# Zpracování dat pomocí OpenRefine

Miroslav Kubásek

*Institut biostatistiky a analýz, Masarykova univerzita, Brno; e-mail: kubasek@iba.muni.cz*

**Abstrakt**

Článek představuje nástroj OpenRefine, který je určený pro pokročilé zpracování a čištění dat. Čtenář bude schopný nástroj nainstalovat, importovat, exportovat data a provádět opravy a transformace dat, včetně importu informací z externích zdrojů.

**Klíčová slova**

OpenRefine, GREL, data mining, regex, fusion tables, tutorial

## 1. Úvod

Data jsou někdy označována jako novodobé zlato, zvláště v dnešní datově-orientované ekonomice, kdy okolo nás existuje nepředstavitelné množství dat, získávají data čím dál více na hodnotě. Zkusme o těchto cenných datech přemýšlet jako o diamantech. Zprvu je to velmi syrový, nezpracovaný materiál, mnohdy skrytý v ohromném množství bezcenné hlušiny. Ale pokud se nám podaří surový diamant najít, pohrajeme si s ním, vybrousíme ho, vyleštíme, najednou získá ohromně na ceně a to je to, o co nám jde. Přesně tak se můžeme dívat i na nástroj OpenRefine.[1] Umožňuje nám pokročilými způsoby importovat, transformovat a exportovat data v nejrůznějších formátech. Ukážeme si,[2] jak v datech opravit drobné chyby, změnit strukturu dat, až po to jak data rozšířit o informace z externích zdrojů a jednoduchým způsobem je vizualizovat.

### 1.1. OpenRefine v porovnání s dalšími nástroji

| OpenRefine | Tabulkový editor (Excel) | Databáze |
|---|---|---|
| **Je možné dávkově zpracovat jak řádky, tak i sloupce.** | V jeden okamžik lze editovat jen jednu buňku. | Je potřeba znát dotazovací jazyk (SQL) |
| **Umožňuje zkoumat a transformovat data.** | Vhodné pro vkládání dat a provádění výpočtů. | Pouze základní možnosti transformací. |
| **Není potřeba definovat schéma dat.** | Není potřeba definovat schéma dat. | Je potřeba definovat schéma zpracovávaných dat. |
| **Data jsou viditelná v každém kroku editace.** | Data jsou viditelná, omezený počet řádků. | Data jsou skryta, dokud je dotazem nevyexportujete. |
| **Mnohem více interaktivní a vizuální nástroj.** | Vizualizace dat je pouze základní. | Použití příkazového řádku pro spouštění dotazů. |

[1] http://openrefine.org/

[2] Všechny použité datové soubory použité v tomto textu včetně instalačních souborů jsou k dispozici na adrese https://github.com/MiroslavKubasek/OpenRefine-tutorial

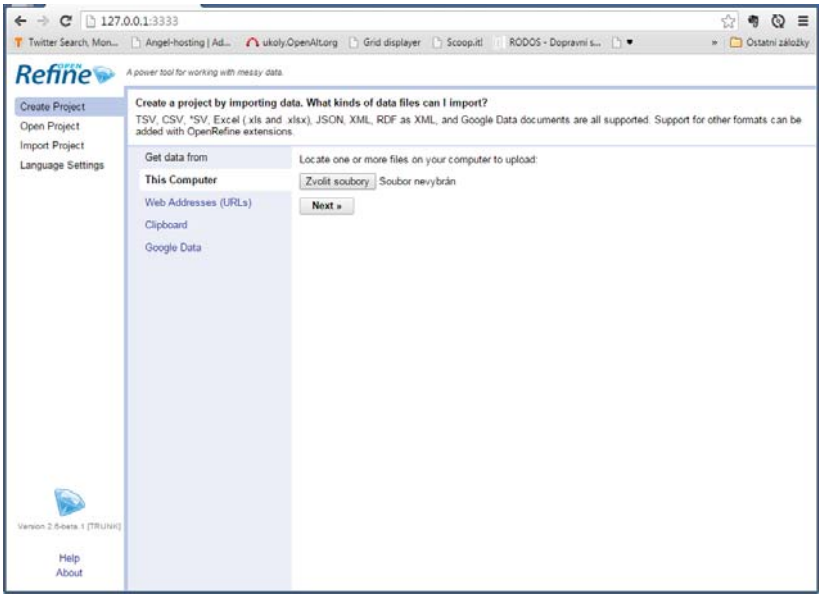
# 2. Základy OpenRefine

## 2.1. Instalace

OpenRefine je aplikace naprogramovaná v jazyce Java, proto se ujistěte předtím, než budete pokračovat dále, zdali máte na počítači nainstalované běhové prostředí tohoto jazyka (JRE[3]). V případě potřeby si JRE stáhněte a nainstalujte na základě instrukcí na adrese `http://java.com/en/download/`.

Stáhněte si s adresy `http://openrefine.org/download.html`[4] instalační soubor pro Váš operační systém. Soubory rozbalte do složky, do které bude mít OpenRefine přístup (ne do `C:/Windows/…`), může se jinak stát, že se vám nebudou ukládat projekty nebo nástroj nepůjde spustit.

Po rozbalení spusťte soubor **`refine.bat`** (v případě MS Windows), měl by vám vyskočit příkazový řádek a po chvíli automaticky otevřít nové okno ve webovém prohlížeči. Pokud se Vám okno samo neotevře, přejdete v prohlížeči na adresu **`http://127.0.0.1:3333/`**. Měla by se Vám ve webovém prohlížeči zobrazit následující stránka.




Ačkoliv se prostředí OpenRefine jeví jako webová stránka, veškeré operace probíhají lokálně (bez nutnosti připojení k internetu).

[3] Java Runtime Environment

[4] Dále v textu budeme předpokládat, že jste si nainstalovali verzi 2.6-beta1 pro OS Windows. Jednotlivé verze programu se mohou navzájem drobně lišit. Návod jak program nainstalovat na počítače s OS Mac nebo Linux najdete přímo na adrese http://openrefine.org/download.html

## 2.2. Orientace v programu

Úvodní menu programu je rozděleno do čtyř základních částí. Vyobrazení výše, ukazuje sekci tvorby nového projektu. Dalšími částmi jsou pak otevření již existujícího projektu, import projektu a nastavení jazyka[5].

**Vytvořit projekt (Create project)**

Vlastní vytvoření nového projektu se skládá ze tří kroků. Prvně vyberete soubor se zdrojovými daty a stiskneme tlačítko „Next". V druhém kroku „Preview" máme možnost vidět, jak se programu podařilo načíst naše data. Některé běžné formáty rozezná sám, s některými mu budeme muset trochu pomoci. Máme zde možnost programu říci, jak má data načíst „Parse data as" a také můžeme nastavit různé parametry těchto importů (např. vynechání řádků od začátku, od konce, jak interpretovat prázdné řádky, nastavit kódování znaků apod.). Jakmile máme správně nastaveny parametry importu, tak již jen projekt pojmenujeme a stisknutím tlačítka „Create Project" nakonec projekt necháme vytvořit.

OpenRefine umožňuje načíst data v následujících formátech:

- Comma-Separated Values (.csv), Tab-Separated Values (.tsv), ostatní *SV
- MS Excel dokumenty (.xls a .xlsx), Open Document Format (.odf, .ods)
- JavaScript Object Notations (.json)
- XML a RDF jako XML (.xml)
- Line-based formats (záznamy, logy, apod.)
- Google Data (Google Spreadsheet, Fusion Table)

Umístění dat pro import si můžete vybrat z následujících možností:

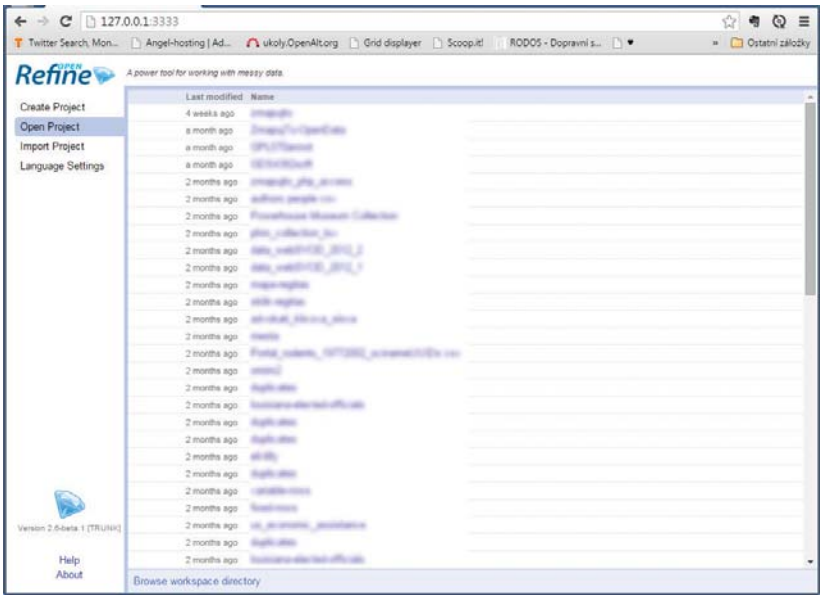
- **This computer -** Možnost tvorby nového projektu na základě dat, která máte uložena ve svém počítači. Zpracovatelné formáty souborů jsou uvedeny výše.
- **Web adresses (URLs) -** Na základě vložené jedné nebo více URL OpenRefine stáhne obsah, který na dané URL nalezne a pokusí se v dalším kroku automaticky najít separátor (často potřebuje pomoci) nebo vás nechá vybrat konkrétní obsah, který chcete stáhnout.
- **Clipboard -** Česky „schránka" vám umožňuje buď zápisem nebo vložením (CTRL+V) textového záznamu vytvořit projekt.
- **Google Data -** V této sekci můžeme povolit OpenRefine propojení s vaším Google Drive účtem[6]. Po propojení máte možnost nahrávat do OpenRefine dokumenty z Google Drive jako nové projekty.

**Otevřít projekt (Open project)**

Objeví se Vám seznam již vytvořených projektů. Projekty je možno v tomto rozhraní také přejmenovat či smazat.

[5] Aktuálně si můžete vybrat pouze mezi anglickou a italskou jazykovou mutací programu.

[6] Pro propojení a načtení dat z Google Spreadsheet nebo Fusion Table Vás program vyzve k přihlášení k účtu Google a k autorizaci.




**Importovat projekt (Import project)**

Import již existujícího projektu se zachovanými všemi úpravami v "Undo/Redo" ve formátu .tar nebo .tar.gz.

## 2.3. Práce s projektem

Základní pracovní plocha každého projektu v OpenRefine sestává ze 4 základních částí.

**Filtrace a změny na projektu**

V části filtrace ***Facet/Filter*** budete později pracovat s facety a textovou filtrací projektu. Facety představují v našem případě seskupení dat na základě zvoleného segmentu. Textovou filtrací můžete buď na základě regulárního výrazu, nebo fráze filtrovat odpovídající obsah projektu. V případě jakékoli filtrace dat je třeba si uvědomit, že operace, prováděné na sloupcích či buňkách budou vždy aplikovány pouze na aktuálně zvolený dataset.

Druhou částí jsou změny projektu, tedy sekce ***Undo/Redo*** (Zpět/Vpřed). V této sekci naleznete veškeré provedené kroky na projektu a můžete se v nich libovolně vracet zpět nebo vpřed. Postup na projektu bude zachován i při exportu celého projektu, tedy kdokoli kdo poté celý projekt importuje, uvidí veškeré podniknuté kroky a může s nimi libovolně pracovat.

Sekce ***Undo/Redo*** zároveň obsahuje dvě důležité funkce a to ***Extract***…(vytáhni) a ***Apply***…(aplikuj). Tyto funkce vám dovolují kdykoli postup na projektu přes Extract vyexportovat do textového souboru a aplikovat na jakýkoli jiný projekt za pomoci Apply.

**Data projektu**

V datové části naleznete náhled dat, která projekt obsahuje a možnost práce s nimi. Šipky u sloupců slouží jako menu funkcí na ně aplikovatelných (v případě zvolení šipky u sloupce se aplikuje daná funkce/filtrace pouze na sloupec, který jste zvolily). Zvláštní funkci plní sloupec úplně vlevo, jehož funkce jsou aplikovány na celý projekt. Jednotlivé řádky je pro pozdější filtraci možno označit buď pomocí vlajky nebo hvězdičky a potom s daným segmentem pracovat tak, že si vytvoříte Facet podle vlajky nebo hvězdičky.

**Základní informace o projektu**

Jednoduchý a přehledný panel o počtu řádků, záznamů s možností nastavení počtu řádků, které se vám v náhledu dat projektu budou zobrazovat.

Rozdíl mezi Row a Record:

- Row = Řádek
- Record = Záznam

Pokud v projektu se 7000 řádky budete mít seřazená data a smažete například přes funkci *Blank Down* duplicitní řádky, budete mít stále 7000 řádků, ale například 6000 záznamů (1000 bylo duplicitních).

**Export a nápověda**

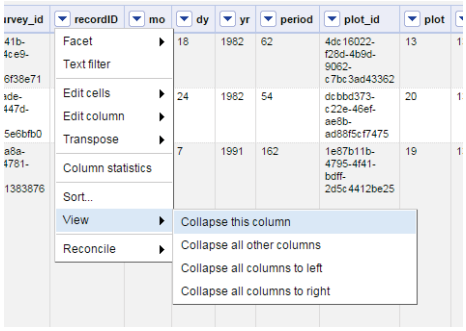
V pravém horním rohu naleznete možnost otevření projektu, nápovědy a především možnost exportu projektu. Exportovat můžete do formátů, které můžete zároveň importovat do OpenRefine. Zajímavou funkcí exportu je především ***Custom tabular exporter***, ve kterém si nejen můžete zvolit, jaké sloupce chcete exportovat, ale je zde i možnost, v případě propojení s vaším Google účtem, exportovat projekt do Google Spreadsheetu či do Google Fusion table.

