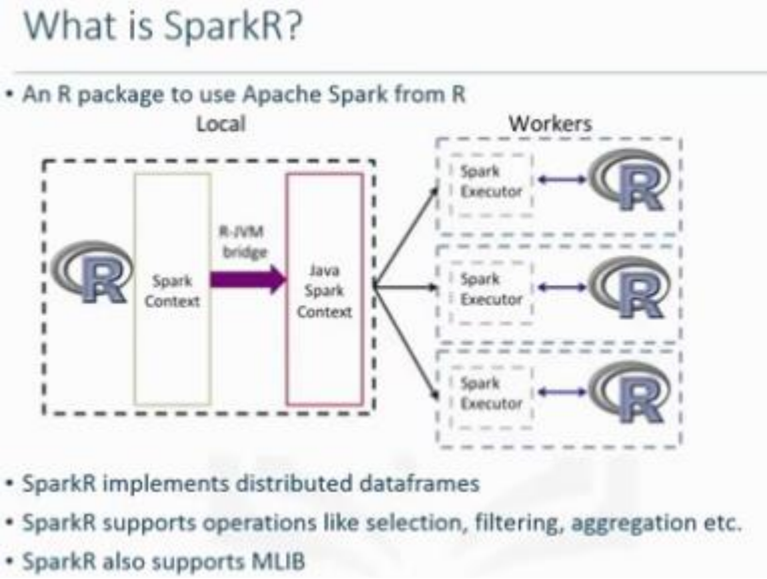
Lesson objectives

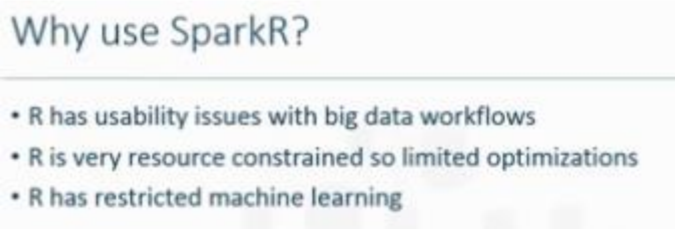
- Learn what SparkR is
- Learn why you would use SparkR
- List the features of SparkR
- Understand the interfaces into SparkR



What is SparkR?

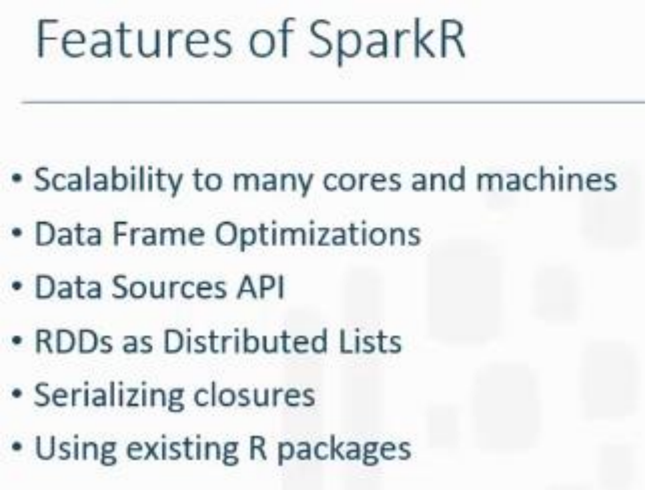
- An R package to use Apache Spark from R

- SparkR implements distributed dataframes
- SparkR supports operations like selection, filtering, aggregation etc.
- SparkR also supports MLIB



Why use SparkR?

- R has usability issues with big data workflows
- R is very resource constrained so limited optimizations
- R has restricted machine learning

