1

Introduction to Python R and Data Science

1.1 What Is Python?

Python is a programming language that lets you work more quickly and integrate your systems more effectively. It was created by Guido van Rossum. You can read Guido's history of Python at the History of Python blog at http://python-history.blogspot.in/2009/01/introduction-and-overview.html.

It is worth reading for beginners and even experienced people in Python. The following is just an extract:

> many of Python's keywords (if, else, while, for, etc.) are the same as in C, Python identifiers have the same naming rules as C, and most of the standard operators have the same meaning as C. Of course, Python is obviously not C and one major area where it differs is that instead of using braces for statement grouping, it uses indentation. For example, instead of writing statements in C like this

```
if (a < b) {
   max = b;
} else {
   max = a;
}
```

> Python just dispenses with the braces altogether (along with the trailing semicolons for good measure) and uses the following structure:

```
if a < b:
   max = b
else:
   max = a
```

COPYRIGHTED MATERIAL

Python® for R Users: A Data Science Approach, First Edition. Ajay Ohri.
© 2018 John Wiley & Sons, Inc. Published 2018 by John Wiley & Sons, Inc.

> The other major area where Python differs from C-like languages is in its use of dynamic typing. In C, variables must always be explicitly declared and given a specific type such as int or double. This information is then used to perform static compile-time checks of the program as well as for allocating memory locations used for storing the variable's value. In Python, variables are simply names that refer to objects.

The Python Package Index (PyPI) https://pypi.python.org/pypi hosts third-party modules for Python. There are currently **91 625** packages there. You can browse Python packages by topic at https://pypi.python.org/pypi?%3A action=browse

1.2 What Is R?

The official definition of what is R is given on the main website at http://www.r-project.org/about.html

> R is an integrated suite of software facilities for data manipulation, calculation and graphical display. It includes an effective data handling and storage facility, a suite of operators for calculations on arrays, in particular matrices, a large, coherent, integrated collection of intermediate tools for data analysis, graphical facilities for data analysis and display either on-screen or on hardcopy, and a well-developed, simple and effective programming language which includes conditionals, loops, user-defined recursive functions and input and output facilities.
>
> The term 'environment' is intended to characterize it as a fully planned and coherent system, rather than an incremental accretion of very specific and inflexible tools, as is frequently the case with other data analysis software.

The Comprehensive R Archive Network (CRAN) hosts thousands of packages for R at https://cran.r-project.org/web/packages/, so does GitHub (see https://github.com/search?utf8=%E2%9C%93&q=stars%3A%3E1+language%3AR) as well as Bioconductor as package repositories. You can see all the packages from these repositories for R at http://www.rdocumentation.org/ (11 885 packages as of 2016).

As per the author, R is both a language in statistics as well as computer science and an analytics software with great usefulness in analyzing business data and applying data science to it. In particular the appeal of R remains: it is a free open source and has a huge number of packages particularly dealing with analysis of data.

Disadvantages of R remain memory handling in production environments, lack of incentives for R developers, and a sometimes turgid documentation that is mildly academic oriented rather than enterprise user oriented.

1.3 What Is Data Science?

Data science lies at the intersection of programming, statistics, and business analysis. It is the use of programming tools with statistical techniques to analyze data in a systematic and scientific way. A famous diagram by Drew Conway put data science as the intersection of the three. It is given at http://drewconway.com/zia/2013/3/26/the-data-science-venn-diagram

The author defines a data scientist as follows:

> A data scientist is simply a person who can **write code** (in languages like R, Python, Java, SQL, Hadoop (Pig, HQL, MR) etc.) **for data** (storage, querying, summarization, visualization) **efficiently** and **quickly on hardware** (local machines, on databases, on cloud, on servers) and **understand enough statistics** to derive **insights** from **data** so business can make **decisions.**

1.4 The Future for Data Scientists

The respectable *Harvard Business Review* defines data scientist to be the sexiest job of the twenty-first century (https://hbr.org/2012/10/data-scientist-the-sexiest-job-of-the-21st-century/).

Surveys on salaries point out to both rising demand and salaries for data scientists and a big shortage for trained professionals (see http://www.forbes.com/sites/gilpress/2015/10/09/the-hunt-for-unicorn-data-scientists-lifts-salaries-for-all-data-analytics-professionals/). Indeed this has coined a new term unicorn data scientists. A unicorn data scientist is rare to find for he has all the skills in programming, statistics, and business aptitude. A modification of the Data Science Venn Diagram in Figure 1.1 is available at http://www.anlytcs.com/2014/01/data-science-venn-diagram-v20.html, which the author found more updated.

In addition, unicorn is a term in the investment industry, and in particular the venture capital industry, which denotes a start-up company whose valuation has exceeded $1 billion. The term has been popularized by Aileen Lee of Cowboy Ventures. They can be seen at http://graphics.wsj.com/billion-dollar-club/ and http://fortune.com/unicorns/

Not surprisingly data science offers a critical edge to these start-ups as well. So we can have both rising demand and short supply of data scientists, leading to a more secure work environment. A list of start-ups can be seen at Y Combinator at http://yclist.com/ including data science related start-ups. You can see a survey here on data scientist salaries at http://www.burtchworks.com/2015/07/14/compensation-of-data-scientists-insights-from-the-past-year. The annual Rexer Analytics survey helps gauge skills and usage by data miners. You can read an interview at http://decisionstats.com/2013/12/25/

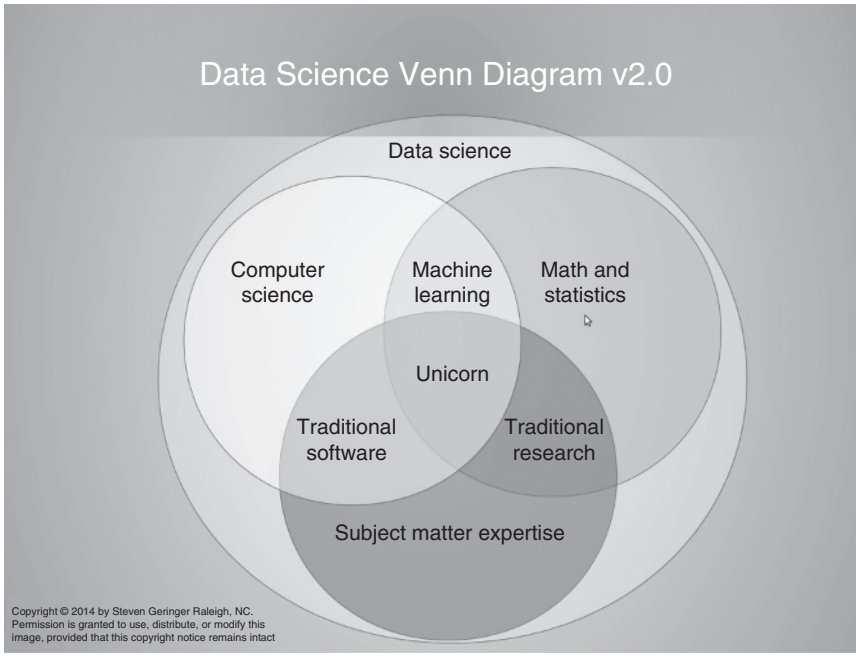

Figure 1.1 Data Science Venn diagram. *Source:* Copyright © 2014 Steven Geringer Raleigh, NC.

karl-rexer-interview-on-the-state-of-analytics/ or read the report at www.rexeranalytics.com. We can thus sum up and say that data scientists who have the right skills have a great future ahead professionally.

A note of caution is that skills need to be updated by data scientists very quickly and they need to be responsive to business needs to frame the data science solutions. So the risk of being obsolete remains an encouragement for data scientists to get multiple skills. An interesting fellowship program for data scientists is run by Insight at http://insightdatascience.com/, and a repository for data science is available for free at https://github.com/okulbilisim/awesome-datascience

Closer home, the NY-based Byte academy offers a Python-based program for data science at http://byteacademy.co/

1.5 What Is Big Data?

Big data is a broad term for datasets so large or complex that traditional data processing applications are inadequate. The 3Vs model helps with understanding big data.

These are:

1) Volume (size and scale of data)
2) Velocity (streaming or data refresh rate)
3) Variety (type: structured or unstructured) of data

The fourth V is veracity.

Typical approaches to deal with big data are hardware based, and use distributed computing, parallel processing, cloud computing, and specialized software like Hadoop stack. An interesting viewpoint to big data is given at https://peadarcoyle.wordpress.com/2015/08/02/interview-with-a-data-scientist-hadley-wickham/ by Dr. Hadley Wickham, a noted R scientist:

> There are two particularly important transition points:
>
> * From in-memory to disk. If your data fits in memory, it's small data. And these days you can get 1 TB of ram, so even small data is big! Moving from in-memory to on-disk is an important transition because access speeds are so different. You can do quite naive computations on in-memory data and it'll be fast enough. You need to plan (and index) much more with on-disk data
> * From one computer to many computers. The next important threshold occurs when you data no longer fits on one disk on one computer. Moving to a distributed environment makes computation much more challenging because you don't have all the data needed for a computation in one place. Designing distributed algorithms is much harder, and you're fundamentally limited by the way the data is split up between computers.

Wes McKinney, the author of pandas, the primary Python package for data science, has this to offer on http://wesmckinney.com/blog/the-problem-with-the-data-science-language-wars/

> "any data processing engine that allows you to extend it with user-defined code written in a "foreign language" like Python or R has to solve at least these 3 essential problems:
>
> - Data movement or access: making runtime data accessible in a form consumable by Python, say. Unfortunately, this often requires expensive serialization or deserialization and may dominate the system runtime. Serialization costs can be avoided by carefully creating shared byte-level memory layouts, but doing this requires a lot of experienced and well-compensated people to agree to make major engineering investments for the greater good.
> - Vectorized computation: enabling interpreted languages like Python or R to amortize overhead and calling into fast compiled code that is

array-oriented (e.g. NumPy or pandas operations). Most libraries in these languages also expect to work with array / vector values rather than scalar values. So if you want to use your favorite Python or R packages, you need this feature.
- IPC overhead: the low-level mechanics of invoking an external function. This might involve sending a brief message with a few curt instructions over a UNIX socket."

The author defines big data as data that requires more hardware (Cloud et al.) or more complicated programming or specialized software (Hadoop) than small data.

1.6 Business Analytics Versus Data Science

The author found the historical evolution from statistical computing to business analytics (BA) to data science both fascinating and amusing in the various claims of hegemonic superiority. This is how he explains it to his students and readers.

1.6.1 Defining Analytics

Analytics is the systematic computational analysis of data or statistics. It is the discovery and communication of meaningful patterns in data. Especially valuable in areas rich with recorded information, analytics relies on the simultaneous application of statistics, computer programming, and operations research to quantify performance.

