000 Other

June-09-14 2:22 PM

Other Cheat Sheets
Vincent's Cheat Sheets for Perl, R, Excel (includes Linest, Vlookup), Linux, cron jobs, gzip, ftp, putty, regular expressions, Cygwin, pipe operators, files management, dashboard design etc. coming soon

- Cheat Sheets for Java introcs.cs.princeton.edu/java/11cheatsheet/
- Linux Cheat Sheet www.linuxstall.com/linux-command-line-tips-that-every-linux-user-sh...

From <http://www.datasciencecentral.com/profiles/blogs/17-short-tutorials-all-data-scientists-should-read-and-practice>

Bokeh Cheat Sheet: https://s3.amazonaws.com/assets.datacamp.com/blog_assets/Python_Bokeh_Cheat_Sheet.pdf

Data Science Cheat Sheet: https://www.datacamp.com/community/tutorials/python-data-science-cheat-sheet-basics

Data Wrangling Cheat Sheet: https://www.rstudio.com/wp-content/uploads/2015/02/data-wrangling-cheatsheet.pdf

Data Wrangling: https://en.wikipedia.org/wiki/Data_wrangling

Ggplot Cheat Sheet: https://www.rstudio.com/wp-content/uploads/2015/03/ggplot2-cheatsheet.pdf

Keras Cheat Sheet: https://www.datacamp.com/community/blog/keras-cheat-sheet#gs.DRKeNMs

Keras: https://en.wikipedia.org/wiki/Keras

Machine Learning Cheat Sheet: https://ai.icymi.email/new-machinelearning-cheat-sheet-by-emily-barry-abdsc/

Machine Learning Cheat Sheet: https://docs.microsoft.com/en-in/azure/machine-learning/machine-learning-algorithm-cheat-sheet

ML Cheat Sheet:: http://peekaboo-vision.blogspot.com/2013/01/machine-learning-cheat-sheet-for-scikit.html

Matplotlib Cheat Sheet: https://www.datacamp.com/community/blog/python-matplotlib-cheat-sheet#gs.uEKySpY

Matpotlib: https://en.wikipedia.org/wiki/Matplotlib

Neural Networks Cheat Sheet: http://www.asimovinstitute.org/neural-network-zoo/

Neural Networks: https://www.quora.com/Where-can-find-a-cheat-sheet-for-neural-network

Numpy Cheat Sheet: https://www.datacamp.com/community/blog/python-numpy-cheat-sheet#gs.AK5ZBgE

NumPy: https://en.wikipedia.org/wiki/NumPy

Pandas Cheat Sheet: https://www.datacamp.com/community/blog/python-pandas-cheat-sheet#gs.oundfxM

Pandas: https://en.wikipedia.org/wiki/Pandas_(software)

Pandas Cheat Sheet: https://www.datacamp.com/community/blog/pandas-cheat-sheet-python#gs.HPFoRIc

Pyspark Cheat Sheet: https://www.datacamp.com/community/blog/pyspark-cheat-sheet-python#gs.L=J1zxQ

Scikit Cheat Sheet: https://www.datacamp.com/community/blog/scikit-learn-cheat-sheet

Scikit-learn: https://en.wikipedia.org/wiki/Scikit-learn

Scikit-learn Cheat Sheet: http://peekaboo-vision.blogspot.com/2013/01/machine-learning-cheat-sheet-for-scikit.html

Scipy Cheat Sheet: https://www.datacamp.com/community/blog/python-scipy-cheat-sheet#gs.JDSg3OI

SciPy: https://en.wikipedia.org/wiki/SciPy

TesorFlow Cheat Sheet: https://www.altoros.com/tensorflow-cheat-sheet.html

Tensor Flow: https://en.wikipedia.org/wiki/TensorFlow

From <https://becominghuman.ai/cheat-sheets-for-ai-neural-networks-machine-learning-deep-learning-big-data-678c51b4b463>

Chatbot

July-17-17 3:55 PM

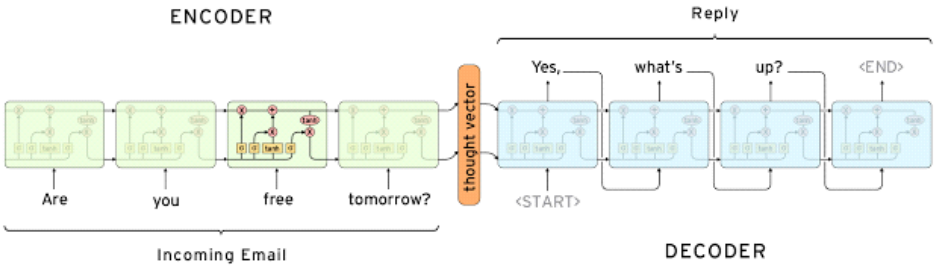
https://becominghuman.ai/cheat-sheets-for-ai-neural-networks-machine-learning-deep-learning-big-data-678c51b4b463

Ultimate Guide to Leveraging NLP & Machine Learning for your Chatbot

RETRIEVAL-BASED VS. GENERATIVE MODELS

Retrieval-based models (easier) use a repository of predefined responses and some kind of heuristic to pick an appropriate response based on the input and context. The heuristic could be as simple as a rule-based expression match, or as complex as an ensemble of Machine Learning classifiers. These systems don't generate any new text, they just pick a response from a fixed set.

Generative models (harder) don't rely on pre-defined responses. They generate new responses from scratch. Generative models are typically based on Machine Translation techniques, but instead of translating from one language to another, we "translate" from an input to an output (response).



Machine Learning

Both approaches have some obvious pros and cons. Due to the repository of handcrafted responses, retrieval-based methods don't make grammatical mistakes. However, they may be unable to handle unseen cases for which no appropriate predefined response exists. For the same reasons, these models can't refer back to contextual entity information like names mentioned earlier in the conversation. Generative models are "smarter". They can refer back to entities in the input and give the impression that you're talking to a human. However, these models are hard to train, are quite likely to make grammatical mistakes (especially on longer sentences), and typically require huge amounts of training data.

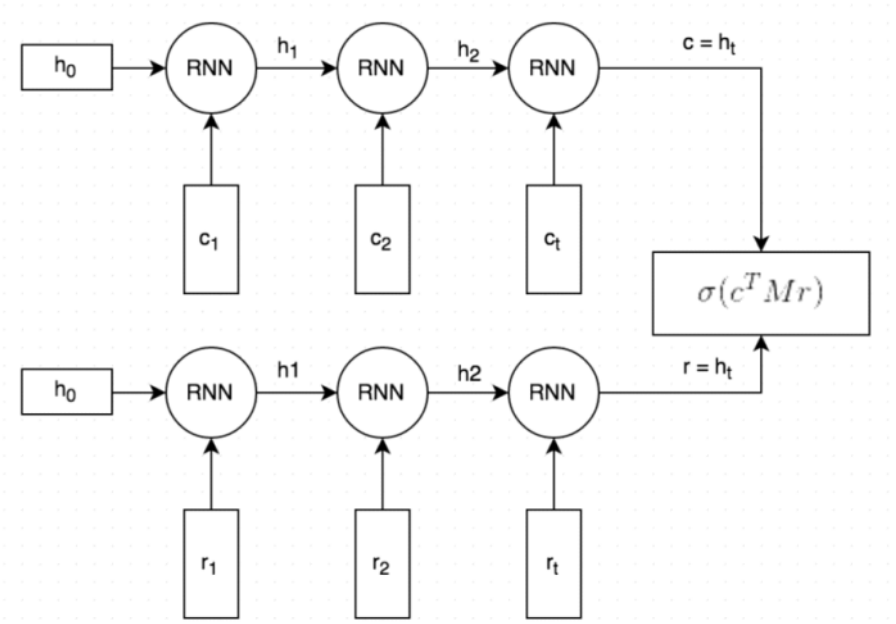
Deep Learning techniques can be used for both retrieval-based or generative models, but research seems to be moving into the generative direction. Deep Learning architectures likeSequence to Sequence are uniquely suited for generating text and researchers are hoping to make rapid progress in this area. However, we're still at the early stages of building generative models that work reasonably well. Production systems are more likely to be retrieval-based for now.

In this post we'll work with the Ubuntu Dialog Corpus (paper, github). The Ubuntu Dialog Corpus (UDC) is one of the largest public dialog datasets available. It's based on chat logs from the Ubuntu channels on a public IRC network. The paper goes into detail on how exactly the corpus was created, so I won't repeat that here. However, it's important to understand what kind of data we're working with, so let's do some exploration first.

The are various ways to evaluate how well our model does. A commonly used metric is *recall@k*. Recall@k means that we let the model pick the k best responses out of the 10 possible responses (1 true and 9 distractors). If the correct one is among the picked ones we mark that test example as correct. So, a larger k means that the task becomes easier. If we set k=10 we get a recall of 100% because we only have 10 responses to pick from. If we set k=1 the model has only one chance to pick the right response.

At this point you may be wondering how the 9 distractors were chosen. In this data set the 9 distractors were picked at random. However, in the real world you may have millions of possible responses and you don't know which one is correct. You can't possibly evaluate a million potential responses to pick the one with the highest score— that'd be too expensive. Google'sSmart Reply uses clustering techniques to come up with a set of possible responses to choose from first. Or, if you only have a few hundred potential responses in total you could just evaluate all of them.

The Dual Encoder LSTM we'll build looks like this (paper):



Duel Encoder

It roughly works as follows:

1. Both the context and the response text are split by words, and each word is embedded into a vector. The word embeddings are initialized with Stanford's GloVe vectors and are fine-tuned during training (Side note: This is optional and not shown in the picture. I found that initializing the word embeddings with GloVe did not make a big difference in terms of model performance).

2. Both the embedded context and response are fed into the same Recurrent Neural Network word-by-word. The RNN generates a vector representation that, loosely speaking, captures the "meaning" of the context and response (c and r in the picture). We can choose how large these vectors should be, but let's say we pick 256 dimensions.

3. We multiply c with a matrix M to "predict" a response r'. If c is a 256-dimensional vector, then M is a 256×256 dimensional matrix, and the result is another 256-dimensional vector, which we can interpret as a generated response. The matrix M is learned during training.

4. We measure the similarity of the predicted response r' and the actual response r by taking the dot product of these two vectors. A large dot product means the vectors are similar and that the response should receive a high score. We then apply a sigmoid function to convert that score into a probability. Note that steps 3 and 4 are combined in the figure.

From <https://chatbotslife.com/ultimate-guide-to-leveraging-nlp-machine-learning-for-you-chatbot-531ff2dd870c>

Code Snippets and Github Includedchatbotslife.com

Artificial Intelligence Periodic Table

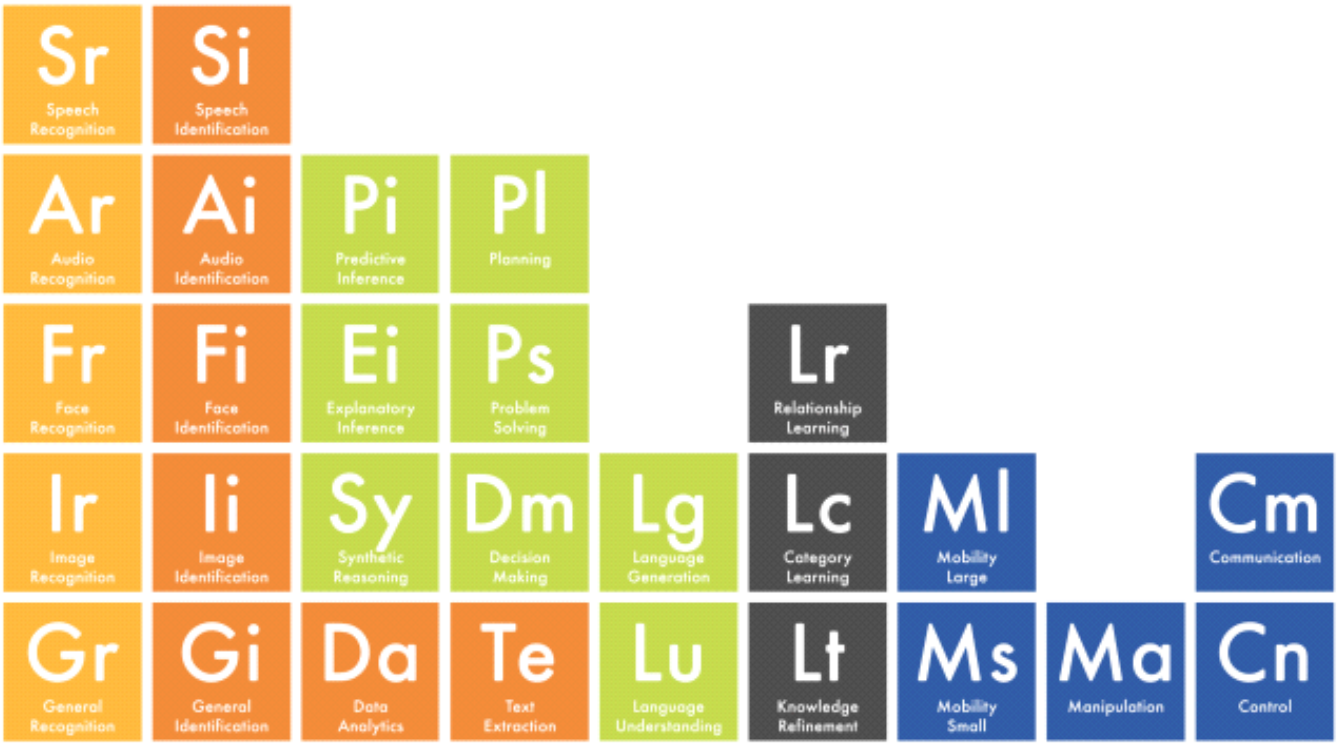
April-13-17 9:38 AM

This article was written by Kris Hammond.

This is an invitation to collaborate. In particular, it is an invitation to collaborate in framing how we look at and develop machine intelligence. Even more specifically, it is an invitation to collaborate in the construction of a Periodic Table of AI.

Let's be honest. Thinking about Artificial Intelligence has proven to be difficult for us. We argue constantly about what is and is not AI. We certainly cannot agree on how to test for it. We have difficultly deciding what technologies should be included within it. And we struggle with how to evaluate it.

Even so, we are looking at a future in which intelligent technologies are becoming commonplace. Take personal assistants, albeit simple, they are now on our phones and in our homes. Intelligent analytical systems are beginning to evaluate our credit worthiness, investment risks and massive transactional flows to alert us about fraud and money manipulation. And even ignoring the possibilities associated with having our health tracked and maintained by agents with access to every piece of online medical information, we are surrounded by systems that track our transactions, interests and connections in order to give us advice about who we might want to be friends with, what we might want to buy and even who we should consider dating.



From <http://www.datasciencecentral.com/profiles/blogs/the-periodic-table-of-ai>

Cheat Sheet - 201507 24 Data Science, R, Python, Excel, and Machine Learning Cheat Sheets

July-30-15 3:12 PM

Data Science

1. Data Science Cheat Sheet - Basic
2. Data Science Cheat Sheet - Advanced

Hadoop

1. Hadoop for Dummies cheatsheet
2. Getting Started Apache Hadoop Reference Card
3. Hadoop Command Line cheatsheet
4. Working with HDFS from the command line - Hadoop Cheat sheet

R

1. R cheat sheet (Google Drive)
2. R functions for Regression Analysis
3. R functions for Time series Analysis
4. R Cheat Sheet
5. Data Visualization with R
6. Data Analysis the data.table way
7. Data Visualisation with ggplot2 cheatsheet by R studio

Python

1. Python 2.7 Quick Reference Sheet
2. Python Cheat Sheet by DaveChild
3. Python Basics Reference sheet
4. NumPy / SciPy / Pandas Cheat Sheet

Machine Learning

1. Choosing the right estimator Machine Learning cheatsheet
2. Patterns for Predictive learning cheatsheet
3. Machine learning algorithm cheat sheet for Microsoft Azure
4. Machine Learning cheatsheet Github 1
5. Machine Learning cheatsheet Github 2
6. Machine Learning which algorithm performs best?
7. Cheat sheet 10 machine learning algorithms R commands

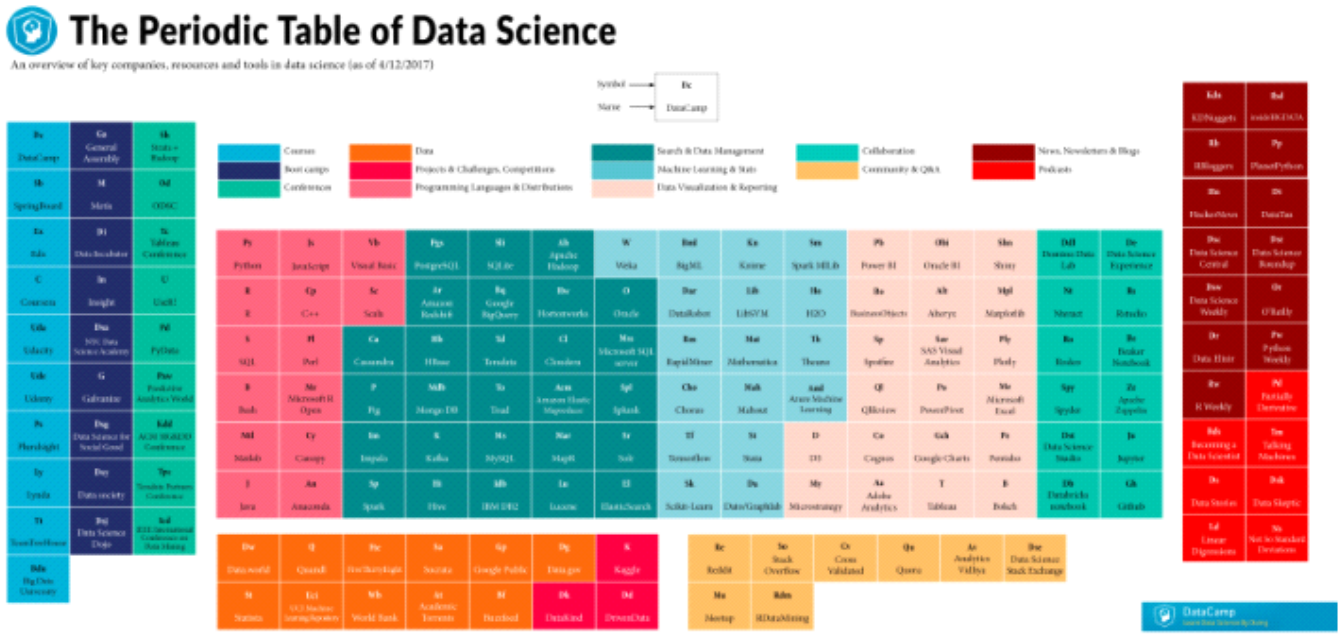
Source for picture: click here

Pasted from <http://www.datasciencecentral.com/profiles/blogs/20-data-science-r-python-excel-and-machine-learning-cheat-sheets>

DataCamp Data Science

April-13-17 10:07 AM

This periodic table can serve as a guide to navigate the key players in the data science space. The resources in the table were chosen by looking at surveys taken from data science users, such as the 2016 Data Science Salary Survey by O'Reilly, the 2017 Magic Quadrant for Data Science Platforms by Gartner, and the KD Nuggets 2016 Software Poll results, among other sources. The categories in the table are not all mutually exclusive.
Check out the full periodic table of data science below:



Navigating The Periodic Table of Data Science

You'll see that on the table's left-hand section lists companies that have to do with education: here, you'll find courses, boot camps and conferences. On the right-hand side, on the other hand, you'll find resources that will keep you up to date with the latest news, hottest blogs and relevant material in the data science community. In the middle, you'll find tools that you can use to get started with data science: you'll find programming languages, projects and challenges, data visualization tools, etc.
The table puts the data science resources, tools and companies in the following 13 categories:
Courses: for those who are looking to learn data science, there are a bunch of sites (companies) out there that offer data science courses. You'll find various options here that will probably suit your learning style: DataCamp for learning by doing, MOOCs by Coursera and Edx, and much more!
Boot camps: this section includes resources for those who are looking for more mentored options to learn data science. You'll see that boot camps like The Data Incubator or Galvanize have been included.
Conferences: learning is not an activity that you do when you go on courses or boot camps. Conferences are something that learners often forget, but they also contribute to learning data science: it's important that you attend them as a data science aspirant, as you'll get in touch with the latest advancements and the best industry experts. Some of the ones that are listed in the table are UseR!, Tableau Conference and PyData.
Data: practice makes perfect, and this is also the case for data science. You'll need to look and find data sets in order to start practicing what you learned in the courses on real-life data or to make your data science portfolio. Data is the basic building block of data science and finding that data can be probably one of the hardest things. Some of the options that you could consider when you're looking for cool data sets are data.world, Quandl and Statista.
Projects & Challenges, Competitions: after practicing, you might also consider taking on bigger projects: data science portfolios, competitions, challenges, ... You'll find all of these in this category of the Periodic Table of Data Science! One of the most popular options is probably Kaggle, but also DrivenData or DataKind are worth checking out!
Programming Languages & Distributions: data scientists generally use not only one, but many programming languages; Some programming languages like Python have recently gained a lot of traction in the community and also Python distributions, like Anaconda, seem to find their way to data science aspirants.
Search & Data Management: this enormous category contains all tools that you can use to search and manage your data in some way. You'll see, on the one hand, a search library like Lucene, but also a relational database management system like Oracle.
Machine Learning & Stats: this category not only offers you libraries to get started with machine learning and stats with programming languages such as Python, but also entire platforms, such as Alteryx or DataRobot.
Data Visualization & Reporting: after you have analyzed and modeled your data, you might be looking to visualize the results and report on what you have been investigating. You can make use of open-source options like Shiny or Matplotlib to do this, or all back on commercial options such as Qlikview or Tableau.
Collaboration: collaboration is a trending topic in the data science community. As you grow, you'll also find the need to work in teams (even if it's just with one other person!) and in those cases, you'll want to make use of notebooks like Jupyter. But even as you're just working on your own, working with an IDE can come in handy if you're just starting out. In such cases, consider Rodeo or Spyder.
Community & Q&A: asking questions and falling back on the community is one of the things that you'll probably do a lot when you're learning data science. If you're ever unsure of where you can find the answer to your data science question, you can be sure to find it in sites such as StackOverflow, Quora, Reddit, etc.
News, Newsletters & Blogs: you'll find that the community is evolving and growing rapidly: following the news and the latest trends is a necessity. General newsletters like Data Science Weekly or Data Elixir, or language-specific newsletters like Python Weekly or R Weekly can give you your weekly dose of data science right in your mailbox. But also blogging sites like RBloggers or KDNuggets are worth following!
Podcasts: last, but definitely not least, are the podcasts. These are great in many ways, as you'll get introduced to expert interviews, like in Becoming A Data Scientist or to specific data science topics, like in Data Stories or Talking Machines!
Are you thinking of another resource that should be added to this periodic table? Leave a comment below and tell us about it!

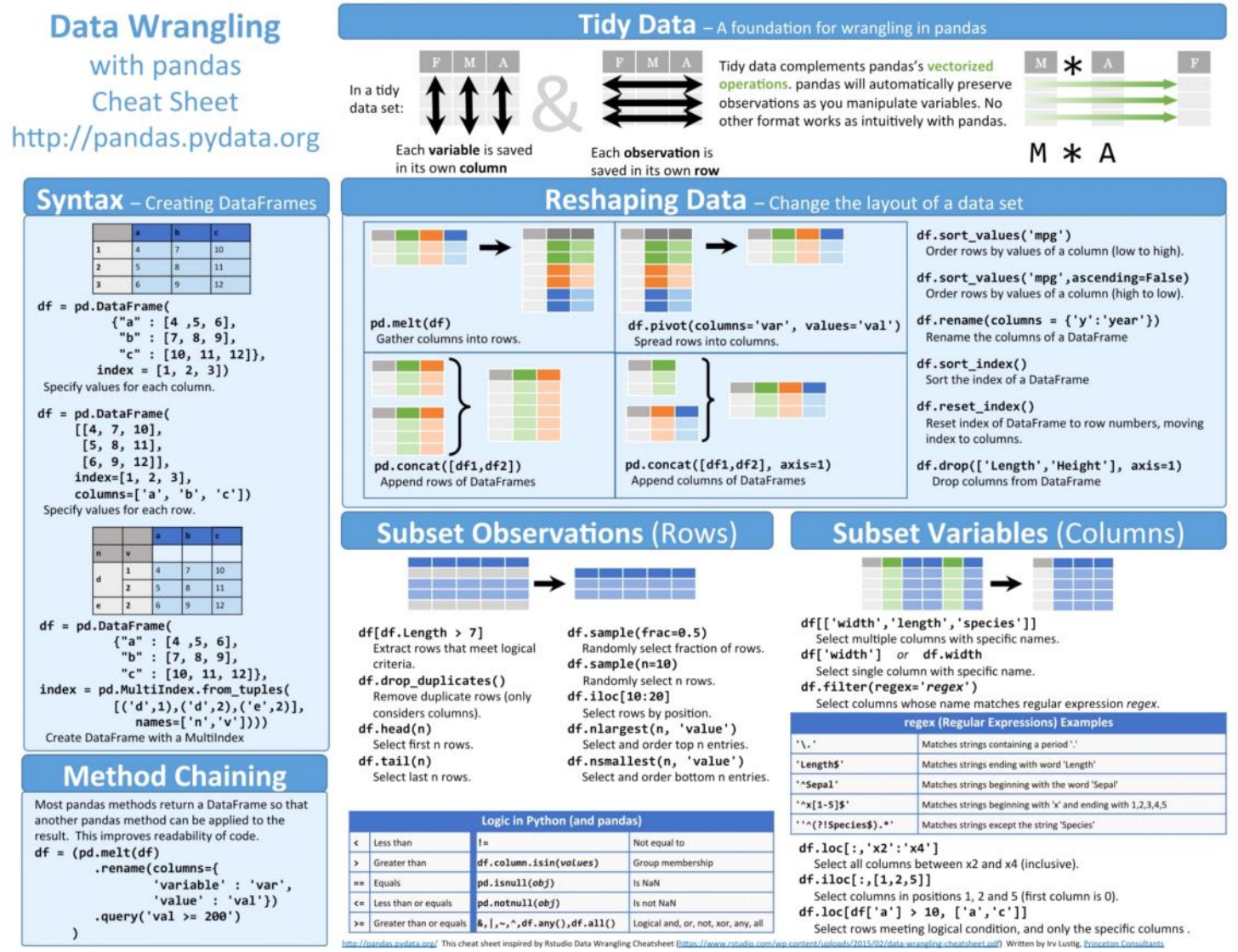
From <https://www.datacamp.com/community/blog/data-science-periodic-table#gs.tIng3H4>

Data Wrangling

July-17-17 4:28 PM

Data Wrangling

The term "data wrangler" is starting to infiltrate pop culture. In the 2017 movie Kong: Skull Island, one of the characters, played by actor Marc Evan Jackson is introduced as "Steve Woodward, our data wrangler".



