Roteiro de Análise de Qualidade de dados

⟨https://github.com/bda-hiae/qualidade⟩

Leandro Furlam Elias Ribeiro Alexandre Rodrigues

Setembro 2020

Sumário

1 Qualidade de dados

O processo de analise de qualidade de dados esta focado na avaliação de conjuntos de dados e na aplicação de acoes corretivas, para garantir que estes estejam adequados aos propósitos para os quais foram originalmente destinados (MERINO et al., 2016). Dessa forma, a qualidade de dados esta diretamente relacionada a confiabilidade dos dados de entrada. Considerando que os dados têm níveis inadequados de qualidade, é provável que ocorram erros, que podem se propagar acidentalmente e inconscientemente por todo o fluxo da informação, prejudicando a eficiência do sistema. Formas regulares de avaliar a qualidade de dados com modelos clássicos geralmente se destinam a detectar e corrigir erros em fontes conhecidas com base em um conjunto limitado de regras. No ambiente de *Big Data*, a quantidade de regras pode ser enorme e o custo da aplicação para correção de erros pode não ser viável e nem apropriado (*e.g.* o enorme volume de dados ou a volatilidade dos dados de *streaming*). Isso ocorre principalmente porque o *Big Data* não é apenas sobre dados, mas também sobre uma pilha conceitual e tecnológica completa, incluindo dados brutos e processados, armazenamento, formas de gerenciar dados, processamento e analise (MERINO et al., 2016).

Uma dimensão de qualidade de dados é um termo descritor de um recurso de dados, o qual pode ser medido ou avaliado de acordo com padrões definidos, a fim de determinar a qualidade de um conjunto de dados (DAMA, 2013). Geralmente, dados só têm valor quando dão suporte a um processo ou a uma tomada de decisão. Em consequência, as regras de qualidade de dados definidas devem levar em consideração o valor que os dados podem fornecer para o sistema.

Neste relatório, seis dimensões de qualidade de dados são analisadas: completude, conformidade, acurácia, consistência e temporalidade (DAMA, 2013). A dimensão unicidade, que objetiva mensurar o grau de duplicidade nos dados, foi excluída deste estudo, uma vez que dados de identificação dos pacientes são removidos da planilha de informa coes.

Completude caracteriza a taxa de preenchimento das variáveis. Para cada variável é calculado o percentual de entradas com informação não nulas, respeitando, quando houver, sua dependência com outras variáveis.

Conformidade detecta concordância nos valores digitados nos campos das variáveis, avaliando se os valores de entrada não nulos estão em conformidade com os padrões descritos pelo dicionario de dados. Para cada variável estudada é calculado o percentual de entradas em conformidade com o padrão adotado.

Acurácia visa detectar se informação registrada reflete o evento ou objeto descrito, isto é, verificar se o dado cadastrado esta em concordância com o evento observado. Devido ao processo de anonimização dos dados, a analise de acurácia se restringe a verificar a possibilidade das informações registradas. Note que acurácia e conformidade são dimensões distintas, pois enquanto conformidade avalia o padrão do dado, acurácia avalia a razoabilidade dos dados. Para cada variável estudada é calculado o percentual de entradas com informações acuradas.

Consistência constitui de testes envolvendo duas ou mais variáveis visando detectar inconsistências entre dados de um mesmo registro. Para cada teste considerado é calculado os percentuais de aprovação e falha.

Temporalidade objetiva efetuar medidas estatísticas nos intervalos de tempos entre eventos, *e.g.* Nascimento de um recém-nascido e inclusão desse registro no sistema. O principal interesse é verificar se o dado é disponibilizado prontamente.

Referências

MERINO, J. et al. A data quality in use model for big data. *Future Generation Computer Systems*, v. 63, p. 123–130, 2016. ISSN 0167-739X. Modeling and Management for Big Data Analytics and Visualization. Disponível em: ⟨http://www.sciencedirect.com/science/article/pii/S0167739X15003817⟩.

DAMA, U. The six primary dimensions for data quality assessment-defining data quality dimensions. *Bristol: np URL: https://www.whitepapers.em360tech.com/wp-content/files_mf/1407250286DAMAUKDQDimensionsWhitePaperR37.pdf*, v. 3, p. 2017, 2013.

2 *Scripts* de qualidade de dados

2.1 Pacotes utilizados

2.1.1 R 3.6.2 (Dark and Stormy Night)

- `bdacodeR 0.0.1`
- `data.table 1.13.0`
- `dplyr 1.0.2`
- `ggplot2 3.3.2`
- `kableExtra 1.2.1`
- `knitr 1.29`
- `reshape 0.8.8`
- `reticulate 1.16`
- `stringi 1.5.3`
- `stringr 1.4.0`
- `viridis 0.5.1`

2.1.2 Python 3.7.6

- `numpy 1.17.3`
- `pandas 1.1.2`
- `scipy 1.4.1`
- `sklearn 0.0.5`
- `sparse_dot_topn 0.2.9`

2.2 Funções gerais de qualidade

qualidade.R

2.2.1 Conformidade dos registros por tamanho e valores validos de domínio

```
    @param data vector. Dados a serem analisados
    @param tamanho numeric[2]. Tamanhos minimos e maximos a serem comparados
    @param variantes character. Valores validos de dominio a serem comparados
    @return vector. Dados com NA em posicoes apresentando falhas

    conformidade <- function(data, tamanho, variantes)
```

