OpenRefine

Fernando Anselmo
http://fernandoanselmo.orgfree.com/wordpress/

Versão 1.0 em 11 de julho de 2020

Resumo

OpenRefine [1] é um aplicativo gratuito e de código aberto para manipular todos os tipos de arquivos de dados. Escrito em linguagem Java, roda em qualquer sistema operacional através do navegador. Criado para obter rapidamente uma visão geral do conteúdo de um conjunto de dados, resolver inconsistências e aprimorá-lo com outros dados - tudo de maneira visual, interativa e eficiente. Pense nele como um editor de Planilhas voltado exclusivamente para o tratamento de dados.

1 Parte inicial

Quase todos os conjuntos de dados que encontramos estão sujos e bagunçados. Existem inconsistências na maneira como os dados são inseridos - que vão desde erros de ortografia, duplicidades a espaços irregulares - que podem dificultar a análise posterior.

Limpar os dados é uma tarefa que leva um considerado tempo, mas tem que ser realizada de qualquer forma antes de tentar usá-los. **OpenRefine** nasceu com o objetivo de realizar esse trabalho sujo, realiza uma infinidade de tarefas para limpeza, organização e melhora da informação.

Figura 1: *Logo do OpenRefine*

Mas tudo isso não poderia ser feito com o **MS-Excel** ou **Calc** (**LibreOffice**)? Diria que talvez sim se sua massa for pequena, mas tente trabalhar com Gigas ou Teras de informação e verá que esses programas não lhe atenderão mais. Por ser extremamente leve e portátil, **OpenRefine** é a solução ideal para qualquer tipo de trabalho.

Seu início foi como um projeto chamado **Freebase Gridworks**, que foi comprado pela **Google** e reconstruído como **GoogleRefine** em 2010. O suporte oficial terminou em 2012 e houve uma

transição para o projeto de código aberto denominado **OpenRefine**. Se olharmos bem **Google-Refine** e **OpenRefine** é o mesmo aplicativo, muitos tutoriais e documentação usam os nomes de forma intercambiável.

Casos que devemos usar:

- Limpeza - descobrir, corrigir inconsistências e agrupamentos.
- Transformar - alterar formatos, remodelar com divisão ou união de colunas com vários valores, transposição de colunas ou linhas.
- Estender - enriquecer os dados combinando-os a arquivos, mesclando projetos, buscando URLs ou com bancos de dados on-line.
- Automatizar - podemos reutilizar sua rotina de processamento exportando o histórico de operações em JSON e criar uma rotina programável.

Este é um programa extremamente flexível, qualquer formato tabular, tais como, planilhas, bancos de dados, XML, RDF, matrizes ou JSON pode ser visualizado. Além disso, foi projetado para ser extensível, a comunidade criou vários *plugins* e extensões especializadas.

OpenRefine não apenas permite diagnosticar rapidamente a precisão de seus dados, mas também a agir em certos tipos de erros, muitas vezes de maneira automatizada. Podemos identificar facilmente erros sistemáticos, tais como, células em branco, duplicatas, inconsistências ortográficas entre outros.

2 OpenRefine no Docker

Facilmente podemos obter o **OpenRefine** através do Docker. Criar uma pasta para associar ao contêiner onde colocaremos nossos dados:

```
$ mkdir $HOME/openrefine_home
```

Fornecer permissões a pasta de modo que o contêiner possa acessá-la:

```
$ chown 1000 $HOME/openrefine_home
```

Baixar a imagem disponível:

```
$ docker pull vimagick/openrefine
```

Criar o container:

```
$ docker run -d --name meu-openrefine -p 3333:3333 -v $HOME/openrefine_home:/data:z
vimagick/openrefine
```

Para executar abrir um navegador e acessar a URL `http://localhost:3333`.

2.1 Para atualizar o OpenRefine no Docker

Quando existir uma nova versão disponível, podemos atualizar diretamente no contêiner do Docker. Contêiner possui um conceito de dinamicidade, ou seja, são atualizáveis e basicamente a única diferença é que devemos entrar nele para proceder isso.

Entrar no container:

```
$ docker exec -u 0 -it meu-openrefine bash
```

Ir para o local correto:
`$ cd /app`

Baixar e instalar:
```
$ set -xe && apk add --no-cache bash curl tar && curl -sSL https://github.com/
OpenRefine/OpenRefine/releases/download/[versão]/openrefine-linux-[versão].tar.gz |
tar xz --strip 1
```

Sair do bash:
`$ exit`

Reiniciar o container:
`$ docker restart meu-openrefine`

E no navegador teremos a nova versão:

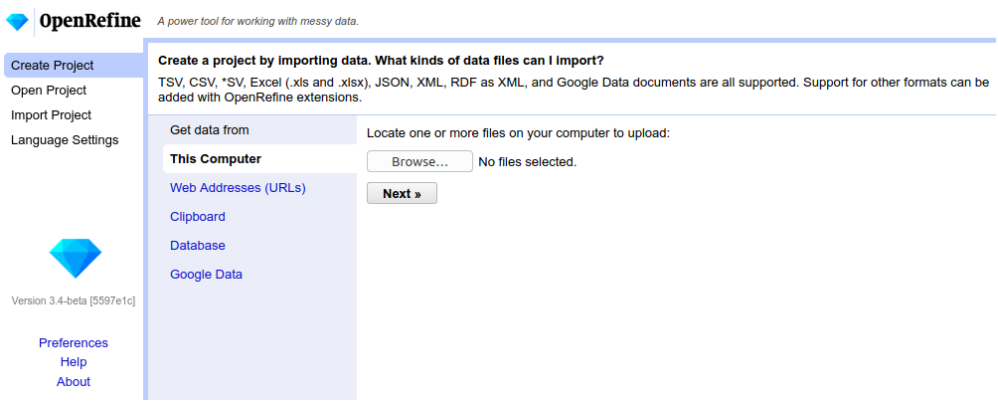


Figura 2: *Versão atual do OpenRefine*

3 Passos Iniciais

No menu a esquerda temos as opções:

- **Criar um Projeto** - que permite importar uma base de dados de um arquivo ou de um banco que graças a linguagem Java podemos - através de uma conexão **JDBC** - utilizar qualquer banco de dados conhecido, buscá-la de um endereço Web que também (graças a sua passagem na Google) pode ser uma Planilha Google.
- **Abrir um Projeto** - uma vez criado, os projetos ficam a nossa disposição para reusá-los.
- **Importar um Projeto** - abrir um projeto criado por terceiros.
- **Escolher a linguagem** - pessoalmente prefiro mantê-lo em língua inglesa, mas se desejar pode optar pelo português.

Para os nossos exemplos vamos utilizar um arquivo chamado **marketSales.csv** que está disponibilizado em `https://github.com/fernandoans/machinelearning/tree/master/bases`. Baixar o arquivo, clicar em **Create Project** ▷ **This computer**. Pressionar **Browse**. Localizar o arquivo e no retorno pressionar **Next**.

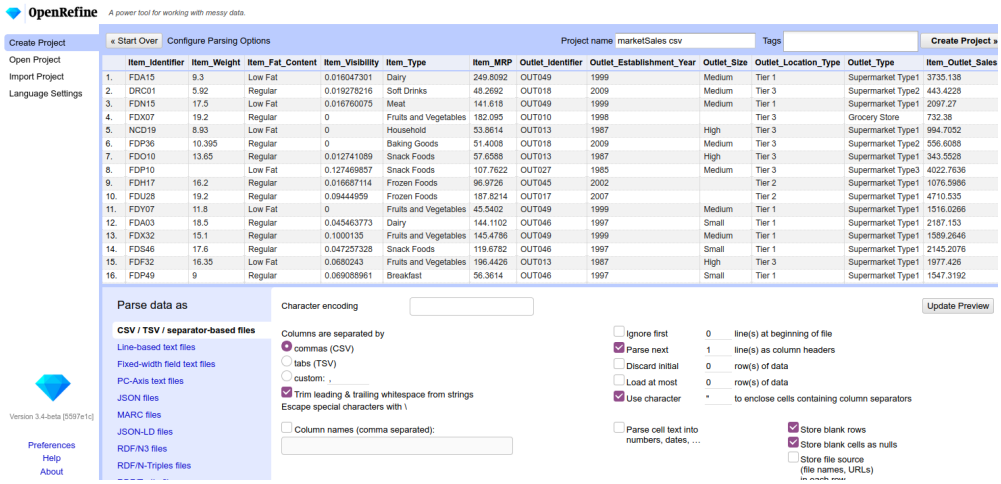


Figura 3: *Dados para Importar*

Essa janela é uma simples conferência de como o OpenRefine visualiza seus dados, podemos realizar os ajustes, caso haja necessidade, tais como, alterar o separador, a partir de qual linha começará a leitura, se linhas em branco devem ser consideradas ou não, o formato correto do arquivo (CSV, JSON, XML, XLS) entre muitas outras opções. Uma vez concluído pressionar **Create Project**.

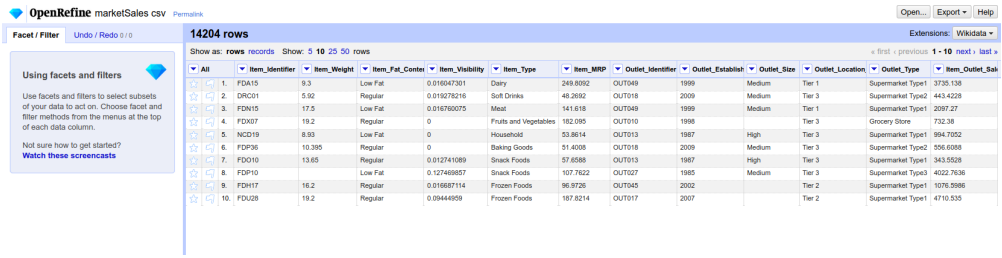


Figura 4: *Dados Importados*

Não devemos nos assustar uma vez que o ambiente de trabalho é o mesmo que qualquer planilha eletrônica, a exceção aqui é o conceito *facet*.

