Sistema de Cache com plotagem gráfica implementado em REDIS com linguagem R

Luiz Paulo T. Gonçalves

2023-04-14

O sistema de banco de dados escolhido

Entre as inúmeras opções de Sistema Gerenciador de Banco de Dados NoSQL, o Redis destaca-se quando o assunto é cache. Dado sua velocidade e mecanismo de chave-valor, portanto, no presente projeto de Cache o Redis foi o escolhido.

Qual tipo de função (gráfico) para ser cacheado

Os gráficos que serão cacheados são gráficos de distribuição, a saber: histograma e gráfico de densidade. Tais gráficos retornam a distribuoção dos dados, assim, possibilitando que o analista tome uma decisão de modelo mais apropriado de acordo com a distribuição. Por exemplo, uma distribuiçao binomial é mais apropriado em determinada modelagem estatística enquanto os que têm distribuição Gaussiana se enquadra melhor em outra modelagem.

Em que formato os gráficos serão armazenados

Qual a solução adotada para geração dos índices

O projeto da solução em linguagem R

Construindo a função em Linguagem R

```
# Dependências\Packages Utilizados
library(tidyverse)
library(DataExplorer)
library(redux)
library(readxl)
```

```
CachePlot <- function(db = as.data.frame(),
                      type = as.character()){

# Conectando ao Redis localmente
  redis_conn <- redux::redis_connection(redis_config())
```

```
  r_conex <- redux::hiredis()

# Gerando a chave
key_redis <- paste0("plot_", type, "_", base::deparse(substitute(db)))

# Verificar se o gráfico já foi plotado anteriormente

  if(r_conex$EXISTS(key_redis) == length(key_redis)){
    # Retorna o plote salvo na key-value
    print(r_conex$PING()) # Para o usuário saber se está conectado ao Redis
    plot_redis <- base::unserialize(r_conex$GET(key = key_redis))
    cat(key_redis, "Disponível no banco de dados Redis!")
    print(plot_redis)

  }else{

# Condicional abre caso não tenha o plote no banco de dados

      if(type == "histogram"){

        histograma = DataExplorer::plot_histogram(data = db)
        hist_serialized = base::serialize(histograma, NULL)
        # Guardando no banco de dados
        print(r_conex$PING())
        r_conex$SET(key = key_redis, hist_serialized)
        cat("Plot", key_redis, "armazenado no Redis com sucesso!")


      } else if(type == "density"){

        density = DataExplorer::plot_density(data = db)
        den_serialized = base::serialize(density, NULL)
        # Guardando no banco de dados
        print(r_conex$PING())
        r_conex$SET(key = key_redis, den_serialized)
        cat("Plot", key_redis, "armazenado no Redis com sucesso!")

      } else{

        stop("ERRO: tipo de plote não disponível no sistema de Cache-Redis")

          }

      }

  }
```

