# pd.DataFrame vs rdd.DataFrame

**Wenqiang Feng**

E-mail: von198@gmail.com, Web: http://web.utk.edu/~wfeng1; https://runawayhorse001.github.io/LearningApacheSpark

## Loading DataFrame

### Creating DataFrame

▷ **From List**

```
pd.DataFrame(my_list,columns= col_name)
spark.createDataFrame(my_list, col_name).show()
```

▷ **From Dictionary**

```
pd.DataFrame(d)
spark.createDataFrame(np.array(list(d.values())).T.tolist(),
                              list(d.keys())).show()
```

### From Data Sources

▷ **From Database**

```
conn = psycopg2.connect(host=host, database=db_name,
                        user=user, password=pw)
cur = conn.cursor()
sql = """select * from table_name
      """.format(table_name=table_name)
dp = pd.read_sql(sql, conn)
-------------------------------------------------------------
url='jdbc:postgresql://'+host+':5432/'+db_name+'?user='+user
      +'&password='+pw
p={'driver':'org.postgresql.Driver','password':pw,'user':user}
ds=spark.read.jdbc(url=url, table=table_name, properties=p)
```

▷ **From .csv**

```
dp = pd.read_csv('Advertising.csv')
ds = spark.read.csv(path='Advertising.csv',
                           header=True,inferSchema=True)
```

▷ **From .json**

```
dp = pd.read_json("data/data.json")
ds = spark.read.json('data/data.json')
```

## Basic Manipulation

### Data Types

```
dp.dtypes
ds.dtypes
```

### Count

```
dp.count()[1]
ds.count()
```

### Column Names

```
dp.columns
ds.columns
```

### Select Columns

```
dp[name_list].head()
ds[name_list].show()
```

### Rename Columns

```
dp.columns = name_list
ds.toDF(*name_list).show()
```

### Drop Columns

```
dp.drop(name_list,axis=1)
ds.drop(*name_list).show()
```

### Distinct Rows

```
dp.drop_duplicates()
ds.drop_duplicates()
```

### Cross Table

```
pd.crosstab(dp.col1,dp.col2)
ds.crosstab('col1','col2')
```

### Replace Values

```
dp.A.replace(['male', 'female'],[1, 0], inplace=True)
ds.na.replace(['male','female'],['1','0']).show()
```

## Basic Manipulation

### Rename one or more columns

```
mapping = 'kay1':'val1','key2':'val2'
dp.rename(columns=mapping).head(4)
new_names = [mapping.get(col,col) for col in ds.columns]
ds.toDF(*new_names).show(4)
```

### Replace one or more data types

```
d = 'col2': 'str','col3':'str' # 'string' for pyspark
dp = dp.astype(d)
ds =  ds.select(*list(set(ds.columns)-set(d.keys())),
                *(col(c[0]).astype(c[1]).alias(c[0]) for c in d.items()))
```

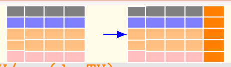
### Random Split

```
from sklearn.model_selection import train_test_split
a, b = train_test_split(dp, test_size=0.8)
a, b = ds.randomSplit([0.2,0.8])
```

### Unixtime to Date

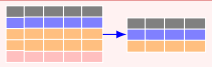
```
dp['date']=pd.to_datetime(dp['ts'],unit='s').dt.tz_localize('UTC')
spark.conf.set("spark.sql.session.timeZone", "UTC")
ds.withColumn('date', F.from_unixtime('ts'))
```

### Make New Vaiables

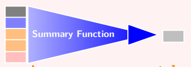


```
dp['tv_norm'] = dp.TV/sum(dp.TV)
ds.withColumn('tv_norm', ds.TV/ds.groupBy()
                     .agg(F.sum("TV")).collect()[0][0]).show(4)
dp['cond'] = dp.apply(lambda c: 1 if ((c.TV>100)&(c.Radio<40))
                                 else 2 if c.Sales> 10
                                 else 3,axis=1)
ds.withColumn('cond',F.when((ds.TV>100)&(ds.Radio<40),1)
                      .when(ds.Sales>10, 2)
                      .otherwise(3)).show(4)
dp['log_tv'] = np.log(dp.TV)
ds.withColumn('log_tv',F.log(ds.TV)).show(4)
dp['tv+10'] = dp.TV.apply(lambda x: x+10)
ds.withColumn('tv+10', ds.TV+10).show(4)
```

## Summarise Data

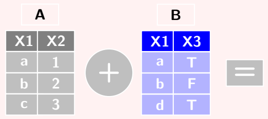


```
dp.describe()
ds.describe().show()
dp.corr(method='pearson')
mat=Statistics.corr(ds.rdd.map(lambda r: r[0:]),method='pearson')
pd.DataFrame(mat,columns=ds.columns,index=ds.columns)
```



```
dp.C.max() #Similar for: min,max,mean,std
ds.agg(F.max(df.C)).head()[0] #Similar for: min,max,avg,stddev
```

## Mutating Joins



| A | |
|---|---|
| X1 | X2 |
| a | 1 |
| b | 2 |
| c | 3 |

| B | |
|---|---|
| X1 | X3 |
| a | T |
| b | F |
| d | T |

**Result** / **Function**

| X1 | X2 | X3 |
|---|---|---|
| a | 1 | T |
| b | 2 | F |
| c | 3 | null |

```
#Join matching rows from B to A
A.merge(B,on='X1',how='left')
A.join(B,'X1',how='left')
 .orderBy('X1', ascending=True).show()
```

| X1 | X2 | X3 |
|---|---|---|
| a | 1 | T |
| b | 2 | F |
| d | null | T |

```
#Join matching rows from A to B
A.merge(B,on='X1',how='right')
A.join(B,'X1',how='right')
 .orderBy('X1', ascending=True).show()
```

| X1 | X2 | X3 |
|---|---|---|
| a | 1 | T |
| b | 2 | F |

```
#Retain only rows in both sets
A.merge(B,on='X1',how='inner')
A.join(B,'X1',how='inner')
 .orderBy('X1', ascending=True).show()
```

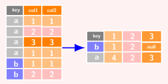
| X1 | X2 | X3 |
|---|---|---|
| a | 1 | T |
| b | 2 | F |
| c | 3 | null |
| d | null | T |

```
#Retain all values,all rows
A.merge(B,on='X1',how='full')
A.join(B,'X1',how='full')
 .orderBy('X1', ascending=True).show()
```

## Group Data

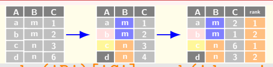
```
dp.groupby(['A']).agg('B':'min','C':'mean')
ds.groupBy(['A']).agg('B':'min','C':'avg').show()
```

## Pivot



```
pd.pivot_table(dp, values='col1', index='key',
                     columns='col2', aggfunc=np.sum)
---------------------------------------------------
df.groupBy(['key'])
  .pivot('col1').sum('col2').show()
```

## Windows



```
dp['rank'] = dp.groupby('B')['C'].rank('dense', ascending=False)
-------------------------------------------------
w = Window.partitionBy('B').orderBy(ds.C.desc())
ds  = ds.withColumn('rank', F.dense_rank().over(w))
```

## SQL

```
sql = """
      SELECT * FROM table_name
      """.format(table_name=table_name)
dp = pd.read_sql(sql, conn)
#
ds.registerTempTable('ds')
spark.sql("SELECT * FROM ds").show()
```