Installing Virtual Machine for Hands On Exercises

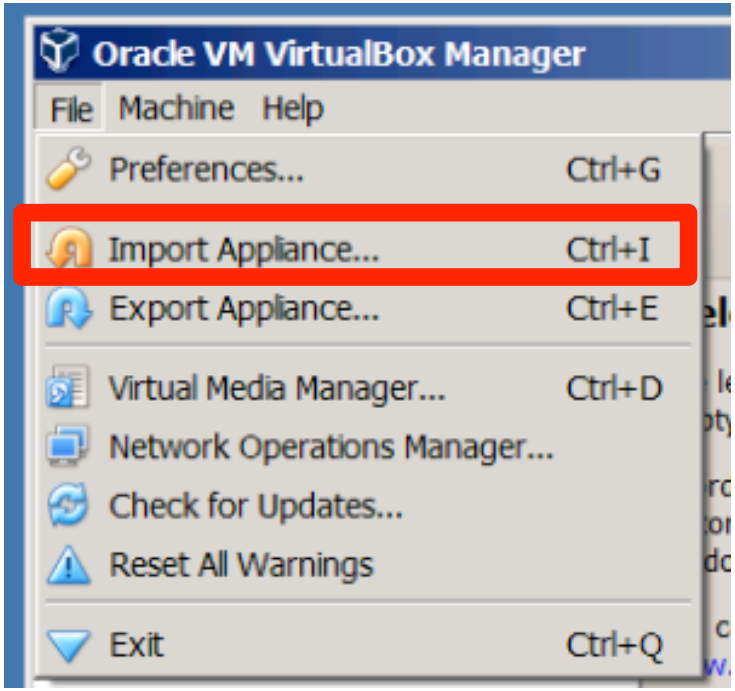
SDSC SAN DIEGO SUPERCOMPUTER CENTER

UC San Diego

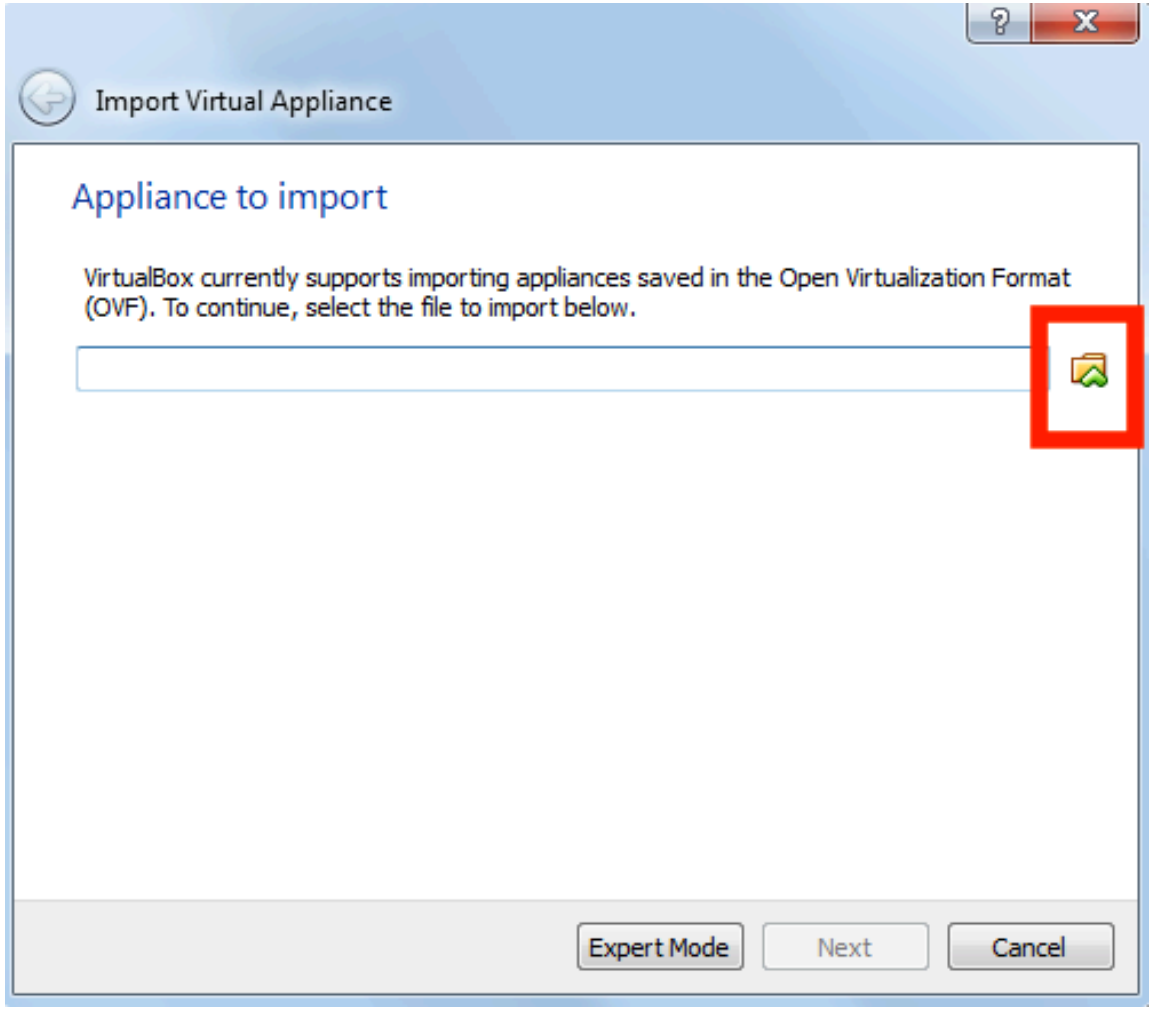
Overview

- Import VM into VirtualBox
- Start VM for Hands On exercises

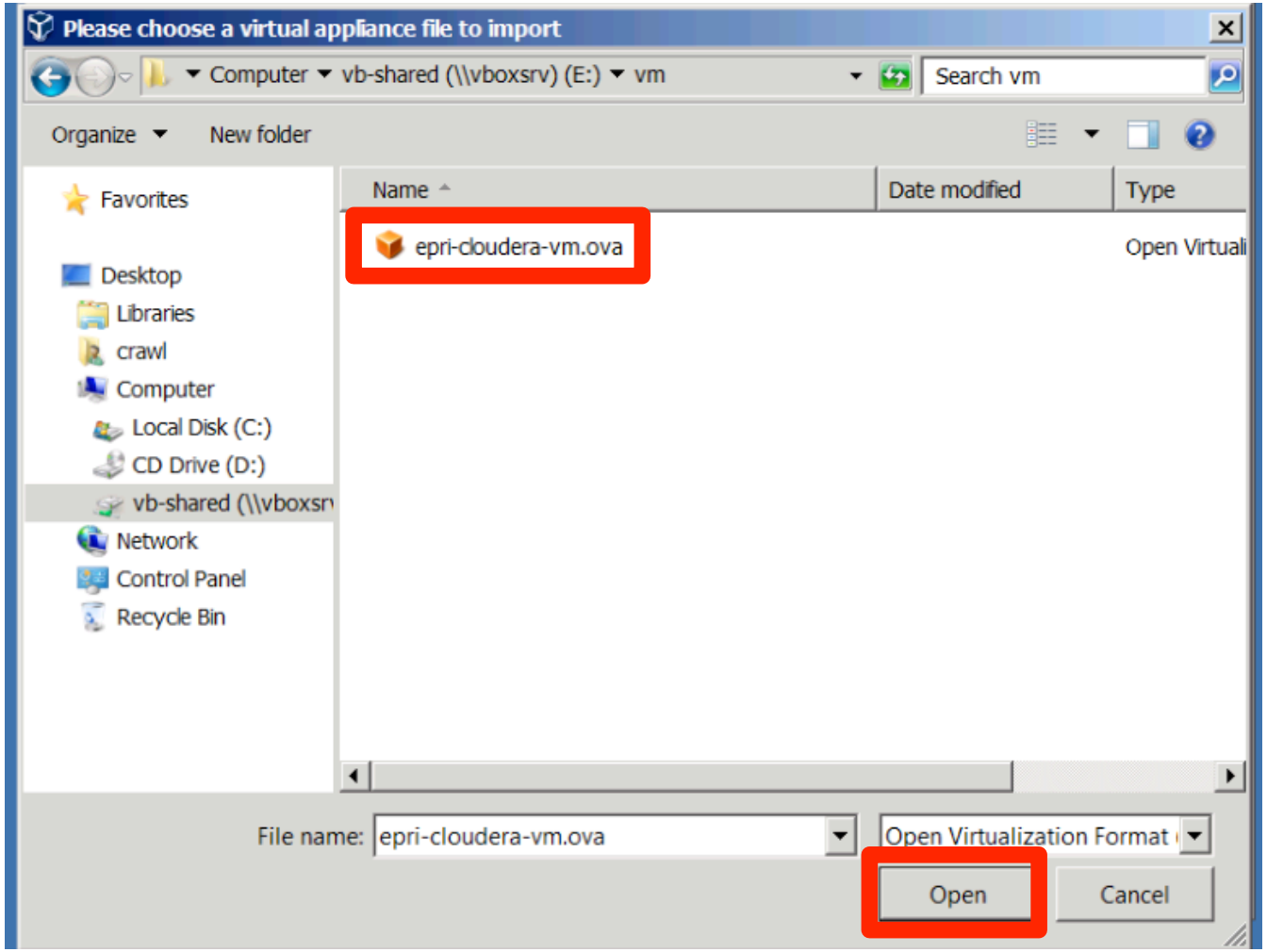
Start VirtualBox, Import Appliance



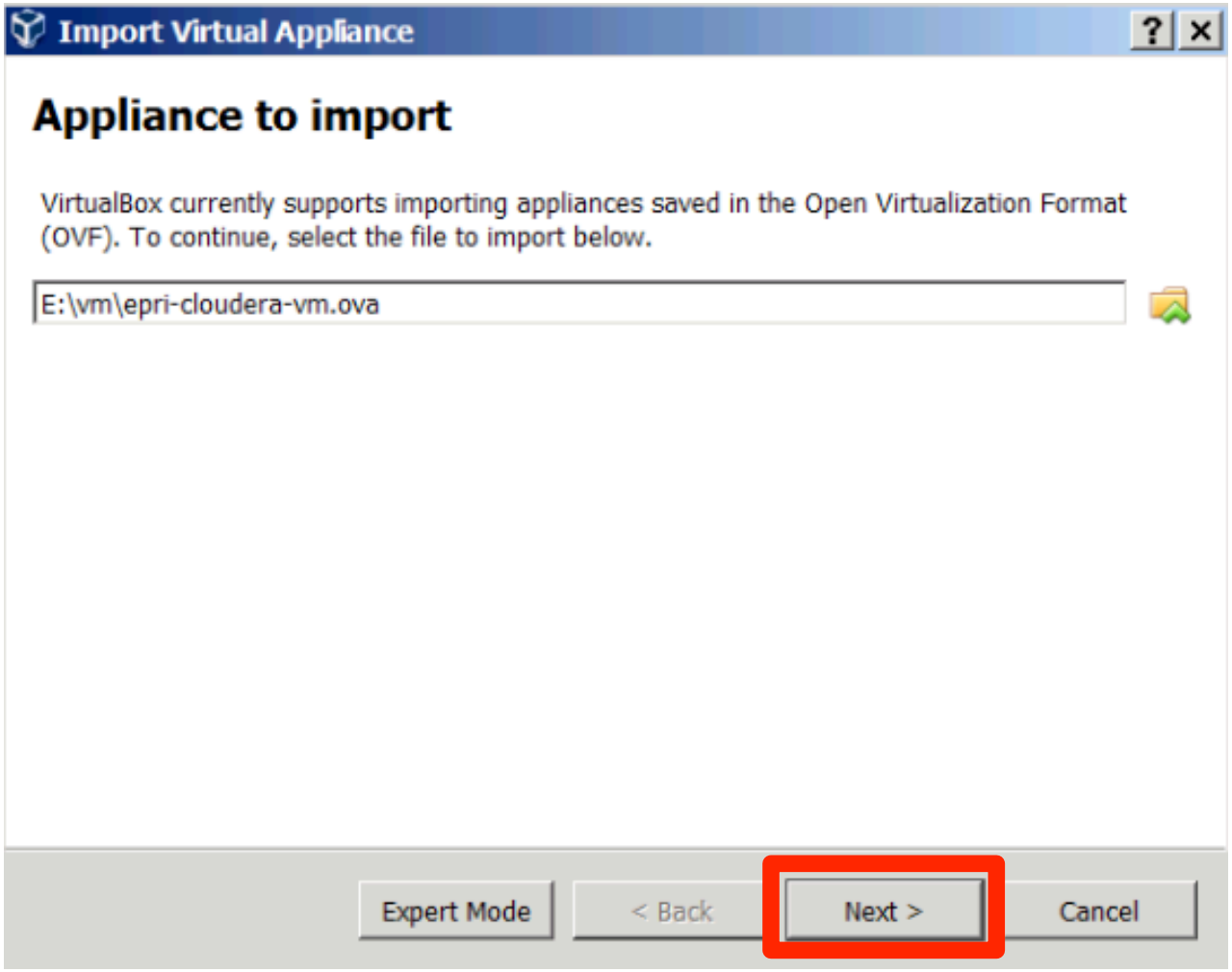