Chamando a Função

```
CachePlot(db = iris,
          type = "density")
```

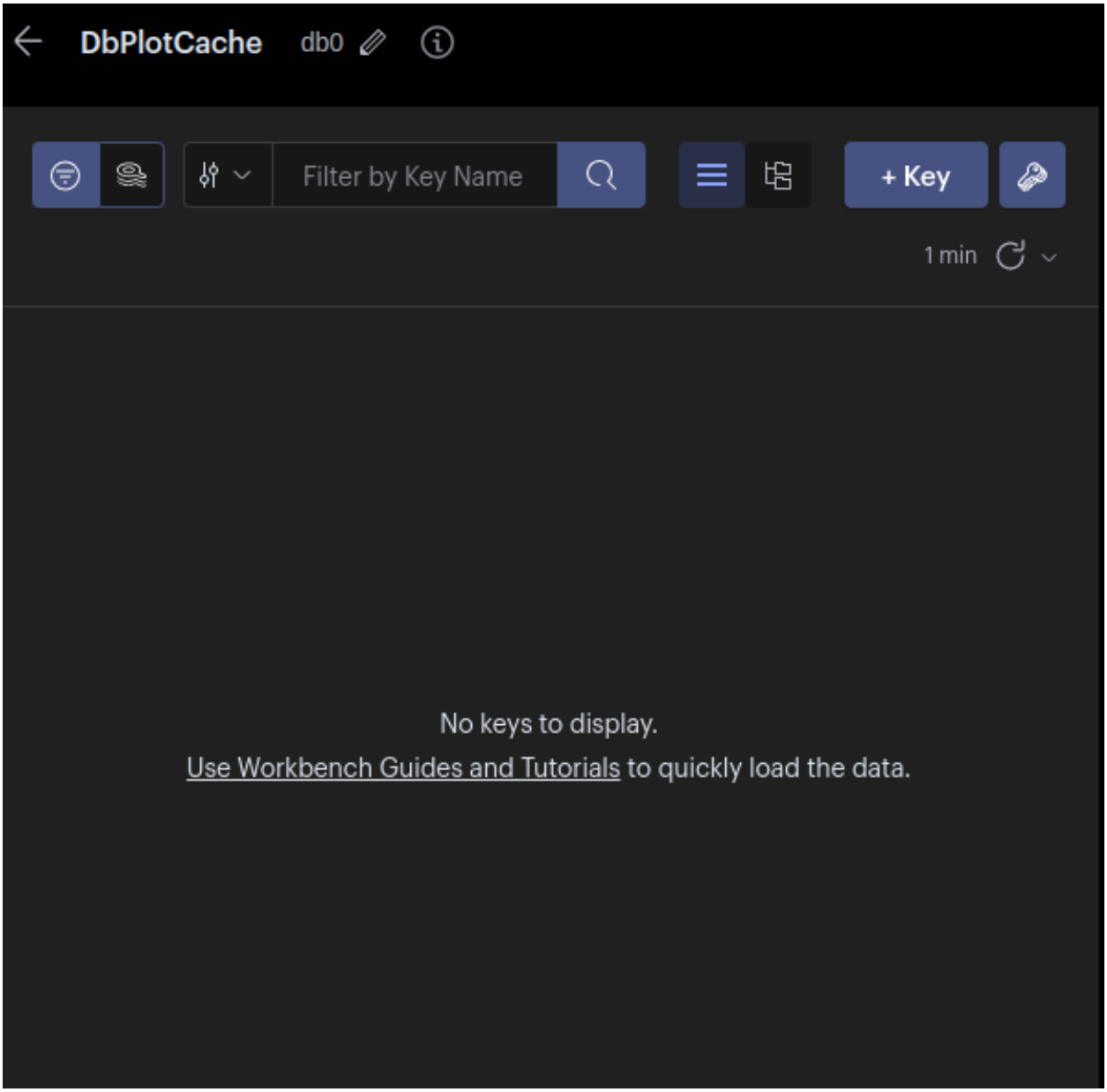
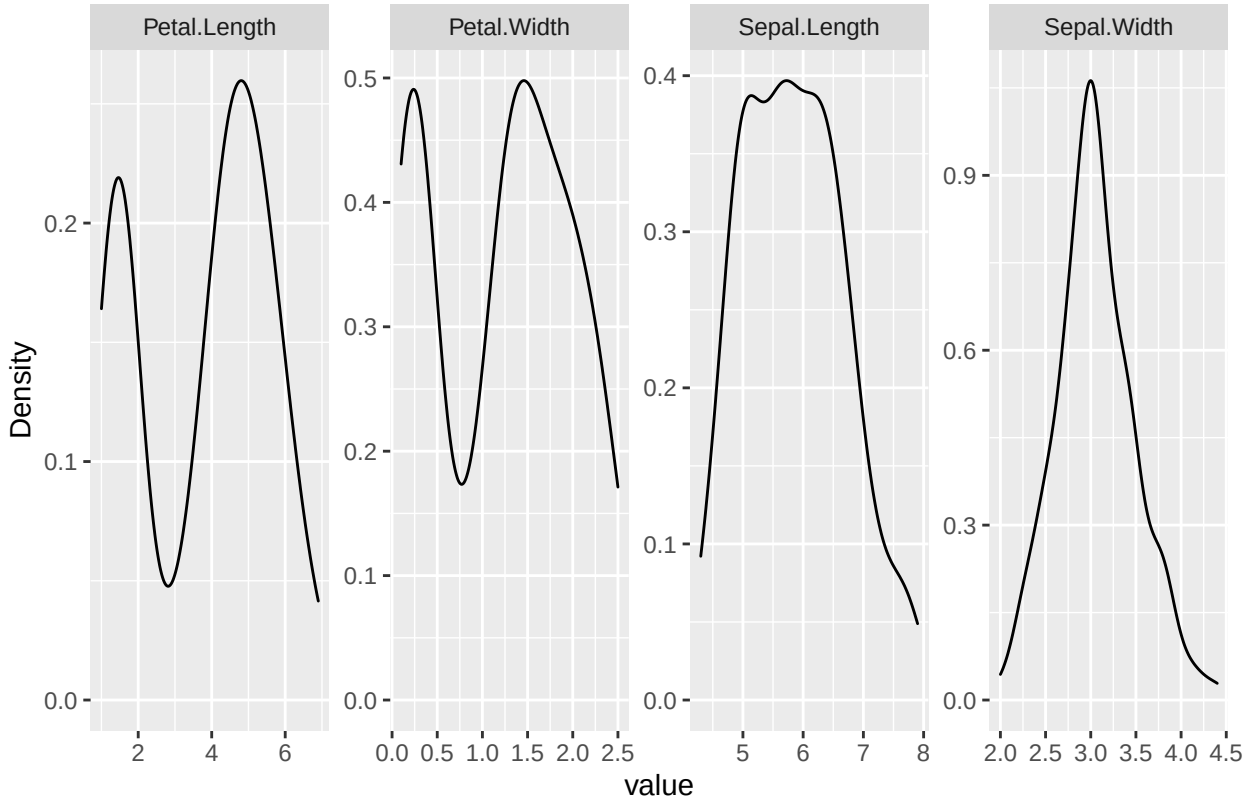


