nameunicorn                        -28.45854   13.50512  -2.107 0.035204
## brand_namesilmet                          20.63810   13.76356   1.499 0.133887
## brand_nameahold - albert                  44.93166   14.20154   3.164 0.001577
## brand_nametesco                           53.62313   15.88979   3.375 0.000751
## brand_nameglobus                         -70.45281   17.11840  -4.116 4.00e-05
## brand_namepasoil                          -4.07437   18.20302  -0.224 0.822910
## brand_nameeurobit                        -30.42978   17.75353  -1.714 0.086660
## brand_namehunsgas                         27.37185   21.82633   1.254 0.209941
## brand_namekontakt                          8.10408   17.59181   0.461 0.645076
## brand_nameprim                           -91.41843   17.64451  -5.181 2.40e-07
## brand_namevena trade                    -102.37257   18.93068  -5.408 7.04e-08
## brand_namečsad                            -6.75445   21.83399  -0.309 0.757080
## brand_namemakro                          -78.92220   19.09428  -4.133 3.70e-05
## brand_namepetrol                          17.49126   18.91855   0.925 0.355293
## brand_namearmex oil                       37.07816   21.85423   1.697 0.089904
## nonstopTRUE                               14.06218    3.50783   4.009 6.30e-05
##
## (Intercept)                             ***
## sc                                      ***
## distance_to_closest_competitor          **
## other_stations_in_driving_distance_0_3  ***
## other_stations_in_driving_distance_3_6  ***
## osm_highwayTRUE                         ***
## prahaTRUE                               ***
## brnoTRUE                                ***
## ostravaTRUE                             ***
## mesto100TRUE                            *
## mesto50TRUE                             **
## mesto20TRUE
## brand_namebenzina                       ***
## brand_nameeurooil
## brand_nameomv                           ***
## brand_nameshell                         ***
## brand_nameagip                          ***
## brand_namepapoil
## brand_namerobin oil                     .
## brand_namelukoil                        **
## brand_namemol
## brand_namekm prona
## brand_nametank ono                      ***
## brand_nameunicorn                       *
## brand_namesilmet
## brand_nameahold - albert                **
## brand_nametesco                         ***
## brand_nameglobus                        ***
```

```
## brand_namepasoil
## brand_nameeurobit                    .
## brand_namehunsgas
## brand_namekontakt
## brand_nameprim                       ***
## brand_namevena trade                 ***
## brand_namečsad
## brand_namemakro                      ***
## brand_namepetrol
## brand_namearmex oil                  .
## nonstopTRUE                          ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 64.99 on 2317 degrees of freedom
## Multiple R-squared:  0.489,  Adjusted R-squared:  0.4806
## F-statistic: 58.35 on 38 and 2317 DF,  p-value: < 2.2e-16
```

7.2.3 Odhad SAR modelu

Model SAR má na průřezových datech následující tvar:

$$y = \rho W y + \alpha \iota_n + X\beta + \varepsilon,$$

kde y je vysvětlovaná veličina, X je matice vysvětlujících veličin, α je úrovňová konstanta, β je vektor parametrů, ρ je parametr prostorové autokorelace, W je váhová matice a ε je vektor náhodné složky.

Pro odhad SAR modelu na průřezových datech musíme nejprve vytvořit váhovou matici W (objekt třídy *listw*), která popisuje strukturu prostorové autokorelace v datech. Uživatel může váhovou matici sestavit sám pomocí nástrojů, které nabízí balík **spdep**, nebo může využít specializovanou funkci `create_weights()` vytvořenou v rámci tohoto softwaru. V základním nastavení funkce vyžaduje tři vstupní parametry: data použitá v ekonometrii, určení, jaké vzdálenosti se mají použít k určení "sousedních" stanic (zde se používají vzdálenosti vzdušnou čarou) a maximální vzdálenost, do které se stanice považují za sousední. Funkce standardně uvažuje, že vliv sousedních stanic klesá se vzdáleností, tj. že váha je (před řádkovou normalizací) $1/d$, pokud je $d \leq$ max_dist a 0 jinak, kde d je vzdálenost mezi stanicí a jejím sousedem. Tuto transformační funkci je však možné změnit pomocí parametru `transform`. Další parametry funkce a možnosti jejího nastavení jsou popsané v její dokumentaci.

```
W <- create_weights(cs_data, neighbors = "beeline", max_dist = 20)
```

Následně odhadneme model pomocí funkce `lagsarlm2()`. Tato funkce je wrapper nad funkcí `lagsarlm()` z balíku **spatialreg**, který do objektu odhadu přidává proměnné potřebné pro tvorbu předpovědí, a tedy i simulaci fúze.

```
estimate_sar <- lagsarlm2(basic_model,
                          data = cs_data,
                          listw = W)
```

Výsledek odhadu můžeme opět vypsat pomocí funkce `summary()`:

```
summary(estimate_sar)
```

```
##
## Call:
## lagsarlm(formula = basic_model, data = cs_data, listw = W, na.action = na.omit)
##
## Residuals:
##         Min          1Q      Median          3Q         Max
## -279.073389  -32.398022   -0.046567   29.526511  555.132778
##
## Type: lag
## Coefficients: (asymptotic standard errors)
##                                        Estimate Std. Error  z value  Pr(>|z|)
## (Intercept)                          1377.33834   75.72439  18.1888 < 2.2e-16
## sc                                     25.79838   12.07016   2.1374 0.0325680
## distance_to_closest_competitor          1.07021    0.46967   2.2787 0.0226866
## other_stations_in_driving_distance_0_3 -3.13807    0.70910  -4.4255 9.624e-06
## other_stations_in_driving_distance_3_6 -1.35312    0.34272  -3.9482 7.873e-05
## osm_highwayTRUE                        60.45078    4.78380  12.6366 < 2.2e-16
## prahaTRUE                              52.51585    7.54893   6.9567 3.483e-12
## brnoTRUE                               19.14075    8.93740   2.1416 0.0322219
## ostravaTRUE                            22.77109    8.92770   2.5506 0.0107534
## mesto100TRUE                           19.96247    7.37015   2.7086 0.0067577
## mesto50TRUE                            15.97656    4.93807   3.2354 0.0012148
## mesto20TRUE                             8.06640    4.06086   1.9864 0.0469912
## brand_namebenzina                      42.72937    3.75442  11.3811 < 2.2e-16
## brand_nameeurooil                       1.67484    4.51210   0.3712 0.7104968
## brand_nameomv                          78.42018    5.02066  15.6195 < 2.2e-16
## brand_nameshell                        92.55647    5.24327  17.6524 < 2.2e-16
## brand_nameagip                         84.54219    5.88342  14.3696 < 2.2e-16
## brand_namepapoil                        9.22227    5.99440   1.5385 0.1239312
## brand_namerobin oil                    -0.10060    7.09917  -0.0142 0.9886942
## brand_namelukoil                      -20.16911    8.90080  -2.2660 0.0234521
## brand_namemol                           4.28354    9.28087   0.4615 0.6444072
## brand_namekm prona                     -1.15163   10.17342  -0.1132 0.9098721
## brand_nametank ono                   -109.87118   10.44156 -10.5225 < 2.2e-16
## brand_nameunicorn                     -26.00448   11.71669  -2.2194 0.0264568
## brand_namesilmet                       23.44710   11.94091   1.9636 0.0495772
## brand_nameahold - albert               30.36250   12.32970   2.4626 0.0137953
## brand_nametesco                        46.96569   13.78907   3.4060 0.0006592
## brand_nameglobus                      -67.12077   14.85181  -4.5194 6.202e-06
```

```
## brand_namepasoil                          10.04973   15.79293   0.6363 0.5245527
## brand_nameeurobit                        -29.82860   15.40278  -1.9366 0.0527977
## brand_namehunsgas                         32.54006   18.93889   1.7182 0.0857672
## brand_namekontakt                         27.23609   15.26359   1.7844 0.0743614
## brand_nameprim                           -98.20279   15.31009  -6.4143 1.415e-10
## brand_namevena trade                    -105.71430   16.42970  -6.4343 1.240e-10
## brand_namečsad                            -1.17344   18.94494  -0.0619 0.9506110
## brand_namemakro                          -73.82316   16.56570  -4.4564 8.335e-06
## brand_namepetrol                          12.77050   16.41640   0.7779 0.4366213
## brand_namearmex oil                       46.96729   18.96234   2.4769 0.0132539
## nonstopTRUE                                9.02742    3.04674   2.9630 0.0030468
##
## Rho: 0.6076, LR test value: 517.3, p-value: < 2.22e-16
## Asymptotic standard error: 0.021129
##     z-value: 28.756, p-value: < 2.22e-16
## Wald statistic: 826.93, p-value: < 2.22e-16
##
## Log likelihood: -12899.17 for lag model
## ML residual variance (sigma squared): 3179, (sigma: 56.383)
## Number of observations: 2356
## Number of parameters estimated: 41
## AIC: 25880, (AIC for lm: 26396)
## LM test for residual autocorrelation
## test value: 31.213, p-value: 2.3122e-08
```

Na rozdíl od modelu OLS, parametry SAR modelu nejsou mezními vlivy, a budou se tedy velmi lišit od odhadů parametrů OLS modelu, a to i v případě, že by modely dávaly velmi podobné predikce. Mezní vlivy spočítá (a rozdělí mezi přímé a nepřímé vlivy) funkce `impacts()`:

```
impacts(estimate_sar, listw = W)
```

7.3 Odhad modelů na panelových datech s fixními efekty na čas i stanice

Dále se podíváme na odhad stejných modelů na panelových datech. Odhad na panelových datech umožňuje kontrolovat o nepozorovanou nehomogenitu mezi jednotlivými obdobími a jednotlivými stanicemi. Tento oddíl ukáže, jak odhadnout model s fixními efekty na čas a individuální stanice. Kompletní ukázku použitého kódu obsahuje první část skriptu `example_panel_estimation_and_merger_simulations.R`. V následujícím oddíle se podíváme na odhad modelu, který obsahuje fixní efekty pouze na čas; místo fixních efektů na individuální stanice tam stejně jako na průřezových datech zavedeme fixní efekty na velké značky, velikost municipality a přítomnost dálnice.

7.3.1 Příprava dat pro odhad

Před vlastním odhadem modelu je opět potřeba data připravit. Obecnou přípravu dat popisuje kapitola 5. Řekněme tedy, že máme data o cenách benzínu E5 v ČR v letech 2013 až 2018 agregovaná

funkcí `prepare_average_fuel_data()` na měsíční bázi a že do dat jsou přidané následující statistiky lokální konkurence:

- *spatial clustering measure* SC,
- počet konkurenčních čerpacích stanic v dojezdových vzdálenostech 0–3, 3–6 a 6–9 km a
- vzdálenost k nejbližšímu konkurentovi v km.

Data dále pro každou stanici a měsíc obsahují údaj o tom, zda je čerpací stanice otevřená non-stop.

Budeme vysvětlovat cenu benzínu na jednotlivých čerpacích stanicích v závislosti na změně značky ve fúzi, výše uvedených měřítcích konkurence, údaji o otevření non-stop. Přitom zahrneme fixní efekty na čas a čerpací stanice. Odhadovaná rovnice tedy bude mít tvar:

```
basic_model <- price ~ brand_change +
    sc +
    other_stations_in_driving_distance_0_3 +
    other_stations_in_driving_distance_3_6 +
    other_stations_in_driving_distance_6_9 +
    distance_to_closest_competitor +
    nonstop
```

Nyní nám zbývá udělat s daty jen dvě drobné a jednu zásadní úpravu. Drobné změny jsou následující: 1) všechny ceny převedeme na haléře (tj. vynásobíme stem), aby odhadnuté parametry byly lépe čitelné, a 2) pomocí funkce `add_brand_changes_in_takeover()` přidáme do tabulky proměnnou `brand_change`, která indikuje změnu značky po fúzi, viz oddíl 5.6.3.

Poslední úprava je zásadnější. Model SAR budeme odhadovat pomocí funkcí z balíku **splm**. Ten umí v současné době odhadovat pouze modely na balancovaných panelech. Proto musíme z panelu vyloučit stanice, pro která nemáme kompletní pozorování. To zajistí funkce `prepare_for_spatial_panel_econometrics()`, která vyžaduje zejména tři vstupní parametry: agregovaná data, jméno sloupce, který indikuje, které stanice v daném čase opravdu existují, a výčet proměnných, které mají být v datech zachovány (a které tedy nesmějí obsahovat chybějící hodnoty). Nejjednodušší způsob, jak předat seznam potřebných proměnných, je pomocí odhadované formule.

```
natural_monthly <- natural_monthly %>%
    mutate(price = price * 100) %>%
    add_brand_changes_in_takeover(takeovers)
panel_data <- natural_monthly %>%
    filter(from_date >= "2013-01-01", from_date <= "2018-12-01") %>%
    prepare_for_spatial_panel_econometrics(existence_criterium, basic_model)
```

Tento přístup k tvorbě balancovaného panelu není zcela bezproblémový: jednak může být odhad prostorové autokorelace mírně vychýlený, jednak nebudeme schopní posoudit vliv fúze na vynechané stanice. Přesto je však takový odhad zřejmě lepší než úplné vynechání prostorového aspektu z dat.

7.3.2 Odhad modelu fixních vlivů bez prostorové autokorelace závislé veličiny

Software podporuje simulaci modelů fixních vlivů bez prostorové složky odhadnutou pomocí funkce `feols()` z balíku **fixest**. Detaily použití této funkce najdete v její dokumentaci.

Nejdříve upravíme odhadovanou rovnici tak, že do ní přidáme fixní efekty na čas a čerpací stanice:

```
update_fe <- function(m) update(as.Formula(m), . ~ . | id + from_date)
basic_model_fe <- update_fe(basic_model)
```

Následně můžeme odhadnout model. Zde použijeme balancovaná data, i když to striktně není nutné, čistě pro srovnatelnost s odhadem SAR modelu:

```
estimate_fe <- feols(basic_model_fe, data = panel_data)
```

Odhad vypíšeme pomocí funkce `summary()`:

```
summary(estimate_fe)
```

```
## OLS estimation, Dep. Var.: price
## Observations: 112,608
## Fixed-effects: id: 1,564,  from_date: 72
## Standard-errors: Clustered (id)
##                                          Estimate Std. Error   t value
## brand_changeagip -> mol                 -2.050700   4.256900 -0.481729
## brand_changeavanti -> benzina          -41.048000   5.397600 -7.604800
## brand_changelukoil -> mol               77.943000   7.490800 10.405000
## brand_changeomv -> benzina             -35.790000   4.975200 -7.193700
## sc                                      32.619000   9.138800  3.569400
## other_stations_in_driving_distance_0_3  -3.245900   1.089300 -2.979700
## other_stations_in_driving_distance_3_6  -2.895700   0.729034 -3.972000
## other_stations_in_driving_distance_6_9  -0.714944   0.665755 -1.073900
## distance_to_closest_competitor           0.349666   0.621646  0.562484
## nonstopTRUE                              5.934500   3.202500  1.853100
##                                         Pr(>|t|)
## brand_changeagip -> mol                 0.630066
## brand_changeavanti -> benzina            4.9e-14 ***
## brand_changelukoil -> mol              < 2.2e-16 ***
## brand_changeomv -> benzina              9.74e-13 ***
## sc                                      0.000369 ***
## other_stations_in_driving_distance_0_3   0.00293 **
## other_stations_in_driving_distance_3_6   7.5e-05 ***
## other_stations_in_driving_distance_6_9   0.28304
## distance_to_closest_competitor          0.573867
## nonstopTRUE                             0.064058 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
## Log-likelihood: -605,432.97   Adj. R2: 0.97085
##                              R2-Within: 0.01969
```

7.3.3 Odhad panelového SAR modelu

Software podporuje při odhadu panelového SAR modelu odhadovou funkci `spml()` z balíku **splm**. Tato funkce z nějakého důvodu označuje parametr prostorové autokorelace λ. Odhad SAR modelu opět vyžaduje váhovou matici W (objekt třídy *listw*). Tu buď vytvoříme sami pomocí nástrojů z balíku **spdep**, nebo využijeme funkci `create_weights()`:

```
W <- create_weights(panel_data, neighbors = "beeline", max_dist = 20)
```

Vlastní odhad provedeme pomocí funkce `spml2()`, což je wrapper nad funkcí `spml()` z balíku **splm**, který do objektu odhadu přidává některé údaje potřebné pro predikci modelu, a tedy i simulaci fúze. Parametry tohoto wrapperu jsou s jednou výjimkou popsanou níže stejné jako parametry funkce `spml()` a jsou tedy popsány v její dokumentaci.

```
estimate_sar <- spml2(basic_model,
                      data = panel_data,
                      index = c("id", "from_date"),
                      listw = W,
                      model = "within",
                      effect = "twoways",
                      lag = TRUE,
                      spatial.error = "none")
```

Výsledný odhad modelu je opět možné vypsat pomocí funkce `summary()`:

```
summary(estimate_sar)
```

```
## Spatial panel fixed effects lag model
##
##
## Call:
## splm::spml(formula = formula, data = data, index = index, listw = listw,
##     listw2 = listw2, na.action = na.action, model = model, effect = effect,
##     lag = lag, spatial.error = spatial.error)
##
## Residuals:
##        Min.     1st Qu.      Median     3rd Qu.        Max.
## -1.0359e+03 -2.3044e+01 -3.5251e-02  2.3078e+01  6.4747e+02
##
## Spatial autoregressive coefficient:
##         Estimate Std. Error t-value  Pr(>|t|)
## lambda 0.6410805  0.0032507  197.21 < 2.2e-16 ***
```

```
##
## Coefficients:
##                                          Estimate Std. Error  t-value  Pr(>|t|)
## brand_changeagip -> mol                  -1.99765    1.13545  -1.7594  0.078518
## brand_changeavanti -> benzina           -45.00055   10.70304  -4.2045 2.617e-05
## brand_changelukoil -> mol                71.14092    1.91754  37.1000 < 2.2e-16
## brand_changeomv -> benzina              -30.89497    1.58729 -19.4640 < 2.2e-16
## sc                                       17.73273    3.26918   5.4242 5.821e-08
## other_stations_in_driving_distance_0_3   -2.41363    0.38551  -6.2609 3.827e-10
## other_stations_in_driving_distance_3_6   -1.85900    0.22615  -8.2201 < 2.2e-16
## other_stations_in_driving_distance_6_9   -0.65838    0.21977  -2.9957  0.002738
## distance_to_closest_competitor            0.43041    0.21267   2.0239  0.042984
## nonstopTRUE                               4.06862    0.84977   4.7879 1.685e-06
##
## brand_changeagip -> mol                 .
## brand_changeavanti -> benzina           ***
## brand_changelukoil -> mol               ***
## brand_changeomv -> benzina              ***
## sc                                      ***
## other_stations_in_driving_distance_0_3  ***
## other_stations_in_driving_distance_3_6  ***
## other_stations_in_driving_distance_6_9  **
## distance_to_closest_competitor          *
## nonstopTRUE                             ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