Click on Folder



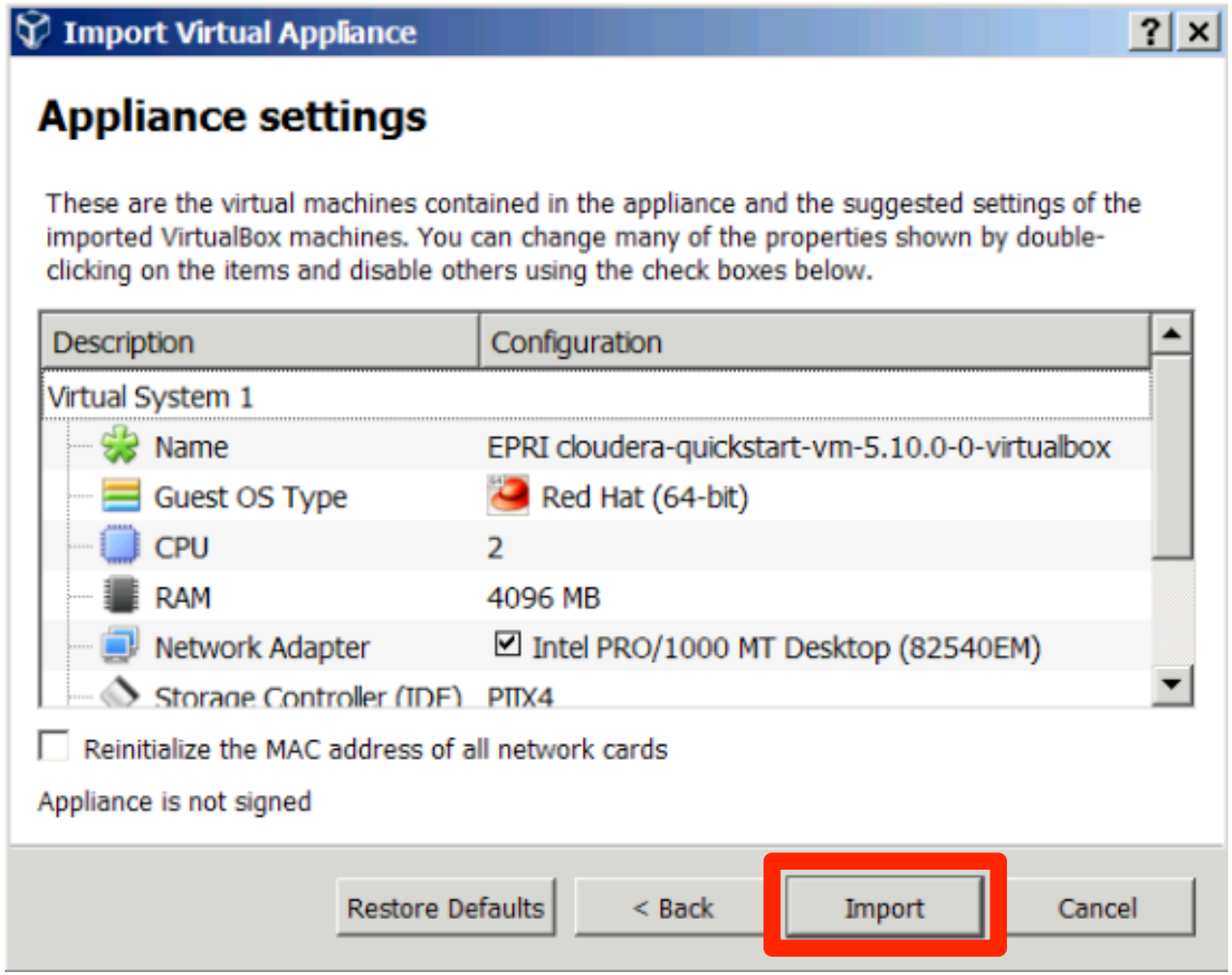
Choose VM Image



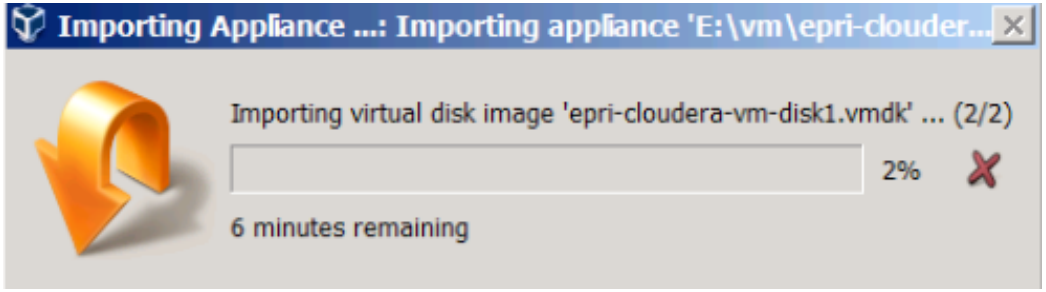
Click Next


Import Virtual Appliance
Appliance to import
VirtualBox currently supports importing appliances saved in the Open Virtualization Format (OVF). To continue, select the file to import below.
E:\vm\epri-cloudera-vm.ova
Expert Mode
< Back
Next >
Cancel