The **information ladder** was created by education professor Norman Longworth to describe the stages in human learning. According to the ladder, a learner moves through the following progression to construct "wisdom" from "data":

Data → Information → Knowledge → Understanding → Insight → Wisdom

BA refers to the skills, technologies, and practices for continuous iterative exploration and investigation of past business performance to gain insight and drive business planning.

Data analytics (DA) is the science of examining raw data with the purpose of drawing conclusions about that information.

Citation from http://www.gartner.com/it-glossary/analytics

Data science is a more recent term and implies much more programming complexity:

Data Science = programming + statistics + business knowledge

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

Business intelligence (BI) is an umbrella term that includes the applications, infrastructure and tools, and best practices that enable access to and analysis of information to improve and optimize decisions and performance.

Overall the most important thing should be assistance to decision-making rendered not just the science of data analysis.

1.7 Tools Available to Data Scientists

Some (and not all) of the widely used tools available to data scientists are the following:

- Data storage—MySQL, Oracle, SQL Server, HBase, MongoDB, and Redis
- Data querying—SQL, Python, Java, and R
- Data analysis—SAS, R, and Python
- Data visualization—JavaScript, R, and Python
- Data mining—Clojure, R, and Python
- Cloud—Amazon AWS, Microsoft Azure, and Google Cloud
- Hadoop Big Data—Spark, HDFS MapReduce (Java), Pig, Hive, and Sqoop

A cheat sheet is a piece of paper bearing written notes intended to aid one's memory. It can also be defined as a compilation of mostly used commands to help you learn that language's syntax at a faster rate. To help with remembering syntax for many tools, cheat sheets can be useful for data scientists.

The author has written an article on KDnuggets on cheat sheets for data science at http://www.kdnuggets.com/2014/05/guide-to-data-science-cheat-sheets.html where he elaborates on his philosophy of what is a data scientist or not.

1.7.1 Guide to Data Science Cheat Sheets

Selection of the most useful Data Science cheat sheets, covering SQL, Python (including NumPy, SciPy, and Pandas), R (including Regression, Time Series, Data Mining), MATLAB, and more. By Ajay Ohri, May 2014.

Over the past few years, as the buzz and apparently the demand for data scientists has continued to grow, people are eager to learn how to join, learn, advance, and thrive in this seemingly lucrative profession. As someone who writes on analytics and occasionally teaches it, I am often asked—How do I become a data scientist?

Adding to the complexity of my answer is data science seems to be a multi-disciplinary field, while the university departments of statistics, computer science, and management deal with data quite differently.

But to cut the marketing created jargon aside, a data scientist is simply a person who can write code in a few languages (primarily R, Python, and SQL)

for data querying, manipulation, aggregation, and visualization using enough statistical knowledge to give back actionable insights to the business for making decisions.

Since this rather practical definition of a data scientist is reinforced by the accompanying words on a job website for "data scientists," ergo, here are some tools for learning the primary languages in data science—Python, R, and SQL.

A cheat sheet or reference card is a compilation of mostly used commands to help you learn that language's syntax at a faster rate. The inclusion of SQL may lead to some to feel surprised (isn't this the NoSQL era?), but it is there for a logical reason. Both PIG and Hive Query Language are closely associated with SQL—the original Structured Query Language. In addition one can solely use the sqldf package within R (and the less widely used python-sql or python-sql-parse libraries for Pythonic data scientists) or even the Proc SQL commands within the old champion language SAS and do most of what a data scientist is expected to do (at least in data munging).

Python Cheat Sheets is a rather partial list given the fact that Python, the most general-purpose language within the data scientist quiver, can be used for many things. But for the data scientist, the packages of NumPy, SciPy, pandas, and scikit-learn seem the most pertinent.

Do all the thousands of R packages have useful interest to the aspiring data scientist? No.

Accordingly we chose the appropriate cheat sheets for you. Note that this is a curated list of lists. If there is anything that can be assumed in the field of data science, it should be that the null hypothesis is that the data scientist is intelligent enough to make his own decisions based on data and its context. Three printouts are all it takes to speed up the aspiring data scientist's journey.

You can also view the presentation on SlideShare at http://www.slideshare.net/ajayohri/cheat-sheets-for-data-scientists that has more than 8000 views.

1.8 Packages in Python for Data Science

Some useful packages for data scientists in Python are as follows:

- **pandas**—A software library written for data structures, data manipulation, and analysis in Python.
- **NumPy**—Adds Python support for large, multidimensional arrays and matrices, along with a large library of high-level mathematical functions to operate on these arrays.
- **IPython Notebook(s)**—Demonstrates Python functionality geared toward data analysis.
- **SciPy**—A fundamental library for scientific computing.

- **Matplotlib**—A comprehensive 2D plotting for graphs and data visualization.
- **Seaborn**—A Python visualization library based on matplotlib. It provides a high-level interface for drawing attractive statistical graphics.
- **scikit-learn**—A machine learning library.
- **statsmodels**—For building statistical models.
- **Beautiful Soup**—For web scraping.
- **Tweepy**—For Twitter scraping.
- **Bokeh** (http://bokeh.pydata.org/en/latest/)—A Python interactive visualization library that targets modern web browsers for presentation. Its goal is to not only provide elegant, concise construction of novel graphics in the style of D3.js but also deliver this capability with high-performance interactivity over very large or streaming datasets. It has interfaces in Python, Scala, Julia, and now R.
- **ggplot** (http://ggplot.yhathq.com/)—A plotting system for Python based on R's ggplot2 and the Grammar of Graphics. It is built for making professional-looking plots quickly with minimal code.

For R the best way to look at packages is see CRAN Task Views (https://cran.r-project.org/web/views/) where the packages are aggregated by usage type. For example, the CRAN Task View on High Performance Computing is available at https://cran.r-project.org/web/views/HighPerformanceComputing.html.

1.9 Similarities and Differences between Python and R

- Python is used in a wide variety of use cases unlike R that is mostly a language for statistics.
- Python has two versions: Python 2 (or 2.7) and Python 3 (3.4). This is not true in R that has one major release.
- R has very good packages in data visualization and data mining and so does Python. R however has a large number of packages that can do the same thing, while Python generally focuses on adding functions to same package. This is both a benefit in terms of options available and a disadvantage in terms of confusing the beginner. Python has comparatively fewer packages (like statsmodels and scikit-learn for data mining).
- Communities differ in terms of communication and interaction. The R community uses the #rstats on Twitter (see https://twitter.com/hashtag/rstats) to communicate.
- R has an R Journal at https://journal.r-project.org/, and Python has a journal at *Python Papers* (http://ojs.pythonpapers.org/). In addition there is a *Journal of Statistical Software* (http://www.jstatsoft.org/index).

1.9.1 Why Should R Users Learn More about Python?

A professional data scientist should hedge his career by not depending on just one statistical computing language. The ease at which a person can learn a new language does decrease with age, and it's best to base your career on more than R. SAS language did lead the world for four decades, but in a fast-changing world, it is best not to bet your mortgage that R skills are all you need for statistical computing in a multi-decade career.

1.9.2 Why Should Python Users Learn More about R?

R will continue to have the maximum number of packages in statistics data science and visualization. Since R is also open source and free, it is best to prototype your solution in R than use Python for scaling up in production environment.

An interesting viewpoint is given at http://www.kdnuggets.com/2015/05/r-vs-python-data-science.html by a founder of DataCamp and at http://multithreaded.stitchfix.com/blog/2015/03/17/grammar-of-data-science/

1.10 Tutorials

A notebook by Radim Rehurek on data science with Python with code and output is available at http://radimrehurek.com/data_science_python/.

A good list of notebooks in data science for Python is also available at https://github.com/donnemartin/data-science-ipython-notebooks.

More general knowledge on data science-related activities in Python can be found at https://github.com/okulbilisim/awesome-datascience.

For more learning on data science, see http://datasciencespecialization.github.io/. It has all nine courses in the Coursera Data Science Specialization from Johns Hopkins University.

It has the following courses:

- The Data Scientist's Toolbox
- R Programming
- Getting and Cleaning Data
- Exploratory Data Analysis
- Reproducible Research
- Statistical Inference
- Regression Models
- Practical Machine
- Learning Developing Data Products

1.11 Using R and Python Together

The author has helped create a SlideShare ppt on a side-by-side comparison of R and Python syntax for data science at http://www.slideshare.net/ajayohri/python-for-r-users (35 000 + views). However a guide for using Python and R for quantitative finance is also found at http://www.slideshare.net/lsbardel/python-and-r-for-quantitative-finance-2409526

Additionally the following methods help to use both R and Python and leverage their tremendous strengths:

1) RPy2 RPy2 helps in using R and Python together. The official documentation is given at http://rpy.sourceforge.net/rpy2/doc-dev/html/introduction.html. The object r in rpy2.robjects represents the running embedded R process. If familiar with R and the R console, r is a little like a communication channel from Python to R.

A lucid example of using RPy2 is given here at A Slug's Guide to Python (https://sites.google.com/site/aslugsguidetopython/data-analysis/pandas/calling-r-from-python):

```
from pandas import*
from rpy2.robjects.packages import importr
import rpy2.robjects as ro
import pandas.rpy.common as com
```

We can pass commands to the R session by putting the R syntax within the ro.r() method as strings, and we can read the R data.frame into pandas data frame with com.load_data method. We can then pass the pandas data frame back to the R instance by first converting pydf to an R data frame by using com.convert_to_r_dataframe method.