## 2.4. Práce s daty

Náhledy na data nám mohou usnadnit orientaci v datasetu projektu. OpenRefine disponuje několika efektivními funkcemi, které Vám mohou zpříjemnit a urychlit práci.

**Views (náhledy)**

Náhledy nám umožňují schovat sloupce, které nás v dané chvíli nezajímají. Tuto funkci naleznete po kliknutí na příslušný sloupec, můžete buď sloupec schovat „***Collapse this column***“, schovat všechny ostatní sloupce „***Collapse all other columns***“ nebo schovat všechny sloupce nalevo či napravo od příslušného sloupce „***Collapse all columns to left/right***“.

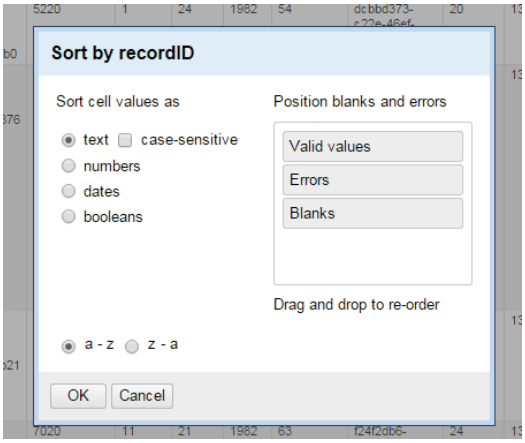



Po dvojitém kliknutí na mezeru mezi sloupci se sloupec znovu objeví. Data zůstávají ve zpracování, jako kdyby byl sloupec normálně viditelný, globální operace se proto mohou projevit i na schovaných sloupcích/řádcích (na rozdíl od filtrace).

**Sort (řazení)**

Řazení dat podle příslušného sloupce. Řadit buňky můžete jako textové, numerickém datové nebo boolean (true/false). Je důležité dbát na to, že pokud seřadíte data, dostáváte pouze nový pohled na ně a doopravdy nejsou seřazeny do doby, než nad sloupci zaškrtnete v

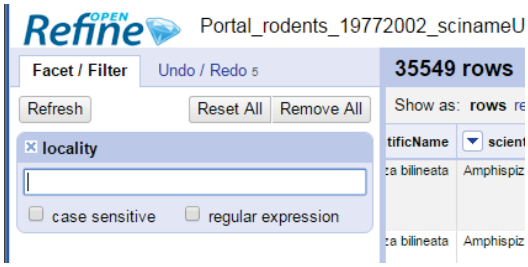
rozbalovacím tlačítku ***Sort -> Reorder rows permanently*** nebo pokud chcete náhled zrušit, tak ***Sort -> Remove sort***.




### Filtrace dat

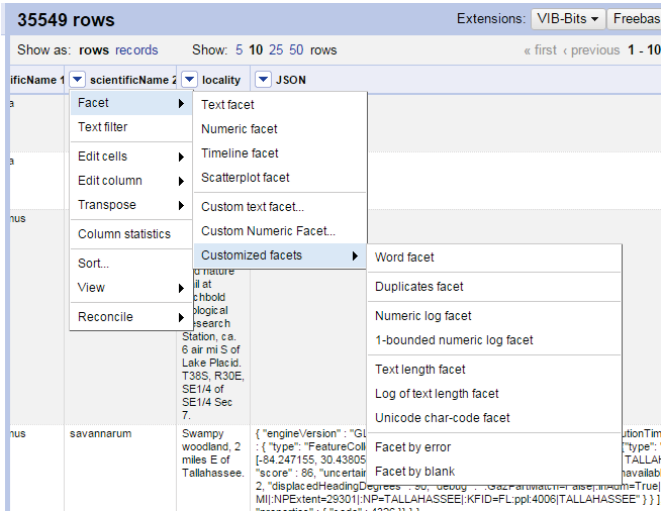
Filtrace dat nám pomáhá vytyčit určitý segment dat, na který se chceme buď podívat nebo přímo s ním pracovat. OpenRefine má tu výhodu, že při filtraci dat pracujete vždy pouze s vybraným segmentem dat, tedy pokud aplikujete jakoukoli funkci, aplikuji se pouze na aktuálně vyfiltrovaná data.

Základním typem filtrace, je filtrace textová. Na základě zadaného řetězce vyhledá buňky v daném sloupci, které jej obshaují. Po zatržení je možné použít regulární výrazy (***regular expression***) nebo zajistit, aby byl řetězec ***case sensitive*** (rozlišoval malá a velká písmena).




### Facety

Facety jsou seskupení dat na základě zvoleného klíče. To znamená, že při zvolení textového facetu se bude snažit OpenRefine dávat do jednoho segmentu k filtraci obsahově stejné buňky. K dispozici máte řadu různých obecně definovaných facetů, od textových, přes numerické po přizpůsobené v sekci ***Customized facets***.




Za zmínku stojí především přizpůsobený s názvem ***Word facet***. Tento facet vám zobrazí četnost výskytu slov ve sloupci. Hodí se především při tvorbě analýzy klíčových frází a po stažení HTML kódu z cizích stránek a jejich analýzy.

Jedinou výjimkou je tzv. ***Scatterplot facet***, což není ani tak facet, jako vizualizace 2D grafu vztahu označeného sloupce s jiným sloupcem. To znamená, že pokud u některého sloupce dáte ***Scatterplot facet***, objeví se vám obrazovka, na které můžete určit se kterým sloupcem má/může být promítnut. OpenRefine má tuto funkci poněkud nedokončenou. Na grafu chybí například měřítko hodnot a nedá se s ním téměř vůbec pracovat.

## 2.5. Práce se sloupci

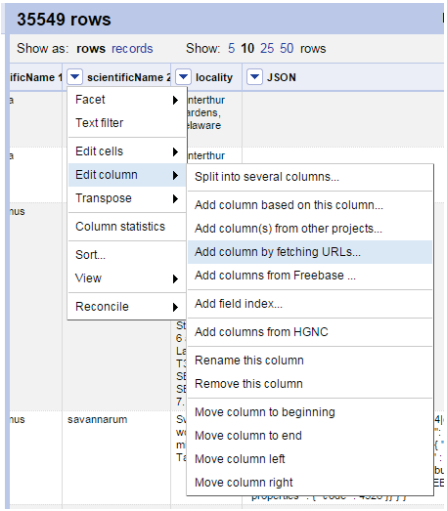
Sloupce jsou jeden ze základních prvků každého projektu. O to důležitější je práce s nimi. Níže naleznete základní operace, které můžete se sloupci provádět a příklady.

***Split into several columns*** **(Rozdělení sloupce na více sloupců)**

Představte si, že máte v datasetu v jednom sloupci více hodnot. Například GPS souřadnice pozice „50°4’15.963″N, 14°24’3.179″E“. Tyto souřadnice chcete rozdělit na dva různé sloupce, přičemž jeden bude obsahovat pouze severní a druhý pouze východní souřadnice. To uděláte za pomoci funkce Split into several columns, kde jako separátor mezi souřadnicemi nastavíte čárku a OpenRefine vám vytvoří nový sloupec s oddělenou částí dat a v původním ponechá první část. ***Split into several columns*** tedy slouží k rozdělení příslušného sloupce na základě separátoru.

***Add column based on this column***

V překladu „Přidat sloupec na základě dat sloupce“. Jedná se o velmi často používanou operaci, která na základě dat v příslušném sloupci vytváří nový sloupec s výstupem (Příklad: Pokud máte sloupec s celým jménem uživatelů a chcete vytvořit nový sloupec pouze s jejich příjmeními). V tom případě u sloupce s celým jménem dáte ***Edit column -> Add column based on this column***, vyplníte název nového sloupce a do ***Expression*** vložíte `value.split(" ")[1]`. Funkce split je detailněji rozebrána v kapitole popisující funkce na řetězcích.




***Add column by fetching URLs***

V českém překladu: „Přidat sloupec na základě dat stažených z URL“. Nutno poznamenat, že při tomto procesu můžete stále pracovat s daty z příslušného sloupce. U přidávání nového sloupce se objevuje jedna nová sekce, nazvaná ***Throttle delay***. Jedná se zpoždění mezi dotazy, které bude OpenRefine odesílat v ms (nikoliv ve vteřinách).

Příklad: Pokud máte sloupec plný URL a chcete k nim nastahovat data o jejich sdílení a likes, uděláte to tak, že u daného sloupce dáte ***Edit column -> Add column by fetching URLs***, nazvete nový sloupec, zadáte zpoždění (v tomto případě doporučuji 500 ms) a do ***Expression*** vložíte následující výraz: `http://graph.facebook.com/?ids= + value` . Po dokončení získáte v novém sloupci výstup ve formátu JSON. O JSON, jeho parsování a širším využití tohoto procesu naleznete více informací v části věnované příkladům.

***Add columns from Freebase***

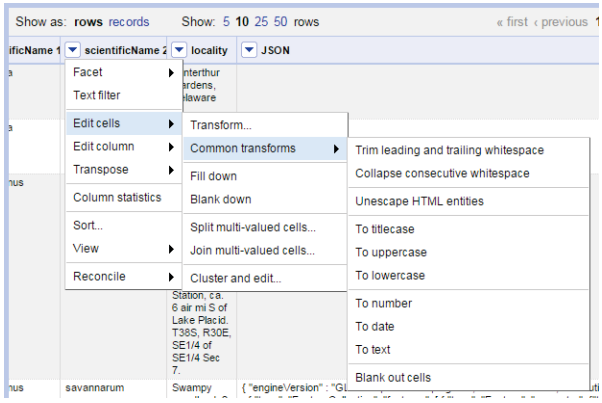
„Přidat sloupec na základě synchronizace obsahu buněk s Freebase”. Tato funkce je přímo navázána na použití ***Reconcilation***, které podrobněji naleznete v sekci praktické příklady.

**Další dostupné akce se sloupci**

- ***Rename this column*** - přejmenování sloupce
- ***Remove this column*** - odstranění sloupce
- ***Move column*** - posun sloupce, na výběr máte ze 4 možností:
    - ***Move column to beginning*** - přesun sloupce na začátek
    - ***Move columnt to end*** - přesun sloupce na konec (za všechny sloupce)
    - ***Move column left*** - přesun sloupce o jednu pozici doleva
    - ***Move column right*** - přesun sloupce o jednu pozici doprava

## 2.6. Základní operace s buňkami

Buňky (*Cells*) jsou základní jednotkou OpenRefine. Každá buňka obsahuje nějakou hodnotu, která může být textového, numerického nebo datového formát a přísluší ji právě jeden sloupec.




***Transform* (transformace)**

Změna obsahu buněk na základě vložené funkce v *GREL*, *Jython* nebo *Closure*. O GREL a funkcích, které se dají využít při transformaci, naleznete více informací v kapitole věnované funkcím.

OpenRefine také nabízí základní set předpřipravených transformací ***Common transforms***, které můžete rovnou aplikovat:

- ***Trim leading and trailing whitespace*** - odstraní mezery na koncích a začátcích buněk ve sloupci
- ***Collapse consecutive whitespace*** - pokud v buňce nalezne dvě po sobě jdoucí mezery, změní je pouze na jednu
- ***Unescape HTML entities*** - převede HTML na klasické znaky (například `&#33;` na !
- ***To titlecase*** - každé slovo buňky bude začínat velkým písmenem
- ***To uppercase*** -změní text v buňkách na velká písmena
- ***To lowercase*** - změní text v buňkách na malá písmena
- ***To number / To date / To text*** - změna formátu buněk na číslo / datum / text
- ***Blank out cells*** - smaže obsah všech buněk příslušného sloupce

***Fill down* (vyplň pod)**

Fill down je funkcí, která vyplní prázdné buňky sloupce buňkami nad nimi. Je proto důležité dbát na pořadí buněk ve chvíli, kdy chcete tuto funkci využít.

***Blank down* (vymaž pod)**

Blank down je funkcí opačnou k Fill down. Pokud OpenRefine nalezne v příslušném sloupci dvě stejné buňky pod sebou, smaže té níže postavené obsah buňky.

Tato funkce se často používá při zbavování se duplicit (příkazy ***Sort* + *Edit cells* -> *Blank down***).

***Split multi-valued cells* (rozdělení řádků s více hodnotami)**

Rozdělení řádku na základě pevně určeného separátoru. Funkce podobná ***Split into several columns***, s tím rozdílem, že nerozdělí obsah buněk na nové sloupce, ale rozdělí hodnoty do nových řádků.

**Příklad:** Při dělení hodnoty „`7.1`“ s použitím separátoru „.“ zůstane na řádku hodnota `7` a do nového řádku se vepíše hodnota 1.

***Join multi-valued cells*** **(spojení řádků s více hodnotami)**

Opačná funkce k funkci ***Split multi-valued cells***. Podívá se o řádek níže, a pokud je v daném řádku hodnota s prázdným sloupcem úplně nalevo, vloží tuto hodnotu do buňky výše a oddělí hodnoty separátorem.

**Příklad:** Po rozdělení hodnot, viz příklad výše, na stejném sloupci spustíme funkci ***Edit cells -> Join multi-valued cells*** a jako separátor určíme `"."`. Dostaneme se tak na původní hodnotu, tedy `"7.1"`.

***Cluster*** **(Clusterizace)**

Clusterizace je slučování hodnot na základě podobnosti. Hodí se především ve chvíli, kdy v jednom sloupci máte soubor hodnot, které jsou podobné, ale ne stejné a chcete je změnit na ucelený název.

Například pokud ve sloupci `"Země"` budete mít uvedeny země původu objektů v datasetu a některé buňky sloupce `"Země"` budou pro označení České republiky užívat názvy jako: `"Česká republika"`, `"Česká Republika"`, `"České Republiky"`, `"Ceska Republika"`, `"CESKA REPUBLIKA"`. Tyto všechny názvy pomocí clusterizace spojíte do jednoho primárního tvaru, který si zvolíte.

Při nacházení podobných frází je třeba vybrat konkrétní metodu, kterou OpenRefine použije. Mezi metody, které k cluesterizaci můžete využít, patří:

- ***Key collision***
- ***Nearest neighbor***
- … a k nim přidružené funkce.