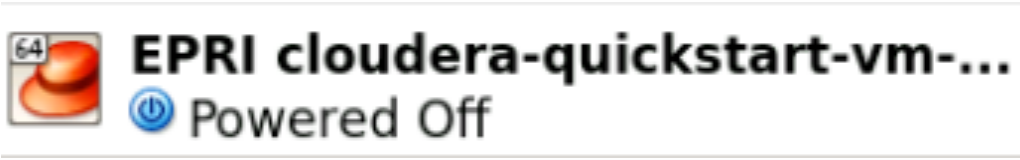
Click Import



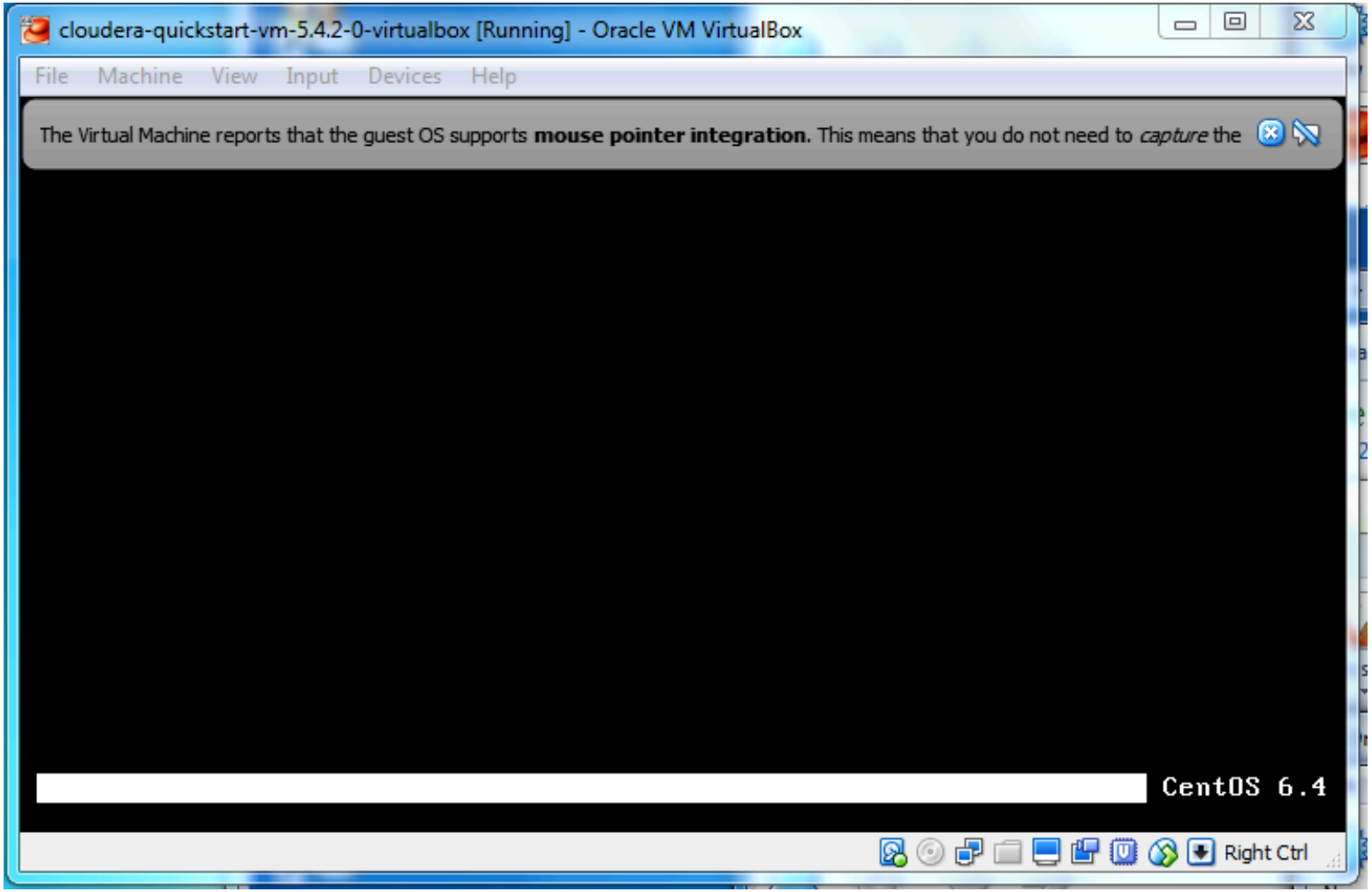
Progress Bar



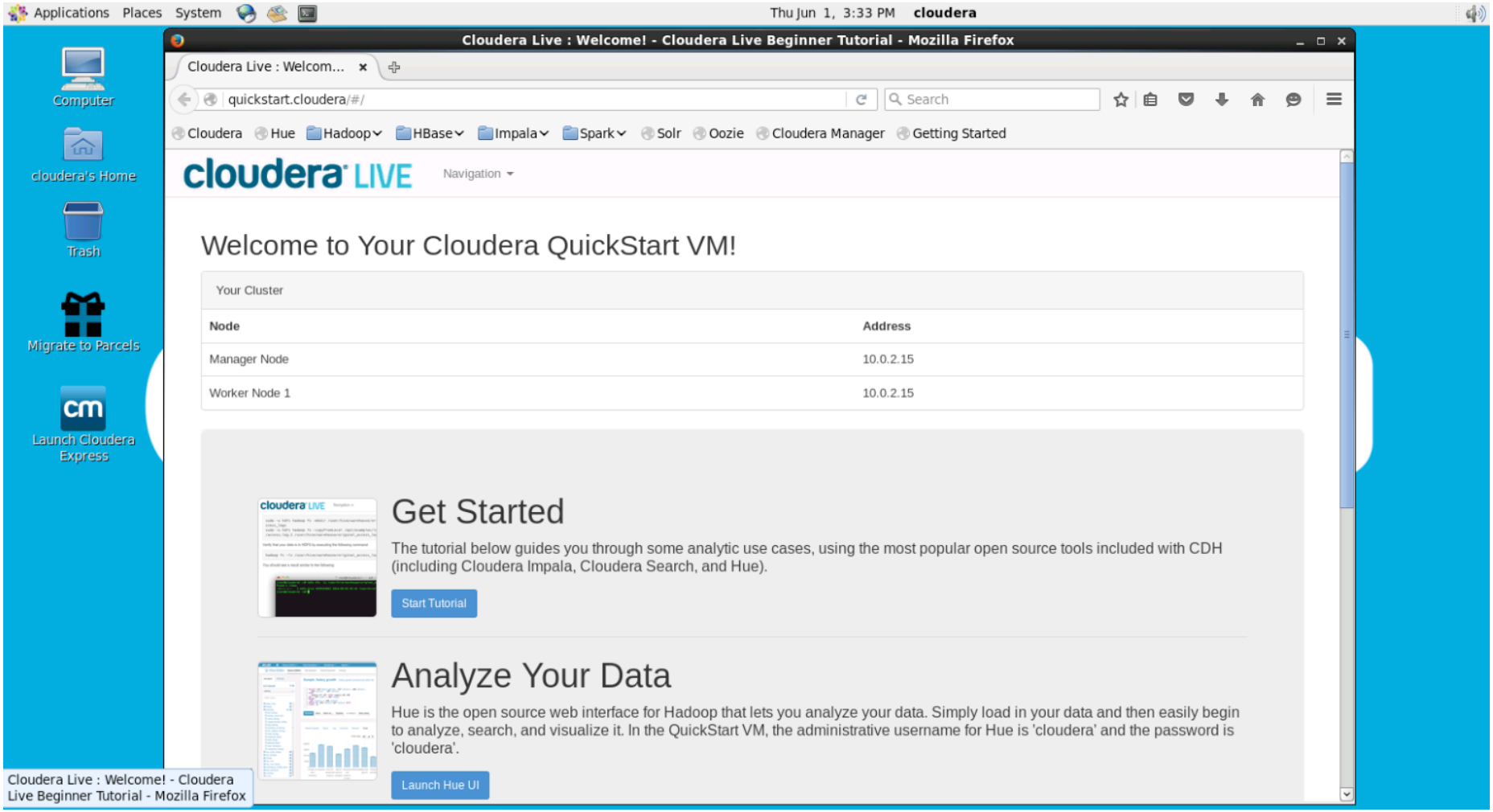
Imported VM



Start VM


cloudera-quickstart-vm-5.4.2-0-virtualbox [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
The Virtual Machine reports that the guest OS supports mouse pointer integration. This means that you do not need to capture the
CentOS 6.4
Right Ctrl

VM Desktop

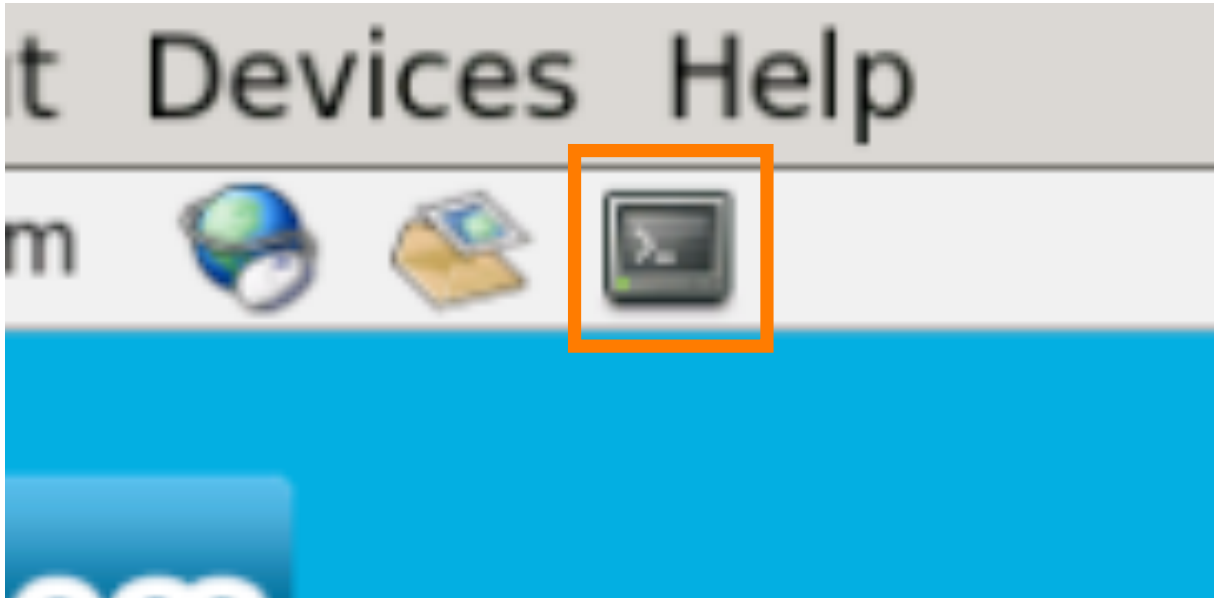

Applications Places System
Thu Jun 1, 3:33 PM cloudera
Cloudera Live : Welcome! - Cloudera Live Beginner Tutorial - Mozilla Firefox
Cloudera Live : Welcom...
quickstart.cloudera/#/
Search
Cloudera Hue Hadoop HBase Impala Spark Solr Oozie Cloudera Manager Getting Started
cloudera LIVE
Navigation
Computer
cloudera's Home
Trash
Migrate to Parcels
Launch Cloudera Express
Welcome to Your Cloudera QuickStart VM!
Your Cluster
Node
Address
Manager Node
10.0.2.15
Worker Node 1
10.0.2.15
Get Started
The tutorial below guides you through some analytic use cases, using the most popular open source tools included with CDH (including Cloudera Impala, Cloudera Search, and Hue).
Start Tutorial
Analyze Your Data
Hue is the open source web interface for Hadoop that lets you analyze your data. Simply load in your data and then easily begin to analyze, search, and visualize it. In the QuickStart VM, the administrative username for Hue is 'cloudera' and the password is 'cloudera'.
Launch Hue UI
Cloudera Live : Welcome! - Cloudera Live Beginner Tutorial - Mozilla Firefox

Hands-On 1: WordCount

Hands-On Overview

- **Hadoop command line**
  - Copy data to HDFS
  - Run WordCount
  - View results
- **Spark in Python Notebook**
  - Load data into an RDD
  - Use Spark API to perform WordCount

Open a terminal window



Input Data

1. **Go to data directory**
   - *cd epri*
   - *ls*
   - *more words.txt*

```
The Project Gutenberg EBook of The Complete Works of William Sh
William Shakespeare

This eBook is for the use of anyone anywhere at no cost and wit
almost no restrictions whatsoever.  You may copy it, give it av
re-use it under the terms of the Project Gutenberg License incl
with this eBook or online at www.gutenberg.org

** This is a COPYRIGHTED Project Gutenberg eBook, Details Belov
**     Please follow the copyright guidelines in this file.

Title: The Complete Works of William Shakespeare

Author: William Shakespeare

Posting Date: September 1, 2011 [EBook #100]
Release Date: January, 1994
```

Copy Data to HDFS

1. **Copy command:**
   - *hadoop fs **-copyFromLocal** words.txt*

2. **Verify file was copied:**
   - *hadoop fs -ls*

```
(py35) hadoop fs -ls
Found 1 items
-rw-r--r--   1 cloudera cloudera    5589889 2017-05-31 17:00 words.txt
```

Run WordCount

hadoop jar hadoop-examples.jar wordcount words.txt count

```
 INFO mapreduce.JobSubmitter: number of splits:1
 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1496251834393_0001
 INFO impl.YarnClientImpl: Submitted application application_1496251834393_0001
 INFO mapreduce.Job: The url to track the job: http://quickstart.cloudera:8088/proxy
01/
 INFO mapreduce.Job: Running job: job_1496251834393_0001
 INFO mapreduce.Job: Job job_1496251834393_0001 running in uber mode : false
 INFO mapreduce.Job:  map 0% reduce 0%
 INFO mapreduce.Job:  map 100% reduce 0%
 INFO mapreduce.Job:  map 100% reduce 100%
 INFO mapreduce.Job: Job job_1496251834393_0001 completed successfully
 INFO mapreduce.Job: Counters: 49
```

List Results

1. List results

- *hadoop fs -ls*

```
(py35) hadoop fs -ls
Found 2 items
drwxr-xr-x   - cloudera cloudera          0 2017-06-02 10:25 count
-rw-r--r--   1 cloudera cloudera    5589889 2017-05-31 17:00 words.txt
```