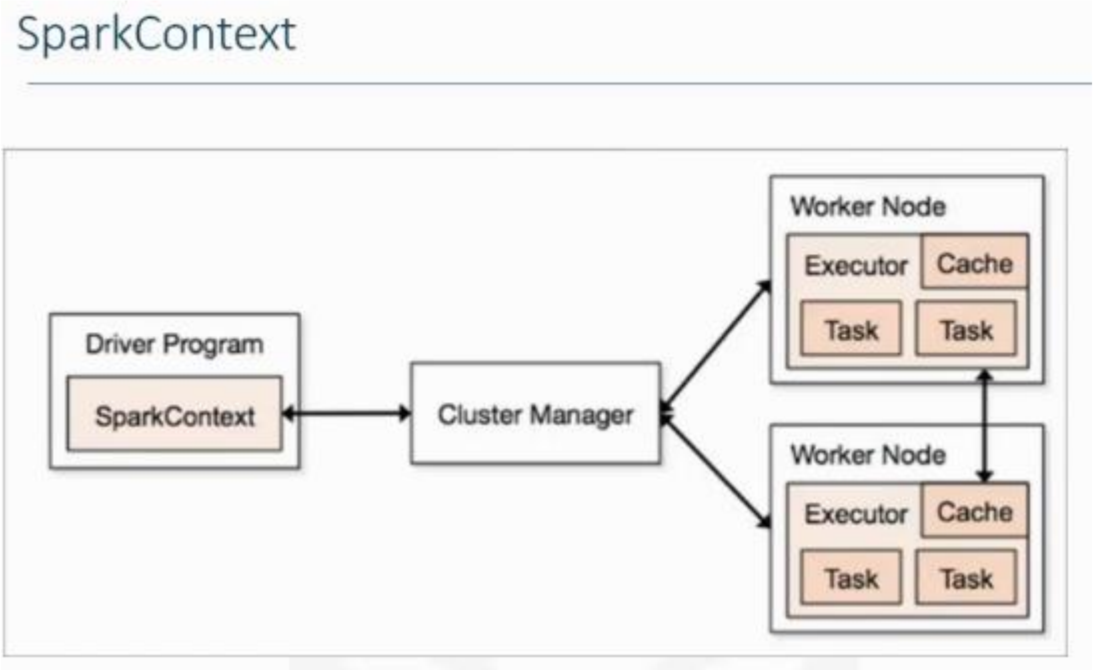


Features of SparkR

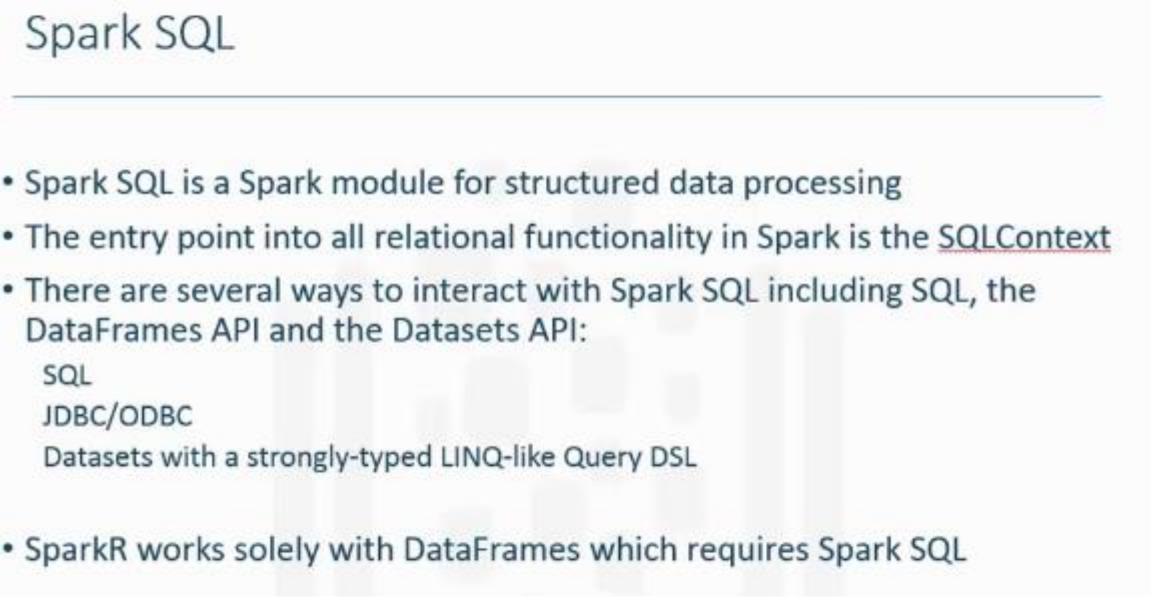
- Scalability to many cores and machines
- Data Frame Optimizations
- Data Sources API
- RDDs as Distributed Lists
- Serializing closures
- Using existing R packages

Interfaces

- Spark shell
- SparkR shell
- Rstudio
- Notebooks
- Data Scientist Workbench (an interactive service providing notebooks interface)



SparkContext



Spark SQL

- Spark SQL is a Spark module for structured data processing
- The entry point into all relational functionality in Spark is the SQLContext
- There are several ways to interact with Spark SQL including SQL, the DataFrames API and the Datasets API:

  SQL

  JDBC/ODBC

  Datasets with a strongly-typed LINQ-like Query DSL

- SparkR works solely with DataFrames which requires Spark SQL

Spark shell

- The following command is used to open Spark shell:
  - Spark-shell

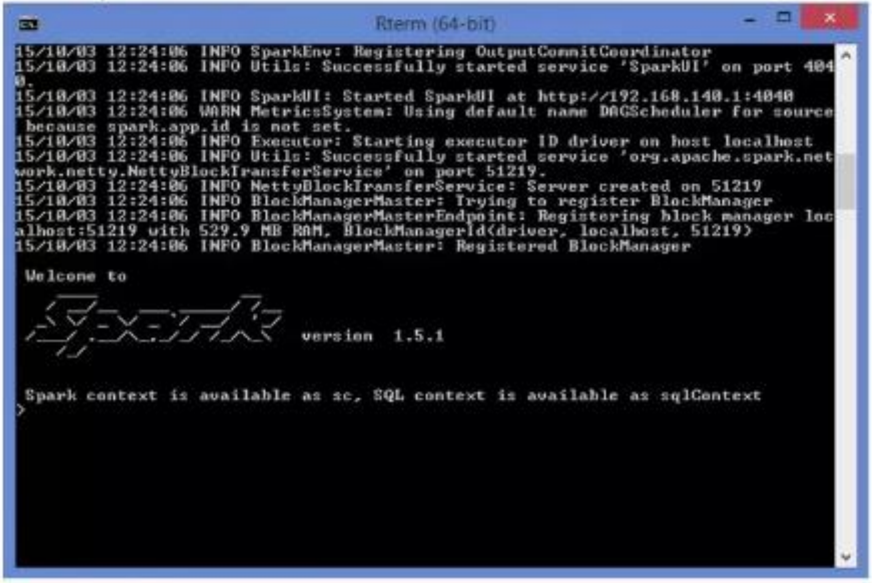
```
$ ./bin/spark-shell
Spark context available as sc.
SQL context available as sqlContext.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 1.6.0-SNAPSHOT
      /_/

Using Scala version 2.11.7 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_66)
Type in expressions to have them evaluated.
Type :help for more information.

scala>
```

- create a SparkContext and an sqlContext

sc <- sparkR.init()
sqlContext <- sparkRSQL.init(sc)