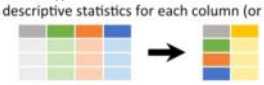
Summarize Data

`df['w'].value_counts()`
Count number of rows with each unique value of variable

`len(df)`
# of rows in DataFrame.

`df['w'].nunique()`
# of distinct values in a column.

`df.describe()`
Basic descriptive statistics for each column (or GroupBy)



pandas provides a large set of **summary functions** that operate on different kinds of pandas objects (DataFrame columns, Series, GroupBy, Expanding and Rolling (see below)) and produce single values for each of the groups. When applied to a DataFrame, the result is returned as a pandas Series for each column. Examples:

`sum()`
Sum values of each object.

`count()`
Count non-NA/null values of each object.

`median()`
Median value of each object.

`quantile([0.25,0.75])`
Quantiles of each object.

`apply(function)`
Apply function to each object.

`min()`
Minimum value in each object.

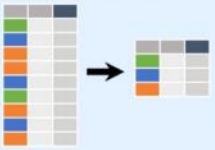
`max()`
Maximum value in each object.

`mean()`
Mean value of each object.

`var()`
Variance of each object.

`std()`
Standard deviation of each object.

Group Data



`df.groupby(by="col")`
Return a GroupBy object, grouped by values in column named "col".

`df.groupby(level="ind")`
Return a GroupBy object, grouped by values in index level named "ind".

All of the summary functions listed above can be applied to a group. Additional GroupBy functions:

`size()`
Size of each group.

`agg(function)`
Aggregate group using function.

The examples below can also be applied to groups. In this case, the function is applied on a per-group basis, and the returned vectors are of the length of the original DataFrame.

`shift(1)`
Copy with values shifted by 1.

`rank(method='dense')`
Ranks with no gaps.

`rank(method='min')`
Ranks. Ties get min rank.

`rank(pct=True)`
Ranks rescaled to interval [0, 1].

`rank(method='first')`
Ranks. Ties go to first value.

`shift(-1)`
Copy with values lagged by 1.

`cumsum()`
Cumulative sum.

`cummax()`
Cumulative max.

`cummin()`
Cumulative min.

`cumprod()`
Cumulative product.

Windows

`df.expanding()`
Return an Expanding object allowing summary functions to be applied cumulatively.

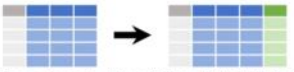
`df.rolling(n)`
Return a Rolling object allowing summary functions to be applied to windows of length n.

Handling Missing Data

`df.dropna()`
Drop rows with any column having NA/null data.

`df.fillna(value)`
Replace all NA/null data with value.

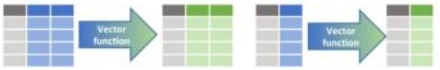
Make New Columns



`df.assign(Area=lambda df: df.Length*df.Height)`
Compute and append one or more new columns.

`df['Volume'] = df.Length*df.Height*df.Depth`
Add single column.

`pd.qcut(df.col, n, labels=False)`
Bin column into n buckets.



pandas provides a large set of **vector functions** that operate on all columns of a DataFrame or a single selected column (a pandas Series). These functions produce vectors of values for each of the columns, or a single Series for the individual Series. Examples:

`max(axis=1)`
Element-wise max.

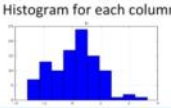
`clip(lower=-10,upper=10)`
Trim values at input thresholds

`min(axis=1)`
Element-wise min.

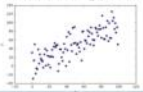
`abs()`
Absolute value.

Plotting

`df.plot.hist()`
Histogram for each column



`df.plot.scatter(x='w',y='h')`
Scatter chart using pairs of points



Combine Data Sets

adf

x1	x2
A	1
B	2
C	3

+

bdf

x1	x3
A	T
B	F
D	T

=

Standard Joins

x1	x2	x3
A	1	T
B	2	F
C	3	NaN

`pd.merge(adf, bdf, how='left', on='x1')`
Join matching rows from bdf to adf.

x1	x2	x3
A	1.0	T
B	2.0	F
D	NaN	T

`pd.merge(adf, bdf, how='right', on='x1')`
Join matching rows from adf to bdf.

x1	x2	x3
A	1	T
B	2	F

`pd.merge(adf, bdf, how='inner', on='x1')`
Join data. Retain only rows in both sets.

x1	x2	x3
A	1	T
B	2	F
C	3	NaN
D	NaN	T

`pd.merge(adf, bdf, how='outer', on='x1')`
Join data. Retain all values, all rows.

Filtering Joins

x1	x2
A	1
B	2

`adf[adf.x1.isin(bdf.x1)]`
All rows in adf that have a match in bdf.

x1	x2
C	3

`adf[~adf.x1.isin(bdf.x1)]`
All rows in adf that do not have a match in bdf.

ydf

x1	x2
A	1
B	2
C	3

+

zdf

x1	x2
B	2
C	3
D	4

=

Set-like Operations

x1	x2
B	2
C	3

`pd.merge(ydf, zdf)`
Rows that appear in both ydf and zdf (Intersection).

x1	x2
A	1
B	2
C	3
D	4

`pd.merge(ydf, zdf, how='outer')`
Rows that appear in either or both ydf and zdf (Union).

x1	x2
A	1

```
pd.merge(ydf, zdf, how='outer',
         indicator=True)
.query('_merge == "left_only"')
.drop(['_merge'],axis=1)
```
Rows that appear in ydf but not zdf (Setdiff).

http://pandas.pydata.org/ This cheat sheet inspired by Rstudio Data Wrangling Cheatsheet (https://www.rstudio.com/wp-content/uploads/2015/02/data-wrangling-cheatsheet.pdf) Written by Irv Lustig, Princeton Consultants

Data Wrangling with dplyr and tidyr Cheat Sheet

RStudio

Tidy Data - A foundation for wrangling in R

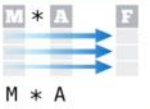
In a tidy data set:

Each **variable** is saved in its own **column**

&

Each **observation** is saved in its own **row**

Tidy data complements R's **vectorized operations**. R will automatically preserve observations as you manipulate variables. No other format works as intuitively with R.



Syntax - Helpful conventions for wrangling

dplyr::tbl_df(iris)

Converts data to tbl class. tbl's are easier to examine than data frames. R displays only the data that fits onscreen:

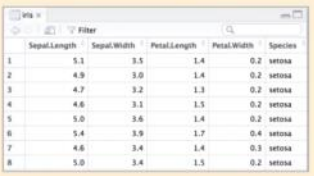
```
Source: local data frame [150 x 5]

   Sepal.Length Sepal.Width Petal.Length
1           5.1         3.5          1.4
2           4.9         3.0          1.4
3           4.7         3.2          1.3
4           4.6         3.1          1.5
5           5.0         3.6          1.4
..          ...         ...          ...
Variables not shown: Petal.Width (dbl),
  Species (fctr)
```

dplyr::glimpse(iris)

Information dense summary of tbl data.

utils::View(iris)

View data set in spreadsheet-like display (note capital V).



	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa
7	4.6	3.4	1.4	0.3	setosa
8	5.0	3.4	1.5	0.2	setosa

dplyr::%>%

Passes object on left hand side as first argument (or . argument) of function on righthand side.

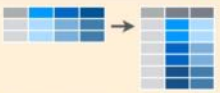
`x %>% f(y)` *is the same as* `f(x, y)`

`y %>% f(x, ., z)` *is the same as* `f(x, y, z )`

"Piping" with %>% makes code more readable, e.g.

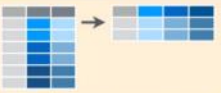
```
iris %>%
  group_by(Species) %>%
  summarise(avg = mean(Sepal.Width)) %>%
  arrange(avg)
```

Reshaping Data - Change the layout of a data set



tidyr::gather(cases, "year", "n", 2:4)

Gather columns into rows.



tidyr::spread(pollution, size, amount)

Spread rows into columns.

tidyr::separate(storms, date, c("y", "m", "d"))

Separate one column into several.

tidyr::unite(data, col, ..., sep)

Unite several columns into one.

dplyr::data_frame(a = 1:3, b = 4:6)

Combine vectors into data frame (optimized).

dplyr::arrange(mtcars, mpg)

Order rows by values of a column (low to high).

dplyr::arrange(mtcars, desc(mpg))

Order rows by values of a column (high to low).

dplyr::rename(tb, y = year)

Rename the columns of a data frame.

Subset Observations (Rows)

dplyr::filter(iris, Sepal.Length > 7)

Extract rows that meet logical criteria.

dplyr::distinct(iris)

Remove duplicate rows.

dplyr::sample_frac(iris, 0.5, replace = TRUE)

Randomly select fraction of rows.

dplyr::sample_n(iris, 10, replace = TRUE)

Randomly select n rows.

dplyr::slice(iris, 10:15)

Select rows by position.

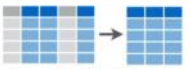
dplyr::top_n(storms, 2, date)

Select and order top n entries (by group if grouped data).

Logic in R - ?Comparison, ?base::Logic

<	Less than	!=	Not equal to
>	Greater than	%in%	Group membership
==	Equal to	is.na	Is NA
<=	Less than or equal to	!is.na	Is not NA
>=	Greater than or equal to	&, |, !, xor, any, all	Boolean operators

Subset Variables (Columns)



dplyr::select(iris, Sepal.Width, Petal.Length, Species)

Select columns by name or helper function.

Helper functions for select - ?select

select(iris, contains("."))
Select columns whose name contains a character string.

select(iris, ends_with("Length"))
Select columns whose name ends with a character string.

select(iris, everything())
Select every column.

select(iris, matches(".t."))
Select columns whose name matches a regular expression.

select(iris, num_range("x", 1:5))
Select columns named x1, x2, x3, x4, x5.

select(iris, one_of(c("Species", "Genus")))
Select columns whose names are in a group of names.

select(iris, starts_with("Sepal"))
Select columns whose name starts with a character string.

select(iris, Sepal.Length:Petal.Width)
Select all columns between Sepal.Length and Petal.Width (inclusive).

select(iris, -Species)
Select all columns except Species.

devtools::install_github("rstudio/EDAWR") for data sets

Learn more with browseVignettes(package = c("dplyr", "tidyr")) • dplyr 0.4.0• tidyr 0.2.0 • Updated: 1/15

Summarise Data

dplyr::**summarise(iris, avg = mean(Sepal.Length))**
Summarise data into single row of values.
dplyr::**summarise_each(iris, funs(mean))**
Apply summary function to each column.
dplyr::**count(iris, Species, wt = Sepal.Length)**
Count number of rows with each unique value of variable (with or without weights).

Summarise uses **summary functions**, functions that take a vector of values and return a single value, such as:

dplyr::**first**
First value of a vector.
dplyr::**last**
Last value of a vector.
dplyr::**nth**
Nth value of a vector.
dplyr::**n**
# of values in a vector.
dplyr::**n_distinct**
# of distinct values in a vector.
IQR
IQR of a vector.
min
Minimum value in a vector.
max
Maximum value in a vector.
mean
Mean value of a vector.
median
Median value of a vector.
var
Variance of a vector.
sd
Standard deviation of a vector.

Group Data

dplyr::**group_by(iris, Species)**
Group data into rows with the same value of Species.
dplyr::**ungroup(iris)**
Remove grouping information from data frame.

iris %>% group_by(Species) %>% summarise(…)
Compute separate summary row for each group.

Make New Variables

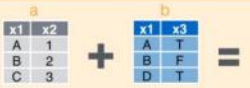
dplyr::**mutate(iris, sepal = Sepal.Length + Sepal. Width)**
Compute and append one or more new columns.
dplyr::**mutate_each(iris, funs(min_rank))**
Apply window function to each column.
dplyr::**transmute(iris, sepal = Sepal.Length + Sepal. Width)**
Compute one or more new columns. Drop original columns.

Mutate uses **window functions**, functions that take a vector of values and return another vector of values, such as:

dplyr::**lead**
Copy with values shifted by 1.
dplyr::**lag**
Copy with values lagged by 1.
dplyr::**dense_rank**
Ranks with no gaps.
dplyr::**min_rank**
Ranks. Ties get min rank.
dplyr::**percent_rank**
Ranks rescaled to [0, 1].
dplyr::**row_number**
Ranks. Ties got to first value.
dplyr::**ntile**
Bin vector into n buckets.
dplyr::**between**
Are values between a and b?
dplyr::**cume_dist**
Cumulative distribution.
dplyr::**cumall**
Cumulative `all`
dplyr::**cumany**
Cumulative `any`
dplyr::**cummean**
Cumulative `mean`
cumsum
Cumulative `sum`
cummax
Cumulative `max`
cummin
Cumulative `min`
cumprod
Cumulative `prod`
pmax
Element-wise `max`
pmin
Element-wise `min`

iris %>% group_by(Species) %>% mutate(…)
Compute new variables by group.

Combine Data Sets



Mutating Joins

dplyr::**left_join(a, b, by = "x1")**
Join matching rows from b to a.

dplyr::**right_join(a, b, by = "x1")**
Join matching rows from a to b.

dplyr::**inner_join(a, b, by = "x1")**
Join data. Retain only rows in both sets.

dplyr::**full_join(a, b, by = "x1")**
Join data. Retain all values, all rows.

Filtering Joins

dplyr::**semi_join(a, b, by = "x1")**
All rows in a that have a match in b.

dplyr::**anti_join(a, b, by = "x1")**
All rows in a that do not have a match in b.

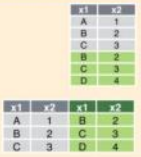
Set Operations

dplyr::**intersect(y, z)**
Rows that appear in both y and z.

dplyr::**union(y, z)**
Rows that appear in either or both y and z.

dplyr::**setdiff(y, z)**
Rows that appear in y but not z.

Binding



dplyr::**bind_rows(y, z)**
Append z to y as new rows.
dplyr::**bind_cols(y, z)**
Append z to y as new columns.
Caution: matches rows by position.

devtools::install_github("rstudio/EDAWR") for data sets

Learn more with browseVignettes(package = c("dplyr", "tidyr")) • dplyr 0.4.0• tidyr 0.2.0 • Updated: 1/15

Machine Learning

April-13-17 9:33 AM

Machine Learning Cheat Sheet: https://ai.icymi.email/new-machinelearning-cheat-sheet-by-emily-barry-abdsc/

This blog about machine learning was written by Emily Barry. Emily is a Data Scientist in San Francisco, California. She really loves emoji. Another thing she loves is data science. The more she learns about machine learning algorithms, the more challenging it is to keep these subjects organized in her brain to recall at a later time. So, she decided to marry these two loves in as productive a fashion as possible.

This is *by no means* a comprehensive guide to machine learning, but rather a study in the basics for herself and the likely small overlap of people who like machine learning and love emoji as much as she do.

MACHINE LEARNING IN EMOJI

SUPERVISED UNSUPERVISED REINFORCEMENT

SUPERVISED — human builds model based on input / output

UNSUPERVISED — human input, machine output
human utilizes if satisfactory

REINFORCEMENT — human input, machine output
human reward/punish, cycle continues

BASIC REGRESSION

LINEAR — linear_model.LinearRegression()

Lots of numerical data

LOGISTIC — linear_model.LogisticRegression()

Target variable is categorical

CLASSIFICATION

NEURAL NET — neural_network.MLPClassifier()

Complex relationships. Prone to overfitting
Basically magic.

K-NN — neighbors.KNeighborsClassifier()

Group membership based on proximity

DECISION TREE — tree.DecisionTreeClassifier()

If/then/else. Non-contiguous data
Can also be regression

RANDOM FOREST — ensemble.RandomForestClassifier()

Find best split randomly
Can also be regression

SVM — svm.SVC() svm.LinearSVC()

Maximum margin classifier. Fundamental
Data Science algorithm

NAIVE BAYES GaussianNB() MultinomialNB() BernoulliNB()

Updating knowledge step by step with new info

CLUSTER ANALYSIS

K-MEANS — cluster.KMeans()

Similar datum into groups
based on centroids

ANOMALY DETECTION — covariance.EllipticalEnvelope()

Finding outliers
through grouping

FEATURE REDUCTION

T-DISTRIB STOCHASTIC NEIB EMBEDDING — manifold.TSNE()

Visualize high dimensional data. Convert
similarity to joint probabilities

PRINCIPLE COMPONENT ANALYSIS — decomposition.PCA()

Distill feature space into components that
describe greatest variance

CANONICAL CORRELATION ANALYSIS — decomposition.CCA()

Making sense of cross-correlation
matrices

LINEAR DISCRIMINANT ANALYSIS — lda.LDA()

Linear combination of features that
separates classes

OTHER IMPORTANT CONCEPTS

BIAS VARIANCE TRADEOFF

UNDERFITTING / OVERFITTING

INERTIA

ACCURACY FUNCTION $(TP + TN) / (P + N)$

PRECISION FUNCTION $TP / (TP + FP)$

SPECIFICITY FUNCTION $TN / (FP + TN)$

SENSITIVITY FUNCTION $TP / (TP + FN)$

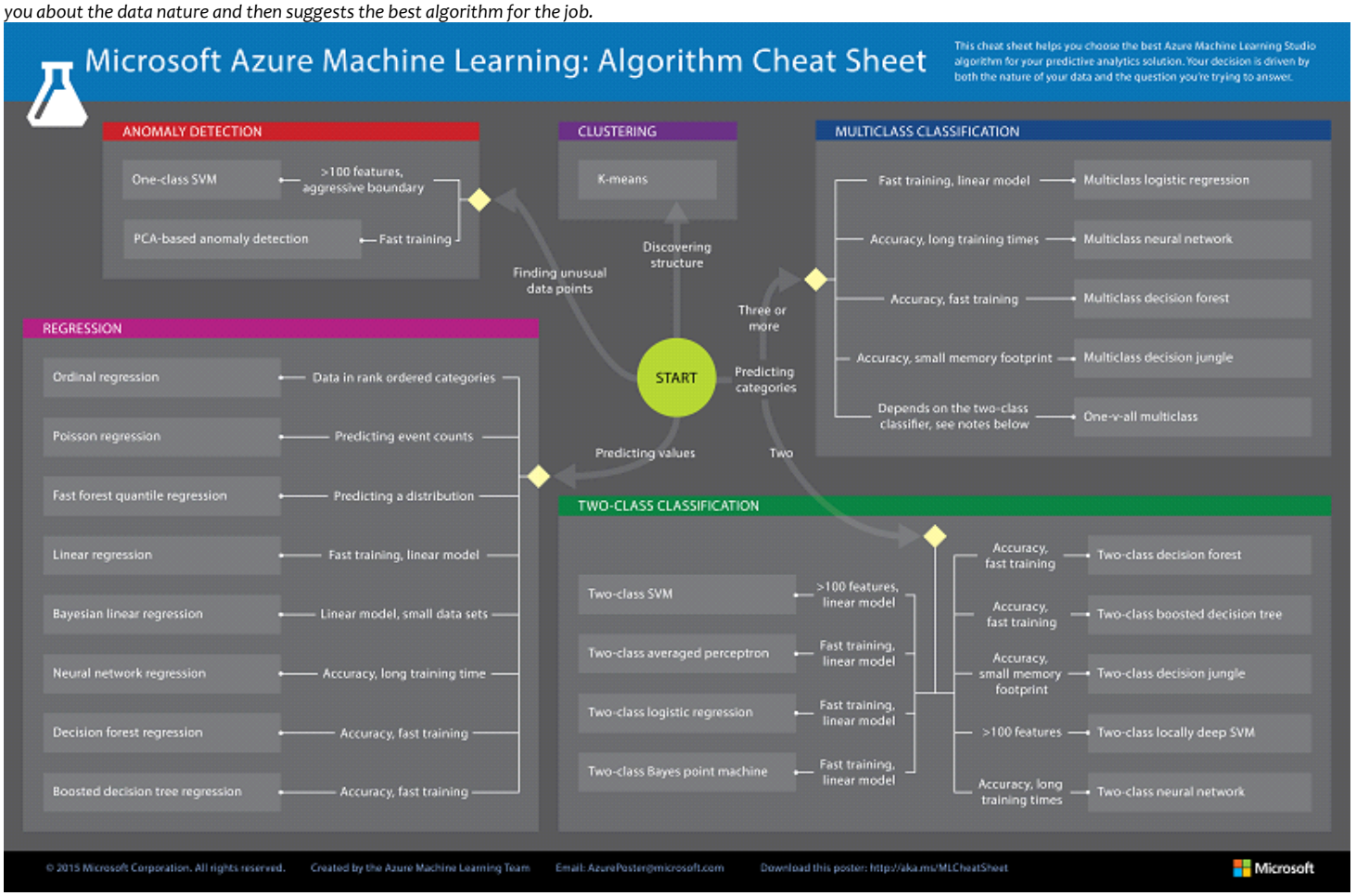
@emilyinamillion made this

From <http://www.datasciencecentral.com/profiles/blogs/the-making-of-a-cheatsheet-emoji-edition>

MACHINE LEARNING Azure : ALGORITHM CHEAT SHEET

July-17-17 4:00 PM

This machine learning cheat sheet from Microsoft Azure will help you choose the appropriate machine learning algorithms for your predictive analytics solution. First, the cheat sheet will asks you about the data nature and then suggests the best algorithm for the job.



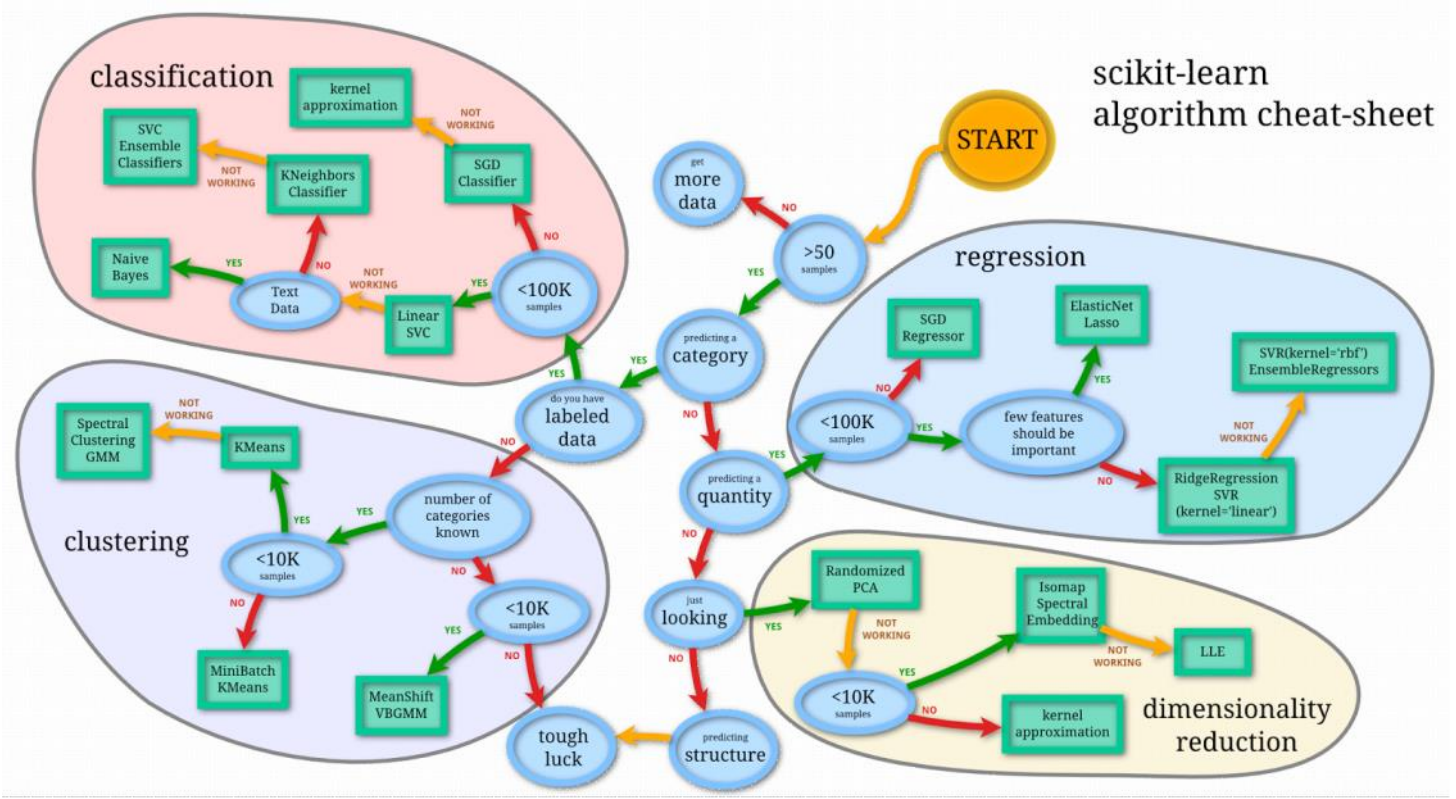
Machine Learning Cheat Sheet: https://docs.microsoft.com/en-in/azure/machine-learning/machine-learning-algorithm-cheat-sheet

From <https://becominghuman.ai/cheat-sheets-for-ai-neural-networks-machine-learning-deep-learning-big-data-678c51b4b463>

Machine Learning: Scikit-learn algorithm

July-17-17 3:58 PM

This machine learning cheat sheet will help you find the right estimator for the job which is the most difficult part. The flowchart will help you check the documentation and rough guide of each estimator that will help you to know more about the problems and how to solve it.

