In [28]:
```
import os
import sys
spark_home = os.environ['SPARK_HOME'] = '/Users/liang/Downloads/spark-1.3
spark_home = os.environ['SPARK_HOME'] = '/Users/jshanahan/Dropbox/Lecture
if not spark_home:
    raise ValueError('SPARK_HOME enviroment variable is not set')
sys.path.insert(0,os.path.join(spark_home,'python'))
sys.path.insert(0,os.path.join(spark_home,'python/lib/py4j-0.8.2.1-src.zi
execfile(os.path.join(spark_home,'python/pyspark/shell.py'))
```

```
---------------------------------------------------------------------------
-----
ValueError                                Traceback (most recent call
last)
<ipython-input-28-9d3b5026fb8b> in <module>()
      7 sys.path.insert(0,os.path.join(spark_home,'python'))
      8 sys.path.insert(0,os.path.join(spark_home,'python/lib/py4j-0.8
.2.1-src.zip'))
----> 9 execfile(os.path.join(spark_home,'python/pyspark/shell.py'))

/Users/jshanahan/Dropbox/Lectures-UC-Berkeley-ML-Class-2015/spark-1.5.
0-bin-hadoop2.6/python/pyspark/shell.py in <module>()
     41     SparkContext.setSystemProperty("spark.executor.uri", os.en
viron["SPARK_EXECUTOR_URI"])
     42
---> 43 sc = SparkContext(pyFiles=add_files)
     44 atexit.register(lambda: sc.stop())
     45

/Users/jshanahan/Dropbox/Lectures-UC-Berkeley-ML-Class-2015/spark-1.5.
0-bin-hadoop2.6/python/pyspark/context.pyc in __init__(self, master, a
ppName, sparkHome, pyFiles, environment, batchSize, serializer, conf,
gateway, jsc, profiler_cls)
    108         """
    109         self._callsite = first_spark_call() or CallSite(None,
None, None)
--> 110         SparkContext._ensure_initialized(self, gateway=gateway
)
    111         try:
    112             self._do_init(master, appName, sparkHome, pyFiles,
environment, batchSize, serializer,

/Users/jshanahan/Dropbox/Lectures-UC-Berkeley-ML-Class-2015/spark-1.5.
0-bin-hadoop2.6/python/pyspark/context.pyc in _ensure_initialized(cls,
instance, gateway)
    248                         " created by %s at %s:%s "
    249                         % (currentAppName, currentMaster,
--> 250                            callsite.function, callsite.file,
callsite.linenum))
    251                 else:
    252                     SparkContext._active_spark_context =
```

```
instance

ValueError: Cannot run multiple SparkContexts at once; existing SparkC
ontext(app=pyspark-shell, master=local[*]) created by <module> at <ipy
thon-input-2-9d3b5026fb8b>:9
```

In [1]:
```
## Set up Spark

import os
import sys
import pyspark
from pyspark.sql import SQLContext

# We can give a name to our app (to find it in Spark WebUI) and configure
# In this case, it is local multicore execution with "local[*]"
app_name = "example-logs"
master = "local[*]"
conf = pyspark.SparkConf().setAppName(app_name).setMaster(master)
sc = pyspark.SparkContext(conf=conf)
sqlContext = SQLContext(sc)
print sc
print sqlContext
```

```
<pyspark.context.SparkContext object at 0x7f58867df510>
<pyspark.sql.context.SQLContext object at 0x7f5885f7e3d0>
```

In [13]:
```
%%writefile callsign_tbl_sorted.txt
3AZ, Monaco (Principality of)
3BZ, Mauritius (Republic of)
3CZ, Equatorial Guinea (Republic of)
3DM, Swaziland (Kingdom of)
3DZ, Fiji (Republic of)
3FZ, Panama (Republic of)
3GZ, Chile
3UZ, China (People's Republic of)
3VZ, Tunisia
3WZ, Viet Nam (Socialist Republic of)
3XZ, Guinea (Republic of)
3YZ, Norway
3ZZ, Poland (Republic of)
4CZ, Mexico
4IZ, Philippines (Republic of the)
4KZ, Azerbaijani Republic
4LZ, Georgia (Republic of)
4MZ, Venezuela (Republic of)
4OZ, Montenegro (Republic of)                          (WRC-07)
4SZ, Sri Lanka (Democratic Socialist Republic of)
4TZ, Peru
4UZ, United Nations
4VZ, Haiti (Republic of)
4WZ, Democratic Republic of Timor-Leste   (WRC-03)
4XZ, Israel (State of)
4YZ, International Civil Aviation Organization
```