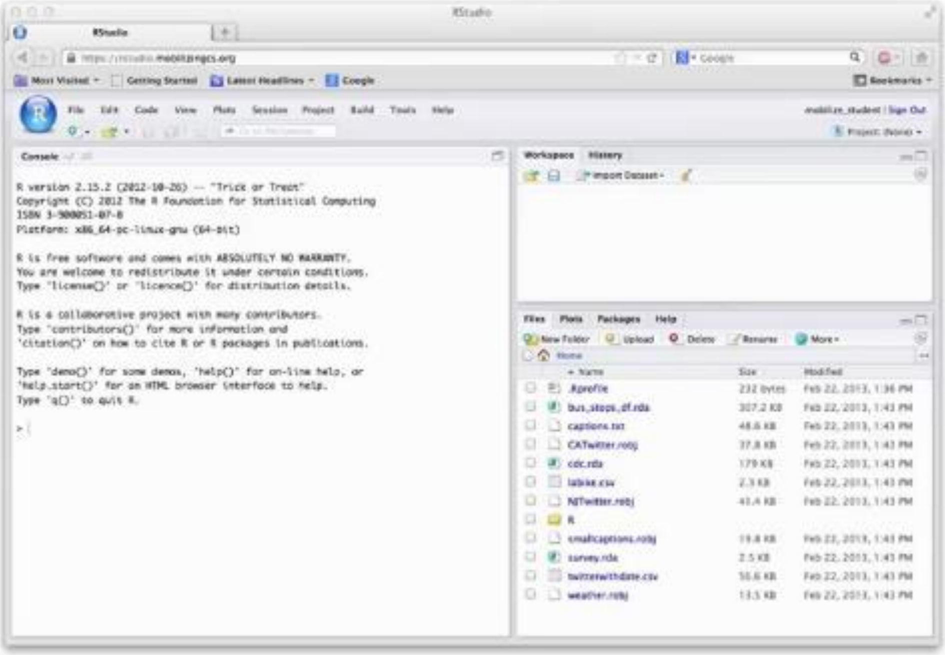
SparkR shell

- The following command is used to open SparkR shell:
  - sparkR shell



```
15/10/03 12:24:06 INFO SparkEnv: Registering OutputCommitCoordinator
15/10/03 12:24:06 INFO Utils: Successfully started service 'SparkUI' on port 404
0.
15/10/03 12:24:06 INFO SparkUI: Started SparkUI at http://192.168.140.1:4040
15/10/03 12:24:06 WARN MetricsSystem: Using default name DAGScheduler for source
 because spark.app.id is not set.
15/10/03 12:24:06 INFO Executor: Starting executor ID driver on host localhost
15/10/03 12:24:06 INFO Utils: Successfully started service 'org.apache.spark.net
work.netty.NettyBlockTransferService' on port 51219.
15/10/03 12:24:06 INFO NettyBlockTransferService: Server created on 51219
15/10/03 12:24:06 INFO BlockManagerMaster: Trying to register BlockManager
15/10/03 12:24:06 INFO BlockManagerMasterEndpoint: Registering block manager loc
alhost:51219 with 529.9 MB RAM, BlockManagerId(driver, localhost, 51219)
15/10/03 12:24:06 INFO BlockManagerMaster: Registered BlockManager

 Welcome to
    ____              __
   / __/__  ___ _____/ /__
  _\ \/ _ \/ _ `/ __/  '_/
 /___/ .__/\_,_/_/ /_/\_\   version  1.5.1
    /_/


 Spark context is available as sc, SQL context is available as sqlContext
>
```

RStudio

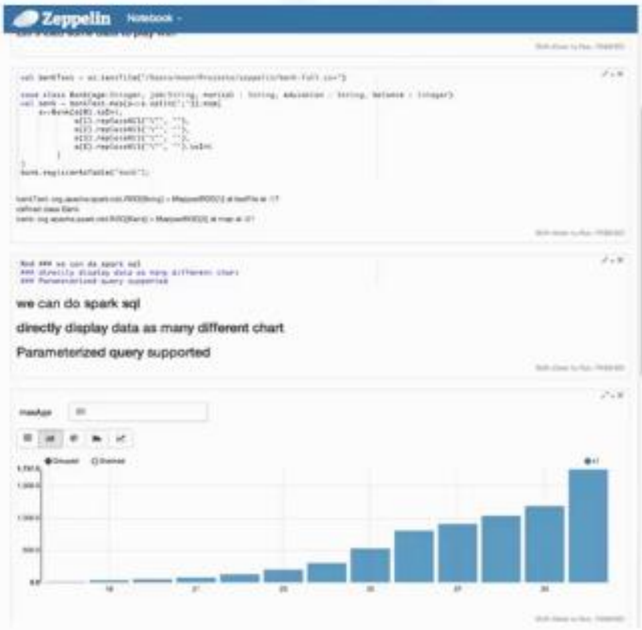


```
R version 2.15.2 (2012-10-26) -- "Trick or Treat"
Copyright (C) 2012 The R Foundation for Statistical Computing
ISBN 3-900051-07-0
Platform: x86_64-pc-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> |
```

Name	Size	Modified
.Rprofile	232 bytes	Feb 22, 2013, 1:36 PM
bus_stops_df.rda	307.2 KB	Feb 22, 2013, 1:43 PM
captions.txt	48.6 KB	Feb 22, 2013, 1:43 PM
CATwitter.robj	37.8 KB	Feb 22, 2013, 1:43 PM
cdc.rda	179 KB	Feb 22, 2013, 1:43 PM
labike.csv	2.3 KB	Feb 22, 2013, 1:43 PM
NJTwitter.robj	41.4 KB	Feb 22, 2013, 1:43 PM
R		
smallcaptions.robj	19.8 KB	Feb 22, 2013, 1:43 PM
survey.rda	2.5 KB	Feb 22, 2013, 1:43 PM
twitterwithdate.csv	50.6 KB	Feb 22, 2013, 1:43 PM
weather.robj	13.5 KB	Feb 22, 2013, 1:43 PM

Notebooks

- Zeppelin notebook



```
val bankText = sc.textFile("/Users/moon/Projects/zeppelin/bank-full.csv")

case class Bank(age:Integer, job:String, marital : String, education : String, balance : Integer)
val bank = bankText.map(s=>s.split(";")).map(
    s=>Bank(s(0).toInt,
            s(1).replaceAll("\"", ""),
            s(2).replaceAll("\"", ""),
            s(3).replaceAll("\"", ""),
            s(5).replaceAll("\"", "").toInt
        )
)
bank.registerAsTable("bank");
```

```
bankText: org.apache.spark.rdd.RDD[String] = MappedRDD[1] at textFile at <console>:17
defined class Bank
bank: org.apache.spark.rdd.RDD[Bank] = MappedRDD[3] at map at <console>:21
```

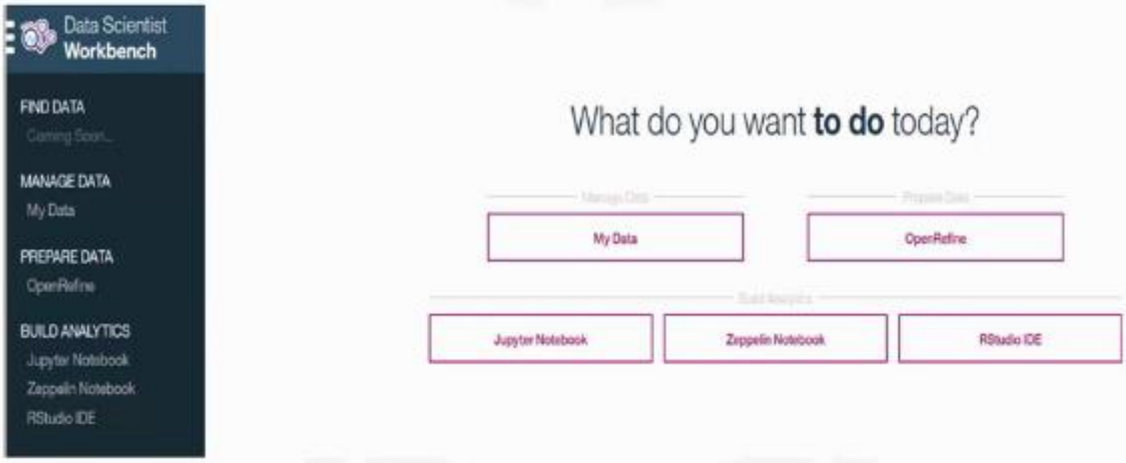
```
%md ### we can do spark sql
### directly display data as many different chart
### Parameterized query supported
```

we can do spark sql

directly display data as many different chart

Parameterized query supported

Data Scientist Workbench (DSWB)



Data Scientist Workbench

FIND DATA
Coming Soon...

