

Student Guide

Introduction to Data Science

Skills Academy: Data Science Technologist



March 2018

NOTICES

This information was developed for products and services offered in the USA.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service. IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

*IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
United States of America*

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you. Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

TRADEMARKS

IBM, the IBM logo, ibm.com, Big SQL, Db2, and Hortonworks are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

Adobe, and the Adobe logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

© Copyright International Business Machines Corporation 2018.

This document may not be reproduced in whole or in part without the prior written permission of IBM.

US Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Preface.....	P-1
Contents.....	P-3
Course overview.....	P-6
Document conventions.....	P-7
Labs	P-8
Additional training resources	P-9
IBM product help	P-10
Unit 1 Data Science and Data Science Notebooks	1-1
Unit objectives	1-3
A lead-in to Data Analysis	1-4
What is methodology?	1-5
Data Science - using the Scientific Method	1-6
Applying methodology to Data Science	1-7
Diagrams for Data Science Methodology	1-8
Data Science Methodology.....	1-9
Data science project lifecycle	1-10
Analytics & the analytic life cycle	1-11
Systematic approach in Three Phases	1-12
Godsey's Three Phases in a diagram.....	1-14
So, then what is Data Science?.....	1-15
AI >> Machine Learning >> Deep Learning	1-16
The Work of the Data Scientist.....	1-17
Drew Conway's Venn diagram	1-18
Drew Conway's Venn diagram (adapted)	1-21
The data science skill set	1-22
Data Roles & Skill Sets.....	1-24
Business Analysts vs Data Scientists	1-26
What type of education do Data Scientists have?.....	1-27
The art of Data Science in 5 steps.....	1-28
Important things to know about Data Science.....	1-29
Now down to the practical steps in the process	1-30
Mommy, where does data come from?.....	1-31
The scientific method - Generic.....	1-32
Data Notebooks: Tools for data scientists	1-33

The Data Science Pipeline - Summarization.....	1-35
We will use the Anaconda distribution of Python	1-36
If your Python 3 is weak? ...and new to Jupyter?	1-37
Getting started with Jupyter.....	1-38
Data Science - according to Continuum Analytics	1-39
Python libraries for Data Analysis.....	1-40
Pandas.....	1-41
The tools of statistics.....	1-42
Hardware & software for Data Science.....	1-43
Notebooks & Workbenches for Data	1-45
Jupyter Notebook (formerly IPython)	1-46
Apache Zeppelin - a short overview.....	1-47
IBM Watson Studio.....	1-48
Demo: Introducing Watson Studio	1-49
Working with Data Science in the Cloud.....	1-51
Checkpoint	1-52
Checkpoint solution	1-53
Unit summary	1-54
Unit 2 Data Science with Open Source Tools	2-1
Unit objectives	2-3
Supported languages	2-4
Data Science readiness	2-5
Jupyter works with 60+ language Kernels	2-7
Getting started with the Jupyter notebook	2-8
Architecture	2-9
Example of Jupyter in action	2-10
How notebooks help a data scientist	2-11
Why do we want to study & use Jupyter?	2-12
Features of Data Science Notebooks	2-13
Browser compatibility	2-14
Jupyter keyboard-shortcuts	2-15
Jupyter magic commands / functions.....	2-16
Markdown.....	2-18
Example Markdown formatting	2-19
Essential packages - for Python 3	2-21
Essential packages - for Python 3 - continued.....	2-22
Essential packages - for Python 3 - visualization.....	2-23

Essential packages - for Python 3 - visualization 2	2-24
Essential packages - for Python 3 - HTML	2-25
Installing Jupyter	2-26
Architecture of Anaconda 5 with Jupyter Notebook	2-27
Jupyter magic for installing Python packages	2-28
Git / GitHub for version control and for distribution	2-29
Python Data Science Handbook.....	2-31
Demonstration 1: Introduction to Jupyter notebooks.....	2-32
Unit summary	2-47

Course overview

Preface overview

This course is designed to introduce the student to data science and some of the tools of the trade. The student will have a better understanding of the methodology of a data scientist and be able to explain what is involved with a data science project. The student will learn how to use notebooks, data science platforms, and various open source tools.

Intended audience

Data Scientist, Data Engineer, Developers, Anyone interested in learning about data science.

Topics covered

Topics covered in this course include:

- Data Science essentials
- Drew Conway's Venn Diagram - and that of others
- The Scientific Process applied to Data Science
- The steps in running a Data Science project
- Languages used for Data Science (Python, R, Scala, Julia, ...)
- Survey of Data Science Notebooks
- Markdown language with notebooks
- Resources for Data Science, including GitHub
- Jupyter Notebook
- Essential packages: NumPy, SciPy, Pandas, Scikit-learn, NLTK, BeautifulSoup...
- Data visualizations: matplotlib, ..., PixieDust
- Using Jupyter "Magic" commands

Course prerequisites

- None

Document conventions

Conventions used in this guide follow Microsoft Windows application standards, where applicable. As well, the following conventions are observed:

- **Bold**: Bold style is used in demonstration and exercise step-by-step solutions to indicate a user interface element that is actively selected or text that must be typed by the participant.
- *Italic*: Used to reference book titles.
- **CAPITALIZATION**: All file names, table names, column names, and folder names appear in this guide exactly as they appear in the application.
To keep capitalization consistent with this guide, type text exactly as shown.

Labs

Lab format

Labs are designed to allow you to work according to your own pace. Practice with product, to become proficient with how tasks and steps are performed. Use the Purpose and Results sections, to guide you on what you will complete in the lab.

Additional training resources

- Visit IBM Analytics Product Training and Certification on the IBM website for details on:
 - Instructor-led training in a classroom or online
 - Self-paced training that fits your needs and schedule
 - Comprehensive curricula and training paths that help you identify the courses that are right for you
 - IBM Analytics Certification program
 - Other resources that will enhance your success with IBM Analytics Software
- For the URL relevant to your training requirements outlined above, bookmark:
 - Information Management portfolio:
<http://www-01.ibm.com/software/data/education/>
 - Predictive and BI/Performance Management/Risk portfolio:
<http://www-01.ibm.com/software/analytics/training-and-certification/>

IBM product help


Help type	When to use	Location
Task-oriented	You are working in the product and you need specific task-oriented help.	<i>IBM Product</i> - Help link
Books for Printing (.pdf)	<p>You want to use search engines to find information. You can then print out selected pages, a section, or the whole book.</p> <p>Use Step-by-Step online books (.pdf) if you want to know how to complete a task but prefer to read about it in a book.</p> <p>The Step-by-Step online books contain the same information as the online help, but the method of presentation is different.</p>	Start/Programs/ <i>IBM Product</i> /Documentation
IBM on the Web	<p>You want to access any of the following:</p> <ul style="list-style-type: none"> • IBM - Training and Certification • Online support • IBM Web site 	<ul style="list-style-type: none"> • http://www-01.ibm.com/software/analytics/training-and-certification/ • http://www-947.ibm.com/support/entry/portal/Overview/Software • http://www.ibm.com

Unit 1

Data Science and Data Science Notebooks

IBM Training

IBM



Data Science and Data Science Notebooks

Data Science Foundations

© Copyright IBM Corporation 2018
Course materials may not be reproduced in whole or in part without the written permission of IBM.

Unit objectives

- Have a better understanding of methodology “scientific approach” methods used & skills practiced by Data Scientists
- Recognize the iterative nature of a data science project
- Outline the benefits of using Data Science Notebooks
- Describe the mechanisms and tools used with Data Science Notebooks
- Compare and contrast the major Notebooks used by Data Scientists

Unit objectives

A lead-in to Data Analysis

The right data, the right questions, ...

At Google major decisions are based on only a tiny sampling of all their data. You don't always need a ton of data to find important insights. You need the right data. ...

Most important, to squeeze insights out of Big Data, you have to ask the right questions. Just as you can't point a telescope at the night sky and have it discover Pluto for you, you can't download a whole bunch of data and have it discover the secrets of human nature for you.

Stephens-Davidowitz, S. (2017). *Everybody lies: Big data, new data, and what the internet can tell us who we really are*. New York: Dey Street Books / William Morrow / Harper Collins. Pp 21-22.

... and the right methodology

A lead-in to Data Analysis

The book is well worth reading (352 pp. Amazon, paperback \$14).

Reviewers have said:

- “This book is about a whole new way of studying the mind . . . an unprecedented peek into people’s psyches . . . Time and again my preconceptions about my country and my species were turned upside-down by Stephens-Davidowitz’s discoveries . . . endlessly fascinating.” (Steven Pinker, author of *The Better Angels of Our Nature*)
- “Move over *Freakonomics*. Move over *Moneyball*. This brilliant book is the best demonstration yet of how big data plus cleverness can illuminate and then move the world. Read it and you’ll see life in a new way.” (Lawrence Summers, President Emeritus and Charles W. Eliot University Professor of Harvard University)
- “Pivotal . . . A book for those who are intensely curious about human nature, informational analysis, and amusing anecdotes to the tune of Steven Levitt and Stephen Dubner’s *Freakonomics*.” (*Library Journal*)

What is methodology?

- General strategy that guides processes and activities within a domain
- Provides the analyst with a *framework* for how to proceed with the methods, processes, and heuristics used to obtain answers or results
 - Doesn't depend on particular technologies or tools
 - Not a set of techniques or recipes
- In the domain of data science, we are concerned with solving a problem or answering a question through analyzing data
 - Often, we construct a model of some sort to predict outcomes or discover underlying patterns > predictive & descriptive models
 - Goal of modeling is to gain *insights* from which we can formulate *actions* to influence *future outcomes or behaviors*
 - For our purpose, we need a data science methodology to guide us in achieving this goal

Data Science and Data Science Notebooks

© Copyright IBM Corporation 2018

What is methodology?

Definition of **methodology** (*pl.: methodologies*) in the Merriam-Webster Dictionary:

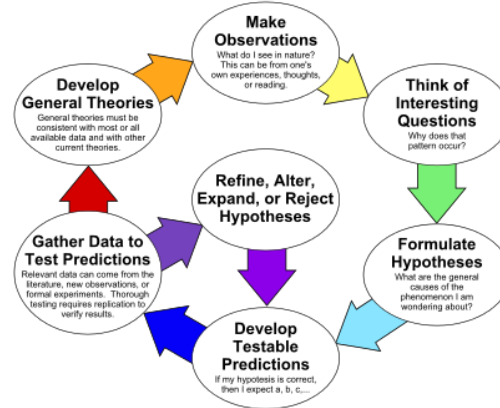
1. a body of methods, rules, and postulates employed by a discipline; a particular procedure or set of procedures
2. the analysis of the principles or procedures of inquiry in a particular field

In the domain of data science, solving problems and answering questions through data analysis is standard practice. Often, data scientists construct a model to predict outcomes or discover underlying patterns, with the goal of gaining insights.

Data Science - using the Scientific Method

- The scientific method is a body of techniques for:
 - investigating phenomena
 - acquiring new knowledge, or
 - correcting and integrating previous knowledge
- To be termed scientific, a method of inquiry is commonly based on empirical or measurable evidence subject to specific principles of reasoning
 - Wikipedia, Scientific Method

The Scientific Method as an Ongoing Process



Applying methodology to Data Science

- We need a data science methodology to provide us with a guiding strategy... regardless of particular technologies & approaches
- Data science methodology presented in the diagram on the next slide
 - Bears similarities to recognized methodologies for data mining, notably CRISP-DM*.
 - Updated for new considerations in data science

Cross Industry Standard Process for Data Mining

- In the next few slides, we will look at a number of different representations of the Data Science methodology process
- Also look at the notes to this slide

Applying methodology to Data Science

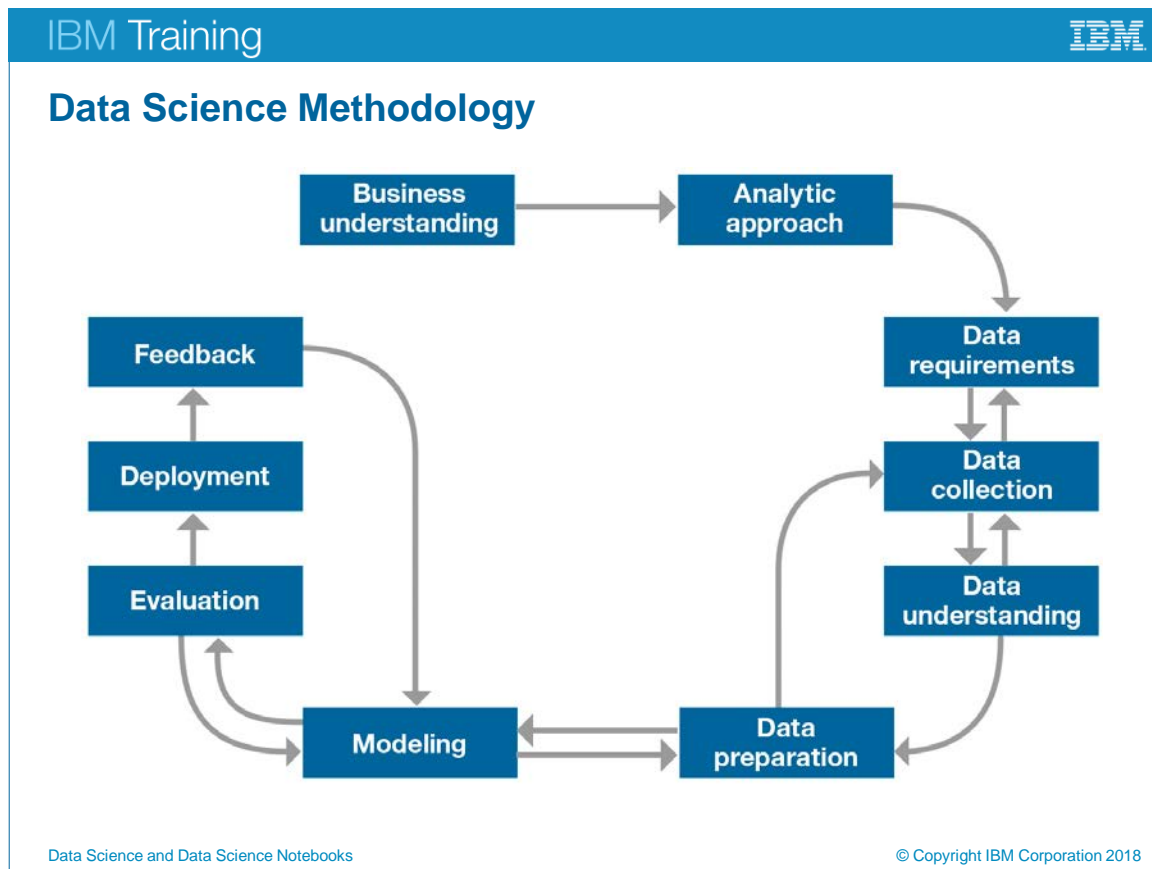
References:

- Brachman, R. & Anand, T., "The process of knowledge discovery in databases," in Fayyad, U. et al., eds., *Advances in knowledge discovery and data mining*, AAAI Press, 1996 (pp. 37-57).
- SAS Institute, <http://en.wikipedia.org/wiki/SEMMA>, http://www.sas.com/en_us/software/analytics/enterprise-miner.html, http://www.sas.com/en_gb/software/small-midsize-business/desktop-data-mining.html.
- http://en.wikipedia.org/wiki/Cross_Industry_Standard_Process_for_Data_Mining, <http://the-modeling-agency.com/crisp-dm.pdf>.
- Ballard, C., Rollins, J., Ramos, J., Perkins, A., Hale, R., Dorneich, A., Milner, E., and Chodagam, J.: *Dynamic Warehousing: Data Mining Made Easy*, IBM Redbook SG24-7418-00 (Sep 2007), pp. 9-26.
- Gregory Piatetsky, *CRISP-DM, still the top methodology for analytics, data mining, or data science projects*, Oct. 28, 2014, <http://www.kdnuggets.com/2014/10/crisp-dm-top-methodology-analytics-data-mining-data-science-projects.html>.

Diagrams for Data Science Methodology

- The goal of the next three slides is to show you some different representations of Data Science Methodology, aka Data Science Lifecycle
- Note the steps and the iteration between the steps
- We will then look at the *specific skills* needed for the Data Scientist
- And, afterwards, compare the skills of the Data Scientist with those expected of the Business Analyst

Diagrams for Data Science Methodology



Data Science Methodology

Source of graphic:

- John Rollins, Data Scientist, IBM Analytics, "Why we need a methodology for data science" <http://www.ibmbigdatahub.com/blog/why-we-need-methodology-data-science>
- This is one of a number of articles on the IBM Big Data & Analytics Hub at <http://www.ibmbigdatahub.com>

1. Here, we see that our data science methodology:

- Begins with understanding the business problem (not with data).
- Is highly iterative.
- Does not "end." As long as the business problem is relevant, we continue to refine the model (including data requirements and data preparation) based on feedback and then re-implement and refresh the model.

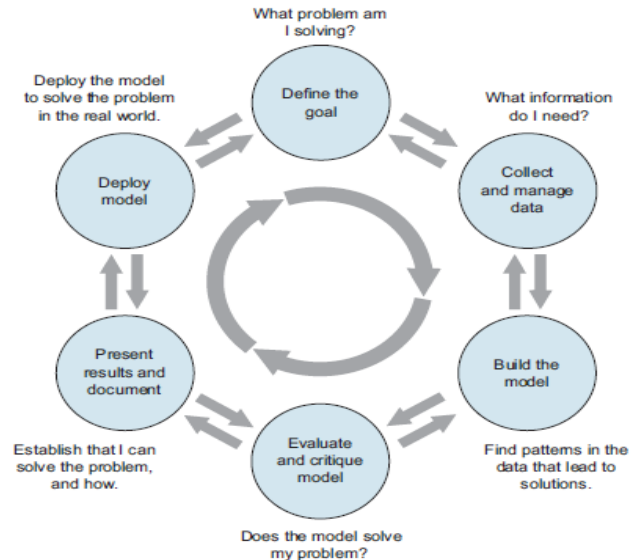
2. This is a general strategy for problem-solving:

- It does not depend on particular technologies or tools.
- It provides a "top-down" approach, but is conceptually consistent with "bottom-up" approach (i.e., we still need to understand data, prepare data, have an analytic approach, build a model, evaluate it, deploy it, and refine it).

Data science project lifecycle

Another description of a data science project lifecycle:

- “The lifecycle of a data science project: loops within loops”
 - Zumel, N., & Mount, J. *Practical data science with R*. Shelter Island, NY: Manning Publications, p. 7.

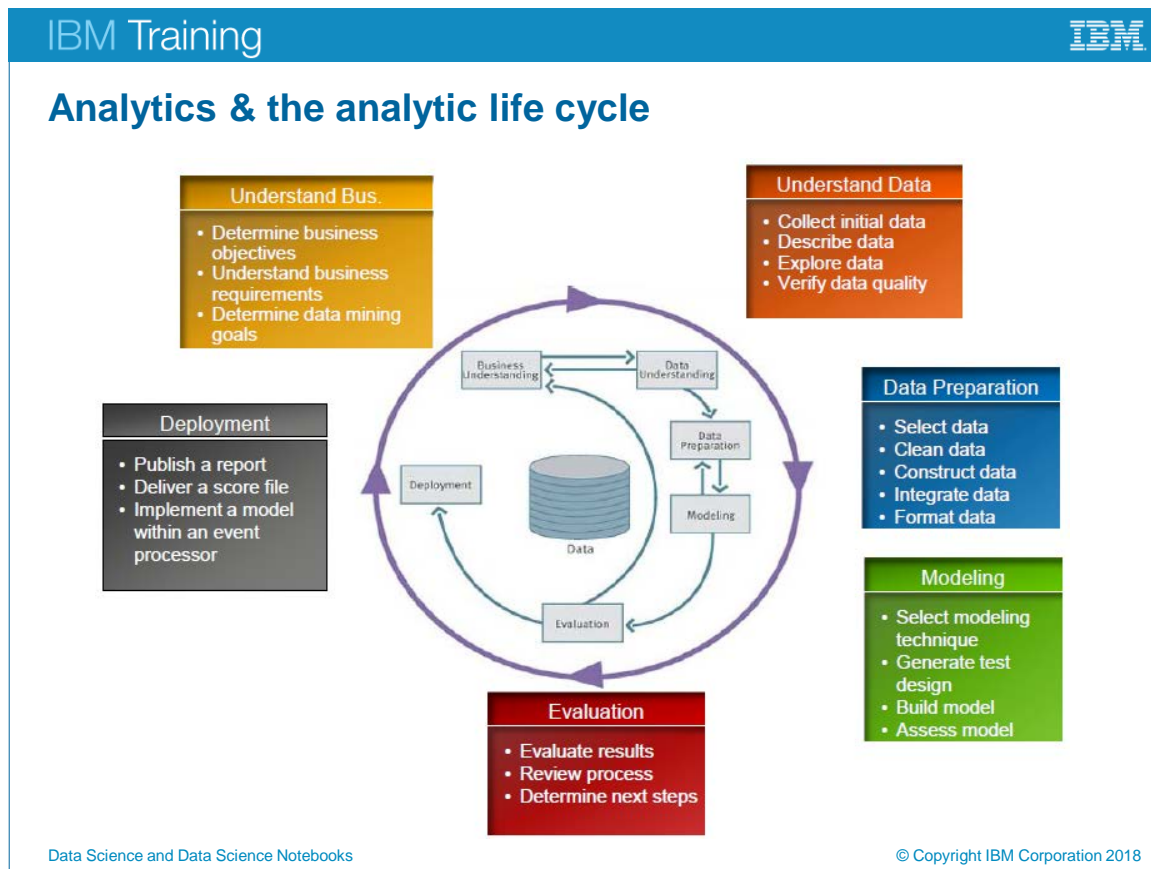


Data Science and Data Science Notebooks

© Copyright IBM Corporation 2018

Data science project lifecycle

Source for graphic: Zumel, N., & Mount, J. *Practical data science with R*. Shelter Island, NY: Manning Publications, p. 7.



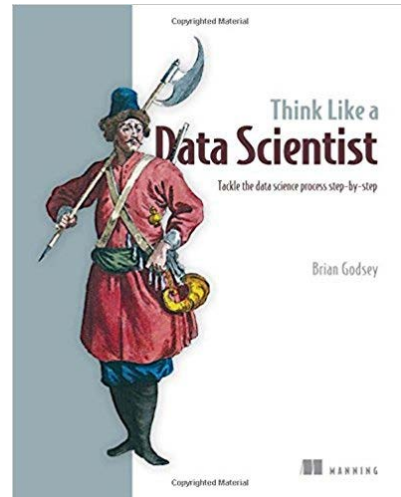
Analytics & the analytic life cycle

Source: **Greg Battas, HPE**

Here the cycle begins with **Understanding Business**.

Systematic approach in Three Phases

- Brian Godsey in *Think Like a Data Scientist* organizes a data science project into three phases:
 - The first phase is **preparation**-time and effort spent gathering information at the beginning of a project can spare big headaches later.
 - The second phase is **building the product**, from planning through execution, using what you learned during the preparation phase and all the tools that statistics and software can provide
 - The third and final phase is **finishing**-delivering the product, getting feedback, making revisions, supporting the product, and wrapping up the project



Systematic approach in Three Phases

The book is: Godsey, B. (2017). *Think like a Data Scientist: Tackle the data science process step-by-step*. Shelter Island, NY: Manning. Amazon \$41. (With all Manning Publications, if you buy the print book, you get the PDF and pBook as free downloads).

“*Think Like a Data Scientist* presents a step-by-step approach to data science, combining analytic, programming, and business perspectives into easy-to-digest techniques and thought processes for solving real world data-centric problems.”

“This book attempts to foresee that future in which the most common, rote, mechanical tasks of data science are stripped away, and we are left with only the core: applying the scientific method to data sets in order to achieve a project’s goals. This, the process of data science, involves software as a necessary set of tools, just as a traditional scientist might use test tubes, flasks, and a Bunsen burner. But, what matters is what’s happening on the inside: what’s happening to the data, what results we get, and why.” - pp. xvii-xviii.

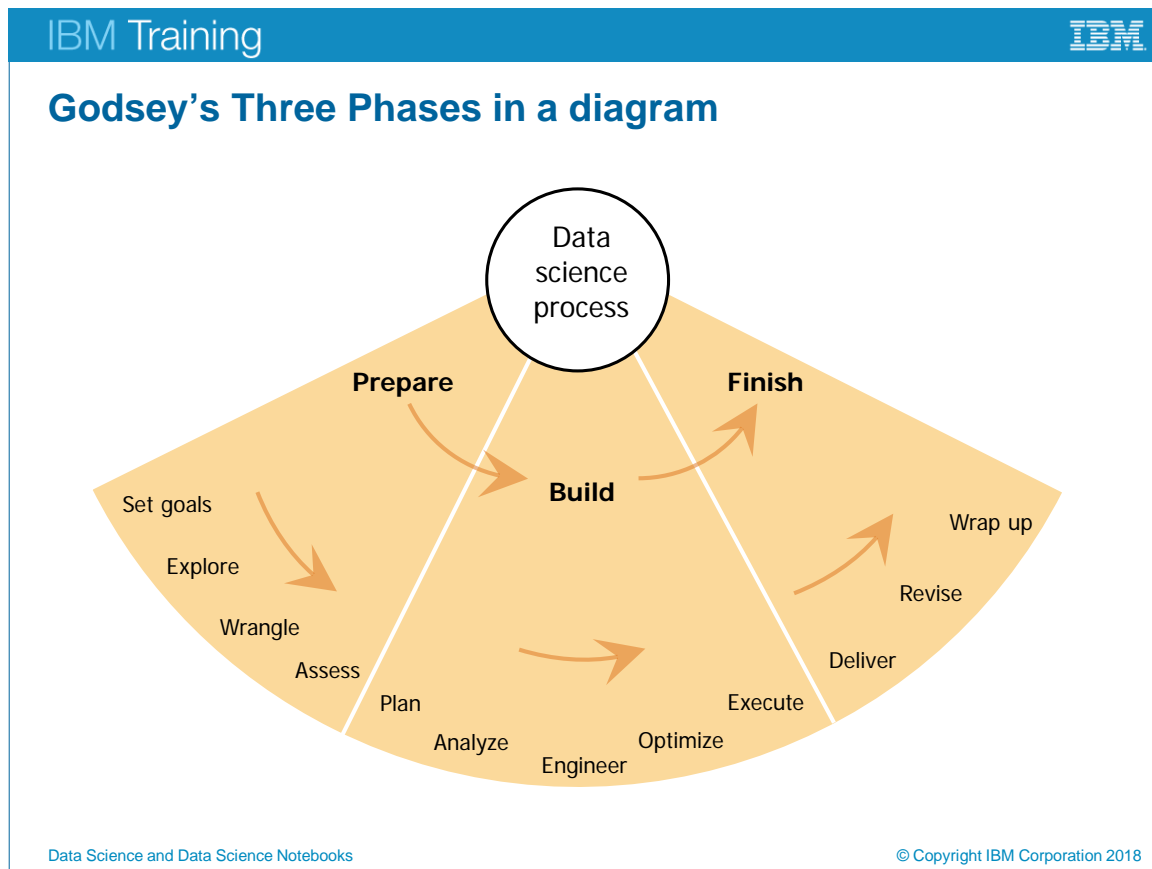
You can read two chapters for free from <https://www.manning.com/books/think-like-a-data-scientist>:

Chapter 1: Philosophies of data science 3

- 1.1 Data science and this book 5
- 1.2 Awareness is valuable 7
- 1.3 Developer vs. data scientist 8
- 1.4 Do I need to be a software developer? 10
- 1.5 Do I need to know statistics? 11
- 1.6 Priorities: knowledge first, technology second, opinions third 12
- 1.7 Best practices 13
- 1.8 Reading this book: how I discuss concepts 17

Chapter 6: Developing a Plan

- 6.1 What have you learned?
- 6.2 Reconsidering expectations and goals
- 6.3 Planning
- 6.4 Communicating new goals
- 6.5 Exercises
- 6.6 Summary



Godsey's Three Phases in a diagram

Source: Godsey, B. (2017). *Think like a Data Scientist: Tackle the data science process step-by-step*. Shelter Island, NY: Manning, p. 6.

So then, what is Data Science?

- Tools are only part of the picture
-- Data Science is much more than the tools that are used
- Data Science is a **mode of thinking**
- And requires an **orchestration of talents**

Source:

Godsey, B. (2017). *Think like a Data Scientist: Tackle the data science process step-by-step*. Shelter Island, NY: Manning Publications, p. 4.



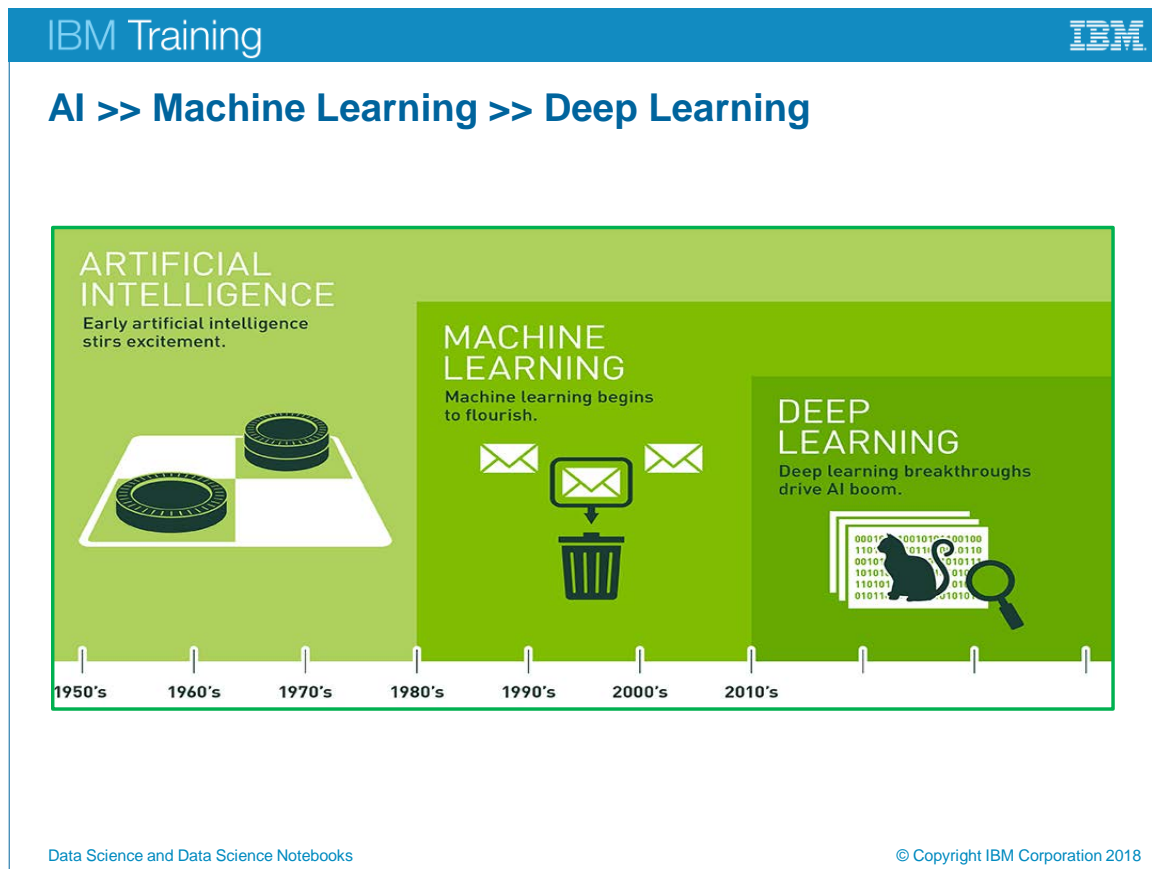
Data Science and Data Science Notebooks

© Copyright IBM Corporation 2018

So, then what is Data Science?

Investigate some humor around Data Science; try a Google search: **data science cartoons**

Also: Dilbert's 20 funniest cartoons on Big Data at <http://bigdata-madesimple.com/dilberts-20-funniest-cartoons-on-big-data/>



AI >> Machine Learning >> Deep Learning

Since an early flush of optimism in the 1950s, smaller subsets of artificial intelligence - first machine learning, then deep learning, a subset of machine learning - have created every larger disruptions.

Artificial intelligence is the future. Artificial intelligence is science fiction. Artificial intelligence is already part of our everyday lives. All those statements are true, it just depends on what flavor of AI you are referring to. The easiest way to think of their relationship is to visualize them as concentric circles with AI - the idea that came first - the largest, then machine learning - which blossomed later, and finally deep learning - which is driving today's AI explosion - fitting inside both.

- What's the Difference Between Artificial Intelligence, Machine Learning, and Deep Learning?
<https://opendatascience.com/blog/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/>

The Work of the Data Scientist

- Drew Conway's Venn diagram
- The Data Scientist's Skill Set
 - Role of Business Analyst vs Data Scientist
 - Education
- The Art of Data Science in 5 Steps

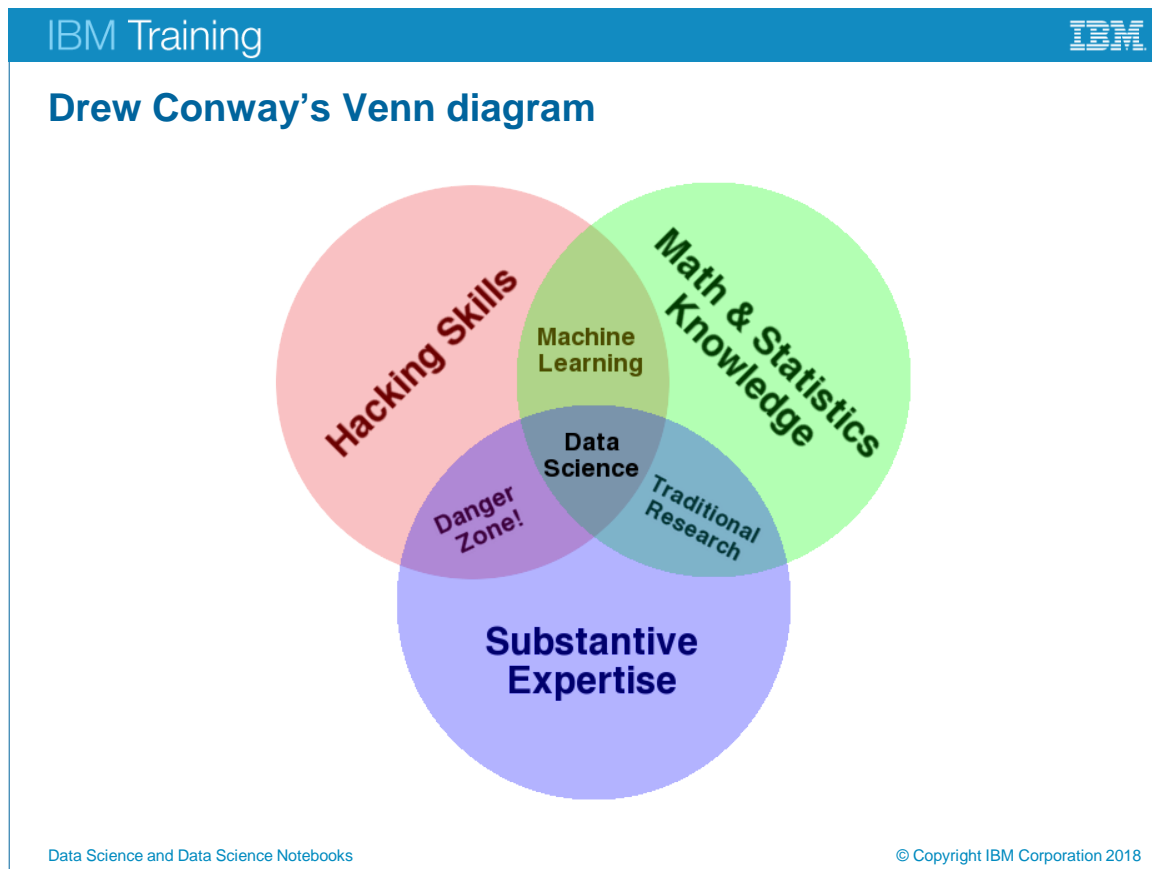


Data Science and Data Science Notebooks

© Copyright IBM Corporation 2018

The Work of the Data Scientist

We will look at several Venn diagrams (generally derived from Conway's initial synthesis). Each has its merits.



Drew Conway's Venn diagram

Source:

<http://drewconway.com/zia/2013/3/26/the-data-science-venn-diagram>

This diagram is the classic Drew Conway diagram that everyone talks about. It starts the discussion on the question of *What is Data Science?*

How to read the Data Science Venn Diagram:

- <http://drewconway.com/zia/2013/3/26/the-data-science-venn-diagram>
- <http://www.dataists.com/2010/09/the-data-science-venn-diagram>

The “official” notes (by Conway himself):

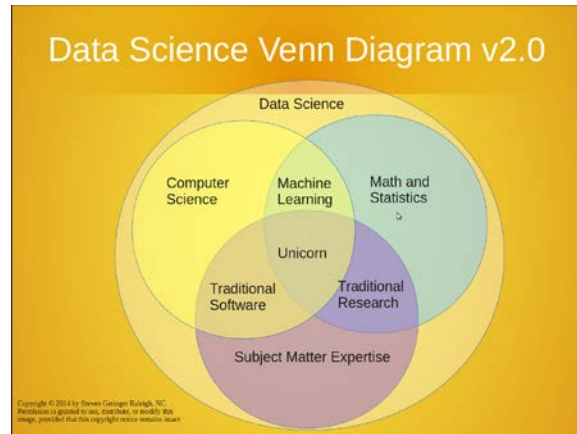
- The primary colors of data: **hacking skills**, **math and stats knowledge**, and **substantive expertise**
- Previously, we spent a lot of time talking about "where" a course on data science might exist at a university. The conversation was largely rhetorical, as everyone was well aware of the inherent interdisciplinary nature of these skills; but then, why have I highlighted these three? First, none is discipline specific, but more importantly, each of these skills are on their own very valuable, but when combined with only one other are at best simply not data science, or at worst downright dangerous.

- For better or worse, data is a commodity traded electronically; therefore, in order to be in this market you need to speak hacker. This, however, does not require a background in computer science-in fact-many of the most impressive hackers I have met never took a single CS course. Being able to manipulate text files at the command-line, understanding vectorized operations, thinking algorithmically; these are the hacking skills that make for a successful data hacker.
- Once you have acquired and cleaned the data, the next step is to actually extract insight from it. In order to do this, you need to apply appropriate math and statistics methods, which requires at least a baseline familiarity with these tools. This is not to say that a PhD in statistics is required to be a competent data scientist, but it does require knowing what an ordinary least squares regression (http://en.wikipedia.org/wiki/Ordinary_least_squares) is and how to interpret it.
- In the third critical piece-substance-is where thoughts on data science diverge from most of what has already been written on the topic. Data plus math and statistics only gets you machine learning, which is great if that is what you are interested in, but not if you are doing data science. Science is about discovery and building knowledge, which requires some motivating questions about the world and hypotheses that can be brought to data and tested with statistical methods. On the flip-side, substantive expertise plus math and statistics knowledge is where most traditional researcher falls. Doctoral level researchers spend most of their time acquiring expertise in these areas, but very little time learning about technology. Part of this is the culture of academia, which does not reward researchers for understanding technology. That said, there are many young academics and graduate students that are eager to bucking that tradition.
- Finally, a word on the hacking skills plus substantive expertise danger zone. This is where people can be placed who, "know enough to be dangerous," and is the most problematic area of the diagram. In this area people who are perfectly capable of extracting and structuring data, likely related to a field they know quite a bit about, and probably even know enough R to run a linear regression and report the coefficients; but they lack any understanding of what those coefficients mean. It is from this part of the diagram that the phrase "lies, damned lies, and statistics" emanates, because either through ignorance or malice this overlap of skills gives people the ability to create what appears to be a legitimate analysis without any understanding of how they got there or what they have created. Fortunately, it requires near willful ignorance to acquire hacking skills and substantive expertise without also learning some math and statistics along the way. As such, the danger zone is sparsely populated, however, it does not take many to produce a lot of damage.

Hopefully this brief illustration has provided some clarity into what data science is and what it takes to get there. By considering these questions at a high level it prevents the discussion from degrading into minutia, such as specific tools or platforms, which I think hurts the conversation.

Drew Conway's Venn diagram (adapted)

- The center is marked "Unicorn"
- real Data Scientists can be as hard to find as unicorns.
- A team of people with complimentary skills is the course of action for most data driven organizations.
- Some individuals might posses Computer Science, Statistics, and Subject Matter Expertise. They are just very hard to find.



Data Science and Data Science Notebooks

© Copyright IBM Corporation 2018

Drew Conway's Venn diagram (adapted)

There have been a number of attempts to get our collective brains around all the skill sets needed to effectively do Data Science.

Here are two...

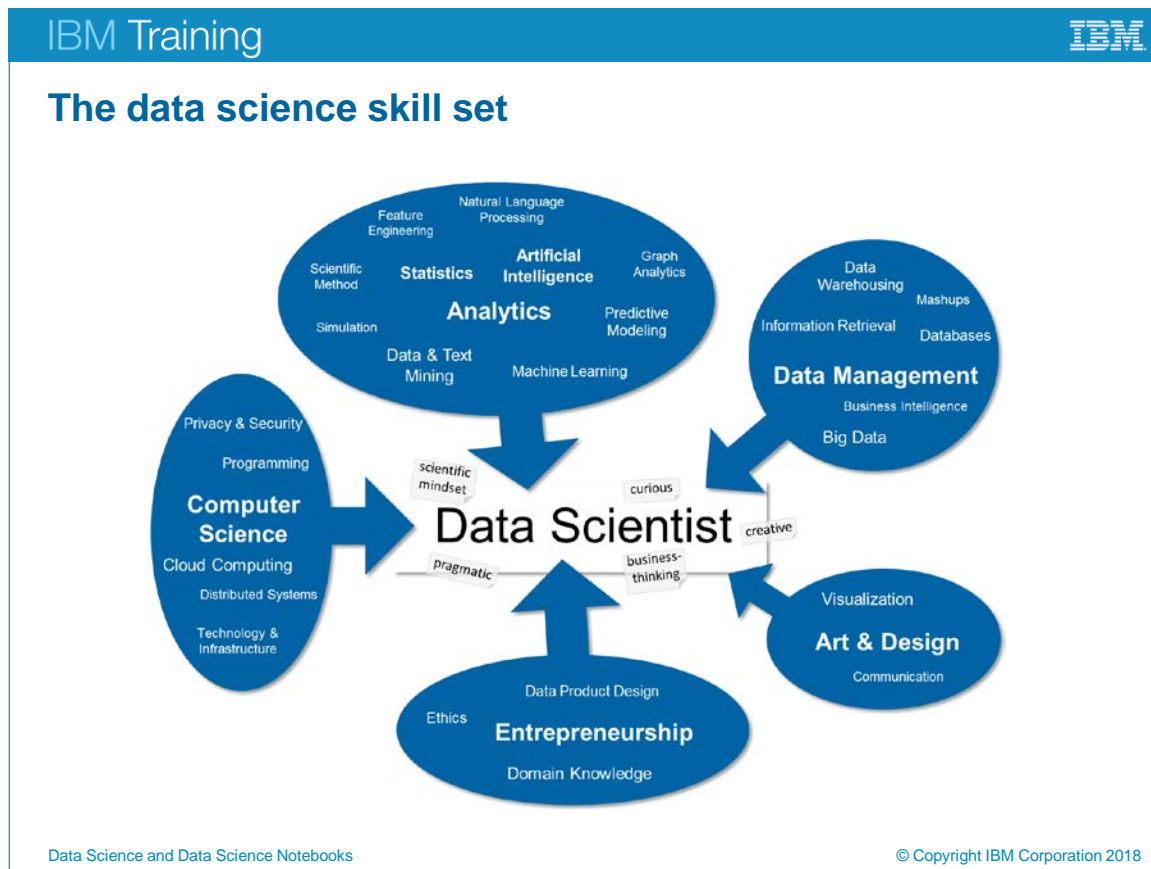
1. <http://drewconway.com/zia/2013/3/26/the-data-science-venn-diagram>
2. <http://upload.wikimedia.org/wikipedia/commons/4/44/DataScienceDisciplines.png>

Below is a position on the subject.

The center is marked "Unicorn." This a reference to the discussions in the press and blogosphere indicating that Data Scientists are as hard to find as unicorns. Finally the mindset is changing that a team of people with complimentary skills is the course of action for most data driven organizations. Certainly some individuals might possess Computer Science, Statistics and Subject Matter Expertise. They are just very hard to find. Many Data Scientist job descriptions don't reflect this reality and so these positions go unfilled for six months or more.

This is an Adaptation of the original Data Science Venn Diagram which is licensed under *Creative Commons Attribution-NonCommercial*.

Source: <http://www.anlytcs.com/2014/01/data-science-venn-diagram-v20.html>



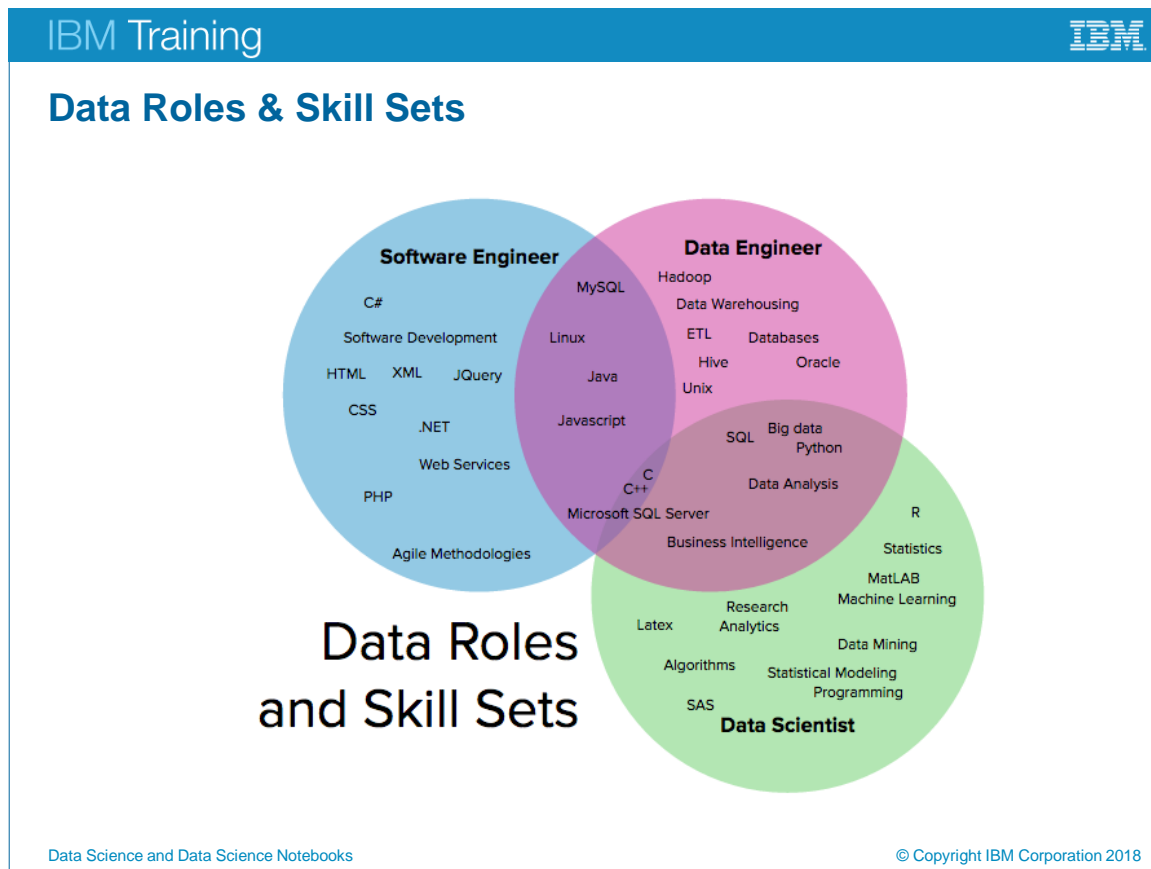
The data science skill set

For us, it is a very useful (and used) tool to explain what data science is *and* what we can do:

- First, it shows the different high-level skills a data scientist should master. This unveils the intrinsic interdisciplinarity of data science (I usually say that an actual data scientist should comprise approximately 80% of these skills, scattered over all 5 blue bubbles). These skills can be trained, usually by earning a degree in one of the top 3 bubbles (with a focus on quantitative analytics) and getting skills in the other bubbles by interdisciplinary work and/or advanced training.
- Second, it shows character traits of a data scientist as grey labels attached to the center of the map. These attitudes are hard to train, but reveal the mindset of a data scientist.
- Third, the history of the field shows up on the map: all the skills have been developed in disciplines of their own right (like, e.g., statistics or artificial intelligence). Data science now is a *unique blend* of these skills (from analytics, engineering and communication) aiming at a *specific goal*: generating (corporate and/or societal) value from (all kinds of) data.

More Information on this skill set map and how we implement it within the Datalab can also be found in a paper “Applied Data Science in Europe.”

This diagram and the accompanying text were posted by Thilo Stadelmann in a blog at <https://blog.zhaw.ch/datascience/the-data-science-skill-set>



Data Roles & Skill Sets

Reference:

- <http://101.datascience.community/2016/11/28/data-scientists-data-engineers-software-engineers-the-difference-according-to-linkedin/>

Software Engineer

A software engineer builds applications and systems. Developers will be involved through all stages of this process from design, to writing code, to testing and review. They are creating the products that create the data. Software engineering is the oldest of these three roles, and has established methodologies and tool sets.

Work includes:

- Frontend and backend development
- Web apps
- Mobile apps
- Operating system development
- Software design

Data Engineer

A data engineer builds systems that consolidate, store, and retrieve data from the various applications and systems created by software engineers. Data engineering emerged as a niche skill set within software engineering. 40% of all data engineers were previously working as a software engineer_ (<https://www.stitchdata.com/resources/reports/the-state-of-data-engineering>), making this the most common career path for data engineers by far.

Work includes:

- Advanced data structures
- Distributed computing
- Concurrent programming
- Knowledge of new & emerging tools: Hadoop, Spark, Kafka, Hive, etc.
- Building ETL/data pipelines

Data Scientist

A data scientist builds analysis on top of data. This may come in the form of a one-off analysis for a team trying to better understand customer behavior, or a machine learning algorithm that is then implemented into the code base by software engineers and data engineers.

Work includes:

- Data modeling
- Machine learning
- Algorithms
- Business Intelligence dashboards

Evolving Data Teams

These roles are still evolving. The process of ETL is getting much easier overall as new tools (like Stitch - <https://www.stitchdata.com/>) enter the market, making it easy for software developers to set up and maintain data pipelines. Larger companies are pulling data engineers off the software engineering team entirely in lieu of forming a centralized data team where infrastructure and analysis sit together. In some scenarios data scientists are responsible for both data consolidation and analysis (<http://multithreaded.stitchfix.com/blog/2016/03/16/engineers-shouldnt-write-etl/>)

At this point, there is no single dominant path. But we expect this rapid evolution to continue, after all, data certainly isn't getting any smaller.

Business Analysts vs Data Scientists

Who are they?



BUSINESS ANALYSTS

Research and extract valuable information from structured and unstructured sources to explain historical, current, and future business performance; determine the best analytical models and approaches to present and explain solutions to business users.



DATA SCIENTISTS

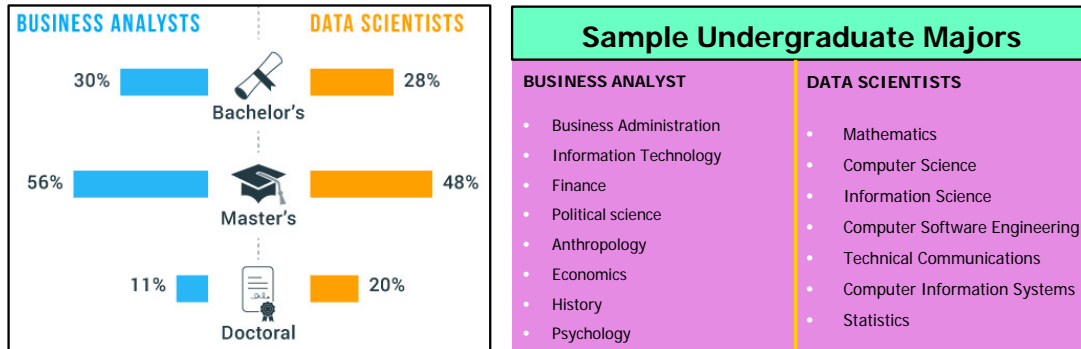
Design, develop, and deploy algorithms through statistical programming that support business decision-making tools; manage large amounts of data; create visualizations to aid in understanding.

Business Analysts vs Data Scientists

From the infographic at: <http://cloudtweaks.com/2016/05/business-analytics-vs-data-science/>

What type of education do Data Scientists have?

- The majority of business analysts come from a variety of backgrounds including business and humanities, while data scientists tend to come from computer science, mathematics, and technology backgrounds



Data Science and Data Science Notebooks

© Copyright IBM Corporation 2018

What type of education do Data Scientists have?

See the infographics at: <http://cloudtweaks.com/2016/05/business-analytics-vs-data-science/>

The art of Data Science in 5 steps

1. Get passionate about Big Data
2. Embrace hands-on learning
3. Learn the language and communicate your insights
4. Learn from people in the industry
5. Keep challenging yourself

“As you set out to learn this field and become a data scientist, you'll find that entering an emerging field is both exciting and overwhelming. As you come across challenges and obstacles, and as the definition of data science seems to change depending on whom you're reading or listening to, the whole endeavor can seem intimidating enough to want to quit. If you're passionate about data and what you can do with it, though, you'll have motivation to keep going.”



Data Science and Data Science Notebooks

© Copyright IBM Corporation 2018

The art of Data Science in 5 steps

From an article on LinkedIn pulse: <https://www.linkedin.com/pulse/learn-art-data-science-five-steps-ronald-van-loon>

“The field of data science is one of the youngest and most exciting fields in the technology sector. In no other industry or field can you combine statistics, data analysis, research, and marketing to do jobs that help businesses make the digital transformation and come to full digital maturity.

Important things to know about Data Science

Handy list of 50 quotes from some of the leading minds in the field

<http://www.datasciencecentral.com/profiles/blogs/50-important-things-you-need-to-know-about-data-science-1>

1. Do data science because you love data

Weaker motivations won't sustain you in the long run.

2. Learn data science by doing it - and then by doing it some more

The word "possessed" rings true here. Longer versions of these quotes reveal a common denominator in these influencers' schedules: they spent, and continue to spend, a lot of time practicing their craft.

3. Don't forget the importance of theory amid all that practice

4. Don't expect to learn data science overnight

Or even over several nights. It's a lifelong learning curve, especially as the field evolves so quickly.

5. Communicate

Learn how to tell a meaningful, understandable story with the insights you draw from your data. This is critical in just about any application of data science.

Important things to know about Data Science

Note that these are **action** verbs: **do**, **learn**, **understand** the importance of theory, **have patience** in learning, and **communicate**.

Now down to the practical steps in the process

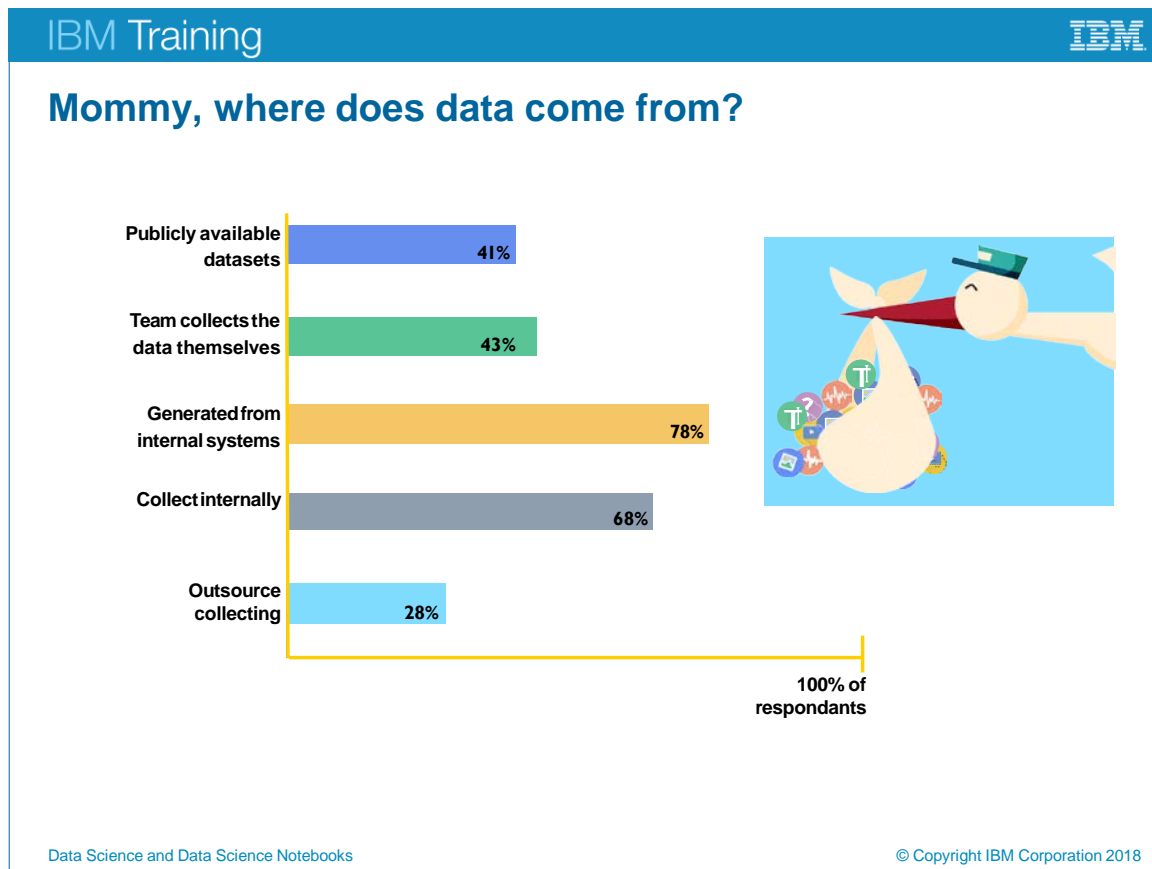
- We will look at the steps in more detail
- And introduce the notion of a **Data Science Notebook**



Data Science and Data Science Notebooks

© Copyright IBM Corporation 2018

Now down to the practical steps in the process



Mommy, where does data come from?

While the majority of data scientists utilize data generated from internal systems (78%), over half of them get data from at least 3 different sources including manual internal collection, publicly available datasets, and outsourcing.

Finally, while 48% list collecting data as one of their 3 least favorite tasks, 43% of data scientists are doing just that - collecting data themselves.

While there's no shortage of data, access to quality data is definitely an issue. Specifically when it comes to AI projects,

51% of respondents listed issues related to quality data ("getting good training data" or "improving the quality of your training dataset") as the biggest bottleneck to successfully completing projects.

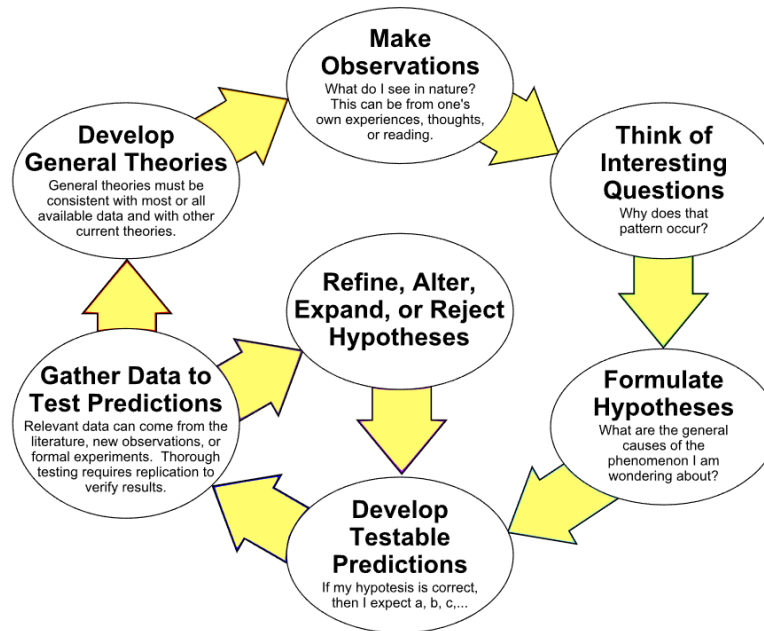
Reference:

For this year's survey, CrowdFlower surveyed 179 data scientists globally representing a balance of company ranging in size from <100 to 10,000+. A variety of industries were represented as well, with a slightly weighted emphasis on 'technology' - representing 40% of respondents. The survey was conducted in February and March of 2017.

- https://visit.crowdfunder.com/rs/416-ZBE-142/images/CrowdFlower_DataScienceReport.pdf

The scientific method - Generic

- An ongoing & iterative process



Data Science and Data Science Notebooks

© Copyright IBM Corporation 2018

The scientific method - Generic

By ArchonMagnus (Own work) [CC BY-SA 4.0 (<http://creativecommons.org/licenses/by-sa/4.0>)], via Wikimedia Commons

Generally speaking, both traditional scientists and data scientists ask questions and/or define a problem, collect and leverage data to come up with answers or solutions, test the solution to see if the problem is solved, and iterate as needed to improve on, or finalize the solution.

Data Notebooks: Tools for data scientists

- Interactive notebooks are experiencing a rise in popularity
 - Traditionally, notebooks have been used to document research and make results reproducible, simply by re-running the notebook on source data.
 - Notebooks are typically used by data scientists for quick exploration tasks. In that regard they offer a number of advantages over any local scripts or tools. When properly set up by the organization, a notebook offers direct connections to all necessary sources of data, without additional effort on the part of the user
- The major notebooks currently used by Data Scientists are:
 - **Jupyter** - based on the **iPython** notebook - and named after 3 of the 60 or so language-kernels now available for it: **Julia**, **Python**, and **R**
 - Apache **Zeppelin** - distributed with ODPI - emphasized by Hortonworks
 - Jupyter notebook within the IBM's **Watson Studio (formerly DSX)** platform. - *proprietary, with a free trial available*
 - **Knitr** - affiliated with Foundation for Open Access Statistics (FOAS) - for R

Data Science and Data Science Notebooks

© Copyright IBM Corporation 2018

Data Notebooks: Tools for data scientists

Data science is inherently an exploratory and creative process because there is usually neither a definitive answer to the problem at hand nor a well-defined approach to reaching one. Data scientists research problems, explore data, visualize patterns across data and use their experience and judgment to choose parameters and processes that may be relevant to the specific problem at hand. This makes sharing and collaboration a critical activity that enables teams of data scientists to build on each other's knowledge and to produce the overall best results.

As data science has evolved over time with big data, new techniques and technologies have emerged. This change is reflected in the background and training of the data scientists across organizations.

References:

- What is a Data Science Workbench and Why Do Data Scientists Need One? (August 17, 2017)
<https://hortonworks.com/blog/data-science-workbench-data-scientists-need-one/>
- The Rise of Data Science Notebooks
<https://www.datanami.com/2016/05/04/rise-data-science-notebooks/>

- A comprehensive comparison of Jupyter vs. Zeppelin
<https://www.linkedin.com/pulse/comprehensive-comparison-jupyter-vs-zeppelin-hoc-q-phan-mba->
- <http://jupyter.org/>
- <https://zeppelin.apache.org/> & <https://www.zepl.com>
- <https://datascience.ibm.com/>

The Data Science Pipeline - Summarization

The five key steps in Data Analysis:

1. **Acquisition:** Acquire data from a variety of sources, including RDBMS systems, NoSQL and document store, webscraping, Data Lakes, HDFS, etc.
2. **Exploration & Understanding:** Understanding the data that you will use and how it was collected, often requiring significant exploration
3. **Munging, Wrangling, & Manipulation:** Single most time-consuming and important step in the pipeline - data is rarely in the needed form for analysis
4. **Analytis & Modeling:** The fun part where the data scientist explores the statistical relationship between the variables in the data and uses a bag of machine learning tricks to cluster, categoralize, classify Predictive Models
5. **Communicate & Operationalize:** Give the data back in a compelling form and structure - one-off report, scalable web product, interactive data, ...

The Data Science Pipeline - Summarization

IBM Training

IBM

We will use the Anaconda distribution of Python

Data Science and Data Science Notebooks

© Copyright IBM Corporation 2018

We will use the Anaconda distribution of Python

<https://www.anaconda.com/anaconda-overview>

From <https://www.anaconda.com/download>, you would install Anaconda 5 (or later) for your computer - Windows, MacOS, or Linux - 64-bit installer.

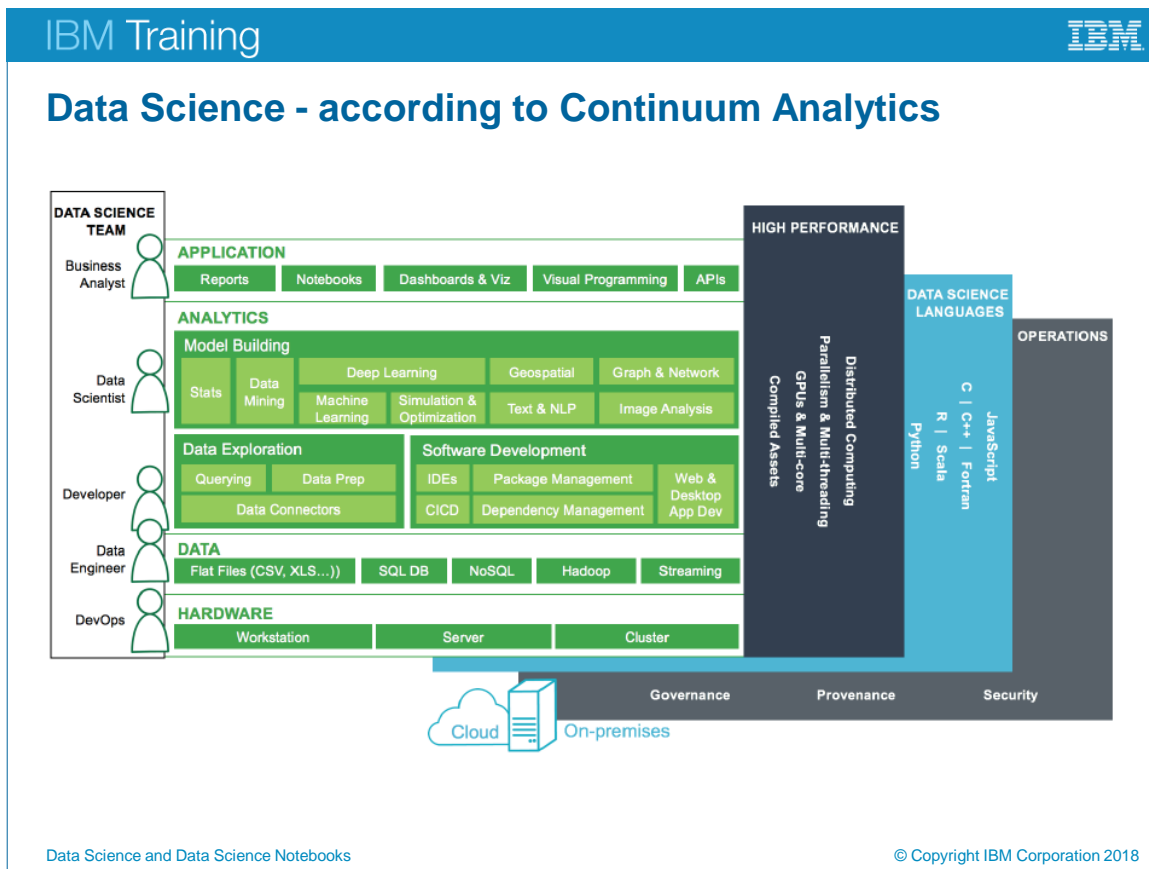
If your Python 3 is weak? ...and new to Jupyter?

- Rather than just reading about what happens when you type some Python, try out the commands for yourself
- You should have Python 3 running as you read, try out every snippet of code, and observe what the language does to respond
 - Reading about a computer language won't teach you the language. You can teach yourself the language by working through all the examples and lab exercises, and then using what you've learning on you own problems later.
 - Ask yourself: "What would happen if ... ?", "How can I accomplish that?" Just try things! And this process is not only more fun than passively acciumulating facts - and it is fr more effective.
 - Make mistakes! - not necessarily deliberately - as each mistake provide the opportunity to learn something new.
 - Good practice: keep a log! That is what any scientist does, and as we are new data scientists, our log is Jupyter Notebook.

If your Python 3 is weak? ...and new to Jupyter?

Getting started with Jupyter

- Use Anaconda 5 to install **Jupyter** on your computer from Anaconda
 - Windows, MacOS, & Linux versions at <https://www.anaconda.com/download/>
 - Install Python 3.6 (or later) version 64-bit
 - The open source version of Anaconda is a high performance distribution of Python and R and includes over 100 of the most popular Python, R, and Scala packages for data science
 - Additionally, you'll have access to over 720 packages that can easily be installed with **conda**, the renowned package, dependency, and environment manager, that is included in Anaconda - from Continuum Analytics
- Start using Jupyter
 - Quick start: <https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/>
 - Tutorials: <http://bebi103.caltech.edu.s3-website-us-east-1.amazonaws.com/2015/tutorials.html>
 - Other tutorials: <http://www.kdnuggets.com/2016/04/top-10-ipython-nb-tutorials.html>
 - **Plus many YouTube videos**



Data Science - according to Continuum Analytics

<https://know.anaconda.com/journey-to-ODS-whitepaper.html>

Anaconda is software from Anaconda (formerly known as Continuum Analytics). This graphic is the final image in an 8 page whitepaper on Open Data Science (ODS), available at <https://know.anaconda.com/journey-to-ODS-whitepaper.html>

Anaconda is the leading modern open source analytics platform powered by Python. Anaconda enables modernization to Open Data Science as a platform that embraces the entire ODS ecosystem - Python, R, Java, Scala, Hadoop and more - across the entire modern analytics stack from desktop to server to clusters and cloud for enterprises. Anaconda makes it easy to:

- Create, collaborate and deploy Data Science solutions
- Leverage modern architectures and frameworks
- Set up and manage open source

Python libraries for Data Analysis

Standard Python libraries useful for Data Analysis with Python are:

- **Pandas** - for data munging and preparation
- **NumPy** - abundance of useful features for operations on n-arrays and matrices in Python
- **SciPy** - library of software for engineering and science
- **Matplotlib** - tailored for the generation of simple and powerful visualizations
- **Statsmodels** - data exploration by using methods of estimation of statistical models
- **scikit-learn** - concise & consistent interface to the common ML algorithms
- **Seaborn** - visualization of statistical models; based & highly dependent on Matplotlib
- **Bokeh** - is aimed at interactive visualizations; independent of Matplotlib - main focus is interactivity, with presentation via modern browsers in the style of Data-Driven Documents
- **Plotly** - web-based toolbox for building visualizations, exposing APIs to Python, etc.
- **Theano / TensorFlow / Keras**
- **NLTK** - Natural Language Toolkit - tasks of symbolic & statistical NL processing
- **Gensim / Scrapy**

Python libraries for Data Analysis

References:

- Top 15 Python Libraries for Data Science in 2017
<http://www.kdnuggets.com/2017/06/top-15-python-libraries-data-science.html>
- Ranked: The Best Python Packages for Data Science
<http://www.kdnuggets.com/2017/05/best-python-packages-data-science.html>

Pandas

- Author: Wes McKinney
 - Book: *Python for Data Analysis*, 2nd ed. (O'Reilly: Sebastopol, October 2017)
 - 10-minute tour of Pandas: <https://vimeo.com/59324550>
- What problem does Pandas solve (from <http://pandas.pydata.org>):
 - Python has long been great for data munging and preparation, but less so for data analysis and modeling. *pandas* helps fill this gap, enabling you to carry out your entire data analysis workflow in Python without having to switch to a more domain specific language like R.
 - Combined with the excellent [IPython](#) toolkit and other libraries, the environment for doing data analysis in Python excels in performance, productivity, and the ability to collaborate.
 - *pandas* does not implement significant modeling functionality outside of linear and panel regression; for this, look to [statsmodels](#) and [scikit-learn](#). More work is still needed to make Python a first class statistical modeling environment, but we are well on our way toward that goal.

The tools of statistics

- The tools of statistics include:
 - **Data collection:** We will use data from a large national survey that was designed explicitly with the goal of generating statistically valid inferences about the U.S. population.
 - **Descriptive statistics:** We will generate statistics that summarize the data concisely, and evaluate different ways to visualize data.
 - **Exploratory data analysis:** We will look for patterns, differences, and other features that address the questions we are interested in. At the same time we will check for inconsistencies and identify limitations.
 - **Estimation:** We will use data from a sample to estimate characteristics of the general population.
 - **Hypothesis testing:** Where we see apparent effects, like a difference between two groups, we will evaluate whether the effect might have happened by chance.
- By performing these steps with care to avoid pitfalls, we can reach conclusions that are more justifiable and more likely to be correct

Data Science and Data Science Notebooks

© Copyright IBM Corporation 2018

The tools of statistics

These notes come from: Downey, A. B. (2014). *Think stats: Exploratory data analysis in Python*. Sebastopol, CA: O'Reilly.

Hardware & software for Data Science

- Data Science algorithms are processing intensive
 - CPUs & multiple cores
 - GPUs
- This is why you will hear discussion about GPUs from companies such as NVidia and AMD
- Specialized algorithms, implemented with interfaces from Python and R, are state of the art
 - Google's TensorFlow
 - Keras
 - Theano
 - ...

Hardware & software for Data Science

A few paragraphs on the hardware side from: Mueller, J. P., & Massaron, L. (2017). *Algorithms for dummies*. Hoboken, NJ: Wiley.

General-purpose processors, CPUs, started out as a means to solve problems using algorithms. However, their general-purpose nature also means that a CPU can perform a great many other tasks, such as moving data around or interacting with external devices. A general-purpose processor does many things well, which means that it can perform the steps required to complete an algorithm, but not necessarily fast. In fact, owners of early general-purpose processors could add math coprocessors (special math-specific chips) to their systems to gain a speed advantage (see <http://www.computerhope.com/jargon/m/mathcopr.htm> for details).

Today, general-purpose processors have the math coprocessor embedded into them, so when you get an Intel i7 processor, you actually get multiple processors in a single package.

You may wonder why this section mentions Graphics Processing Units (GPUs). After all, GPUs are supposed to take data, manipulate it in a special way, and then display a pretty picture onscreen. Any computer hardware can serve more than one purpose. It turns out that GPUs are particularly adept at performing data transformations, which is a key task for solving algorithms in many cases. A GPU is a special-purpose processor, but one with capabilities that lend themselves to faster algorithm execution. It shouldn't surprise you to discover that people who create algorithms spend a lot of time thinking outside the box, which means that they often see methods of solving issues in nontraditional approaches.

The point is that CPUs and GPUs form the most commonly used chips for performing algorithm-related tasks. The first performs general-purpose tasks quite well, and the second specializes in providing support for math-intensive tasks, especially those that involve data transformations. Using multiple cores makes parallel processing (performing more than one algorithmic step at a time) possible. Adding multiple chips increases the number of cores available. Having more cores adds speed, but a number of factors keeps the speed gain to a minimum. Using two i7 chips won't produce double the speed of just one i7 chip.

A math coprocessor and a GPU are two examples of common special-purpose chips in that you don't see them used to perform tasks such as booting the system. However, algorithms often require the use of uncommon special-purpose chips to solve problems. ... Spending a little time looking around can show you all sorts of interesting chips, such as the new artificial neurons that IBM is working on (see the story at <http://www.computerworld.com/article/3103294/computer-processors/ibm-creates-artificial-neurons-from-phase-change-memory-for-cognitive-computing.html>). Imagine performing algorithmic processing using memory that simulates the human brain. It would create an interesting environment for performing tasks that might not otherwise be possible today.

Neural networks, a technology that is used to simulate human thought and make deep learning techniques possible for machine learning scenarios, are now benefitting from the use of specialized chips, such as the Tesla P100 from NVidia (see the story at <https://www.technologyreview.com/s/601195/a-2-billion-chip-to-accelerate-artificial-intelligence/> for details). These kinds of chips not only perform algorithmic processing extremely fast, but learn as they perform the tasks, making them faster still with each iteration. Learning computers will eventually power robots that can move (after a fashion) on their own, akin to the robots seen in the movie *I Robot* (see one such robot described at <http://www.cbsnews.com/news/this-creepy-robot-is-powered-by-a-neural-network/>). There are also special chips that perform tasks such as visual recognition (see <https://www.technologyreview.com/s/537211/a-better-way-to-build-brain-inspired-chips/> for details).

Notebooks & Workbenches for Data

- Jupyter Notebook (formerly IPython)
- Apache Zeppelin
- BeakerX (Beaker Notebook)
- IBM Watson Studio - a platform for data scientists
- Databricks Unified Analytics Platform
- Cloudera Data Science Workbench
- MapR Data Science Refinery
- Dataiku's Data Science Studio
- ...

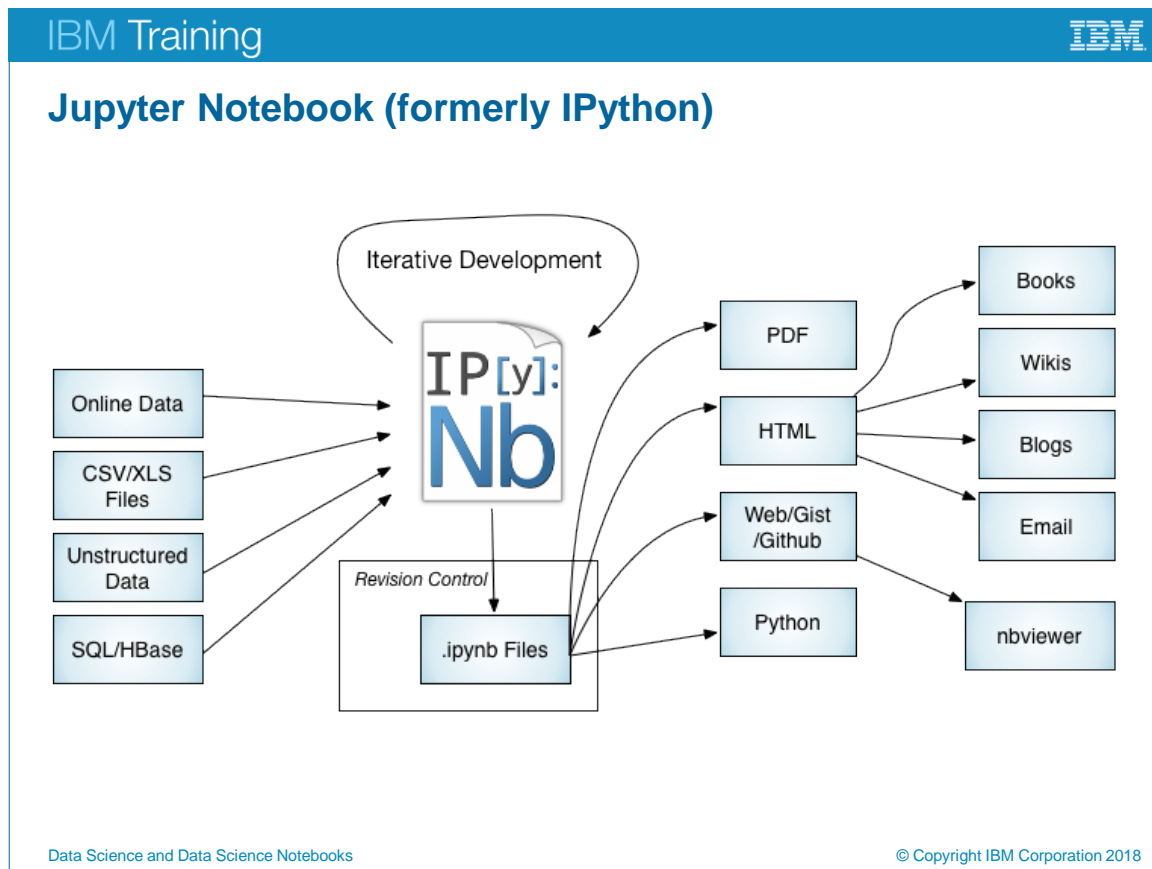
Data Science and Data Science Notebooks

© Copyright IBM Corporation 2018

Notebooks & Workbenches for Data

References:

- <http://beakernotebook.com>
- Beaker Extensions for Jupyter Notebook:
 - <http://beakerx.com>
 - <http://github.com/twosigma/beakerx>



Jupyter Notebook (formerly IPython)

Reference:

- Jupyter, Zeppelin, Beaker: The Rise of the Notebooks
<https://opendatascience.com/blog/jupyter-zeppelin-beaker-the-rise-of-the-notebooks/>

Apache Zeppelin - a short overview

- Web-based notebook that enables data-driven, interactive data analytics and collaborative documents with SQL, Scala, and more - Supports over 20+ different interpreters - <https://zeppelin.apache.org/>
 - Pluggable visualization using the Helium framework
 - Latest version (Apache Zeppelin 0.7) supports:
 - Latest version of [Apache Spark 2.1.0](#); Integrated [Matplotlib](#) with Python & Pyspark interpreter; [Conda](#) is available; can use Apache Beam, Scio, and Apache Pig as backend interpreters.
- Usage:
 - **Single User:** Local Spark, 6 Built-in visualizations, display system, dynamic forms, multiple backends supported,
 - **Multi-User:** Has multi-user support w/ LDAP. Can be configured for a YARN cluster. Collaborative Mode is the default, but you can switch to a Personal Mode.
- Interactive demonstration: <https://www.zepl.com/>

Data Science and Data Science Notebooks

© Copyright IBM Corporation 2018

Apache Zeppelin - a short overview

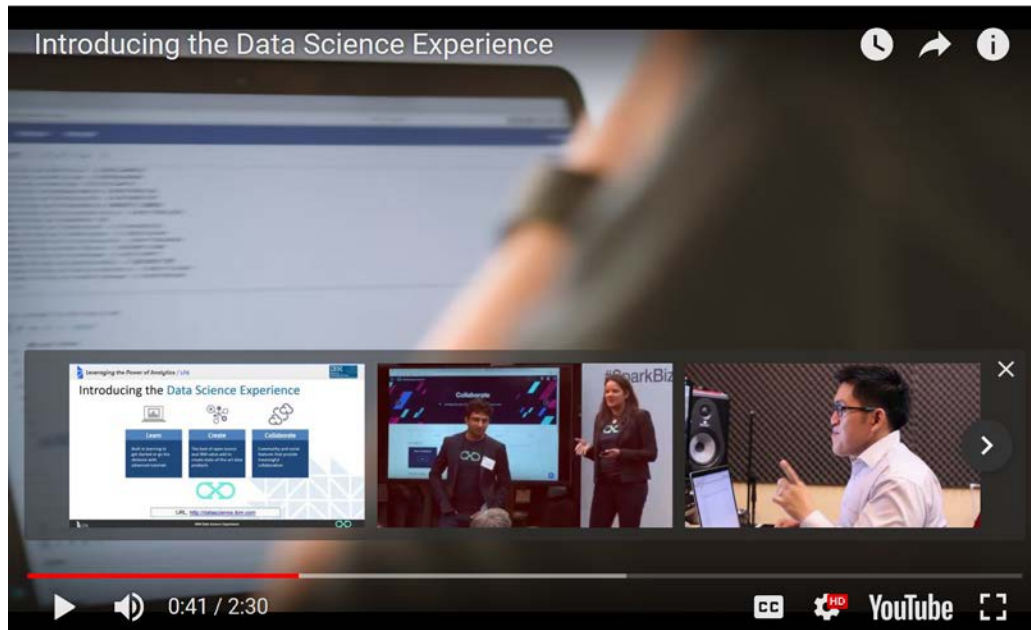
References:

- <https://www2.thelinuxfoundation.org/odpi-data-science-webinar-20171116-slides.pdf>
- <https://zeppelin.apache.org/#pluggable-visualization-via-helium>
https://zeppelin.apache.org/helium_packages.html
- Introduction to Zeppelin
<https://hortonworks.com/tutorial/getting-started-with-apache-zeppelin>

IBM Watson Studio

- IBM Watson Studio is an interactive, collaborative, cloud-based environment where data scientists can use multiple tools to activate their insights. Data scientists can work with a growing set of data science tools such as RStudio, Jupyter, Python, Scala, Spark, IBM Watson Machine Learning, and more
 - <https://datascience.ibm.com/>
 - <https://developer.ibm.com/clouddataservices/docs/ibm-data-science-experience/>
- Watson Studio can be best described as a Platform

Demo: Introducing Watson Studio



Data Science and Data Science Notebooks

© Copyright IBM Corporation 2018

Demo: Introducing Watson Studio

Play at full-screen the YouTube Video at: <https://youtu.be/1HjzkLRdP5k>

Watch this short video as an introduction to Watson Studio as part of the Watson Data Platform.

Other videos on the IBM Watson Studio are available at:

- <https://developer.ibm.com/clouddataservices/docs/ibm-data-science-experience/>

Resources at this location include:

Get Started

- Load and Analyze Public Data Sets
- Manage Object Storage
- Work with Data Connections
- Collaborate on Projects
- Publish Notebooks to GitHub

Notebooks

- Use the Machine Learning Library
- Build SQL Queries
- Build a Custom Library for Apache Spark
- Introduction to Decision Tree Learning
- Introduction to Machine Learning: Predict Cancer Diagnosis
- Introduction to Natural Language Processing (NLP)
- Community Notebook: Precipitation Analysis
- Community Notebook: NY Motor Vehicle Accidents Analysis
- Community Notebook: Use Spark R to Load and Analyze Data

Integrate

- Analyze Data Using RStudio
- Load Db2 Warehouse on Cloud Data with Apache Spark
- Use the SQL-Cloudant Connector in Scala Notebook
- Use the SQL-Cloudant Connector in Python Notebook
- Use GraphFrames
- Sentiment Analysis of Twitter Hashtags Using Spark Streaming
- Sentiment Analysis of Reddit AMAs
- Reddit sentiment analysis in SparkR and CouchDB
- R in Jupyter Notebooks

Working with Data Science in the Cloud

- White Paper by Hortonworks, Power Data Science with Apache Spark in the Cloud: <http://info.hortonworks.com/rs/549-QAL-086/images/hwx-spark-in-the-cloud.pdf>
 - Key uses cases to illustrate the role that Apache Hadoop, Apache Spark and Apache Zeppelin technologies in the daily work of the data scientists
 - Why do data science with Apache Spark and Apache Zeppelin
 - How to best take advantage of solving data science problems with the cloud
- Features:
 - Designed for AWS: Optimized for and integrated with AWS services such as EC2, S3, RDS
 - Focus on your data: Ease of use for developers and data scientists - Zeppelin notebook integrated with Spark & Hive
 - Interoperable HDP services: no rework or recoding from on premises to cloud
 - Enterprise ready: simple, flexible, efficient, secure
- Visit: <https://hortonworks.com/products/cloud/aws/>

Data Science and Data Science Notebooks

© Copyright IBM Corporation 2018

Working with Data Science in the Cloud

Checkpoint

1. What are the generic stages in using the scientific method?
2. What are the 5 major steps in the Data Analytics cycle? Which one typically take the most time?
3. What are the most important computer languages for Data Analytics?
4. Why are Graphical Processing Units (GPUs) important for Data Analytics?
5. How does Jupyter store its workbooks?

Checkpoint

Checkpoint solution

1. What are the generic stages in using the scientific method?
 - make observations; develop questions; formulate hypotheses; develop testable predictions; test predictions; develop general theories.
2. What are the 5 major steps in the Data Analytics cycle? Which one typically takes the most time?
 - Acquisition; Exploration & Understanding; Munging, Wrangling, & Manipulating; Analysis & Modeling; Communicate & Operationalize.
 - Data munging or wrangling can often consume 80% or more of project time and resources
3. What are the most important computer languages for Data Analytics?
 - Python, R, Scala
4. Why are Graphical Processing Units (GPUs) important for Data Analytics?
 - GPUs are special-purpose processors that traditionally can be used to power graphical displays, but for Data Analytics lend themselves to faster algorithm execution because of the large number of independent processing cores.
5. How does Jupyter store its workbooks?
 - Jupyter stores its workbooks in files with the .ipynb suffix. These files can be stored locally or on a hub server.

Data Science and Data Science Notebooks

© Copyright IBM Corporation 2018



Checkpoint solution


Unit summary

- Have a better understanding of methodology “scientific approach” methods used & skills practiced by Data Scientists
- Recognize the iterative nature of a data science project
- Outline the benefits of using Data Science Notebooks
- Describe the mechanisms and tools used with Data Science Notebooks
- Compare and contrast the major Notebooks used by Data Scientists

Unit summary

Unit 2 Data Science with Open Source Tools





Data Science with Open Source Tools

Data Science Foundations

© Copyright IBM Corporation 2018
Course materials may not be reproduced in whole or in part without the written permission of IBM.

Unit objectives

- Getting started with Jupyter Notebook
- Data and notebooks in Jupyter
- How notebooks help data scientists
- Essential packages: NumPy, SciPy, Pandas, Scikit-learn, NLTK, BeautifulSoup, ...
- Data visualizations: matplotlib, ..., PixieDust
- Using Jupyter “Magic” commands

Unit objectives

We will concentrate this unit on the Jupyter Notebook and Python

Supported languages

- Jupyter: Python, R, Julia, and dozens of community maintained kernels.
- Zeppelin: Various languages supports are included in the binary package: Spark, Python, JDBC, JDBC, etc. Third-party interpreters are available through an online registry.



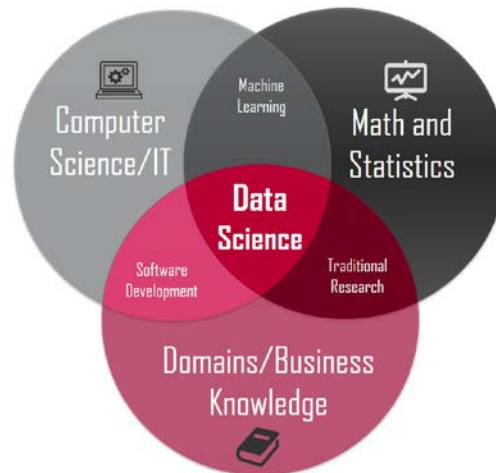
Data Science with Open Source Tools

© Copyright IBM Corporation 2018

Supported languages

Data Science readiness

- Jupyter: Widely used by data scientists for a variety of tasks including quick exploration, documentation of findings, reproducibility, teaching, and presentations
- Zeppelin: Data scientists can collaborate with each other. Also business users can login and collaborate with data scientists directly on notebooks



Data Science with Open Source Tools

© Copyright IBM Corporation 2018

Data Science readiness

Excellent comparison of Jupyter and Zeppelin by the Open Data Platform Initiative (ODPi) at the Linux Foundation:

- Data Science Notebook Guidelines
<http://go.linuxfoundation.org/datasciencenotebookguidelines>
- ODPi Webinar on How BI and Data Science Gets Results
<https://www.odpi.org/tag/data-science>

Multi-user support:

- Jupyter: Native Jupyter does not support multi-user. However, JupyterHub can be used to serve notebooks to users working in separate sessions.
- Zeppelin: Multiple users can collaborate in real-time on a notebook. Multiple users can work with multiple languages in the same notebook.

Architecture:

- Jupyter: The notebook server sends code to language kernels, renders in a browser, and stores code/output/Markdown in JSON files.

- Zeppelin: Zeppelin server daemon manages multiple interpreters (backend integrations). Web application communicates to server using web socket for real-time communication.

BigData Ecosystem:

- Jupyter: Can be connected to a variety of big data execution engines and frameworks: Spark, massively parallel processing (MPP) databases, Hadoop, etc.
- Zeppelin: Tightly integrated with Apache Spark and other big data processing engines.

Security:

- Jupyter: Code executed in the notebook is trusted, like any other Python program. Token-based authentication on by default. Root use disabled by default.
- Zeppelin: User authentication (LDAP, AD integration) Notebook ACL. Interpreter ACL. SSL connection.

Support and Community:

- Jupyter: Mature project with active community and good support. Jupyter project born in 2014 but has roots going back to iPython in 2001. Steering council of ~15 members from academia and commercial companies. Search <https://stackoverflow.com/search?q=jupyter> returns ~10,500 results
- Zeppelin: Apache Zeppelin is one of the most active projects in Apache Software Foundation. Project born in 2013 and became a top level project of ASF in 2015

Jupyter works with 60+ language Kernels



Data Science with Open Source Tools

© Copyright IBM Corporation 2018

Jupyter works with 60+ language Kernels

Getting started with the Jupyter notebook

- Project Jupyter was born out of the IPython project as the project evolved to become a notebook that could support multiple languages Hence the historical name as the IPython notebook
 - The name Jupyter is an indirect acronym of the three core languages it was designed for: **J**ULia, **PY**Thon, and **R** - and is inspired by the planet Jupiter.
- When working with Python in Jupyter, the IPython kernel is used, which gives us some handy access to IPython features from within our Jupyter notebooks
- Jupyter, like IPython, can be hosted locally on your own notebook or on almost any server with ssh or http access
 - Projects & team work can be supported with JupyterHub and similar servers
- You can write code in any of the supported languages, annotate the code with comments using markdown, and generate graphical output

Data Science with Open Source Tools

© Copyright IBM Corporation 2018

Getting started with the Jupyter notebook

References:

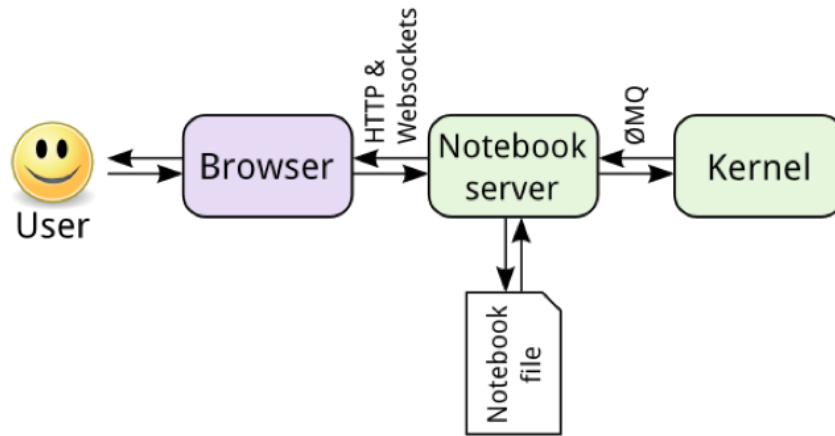
28 Jupyter Notebook tips, tricks, and shortcuts (Dataquest)

<https://www.dataquest.io/blog/jupyter-notebook-tips-tricks-shortcuts/>

Comprehensive Guide to Jupyter Notebook

<http://jupyter-notebook.readthedocs.io/en/stable/notebook.html>

Architecture



Data Science with Open Source Tools

© Copyright IBM Corporation 2018

Architecture

Python

Rkernel

Julia

Dozens of community-maintained kernels

<https://github.com/jupyter/jupyter/wiki/Jupyter-kernels>

IBM Training
IBM

Example of Jupyter in action

Jupyter runs in a web browser

Markdown - text that can include formatting, equations, etc.

Python language statements . . .

that produce, for example, various visualizations of the output data of the analysis

Data Science with Open Source Tools
© Copyright IBM Corporation 2018

Example of Jupyter in action

This example has 5 “cells”:

- Markdown (with an included LaTeX equation)
- Python 3 code
- More markdown (a single line of text)
- Three lines of Python 3
- Two visualizations generated by the Python 3 statements in the prior cell

How notebooks help a data scientist

- Notebooks originated in packages such as Mathematica & Matlab
- Traditionally, notebooks have been used to *document research* and *make results reproducible*, simply by rerunning the notebook on source data
 - Notebooks are typically used by data scientists for quick exploration tasks
 - When properly set up by the organization, a notebook offers direct connections to all necessary sources of data, without additional effort on the part of a future researcher
 - Setting up connections to the right data sources - whether traditional databases or NoSQL data stores, data lakes, and various object and blob stores, each requiring drivers, APIs, permissions, and credentials - is non-trivial, and hence documentation is essential
- Notebooks can be shared and collaborated - and full documentation is typically built in through markdown language(s)
- The end product of the research is available through output and visualization

Data Science with Open Source Tools

© Copyright IBM Corporation 2018

How notebooks help a data scientist

Data notebooks such as the Jupyter notebook take the place often of previous approaches to presenting the end products of research that have been regularly presented through:

- Powerpoint slides
- Written reports

Notebooks also tend to be set up in a cluster environment, allowing the data scientist to take advantage of computational resources beyond what is available on her laptop, and operate on the full data set without having to downsample and download local copy. Additional computational power also enables complex processing, even quick machine learning model training that would be intractable on a local machine.

References:

- The Rise of Data Science Notebooks
<https://www.datanami.com/2016/05/04/rise-data-science-notebooks/>
- An example of what can be done with a shared notebook can be seen with a demo of Databricks notebook running Spark:
https://www.youtube.com/watch?v=NR1MYg_7oSg

Why do we want to study & use Jupyter?

- Jupyter is the prototypical Data Science Notebook
 - It is the basis for other Notebooks: e.g., IBM Watson Studio, Cloudera Data Science Workbench, etc.
- Notebooks support Data Science using a scientific *notebook approach*
- Jupyter is quite extensible, supports many programming languages, and is easily hosted on your computer or on almost any server - you only need to have ssh or http access
 - Supports Python 3 out of the box. Other languages are available by installing appropriate language kernels: e.g., Julia, R, Scala, Python 2, C++, Java, Kotlin, NodeJS, Haskell, etc.
- And is, of course, Open Source and completely free
- DS projects are often distributed in the form of .ipynb notebooks

Why do we want to study & use Jupyter?

References:

Jupyter kernels - a list of available kernels

<https://github.com/jupyter/jupyter/wiki/Jupyter-kernels>

Features of Data Science Notebooks

- Supported languages for Data Science Notebooks
 - Jupyter, for instance, offers over 60 supported Kernels (Python, R, Julia, Scala, ...)
 - Some notebooks are designed around a single language (e.g., R)
- A Jupyter Notebook consists of a **Sequence of Cells** that are:
 - **Code cells** that hold code written in the current language of choice
 - Code is in one of the supported Computer Languages (our examples will use Python 3)
 - Code cells can include Magic statements (discussed soon)
 - **Markdown cells** for documenting the computational process in a literate way
 - **Raw cells** - used for output, including, optionally, graph visualization of output
- Notes & documentation can be included, intermixed with language elements
 - Markdown language (`.md` suffix) - with some variations
 - Markdown can include LaTeX statements (for equations) and HTML5 statements

Data Science with Open Source Tools

© Copyright IBM Corporation 2018

Features of Data Science Notebooks

A Jupyter notebook consists of a sequence of cells. A cell is a multiline text input field, and its contents can be executed by using Shift-Enter, or by clicking either the “Play” button the toolbar, or Cell | Run in the menu bar. The execution behavior of a cell is determined by the cell’s type. There are three types of cells: code cells, markdown cells, and raw cells. Every cell starts off being a code cell, but its type can be changed by using a drop-down on the toolbar (which will be “Code”, initially), or via Keyboard shortcuts.

For more information on the different things you can do in a notebook, see the following (running in an nbviewer):

<http://nbviewer.jupyter.org/github/jupyter/notebook/tree/master/docs/source/examples/Notebook/>

Browser compatibility

- Jupyter Notebook is officially supported by the latest stable versions of the following browsers:
 - Chrome
 - Safari
 - Firefox
 - This mainly due to the notebook's usage of WebSockets and the flexible box model.
- The following browsers are unsupported:
 - Safari < 5
 - Firefox < 6
 - Chrome < 13
 - Opera (any): CSS issues, but execution might work
 - Internet Explorer < 10
 - Internet Explorer ≥ 10 (same as Opera)
 - Using Safari with HTTPS and an untrusted certificate is known to not work (websockets will fail)

Jupyter keyboard-shortcuts

- All actions in the notebook can be performed with the mouse, but keyboard shortcuts are also available for the most common ones. The essential shortcuts to remember are the following:
 - **Shift-Enter** → **run cell** - Execute the current cell, show output (if any), and jump to the next cell below. If **Shift-Enter** is invoked on the *last* cell, a new code cell will be created
 - **Ctrl-Enter** → **run cell in-place** - Execute the current cell as if it were in “terminal mode”, where any output is shown, but the cursor remains in the current cell. The cell's entire contents are selected after execution, so you can just start typing and only the new input will be in the cell
 - **Alt-Enter** → **run cell, insert below** - Executes the current cell, shows the output, and inserts a new cell between the current cell and the cell below (if one exists)
 - **Esc & Enter** → **Command mode & edit mode** - In command mode, you navigate around the notebook using keyboard shortcuts. In edit mode, you can edit text in cells

Data Science with Open Source Tools

© Copyright IBM Corporation 2018

Jupyter keyboard-shortcuts

For the full list of available shortcuts, click Help > Keyboard Shortcuts in the notebook menus.

For a more nuanced description, see:

<http://jupyter-notebook.readthedocs.io/en/stable/notebook.html#keyboard-shortcuts>

Jupyter *magic* commands / functions

- Jupyter kernels support varying *Magic* commands that extend the core language with useful shortcuts - Magic commands start with `%`
- `%lsmagic` (“list magic”) can be used to list all available Magic commands

```
In [1]: 1 %lsmagic
        2

Out[1]: Available line magics:
%alias %alias_magic %autocall %automagic %autosave %bookmark %cd %clear %cls %colors %config %connect_info
%copy %ddir %debug %dhist %dirs %doctest_mode %echo %ed %edit %env %gui %hist %history %killbgscripts %
%ldir %less %load %load_ext %loadpy %logoff %logon %logstart %logstate %logstop %ls %lsmagic %macro %magi
c %matplotlib %mkdir %more %notebook %page %paste %pastebin %pdb %pdef %pdoc %pfile %pinfo %pinfo2 %popd %ppri
nt %precision %profile %prun %psearch %psource %pushd %pwd %pycat %pylab %qtconsole %quickref %recall %r
ehashx %reload_ext %ren %rep %rerun %reset %reset_selective %rmkdir %run %save %sc %set_env %store %sx %
system %tb %time %timeit %unalias %unload_ext %who %who_ls %whos %xdel %xmode

Available cell magics:
%%! %%HTML %%SVG %%bash %%capture %%cmd %%debug %%file %%html %%javascript %%js %%latex %%markdown %%per
l %%prun %%pypy %%python %%python2 %%python3 %%ruby %%script %%sh %%svg %%sx %%system %%time %%timeit %
%writefile

Automagic is ON, % prefix IS NOT needed for line magics.
```

Data Science with Open Source Tools

© Copyright IBM Corporation 2018

Jupyter magic commands / functions

References:

<https://www.dataquest.io/blog/jupyter-notebook-tips-tricks-shortcuts/>

<http://ipython.readthedocs.io/en/stable/interactive/magics.html> (explains all the magic commands in detail with examples of usage)

Book: VanderPlas, J. (2016). Python data science handbook: Essential tools for working with data. Sebastopol, CA: O'Reilly Media. ISBN: 978-1-491-91205-8. 541 pp. List price, \$28. Amazon, US \$32.

Publisher's description of this book:

Working scientists and data crunchers familiar with reading and writing Python code will find this comprehensive desk reference ideal for tackling day-to-day issues: manipulating, transforming, and cleaning data; visualizing different types of data; and using data to build statistical or machine learning models. Quite simply, this is the must-have reference for scientific computing in Python.

With this handbook, you'll learn how to use:

- IPython and Jupyter: provide computational environments for data scientists using Python
- NumPy: includes the ndarray for efficient storage and manipulation of dense data arrays in Python
- Pandas: features the DataFrame for efficient storage and manipulation of labeled/columnar data in Python
- Matplotlib: includes capabilities for a flexible range of data visualizations in Python
- Scikit-Learn: for efficient and clean Python implementations of the most important and established machine learning algorithms

Markdown

- Markdown is a lightweight markup language with plain text formatting syntax. It is designed so that it can be converted to HTML and many other formats using a tool by the same name
 - There is no clearly defined Markdown standard, apart from the original writeup and implementation by John Gruber
 - From 2012, a group of people including Jeff Atwood launched what Atwood characterized as a standardization effort → **CommonMark**
 - Wikipedia has an excellent introduction:
<https://en.wikipedia.org/wiki/Markdown>
- There are some flavors of Markdown
 - Markdown Extra (https://en.wikipedia.org/wiki/Markdown#Markdown_Extra)
 - GitHub-Flavored Markdown Editor (<https://jbt.github.io/markdown-editor/>)
 - WordPress Markdown (<https://en.support.wordpress.com/markdown-quick-reference/>)

Markdown

On the use of LaTeX in Markdown:

<https://www.youtube.com/watch?v=-F4WS8o-G2A>

<https://upmath.me/>

<https://github.com/parpalak/upmath.me>

<https://www.overleaf.com/blog/501-markdown-into-latex-with-style>

Example Markdown formatting

- Basic (plain text) formatting shown on left from <http://commonmark.org/help/>
- With MathJax, you can included mathematical expressions both inline $e^{i\pi}+1=0$ and displayed

$$e^x = \sum_{i=0}^{\infty} \frac{1}{i!} x^i$$

This is important to express the mathematical and statistical intent of the work in hand

- To convert from Markdown to LaTeX, visit <http://kesdev.com/you-got-latex-in-my-markdown/>

Type	Or	...to Get
<code><Italic></code>	<code><Italic></code>	<i>Italic</i>
<code><<Bold>></code>	<code><<Bold>></code>	Bold
<code># Heading 1</code>	Heading 1 =====	Heading 1
<code>## Heading 2</code>	Heading 2 =====	Heading 2
<code>[Link](http://a.com)</code>	<code>[Link][1]</code> <code>[1]: http://b.org</code>	Link
<code>[[Image]](http://url/a.png)</code>	<code>[[Image]](1)</code> <code>[1]: http://url/b.jpg</code>	
<code>< Blockquote</code>		Blockquote
<code>< LIST</code> <code>< LIST</code> <code>< LIST</code>	<code>< LIST</code> <code>< LIST</code> <code>< LIST</code>	<ul style="list-style-type: none">• List• List• List
<code>1. One</code> <code>2. Two</code> <code>3. Three</code>	<code>1) One</code> <code>2) Two</code> <code>3) Three</code>	<ol style="list-style-type: none">1. One2. Two3. Three
Horizontal Rule <code>---</code>	Horizontal Rule <code>---</code>	Horizontal Rule
<code>Inline code` with backticks</code>		Inline code` with backticks
<code>...` # code block print "3 backticks or" print "indent 4 spaces"</code>	<code>...` # code block print "3 backticks or" print "indent 4 spaces"</code>	<pre># code block print "3 backticks or" print "indent 4 spaces"</pre>

Example Markdown formatting

References:

Markdown for Jupyter notebooks Cheatsheet (from IBM Watson Data Platform)

<https://medium.com/ibm-data-science-experience/markdown-for-jupyter-notebooks-cheatsheet-386c05aeebed>

<http://commonmark.org/help/>

[http://jupyter-](http://jupyter-notebook.readthedocs.io/en/stable/examples/Notebook/Working%20With%20Markdown%20Cells.html)

[notebook.readthedocs.io/en/stable/examples/Notebook/Working%20With%20Markdown%20Cells.html](http://jupyter-notebook.readthedocs.io/en/stable/examples/Notebook/Working%20With%20Markdown%20Cells.html)

The examples with MathJax shown on the slide can be achieved in the following manner:

Inline expressions can be added by surrounding the latex code with `$`:

`$e^{i\pi} + 1 = 0$`

Expressions on their own line are surrounded by `$$`:

`$$e^x = \sum_{i=0}^{\infty} \frac{1}{i!} x^i$$`

See: <http://jupyter-notebook.readthedocs.io/en/stable/examples/Notebook/Working%20With%20Markdown%20Cells.html#LaTeX-equations>

Essential packages - for Python 3

- NumPy
 - The most fundamental package, around which the scientific computation stack is built, is NumPy (stands for Numerical Python). It provides an abundance of useful features for operations on n-arrays and matrices in Python.
- SciPy
 - SciPy is a library of software for engineering and science with modules for linear algebra, optimization, integration, and statistics. The main functionality of SciPy library is built upon NumPy, and its arrays thus make substantial use of NumPy.
- Pandas
 - Pandas is a Python package designed to do work with “labeled” and “relational” data simple and intuitive. Pandas is a perfect tool for data wrangling. It designed for quick and easy data manipulation, aggregation, and visualization.

Data Science with Open Source Tools

© Copyright IBM Corporation 2018

Essential packages - for Python 3

References:

Top 15 Python Libraries for Data Science in 2017

<https://activewizards.com/blog/top-15-libraries-for-data-science-in-python/>

Essential packages - for Python 3 - *continued*

- Scikit-learn
 - Exposes a concise and consistent interface to the common machine learning algorithms, making it simple to bring ML into production systems. The library combines quality code and good documentation, ease of use and high performance and is a de-facto industry standard for machine learning with Python
- NLTK
 - The Natural Language Toolkit is used for common tasks of symbolic and statistical NLP
 - Allows such as text tagging, classification, and tokenizing, name entities identification, building corpus tree that reveals inter and intra-sentence dependencies, stemming, and semantic reasoning.
 - All of the building blocks allow for building complex research systems for different tasks, for example, sentiment analytics, automatic summarization
- And more ...

Essential packages - for Python 3 - continued

Essential packages - for Python 3 - *HTML*

- Data mining of websites with Python and BeautifulSoup - aka web scraping
- **BeautifulSoup v4**
 - Python package for parsing HTML and XML documents
 - Creates a parse tree from webpages that can be used to extract data from HTML
 - Provides a scalable way to collect, organize, and analyze information from the Internet



Data Science with Open Source Tools

© Copyright IBM Corporation 2018

Essential packages - for Python 3 - HTML

References:

- <https://medium.freecodecamp.org/how-to-scrape-websites-with-python-and-beautifulsoup-5946935d93fe>
- Web Scraping with Beautiful Soup - as a Jupiter-based Tutorial
http://web.stanford.edu/~zlotnick/TextAsData/Web_Scraping_with_Beautiful_Soup.html
- BeautifulSoup v4 Documentation
<https://www.crummy.com/software/BeautifulSoup/bs4/doc/>

Installing Jupyter

- If Jupyter is not already installed, it is installed best using Anaconda 5 using:
 - Anaconda 5.0.1 for Linux 64-bit with Python 3.6 (Anaconda3-5.0.1-Linux-x86_64.sh) from <https://www.anaconda.com/download/#linux>
 - Anaconda Distribution Starter Guide http://know.anaconda.com/rs/387-XNW-688/images/2017-08_Anaconda_Starter_Guide_CheatSheet_Web.pdf
- Distributions of Jupyter are available from Anaconda.com for:
 - Windows 64-bit & 32-bit
 - MacOS 64-bit
 - Linux 64-bit (x86 & Power8) & 32-bit
 - In each case with Python 3.6 or Python 2.7 - and earlier
 - With BSD licensing and thus permission to use commercially and for redistribution
 - Free Community support or paid Enterprise support

Data Science with Open Source Tools

© Copyright IBM Corporation 2018

Installing Jupyter

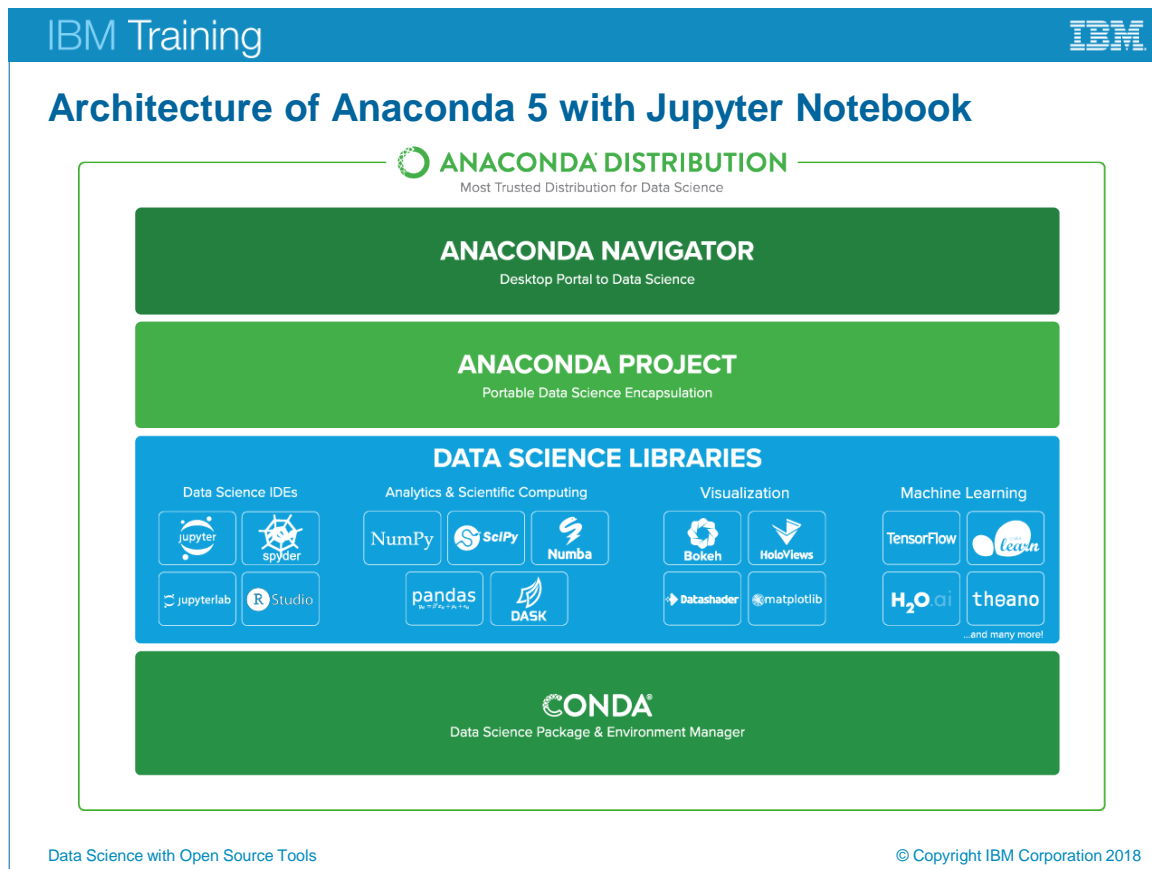
Project Jupyter: <http://jupyter.org/> “Project Jupyter exists to develop open-source software, open-standards, and services for interactive computing across dozens of programming languages.”

Jupyter is best run as a web application.

You can also run the command line version QtConsole or use the Spyder IDE (Integrated Development Environment). When you install Anaconda 5, you get the local web-based browser version as well as QtConsole and Spyder. The QtConsole can use any Jupyter kernel.

References:

- The Qt Console for Jupyter - Jupyter Qt Console 4.3.1 documentation <https://qtconsole.readthedocs.io/>
- Demonstration of the Jupyter QtConsole + Vim integration - YouTube https://www.youtube.com/watch?v=Fi_Xd6h4ncw
- Integrating Vim and jupyter-qtconsole <https://blogs.aalto.fi/marijn/tag/jupyter-qtconsole/>



Architecture of Anaconda 5 with Jupyter Notebook

Reference:

- <https://www.anaconda.com/distribution/>

You should make the following choices:

- 64-bit distribution
- Based on Python 3.6+

Even though Python 2.7 distributions are available, you should note that Python 2.x will be no longer supported after 2020 (extended from 2015). All packages from Python 2.x are now available with Python 3.

Jupyter *magic* for installing Python packages

- How can I install a Python package so that it works with my Python notebook, using `pip` and/or `conda` ?
 - Fundamentally the problem is usually rooted in the fact that the Jupyter kernels are disconnected from Jupyter's shell; in other words, the installer points to a different Python version than is being used in the notebook
- The solution requires you to define a `%pip` magic function (and/or a `%conda` magic function) that works in the current notebook kernel

```
In [21]: from IPython.core.magic import register_line_magic

@register_line_magic
def pip(args):
    """Use pip from the current kernel"""
    from pip import main
    main(args.split())
```

Running it as follows will install packages in the expected location

```
In [22]: %pip install numpy
```

Jupyter magic for installing Python packages

Reference:

Installing Python Packages from a Jupyter Notebook

<https://jakevdp.github.io/blog/2017/12/05/installing-python-packages-from-jupyter/>

Git / GitHub for version control and for distribution

- **Git** is a version control system for tracking changes in computer files and coordinating work on those files among multiple people
 - Linus Torvalds wanted in 2005 a distributed system that he could use, but none of the available free systems met his needs, especially for performance
- **GitHub** is a Web-based Git version control repository hosting service. It is mostly used for computer code
 - It offers all of the distributed version control and source code management (SCM) functionality of Git as well as adding its own features
 - It provides access control and several collaboration features such as bug tracking, feature requests, task management, and wikis for every project
- GitHub has become an effective **public repository** for a large number of Notebooks - introductory & advanced - on Data Science and other topics

Data Science with Open Source Tools

© Copyright IBM Corporation 2018

Git / GitHub for version control and for distribution

A Gallery of Interesting Jupyter Notebooks

<https://github.com/jupyter/jupyter/wiki/A-gallery-of-interesting-Jupyter-Notebooks>

Entire books or other large collections of notebooks on a topic

- Introductory Tutorials
- Programming and Computer Science
- Statistics, Machine Learning and Data Science
- Mathematics, Physics, Chemistry, Biology
- Earth Science and Geo-Spatial data
- Linguistics and Text Mining
- Signal Processing
- Engineering Education

Scientific computing and data analysis with the SciPy Stack:

- General topics in scientific computing
- Social data
- Psychology and Neuroscience
- Machine Learning, Statistics and Probability
- Physics, Chemistry and Biology
- Economics and Finance
- Earth science and geo-spatial data
- Data visualization and plotting
- Mathematics
- Signal, Sound and Image Processing
- Natural Language Processing
- Pandas for data analysis

General Python Programming Notebooks in languages other than Python:

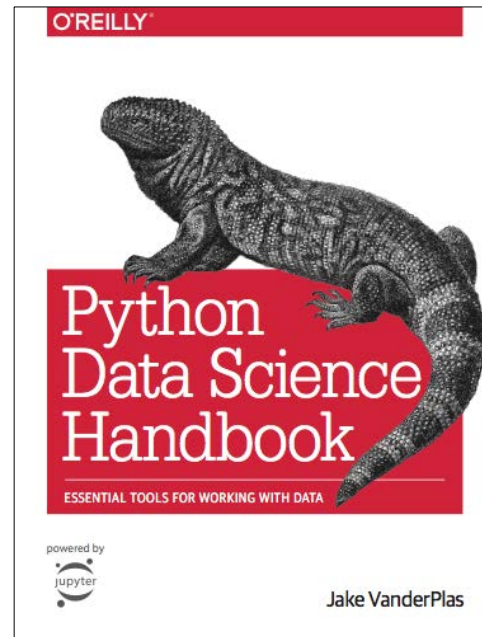
- Julia
- Haskell
- Ruby
- Perl
- F#
- C#
- Javascript

Miscellaneous topics about doing various things with the Notebook itself:

- Reproducible academic publications
- Other publications using Notebooks
- Data-driven journalism

Python Data Science Handbook

- The full text of the book *Python Data Science Handbook* is available for free on the website <https://jakevdp.github.io/PythonDataScienceHandbook/> - this content is available there as text pages and on **GitHub** as a set of Jupyter notebooks
- Go to the author's GitHub site at <https://github.com/jakevdp/PythonDataScienceHandbook> and click on the green “**clone or download**” to get the full set of Jupyter **.ipynb** files (use **download** to get a zipfile with everything)
- Once downloaded, any of the **.ipynb** files can then be uploaded *locally* into your Jupyter browser



Data Science with Open Source Tools

© Copyright IBM Corporation 2018

Python Data Science Handbook

VanderPlas, J. (2016). *Python data science handbook: Essential tools for working with data*. Sebastopol, CA: O'Reilly Media. 548pp. ISBN 978-1-491-91205-8. List price USD \$60. Amazon USA \$32.

The author states: “The text is released under the CC-BY-NC-ND license, and code is released under the MIT license. If you find this content useful, please consider supporting the work by buying the book!”

Lab 1

Introduction to Jupyter notebooks

- Start Jupyter - it will open in a web browser
- Import the lab file (all Jupyter files have a `.ipynb` suffix) into your default workspace
 - This is now a copy of the provided lab file and you can do anything with it
 - If you mess it up, you can re-import again later
- Explore the component panels - some are markdown, some are code, some are results of running the code (output data, visualizations, ...)
- Learn how to run single panels - and then the whole script
 - You may need to adjust the provided script to locate the data files that accompany the Jupyter `.ipynb` file
 - Add some additional panels, as described in the lab script

Demonstration 1: Introduction to Jupyter notebooks

“But concepts are useless without practice. You can’t understand how to build software in a new way, without building software in a new way. ... There’s such a thing as knowledge that you have but don’t ‘know.’ The neural network of your brain is better at learning by example than by study; and the craft of software development, like all crafts, is best learned this way. The discrete knowledge in this [lab]-the abstract concepts, the classifications and definitions-is a pale shadow of the skills you’ll eventually master. Taking inspiration from Taoist philosophy, this [lab] can only show you the way: you must walk it yourself to truly learn. And although the map isn’t the territory, having it is definitely better than being lost.”

Rodger, R. (2018). The Tao of Microservices. Shelter Island, NY: Manning, p. xiii-xiv.

Hortonworks also provides a more sophisticated of tutorial labs that go beyond the material in this course/workshop, e.g.:

- Hands-on Tour of Apache Spark in 5 minutes
<https://hortonworks.com/tutorial/hands-on-tour-of-apache-spark-in-5-minutes/>

This tutorial uses an Apache Zeppelin notebook for the development environment to keep things simple and elegant. Zeppelin will allow you to run in a pre-configured environment and execute code written for Spark in Scala and SQL, a few basic Shell commands, pre-written Markdown directions, and an HTML formatted table.

This tutorial is a part of series of hands-on tutorials using the Hortonworks Data Platform (HDP) via either the Hortonworks Data Cloud (HDCloud) or a pre-configured downloadable HDP Sandbox.

The Zeppelin notebook uses basic Scala syntax. A Python version is coming soon.

If you are new to Zeppelin, review the following tutorial: Getting Started with Apache Zeppelin.

There are also videos on YouTube:

- Jupyter in Depth (3 hr 19 min)
<https://www.youtube.com/watch?v=k7WXVWej-NY>
- Jupyter For Everything Else (45 mins)
https://www.youtube.com/watch?v=fSbuAKIRs_4

Lab 1: Introduction to Jupyter notebooks

Purpose:
Introduction to Jupyter notebooks.

Task 1. Configure Jupyter on first use.

The following setup is needed for the **nbuser** user.

1. Login for the first-time as **nbuser** with password **passw0rd**.
Verify that you have access to both Jupyter and Python 3.
2. Open a new terminal window.

```
jupyter --version
```

```
4.4.0
```

```
python --version
```

```
Python 3.6.3 :: Anaconda, Inc.
```
3. Create a Jupyter configuration file for the user.

```
jupyter notebook --generate-config
```
4. Generate a password for this user using Jupyter

```
jupyter notebook password
```

Use **passw0rd** for your password. This must be entered twice.
5. You can find all the appropriate paths for files by using the following command (with the results for Linux [CentOS 7] and Anaconda installed at /opt/anaconda3):

```
jupyter -paths
```

config:

```
/home/nbuser/.jupyter
/opt/anaconda3/etc/jupyter
/user/local/etc/jupyter
/etc/jupyter
```

data:

```
/home/nbuser/.local/share/jupyter
/opt/anaconda3/share/jupyter
/user/share/jupyter
```

runtime:

```
/home/nbuser/.local/share/jupyter/runtime
```

You are already familiar with the *config* file created in step 2 above - it is a text file (though it can be useful to review the contents). Most of the entries are preceded by a pound/hash sign (“#”) and are not active.

The *data* location is where notebook files can be currently found. New ones (uploaded by this user) are stored in the first of these directories.

The runtime location is where any currently running Jupyter server will store its runtime information (in a JSON formatted file) and any cookie information.

If you are running with Jupyter locally on Window 10 and using PowerShell, you will get something similar to this (for user glen):

jupyter -paths

```
PS C:\Users\glen> jupyter --paths
config:
    C:\Users\glen\.jupyter
    c:\python36-32\etc\jupyter
    C:\ProgramData\jupyter
data:
    C:\Users\glen\AppData\Roaming\jupyter
    c:\python36-32\share\jupyter
    C:\ProgramData\jupyter
runtime:
    C:\Users\glen\AppData\Roaming\jupyter\runtime
```

Task 2. Download any external Jupyter notebook files.

1. Download any external Jupyter Notebook files (suffix `.ipynb`) to your user space from location(s) specified by your instructor - either your personal Desktop or into a folder Downloads that you would create for your user space or to your system temporary directory (e.g. on Linux, `/tmp`)

URL Location(s):

The three simplest ways to work with or download these Jupyter Notebook files are:

- Open the remote Jupyter notebook with nbviewer (for read-only use only)
 - Use a web browser to download and save locally
 - Download with the wget utility (generally available on Linux/Mac OS, and available on Windows 10 Powershell)
2. For example, using the example of the *first* notebook available at:
<https://github.com/jupyter/jupyter/wiki/A-gallery-of-interesting-Jupyter-Notebooks#introductory-tutorials>

- **Use the Notebook Viewer** (nbviewer):
<https://nbviewer.jupyter.org/github/jupyter/notebook/blob/master/docs/source/examples/Notebook/Running%20Code.ipynb>
- **Download the notebook**, from the download prompt from the top-right-corner of the Notebook viewer:



- **Use wget** to download from the command line:

wget

```
https://raw.githubusercontent.com/jupyter/notebook/master/docs/source/examples/Notebook/Running%20Code.ipynb -O firstlab.ipynb
```

Note that this must be entered as *one line* and with *precisely three spaces in it* (one after wget, and one before & one after -O). The dash-Capital-Oh provides the file name that is used to store the file into the current directory (the directory where the command is run).

Task 3. Bring up Jupyter for the first time.

1. From a terminal window, and logged in to **nbuser**, run:

```
jupyter notebook &
```

The ampersand (&) is best used on Linux and Mac OS systems to free up the terminal window for future commands. You cannot use the ampersand, however, if you are running this from a Windows 10 PowerShell terminal window.

Do not be concerned about any error messages as Firefox may not have been setup with an appropriate default display.

2. If there is no Firefox Web Browser icon on the user desktop, create one for your future convenience (on Linux CentOS: under **Applications > Internet**, find the **Firefox Web Brower**. Do not click, but rather **drag** this icon to the user Desktop.
3. Double-click on the **Firefox Web Browser Icon** that is on the Desktop in order to start Firefox.
4. If there is a **Jupyter** entry on your Bookmarks Toolbar, double-click on that. If there no Jupyter entry on your Bookmarks Toolbar, you can create one: Enter the following URL in the browser window: **localhost:8888** (the *default* location for the Jupyter; your port number may be different if set otherwise or if something else is running on this port).
Before entering a password, drag the **(i)** to the left of the URL window to the Bookmark Toolbar of the open Firefox web browser **Bookmarks Toolbar**.
If the entry on the Toolbar appears as **Home** rather than Jupyter Notebook, right-click on the entry on the toolbar and select Properties; then change the Name from Home to **Jupyter**.
5. You can now enter the password **passw0rd** on the webpage to connect to the local Jupyter server.
Jupyter is now ready to use.

6. On all future connections to Jupyter you can go directly to **Step 6**. If the CentOS server has been down or the user nbuser logged out, you will probably also need to start at **Step 1** as well.

The steps in the previous tasks can be used at any time to download files that would be later uploaded to a Jupyter notebook.

7. Any previously *used* notebook files will appear in the browser window. If you have a notebook file that has not been previously used, but you want to work on it now, select **Upload** from the Icon at the top right-hand corner of the browser window - this will upload the notebook file (suffix .ipynb) to the workspace).

Task 4. Explore how to use Jupyter.

These labs presume that you have Internet access to be able to pull files from the Internet and to be able to access web pages on the internet.

We will download a file from the Internet for this first exploratory lab. This can be done in an exploratory terminal window on your desktop.

The file needed is available at:

<https://nbviewer.jupyter.org/github/jupyter/notebook/blob/master/docs/source/examples/Notebook/Running%20Code.ipynb>

1. In Linux, right-click on your desktop and select **Open Terminal**.

You can download it directly from the nbviewer window (the download symbol is at the end of the set of icons at the top right corner of the web browser window). Or, you can read this file remotely using a web browser or download it to your system with (all one line) and run it there:

```
wget
https://raw.githubusercontent.com/jupyter/notebook/master/
docs/source/examples/Notebook/Running%20Code.ipynb -O
firstlab.ipynb
```

2. Take note of the directory/folder where you downloaded this file (now named firstlab.ipynb) by using the command:

```
pwd
```

Task 5. Work with Jupyter yourself.

The following general directions apply regardless of whether you are running Jupyter on Linux, Windows, or Mac OS, and with any web browser (screen captures are from Mozilla Firefox). Everything is based on the current version of Jupyter as installed with Anaconda 5.

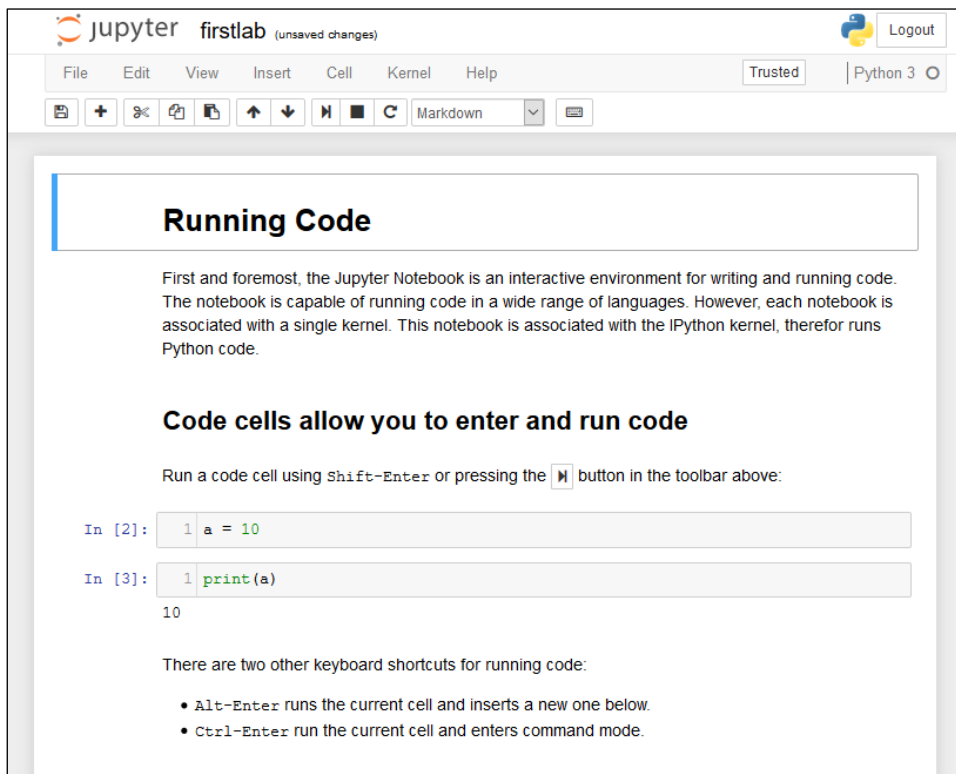
Upload your file to Jupyter.

1. Use the **Upload** button (A) to open a browser from which you can locate the file. This places the file at the file list (B). Then hit the **Colored-Upload** button (C) to actually move a copy of the file into your work directory.

Note the two buttons - **Name** and **Last Modified** - enable you to sort files by name and by late modify data and time. Use **Last Modified** to bring your new *Jupyter Notebook File* to the top of the list.



2. Once your file has been uploaded, double-click on it to open it. Note the name at the top (**firstlab**) and that this particular Jupyter Notebook File is based on the **Python 3** programming language.



3. Place your mouse in the two cells **In [2]** and **In [3]** successively and run the

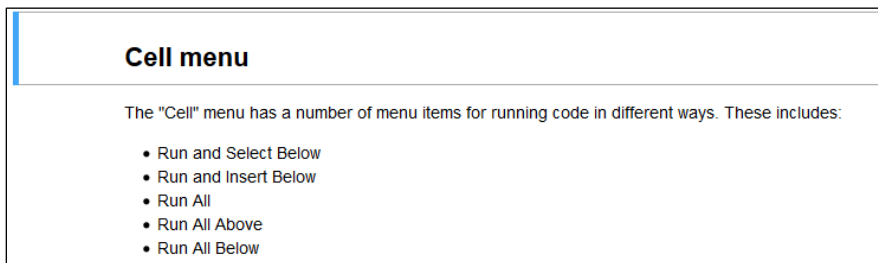
Python 3 code in those cells, with the **Run**  button.

This button is used to execute code in the current cell (and that only). There will often be code in prior cells that must be executed first to set up the environment for the current cell.

A full set of Jupyter Shortcut Keys can be found at:

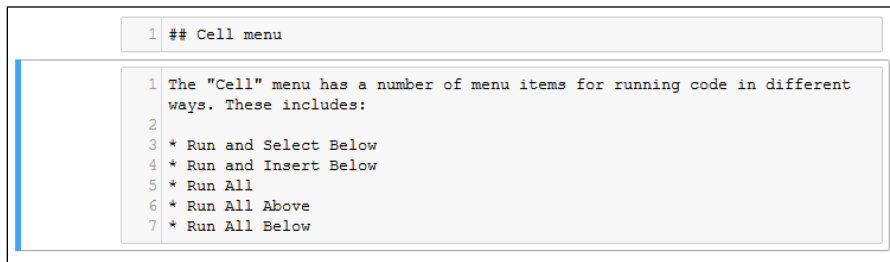
- from the Browser itself: **Help > Keyboard Shortcuts**
- or various websites, e.g.,:
<https://gist.github.com/kidpixo/f4318f8c8143adee5b40>

4. Scroll down to this pair of cells:




5. For each of these two cells, individually, take the following steps:

- Click on the cell - and note that it is classified as Markdown
- Select instead the value of **Raw NBConvert** from this dropdown



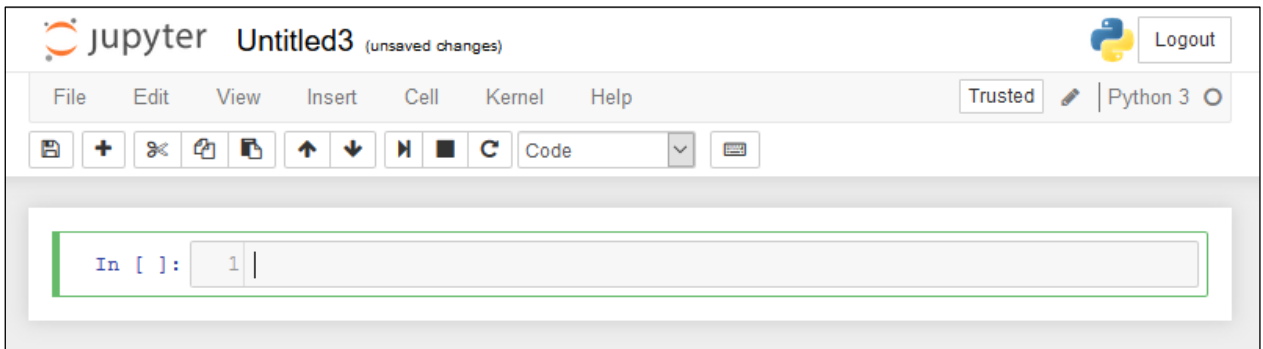
These two cells will now show the actual raw text that is held in the cell. By the way, the same could have been coded as just one cell with these textual values in the one call. Note the following:

- The **##** is the way to represent the second level of heading (this is slightly smaller than heading 1 that is shown in Step 2, where you can see examples of both a Heading 1 and a Heading 2). There are eight levels of heading.
- The second of these two cells illustrates a paragraph of text, delimited by a blank-line as well as five bullets, introduced by a ***** at the beginning of the line.
- These two cells are no longer seen as Markdown, as such.

6. Return the two cells to their former formatting, individually, by:
 - Selecting **Markdown** for each cell
 - Re-running each cell with the **Run**  button
7. Explore other cells of this Notebook, both Code (which you can run) and Markdown (which you can look at for their formatting).

Task 6. Create your own notebook from scratch.

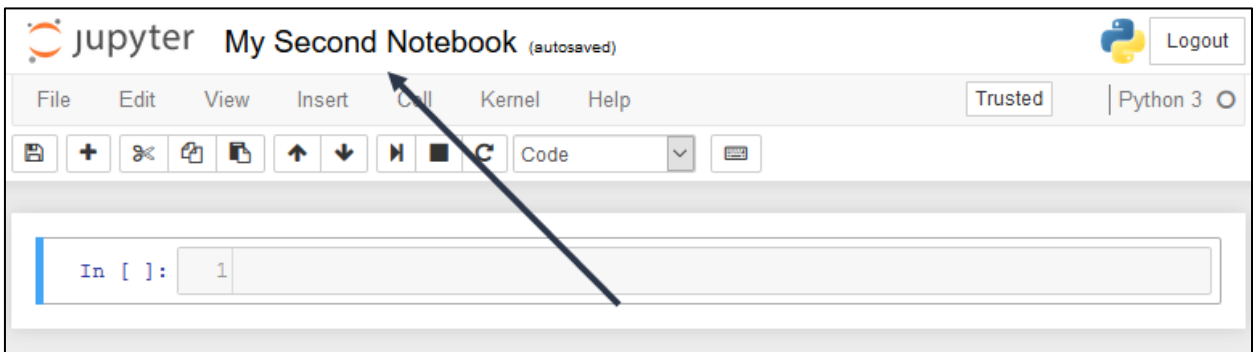
1. Start a new Jupyter Notebook: **File > New Notebook > Python 3**



We currently do not have any other kernels configured. Thus, our only choice is Python 3.

In this lab we will not have time to deploy other kernels such as R, etc.

2. Rename the notebook from Untitled to a value you select, e.g., **My Second Notebook**, by using: **File > Rename** and entering a value.



The new notebook currently has one cell. We will use it for Markdown.

3. Choose **Markdown** from the drop-down.

4. Enter the following text (the left hand edge is the beginning of each line):

```
# Sample Markdown text
## Heading 2
### Heading 3
```

There are eight levels of headings

Here is a list:

- One
- Two - with some *italic* text
- Three - with some **bold** text
- Four

And this sentence has an embedded formula $\sqrt{3x-1}+(1+x)^2$ in the text itself using \LaTeX between single dollar-signs.

For more complex formulas such as the following where the formula is place between double dollar-signs. Both this and the previous use MathJax to interpret the LaTeX-coded formula.

The standard normal cumulative distribution function is:

$$\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^x e^{-\frac{z^2}{2}} dz$$

5. And then the **Run** the cell. You will see that we have used an equation written in LaTeX between two \$-signs. This will be set as an equation once the markdown is run.

6. Create a new cell using the **+** in the menu bar.
7. Set the new cell as **Code**.

We can use the matplotlib plotting framework in Python 3 to display a graph. We will draw a simple graph, unrelated to the previous cell (in a notebook, normally each cell tells part of an ongoing story - but not here).

8. Use the following Python 3 code which invokes the matplotlib library:

```
import numpy as np
import matplotlib.pyplot as plt

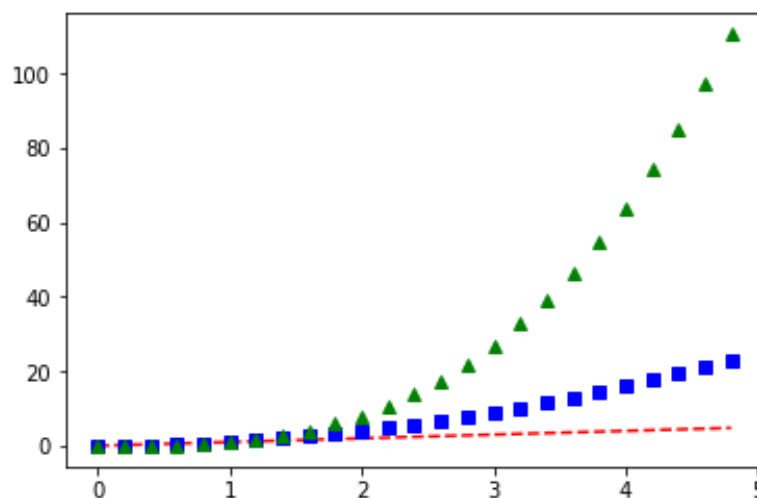
# evenly sampled time at 200ms intervals
t = np.arange(0., 5., 0.2)

# red dashes, blue squares and green triangles
plt.plot(t, t, 'r--', t, t**2, 'bs', t, t**3, 'g^')
plt.show()
```

```
In [1]: import numpy as np
import matplotlib.pyplot as plt

# evenly sampled time at 200ms intervals
t = np.arange(0., 5., 0.2)

# red dashes, blue squares and green triangles
plt.plot(t, t, 'r--', t, t**2, 'bs', t, t**3, 'g^')
plt.show()
```



This example comes from the **Pyplot Tutorial** on the official **matplotlib** website at: https://matplotlib.org/users/pyplot_tutorial.html

Jupyter can be used to display HTML text and to embed graphics / videos, e.g., YouTube videos.

Try the following **Code** example:

```
In [13]: 1 from IPython.display import HTML
          2
          3 # Youtube
          4 HTML('<iframe width="560" height="315"
          5       src="https://www.youtube.com/embed/UsCQXe1OHZk"
          6       frameborder="0" allowfullscreen>
          7       </iframe>')
          8
```

Out[13]:



The output of the code is a YouTube video player. The video title is "Python 3 Tutorial: Print Function and Strings". The video content shows the text "Python 3 Basics #2 Print & Strings" with a play button icon. The video player has a dark grey header with the title and a clock icon. The main content area has a white background with the text "Python 3 Basics #2 Print & Strings" in various colors and fonts. The video player has a white border and a dark grey footer.

```
In [ ]: 1
```

Note: The triple-single-quote (""") is used to allow a Python multi-line string to extend over multiple lines for clarity.

Having used the following import

```
from IPython.display import HTML
```

...any text in HTML format can be used as parameter to the Python function `HTML()` that has been imported.

9. Jupyter Notebooks are typically written to provide textual comment (“markdown”) on sections of Python and other language code (“code”). Thus, you can include HTML, e.g., the following code outputs a table with two columns and three rows, including a header row:

```
from IPython.display import HTML

from IPython.display import HTML

table = """<table style="width:50%">
<tr>
    <th>Firstname</th>
    <th>Lastname</th>
    <th>Age</th>
</tr>
<tr>
    <td>Jill</td>
    <td>Smith</td>
    <td>50</td>
</tr>
<tr>
    <td>Eve</td>
    <td>Jackson</td>
    <td>94</td>
</tr>
<tr>
    <td>John</td>
    <td>Doe</td>
    <td>80</td>
</tr>
</table>"""

HTML(table)
```

The `html` function here will render HTML tags as a string in its input argument. We can see the final output as follows:

Out[19]:


Firstname	Lastname	Age
Jill	Smith	50
Eve	Jackson	94
John	Doe	80

To work with images, such as JPEG and PNG, use the **Image** class of the IPython display module.

10. Run the following code to display a sample image; when run, this is the output:

```
In [20]: 1 from IPython.display import Image
          2 Image(url='http://python.org/images/python-logo.gif')
```

Out[20]:



Naturally, you can write Python 3 code in any cell marked as **Code** and execute it. Feel free to experiment.

Other language kernels such as R can be added to Jupyter, but these are beyond the scope of this lab.

Results:

You have been introduced to Jupyter notebooks.

Unit summary

- Getting started with Jupyter Notebook
- Data and notebooks in Jupyter
- How notebooks help data scientists
- Essential packages: NumPy, SciPy, Pandas, Scikit-learn, NLTK, BeautifulSoup, ...
- Data visualizations: matplotlib, ..., PixieDust
- Using Jupyter “Magic” commands

Unit summary



IBM Training