A truncated screenshot of the website is given in Figure 1.2 to help the reader understand and refer back to https://sites.google.com/site/aslugsguidetopython/data-analysis/pandas/calling-r-from-python

1.11.1 Using R Code for Regression and Passing to Python

An example of using rpy2 and caret package in R is given for kaggle at https://www.kaggle.com/c/bike-sharing-demand/forums/t/12923/rpy2-caret-example

```
a caret use from python environment:
import pandas.rpy.common as com
import rpy2
```

```
from rpy2.robjects.packages import importr
graphics = importr('graphics')
grdevices = importr('grDevices')
base = importr('base')
stats = importr('stats')

import array

x = array.array('i', range(10))
y = stats.rnorm(10)

grdevices.X11()

graphics.par(mfrow = array.array('i', [2,2]))
graphics.plot(x, y, ylab = "foo/bar", col = "red")

kwargs = {'ylab':"foo/bar", 'type':"b", 'col':"blue", 'log':"x"}
graphics.plot(x, y, **kwargs)


m = base.matrix(stats.rnorm(100), ncol=5)
pca = stats.princomp(m)
graphics.plot(pca, main="Eigen values")
stats.biplot(pca, main="biplot")
```

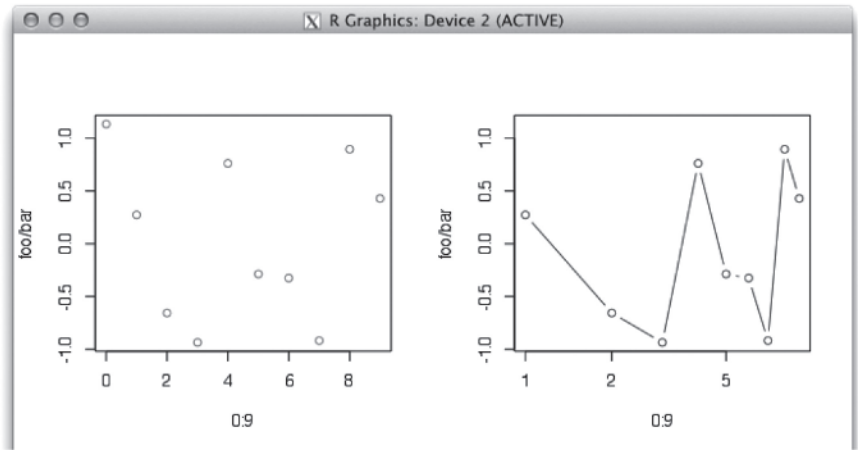

Figure 1.2 Using R code for regression and passing to Python.

```
import rpy2.robjects as ro
from rpy2.robjects import Formula
from rpy2.robjects.packages import importr
caretr = importr("caret")
data_trainr = com.convert_to_r_dataframe(data_train)
param1 = {'method' : 'repeatedcv', 'number' : 3,
'repeats' : 5}
ctrl = caretr.trainControl(**param1)
param2 = {'method' : 'rf', 'trControl' : ctrl}
rf_for = Formula("log(casual + 1) ~ dm + t + wd + tp +
hum + ws")
```

```
rfmod = caretr.train(rf_for, data = data_trainr,
**param2)
print(rfmod)
```

A better but slightly old demo of using R and Python together in rpy2 is given at http://www.bytemining.com/wp-content/uploads/2010/10/rpy2.pdf

Another good example is given by Laurent Gautier in her talk "Polyglot applications with R and Python [BARUG Meeting]" at http://files.meetup.com/1225993/Laurent%20Gautier_R_toPython_bridge_to_R.pdf#!

A minimal example of rpy2 regression using pandas data frame is given at Stack Overflow at http://stackoverflow.com/questions/30922213/minimal-example-of-rpy2-regression-using-pandas-data-frame

```
from rpy2.robjects import pandas2ri
pandas2ri.activate()
robjects.globalenv['dataframe'] = dataframe
M = stats.lm('y~x', data=base.as_symbol('dataframe'))
```

The result is:

```
>>> print(base.summary(M).rx2('coefficients'))
      Estimate Std. Error  t value  Pr(>|t|)
(Intercept) 0.6  1.1489125 0.522233 0.6376181
x        0.8  0.3464102 2.309401 0.1040880
```

CONDA—Conda is an open-source package management system and environment management system for installing multiple versions of software packages and their dependencies and switching easily between them. It works on Linux, OS X, and Windows and was created for Python programs but can package and distribute any software. Using conda we can use Python and R together. We can then use the familiar interface of Jupyter/IPython Notebook. You can refer to https://www.continuum.io/conda-for-r.

You can see the demo for R within Jupyter at https://try.jupyter.org/. A good blog post on using Jupyter to R is found at https://www.continuum.io/blog/developer/jupyter-and-conda-r.

The Anaconda team has created an "R Essentials" bundle with the IRkernel and over 80 of the most used R packages for data science, including dplyr, shiny, ggplot2, tidyr, caret, and nnet.

Once you have conda, you may install "R Essentials" into the current environment:

```
conda install -c r r-essentials
  Bash
```

or create a new environment just for "R essentials":

```
conda create -n my-r-env -c r r-essentials
```

(https://www.continuum.io/content/preliminary-support-r-conda)
conda create -c r -n r r will download R from our official R channel on Anaconda.org:

- Revolution Analytics—A Microsoft company that is one of the leading vendors of R has a blog post on this at http://blog.revolutionanalytics.com/2015/09/using-r-with-jupyter-notebooks.html
- Official documentation is also at http://conda.pydata.org/docs/r-with-conda.html

DOCKER—Docker is an open platform for developers and sysadmins to build, ship, and run distributed applications, whether on laptops, data center VMs, or the cloud.

- You can use Docker to run Jupyter. This is available at https://hub.docker.com/r/jupyter/datascience-notebook/ and https://github.com/jupyter/docker-stacks

A good discussion for Docker is given at http://stackoverflow.com/questions/16047306/how-is-docker-different-from-a-normal-virtual-machine. I am reproducing a part of the technical answer in the following text.

Docker was using Linux Containers (LXC) earlier but switched to runC (formerly known as libcontainer) that runs in the same operating system as its host. **This allows it to share a lot of the host operating system resources.** It also uses layered file systems like AuFS. It also manages the networking for you as well.

AuFS is a layered file system, so you can have a read-only part and a write part and merge those together. So you could have the common parts of the operating system as read only, which are shared among all of your containers, and then give each container its own mount for writing.

So let's say you have a container image that is 1GB in size. If you wanted to use a full VM, you would need to have 1GB times × number of VMs you want. With LXC and AuFS you can share the bulk of the 1GB, and if you have 1000 containers, you still might only have a little over 1GB of space for the container OS, assuming they are all running the same OS image.

A full virtualized system gets its own set of resources allocated to it and does minimal sharing. You get more isolation, but it is much heavier (requires more resources).

With LXC you get less isolation, but they are more lightweight and require fewer resources. So you could easily run 1000's on a host.

You can build your own docker environment for data science.

For Jupyter and Docker, see "A data science environment in minutes using Docker and Jupyter" at https://www.dataquest.io/blog/data-science-quickstart-with-docker/, and for Docker and R, you can use the instructions and file at https://hub.docker.com/r/library/r-base/. The official R base is available at https://store.docker.com/images/f2e50720-cada-432f-85a5-1ade438d537b?tab=description, and you can just copy and paste to pull the image

docker pull r-base

- Python and R together using Beaker—You can use Python and R together using Jupyter, Rpy2, or Beaker. While Jupyter and rpy2 have been covered before, we can also use Beaker. You can use R Python and JavaScript within the same notebook in Beaker (and other languages too).

You can see examples here http://beakernotebook.com/examples and read about it at http://blog.dominodatalab.com/interactive-data-science/ and https://github.com/twosigma/beaker-notebook

1.12 Other Software and Python

- SAS and Python—You can use the SAS language to talk to both Python and R. This is done using Java (passed to the Java class SASJavaExec using the Base SAS Java Object). More specifically you can see the instructions at https://github.com/sassoftware/enlighten-integration/ and https://communities.sas.com/docs/DOC-10746

1.13 Using SAS with Jupyter

You can also use SAS from within Jupyter (Figure 1.3; see http://blogs.sas.com/content/sasdummy/2016/04/24/how-to-run-sas-programs-in-jupyter-notebook/).

1.14 How Can You Use Python and R for Big Data Analytics?

Big data is synonymous with Hadoop. For using Python with Hadoop, you can use the following packages:

1) Hadoop Streaming
2) mrjob
3) dumbo
4) hadoopy
5) pydoop

```
In [3]: ods html5 style=statistical;
        ods graphics /width=500 height=400;
        proc means data=sashelp.cars; run;
        proc sgplot data=sashelp.cars;
            histogram msrp;
            density msrp;
        run;
```

Out[3]:

The SAS System

The MEANS Procedure

Variable	Label	N	Mean	Std Dev	Minimum	Maximum
MSRP		428	32774.86	19431.72	10280.00	192465.00
Invoice		428	30014.70	17642.12	9875.00	173560.00
EngineSize	Engine Size (L)	428	3.1967290	1.1085947	1.3000000	8.3000000
Cylinders		426	5.8075117	1.5584426	3.0000000	12.0000000
Horsepower		428	215.8855140	71.8360316	73.0000000	500.0000000
MPG_City	MPG (City)	428	20.0607477	5.2382176	10.0000000	60.0000000
MPG_Highway	MPG (Highway)	428	26.8434579	5.7412007	12.0000000	66.0000000
Weight	Weight (LBS)	428	3577.95	758.9832146	1850.00	7190.00
Wheelbase	Wheelbase (IN)	428	108.1542056	8.3118130	89.0000000	144.0000000
Length	Length (IN)	428	186.3621495	14.3579913	143.0000000	238.0000000

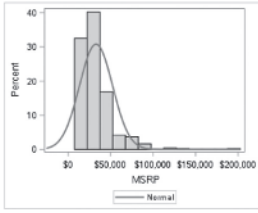

Figure 1.3 Using SAS from within Jupyter Notebook. *Source:* Chris Hemedinger on The SAS Dummy, SAS Institute. Reproduced with the permission of SAS Institute Inc.

An example is given at https://blog.cloudera.com/blog/2013/01/a-guide-to-python-frameworks-for-hadoop/

A recent innovation is Apache Arrow (see https://blog.cloudera.com/blog/2016/02/introducing-apache-arrow-a-fast-interoperable-in-memory-columnar-data-structure-standard/). As per the article, "For the Python and R communities, Arrow is extremely important, as data interoperability has been one of the biggest roadblocks to tighter integration with big data systems (which largely run on the JVM)."

The next innovation is Feather (see https://blog.rstudio.org/2016/03/29/feather/). Feather is a fast, lightweight, and easy-to-use binary file format for storing data frames, and Feather files are the same whether written by Python or R code. The Python interface uses Cython to expose Feather's C++11 core to users, while the R interface uses Rcpp for the same task.