MANAGE DATA
My Data

PREPARE DATA
OpenRefine

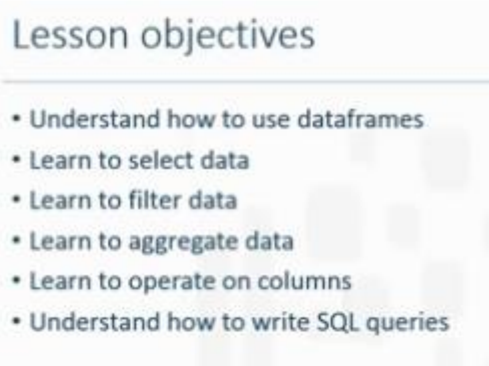
BUILD ANALYTICS
Jupyter Notebook
Zeppelin Notebook
RStudio IDE

What do you want **to do** today?

Manage Data: My Data

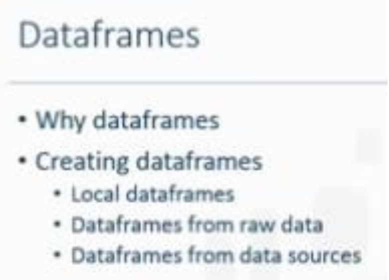
Prepare Data: OpenRefine

Build Analytics: Jupyter Notebook | Zeppelin Notebook | RStudio IDE



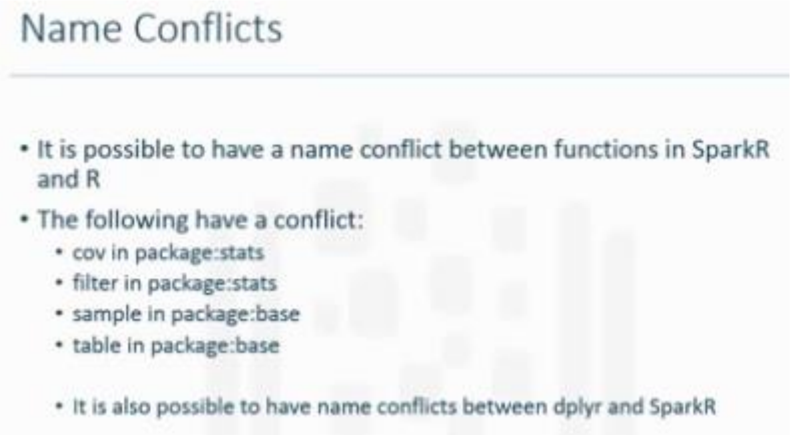
Lesson objectives

- Understand how to use dataframes
- Learn to select data
- Learn to filter data
- Learn to aggregate data
- Learn to operate on columns
- Understand how to write SQL queries



Dataframes

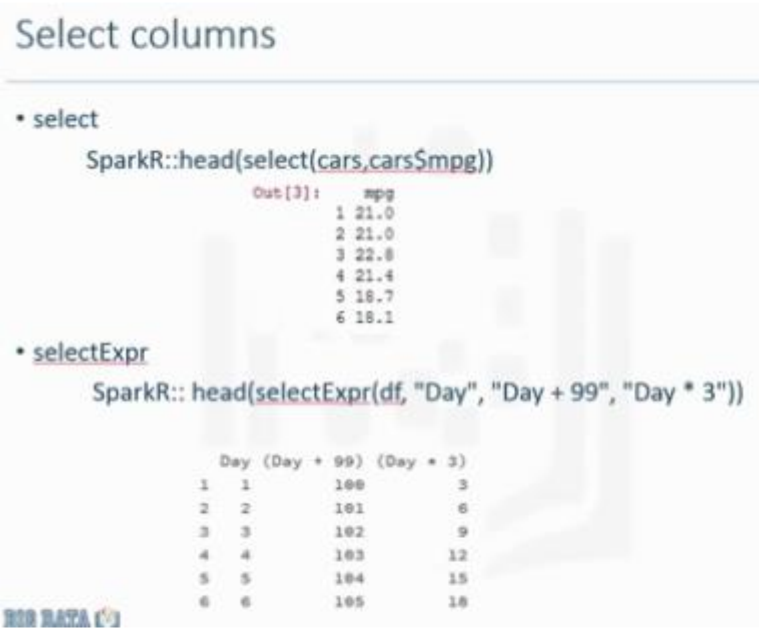
- Why dataframes
- Creating dataframes
  - Local dataframes
  - Dataframes from raw data
  - Dataframes from data sources



Name Conflicts

- It is possible to have a name conflict between functions in SparkR and R
- The following have a conflict:
  - cov in package:stats
  - filter in package:stats
  - sample in package:base
  - table in package:base

  - It is also possible to have name conflicts between dplyr and SparkR



Select columns

- select

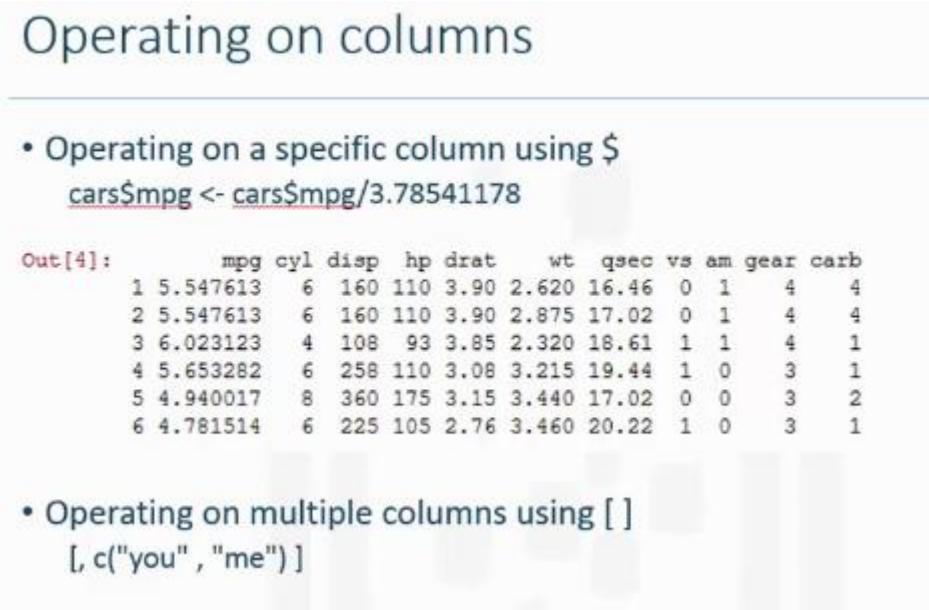
```
SparkR::head(select(cars,cars$mpg))
```

```
Out[3]:    mpg
         1 21.0
         2 21.0
         3 22.8
         4 21.4
         5 18.7
         6 18.1
```