V SAR modelu nepředstavují parametry mezní vlivy jednotlivých proměnných. Vlivy těchto proměnných můžeme odhadnout a na přímé a nepřímé vlivy rozložit pomocí funkce `impacts()`. (Pozor: Výpočet vlivů vyžaduje velké množství paměti a trvá poměrně dlouho.)

```
imp <- impacts(estimate_sar, listw = W, time = 72, R = 1000)
```

Výsledek pak můžeme vypsat buď pomocí funkce `summary()`, nebo kompaktněji pomocí funkce `present_impacts()` implementované v tomto softwaru:

```
present_impacts(imp)
```

```
##                                 parameter              direct
## 1                 brand_changeagip -> mol     -2.199* (1.226)
## 2           brand_changeavanti -> benzina -49.350*** (11.552)
## 3               brand_changelukoil -> mol   77.155*** (2.062)
## 4             brand_changeomv -> benzina  -33.433*** (1.655)
## 5                                      sc   19.145*** (3.475)
## 6  other_stations_in_driving_distance_0_3   -2.609*** (0.426)
## 7  other_stations_in_driving_distance_3_6   -2.011*** (0.237)
## 8  other_stations_in_driving_distance_6_9   -0.696*** (0.233)
```

```
## 9          distance_to_closest_competitor     0.467** (0.238)
## 10                            nonstopTRUE    4.426*** (0.923)
##                indirect                total
## 1       -3.458* (1.930)      -5.66* (3.156)
## 2   -77.579*** (18.150) -126.93*** (29.690)
## 3    121.297*** (3.681)   198.45*** (5.617)
## 4    -52.559*** (2.686)   -85.99*** (4.308)
## 5     30.098*** (5.477)    49.24*** (8.946)
## 6     -4.102*** (0.672)    -6.71*** (1.098)
## 7     -3.161*** (0.375)    -5.17*** (0.611)
## 8     -1.095*** (0.368)    -1.79*** (0.601)
## 9       0.734** (0.374)      1.20** (0.612)
## 10     6.959*** (1.459)    11.38*** (2.381)
```

7.4 Odhad modelů na panelových datech s fixními efekty na čas a velké značky

Někdy může být potřeba zahrnout do modelu odhad rozdílů v cenových hladinách mezi jednotlivými značkami (důvod je blíže vysvětlen v kapitole 8). V takovém případě však typicky nemůžeme do modelu zahrnout fixní efekty na jednotlivé stanice. Příklad takového odhadu ukazuje první část skriptu `R/example_panel_estimation_and_merger_simulations_time_fixed_effects_only.R`.

V takovém případě vysvětlujeme cenu na základě měřítek konkurence a fixních efektů pro změny jmen ve fúzi, jmen velkých značek, velikosti municipalit, přítomnosti dálnice a případně dalších vlastností stanic (zde, zda má daná stanice otevřeno nonstop). Odhadujeme tedy rovnici

```
basic_model <- price ~ brand_change + brand_name +
    sc +
    other_stations_in_driving_distance_0_3 +
    other_stations_in_driving_distance_3_6 +
    other_stations_in_driving_distance_6_9 +
    distance_to_closest_competitor +
    osm_highway +
    praha + brno + ostrava + mesto100 + mesto50 + mesto20 +
    nonstop
```

Data připravíme stejně jako v předchozím případě:

```
panel_data <- natural_monthly %>%
    filter(from_date >= "2013-01-01", from_date <= "2018-12-01") %>%
    prepare_for_spatial_panel_econometrics(existence_criterium, basic_model)
```

Model fixních vlivů pak odhadneme takto:

```
update_fe <- function(m) update(as.Formula(m), . ~ . | from_date)
basic_model_fe <- update_fe(basic_model)
estimate_fe2 <- feols(basic_model_fe, data = panel_data)
```

Při odhadu SAR modelu kontrolujeme pouze o fixní efekt období. Parametr `index` tak bude mít hodnotu pouze `"from_date"`. Pro simulaci fúze však v datech musí být zachován i sloupec, který identifikuje id jednotlivých stanic. Funkce `spml2()` pak vyžaduje zadání jména tohoto sloupce pomocí parametru `index_id`. Zároveň je třeba změnit parametr `effect` na hodnotu `"time"`:

```
W <- create_weights(panel_data, neighbors = "beeline", max_dist = 20)
estimate_sar2 <- spml2(basic_model,
                       data = panel_data,
                       index = c("from_date"),
                       listw = W,
                       model = "within",
                       effect = "time",
                       lag = TRUE,
                       spatial.error = "none",
                       index_id = "id")
```

