Quarto

Table of contents

1 Quarto

Quarto enables you to weave together content and executable code into a finished document. To learn more about Quarto see https://quarto.org.

1.1 Running Code

R version

```
print(R.version)
#>                _
#> platform       aarch64-apple-darwin20
#> arch           aarch64
#> os             darwin20
#> system         aarch64, darwin20
#> status
#> major          4
#> minor          4.0
#> year           2024
#> month          04
#> day            24
#> svn rev        86474
#> language       R
```

```
#> version.string R version 4.4.0 (2024-04-24)
#> nickname       Puppy Cup
```

Python version

```
import sys
sys.version
#> '3.11.10 | packaged by conda-forge | (main, Oct 16 2024, 01:28:12) [Clang 17.0.6 ]'
```

```
#install.packages("sparklyr")
library(sparklyr)
#>
#> Attaching package: 'sparklyr'
#> The following object is masked from 'package:stats':
#>
#>     filter
#install.packages("pysparklyr")
library("pysparklyr")
library(devtools)
#> Loading required package: usethis
library(httr)
#install_github("nagdevAmruthnath/minio.s3")
library("minio.s3")
library(aws.s3)
#> Registered S3 methods overwritten by 'aws.s3':
#>   method                  from
#>   as.data.frame.s3_bucket minio.s3
#>   print.aws_error         minio.s3
#>   print.s3_bucket         minio.s3
#>   print.s3_object         minio.s3
#>
#> Attaching package: 'aws.s3'
#> The following objects are masked from 'package:minio.s3':
#>
#>     bucket_exists, bucket_list_df, bucketexists, bucketlist,
#>     copy_bucket, copy_object, copybucket, copyobject, delete_bucket,
#>     delete_bucket_policy, delete_cors, delete_lifecycle, delete_object,
#>     delete_replication, delete_tagging, delete_website, deletebucket,
#>     deleteobject, get_acceleration, get_acl, get_bucket, get_bucket_df,
#>     get_bucket_policy, get_bucketname, get_cors, get_lifecycle,
#>     get_location, get_notification, get_object, get_replication,
```

```
#>     get_requestpayment, get_tagging, get_torrent, get_uploads,
#>     get_versioning, get_versions, get_website, getbucket, getobject,
#>     head_object, headobject, put_acceleration, put_acl, put_bucket,
#>     put_bucket_policy, put_cors, put_folder, put_lifecycle,
#>     put_notification, put_object, put_replication, put_requestpayment,
#>     put_tagging, put_versioning, put_website, putbucket, putobject,
#>     s3HTTP, s3load, s3read_using, s3readRDS, s3save, s3save_image,
#>     s3saveRDS, s3source, s3sync, s3write_using, save_object, saveobject
library(tidyverse)
#> -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
#> v dplyr     1.1.4     v readr     2.1.5
#> v forcats   1.0.0     v stringr   1.5.1
#> v ggplot2   3.5.1     v tibble    3.2.1
#> v lubridate 1.9.3     v tidyr     1.3.1
#> v purrr     1.0.2
#> -- Conflicts ------------------------------------------ tidyverse_conflicts() --
#> x dplyr::filter() masks sparklyr::filter(), stats::filter()
#> x purrr::invoke() masks sparklyr::invoke()
#> x dplyr::lag()    masks stats::lag()
#> i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to be
library(dplyr)
library(readr)
library(DBI)
```

1.2 Local Spark Cluster

```
#sparklyr::spark_install(version = "3.5.0")
spark_installed_versions()
#>   spark hadoop                                          dir
#> 1 2.4.3    2.7 /Users/ben/spark/spark-2.4.3-bin-hadoop2.7
#> 2 3.5.0      3   /Users/ben/spark/spark-3.5.0-bin-hadoop3
sc <- spark_connect(master = "local")
```

```
# Setting the environment variables for MinIO
# https://localhost:9001/browser/templategenerator
set_config(config(ssl_verifyhost = 0L, ssl_verifypeer = 0L))
Sys.setenv("AWS_ACCESS_KEY_ID" = "health",
           "AWS_SECRET_ACCESS_KEY" = "NOentry#23",
           "AWS_SSL_ENABLED" = "TRUE",
           "AWS_S3_ENDPOINT" = "127.0.0.1:9000")
```

```
# Get file from the MinIO server
b <- get_bucket(bucket = 'templategenerator', region = "", use_https = TRUE)
iris <- aws.s3::s3read_using(FUN = read.csv, object = "iris.csv", bucket = b, opts = list(use

# Spark dataframe
iris_tbl <- copy_to(sc, iris)

# Show dataframe
iris_tbl
#> # Source:   table<`iris`> [?? x 5]
#> # Database: spark_connection
#>    sepal_length sepal_width petal_length petal_width species
#>           <dbl>       <dbl>        <dbl>       <dbl> <chr>
#>  1          5.1         3.5          1.4         0.2 setosa
#>  2          4.9         3            1.4         0.2 setosa
#>  3          4.7         3.2          1.3         0.2 setosa
#>  4          4.6         3.1          1.5         0.2 setosa
#>  5          5           3.6          1.4         0.2 setosa
#>  6          5.4         3.9          1.7         0.4 setosa
#>  7          4.6         3.4          1.4         0.3 setosa
#>  8          5           3.4          1.5         0.2 setosa
#>  9          4.4         2.9          1.4         0.2 setosa
#> 10          4.9         3.1          1.5         0.1 setosa
#> # i more rows
```

```
# Use SQL on Spark
dbGetQuery(sc, "SELECT count(*) FROM iris")
#>   count(1)
#> 1      150

# Disconnect
spark_disconnect(sc)
```