1.15 What Is Cloud Computing?

The official definition of cloud computing is given at http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf

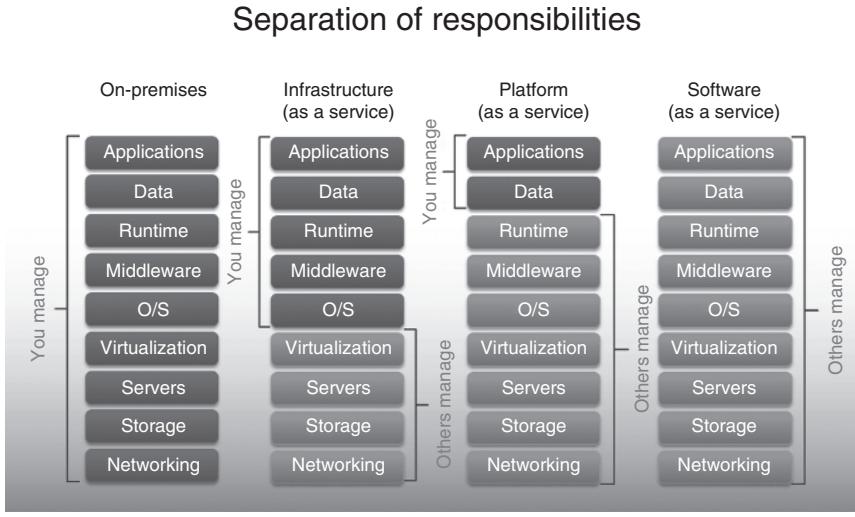

Figure 1.4 The difference between infrastructure as service, platform as a service, and software as a service. *Source:* https://blogs.technet.microsoft.com/kevinremde/2011/04/03/saas-paas-and-iaas-oh-my-cloudy-april-part-3/. © Microsoft.

Cloud computing is a model for enabling:

1) Ubiquitous, convenient on-demand network access
2) A shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction

Amazon (EC2), Google, Oracle, IBM, and Microsoft Azure are some examples of cloud providers. For a data scientist, it is important to know the difference between Infrastructure as a Service, Platform as a Service, and Software as a Service (Figure 1.4).

1.16 How Can You Use Python and R on the Cloud?

If you want to host and run Python in the cloud, these implementations may be right for you: PythonAnywhere (freemium hosted Python installation that lets you run Python in the browser, e.g., for tutorials, showcases, etc.). It has an additional use case for education.

From https://www.pythonanywhere.com/details/education, Python is a great language for teaching, but getting it installed and set up on all your students' computers can be less than easy. PythonAnywhere provides an

environment that's ready to go—including a syntax-highlighting, error-checking editor and Python 2 and 3 consoles.

You can use web scraping from Python on the cloud through http://scrapinghub.com/scrapy-cloud/. Scrapy is the most popular and advanced web crawling framework for Python. It makes writing web crawlers fast, easy, and fun. However, you still need to deploy and run your crawler periodically, manage servers, monitor performance, review scraped data, and get notified when spiders break. This is where Scrapy Cloud comes in.

Additionally, you can run RStudio Server on the cloud if you prefer the RStudio interface using the instructions at http://www.louisaslett.com/RStudio_AMI/. As of May 2016 there is experimental support for Julia (and Python).

1.17 Commercial Enterprise and Alternative Versions of Python and R

Two principal commercial distributions of Python for data scientists are as follows:

- Anaconda from Continuum Analytics (https://www.continuum.io/downloads)

  Anaconda is a completely free Python distribution (including for commercial use and redistribution). It includes more than 300 of the most popular Python packages for science, math, engineering, and data analysis.
- Enthought Canopy (https://www.enthought.com/products/canopy/)

  Enthought Canopy is a Python analysis environment that provides easy installation of the core scientific analytic and scientific Python packages.

A number of alternative implementations are also available (see https://www.python.org/download/alternatives/):

- IronPython (Python running on.NET).
- Jython (Python running on the Java virtual machine).
- PyPy (http://pypy.org/). PyPy is a fast, compliant alternative implementation of the Python language (2.7.10 and 3.2.5). It has several advantages in terms of speed and distinct features but is currently trying to port NumPy package (NumPy is the basic package for many numerical operations in Python).
- Stackless Python (branch of CPython supporting microthreads).

Some repackagings of Python are the following:

- ActiveState ActivePython (commercial and community versions, including scientific computing modules)

- pythonxy (scientific-oriented Python Distribution based on Qt and Spyder)
- winpython (WinPython is a portable scientific Python distribution for Windows)
- Conceptive Python SDK (targets business, desktop, and database applications)
- PyIMSL Studio (a commercial distribution for numerical analysis—free for noncommercial use)
- eGenix PyRun (a portable Python runtime, complete with stdlib, frozen into a single 3.5–13 MB executable file)

In addition there is Cython (http://cython.org/), an optimizing static compiler for both the Python programming language and the extended Cython programming language (based on Pyrex). It makes writing C extensions for Python easier:

- For R the commercial versions are by Revolution Analytics, a Microsoft-acquired subsidiary (www.revolutionanalytics.com). Revolution Analytics makes RevoScaleR package that helps scale up to bigger datasets. RStudio also makes software around R (including Shiny Package and a widely used IDE at www.rstudio.com).
- Renjin is a JVM-based interpreter for the R language for statistical computing (http://www.renjin.org/).
- pqR, a pretty quick version of R (http://www.pqr-project.org/), is a new version of the R interpreter. It is based on R-2.15.0 later versions distributed by the R Core Team (at r-project.org).
- Oracle R Enterprise (ORE) (see https://blogs.oracle.com/R/) (http://www.oracle.com/technetwork/database/database-technologies/r/r-enterprise/overview/index.html). Oracle R Enterprise, a component of the Oracle Advanced Analytics Option, makes the open-source R statistical programming language and environment ready for the enterprise and big data. Designed for problems involving large volumes of data, it integrates R with Oracle Database. R users can run R commands and scripts for statistical and graphical analyses on data stored in Oracle Database.
- **TIBCO Enterprise Runtime for R (TERR)** (http://spotfire.tibco.com/discover-spotfire/what-does-spotfire-do/predictive-analytics/tibco-enterprise-runtime-for-r-terr). TERR, a key component of Spotfire Predictive Analytics, is an enterprise-grade analytic engine that TIBCO has built from the ground up to be fully compatible with the R language, leveraging our long-time expertise in the closely related S+ analytic engine. This allows customers not only to continue to develop in open source R but also to then integrate and deploy their R code on a commercially supported and robust platform.

1.17.1 Commonly Used Linux Commands for Data Scientists

It is important for the budding data scientist to learn the right operating system before a language; hence here are some Linux tips:

- **ls**—Directory listing
- **cd dir**—Change directory to dir
- **mkdir dirname**—Makes a directory named dirname
- **cd**—Change to home
- **sudo**—Gives superuser or admin rights
- **sudo bash**—Changes to root
- **pwd**—Shows present working directory
- **rm filename**—Removes file named filename
- **cat > filename**—Puts standard output in a file
- **cp filename1 filename2**—Copies filename1 to filename2
- **mv filename1 filename2**—Moves filename1 to filename2

Refer to http://www.linuxstall.com/linux-command-line-tips-that-every-linux-user-should-know/ and http://i0.wp.com/www.linuxstall.com/wp-content/uploads/2012/01/linux-command-line-cheat-sheet.png (Figure 1.5).

1.17.2 Learning Git

Git is a version control system that enables teams to work together on projects as well as share code. GitHub is a popular website for sharing packages and libraries under development in R.

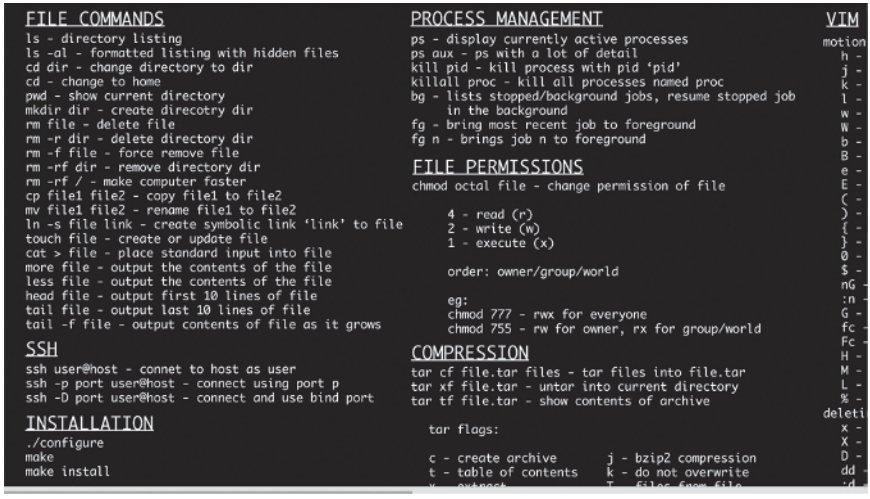

Figure 1.5 Linux cheat sheet.

The following cheat sheet will help you get started in Git (http://overapi.com/static/cs/git-cheat-sheet.pdf) (Figures 1.6 and 1.7). You can test your knowledge by a tutorial at https://try.github.io/levels/1/challenges/1. Lastly the author believes the best way to learn Git is to start contributing to a project.

Figure 1.6 Git cheat sheet. *Source:* © Github.

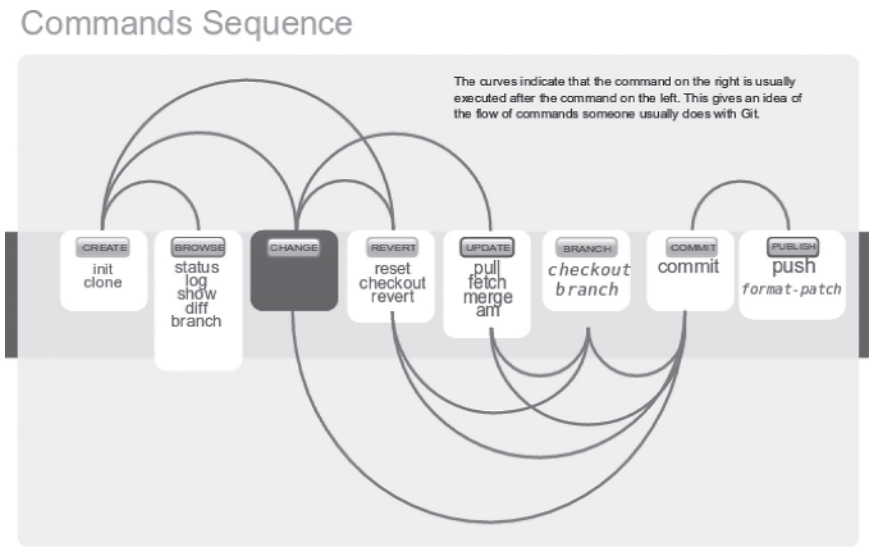

Figure 1.7 Git command sequence. *Source:* © Github.

Let's begin learning the basics of Python (https://nbviewer.jupyter.org/gist/decisionstats/ce2c16ee98abcf328177).

Bold font is code; normal font is output.