3.1 Facets

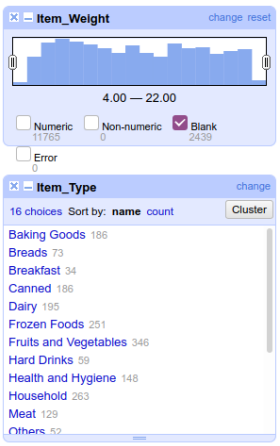
Isso seria algo como um filtro que é aplicado a cada coluna (ou a forma como a enxergaremos nossos dados - isso vem de "faceta" por essa palavra não ser comum adotaremos nesta apostila o termo FILTRO), são extremamente importantes para a visualização. Esses filtros uma vez criados são trabalhados na parte a esquerda.

O tipo da coluna define a ela propriedades únicas, por exemplo duas colunas numéricas são unidas com operações matemáticas, já se forem caracteres muda para concatenação. Sendo assim vamos definir que para as colunas: **Item_Weight**, **Item_Visibility**, **Item_MRP** e **Item_Outlet_Sales** o tipo numérico. Clicar $\triangledown$ da coluna e selecionar **Edit Cells** ▷ **Common transforms** ▷ **To number**.

Errou alguma coluna? Não se desespere, isso aqui é como jogos de *video game*, a cada ação é criado um ponto de controle, na aba *Undo/Redo* (no menu a esquerda) podemos visualizar todas as ações que foram realizadas e a qualquer momento retornar a um ponto de mudança. Isso significa que

iremos perder tudo o que fizemos? Depende, isso varia da quantidade de ações que retornamos. É como um histórico armazenado em memória.

As colunas modificadas ficaram na cor verde o que indica que são numéricas, vamos tentar abrir a coluna **Item_Type** como um filtro numérico. Clicar ▽ desta e selecionar **Facet** ▷ **Numeric Facet**. Obtemos uma janela de erro (no lado esquerdo) com a informação que esta coluna não possui esse tipo.



Tentemos proceder de maneira similar em qualquer uma das colunas que definimos como numéricas. Na esquerda aparece um histograma com o que existe nessa coluna: quantidade de numéricos, não numéricos, brancos e erros. Se desmarcarmos ou marcamos qualquer opção esta ação é refletida na visão dos dados.

Para as tipo texto essa janela aparece de forma diferente, como um filtro de opções mostrando a quantidade de cada valor (fica horrível para uma coluna com valores únicos, como **Item_Identifier**). Podemos ver isso aplicado corretamente. Clicar ▽ de **Item_Type** e selecionar **Facet** ▷ **Text Facet**. Serve para verificarmos como estão os dados, se existem nulos os discrepâncias. Também temos a possibilidade de abrir um histograma para este tipo, opção **Facet by choice counts** que aparece ao final da lista.

3.2 Variáveis

Antes de começarmos a ver qualquer coisa prática temos que ter a noção do que são variáveis para o **OpenRefine**, na verdade para o **GREL** (*General Refine Expression Language*). Existem as básicas que são:

- *row* - objeto contendo a linha atual.
- *cell* - célula específica que está baseado em uma coluna da linha.
- *value* - valor de uma determinada *cell* ou *record*, pode ser nulo.

Devemos nos atentar para que existe uma diferença entre *cell* (célula) e *column* (coluna). Podemos realizar ações para cada uma delas separadamente, por exemplo a função *split* em célula significa criar várias colunas, já em coluna diversas linhas. A transformação (modificação do valor) ocorre na célula nunca na coluna.

Row que se refere a linha atual, possui as seguintes propriedades:

- *row.index* - índice de cada linha (inicia por 0).
- *row.cells* - objeto contendo as células de uma linha.
- *row.columnNames* - lista com o nome das colunas.
- *row.starred* - indica se a linha está marcada com a estrela.
- *row.flagged* - indica se a linha está marcada com a bandeira.

Por exemplo, as expressões:

```
row.cells["Item_Type"].value
cell.value
```

Retornam exatamente o mesmo valor se estivermos na coluna **Item_Type**, se estamos em outra e desejamos obter o valor de **Item_Type** somente a primeira expressão serviria.

Rows (linhas) e *Records* são diferentes, mesmo que aparentem serem iguais, quando realizamos o *split* em uma célula, provavelmente teremos mais linhas que registros, ou seja, o registro não muda. Por exemplo, na coluna **Outlet_Type** desejamos saber quantos supermercados existem pouco importa seu tipo. Ao ativarmos *text facet* veremos que temos 3 tipos, não é necessário fazer contas, vamos separar os valores.

Clicar ▽ de **Outlet_Type** e selecionar **Edit cells** ▷ **Split multi-valued cells...**. Em **Separator** colocar um espaço em branco. E através de um filtro de texto vemos que são 12.399 supermercados. E na informação da janela principal são 28.408 linhas, se marcarmos **records** continuamos com os mesmos 14.204 registros. Ou seja, apenas a visão deles mudou.

3.3 Ordenações

A opção ▽, além dos filtros, temos as possibilidades de ações que podemos proceder, como ordenar por exemplo. A partir do momento que ordenamos por determinada coluna esta se torna o padrão, a próxima coluna que procedemos a mesma ação vira uma espécie de segunda ordem, ou seja, uma ordenação dentro da outra.