Clusterizace se také často využívá v pozdějších fázích analýzy klíčových frází pro slučování frází na jednotný tvar. Konkrétní návod jak na to naleznete v příkladu clusterizace a slučování hodnot.

## 3. Funkce jazyka GREL

Zkratka **GREL** *Google Refine Expression Language* (nyní již *OpenRefine Expression Language*, ale zkratka zůstala zachovaná). Jak už název sám o sobě napovídá, jedná se o jazyk, jehož zápisem můžete provádět základní i pokročilé funkce pro práci s obsahem projektu. Níže si představíme výběr nejpoužívanějších funkcí. Kompletní popis funkcí lze najít v nápovědě programu, nebo na adrese https://github.com/OpenRefine/OpenRefine/wiki/GREL-Functions.

Mimo GREL dovoluje OpenRefine využít také jazyky *Clojure*[7] a *Jython*[8]. Pro naše ukázkové účely je zde ale jednodušší a používanější právě GREL, proto funkce a příklady na nich

[7] Clojure je moderní dialekt programovacího jazyka Lisp. Jedná se o univerzální jazyk podporující funkcionální programování

[8] Jython (dříve známý jako JPython) představuje implementaci programovacího jazyka Python v jazyce Java

budou řešeny právě tímto zápisem. Pozor! GREL je *case sensitive jazyk*, tedy je rozdíl, zda použijete „`value`“ nebo „`Value`“.

### 3.1. Funkce pro logické výrazy

Jedná se o funkce s možnou návratovou hodnotou pouze `0` (False – není pravda), nebo `1` (True – je pravda).

**`and(boolean 1, boolean 2,…)`** - logická spojka `AND`. Vrací `TRUE`, pokud jsou všechny výrazy, zapsané v and platné. Tedy výsledkem `and(1<3, 7>4, 10>1)` bude `TRUE`, jelikož jsou všechny výrazy pravdivé. Oproti tomu `and(1<3, 7>4, 10<1)` vrátí `FALSE`, jelikož jeden z výrazů je nepravdivý.

**`or(boolean 1, boolean 2,…)`** - logická spojka `OR`. Vrací `TRUE` v případě, že je alespoň jeden z výrazů v jejím zápisu platný. Tedy výsledkem `or(1<3, 7<4)` bude `TRUE`, jelikož první výraz je pravdivý. Výsledkem `or(1>3, 7<4)` bude `FALSE`, jelikož jsou oba výrazy nepravdivé.

**`not(boolean 1)`** - negace jakéhokoli boolean výrazu. Výsledkem `not(1<4)` by tedy bylo `FALSE`, jelikož výraz je sám o sobě pravdivý a not jej zneguje.

### 3.2. Vlastnosti řetězce

**`length(string s)`** - navrací délku daného řetězce. Tedy v případě `length("příklad")` bude výsledkem funkce `7`.

### 3.3. Testování řetězce

**`startsWith(string s, string sub)`** - funkce, která vrací `TRUE` pokud zadaný řetězec `s` začíná řetězcem `sub`. Například výsledkem `startsWith("příklad", "pří")` bude `TRUE`, jelikož `"příklad"`, začíná na `"pří"`. Výsledkem `startsWith("překlad", "pří")` bude `FALSE`, jelikož `"překlad"` nezačíná na `"pří"`.

**`endsWith(string s, string sub)`** - opačná funkce k funkci `startsWith`, analyzující konec řetězce. Výsledkem `endsWith("příklad", "lad")` tedy bude `TRUE`, jelikož řetězec `"příklad"` končí na `"lad"`. Výsledkem `endsWith("příklad", "dný")` bude `FALSE`, jelikož řetězec `"příklad"` nekončí na `"dný"`.

**`contains(string s, string sub)`** - funkce, která slouží jako indikátor, zda v řetězci `s` existuje subřetězec `sub`. Například tedy `contains("příklad", "kla")` vrátí `TRUE`, jelikož řetězec `"příklad"` obsahuje řetězec `"kla"`.

### 3.4. Úpravy řetězců

**`toLowercase(string s)`** - převede obsah řetězce `s` na malá písmena.

**`toUppercase(string s)`** - převede obsah řetězce `s` na velká písmena.

**`toTitlecase(string s)`** - převede obsah řetězce `s` na velká písmena na začátku každého slova.

### 3.5. Odsekávání částí

**`trim(string s)`** - `trim` odsekne na začátku a konci řetězce mezery, pokud nějaké nalezne. Je skvělou funkcí k očištění řetězců. Například `trim("  příklad")` vrátí hodnotu `"příklad"` (tedy bez úvodních mezer).

**`chomp(string s, string sep)`** - `chomp` (česky žvýkání) je funkce, která zadanému řetězci `s` v případě shody odsekne řetězec `sep`. Tedy například `chomp("příklady", "dy")` navrátí řetězec `"příkla"`.

### 3.6. Subřetězce

**`substring(s, number from, optional number to)`** - navrací subřetězec dle zadaného rozsahu (od parametru `from` po parametr `to`). Například `substring("příklad", 2, 4)` navrací řetězec `"ík"`, jelikož je od druhého do čtvrtého písmena daného řetězce[9].

### 3.7. Nalezení a nahrazení

**`indexOf(string s, string sub)`** - navrací pozici řetězce `sub` v řetězci `s` nebo `-1` v případě, že řetězec `sub` nebyl v řetězci `s` nalezen. Například `indexOf("příklad", "kl")` navrátí `3`, ale `indexOf("příklad", "lk")` navrátí `-1`.

**`lastIndexOf(string s, string sub)`** - navrací poslední pozici řetězce `sub` v řetězci `s`. Tedy pokud se řetězec `sub` vyskytuje v řetězci `s` vícekrát, navrátí jeho poslední pozici. Například `lastIndexOf("filip", "i")` navrátí `3`. V případě, že není řetězec nalezen, navrací funkce `-1`.

**`replace(string s, string f, string r)`** - nahrazení textového řetězce `f` v řetězci `s` řetězcem `r`. Například `replace("příklad", "pří", "pře")` nám navrátí řetězec `"překlad"`, jelikož vymění řetězec `"pří"` za řetězec `"pře"`.

**`replaceChars(string s, string f, string r)`** - funkce, podobná `replace` s tím rozdílem, že místo nahrazení řetězce `f` nahrazuje za každé písmeno v řetězci `f` daným písmenem řetězce `r`. Například `replaceChars("příklad", "íd", "*-")` navrátí řetězec `"př*kla-"`, jelikož nahradí první písmeno odpovídající `f` (což je `"í"`) prvním písmenem, odpovídajícím `r` (což je `"*"`) a stejně tak s druhým písmenem.

**`match(string s, regexp p)`** - pokusí se porovnat řetězec `s` s regulárním výrazem `p` a případně navrátit odpovídající pole hodnot regulárnímu výrazu. Například `match("230.22398, 12.3480", /\.(\d\d\d+)/)` navrátí hodnotu `3480`.

### 3.8. Parsování a dělení

**`toNumber(s)`** - převede řetězec `s` na číslo.

[9] pozice znaků v řetězci se indexují od 0, tedy první znak má index 0, druhý 1 atd.

**split(s, sep)** - rozdělí `s` separátorem `sep` na pole hodnot. Například `split("Filip,Pavel,Petr,Adam",",")` navrací pole hodnot `["Filip", "Pavel", "Petr", "Adam"]`.

**splitByLengths(string s, number n1, number n2, …)** - rozdělí řetězec `s` dle zadaných jednotlivých délek na části. Například `splitByLengths("Příkladová věta", 3, 2, 6)` navrací pole `["Pří", "kl", "adová "]`.

**smartSplit(string s, optional string sep)** - chytřejší (automatické) dělení řetězce. V případě, že není zadán separátor, hledá čárku nebo tečku jako oddělovač řetězce.

**splitByCharType(s) -** vrátí pole řetězců, rozdělených do skupin dle typu řetězce `s` (shlukování na základ velikosti písmen,…). Například pro řetězec `"CTFinder"` jsou návratové hodnoty jednotlivých částí pole následující: `splitByCharType("CTFinder")[0]` navrátí řetězec `"CTF"`, `splitByCharType("CTFinder")[1]` navrátí řetězec `"inder"`[10].

**partition(string s, string or regex frag, optional boolean omitFragment)** - participace rozděluje řetězec na odpovídající fragment a zbývající text. Návratovou hodnotou je pole hodnot, složené z textu před fragmentem, fragmentu a textu za fragmentem. Například `partition("Příkladová věta na ukázku", "věta")` vrátí pole hodnot `["Příkladová ", "věta", " na ukázku"]`.

### 3.9. Kódování a hash

**diff(o1, o2, optional string timeUnit)** - pro řetězce vrací index, kde se dané řetězce liší. Pro datum vrací rozdíl dat mezi sebou.

**escape(string s, string mode)** - převede obsah řetězce do požadovaného formátu (html, url, xml, csv, javascript).

**unescape(string s, string mode)** - převede obsah řetězce z požadovaného formátu do plain textu.

**md5(string s)** - navrací zahashovaný obsah řetězce s v MD5.

**phonetic(string s, optional string encoding)** - navrací fonetické kódování řetězce `s`. Kódování je možno změnit (přednastaveno DoubleMetaphone).

### 3.10. Funkce pro práci s poli

**length(array a)** - navrací počet prvků pole `a`.

**slice(array a, number from, optional number to)** - funkce `slice` slouží k získání prvků pole od (`from`) / do (`to`) určitého rozsahu. Například `slice([1,2,3,4], 1,3)` navrátí pole hodnot `[2,3]`.

**reverse(array a)**- obrátí pořadí prvků v poli `a`. Například `reverse([1,2,3,4,5])` navrátí pole `[5,4,3,2,1]`.

[10] Zápis pole[0] vrátí první prvek pole, pole[1] vrátí druhý prvek apod.

**sort(array a)**- seřadí vzestupně prvky pole. Například `sort([1,4,3,5])` navrátí pole `[1,3,4,5]`.

**sum(array a)**- navrátí sumu (součet) hodnot pole `a`. Například `sum([1,2,3])` navrátí hodnotu `6`.

**join(array a, string sep)**- vrátí obsah pole jako řetězec, oddělený přednastaveným separátorem. Například `join(["cokoli1","cokoli2","cokoli3"],";")` vrátí řetězec `"cokoli1;cokoli2;cokoli3"`.

**uniques(array a)** - vyčístí pole `a` od duplicitních záznamů. Například: `uniques([1,1,3,5,3,6])` vrátí pole hodnot `[1,3,5,6]`.

### 3.11. Matematické funkce

**floor(number d)** - dolní celá část čísla `d`. Například `floor(3.7)` je hodnota `3`.

**ceil(number d)** - horní celá část čísla `d`. Například `ceil(3.7)` je hodnota `4`.

**round(number d)** - zaokrouhlení čísla `d` k nejbližšímu celému číslu. Například `round(3.7)` navrací hodnotu `4`.

**min(number d1, number d2)** - návratovou hodnotou je menší ze dvou vložených hodnot.

**max(number d1, number d2)** - návratovou hodnotou je větší ze dvou vložených hodnot.

**mod(number d1, number d2)** - návratovou hodnotou je `d1 modulo d2` (zbytek po dělení čísla `d1` číslem `d2`). Například `mod(74,9)` vrátí hodnotu `2`.

**ln(number d)** - návratovou hodnotou je přirozený logaritmus čísla `d`.

**log(number d)** - návratovou hodnotou je logaritmus o základu `10` čísla `d`.

**exp(number d)** - návratovou hodnotou je Eulerovo číslo umocněné číslem `d`.

**pow(number d, number e)** - návratovou hodnotou je číslo `d` umocněné číslem `e`. V jednoduchosti si lze pow představit jako `d^e`. Například `pow(2,3)` navrátí hodnotu `8`. Místo odmocniny můžete použít `e = 0.5`.

### 3.12. Datové a časové funkce

**now()–** návratovou hodnotou je aktuální čas.

**toDate(o, boolean month_first / format1, format2, … )** - převede `o` na datum. Veškeré ostatní argumenty jsou nepovinné. **`month_first`**: nastavte jako `FALSE`, pokud chcete zobrazovat den měsíce na první pozici. **`formatN`**: zápis formátu data podle SimpleDateFormat[11]. Pokud tedy chcete používat datum ve formátu `"1/7/2013 19:45:15"`, použijete následující GREL funkci: `toDate(value,"dd/mm/YYYY H:m:s")`

[11] http://docs.oracle.com/javase/7/docs/api/java/text/SimpleDateFormat.html

**`toString(o, optional string format)`** - pokud je `o` typu datum, format specifikuje, jakým způsobem se bude formátovat datum. Pro formátování se použije SimpleDateFormat tak jako ve funkci `toDate`. Pokud například budete chtít parsovat datum ve fromátech `"Nov-09"` a `"11/09"` a výsledek zobrazit jako řetězec ve formátu `"2009-11"` tak můžeme použít zápis `value.toDate("MM/yy","MMM-yy").toString("yyyy-MM")`.

**`diff(date d1, date d2, optional string timeUnit)`** - návratovou hodnotou je rozdíl mezi dvěma daty.

**`inc(date d, number value, string unit)`** - návratovou hodnotou je datum `d`, posunutý o `value`. Přednastavenými jednotkami jsou hodiny.