2.2.2 Conformidade dos registros a partir de um arquivo *.cnv*

Caso hajam sequências do tipo *A00-A99*, `@function create_sequence` será utilizada.

```
    @param data vector. Dados a serem analisados
    @param cnv_path character. Caminho para o arquivo .cnv
    @return vector. Dados com NA em posicoes apresentando falhas

    conformidade_cnv <- function(data, cnv_path)
```

2.2.3 Acurácia dos registros a partir de valores validos de domínio

```
    @param data vector. Dados a serem analisados
    @param variantes character. Valores validos de dominio a serem comparados
    @return vector. Dados com NA em posicoes apresentando falhas

    acuracia <- function(data, variantes)
```

2.2.4 Acurácia dos registros a partir de valores de datas

```
    @param data vector. Dados a serem analisados
    @param formato character. Formato dos valores de data. Vide classe Date
    @param referencias character[2]. Referencias inferiores e superiores para
        comparacao
    @return vector. Dados com NA em posicoes apresentando falhas

    acuracia_data <- function(data, formato, referencias)
```

2.2.5 Acurácia dos registros a partir de valores validos de código de município

```
    @param data vector. Dados a serem analisados
    @param digitos numeric. Quantidade de digitos para avaliacao
    @return vector. Dados com NA em posicoes apresentando falhas

    acuracia_mun <- function(data, digitos)
```

2.2.6 Acurácia dos registros a partir de valores válidos de UF

```
    @param data vector. Dados a serem analisados
    @param tipo character. Tipo do registro representando a UF. S para siglas,
        N para nome completo do estado e C para o codigo
    @return vector. Dados com NA em posicoes apresentando falhas

    acuracia_uf <- function(data, tipo)
```

2.2.7 Acurácia dos registros a partir de registros numéricos

```
    @param data vector. Dados a serem analisados
    @param positivo boolean. Numerico e positivo?
    @param inteiro boolean. Numerico e inteiro?
    @return vector. Dados com NA em posicoes apresentando falhas

    acuracia_numerico <- function(data, positivo, inteiro)
```

2.2.8 Acurácia dos registros para identificar sequências de zeros ou espaços em branco, visando identificar registros ignorados

```
    @param data vector. Dados a serem analisados
    @param rm.zeros boolean. Remover sequencias de zeros?
    @param rm.whitespace boolean. Remover sequencias de espacos em branco?
    @return vector. Dados com NA em posicoes apresentando falhas

    acuracia_zeros_whitespace <- function(data, rm.zeros, rm.whitespace)
```

2.2.9 Temporalidade entre duas datas

Retornando `@function summary_all(data2 - data1)`.

```
    @param data1 vector. Dados a serem analisados
    @param formato1 character. Formato dos valores de data. Vide classe Date
    @param referencias1 character[2]. Referencias inferiores e superiores para
        comparacao
    @param data2 vector. Dados a serem analisados
    @param formato2 character. Formato dos valores de data. Vide classe Date
    @param referencias2 character[2]. Referencias inferiores e superiores para
        comparacao
    @return numeric

    temporalidade <- function(data1, formato1, referencia1, data2, formato2,
        referencia2)
```

2.2.10 Conversão em tipo `Date`

```
    @param data vector. Dados a serem convertidos
    @param formato character. Formato dos valores de data. Vide classe Date
    @return Date

    toDate <- function(data, formato)
```

2.2.11 Sumário

Cálculo de valores mínimo, 1o quartil, mediana, média, 3o quartil, máximo, e não disponível.

```
    @param data vector. Dados a serem analisados
    @return numeric

    summary_all <- function(data)
```

2.2.12 Cria sequências do tipo *A00-A99*

```
    @param data vector. Dados a serem analisados
    @return character

    create_sequence <- function(data)
```

string-matching.R / string_matching.py

2.2.13 *String matching*

⟨https://bergvca.github.io/2017/10/14/super-fast-string-matching.html⟩

```
    @param clean character. Registros para comparacao
    @param dirty character. Registros para comparacao
    @param name character. Nome para arquivo de saida. Vide arquivo fonte para
        mais detalhes

    string_matching <- function(dirty, clean, name)
```

2.3 Testes realizados em todas as variáveis

O diagnóstico realizado é com as colunas agrupadas pelas quantidades de registros. Possuem a mesma ideia que um vetor agrupado pela fincão `plyr::count`, com nomes de colunas `variavel` e `total`. analise-agrupada.R

2.3.1 Diagnóstico geral (por variável)

```
    @param variaveis character. Variaveis a serem analisadas
    @param parametros_database list. Parametros para leitura dos dados
    @param out boolean. Escrever as saidas em "diagnostico.csv"?
    @return data.table. Colunas: variavel, total, nao.nulo, conformes, acurados
    diagnostico_agr <- function(variaveis, parametros_database, out=T)
```

2.3.2 Diagnóstico agrupado por ano/estado

```
    @param variaveis character. Variaveis a serem analisadas
    @param parametros_database list. Parametros para leitura dos dados
    @param referencia_subs data.frame. Tabela contendo parametros para obtencao
         do ano/mes/estado dos registros. Vide arquivo fonte para mais detalhes
    @param sub character. ANO ou UF
    @param out boolean. Escrever as saidas em "diagnostico-sub.csv"?
    @return data.table. Colunas: sub, total, nao.nulo, nao.nulo_conformes ,
        conformes, conformes_acurados, acurados

    diagnostico_sub_agr <- function(variaveis, parametros_database, referencia_
        subs, sub, out=T)
```

2.4 Testes envolvendo mais de uma variável

Testes específicos envolvendo diferentes variáveis de uma mesma tabela.