4YZ, International Civil Aviation Organization
4ZZ, Israel (State of)
5AZ, Libya (Socialist People's Libyan Arab Jamahiriya)
5BZ, Cyprus (Republic of)
5GZ, Morocco (Kingdom of)
5IZ, Tanzania (United Republic of)
5KZ, Colombia (Republic of)
5MZ, Liberia (Republic of)
5OZ, Nigeria (Federal Republic of)
5QZ, Denmark
5SZ, Madagascar (Republic of)
5TZ, Mauritania (Islamic Republic of)
5UZ, Niger (Republic of the)
5VZ, Togolese Republic
5WZ, Samoa (Independent State of)
5XZ, Uganda (Republic of)
5ZZ, Kenya (Republic of)
6BZ, Egypt (Arab Republic of)
6CZ, Syrian Arab Republic
6JZ, Mexico
6NZ, Korea (Republic of)
6OZ, Somali Democratic Republic
6SZ, Pakistan (Islamic Republic of)
6UZ, Sudan (Republic of the)
6WZ, Senegal (Republic of)
6XZ, Madagascar (Republic of)
6YZ, Jamaica
6ZZ, Liberia (Republic of)
7IZ, Indonesia (Republic of)
7NZ, Japan
7OZ, Yemen (Republic of)
7PZ, Lesotho (Kingdom of)
7QZ, Malawi
7RZ, Algeria (People's Democratic Republic of)
7SZ, Sweden
7YZ, Algeria (People's Democratic Republic of)
7ZZ, Saudi Arabia (Kingdom of)
8IZ, Indonesia (Republic of)
8NZ, Japan
8OZ, Botswana (Republic of)
8PZ, Barbados
8QZ, Maldives (Republic of)
8RZ, Guyana
8SZ, Sweden
8YZ, India (Republic of)
8ZZ, Saudi Arabia (Kingdom of)
9AZ, Croatia (Republic of)
9DZ, Iran (Islamic Republic of)
9FZ, Ethiopia (Federal Democratic Republic of)
9GZ, Ghana
9HZ, Malta
9JZ, Zambia (Republic of)
9KZ, Kuwait (State of)
9LZ, Sierra Leone

9MZ, Malaysia
9NZ, Nepal
9TZ, Democratic Republic of the Congo
9UZ, Burundi (Republic of)
9VZ, Singapore (Republic of)
9WZ, Malaysia
9XZ, Rwandese Republic
9ZZ, Trinidad **and** Tobago
A2Z, Botswana (Republic of)
A3Z, Tonga (Kingdom of)
A4Z, Oman (Sultanate of)
A5Z, Bhutan (Kingdom of)
A6Z, United Arab Emirates
A7Z, Qatar (State of)
A8Z, Liberia (Republic of)
A9Z, Bahrain (State of)
ALZ, United States of America
AOZ, Spain
ASZ, Pakistan (Islamic Republic of)
AWZ, India (Republic of)
AXZ, Australia
AZZ, Argentine Republic
BZZ, China (People's Republic of)
C2Z, Nauru (Republic of)
C3Z, Andorra (Principality of)
C4Z, Cyprus (Republic of)
C5Z, Gambia (Republic of the)
C6Z, Bahamas (Commonwealth of the)
C7Z, World Meteorological Organization
C9Z, Mozambique (Republic of)
CEZ, Chile
CKZ, Canada
CMZ, Cuba
CNZ, Morocco (Kingdom of)
COZ, Cuba
CPZ, Bolivia (Republic of)
CUZ, Portugal
CXZ, Uruguay (Eastern Republic of)
CZZ, Canada
D3Z, Angola (Republic of)
D4Z, Cape Verde (Republic of)
D5Z, Liberia (Republic of)
D6Z, Comoros (Islamic Federal Republic of the)
D9Z, Korea (Republic of)
DRZ, Germany (Federal Republic of)
DTZ, Korea (Republic of)
DZZ, Philippines (Republic of the)
E2Z, Thailand
E3Z, Eritrea
E4Z, Palestinian Authority
E5Z, New Zealand - Cook Islands (WRC-07)
E6Z, New Zealand - Niue
E7Z, Bosnia **and** Herzegovina (Republic of) (WRC-07)
EHZ, Spain