**`datePart(date d, string unit)`** - návratovou hodnotou je zadaná část data. Zápisy pro získány části data mohou být následující:

| jednotka | vrácená část data | vrácený typ | Příklad použití pro value = 2014-03-14T05:30:04Z |
|---|---|---|---|
| `years, year` | Year | Number | `value.datePart("years") -> 2014` |
| `Months, month` | Month | Number | `value.datePart("months") -> 2` |
| `Weeks, week, w` | Week (of the year) | Number | `value.datePart("weeks") -> 2` |
| `weekday` | Day of the week | String | `value.datePart("weekday") -> Friday` |
| `Hours, hour, h` | Hour | Number | `value.datePart("hours") -> 5` |
| `Minutes, minute, min` | Minute | Number | `value.datePart("minutes") -> 30` |
| `Seconds, sec, s` | Seconds | Number | `value.datePart("seconds") -> 04` |
| `Milliseconds, ms, S` | Millseconds | Number | `value.datePart("milliseconds") -> 0` |
| `time` | Date expressed as milliseconds since the Unix Epoch | Number | `value.datePart("time") -> 1394775004000` |

### 3.13. Práce s JSON řetězci

**`jsonize(value)`** - převede výstup vložené hodnoty (proměnnou) na JSON literál.

**parseJson(string s)** - parsuje JSON řetězec `s`. K získání dat může být využit i v kombinaci s funkcí get. Například:

`parseJson("{'test':1}").get("test")` jejíž návratovou hodnotou bude `1`.

V případě, že potřebujete z JSON výstupu získat více proměnných, je potřeba k nim buď přistupovat jednotlivě nebo projít výstup pomocí funkce `forEach()`. Konkrétní příklad:

```
{
      "status":"OK",
      "language":"czech",
      "keywords":[
            {"text":"Příklad1"},
            {"text":"Příklad2"},
            {"text":"Příklad3"}
      ]
}
```

GREL výraz:
`forEach(value.parseJson().keywords,v,v.text).join("::")`,
jehož výstupem bude `Příklad1::Příklad2::Příklad3`.

### 3.14. Parsování HTML

**parseHtml(string s)** - navrací celý HTML dokument a doplňuje uzavírací tagy tam, kde ve zdrojovém kódu chybí. Často se používá ve spojení s funkcí `select()`, která vám umožní získávat pouze konkrétní části HTML dokumentu.

**select(Element e, String s)** - návratovou hodnotou funkce `select` je konkrétní element HTML za použití selektoru nebo syntaxe k jeho specifikaci[12].

**htmlAttr(Element e, String s)** - návratovou hodnotou je atribut HTML elementu.

**htmlText(Element e)** - návratovou hodnotou je text uvnitř HTML elementu.

**innerHtml(Element e)** - návratovou hodnotou je innerHTML daného HTML elementu.

**outerHtml(Element e)** - návratovou hodnotou je outerHTML daného HTML elementu (tedy jak obsah, tak počateční a ukončovací tag).

**ownText(Element e)** - návratovou hodnotou je pouze text, příslušející danému HTML elementu (ne obsah potomků elementu).

### 3.15. Ostatní funkce

**type(o)–** návratovou hodnotou je typ proměnné `o` (number, string,…).

[12] Více informací o selector-syntax naleznete na adrese http://jsoup.org/cookbook/extracting-data/selector-syntax

**`hasField(o, string name)`** - návratovou hodnotou je informace, zda `o` obsahuje pole (atribut, parametr), které se nazývá `name`. Například `cell.hasField("value")` vrací vždy `TRUE`, jelikož každá buňka má hodnotu.

**`cross(cell c, string projectName, string columnName)`** - návratovou hodnotou je pole `0` a více řádků z projektu `projectName`, pro které má sloupec `columnName` stejný obsah, jako buňka `c`. Jedná se tedy o získávání dat na základě shody buněk se sloupcem z jiného projektu.

Praktičtější pro nás ale bude v případě požadavku na propojení dat z jiného projektu využít přímo nabídku ***Edit column -> Add column(s) from other projects.***

**`facetCount(choiceValue, string facetExpression, string columnName)`** - návratovou hodnotou `facetCount` je počet shod, odpovídajících zadanému výrazu. Uvažujme následující příklad:

| Dárek | Hodné dítě |
|---|---|
| Lego | Pepík |
| Traktůrek | Fanda |
| Autíčko | Pepík |
| Ponožky | Péťa |

Pokud bychom chtěli zjistit, které z dětí dostalo kolik dárku, vytvoříme si na to jednoduchý facet do nového sloupce. Výraz by vypadal následovně: `facetCount(value, "value", "Hodné dítě")`. Výstup by poté vypadal následovně:

| Dárek | Hodné dítě | Count |
|---|---|---|
| Lego | Pepík | 2 |
| Traktůrek | Fanda | 1 |
| Autíčko | Pepík | 2 |
| Ponožky | Péťa | 1 |

## 4. Regulární výrazy

Regulární výrazy[13] (někdy nazývané zkratkou „regex“) jsou způsob, jak zapsat pomocí textového řetězce výraz, který popisuje celou množinu řetězců (tzv. regulární jazyk). Regulární výraz si můžeme představit jako "šablonu" pro konkrétní podmnožinu slov, která

[13] Pro důkladné studium regulárních výrazů v češtině doporučuji např.: http://www.root.cz/serialy/regularni-vyrazy/

mají určité společné vlastnosti. Tyto regulární výrazy pak umožňují například testovat, jestli vstupní text vyhovuje danému regulárnímu výrazu, zjištění pozice ve vstupním textu, kde shoda s regulárním výrazem začíná, umožňují záměnu textu v podvýrazech regulárního výrazu, extrahovat všechny shody s regulárním výrazem do předem daných proměnných apod.

Pro vytvoření regulárního výrazu je potřeba znát speciální syntaxi, která reprezentuje různé typy znaků, které se mohou v textovém řetězci objevit. My si představíme alespoň základní syntaxi.

- `.` - znak tečka reprezentuje jakýkoliv znak
- `[<seznam/rozsah>]` - když mezi hranaté závorky vložíme libovolný seznam znaků nebo rozsah znaků, tak to bude zástupným znakem pro kterýkoliv prvek z této závorky. Například výraz `[ABC]` bude odpovídat znaku `A` nebo `B` nebo `C` (pozor, záleží na velikosti písmen!). Výraz `[A-B]` odpovídá velkým znakům a výraz `[A-Za-z0-9]` odpovídá velkým a malým znakům a číslicím.
- `\d` - odpovídá libovolné číslici (ekvivalentní k výrazu `[0-9]`)
- `\w` - odpovídá znakům, které mohou být součástí slova (ekvivalentní k výrazu `[A-Za-z0-9_]`)
- `\s` - odpovídá jakémukoliv netisknutelnému znaku, jako je mezera, tabulátor nebo znak nového řádku.
- `^` - odpovídá začátku řetězce. Používá se to v případě, pokud chceme určit, že řetězec musí začínat na konkrétní regulární výraz.
- `$` - odpovídá konci řetězce. Používá se to v případě, pokud chceme určit, že řetězec musí končit na konkrétní regulární výraz.

Výše uvedené symboly se mohou kombinovat, takže pokud například chceme zapsat regulární výraz, který bude představovat jak slovo `"organise"` tak slovo `"organize"`, tak ho zapíšeme následujícím způsobem: `/organi[sz]e/`[14].

Výše uvedené symboly pak mohou být kombinovány s operátory, které vyznačují, kolikrát se ten který podvýraz může opakovat. Mezi nejpoužívanější operátory patří:

- `*` - představuje opakování výrazu libovolně-krát (včetně nuly). Například regulární výraz `/.*/` představuje jakýkoliv libovolný řetězec.
- `+` - představuje opakování jednou a vícekrát, znamená to, že opakující podvýraz se musí objevit alespoň jedenkrát. Například regulární výraz `/head\s+rest/` bude odpovídat `"head rest"` s jednou mezerou, `"head  rest"` s dvěma mezerami ale již nebude odpovídat řetězci `"headrest"` bez mezery.
- `?` - představuje, že se výraz bude opakovat buď 0x nebo 1x. Například regulární výraz, který bude odpovídat jak řetězci `"colour"` tak `"color"` zapíšeme `/colou?r/`.
- `{}` - ve složené závorce můžeme i přímo vyznačit, kolikrát chceme, aby se podvýraz opakoval. Například `/a{2}/` odpovídá řetězci `"aa"` a výraz `/a{2,4}/` odpovídá libovolnému slovu z `"aa"`, `"aaa"`, `"aaaa"`.

Níže následují praktické příklady regulárních výrazů:

[14] Znak / se používá k označení, že se jedná o regulární výraz, je to podobné jako když pomocí uvozovek označujeme, že se jedná o textový řetězec.

`/[0-9]|[1-9][0-9]/` čísla 0 až 99

`/\d{2}/` sekvence dvou číslic desítkové soustavy (00, 01, ...,98, 99)

`/[0-9a-fA-F]|[1-9a-fA-F][0-9a-fA-F]+/` hexadecimální čísla

`/(19|20)\d{2}/` letopočty 1900-2099

`/\d+\.\d+\.\d+\-.\d+/` odpovídá IP adrese (jednoduchá verze)

`/http://[a-zA-Z_.]+/` odpovídá webové adrese (jednoduchá verze)

`/(http://)?w{3}[a-zA-Z_.]+\.cz/` odpovídá české doméně začínající na www

`/[a-zA-Z_.]+@[a-zA-Z_.]+/` odpovídá e-mailové adrese (jednoduchá verze)

Tvorba regulárních výrazů je náročný proces, zvláště pokud je potřeba podchytit všechny vlastnosti popisované množiny řetězců. V našem případě bychom za ideální regulární výraz pro IP adresu mohli považovat:

```
/^(?:(?:25[0-5]|2[0-4]\d|[01]\d\d|\d?\d)(?(\.?\d)\.)){4}$/
```

pro URL adresu:

```
/^http://([a-zA-Z0-9_\-]+)([\.][a-zA-Z0-9_\-]+)+([/][a-zA-Z0-
9\~\(\)_\-]*)+([\.][a-zA-Z0-9\(\)_\-]+)*$/
```

a pro e-mailovou adresu:

```
/^[\w-]+(?:\.[\w-]+)*@(?:[\w-]+\.)+[a-zA-Z]{2,7}$/
```

Je dobré předtím, než začnete nějaký složitější regulární výraz sami konstruovat, pokusit se najít, jestli již požadovaný výraz nesestrojil někdo před Vámi. Můžete využít např. databázi regulárních výrazů na adrese http://www.regxlib.com/.

Pro praktickou práci s regulárními výrazy může být praktické si vytisknout tzv. Cheat Sheet kde jsou všechny možnosti regulárních výrazů popsány stručně na několika stranách A4, k dispozici je například na adrese: http://arcadiafalcone.net/GoogleRefineCheatSheets.pdf.

### 4.1. Použití regulárních výrazu v jazyce GREL

**`value.match(/regex/)`-** pokusí se zjistit, jestli regulární výraz odpovídá řetězci `value`. Funkce vrátí pole řetězců (pokud proběhla shoda), které byly nalezeny. Pomocí indexu můžeme získat konkrétní výskyt řetězce.

Příklad:
```
value = "The cat can't lay on the cot"
value.match(/c.t/) -> ["cat", "cot"]
value.match(/c.t/)[0] -> "cat"
```

**`value.contains(/regex/)`** - zjistí, jestli řetězec `value` odpovídá regulárnímu výrazu. Vrací pravdivostní hodnotu (`true`/`false`).

Příklad:
```
value = "coffee and tea and chai and mate"
value. contains(/and/) -> TRUE
```

**`value.replace(/regex/, u)`** - vrátí řetězec `value`, kde budou všechny výskyty regulárního výrazu nahrazeny řetězcem `u`.

Příklad:
```
value = "coffee and tea and chai and mate"
value. replace(/\sand\s/, ", ") -> "coffee,
tea, chai, mate"
```

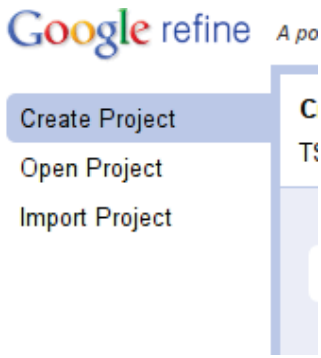
# 5. Praktické příklady

## 5.1. Čištění dat

V tomto příkladu si ukážeme nejběžnější postup, jak v OpenRefine vyčistit data, která jsme dostali ke zpracování. Jako příklad dat nám poslouží seznam nemocnic v Zimbabwe, která jsou dostupná ve formě otevřených dat. Tato data naleznete v souboru s názvem `provincial_and_district_hospitals.csv` který je možné stáhnout z adresy `https://github.com/MiroslavKubasek/OpenRefine-tutorial` v adresáři `data`.

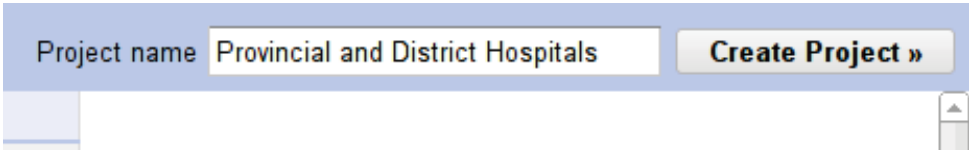
**Krok 1: Vytvoříme nový projekt**

1. Spustíme program OpenRefine a ve webovém prohlížeči si otevřeme stránku http://127.0.0.1:3333
2. Klikneme na záložku ***Create Project*** v levé části stránky.