consistencia.R

2.4.1 Função auxiliar para calcular a consistência anual entre datas

Para consistência geral, basta somar.

```
    @param V1 character
    @param V2 character. Teste: V1 > V2
    @param referencia_subs data.frame. Tabela contendo parametros para obtencao
         do ano/mes/estado dos registros. Vide arquivo fonte para mais detalhes
```

```
    @param sub character. ANO ou UF
    @param parametros_database list. Parametros para leitura dos dados
    @return data.table. Colunas total, falhas, sub
    consistencia_data_subs <- function(V1, V2, referencia_subs, sub, parametros
        _database)
```

2.4.2 Consistência anual entre uma *query*

Para consistência geral, basta somar.

```
    @param query character. uma query SQL, por exemplo.
    @param variaveis_query character. variaveis envolvidas na query
    @param referencia_subs data.frame. Tabela contendo parametros para obtencao
         do ano/mes/estado dos registros. Vide arquivo fonte para mais detalhes
    @param sub character. ANO ou UF
    @param parametros_database list. Parametros para leitura dos dados
    @return data.table. Colunas total, falhas, sub

    consistencia_query_subs <- function(variaveis, query, referencia_subs, sub,
         parametros_database)
```

2.5 Testes realizados apenas em variáveis contendo o identificador do paciente e contendo informações deste

O diagnóstico realizado é com as colunas referentes à identificadores de pacientes que se repetem ao menos uma vez na base de dados. Estas, então, são agrupadas pelos identificadores.

unicidade-agrupada.R

2.5.1 Diagnóstico geral

Por comparação + geral.

```
    @param variavel_id character. Variavel contendo o identificador do paciente
    @param variaveis_comp character. Variaveis para comparacao
    @param parametros_database list. Parametros para leitura dos dados
    @param out boolean. Escrever as saidas em "unicidade.csv"?
    @return data.table. Colunas: falhas, total, falhas_geral, total_geral,
        variavel, teste

    unicidade_agr <- function(variavel_id, variaveis_comp, parametros_database,
         out=T)
```

2.5.2 Diagnóstico agrupado por ano/estado

```
    @param variavel_id character. Variavel contendo o identificador do paciente
    @param variaveis_comp character. Variaveis para comparacao
    @param parametros_database list. Parametros para leitura dos dados
    @param referencia_subs data.frame. Tabela contendo parametros para obtencao
         do ano/mes/estado dos registros. Vide arquivo fonte para mais detalhes
    @param sub character. ANO ou UF
    @param out boolean. Escrever as saidas em "unicidade-sub.csv"?
    @return data.table. Colunas: sub, total, falhas
```

```
8
9     unicidade_sub_agr <- function(variavel_id, variaveis_comp, referencia_subs,
          sub, parametros_database, out=T)
```

2.5.3 Frequência dos identificadores de de pacientes

```
1     @param variavel_id character. Variavel contendo o identificador do paciente
2     @param parametros_database list. Parametros para leitura dos dados
3     @param out boolean. Escrever as saidas em "unicidade-frequencia.csv"?
4     @return data.table. Colunas: id, categoria
5
6     freq_identificadores <- function(variavel_id, parametros_database, out=T)
```

2.6 Testes realizados em variáveis representando datas

O diagnóstico realizado é com as colunas referentes à identificadores de pacientes que se repetem ao menos uma vez na base de dados. Estas, então, são agrupadas pelos identificadores.

temporalidade-agrupada.R

2.6.1 Temporalidade dos registros

```
1     @param v_temp list. Lista contendo as variaveis a serem analisadas. Vide
          arquivo fonte para mais detalhes.
2     @param parametros_database list. Parametros para leitura dos dados
3     @param out boolean. Escrever as saidas em "temporalidade.csv"?
4     @return data.table. Colunas: Min., 1st Qu., Median, Mean, 3rd Qu., Max., NA
          s, V1, V2, teste
5
6     temporalidade_agr <- function(v_temp, parametros_database, out=T)
```

2.6.2 Temporalidade dos registros, agrupado por ano/estado

```
1     @param v_temp list. Lista contendo as variaveis a serem analisadas. Vide
          arquivo fonte para mais detalhes.
2     @param parametros_database list. Parametros para leitura dos dados
3     @param referencia_subs data.frame. Tabela contendo parametros para obtencao
           do ano/mes/estado dos registros. Vide arquivo fonte para mais detalhes
4     @param sub character. ANO ou UF
5     @param out boolean. Escrever as saidas em "temporalidade-sub.csv"?
6     @return data.table. Colunas Tx, sub
7
8     temporalidade_sub_agr <- function(v_temp, parametros_database, referencia_
          subs, sub, out=T)
```

2.7 Tabelas auxiliares

Levantamento de tabelas pertinentes à analise de qualidade.

auxiliares.R

2.7.1 *Outliers*

Para variáveis numéricas.

```
    @param parametros_database list. Parametros para leitura dos dados
    @param out boolean. Escrever as saidas em "outliers.csv"?
    @return data.table. Colunas variavel, outliers

    outliers_agr <- function(parametros_database, out=T)
```

2.7.2 Registros inconformes/inacurados de variáveis

```
    @param variaveis character. Variaveis a serem analisadas
    @param parametros_database list. Parametros para leitura dos dados
    @param out boolean. Escrever as saidas em "inconformes-inacurados.csv"?
    @return data.table. Colunas variavel, inconformes, acurados

    inconformes_inacurados_agr <- function(variaveis, parametros_database, out=
        T)
```