Vamos clarear esse conceito, ordenar numericamente **Item_Visibility** (clicar ▽ dessa e selecionar **Sort**, marcar a opção **smallest first**), em seguida ordenar de mesma forma a coluna **Item_Height**. A ordem é: por **Item_Visibility** e para cada valor repetido ordena por **Item_Height**. Além disso, o **OpenRefine** corrigiu todos os menus, clicar ▽ de **Item_Visibility** e selecionar **Sort** que agora possui mais possibilidades como **Reverse** (inverter a ordenação) ou **Remove Sort** (remover esta ordenação). Além disso, acima da planilha de dados aparece a opção **Sort** que possui as seguintes opções: **Remove Sort** (remover todas as ordenações), **Reorder rows permanently** (aplicar esta ordenação de forma permanente) e as opções para cada coluna na ordem que aparecem no filtro.

3.4 Duplicados

Não é o caso de nenhuma coluna neste conjunto, mas vamos imaginar que uma delas deveria ser a chave primária da coleção, ou seja, conter apenas valores únicos, como saber se nela existem termos duplicados? Clicar ▽ da coluna e selecionar **Facet** ▷ **Customized facets** ▷ **Facet by blank** é mostrado dois valores *true* (linhas que contém valores duplicados) e *false* (que não contém), ou seja, basta clicarmos em *true* e veremos que linhas são essas para arrumarmos.

4 Limpeza dos Dados

Para praticarmos um pouco feche este projeto (clicar no símbolo do OpenRefine no canto superior esquerdo) baixar e abrir a planilha **LouisianaSecretaryOfStateOfficials.xls** (disponibilizado no mesmo endereço do GitHub). Essa contém os dados dos Funcionários da Secretaria do Estado de Louisiana nos EUA.

Nossa primeira preocupação é transformar corretamente os dados, para as colunas **Expiration Date** e **Commissioned Date** devemos transformar para data. Clicar ▽ de cada coluna e selecionar **Edit Cells** ▷ **Common transforms** ▷ **To date**. Agora podemos aplicar para essa o *timeline facet*.

Aplicar aos seguintes filtros:

- *timeline facet* das colunas **Expiration Date** e **Commissioned Date** marcar somente a opção **Blank**
- *text facet* das colunas **Candidate Name** e **Office Phone** marcar a opção **Blank**.

Existem 901 registros a serem descartados pois não contém nenhuma informação essencial.

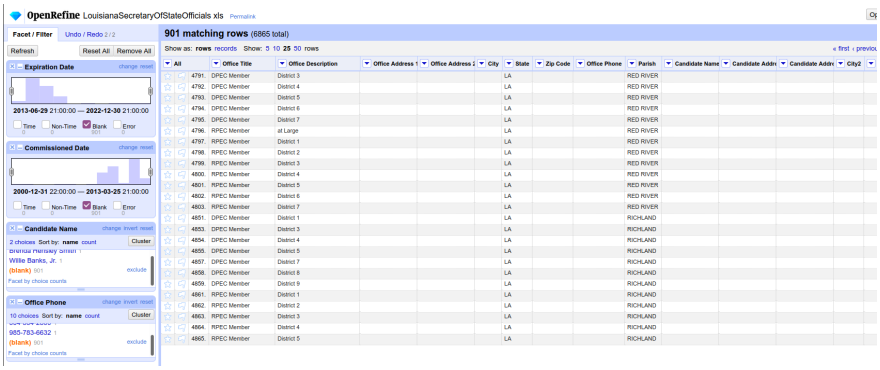


Figura 5: *Filtros aplicados*

São descartáveis pois não contém qualquer informação que pode nos auxiliar no seu preenchimento, por exemplo sabemos que temos o título, a descrição do escritório (alguns tem até a paróquia), porém sem o nome, telefone e datas do cargo não temos como saber quem são.

Para eliminá-los, clicar ▽ da primeira coluna e selecionar **Edit Rows** ▷ **Remove matching rows**. Se desejar eliminar todos selecionar **Remove All** de *Facet/Filter* (no lado superior esquerdo).

4.1 Outro tipo de duplicados

Um tipo mais comum de registros duplicados é quando eles não se batem por alguma diferença, por exemplo, maiúsculas e minúsculas, zero na frente, mesma palavra mas escrita com grafia diferente.

Por exemplo, aplicar o *text facet* na coluna **Office Title** e vemos que existem vários títulos nessas condições.

Neste caso devemos ter cuidado pois é muito fácil tratá-los com o OpenRefine, ao nos posicionarmos que qualquer palavra do filtro (por exemplo, em *Alderman*) aparece as opçoes **Edit** e **Include**. A primeira serve para modificar os registros e a segunda para mostrá-los na janela principal.

Para procedermos a alteração, selecionamos o segundo *Alderman* e pressionamos a opção **Edit** observamos que existem espaços ao final da palavra, retirar e pressionar **Apply**.