Výsledný odhad modelu je opět možné vypsat pomocí funkce `summary()`:

```
summary(estimate_sar2)
```

```
## Spatial panel fixed effects lag model
##
##
## Call:
## splm::spml(formula = formula, data = data, index = index, listw = listw,
##     listw2 = listw2, na.action = na.action, model = model, effect = effect,
##     lag = lag, spatial.error = spatial.error)
##
## Residuals:
##       Min.    1st Qu.     Median    3rd Qu.       Max.
## -1040.3604   -33.6844    -1.3071    30.6427   731.7016
##
## Spatial autoregressive coefficient:
##         Estimate Std. Error t-value  Pr(>|t|)
## lambda 0.5958928  0.0028503  209.06 < 2.2e-16 ***
##
## Coefficients:
##                                          Estimate  Std. Error  t-value
## brand_changeagip -> mol                 62.726487    1.603831  39.1104
## brand_changeavanti -> benzina          -63.848482   10.339029  -6.1755
## brand_changelukoil -> mol               24.707242    1.828492  13.5124
## brand_changeomv -> benzina               1.725859    1.602542   1.0770
## brand_namea+s                          -45.956522    7.202990  -6.3802
## brand_nameagip                          95.171058    1.221356  77.9224
## brand_nameahold - albert                 0.139373    1.772336   0.0786
## brand_namearmex oil                      2.015193    5.110505   0.3943
## brand_nameavanti                        53.082156    6.852858   7.7460
```

```
## brand_namebenzina                          47.754182    0.601227  79.4278
## brand_namečsad                             33.966699    3.890829   8.7299
## brand_nameeurobit                         -44.047010    2.123954 -20.7382
## brand_nameeurooil                          12.097430    0.740114  16.3454
## brand_namefree 1                           56.000829   13.991505   4.0025
## brand_nameglobus                          -52.993304    1.923334 -27.5528
## brand_nameherst                           -56.100402    3.338372 -16.8047
## brand_namehunsgas                          -3.234383    5.224699  -0.6191
## brand_namekm prona                         22.415861    1.497066  14.9732
## brand_namekontakt                          39.597433    2.356937  16.8004
## brand_namelukoil                          -16.994208    2.264261  -7.5054
## brand_namemakro                           -49.281775    2.135950 -23.0725
## brand_namemol                              30.869664    1.350126  22.8643
## brand_nameoktan                            -3.109718    3.334079  -0.9327
## brand_nameomv                              91.747735    0.812749 112.8857
## brand_namepapoil                           20.359845    1.036019  19.6520
## brand_namepasoil                           17.888415    4.472951   3.9992
## brand_namepetrol                          -21.010211    2.862867  -7.3389
## brand_nameprim                            -71.918411    1.675119 -42.9333
## brand_namerobin oil                        10.192207    1.051993   9.6885
## brand_nameshell                           106.068757    0.756171 140.2708
## brand_namesilmet                           43.302044    1.915179  22.6099
## brand_nametank ono                       -120.437771    1.345084 -89.5392
## brand_nametesco                           -44.553214    1.815336 -24.5427
## brand_nameunicorn                         -24.462687    1.593825 -15.3484
## brand_namevena trade                      -83.426951    2.356985 -35.3956
## sc                                         20.607236    1.878808  10.9683
## other_stations_in_driving_distance_0_3     -0.887641    0.105999  -8.3740
## other_stations_in_driving_distance_3_6     -0.344069    0.056643  -6.0744
## other_stations_in_driving_distance_6_9     -0.492045    0.046123 -10.6681
## distance_to_closest_competitor              2.751546    0.087659  31.3894
## osm_highwayTRUE                            48.123227    0.700220  68.7259
## prahaTRUE                                  30.605388    1.241181  24.6583
## brnoTRUE                                   25.941995    1.356921  19.1183
## ostravaTRUE                                 5.061095    1.268051   3.9912
## mesto100TRUE                               11.424713    1.044068  10.9425
## mesto50TRUE                                 6.151059    0.717915   8.5680
## mesto20TRUE                                 3.464265    0.622698   5.5633
## nonstopTRUE                                10.540937    0.453026  23.2679
##                                           Pr(>|t|)
## brand_changeagip -> mol                  < 2.2e-16 ***
## brand_changeavanti -> benzina            6.596e-10 ***
## brand_changelukoil -> mol                < 2.2e-16 ***
## brand_changeomv -> benzina                  0.2815
## brand_namea+s                            1.769e-10 ***
## brand_nameagip                           < 2.2e-16 ***
## brand_nameahold - albert                    0.9373
## brand_namearmex oil                         0.6933
```

```
## brand_nameavanti                          9.484e-15 ***
## brand_namebenzina                         < 2.2e-16 ***
## brand_namečsad                            < 2.2e-16 ***
## brand_nameeurobit                         < 2.2e-16 ***
## brand_nameeurooil                         < 2.2e-16 ***
## brand_namefree 1                          6.268e-05 ***
## brand_nameglobus                          < 2.2e-16 ***
## brand_nameherst                           < 2.2e-16 ***
## brand_namehunsgas                            0.5359
## brand_namekm prona                        < 2.2e-16 ***
## brand_namekontakt                         < 2.2e-16 ***
## brand_namelukoil                          6.124e-14 ***
## brand_namemakro                           < 2.2e-16 ***
## brand_namemol                             < 2.2e-16 ***
## brand_nameoktan                              0.3510
## brand_nameomv                             < 2.2e-16 ***
## brand_namepapoil                          < 2.2e-16 ***
## brand_namepasoil                          6.355e-05 ***
## brand_namepetrol                          2.154e-13 ***
## brand_nameprim                            < 2.2e-16 ***
## brand_namerobin oil                       < 2.2e-16 ***
## brand_nameshell                           < 2.2e-16 ***
## brand_namesilmet                          < 2.2e-16 ***
## brand_nametank ono                        < 2.2e-16 ***
## brand_nametesco                           < 2.2e-16 ***
## brand_nameunicorn                         < 2.2e-16 ***
## brand_namevena trade                      < 2.2e-16 ***
## sc                                        < 2.2e-16 ***
## other_stations_in_driving_distance_0_3 < 2.2e-16 ***
## other_stations_in_driving_distance_3_6 1.245e-09 ***
## other_stations_in_driving_distance_6_9 < 2.2e-16 ***
## distance_to_closest_competitor            < 2.2e-16 ***
## osm_highwayTRUE                           < 2.2e-16 ***
## prahaTRUE                                 < 2.2e-16 ***
## brnoTRUE                                  < 2.2e-16 ***
## ostravaTRUE                               6.573e-05 ***
## mesto100TRUE                              < 2.2e-16 ***
## mesto50TRUE                               < 2.2e-16 ***
## mesto20TRUE                               2.647e-08 ***
## nonstopTRUE                               < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

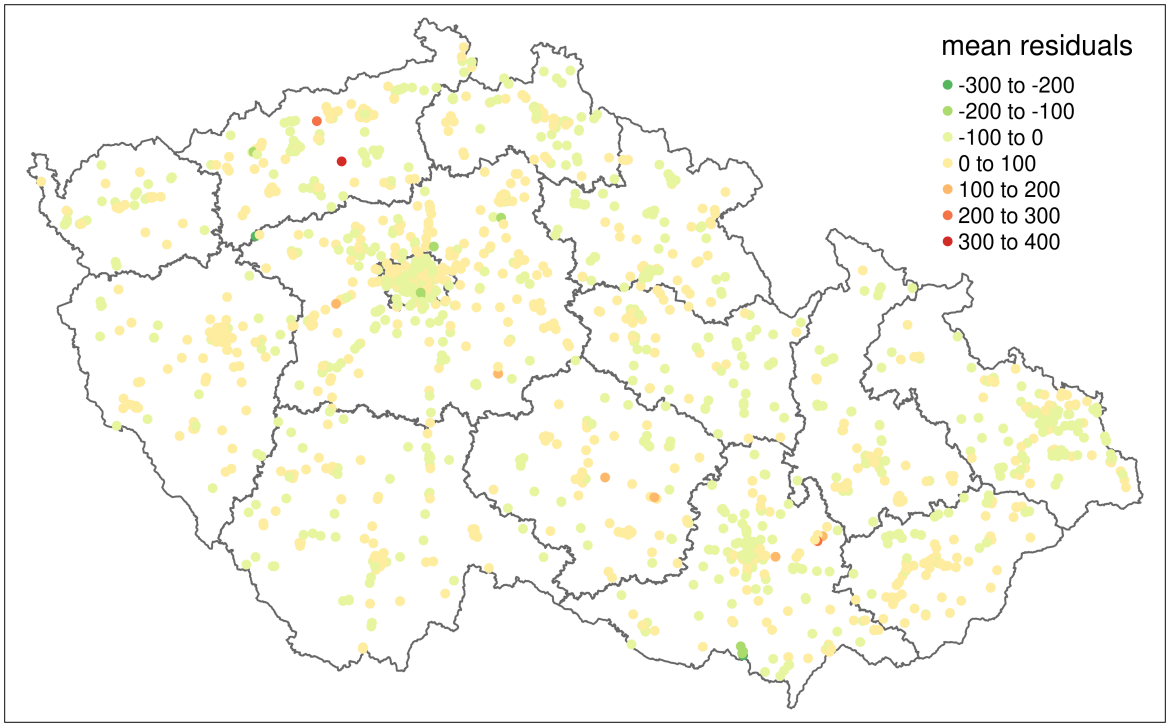
7.5 Vizualizace reziduí odhadnutých modelů na mapě

Někdy může být užitečné podívat se na rezidua modelu a v případě prostorových dat vykreslit tato rezidua do mapy. K tomu slouží funkce `plot_station_residuals()`,

která vykresluje rezidua na jednotlivých stanicích, a funkce `plot_average_residuals()`, která vykresluje jejich průměry za jednotlivé regiony. Detailní popis použití těchto funkcí je v jejich dokumentaci. Příklady použití najdete pro průřezová data ve skriptu `example_crosssectional_estimation_and_merger_simulations.R` a pro panelová data ve skriptu `example_panel_estimation_and_merger_simulations.R`. Zde proto uvádíme jen dva vybrané příklady:

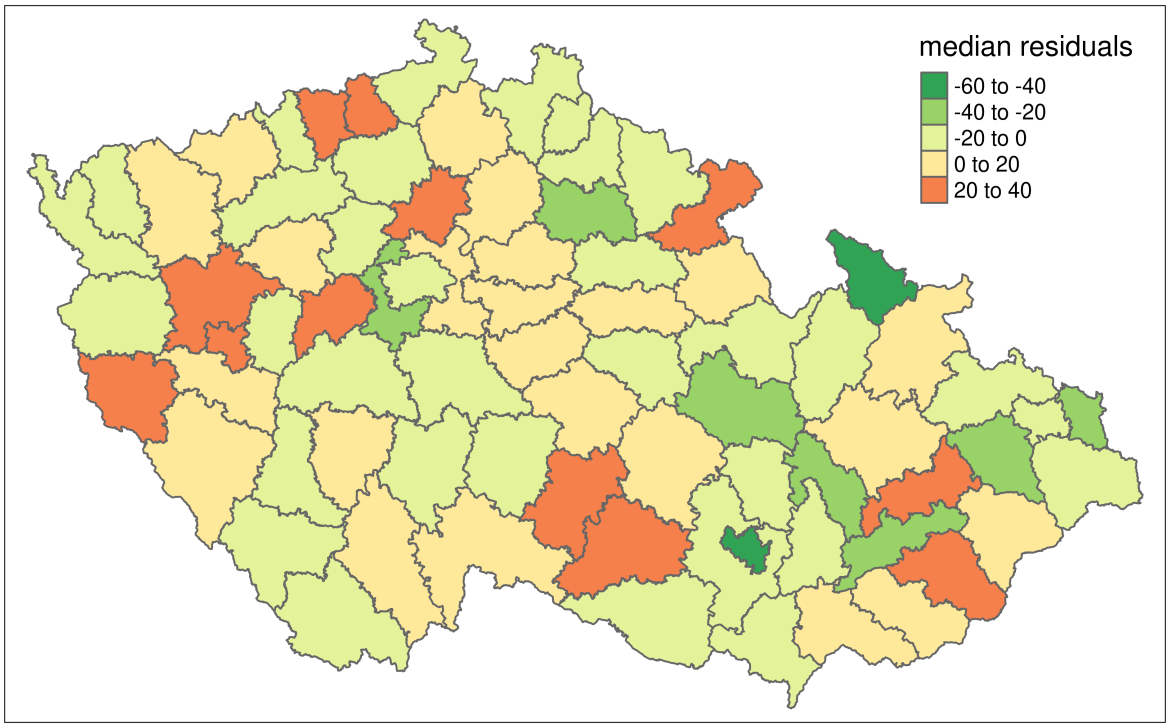
Rezidua pro jedno období vykreslená po jednotlivých stanicích:

```
panel_data %>%
    filter(from_date == "2018-01-01") %>%
    plot_station_residuals(estimate_sar,
                           natural_monthly,
                           arccr_folder = PATH_TO_ARCCR,
                           regions = "kraje",
                           size = 0.08)
```



Medián reziduí v jednotlivých okresech pro jedno období