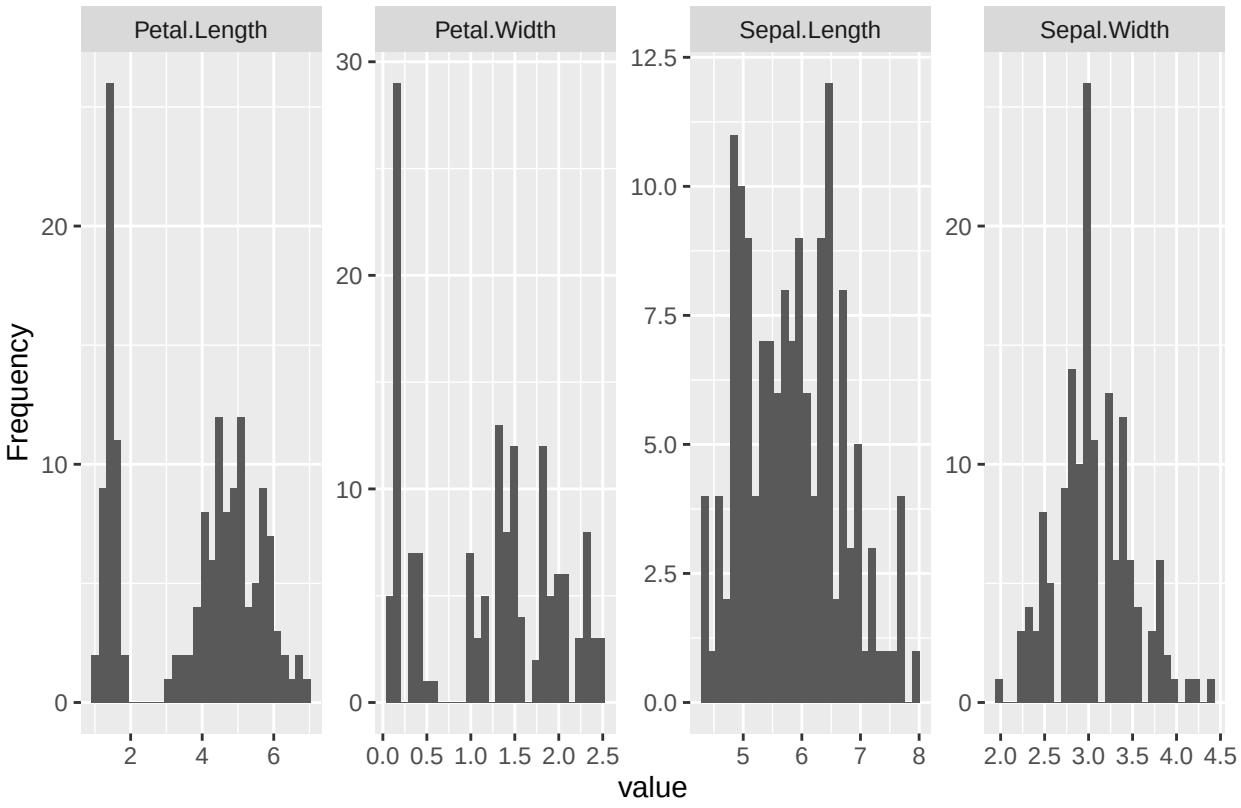
Figure 1: Redis Vazio

```
## [Redis: PONG]
## plot_density_iris Disponível no banco de dados Redis!$page_1
```




```
CachePlot(db = iris,
          type = "histogram")
```

```
## [Redis: PONG]
## plot_histogram_iris Disponível no banco de dados Redis!$page_1
```