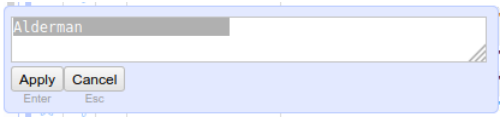


Figura 6: *Opção de Edição*

Porém essa forma só é aplicada quanto a grafia diferente, mudar os títulos *Alderman at large* e *Aldermen* para *Alderman* da mesma forma. Quando temos muitas informações na mesma situação,

por exemplo remover espaços, ao invés de editar cada uma podemos aplicar a função *trim* (elimina os espaços em branco tanto da direita quanto da esquerda) e eliminar todos de uma só vez.

Clicar ▽ de **Office Title** e selecionar **Edit Cells** ▷ **Common transforms** ▷ **Trim leading and trailing whitespace**. Assim todos com esse tipo de duplicidade se resolvem.

Outra forma de vermos os duplicados é aplicando a técnica de **clusterização**, esta localiza automaticamente os semelhantes e os agrupa. Clicar ▽ de **Office Title** e selecionar **Edit Cells** ▷ **Cluster and Edit**. Na janela *Cluster & Edit* alterar **Method** para *nearest neighbor*.

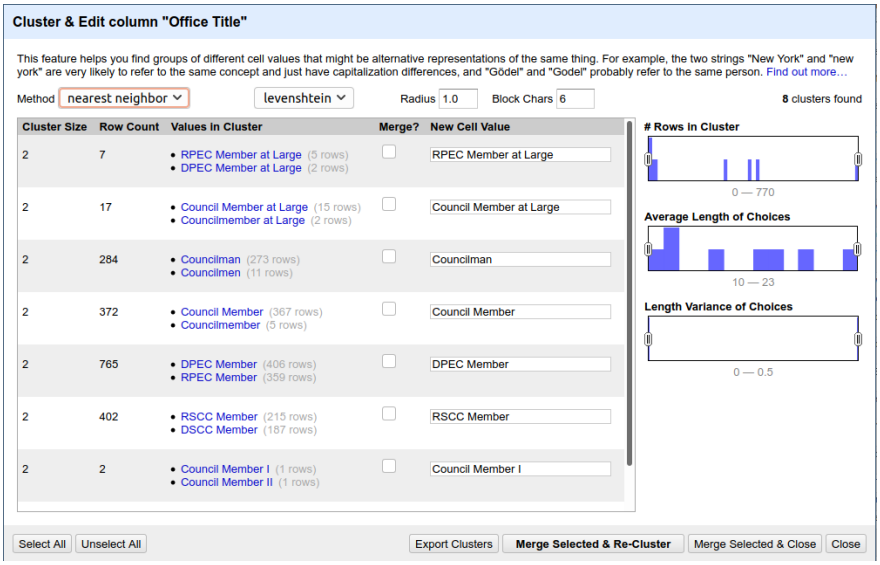


Figura 7: *Clusterização e Edição*

Outra forma de lidarmos com esses duplicados é através de uma troca (Replace) direta, por exemplo, alguns títulos possuem a palavra *at large* ao final. Clicar ▽ de **Office Title** e selecionar **Edit Cells** ▷ **Replace**. Em **find** digitar " at large" (sem as aspas e com o espaço) e marcar **case insensitive**. Não colocar nada em **Replace with**. Pressionar **OK**. Outro caso é com o texto "(s)" que aparece em alguns títulos, realizar a troca de mesma forma para praticar.

Algumas realmente não existirá forma a não ser realizarmos a edição da maneira como foi mostrado no início dessa sessão, como por exemplo o título *Councilmember* que aparece junto e deve ser trocado para "Council Member". Mas mesmo assim, observamos que o OpenRefine torna fácil o trabalho de lidar com duplicados.

4.2 Separar Informação

Vamos pegar um caso curioso que acontece com a coluna **Office Address**, muitos endereços são fornecidos apenas como uma caixa postal (P.O. Box), então vamos criar uma nova coluna que conterá essa informação. Primeiro usar as técnicas aprendidas para deixar todas as caixas postais com o formato correto: "P.O. Box" (sem aspas). Algumas estão muito separadas e outras muito juntas. E ainda existe o caso que ao invés de *Box* está escrito "Drawer" (gaveta).

Uma vez que temos todas em um padrão único. Podemos criar um filtro para nos mostrar as que possuem ou não "P.O. Box" sem termos o trabalho de fazer essa ação a cada uma. Clicar ▽ de **Office Address** e selecionar **Facet** ▷ **Custom text facet...**.