```
panel_data %>%
    filter(from_date == "2018-01-01") %>%
    plot_average_residuals(estimate_sar,
                           arccr_folder = PATH_TO_ARCCR,
                           regions = "okresy",
                           stat = median)
```


median residuals
-60 to -40
-40 to -20
-20 to 0
0 to 20
20 to 40

8 Simulace fúze

Tato kapitola popisuje, jak simulovat fúzi značek stanic a jak vizualizovat výsledky této fúze. Funkce pro simulaci fúze jsou implementované ve skriptu `econometric_functions.R`, funkce pro vykreslení výsledků fúze na mapě jsou implementované ve skriptu `graphing_functions.R`. Praktické příklady provedení a vizualizace fúze ukazuje pro průřezová data skript `example_crosssectional_estimation_and_merger_simulations.R`. Pro panelové modely ukazují simulace fúze skripty `example_panel_estimation_and_merger_simulations.R` a `example_panel_estimation_and_merger_simulations_time_fixed_effects_only.R`.

8.1 Princip simulace fúze

Při fúzi se ceny změní ze dvou důvodů:

1. Změní se lokální konkurence. Pokud jedna firma převezme všechny stanice jiné firmy, lokální konkurence na všech stanicích klesne nebo zůstane stejná, a ceny tedy buď vzrostou nebo se nezmění. Naproti tomu, pokud jedna firma koupí pouze vybrané stanice jiné firmy a ta stále zůstane na trhu, pak může lokální konkurence na některých stanicích vzrůst, na jiných klesnout a na jiných se nezmění. Ceny se pak mohou na jednotlivých stanicích měnit libovolným směrem.
2. Různé značky mají různé cenové hladiny, které jsou dané reputací, dodatečnými službami, modifikací produktu apod. Pokud dražší značka koupí levnější značku a převede koupené stanice pod své jméno, dojde na těchto stanicích ke změně cen bez ohledu na změny lokální konkurence. (Podobně, když levnější značka koupí dražší značku.)

   (Cenové hladiny jednotlivých značek nejsou přímo pozorovatelné, protože ceny na jednotlivých stanicích nezávisí jen na cenové hladině značky, ale i na lokální konkurenci, přítomnosti levných nebo drahých stanic v okolí a mnoha idiosynkratických vlivech. Jediný způsob, jak je odhalit, je pomocí regrese.)

Celý problém je dál komplikován tím, že na trzích homogenních produktů s prostorovým aspektem typicky existuje silná prostorová autokorelace cen. V takové situaci se změna ceny v důsledku změny lokální konkurence nebo změny cenové hladiny v důsledku změny značky projeví nejen na přímo zasažených stanicích, ale rozlije se spojitě (s intenzitou klesající se vzdáleností) celou zemí. Z hlediska antimonopolního posuzování fúzí je samozřejmě otázka, zda při posuzování vlivu fúzí uvažovat vliv změny cenové hladiny na koupené stanice. Zřejmě je však důležité posoudit i vliv této změny na stanice v jejich okolí, kam se tato změna rozlije v důsledku prostorové autokorelace cen.

K plnému posouzení vlivu fúze je tedy nutné, abychom měli k dispozici ekonometrický model, který odhaduje 1) vliv lokální konkurence, 2) cenové hladiny jednotlivých značek nebo změnu cenové hladiny při přechodu stanic z jedné značky na druhou a 3) ideálně i odhad prostorové autokorelace cen (v případě modelu typu SAR).

Vlastní princip simulace fúze je pak velmi jednoduchý. Jeho základem je odhadnutý ekonometrický model a data pro jedno konkrétní období před fúzí. Na těchto datech předpovíme, jak by podle modelu vypadaly ceny na jednotlivých stanicích ve třech různých situacích:

1. Odhadneme, jak by ceny vypadaly bez fúze, tj. provedeme predikci cen pro dané období na základě skutečných dat a odhadnutého modelu (jedná se tedy o "vyrovnané hodnoty"). Tyto ceny (označíme je jako P^0) nejsou samy o sobě nijak zajímavé – slouží pouze pro porovnání s predikcemi po fúzi.
2. Upravíme vstupní data tak, že změníme vlastnictví stanic tak, jak bude vypadat po fúzi, a přepočítáme statistiky konkurence, jak by vypadaly po fúzi. Na těchto datech provedeme druhou predikci cen, tj. odhadneme, jak by ceny vypadaly po fúzi. Tyto ceny označíme jako P^F.
3. Vstupní data upravíme tak, že zkombinujeme hodnoty statistik konkurence, jak by vypadaly po fúzi, s úrovňovými konstantami vlastnictví (`brand_name` nebo `brand_change`), jak vypadaly před fúzí. S pomocí těchto dat získáme třetí predikci cen, P^M, která ukazuje, jak by ceny vypadaly, kdyby se změnila pouze lokální konkurence, ale ne cenové hladiny na jednotlivých stanicích.

Odhad cen P^M tedy zahrnuje pouze změny v lokální konkurenci bez změny vlastnictví značek; odhad cen P^F zahrnuje jak důsledky změn v lokální konkurenci, tak změn cenových hladin na jednotlivých stanicích v důsledku změny značky při fúzi.

S pomocí těchto hodnot spočítáme, jak by se ceny změnily v důsledku fúze:

- rozdíl $Z^M = P^M - P^0$ ukazuje, jak by fúze změnila ceny v důsledku změn lokální konkurence beze změn cenových hladin na jednotlivých stanicích a
- rozdíl $Z^F = P^F - P^0$ ukazuje, jak by fúze změnila ceny v důsledku změn lokální konkurence i změn cenových úrovní jednotlivých koupených stanic, které nastaly v důsledku změny značky.

Srovnáme vždy předpovědi s předpověďmi, ne se skutečnými hodnotami, protože tímto způsobem se zbavíme náhodné složky, tj. té části dat, kterou model neumí předpovědět.

Každý ze tří typů modelů popsaných v kapitole 7 v oddílech 7.2 až 7.4 umožňuje odhad cen P^M a jejich změn Z^M, protože všechny tyto modely odhadují vliv jednotlivých měřítek lokální konkurence. Modely odhadnuté na průřezových datech nebo na panelových datech bez fixních efektů na jednotlivé stanice umožňují získat i odhad cen P^F a jejich změn Z^F, protože mohou obsahovat efekty pro jména jednotlivých značek. Naproti tomu model odhadnutý na panelových datech s fixními efekty nejen na čas, ale i na jednotlivé stanice (viz oddíl 7.3) typicky nemůže obsahovat fixní efekty `brand_name` na jména jednotlivých značek, protože většina stanic má v čase stále stejné značky, takže by došlo k dokonalé multikolinearitě s fixními efekty na jednotlivé stanice. Takový model proto umožňuje získat odhad cen P^F jen ve speciální situaci, kdy 1) buď simulujeme efekt změny ex post nebo 2) v minulosti značka A koupila část stanic značky B a my nyní simulujeme, co by se stalo, kdyby značka A koupila zbytek stanic značky B, takže je změna cenové úrovně daná změnou značky zahrnutá do efektu `brand_change`.

8.2 Provedení simulace fúze

Vlastní simulaci fúze provede funkce `simulate_merger()` implementovaná ve skriptu `econometric_functions.R`. Tato funkce automaticky zaktualizuje data, provede potřebné predikce a spočítá hodnoty P^0, P^M, P^F, Z^M a Z^F. Aby to mohla provést, potřebuje funkce následující vstupy:

- použitý odhad modelu,
- data, na kterých byl model odhadnut,
- data, ze kterých jsou odhadová data výběrem (musí obsahovat všechny proměnné použité v odhadnutém modelu),

- výběr jednoho období (pouze pro panelová data),
- tabulku fúze,
- jméno sloupce, který v datech indikuje, které stanice v daném období skutečně existují,
- tabulku, která říká, které značky mají stejného majitele,
- cestu k adresáři `AdministrativniCleneni_v13.gdb` a
- tabulku dojezdových vzdáleností.

Výsledkem simulace fúze je tabulka třídy *tibble*, která obsahuje následující sloupce:

- `id` (třída *integer*) ... id stanice,
- `from_date`, `to_date` (třída *Date*) ... začátek a konec období, pro které jsou data simulovaná,
- `name` (třída *character*) ... skutečné značka stanice v daném období,
- `name_after_merger` (třída *character*) ... značka stanice po provedení fúze,
- `merged_station` (třída *logical*) ... `TRUE`, pokud byla stanice předmětem simulované fúze,
- `price_prediction_basic` (třída *double*) ... cena P^0, kterou použitý model predikuje za nezměněných okolnosti,
- `price_prediction_medium` (třída *double*) ... cena P^M, kterou použitý model predikuje v situaci, kdy by fúze změnila statistiky konkurence, ale ne značky firem (tj. cenové hladiny stanic),
- `price_prediction_full` (třída *double*) ... cena P^F, kterou použitý model predikuje v situaci, kdy by fúze změnila jak statistiky konkurence, tak značky firem (tj. cenové hladiny stanic),
- `price_change_medium` (třída *double*) ... změna ceny Z^M pouze v důsledku změn statistik konkurence, ale ne cenové hladiny stanic, tj. rozdíl mezi `price_prediction_medium` a `price_prediction_basic`, a
- `price_change_full` (třída *double*) ... plná změna ceny Z^F v důsledku fúze, tj. rozdíl mezi `price_prediction_full` a `price_prediction_basic`.

V případě simulace fúze na panelových datech s fixními efekty na jednotlivé stanice, kdy není možné odhadnout ceny P^F a změny cen Z^F, je možné funkci `simulate_merger()` požádat, aby tyto statistiky nepočítala. K tomu stačí nastavit parametr `full_impact` na hodnotu `FALSE`. Ve výsledné tabulce pak budou mít sloupce `price_prediction_full` a `price_change_full` hodnoty `NA`.

Poznámka: Funkce `simulate_merger()` používá k aktualizaci statistik konkurence funkci `update_competition_statistics()`. Tato funkce umí aktualizovat tři statistiky konkurence, které jsou v softwaru implementované: *spatial clustering measure* SC, počet konkurenčních (a případně vlastních) stanic ve vybraných dojezdových vzdálenostech a vzdálenost k nejbližšímu konkurentovi. Funkce `simulate_merger()` sama aktualizuje dvě proměnné obsahující jméno značky `brand_name` a změnu značky `brand_change`. Pokud si uživatel přidá jakékoli další měřítko konkurence nebo jinou proměnou, která se má při simulaci fúze aktualizovat, musí odpovídajícím způsobem upravit i funkce `simulate_merger()` a `update_competition_statistics()`.

Následující příklad ukazuje, jak provést ex post simulaci, kdy Mol koupil v roce 2014 Lukoil a v roce 2015 Agip, pomocí panelového modelu SAR s fixními efekty na čas a jednotlivé stanice. Fúze se simuluje v období před fúzí. Plný kód simulace obsahuje skript `example_panel_estimation_and_merger_simulations.R`.