- *hadoop fs -ls count*

```
(py35) hadoop fs -ls count
Found 2 items
-rw-r--r--   1 cloudera cloudera          0 2017-06-02 10:25 count/_SUCCESS
-rw-r--r--   1 cloudera cloudera     720972 2017-06-02 10:25 count/part-r-00000
```

View Results

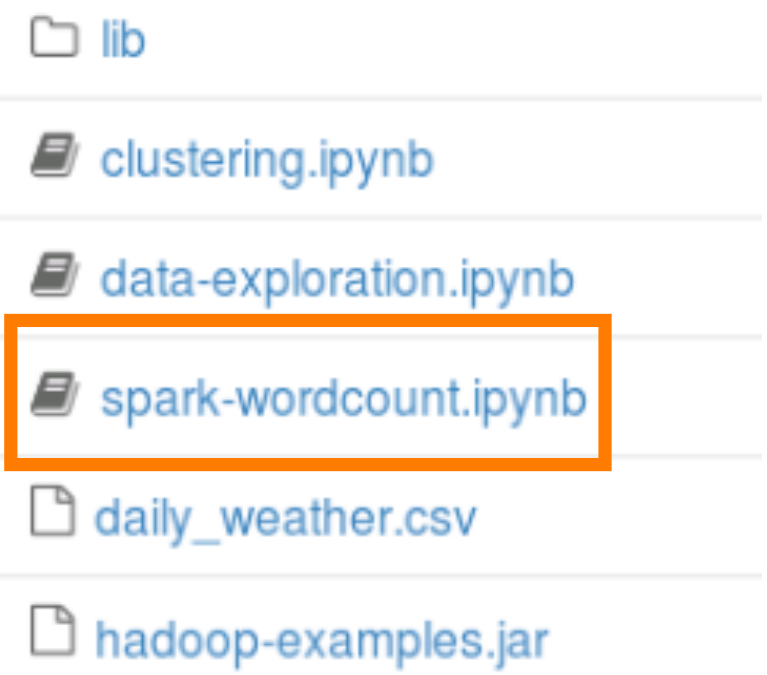
1. **Copy results from HDFS to local file system**
   - *hadoop fs* ***-copyToLocal*** *count/part-r-00000 count.txt*

2. **View results**
   - *more count.txt*

```
"       241
"'Tis   1
"A      4
"AS-IS".        1
"Air,"  1
"Alas,  1
"Amen"  2
"Amen"? 1
"Amen," 1
"And    1
"Aroint 1
"B      1
"Black  1
"Break  1
"Brutus"        1
"Brutus,        2
"C      1
"Caesar"?       1
"Caesar,        1
"Caesar."       2
```

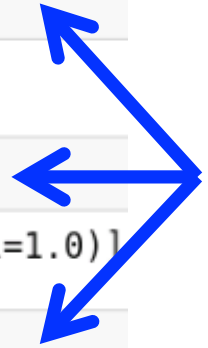
Open Spark Wordcount Notebook

1. Open new terminal window
2. *cd epri*
3. Run *pyspark*
4. Click on *spark-wordcount.ipynb*



iPython Notebooks

```
In [4]: accuracy = evaluator.evaluate(predictions)
        print("Accuracy = %g " % (accuracy))

        Accuracy = 0.809524

In [5]: predictions.rdd.take(2)

Out[5]: [Row(prediction=1.0, label=1.0), Row(prediction=1.0, label=1.0)]

In [6]: predictions.rdd.map(tuple).take(2)

Out[6]: [(1.0, 1.0), (1.0, 1.0)]
```



Input Cells

Shift-Enter or  to execute cell

iPython Notebooks

```
In [4]: accuracy = evaluator.evaluate(predictions)
        print("Accuracy = %g " % (accuracy))

        Accuracy = 0.809524

In [5]: predictions.rdd.take(2)

Out[5]: [Row(prediction=1.0, label=1.0), Row(prediction=1.0, label=1.0)]

In [6]: predictions.rdd.map(tuple).take(2)

Out[6]: [(1.0, 1.0), (1.0, 1.0)]
```

Output Cells

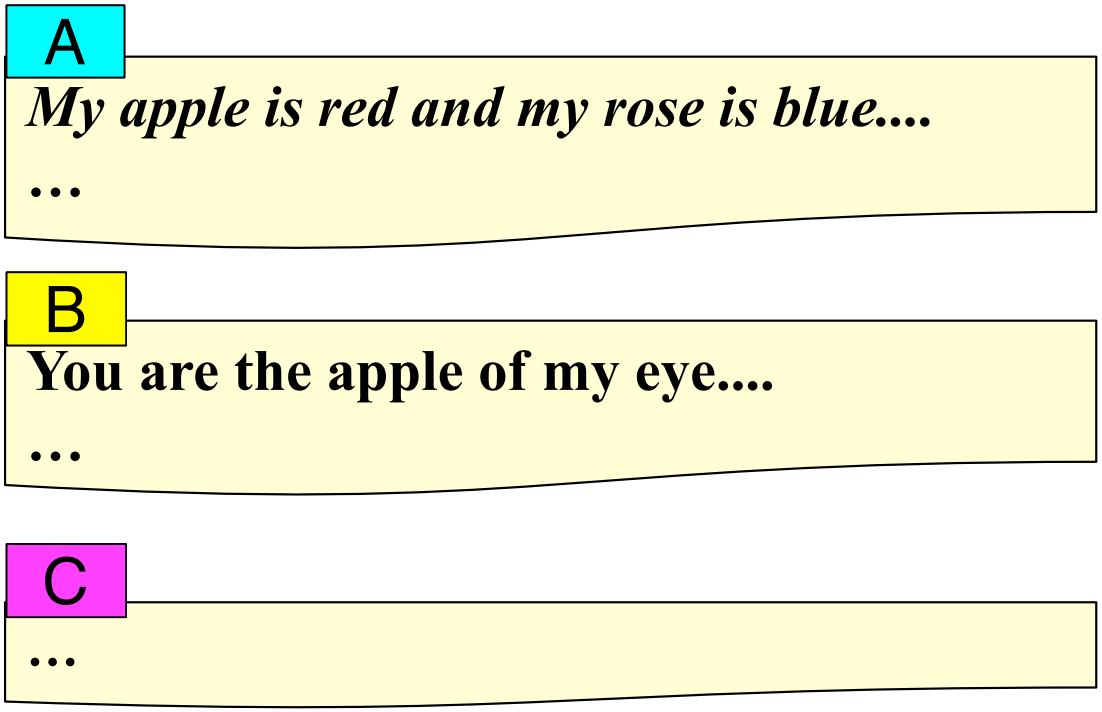
Shift-Enter or  to execute cell

Load Data into Spark RDD

lines = sc.textFile("file:/home/cloudera/epri/words.txt")