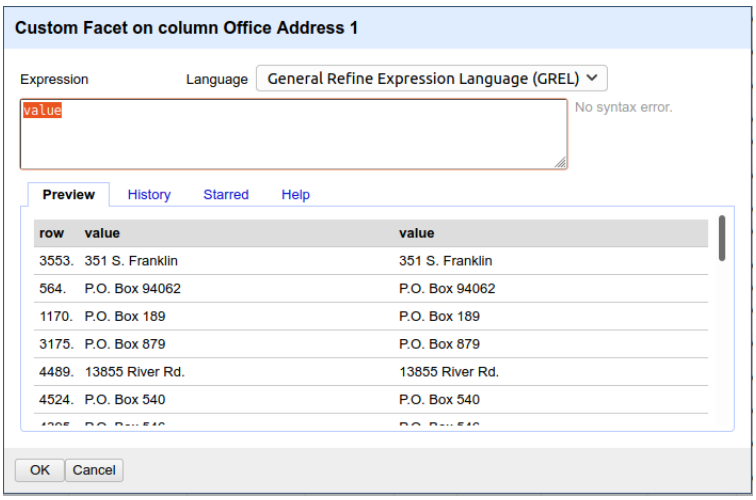


Figura 8: *Filtro Customizado*

Agora precisamos de conhecimento em **GREL**, não se preocupe tanto pois também podemos usar **Python** ou **Cloujure**. GREL é uma linguagem de Script bem simples de aprender, por exemplo para fazer o que desejamos basta escrever na caixa a expressão:
`value.startsWith("P.O. Box")`

A função *startsWith* (sua contrária é *endsWith*) verifica se o valor inicia com um texto definido entre os parênteses. E ao pressionar **OK** teremos nosso filtro construído. Neste selecione a opção *true*.

Para criar a nova coluna. Clicar ▽ de **Office Address** e selecionar **Edit column** ▷ **Add column based on this column...**. Informar o nome da coluna como **P.O. Box**. E a expressão em GREL:
`value.split(" ")[2]`

A função *split* quebra o texto em várias partes, a primeira (índice 0) é "P.O.", a segunda (índice 1) é "Box" e a terceira (índice 2) é a que nos interessa pois contém o valor da caixa postal.

Como última troca vamos remover o número. Clicar ▽ de **Office Address** e selecionar **Edit cells** ▷ **Transform**. E a expressão em GREL:
`value.split(" ")[0]+" "+value.split(" ")[1]`

Aplicamos a mesma função *split* porém obtendo as duas primeira palavras. E ao liberar o filtro, notamos que apenas os registros que continham a palavra "P.O. Box" ganharam seu número em uma nova coluna e este foi retirado da anterior. E temos 2.991 Secretários recebem sua correspondência por meio de Caixa Postal.

4.3 Dependência de Informação

Algo que acontece muito em limpeza de dados são informações conflitantes, por exemplo, acionar dois *facet text* para as colunas **City** e **Zip Code**. O prefixo do código postal de uma cidade deve ser único, se selecionamos *Alexandria* vemos que tem 3 prefixos diferentes: 71302 (com 1 registro), 71303 (com 2 registros) e 71309 (com 33 registros).

Outro caso é parte de informação faltante, por exemplo, para o mesmo filtro desmarque a cidade e aparece um código postal com o valor 1022. Poderíamos chegar a uma conclusão errada que falta

um ”7” na frente (visto que a maioria dos valores iniciam com 7). Porém se selecionamos este veremos que sua cidade é *Independence*, então desmarcamos este filtro e selecionamos esta cidade e vemos que os outros registros possuem o prefixo com valor ”70443”.

Existem outras três colunas que também são dependentes: **Candidate Name**, **Sex** e **Salutation**. porém para vê-las (sem ter que ficar indo de cá para lá na planilha) vamos ”encolher” as colunas que não nos interessa. Clicar ▽ de **Office Description** e selecionar **View** ▷ **Collapse this column**. E dessa forma conseguimos obter a visão das colunas que nos interessa. Para voltar ao tamanho padrão basta clicar na coluna fechada.

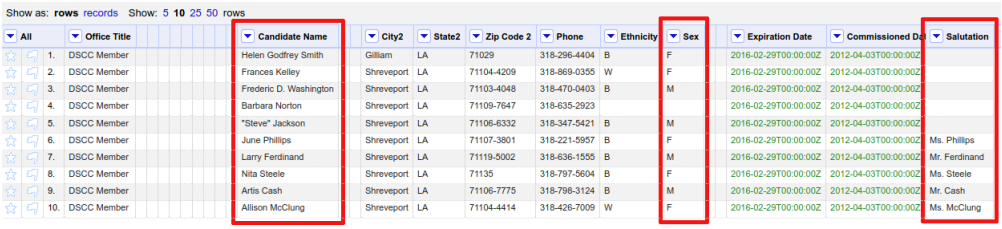


Figura 9: *Visão das Colunas de Interesse*

Também podemos expandir ou encolher todas as colunas de uma só vez. Clicar ▽ da primeira coluna **All** e selecionar **View**. Existem as seguintes opções:

- *Collapse all columns* - encolher todas as colunas.
- *Expand all columns* - espandir todas as colunas.
- *Show/Hide 'null' values in cells* - mostrar ou esconder células com valores nulos.

Devemos preencher os dados faltantes da coluna **Salutation** da seguinte forma: O pronome de tratamento (Mr. para **Sex** igual a ”M” e Mrs. igual a ”F”) seguido do último nome. Nosso primeiro problema é na coluna **Sex** no qual pode conter informação vazia, aplicar (conforme já mostrado) a função *trim* que elimina os espaços em branco nesta coluna.

Próximo passo é criar um filtro para isolarmos somente os registros faltantes. Clicar ▽ de **Salutation** e selecionar **Facet** ▷ **Customize facets** ▷ **Facet by null**. Aplicar o mesmo filtro para a coluna **Candidate Name**. Proceder de mesma forma para a coluna **Sex** porém usando a opção **Facet by empty string**.