```
period <- as.Date("2014-01-01")  # měsíc, ke kterému se simulace provádí
mol_merger_sar <- simulate_merger(
    estimate_sar,
    panel_data,
```

```
    natural_monthly,
    periods = period,
    takeovers = takeovers[1:2, ],
    existence = existence_criterium,
    same_group = same_owners_correction_table,
    arccr_folder = PATH_TO_ARCCR,
    driving_distances = station_driving_distances
)
```

Druhý příklad ukazuje, jak využít panelový model s fixními efekty na čas a dodatečnými efekty na vlastnictví stanice, velikost municipality a umístnění stanice na dálnici k simulaci budoucí hypotetické fúze.

Nejdříve opět zvolíme období, pro které fúzi simulovat. Dále vytvoříme tabulku fúze. V ní zadáme, že Benzina kupuje všechny stanice Eurooil a bude je nadále provozovat pod svou značkou. Skript `example_panel_estimation_and_merger_simulations_time_fixed_effects_only.R` obsahuje plný kód této simulace.

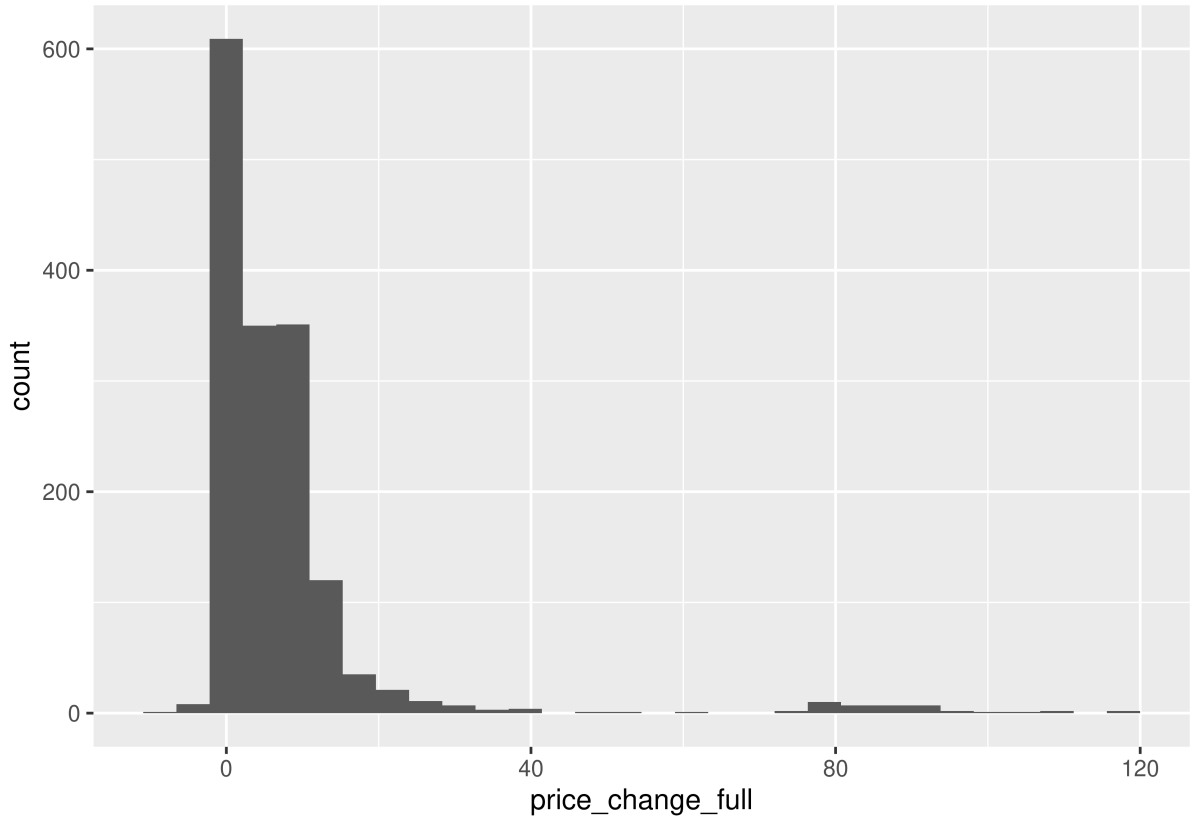
```
# simulace hypotetické fúze, ve které by Benzina koupila všechny stanice Eurooilu
# - období, ve kterém se fúze simuluje
fake_period <- as.Date("2018-01-01")
# - zjištění id čerpacích stanic Eurooil a tvorba tabulky fúzí
eurooil <- natural_monthly %>%
    filter(from_date == fake_period, name == "eurooil") %>%
    pull(id)
fakeovers <- tribble(
    ~takeover_date, ~terminal_date, ~source, ~target, ~ids,
    NA, NA, "eurooil", "benzina", ids = eurooil
)
# - simulace fúze
benzina_merger_sar <- simulate_merger(
    estimate_sar2,
    panel_data,
    natural_monthly,
    periods = fake_period,
    takeovers = fakeovers,
    existence = existence_criterium,
    same_group = same_owners_correction_table,
    arccr_folder = "map_shapes/AdministrativniCleneni_v13.gdb",
    driving_distances = station_driving_distances
)
```

Nakonec by nás mohla zajímat situace, kdy by Benzina převzala kontrolu nad všemi stanicemi Eurooil, ale nadále je provozovala pod značkou (a tedy s cenovou hladinou) Eurooil. V tomto případě by se tedy změnily jen měřítka konkurence, ale ne cenové hladiny jednotlivých čerpacích stanic. Tato situace je už zahrnutá v předchozí simulaci: jedná se o cenu P^M a změnu Z^M označené v tabulce fúze jako `price_prediction_medium` a `price_change_medium`.

8.3 Popisné statistiky fúze

Protože je predikce fúze uložená v tabulce, může uživatel predikci zobrazit libovolným způsobem, který mu vyhovuje. Může např. vykreslit histogram změn cen, o kterých model předpovídá, že k nim v důsledku fúze dojde:

```
library(ggplot2)
ggplot(mol_merger_sar, aes(price_change_full)) + geom_histogram()
```



Software navíc implementuje funkci `summarize_merger()`, která zobrazí základní popisné statistiky pro zvolenou statistiku fúze. Funkce bere tři základní parametry: tabulku s předpověďmi fúze vytvořená funkcí `simulate_merger()`, jména značek, které už před fúzí patřily firmě, která ve fúze nakupuje nové stanice, a vektor jmen sloupců z tabulky s předpověďmi fúze, jejichž statistiky se mají vypsat. Pokud je zvolená statistika jen jedna, je možné požádat o zjednodušení výpisu.

Následující kód vypíše průměrnou, minimální a maximální změnu ceny P^F rozdělenou do kategorií značek patřících firmě, která nakupuje (`former`), koupených značek (`purchased`), ostatních stanic (`other`) a všech stanic bez rozdílu (`all`). Sloupec `n` označuje počet stanic v každé kategorii. Sloupec `signif` indikuje statistickou významnost rozdílu střední hodnoty změny ceny v dané kategorii od nuly (s kódováním *** 0.01, ** 0.05 a * 0.1). Statistická významnost je počítaná bootstrapem.

Tabulka sumarizuje ex-post hodnocené důsledky fúze, kdy Mol koupil čerpací stanice Lukoil a Agip:

```
summarize_merger(mol_merger_sar, c("mol", "papoil", "slovnaft"),
                 "price_change_full", simplify = TRUE)
```

```
## # A tibble: 4 x 7
##   group         n  mean    min   max    sd signif
##   <fct>     <int> <dbl>  <dbl> <dbl> <dbl> <chr>
## 1 former       95  7.97 -0.370  46.8  8.49 ***
## 2 purchased   141 31.1  -8.75  118.  38.0  ***
## 3 other      1328  5.04 -1.54   63.2  5.84 ***
## 4 all        1564  7.57 -8.75  118.  14.8  ***
```

Je možné zadat i více statistik naráz:

```
summarize_merger(mol_merger_sar,  c("mol", "papoil", "slovnaft"),
                 c("price_change_full", "price_change_medium"))
```

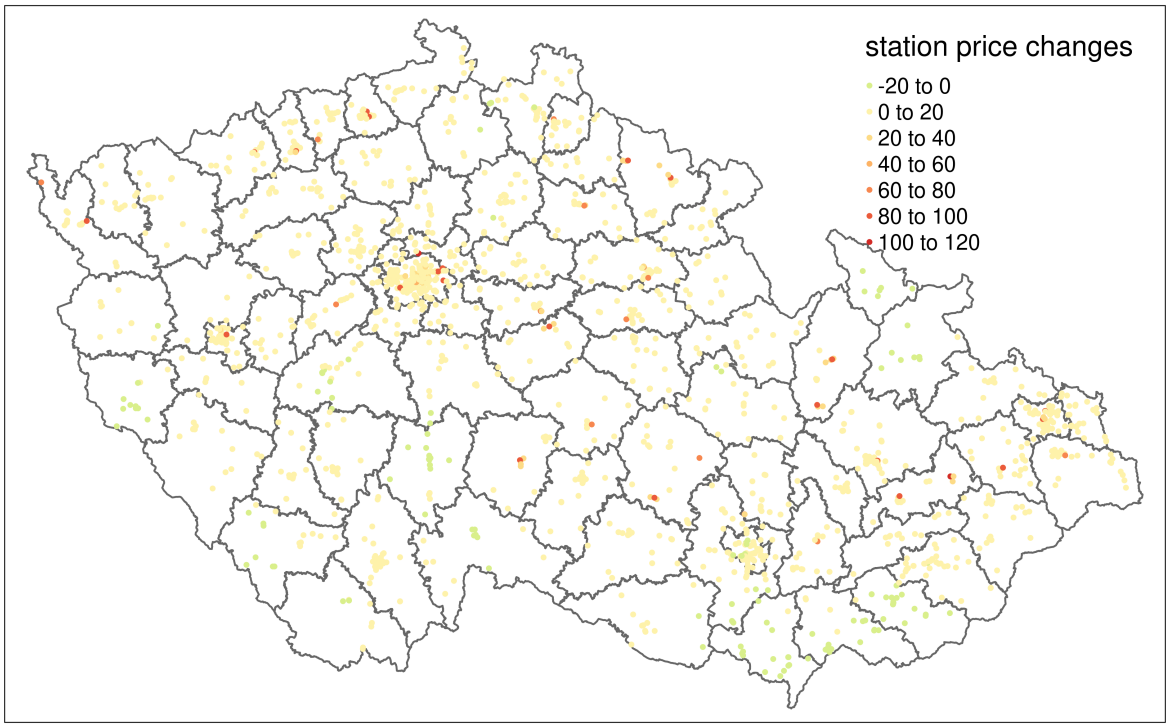
```
## # A tibble: 4 x 13
##   group price_change_fu~ price_change_fu~ price_change_fu~ price_change_fu~
##   <fct>            <int>            <dbl>            <dbl>            <dbl>
## 1 form~               95             7.97           -0.370             46.8
## 2 purc~              141            31.1            -8.75             118.
## 3 other             1328             5.04           -1.54              63.2
## 4 all               1564             7.57           -8.75             118.
## # ... with 8 more variables: price_change_full_sd <dbl>,
## #   price_change_full_signif <chr>, price_change_medium_n <int>,
## #   price_change_medium_mean <dbl>, price_change_medium_min <dbl>,
## #   price_change_medium_max <dbl>, price_change_medium_sd <dbl>,
## #   price_change_medium_signif <chr>
```

8.4 Vizualizace výsledků fúze na mapě

Software umožňuje vykreslovat předpovězené ceny a jejich změny do mapy. K tomu slouží funkce `plot_station_merger_prices()` a `plot_average_merger_prices()`. První z nich vykresluje ceny nebo změny cen na individuálních stanicích, druhá průměrné ceny a změny cen ve vybraných typech regionů. Základní vstupy funkcí jsou tabulka předpovězených cen po fúzi vytvořená funkcí `simulate_merger()`, původní data, jejichž podmnožina byla použita pro odhad modelu, cesta k adresáři `AdministrativniCleneni_v13.gdb`. Dále je možné pomocí parametru `var` zadat, která veličina se do grafu vykreslí (implicitně Z^F). Jaké regiony se mají vykreslit určuje parametr `regions` (při vykreslování průměrů v regionech se implicitně použijí okresy, při vykreslování cen na jednotlivých stanicích se implicitně regiony nevykreslují).

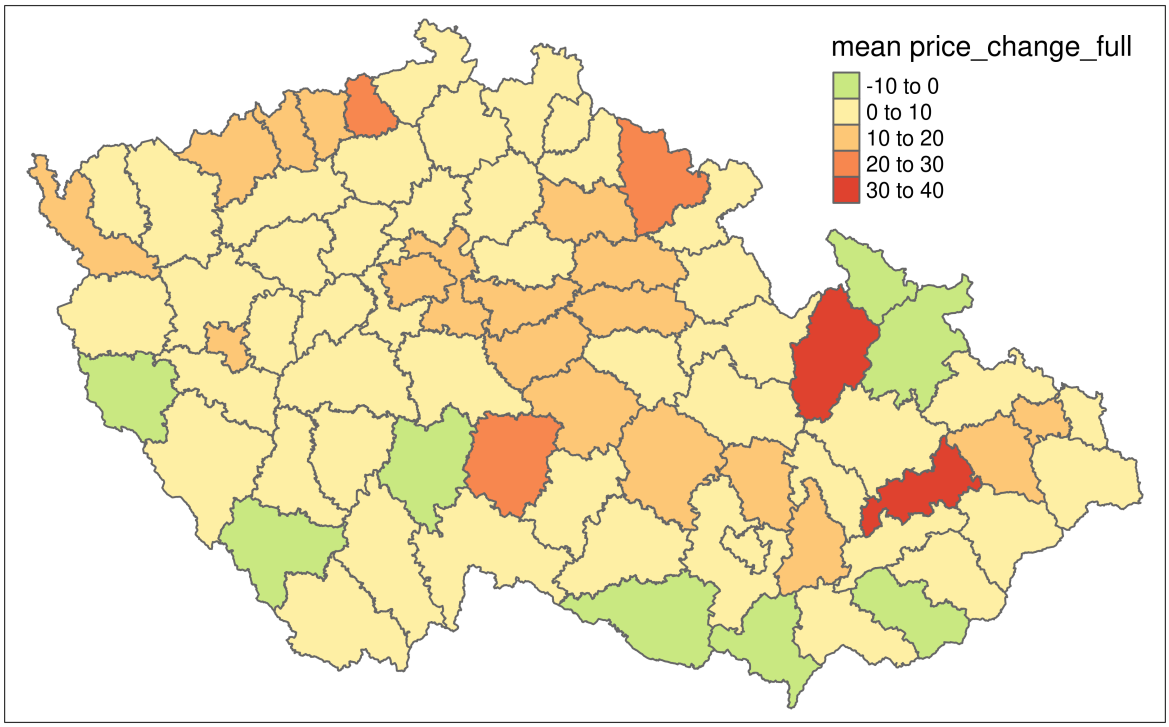
Příklad ukazuje, jak zobrazit plné změny cen Z^F v důsledku fúze, kdy Mol koupil Lukoil a Agip:

```
plot_station_merger_prices(mol_merger_sar, natural_monthly,
                           arccr_folder = PATH_TO_ARCCR,
                           regions = "okresy")
```



Následující příklad ukazuje, jak vykreslit průměrné změny cen Z^F v okresech v důsledku stejné fúze:

```
plot_average_merger_prices(mol_merger_sar, natural_monthly,
                           arccr_folder = PATH_TO_ARCCR,
                           regions = "okresy")
```



Další parametry funkcí jsou popsané v dokumentaci ve skriptu `graphing_functions.R`. Příklady použití najdete ve skriptech `example_panel_estimation_and_merger_simulations.R` (pro panelové odhady) a `example_crosssectional_estimation_and_merger_simulations.R` (pro pro průřezová data).

8.5 Simulace uvalení závazku prodeje vybraných stanic

V některých situacích může být schválení fúze podmíněno uvalením závazku. V tomto oddílu se podíváme nato, jak nasimulovat dopad uvalení závazku, který má formu povinnosti fúzujících značek odprodat vybrané stanice.

Vlastní simulace je velmi jednoduchá – stačí změnit tabulku fúzí tak, že vybrané stanice buď nebudou součástí fúze (a tedy budou i nadále operovat pod původní značkou), nebo budou prodány vybrané značce. Zde budeme pro jednoduchost uvažovat situaci, kdy čtyři stanice, na kterých by v důsledku fúze nejvíce stouply ceny, zůstanou i nadále pod samostatnou značkou. Použijeme k tomu hypotetický nákup Eurooilu Benzinou popsaný na konci oddílu 8.2.

Nejdříve najdeme čtyři stanice (zde všechny patří Eurooilu), kde by v důsledku fúze nejvíce stoupla cena:

```
benzina_merger_sar %>%
    filter(price_change_full > 35) %>%
    arrange(desc(price_change_full)) %>%
```

```
    select(id, name, merged_station, price_change_full, price_change_medium) %>%
    slice(1:4)
```

```
## # A tibble: 4 x 5
##      id name    merged_station price_change_full price_change_medium
##   <int> <chr>   <lgl>                      <dbl>                <dbl>
## 1   637 eurooil TRUE                        78.5                 35.4
## 2   634 eurooil TRUE                        77.5                 37.2
## 3   748 eurooil TRUE                        75.1                 31.5
## 4   655 eurooil TRUE                        70.2                 27.5
```