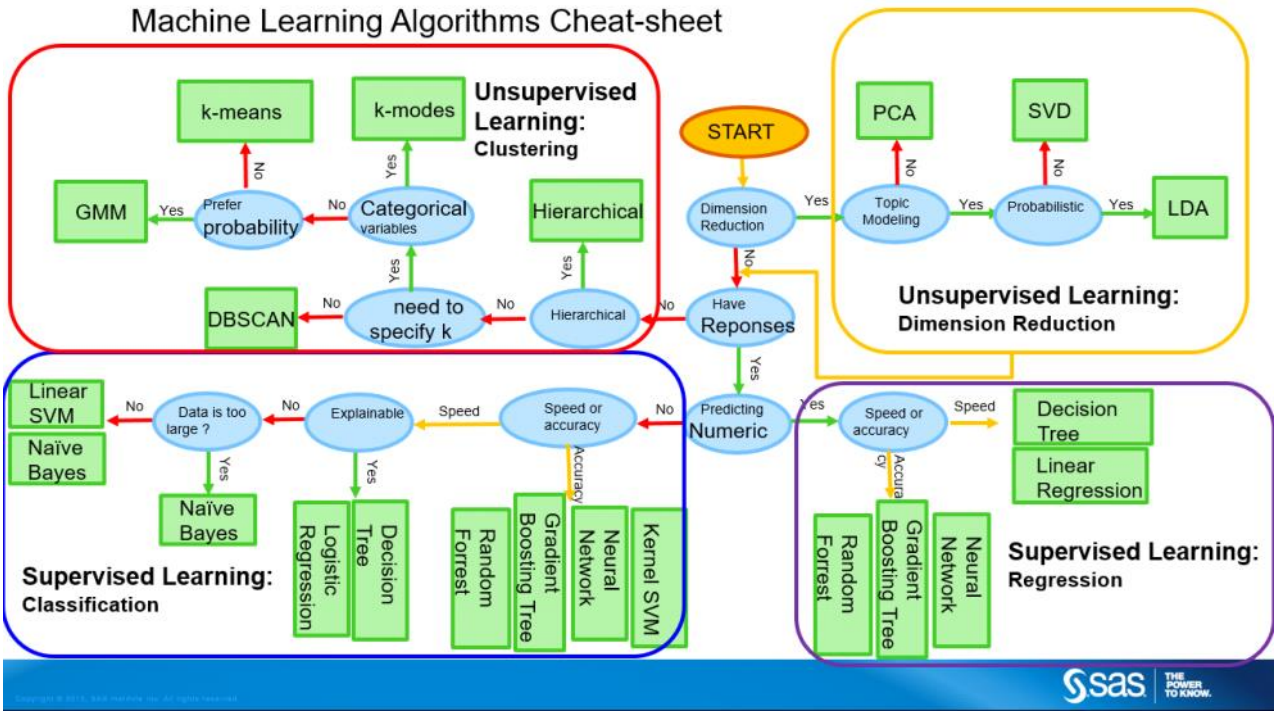


Scikit-learn Cheat Sheet: http://peekaboo-vision.blogspot.com/2013/01/machine-learning-cheat-sheet-for-scikit.html

From <https://becominghuman.ai/cheat-sheets-for-ai-neural-networks-machine-learning-deep-learning-big-data-678c51b4b463>

Machine Learning SAS

April-24-17 4:49 PM



Which machine learning algorithm should I use?

6

By Hui Li on Subconscious Musings April 12, 2017Advanced Analytics | Machine Learning

This resource is designed primarily for beginning data scientists or analysts who are interested in identifying and applying machine learning algorithms to address the problems of their interest. A typical question asked by a beginner, when facing a wide variety of machine learning algorithms, is "which algorithm should I use?" The answer to the question varies depending on many factors, including:

- The size, quality, and nature of data.
- The available computational time.
- The urgency of the task.
- What you want to do with the data.

Even an experienced data scientist cannot tell which algorithm will perform the best before trying different algorithms. We are not advocating a one and done approach, but we do hope to provide some guidance on which algorithms to try first depending on some clear factors.

The **machine learning algorithm cheat sheet** helps you to choose from a variety of machine learning algorithms to find the appropriate algorithm for your specific problems. This article walks you through the process of how to use the sheet.

Since the cheat sheet is designed for beginner data scientists and analysts, we will make some simplified assumptions when talking about the algorithms.

The algorithms recommended here result from compiled feedback and tips from several data scientists and machine learning experts and developers. There are several issues on which we have not reached an agreement and for these issues we try to highlight the commonality and reconcile the difference.

Additional algorithms will be added in later as our library grows to encompass a more complete set of available methods.

How to use the cheat sheet

Read the path and algorithm labels on the chart as "If *<path label>* then use *<algorithm>*." For example:

- If you want to perform dimension reduction then use principal component analysis.
- If you need a numeric prediction quickly, use decision trees or logistic regression.
- If you need a hierarchical result, use hierarchical clustering.

Sometimes more than one branch will apply, and other times none of them will be a perfect match. It's important to remember these paths are intended to be rule-of-thumb recommendations, so some of the recommendations are not exact. Several data scientists I talked with said that the only sure way to find the very best algorithm is to try all of them.

Types of machine learning algorithms

This section provides an overview of the most popular types of machine learning. If you're familiar with these categories and want to move on to discussing specific algorithms, you can skip this section and go to "When to use specific algorithms" below.

Supervised learning

Supervised learning algorithms make predictions based on a set of examples. For example, historical sales can be used to estimate the future prices. With supervised learning, you have an input variable that consists of labeled training data and a desired output variable. You use an algorithm to analyze the training data to learn the function that maps the input to the output. This inferred function maps new, unknown examples by generalizing from the training data to anticipate results in unseen situations.

- **Classification:** When the data are being used to predict a categorical variable, supervised learning is also called classification. This is the case when assigning a label or indicator, either dog or cat to an image. When there are only two labels, this is called binary classification. When there are more than two categories, the problems are called multi-class classification.
- **Regression:** When predicting continuous values, the problems become a regression problem.
- **Forecasting:** This is the process of making predictions about the future based on the past and present data. It is most commonly used to analyze trends. A common example might be estimation of the next year sales based on the sales of the current year and previous years.

Semi-supervised learning

The challenge with supervised learning is that labeling data can be expensive and time consuming. If labels are limited, you can use unlabeled examples to enhance supervised learning. Because the machine is not fully supervised in this case, we say the machine is semi-supervised. With semi-supervised learning, you use unlabeled examples with a small amount of labeled data to improve the learning accuracy.

Unsupervised learning

When performing unsupervised learning, the machine is presented with totally unlabeled data. It is asked to discover the intrinsic patterns that underlies the data, such as a clustering structure, a low-dimensional manifold, or a sparse tree and graph.

- **Clustering:** Grouping a set of data examples so that examples in one group (or one cluster) are more similar (according to some criteria) than those in other groups. This is often used to segment the whole dataset into several groups. Analysis can be performed in each group to help users to find intrinsic patterns.
- **Dimension deduction:** Reducing the number of variables under consideration. In many applications, the raw data have very high dimensional features and some features are redundant or irrelevant to the task. Reducing the dimensionality helps to find the true, latent relationship.

Reinforcement learning

Reinforcement learning analyzes and optimizes the behavior of an agent based on the feedback from the environment. Machines try different scenarios to discover which actions yield the greatest reward, rather than being told which actions to take. Trial-and-error and delayed reward distinguish reinforcement learning from other techniques.

Considerations when choosing an algorithm

When choosing an algorithm, always take these aspects into account: accuracy, training time and ease of use. Many users put the accuracy first, while beginners tend to focus on algorithms they know best.

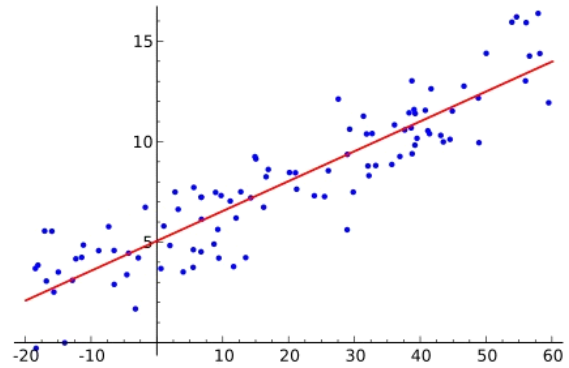
When presented with a dataset, the first thing to consider is how to obtain results, no matter what those results might look like. Beginners tend to choose algorithms that are easy to implement and can obtain results quickly. This works fine, as long as it is just the first step in the process. Once you obtain some results and become familiar with the data, you may spend more time using more sophisticated algorithms to strengthen your understanding of the data, hence further improving the results.

Even in this stage, the best algorithms might not be the methods that have achieved the highest reported accuracy, as an algorithm usually requires careful tuning and extensive training to obtain its best achievable performance.

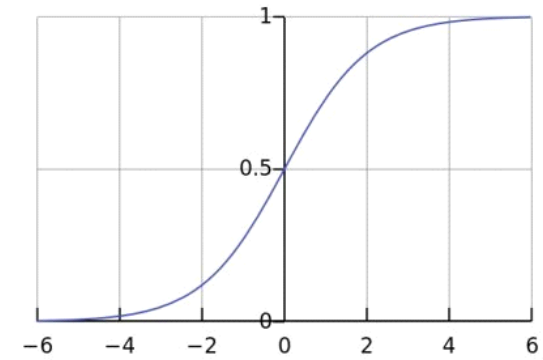
When to use specific algorithms

Looking more closely at individual algorithms can help you understand what they provide and how they are used. These descriptions provide more details and give additional tips for when to use specific algorithms, in alignment with the cheat sheet.

Linear regression and Logistic regression

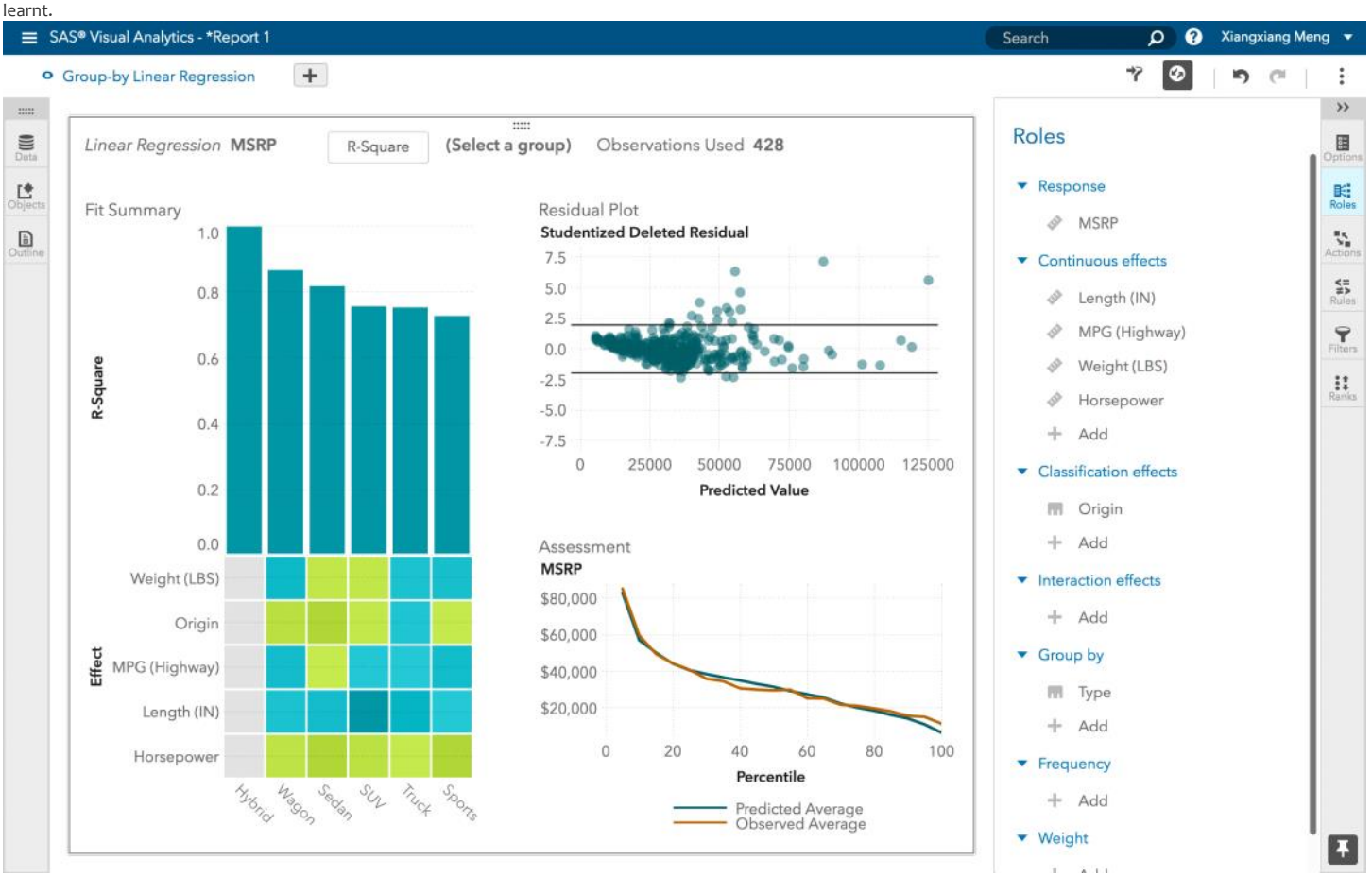


Linear regression



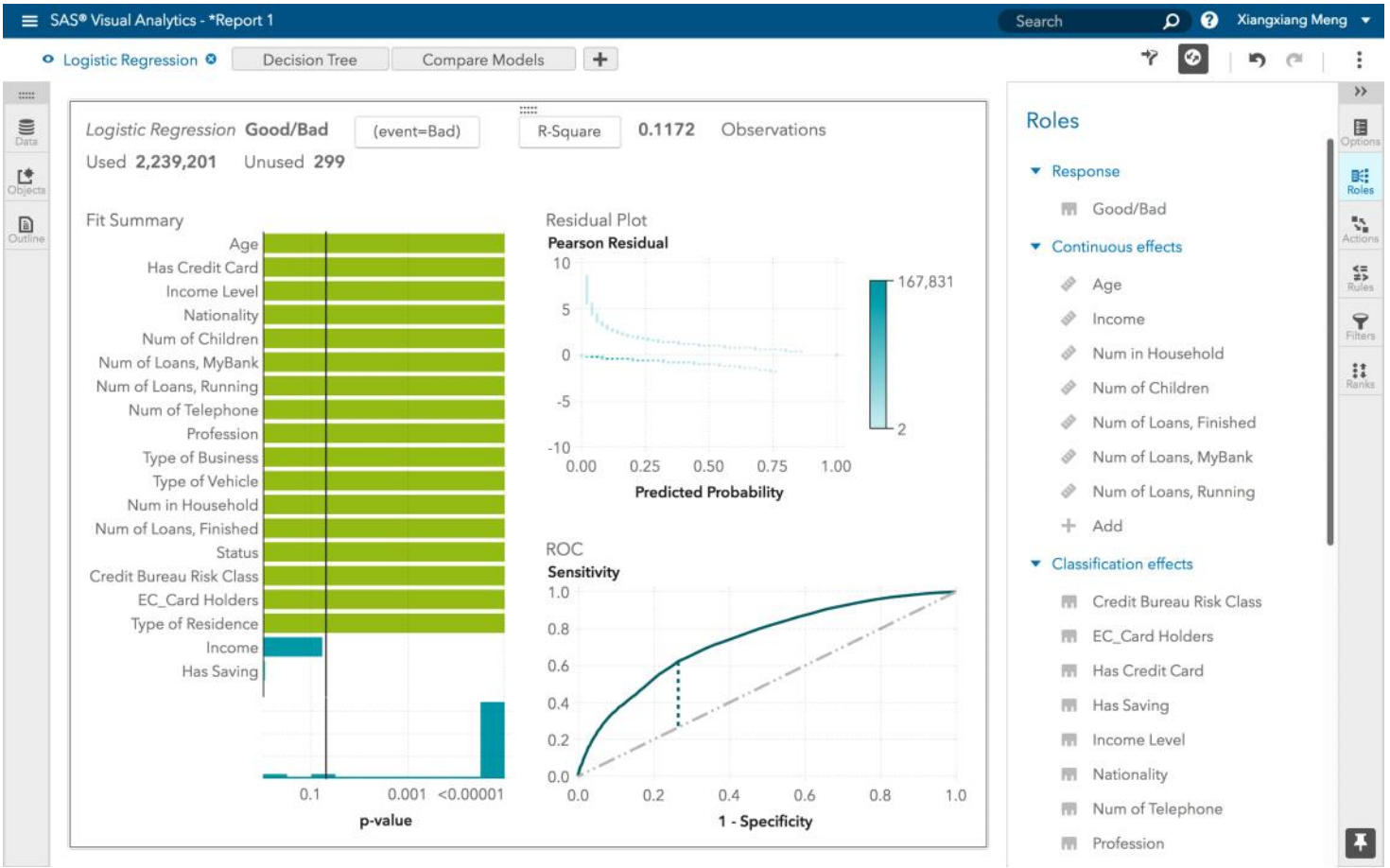
Logistic regression

Linear regression is an approach for modeling the relationship between a continuous dependent variable y and one or more predictors X. The relationship between yand X can be linearly modeled as y=βTX+ϵ Given the training examples {xi,yi}Ni=1, the parameter vector β can be learnt.



Linear regression example in SAS Visual Analytics

If the dependent variable is not continuous but categorical, linear regression can be transformed to logistic regression using a logit link function. Logistic regression is a simple, fast yet powerful classification algorithm. Here we discuss the binary case where the dependent variable y only takes binary values {yi∈(−1,1)}Ni=1 (it which can be easily extended to multi-class classification problems).



Logistic regression example in SAS Visual Analytics

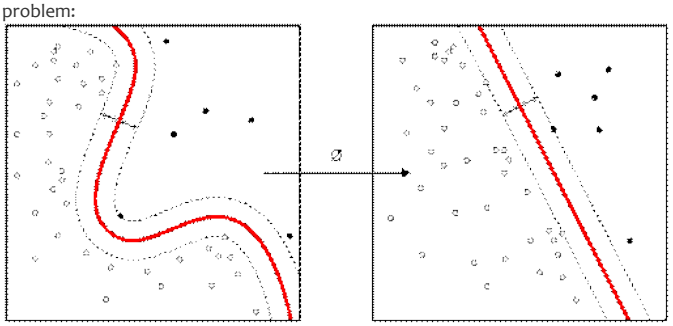
In logistic regression we use a different hypothesis class to try to predict the probability that a given example belongs to the "1" class versus the probability that it belongs to the "-1" class. Specifically, we will try to learn a function of the form:$p(y_i=1|x_i)=\sigma(\beta^T x_i)$ and $p(y_i=-1|x_i)=1-\sigma(\beta^T x_i)$. Here $\sigma(x)=\frac{1}{1+\exp(-x)}$ is a sigmoid function. Given the training examples$\{x_i,y_i\}_{i=1}^N$, the parameter vector β can be learnt by maximizing the log-likelihood of β given the data set.

Linear SVM and kernel SVM

Kernel tricks are used to map a non-linearly separable functions into a higher dimension linearly separable function. A support vector machine (SVM) training algorithm finds the classifier represented by the normal vector w and bias b of the hyperplane. This hyperplane (boundary) separates different classes by as wide a margin as possible. The problem can be converted into a constrained optimization problem:

$\text{minimize} \|w\| \text{subject to} y_i(w^T x_i - b) \geq 1, i=1,\ldots,n.$

A support vector machine (SVM) training algorithm finds the classifier represented by the normal vector and bias of the hyperplane. This hyperplane (boundary) separates different classes by as wide a margin as possible. The problem can be converted into a constrained optimization problem:

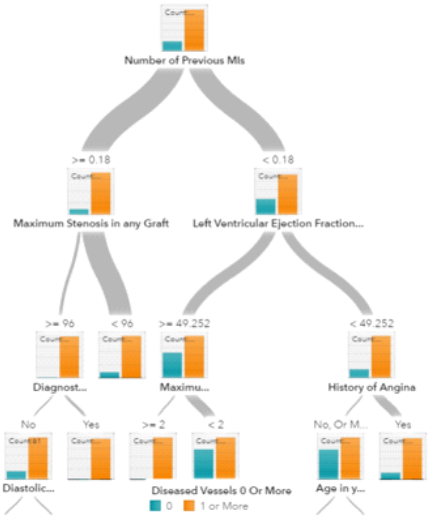


Kernel tricks are used to map a non-linearly separable functions into a higher dimension linearly separable function.

When the classes are not linearly separable, a kernel trick can be used to map a non-linearly separable space into a higher dimension linearly separable space.

When most dependent variables are numeric, logistic regression and SVM should be the first try for classification. These models are easy to implement, their parameters easy to tune, and the performances are also pretty good. So these models are appropriate for beginners.

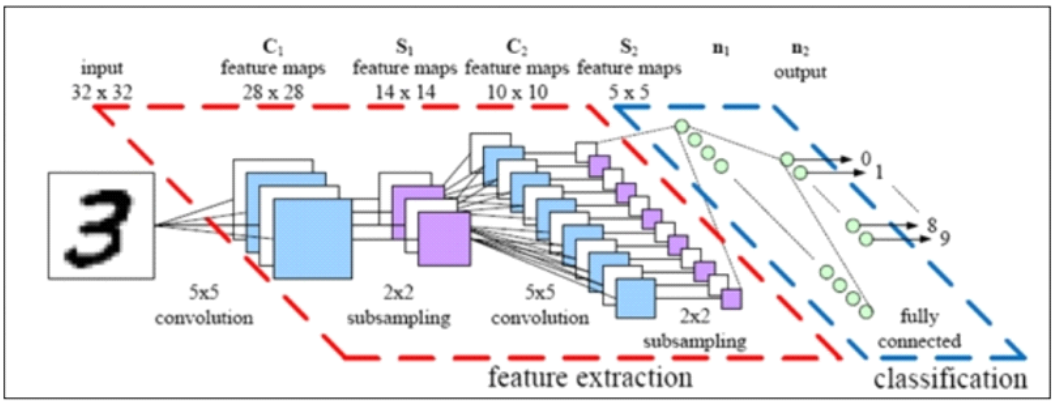
Trees and ensemble trees



A decision tree for prediction model

Decision trees, random forest and gradient boosting are all algorithms based on decision trees. There are many variants of decision trees, but they all do the same thing – subdivide the feature space into regions with mostly the same label. Decision trees are easy to understand and implement. However, they tend to over fit data when we exhaust the branches and go very deep with the trees. Random Forrest and gradient boosting are two popular ways to use tree algorithms to achieve good accuracy as well as overcoming the over-fitting problem.