2.7.3 Registros representando informações ignoradas de variáveis

```
    @param variaveis character. Variaveis a serem analisadas
    @param parametros_database list. Parametros para leitura dos dados
    @param out boolean. Escrever as saidas em "ignorados.csv"?
    @return data.table. Colunas variavel, inconformes, acurados

    ignorados_agr <- function(variaveis, parametros_database, out=T)
```

2.8 Análise de qualidade

Suma das funções de qualidade mencionadas acima. Necessário definir `@param parametros_database`, e, opcionalmente, `@param referencia_subs` (para analises por ano/estado), `@param v_temp` para analise da temporalidade, `@param variavel_id` e `@param variaveis_comp` para analises de unicidade.

geral.R

2.9 Dicionário de dados

Para cada variável são definidas colunas especificas utilizadas na analise de qualidade.

dicionario.R

2.9.1 Formatação do dicionário

Sugere-se que `@param dict` seja construído após o levantamento de domínio, em uma planilha adequada.

```
    @param dict data.frame. Tabela com os parametros para analise:
        VARIAVEL: variavel a ser analisada
        MIN/MAX: tamanho minimo/maximo da quantidade de caracteres
        VARIANTES: dominio aceito, separados por <, >
        CLASSE: classe da variavel (usada para obter os tipos numeric)
```

```
        FORMATO: formato da data. ex: %d%m%Y
        REFERENCIA.INF/REFERENCIA.SUP: referencias inferiores e superiores de
            datas
        DIGITOS: quantidade de digitos de codigos de estados
        TIPO: tipo do dado representando UF. S para siglas, N para nome
            completo do estado e C para o codigo
        RM.ZEROS: zeros sao considerados como invalidos? TRUE/FALSE para todas
            as variaveis
        RM.WHITESPACE: whitespaces sao considerados como invalidos? TRUE/FALSE
            para todas as variaveis
        IGNORADOS: palavras representando valores invalidos
        POSITIVO: numerico e positivo?
        INTEIRO: numerico e inteiro? usando tambem em unicidade, caso a
            variavel seja numerica
    dicionario_formatado <- function(dict)
```

2.9.2 Atualizar as datas limites do dicionário de dados?

```
    @param parametros_database list. Parametros para leitura dos dados
    @param referencia_subs data.frame. Vide arquivo fonte para mais detalhes

    atualizar_datas_dict <- function(parametros_database, referencia_subs)
```

2.10 Leitura dos dados

Leitura das colunas da grande tabela de dados, visando realizar as analises mencionadas. Caso mude o sistema de armazenamento, este arquivo deve ser alterado.

A construção, neste momento, se deu com os dados armazenados em *AWS Athena*, com o auxilio do pacote *bdacode*, produzido no âmbito deste projeto.

leitura-data.R

2.10.1 Leitura dos dados agrupados pelas quantidades de cada registro

```
    @param variavel. Variavel de leitura
    @param parametros_database list. Parametros para leitura dos dados
    @return data.table. Colunas variavel e total

    coluna_agregada <- function(variavel, parametros_database)
```

Coluna SEXO. `@param variavel = SEXO`.

	variavel	total
1:	FEMININO	3465363
2:	MASCULINO	3012383

2.10.2 Leitura dos dados agrupados pelas quantidades de cada registro, a partir da comparação de *substrings*

Substrings obtidas de `@param variavel_ref`.

```
    @param variavel. Variavel de leitura
    @param parametros_database list. Parametros para leitura dos dados
    @param variavel_ref character. Variavel utilizada como referencia
    @param inicio_ref numeric. Posicao inicial de @param ref
    @param ref character. Substring a ser analisada
    @return data.table. Colunas variavel e total

    coluna_ref_agregada <- function(variavel, parametros_database, variavel_ref
        , inicio_ref, ref)
```

Registros da coluna SEXO referente ao CNES 2007061. `@param variavel = SEXO`, `@param variavel_ref = CNES`, `@param inicio_ref = 1`, `@param ref = 2007061`.

	variavel	total
1:	FEMININO	7596
2:	MASCULINO	6773

2.10.3 Variáveis de uma tabela de dados

```
    @param parametros_database list. Parametros para leitura dos dados
    @return character. Vetor contendo os nomes das colunas

    variaveis_tabela <- function(parametros_database)
```

2.10.4 Leitura dos dados agrupados pelas quantidades de cada registro, a partir de valores armazenados referentes a um identificador de paciente

```
    @param variavel_id character. Variavel contendo o identificador do paciente
    @param variaveis_comp character. Variaveis contendo informacoes do paciente
    @param parametros_database list. Parametros para leitura dos dados
    @return data.table. Colunas variavel_id, variaveis_comp e total

    coluna_unicidade_agregada <- function(variavel_id, variaveis_comp,
        parametros_database)
```

Dado o identificador do paciente (coluna ID_PACIENTE), obter informações destes pacientes (data de nascimento e sexo). O resultado refere-se apenas a identificadores que se repetem ao menos uma vez. Identificadores únicos não devem ser levados em consideração. `@param variavel_id = ID_PACIENTE`, `@param variaveis_comp = [DT_NASCIMENTO, SEXO]`.