Numerical Operations

```
2+3+5
```

```
10
```

```
66-3-(-4)
```

```
67
```

```
32*3
```

```
96
```

```
2**3 #2 raised to power of 3
```

```
8
```

```
43/3
```

```
14.333333333333334
```

```
32//3 #Gives quotient
```

```
10
```

```
44%3 #Gives remainder
```

```
2
```

For R it is almost the same except for the last two. There the syntax is:

```
32 % / % 3 #Gives quotient
44 % % 3 #Gives remainder
```

For Loops

See https://docs.python.org/3/tutorial/controlflow.html

```
#numbers from 0 to 30 increment 6
for x in range(0, 30, 6):
    print (x)
```

```
0
6
12
18
24
```

For Loops can be slightly different in R. Note the + sign denotes a new line in R code:

```
for(i in seq(0,30,6)){
+   print(i)
+ }
[1] 0
[1] 6
[1] 12
[1] 18
[1] 24
[1] 30
```

Functions

```
def myfirstfunction(x):
  y=x**3+3*x+20
  print(y)

myfirstfunction(20)
```

```
8080
```

In R creating a function would be different. You would need to use a function like function(x) as follows, then write the function within brackets, and print out the value. This is because R like most computer languages does not use

space indentation. In **R**, you can view a **function's** code by typing the **function** name without the (). This is especially useful to see the algorithms in an existing package and to tweak it if possible:

```
myfirstfunction =function(x){
  y=x**3+3*x+20
print(y)}

myfirstfunction(20)

for x in range(0,30,6):
     myfirstfunction (x)
```

```
20
254
1784
5906
13916
```

```
def mynewfunction(x,y):
  z=x**3+3*x*y+20*y
  print(z)

mynewfunction(1,3)
```

```
70
```

```
mynewfunction(10,3)
```

```
1150
```

See http://rpubs.com/ajaydecis/forfunctions, https://docs.python.org/2/library/functions.html, and http://stackoverflow.com/questions/7969949/whats-the-difference-between-globals-locals-and-vars

- globals()—*Always* returns the dictionary of the *module* namespace
- locals()—*Always* returns *a* dictionary of the *current* namespace
- vars()—Returns *either a* dictionary of the current namespace (if called with no argument) or *the* dictionary of the argument

```
locals() #gives objects in local space
```

```
{'In': ['',
  '2+3+5',
  '66-3-(-4)',
```

```
  '2^3',
  '3^3',
  '44%%3',
  '44%3',
  '43/3',
  '32*3',
  '2**3',
..... truncated by author
globals()

{'In': ['',
  '2+3+5',
  '66-3-(-4)',
  '2^3',
  '3^3',
  '44%%3',
  '44%3',
  '43/3',
  '32*3',
  '2**3',
  '32//3',
  'for i in 1:30\n   print i',
  'for i in 1:30:\n  print i',
  'for i in 1:30:\n  print i',
  'for i in range(1,30):\n  print i',
  'for i in range(1,30):\n  print i',
  'for i in range(1,30):\n  print %i',
  'for i in range(1,30):\n  print %(i)',
  'for i in range(1,30):\n  print % (i)',
  'for i in range(1,30):\n  print % (i)',
  'for x in range(0,30):\n  print % (x)',
 ------truncated by author
```

More Numerical Operations

```
import math

math.exp(2)

7.38905609893065
math.log(2)

0.6931471805599453
math.log(2,10)
```

```
0.30102999566398114
math.sqrt(10)

3.1622776601683795
dir(math) #dir gives all the identifiers a module defines

['__doc__',
 '__file__',
 '__loader__',
 '__name__',
 '__package__',
 '__spec__',
 'acos',
 'acosh',
 'asin',
 'asinh',
 'atan',
 'atan2',
 'atanh',
 'ceil',
 'copysign',
 'cos',
 'cosh',
 'degrees',
 'e',
 'erf',
 'erfc',
 'exp',
 'expm1',
 'fabs',
 'factorial',
 'floor',
 'fmod',
 'frexp',
 'fsum',
 'gamma',
 'hypot',
 'isfinite',
 'isinf',
 'isnan',
 'ldexp',
 'lgamma',
 'log',
```

```
 'log10',
 'log1p',
 'log2',
 'modf',
 'pi',
 'pow',
 'radians',
 'sin',
 'sinh',
 'sqrt',
 'tan',
 'tanh',
 'trunc']
a=[23,45,78,97,89]

type(a)

list
len(a)

5
max(a)

97
min(a)

23
sum(a)

332
import numpy

numpy.mean(a)

66.400000000000006

numpy.std(a)

28.011426240018555
numpy.var(a)

784.63999999999999
#Example of Help (note the ? is almost the same as R)
numpy.random?
```

```
from random import randint,randrange
print(randint(0,9))

5

randrange(10)

4
for x in range(0,5):
    print(randrange(10))

8
3
7
8
5
```

Strings, Lists, Tuples, and Dicts

```
newstring='Hello World'

newstring

'Hello World'
print(newstring)

Hello World

newstring2='Hello World's'
 File "<ipython-input-56-8c5b85561ed9>", line 1
   newstring2='Hello World's'
                     ^
SyntaxError: invalid syntax

#Double Quotes and Single Quotes
newstring2="Hello World's"

print(newstring2)

Hello World's

#Escape character \
newstring3="Hello, World\'s"

print(newstring3)
```

```
Hello, World's

10*newstring3

"Hello, World'sHello, World'sHello, World'sHello,
World'sHello, World'sHello, World'sHello,
World'sHello, World'sHello, World'sHello World's "
```

Passing Variables in Strings in Python

```
myname1= 'Ajay'
myname2= 'John'
message =" Hi I am %s. How do you do"
message %myname1
' Hi I am Ajay. How do you do'

message %myname2
' Hi I am John. How do you do'

new1= "Why did the %s cross the %s"
print(new1%('chicken','road'))

Why did the chicken cross the road

print(new1%(10,40))
Why did the 10 cross the 40

new2= "Why did the %d cross the %d"

print(new2%('chicken','road'))
---------------------------------------------------------
TypeError            Traceback (most recent call last)
<ipython-input-11-b2f398d16f9c> in <module>()
----> 1 print(new2%('chicken','road'))

TypeError: %d format: a number is required, not str

Note the error caused by %d and %s
```

Lists

```
newnames='ajay,vijay,john,donald,hillary,bill,ashok'

type(newnames)

str
newnames[0:9]
```

```
'ajay,vija'
newnames2=['ajay','vijay','john','donald','hillary',
'bill','ashok']

type(newnames2)

list
```

In R, a list would be created like this:

```
newnames2=c('ajay','vijay','john','donald','hillary',
'bill','ashok')

newnames2[0]

'ajay'
So in R the index starts from 1, while in Python the
index starts with 0.

newnames2[0]='micky mouse' #substituting members in a list

newnames2

['micky mouse', 'vijay', 'john', 'donald', 'hillary',
'bill', 'ashok']
newnames2[2]

'john'
newnames2.append('daisy')

newnames2

['micky mouse', 'vijay', 'john', 'donald', 'hillary',
'bill', 'ashok', 'daisy']

.append to add and del to delete members in a list

del newnames2[2]

newnames2

['micky mouse', 'vijay', 'donald', 'hillary', 'bill',
'ashok', 'daisy']
newlist=[1,2,4,7]
```

```
newnames2+newlist
```

```
['micky mouse',
 'vijay',
 'donald',
 'hillary',
 'bill',
 'ashok',
 'daisy',
 1,
 2,
 4,
 7]
```

```
newlist*3
```

```
[1, 2, 4, 7, 1, 2, 4, 7, 1, 2, 4, 7]
```

```
a tuple is a list that uses parenthese () not square
brackets [] and it CANNOT  be modified at all once
created
scores=(23,46,69,7,5)
```

```
type(scores)
```

```
tuple
```

```
scores[3]
```

```
7
```

```
dir(scores) #dir command gives various operations that
can be done to that object
```

```
['__add__',
 '__class__',
 '__contains__',
 '__delattr__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__getitem__',
 '__getnewargs__',
 '__gt__',
```

```
 '__hash__',
 '__init__',
 '__iter__',
 '__le__',
 '__len__',
 '__lt__',
 '__mul__',
 '__ne__',
 '__new__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__rmul__',
 '__setattr__',
 '__sizeof__',
 '__str__',
 '__subclasshook__',
 'count',
 'Index']
```

```
favourite_movie=['micky mouse,steamboat willie',
'vijay,slumdog millionaire', 'john,passion of christ',
'donald,arthur']
```

```
type(favourite_movie)
```

```
list
```

```
favourite_movie2={'micky mouse':'steamboat
willie','vijay':'slumdog millionaire','john':'passion
of christ','donald':'arthur'}
```

```
type(favourite_movie2)
```

```
dict
favourite_movie2['micky mouse']
```

```
'steamboat willie'
favourite_movie2['vijay']
```

```
'slumdog millionaire'
```

Refer to https://nbviewer.jupyter.org/gist/decisionstats/752ff727101cf6fc13225bd94eef358a for the code in this example.

Strings—We use str function to convert data to string data (we use int to convert data to integer values). We can use *slicing* on index to create substrings and concatenate strings using + sign. The following example shows some of the things that can be done with string data:

```
names=['Ajay','Vijay','Ra Jay','Jayesh']

type(names)

list
names[1]

'Vijay'
type(names[1])

str
names[0][1:3]

'ja'
names[2][2:]

' Jay'
names[2][2:] + names[3][2:]

' Jayyesh'
names[1].lower()

'vijay'
names[2].replace(" ","")

'RaJay'
```

Let's try to do the same thing in R (http://rpubs.com/ajaydecis/strings4). There are important differences we want to highlight:

```
names=c('Ajay','Vijay','Ra Jay','Jayesh')
```

R uses c to make a list. Python does not—but uses square brackets.
Python uses type while R uses class to find out the object's type:

```
class(names)
## [1] "character"
```

Python starts the index from 0 while begins the index of a list from 1:

```
names[1]
## [1] "Ajay"
class(names[1])
## [1] "character"
```

You have to use substr in R to find part of a string. In Python you simply can look this from within square brackets:

```
substr(names[1],2,3)
## [1] "ja"
substr(names[3],3,nchar(names[3]))
## [1] " Jay"
```

While Python simple combined strings using +, R used paste:

```
paste(substr(names[3],3,nchar(names[3])),substr(names
[2],3,nchar(names[2])))
## [1] " Jay jay"
```

R uses **tolower** while Python uses.lower():

```
tolower(names[1])
## [1] "ajay"
```

Python used replace while R used gsub:

```
gsub(" ","",names[3])
## [1] "RaJay"
```

The biggest difference is R mostly uses function(object), while Python uses object.function() to get things done. This is an important difference

File and Folder Operations

In Python we use the os package for file operations to refer and read the file from a particular directory. We also use !pip freeze to get the list of packages (versions). We use print (IPython.sys_info()) and version_information package (%load_ext version_information

%version_information) to get System Information (see Python code at https://nbviewer.jupyter.org/gist/decisionstats/29f3adfb6980db52a61130aa8c8f9166).