EJZ, Ireland
EKZ, Armenia (Republic of)
ELZ, Liberia (Republic of)
EOZ, Ukraine
EQZ, Iran (Islamic Republic of)
ERZ, Moldova (Republic of)
ESZ, Estonia (Republic of)
ETZ, Ethiopia (Federal Democratic Republic of)
EWZ, Belarus (Republic of)
EXZ, Kyrgyz Republic
EYZ, Tajikistan (Republic of)
EZZ, Turkmenistan
FZZ, France
GZZ, United Kingdom of Great Britain **and** Northern Ireland
H2Z, Cyprus (Republic of)
H3Z, Panama (Republic of)
H4Z, Solomon Islands
H7Z, Nicaragua
H9Z, Panama (Republic of)
HAZ, Hungary (Republic of)
HBZ, Switzerland (Confederation of)
HDZ, Ecuador
HEZ, Switzerland (Confederation of)
HFZ, Poland (Republic of)
HGZ, Hungary (Republic of)
HHZ, Haiti (Republic of)
HIZ, Dominican Republic
HKZ, Colombia (Republic of)
HLZ, Korea (Republic of)
HMZ, Democratic People's Republic of Korea
HNZ, Iraq (Republic of)
HPZ, Panama (Republic of)
HRZ, Honduras (Republic of)
HSZ, Thailand
HTZ, Nicaragua
HUZ, El Salvador (Republic of)
HVZ, Vatican City State
HYZ, France
HZZ, Saudi Arabia (Kingdom of)
IZZ, Italy
J2Z, Djibouti (Republic of)
J3Z, Grenada
J4Z, Greece
J5Z, Guinea-Bissau (Republic of)
J6Z, Saint Lucia
J7Z, Dominica (Commonwealth of)
J8Z, Saint Vincent **and** the Grenadines
JSZ, Japan
JVZ, Mongolia
JXZ, Norway
JYZ, Jordan (Hashemite Kingdom of)
JZZ, Indonesia (Republic of)
KZZ, United States of America
L9Z, Argentine Republic

LNZ, Norway
LWZ, Argentine Republic
LXZ, Luxembourg
LYZ, Lithuania (Republic of)
LZZ, Bulgaria (Republic of)
MZZ, United Kingdom of Great Britain **and** Northern Ireland
NZZ, United States of America
OCZ, Peru
ODZ, Lebanon
OEZ, Austria
OJZ, Finland
OLZ, Czech Republic
OMZ, Slovak Republic
OTZ, Belgium
OZZ, Denmark
P2Z, Papua New Guinea
P3Z, Cyprus (Republic of)
P4Z, Netherlands (Kingdom of the) **-** Aruba
P9Z, Democratic People's Republic of Korea
PIZ, Netherlands (Kingdom of the)
PJZ, Netherlands (Kingdom of the) **-** Netherlands Caribbean
POZ, Indonesia (Republic of)
PYZ, Brazil (Federative Republic of)
PZZ, Suriname (Republic of)
RZZ, Russian Federation
S3Z, Bangladesh (People's Republic of)
S5Z, Slovenia (Republic of)
S6Z, Singapore (Republic of)
S7Z, Seychelles (Republic of)
S8Z, South Africa (Republic of)
S9Z, Sao Tome **and** Principe (Democratic Republic of)
SMZ, Sweden
SRZ, Poland (Republic of)
SSM, Egypt (Arab Republic of)
STZ, Sudan (Republic of the)
SUZ, Egypt (Arab Republic of)
SZZ, Greece
T2Z, Tuvalu
T3Z, Kiribati (Republic of)
T4Z, Cuba
T5Z, Somali Democratic Republic
T6Z, Afghanistan (Islamic State of)
T7Z, San Marino (Republic of)
T8Z, Palau (Republic of)
TCZ, Turkey
TDZ, Guatemala (Republic of)
TEZ, Costa Rica
TFZ, Iceland
TGZ, Guatemala (Republic of)
THZ, France
TIZ, Costa Rica
TJZ, Cameroon (Republic of)
TKZ, France
TLZ, Central African Republic