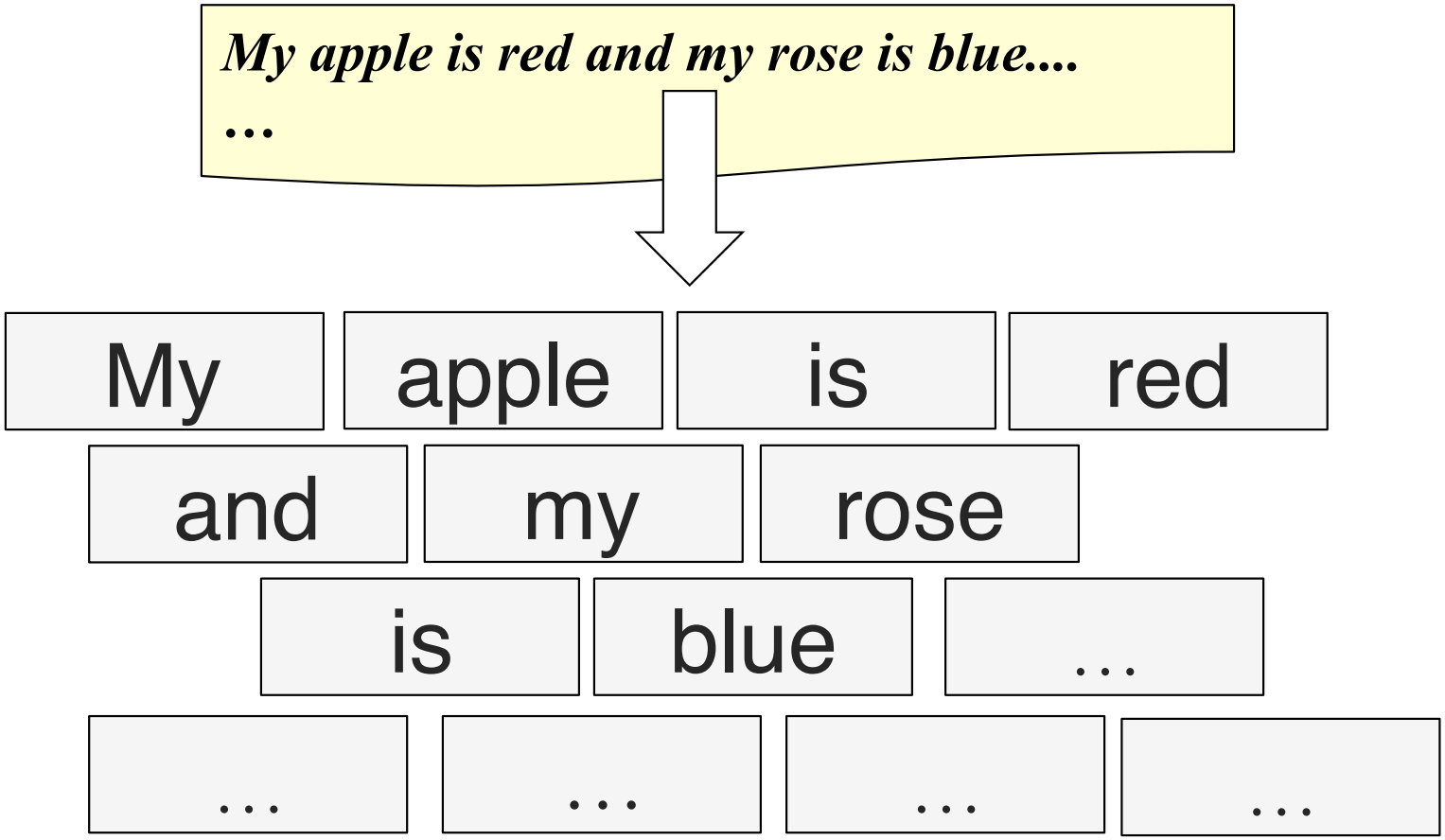
- **Count the lines:**

lines.count()
= 124787



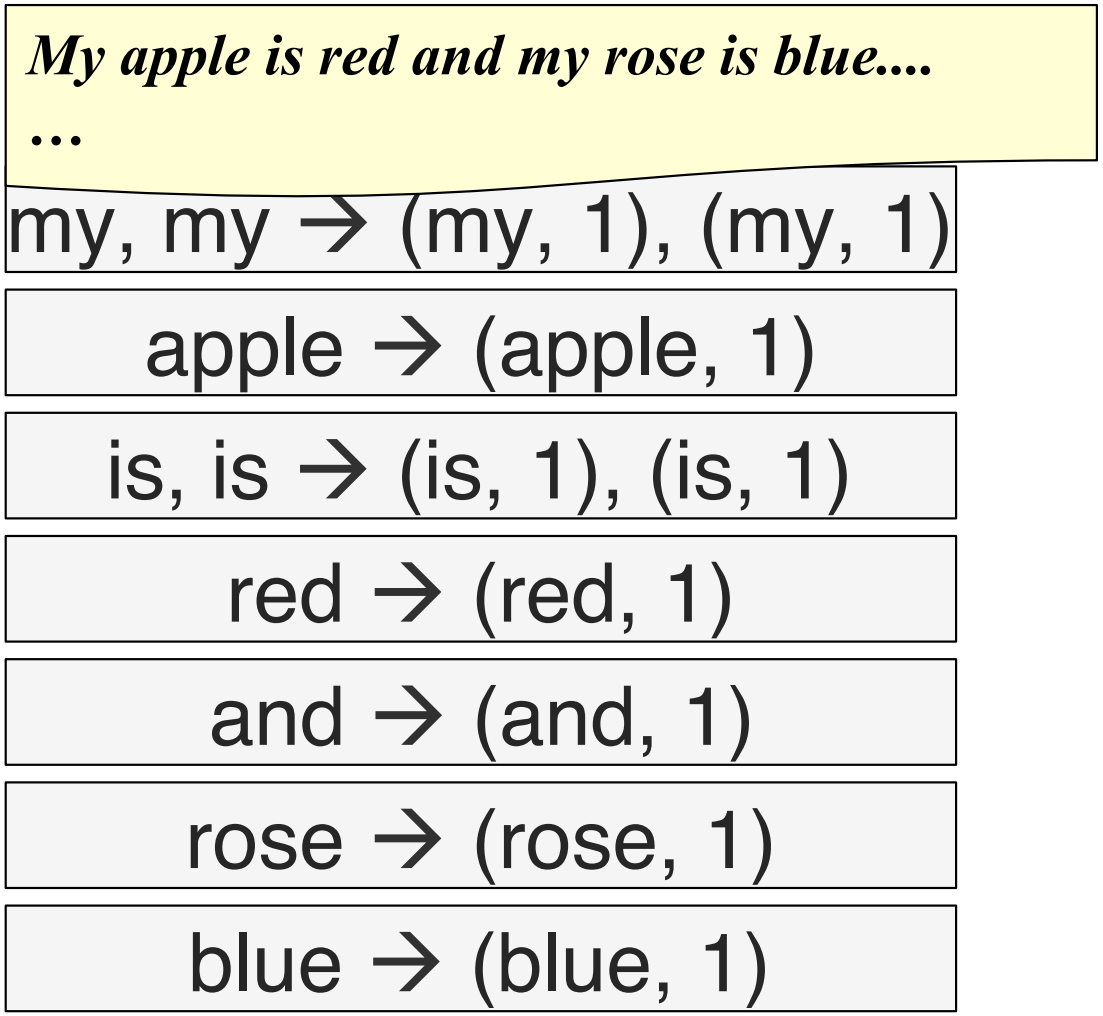
Convert Lines into Words

words = lines.flatMap(lambda line : line.split(" "))



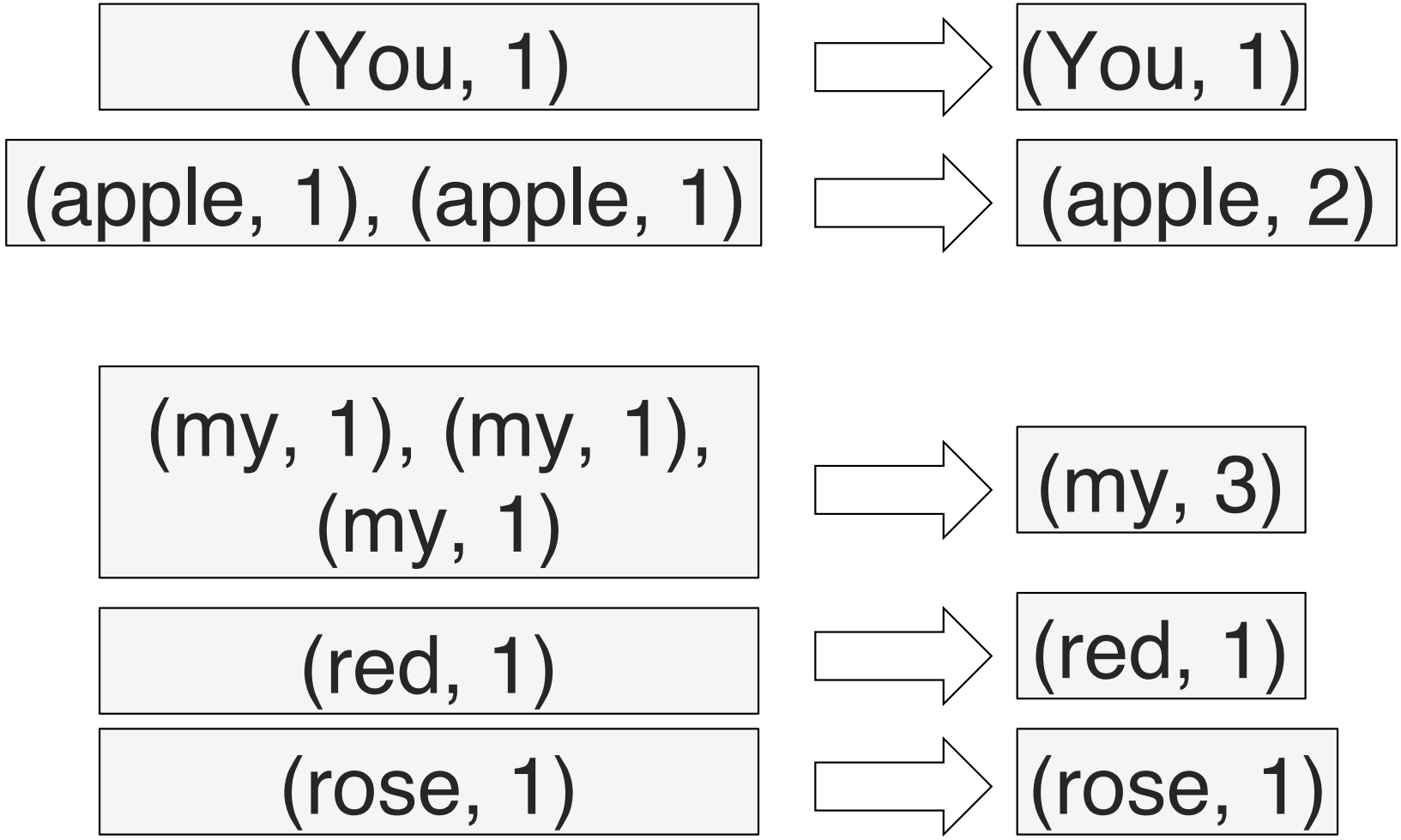
Create Tuples from Words

tuples = words.map(lambda word : (word, 1))



Count Word-Tuples

counts = tuples.reduceByKey(lambda a, b : (a + b))



Write Counts and View Results

counts.coalesce(1).sortByKey()
.saveAsTextFile("file:/home/cloudera/epri/count2")

- **In terminal:**
  - *more count2/part-00000*

```
('', 506610)
('"', 241)
('"\'Tis', 1)
('"A', 4)
('"AS-IS".', 1)
('"Air,"', 1)
('"Alas,', 1)
('"Amen"', 2)
('"Amen"?', 1)
('"Amen,"', 1)
('"And', 1)
('"Aroint', 1)
('"B', 1)
('"Black', 1)
('"Break', 1)
('"Brutus"', 1)
('"Brutus,', 2)
('"C', 1)
('"Caesar"?', 1)
('"Caesar,', 1)
('"Caesar."', 2)
```

Hands-On 2: Data Exploration and Clustering

SDSC SAN DIEGO SUPERCOMPUTER CENTER

UC San Diego

Overview

- **Weather station measurements**
- **Data Exploration**
  - Load into Spark DataFrame
  - Describe schema
  - Show summary statistics
  - Calculate correlation between features
- **Cluster to identify different weather patterns**
  - Spark KMeans
  - Elbow plot
  - Parallel plots

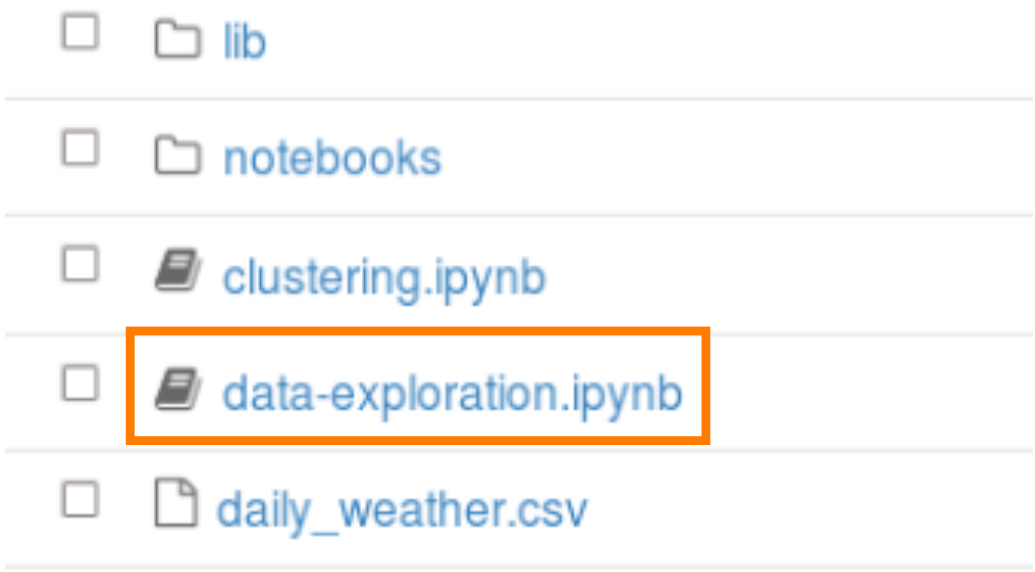
Dataset Description

- **Measurements from weather station on Mt. Woodson, San Diego**
- **Air temperature, humidity, wind speed, wind direction, etc.**
- **Three years of data: Sep. 2011 - Sep. 2014**
- ***minute_weather.csv***
  - measurement every minute
- ***daily_weather.csv***
  - aggregated measurements

Dataset Description

- **Measurements from weather station on Mt. Woodson, San Diego**
- **Air temperature, humidity, wind speed, wind direction, etc.**
- **Three years of data: Sep. 2011 - Sep. 2014**
- ***minute_weather.csv*** Clustering
  - measurement every minute
- ***daily_weather.csv*** Data Exploration
  - aggregated measurements