	ID_PACIENTE	DT_NASCIMENTO	SEXO	total
1:	20584815000	29/11/1971	FEMININO	2407
2:	1322355600	16/04/1963	MASCULINO	2395
3:	15645467400	13/04/1934	FEMININO	2352
	—			
253035:	6257451600	15/05/1952	MASCULINO	1
253036:	2299756400	22/06/2010	FEMININO	1

2.10.5 Leitura dos dados agrupados pelas quantidades de cada registro, a partir de valores armazenados referentes a um identificador de paciente

```
    @param variavel_id character. Variavel contendo o identificador do paciente
    @param variaveis_comp character. Variaveis contendo informacoes do paciente
    @param parametros_database list. Parametros para leitura dos dados
    @param variavel_ref character. Variavel utilizada como referencia
    @param inicio_ref numeric. Posicao inicial de @param ref
    @param ref character. Substring a ser analisada
    @return data.table. Colunas variavel_id, variaveis_comp e total

    coluna_ref_unicidade_agregada <- function(variavel_id, variaveis_comp,
        parametros_database, variavel_ref, inicio_ref, ref)
```

Dado o identificador do paciente (coluna ID_PACIENTE), obter informações destes pacientes (data de nascimento e sexo) referente ao CNES 2007061. O resultado refere-se apenas a identificadores que se repetem ao menos uma vez. Identificadores únicos não devem ser levados em consideração. `@param variavel_id = ID_PACIENTE`, `@param variaveis_comp = [DT_NASCIMENTO, SEXO]`, `@param variavel_ref = CNES`, `@param inicio_ref = 1`, `@param ref = 2007061`.

	ID_PACIENTE	DT_NASCIMENTO	SEXO	total
1:	5743875500	01/12/1978	FEMININO	1474
2:	3995624900	15/09/1982	FEMININO	1359
3:	12540836800	10/05/1957	MASCULINO	309
	—			
2436:	19234106400	13/04/1989	FEMININO	1
2437:	7183677200	12/03/1949	FEMININO	1

2.10.6 Coluna agrupada usada para fins de comparação entre datas, agrupado pela quantidade de registros

```
    @param x character
    @param y character. Variaveis de comparacao: y - x
    @param parametros_database list. Parametros para leitura dos dados
    @return data.table. Colunas: day_diff, total

    coluna_temp_agregada <- function(x, y, parametros_database)
```

Quantidade de dias entre a data de ocorrência e a data de nascimento. `@param x = DT_NASCIMENTO`, `@param y = DT_NASCIMENTO` (DATA_OCORR - DT_NASCIMENTO).

	day_diff	total
1:	0	391
2:	23452	113
3:	24931	107
	—	
34471:	2403	1
34472:	35828	1

2.10.7 Coluna agrupada usada para fins de comparação entre datas, agrupado pela quantidade de registros

```
    @param x character
    @param y character. Variaveis de comparacao: y - x
    @param parametros_database list. Parametros para leitura dos dados
    @param variavel_ref character. Variavel utilizada como referencia
    @param inicio_ref numeric. Posicao inicial de @param ref
    @param ref character. Substring a ser analisada
    @return data.table. Colunas: day_diff, total

    coluna_ref_temp_agregada <- function(x, y, parametros_database,
                                         variavel_ref, inicio_ref, ref)
```

Quantidade de dias entre a data de ocorrência e a data de nascimento (DATA_OCORR - DT_NASCIMENTO), referente ao CNES 2007061. `@param x = DT_NASCIMENTO`, `@param y = DT_NASCIMENTO` (DATA_OCORR - DT_NASCIMENTO), `@param variavel_ref = CNES`, `@param inicio_ref = 1`, `@param ref = 2007061`.

	day_diff	total
1:	26555	18
2:	2166	16
3:	2632	16
	—	
4550:	26791	1
4551:	19416	1

2.10.8 Leitura do dicionario de dados padronizado

Padronizado para os *scripts* de qualidade. Perceba que este possui colunas definidas para realizar a analise de qualidade, e é especificado neste guia de qualidade.

```
    @param parametros_database list. Parametros para leitura dos dados
    @param referencia_subs data.frame. Tabela contendo parametros para obtencao
         do ano/mes/estado dos registros. Vide arquivo fonte para mais detalhes
    @param atualizar_datas boolean. Atualizar as datas contidas no dicionario?
    @return data frame. Dicionario de dados, vide a definicao do dicionario de
        dados

    leitura_dict <- function(parametros_database, referencia_subs=NULL,
        atualizar_datas=F)
```

No *script* construído, deixamos o arquivo *.csv* salvo "bases/scripts/database_name.table_name/dict-table_name.csv")

2.11 Funções exclusivas de AWS Athena

Funções construídas de forma que, dada sua complexidade, funcionam exclusivamente em *AWS Athena*, com o auxilio do pacote *bdacode*, produzido no âmbito deste projeto.

dominio-athena.R

2.11.1 Levantamento de domínio

Levantamento de `@param table_name`. São também calculados tamanhos, classe, porcentagem de registros e completude.

```
    @param variaveis character. Variaveis a serem analisadas
    @param referencia_subs data.frame. Tabela contendo parametros para obtencao
        do ano/mes/estado dos registros. Vide arquivo fonte para mais detalhes
    @param database_name character
    @param table_name character
    Saidas escritas em "dominio-unique-athena.csv"

    tabela_dominio <- function(variaveis, referencia_subs, database_name, table
        _name)
```

2.11.2 Levantamento da disponibilidade dos arquivos, por ano e estado

```
    @param database_name character
    @param table_name character
    @param referencia_subs data.frame. Tabela contendo parametros para obtencao
        do ano/mes/estado dos registros. Vide arquivo fonte para mais detalhes
    Saidas escritas em "arquivos-faltantes-athena.csv"

    disponibilidade_arquivos <- function(database_name, table_name, referencia_
        subs)
```

```
@param referencia_subs <- [[VARIAVEL=REMESSA], [INICIO.ANO=12], [QNT.ANO=4],
[INICIO.MES=16], [QNT.MES 2]]
```