TMZ, France
TNZ, Congo (Republic of the)
TQZ, France
TRZ, Gabonese Republic
TSZ, Tunisia
TTZ, Chad (Republic of)
TUZ, Côte d'Ivoire (Republic of)
TXZ, France
TYZ, Benin (Republic of)
TZZ, Mali (Republic of)
UIZ, Russian Federation
UMZ, Uzbekistan (Republic of)
UQZ, Kazakhstan (Republic of)
UZZ, Ukraine
V2Z, Antigua **and** Barbuda
V3Z, Belize
V4Z, Saint Kitts **and** Nevis
V5Z, Namibia (Republic of)
V6Z, Micronesia (Federated States of)
V7Z, Marshall Islands (Republic of the)
V8Z, Brunei Darussalam
VGZ, Canada
VNZ, Australia
VOZ, Canada
VQZ, United Kingdom of Great Britain **and** Northern Ireland
VRZ, China (People's Republic of) - Hong Kong
VSZ, United Kingdom of Great Britain **and** Northern Ireland
VWZ, India (Republic of)
VYZ, Canada
VZZ, Australia
WZZ, United States of America
XIZ, Mexico
XOZ, Canada
XPZ, Denmark
XRZ, Chile
XSZ, China (People's Republic of)
XTZ, Burkina Faso
XUZ, Cambodia (Kingdom of)
XVZ, Viet Nam (Socialist Republic of)
XWZ, Lao People's Democratic Republic
XXZ, China (People's Republic of) - Macao (WRC-07)
XZZ, Myanmar (Union of)
Y9Z, Germany (Federal Republic of)
YAZ, Afghanistan (Islamic State of)
YHZ, Indonesia (Republic of)
YIZ, Iraq (Republic of)
YJZ, Vanuatu (Republic of)
YKZ, Syrian Arab Republic
YLZ, Latvia (Republic of)
YMZ, Turkey
YNZ, Nicaragua
YRZ, Romania
YSZ, El Salvador (Republic of)
YUZ, Serbia (Republic of) (WRC-07)

```
YYZ, Venezuela (Republic of)
Z2Z, Zimbabwe (Republic of)
Z3Z, The Former Yugoslav Republic of Macedonia
Z8Z, South Sudan (Republic of)
ZAZ, Albania (Republic of)
ZJZ, United Kingdom of Great Britain and Northern Ireland
ZMZ, New Zealand
ZOZ, United Kingdom of Great Britain and Northern Ireland
ZPZ, Paraguay (Republic of)
ZQZ, United Kingdom of Great Britain and Northern Ireland
ZUZ, South Africa (Republic of)
ZZZ, Brazil (Federative Republic of)
```

```
Overwriting callsign_tbl_sorted.txt
```

In [14]:
```
#..... Other code...
#Country lookup code

# Helper functions for looking up the call signs

def lookupCountry(sign, prefixes):
    pos = bisect.bisect_left(prefixes, sign)
    return prefixes[pos].split(",")[1]


def loadCallSignTable():
    f = open("callsign_tbl_sorted.txt", "r")
    return f.readlines()

# Lookup the locations of the call signs on the
# RDD contactCounts. We load a list of call sign
# prefixes to country code to support this lookup.
signPrefixes = loadCallSignTable()


def processSignCount(sign_count, signPrefixes):
    country = lookupCountry(sign_count[0], signPrefixes)
    count = sign_count[1]
    return (country, count)

contactCounts = sc.parallelize([["ZMZ", 1], ["ZMZ", 3]])

countryContactCounts = (contactCounts
                        .map(lambda signCount: processSignCount(signCount
                        .reduceByKey((lambda x, y: x + y)))

#countryContactCounts.saveAsTextFile("tmp/countries.txt")
```