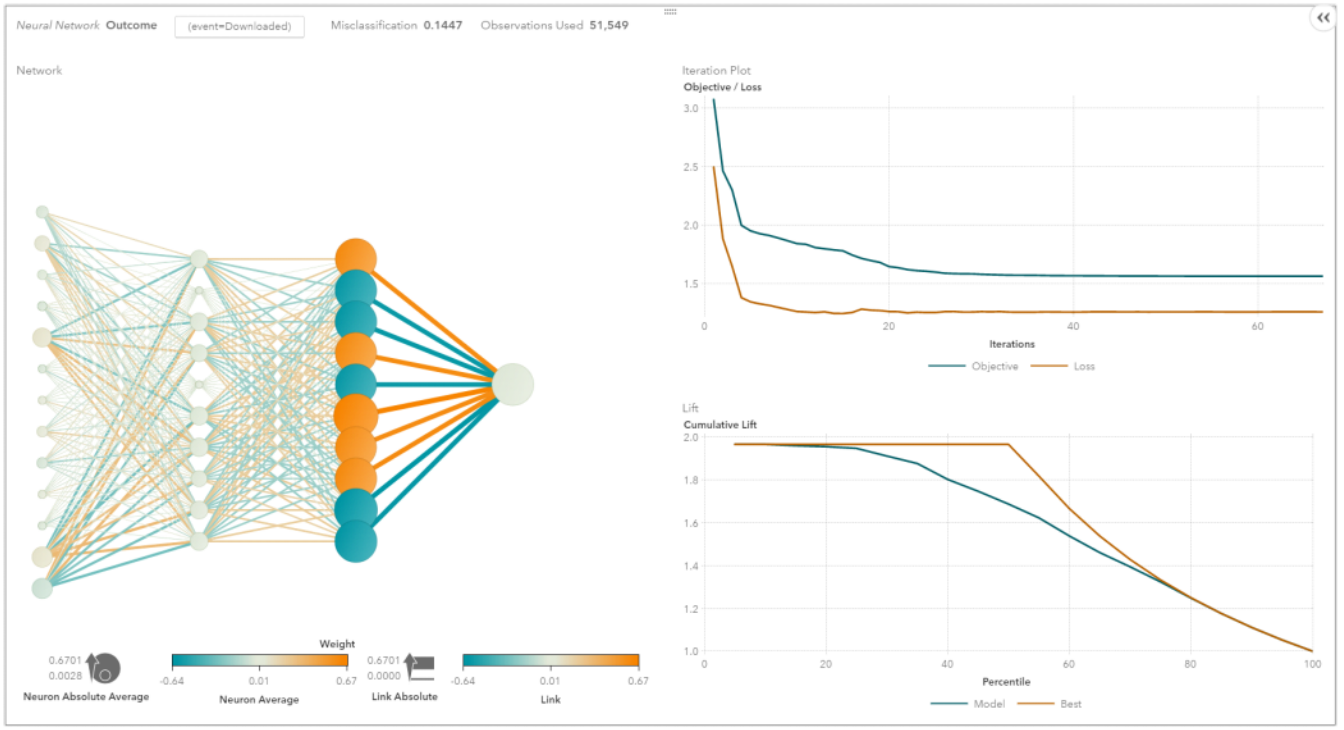
Neural networks and deep learning



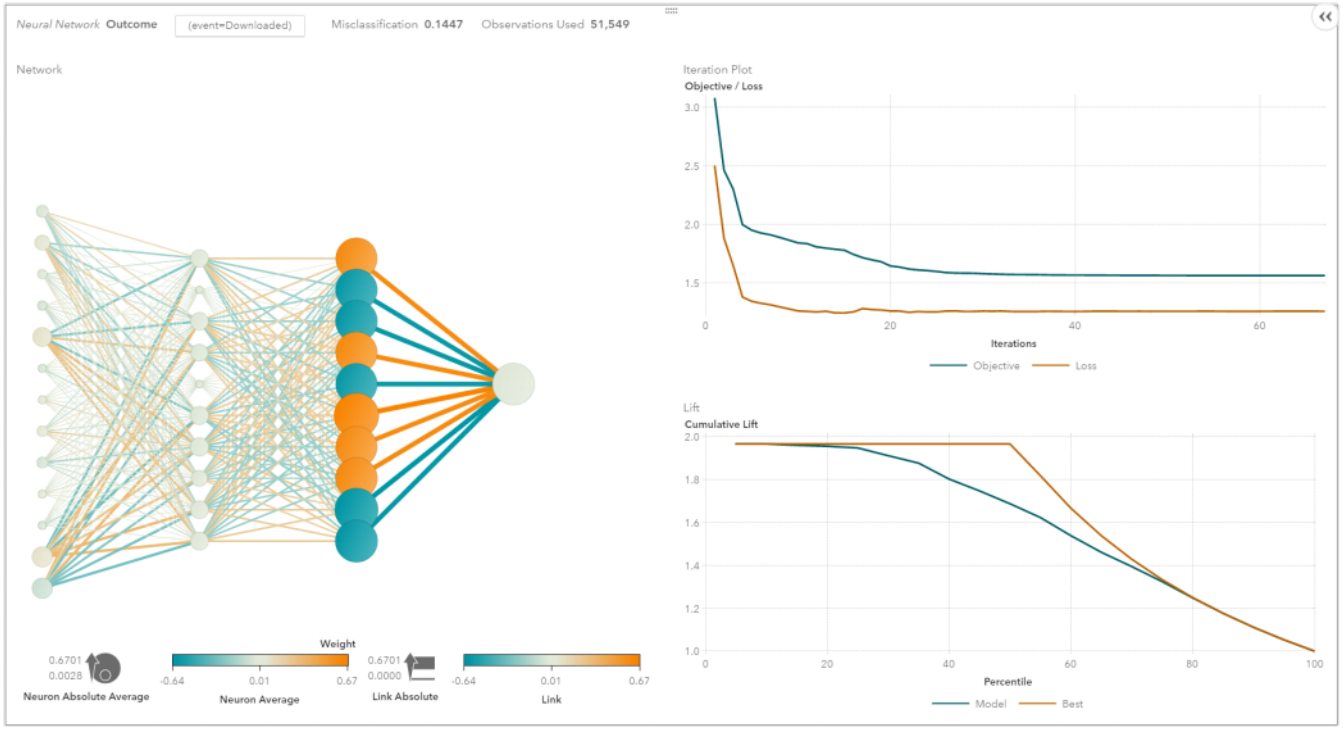
A convolution neural network architecture

Neural networks flourished in the mid-1980s due to their parallel and distributed processing ability. But research in this field was impeded by the ineffectiveness of the back-propagation training algorithm that is widely used to optimize the parameters of neural networks. Support vector machines (SVM) and other simpler models, which can be easily trained by solving convex optimization problems, gradually replaced neural networks in machine learning.

In recent years, new and improved training techniques such as unsupervised pre-training and layer-wise greedy training have led to a resurgence of interest in neural networks. Increasingly powerful computational capabilities, such as graphical processing unit (GPU) and massively parallel processing (MPP), have also spurred the revived adoption of neural networks. The resurgent research in neural networks has given rise to the invention of models with thousands of layers.



A neural network

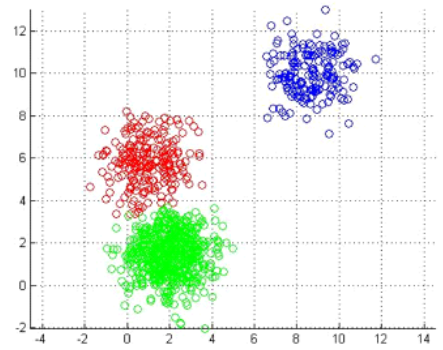


A neural network

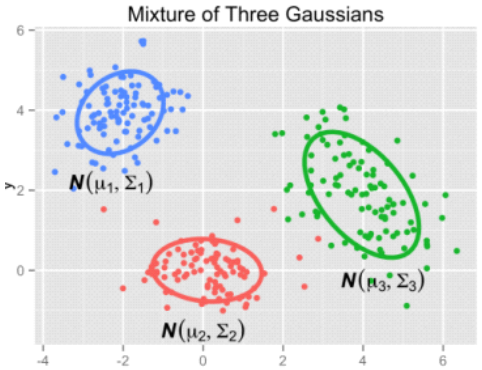
In other words, shallow neural networks have evolved into deep learning neural networks. Deep neural networks have been very successful for supervised learning. When used for speech and image recognition, deep learning performs as well as, or even better than, humans. Applied to unsupervised learning tasks, such as feature extraction, deep learning also extracts features from raw images or speech with much less human intervention.

A neural network consists of three parts: input layer, hidden layers and output layer. The training samples define the input and output layers. When the output layer is a categorical variable, then the neural network is a way to address classification problems. When the output layer is a continuous variable, then the network can be used to do regression. When the output layer is the same as the input layer, the network can be used to extract intrinsic features. The number of hidden layers defines the model complexity and modeling capacity.

k-means/k-modes, GMM (Gaussian mixture model) clustering



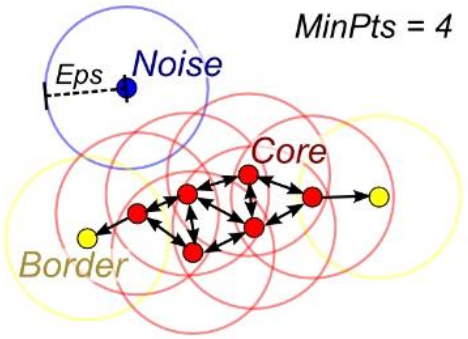
K-means clustering



Gaussian Mixture Model

Kmeans/k-modes, GMM clustering aims to partition n observations into k clusters. K-means define hard assignment: the samples are to be and only to be associated to one cluster. GMM, however define a soft assignment for each sample. Each sample has a probability to be associated with each cluster. Both algorithms are simple and fast enough for clustering when the number of clusters k is given.

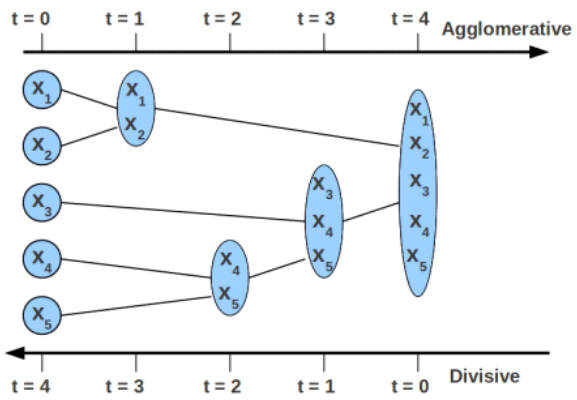
DBSCAN



A DBSCAN illustration

When the number of cluster k is not given, DBSCAN (density-based spatial clustering) can be used by connecting samples through density diffusion.

Hierarchical clustering



Two ways of Hierarchical clustering illustration

Hierarchical partitions can be visualized using a tree structure (a dendrogram). It does not need the number of clusters as an input and the partitions can be viewed at different levels of granularities (i.e., can refine/coarsen clusters) using different K.

PCA, SVD and LDA

We generally do not want to feed a large number of features directly into a machine learning algorithm since some features may be irrelevant or the "intrinsic" dimensionality may be smaller than the number of features. The PCA, SVD and LDA all can be used to perform dimension reduction.

The PCA is an unsupervised clustering method which maps the original data space into a lower dimensional space while preserving as much information as possible. The PCA basically finds a subspace that most preserves the data variance, with the subspace defined by the dominant eigenvectors of the data's covariance matrix.

The SVD is related to PCA in the sense that SVD of the centered data matrix (features versus samples) provides the dominant left singular vectors that define the same subspace as found by PCA. However, SVD is a more versatile technique as it can also do things that PCA may not do. For example, the SVD of a user-versus-movie matrix is able to extract the user profiles and movie profiles which can be used in a recommendation system. In addition, SVD is also widely used as a topic modeling tool, known as latent semantic analysis, in natural language processing (NLP).

A related technique in NLP is latent Dirichlet allocation (LDA). LDA is probabilistic topic model and it decomposes documents into topics in a similar way as a Gaussian mixture model (GMM) decomposes continuous data into Gaussian densities. Differently from the GMM, an LDA models discrete data (words in documents) and it constrains that the topics are *a priori* distributed according to a Dirichlet distribution.

Conclusions

This is the work flow which is easy to follow. The takeaway messages when trying to solve a new problem are:

- Define the problem. What problems do you want to solve?
- Start simple. Be familiar with the data and the baseline results.
- Then try something more complicated.

SAS Visual Data Mining and Machine Learning provides a good platform for beginners to learn machine learning and apply machine learning methods to their problems.

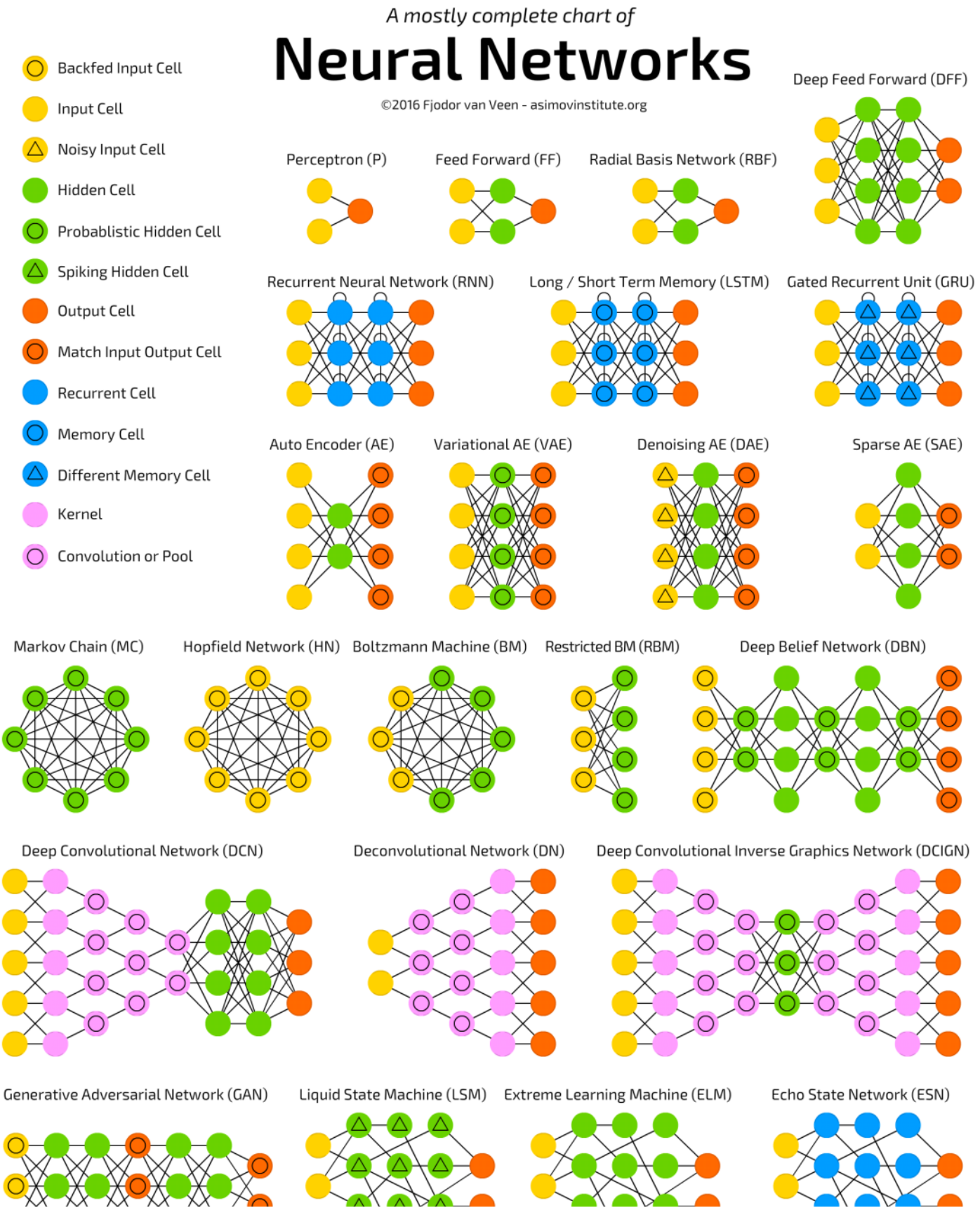
From <http://blogs.sas.com/content/subconsciousmusings/2017/04/12/machine-learning-algorithm-use/?imm_mid=0f1128&cmp=em-data-na-na-newsltr_ai_20170424#prettyPhoto>

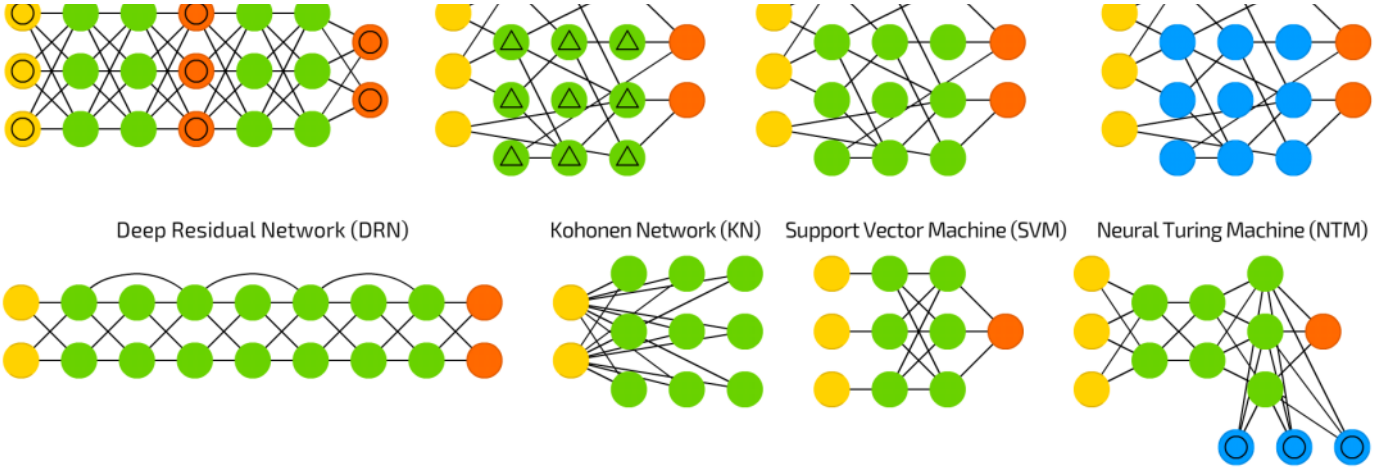
Neural Networks

July-17-17 3:54 PM

https://becominghuman.ai/cheat-sheets-for-ai-neural-networks-machine-learning-deep-learning-big-data-678c51b4b463

Neural Networks





Neural Networks Cheat Sheet

Linear Vector Spaces:

Definition: A linear vector space, X, is a set of elements (vectors) defined over a scalar field, F, that satisfies the following conditions:

1) if $x \in X$ and $y \in X$ then $x+y \in X$. **2)** $x+y=y+x$ **3)** $(x+y)+z=x+(y+z)$

4) There is a unique vector $0 \in X$, such that $x+0=x$ for all $x \in X$.

5) For each vector $x \in X$ there is a unique vector in X, to be called $(-x)$, such that $x+(-x)=0$. **6)** multiplication, for all scalars $a \in F$, and all vectors $x \in X$,

7) For any $x \in X$, $1x=x$ (for scalar 1).

8) For any two scalars $a \in F$ and $b \in F$ and any $x \in X$, $a(bx)=(ab)x$.

9) $(a+b)x=ax+bx$. **10)** $a(x+y)=ax+ay$.

Linear Independence:

Consider n vectors $\{x_1, x_2, .., x_n\}$. If there exists n scalars $a_1, a_2, \ldots, a_n$, at least one of which is nonzero, such that $a_1x_1 + a_2x_2 + \ldots + a_nx_n = 0$, then the $\{x_i\}$ are linearly dependent.

Spanning a Space:

Let X be a linear vector space and let $\{u_1, u_2, \ldots, u_n\}$ be a subset of vectors in X. This subset spans X if and only if for every vector $x \in X$ there exist scalars $x_1, x_2, \ldots, x_n$ such that $x = x_1u_1 + x_2u_2 + \ldots + x_nu_n$.

Inner Product:

(x, y) for any scalar function of x and y.

1. $(x,y) = (y,x)$ 2. $(x, ay_1 + by_2) = a(x,y_1) + b(x,y_2)$
3. $(x,x) \geq 0$, where equality holds iff x is the zero vector.

Norm:

Ascalar function $\|x\|$ is called a norm if it satisfies:

1. $\|x\| \geq 0$ 2. $\|x\| = 0$ if and only if $x = 0$.
3. $\|ax\| = |a|\|x\|$ 4. $\|x + y\| \leq \|x\| + \|y\|$

Angle:

The angle θ bet. 2 vectors x and y is defined by $\cos\theta = \frac{(x,y)}{\|x\|\,\|y\|}$

Orthogonality:

2 vectors $x, y \in X$ are said to be orthogonal if $(x,y) = 0$.

Gram Schmidt Orthogonalization:

Assume that we have n independent vectors $y_1, y_2, \ldots, y_n$. From these vectors we will obtain n orthogonal vectors $v_1, v_2, \ldots, v_n$.

$$v_1 = y_1, \quad v_k = y_k - \sum_{i=1}^{k-1} \frac{(v_i, y_k)}{(v_i, v_i)} v_i,$$

$$where\ \frac{(v_i, y_k)}{(v_i, v_i)} v_i\ is\ the\ projection\ of\ y_k\ on\ v_i$$

Vector Expansions:

$$x = \sum_{i=1}^{n} x_i v_i = x_1v_1 + x_2v_2 + \cdots + x_nv_n,$$

$$for\ orthogonal\ vectors, x_j = \frac{(v_j, x)}{(v_j, v_j)}$$

Reciprocal Basis Vectors:

$$(r_i, v_j) = \begin{cases} 0 & i \neq j \\ 1 & i = j \end{cases}, \quad x_j = (r_j, x)$$

To compute the reciprocal basis vectors: set $\mathbf{B} = [v_1\ v_2\ \ldots\ v_n]$,

$\mathbf{R} = [r_1\ r_2\ \ldots\ r_n]$, $\mathbf{R}^T = \mathbf{B}^{-1}$ In matrix form: $\mathbf{x}^v = \mathbf{B}^{-1}\mathbf{x}^s$

Transformations:

A *transformation* consists of three parts:

domain: $X = \{x_i\}$, range: $Y = \{y_i\}$, and a rule relating each $x_i \in X$ to an element $y_i \in Y$.

Linear Transformations: transformation A is *linear* if:

1. for all $x_1, x_2 \in X$, $A(x_1 + x_2) = A(x_1) + A(x_2)$

Perceptron Architecture:

$\mathbf{a} = hardlim(\mathbf{Wp} + \mathbf{b})$, $\mathbf{W} = [{}_1\mathbf{w}^T\ {}_2\mathbf{w}^T\ \ldots\ {}_S\mathbf{w}^T]^T$,

$a_i = hardlim(n_i) = hardlim({}_i\mathbf{w}^T\,\mathbf{p} + b_i)$

Decision Boundary:

${}_i\mathbf{w}^T\,\mathbf{p} + b_i = 0$

The decision boundary is always orthogonal to the weight vector.

Single-layer perceptrons can only classify linearly separable vectors.

Perceptron Learning Rule

$$\mathbf{W}^{new} = \mathbf{W}^{old} + \mathbf{ep}^T, \mathbf{b}^{new} = \mathbf{b}^{old} + \mathbf{e},$$

$$where\ \ \mathbf{e} = \mathbf{t} - \mathbf{a}$$

Hebb's Postulate:

"When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased."