In R we get System Information using sessionInfo(). (For R Code see http://rpubs.com/newajay/systeminfo)

```
import IPython
print (IPython.sys_info())
```

```
{'commit_hash': 'c963f6b',
 'commit_source': 'installation',
 'default_encoding': 'UTF-8',
 'ipython_path': '/home/ajayohri/anaconda3/lib/
python3.5/site-packages/IPython',
 'ipython_version': '4.2.0',
 'os_name': 'posix',
 'platform': 'Linux-4.4.0-53-generic-x86_64-with-
debian-stretch-sid',
 'sys_executable': '/home/ajayohri/anaconda3/bin/
python',
 'sys_platform': 'linux',
 'sys_version': '3.5.2 |Anaconda 4.1.1 (64-bit)|
  (default, Jul  2 2016, '
      '17:53:06) \n'
      '[GCC 4.4.7 20120313 (Red Hat 4.4.7-1)]'}
```

```
!pip install version_information
```

The directory "/home/ajayohri/.cache/pip/http" or its parent directory is not owned by the current user, and the cache has been disabled. Please check the permissions and owner of that directory. If executing pip with sudo, you may want sudo's -H flag.

The directory "/home/ajayohri/.cache/pip" or its parent directory is not owned by the current user, and caching wheels has been disabled. Check the permissions and owner of that directory. If executing pip with sudo, you may want sudo's -H flag.

Collecting version_information

Downloading version_information-1.0.3.tar.gz

Installing collected packages: version-information

Running setup.py install for version-information ... - \ | done

Successfully installed version-information-1.0.3

You are using pip version 8.1.2; however version 9.0.1 is available.

You should consider upgrading via the "pip install --upgrade pip" command.

%load_ext version_information

%version_information

Software	Version
Python	3.5.2 64bit [GCC 4.4.7 20120313 (Red Hat 4.4.7-1)]
IPython	4.2.0
OS	Linux 4.4.0 53 generic x86_64 with Debian stretch sid
Sat Dec 24 19:47:41 2016 IST	

!pip freeze

```
alabaster==0.7.8
anaconda-client==1.4.0
anaconda-navigator==1.2.1
argcomplete==1.0.0
astropy==1.2.1
Babel==2.3.3
backports.shutil-get-terminal-size==1.0.0
beautifulsoup4==4.4.1
-------list truncated by author
SQLAlchemy==1.0.13
statsmodels==0.6.1
sympy==1.0
tables==3.2.2
terminado==0.6
toolz==0.8.0
tornado==4.3
traitlets==4.2.1
unicodecsv==0.14.1
version-information==1.0.3
Werkzeug==0.11.10
xlrd==1.0.0
XlsxWriter==0.9.2
xlwt==1.1.2
```

The directory "/home/ajayohri/.cache/pip/http" or its parent directory is not owned by the current user, and the cache has been disabled. Please check the permissions and owner of that directory. If executing pip with sudo, you may want sudo's -H flag.

You are using pip version 8.1.2; however version 9.0.1 is available.

You should consider upgrading via the "pip install --upgrade pip" command.
(Authors note-warning message by system)

```
import os as os
os.getcwd()
'/home/ajayohri/Desktop'
os.chdir('/home/ajayohri/Desktop')
```

```
os.getcwd()
'/home/ajayohri/Desktop'
os.listdir()

['Data Analytics Course: Master Data Analytics Using
Python in 2.5 Months_files',
 'dump 4 nov 2016',
 'Hadoop Tutorial | All you need to know about Hadoop
| Edureka_files',
 'Data Analytics Course: Master Data Analytics Using
Python in 2.5 Months.html',
 'Hadoop Tutorial | All you need to know about Hadoop
| Edureka.html',
 'Note to R Users — Data Analysis in Python 0.1
documentation_files',
 'Note to R Users — Data Analysis in Python 0.1
documentation.html',
 'Jupyter Notebook Viewer.html',
 'py4r.jpg',
 'test',
 'hackerearth',
 'Jupyter Notebook Viewer_files',
 'logo-ds.png']
```

In R this would be slightly different:

```
sessionInfo()
## R version 3.3.1 (2016-06-21)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 16.04.1 LTS
##
## locale:
##  [1] LC_CTYPE=en_IN.UTF-8    LC_NUMERIC=C
##  [3] LC_TIME=en_IN.UTF-8    LC_COLLATE=en_IN.UTF-8
##  [5] LC_MONETARY=en_IN.UTF-8  LC_MESSAGES=en_IN.UTF-8
##  [7] LC_PAPER=en_IN.UTF-8   LC_NAME=C
##  [9] LC_ADDRESS=C        LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_IN.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] stats  graphics grDevices utils  datasets
       methods  base
##
```

```
## loaded via a namespace (and not attached):
## [1] magrittr_1.5  tools_3.3.1  htmltools_0.3.5
       Rcpp_0.12.8
## [5] stringi_1.1.1  rmarkdown_1.0 knitr_1.13
       stringr_1.0.0
## [9] digest_0.6.9    evaluate_0.9

getwd()
## [1] "/home/ajayohri"

setwd("/home/ajayohri/Desktop/")

dir()
##  [1] "Data Analytics Course: Master Data Analytics
       Using Python in 2.5 Months_files"
##  [2] "Data Analytics Course: Master Data Analytics
       Using Python in 2.5 Months.html"
##  [3] "dump 4 nov 2016"
##  [4] "hackerearth"
##  [5] "Hadoop Tutorial | All you need to know about
       Hadoop | Edureka_files"
##  [6] "Hadoop Tutorial | All you need to know about
       Hadoop | Edureka.html"
##  [7] "Jupyter Notebook Viewer_files"
##  [8] "Jupyter Notebook Viewer.html"
##  [9] "logo-ds.png"
## [10] "Note to R Users — Data Analysis in Python 0.1
       documentation_files"
## [11] "Note to R Users — Data Analysis in Python 0.1
       documentation.html"
## [12] "py4r.jpg"
## [13] "test"
```

The following deals with the business part (or domain expertise part) of the decision science triad (programming, statistics, and domain expertise).

1.18 Data-Driven Decision Making: A Note

A fundamental principle of data-driven decision making is a famous quote: If you can't measure it, you can't manage it—Peter Drucker.

As per http://whatis.techtarget.com/definition/data-driven-decision-management-DDDM, data-driven decision management (DDDM) is an approach to business governance that values decisions that can be backed

up with verifiable data. The success of the data-driven approach is reliant upon the quality of the data gathered and the effectiveness of its analysis and interpretation.

As per author, the following constitutes data-driven decision making:

- Using past data and trending historical data
- Validating assumptions if any after listing all assumptions
- Using champion challenger scenarios to test scenarios
- Using experiments for various tests
- Use baselines for continuous improvement in customer experiences, costs, and revenues
- Taking decisions based on the previous process

As per HBR.org, the more frequent the correlation in a company's data and the lower the risk of being wrong, the more it makes sense to act based on that correlation (Citation: https://hbr.org/2014/05/an-introduction-to-data-driven-decisions-for-managers-who-dont-like-math).

1.18.1 Strategy Frameworks in Business Management: A Refresher for Non-MBAs and MBAs Who Have to Make Data-Driven Decisions

Some frameworks are used for business strategy—to come up with decisions after analyzing the huge reams of qualitative and uncertain data that business generates. This is also part of the substantive expertise circle in Conway's Venn diagram definition of data science at http://drewconway.com/zia/2013/3/26/the-data-science-venn-diagram (Figure 1.8).

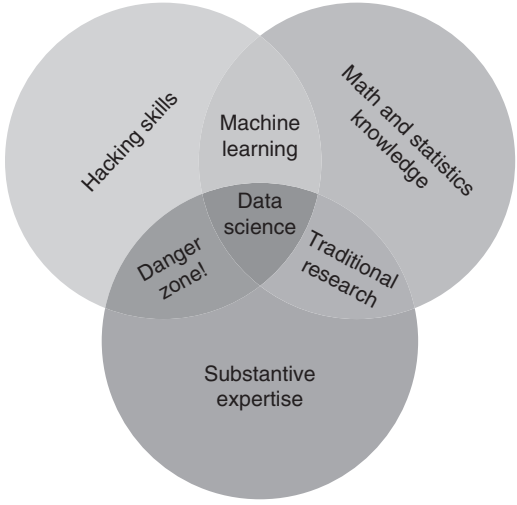

Figure 1.8 Conway's Venn diagram. *Source:* © Drew Conway Data Consulting, LLC.

- **Porter's five forces model**—To analyze industries. Porter's famous model is used to derive five forces that determine the competitive intensity and therefore attractiveness of a market. Attractiveness in this context refers to the overall industry profitability. An "unattractive" industry is one in which the combination of these five forces acts to drive down overall profitability. A very unattractive industry would be one approaching "pure competition" (Figure 1.9).
- **Business canvas**—The business model canvas is used for developing new or documenting existing business models. It describes a firm's value proposition, infrastructure, customers, and finances and thus assists firms by illustrating potential trade-offs in various activities. The business model canvas was initially proposed by Alexander Osterwalder. A bigger graphic can be obtained at https://en.wikipedia.org/wiki/File:Business_Model_Canvas.png (Figure 1.10).
- **BCG matrix**—To analyze product portfolios. BCG Matrix is best used to analyze your own or target organization's product portfolio—applicable for companies with multiple products. This helps corporations allocate resources by analyzing their business units or product lines (Figure 1.11).
- Porter's diamond model—To analyze locations. An economical model developed by Michael Porter in his book *The Competitive Advantage of Nations*, where he published his theory of why particular industries become competitive in particular locations. This helps to analyze countries, states, or locations for both customers and vendors (Figure 1.12).
- McKinsey 7S model—To analyze teams. To check which teams work and which teams are done (within an organization), we can use the 7S model. It is a strategic vision for groups to include businesses, business units, and teams. The 7S are structure, strategy, systems, skills, style, staff, and shared values. The model is most often used as a tool to assess and monitor changes in the internal situation of an organization (Figure 1.13).

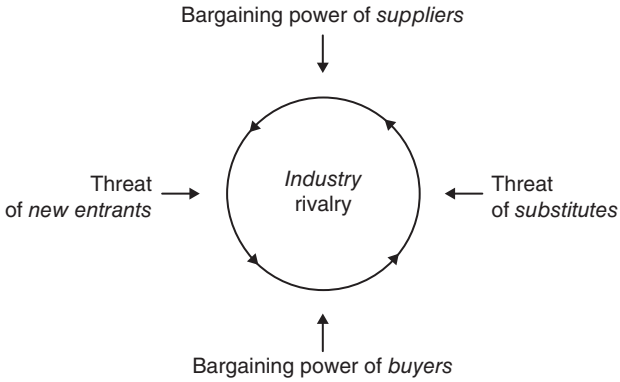

Figure 1.9 Porter five forces for competitive strategy. *Source:* © Wikipedia.