- selectExpr

```
SparkR:: head(selectExpr(df, "Day", "Day + 99", "Day * 3"))
```

```
  Day (Day + 99) (Day * 3)
1   1        100         3
2   2        101         6
3   3        102         9
4   4        103        12
5   5        104        15
6   6        105        18
```



Operating on columns

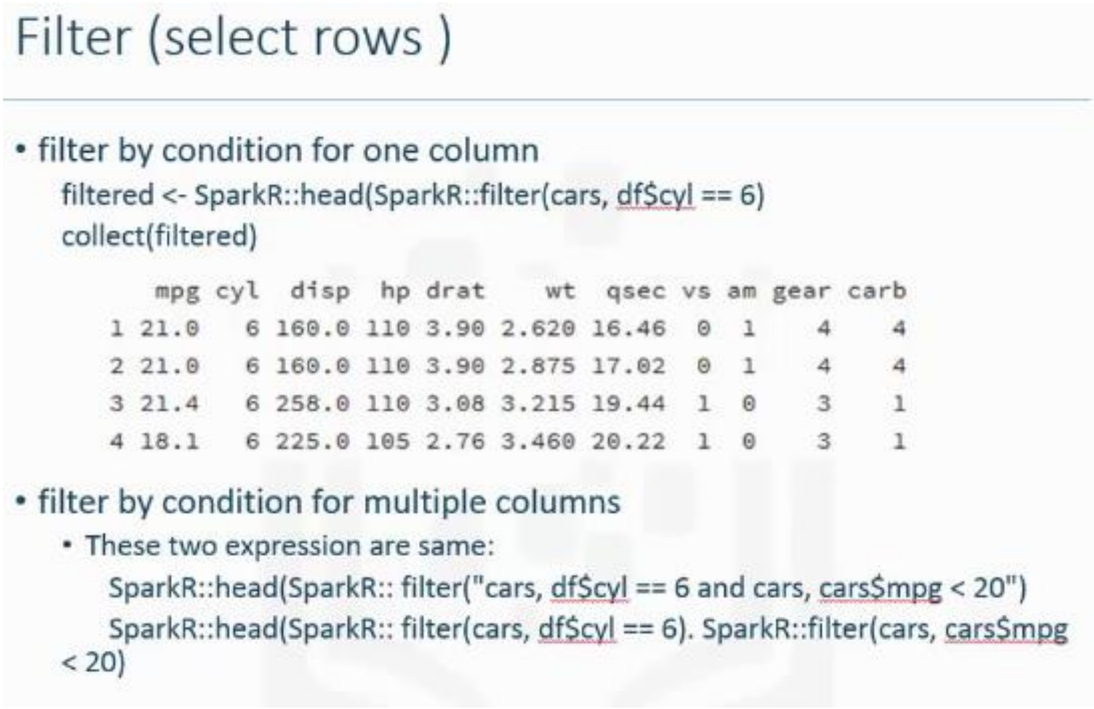
- Operating on a specific column using $

```
cars$mpg <- cars$mpg/3.78541178
```

```
Out[4]:       mpg cyl disp  hp drat    wt  qsec vs am gear carb
        1 5.547613   6  160 110 3.90 2.620 16.46  0  1    4    4
        2 5.547613   6  160 110 3.90 2.875 17.02  0  1    4    4
        3 6.023123   4  108  93 3.85 2.320 18.61  1  1    4    1
        4 5.653282   6  258 110 3.08 3.215 19.44  1  0    3    1
        5 4.940017   8  360 175 3.15 3.440 17.02  0  0    3    2
        6 4.781514   6  225 105 2.76 3.460 20.22  1  0    3    1
```

- Operating on multiple columns using [ ]

```
[, c("you" , "me") ]
```



Filter (select rows)

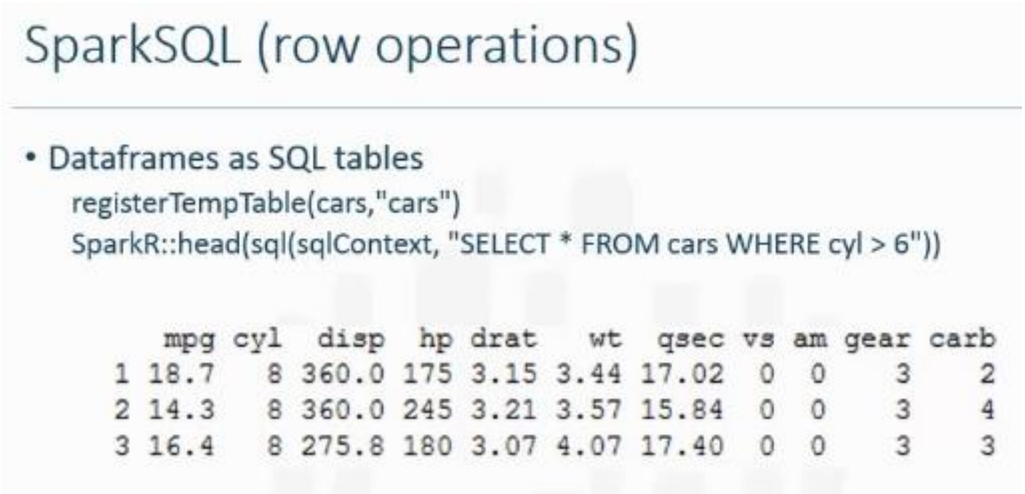
- filter by condition for one column

  filtered <- SparkR::head(SparkR::filter(cars, df$cyl == 6)
  collect(filtered)

```
   mpg cyl  disp  hp drat    wt  qsec vs am gear carb
1 21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
2 21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
3 21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
4 18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
```

- filter by condition for multiple columns
  - These two expression are same:

    SparkR::head(SparkR:: filter("cars, df$cyl == 6 and cars, cars$mpg < 20")
    SparkR::head(SparkR:: filter(cars, df$cyl == 6). SparkR::filter(cars, cars$mpg < 20)

