Desafio

Lucas dos Santos Almeida

2025-09-01

Introdução

Este relatório descreve o processo de importação de dados para o ambiente R, utilizando dois formatos de arquivo distintos: Parquet e JSON. A importação foi realizada a partir de conjuntos de dados obtidos do Kaggle, com o objetivo de demonstrar a utilização das bibliotecas R mais adequadas para cada formato.

O processo de importação utiliza os pacotes Arrow (para o formato Parquet) e Jsonlite (para o formato JSON), além do pacote rio, que foi visto em aula. A documentação dos pacotes também pode ser consultada em sites como o Posit Cheatsheets, que fornecem informações essenciais sobre as bibliotecas de importação.

Importação de dados no formato parquet

Para a importação deste tipo de arquivo, a biblioteca recomendada é o pacote arrow. Está informação pode ser obtida atráves do site da Posit Cheatsheets, e também com ajuda das aulas ministradas no curso, que me auxiliaram durante a progressão desta atividade.

O arquivo utilizado neste exemplo, "Benign-Monday-no-metadata.parquet", foi carregado do kaggle. Este é um conjunto de dados acadêmicos de detecção de intrusão. Foi publicado pelo Instituto Canadense de Segurança Cibernética.

```
# Carregando os pacotes necessários
library(arrow)
library(rio)
```

```
# A função read_parquet lê arquivos com extensão .parquet

dados_read_parquet = read_parquet("Benign-Monday-no-metadata.parquet")

#Como uma alternativa, o pacote rio também foi utilizado. O rio detecta automaticamente o tipo do arqui


# Importando o mesmo arquivo usando a função 'import' do pacote rio


dados_rio = import("Benign-Monday-no-metadata.parquet"
                   )
#Ambos os métodos resultaram na criação de um objeto data.frame como especificado no enunciado
```

Importação de Dados no Formato JSON

Para importar arquivos JSON, a biblioteca padrão e mais recomendada no R é o pacote jsonlite, está descoberta foI graças a Inteligência Artificial e também as notas de aula de ME315

O conjunto de dados utilizado neste exemplo é o "pokemonDB_dataset.json", também retirado do Kaggle.

Este conjunto de dados inclui dois diretórios com uma coleção de aproximadamente 32.000 imagens de Pokémon. Contém vários detalhes sobre cada Pokémon, como nome, tipo, espécie, altura, peso, habilidades, rendimento de EV, taxa de captura, amizade base, experiência base, taxa de crescimento, grupos de ovos, gênero, ciclos de ovos e estatísticas base.

```
# Carregando a biblioteca para importação de arquivos JSON
library(jsonlite)

# Carregando o conjunto de dados JSON, a função que importa os dados com a exstensão json é FromJSON

dados_jsonlite = fromJSON("pokemonDB_dataset.json")

#Similarmente como fizemos acima, iremos importar o mesmo conjunto de dados, porém usando o pacote RIO,

# Importando o mesmo arquivo usando a função 'import' do pacote rio

dados_riojson = import("pokemonDB_dataset.json")
```

Os links acessados durante a realização deste desafio se encontrão aqui

- Link Posit Cheatsheets (Site utilizado para o melhor entendimento dos pacotes)
- Link do primeiro arquivo, “https://www.kaggle.com/datasets/dhoogla/cicids2017”, obtido atráves do site Kaggle.
- Link do segundo arquivo, “https://www.kaggle.com/datasets/divyanshusingh369/complete-pokemon-library-32k-images-and-csv?resource=download”

JSON (JavaScript Object Notation)

O JSON, sigla para JavaScript Object Notation, é um formato de intercâmbio de dados de texto que se originou no início dos anos 2000. Sua criação é amplamente atribuída a Douglas Crockford, que, junto a outros membros da State Software, buscava uma alternativa ao XML que fosse mais leve e de fácil leitura para a comunicação assíncrona entre o servidor e o navegador.

A principal característica do JSON é sua sintaxe, que representa dados estruturados em uma notação baseada em pares de chave-valor. Essa sintaxe, herdada da linguagem JavaScript, facilitou sua adoção na comunidade de desenvolvimento web, tornando-o o formato padrão para APIs RESTful e o principal método de comunicação em aplicações modernas. Em 2013, o JSON foi formalmente padronizado pela ECMA International (padrão ECMA-404), solidificando seu status como um formato universal para a representação e a troca de dados estruturados.

Parquet

O Parquet, diferentemente do JSON, foi concebido com um propósito específico no contexto do Big Data: a otimização do armazenamento e do processamento de dados em larga escala. O projeto foi iniciado em 2013 por engenheiros do Twitter e da Cloudera, que buscavam um formato de arquivo mais eficiente para ser utilizado em ecossistemas como o Apache Hadoop.

A inovação fundamental do Parquet reside em sua arquitetura colunar. Em vez de armazenar os dados linha por linha, o Parquet os armazena por coluna. Essa abordagem oferece diversas vantagens para o processamento analítico:

Compressão Eficiente: Dados de uma mesma coluna tendem a ter o mesmo tipo, o que permite o uso de algoritmos de compressão mais eficazes, reduzindo o tamanho dos arquivos.

Leitura Seletiva: Para consultas que exigem apenas um subconjunto de colunas, o Parquet permite que o sistema de processamento leia apenas os blocos de dados necessários, ignorando o restante do arquivo.

Otimização para Processamento Analítico: O formato colunar é ideal para cargas de trabalho de data warehouse e análise de dados que escaneiam um grande número de linhas para computar agregações em poucas colunas.

O Parquet é um projeto de código aberto sob a Apache Software Foundation e tornou-se o formato de armazenamento padrão para processamento analítico em plataformas como Apache Spark, Hive e Presto.

Quando usar cada um??

Os formatos JSON e Parquet atendem a propósitos distintos e complementares no ecossistema de dados.

Use JSON para:

- Intercâmbio de Dados na Web: É a escolha ideal para comunicação entre um cliente (como um navegador ou aplicativo móvel) e um servidor. É o formato principal em APIs RESTful e GraphQL devido à sua flexibilidade e simplicidade.
- Legibilidade e Simplicidade: Por ser um formato de texto, ele é fácil de ser lido, escrito e depurado por desenvolvedores, acelerando o ciclo de desenvolvimento de software.
- Flexibilidade de Esquema: O JSON não impõe um esquema rígido. Diferentes objetos em um mesmo arquivo podem ter estruturas variadas, o que é útil para dados semiestruturados, como registros de log ou informações de perfis de usuário.

Use Parquet para:

- Análise de Big Data: É o formato preferido para armazenar dados em data warehouses e data lakes. Ele é otimizado para plataformas como Apache Spark, Hive e Presto, que realizam consultas analíticas em vastos volumes de dados.
- Eficiência de Armazenamento e Custo: A compressão avançada do Parquet reduz significativamente o tamanho dos arquivos, diminuindo os custos de armazenamento em nuvem e o consumo de largura de banda.
- Performance em Consultas Analíticas: Por armazenar dados por coluna, o Parquet permite que os sistemas de processamento leiam apenas as colunas necessárias para uma consulta. Isso acelera drasticamente as operações analíticas e de agregação, como GROUP BY e AVG, que geralmente envolvem um subconjunto das colunas.
- Estrutura de Esquema: Diferentemente do JSON, o Parquet é um formato com esquema. Isso garante consistência nos dados e permite otimizações no lado do processamento, tornando-o ideal para dados estruturados.

Referências

- JSON:

Crockford, D. (2006). JSON: The Fat-Free Alternative to XML. Disponível em: http://www.json.org/xml.html

ECMA International. (2013). The JSON Data Interchange Syntax (ECMA-404).

O'Reilly, T. (2012). What Is JSON?. O'Reilly Media.

- Parquet:

Apache Parquet. (2023). About Apache Parquet. Disponível em: https://parquet.apache.org/documentation/latest/

Twitter Engineering Blog. (2013). Dremel Made Simple with Parquet. Disponível em: https://blog.twitter.com/engineering/en_us/a/2013/dremel-made-simple-with-parquet

Cloudera Blog. (2013). Announcing the Parquet Project. Disponível em: https://blog.cloudera.com/announcing-the-parquet-project/