pd.DataFrame vs rdd.DataFrame

Wengiang Feng

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

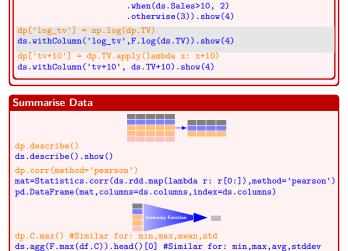

```
From Data Sources
> From Database
  conn = psycopg2.connect(host=host, database=db_name,
                         user=user, password=pw)
  cur = conn.cursor()
  sql = """select * from table name
       """.format(table_name=table_name)
  dp = pd.read_sql(sql, conn)
  url='jdbc:postgresql://'+host+':5432/'+db_name+'?user='+user
         +'&password='+pw
  p='driver':'org.postgresql.Driver','password':pw,'user':user
  ds=spark.read.idbc(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 Count dp.dtypes dp.count()[1] ds.dtypes ds.count() Column Names Select Columns dp[name list].head() dp.columns ds.columns ds[name list].show() Rename Columns **Drop Columns** dp.columns = name_list dp.drop(name_list,axis=1) ds.toDF(*name_list).show() ds.drop(*name_list).show() Distinct Rows Cross Table dp.drop_duplicates() pd.crosstab(dp.col1,dp.col2) ds.drop_duplicates() ds.crosstab('col1','col2') Replace Values dp.A.replace(['male', 'female'],[1, 0], inplace=True)

ds.na.replace(['male','female'],['1','0']).show()

©All Rights Reserved by Dr.Wenqiang Feng. Powered by IATEX. Updated:03-02-2019. von198@gmail.com

```
Basic Manipulation
Rename one or more columns
mapping = 'kav1':'val1'.'kev2':'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.astvpe(d)
ds = ds.select(*list(set(ds.columns)-set(d.kevs())).
                *(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))</pre>
                                    else 2 if c.Sales> 10
                                    else 3,axis=1)
 ds.withColumn('cond',F.when((ds.TV>100)&(ds.Radio<40),1)
```



```
Mutating Joins
                     X1 X2
                                  X1 X3
              #Join matching rows from B to A
X1 X2 X3
              A.merge(B,on='X1',how='left')
              A.join(B.'X1',how='left')
               .orderBy('X1', ascending=True).show()
              #Join matching rows from A to B
X1 X2 X3
              A.merge(B,on='X1',how='right')
              A.join(B,'X1',how='right')
               .orderBy('X1', ascending=True).show()
c null T
              #Retain only rows in both sets
X1 X2 X3
              A.merge(B,on='X1',how='inner')
              A.join(B,'X1',how='inner')
               .orderBy('X1', ascending=True).show()
X1 X2 X3
              #Retain all values, all rows
              A.merge(B,on='X1',how='full')
              A. join(B, 'X1', how='full')
               .orderBy('X1', ascending=True).show()
```



```
pd.pivot_table(dp, values='col1', index='key', columns='col2', aggfunc=np.sum)

df.groupBy(['key'])
.pivot('col1').sum('col2').show()
```