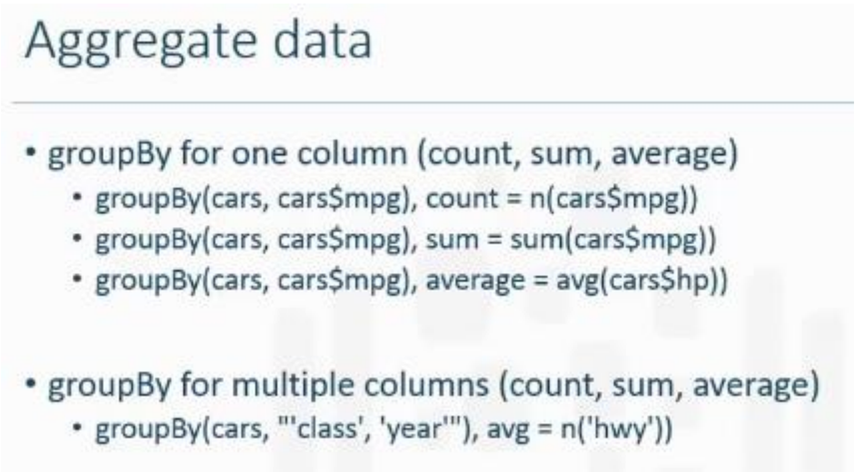


SparkSQL (row operations)

- Dataframes as SQL tables

  registerTempTable(cars,"cars")
  SparkR::head(sql(sqlContext, "SELECT * FROM cars WHERE cyl > 6"))

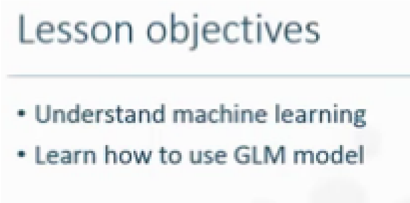
```
   mpg cyl  disp  hp drat   wt  qsec vs am gear carb
1 18.7   8 360.0 175 3.15 3.44 17.02  0  0    3    2
2 14.3   8 360.0 245 3.21 3.57 15.84  0  0    3    4
3 16.4   8 275.8 180 3.07 4.07 17.40  0  0    3    3
```



Aggregate data

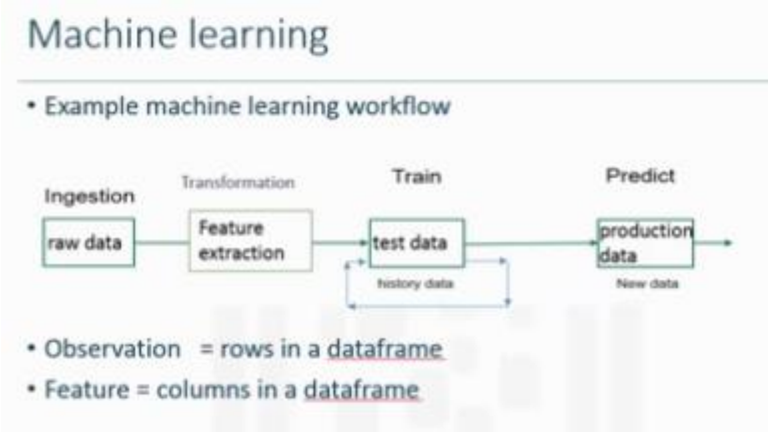
- groupBy for one column (count, sum, average)
  - groupBy(cars, cars$mpg), count = n(cars$mpg))
  - groupBy(cars, cars$mpg), sum = sum(cars$mpg))
  - groupBy(cars, cars$mpg), average = avg(cars$hp))

- groupBy for multiple columns (count, sum, average)
  - groupBy(cars, "'class', 'year'"), avg = n('hwy'))