Linear Associator:

$\mathbf{a} = purelin(\mathbf{Wp})$

The Hebb Rule:

Supervised Form: $w_{ij}^{new} = w_{ij}^{old} + t_{qi}P_{qi}$

$$\mathbf{W} = \boldsymbol{t}_1\boldsymbol{P}_1^T + \boldsymbol{t}_2\boldsymbol{P}_2^T + \cdots + \boldsymbol{t}_Q\boldsymbol{P}_Q^T$$

$$\mathbf{W} = [\mathbf{t}_1\ \mathbf{t}_2\ \ldots \mathbf{t}_Q]\begin{bmatrix}\mathbf{p}_1^T \\ \mathbf{p}_2^T \\ \vdots \\ \mathbf{p}_Q^T\end{bmatrix} = \mathbf{TP}^T$$

Pseudoinverse Rule:

$\mathbf{W} = \mathbf{TP}^+$

When the number, R, of rows of $\mathbf{P}$ is greater than the num ber of columns,Q, of $\mathbf{P}$ and the columns of $\mathbf{P}$ are independent, then the pseudoinverse can be computed by $\mathbf{P}^+ = (\mathbf{P}^T\mathbf{P})^{-1}\mathbf{P}^T$

Variations of Hebbian Learning:

Filtered Learning (Ch.14): $\mathbf{W}^{new} = (1-\gamma)\mathbf{W}^{old} + \alpha\mathbf{t}_q\mathbf{p}_q^T$

Delta Rule (Ch.10): $\mathbf{W}^{new} = \mathbf{W}^{old} + \alpha(\mathbf{t}_q - \mathbf{a}_q)\mathbf{p}_q^T$

Unsupervised Hebb (Ch.13): $\mathbf{W}^{new} = \mathbf{W}^{old} + \alpha\mathbf{a}_q\mathbf{p}_q^T$

Taylor:

$$F(\mathbf{x}) = F(\mathbf{x}^*) + \nabla F(\mathbf{x})^T|_{\mathbf{x}=\mathbf{x}^*}(\mathbf{x}-\mathbf{x}^*) + \frac{1}{2}(\mathbf{x}-\mathbf{x}^*)\nabla^2 F(\mathbf{x})^T|_{\mathbf{x}=\mathbf{x}^*}(\mathbf{x}-\mathbf{x}^*) + \cdots$$

Grad

$$\nabla F(\mathbf{x}) = \left[\frac{\partial}{\partial x_1}F(\mathbf{x})\ \ \frac{\partial}{\partial x_2}F(\mathbf{x})\ \ \ldots\ \frac{\partial}{\partial x_n}F(\mathbf{x})\right]^T$$

Hessian:

$$\nabla^2 F(\mathbf{x}) = \begin{bmatrix} \frac{\partial}{\partial x_1^2}F(\mathbf{x}) & \frac{\partial}{\partial x_1 \partial x_2}F(\mathbf{x}) \ldots & \frac{\partial}{\partial x_1 \partial x_n}F(\mathbf{x}) \\ \frac{\partial}{\partial x_2 \partial x_1}F(\mathbf{x}) & \frac{\partial}{\partial x_2^2}F(\mathbf{x}) \ldots & \frac{\partial}{\partial x_2 \partial x_n}F(\mathbf{x}) \\ \vdots & \vdots & \vdots \\ \frac{\partial}{\partial x_n \partial x_1}F(\mathbf{x}) & \frac{\partial}{\partial x_n \partial x_2}F(\mathbf{x}) \ldots & \frac{\partial}{\partial x_n^2}F(\mathbf{x}) \end{bmatrix}$$

Directional Derivatives:

1st Dir.Der.: $\frac{\mathbf{p}^T\nabla F(\mathbf{x})}{\|\mathbf{p}\|}$, **2nd Dir.Der.:** $\frac{\mathbf{p}^T\nabla^2 F(\mathbf{x})\mathbf{p}}{\|\mathbf{p}\|^2}$

Minima:

domain: $X = \{ x_i \}$, range: $Y = \{ y_i \}$, and a rule relating each $x_i \in X$ to an element $y_i \in Y$.

Linear Transformations: transformation A is *linear* if:

1. for all $x_1, x_2 \in X$, $A(x_1+x_2) = A(x_1) + A(x_2)$
2. for all $x \in X$, $a \in R$, $A(ax) = aA(x)$

Matrix Representations:

Let $\{v_1, v_2, \dots, v_n\}$ be a basis for vector space X, and let $\{u_1, u_2, \dots, u_n\}$ be a basis for vector space Y. Let A be a linear transformation with domain X and range Y: $A(x) = y$

The coefficients of the matrix representation are obtained from

$$A(v_j) = \sum_{i=1}^{m} a_{ij} u_i$$

Change of Basis: $\mathbf{B}_t = [\mathbf{t}_1\ \mathbf{t}_2\ \dots \mathbf{t}_n]$, $\mathbf{B}_w = [\mathbf{w}_1\ \mathbf{w}_2\ \dots \mathbf{w}_n]$

$$\mathbf{A}' = [\mathbf{B}_w^{-1}\mathbf{A}\mathbf{B}_t]$$

Eigenvalues & Eigenvectors: $\mathbf{Az} = \lambda\mathbf{z}$, $|[\mathbf{A} - \lambda\mathbf{I}]| = 0$

Diagonalization: $\mathbf{B} = [\mathbf{z}_1\ \mathbf{z}_2\ \dots \mathbf{z}_n]$,

where $\{\mathbf{z}, \mathbf{z}_2, \dots, \mathbf{z}_n\}$ are the eigenvectors of a square matrix A,

$$[\mathbf{B}^{-1}\mathbf{A}\mathbf{B}] = \text{diag}([\lambda_1\ \lambda_2 \dots \lambda_n])$$

Directional Derivatives:

1st Dir.Der.: $\frac{\mathbf{p}^T \nabla F(\mathbf{x})}{\|\mathbf{p}\|}$, **2nd Dir.Der.:** $\frac{\mathbf{p}^T \nabla^2 F(\mathbf{x})\mathbf{p}}{\|\mathbf{p}\|^2}$

Minima:

Strong Minimum: if a scalar $\delta > 0$ exists, such that $F(x) < F(x+\Delta x)$ for all Δx such that $\delta > \|\Delta x\| > 0$.

Global Minimum: if $F(x) < F(x+\Delta x)$ for all $\Delta x \neq 0$

Weak Minimum: if it is not a strong minimum, and a scalar $\delta > 0$ exists, such that $F(x) \leq F(x+\Delta x)$ for all Δx such that $\delta > \|\Delta x\| > 0$.

Necessary Conditions for Optimality:

1st-Order Condition: $\nabla F(\mathbf{x})|_{\mathbf{x}=\mathbf{x}^*} = 0$ (Stationary Points)

2nd-Order Condition: $\nabla^2 F(\mathbf{x})|_{\mathbf{x}=\mathbf{x}^*} \geq 0$ (Positive Semi-definite Hessian Matrix).

Quadratic fn.: $F(x) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\,\mathbf{x} + \mathbf{d}^T\mathbf{x} + c$

$$\nabla F(x) = \mathbf{Ax} + \mathbf{d},\ \nabla^2 F(x) = \mathbf{A},\ \boldsymbol{\lambda_{min}} \leq \frac{\mathbf{p}^T\mathbf{A}\,\mathbf{p}}{\|\mathbf{p}\|^2} \leq \boldsymbol{\lambda_{max}}$$

General Minimization Algorithm:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k \quad \textit{or} \quad \Delta\mathbf{x}_k = (\mathbf{x}_{k+1} - \mathbf{x}_k) = \alpha_k \mathbf{p}_k$$

Steepest Descent Algorithm:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \mathbf{g}_k \quad \text{where,} \quad \mathbf{g}_k = \nabla F(x)|_{\mathbf{x}=\mathbf{x}_k}$$

Stable Learning Rate: ($\alpha_k = \alpha$, constant) $\alpha < \frac{2}{\lambda_{max}}$

$\{\lambda_1\ \lambda_2, \dots, \lambda_n\}$ Eigenvalues of Hessian matrix A

Learning Rate to Minimize Along the Line:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k \overset{is}{\Rightarrow} \alpha_k = -\frac{\mathbf{g}_k^T\mathbf{P}_k}{\mathbf{P}_k{}^T\mathbf{A}\mathbf{P}_k} \text{ (For quadratic fn.)}$$

After Minimization Along the Line:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k \ \Rightarrow\ \mathbf{g}_{k+1}^T \mathbf{p}_k = 0$$

ADALINE: $\mathbf{a} = purelin(\mathbf{Wp} + \mathbf{b})$

Mean Square Error: *(for ADALINE it is a quadratic fn.)*

$$F(\mathbf{x}) = E[e^2] = E[(t - a)^2] = E[(t - \mathbf{x}^T\mathbf{z})^2]$$

$$F(\mathbf{x}) = c - 2\mathbf{x}^T\mathbf{h} + \mathbf{x}^T\mathbf{R}\,\mathbf{x},$$

$c = E[t^2]$, $\mathbf{h} = E[t\mathbf{z}]$ *and* $\mathbf{R} = E[\mathbf{z}\mathbf{z}^T] \Rightarrow \mathbf{A} = 2\mathbf{R}, \mathbf{d} = -2\mathbf{h}$

Unique minimum, if it exists, is $\mathbf{x}^* = \mathbf{R}^{-1}\mathbf{h}$,

where $\mathbf{x} = \begin{bmatrix} {}_1\mathbf{w} \\ b \end{bmatrix}$ and $\mathbf{z} = \begin{bmatrix} \mathbf{p} \\ 1 \end{bmatrix}$

LMS Algorithm: $\mathbf{W}(k+1) = \mathbf{W}(k) + 2\alpha\,\mathbf{e}(k)\,\mathbf{p}^T(k)$

$$\mathbf{b}(k+1) = \mathbf{b}(k) + 2\alpha\,\mathbf{e}(k)$$

Convergence Point: $\mathbf{x}^* = \mathbf{R}^{-1}\mathbf{h}$

Stable Learning Rate: $0 < \alpha < 1/\boldsymbol{\lambda_{max}}$ where $\boldsymbol{\lambda_{max}}$ is the maximum eigenvalue of R

Adaptive Filter ADALINE:

$$a(k) = purelin(\mathbf{W}\mathbf{p}(k) + b) = \sum_{i=1}^{R} \mathbf{w}_{1,i}\, y(k - i + 1) + b$$

Backpropagation Algorithm:

Performance Index:

Mean Square error: $F(\mathbf{x}) = E[\mathbf{e}^T\mathbf{e}] = E[(\mathbf{t} - \mathbf{a})^T(\mathbf{t} - \mathbf{a})]$

Approximate Performance Index: (single sample)

$$\hat{F}(x) = \mathbf{e}^T(k)\mathbf{e}(k) = \big(\mathbf{t}(k) - \mathbf{a}(k)\big)^T\big(\mathbf{t}(k) - \mathbf{a}(k)\big)$$

Sensitivity: $\mathbf{s}^m = \frac{\partial \hat{F}}{\partial \mathbf{n}^m} = \left[\frac{\partial \hat{F}}{\partial n_1^m} \quad \frac{\partial \hat{F}}{\partial n_2^m} \quad \dots \quad \frac{\partial \hat{F}}{\partial n_{s^m}^m}\right]^T$

Forward Propagation: $\mathbf{a}^0 = \mathbf{p}$,

$$\mathbf{a}^{m+1} = \mathbf{f}^{m+1}(\mathbf{W}^{m+1}\mathbf{a}^m + \mathbf{b}^{m+1}) \ for\ m = 0, 1, \dots, M-1$$

$$\mathbf{a} = \mathbf{a}^M$$

Backward Propagation: $\mathbf{s}^M = -2\dot{\mathbf{F}}^M(\mathbf{n}^M)(\mathbf{t} - \mathbf{a})$,

***Heuristic Variations of Backpropagation:**

Batching: The parameters are updated only after the entire training set has been presented. The gradients calculated for each training example are averaged together to produce a more accurate estimate of the gradient.(If the training set is complete, i.e., covers all possible input/output pairs, then the gradient estimate will be exact.)

Backpropagation with Momentum (MOBP):

$$\Delta\mathbf{W}^m(k) = \gamma\Delta\mathbf{W}^m(k-1) - (1-\gamma)\alpha\,\mathbf{s}^m(\mathbf{a}^{m-1})^T$$

$$\Delta\mathbf{b}^m(k) = \gamma\Delta\mathbf{b}^m(k-1) - (1-\gamma)\alpha\,\mathbf{s}^m$$

Variable Learning Rate Backpropagation (VLBP)

1. If the squared error (over the entire training set) increases by more than some set percentage ζ (typically one to five percent) after a weight update, then the weight update is discarded, the learning rate is multiplied by some factor $\rho < 1$, and the momentum coefficient γ (if it is used) is set to zero.
2. If the squared error decreases after a weight update, then the weight update is accepted and the learning rate is multiplied by some factor $\eta > 1$. If γ has been previously set to zero, it is reset to its original value.
3. If the squared error increases by less than ζ, then the weight update is accepted but the learning rate and the momentum coefficient are unchanged.

Association: $\mathbf{a} = hardlim(\mathrm{W}^0\mathrm{P}^0 + \mathbf{Wp} + b)$

An association is a link between the inputs and outputs of a network so that when a stimulus A is presented to the network, it will output a response B.

Associative Learning Rules:

Unsupervised Hebb Rule:

$$\mathbf{W}(q) = \mathbf{W}(q-1) + \alpha\,\mathbf{a}(q)\mathbf{p}^T(q)$$

Hebb with Decay:

$$\mathbf{W}(q) = (1-\gamma)\mathbf{W}(q-1) + \alpha\,\mathbf{a}(q)\mathbf{p}^T(q)$$

Instar: $\mathbf{a} = hardlim(\mathbf{Wp} + b)$, $\mathbf{a} = hardlim({}_1\mathbf{w}^T\mathbf{p} + b)$

The instar is activated for ${}_1\mathbf{w}^T\mathbf{p} = \|{}_1\mathbf{w}\|\|\mathbf{p}\| cos\theta \geq -b$

where θ is the angle between $\mathbf{p}$ and ${}_1\mathbf{w}$.

Instar Rule:

$${}_i\mathbf{w}(q) = {}_i\mathbf{w}(q-1) + \alpha\, a_i(q)(\mathbf{p}(q) - {}_i\mathbf{w}(q-1))$$

$${}_i\mathbf{w}(q) = (1-\alpha)\,{}_i\mathbf{w}(q-1) + \alpha\,\mathbf{p}(q),\ if\ (a_i(q) = 1)$$

Kohonen Rule:

${}_i\mathbf{w}(q) = {}_i\mathbf{w}(q-1) + \alpha\,(\mathbf{p}(q) - {}_i\mathbf{w}(q-1))$ *for* $i \in X(q)$

Outstar Rule: $\mathbf{a} = satlins(\mathbf{Wp})$

$$\mathbf{w}_j(q) = \mathbf{w}_j(q-1) + \alpha\,\Big(\mathbf{a}(q) - \mathbf{w}_j(q-1)\Big)\,\mathrm{p}_j(q)$$

Competitive Layer: $\mathbf{a} = compet(\mathbf{Wp}) = compet(\mathbf{n})$

Competitive Learning with the Kohonen Rule:

$${}_{i^*}\mathbf{w}(q) = {}_{i^*}\mathbf{w}(q-1) + \alpha\,\Big(\mathbf{p}(q) - {}_{i^*}\mathbf{w}(q-1)\Big)$$

$$= (\mathbf{1} - \alpha)\,{}_{i^*}\mathbf{w}(q-1) + \alpha\,\mathbf{p}(q)$$

${}_{i^*}\mathbf{w}(q) = {}_{i^*}\mathbf{w}(q-1)$, $i \neq i^*$ where i^* is the winning neuron.

Self-Organizing with the Kohonen Rule:

$${}_i\mathbf{w}(q) = {}_i\mathbf{w}(q-1) + \alpha\,\Big(\mathbf{p}(q) - {}_i\mathbf{w}(q-1)\Big)$$

$$= (\mathbf{1} - \alpha)\,{}_i\mathbf{w}(q-1) + \alpha\,\mathbf{p}(q),\ i \in N_{i^*}(d)$$

$$\mathbf{a} = \mathbf{a}^M$$

Backward Propagation: $\mathbf{s}^M = -2\dot{\mathbf{F}}^M(\mathbf{n}^M)(\mathbf{t} - \mathbf{a})$,

$\mathbf{s}^m = \dot{\mathbf{F}}^m(\mathbf{n}^m)(\mathbf{W}^{m+1})^T\mathbf{s}^{m+1}$ $for\ m = M-1, \dots, 2, 1$,where

$\dot{\mathbf{F}}^m(\mathbf{n}^m) = \text{diag}\left(\left[\dot{f}^m(n_1^m) \quad \dot{f}^m(n_2^m) \quad \dots \quad \dot{f}^m\left(n_{\mathbf{s}^m}^m\right)\right]\right)$

$$\dot{f}^m\left(n_j^m\right) = \frac{\partial f^m\left(n_j^m\right)}{\partial n_j^m}$$

Weight Update (Approximate Steepest Descent):

$$\mathbf{W}^m(k+1) = \mathbf{W}^m(k) - \alpha\mathbf{s}^m(\mathbf{a}^{m-1})^{\mathrm{T}}$$
$$\mathbf{b}^m(k+1) = \mathbf{b}^m(k) - \alpha\mathbf{s}^m$$

Self-Organizing with the Kohonen Rule:

$$_i\mathbf{w}(q) = {}_i\mathbf{w}(q-1) + \alpha\left(\mathbf{p}(q) - {}_i\mathbf{w}(q-1)\right)$$
$$= (\mathbf{1} - \alpha)\ {}_i\mathbf{w}(q-1) + \alpha\,\mathbf{p}(q)\ ,\ i \in N_{i^*}(d)$$
$$N_i(d) = \{j, d_{i,j} \le d\}$$

LVQ Network: $(w_{k,i}^2 = 1) \Rightarrow$ subclass i is a part of class k

$$n_i^1 = -\left\| {}_i\mathbf{w}^1 - \mathbf{p}\right\|, \mathbf{a}^1 = compet(\mathbf{n}^1)\ ,\ \mathbf{a}^2 = \mathbf{W}^2\mathbf{a}^1$$

LVQ Network Learning with the Kohonen Rule:

$$_{i^*}\mathbf{w}^1(q) = {}_{i^*}\mathbf{w}^1(q-1) + \alpha\left(\mathbf{p}(q) - {}_{i^*}\mathbf{w}^1(q-1)\right),$$
$$if\ a_{k^*}^2 = t_{k^*} = 1$$
$$_{i^*}\mathbf{w}^1(q) = {}_{i^*}\mathbf{w}^1(q-1) - \alpha\left(\mathbf{p}(q) - {}_{i^*}\mathbf{w}^1(q-1)\right),$$
$$if\ a_{k^*}^2 = 1 \neq t_{k^*} = 0$$

hardlim: $a = \begin{cases} 0 & n < 0 \\ 1 & n \ge 0 \end{cases}$, ***hardlims***: $a = \begin{cases} -1 & n < 0 \\ +1 & n \ge 0 \end{cases}$,***purelin***: $a = n$, ***Logsig***: $a = \frac{1}{1+e^{-n}}$, ***tansig***: $a = \frac{e^n - e^{-n}}{e^n + e^{-n}}$,***poslin***: $a = \begin{cases} 0 & n < 0 \\ n & n \ge 0 \end{cases}$,

compet: $a = \begin{cases} 1 & \text{neuron with max } n \\ 0 & \text{all other neurons} \end{cases}$, ***satlin***: $a = \begin{cases} 0 & n < 0 \\ n & -1 \le n \le 1 \\ 1 & n > 1 \end{cases}$, ***satlins***: $a = \begin{cases} -1 & n < 0 \\ n & -1 \le n \le 1 \\ 1 & n > 1 \end{cases}$,

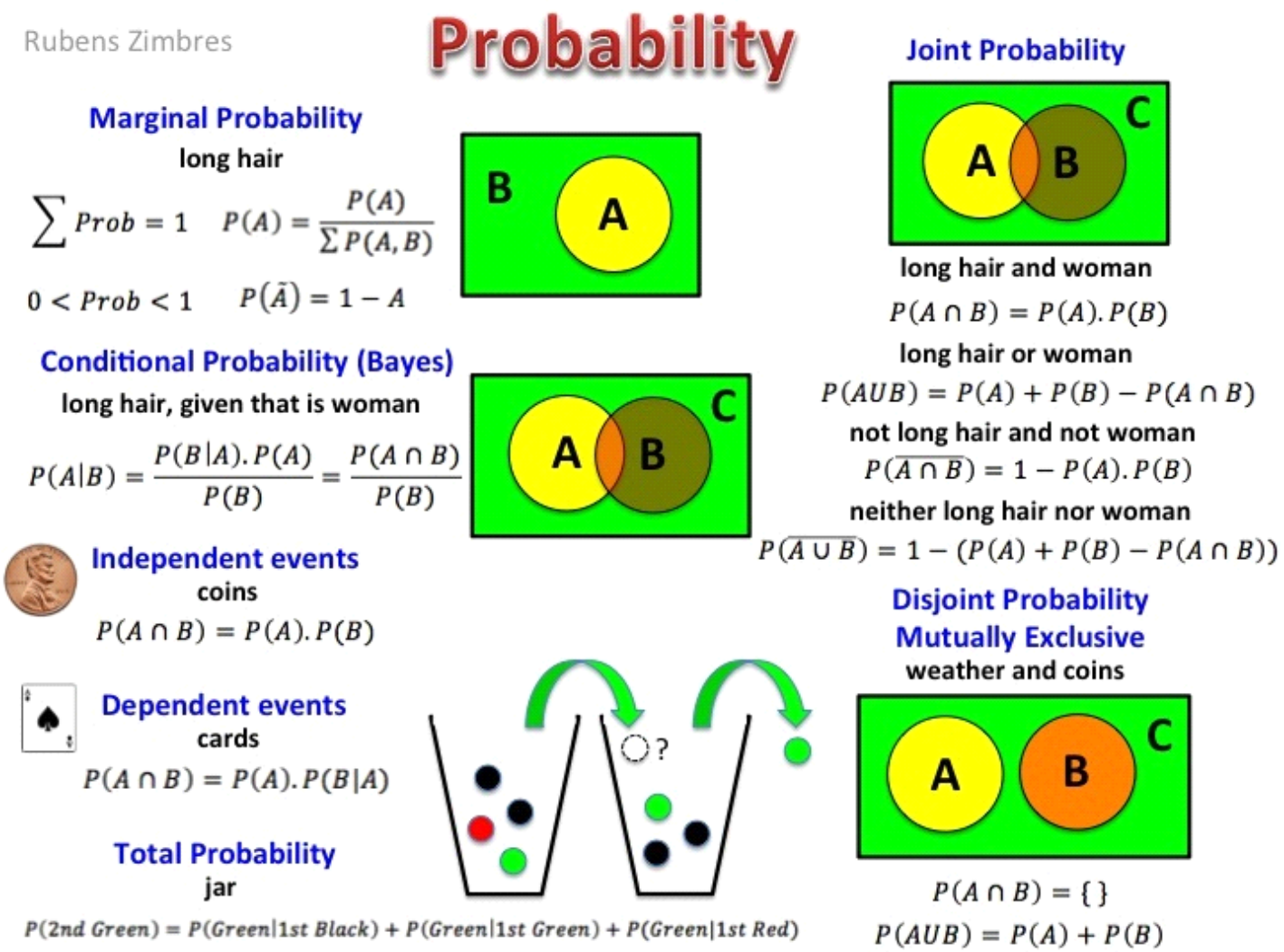
Delay: $a(t) = u(t-1)$, ***Integrator***: $a(t) = \int_0^t u(\tau)d\tau + a(0)$

**HINT:

$$diag([1\ 2\ 3]) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix}$$

Probability

April-12-17 5:39 PM



From <http://www.datasciencecentral.com/profiles/blogs/a-cheat-sheet-on-probability>

Python Quick Reference 2.7

April-13-17 11:59 AM

Python_qr

Python 2.7 Quick Reference Sheet

ver 2.01 – 110105 (sjd)

Interactive Help in Python Shell

help()	Invoke interactive help
help(*m*)	Display help for module *m*
help(*f*)	Display help for function *f*
dir(*m*)	Display names in module *m*

Small Operator Precedence Table

func_name(*args, ...*)	Function call
x[*index* : *index*]	Slicing
x[*index*]	Indexing
x.attribute	Attribute reference
**	Exponentiation
*, /, %	Multiply, divide, mod
+, -	Add, subtract
>, <, <=, >=, !=, ==	Comparison
in, not in	Membership tests
not, and, or	Boolean operators NOT, AND, OR

Module Import

import *module_name*
from *module_name* import *name* , ...
from *module_name* import *

Common Data Types

Type	Description	Literal Ex
int	32-bit Integer	3, -4
long	Integer > 32 bits	101L
float	Floating point number	3.0, -6.55
complex	Complex number	1.2J
bool	Boolean	True, False
str	Character sequence	"Python"
tuple	Immutable sequence	(2, 4, 7)
list	Mutable sequence	[2, x, 3.1]
dict	Mapping	{ x:2, y:5 }

Common Syntax Structures

Assignment Statement
var = exp

Console Input/Output
var = input([*prompt*])
var = raw_input([*prompt*])
print *exp*[,] ...

Selection
if (*boolean_exp*):
  stmt ...
[elif (*boolean_exp*):
  stmt ...] ...
[else:
  stmt ...]

Repetition
while (*boolean_exp*):
  stmt ...

Traversal
for *var* in *traversable_object*:
  stmt ...

Function Definition
def *function_name*(*parmameters*):
  stmt ...

Function Call
function_name(*arguments*)

Class Definition
class *Class_name* [(*super_class*)]:
  [*class variables*]
  def *method_name*(self, *parameters*):
    stmt

Object Instantiation
obj_ref = *Class_name*(*arguments*)

Method Invocation
obj_ref.method_name(*arguments*)

Exception Handling
try:
  stmt ...
except [*exception_type*] [, *var*]:
  stmt ...