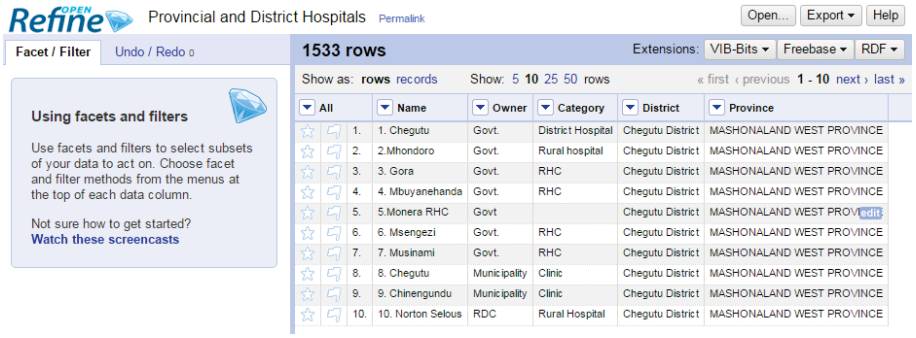



3. V OpenRefine v záložce ***This Computer*** klikneme na tlačítko ***Zvolit soubory*** (***Choose Files***) a vybereme z disku stáhnutý soubor `provincial_and_district_hospitals.csv`. Alternativně bychom mohli také postupovat tak, že klikneme na záložku ***Web Address (URLs)*** a zde vložíme přímo URL adresu, na které se soubor s daty nachází[15]. Jakmile máme zdroj dat nastaven, tak stiskneme tlačítko ***Next »*.**
4. Nyní se nám zobrazí náhled importovaných dat, tak jak je OpenRefine automaticky interpretuje. Pokud jsme použili dobře formátovaný soubor CSV, či jiný běžný formát pro uložení dat, tak by měl OpenRefine automaticky odhadnout jak data naimportovat.
5. Prohlédneme si v náhledu pečlivě, jestli OpenRefine data správně interpretoval. Důležité je zkontrolovat, zdali se v datech nevyskytují podivné znaky. Pokud ano, tak v tom případně bude potřeba nastavit správně kódování importovaných dat (***Character encoding***). V dnešní době jsou většinou data dostupná v kódování UTF-8.
6. Pojmenujeme náš projekt v políčku ***Project name*** a nazveme ho například `"Provincial and District Hospitals"`

[15] V našem případě by se jednalo a adresu: https://raw.githubusercontent.com/MiroslavKubasek/OpenRefine-tutorial/master/data/provincial_and_district_hospitals.csv




7. Po stisknutí tlačítka ***Create Project »*** se nám data naimportují a zobrazí se nám okno s vytvořeným projektem. Přednastaveno máme, že vidíme prvních 10 záznamů, je možné tento počet zvýšit až na 50, vpravo nahoře pak můžeme stránkovat zobrazovaná data.



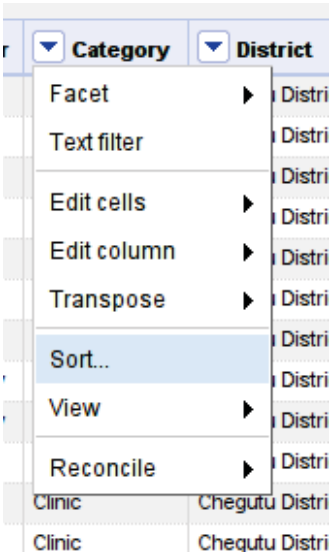

**Poznámka:** Nyní máme úspěšně vytvořený nový projekt. Nezapomeňte, že ačkoliv s OpenRefine pracujeme přes webový prohlížeč, tak server který stránky generuje je stále Váš počítač. Nemusíte se tedy obávat, že by se Vaše případně důvěrná data dostala někam veřejně na internet.

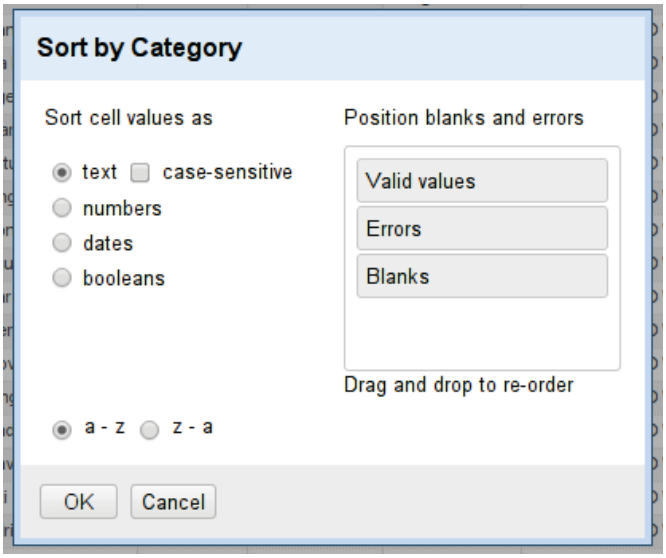
Když již máme vytvořen náš projekt, tak můžeme prozkoumat jak naimportovaná data, tak i samotné rozhraní OpenRefine. Na první pohled se Vám může zdát ovládání velmi odlišné od toho, na které jste zvyklí např. z Excelu, ale jakmile jej začnete používat, rychle si na něj zvyknete.

**Krok 2: Řazení dat**

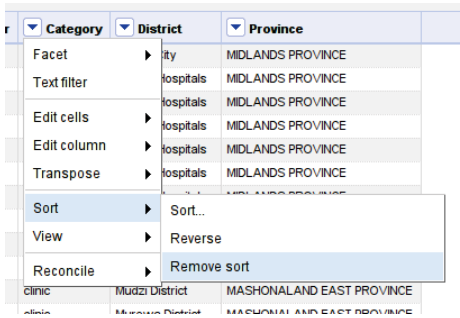
Velmi častým úkolem je řazení a filtrování dat. Chceme například najít minima, maxima nebo informace o kategoriích v datech.

1. OpenRefine pracuje s daty velmi podobně, jako jsme zvyklí například z Excelu. Máme zde řádky, sloupce a buňky. Buňka je definována svým řádkem a sloupcem.
2. Pokud budeme chtít seřadit řádky na základě konkrétního sloupce (v našem případě „Category“), klikneme v záhlaví tohoto sloupce na malý černý trojúhelník a vybereme z nabídky ***Sort...***, který nám otevře dialog pro řazení záznamů.



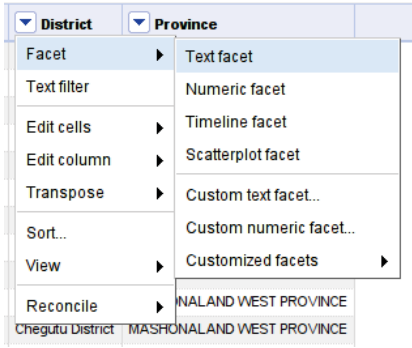

3. V tomto dialogu můžeme vybrat jak se má prořazení OpenRefine na buňky dívat (my použijeme řazení jako text) a také jestli budeme chtít data seřadit vzestupně nebo sestupně. Jakmile klikneme na tlačítko ***OK***, tak se nám záznamy setřídí.
4. Pokud budeme chtít setřídění zrušit, tak ze stejné nabídky ***Sort..*** u sloupce „Category“ vybereme ***Remove sort***.



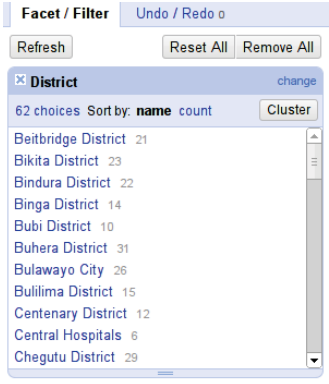

**Krok 3: Práce s facety**

Velmi častým úkolem je filtrovat, třídit data. K tomu jsou v OpenRefine určeny tzv. facety. Jedná se o velmi silný nástroj, který budeme používat i na mnoha dalších místech.

1. Klikneme na malý černý trojúhelník v záhlaví sloupce, který budeme chtít zkoumat (v našem případě „District“) a vybereme z nabídky ***Facet***. Z nabídky pak můžeme vybrat datový typ našeho sloupce, my zde vybereme ***Text facet***, protože chceme zkoumat textové řetězce.




2. Po tomto výběru se nám v levé části prohlížeče zobrazí okno s nadpisem „District“, v kterém uvidíme všechny řetězce, které se v našem sloupci vyskytují.



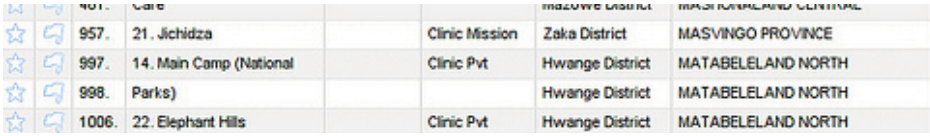

3. Po tomto výběru se nám v levé části prohlížeče zobrazí okno s nadpisem „District“, v kterém uvidíme všechny řetězce, které se v našem sloupci vyskytují.

4. Nyní můžete vybrat jednu či více možností (facetů) a uvidíte, jak se Vám data v hlavním okně budou filtrovat na základě Vašeho výběru.
5. Těchto oken s facety můžete přidat více (v našem případě můžeme přidat další facety např. nad sloupcem „Category“, „Province“ atd.). Pro filtrování pak můžeme tyto facety kombinovat a filtrovat tak data na základě více sloupců.

**Krok 4: Eliminace prázdných buněk**

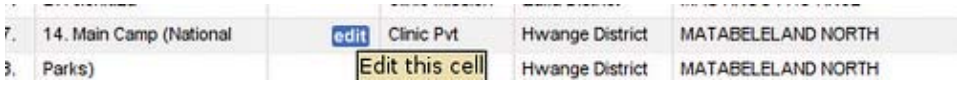
Pokud se podrobněji podíváme na vytvořený facet například pro sloupec „Owner“, tak uvidíte, že na konci seznamu máte na výběr řetězec „(blank)“. Jedná se právě o prázdné hodnoty, s kterými si musíme nějak poradit, abychom měli data konzistentní.

1. Vyberte hodnotu „(blank)“ pro facet vytvořený nad sloupcem „Owner“.
2. Když se podíváme na zobrazené záznamy, tak uvidíme, že zde vznikla v několika případech chyba při rozdělování buněk a tím pádem „Owner“ skončil ve sloupci „Category“.



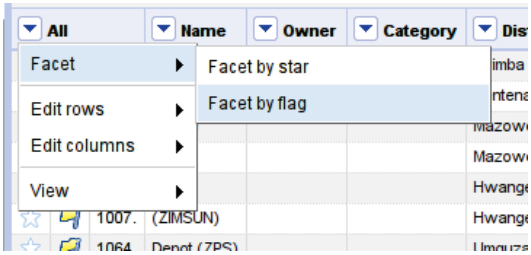
| | | | | | |
|---|---|---|---|---|---|
| 957. | 21. Jichidza | | Clinic Mission | Zaka District | MASVINGO PROVINCE |
| 997. | 14. Main Camp (National | | Clinic Pvt | Hwange District | MATABELELAND NORTH |
| 998. | Parks) | | | Hwange District | MATABELELAND NORTH |
| 1006. | 22. Elephant Hills | | Clinic Pvt | Hwange District | MATABELELAND NORTH |

3. Těchto několik chyb opravíme ručně tak, že najedeme myší do prázdné buňky ve sloupci „Owner“ a klikneme na tlačítko ***edit***. Do textového pole vepíšeme správnou hodnotu ze sloupce „Category“. Nezapomeňte také opravit příslušnou buňku ve sloupci „Category“.

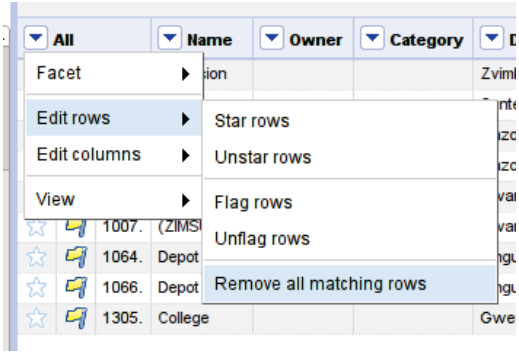



4. Zbytek záznamů, které mají prázdný sloupec „Owner“ již nedávají smysl (chyby v původním souboru, můžete si je porovnat se zdrojovým souborem) a můžeme je považovat za chybná. Tyto chybné záznamy si označíme příznakem ***flag*** tím, že klikneme na symbol vlaječky u každého záznamu. Při výběru prázdného facetu stejného cíle dostaneme při kliknutí na ***All -> Edit rows -> Flag rows***.
5. Stejný postup z bodů 1 až 4 zopakujeme i pro sloupec „Category“. V kategoriích vznikla na několika místech chyba spojením se sloupcem „Name“.
6. Dříve než vymažeme vlaječkou označené záznamy, tak si zkontrolujte, zdali jsme v řádkovém módu ***row mode***. To zjistíte tak, že nad záhlavím sloupců bude tučně vyznačeno ***rows***.


7. Rozbalíme nabídku nad sloupečkem ***All*** a vybereme ***Facet -> Facet by flag***.



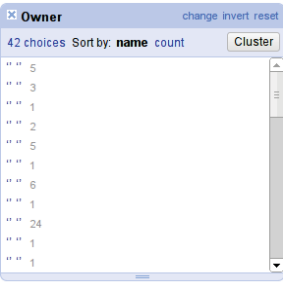

8. Nyní ve facetu na levé straně vybereme ***true***.
9. Nyní provedeme samotné vymazání tak, že v nabídce nad sloupečkem ***All*** vybereme ***Edit rows*** a zde zvolíme ***Remove all matching rows***.




**Krok 5: Eliminace netisknutelných znaků**

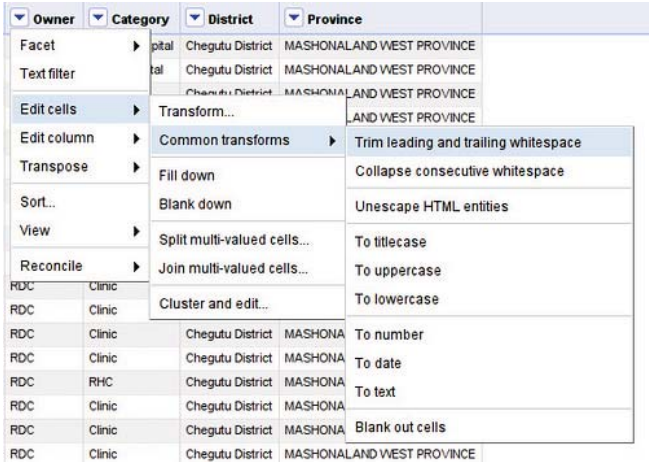
Bílé, netisknutelné znaky (mezera, tabulátor, znak nového řádku) v datech jsou vždy problém. Vzhledem k tomu, že to je velmi častý jev, tak OpenRefine nabízí speciální funkce pro odstranění nechtěných bílých mezer.