Lesson objectives

- Understand machine learning
- Learn how to use GLM model



Machine learning

- Example machine learning workflow

Ingestion

raw data

Transformation

Feature extraction

Train

test data

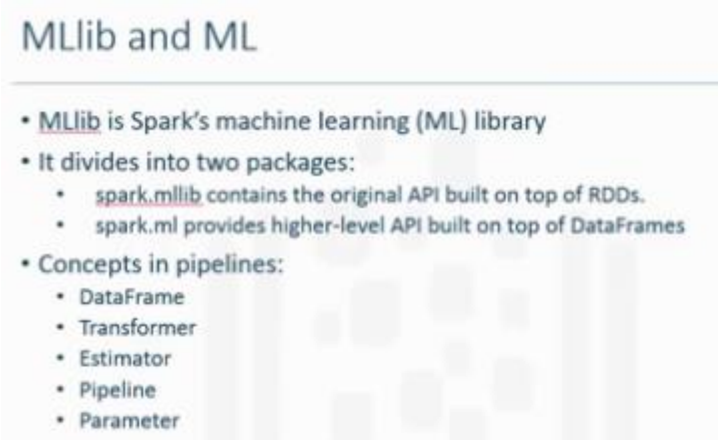
history data

Predict

production data

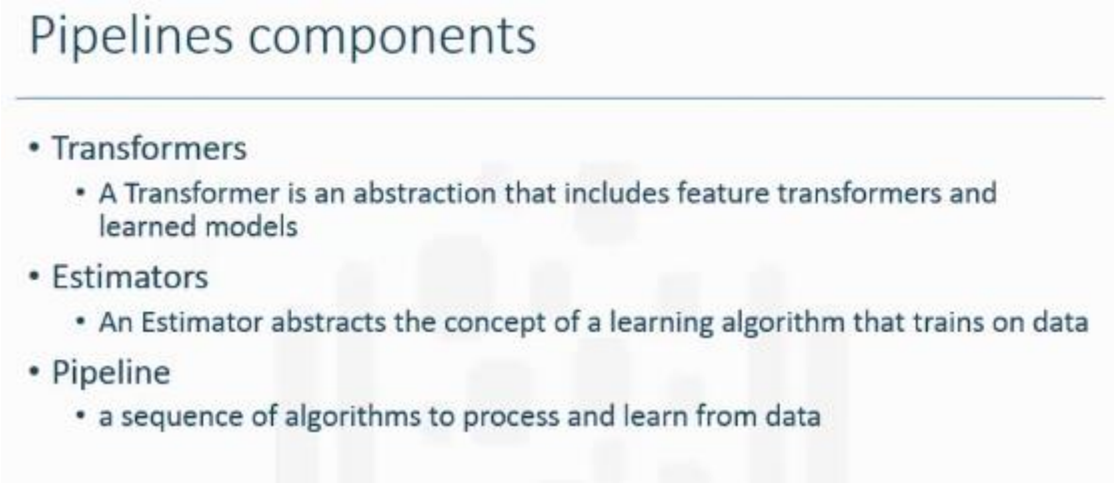
New data

- Observation = rows in a dataframe
- Feature = columns in a dataframe



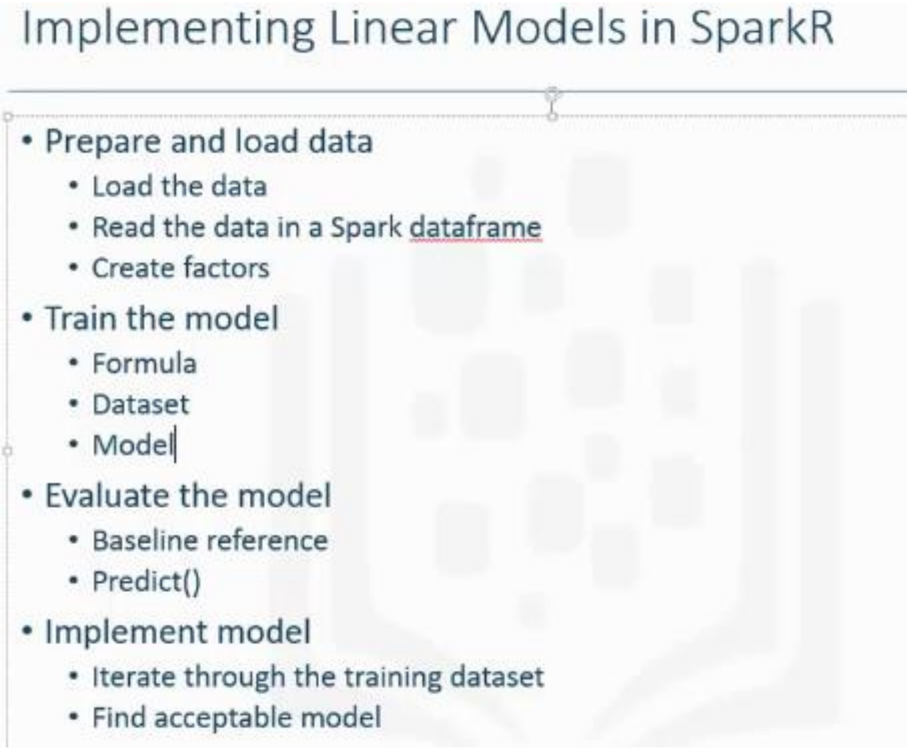
MLlib and ML

- MLlib is Spark's machine learning (ML) library
- It divides into two packages:
  - spark.mllib contains the original API built on top of RDDs.
  - spark.ml provides higher-level API built on top of DataFrames
- Concepts in pipelines:
  - DataFrame
  - Transformer
  - Estimator
  - Pipeline
  - Parameter



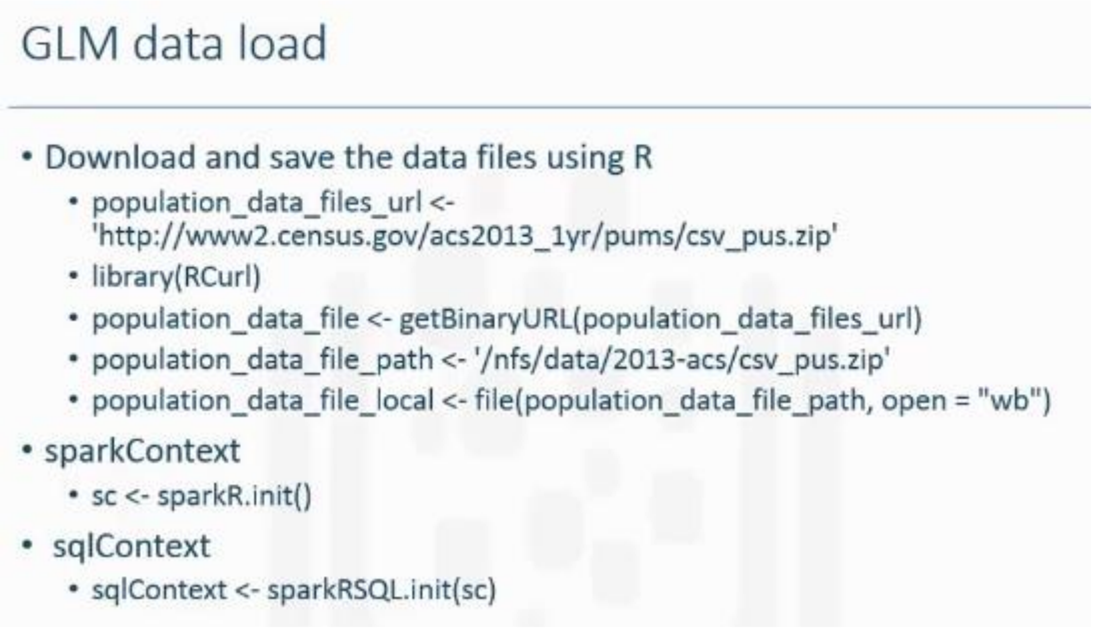
Pipelines components

- Transformers
  - A Transformer is an abstraction that includes feature transformers and learned models
- Estimators
  - An Estimator abstracts the concept of a learning algorithm that trains on data
- Pipeline
  - a sequence of algorithms to process and learn from data



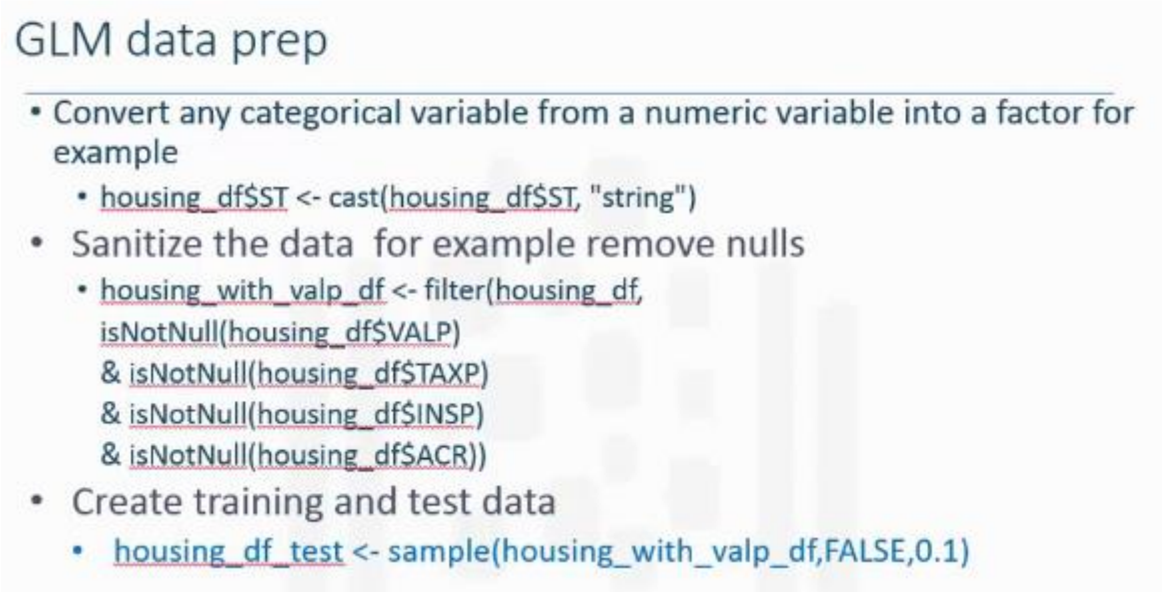
Implementing Linear Models in SparkR

- Prepare and load data
  - Load the data
  - Read the data in a Spark dataframe
  - Create factors
- Train the model
  - Formula
  - Dataset
  - Model
- Evaluate the model
  - Baseline reference
  - Predict()
- Implement model
  - Iterate through the training dataset
  - Find acceptable model



