Data Science Workshop

Safouene Kallel

2022-11-24

SQL

- SQL is a ... language.
- ... clause is used to choose the columns
- ... clause is used to filter the rows
- ... gets rid of the duplicates
- ... is used to create aggregated summaries
- A Primary key is ...

SQL

Please complete the Query:

```
SELECT Sales.*, Customers.*
FROM Sales
LEFT JOIN Customers
... Customers.CustomerID = Sales.CustomerID
```

R & RStudio

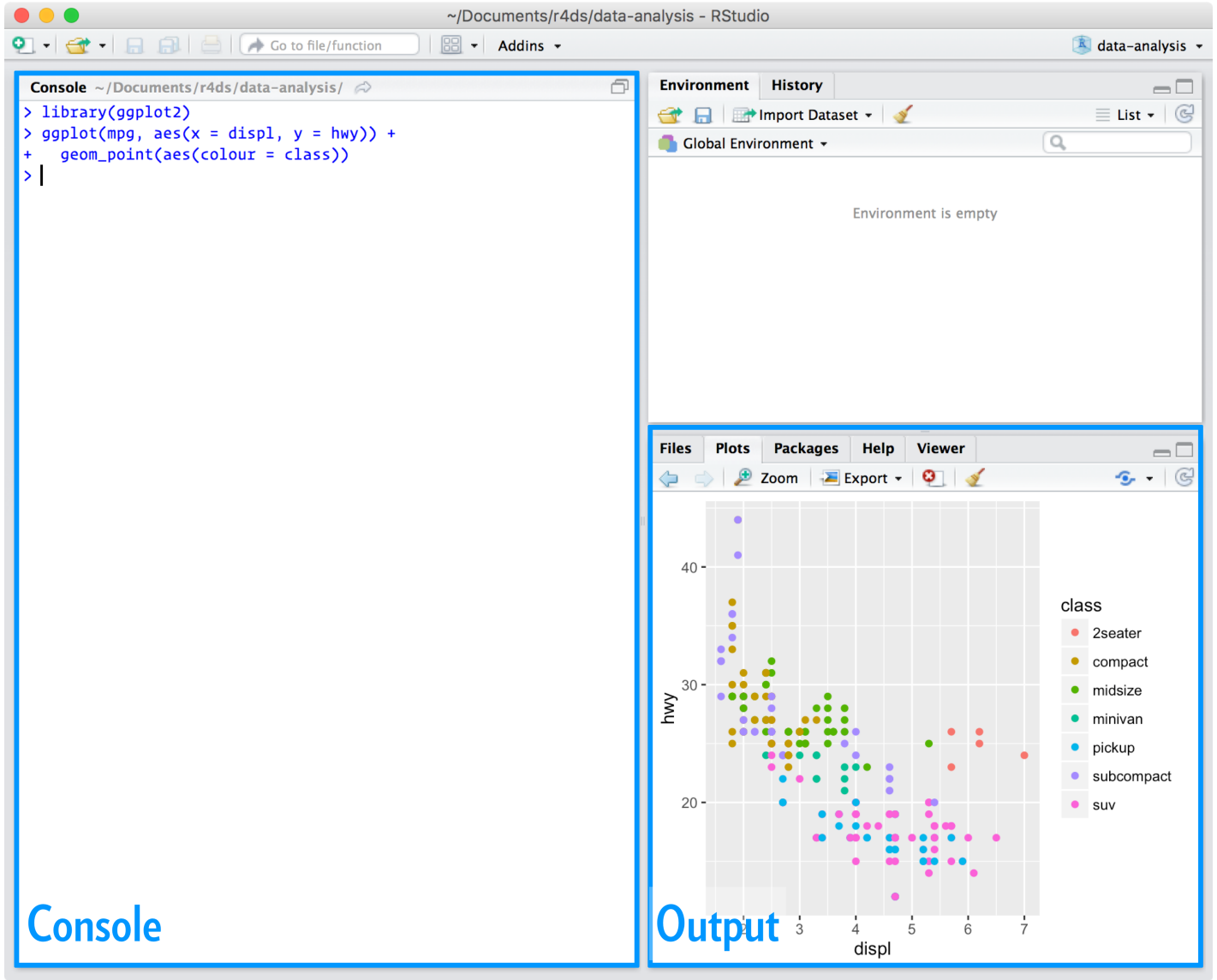

Figure 1: RStudio Console

Create a Project

Please do this at all times.