2.12 Relatório de padronização

Funções relativas ao relatório de padronização das bases, contendo, por exemplo, colunas que foram acrescentadas/removidas ao decorrer dos anos.

proadi-athena.R

2.12.1 Leitura e atualização do relatório de padronização, presente em *proadi-misc*

```
    @param referencia_subs data.frame. Tabela contendo parametros para obtencao
        do ano/mes/estado dos registros. Vide arquivo fonte para mais detalhes
    @param table_name character. Tabela de database_name
    @return dataframe. Colunas FILENAME, QNT_LINHAS, QNT_COLUNAS, colunas, ano

    relatorio_atualizado_S3 <- function(table_name, referencia_subs=NULL)
```

2.12.2 Obtenção do número total de registros por variável vindo do relatório de padronização, presente em *proadi-misc*

```
    @param relatorio data.frame. Idem retorno de relatorio_atualizado_S3()
    @param referencia_subs data.frame. Tabela contendo parametros para obtencao
        do ano/mes/estado dos registros. Vide arquivo fonte para mais detalhes
    @param table_name character. Tabela de database_name
    @return list. Posicoes geral e as subs presentes em referencia_subs

    totais_variavel_relatorio_S3 <- function(relatorio, referencia_subs, table_
        name)
```

2.12.3 Houve acréscimo dos registros?

```
    @param database_name character. database_name
    @param table_name character. Tabela de @param database_name
    @param path_to_data character. Caminho em que o arquivo de quantidade de
        registros sera/esta salvo
    @return boolean

    acrescimo_de_registros <- function(database_name, table_name, path_to_data)
```

2.12.4 Informação geral da base de dados

```
    @param variaveis character. Variaveis analisadas.
    @param referencia_subs data.frame. Tabela contendo parametros para obtencao
         do ano/mes/estado dos registros. Vide arquivo fonte para mais detalhes
    @param database_name character. database_name
    @param table_name character. Tabela de database_name
    @param path_to_data character. Caminho em que o arquivo de quantidade de
        registros sera/esta salvo
    @return data.frame. Possiveis informacoes: Base de dados, Data de obtencao
        dos dados, Periodo, Regiao geografica, Numero maximo de variaveis,
        Numero de registros. Saidas salvas em "informacoes-gerais.csv"

    informacoes_gerais <- function(variaveis, referencia_subs, database_name,
        table_name, path_to_data)
```

2.12.5 Análise de qualidade no âmbito *Proadi*

Suma das funções de qualidade mencionadas acima. Necessário definir `@param parametros_database`, e, opcionalmente, `@param referencia_subs` (para análises por ano/estado), `@param v_temp` para analise da temporalidade, `@param variavel_id` e `@param variaveis_comp` para analises de unicidade. Ë muito semelhante a análise geral, entretanto, neste ocorrem ajustes referentes ao relatório de padronização. Por exemplo, uma variável pode não existir em um determinado ano, logo, seus resultados devem levar este fato em consideração.

relatorio-athena.R

3 Relatório de qualidade

Requisitos:

- dados armazenados em *Amazon Athena*;
- pacote *bdacode* para obtenção destes dados;
- *scripts* de qualidade.

Para todas as funções mencionadas, os respectivos metadados estão nos cabeçalhos, explicando cada um dos parâmetros de entrada. Para exemplos de utilização, vide um *script* de qualidade da base de dados qualidade/bases/scripts/table_name/table_name.R, onde o `table_name` é o nome da tabela em *Amazon Athena*.

Acerca da confecção do relatório, utilizou-se o pacote *RMarkdown* para inserção dos textos e tabelas. Esta parte textual está presente em *API/relatorio*, nos arquivos correspondentes. Veja que alterações globais no relatório de qualidade devem ser realizadas nestes arquivos, a saber:

- qualidade_dados.Rmd;
- conteudos/disponibilidade.Rmd;
- conteudos/completude.Rmd;
- conteudos/conformidade.Rmd;
- conteudos/acuracia.Rmd;
- conteudos/consistencia.Rmd;
- conteudos/unicidade.Rmd;
- conteudos/temporalidade.Rmd;

Ressalta-se que os passos aqui mencionados referem-se à análise de uma tabela nova, partindo do princípio de nenhum conhecimento acerca desta. Uma vez realizados todos os passos descritos na Tabela a seguir, a atualização dos resultados podem ocorrer a qualquer momento, entretanto, sempre ocorre a verificação em qualidade/bases/scripts/base.R, através de `@function acrescimo_de_registros` se houve a inclusão de novos registros.

A confecção de um novo relatório de dados é descrita após a tabela a seguir, ficando estes salvos em qualidade/bases/relatorios/table_name.pdf. Já os dicionários de dados levantados, descritos na tabela abaixo como Dicionário de dados inicial é encontrado em qualidade/bases/dicionarios/table_name.csv, com os respectivos arquivos de tabulação .cnv (quando necessário) salvos em qualidade/bases/dicionarios/cnv/table_name/.