Agora temos 3 filtros: marcar **true** para o filtro **Salutation** e **false** para **Candidate Name** e **Sex**. Nossa visão está restrita aos 1.773 registros que possuem nome e gênero mas não a forma de tratamento.

Precisamos criar uma coluna que contenha o pronome de tratamento devido com o último nome, devemos tomar cuidado pois esse pode conter Jr, III ou seja um sufixo de família. O nome anterior indica isso pois possui uma vírgula.

Clicar ▽ de **Salutation** e selecionar **Edit cells** ▷ **Transform**. E a expressão em GREL:

```
cells["Sex"].value.replace("M","Mr.  ").replace("F","Mrs.  ") +
cells["Candidate Name"].value.split(",")[-2].split(" ")[-1].trim()
```

Primeiro a partir do gênero montamos se será ”Mr.” ou ”Mrs.”, na segunda parte usamos uma função *split* para isolar o nome de família e uma segunda para obter o último nome. Ou seja, com uma única linha resolvemos nosso problema. O valor negativo para o índice trabalha de trás para frente neste, ou seja, valor -1 corresponde ao último valor, -2 ao penúltimo e assim sucessivamente. Pressionar **OK**. E remover os filtros.

5 Funções do GREL

Existem funções bem interessantes para se usar com o GREL, vejamos algumas que podem ser bem úteis para outros casos de limpeza.

`if (condição, caso Verdadeiro, caso Falso)`

Esse comando só serve para criamos uma nova coluna, por exemplo, **Salutation Complete** criada a partir da coluna **Sex** com a expressão:
```
if(value.contains("M"), "Mr.  ", "Mrs.  ") +
cells["Candidate Name"].value.split(",")[-2].split(" ")[-1].trim()
```

Esta função deve ser usada na coluna de origem, isso é **Sex**, e não em **Candidate Name**, pois senão retornaria errado. Como condicionais podemos usar:

- *isBlank(valor)* - se o valor é vazio.
- *isNonBlank(valor)* - se o valor não é vazio.
- *isNull(valor)* - se o valor é nulo.
- *isNotNull(valor)* - se o valor não é nulo.
- *isNumeric(valor)* - se o valor não é numérico.
- *isError(expressão)* - se a expressão retorna um erro.
- *contains(valor)* - contém um determinado valor ou expressão.
- *match(valor)* - é exatamente o valor ou expressão.

`diff(data final, data inicial, período)`

Temos duas datas em questão **Expiration Date** e **Commissioned Date**, a primeira corresponde ao dia de término do mandato e a outra seu início, se precisamos criar uma outra coluna com a diferença em dias. Clicar ▽ de **Commissioned Date** e selecionar **Edit collumn ▷ Add column based on this column...**. E a expressão em GREL:
`diff(cells['Expiration Date'].value, value, "days")`

Para o perído podemos usar para esta diferença:

- *minutes* - em minutos
- *hours* - em horas
- *days* - em dias
- *months* - em meses
- *years* - em anos

`row.index`

Esta função é útil quando necessitamos de uma chave primária, por exemplo, precisamos exportar os dados para uma tabela porém não existe nenhum campo que é candidato a ser a chave única, então basta escolher qualquer coluna e adicionar uma nova com a expressão:
`row.index + 1`

`with(avaliação, variável, expressão)`

Coloca o resultado da avaliação na variável e assim podemos simplificar a expressão a ser usada. Por exemplo, necessitamos criar uma coluna contendo o último nome do Funcionário (lembrar das regras). E a expressão em GREL:
`with(value.split(",")[-2], a, a.split(" ")[-1])`

Temos as seguintes expressões matemáticas em GREL:

- *floor(num)* - causa o arredondamento para baixo, $floor(3.9) = 3$
- *ceil(num)* - causa o arredondamento para cima, $ceil(3.2) = 4$
- *round(num)* - faz o arredondamento da forma padrão, $round(3.7) = 4$
- *min(num1, num2)* - menor valor entre dois números.
- *max(num1, num2)* - maior valor entre dois números.
- *mod(num1, num2)* - módulo entre dois números. Ideal para sabermos quando o número é par, mod(num_par, 2) é sempre igual a 0. Também quando é um múltiplo de outro valor $mod(9, 3) = 0$
- *ln(num)* - logaritmo natural do número.
- *log(num)* - logaritmo na base 10 do número.
- *exp(num)* - exponencial do número.
- *pow(base, expoente)* - base elevada ao expoente. A partir dessa podemos tirar a raiz quadrada elevando o número a 0.5, $pow(49, 0.5) = 7$
- *sum(lista)* - somatório dos números, $sum([3, 4, 8, 7, 9]) = 31$

Não confundir, se estamos usando um valor definido (i.e **value** para criar uma nova coluna) então a função $floor$ é usada da seguinte maneira $value.floor()$, sem nenhum parâmetro, porém também podemos usá-la $floor(value)$. A mesma regra se aplica a qualquer outra função.

```
filter(lista, variável, expressão)
```