1. Začneme čistit sloupec „Owner“
2. Vytvoříme textový facet nad sloupcem „Owner“
3. Vidíme, že vytvořený facet začíná dlouhým seznamem, kde jsou dvojice uvozovek. Ačkoliv vypadají všechny stejně, není tomu tak. Každá totiž obsahuje rozdílný počet mezer mezi uvozovkami.

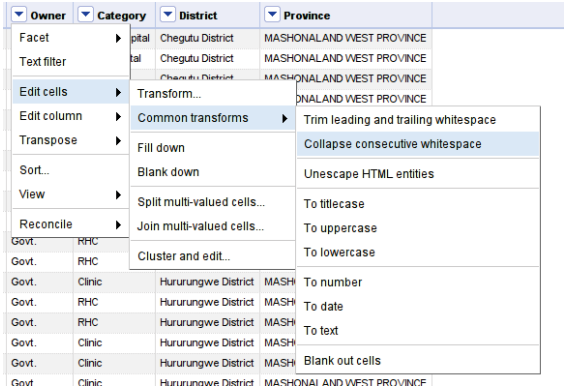



4. Když se v seznamu podíváme trochu níže tak uvidíme, že jsou zde některé záznamy vícekrát, i když vypadají stejně. Například jsou zde dva záznamy pro „Municipality“, protože jedna z nich má mezeru na konci textu.

5. Nyní odstraníme bílé mezery ze začátku a z konce textů ve sloupci „Owner“. Provedem to tak, že rozbalíme menu nad sloupcem „Owner“ a vybereme ***Edit Cells -> Common Transforms -> Trim leading and trailing whitespaces***.




6. Když se nyní podíváme do seznamu facetů na záznam „Municipality“, tak uvidíme, že je zde již pouze jednou.
7. Nyní eliminujeme seznam mezer ze začátku seznamu. Uděláme to tak, že rozbalíme menu nad sloupcem „Owner“ a vybereme ***Edit Cells -> Common Transforms -> Collapse consecutive Whitespaces***.



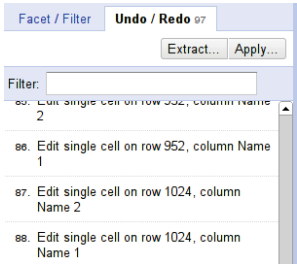

8. Nyní v seznamu facetů vidíme, že se nám mezery zredukovaly pouze na dva záznamy.

9. Nyní již seznam facetů pro sloupec „Owner“ je pročištěn a vypadá mnohem lépe. Zkuste si nyní kroky 2–8 aplikovat znovu i pro ostatní sloupce.

**Poznámka: Historie operací**

Když jsme prováděli tolik transformací, tak Vás může právem napadnout otázka, co když jsem udělal někde chybu? Když pracujete s daty, tak je dobré si někam zapisovat, jaké operace, transformace změny jsme nad daty prováděli. Vzhledem k tomu, že OpenRefine je již od začátku navržený jako nástroj pro zpracování dat, tak sám uchovává všechny změny, které jsme nad daty provedli.

Podívejte se do záložky ***Undo / Redo***, uvidíte zde všechny kroky, které jste doposud udělali. V případě že provedete nějaký krok, který byste chtěli vrátit zpět, tak zde jednoduše klikněte na libovolný krok z tohoto seznamu a tím zrušíte všechny změny, které jste prováděli po něm.

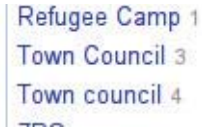



Celou historii lze jednoduše pomocí tlačítka ***Extract...*** vyexportovat ve formátu JSON. Tento textový soubor si můžete uschovat pro pozdější použití či archivaci kroků, které jste prováděli nad daty. Pokud budete někdy v budoucnu muset zpracovat podobná data, tak je pouze stačí naimportovat do OpenRefine a zde v ***Undo / Redo*** kliknout na ***Apply...*** a vložit tento uschovaný JSON soubor. Všechny operace, které jste dříve prováděli nad daty, se pak provedou i nad daty novými.

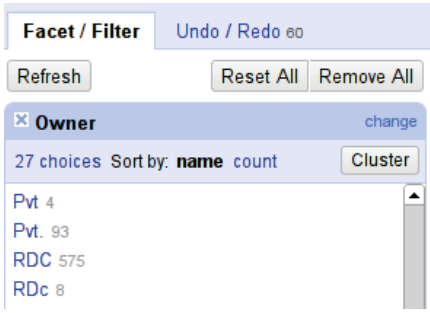
**Krok 6: Sjednocení kategorií**

Když se podíváme na facet pro sloupec „Owner“ tak zjistíte, že zde máme stále některé řetězce vícekrát, i když představují ten samý význam. To samé uvidíme i ve facetu pro sloupec „Category“. Nyní tedy zkusíme sjednotit dohromady kategorie, které mají stejný význam.

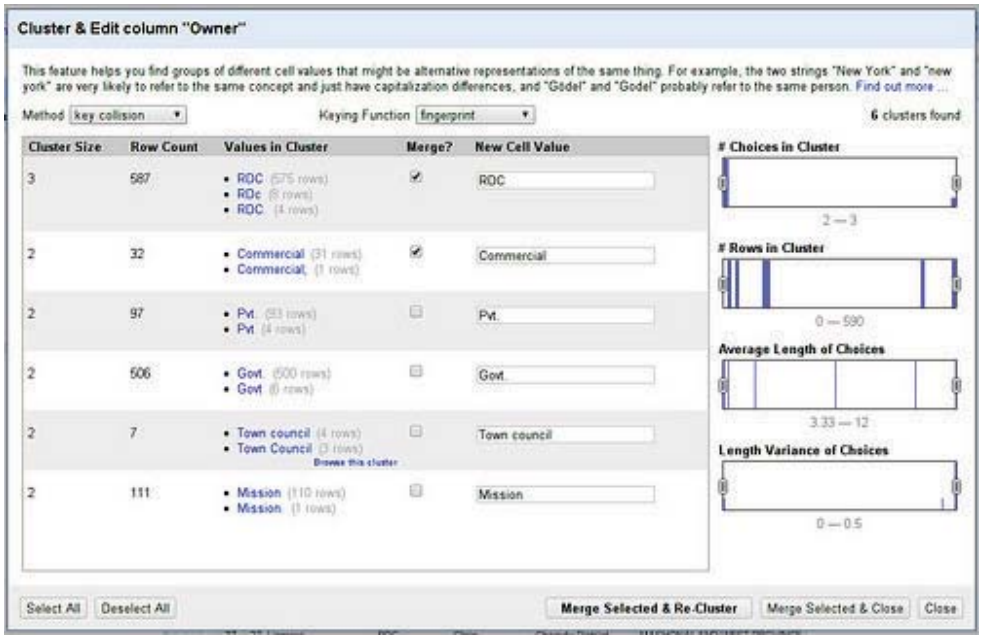
1. Vytvořme facet pro sloupec, který budeme chtít sjednocovat, v našem případě pro sloupec „Owner“.
2. Prvním krokem bude sjednotit kategorie tak, aby používaly stejným způsobem malé a velké znaky. Například kategorie „Town Council“ and „Town council“ se liší pouze v jednom znaku.




3. OpenRefine nám umožňuje pomocí tzv. clusteringu „Clustering“ najít automaticky kategorie, které by měly představovat jednu kategorii. K aktivaci tohoto nástroje stačí, když kliknete na tlačítko ***Cluster*** ve facetu.



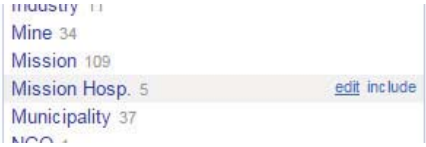

4. Objeví se Vám nové okno, které obsahuje všechny nástroje, které jsou k dispozici pro clustering. OpenRefine je dostatečně chytrý a nabídne Vám, které kategorie by měly patřit k sobě.




5. Zaklikněte checkbox ***merge*** pro kategorie, které budete chtít sjednotit. Jakmile budete mít zaškrtnuty všechny kategorie, které chcete sjednotit, tak samotné sjednocení provedete kliknutím na tlačítko ***Merge selected & Re-Cluster***.
6. Pokud Vám OpenRefine nenabídne žádne možnosti, jak kategorie sjednotit, tak můžete zkusit změnit ***Keying Function*** (výběr metody[16], které se použijí pro zjištění, jestli se jedná o tu samou kategorii) a uvidíte, jestli Vám OpenRefine nabídne další možnost jak sloučit kategorie. Pokud Vám již nenabídne další možnost, tak dialogové okno zavřeme pomocí tlačítka ***Close***.
7. Nyní se ve facetu podívejme na kategorii „Mission“, máme zde dva výskyty „Mission“ a „Mission Hosp.“, které OpenRefine sám automaticky neodhalil, ale pro

[16] Metody které jsou pro clustering k dispozici jsou podrobně popsány na adrese https://github.com/OpenRefine/OpenRefine/wiki/Clustering-In-Depth#methodologies

nás představují ten samý význam. Změníme je tedy ručně na jednu kategorii „Mission“ a to tak, že najedete myší na „Mission Hosp.“ a kliknete na odkaz ***edit*** a zde řetězec upravíme na „Mission“. Podobně můžeme kategorie ručně doopravit tam kde to bude potřeba.

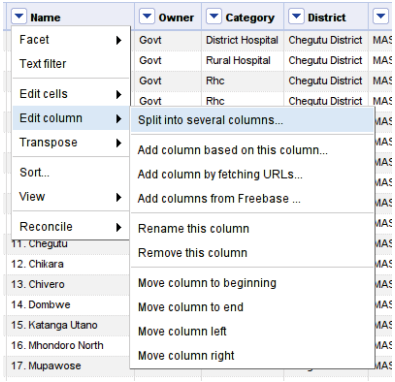



8. Zopakujte kroky 1-7 i pro sloupec „Category“.

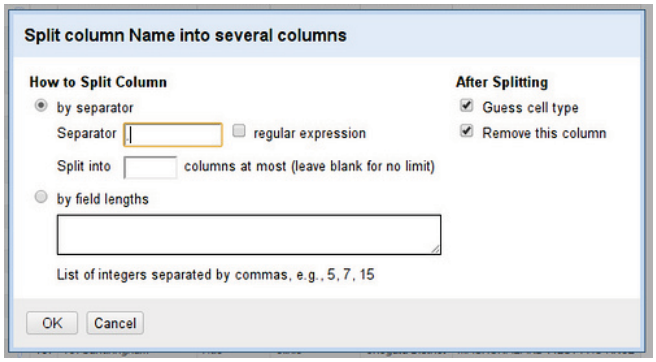
**Krok 7: Rozdělení sloupce**

Když se podíváte na sloupec „Name“ v našem datasetu, tak zjistíte, že název ve většině případů začíná číslem (v tomto případě se jedná o pořadí jednotlivých nemocnic v daném regionu), které pro naše účely nemá žádný význam. Zkusíme je tedy odstranit.

1. Pro rozdělení sloupce „Name“ vyberte z menu nad sloupcem ***Edit Column*** a pak ***Split into several columns***.




2. My nyní rozdělíme sloupec podle znaku „.“ vzhledem k tomu, že čísla v našem případě končí tečkou. Vepíšeme tedy znak „.“ do políčka ***Separator*** v dialogovém okně, které se nám otevřelo. Vzhledem k tomu, že chceme sloupec rozdělit pouze do dvou nových sloupců, tak vepíšeme číslici „2“ do políčka vedle textu ***Split into***, takže celá věta zde bude ***Split into 2 columns at most***.



3. Klikneme na tlačítko ***OK*** a sloupec „Name“ se nám změnil na dva sloupce „Name 1“ a „Name 2“.
4. V několika případech toto rozdělení neproběhlo úplně správně. Jednoduše tedy vytvoříme facet na sloupci „Name 2“, vybereme řetězec „(blank)“ a ručně opravíme chybné záznamy.

Pomocí výše uvedených kroků můžeme zpracovávaný dataset ještě vyčistit od dalších drobností, jak je například neznámá hodnota ve sloupcích „Owner“ a „Category“, sloupec „Province“ upravit tak, aby nebyl kapitálkami apod. Výsledný zpracovaný dataset si můžete prohlédnout a případně načíst do OpenRefine ze souboru `Provincial-and-District-Hospitals-FINAL.tsv` který je možné stáhnout z adresy `https://github.com/MiroslavKubasek/OpenRefine-tutorial` v adresáři `data`.

## 5.2. Získání dat z PDF souborů

Ne vždy jsou data dostupná v pěkně nachystaných datových souborech CSV, EXCEL či JSON. Někdy jsou data, která chceme použít, uložena v souborech ve formátu PDF. Získat data ze souborů PDF, které jsou určeny primárně pro tisk, je velmi náročný úkol, někdy skoro až nemožný. Můžete zkusit zkopírovat PDF jako text a vložit např. do Excelu, někdy bude výsledek použitelný, ale většinou ne a je potřeba použít hodně manuální a mravenčí práce tato data dostat do podoby tak, aby šly dále strojově zpracovávat.

Naštěstí existuje zdarma nástroj Tabula[17], který umožňuje na jedno kliknutí extrahovat tabulky ze souborů PDF a ty pak jednoduše načíst do OpenRefine a zde dále zpracovávat. Instalace programu je jednoduchá, z webu `http://tabula.technology/` si stáhněte zip určený pro Váš operační systém, ten rozbalte na disk a spusťte soubor `tabula.exe` (v případě MS Windows). Po spuštění se Vám objeví příkazová řádka a po chvíli vy se měl spustit automaticky webový prohlížeč na adrese `http://127.0.0.1:8080/`[18].

Pak již jen stačí pomocí tlačítka ***Submit*** nahrát PDF soubor z disku Vašeho počítače, program Tabula PDF zpracuje a nabídne Vám náhled jednotlivých stránek. Pokud se jedná o menší soubor, případně s ne moc složitou strukturou, můžete před odesláním zaškrtnout možnost ***Auto-Detect Tables***.