DESCRIÇÃO	ONDE? COMO?
Definição do database	`@param database_name`
Definição da tabela	`@param table_name`
Verificar disponibilidade da base de dados no Athena	Amazon Athena
Definição de substrings para obtenção de ano/mês/estado dos registros	`@param referencia_subs` *FILENAME* contendo *ES-01/01/2020* `referencia_subs <- data.frame("variavel"= "FILENAME", "inicio.ano"= 10, "qnt.ano"= 4, "inicio.mes"= 7, "qnt.mes"= 2, "inicio.uf"=1, "qnt.uf"=2)`
Levantamento da disponibilidade dos dados: falta algum ano/mês/estado?	`disponibilidade_arquivos <- function(database_name, table_name, referencia_subs)`
Procurar se esses arquivos faltantes existem. Caso existam, inserir no Athena	FTP e Amazon S3
Obtenção de microdados e arquivos auxiliares e de tabulação	TabWin
Obtenção das variáveis	`variaveis_tabela <- function(parametros_database)`
Levantamento preliminar de domínio existente, calculando tamanhos, classe, porcentagem de registros e completude	`tabela_dominio <- function(variaveis, referencia_subs, database_name, table_name)`
Realizar, onde for possível, casamento entre as variáveis de domínio finito, os arquivos de tabulação mais recentes e o dicionário de dados	Dicionário de dados inicial + Arquivos de tabulação
Análise das mudanças de variáveis: quem saiu? Quem entrou? Quem alterou?	Tabela de domínio + Dicionário de dados inicial + Relatório de integração
Buscar registros que identificam sem informação/ignorado	Tabela de domínio + `ignorados_agr <- function(variaveis, parametros_database, out=T)`

Definição de variáveis representando datas, seus formatos e referências	Dicionário de dados inicial. Conferir com o conteúdo das variáveis!
Definição de variáveis representando códigos de municípios e estados	Dicionário de dados inicial. Conferir com o conteúdo das variáveis!
Definição de variáveis com nomes para string matching	Dicionário de dados inicial. Conferir com o conteúdo das variáveis!
Levantamento de nomes para comparação (nomes de municípios, bairros, ...)	⟨https://www.google.com.br/⟩
Realizar matching	`string_matching <- function(dirty, clean, name)`
Definição da classe das variáveis	Dicionário de dados inicial. Conferir com o conteúdo das variáveis!
Preenchimento do dicionário de entrada dos scripts	Dicionário de dados padronizado
Diagnóstico geral	`diagnostico_agr <- function(variaveis, parametros_database)`
Conferir todo o diagnóstico geral	Observar menores e maiores completudes: faz sentido? Observar menores e maiores conformidades: faz sentido? Observar menores e maiores acurácias: faz sentido? Observar *NA*s: faz sentido? Foi encontrado alguma sequência de *whitespaces* ou zeros?
Definição das variáveis para temporalidade	`@param v_temp` `v_temp <- list(c("dt_receita", "dt_atendimento"))`

Definição das variáveis para unicidade	`@param variavel_id, @param variaveis_comp`
	`variavel_id <- "id_paciente";`
	`variaveis_comp <- c("dt_nascimento", "tp_sexo")`

Realização dos testes	relatorio-athena.R
Definição de testes de consistência	Dicionário de dados
Implementação dos testes de consistência. Note que estes devem ser salvos em arquivos junto as demais saídas	`consistencia_data_subs <- function(V1, V2, referencia_subs, sub, parametros_database)` `consistencia_query_subs <- function(variaveis, query, referencia_subs, sub, parametros_database)`

Para confecção do relatório automático, são necessários:

- script de qualidade da base de dados qualidade/bases/scripts/table_name/table_name.R;
- descrição da base de dados qualidade/bases/scripts/table_name/table_name.txt;
- resultados gerados automaticamente pela chamada das funções definidas (vide alguns exemplos para obter o formato) qualidade/bases/scripts/table_name/out/:
  - arquivos-faltantes-athena.csv;
  - diagnostico-sub.csv;
  - diagnostico.csv;
  - dominio-unique-athena.csv;

  - ignorados.csv;
  - inconformes-inacurados.csv;
  - informacoes-gerais.csv;
  - outliers.csv;
  - temporalidade-sub.csv;
  - temporalidade.csv;
  - unicidade-frequencia.csv;
  - unicidade-sub.csv;
  - unicidade.csv.
- dicionário de dados definido pelos passos apresentados anteriormente qualidade/bases/scripts/table_name/dict-table_name.csv;
- quantidade de registros da última atualização (opcional) qualidade/bases/scripts/table_name/table_name.info.

Com os arquivos gerados, basta carregar o *script* contido em API/relatorio.R, que o respectivo relatório será gerado em API/gerados, em um arquivo *.pdf* e em um arquivo .tex (qualidade-dados-table_name.pdf e qualidade-dados-table_name.tex). Destaca-se que caso sejam necessários adicionar mais análises a um relatório específico, recomenda-se que estas sema realizadas no arquivo .tex, e após seja realizada a compilação do mesmo em qualquer editor equivalente.