Common Built-in Functions

Function	Returns
abs(*x*)	Absolute value of *x*
dict()	Empty dictionary, eg: d = dict()
float(*x*)	int or string *x* as float
id(*obj*)	memory addr of *obj*
int (*x*)	float or string *x* as int
len(*s*)	Number of items in sequence *s*
list()	Empty list, eg: m = list()
max(*s*)	Maximum value of items in *s*
min(*s*)	Minimum value of items in *s*
open(*f*)	Open filename *f* for input
ord(*c*)	ASCII code of *c*
pow(*x,y*)	x ** y
range(*x*)	A list of x ints 0 to *x* - 1
round(*x,n*)	float *x* rounded to *n* places
str(*obj*)	str representation of *obj*
sum(*s*)	Sum of numeric sequence *s*
tuple(*items*)	tuple of *items*
type(*obj*)	Data type of *obj*

Common Math Module Functions

Function	Returns (all float)
ceil(*x*)	Smallest whole nbr >= *x*
cos(*x*)	Cosine of *x* radians
degrees(*x*)	*x* radians in degrees
radians(*x*)	*x* degrees in radians
exp(*x*)	e ** *x*
floor(*x*)	Largest whole nbr <= *x*
hypot(*x, y*)	sqrt(*x* * *x* + *y* * *y*)
log(*x* [, *base*])	Log of *x* to *base* or natural log if *base* not given
pow(*x, y*)	x ** y
sin(*x*)	Sine of *x* radians
sqrt(*x*)	Positive square root of *x*
tan(*x*)	Tangent of *x* radians
pi	Math constant pi to 15 sig figs
e	Math constant e to 15 sig figs

Common String Methods

S.method()	Returns (str unless noted)
capitalize	*S* with first char uppercase
center(*w*)	*S* centered in str *w* chars wide
count(*sub*)	int nbr of non-overlapping occurrences of *sub* in *S*
find(*sub*)	int index of first occurrence of *sub* in *S* or -1 if not found
isdigit()	bool True if *S* is all digit chars, False otherwise
islower() isupper()	bool True if *S* is all lower/upper case chars, False otherwise
join(*seq*)	All items in *seq* concatenated into a str, delimited by *S*
lower() upper()	Lower/upper case copy of *S*
lstrip() rstrip()	Copy of *S* with leading/ trailing whitespace removed, or both
split([*sep*])	List of tokens in *S*, delimited by *sep*; if *sep* not given, delimiter is any whitespace

Formatting Numbers as Strings

Syntax: *"format_spec" % numeric_exp*
***format_spec* syntax:** *% width.precision type*

- *width* (optional): align in number of colums specified; negative to left-align, precede with 0 to zero-fill
- *precision* (optional): show specified digits of precision for floats; 6 is default
- *type* (required): d (decimal int), f (float), s (string), e (float – exponential notation)
- Examples for x = 123, y = 456.789
  "%6d" % x -> . . . 123
  "%06d" % x -> 000123
  "%8.2f % y -> . . 456.79
  "8.2e" % y -> 4.57e+02
  "-8s" % "Hello" -> Hello . . .

Common List Methods

L.method()	Result/Returns
append(*obj*)	Append *obj* to end of *L*
count(*obj*)	Returns int nbr of occurrences of *obj* in *L*
index(*obj*)	Returns index of first occurrence of *obj* in *L*; raises ValueError if *obj* not in *L*
pop([*index*])	Returns item at specified *index* or item at end of L if *index* not given; raises IndexError if *L* is empty or *index* is out of range
remove(*obj*)	Removes first occurrence of *obj* from *L*; raises ValueError if *obj* is not in *L*
reverse()	Reverses *L* in place
sort()	Sorts *L* in place

Common Tuple Methods

T.method()	Returns
count(*obj*)	Returns nbr of occurrences of *obj* in *T*
index(*obj*)	Returns index of first occurrence of *obj* in *T*; raises ValueError if *obj* is not in *T*

Common Dictionary Methods

D.method()	Result/Returns
clear()	Remove all items from *D*
get(*k* [,*val*])	Return *D*[*k*] if *k* in *D*, else *val*
has_key(*k*)	Return True if *k* in *D*, else False
items()	Return list of key-value pairs in *D*; each list item is 2-item tuple
keys()	Return list of *D*'s keys
pop(*k*, [*val*])	Remove key *k*, return mapped value or *val* if *k* not in *D*
values()	Return list of *D*'s values

Common File Methods

F.method()	Result/Returns
read([*n*])	Return str of next *n* chars from *F*, or up to EOF if *n* not given
readline([*n*])	Return str up to next newline, or at most *n* chars if specified
readlines()	Return list of all lines in *F*, where each item is a line
write(*s*)	Write str *s* to *F*
writelines(*L*)	Write all str in seq *L* to *F*
close()	Closes the file

Other Syntax

Hold window for user keystroke to close:

```
raw_input("Press <Enter> to quit.")
```

Prevent execution on import:

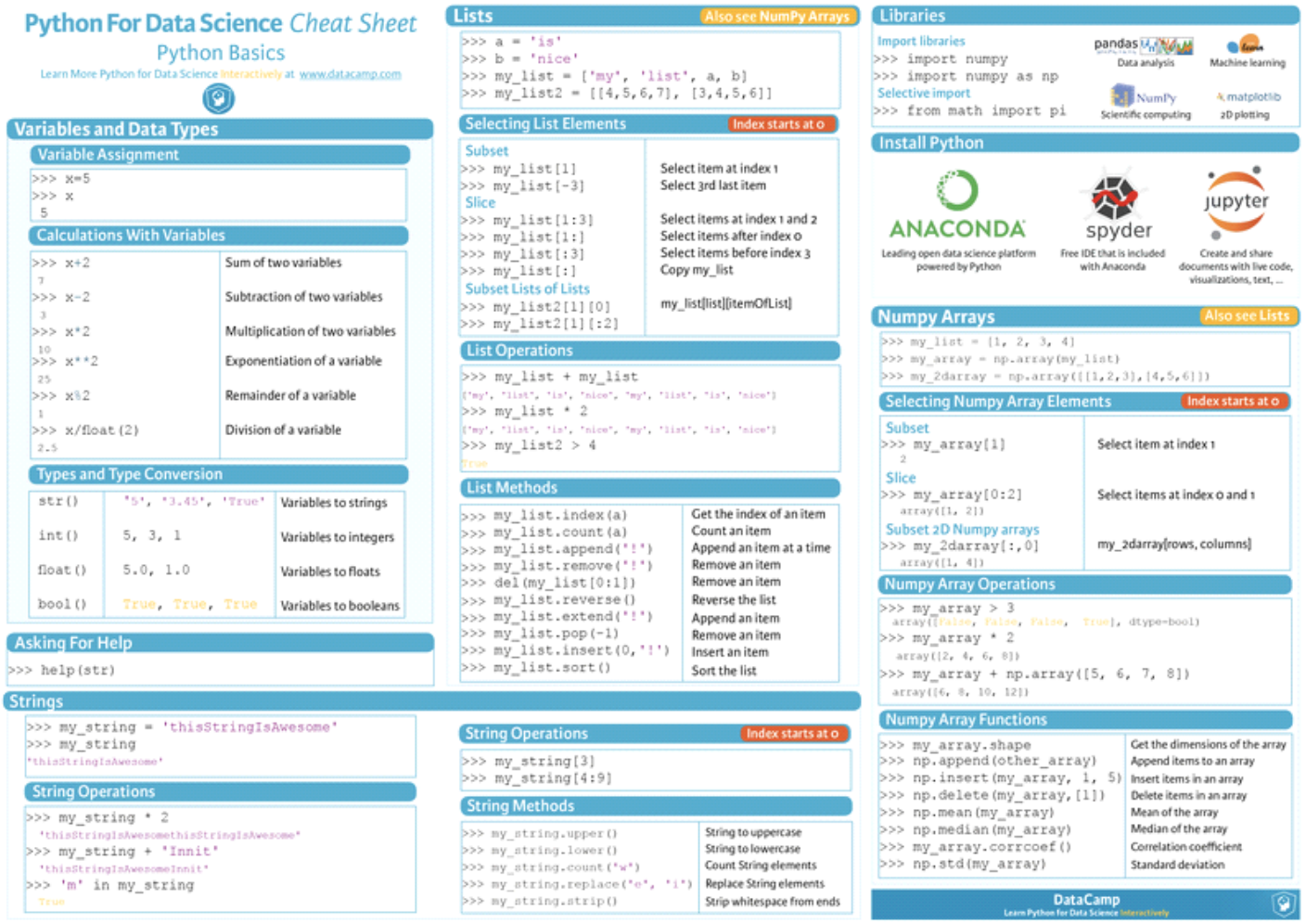
```
if __name__ == "__main__":
    main()
```

Displayable ASCII Characters

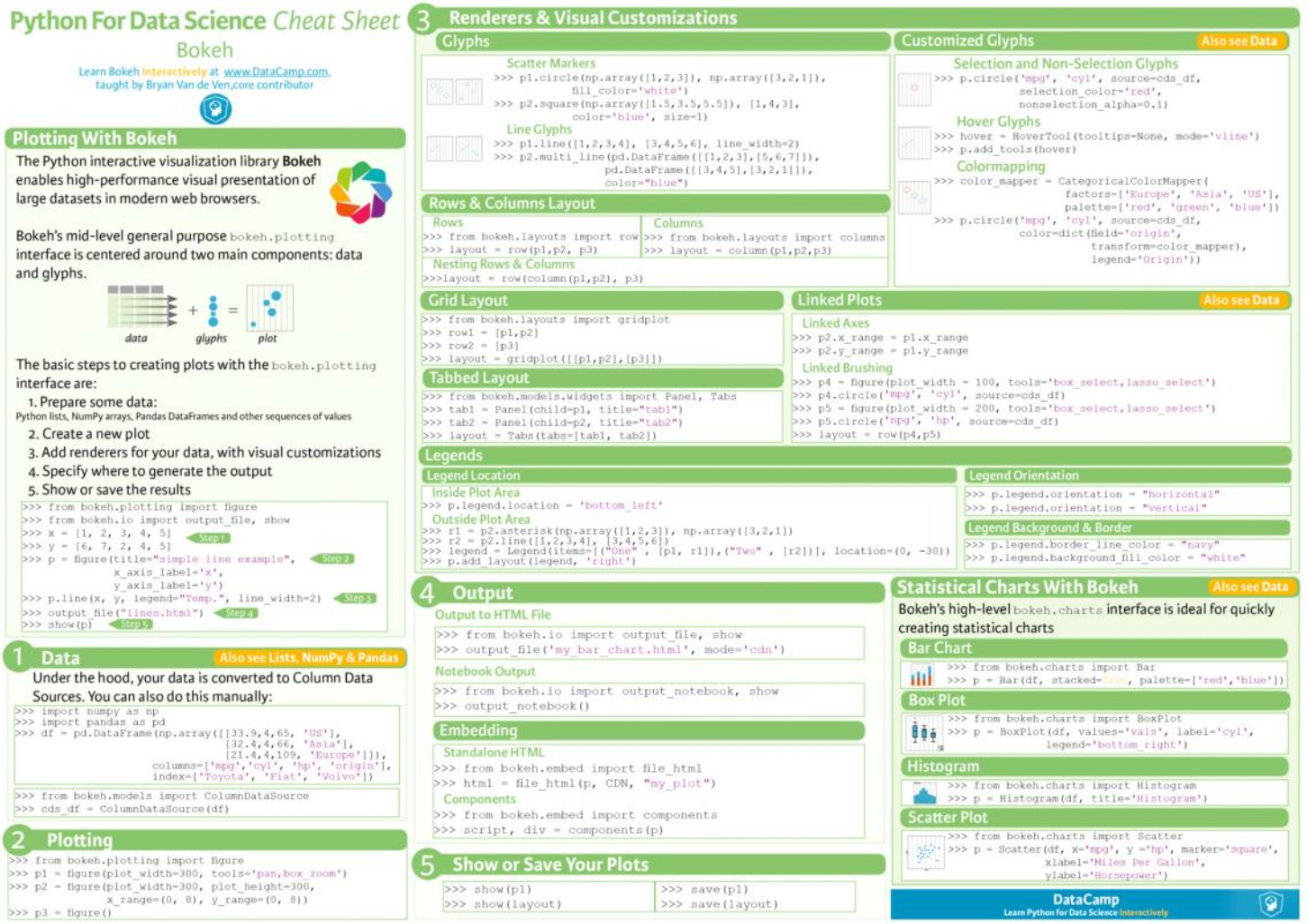
32	SP	48	0	64	@	80	P	96	`	112	p
33	!	49	1	65	A	81	Q	97	a	113	q
34	"	50	2	66	B	82	R	98	b	114	r
35	#	51	3	67	C	83	S	99	c	115	s
36	$	52	4	68	D	84	T	100	d	116	t
37	%	53	5	69	E	85	U	101	e	117	u
38	&	54	6	70	F	86	V	102	f	118	v
39	'	55	7	71	G	87	W	103	g	119	w
40	(	56	8	72	H	88	X	104	h	120	x
41	)	57	9	73	I	89	Y	105	i	121	y
42	*	58	:	74	J	90	Z	105	j	122	z
43	+	59	;	75	K	91	[	107	k	123	{
44	,	60	<	76	L	92	\	108	l	124	|
45	-	61	=	77	M	93	]	109	m	125	}
46	.	62	>	78	N	94	^	110	n	126	~
47	/	63	?	79	O	95	_	111	o	127	DEL

'\0' = 0, '\t' = 9, '\n' = 10

Python for Data Science



Python Data Science Cheat Sheet



Bokeh Cheat Sheet: https://s3.amazonaws.com/assets.datacamp.com/blog_assets/Python_Bokeh_Cheat_Sheet.pdf

Data Science Cheat Sheet: https://www.datacamp.com/community/tutorials/python-data-science-cheat-sheet-basics

From <https://becominghuman.ai/cheat-sheets-for-ai-neural-networks-machine-learning-deep-learning-big-data-678c51b4b463>

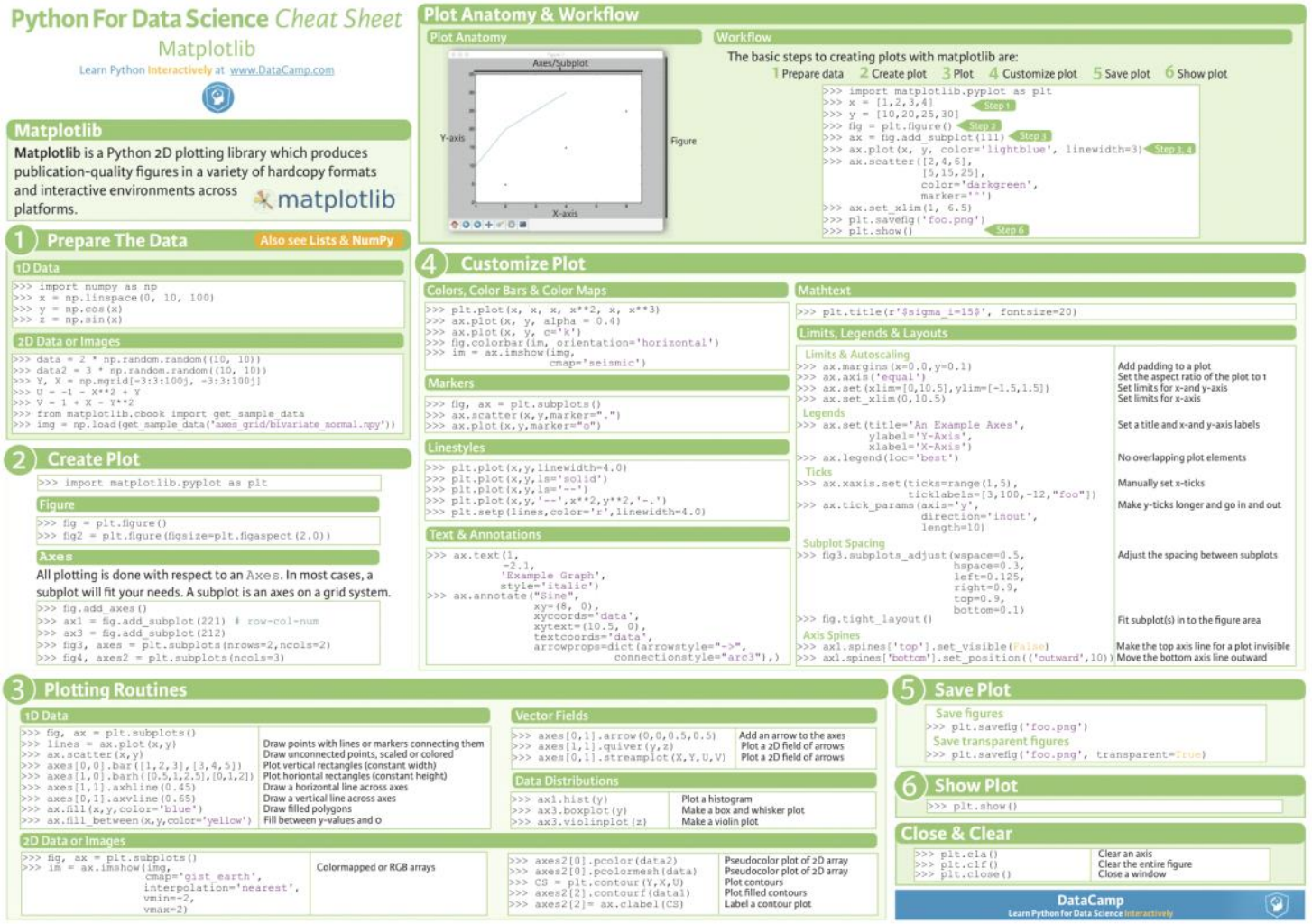
Python-MatplotLib

July-17-17 4:31 PM

Matplotlib

matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-orientedAPI for embedding plots into applications using general-purpose GUI toolkitslike Tkinter, wxPython, Qt, or GTK+. There is also a procedural "pylab" interface based on a state machine (like OpenGL), designed to closely resemble that of MATLAB, though its use is discouraged.[2] SciPy makes use of matplotlib.

pyplot is a matplotlib module which provides a MATLAB-like interface.[6]matplotlib is designed to be as usable as MATLAB, with the ability to use Python, with the advantage that it is free.



Matplotlib Cheat Sheet

Matplotlib Cheat Sheet: https://www.datacamp.com/community/blog/python-matplotlib-cheat-sheet#gs.uEKySpY

Matpotlib: https://en.wikipedia.org/wiki/Matplotlib

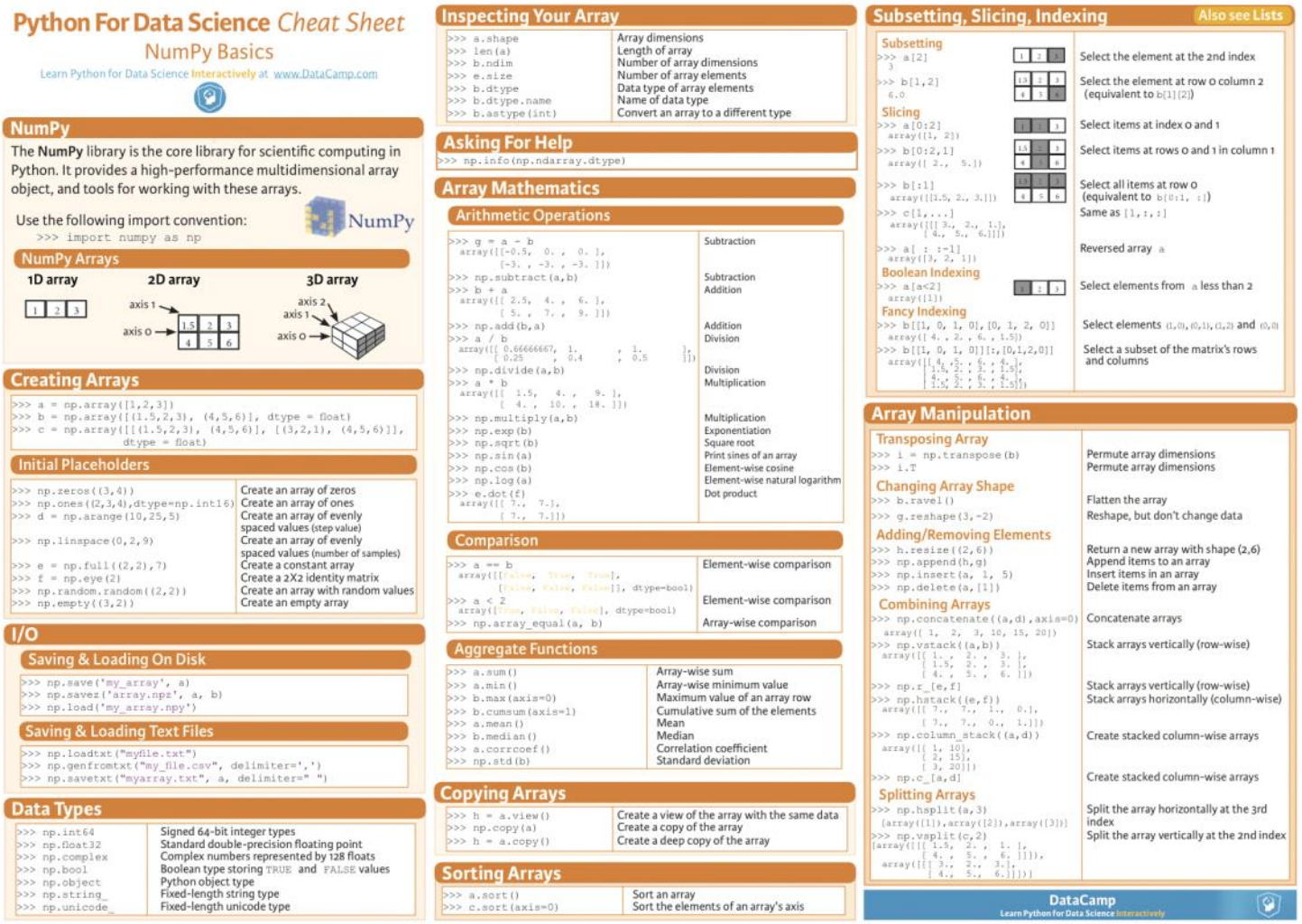
From <https://becominghuman.ai/cheat-sheets-for-ai-neural-networks-machine-learning-deep-learning-big-data-678c51b4b463>

Python-numpy

July-17-17 4:27 PM

Numpy

NumPy targets the CPython reference implementation of Python, which is a non-optimizing bytecode interpreter. Mathematical algorithms written for this version of Python often run much slower than compiled equivalents. NumPy address the slowness problem partly by providing multidimensional arrays and functions and operators that operate efficiently on arrays, requiring rewriting some code, mostly inner loops using NumPy.



Numpy Cheat Sheet: https://www.datacamp.com/community/blog/python-numpy-cheat-sheet#gs.AK5ZBgE

NumPy: https://en.wikipedia.org/wiki/NumPy

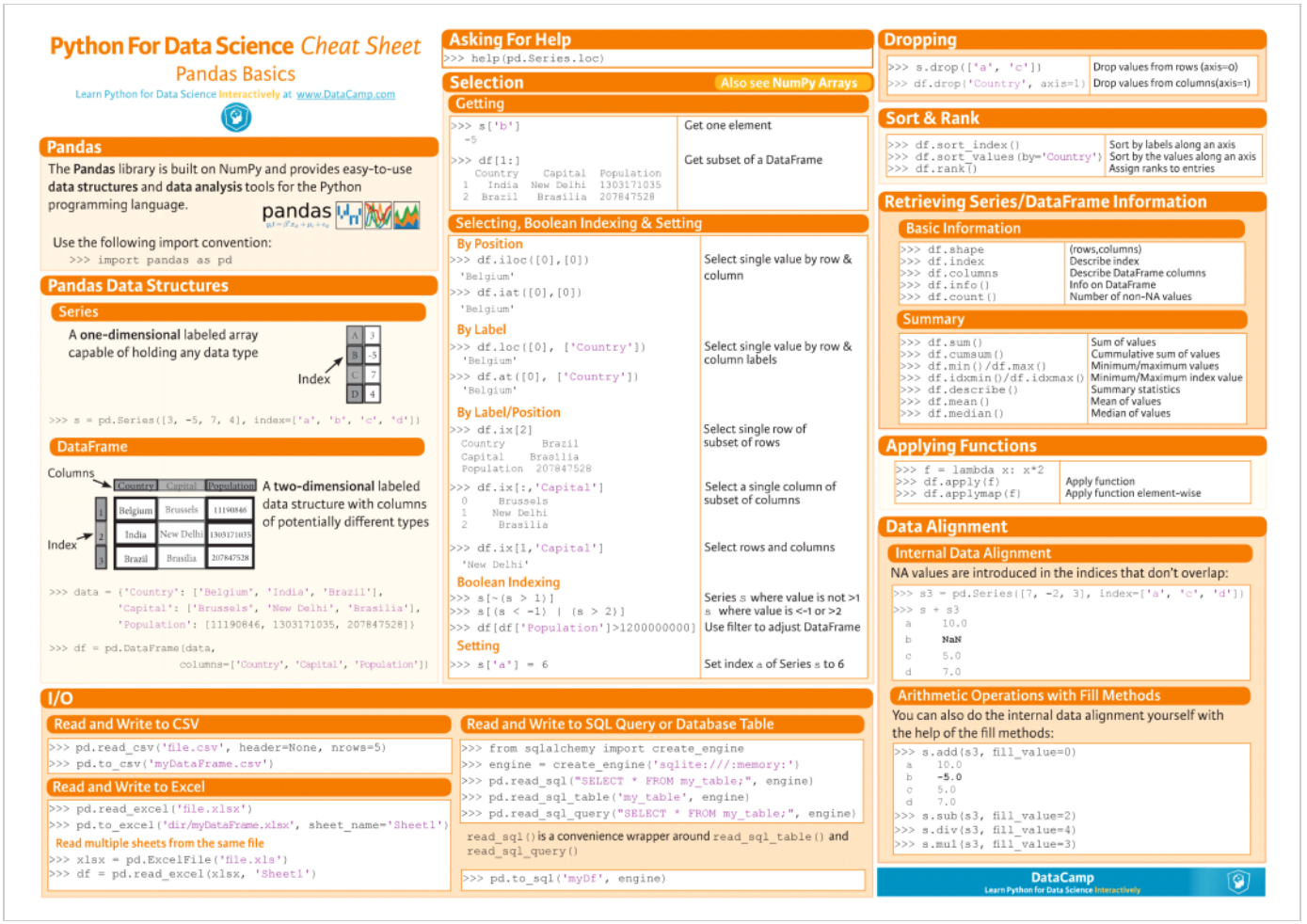
From <https://becominghuman.ai/cheat-sheets-for-ai-neural-networks-machine-learning-deep-learning-big-data-678c51b4b463>

Python-Pandas

April-12-17 5:42 PM

https://www.datacamp.com/community/blog/python-pandas-cheat-sheet#gs.ub0nwh8

- *NumPy, SciPy and Pandas:* s3.amazonaws.com/quandl-static-content/Documents/Quandl+-+Pandas,+S...



