CS 696 Intro to Big Data: Tools and Methods
Fall Semester, 2019
Doc 15 Spark Intro 3
Mar 14, 2019

Course Datasets

GRE data

- Use Scikitlearn to compute linear regression
  - Given a GRE score what GPA does it predict?

Dwell Data

- muitiple-site.tsv
  - Using Scikitearn
    - mean, standard deviation of dwell time per site
  - Using Spark
    - mean, standard deviation of dwell time per site

Random Samples

```
tenDF = df.limit(10)
tenDF.show
```

```
+-----------------+-------------------+-----+
|DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME|count|
+-----------------+-------------------+-----+
|    United States|            Romania|   15|
|    United States|            Croatia|    1|
|    United States|            Ireland|  344|
|            Egypt|      United States|   15|
|    United States|              India|   62|
|    United States|          Singapore|    1|
|    United States|            Grenada|   62|
|       Costa Rica|      United States|  588|
|          Senegal|      United States|   40|
|          Moldova|      United States|    1|
+-----------------+-------------------+-----+
```

```
seed = 5
withReplacement = False
fraction = 0.5

smallSample = tenDF.sample(withReplacement, fraction, seed)
smallSample.show()
```

```
+-----------------+-------------------+-----+
|DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME|count|
+-----------------+-------------------+-----+
|    United States|            Romania|   15|
|            Egypt|      United States|   15|
|    United States|              India|   62|
|       Costa Rica|      United States|  588|
|          Senegal|      United States|   40|
+-----------------+-------------------+-----+
```

Random Splits

Split DF into two disjoint parts randomly

One dataframe for training
One for validation

randomSplit

```
seed = 5
twoDF = tenDF.randomSplit((0.25, 0.75), seed)
twoDF[0].show()
twoDF[1].show()
```

```
+-----------------+-------------------+-----+
|DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME|count|
+-----------------+-------------------+-----+
|    United States|            Romania|   15|
|    United States|            Croatia|    1|
|    United States|            Ireland|  344|
|            Egypt|      United States|   15|
|    United States|              India|   62|
|    United States|          Singapore|    1|
|    United States|            Grenada|   62|
|       Costa Rica|      United States|  588|
|          Senegal|      United States|   40|
|          Moldova|      United States|    1|
+-----------------+-------------------+-----+
```

```
+-----------------+-------------------+-----+
|DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME|count|
+-----------------+-------------------+-----+
|       Costa Rica|      United States|  588|
|    United States|            Ireland|  344|
|    United States|            Romania|   15|
+-----------------+-------------------+-----+
```

```
+-----------------+-------------------+-----+
|DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME|count|
+-----------------+-------------------+-----+
|            Egypt|      United States|   15|
|          Moldova|      United States|    1|
|          Senegal|      United States|   40|
|    United States|            Croatia|    1|
|    United States|            Grenada|   62|
|    United States|              India|   62|
|    United States|          Singapore|    1|
+-----------------+-------------------+-----+
```

Weights Normalized to 1

twoDF = tenDF.randomSplit([0.25, 0.50])
twoDF[0].show()
twoDF[1].show()

```
+-----------------+-------------------+-----+
|DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME|count|
+-----------------+-------------------+-----+
|       Costa Rica|      United States|  588|
|    United States|            Ireland|  344|
|    United States|            Romania|   15|
+-----------------+-------------------+-----+

+-----------------+-------------------+-----+
|DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME|count|
+-----------------+-------------------+-----+
|            Egypt|      United States|   15|
|          Moldova|      United States|    1|
|          Senegal|      United States|   40|
|    United States|             Croatia|    1|
|    United States|             Grenada|   62|
|    United States|               India|   62|
|    United States|           Singapore|    1|
+-----------------+-------------------+-----+
```

pyspark.sql.functions

283 functions

from pyspark.sql.functions import **sin**

df = spark.createDataFrame([(1, "John Doe", 21)], ("id", "name", "age"))
df.select("id","name",**sin**("age")).show()

```
+---+--------+--------------------+
| id|    name|           SIN(age)|
+---+--------+--------------------+
|  1|John Doe|0.8366556385360561|
+---+--------+--------------------+
```

User Defined Functions on DataFrames

```
def to_upper(s):
    if s is not None:
        return s.upper()

df = spark.createDataFrame([(1, "John Doe", 21)], ("id", "name", "age"))
df.select("id", to_upper("name"),"age").show()
```

```
+---+--------+---+
| id|    NAME|age|
+---+--------+---+
|  1|John Doe| 21|
+---+--------+---+
```

Function is applied to the column name not the data!

User Defined Functions on DataFrames

```
from pyspark.sql.types import IntegerType, StringType
from pyspark.sql.functions import udf

slen = udf(lambda s: len(s), IntegerType())

def add_one(x):
    if x is not None:
        return x + 1

add_one_udf = udf(add_one, IntegerType())

def to_upper(s):
        if s is not None:
            return s.upper()
to_upper_udf = udf(to_upper, StringType())
```

User Defined Functions on DataFrames

df = spark.createDataFrame([(1, "John Doe", 21)], ("id", "name", "age"))
df.select(**slen**("name").alias("slen(name)"), **to_upper_udf**("name"), **add_one_udf**("age")).show()

```
+----------+--------------+------------+
|slen(name)|to_upper(name)|add_one(age)|
+----------+--------------+------------+
|         8|      JOHN DOE|          22|
+----------+--------------+------------+
```

Type Mismatch

```
df = spark.createDataFrame([(1, "John Doe", 21)], ("id", "name", "age"))
df.select("name", add_one_udf("name"), "age").show()
```

Python
    Exception

Scala
    Null values

Aggregations

Summarize

groupBy

roll up

cube

window

Aggregation Functions

count
countDistinct
approx_count_distinct
first, last
min, max
sum
sumDistinct
avg, mean
variance, var_samp, var_pop
stddev, stddev_samp, stddev_pop
skewness, kurtosis

Covariance & Correlation
    corr, covar_samp, covar_pop

Example

```
flight_file = "/Users/whitney/Courses/696/Fall17/SparkBookData/flight-data/json/2015-summary.json"
flight_df = spark.read.json(flight_file)
flight_df.show(3)
```

```
+-----------------+-------------------+-----+
|DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME|count|
+-----------------+-------------------+-----+
|    United States|            Romania|   15|
|    United States|            Croatia|    1|
|    United States|            Ireland|  344|
+-----------------+-------------------+-----+
```

Example

```
import pyspark.sql.functions as F
newDf = flight_df.select(F.sum("count").alias('Sum'),
                         F.mean("count").alias('Mean'),
                         F.max("count").alias("Max"),
                         F.stddev_samp("count").alias("Sample StdDev"),
                         F.stddev_pop("count").alias("Pop StdDev"),
                         F.count("DEST_COUNTRY_NAME").alias("Count"))
newDf.show()
```

```
+------+-----------+------+------------------+------------------+-----+
|   Sum|       Mean|   Max|     Sample StdDev|       Pop StdDev|Count|
+------+-----------+------+------------------+------------------+-----+
|453316|1770.765625|370002|23126.516918551915|23081.30374350104|  256|
+------+-----------+------+------------------+------------------+-----+
```

countDistinct

```
import pyspark.sql.functions as F
newDf = flight_df.select(
    F.countDistinct(F.col("DEST_COUNTRY_NAME")).alias("Distinct Dest"),
    F.countDistinct(F.col("ORIGIN_COUNTRY_NAME")).alias("Distinct Origin"))

newDf.show()
```

```
+-------------+---------------+
|Distinct Dest|Distinct Origin|
+-------------+---------------+
|          132|            125|
+-------------+---------------+
```

countDistinct

```
import pyspark.sql.functions as F
newDf = flight_df.select(
    F.countDistinct(
        F.col("DEST_COUNTRY_NAME"),
        F.col("ORIGIN_COUNTRY_NAME")).alias("Distinct Pair"))

newDf.show()
```

```
+-------------+
|Distinct Pair|
+-------------+
|          256|
+-------------+
```

countDistinct

```
import pyspark.sql.functions as F
newDf = flight_df.select(
    F.countDistinct(
        "DEST_COUNTRY_NAME",
        "ORIGIN_COUNTRY_NAME",
        "count").alias("Distinct Rows"))
newDf.show()
```

```
+--------------+
|Distinct Rows|
+--------------+
|           256|
+--------------+
```

countDistinct

pyspark.sql.functions.countDistinct(col, *cols)

pyspark.sql.functions.count(col)

Group By Data

```
reader = spark.read
reader.option("header",True).option("inferSchema",True)
ordersDF = reader.csv("orders.csv")
ordersDF.show()
```

```
+--------+------+
|customer|amount|
+--------+------+
|       a|     2|
|       b|     8|
|       a|     3|
|       c|     9|
|       a|     4|
|       b|    16|
|       c|    11|
|       b|    24|
|       c|    30|
+--------+------+
```

groupBy

```
import pyspark.sql.functions as F

amountGrouped = ordersDF.groupBy("customer") \
    .agg(
        F.sum("amount").alias("Total"),
        F.mean("amount").alias("Average"),
        F.count("amount").alias("Number of Orders"))
amountGrouped.sort("customer").show()
```

```
+--------+------+
|customer|amount|
+--------+------+
|       a|     2|
|       b|     8|
|       a|     3|
|       c|     9|
|       a|     4|
|       b|    16|
|       c|    11|
|       b|    24|
|       c|    30|
+--------+------+
```

```
+--------+-----+------------------+----------------+
|customer|Total|           Average|Number of Orders|
+--------+-----+------------------+----------------+
|       a|    9|               3.0|               3|
|       b|   48|              16.0|               3|
|       c|   50|16.666666666666668|               3|
+--------+-----+------------------+----------------+
```

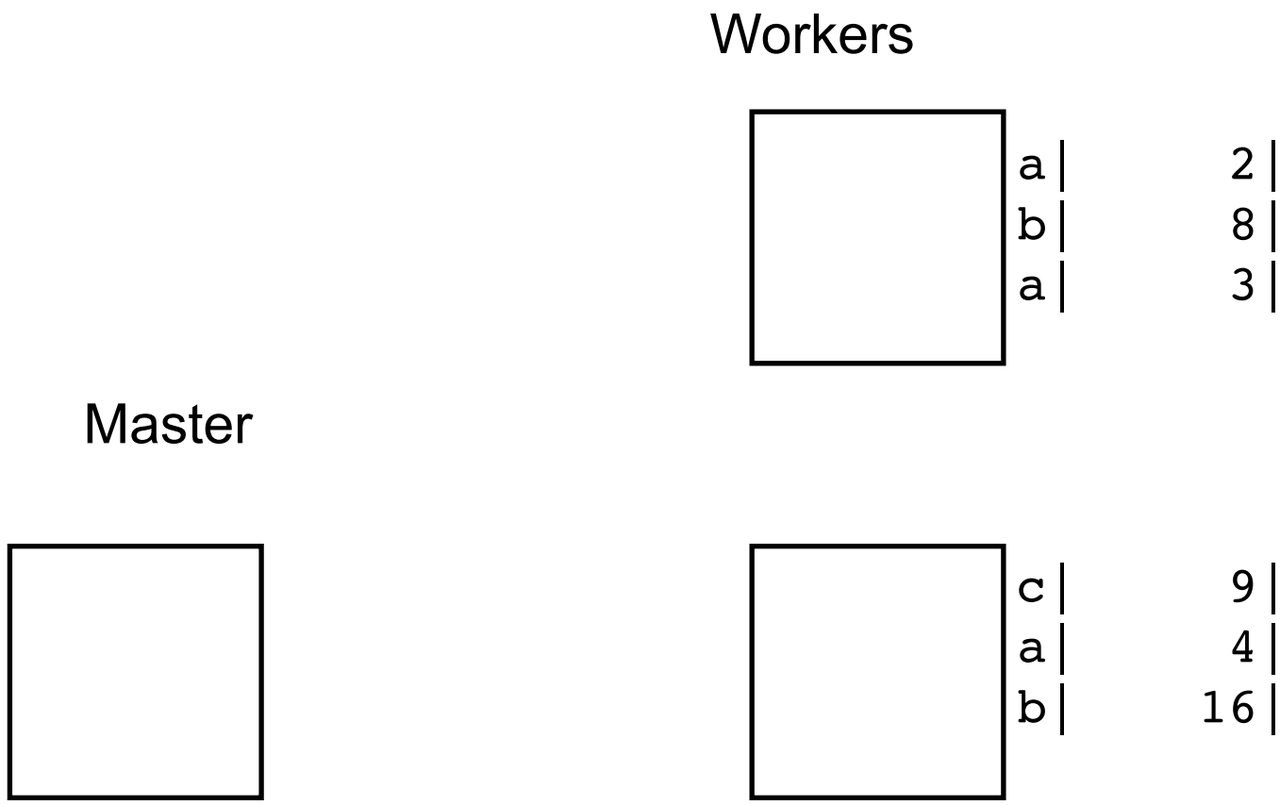
How does this work?

```
import pyspark.sql.functions as F

amountGrouped = ordersDF.groupBy("customer") \
                        .agg(F.sum("amount").alias("Total"))
amountGrouped.sort("customer").show()
```

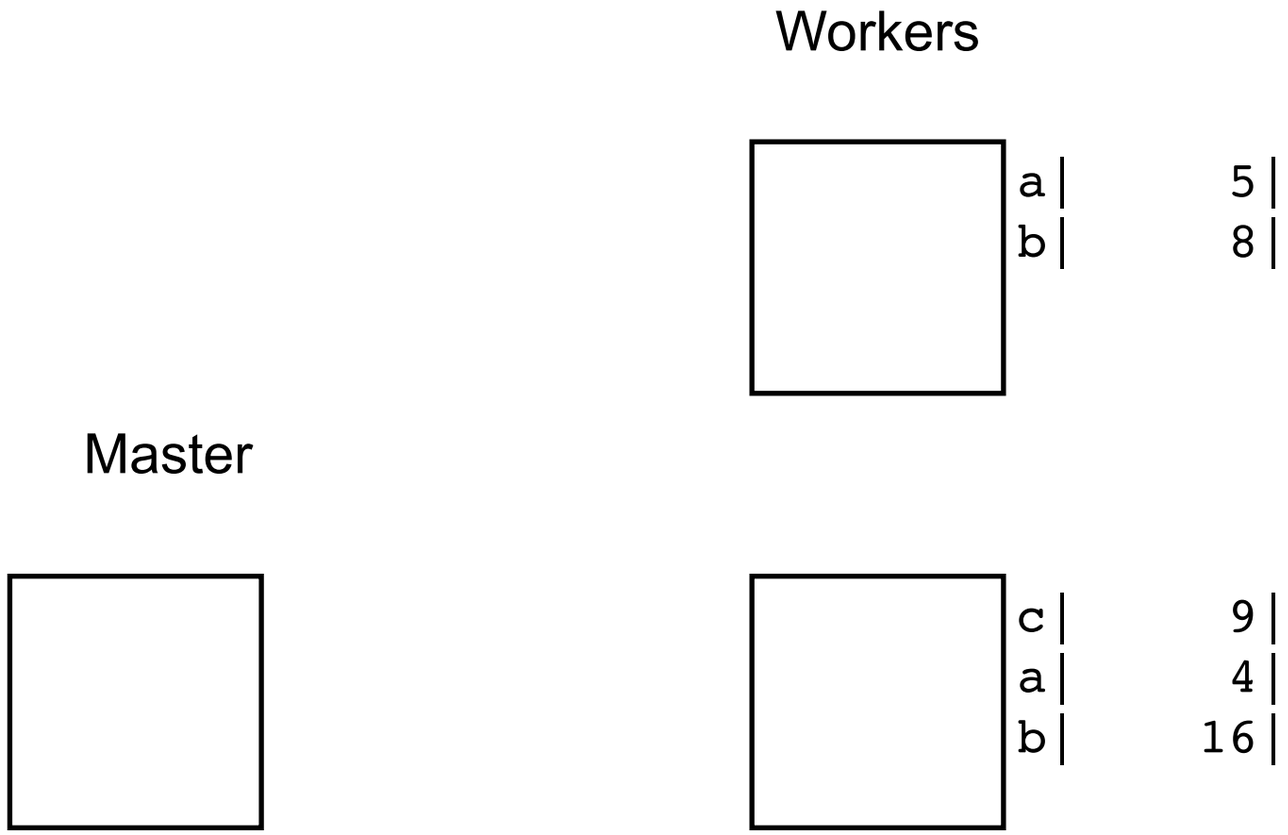
```
+--------+-----+
|customer|Total|
+--------+-----+
|       a|    9|
|       b|   48|
|       c|   50|
+--------+-----+
```

groupBy



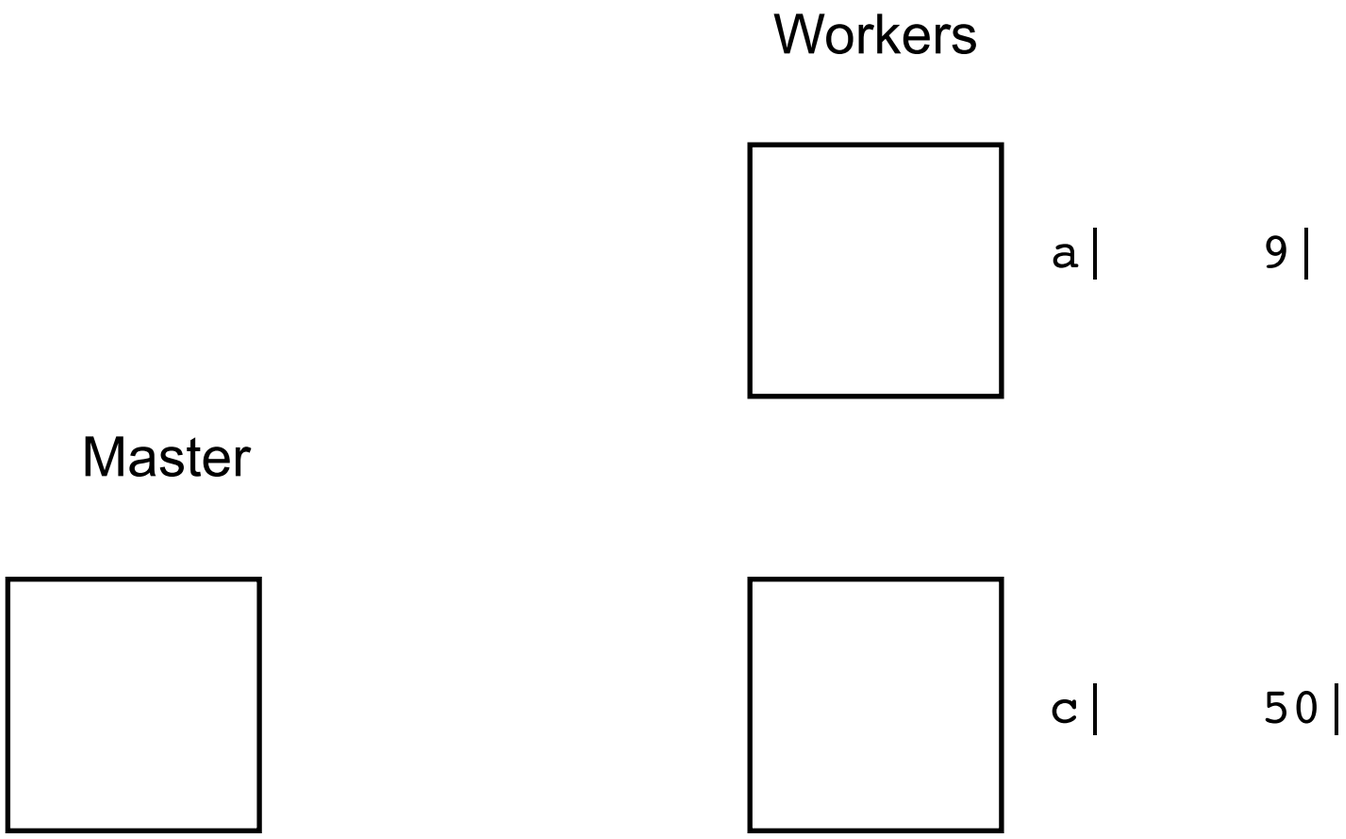

```
+--------+------+
|customer|amount|
+--------+------+
|       a|     2|
|       b|     8|
|       a|     3|
|       c|     9|
|       a|     4|
|       b|    16|
|       c|    11|
|       b|    24|
|       c|    30|
+--------+------+
```

groupBy



customer	amount
a	2
b	8
a	3
c	9
a	4
b	16
c	11
b	24
c	30

groupBy




```
+--------+------+
|customer|amount|
+--------+------+
|       a|     2|
|       b|     8|
|       a|     3|

|       c|     9|
|       a|     4|
|       b|    16|

|       c|    11|
|       b|    24|
|       c|    30|
+--------+------+
```

groupBy
Transformation