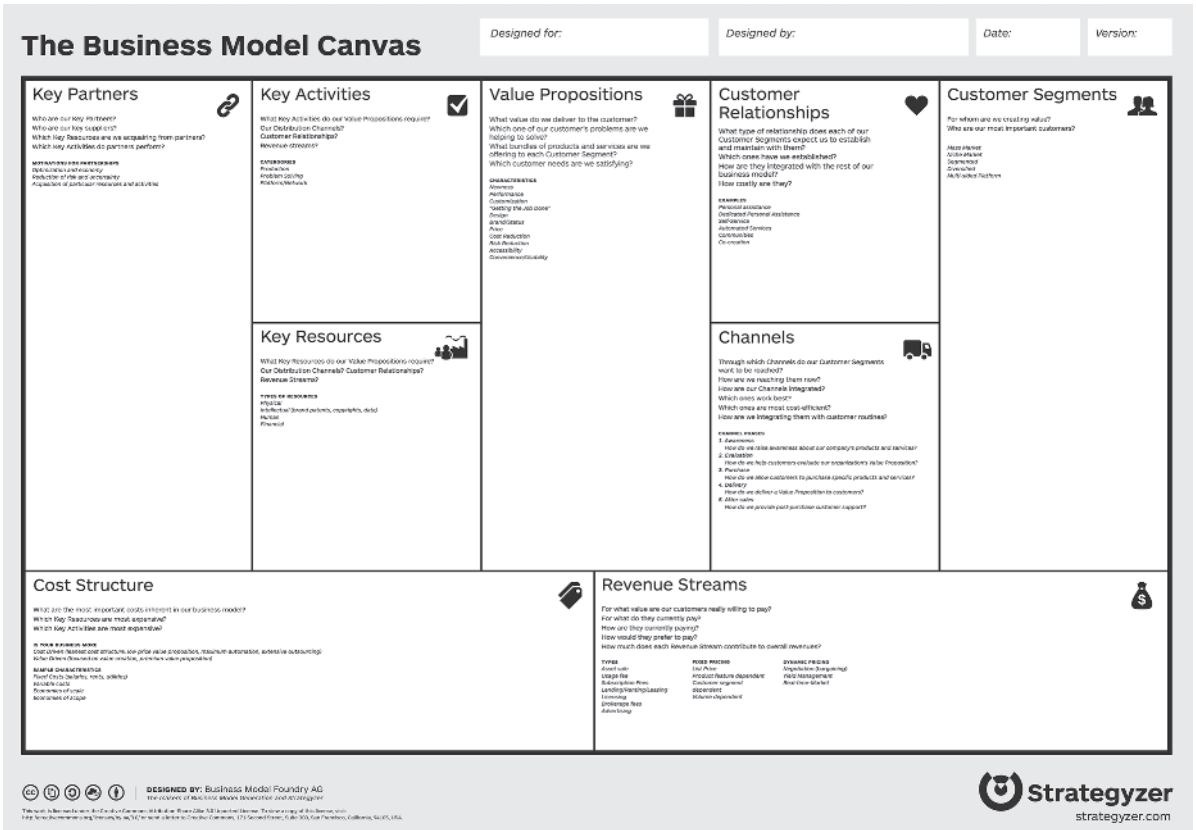

Figure 1.10 Business model canvas. *Source:* © Strategyzer AG Services.

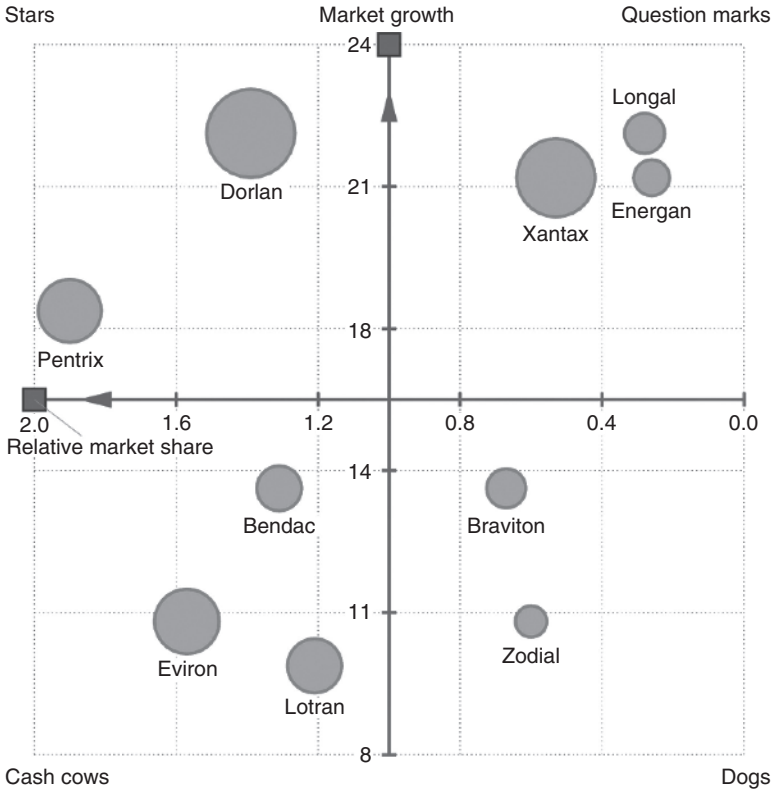

Figure 1.11 BCG matrix. *Source:* http://en.wikipedia.org/wiki/Growth-share_matrix. © Wikipedia.

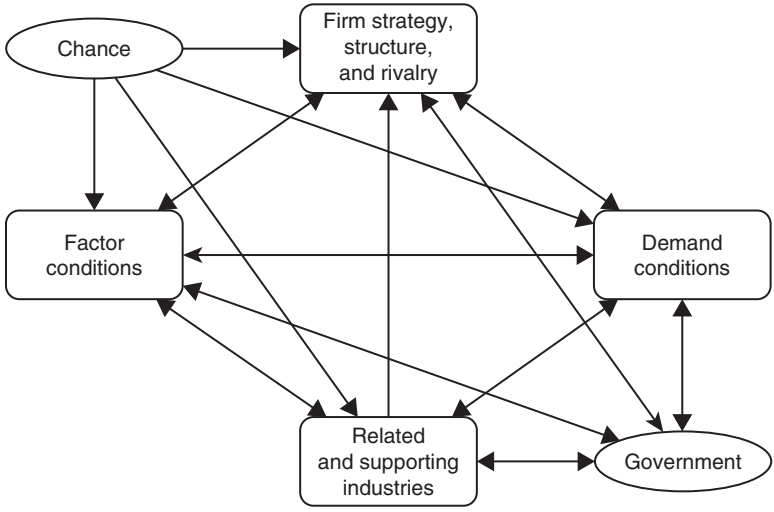

Figure 1.12 Porter's diamond model for competitiveness in locations. *Source:* http://en.wikipedia.org/wiki/Diamond_model. © Wikipedia.

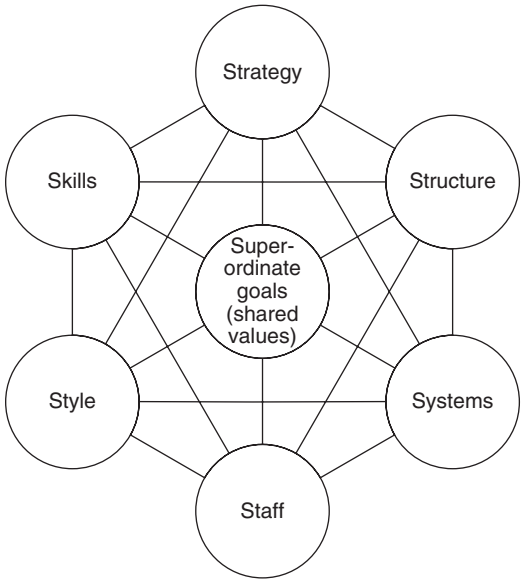

Figure 1.13 Mckinsey 7s model. *Source:* http://en.wikipedia.org/wiki/McKinsey_7S_Framework. © Wikipedia.

- Grenier's theory—To analyze growth of organization. It was developed by Larry E. Greiner and is helpful when examining the problems associated with growth on organizations and the impact of change on employees. It can be argued that growing organizations move through five relatively calm periods of evolution, each of which ends with a period of crisis and revolution. Each evolutionary period is characterized by the dominant management style used to achieve growth, while each revolutionary period is characterized by the dominant management problem that must be solved before growth will continue (Figure 1.14).
- Herzberg's hygiene theory—To analyze soft aspects of individuals.

The following table presents the top seven factors causing dissatisfaction and the top six factors causing satisfaction, listed in the order of higher to lower importance.

Leading to satisfaction

- Achievement
- Recognition
- Work itself
- Responsibility
- Advancement
- Growth

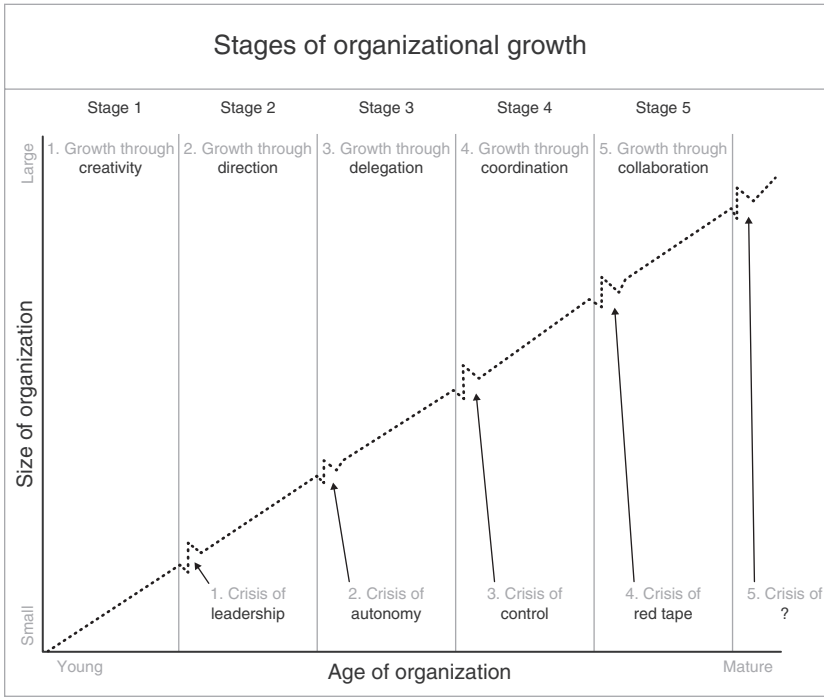

Figure 1.14 Grenier theory. *Source:* Adapted from Greiner (1998). Evolution and Revolution as Organizations Grow. © Harvard Business Publishing.

