Contents

[WEEK 1 Welcome 2](#_Toc12736882)

[Software Installation Directions 2](#_Toc12736883)

[Installation 2](#_Toc12736884)

[Setup Using Docker 2](#_Toc12736885)

[Installing Docker 2](#_Toc12736886)

[Pulling the Docker image 2](#_Toc12736887)

[Running Docker Images 2](#_Toc12736888)

[Setup From Scratch 2](#_Toc12736889)

[Map-Reduce and Spark 3](#_Toc12736890)

[The Memory Hierarchy 3](#_Toc12736891)

[Spark Basics 3](#_Toc12736892)

[Quiz 1 3](#_Toc12736893)

[Programming Assignment 1 3](#_Toc12736894)

[Pyspark and RDDs 3](#_Toc12736895)

[Spark SQL and DataFrames 3](#_Toc12736896)

[Preparing for Data Analysis 3](#_Toc12736897)

[Quiz 2 3](#_Toc12736898)

[Programming Assignment 2 3](#_Toc12736899)

[PCA and Weather Analysis 3](#_Toc12736900)

[Covariance and PCA 3](#_Toc12736901)

[Visualizing PCA Coefficients 3](#_Toc12736902)

[Quiz 3 3](#_Toc12736903)

[Programming Assignment 3 3](#_Toc12736904)

[Vizualizing PCA Residuals I 3](#_Toc12736905)

[Vizualizing PCA Residuals II 3](#_Toc12736906)

[Quiz 4 3](#_Toc12736907)

[Programming Assignment 4 3](#_Toc12736908)

[K-Means and Intrinsic Dimensions 3](#_Toc12736909)

[K-Means Clustering 3](#_Toc12736910)

[Quiz 5 3](#_Toc12736911)

[Intrinsic Dimension 3](#_Toc12736912)

[Quiz 6 3](#_Toc12736913)

[Programming Assignment 5 3](#_Toc12736914)

[Decision Trees, Random Forests and Boosting 3](#_Toc12736915)

[Decision Trees 3](#_Toc12736916)

[Quiz 7 3](#_Toc12736917)

[Programming Assignmnet 6 3](#_Toc12736918)

[Boosting 3](#_Toc12736919)

[Ensembles 3](#_Toc12736920)

[Real-World Application for PCA and Boosting 3](#_Toc12736921)

[Quiz 8 3](#_Toc12736922)

[Appendix 1 Jupyter Nbextensions Configurator 4](#_Toc12736923)

[Usage 4](#_Toc12736924)

[base\_url 4](#_Toc12736925)

[tree tab 4](#_Toc12736926)

[edit menu item 4](#_Toc12736927)

[YAML file format 4](#_Toc12736928)

[APPENDIX 2 Spark Install Instructions - Windows 5](#_Toc12736929)

[Install and Setup 5](#_Toc12736930)

[First Spark Application 5](#_Toc12736931)

[**Next Steps** 6](#_Toc12736932)

[**References** 6](#_Toc12736933)

[APPENDIX 3 Gow - The lightweight alternative to Cygwin 6](#_Toc12736934)

[Introduction 6](#_Toc12736935)

[Features and Benefits 6](#_Toc12736936)

[Win32 Utilities Overview 6](#_Toc12736937)

Big Data Analytics using SPARK

Yoav Freund, Professor Computer Science & Engineering UC San Diego

[**LINK**](https://courses.edx.org/courses/course-v1:UCSanDiegoX+DSE230x+1T2019a/course/#block-v1:UCSanDiegoX+DSE230x+1T2019a+type@chapter+block@b341cd4498054fa089cc99dcadd5875a)

# WEEK 1 Welcome

Welcome to Big Data Analytics using Spark! This course teaches you how to perform statistical analysis of very large datasets that do not fit on a single computer. You will learn some of the most popular tools for performing this type of analysis: apache spark, XGBoost and TensorFlow. You will learn how to use these tools through Jupyter Notebooks and experience the power of combining narrative, code and graphics to create convincing analytical documents.

* Memory Hierarchy, Latency vs. Throughput
* Spark Basics
* Dataframes and SQL
* PCA and Weather Analysis
* K-Means and Intrinsic Dimensions
* Neural Networks and TensorFlow.

## Software Installation Directions

### Installation

* This class will use python3
* The easiest way to install everything is to use the Docker instance that we provide. If for whatever reason, you cannot install Docker then we also provide instructions to download and install the required Python implementation and associated libaries.
* If you're a verified learner, then it's up to you to decide if you want to have a local development environment. We will provide you with a cloud hosted jupyter notebook interface for some of the Programming Assignments where you can code directly. We still recommend verified learners to setup the local environment anyhow.
* If you're auditing the course, then you should follow the instructions below. We highly recommend setting up using docker.

### Setup Using Docker

The purpose of this part is to ensure you have a working and compatible Python and PySpark installation. In order to avoid potential compatibility issues generated from students using different versions than the expected, we provide a Docker image with Ubuntu 16.04 and a clean Anaconda 4.3 with python 3.6, jupyter 5.4, spark 2.2 installation. We also provide a script to run the docker image and get Jupyter running on it so that you can program on it directly. In this guide, we provide instructions on how to install Docker and pull the Docker image. In case you are not able to use Docker, you will have to install Python and Pyspark manually.

* Using the provided Docker container requires installing Docker, 6-7 GB of free space and root access on the host machine (admin rights for windows).
* If you are not able to use the provided container, you can install Python and Pyspark on your own. Make sure you follow the instructions given below. We will expect your results to match ours.
* Docker containers are not intended to store data. We highly recommend you develop your solutions locally and only use docker to compile and run. The following guides show you how to do that. Obtained results should be stored locally as well. If you develop within the container you are at risk of losing your work. You have been warned.
* When you work within the provided container (interactively or not) you are automatically logged in as a user named ucsddse230. Your homework notebook is mounted in the directory /home/ucsddse230/work. If you delete the mounted directory containing your work it will be deleted from the host system. Make sure your work is secure at all times. We recommend you use some sort of version control such as git.

### Installing Docker

Installing Docker should be straightforward for Windows and Mac OS users. Mac and Windows users can download it from from the [Docker website](https://store.docker.com/search?type=edition&offering=community). Linux users will have to use this [guide](https://docs.docker.com/install/linux/docker-ce/ubuntu/). The linux guides essentially try to upgrade your system to a compatible version (for example upgrading to Ubuntu 16.04). Be careful not to break your current system. If you are working with linux, having a Ubuntu 16.04 system should result in an easier docker installation. For Windows users, Docker will require you enable Hyper-V and restart your computer. Some Windows 10 versions do not have Hyper-V. If you face any issues with installing Docker on Windows, installing Docker Toolbox instead of Docker should be the easiest way out.

### Pulling the Docker image

After successful installation of Docker, open a command prompt or shell and execute the following command (Windows users should skip the “sudo” part): Linux/Mac:

$ sudo docker pull ucsddse230/cse255-dse230

Windows (Powershell prefered):

$ docker pull ucsddse230/cse255-dse230

Docker should automatically start downloading and extracting the provided image. If you skip this step the image will automatically be downloaded the first time you attempt to start it. Once finished you can verify you have it by typing Linux/Mac:

$ sudo docker images

Windows:

$ docker images

Students using Docker Toolbox for their Windows OS that does not support Hyper-V would now need to execute an additional command to identify their Docker IP address.

$ docker-machine ip

Note down the IP address that is returned as the output. You will need to use this in the next section.

### Running Docker Images

Next, download the required content (for example, Programming assignment or Section Notebooks) files from EdX directory to some location on your computer. ex: /local/path/to/pa1. An an illustration, let's use Programming Assignment 1. You may open the [Programming Assignment 1 section](https://courses.edx.org/courses/course-v1:UCSanDiegoX+DSE230x+1T2019a/jump_to_id/2cb4714117c04c50a41e5d97b3af2e0c) on EdX which has the link to the necessary started code. Let's say after unpacking the files pa1 files are present in  /local/path/to/pa1

NOTE: This path should be the absolute path.

Then run the following following line of code in your terminal (first time might take a while).

$ docker run -it -p 8889:8888 -v /local/path/to/pa1:/home/ucsddse230/work ucsddse230/cse255-dse230 /bin/bash

This command will:

1. Start the docker container

2. mount the local directory "/local/path/to/pa1" inside the container at the location "/home/ucsddse230/work"

3. Forward requests to port 8889 on the local system from port 8888 inside the docker container.

Notice the terminal has changed, you are now inside a virtual machine. Run the following commands to start jupyter at http://localhost:8889 by issuing the command

$ jupyter notebook

This will start jupyter at port 8888 inside the docker container, which will be accessible outside the docker at port 8889.

Now you can view notebooks and work on homework at the localhost:8889 port. Go ahead open your web browser and put "localhost:8889" in the address bar. You should now be able to see the Jupyter Notebook webpage.

Students using Docker Toolbox can access the Jupyter notebook running in their Docker container at the DockerIP:8888 port, where DockerIP is the IP address returned in the previous section.

Whatever changes you make will also happen to /local/path/to/pa1.

### Setup From Scratch

First install the Python 3.6 version using the [anaconda](https://www.continuum.io/Downloads) distribution.

#### Install jupyter

If you install Anaconda, jupyter and almost all the necessary packages are installed for you.

#### Install notebook extensions

This step is not required, but extensions can make your work on notebooks significantly easier. To install a bunch of useful extensions, together with a configurator for managing thses extensions, follow the directions on:

<https://github.com/Jupyter-contrib/jupyter_nbextensions_configurator>

#### Install python packages

Make sure to install the python package [findspark](https://github.com/minrk/findspark). The typing the following command in the terminal installs the package:

Anaconda: conda install -c conda-forge findspark=1.0.0[[1]](#footnote-1)

pip: sudo pip install findspark

If you are using pip instead of anaconda, you also must install the following packages:

* numpy
* matplotlib
* pandas
* Some notebooks require additional packages, or packages of a later version. If an import command in a notebook fails, use pip or conda to install the missing package.

Install Spark on your computer

* [Install on Linux or Mac OS X](https://mas-dse.github.io/DSE230/installation/linux/)
* [Install on Windows](https://mas-dse.github.io/DSE230/installation/windows/)

# Map-Reduce and Spark

## The Memory Hierarchy

## Spark Basics

## Quiz 1

## Programming Assignment 1

## Pyspark and RDDs

## Spark SQL and DataFrames

## Preparing for Data Analysis

## Quiz 2

## Programming Assignment 2

# PCA and Weather Analysis

## Covariance and PCA

## Visualizing PCA Coefficients

## Quiz 3

## Programming Assignment 3

## Vizualizing PCA Residuals I

## Vizualizing PCA Residuals II

## Quiz 4

## Programming Assignment 4

# K-Means and Intrinsic Dimensions

## K-Means Clustering

## Quiz 5

## Intrinsic Dimension

## Quiz 6

## Programming Assignment 5

## Decision Trees, Random Forests and Boosting

## Decision Trees

## Quiz 7

## Programming Assignmnet 6

## Boosting

## Ensembles

## Real-World Application for PCA and Boosting

## Quiz 8

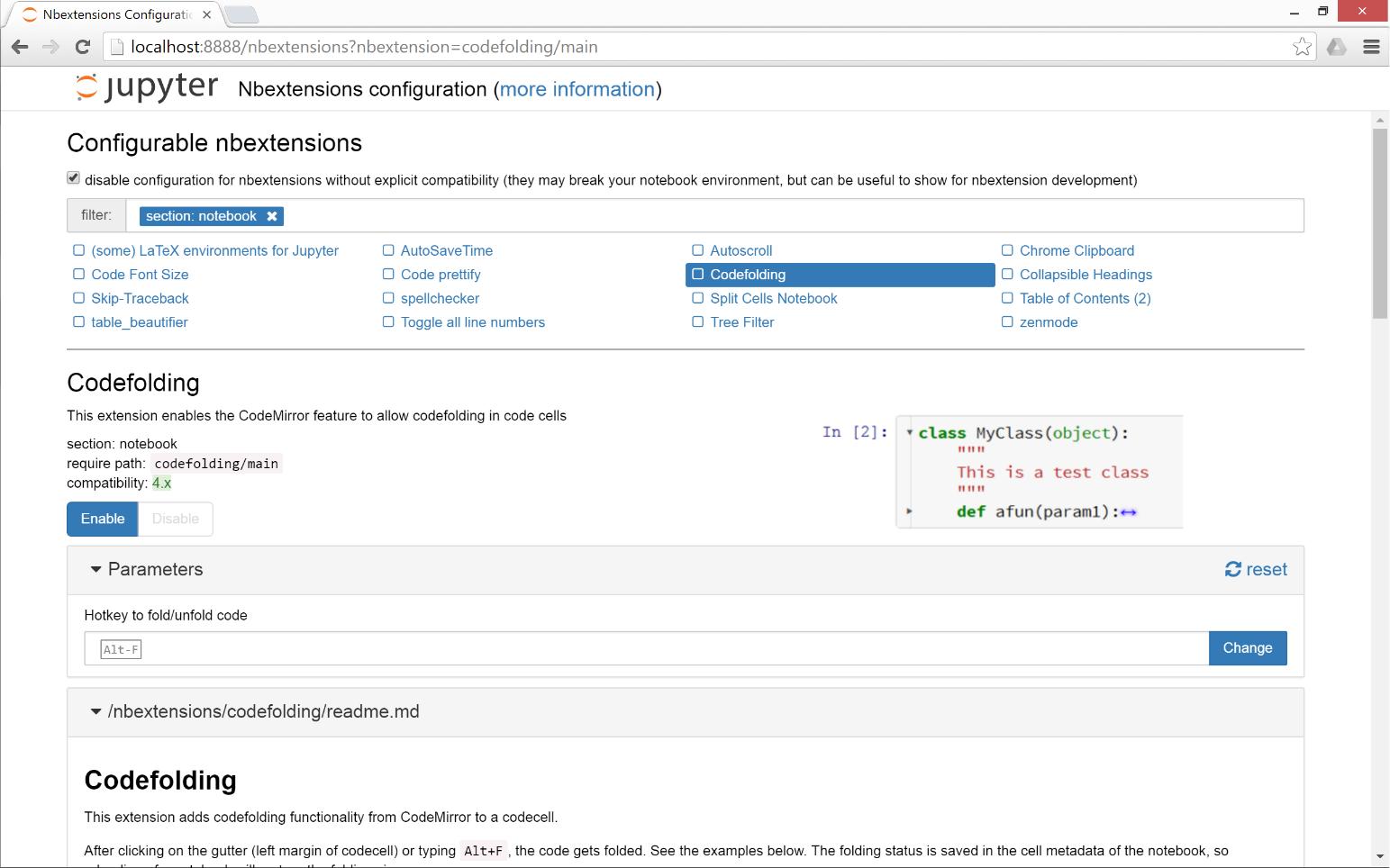
# Appendix 1 Jupyter Nbextensions Configurator

[**LINK**](https://github.com/Jupyter-contrib/jupyter_nbextensions_configurator)

A server extension for jupyter notebook which provides configuration interfaces for notebook extensions (nbextensions). The jupyter\_nbextensions\_configurator jupyter server extension provides graphical user interfaces for configuring which nbextensions are enabled (load automatically for every notebook). In addition, for nbextensions which include an appropriate yaml descriptor file (see below), the interface also renders their markdown readme files, and provides controls to configure the nbextensions' options.

## Usage

Once jupyter\_nbextensions\_configurator is installed and enabled, and your notebook server has been restarted, you should be able to find the nbextensions configuration interface at the url <base\_url>nbextensions, where <base\_url> is described below (for simple installs, it's usually just /, so the UI is at /nbextensions).

[](https://github.com/Jupyter-contrib/jupyter_nbextensions_configurator/blob/master/src/jupyter_nbextensions_configurator/static/nbextensions_configurator/icon.png)

### base\_url

For most single-user notebook servers, the dashboard (the file-browser or 'tree' view) is at

http://localhost:8888/tree

So the base\_url is the part between the host (http://localhost:8888) and tree, so in this case it's the default value of just /. If you have a non-default base url (such as with JupyterHub), you'll need to prepend it to the url. So, if your dashboard is at

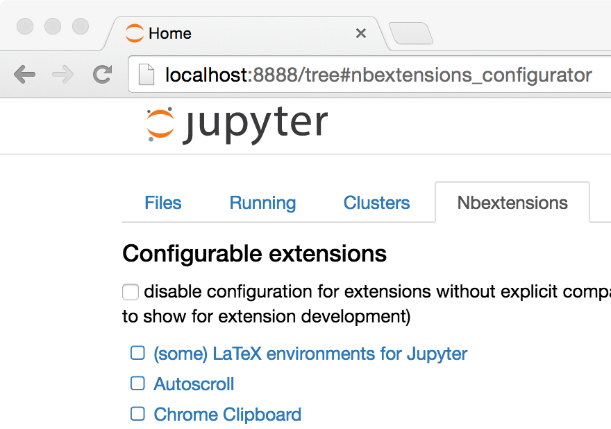
http://localhost:8888/custom/base/url/tree

then you'll find the configurator UI page at

http://localhost:8888/custom/base/url/nbextensions

### tree tab

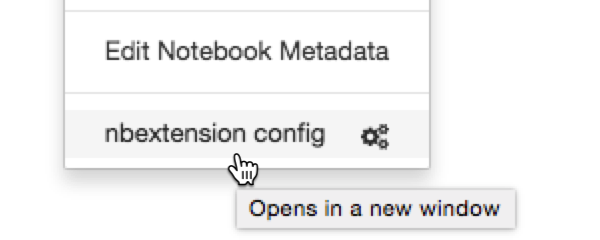
In addition to the main standalone page, the nbextensions configurator interface is also available as a tab on the dashboard:

[](https://github.com/Jupyter-contrib/jupyter_nbextensions_configurator/blob/master/src/jupyter_nbextensions_configurator/static/nbextensions_configurator/tree_tab/icon.png)

The dashboard tab is provided via an nbextension called "Nbextensions dashboard tab", with requirejs urinbextensions\_configurator/tree\_tab/main. Since version [0.2.0](https://github.com/Jupyter-contrib/jupyter_nbextensions_configurator#020), this nbextension is enabled by default on enabling the jupyter\_nbextensions\_configurator serverextension, but it can be disabled as with any other nbextension if you don't want to use it.

### edit menu item

jupyter\_nbextensions\_configurator provides a second small nbextension, which simply adds an item to the notebook-view edit menu, which links to the configurator UI page:

[](https://github.com/Jupyter-contrib/jupyter_nbextensions_configurator/blob/master/src/jupyter_nbextensions_configurator/static/nbextensions_configurator/config_menu/icon.png)

Similarly to the tree tab nbextension detailed above, since version [0.2.0](https://github.com/Jupyter-contrib/jupyter_nbextensions_configurator#020), the edit menu item nbextension is enabled by default when enabling the main jupyter\_nbextensions\_configurator serverextension, but can be disabled at any time in the same way as other nbextensions.

## YAML file format

You don't need to know about the yaml files in order simply to use jupyter\_nbextensions\_configurator. An nbextension is 'found' by the jupyter\_nbextensions\_configurator server extension when a special yaml file describing the nbextension and its options is found in the notebook server's nbextensions\_path. The yaml file can have any name with the file extension .yaml or .yml, and describes the nbextension and its options to jupyter\_nbextensions\_configurator.

The case-sensitive keys in the yaml file are as follows:

* Type, (**required**) a case-sensitive identifier, must be IPython Notebook Extension or Jupyter Notebook Extension
* Main, (**required**) the main javascript file that is loaded, typically main.js
* Name, the name of the nbextension
* Section, which view the nbextension should be loaded in (defaults to notebook, but can alternatively be tree, edit, or to load in all views, common).
* Description, a short explanation of the nbextension
* Link, a URL for more documentation. If this is a relative url with a .md file extension (recommended!), the markdown readme is rendered in the configurator UI.
* Icon, a URL for a small icon for the configurator UI (rendered 120px high, should preferably end up 400px wide. Recall HDPI displays may benefit from a 2x resolution icon).
* Compatibility, Jupyter major version compatibility, e.g. 3.x or 4.x, 3.x 4.x, 3.x, 4.x, 5.x
* Parameters, an optional list of configuration parameters. Each item is a dictionary with (some of) the following keys
  + name, (**required**) the name used to store the configuration variable in the config json. It follows a json-like structure, so you can use . to separate sub-objects e.g. myextension.buttons\_to\_add.play.
  + description, a description of the configuration parameter
  + default, a default value used to populate the tag in the configurator UI, if no value is found in config. Note that this is more of a hint to the user than anything functional - since it's only set in the yaml file, the javascript implementing the nbextension in question might actually use a different default, depending on the implementation.
  + input\_type, controls the type of html tag used to render the parameter in the configurator UI. Valid values include text, textarea, checkbox, [html5 input tags such as number, url, color, ...], plus a final type of list
  + list\_element, a dictionary with the same default and input\_type keys as a Parameters entry, used to render each element of the list for parameters with input\_type list
  + finally, extras such as min, step and max may be used by number tags for validation
* tags, a list of string tags describing the nbextension, to allow for filtering

**Example:**

Type: Jupyter Notebook Extension

Name: Limit Output

Section: notebook

Description: This nbextension limits the number of characters that can be printed below a codecell

tags:

- usability

- limit

- output

Link: readme.md

Icon: icon.png

Main: main.js

Compatibility: 4.x

Parameters:

- name: limit\_output

description: Number of characters to limit output to

input\_type: number

default: 10000

step: 1

min: 0

- name: limit\_output\_message

description: Message to append when output is limited

input\_type: text

default: '\*\*OUTPUT MUTED\*\*'

# APPENDIX 2 Spark Install Instructions - Windows

Instructions tested with Windows 10 64-bit. It is highly recommend that you use Mac OS X or Linux for this course, these instructions are only for people who cannot run Mac OS X or Linux on their computer.

## Install and Setup

Spark provides APIs in Scala, Java, Python (PySpark) and R. We use PySpark and Jupyter, previously known as IPython Notebook, as the development environment. There are many articles online that talk about Jupyter and what a great tool it is, so we won’t introduce it in details here.

This Guide Assumes you already have Anaconda and Gnu On Windows installed. See <https://mas-dse.github.io/startup/anaconda-windows-install/>

1. Go to [http://www.java.com](http://www.java.com/) and install Java 7+.

2. Get Spark pre-built package from the [downloads page](http://spark.apache.org/downloads.html) of the Spark project website.

3. Open PowerShell by pressing ⊞ Win-R, typing “powershell” in Run dialog box and clicking “OK”. Change your working directory to where you downloaded the Spark package.

4. Type the commands in red to uncompress the Spark download. Alternatively, you can use any other software of your preference to uncompress.

> **gzip -d spark-2.1.0-bin-hadoop2.7.tgz**  
> **tar xvf spark-2.1.0-bin-hadoop2.7.tar**

5. Type the commands in red to move Spark to the c:\opt\spark\ directory.

> **mkdir C:\opt\**  
> **move spark-2.1.0-bin-hadoop2.7 C:\opt\spark\**

6. Type the commands in red to download winutils.exe for Spark.

> **cd C:\opt\spark\bin\**  
> **curl -k -L -o winutils.exe https://github.com/steveloughran/winutils/blob/master/hadoop-2.6.0/bin/winutils.exe?raw=true**

7. Create an environment variable with variable name = SPARK\_HOME and variable value = C:/opt/spark. This link provides a good description of [how to set environment variable in windows](http://www.forbeslindesay.co.uk/post/42833119552/permanently-set-environment-variables-on-windows)

8. Type the commands in red to create a temporary directory.

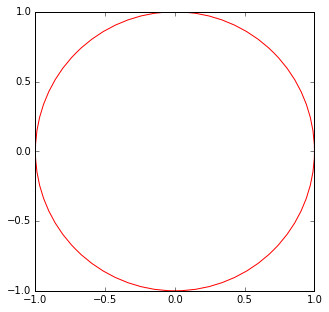
> **mkdir ~/Documents/jupyter-temp/**  
> **cd ~/Documents/jupyter-temp/**

9. Type the commands in red to install, configure and run Jupyter Notebook. Jupyter Notebook will launch using your default web browser.

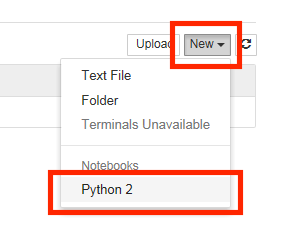
> **conda install jupyter -y**  
> **ipython kernelspec install-self**  
> **jupyter notebook**

## First Spark Application

In our first Spark application, we will run a Monte Carlo experiment to find an estimate for $\pi$. Here is how we are going to do it. The figure bellow shows a circle with radius $r = 1$ inscribed within a 2×2 square. The ratio between the area of the circle and the area of the square is $\frac{\pi}{4}$. If we sample enough points in the square, we will have approximately $\rho = \frac{\pi}{4}$ of these points that lie inside the circle. So we can estimate $\pi$ as $4 \rho$.



1. Create a new Notebook by selecting **Python 2** from the **New** drop down list at the right of the page.



2. First we will create the Spark Context. Copy and paste the red text into the first cell then click the  (run cell) button:

**import os**  
**import sys**  
  
**import findspark**  
**findspark.init()**  
  
**from pyspark import SparkContext**  
  
**sc = SparkContext(master="local[4]")**

3. Next, we draw a sufficient amount of points inside the square. Copy and paste the red text into the next cell then click the  (run cell) button:

**import numpy as np**  
  
**TOTAL = 1000000**  
**dots = sc.parallelize([2.0 \* np.random.random(2) - 1.0 for i in range(TOTAL)]).cache()**  
**print("Number of random points:", dots.count())**  
  
**stats = dots.stats()**  
**print('Mean:', stats.mean())**  
**print('stdev:', stats.stdev())**

Output:

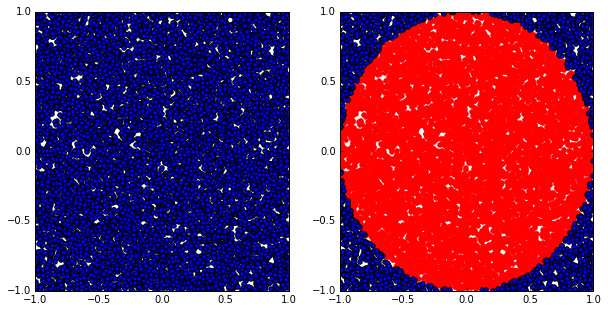
('Number of random points:', 1000000)  
('Mean:', array([-0.0004401 , 0.00052725]))  
('stdev:', array([ 0.57720696, 0.57773085]))

4. We can sample a small fraction of these points and visualize them. Copy and paste the red text into the next cell then click the  (run cell) button:

**%matplotlib inline**  
**from operator import itemgetter**  
**from matplotlib import pyplot as plt**  
  
**plt.figure(figsize = (10, 5))**  
  
**# Plot 1**  
**plt.subplot(1, 2, 1)**  
**plt.xlim((-1.0, 1.0))**  
**plt.ylim((-1.0, 1.0))**  
  
**sample = dots.sample(False, 0.01)**  
**X = sample.map(itemgetter(0)).collect()**  
**Y = sample.map(itemgetter(1)).collect()**  
**plt.scatter(X, Y)**  
  
**# Plot 2**  
**plt.subplot(1, 2, 2)**  
**plt.xlim((-1.0, 1.0))**  
**plt.ylim((-1.0, 1.0))**  
  
**inCircle = lambda v: np.linalg.norm(v) <= 1.0**  
**dotsIn = sample.filter(inCircle).cache()**  
**dotsOut = sample.filter(lambda v: not inCircle(v)).cache()**  
  
**# inside circle**  
**Xin = dotsIn.map(itemgetter(0)).collect()**  
**Yin = dotsIn.map(itemgetter(1)).collect()**  
**plt.scatter(Xin, Yin, color = 'r')**  
  
**# outside circle**  
**Xout = dotsOut.map(itemgetter(0)).collect()**  
**Yout = dotsOut.map(itemgetter(1)).collect()**  
**plt.scatter(Xout, Yout)**

Output:

﻿<matplotlib.collections.PathCollection at 0x17a78780>



5. Finally, let’s compute the estimated value of $\pi$. Copy and paste the red text into the next cell then click the  (run cell) button:

**pi = 4.0 \* (dots.filter(inCircle).count() / float(TOTAL))**  
**print("The estimation of \pi is:", pi)**

Output:

('The estimation of \\pi is:', 3.142204)

**Next Steps**

* [Spark Programming Guide](http://spark.apache.org/docs/latest/programming-guide.html)
* [Example Spark Programs](http://spark.apache.org/examples.html)

**References**

* [Spark official documents](http://spark.apache.org/docs/latest/)
* Example Python Spark programs [on the Spark Github repository](https://github.com/apache/spark/tree/master/examples/src/main/python)

# APPENDIX 3 Gow - The lightweight alternative to Cygwin

## Introduction

Gow (Gnu On Windows) is the lightweight alternative to Cygwin. It uses a convenient [NSIS installer](http://bit.ly/cop1wN)that installs over 100 extremely useful [open source UNIX applications](https://github.com/bmatzelle/gow/wiki/executables_list) compiled as native win32 binaries. It is designed to be as small as possible, about 18 MB, as opposed to Cygwin which can run well over 100 MB depending upon options.

Here are a couple quotes from happy Gow users:

"Gow is one of the few things that makes Windows bearable/usable"

"I use Gow constantly. It's awesome."

"I just wanted to let you know that the GOW Suite is simply great - it is far lighter than the Cygwin tool, and is extremely useful. "

## Features and Benefits

* *Ultra light*: Small, light subset (about 18 MB) of very [useful UNIX binaries](https://github.com/bmatzelle/gow/wiki/executables_list) that do not have decent installers (until now!).
* *Shell window from any directory*: Adds a Windows Explorer shell window ([screenshot](http://bit.ly/9Pk1us)) so that you can right-click on any directory and open a command (cmd.exe) window from that directory.
* *Simple install/remove*: Easy to install and remove, all files contained in a single directory in a standard C:\Program Files path.
* *Included in PATH*: All binaries are conveniently installed into the Windows PATH so they are accessible from a command-line window.
* *Stable binaries*: All commands are stable and tested.

## Win32 Utilities Overview

Below are just a few of the [100+ applications](https://github.com/bmatzelle/gow/wiki/executables_list) found in Gow.

* *Shell scripting*: bash, zsh
* *Compression*: gzip, zip, bzip2, compress
* *SSH*: putty, psftp, pscp, pageant, plink
* *Download/upload*: cURL, wget
* *FTP*: NcFTP
* *Editing*: vim, nano
* *Text search/view*: grep, agrep, less, cat, tail, head
* *File system*: mv, cp, du, ls, pwd, rmdir, whereis
* *Development*: make, diff, diff3, sleep, cvs, dos2unix, unix2dos

1. Find pyspark to make it importable. [↑](#footnote-ref-1)