Python For Data Science *Cheat Sheet*

Pandas Basics

Learn Python for Data Science Interactively at www.DataCamp.com

Pandas

The **Pandas** library is built on NumPy and provides easy-to-use **data structures** and **data analysis** tools for the Python programming language.

pandas

Use the following import convention:

```
>>> import pandas as pd
```

Pandas Data Structures

Series

A **one-dimensional** labeled array capable of holding any data type

Index

```
>>> s = pd.Series([3, -5, 7, 4], index=['a', 'b', 'c', 'd'])
```

DataFrame

Columns

Index

A **two-dimensional** labeled data structure with columns of potentially different types

	Country	Capital	Population
1	Belgium	Brussels	11190846
2	India	New Delhi	1303171035
3	Brazil	Brasilia	207847528

```
>>> data = {'Country': ['Belgium', 'India', 'Brazil'],
            'Capital': ['Brussels', 'New Delhi', 'Brasilia'],
            'Population': [11190846, 1303171035, 207847528]}

>>> df = pd.DataFrame(data,
                      columns=['Country', 'Capital', 'Population'])
```

Asking For Help

```
>>> help(pd.Series.loc)
```

Selection

Also see NumPy Arrays

Getting

```
>>> s['b']
 -5
```
Get one element

```
>>> df[1:]
   Country    Capital  Population
1   India  New Delhi  1303171035
2  Brazil   Brasilia   207847528
```
Get subset of a DataFrame

Selecting, Boolean Indexing & Setting

By Position

```
>>> df.iloc[[0],[0]]
'Belgium'
>>> df.iat[[0],[0]]
'Belgium'
```
Select single value by row & column

By Label

```
>>> df.loc[[0], ['Country']]
'Belgium'
>>> df.at[[0], ['Country']]
'Belgium'
```
Select single value by row & column labels

By Label/Position

```
>>> df.ix[2]
Country      Brazil
Capital    Brasilia
Population  207847528
```
Select single row of subset of rows

```
>>> df.ix[:,'Capital']
0     Brussels
1    New Delhi
2     Brasilia
```
Select a single column of subset of columns

```
>>> df.ix[1,'Capital']
'New Delhi'
```
Select rows and columns

Boolean Indexing

```
>>> s[~(s > 1)]
>>> s[(s < -1) | (s > 2)]
```
Series `s` where value is not >1 `s` where value is <-1 or >2

```
>>> df[df['Population']>1200000000]
```
Use filter to adjust DataFrame

Setting

```
>>> s['a'] = 6
```
Set index `a` of Series `s` to 6

Dropping

```
>>> s.drop(['a', 'c'])
>>> df.drop('Country', axis=1)
```
Drop values from rows (axis=0)
Drop values from columns(axis=1)

Sort & Rank

```
>>> df.sort_index()
>>> df.sort_values(by='Country')
>>> df.rank()
```
Sort by labels along an axis
Sort by the values along an axis
Assign ranks to entries

Retrieving Series/DataFrame Information

Basic Information

```
>>> df.shape
>>> df.index
>>> df.columns
>>> df.info()
>>> df.count()
```
(rows,columns)
Describe index
Describe DataFrame columns
Info on DataFrame
Number of non-NA values

Summary

```
>>> df.sum()
>>> df.cumsum()
>>> df.min()/df.max()
>>> df.idxmin()/df.idxmax()
>>> df.describe()
>>> df.mean()
>>> df.median()
```
Sum of values
Cummulative sum of values
Minimum/maximum values
Minimum/Maximum index value
Summary statistics
Mean of values
Median of values

Applying Functions

```
>>> f = lambda x: x*2
>>> df.apply(f)
>>> df.applymap(f)
```
Apply function
Apply function element-wise

Data Alignment

Internal Data Alignment

NA values are introduced in the indices that don't overlap:

```
>>> s3 = pd.Series([7, -2, 3], index=['a', 'c', 'd'])
>>> s + s3
a    10.0
b     NaN
c     5.0
d     7.0
```

Arithmetic Operations with Fill Methods

You can also do the internal data alignment yourself with the help of the fill methods:

```
>>> s.add(s3, fill_value=0)
a    10.0
b    -5.0
c     5.0
d     7.0
>>> s.sub(s3, fill_value=2)
>>> s.div(s3, fill_value=4)
>>> s.mul(s3, fill_value=3)
```

I/O

Read and Write to CSV

```
>>> pd.read_csv('file.csv', header=None, nrows=5)
>>> pd.to_csv('myDataFrame.csv')
```

Read and Write to Excel

```
>>> pd.read_excel('file.xlsx')
>>> pd.to_excel('dir/myDataFrame.xlsx', sheet_name='Sheet1')
```

Read multiple sheets from the same file

```
>>> xlsx = pd.ExcelFile('file.xls')
>>> df = pd.read_excel(xlsx, 'Sheet1')
```

Read and Write to SQL Query or Database Table

```
>>> from sqlalchemy import create_engine
>>> engine = create_engine('sqlite:///:memory:')
>>> pd.read_sql("SELECT * FROM my_table;", engine)
>>> pd.read_sql_table('my_table', engine)
>>> pd.read_sql_query("SELECT * FROM my_table;", engine)
```

`read_sql()` is a convenience wrapper around `read_sql_table()` and `read_sql_query()`

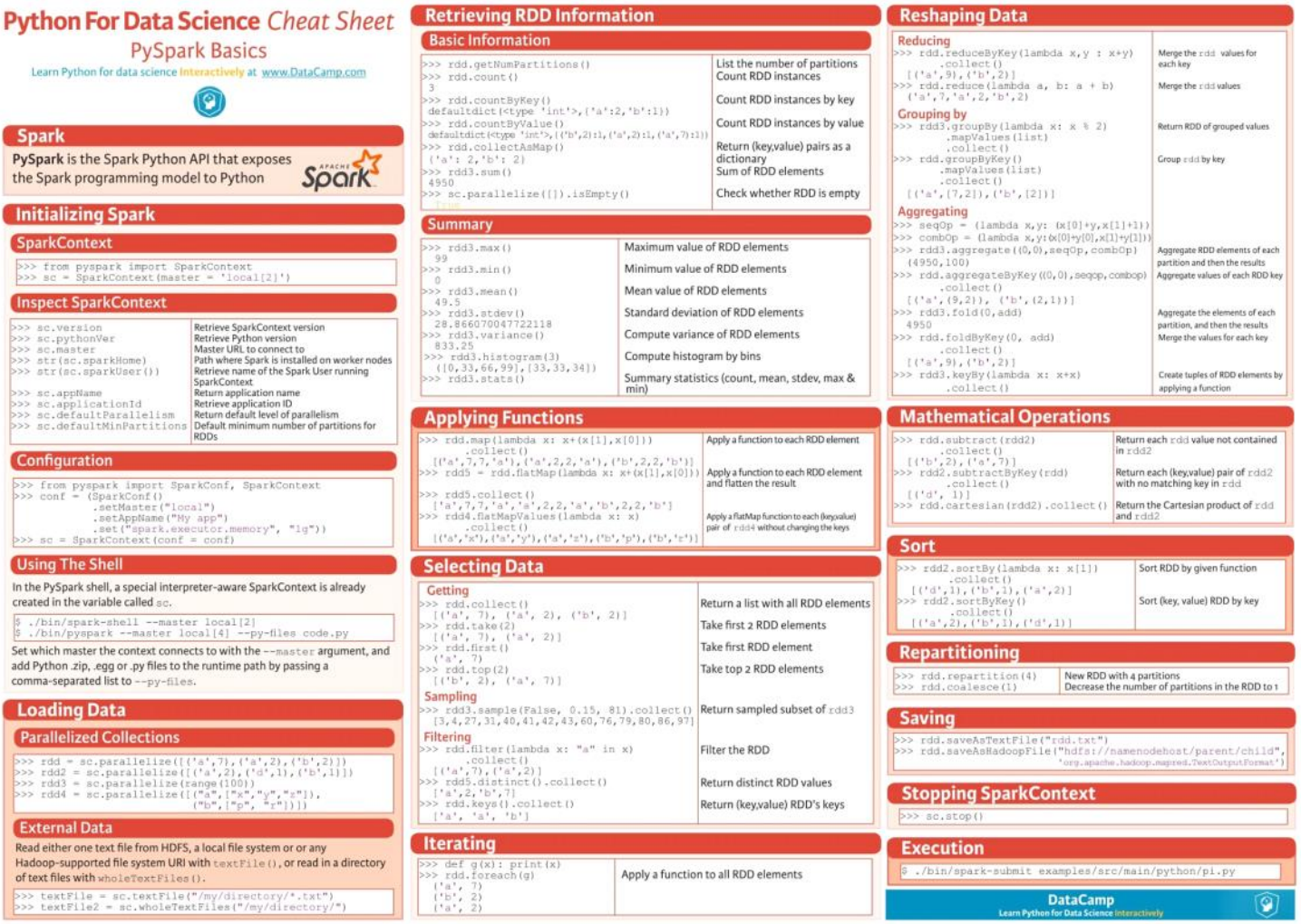
```
>>> pd.to_sql('myDf', engine)
```

DataCamp
Learn Python for Data Science Interactively

Python-pySpark

July-17-17 4:33 PM

pySpark



Pyspark Cheat Sheet: https://www.datacamp.com/community/blog/pyspark-cheat-sheet-python#gs.L=J1zxQ

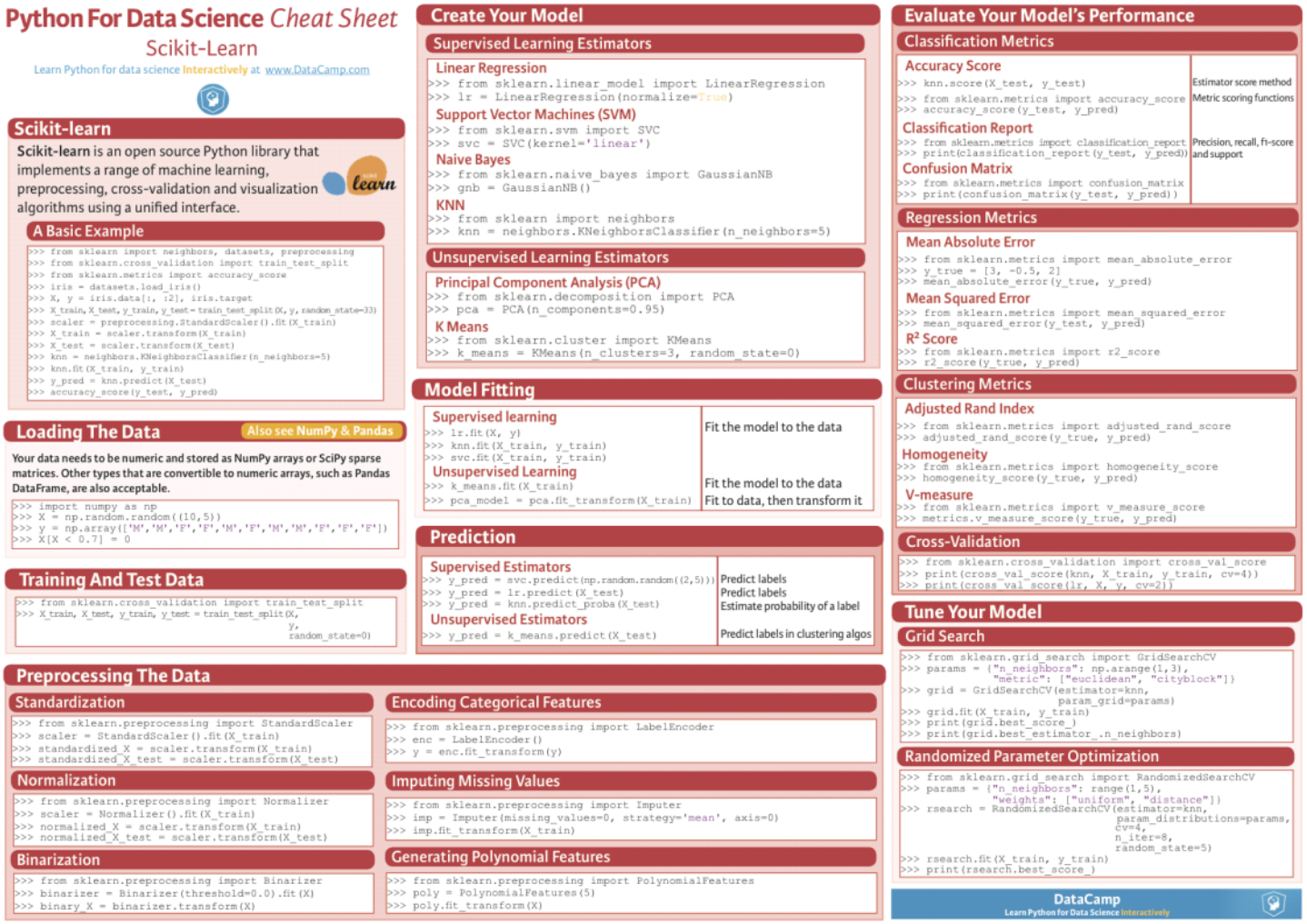
From <https://becominghuman.ai/cheat-sheets-for-ai-neural-networks-machine-learning-deep-learning-big-data-678c51b4b463>

Python-Scikit-Learn

July-17-17 3:59 PM

Scikit-Learn

Scikit-learn (formerly **scikits.learn**) is a free software machine learninglibrary for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, *k*-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.



Scikit Cheat Sheet: https://www.datacamp.com/community/blog/scikit-learn-cheat-sheet

Scikit-learn: https://en.wikipedia.org/wiki/Scikit-learn

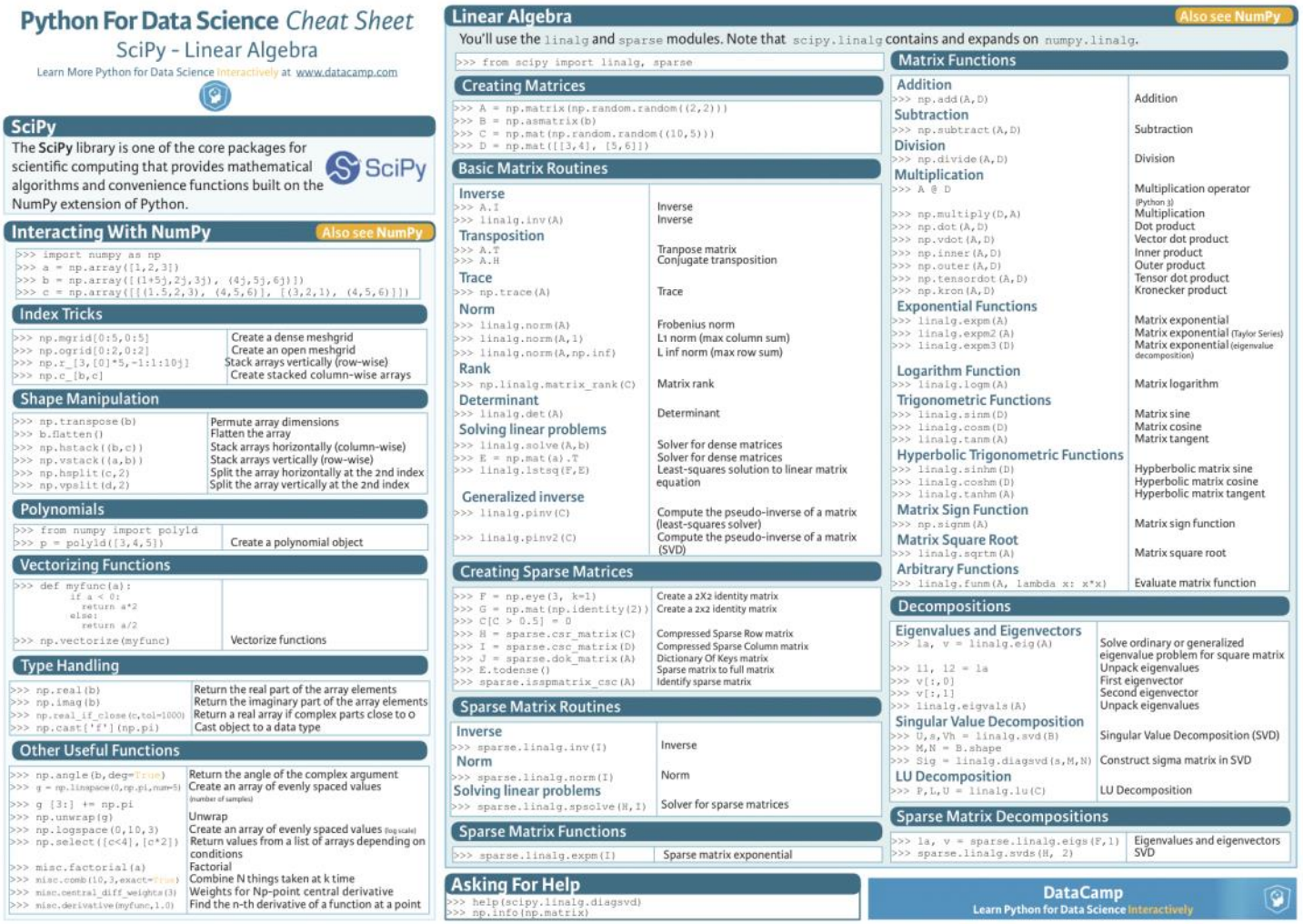
From <https://becominghuman.ai/cheat-sheets-for-ai-neural-networks-machine-learning-deep-learning-big-data-678c51b4b463>

Python-SciPy

July-17-17 4:30 PM

Scipy

SciPy builds on the NumPy array object and is part of the NumPy stack which includes tools like Matplotlib, pandas and SymPy, and an expanding set of scientific computing libraries. This NumPy stack has similar users to other applications such as MATLAB, GNU Octave, and Scilab. The NumPy stack is also sometimes referred to as the SciPy stack.[3]



Scipy Cheat Sheet: https://www.datacamp.com/community/blog/python-scipy-cheat-sheet#gs.JDSg3OI

SciPy: https://en.wikipedia.org/wiki/SciPy

From <https://becominghuman.ai/cheat-sheets-for-ai-neural-networks-machine-learning-deep-learning-big-data-678c51b4b463>

R

April-13-17 12:06 PM

- R Functions for Regression Analysis cran.r-project.org/doc/contrib/Ricci-refcard-regression.pdf
- Time Series cran.r-project.org/doc/contrib/Ricci-refcard-ts.pdf

https://cran.r-project.org/doc/contrib/Short-refcard.pdf

Short-refcard2

R Reference Card

by Tom Short, EPRI PEAC, tshort@epri-peac.com 2004-11-07 Granted to the public domain. See www.Rpad.org for the source and latest version. Includes material from *R for Beginners* by Emmanuel Paradis (with permission).

Getting help

Most R functions have online documentation.

`help(topic)` documentation on topic
`?topic` id.
`help.search("topic")` search the help system
`apropos("topic")` the names of all objects in the search list matching the regular expression "topic"
`help.start()` start the HTML version of help `str(a)` display the internal *str*ucture of an R object `summary(a)` gives a "summary" of a, usually a statistical summary but it is *generic* meaning it has different operations for different classes of a
`ls()` show objects in the search path; specify pat="pat" to search on a pattern
`ls.str()` str() for each variable in the search path `dir()` show files in the current directory `methods(a)` shows S3 methods of a
`methods(class=class(a))` lists all the methods to handle objects of class a

Input and output

`load()` load the datasets written with save `data(x)` loads specified data sets `library(x)` load add-on packages `read.table(file)` reads a file in table format and creates a data frame from it; the default separator sep="" is any whitespace; use header=TRUE to read the first line as a header of column names; use as.is=TRUE to prevent character vectors from being converted to factors; use comment.char="" to prevent "#" from being interpreted as a comment; use skip=n to skip n lines before reading data; see the help for options on row naming, NA treatment, and others
`read.csv("filename",header=TRUE)` id. but with defaults set for reading comma-delimited files
`read.delim("filename",header=TRUE)` id. but with defaults set for reading tab-delimited files
`read.fwf(file,widths,header=FALSE,sep="",as.is=FALSE)` read a table of *f*ixed width *f*ormatted data into a 'data.frame'; widths is an integer vector, giving the widths of the fixed-width fields
`save(file,...)` saves the specified objects (...) in the XDR platformindependent binary format
`save.image(file)` saves all objects `cat(..., file="", sep=" ")` prints the arguments after coercing to character; sep is the character separator between arguments
`print(a, ...)` prints its arguments; generic, meaning it can have different methods for different objects
`format(x,...)` format an R object for pretty printing
`write.table(x,file="",row.names=TRUE,col.names=TRUE, sep=" ")` prints x after converting to a data frame; if quote is TRUE, character or factor columns are surrounded by quotes ("); sep is the field separator; eol is the end-of-line separator; na is the string for missing values; use col.names=NA to add a blank column header to get the column headers aligned correctly for spreadsheet input
`sink(file)` output to file, until sink()

Most of the I/O functions have a file argument. This can often be a character string naming a file or a connection. file="" means the standard input or output. Connections can include files, pipes, zipped files, and R variables.

On windows, the file connection can also be used with description = "clipboard". To read a table copied from Excel, use x <- read.delim("clipboard") To write a table to the clipboard for Excel, use write.table(x,"clipboard",sep="\t",col.names=NA) For database interaction, see packages RODBC, DBI, RMySQL, RPgSQL, and ROracle. See packages XML, hdf5, netCDF for reading other file formats.

Data creation

`c(...)` generic function to combine arguments with the default forming a vector; with recursive=TRUE descends through lists combining all elements into one vector
`from:to` generates a sequence; ":" has operator priority; 1:4 + 1 is "2,3,4,5" `seq(from,to)` generates a sequence by= specifies increment; length= specifies desired length
`seq(along=x)` generates 1, 2, ..., length(along); useful for for loops
`rep(x,times)` replicate x times; use each= to repeat "each" element of x each times; rep(c(1,2,3),2) is 1 2 3 1 2 3; rep(c(1,2,3),each=2) is 1 1 2 2 3 3
`data.frame(...)` create a data frame of the named or unnamed arguments; data.frame(v=1:4,ch=c("a","B","c","d"),n=10); shorter vectors are recycled to the length of the longest
`list(...)` create a list of the named or unnamed arguments; list(a=c(1,2),b="hi",c=3i);
`array(x,dim=)` array with data x; specify dimensions like dim=c(3,4,2); elements of x recycle if x is not long enough
`matrix(x,nrow=,ncol=)` matrix; elements of x recycle
`factor(x,levels=)` encodes a vector x as a factor
`gl(n,k,length=n*k,labels=1:n)` generate levels (factors) by specifying the pattern of their levels; k is the number of levels, and n is the number of replications
`expand.grid()` a data frame from all combinations of the supplied vectors or factors
`rbind(...)` combine arguments by rows for matrices, data frames, and others
`cbind(...)` id. by columns

Slicing and extracting data

Indexing vectors

x[n]	n^{th} element
x[-n]	all *but* the n^{th} element
x[1:n]	first n elements
x[-(1:n)]	elements from n+1 to the end
x[c(1,4,2)]	specific elements
x["name"]	element named "name"
x[x > 3]	all elements greater than 3
x[x > 3 & x < 5]	all elements between 3 and 5
x[x %in% c("a","and","the")]	elements in the given set

Indexing lists

x[n]	list with elements n
x[[n]]	n^{th} element of the list
x[["name"]]	element of the list named "name"
x$name	id.

Indexing matrices

x[i,j]	element at row i, column j
x[i,]	row i
x[,j]	column j
x[,c(1,3)]	columns 1 and 3
x["name",]	row named "name"

Indexing data frames (matrix indexing plus the following) x[["name"]] column named "name" x$name id.