Open Data Exploration Notebook



Load Data into Spark DataFrame

```
from pyspark.sql import SQLContext

sqlContext = SQLContext(sc)

df = sqlContext.read.load('file:/home/cloudera/epri/daily_weather.csv',
                          format='com.databricks.spark.csv',
                          header='true',inferSchema='true')
```

1-2: Create the Spark SQL Context

3: Read the daily weather data into a DataFrame

Examine Schema

df.printSchema()

```
root
 |-- number: integer (nullable = true)
 |-- air_pressure_9am: double (nullable = true)
 |-- air_temp_9am: double (nullable = true)
 |-- avg_wind_direction_9am: double (nullable = true)
 |-- avg_wind_speed_9am: double (nullable = true)
 |-- max_wind_direction_9am: double (nullable = true)
 |-- max_wind_speed_9am: double (nullable = true)
 |-- rain_accumulation_9am: double (nullable = true)
 |-- rain_duration_9am: double (nullable = true)
 |-- relative_humidity_9am: double (nullable = true)
 |-- relative_humidity_3pm: double (nullable = true)
```

Show Summary Statistics

df.describe().toPandas().transpose()

	0	1	2	3	4
summary	count	mean	stddev	min	max
number	1095	547.0	316.24357700987383	0	1094
air_pressure_9am	1092	918.8825513138094	3.184161180386833	907.9900000000024	929.3200000000012
air_temp_9am	1090	64.93300141287072	11.175514003175877	36.752000000000685	98.90599999999992
avg_wind_direction_9am	1091	142.2355107005759	69.13785928889189	15.500000000000046	343.4
avg_wind_speed_9am	1092	5.50828424225493	4.5528134655317185	0.69345139999974	23.554978199999763
max_wind_direction_9am	1092	148.95351796516923	67.23801294602953	28.89999999999991	312.19999999999993
max_wind_speed_9am	1091	7.019513529175272	5.598209170780958	1.1855782000000479	29.84077959999996
rain_accumulation_9am	1089	0.20307895225211126	1.5939521253574893	0.0	24.01999999999907
rain_duration_9am	1092	294.1080522756142	1598.0787786601481	0.0	17704.0
relative_humidity_9am	1095	34.24140205923536	25.472066802250055	6.090000000001012	92.6200000000002
relative_humidity_3pm	1095	35.34472714825898	22.524079453587273	5.3000000000006855	92.2500000000003

Calculate Correlation of Air Temperature vs Humidity

df.stat.corr("air_temp_9am", "relative_humidity_9am")

= -0.536670…

Show Plots in Notebook

%matplotlib inline

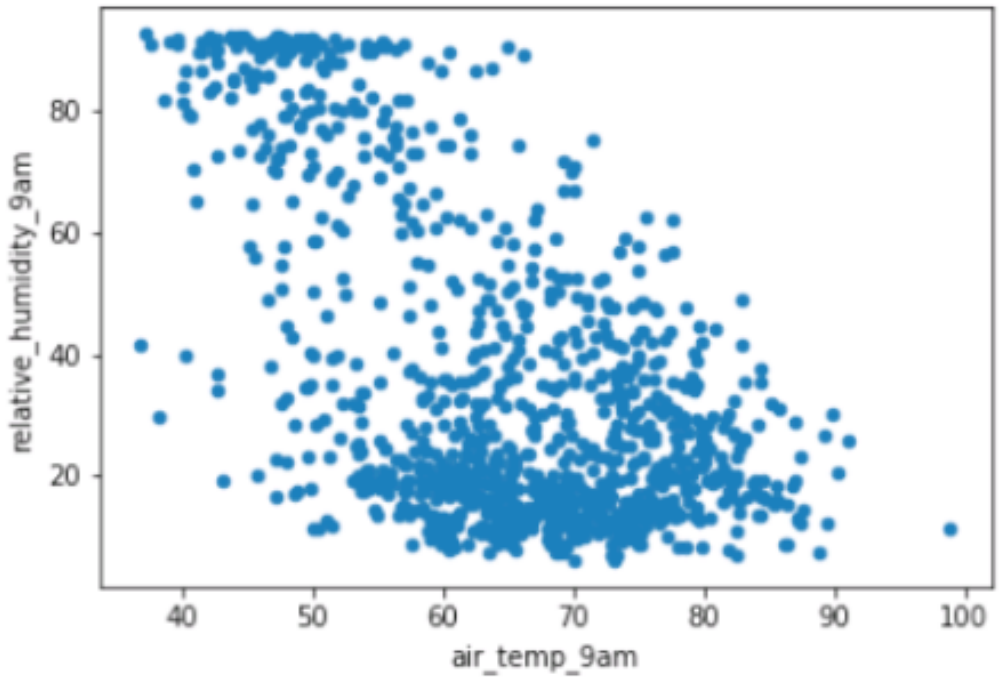
Scatter Plot of Air Temperature vs Humidity

df.select('air_temp_9am', 'relative_humidity_9am')

.toPandas()

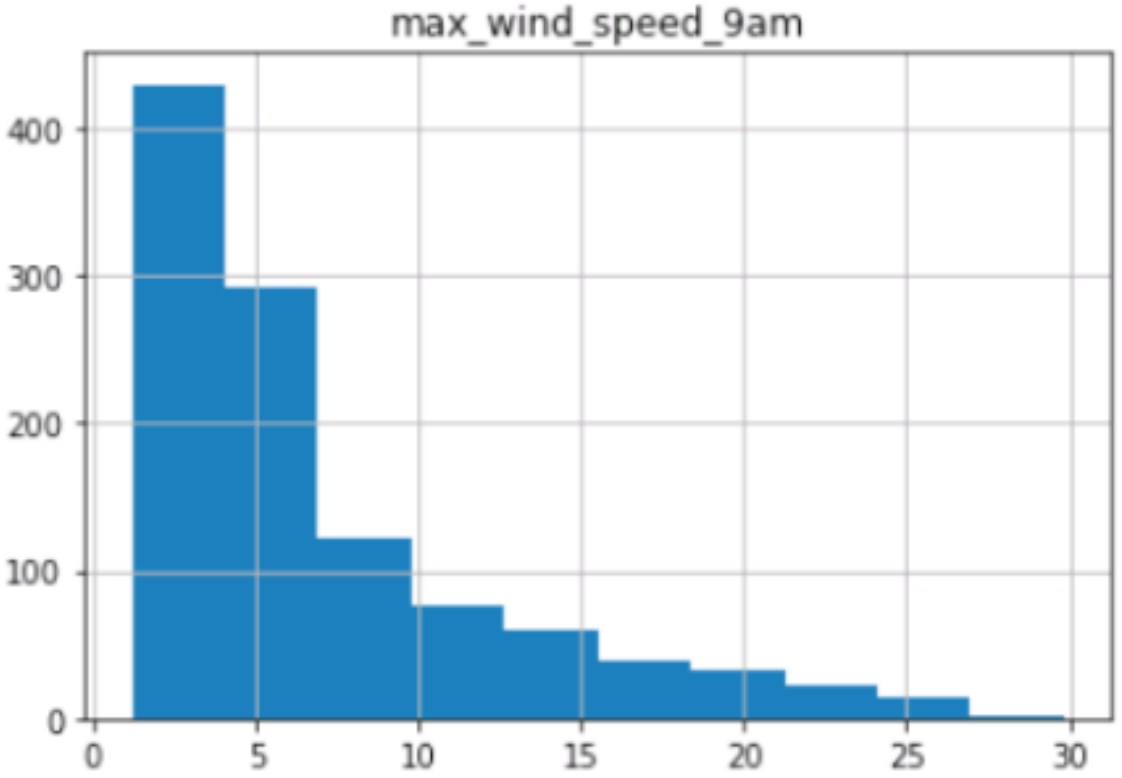
.plot

.scatter(x='air_temp_9am', y='relative_humidity_9am')



Histogram of Max Wind Speed

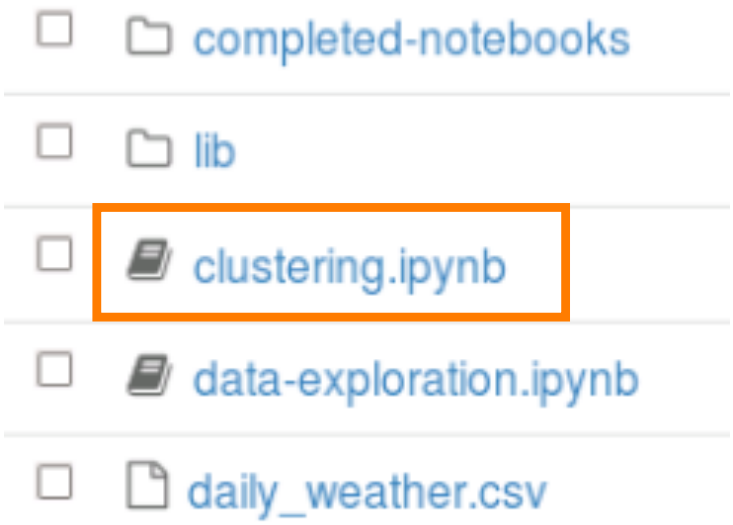
df.select('max_wind_speed_9am').toPandas().hist()



Clustering to Identify Santa Ana Conditions

- **Strong, dry winds in Southern California**
  - wind speed > 30mph
  - relative humidity < 10%
- **Extreme fire danger**
  - May 2014, swarm of 14 wildfires in San Diego County
  - 2008, Witch Fire, ~200,000 acres
  - 2003, Cedar Fire, ~280,000 acres

Open Clustering Notebook



Load Libraries and Minute Weather Data

In [ ]:
```
from pyspark.sql import SQLContext
from pyspark.ml.clustering import KMeans
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.feature import StandardScaler
import utils
%matplotlib inline
```

In [ ]:
```
sqlContext = SQLContext(sc)
df = sqlContext.read.load('file:/home/cloudera/epri/minute_weather.csv',
                          format='com.databricks.spark.csv',
                          header='true',inferSchema='true')
```

Count Rows and Filter Data

df.count()
= 1587257

filteredDF = df.filter((df.rowID % 100) == 0)

filteredDF.count()
= 15873

Show Summary Statistics

filteredDF.describe().toPandas().transpose()

	0	1	2	3	4
summary	count	mean	stddev	min	max
rowID	15873	793600.0	458228.4746717515	0	1587200
air_pressure	15873	916.8291627291587	3.0517222151797943	905.1	929.4
air_temp	15873	61.854689094688936	11.83541379082148	32.36	96.44
avg_wind_direction	15870	161.2875236294896	95.3131612965649	0.0	359.0
avg_wind_speed	15870	2.7928040327662296	2.0705061984600173	0.1	20.1
max_wind_direction	15870	162.70094517958412	92.26960112663167	0.0	359.0
max_wind_speed	15870	3.41462507876495	2.428906406812135	0.1	20.9
min_wind_direction	15870	166.64429741650915	97.82483630682509	0.0	359.0
min_wind_speed	15870	2.1522684310018896	1.7581135042599596	0.0	19.5

Drop Null Data

workingDF = filteredDF.na.drop()
workingDF.count()

= 15869

Create Feature Vector

```
featuresUsed = ['air_pressure', 'air_temp', 'avg_wind_direction', 'avg_wind_speed',
        'max_wind_speed','relative_humidity']
assembler = VectorAssembler(inputCols=featuresUsed, outputCol="features_unscaled")
assembled = assembler.transform(workingDF)
```

Scale Data

```
scaler = StandardScaler(inputCol="features_unscaled", outputCol="features", withStd=True,
scalerModel = scaler.fit(assembled)
scaledData = scalerModel.transform(assembled)
```

Use One-third Data for Elbow Plot

```
# Use one-third data for elbow plot

scaledData = scaledData.select("features", "rowID")

elbowset = scaledData.filter((scaledData.rowID % 3) == 0).select("features")
elbowset.persist()
elbowset.count()
```

```
5289
```

Generate Clusters for Elbow Plot

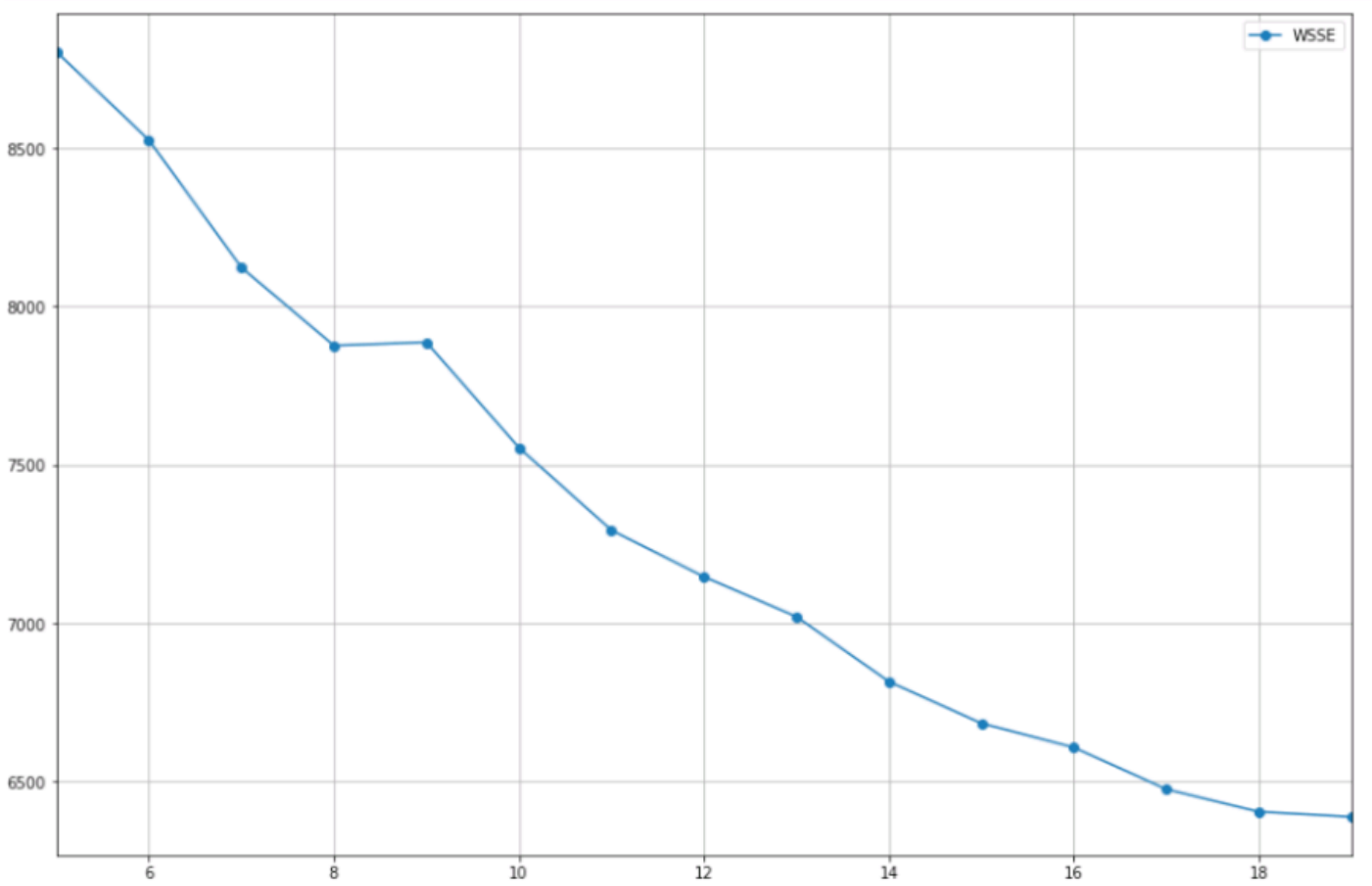
clusters = range(5, 20)

wsseList = utils.elbow(elbowset, clusters)

```
Training for cluster size 5
.....................WSSE = 8804.118354684928
Training for cluster size 6
.....................WSSE = 8525.400895523859
Training for cluster size 7
.....................WSSE = 8123.225352699479
Training for cluster size 8
.....................WSSE = 7876.960906428347
Training for cluster size 9
.....................WSSE = 7887.3819888393555
Training for cluster size 10
.....................WSSE = 7554.953661841033
Training for cluster size 11
.....................WSSE = 7294.206903885911
Training for cluster size 12
```

Show Elbow Plot

utils.elbow_plot(wsseList, clusters)



10x Data Elbow Plot

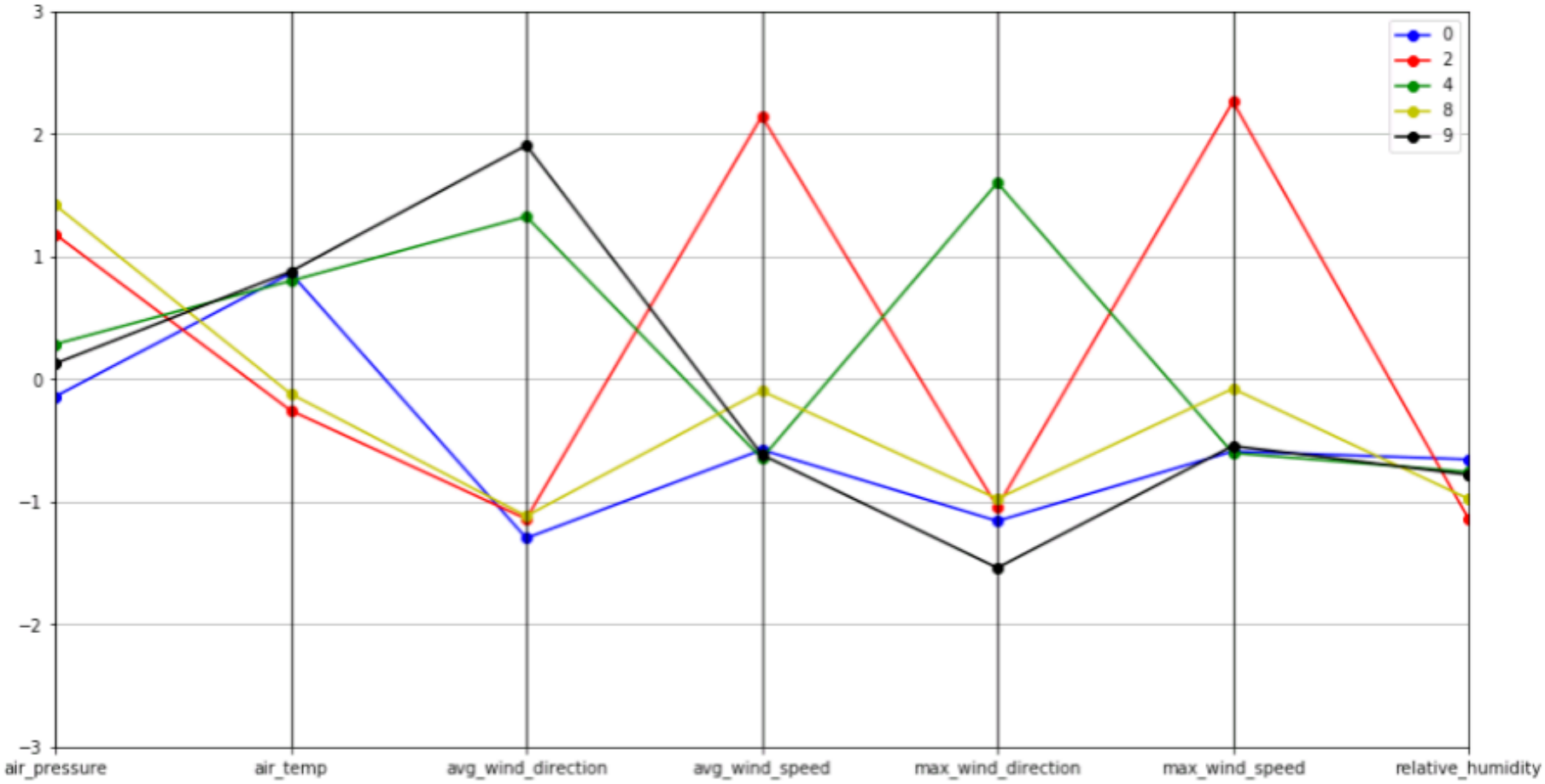
Run KMeans for k = 12 and Extract Cluster Centers

```
# Run KMeans for k = 12

scaledDataFeat = scaledData.select("features")
scaledDataFeat.persist()

kmeans = KMeans(k=12, seed=1)
model = kmeans.fit(scaledDataFeat)
transformed = model.transform(scaledDataFeat)
```

```
# Compute cluster centers

centers = model.clusterCenters()
P = utils.pd_centers(featuresUsed, centers)
centers
```

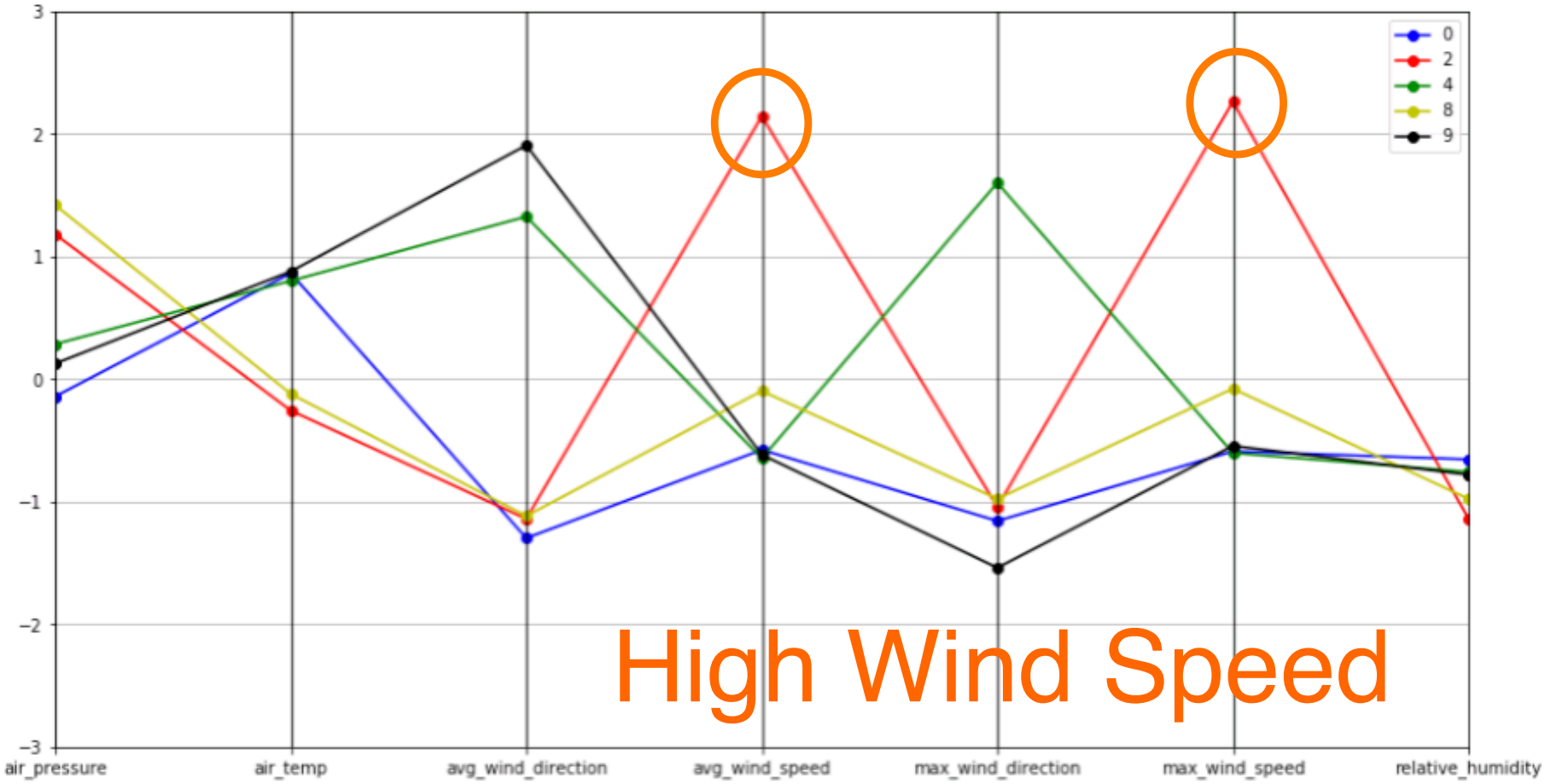
Parallel Plot: Dry Days

utils.parallel_plot(P[P['relative_humidity'] < -0.5], P)



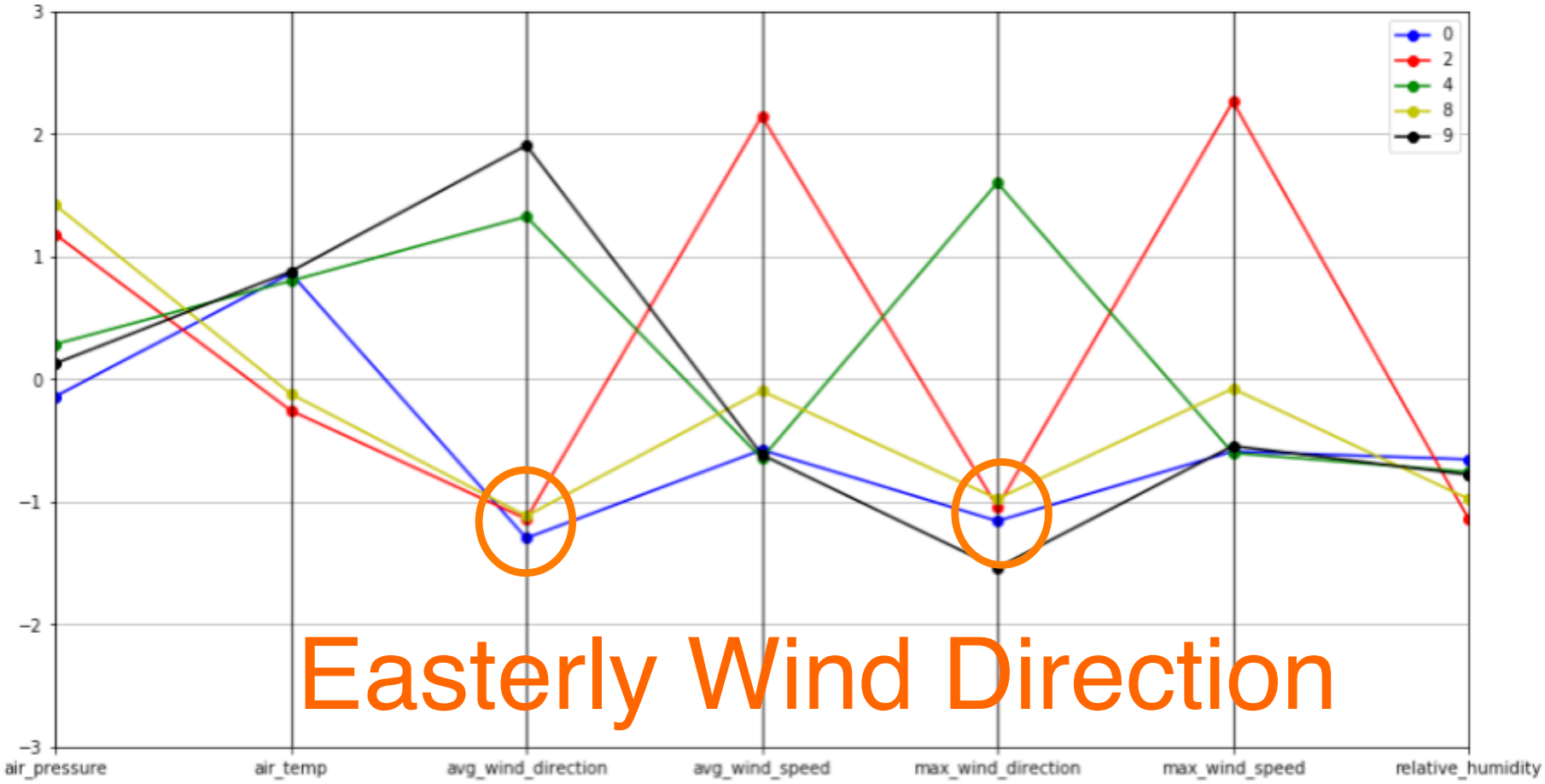
Parallel Plot: Dry Days

utils.parallel_plot(P[P['relative_humidity'] < -0.5], P)



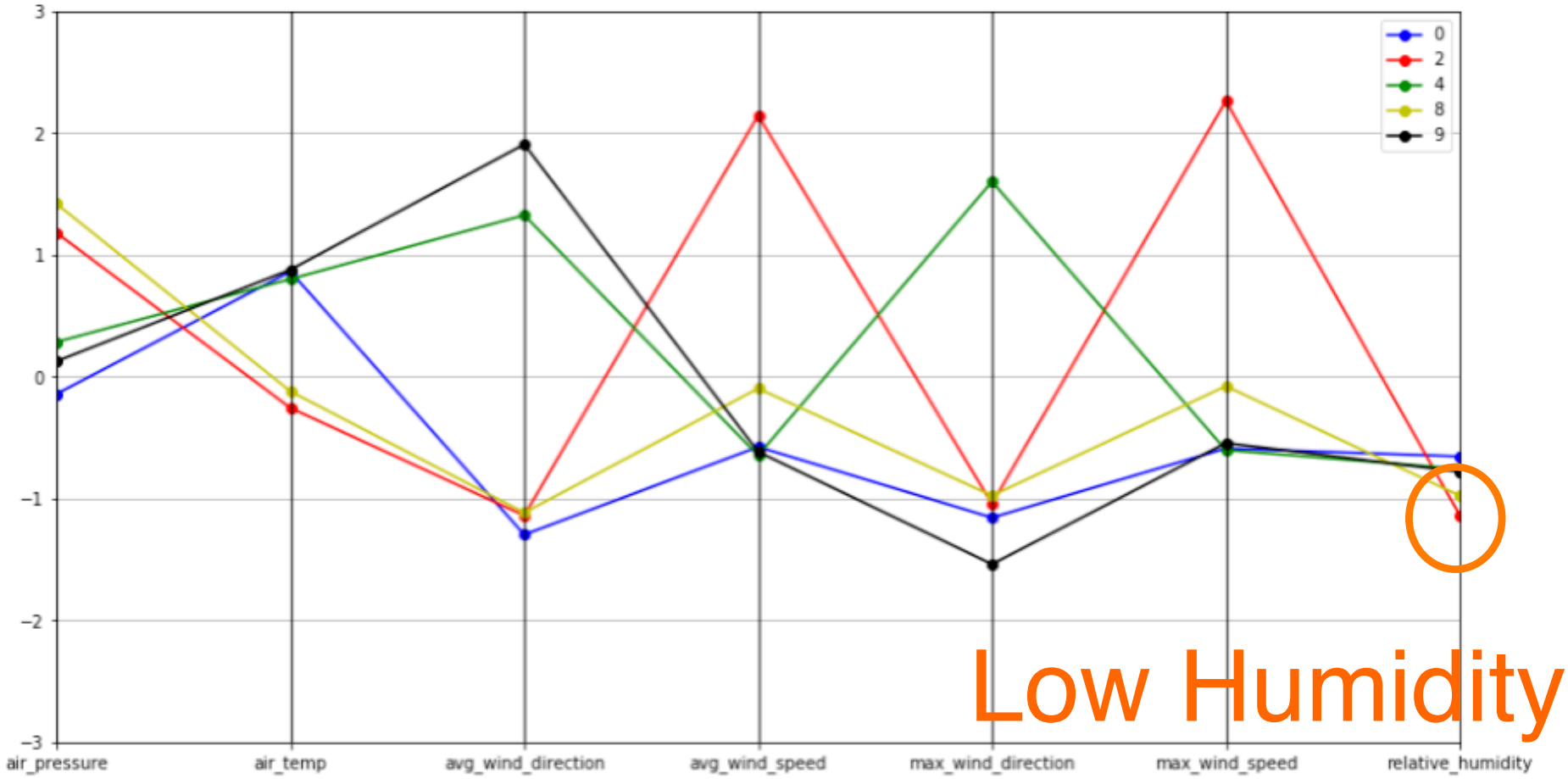
Parallel Plot: Dry Days

utils.parallel_plot(P[P['relative_humidity'] < -0.5], P)



Parallel Plot: Dry Days

utils.parallel_plot(P[P['relative_humidity'] < -0.5], P)



Parallel Plot Exercises

- Humid days
- Hot days
- Cool days

Thank You!

- Data and Notebooks for exercises at:

https://github.com/words-sdsc/epri