In [15]:
```
%%writefile beerSales.txt
Week    PRICE12PK   PRICE18PK   PRICE30PK   CASES12PK   CASES18PK   CASES
1   19.98   14.10   15.19   223.5   439 55.00
```

```
2   19.98   18.65   15.19   215.0   98  66.75
3   19.98   18.65   13.87   227.5   70  242.00
4   19.98   18.65   12.83   244.5   52  488.50
5   19.98   18.65   13.16   313.5   64  308.75
6   19.98   18.65   15.19   279.0   72  111.75
7   19.98   18.65   13.92   238.0   47  252.50
8   20.10   18.73   14.42   315.5   85  221.25
9   20.12   18.75   13.83   217.0   59  245.25
10  20.13   18.75   14.50   209.5   63  148.50
11  20.14   18.75   13.87   227.0   57  229.75
12  20.12   18.75   13.64   216.5   54  312.00
13  20.12   13.87   14.31   169.0   404 96.75
14  20.13   14.27   13.85   178.0   380 123.25
15  20.14   18.76   14.20   301.5   65  200.50
16  20.14   18.77   13.64   266.5   40  359.75
17  20.13   13.87   14.33   182.5   456 113.50
18  20.13   14.14   13.14   159.0   176 136.50
19  20.13   18.76   13.81   285.5   61  225.50
20  20.13   18.72   15.19   360.0   91  122.25
21  20.13   18.76   13.13   263.0   59  443.75
22  19.18   18.76   13.63   443.5   83  322.75
23  14.78   18.74   15.19   1101.5  41  53.00
24  16.04   18.75   13.89   814.0   47  140.75
25  20.12   18.75   14.28   365.0   84  210.75
26  19.75   18.75   15.19   510.0   85  110.50
27  19.65   18.75   13.12   580.5   116 568.25
28  19.69   13.79   13.78   251.0   544 115.50
29  20.12   13.49   15.19   237.0   890 58.75
30  20.12   14.89   15.19   302.5   371 77.25
31  20.13   13.94   15.19   229.5   557 66.25
32  20.14   13.67   15.19   188.5   775 50.00
33  15.14   14.43   15.19   795.5   236 46.50
34  14.33   18.75   15.19   1556.5  43  65.75
35  16.24   18.22   13.14   807.5   63  252.75
36  19.93   14.06   13.45   243.0   469 179.00
37  21.06   14.43   13.00   201.5   335 226.25
38  21.19   19.48   13.60   294.0   75  288.50
39  21.23   15.15   14.46   220.5   461 114.25
40  20.12   13.79   14.94   255.5   817 70.00
41  14.73   14.31   15.19   920.5   200 47.75
42  14.57   19.50   15.19   730.0   32  98.75
43  15.94   13.85   15.19   262.5   460 77.00
44  20.70   14.23   13.43   209.5   751 160.50
45  19.57   19.31   14.37   283.0   70  143.50
46  19.60   19.29   15.19   262.5   80  133.00
47  19.94   13.76   15.19   310.0   523 68.75
48  21.28   13.45   15.19   278.5   741 81.75
49  14.56   15.13   15.19   741.5   130 56.25
50  14.39   19.43   15.19   1316.0  69  68.75
51  16.81   13.26   15.19   449.0   493 49.25
52  19.86   13.92   15.19   505.0   814 76.50
```

```
Overwriting beerSales.txt
```

In [7]:
```
df=sc.textFile("beerSales.txt")
```

In [38]:
```
for i in df.take(100):
    print i
```

```
24      16.04   18.75   13.89   814.0   47      140.75
25      20.12   18.75   14.28   365.0   84      210.75
26      19.75   18.75   15.19   510.0   85      110.50
27      19.65   18.75   13.12   580.5   116     568.25
28      19.69   13.79   13.78   251.0   544     115.50
29      20.12   13.49   15.19   237.0   890     58.75
30      20.12   14.89   15.19   302.5   371     77.25
31      20.13   13.94   15.19   229.5   557     66.25
32      20.14   13.67   15.19   188.5   775     50.00
33      15.14   14.43   15.19   795.5   236     46.50
34      14.33   18.75   15.19   1556.5  43      65.75
35      16.24   18.22   13.14   807.5   63      252.75
36      19.93   14.06   13.45   243.0   469     179.00
37      21.06   14.43   13.00   201.5   335     226.25
38      21.19   19.48   13.60   294.0   75      288.50
39      21.23   15.15   14.46   220.5   461     114.25
40      20.12   13.79   14.94   255.5   817     70.00
41      14.73   14.31   15.19   920.5   200     47.75
42      14.57   19.50   15.19   730.0   32      98.75
```

In [20]:
```
## ET 16

#first, we define the linear regression function

from pyspark.mllib.regression import LabeledPoint, LinearRegressionWithSG
import os
import sys
import re
import string


df= sc.textFile("beerSales.txt") #, use_unicode=False)
df2=df.zipWithIndex().filter(lambda x: x[1]>0).map(lambda x: [float(y) fo

def LinReg(dataset):
    lr=LinearRegressionWithSGD.train(dataset, iterations=10000, step=0.00
    predicted_values=dataset.map(lambda pt: (pt.label, lr.predict(pt.feat
    mape = predicted_values.map(
        lambda (v, p): 100*abs(v - p)/v
    ).reduce(
        lambda x, y: x + y) / predicted_values.count()
    print("Mean Abs Pred Error = " + str(mape))

LinReg(df2.map(lambda x: LabeledPoint(x[5],x[1:4])))
```

```
Mean Abs Pred Error = 171.737094196
```

In [21]:
```
## ET 17

import math
LinReg(df2.map(lambda x: LabeledPoint(math.log(x[5]),x[1:4])))
```

```
Mean Abs Pred Error = 37.4346369254
```

In [22]:
```
## ET 18

LinReg(df2.map(lambda x: LabeledPoint(math.log(x[5]), [math.log(xx) for x
```

```
Mean Abs Pred Error = 99.1063425636
```

In [ ]: