PYTHON
for
DATA SCIENCE

MEET YOUR INSTRUCTOR

Hi!

I am Dushyant Khosla

Currently doing Data Science things at limeroad.com

- R and Python SME
- Advanced Analytics Evangelist
- 6 years in Data and Decision Sciences
- Previously: AbsolutData, Dell, @WalmartLabs
- Bachelors in Engineering, Masters in Analytics
- Enterprise Trainer on Statistics and Machine Learning

WHAT IS DATA SCIENCE?

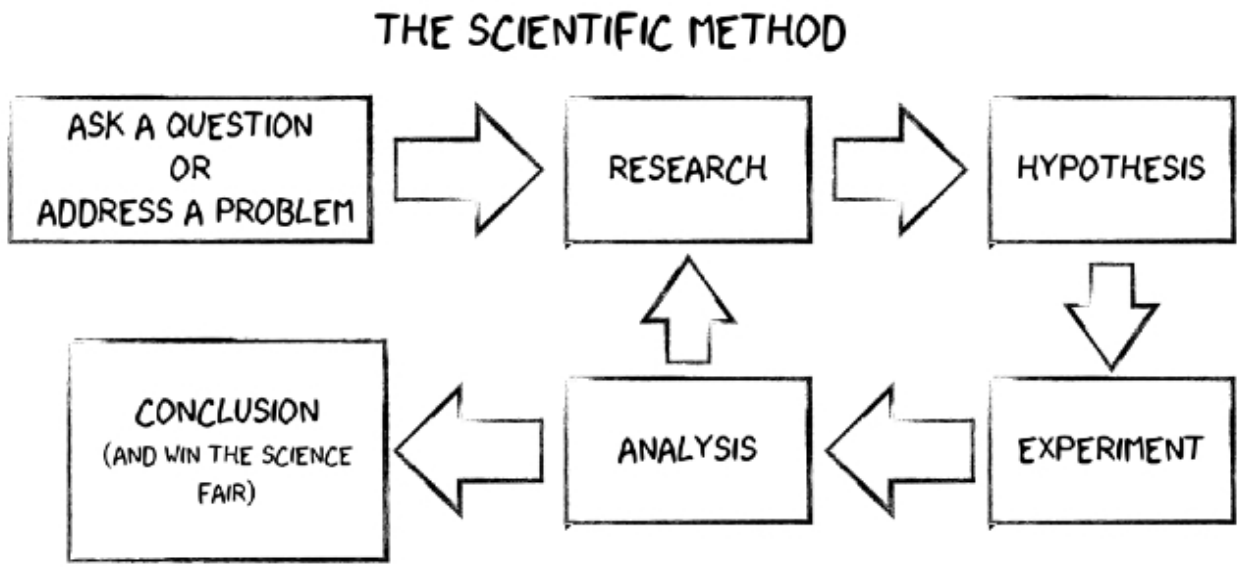
> *"To gain insights into data through computation, statistics, and visualization."*

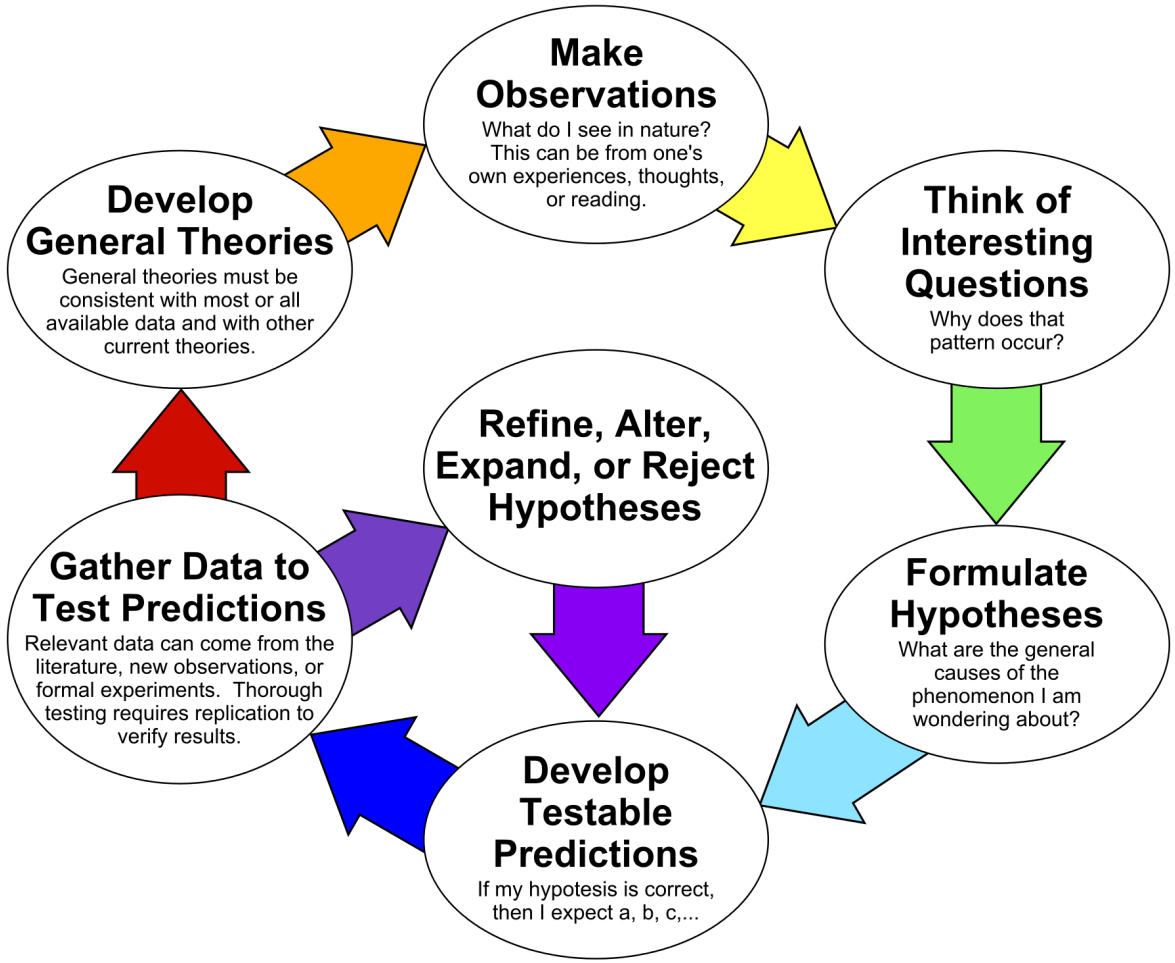
Quora Threads for Expert Definitions

- What is Data Science?
- What does a Data Scientist do?

DS IS MULTIDISCIPLINARY

- The Scientific Method (wiki)
- Programming
- Databases
- Statistics
- Machine Learning
- Domain Knowledge




Make Observations
What do I see in nature? This can be from one's own experiences, thoughts, or reading.
Think of Interesting Questions
Why does that pattern occur?
Formulate Hypotheses
What are the general causes of the phenomenon I am wondering about?
Develop Testable Predictions
If my hypotesis is correct, then I expect a, b, c,...
Gather Data to Test Predictions
Relevant data can come from the literature, new observations, or formal experiments. Thorough testing requires replication to verify results.
Develop General Theories
General theories must be consistent with most or all available data and with other current theories.
Refine, Alter, Expand, or Reject Hypotheses


SCIENTIFIC METHOD
PURPOSE
State the problem.
RESEARCH
Find out about the topic.
HYPOTHESIS
Predict the outcome to the problem.
EXPERIMENT
Develop a procedure to test the hypothesis.
ANALYSIS
Record the results of the experiment.
CONCLUSION
Compare the hypothesis to the experiment's conclusion.

Science Paradigms

- Thousand years ago:
  science was **empirical**
  describing natural phenomena
- Last few hundred years:
  theoretical branch
  using models, generalizations
- Last few decades:
  a **computational** branch
  simulating complex phenomena
- Today: **data exploration** (eScience)
  unify theory, experiment, and simulation
  - Data captured by instruments or generated by simulator
  - Processed by software
  - Information/knowledge stored in computer
  - Scientist analyzes database/files using data management and statistics

$$\left(\frac{\dot{a}}{a}\right)^2 = \frac{4\pi Gp}{3} - \mathrm{K}\frac{c^2}{a^2}$$

WHY DATA SCIENCE?

The ability to take **data** – to be able to **understand** it, to **process** it, to **extract value** from it, to **visualize** it, to **communicate** it's going to be a hugely important skill in the next decades, not only at the professional level but even at the educational level for elementary school kids, for high school kids, for college kids. Because now we really do have essentially free and **ubiquitous data**."

– Hal Varian

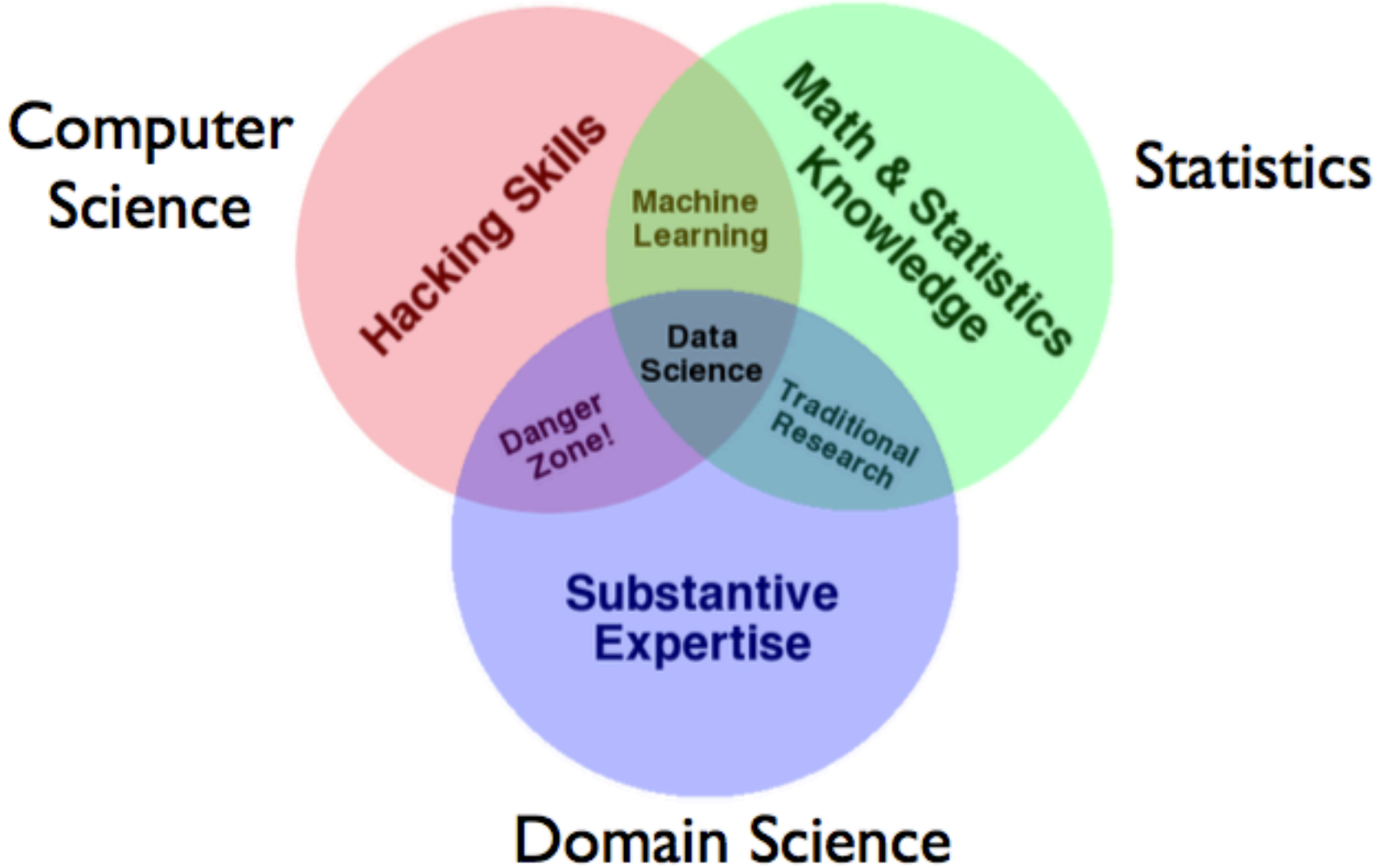
WHO'S A DATA SCIENTIST

"A data scientist is someone who knows more statistics than a computer scientist and more computer science than a statistician."

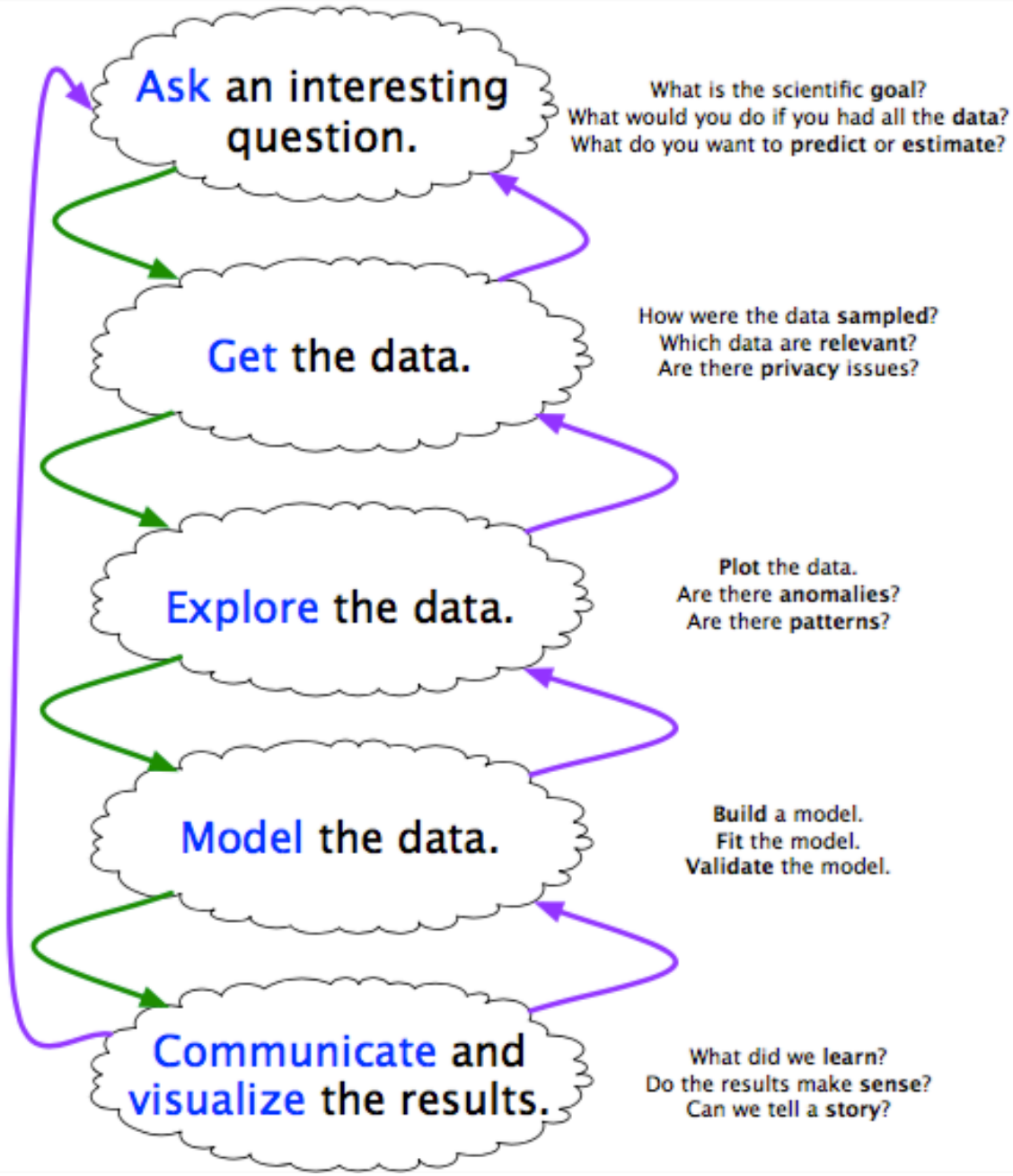
- Josh Blumenstock

"Data Scientist = statistician + programmer + coach + storyteller + artist"

- Shlomo Aragmon


Computer Science
Hacking Skills
Machine Learning
Math & Statistics Knowledge
Statistics
Data Science
Danger Zone!
Traditional Research
Substantive Expertise
Domain Science

Drew Conway


Ask an interesting question.
What is the scientific goal?
What would you do if you had all the data?
What do you want to predict or estimate?
Get the data.
How were the data sampled?
Which data are relevant?
Are there privacy issues?
Explore the data.
Plot the data.
Are there anomalies?
Are there patterns?
Model the data.
Build a model.
Fit the model.
Validate the model.
Communicate and visualize the results.
What did we learn?
Do the results make sense?
Can we tell a story?


1. Fundamentals
Matrices & Linear Algebra Fundamentals
Hash Functions, Binary Tree, O(n)
Relational Algebra, DB Basics
Inner, Outer, Cross, Theta Join
CAP Theorem
Tabular Data
Data Frames & Series
Entropy
Sharding
OLAP
Multidimensional Data Model
ETL
Reporting Vs BI Vs Analytics
JSON & XML
NoSQL
Regex
Vendor Landscape
Env Setup
5%
2. Statistics
Prob Den Fn (PDF)
ANOVA
Skewness
Continuos Distributions (Normal, Poisson, Gaussian)
Cumul Dist Fn (CDF)
Random Variables
Bayes Theorem
Probability Theory
Percentiles & Outliers
Histograms
Exploratory Data Analysis
Central Limit Theorem
Monte Carlo Method
Hypothesis Testing
p-Value
Chi² Test
Estimation
Confid Int (CI)
MLE
Kernel Density Estimate
Regression
Covariance
Correlation
15%
Factor Analysis
Install Pkgs
Descriptive Statistics (mean, median, range, SD, Var)
Pick a Dataset (UCI Repo)
3. Programming
Data Frames
Reading CSV Data
Reading Raw Data
Subsetting Data
Manipulate Data Frames
Functions
Lists
Factors
Arrays
Matrices
Vectors
Variables
Expressions
R Basics
R Setup R Studio
Python Basics
Working in Excel
Rapid Miner
IBM SPSS
4. Machine Learning
Regression
Perceptron
Linear Regression
Ranking
Logistic Regression
Classification
K-Nearest Neighbor
Naïve Bayes Classifiers
Boosting
Decision Trees
Classification Rate
Trees & Classification
Bias & Variance
Overfitting
Lift
Prediction
Classifier
Training & Test Data
Concepts, Inputs & Attributes
Unsupervised Learning
Supervised Learning
Categorical Var
Numerical Var
What is ML ?
30%
Euclidean Distance
Least² Fit
Causation
Pearson Coeff
Correlation
Clustering
Hierarchical Clustering
K-means Clustering
Neural Networks
Sentiment Analysis
Collaborative Filtering
Tagging
Vocabulary Mapping
Classify Text
50%
Using NLTK
Using Weka
Using Mahout
Feature Extraction
Named Entity Recognition
Text Analysis
UIMA
Term Document Matrix
Term Document Matrix
Term Frequency & Weight
Support Vector Machines
Association Rules
Market Based Analysis
Corpus
5. Text Mining / NLP
6. Visualization
Data Exploration in R (Hist, Boxplot etc)
Uni, Bi & Multivariate Viz
ggplot2
Histogram & Pie (Uni)
Tree & Tree Map
Scatter Plot (Bi)
Line Charts (Bi)
Spatial Charts
Survey Plot
Timeline
Decision Tree
40%
Tableau
IBM ManyEyes
InfoVis
D3.js
7. Big Data
Job & Task Tracker
MR Programming
Sqoop: Loading Data in HDFS
Flume, Scribe : For Unstruct Data
SQL with Pig
DWH with Hive
Scribe, Chukwa For Weblog
Using Mahout
Name & Data Nodes
Setup Hadoop (IBM / Cloudera / HortonWorks)
Data Replication Principles
HDFS
Hadoop Components
Map Reduce Fundamentals
Zookeeper Avro
Storm: Hadoop Realtime
Rhadoop, RHIPE
rmr
Cassandra
MongoDB, Neo4j
60%
8. Data Ingestion
Sampling
Stratified Sampling
Principal Component Analysis
Transformation & Enrichment
Data Fusion
Data Integration
Data Sources & Acquisition
Data Discovery
Summary of Data Formats
Data Survey
Using ETL
How much Data?
Google OpenRefine
80%
9. Data Munging
Denoising
Feature Extraction
Binning Sparse Values
Unbiased Estimators
Handling Missing Values
Data Scrubbing
Normalization
Dimensionality & Numerosity Reduction
10. Toolbox
MS Excel w/ Analysis ToolPak
Java, Python
R, R-Studio, Rattle
Weka, Knime, RapidMiner
Hadoop Dist of Choice
Spark, Storm
Flume, Scibe, Chukwa
Nutch, Talend, Scraperwiki
Webscraper, Flume, Sqoop
tm, RWeka, NLTK
RHIPE
D3.js, ggplot2, Shiny
IBM Languageware
Cassandra, MongoDB
100%
Author: Swami Chandrasekaran

WHAT DOES A DATA SCIENTIST DO?

BUILD DATA PRODUCTS

tools built with data to inform decision making

DESCRIPTIVE
PREDICTIVE
PRESCRIPTIVE

OSEMN Things!

Obtain data

Scrub data

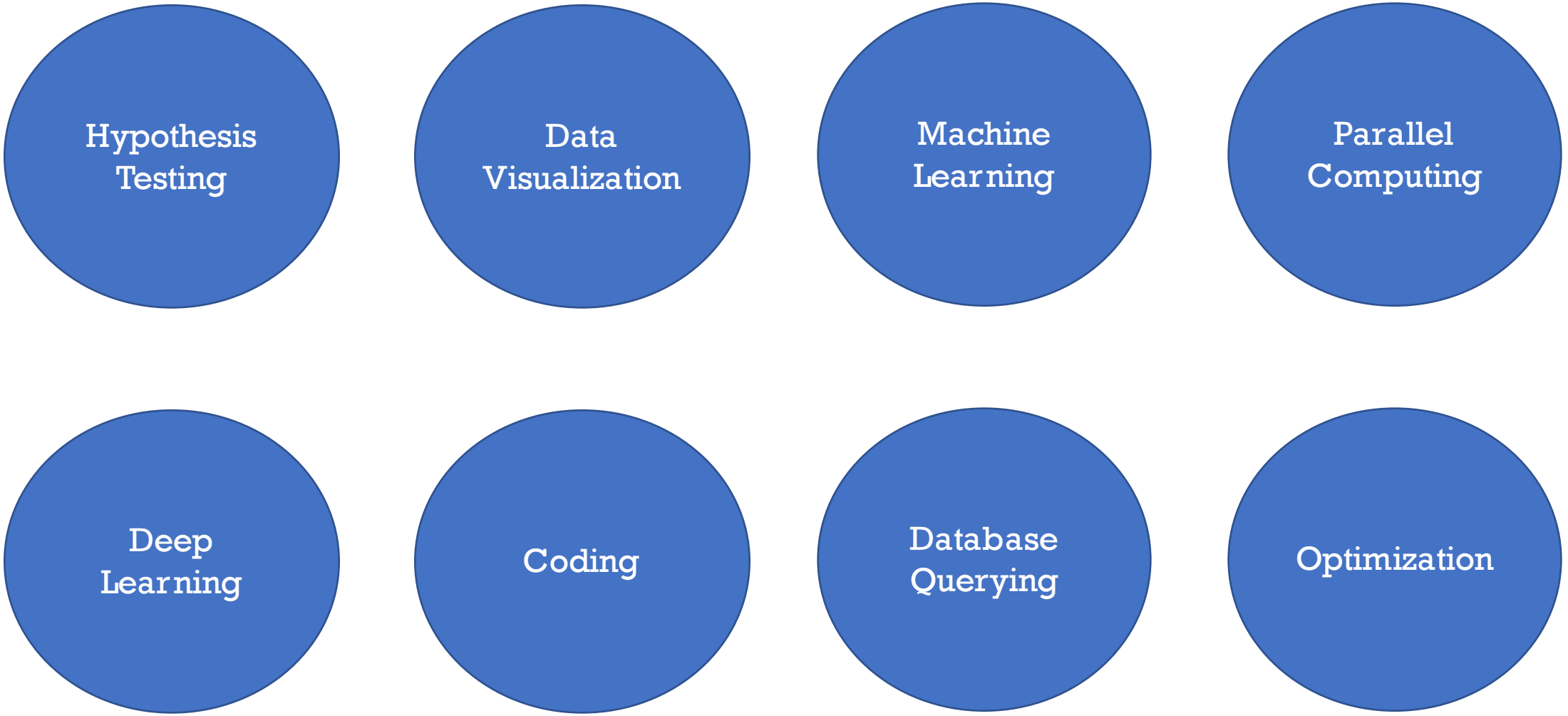
Explore data

Build Models

iNterpret results

Hence the acronym
O-S-E-M-N
(pronounced, 'awesome')

.. AND THIS


Hypothesis Testing
Data Visualization
Machine Learning
Parallel Computing
Deep Learning
Coding
Database Querying
Optimization

WHO DOES DATA SCIENCE?

Google
Facebook
Amazon
Twitter
LinkedIN
Apple
Netflix
GE
Boeing
Walmart
…

Astronomy

Discover new galaxies and better define the properties of observable star clusters.

Biology

Interpret simple relationships in the complex interactions between molecules.

Chemistry

Evaluate hundreds of millions of chemicals for potential pharmaceutical use.

Computer Science

Uncover new techniques to improve and apply current machine learning techniques.

Environmental Science

Support conservation and sustainability efforts by predicting environmental impacts.

Evolutionary Computation

Generate new solutions using evolutionary strategies to solve complex problems.

General

Investigate and predict the growing impact of machines on the human experience.

Material Science

Search for tomorrow's materials by defining properties of unknown materials.

Medical

Develop new models of healthcare and generate personalized prediction models.

Music

Discover new methods of music creation and distribution to keep pace with changing needs.

Neurology

Improve diagnoses of neurological disorders and understand how conditions impact the brain.

Physics/Mechanics/ Biomechanics

Produce accurate predictive simulations and find new opportunities in existing data.

… AND THESE GUYS

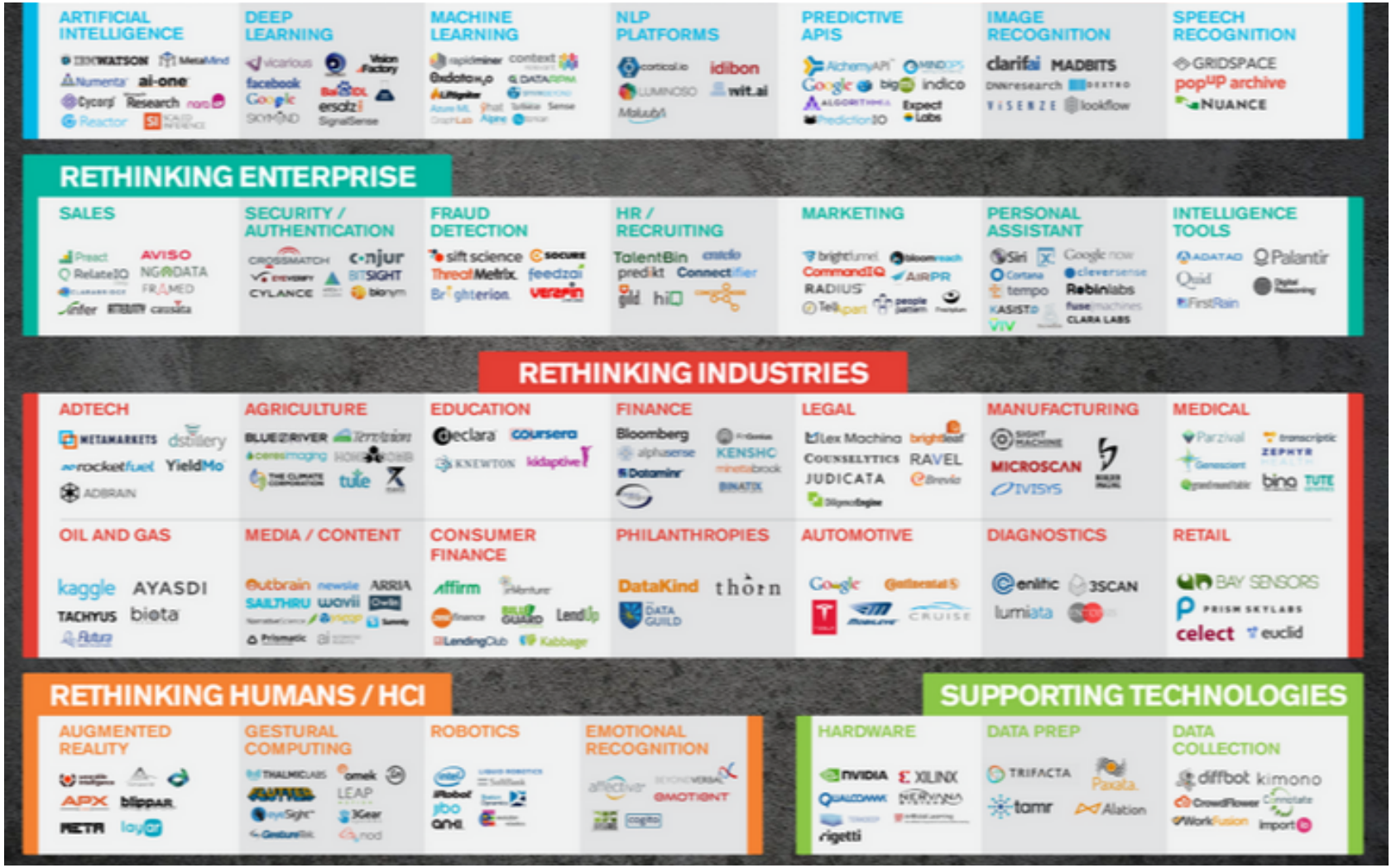
$4.6M AVERAGE VALUATION

[redacted]

700 COMPANIES | 2,974 INVESTORS | 8,582 FOLLOWERS [redacted]

Company	Joined	Followers	Signal
Mattermark Deal Intelligence - Powerful Anal... San Francisco · SaaS	Apr '12	2894	
uBiome Big Data from Bacteria San Francisco · Quantified Self	Mar '13	2520	
Kaggle World's largest data science com... San Francisco · Data Mining	May '11	1373	
Vurb Mobile search reinvented - find, s... San Francisco · Disruptive Models	Oct '11	873	
Granify Granify knows which shoppers ar...	Jun '11	626	

… AND THESE GUYS



KEY CONCEPTS

- *use many data sources*
- *understand how the data were collected* (sampling is essential)
- *weight the data thoughtfully* (not all polls are equally good)
- *use statistical models* (not just hacking around in Excel)
- *understand correlations* (e.g., states that trend similarly)
- *think like a Bayesian, check like a frequentist* (reconciliation)
- *have good communication skills* (What does a 60% probability even mean?
- *visualize, validate, and understand the conclusions*

COMMON CHALLENGES

- *massive data* (millions of users, billions of events)
- *curse of dimensionality* (hundreds of variables)
- *missing data* (*not* missing at random)
- *need to avoid overfitting* (test data vs. training data)

COMMON TASKS

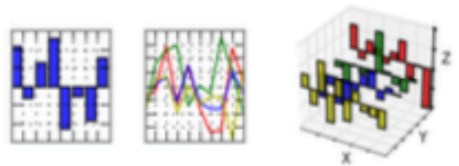
- *data munging/scraping/sampling/cleaning in order to get an informative, manageable data set;*
- *data storage and management in order to be able to access data quickly and reliably during subsequent analysis;*
- *exploratory data analysis to generate hypotheses and intuition about the data;*
- *prediction based on statistical tools such as regression, classification, and clustering; and*
- *communication of results through visualization, stories, and interpretable summaries.*

TOOLS OF THE TRADE

IP[y]: IPython
Interactive Computing

pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$



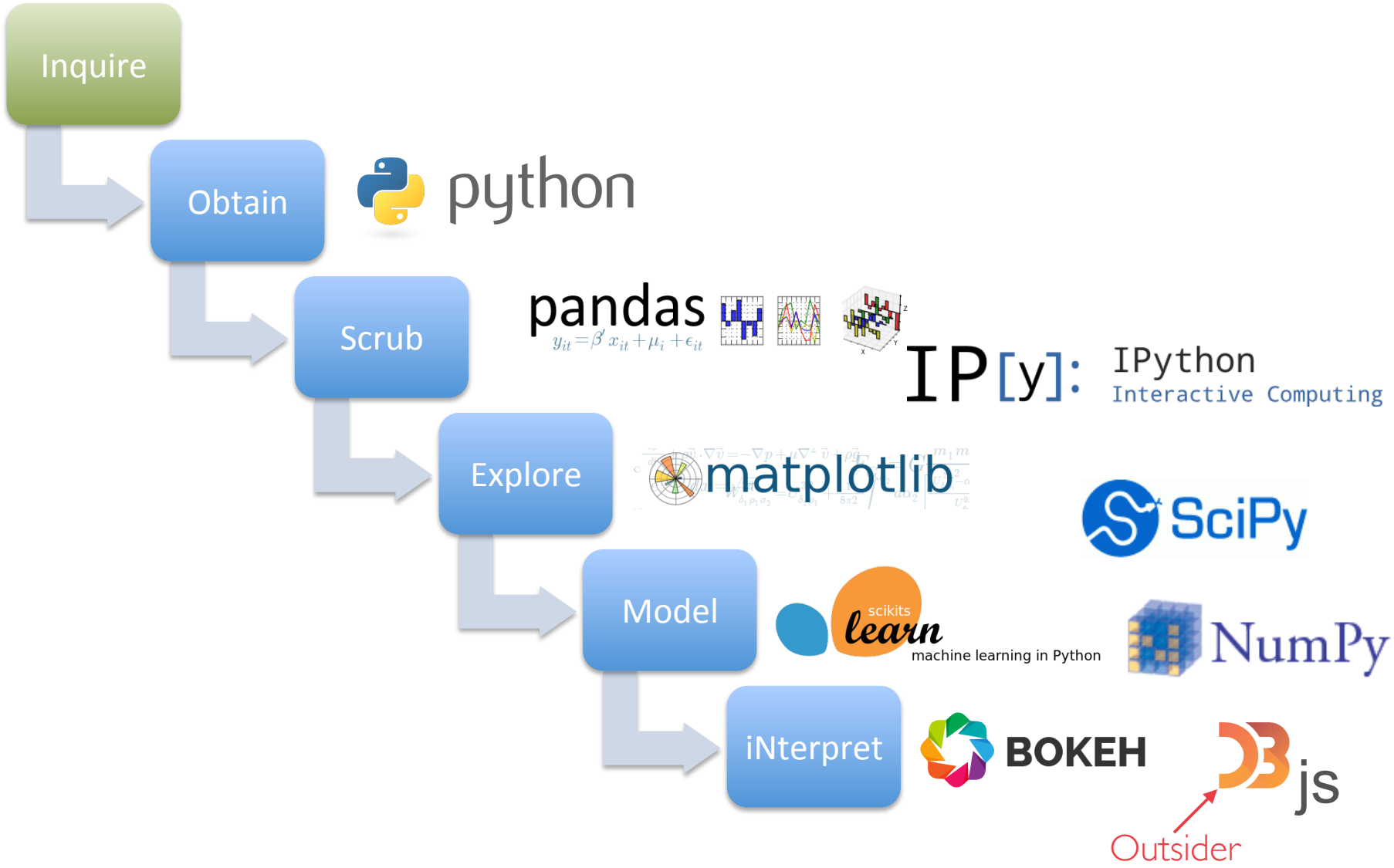
scikits learn
machine learning in Python

NumPy

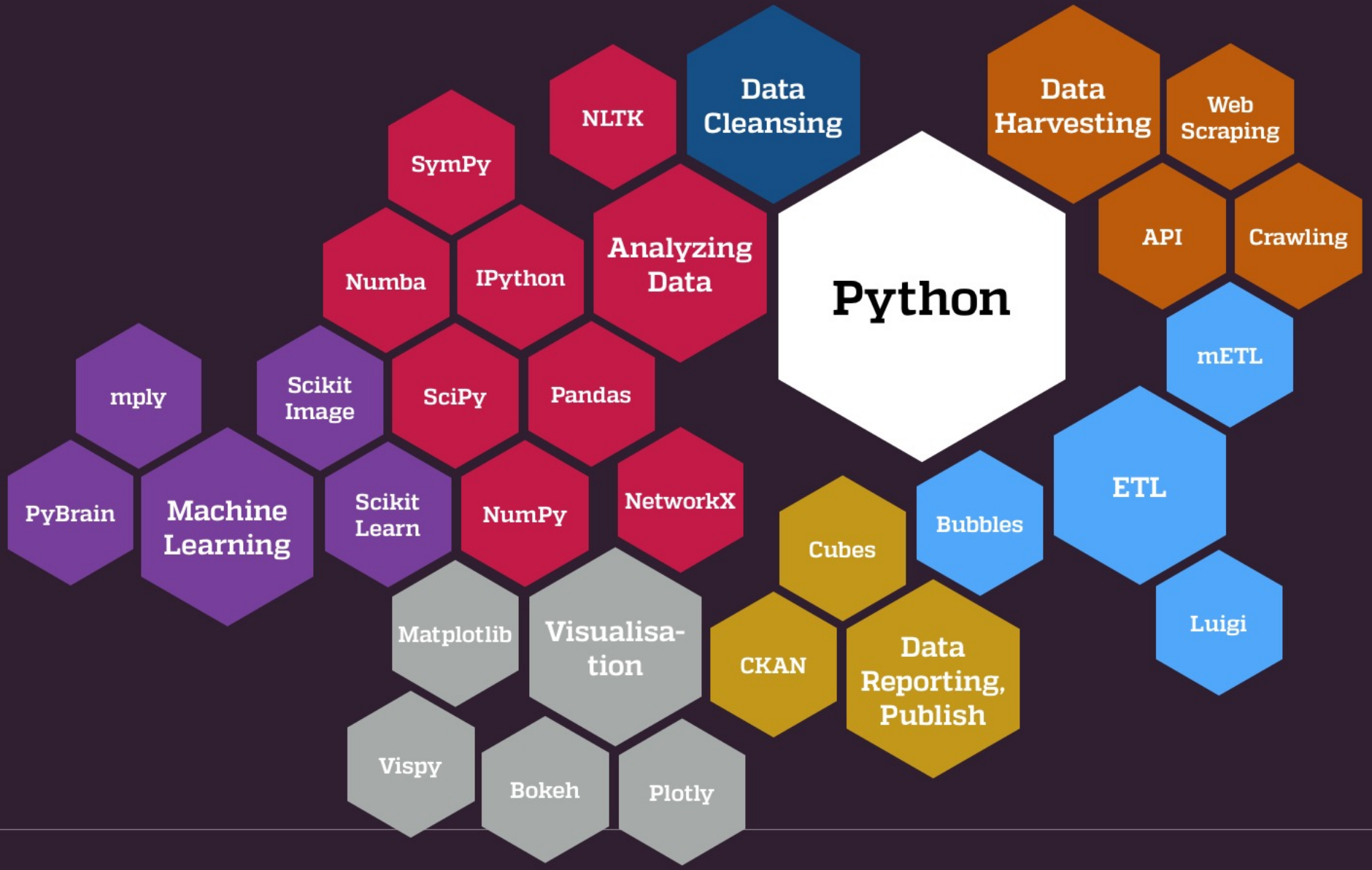
SciPy.org Sponsored By ENTHOUGHT

matplotlib

PYTHON IS IOSEMN



Python Data Science Ecosystem


Data Cleansing
NLTK
SymPy
Numba
IPython
Analyzing Data
Python
Data Harvesting
Web Scraping
API
Crawling
mETL
ETL
Luigi
Bubbles
Cubes
Data Reporting, Publish
CKAN
mply
Scikit Image
SciPy
Pandas
PyBrain
Machine Learning
Scikit Learn
NumPy
NetworkX
Matplotlib
Visualisa-tion
Vispy
Bokeh
Plotly

Packages - Data Manipulation

- NumPy
  - Low level array operations
- pandas
  - Data tables and in-memory manipulation
- Dask
  - Parallel out-of-core array manipulation
- Blaze
  - High level interface for databases and different computational backends

Packages - Modelling

- SciPy
  - FFTs, integration, other general algorithms
- StatsModels (Statistics in Python)
  - Statistical distributions and tests
- scikit learn
  - Machine Learning pipelines
- PyMC3
  - Bayesian Probabilistic Programming

Packages - Visualisation

- matplotlib
  - Widely used and powerful plotting package
- seaborn
  - Opinionated but beautiful data visualisations
- Bokeh
  - Interactive plotting with server option
- plotly
  - Graphics API with translation between languages (e.g. Python -> D3)

IPython Notebooks

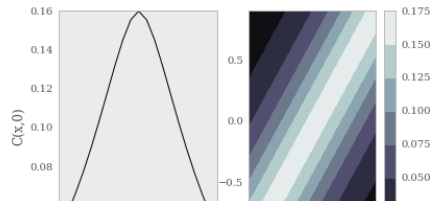
IPython Interactive Computing

```
In [34]: from pymc.gp.cov_funs import matern
         import numpy as np
         C = Covariance(eval_fun=matern.euclidean, diff_degree=1.4, amp=0.4, scale=1, rank_limit=1000)

         subplot(1,2,2)
         contourf(x, x, C(x,x).view(ndarray), origin='lower', extent=(-1,1,-1,1), cmap=cm.bone)
         colorbar()

         subplot(1,2,1)
         plot(x, C(x,0).view(ndarray), 'k-')
         ylabel('C(x,0)')

Out[34]: <matplotlib.text.Text at 0x112713290>
```



jupyter

Why should I become a Data Scientist?

DEMAND & SUPPLY

"We project a need for 1.5 million additional managers and analysts in the United States who can ask the right questions and consume the results of the analysis of Big Data effectively."

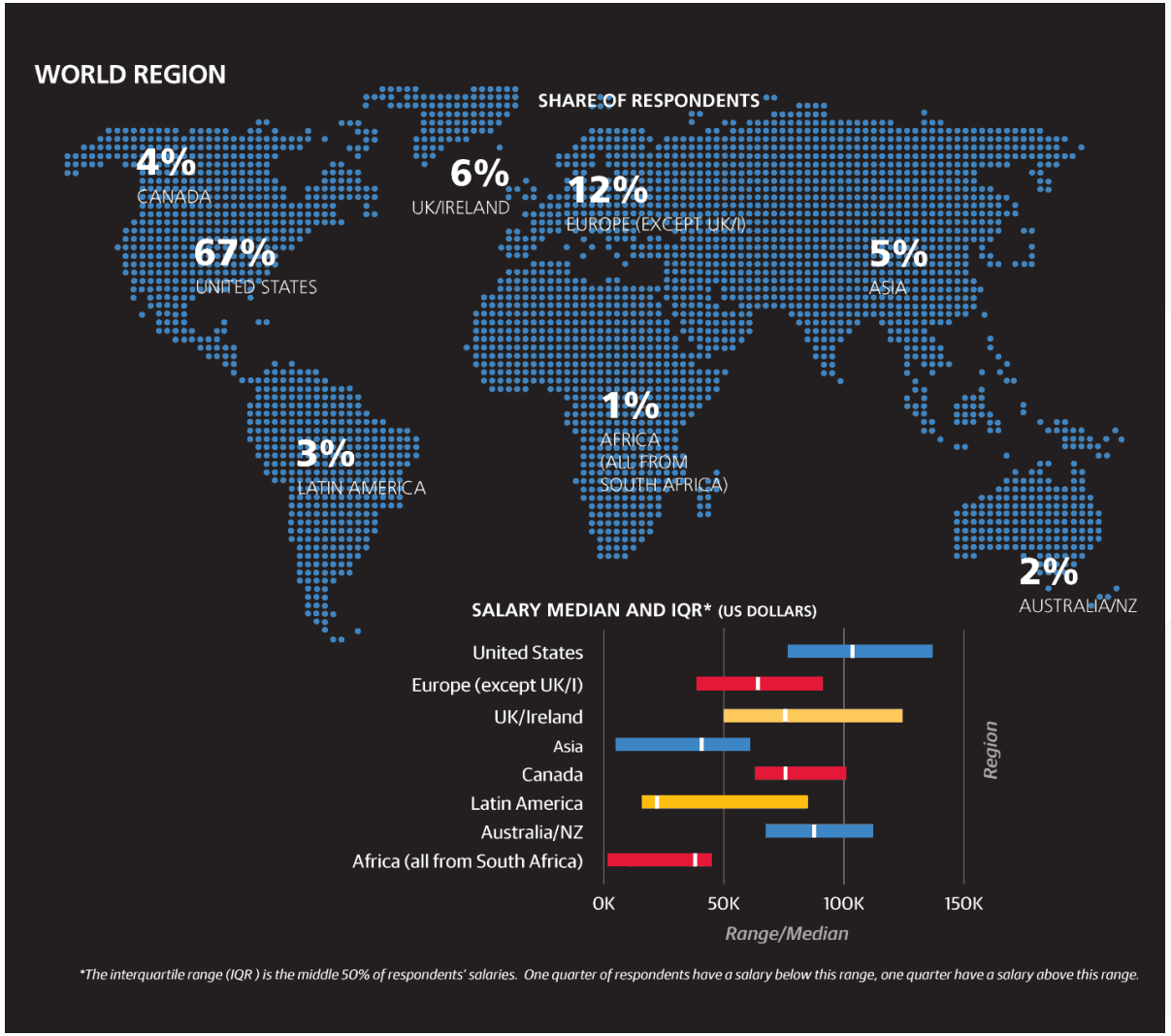
"A significant constraint on realizing value from Big Data will be a shortage of talent, particularly of people with deep expertise in statistics and machine learning, and the managers and analysts who know how to operate companies by using insights from Big Data."

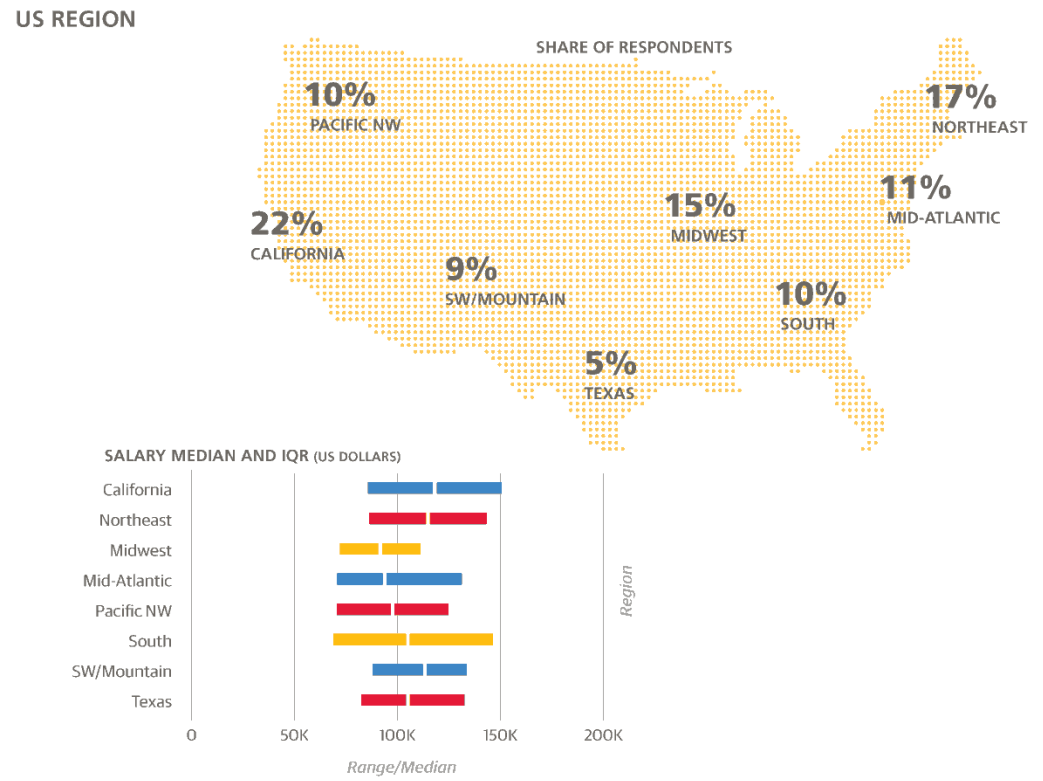
Big data: The next frontier for innovation, competition, and productivity, McKinsey report

"By 2018 the United States will experience a shortage of 190,000 skilled data scientists, and 1.5 million managers and analysts capable of reaping actionable insights from the big data deluge."

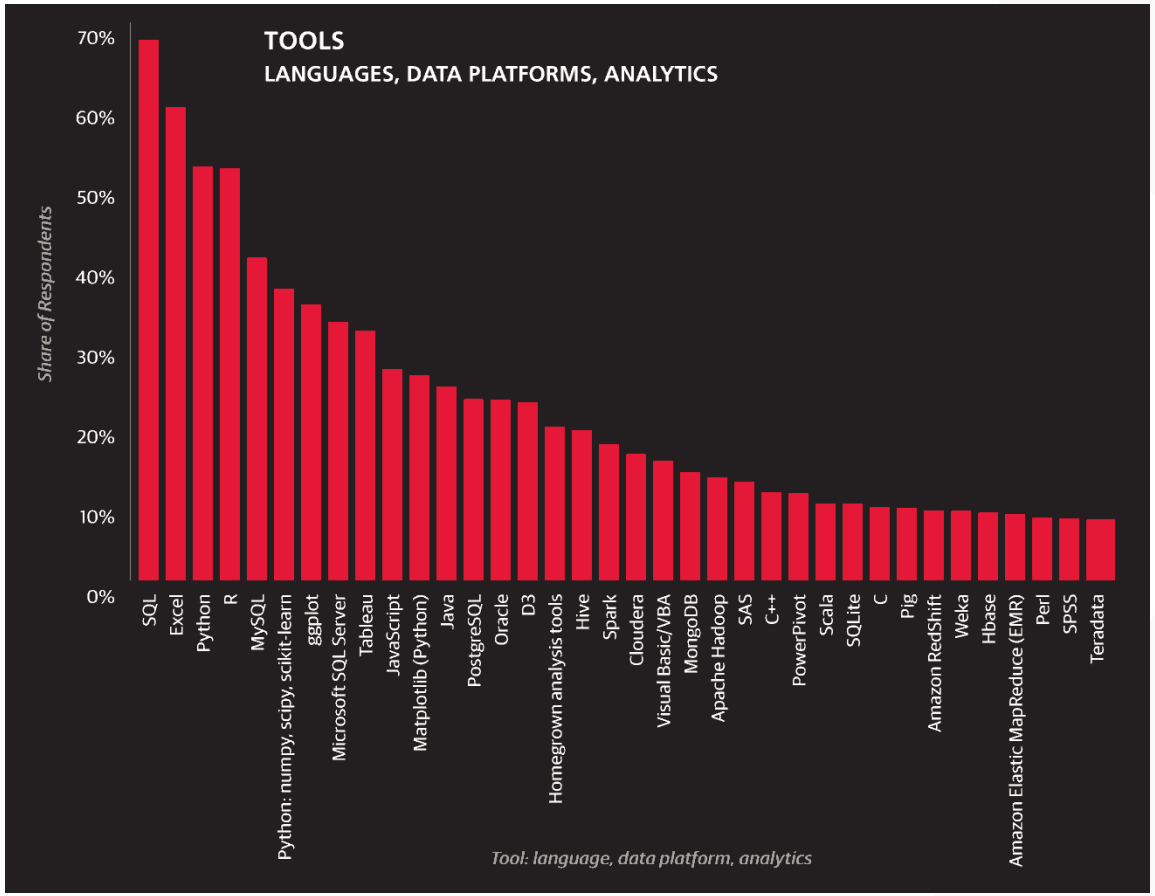
Game changers: Five opportunities for US growth and renewal, McKinsey report

Show me the money





OK. How so do I become a Data Scientist?



Read books on
- Statistics
- Machine Learning
- Programming
- Databases

Take University courses

Apply for internships to work on real-life projects

Spend hours debugging on StackOverflow

Why python?

- It’s awesome and popular!
- Free and Open Source language.
- Readable syntax.
- Great for interactive work
- Easy to learn and has an active community.
- Large amount of libraries.
- High level, general purpose.
- Backed up with fast C & Fortran numerical libraries

Get ANACONDA

- Free enterprise-ready cross platform Python distribution for large- scale data processing, predictive analytics, and scientific computing.

- Download here http://docs.continuum.io/anaconda/install

- Features https://www.continuum.io/why-anaconda

Python for big data

hadoop Streaming

PYDOOP

Pig UDFs in Jython

HADOOPY

pySpark

In-memory cluster computing framework for large-scale data processing

Some facts about Apache Spark

- Started in 2009 by AMP Lab at UC Berkeley.
- Graduated from Apache Incubator earlier this year.
- Close to 300 contributors on Github.
- Developed using Scala, with Java and Python APIs.
- Can sit on an existing Hadoop cluster.
- Processes data up to 100x faster than Hadoop Map-Reduce in memory or up to 10x faster in disk.

Some misconceptions About Spark

Misconception #1

You need to know **Scala** or **Java** to use Spark

Misconception #2

There are not enough documentations or example codes available to get started on **PySpark**

Misconception #3

Not all Spark features are available for **Python** or **PySpark**

ALL FALSE

Misconception #1

You need to know **Scala** or **Java** to use Spark

FALSE

Misconception #2

There are not enough documentations or example codes available to get started on **PySpark**

FALSE

Misconception #3

Not all Spark features are available for **Python** or **PySpark**

FALSE*

Pyspark features

- Provides interactive shell for processing data from command line.
- 2x to 10x less code than standalone programs.
- Can be used from iPython shell or notebook.
- Full support for Spark SQL (previously Shark).
- Spark Streaming

MLLIB

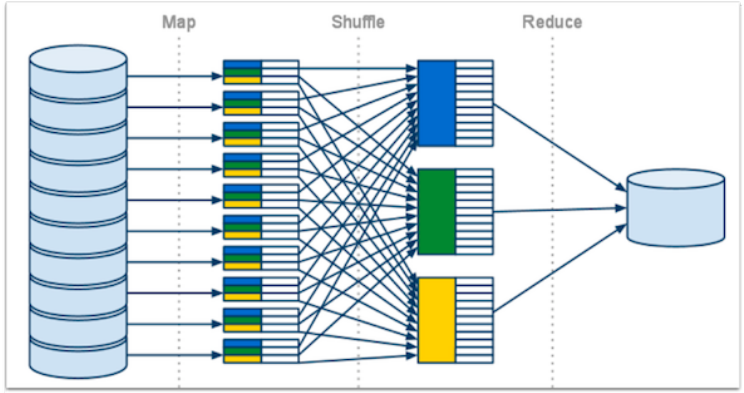
- Model building and eval
- Fast
- Basics covered
  - LR, SVM, Decision tree
  - PCA, SVD
  - K-means
  - ALS
- Algorithms expect RDDs of consistent types (i.e. `LabeledPoints`)

Data Scientists

- Rich, scalable machine learning libraries (MLlib)
  - Statistics - Correlation, sampling, hypothesis testing
  - ML - Classification, Regression, Collaborative filtering, Clustering, Dimensionality Reduction etc.
- Seamless integration of Numpy, Matplotlib and Pandas for data wrangling and visualizations.
- Advantage of in-memory processing for iterative tasks

Map-reduce vs spark

Hadoop Map-Reduce



- A programming paradigm for batch processing.
- Data loaded and read from disk for each iteration and finally written to disk.
- Fault tolerance achieved through data replication on data nodes.
- Each Pig/Hive query spawns a separate Map-Reduce job and reads from disk.

What is different in Spark?

- Data is cached in RAM from disk for iterative processing.
- If data is too large for memory, rest is spilled into disk.
- Interactive processing of datasets without having to reload in the memory.
- Dataset is represented as RDD (Resilient Distributed Dataset) when loaded into Spark Context.
- Fault tolerance achieved through RDD and lineage graphs.

Hadoop Issues

- Pros
  - Reliable in face of failure (*will* happen at scale) - disk, network, node, rack …
  - Very scalable: ~40,000 nodes at Yahoo!
- Cons
  - Disk I/O for every job
  - Unwieldy API (hence Cascading, Scalding, Crunch, Hadoopy …)
  - Very hard to debug - especially Streaming jobs 😱

So Why Spark?

- **In-memory** caching == fast!
- Broadcast variables and accumulator primitives built-in
- Resilient Distributed Datasets (RDD) - recomputed on the fly in case of failure
- **Hadoop compatible**
- Rich, functional API in **Scala**, **Python**, **Java** and **R**
- One platform for multiple use cases:
  - Shark / SparkSQL - **SQL** on Spark
  - Spark **Streaming** - Real time processing
  - **Machine Learning** - MLlib
  - **Graph Processing** - GraphX

Functional API

- Functional Python

```
lines = open("somefile").readlines()
words = [(word, 1) for line in lines for word in line.split(" ")]
counts = [(key, len(list(group))) for key, group in
      groupby(sorted(counts, key=lambda x: x[0]), lambda x: x[0])]
```

- Functional PySpark

```
counts = file.flatMap(lambda line: line.split(" ")) \
             .map(lambda word: (word, 1)) \
             .reduceByKey(lambda a, b: a + b)
```

Plugging in Python Libraries

- Such as scikit-learn

```
from sklearn import linear_model as lm

# init stochastic gradient descent
sgd = lm.SGDClassifier(loss='log')
# training
for i in range(100):
  sgd = sc.parallelize(data, numSlices=slices) \
          .mapPartitions(lambda x: train(x, sgd)) \
          .reduce(lambda x, y: merge(x, y))
  sgd = avg_model(sgd, slices) # averaging weight vector
```

SparkSQL

- SparkSQL

```
events = rdd.map(lambda row: Row(time = row[0], event=row[1], …))

# complex join and filter in Spark
...

schema = sqlContext.inferSchema(events)
events.registerTempTable('events')
val aggs = sql("select
              from_unixtime(cast(time/1000.0 as bigint), 'yyyy-MM-dd HH:00:00') hour,
              event,
              count(1)
            from events …")

# aggs is a normal PySpark RDD, with all the normal operations available
aggs.map( … )
...
```

Let’s begin!

One of Python's most useful features is its interactive interpreter. It allows for very fast testing of ideas without the overhead of creating test files as is typical in most programming languages. However, the interpreter supplied with the standard Python distribution is somewhat limited for extended interactive use.

IPython

A comprehensive environment for **interactive** and **exploratory** computing

Three Main Components:

- An enhanced interactive Python **shell**.
- A decoupled two-process *communication model*, which allows for multiple clients to connect to a computation kernel, most notably the web-based notebook.
- An *architecture* for interactive **parallel computing**.

IP[y]: IPython
Interactive Computing

Some of the many useful features of IPython includes:

- Command history, which can be browsed with the up and down arrows on the keyboard.
- Tab auto-completion.
- In-line editing of code.
- Object introspection, and automatic extract of documentation strings from python objects like classes and functions.
- Good interaction with operating system shell.
- Support for multiple parallel back-end processes, that can run on computing clusters or cloud services like Amazon EC2.

IP[y]: IPython
Interactive Computing

IPython provides a rich architecture for **interactive** computing with:

- A powerful interactive shell.
- A kernel for Jupyter.
- Easy to use, high performance tools for parallel computing.
- Support for interactive data visualization and use of GUI toolkits.
- Flexible, embeddable interpreters to load into your own projects.

IPython

IPython is an interactive shell that addresses the limitation of the standard python interpreter, and it is a work-horse for scientific use of python. It provides an interactive prompt to the python interpreter with a greatly improved user-friendliness.

It comes with a browser-based **notebook** with support for code, text, mathematical expressions, inline plots and other rich media.

You don't need to know anything beyond Python to start using IPython – just type commands as you would at the standard Python prompt. But IPython can do much more than the standard prompt...

The Four Most Helpful Commands

The four most helpful commands is shown to you in a banner, every time you start IPython:

Command	Description
`?`	Introduction and overview of IPython's features.
`%quickref`	Quick reference.
`help`	Python's own help system.
`object?`	Details about `object`, use `object??` for extra details.

Tab Completion

Tab completion, especially for attributes, is a convenient way to explore the structure of any object you're dealing with. Simply type `object_name.<TAB>` to view the object's attributes. Besides Python objects and keywords, tab completion also works on file and directory names.

Running and Editing

The `%run` magic command allows you to run any python script and load all of its data directly into the interactive namespace. Since the file is re-read from disk each time, changes you make to it are reflected immediately (unlike imported modules, which have to be specifically reloaded). IPython also includes `dreload`, a recursive reload function.

`%run` has special flags for timing the execution of your scripts (`-t`), or for running them under the control of either Python's pdb debugger (`-d`) or profiler (`-p`).

The `%edit` command gives a reasonable approximation of multiline editing, by invoking your favorite editor on the spot. IPython will execute the code you type in there as if it were typed interactively.

Magic Functions ...

The following examples show how to call the builtin `%timeit` magic, both in line and cell mode:

```
In [1]: %timeit range(1000)
100000 loops, best of 3: 7.76 us per loop

In [2]: %%timeit x = range(10000)
   ...: max(x)
   ...:
1000 loops, best of 3: 223 us per loop
```

The builtin magics include:

- Functions that work with code: `%run`, `%edit`, `%save`, `%macro`, `%recall`, etc.
- Functions which affect the shell: `%colors`, `%xmode`, `%autoindent`, `%automagic`, etc.
- Other functions such as `%reset`, `%timeit`, `%%writefile`, `%load`, or `%paste`.

Exploring your Objects

Typing `object_name?` will print all sorts of details about any object, including docstrings, function definition lines (for call arguments) and constructor details for classes. To get specific information on an object, you can use the **magic** commands `%pdoc`, `%pdef`, `%psource` and `%pfile`.

Magic Functions

IPython has a set of predefined **magic functions** that you can call with a command line style syntax. There are two kinds of magics, line-oriented and cell-oriented.

- Line magics are prefixed with the `%` character and work much like OS command-line calls: they get as an argument the rest of the line, where arguments are passed without parentheses or quotes.
- Cell magics are prefixed with a double `%%`, and they are functions that get as an argument not only the rest of the line, but also the lines below it in a separate argument.

System Shell Commands

To run any command at the system shell, simply prefix it with `!`. You can capture the output into a Python list. To pass the values of Python variables or expressions to system commands, prefix them with `$`.

System Aliases

It's convenient to have aliases to the system commands you use most often. This allows you to work seamlessly from inside IPython with the same commands you are used to in your system shell. IPython comes with some pre-defined aliases and a complete system for changing directories, both via a stack (`%pushd`, `%popd` and `%dhist`) and via direct %cd. The latter keeps a history of visited directories and allows you to go to any previously visited one.

Magic Functions ...

You can always call them using the `%` prefix, and if you're calling a line magic on a line by itself, you can omit even that: `run thescript.py`. You can toggle this behavior by running the `%automagic` magic.

Cell magics must always have the `%%` prefix.

A more detailed explanation of the magic system can be obtained by calling `%magic`, and for more details on any magic function, call `%somemagic?` to read its docstring. To see all the available magic functions, call `%lsmagic`.

System Shell Commands ...

```
!ping www.bbc.co.uk
files = !ls                  # capture
!grep -rF $pattern ipython/*  # passing vars
```


History ...

Input and output history are kept in variables called In and Out, keyed by the prompt numbers. The last three objects in output history are also kept in variables named _, __ and ___.

You can use the `%history` magic function to examine past input and output. Input history from previous sessions is saved in a database, and IPython can be configured to save output history.

Several other magic functions can use your input history, including `%edit`, `%rerun`, `%recall`, `%macro`, `%save` and `%pastebin`. You can use a standard format to refer to lines:

```
%pastebin 3 18-20 ~1/1-5
```

This will take line 3 and lines 18 to 20 from the current session, and lines 1-5 from the previous session.

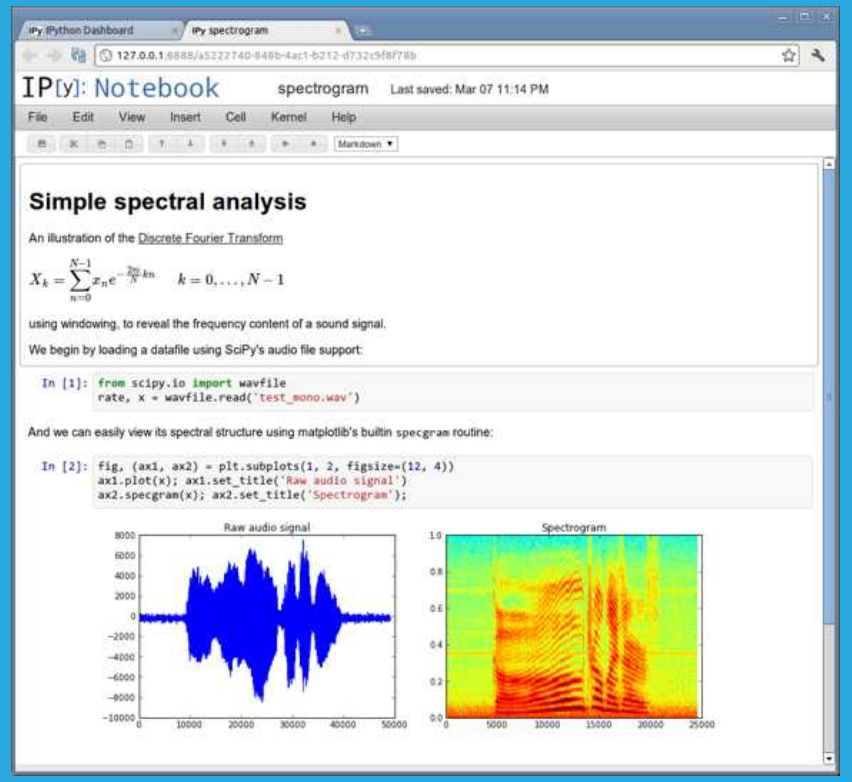
History

IPython stores both the commands you enter, and the results it produces. You can easily go through previous commands with the up- and down-arrow keys, or access your history in more sophisticated ways.

Debugging

After an exception occurs, you can call `%debug` to jump into the Python debugger (pdb) and examine the problem. Alternatively, if you call `%pdb`, IPython will automatically start the debugger on any uncaught exception. You can print variables, see code, execute statements and even walk up and down the call stack to track down the true source of the problem. This can be an efficient way to develop and debug code, in many cases eliminating the need for print statements or external debugging tools.

You can also step through a program from the beginning by calling `%run -d theprogram.py`.

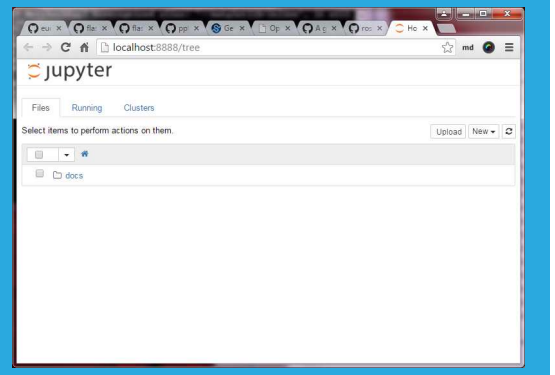


Project Jupyter

The IPython Notebook (2011)

- Rich Web Client
- Text & Math
- Code
- Results
- Share, Reproduce.

See: **Fernando Perez**, IPython & Project Jupyter



Jupyter Notebook

The Jupyter Notebook is a web application for **interactive** data science and scientific computing.

Using the Jupyter Notebook, you can author engaging documents that combine live-code with narrative text, equations, images, video, and visualizations. By encoding a complete and reproducible record of a computation, the documents can be shared with others on GitHub, Dropbox, and the Jupyter Notebook Viewer.

```
# Install
sudo apt-get install build-essential python-dev
pip install jupyter

# Start
jupyter notebook

# Previously
pip install "ipython[notebook]"
ipython notebook
```

See: Jupyter@RTD

- Open source, interactive data science and scientific computing across over 40 programming languages.
- The Jupyter Notebook is a web application that allows you to create and share documents that contain live code, equations, visualizations and explanatory text.
- Uses include: data cleaning and transformation, numerical simulation, statistical modeling, machine learning and much more.

Language of Choice

The Notebook has support for over 40 programming languages, including those popular in Data Science such as Python, R, Julia and Scala.

Share Notebooks

Notebooks can be shared with others using email, Dropbox, GitHub and the Jupyter Notebook Viewer.

Interactive Widgets

Code can produce rich output such as images, videos, LaTeX, and JavaScript. Interactive widgets can be used to manipulate and visualize data in realtime.

Big-Data Integration

Leverage big data tools, such as Apache Spark, from Python, R and Scala. Explore that same data with pandas, scikit-learn, ggplot2, dplyr, etc.

See: Jupyter.ORG Website

Project Jupyter

IPython

- Interactive Python shell at the terminal
- Kernel for this protocol in Python
- Tools for Interactive Parallel computing

Jupyter

- Network protocol for interactive computing
- Clients for protocol
  - Console
  - Qt Console
  - Notebook
- Notebook file format & tools (nbconvert...)
- Nbviewer

jupyter

What's in a name?

- Inspired by the open languages of science:
  - Julia, Python & R
  - Not an acronym: all languages equal class citizens.
- Astronomy and Scientific Python: A long and fruitful collaboration
- Galileo's notebooks:
  - The original, open science, data-and-narrative papers
  - Authorea: "Science was Always meant to be Open"

See: **Fernando Perez**, IPython & Project Jupyter

Project Jupyter

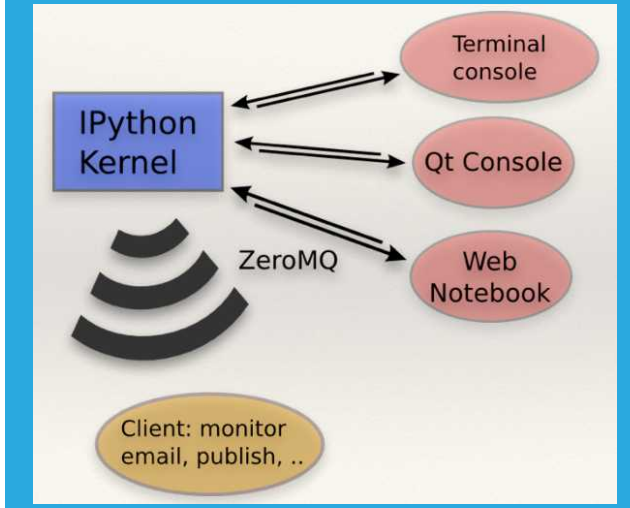
From IPython to Project Jupyter

- Not just about Python: Kernels in **any** language
- IPython : "Official"
- IJulia, IRKernel, IHaskell, IFSharp, Ruby, IScala, IErlang, .. Lots more! ~37 and counting
- Why is it called IPython, if it can do Julia, R, Haskell, Ruby, **...** ?"

TL;DR

- Separation of the language-agnostic components
- **Jupyter**: protocol, format, multi-user server
- **IPython**: interactive Python console, Jupyter kernel
- Jupyter kernels = Languages which can be used from the notebook (37 and counting)

A Simple and Generic Architecture



See: **Fernando Perez**, IPython & Project Jupyter

ANALYTIXLABS

Advance Big Data Science is a comprehensive program with following modules, weekly assignments and case studies

Module 1	***Python – 21 hours + Practice exercises*** Basic data handling, data manipulation, descriptive analytics and visualization
Module 2	***Hadoop & it's components – 15 hours + Practice*** Introduction to the ecosystem and focus on core components – Pig, Hive, MapReduce, HDFS, Impala Hbase and other Apache projects
Module 3	***Spark – 33 hours + Practice*** Working on Spark and connecting to Hive and Python for advance analytics and Machine Learning
Module 4	***R (video based) – 12 hours*** Data manipulation, descriptive analytics and visualization using R

Crafted by team of experts and maintains a balance between theoretical concepts and practical applications

Advance Big Data Science using Python-R-Hadoop-Spark (1/3)

Total Duration: 84 hours + Practice

Introduction to Data Science

- What is Data Science?
- Data Science Vs. Analytics vs. Data warehousing, OLAP, MIS Reporting
- Relevance in industry and need of the hour
- Type of problems and objectives in various industries
- How leading companies are harnessing the power of Data Science?
- Different phases of a typical Analytics/Data Science projects

Python: Introduction & Essentials

- Overview of Python- Starting Python
- Introduction to Python Editors & IDE's(Canopy, pycharm, Jupyter, Rodeo, Ipython etc…)
- Custom Environment Settings
- Concept of Packages/Libraries - Important packages(NumPy, SciPy, scikit-learn, Pandas, Matplotlib, etc)
- Installing & loading Packages & Name Spaces
- Data Types & Data objects/structures (Tuples, Lists, Dictionaries)
- List and Dictionary Comprehensions
- Variable & Value Labels – Date & Time Values
- Basic Operations - Mathematical - string - date
- Reading and writing data
- Simple plotting
- Control flow
- Debugging
- Code profiling

Python: Accessing/Importing and Exporting Data

- Importing Data from various sources (Csv, txt, excel, access etc)
- Database Input (Connecting to database)
- Viewing Data objects - subsetting, methods
- Exporting Data to various formats

Python: Data Manipulation – cleansing

- Cleansing Data with Python
- Data Manipulation steps(Sorting, filtering, duplicates, merging, appending, subsetting, derived variables, sampling, Data type conversions, renaming, formatting etc)
- Data manipulation tools(Operators, Functions, Packages, control structures, Loops, arrays etc)
- Python Built-in Functions (Text, numeric, date, utility functions)
- Python User Defined Functions
- Stripping out extraneous information
- Normalizing data
- Formatting data
- Important Python Packages for data manipulation (Pandas, Numpy etc)

Python: Data Analysis – Visualization

- Introduction exploratory data analysis
- Descriptive statistics, Frequency Tables and summarization
- Univariate Analysis (Distribution of data & Graphical Analysis)
- Bivariate Analysis(Cross Tabs, Distributions & Relationships, Graphical Analysis)
- Creating Graphs- Bar/pie/line chart/histogram/boxplot/scatter/density etc)
- Important Packages for Exploratory Analysis(NumPy Arrays, Matplotlib, Pandas and scipy.stats etc)

Python: Basic statistics

- Basic Statistics - Measures of Central Tendencies and Variance
- Building blocks - Probability Distributions - Normal distribution - Central Limit Theorem
- Inferential Statistics -Sampling - Concept of Hypothesis Testing
- Statistical Methods - Z/t-tests (One sample, independent, paired), Anova, Correlation and Chi-square

Python: Polyglot Programming

- Making Python talk to other languages and database systems
- How do R and Python play with each other, why it's essential to know both

Hadoop: Introduction to Hadoop & Ecosystem

- Introduction to Hadoop
- Hadoopable Problems - Uses of Big Data analytics in various industries like Telecom, E- commerce, Finance and Insurance etc
- Problems with Traditional Large-Scale Systems & Existing Data analytics Architecture
- Key technology foundations required for Big Data
- Comparison of traditional data management systems with Big Data management systems
- Evaluate key framework requirements for Big Data analytics
- Apache projects in the Hadoop Ecosystem
- Hadoop Ecosystem & Hadoop 2.x core components
- Explain the relevance of real-time data
- Explain how to use Big Data and real-time data as a Business planning tool

Advance Big Data Science using Python-R-Hadoop-Spark (2/3)

Total Duration: 84 hours + Practice

Hadoop core components- HDFS
- HDFS Overview & Data storage in HDFS
- Get the data into Hadoop from local machine(Data Loading Techniques) - vice versa

Hadoop core components- MapReduce (YARN)
- Map Reduce Overview (Traditional way Vs. MapReduce way)
- Concept of Mapper & Reducer
- Understanding Map reduce program skeleton
- Running MapReduce job in Command line

Hadoop Data Analysis Tools: Hadoop-PIG
- Introduction to PIG - MapReduce Vs Pig, Pig Use Cases
- Pig Latin Program & Execution
- Pig Latin : Relational Operators, File Loaders, Group Operator, COGROUP Operator, Joins and COGROUP, Union, Diagnostic Operators, Pig UDF
- Use Pig to automate the design and implementation of MapReduce applications
- Data Analysis using PIG

Hadoop Data Analysis Tools: Hadoop-Hive
- Introduction to Hive - Hive Vs. PIG - Hive Use Cases
- Discuss the Hive data storage principle
- Explain the File formats and Records formats supported by the Hive environment
- Perform operations with data in Hive
- Hive QL: Joining Tables, Dynamic Partitioning, Custom Map/Reduce Scripts
- Hive Script, Hive UDF

Hadoop Data Analysis Tools: Impala
- Introduction to Impala & Architecture
- How Impala executes Queries and its importance.
- Hive vs. PIG vs. Impala
- Extending Impala with User Defined functions
- Improving Impala Performance

Hadoop Data Analysis Tools: Hbase (NOSQL Database)
- Introduction to NoSQL Databases, types, and Hbase
- HBase v/s RDBMS, HBase Components, HBase Architecture
- HBase Cluster Deployment

Hadoop: Introduction to other Apache Projects
- Introduction to Zookeeper/Oozie/Sqoop/Flume

SPARK: Introduction
- Introduction to Apache Spark
- Streaming Data Vs. In Memory Data
- Map Reduce Vs. Spark
- Modes of Spark
- Spark Installation Demo
- Overview of Spark on a cluster
- Spark Standalone Cluster

Spark: Spark in practice
- Invoking Spark Shell
- Creating the Spark Context
- Loading a File in Shell
- Performing Some Basic Operations on Files in Spark Shell
- Building a Spark Project with sbt
- Running Spark Project with sbt
- Caching Overview
- Distributed Persistence
- Spark Streaming Overview(Example: Streaming Word Count)

Spark: Spark meets Hive
- Analyze Hive and Spark SQL Architecture
- Analyze Spark SQL
- Context in Spark SQL
- Implement a sample example for Spark SQL
- Integrating hive and Spark SQL
- Support for JSON and Parquet File Formats Implement Data Visualization in Spark
- Loading of Data
- Hive Queries through Spark
- Performance Tuning Tips in Spark
- Shared Variables: Broadcast Variables & Accumulators

Data Science using SPARK – Python
- Hadoop - Python Integration
- Spark - Python Integration (PySpark)

Spark -Python: Machine Learning -Predictive Modeling – Basics
- Introduction to Machine Learning & Predictive Modeling
- Types of Business problems - Mapping of Techniques
- Major Classes of Learning Algorithms -Supervised vs Unsupervised Learning,
- Different Phases of Predictive Modeling (Data Pre-processing, Sampling, Model Building, Validation)
- Overfitting (Bias-Variance Trade off) & Performance Metrics
- Types of validation(Bootstrapping, K-Fold validation etc)

Advance Big Data Science using Python-R-Hadoop-Spark (3/3)

Total Duration: 84 hours + Practice

Spark -Python: Machine Learning in Practice

- Linear Regression
- Logistic Regression
- Segmentation - Cluster Analysis (K-Means)
- Decision Trees (CHAID/CART/CD 5.0)
- Artificial Neural Networks(ANN)
- Support Vector Machines(SVM)
- Ensemble Learning (Random Forest, Bagging & boosting)
- Other Techniques (KNN, Naïve Bayes, LDA/QDA etc)
- Important Packages for Machine Learning (Sci Kit Learn, scipy.stats etc)

R: Introduction - Data Importing/Exporting - (Video Based)

- Introduction R/R-Studio - GUI
- Concept of Packages - Useful Packages (Base & other packages) in R
- Data Structure & Data Types (Vectors, Matrices, factors, Data frames, and Lists)
- Importing Data from various sources
- Database Input (Connecting to database)
- Exporting Data to various formats)
- Viewing Data (Viewing partial data and full data)
- Variable & Value Labels – Date Values

R: Data Manipulation - (Video Based)

- Data Manipulation steps (Sorting, filtering, duplicates, merging, appending, subsetting, derived variables, sampling, Data type conversions, renaming, formatting, etc)
- Data manipulation tools(Operators, Functions, Packages, control structures, Loops, arrays, etc)
- R Built-in Functions (Text, Numeric, Date, utility)
- R User Defined Functions
- R Packages for data manipulation(base, dplyr, plyr, reshape, car, sqldf, etc)

R: Data Analysis – Visualization - (Video Based)

- Introduction exploratory data analysis
- Descriptive statistics, Frequency Tables and summarization
- Univariate Analysis (Distribution of data & Graphical Analysis)
- Bivariate Analysis(Cross Tabs, Distributions & Relationships, Graphical Analysis)
- Creating Graphs- Bar/pie/line chart/histogram/boxplot/scatter/density etc)
- R Packages for Exploratory Data Analysis(dplyr, plyr, gmodes, car, vcd, Hmisc, psych, doby etc)
- R Packages for Graphical Analysis (base, ggplot, lattice,etc)

Integration of Hadoop with R

Contact Us

Visit us on: http://www.analytixlabs.in/

For course registration, please visit: http://www.analytixlabs.co.in/course-registration/

For more information, please contact us: http://www.analytixlabs.co.in/contact-us/

Or email: info@analytixlabs.co.in

Call us we would love to speak with you: (+91) 7530818107

Join us on:

Twitter - http://twitter.com/#!/AnalytixLabs

Facebook - http://www.facebook.com/analytixlabs

LinkedIn - http://www.linkedin.com/in/analytixlabs

Blog - http://www.analytixlabs.co.in/category/blog/