```
eurooil_exception <- benzina_merger_sar %>%
    arrange(desc(price_change_full)) %>%
    slice(1:4) %>%
    pull(id)
```

Vektor `eurooil_exception` obsahuje id těchto čtyř stanic. S jeho pomocí vytvoříme novou tabulku fúze tak, že v simulaci Benzina koupí všechny stanice Eurooil s výjimkou těchto čtyř vybraných stanic:

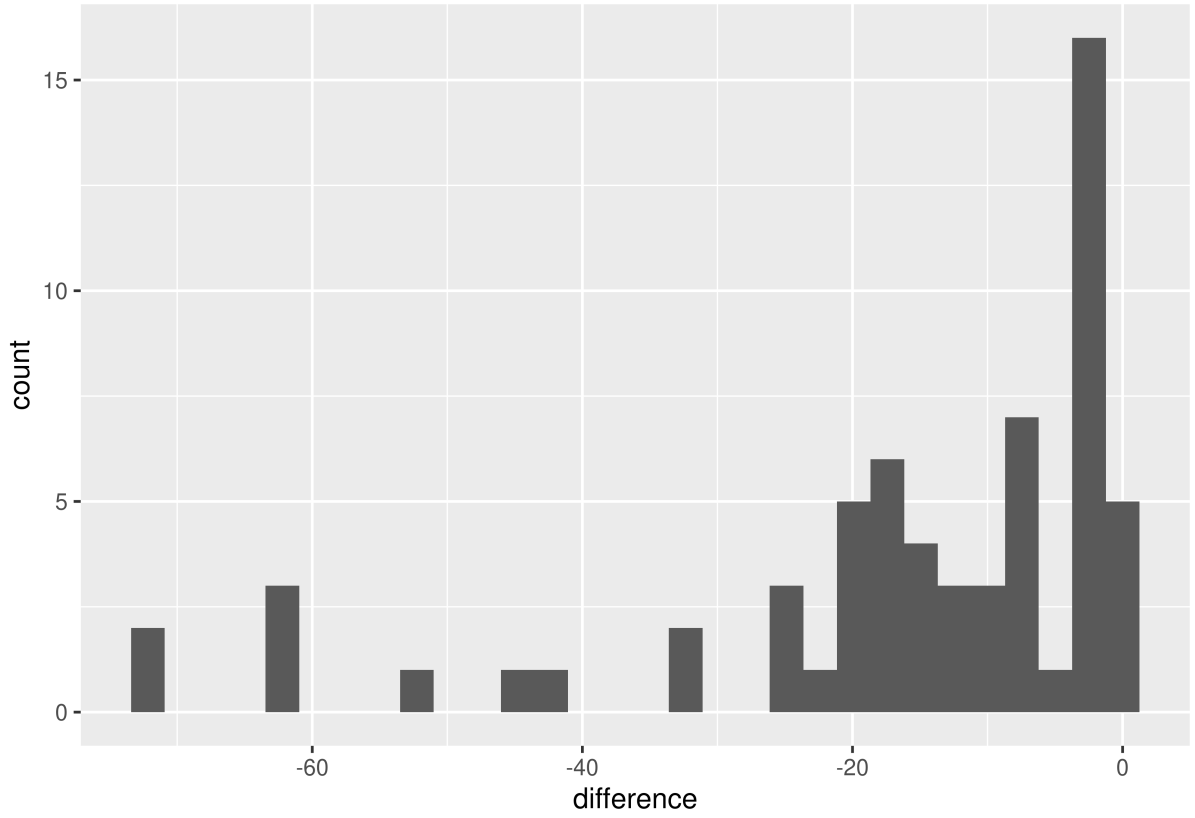
```
eurooil_taken_over <- setdiff(eurooil, eurooil_exception)
fakeover_exception <- tribble(
    ~takeover_date, ~terminal_date, ~source, ~target, ~ids,
    NA, NA, "eurooil", "benzina", ids = eurooil_taken_over
)
```

Nakonec nasimulujeme fúzi stejným způsobem jako v oddíle 8.2. Jediný rozdíl spočívá v použití odlišné tabulky fúze:

```
benzina_merger_sar_exceptions <- simulate_merger(
    estimate_sar2,
    panel_data,
    natural_monthly,
    periods = fake_period,
    takeovers = fakeover_exception,
    existence = existence_criterium,
    same_group = same_owners_correction_table,
    arccr_folder = "map_shapes/AdministrativniCleneni_v13.gdb",
    driving_distances = station_driving_distances
)
```

Důsledky uvalení závazku můžeme posoudit mnoha způsoby. Např. se můžeme podívat, jak se na jednotlivých stanicích změnily ceny v důsledku uvalení tohoto závazku. Jednou možností je vykreslení histogramu takových změn (aby byl rozdíl viditelný, ukazujeme histogram změn pouze pro stanice, kde se cen v důsledku uvalení závazku změnila více než o 1 halíř).

```
comp <- left_join(benzina_merger_sar %>%
                      select(id, name, price_chage_merger = price_change_full),
                  benzina_merger_sar_exceptions %>%
                      select(id, price_change_exceptions = price_change_full),
                  by = "id") %>%
    mutate(difference = price_change_exceptions - price_chage_merger) %>%
    arrange(difference)
ggplot(comp %>% filter(abs(difference) > 1), aes(difference)) + geom_histogram()
```



Je zřejmé, že se ceny změnily na více než jen vybraných stanicích. K tomu v našem případě došlo ze dvou důvodů: 1) závazek změnil sílu lokální konkurence v okolí jednotlivých stanic a 2) prostorová autokorelace rozlévá změnu do okolí zasažených stanic.

Samozřejmě se můžeme podívat i změnu na vybraných stanicích přímo (zde jen 10 největších změn). První čtyři jsou samozřejmě stanice vybrané jako závazek:

```
comp %>%
    select(-price_change_exceptions) %>%
    slice(1:10) %>%
    knitr::kable()
```

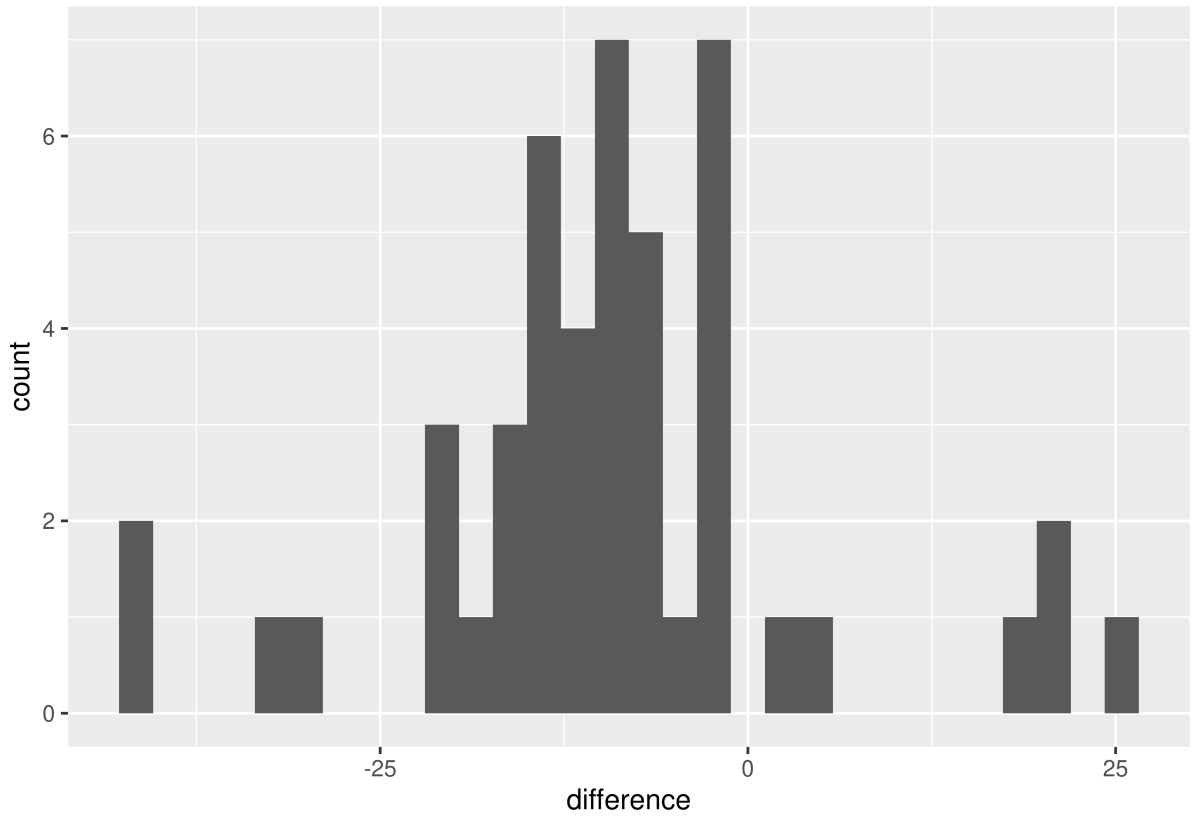
id	name	price_chage_merger	difference
637	eurooil	78.45287	-73.26967
748	eurooil	75.11324	-71.55515
655	eurooil	70.17449	-63.20682
634	eurooil	77.52208	-63.11587
1907	noprosu	64.27358	-61.04282
91	shell	52.45386	-51.48270
328	benzina	47.48417	-45.84306
883	robin oil	43.09959	-42.17660
926	benzina	41.92885	-32.44660
458	benzina	33.40134	-32.15451

Simulace závazku, kdy vybrané stanice převezme jiná značka, je obdobně jednoduchá: stačí nastavit příslušné převzetí v tabulce fúzí. Následující příklad ukazuje, co by se stalo, pokud by čtyři stanice, které by při fúzi s Benzinou zdražily nejvíc, místo Benziny koupilo OMV:

```
fakeover_exception2 <- tribble(
    ~takeover_date, ~terminal_date, ~source, ~target, ~ids,
    NA, NA, "eurooil", "benzina", eurooil_taken_over,
    NA, NA, "eurooil", "omv", eurooil_exception
)
benzina_merger_sar_exceptions2 <- simulate_merger(
    estimate_sar2,
    panel_data,
    natural_monthly,
    periods = fake_period,
    takeovers = fakeover_exception2,
    existence = existence_criterium,
    same_group = same_owners_correction_table,
    arccr_folder = "map_shapes/AdministrativniCleneni_v13.gdb",
    driving_distances = station_driving_distances
)
```

V tomto případě by některé stanice oproti fúzi s Benzinou zdražily, protože OMV je obecně dražší značka než Benzina, jak ukazují koeficienty `brand_namebenzina` a `brand_nameomv` v regresní tabulce. Rozdíl opět ukazuje histogram (kde byly opět vynechané stanice, jejichž cena se mezi simulací se závazkem a bez závazku neliší více než o 1 halíř):

```
comp2 <- left_join(benzina_merger_sar %>%
                       select(id, name, price_chage_merger = price_change_full),
                   benzina_merger_sar_exceptions2 %>%
                       select(id, price_change_exceptions = price_change_full),
                   by = "id") %>%
    mutate(difference = price_change_exceptions - price_chage_merger) %>%
    arrange(difference)
ggplot(comp2 %>% filter(abs(difference) > 1), aes(difference)) + geom_histogram()
```


count
6
4
2
0
-25
0
25
difference