Petal.Length
Petal.Width
Sepal.Length
Sepal.Width
Frequency
value
0
10
20
30
0.0
2.5
5.0
7.5
10.0
12.5
2
4
6
0.0 0.5 1.0 1.5 2.0 2.5
5
6
7
8
2.0 2.5 3.0 3.5 4.0 4.5

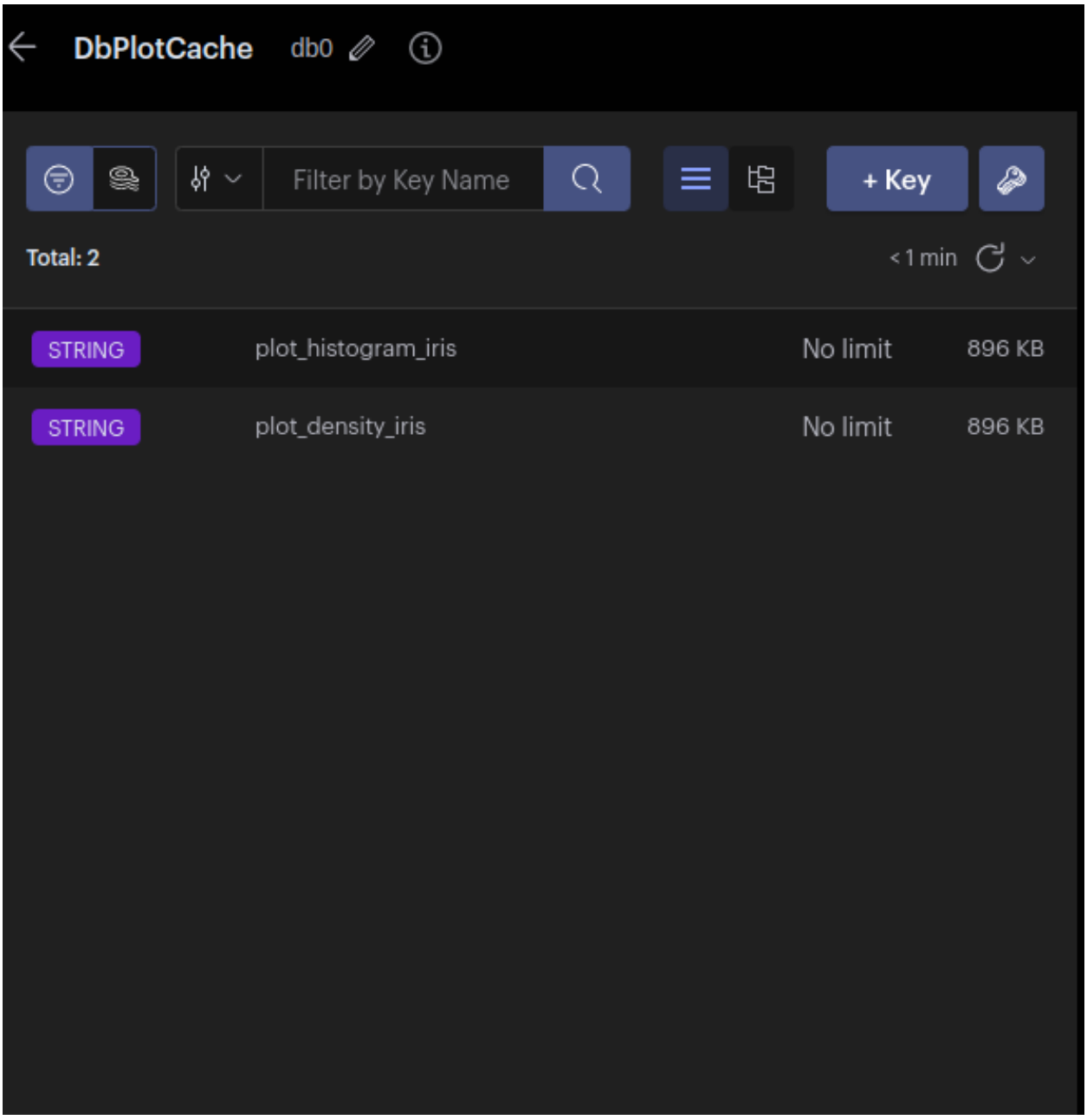


Figure 2: Redis com os plots gerados

```
> db_iris = iris
> CachePlot(db = iris,
+           type = "density")
[Redis: PONG]
Plot plot_density_iris armazenado no Redis com sucesso!
> CachePlot(db = iris,
+           type = "histogram")
[Redis: PONG]
Plot plot_histogram_iris armazenado no Redis com sucesso!
```

Figure 3: Tela do Usuário