Please create a new Project File -> New Directory -> New Project

R Packages

R Packages are extensions to the R language. They contain code (functions), data and documentation.

```
## installing ODBC on our Computer
install.packages("odbc")
install.packages("RMySQL")
```

```
## loading odbc in our environment
library(odbc)
library(RMySQL)
```

Connecting to the Database

```
## calling a function from odbc

db_user <- 'DS_Training_u01'
db_password <- ''
db_name <- 'DS_Training'
db_host <- 'stgux-db.vih.infineon.com'
db_port <- 3306

conn <-  dbConnect(MySQL(),
                   user = db_user,
                   password = db_password,
                   dbname = db_name,
                   host = db_host,
                   port = db_port)
```

R Packages

Figure 2: R Packages

R Packages

Now we created a connection with the database, how do we get the data into our computer.

```
install.packages("tidyverse")
```

```
library(dbplyr)
library(dplyr)
```

Figure 3: Tidyverse

SQL in R

in order to get data from a database, you have several choices:

Write a SQL Query and Send it with R:

```
res <- dbSendQuery(conn,
                   "SELECT *
                   FROM employees
                   WHERE EmployeeID = 2")
dbFetch(res)
dbClearResult(res)
dbDisconnect(conn)
```

dplyr

```
## collecting data from the database
Employees <- tbl(conn, "employees") %>% collect()
```

```
## filtering data
Employees <- Employees %>% filter(EmployeeID==7)
```

```
## the preferred way

Employees <- tbl(conn, "employees") %>%
  filter(EmployeeID==7) %>%
  collect()
```

filter first (on the server side), load later
no need to fill the memory with data we are not going to use

group by, summarize

```
Sales <- tbl(conn,"Sales") %>%
  group_by(ProductID) %>%
  summarise(n=n(),sum=sum(Quantity)) %>%
  collect()
```

mutate

```
Sales <- Sales %>%
  mutate(Average= sum/n) %>%
  arrange(desc(Average))
```

ggplot2, visualize

```
library(ggplot2)
```

```
sp1 <- ggplot(Sales,aes(x=n,y=Average)) +
  geom_point()

sp1
```

Cheat sheet

https://github.com/rstudio/cheatsheets/blob/main/data-visualization-2.1.pdf

Exercise

Pulsed IV data

1. load .csv file in your Environment ({readr})
2. filter data to have only Example = 1
3. create a plot to show the raw data (x axis : V2, y axis: I)
4. create a line between the points
5. add colors
6. modify labels (to add the units)
7. modify title
8. get the maximum value of current per line the data

Solution

```
library(readr)
library(dplyr)

pulsedIV <- read_csv("./Data/TestID2.csv")

pulsedIV <- pulsedIV %>%
  filter(Example==1)
```

Solution

```
ggplot(pulsedIV,aes(x=V2, y=I)) +
  geom_point()
```

Solution

```
ggplot(pulsedIV,aes(x=V2, y=I)) +
  geom_point() + geom_line()
```

```
ggplot(pulsedIV,aes(x=V2,
                    y=I,
                    group = V1)) +
  geom_point() +
  geom_line()
```

Solution

```
ggplot(pulsedIV,aes(x=V2,
                    y=I,
                    group = V1,
                    color=V1)) +
  geom_point() +
  geom_line()
```

```
ggplot(pulsedIV,aes(x=V2,
                    y=I,
                    group = V1,
                    color=as.factor(V1))) +
  geom_point() +
  geom_line()
```

Solution

```
ggplot(pulsedIV,aes(x=V2,
                    y=I,
                    group = V1,
                    color=as.factor(V1))) +
  geom_point() +
  geom_line() +
  xlab("V1 [V]") +
  ylab("I [A]") +
  labs(color="V2 [V]") +
  theme_light() +
  ggtitle("IV plot")
```

Solution

```
pulsedIV_summary <- pulsedIV %>%
  group_by(V1) %>%
  summarise(maxI=max(I, na.rm = TRUE))
```

```
ggplot(pulsedIV_summary, aes(x=V1,
                             y=maxI)) +
  geom_point() +
  geom_line()
```