V náhledu si pak můžete označovat jednotlivé tabulky, které chcete z PDF souboru získat. Jakmile pustíte tlačítko myši, Tabula Vám zobrazí zformátovanou vybranou tabulku. Pokud nejsou data pěkně zformátovaná, zkuste odstranit z výběru záhlaví případně zápatí tabulky.

Pokud jsme s výběrem spokojeni, můžeme si tabulku uložit jako CSV nebo TSV soubor. Případně můžeme data pouze zkopírovat do schránky a vložit je přímo do Excelu nebo OpenRefine.

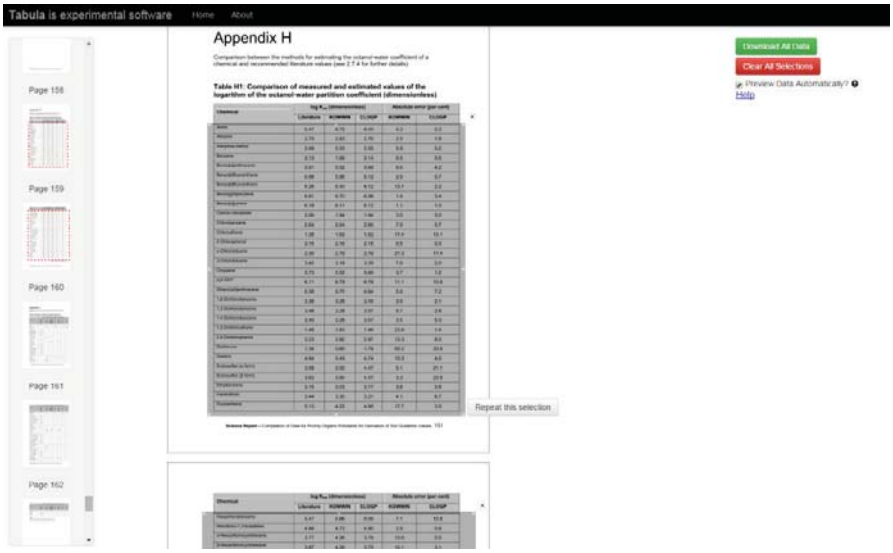
Ukážeme si export dat z PDF souboru na konkrétním příkladu:

---

[17] Program Tabula je dostupný pro MS Windows, Mac i Linux. Pro běh programu je nutné mít na počítači nainstalován běhové prostředí jazyka Java (JRE), viz http://java.com/en/download/

[18] Pokud se webový prohlížeč nespustí sám, tak jej spusťte ručně a tuto webovou adresu do něj vepište.

1. Stáhněte si soubor `scho1008bovo-e-e.pdf` z adresy `https://github.com/MiroslavKubasek/OpenRefine-tutorial` v adresáři `data`[19]
2. Stažený soubor načtěte do programu Tabula (***Vybrat soubor*** a stisknout tlačítko ***Submit***).
3. Chvíli trvá, než program zpracuje PDF soubor. Poté se Vám objeví naáhledy jednotlivých stránek. My budeme exportovat tabulku H1 z přílohy H (Appendix H), která začíná na stránce číslo 158.
4. Nyní myší označíme data v tabulce H1 na straně 158, zobrazí se nám náhled dat. Pokud jsou data v pořádku, klikneme „Close“ a označíme stejným způsobem data v tabulce na následující straně 159. Také po vizuální kontrole zavřeme náhled dat.

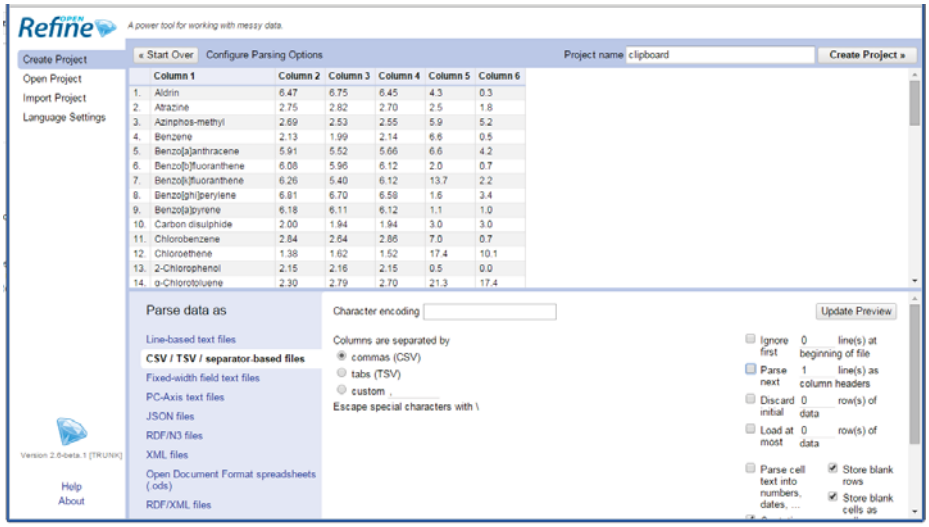



5. Nyní máme v PDF náhledech označeny šedým obdélníkem dvě oblasti s daty na stranách 158 a 159. Teď klikneme na tlačítko ***Download All Data***. Zde můžeme data uložit v běžných formátech, ale my data zkopírujeme do schránky přes tlačítko ***Copy to Clipboard***.

[19] Originální soubor je umístěn na adrese https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/291006/scho1008bovo-e-e.pdf


6. Otevřeme aplikaci OpenRefine, zvolíme ***Create project*** a z nabídky vybereme ***Clipboard*** data pomocí kombinace kláves „CTRL + V“ data vložíme do textového políčka a klikneme na tlačítko ***Next »***.


7. Zobrazí se nám známí obrazovka importu dat, kde je potřeba vybrat z nabídky druhou možnost importu ***CSV / TSV / separator-based files*** a je potřeba odškrtnout možnost ***Parse next 1 line(s) as column headers***.

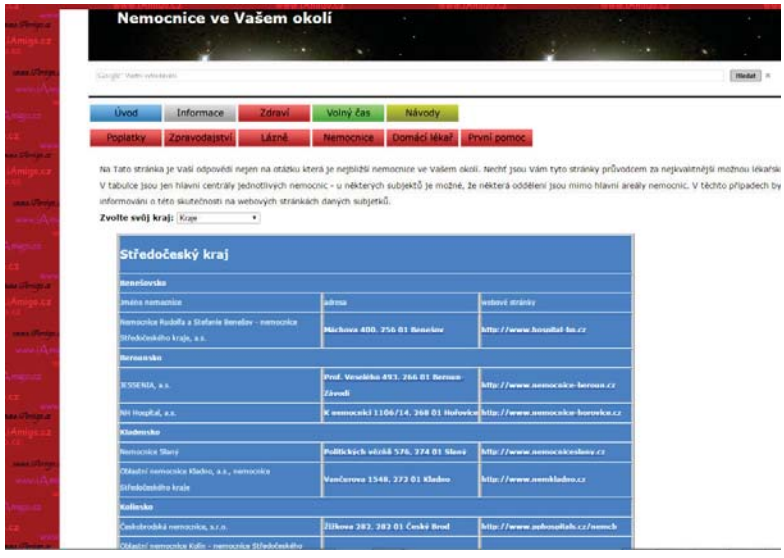



8. Pojmenujeme projekt v políčku ***Project name*** a klikneme tlačítko ***Create Project »***. Data se nám načtou a už nám jen zbývá pojmenovat podle originálního pdf vhodně záhlaví jednotlivých sloupců.

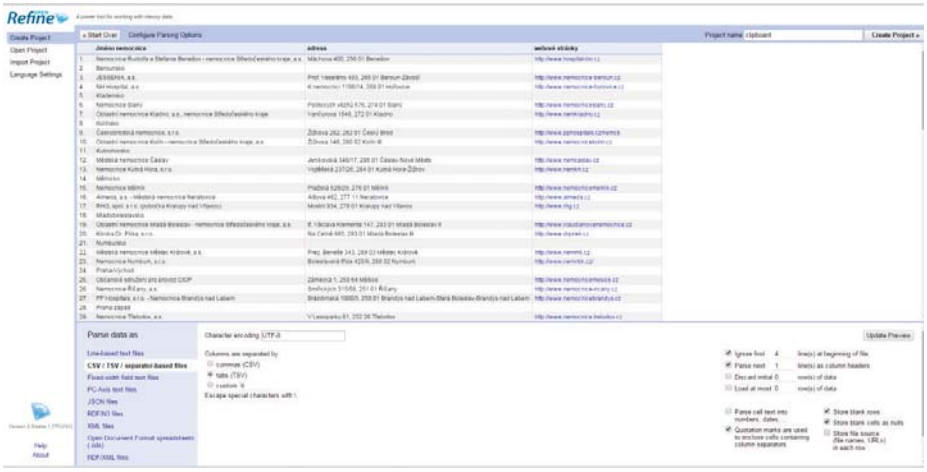
## 5.3. Získání dat z webu, využití API pro doplnění dat a mapová vizualizace

V tomto příkladu použijeme veřejně dostupný seznam nemocnic na webové adrese `http://www.iamigo.cz/nemocnice.htm` který obsahuje název nemocnice, její adresu a web nemocnice. Tyto data doplníme pomocí Google maps API o geografické souřadnice jednotlivých nemocnic a výsledná data vizualizujeme v interaktivní mapě.

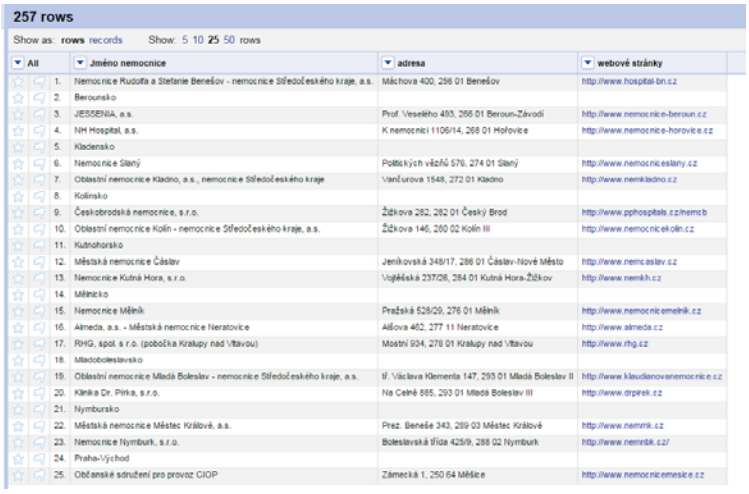
1. Na webové stránce `http://www.iamigo.cz/nemocnice.htm` označíme pomocí myši celou tabulku se seznamem nemocnic a tento výběr zkopírujeme do schránky (CTRL + C).



2. Otevřeme OpenRefine, zvolíme ***Create project*** a z nabídky vybereme ***Clipboard***, pomocí kombinace kláves „CTRL + V“ data vložíme do textového políčka a klikneme na tlačítko ***Next »***.
3. Zobrazí se nám známí obrazovka importu dat, kde je potřeba vybrat z nabídky druhou možnost importu ***CSV / TSV / separator-based files*** a je potřeba zaškrtnout možnost ***Ignore first 4 line(s) at begining of file*** a nechat zaškrtlou možnost ***Parse next 1 line(s) as column headers***. Také v ***Character encoding*** zvolte „UTF-8“.
4. Pojmenujeme v políčku ***Project name*** projekt například „Nemocnice v ČR“ a klikneme na tlačítko ***Create Project »***.



5. Tím vytvoříme v OpenRefine nový dataset se 257 záznamy, které budeme muset nyní pročistit tak, aby nám v datasetu zůstali pouze záznamy s nemocnicemi.

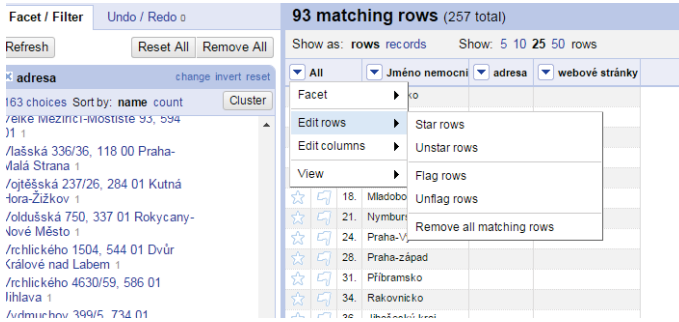


257 rows

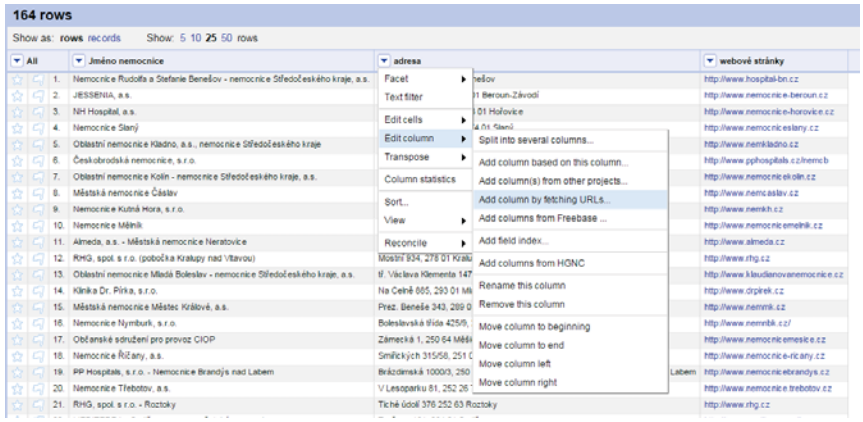
Show as: **rows** records Show: 5 10 **25** 50 rows