É ideal quando temos um lista de valores e queremos reduzir esta, somente para entendermos, temos valores contendo 3 possibilidades "Sr", "Sr." e "Senhor" queremos deixar somente o "Sr.". Aplicamos a expressão em GREL:

```
filter(["Sr", "Sr.", "Senhor"], v, v.length() == 3)
```

Podemos ter as seguintes expressões de texto em GREL (a posição do primeiro caractere de um texto é 0):

- *replace(val1, val2)* - troca um valor por outro.
- *replaceChars(val1, val2)* - troca uma cadeia de caracteres por outra, é útil quando temos "sujeira" nos campos texto.
- *startsWith(val)* - se o texto inicia por determinado valor.
- *endsWith(val)* - se o texto termina por determinado valor.
- *toString()* - transforma o valor para texto.
- *indexOf(val)* - posição do texto para o valor informado, retorna -1 se não existir.
- *lastIndexOf(val)* - posição do texto para o valor informado porém conta de trás para frente, retorna -1 se não existir.
- *toLowercase()* - texto em minúsculas.
- *toUppercase()* - texto em maiúsculas.
- *toTitlecase()* - início de cada palavra em maiúscula, ideal para corrigir nomes.
- *chomp(val)* - corta o termo final do texto se for similar ao valor.
- *substring(inicial, final)* - corta um texto. O **final** é opcional mas se passado não será incluído, $substring("Fernando", 3) = "nando"$, porém $substring("Fernando", 3, 6) = "nan"$
- *splitByLengths(val1, val2, ...)* - transforma o texto em uma lista sendo que cada elemento tem seu tamanho definido.

```
find(expressão)
```

Se determinada expressão for encontrada no texto então esta será retornada, por exemplo, será que um determinado campo contém vogais? E como estas aparecem? Usamos a expressão:

```
value.toLowercase().find(/[aeiou]+/).sort()
```

Retorna uma lista com todas as vogais da expressão ordenadas alfabeticamente. Podemos ter as seguintes expressões de lista em GREL, normalmente algumas delas também podem ser aplicadas a texto:

- *length()* - quantidade de elementos.
- *sort()* - ordenar uma lista, numérica ou não.

```
forEach(lista, variável, expressão)
```

Avalia a lista, colocando cada valor na variável e chamando a expressão.

Possui algumas funções similares:

- *forEachIndex(lista, índice, variável, expressão)* - além do valor passa o índice.
- *forRange(início, fim, passo, variável, expressão)* - ao invés de usar uma lista de valores cria um laço com um valor inicial e final percorrendo os elementos conforme o passo.
- *forNonBlank(lista, variável, expressãoNãoVazios, expressãoVazios)* - verifica se a variável em questão não é vazia e chama a primeira expressão, caso contrário a segunda.

Uma última função antes de terminarmos, para criarmos uma coluna com TODOS os valores das outras colunas? Use a expressão (que pode ser colocada na opção **Add column based on this column...** em qualquer coluna):
`forEach(row.columnNames,cn,cells[cn].value).join("|")`

Acabou? Recomendo que veja o tutorial sobre Limpeza de Dados de **John Little** que está disponível no endereço `https://libjohn.github.io/openrefine/` e o blog **Open Refine for Librarians** em `http://liwong.blogspot.com/`. Sempre que tenho dúvidas recorro a ambos.

6 Conclusão

Como vimos os casos de uso do OpenRefine mostra que, como os pesquisadores, podemos diagnosticar e agir com base na precisão dos dados. Vimos os princípios e práticas da limpeza de dados, bem como o OpenRefine pode ser usado para executar tarefas essenciais que nos ajudam a obter o melhor dos dados.

Registros duplicados, valores vazios e formatos inconsistentes são fenômenos com os quais devemos estar preparados para lidar ao usar quaisquer conjuntos de dados. Levamos muito tempo para descobrir inconsistências nos dados contidos em uma simples planilha ou banco de dados. À medida que cada vez mais compartilhamos, agregamos e reutilizamos dados da Web, e devemos estar atentos a questões de qualidade de dados que inevitavelmente surgem.

Sou um entusiasta do mundo **Open Source** e novas tecnologias. Qual a diferença entre Livre e Open Source? Livre significa que esta apostila é gratuita e pode ser compartilhada a vontade. Open Source além de livre todos os arquivos que permitem a geração desta (chamados de arquivos fontes) devem ser disponibilizados para que qualquer pessoa possa modificar ao seu prazer, gerar novas, complementar ou fazer o que quiser. Os fontes da apostila (que foi produzida com o LaTex) está disponibilizado no GitHub [4]. Veja ainda outros artigos que publico sobre tecnologia através do meu Blog Oficial [2].

Referências

[1] Página Oficial do OpenRefine
`https://openrefine.org/`

[2] Fernando Anselmo - Blog Oficial de Tecnologia
`http://www.fernandoanselmo.blogspot.com.br/`

[3] Encontre essa e outras publicações em
`https://cetrex.academia.edu/FernandoAnselmo`

[4] Repositório para os fontes da apostila
`https://github.com/fernandoans/publicacoes`