Leading to dissatisfaction

- Company policy
- Supervision
- Relationship with boss
- Work conditions
- Salary
- Relationship with peers
- Security

This framework helps to explain what motivates people to contribute (or fail to contribute) to teams, products, organizations, and nations. Alternative motivational models are Maslow's hierarchy of needs (shown here) and McGregor Theory X and Theory Y. McGregor terms the two models as "Theory X," which stresses the importance of strict supervision and external rewards and penalties, and "Theory Y," which highlights the motivating role of job satisfaction and allows scope for workers to approach tasks creatively.

- Marketing mix modeling—To analyze marketing mix for determining a product or a brand's offer. It has the four P's: price, product, promotion, and place. This can also be shown by four C's model: consumer, cost, communication, and convenience.

1.18.2 Additional Frameworks for Business Analysis

Pareto Principle

The Pareto principle (also known as the 80/20 rule, the law of the vital few, and the principle of factor sparsity) is a heuristic or a thumb rule that tells analysts to prioritize their analysis. It helps states that, for many events, roughly 80% of the effects come from 20% of the following causes:

- 80% of a company's profits come from 20% of its customers.
- 80% of a company's complaints come from 20% of its customers.
- 80% of a company's profits come from 20% of the time its staff spend.
- 80% of a company's sales come from 20% of its products.
- 80% of a company's sales are made by 20% of its sales staff.

Thus a business analyst should look at the top and bottom 20% of the products, orders, customers, and staff when doing an analysis to determine the cause and effect relationships that can be then modified for positive value creation.

An additional framework is root cause analysis where one can ask five successive why's to determine the root cause of an effect. The process is to ask "why" and identify the causes associated with each sequential step toward the event. "Why" here stands for "What were the factors that directly resulted in the effect?"

LTV Analysis

Lifetime value (LTV) analysis is often a widely used technique within BA to help businesses which customers to retain and which to churn. It also helps with promotions and customer acquisition strategy. LTV is the cumulative revenue a customer will generate for a business over his active lifetime when associated with the products and brands of that business—from acquisition to churn. LTV helps us answer three fundamental questions:

1) Did the business pay enough to acquire customers from each marketing channel (cost of acquisition)?
2) Did the business acquire the best kind of customers (profitability analysis)?
3) How much could the business spend on keeping or retaining them as your customers (prevent churn by offers, calls, email, and social media)?

You can calculate LTV analysis using the methods given at https://blog.kissmetrics.com/how-to-calculate-lifetime-value/ and at http://www.kaushik.net/avinash/analytics-tip-calculate-ltv-customer-lifetime-value/

For LTV analysis in R, you can see this R package at https://cran.r-project.org/web/packages/BTYD/vignettes/BTYD-walkthrough.pdf. The BTYD package contains models to capture noncontractual purchasing behavior of customers—or, more simply, models that tell the story of people buying until they die (become inactive as customers). The main models presented in the package are the Pareto/NBD, BG/NBD, and BG/BB models.

For Python, LTV can be calculated at http://srepho.github.io/CLV/CLV or by the python package lifetimes (https://pypi.python.org/pypi/Lifetimes) or see home page https://github.com/CamDavidsonPilon/lifetimes

RFM Analysis

RFM stands for recency, frequency, and monetization. RFM is thus a method used for analyzing customer value of current customers:

Recency—How recently did the customer purchase?
Frequency—How often do they purchase?
Monetary value—How much do they spend?

This can be quantified in the following way:

Recency = 10 - The number of months that have passed since the customer last purchased
Frequency = Number of purchases in the last 12 months (maximum of 10)
Monetary = Value of the highest order from a given customer (benchmarked against a standard, say, 1000$ or something relevant)

Alternatively, one can create categories for each metric.

For instance, the recency attribute might be broken into three categories: customers with purchases within the last 90 days, between 91 and 365 days, and longer than 365 days. Such categories may be arrived at by applying business rules, or using a data mining technique, to find meaningful breaks (like CHAID). A commonly used shortcut is to use deciles. One is advised to look at distribution of data before choosing breaks.

Practice

You can see RFM analysis in action at https://decisionstats.com/2010/10/03/ibm-spss-19-marketing-analytics-and-rfm/ and some R code for it here at https://github.com/hoxo-m/easyRFM. You should also see http://www.dataapple.net/?p=133. For doing the RFM Analysis in Python, you can see http://www.marketingdistillery.com/2014/11/02/rfm-customer-segmentation-in-r-pandas-and-apache-spark/

Biases in Decision Making

Though not often taught in a standard BA or data science course, the author feels biases in decision making should be useful for a data scientist since the data scientist influences decisions. Even though decision making driven by

data should be objective, it is not as it is driven by humans, not machines, and humans make errors due to multiple reasons.

The author would like to point out these resources.

Logical Fallacies—These would help the data scientist in recognizing the erroneous arguments used by various stakeholders in decision making. A fallacy is an incorrect argument in logic and rhetoric that undermines an argument's logical validity. The following refers to https://yourlogicalfallacyis.com/, which is created by Jesse Richardson, Andy Smith, Som Meaden, and Flip Creative.

Some of the top logical fallacies are as follows:

- **Ad hominem**—You attacked your opponent's character or personal traits in an attempt to undermine their argument.
- **Slippery slope**—You said that if we allow A to happen, then Z will eventually happen too; therefore A should not happen. The problem with this reasoning is that it avoids engaging with the issue at hand and instead shifts attention to extreme hypotheticals.
- **Straw man**—You misrepresented someone's argument to make it easier to attack. By exaggerating, misrepresenting, or just completely fabricating someone's argument, it's much easier to present your own position as being reasonable.

Cognitive Biases—These impact decisions based on the own psychology of the decision maker. A cognitive bias refers to a systematic pattern of deviation from norm or rationality in judgment, whereby inferences about other people and situations may be drawn in an illogical fashion. Individuals create their own "subjective social reality" from their perception of the input.

Some prominent cognitive biases are as follows:

Confirmation bias—In this the individual only selects data or analysis that supports his preconception and tries to discredit, ignore, or trivialize information that is against the preconceived views. This is a very common confirmation bias in practice. One common reason for doing so is agency–owner conflict in which decision makers in an organization take decisions to maximize their own self-interests (like their annual bonuses) rather than team or organizational goals.

Some other common biases are the following:

Self-serving bias	The tendency to claim more responsibility for successes than failures
Belief bias	Evaluating the strength of an argument by your own belief in the truth or falsity of the conclusion
Framing	Using a narrow approach and scope of the problem to avoid difficult to solve issues
Hindsight bias	The inclination to see past events as being predictable

An excellent article on this is also available at Hilbert (2012).

Statistical Bias Versus Variance

This is a more realistic and statistical description of the kind of error a statistical modeler or a data scientist faces when confronted with data. The following is taken from Fortmann Roe (2012)

Error due to bias: The error due to statistical bias is taken as the difference between the expected (or average) prediction of our model and the correct value that we are trying to predict. Of course you only have one model, so talking about expected or average prediction values might seem a little strange. However, imagine you could repeat the whole model building process more than once: each time you gather new data and run a new analysis, creating a new model. Due to randomness in the underlying datasets, the resulting models will have a range of predictions. Bias measures how far off in general these models' predictions are from the correct value.

Error due to variance: The error due to variance is taken as the variability of a model prediction for a given data point. Again, imagine you can repeat the entire model building process multiple times. The variance is how much the predictions for a given point vary between different realizations of the model (Figure 1.15).

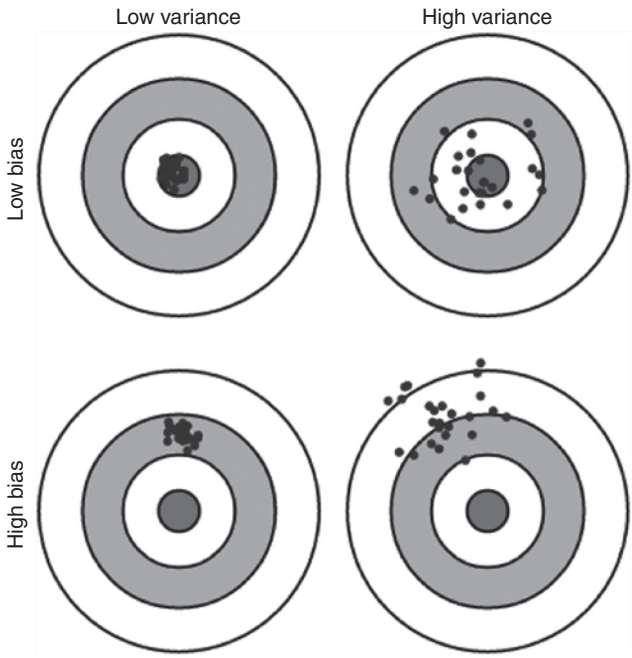

Figure 1.15 Graphical illustration of bias and variance. *Source:* Scott Fortmann-Roe.
© CSS from Substance.io.

Bibliography

Jason R. Briggs. A Playful Introduction to Programming. No Starch Press, 2012, 344 pp, 978-1-59327-407-8.

Scott Fortmann Roe (June 2012). Understanding the Bias-Variance Tradeoff. http://scott.fortmann-roe.com/ and http://scott.fortmann-roe.com/docs/BiasVariance.html (accessed April 29, 2017).

Larry E. Greiner (1998). Evolution and Revolution as Organizations Grow. https://hbr.org/1998/05/evolution-and-revolution-as-organizations-grow. May–June 1998 issue of *Harvard Business Review* (accessed May 22, 2017).

Martin Hilbert (2012). Toward a Synthesis of Cognitive Biases: How Noisy Information Processing Can Bias Human Decision Making. Psychological Bulletin, 138(2), 211–237. Available at martinhilbert.net/HilbertPsychBull.pdf. 10.1037/a0025940, http://dx.doi.org/10.1037/a0025940, http://supp.apa.org/psycarticles/supplemental/a0025940/160972-2010-0470-RRAppendix.pdf (accessed April 29, 2017).

R Core Team. R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, 2013, http://www.R-project.org/ (accessed May 9, 2017).