| All | | Jméno nemocnice | adresa | webové stránky |
|---|---|---|---|---|
| | 1. | Nemocnice Rudolfa a Stefanie Benešov - nemocnice Středočeského kraje, a.s. | Máchova 400, 256 01 Benešov | http://www.hospital-bn.cz |
| | 2. | Berounsko | | |
| | 3. | JESSENIA, a.s. | Prof. Veselého 493, 266 01 Beroun-Závodí | http://www.nemocnice-beroun.cz |
| | 4. | NH Hospital, a.s. | K nemocnici 1106/14, 268 01 Hořovice | http://www.nemocnice-horovice.cz |
| | 5. | Kladensko | | |
| | 6. | Nemocnice Slaný | Politických vězňů 576, 274 01 Slaný | http://www.nemocniceslany.cz |
| | 7. | Oblastní nemocnice Kladno, a.s., nemocnice Středočeského kraje | Vančurova 1548, 272 01 Kladno | http://www.nemkladno.cz |
| | 8. | Kolínsko | | |
| | 9. | Českobrodská nemocnice, s.r.o. | Žižkova 282, 282 01 Český Brod | http://www.pphospitals.cz/nemcb |
| | 10. | Oblastní nemocnice Kolín - nemocnice Středočeského kraje, a.s. | Žižkova 146, 280 02 Kolín III | http://www.nemocnicekolin.cz |
| | 11. | Kutnohorsko | | |
| | 12. | Městská nemocnice Čáslav | Jeníkovská 348/17, 286 01 Čáslav-Nové Město | http://www.nemcaslav.cz |
| | 13. | Nemocnice Kutná Hora, s.r.o. | Vojtěšská 237/26, 284 01 Kutná Hora-Žižkov | http://www.nemkh.cz |
| | 14. | Mělnicko | | |
| | 15. | Nemocnice Mělník | Pražská 528/29, 276 01 Mělník | http://www.nemocnicemelnik.cz |
| | 16. | Almeda, a.s. - Městská nemocnice Neratovice | Alšova 462, 277 11 Neratovice | http://www.almeda.cz |
| | 17. | RHG, spol. s r.o. (pobočka Kralupy nad Vltavou) | Mostní 934, 278 01 Kralupy nad Vltavou | http://www.rhg.cz |
| | 18. | Mladoboleslavsko | | |
| | 19. | Oblastní nemocnice Mladá Boleslav - nemocnice Středočeského kraje, a.s. | tř. Václava Klementa 147, 293 01 Mladá Boleslav II | http://www.klaudienovanemocnice.cz |
| | 20. | Klinika Dr. Pirka, s.r.o. | Na Celně 885, 293 01 Mladá Boleslav III | http://www.drpirek.cz |
| | 21. | Nymbursko | | |
| | 22. | Městská nemocnice Městec Králové, a.s. | Prez. Beneše 343, 289 03 Městec Králové | http://www.nemmk.cz |
| | 23. | Nemocnice Nymburk, s.r.o. | Boleslavská třída 425/9, 288 02 Nymburk | http://www.nemnbk.cz/ |
| | 24. | Praha-Východ | | |
| | 25. | Občanské sdružení pro provoz CIOP | Zámecká 1, 250 64 Měšice | http://www.nemocnicemesice.cz |

6. Vytvoříme textový facet nad sloupcem „adresa“ ***Facet -> Text facet*** a ve vytvořeném facetu vybereme poslední položku „(blank)“ (to jsou řádky, které nemají adresu a neobsahují tak informace o nemocnici). Tyto záznamy vymažeme pomocí sloupce „All“ ***Edit rows -> Remove all matching rows***. Poté můžeme facet nad sloupcem „adresa“ zavřít.

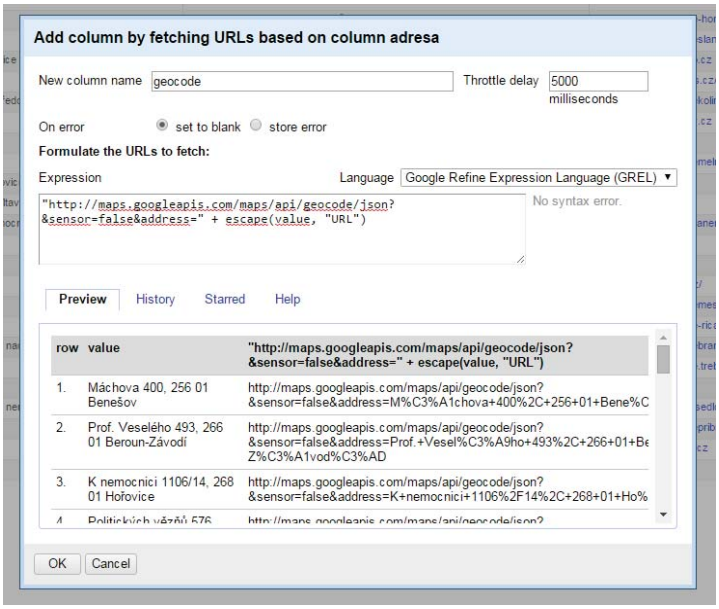



7. Nyní pomocí Google maps API dohledáme souřadnice jednotlivých nemocnic. Provedeme to tak, že pro sloupec „adresa“ vybereme z menu ***Edit column -> Add column by fetching URLs...***.

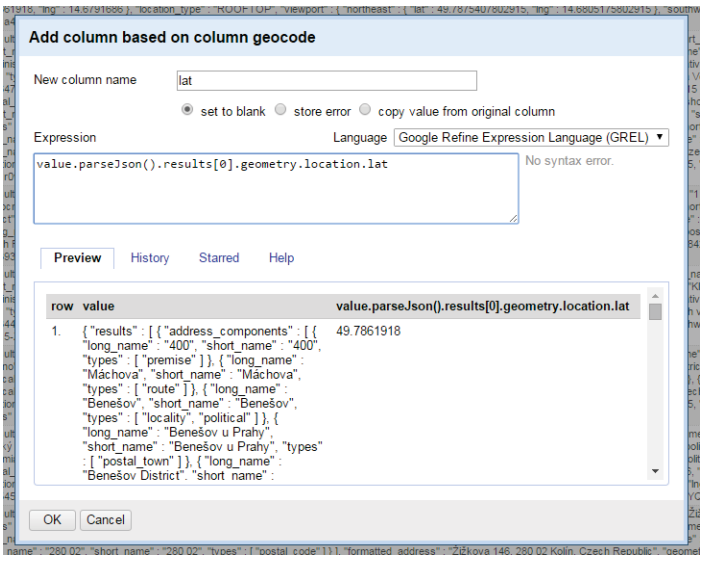



8. Zobrazí se nám dialogové okno, kde do políčka ***Expression*** vepíšeme následující řetězec:
`"http://maps.googleapis.com/maps/api/geocode/json?&sensor=false&address=" + escape(value, "URL")`[20] do políčka ***New column name*** vepíšeme „geocode“, a v políčku ***Throttle delay*** necháme „5000“ milisekund. Klikneme na tlačítko ***OK*** a necháme program, aby nám stáhl z Google maps API data (budeme se dotazovat Google API na 164 adres, což dle rychlosti připojení zabere přibližně 15 minut.).

[20] Lze použít alternativně podobnou službu pomocí kódu http://nominatim.openstreetmap.org /search? + "format=json&" + "q=" + escape(value, "url")



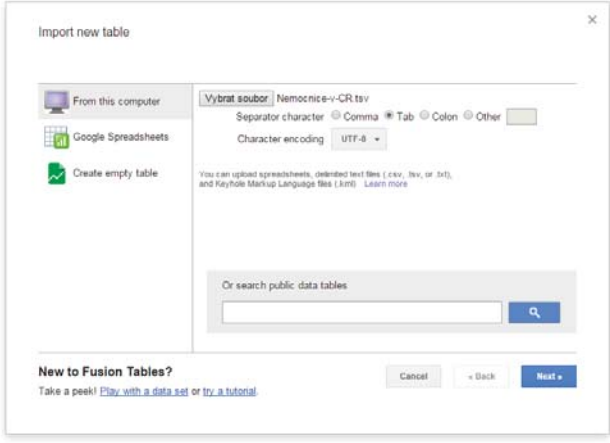

9. Jakmile budou data načtena, tak je potřeba z nich získat geografické souřadnice (lat, lng) jednotlivých nemocnic. Data vrácená z API jsou ve formátu JSON, které prvně musíme parsovat a pak získat data která potřebujeme. Vybereme z menu nad sloupcem „geocode“ nabídku ***Edit column -> Add column based on thic column...***. Zde v dialogovém okně do políčka ***Expression*** vepíšeme následující příkaz v jazyce GREL `value.parseJson().results[0].geometry.location.lat`, do políčka ***New column name*** vepíšeme řetězec „lat“ a klikneme na tlačítko ***OK***.



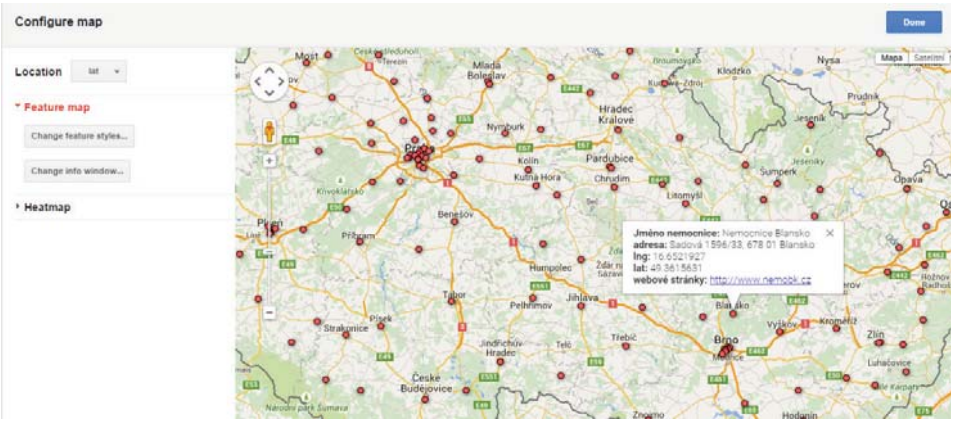

10. Vedle sloupečku „geocode“ se nám objevil nový sloupec s názvem „lat“ který obsahuje část souřadnice adresy. Ten samý postup z bodu 9 uděláme i pro souřadnici s názvem „lng“.

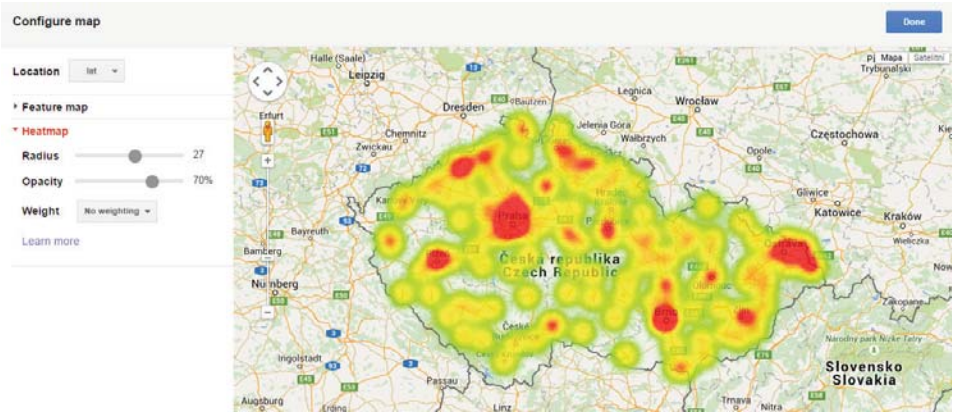
11. Jakmile máme obě souřadnice k dispozici, tak již sloupec „geocode“ nepotřebujeme a můžeme jej odstranit („geocode“ nabídka ***Edit column -> Remove thic column***.
12. Výsledkem budou data, která kromě názvu nemocnice, její adresy a webu budou obsahovat i geografické souřadnice („lng“ a „lat“) těchto adres. Výsledný dataset vyexportujte pomocí tlačítka ***Export -> Tab-separed value*** a uložte na Váš disk. Výsledný zpracovaný dataset si můžete prohlédnout a případně načíst do OpenRefine ze souboru `Nemocnice-v-CR.tsv` který je možné stáhnout z adresy `https://github.com/MiroslavKubasek/OpenRefine-tutorial` v adresáři `data`.


13. Nyní je potřeba, abyste měli zřízen účet na www.google.com. Přejděte na stránku `https://drive.google.com/`, klikněte na tlačítko ***Přidat*** a zde vyberte ***Dynamické tabulky Google***. V dialogu, který se Vám otevře vyberte z disku vyexportovaný soubor z předchozího kroku. Jako ***Separator character*** vyberte „Tab“ a ***Character encoding*** „UTF-8“ a klikněte na tlačítko ***Next***. Dále pak klikněte ještě jednou na ***Next*** a pak na ***Finish***.




14. Google fusion tables automaticky z našich dat rozpoznal, že se jedná o souřadnice a v záložce ***Map*** nám vygeneroval mapu včetně možností kliknutí na bod se zobrazením detailu. V Google fusion tables můžete v případě potřeby upravit, jak má vypadat okno, které se zobrazí po kliknutí na bod v mapě. Jsou zde také bohaté možnosti, jak naformátovat samotný mapový výstup.




15. Vytvořenou mapu můžete dále sdílet pomocí nabídky ***Tools -> Publish...***, kde je na výběr několik možností, jak výslednou mapu nasdílet.

## 6. Použitá literatura

Verborgh R, Wilde MD. Using OpenRefine. 2013. ISBN: 9781783289080

Stephens O. Introduction to OpenRefine, 2014.
`http://www.meanboyfriend.com/overdue_ideas/wp-content/uploads/2014/11/Introduction-to-OpenRefine-handout-CC-BY.pdf`

Atima, Zhuang H, Vedvyas I, Dole R. Tutorial: OpenRefine. 2014.
`http://casci.umd.edu/wp-content/uploads/2013/12/OpenRefine-tutorial-v1.5.pdf`

`http://www.openrefine.cz/`

`http://openrefine.org/`