4 Exemplo de aplicação

4.1 *Script* de qualidade da base de dados

qualidade/bases/scripts/vinculasus_al.horus/horus.R

```
database_name <- "vinculasus_al"
table_name <- "horus"

referencia_subs <- data.frame("VARIAVEL" = "DT_ATENDIMENTO",
                              "INICIO.ANO" = 1, "QNT.ANO" = 4,
                              "INICIO.MES" = 6, "QNT.MES" = 2)

# Temporalidade
v_temp <- list(c("dt_receita", "dt_atendimento"))

# Unicidade
variavel_id <- "id_paciente"
variaveis_comp <- c("dt_nascimento", "tp_sexo")

source("R/relatorio-athena.R")

# Consistencia
    ######################################################################
source("R/consistencia.R")

parametros_database <- list("database_name"=database_name,
                            "table_name"=table_name)

cons_anual <- list(
  consistencia_data_subs("dt_receita", "dt_atendimento", referencia_subs, "ANO"
      , parametros_database),
  consistencia_data_subs("dt_nascimento", "dt_receita", referencia_subs, "ANO",
       parametros_database),
```

```
  consistencia_data_subs("dt_nascimento", "dt_atendimento",  referencia_subs, "ANO", parametros_database),
  consistencia_query_subs("\"ds_tipo_produto\" LIKE 'MEDICAMENTO' AND (NOT \"sg_tipo_produto\" LIKE 'M')", referencia_subs, "ANO", parametros_database),
  consistencia_query_subs("\"ds_tipo_produto\" LIKE 'PRODUTO PARA SAUDE' AND (NOT \"sg_tipo_produto\" LIKE 'I')", referencia_subs, "ANO", parametros_database)
)

for(i in seq(cons_anual))
  colnames(cons_anual[[i]]) <- c("ano", paste0("t", i), paste0("T", i))

dt <- Reduce(function(x, y) full_join(x, y, by = "ano"), cons_anual)
write.csv2(dt, paste0(path_to_data, "out/consistencia-ano.csv"), row.names = F)

testes_cons <- data.frame("teste" = c("T1", "T2", "T3", "T4", "T5"),
                          "desc" = c("dt_receita > dt_atendimento",
                                     "dt_nascimento > dt_receita",
                                     "dt_nascimento > dt_atendimento",
                                     "ds_tipo_produto == MEDICAMENTO & sg_tipo_produto != M",
                                     "ds_tipo_produto == PRODUTO PARA SAUDE & sg_tipo_produto != I"),
                          "desc_completa" = c("A data de emissao do paciente nao pode ser maior que a data de atendimento do paciente",
                                     "A data de nascimento do paciente nao pode ser maior que a data de emissao do paciente",
                                     "A data de nascimento do paciente nao pode ser maior que a data de atendimento do paciente",
                                     "Se a descricao do tipo de produto for \\textit{MEDICAMENTO}, a sigla do tipo de produto precisa ser \\textit{M}",
                                     "Se a descricao do tipo de produto for \\textit{PRODUTO PARA SAUDE}, a sigla do tipo de produto precisa ser \\textit{I}"))
write.csv2(testes_cons, paste0(path_to_data, "out/consistencia-descricao.csv"), row.names = F)

source("API/relatorio.R")
```

4.2 Dicionário de dados padronizado

Versão simplificada

VARIAVEL	CLASSE	MIN	MAX	FORMATO	REFERENCIA.INF	REFERENCIA.SUP	DIGITOS	POSITIVO	INTEIRO	RM.ZEROS	RM.WHITESPACE	IGNORADOS
CO_MUNICIPIO_IBGE	character	6	6				6			TRUE	TRUE	
CO_PRODUTO	character	1	6							TRUE	TRUE	
CO_SEQ_PACIENTE	character	1	12							TRUE	TRUE	
CO_TIPO_PRODUTO	character	1	6							TRUE	TRUE	
DS_PRODUTO	character	1	250							TRUE	TRUE	
DS_TIPO_PRODUTO	character	1	50							TRUE	TRUE	
DT_ATENDIMENTO	date	10	10	%Y-%m-%d	2011-01-01	2019-10-31				FALSE	FALSE	
DT_NASCIMENTO	date	10	10	%Y-%m-%d	1850-01-01	2019-10-31				FALSE	FALSE	
DT_RECEITA	date	10	10	%Y-%m-%d	1850-01-01	2019-10-31				FALSE	FALSE	
ID_PACIENTE	character	4	12							TRUE	TRUE	
NU_DIAS_DISPENSAR	numeric	1	3					TRUE	TRUE	FALSE	TRUE	
NU_FATOR_CORRECAO	numeric	1	6					TRUE	TRUE	FALSE	TRUE	
NU_PRODUTO	character	1	20							TRUE	TRUE	
QT_DOSE	numeric	1	6					TRUE	TRUE	FALSE	TRUE	
QT_POSOLOGIA	numeric	1	4					TRUE	TRUE	FALSE	TRUE	
QT_SOLICITADO	numeric	1	6					TRUE	TRUE	FALSE	TRUE	
SG_TIPO_PRODUTO	character	1	1							TRUE	TRUE	
ST_RENAME	character	1	1							TRUE	TRUE	
TP_PRODUTO	character	1	1							TRUE	TRUE	
TP_SEXO	character	1	1							TRUE	TRUE	I
VL_ITEM_DISPENSACAO	numeric	1	30					FALSE	FALSE	FALSE	TRUE	

5 Algumas considerações

- É necessário definir um arquivo `.Renviron` contendo informações das chaves de acesso ao sistema: *AWS*

```
AWS_ACCESS_KEY_ID = "*"
AWS_SECRET_ACCESS_KEY = "*"
AWS_DEFAULT_REGION = "sa-east-1"
```

- O pacote `bdacode` mencionado foi desenvolvido no âmbito deste projeto, e portanto, para se obter acesso deve-se procurar um dos responsáveis;
- Note que o processo de qualidade envolve o tráfego de uma grande quantidade de dados, logo não é um processo trivial, tampouco rápido. Para se ter uma ideia, todo o processo do `table_name` *horus* no `database_name` *vinculasus_al* (apenas o estado do Alagoas), dura cerca de 4h.

Dúvidas?

Alexandre Rodrigues ⟨arodrigues.ufes@gmail.com⟩
Leandro Furlam ⟨leandrofturi@gmail.com⟩
Elias Ribeiro ⟨elias.junior_@outlook.com⟩