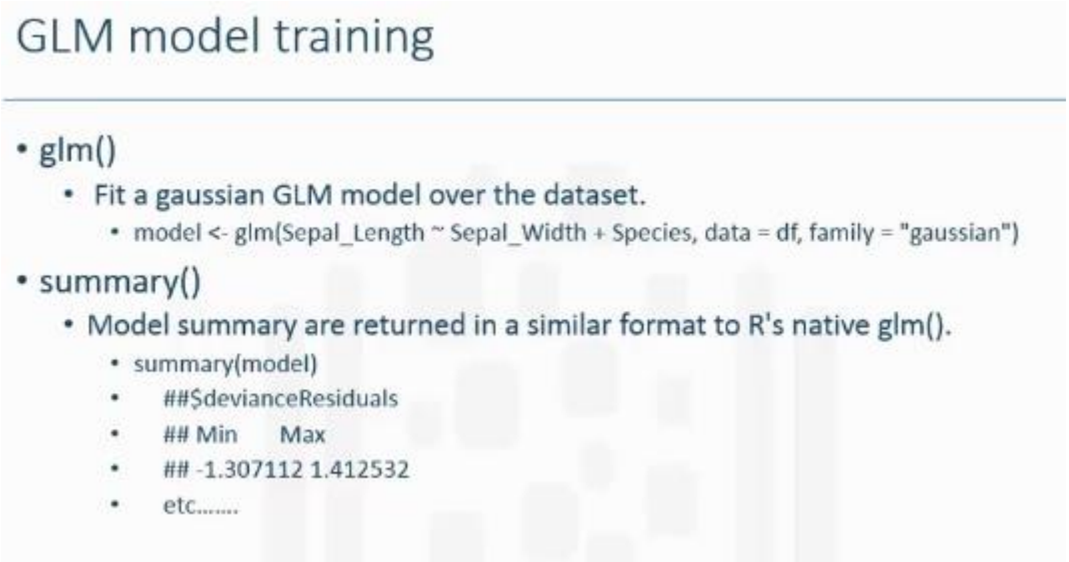
GLM data load

- Download and save the data files using R
  - population_data_files_url <- 'http://www2.census.gov/acs2013_1yr/pums/csv_pus.zip'
  - library(RCurl)
  - population_data_file <- getBinaryURL(population_data_files_url)
  - population_data_file_path <- '/nfs/data/2013-acs/csv_pus.zip'
  - population_data_file_local <- file(population_data_file_path, open = "wb")
- sparkContext
  - sc <- sparkR.init()
- sqlContext
  - sqlContext <- sparkRSQL.init(sc)



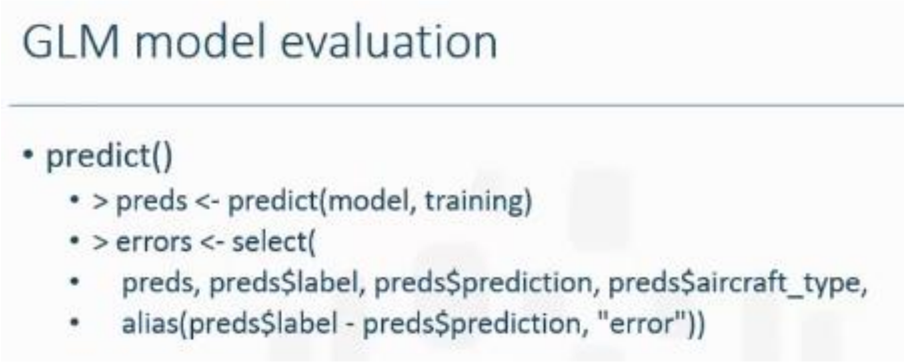
GLM data prep

- Convert any categorical variable from a numeric variable into a factor for example
  - housing_df$ST <- cast(housing_df$ST, "string")
- Sanitize the data for example remove nulls
  - housing_with_valp_df <- filter(housing_df,
isNotNull(housing_df$VALP)
& isNotNull(housing_df$TAXP)
& isNotNull(housing_df$INSP)
& isNotNull(housing_df$ACR))
- Create training and test data
  - housing_df_test <- sample(housing_with_valp_df,FALSE,0.1)



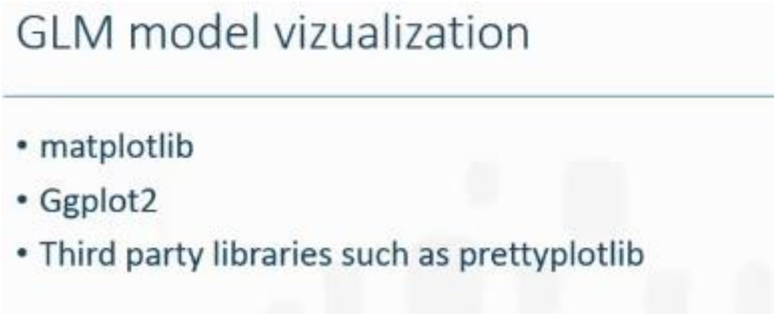
GLM model training

- glm()
  - Fit a gaussian GLM model over the dataset.
    - model <- glm(Sepal_Length ~ Sepal_Width + Species, data = df, family = "gaussian")
- summary()
  - Model summary are returned in a similar format to R's native glm().
    - summary(model)
    - ##$devianceResiduals
    - ## Min   Max
    - ## -1.307112 1.412532
    - etc.......



GLM model evaluation

- predict()
  - > preds <- predict(model, training)
  - > errors <- select(
  - preds, preds$label, preds$prediction, preds$aircraft_type,
  - alias(preds$label - preds$prediction, "error"))



GLM model vizualization

- matplotlib
- Ggplot2
- Third party libraries such as prettyplotlib