Variable conversion

`as.array(x)` `as.data.frame(x)` `as.numeric(x)`, `as.logical(x)` `as.complex(x)`, `as.character(x)`, ... convert type; for a complete list, use methods(as)

Variable information

`is.na(x)` `is.null(x)` `is.array(x)`, `is.data.frame(x)`, `is.numeric(x)` `is.complex(x)` `is.character(x)`, ... test for type; for a complete list, use methods(is)
`length(x)` number of elements in x `dim(x)` Retrieve or set the dimension of an object; dim(x) <- c(3,2) `dimnames(x)` Retrieve or set the dimension names of an object `nrow(x)` number of rows; NROW(x) is the same but treats a vector as a onerow matrix
`ncol(x)` and `NCOL(x)` id. for columns `class(x)` get or set the class of x; class(x) <- "myclass" `unclass(x)` remove the class attribute of x `attr(x,which)` get or set the attribute which of x `attributes(obj)` get or set the list of attributes of obj

Data selection and manipulation

`which.max(x)` returns the index of the greatest element of x
`which.min(x)` returns the index of the smallest element of x `rev(x)` reverses the elements of x `sort(x)` sorts the elements of x in increasing order; to sort in decreasing order: rev(sort(x))
`cut(x,breaks)` divides x into intervals (factors); breaks is the number of cut intervals or a vector of cut points
`match(x, y)` returns a vector of the same length than x with the elements of x which are in y (NA otherwise)
`which(x == a)` returns a vector of the indices of x if the comparison operation is true (TRUE), in this example the values of i for which x[i] == a (the argument of this function must be a variable of mode logical)

choose(n, k) computes the combinations of k events among n repetitions = $n!/[(n-k)!k!]$ **na.omit(x)** suppresses the observations with missing data (NA) (suppresses the corresponding line if x is a matrix or a data frame) **na.fail(x)** returns an error message if x contains at least one NA **unique(x)** if x is a vector or a data frame, returns a similar object but with the duplicate elements suppressed

table(x) returns a table with the numbers of the differents values of x (typically for integers or factors) **subset(x, ...)** returns a selection of x with respect to criteria (..., typically comparisons: x$V1 < 10); if x is a data frame, the option select gives the variables to be kept or dropped using a minus sign

sample(x, size) resample randomly and without replacement size elements in the vector x, the option replace = TRUE allows to resample with replacement

prop.table(x,margin=) table entries as fraction of marginal table

Math

sin,cos,tan,asin,acos,atan,atan2,log,log10,exp

max(x) maximum of the elements of x **min(x)** minimum of the elements of x **range(x)** id. then c(min(x), max(x)) **sum(x)** sum of the elements of x **diff(x)** lagged and iterated differences of vector x **prod(x)** product of the elements of x **mean(x)** mean of the elements of x **median(x)** median of the elements of x **quantile(x,probs=)** sample quantiles corresponding to the given probabilities (defaults to 0,.25,.5,.75,1)

weighted.mean(x, w) mean of x with weights w **rank(x)** ranks of the elements of x **var(x)** or cov(x) variance of the elements of x (calculated on $n-1$); if x is a matrix or a data frame, the variance-covariance matrix is calculated

sd(x) standard deviation of x

cor(x) correlation matrix of x if it is a matrix or a data frame (1 if x is a vector)

var(x, y) or cov(x, y) covariance between x and y, or between the columns of x and those of y if they are matrices or data frames

cor(x, y) linear correlation between x and y, or correlation matrix if they are matrices or data frames

round(x, n) rounds the elements of x to n decimals **log(x, base)** computes the logarithm of x with base base **scale(x)** if x is a matrix, centers and reduces the data; to center only use the option center=FALSE, to reduce only scale=FALSE (by default center=TRUE, scale=TRUE)

pmin(x,y,...) a vector which ith element is the minimum of x[i], y[i], ...

pmax(x,y,...) id. for the maximum **cumsum(x)** a vector which ith element is the sum from x[1] to x[i] **cumprod(x)** id. for the product **cummin(x)** id. for the minimum **cummax(x)** id. for the maximum

union(x,y), **intersect(x,y)**, **setdiff(x,y)**, **setequal(x,y)**, **is.element(el,set)** "set" functions

Re(x) real part of a complex number

Im(x) imaginary part

Mod(x) modulus; abs(x) is the same

Arg(x) angle in radians of the complex number

Conj(x) complex conjugate

convolve(x,y) compute the several kinds of convolutions of two sequences

fft(x) Fast Fourier Transform of an array **mvfft(x)** FFT of each column of a matrix **filter(x,filter)** applies linear filtering to a univariate time series or to each series separately of a multivariate time series Many math functions have a logical parameter na.rm=FALSE to specify missing data (NA) removal.

Matrices

t(x) transpose

diag(x) diagonal

%*% matrix multiplication

solve(a,b) solves a %*% x = b for x **solve(a)** matrix inverse of a

rowsum(x) sum of rows for a matrix-like object; **rowSums(x)** is a faster version

colsum(x), **colSums(x)** id. for columns

rowMeans(x) fast version of row means

colMeans(x) id. for columns

Advanced data processing

apply(X,INDEX,FUN=) a vector or array or list of values obtained by applying a function FUN to margins (INDEX) of X

lapply(X,FUN) apply FUN to each element of the list X **tapply(X,INDEX,FUN=)** apply FUN to each cell of a ragged array given by X with indexes INDEX

by(data,INDEX,FUN) apply FUN to data frame data subsetted by INDEX **merge(a,b)** merge two data frames by common columns or row names **xtabs(a b,data=x)** a contingency table from cross-classifying factors **aggregate(x,by,FUN)** splits the data frame x into subsets, computes summary statistics for each, and returns the result in a convenient form; by is a list of grouping elements, each as long as the variables in x

stack(x, ...) transform data available as separate columns in a data frame or list into a single column

unstack(x, ...) inverse of stack() **reshape(x, ...)** reshapes a data frame between 'wide' format with repeated measurements in separate columns of the same record and 'long' format with the repeated measurements in separate records; use (direction="wide") or (direction="long")

Strings

paste(...) concatenate vectors after converting to character; sep= is the string to separate terms (a single space is the default); collapse= is an optional string to separate "collapsed" results

substr(x,start,stop) substrings in a character vector; can also assign, as substr(x, start, stop) <- value

strsplit(x,split) split x according to the substring split **grep(pattern,x)** searches for matches to pattern within x; see ?regex **gsub(pattern,replacement,x)** replacement of matches determined by regular expression matching sub() is the same but only replaces the first occurrence. **tolower(x)** convert to lowercase **toupper(x)** convert to uppercase **match(x,table)** a vector of the positions of first matches for the elements of x among table

x %in% table id. but returns a logical vector **pmatch(x,table)** partial matches for the elements of x among table **nchar(x)** number of characters

Dates and Times

The class Date has dates without times. POSIXct has dates and times, including time zones. Comparisons (e.g. >), seq(), and difftime() are useful. Date also allows + and −. ?DateTimeClasses gives more information. See also package chron.

as.Date(s) and **as.POSIXct(s)** convert to the respective class; format(dt) converts to a string representation. The default string format is "2001-02-21". These accept a second argument to specify a format for conversion. Some common formats are:

%a, %A Abbreviated and full weekday name.
%b, %B Abbreviated and full month name.
%d Day of the month (01–31).
%H Hours (00–23).
%I Hours (01–12).
%j Day of year (001–366).
%m Month (01–12).
%M Minute (00–59).
%p AM/PM indicator.
%S Second as decimal number (00–61).
%U Week (00–53); the first Sunday as day 1 of week 1.
%w Weekday (0–6, Sunday is 0).
%W Week (00–53); the first Monday as day 1 of week 1.
%y Year without century (00–99). Don't use.
%Y Year with century.
%z (output only.) Offset from Greenwich; -0800 is 8 hours west of.
%Z (output only.) Time zone as a character string (empty if not available).

Where leading zeros are shown they will be used on output but are optional on input. See ?strftime.

Plotting

plot(x) plot of the values of x (on the y-axis) ordered on the x-axis

plot(x, y) bivariate plot of x (on the x-axis) and y (on the y-axis)

hist(x) histogram of the frequencies of x **barplot(x)** histogram of the values of x; use horiz=FALSE for horizontal bars

dotchart(x) if x is a data frame, plots a Cleveland dot plot (stacked plots line-by-line and column-by-column)

pie(x) circular pie-chart **boxplot(x)** "box-and-whiskers" plot **sunflowerplot(x, y)** id. than plot() but the points with similar coordinates are drawn as flowers which petal number represents the number of points

stripplot(x) plot of the values of x on a line (an alternative to boxplot() for small sample sizes)

coplot(x~y | z) bivariate plot of x and y for each value or interval of values of z

interaction.plot (f1, f2, y) if f1 and f2 are factors, plots the means of y (on the *y*-axis) with respect to the values of f1 (on the *x*-axis) and of f2 (different curves); the option fun allows to choose the summary statistic of y (by default fun=mean)

matplot(x,y) bivariate plot of the first column of x *vs.* the first one of y, the second one of x *vs.* the second one of y, etc.

fourfoldplot(x) visualizes, with quarters of circles, the association between two dichotomous variables for different populations (x must be an array with dim=c(2, 2, k), or a matrix with dim=c(2, 2) if $k = 1$)

assocplot(x) Cohen–Friendly graph showing the deviations from independence of rows and columns in a two dimensional contingency table

mosaicplot(x) 'mosaic' graph of the residuals from a log-linear regression of a contingency table

pairs(x) if x is a matrix or a data frame, draws all possible bivariate plots between the columns of x

plot.ts(x) if x is an object of class "ts", plot of x with respect to time, x may be multivariate but the series must have the same frequency and dates

ts.plot(x) id. but if x is multivariate the series may have different dates and must have the same frequency

qqnorm(x) quantiles of x with respect to the values expected under a normal law

qqplot(x, y) quantiles of y with respect to the quantiles of x

contour(x, y, z) contour plot (data are interpolated to draw the curves), x and y must be vectors and z must be a matrix so that dim(z)=c(length(x), length(y)) (x and y may be omitted)

filled.contour(x, y, z) id. but the areas between the contours are coloured, and a legend of the colours is drawn as well

image(x, y, z) id. but with colours (actual data are plotted)

persp(x, y, z) id. but in perspective (actual data are plotted)

stars(x) if x is a matrix or a data frame, draws a graph with segments or a star where each row of x is represented by a star and the columns are the lengths of the segments

symbols(x, y, ...) draws, at the coordinates given by x and y, symbols (circles, squares, rectangles, stars, thermometres or "boxplots") which sizes, colours ... are specified by supplementary arguments

termplot(mod.obj) plot of the (partial) effects of a regression model (mod.obj)

The following parameters are common to many plotting functions:

add=FALSE if TRUE superposes the plot on the previous one (if it exists)

axes=TRUE if FALSE does not draw the axes and the box **type="p"** specifies the type of plot, "p": points, "l": lines, "b": points connected by lines, "o": id. but the lines are over the points, "h": vertical lines, "s": steps, the data are represented by the top of the vertical lines, "S": id. but the data are represented by the bottom of the vertical lines

xlim=, ylim= specifies the lower and upper limits of the axes, for example with xlim=c(1, 10) or xlim=range(x)

xlab=, ylab= annotates the axes, must be variables of mode character **main=** main title, must be a variable of mode character **sub=** sub-title (written in a smaller font)

Low-level plotting commands

points(x, y) adds points (the option type= can be used) **lines(x, y)** id. but with lines **text(x, y, labels, ...)** adds text given by labels at coordinates (x,y); a typical use is: plot(x, y, type="n"); text(x, y, names)

mtext(text, side=3, line=0, ...) adds text given by text in the margin specified by side (see axis() below); line specifies the line from the plotting area

segments(x0, y0, x1, y1) draws lines from points (x0,y0) to points (x1.y1)

arrows(x0, y0, x1, y1, angle= 30, code=2) id. with arrows at points (x0,y0) if code=2, at points (x1,y1) if code=1, or both if code=3; angle controls the angle from the shaft of the arrow to the edge of the arrow head

abline(a,b) draws a line of slope b and intercept a **abline(h=y)** draws a horizontal line at ordinate y **abline(v=x)** draws a vertical line at abcissa x **abline(lm.obj)** draws the regression line given by lm.obj

rect(x1, y1, x2, y2) draws a rectangle which left, right, bottom, and top limits are x1, x2, y1, and y2, respectively

polygon(x, y) draws a polygon linking the points with coordinates given by x and y

legend(x, y, legend) adds the legend at the point (x,y) with the symbols given by legend

title() adds a title and optionally a sub-title **axis(side, vect)** adds an axis at the bottom (side=1), on the left (2), at the top (3), or on the right (4); vect (optional) gives the abcissa (or ordinates) where tick-marks are drawn

rug(x) draws the data x on the *x*-axis as small vertical lines **locator(n, type="n", ...)** returns the coordinates (*x*,*y*) after the user has clicked n times on the plot with the mouse; also draws symbols (type="p") or lines (type="l") with respect to optional graphic parameters (...); by default nothing is drawn (type="n")

Graphical parameters

These can be set globally with **par(...)**; many can be passed as parameters to plotting commands.

adj controls text justification (0 left-justified, 0.5 centred, 1 right-justified)

bg specifies the colour of the background (ex. : bg="red", bg="blue", ... the list of the 657 available colours is displayed with colors())

bty controls the type of box drawn around the plot, allowed values are: "o", "l", "7", "c", "u" ou "]" (the box looks like the corresponding character); if bty="n" the box is not drawn

cex a value controlling the size of texts and symbols with respect to the default; the following parameters have the same control for numbers on the axes, cex.axis, the axis labels, cex.lab, the title, cex.main, and the sub-title, cex.sub

col controls the color of symbols and lines; use color names: "red", "blue" see colors() or as "#RRGGBB"; see rgb(), hsv(), gray(), and rainbow(), as for cex there are: col.axis, col.lab, col.main, col.sub

font an integer which controls the style of text (1: normal, 2: italics, 3: bold, 4: bold italics); as for cex there are: font.axis, font.lab, font.main, font.sub

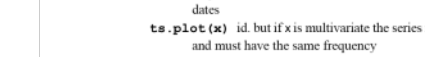
las an integer which controls the orientation of the axis labels (0: parallel to the axes, 1: horizontal, 2: perpendicular to the axes, 3: vertical)

lty controls the type of lines, can be an integer or string (1: "solid", 2: "dashed", 3: "dotted", 4: "dotdash", 5: "longdash", 6: "twodash", or a string of up to eight characters (between "0" and "9") which specifies alternatively the length, in points or pixels, of the drawn elements and the blanks, for example lty="44" will have the same effect than lty=2

lwd a numeric which controls the width of lines, default 1 **mar** a vector of 4 numeric values which control the space between the axes and the border of the graph of the form c(bottom, left, top, right), the default values are c(5.1, 4.1, 4.1, 2.1)

mfcol a vector of the form c(nr,nc) which partitions the graphic window as a matrix of nr lines and nc columns, the plots are then drawn in columns

mfrow id. but the plots are drawn by row **pch** controls the type of symbol, either an integer between 1 and 25, or any single character within ""



ps an integer which controls the size in points of texts and symbols

pty a character which specifies the type of the plotting region, "s": square, "m": maximal **tck** a value which specifies the length of tick-marks on the axes as a fraction of the smallest of the width or height of the plot; if tck=1 a grid is drawn

tcl a value which specifies the length of tick-marks on the axes as a fraction of the height of a line of text (by default tcl=-0.5)

xaxt if xaxt="n" the *x*-axis is set but not drawn (useful in conjonction with axis(side=1, ...))

yaxt if yaxt="n" the *y*-axis is set but not drawn (useful in conjonction with axis(side=2, ...))

Lattice (Trellis) graphics

xyplot(y~x) bivariate plots (with many functionalities)

barchart(y~x) histogram of the values of y with respect to those of x

dotplot(y~x) Cleveland dot plot (stacked plots line-by-line and columnby-column)

densityplot(~x) density functions plot **histogram(~x)** histogram of the frequencies of x **bwplot(y~x)** "box-and-whiskers" plot **qqmath(~x)** quantiles of x with respect to the values expected under a theoretical distribution

stripplot(y~x) single dimension plot, x must be numeric, y may be a factor

qq(y~x) quantiles to compare two distributions, x must be numeric, y may be numeric, character, or factor but must have two 'levels'

splom(~x) matrix of bivariate plots **parallel(~x)** parallel coordinates plot **levelplot(z~x*y|g1*g2)** coloured plot of the

values of z at the coordinates given by x and y (x, y and z are all of the same length)

wireframe(z~x*y|g1*g2) 3d surface plot

cloud(z~x*y|g1*g2) 3d scatter plot

In the normal Lattice formula, y x|g1*g2 has combinations of optional conditioning variables g1 and g2 plotted on separate panels. Lattice functions take many of the same arguments as base graphics plus also data= the data frame for the formula variables and subset= for subsetting. Use panel= to define a custom panel function (see apropos("panel") and ?llines). Lattice functions return an object of class trellis and have to be print-ed to produce the graph. Use print(xyplot(...)) inside functions where automatic printing doesn't work. Use lattice.theme and lset to change Lattice defaults.

Optimization and model fitting

optim(par, fn, method = c("Nelder-Mead", "BFGS", "CG", "L-BFGS-B", "SANN") general-purpose optimization; par is initial values, fn is function to optimize (normally minimize)

nlm(f,p) minimize function f using a Newton-type algorithm with starting values p

lm(formula) fit linear models; formula is typically of the form response termA + termB + ...; use I(x*y) + I(x^2) for terms made of nonlinear components

glm(formula,family=) fit generalized linear models, specified by giving a symbolic description of the linear predictor and a description of the error distribution; family is a description of the error distribution and link function to be used in the model; see ?family

nls(formula) nonlinear least-squares estimates of the nonlinear model parameters

approx(x,y=) linearly interpolate given data points; x can be an xy plotting structure

spline(x,y=) cubic spline interpolation **loess(formula)** fit a polynomial surface using local fitting Many of the formula-based modeling functions have several common arguments: data= the data frame for the formula variables, subset= a subset of variables used in the fit, na.action= action for missing values: "na.fail", "na.omit", or a function. The following generics often apply to model fitting functions:

predict(fit,...) predictions from fit based on input data

df.residual(fit) returns the number of residual degrees of freedom

coef(fit) returns the estimated coefficients (sometimes with their standard-errors)

residuals(fit) returns the residuals **deviance(fit)** returns the deviance **fitted(fit)** returns the fitted values **logLik(fit)** computes the logarithm of the likelihood and the number of parameters

AIC(fit) computes the Akaike information criterion or AIC

Statistics

aov(formula) analysis of variance model **anova(fit,...)** analysis of variance (or deviance) tables for one or more fitted model objects

density(x) kernel density estimates of x

binom.test(), **pairwise.t.test()**, **power.t.test()**, **prop.test()**, **t.test()**, ... use help.search("test")

Distributions

rnorm(n, mean=0, sd=1) Gaussian (normal) **rexp(n, rate=1)** exponential **rgamma(n, shape, scale=1)** gamma **rpois(n, lambda)** Poisson **rweibull(n, shape, scale=1)** Weibull **rcauchy(n, location=0, scale=1)** Cauchy **rbeta(n, shape1, shape2)** beta **rt(n, df)** 'Student' (*t*) **rf(n, df1, df2)** Fisher–Snedecor (*F*) (χ²) **rchisq(n, df)** Pearson **rbinom(n, size, prob)** binomial **rgeom(n, prob)** geometric **rhyper(nn, m, n, k)** hypergeometric **rlogis(n, location=0, scale=1)** logistic **rlnorm(n, meanlog=0, sdlog=1)** lognormal **rnbinom(n, size, prob)** negative binomial **runif(n, min=0, max=1)** uniform **rwilcox(nn, m, n)**, rsignrank(nn, n) Wilcoxon's statistics All these functions can be used by replacing the letter r with d, p or q to get, respectively, the probability density (d*func*(x, ...)), the cumulative probability density (p*func*(x, ...)), and the value of quantile (q*func*(p, ...), with 0 < p < 1).

Programming

function(arglist) expr function definition **return(value) if(cond) expr if(cond) cons.expr else alt.expr for(var in seq) expr while(cond) expr repeat expr break next**

Use braces {} around statements **ifelse(test, yes, no)** a value with the same shape as test filled with elements from either yes or no

do.call(funname, args) executes a function call from the name of the function and a list of arguments to be passed to it

YanchangZhao-refcard-data-mining Data Mining cran.r-project.org/doc/contrib/YanchangZhao-refcard-data-mining.pdf

R Reference Card for Data Mining

by Yanchang Zhao, yanchang@rdatamining.com, January 3, 2013
The latest version is available at http://www.RDataMining.com. Click the link also for document *R and Data Mining: Examples and Case Studies*.
The package names are in parentheses.

Association Rules & Frequent Itemsets

APRIORI Algorithm

a level-wise, breadth-first algorithm which counts transactions to find frequent itemsets
`apriori()` mine associations with APRIORI algorithm (*arules*)

ECLAT Algorithm

employs equivalence classes, depth-first search and set intersection instead of counting
`eclat()` mine frequent itemsets with the Eclat algorithm (*arules*)

Packages

arules mine frequent itemsets, maximal frequent itemsets, closed frequent itemsets and association rules. It includes two algorithms, Apriori and Eclat.
arulesViz visualizing association rules

Sequential Patterns

Functions

`cspade()` mining frequent sequential patterns with the cSPADE algorithm (*arulesSequences*)
`seqefsub()` searching for frequent subsequences (*TraMineR*)

Packages

arulesSequences add-on for *arules* to handle and mine frequent sequences
TraMineR mining, describing and visualizing sequences of states or events

Classification & Prediction

Decision Trees

`ctree()` conditional inference trees, recursive partitioning for continuous, censored, ordered, nominal and multivariate response variables in a conditional inference framework (*party*)
`rpart()` recursive partitioning and regression trees (*rpart*)
`mob()` model-based recursive partitioning, yielding a tree with fitted models associated with each terminal node (*party*)

Random Forest

`cforest()` random forest and bagging ensemble (*party*)
`randomForest()` random forest (*randomForest*)
`varimp()` variable importance (*party*)
`importance()` variable importance (*randomForest*)

Neural Networks

`nnet()` fit single-hidden-layer neural network (*nnet*)

Support Vector Machine (SVM)

`svm()` train a support vector machine for regression, classification or density-estimation (*e1071*)
`ksvm()` support vector machines (*kernlab*)

Performance Evaluation

`performance()` provide various measures for evaluating performance of prediction and classification models (*ROCR*)
`roc()` build a ROC curve (*pROC*)
`auc()` compute the area under the ROC curve (*pROC*)
`ROC()` draw a ROC curve (*DiagnosisMed*)
`PRcurve()` precision-recall curves (*DMwR*)
`CRchart()` cumulative recall charts (*DMwR*)

Packages

rpart recursive partitioning and regression trees
party recursive partitioning
randomForest classification and regression based on a forest of trees using random inputs
rpartOrdinal ordinal classification trees, deriving a classification tree when the response to be predicted is ordinal
rpart.plot plots rpart models with an enhanced version of `plot.rpart` in the *rpart* package
ROCR visualize the performance of scoring classifiers
pROC display and analyze ROC curves

Regression

Functions

`lm()` linear regression
`glm()` generalized linear regression
`nls()` non-linear regression
`predict()` predict with models
`residuals()` residuals, the difference between observed values and fitted values
`gls()` fit a linear model using generalized least squares (*nlme*)
`gnls()` fit a nonlinear model using generalized least squares (*nlme*)

Packages

nlme linear and nonlinear mixed effects models

Clustering

Partitioning based Clustering

partition the data into k groups first and then try to improve the quality of clustering by moving objects from one group to another
`kmeans()` perform k-means clustering on a data matrix
`kmeansCBI()` interface function for `kmeans` (*fpc*)
`kmeansruns()` call `kmeans` for the k-means clustering method and includes estimation of the number of clusters and finding an optimal solution from several starting points (*fpc*)
`pam()` the Partitioning Around Medoids (PAM) clustering method (*cluster*)
`pamk()` the Partitioning Around Medoids (PAM) clustering method with estimation of number of clusters (*fpc*)
`cluster.optimal()` search for the optimal k-clustering of the dataset (*bayesclust*)
`clara()` Clustering Large Applications (*cluster*)
`fanny(x,k,...)` compute a fuzzy clustering of the data into `k` clusters (*cluster*)
`kcca()` k-centroids clustering (*flexclust*)
`ccfkms()` clustering with Conjugate Convex Functions (*cba*)
`apcluster()` affinity propagation clustering for a given similarity matrix (*apcluster*)
`apclusterK()` affinity propagation clustering to get K clusters (*apcluster*)
`cclust()` Convex Clustering, incl. k-means and two other clustering algorithms (*cclust*)
`KMeansSparseCluster()` sparse k-means clustering (*sparcl*)
`tclust(x,k,alpha,...)` trimmed k-means with which a proportion `alpha` of observations may be trimmed (*tclust*)

Hierarchical Clustering

a hierarchical decomposition of data in either bottom-up (agglomerative) or top-down (divisive) way
`hclust(d, method, ...)` hierarchical cluster analysis on a set of dissimilarities `d` using the `method` for agglomeration
`birch()` the BIRCH algorithm that clusters very large data with a CF-tree (*birch*)
`pvclust()` hierarchical clustering with p-values via multi-scale bootstrap resampling (*pvclust*)
`agnes()` agglomerative hierarchical clustering (*cluster*)
`diana()` divisive hierarchical clustering (*cluster*)
`mona()` divisive hierarchical clustering of a dataset with binary variables only (*cluster*)
`rockCluster()` cluster a data matrix using the Rock algorithm (*cba*)
`proximus()` cluster the rows of a logical matrix using the Proximus algorithm (*cba*)
`isopam()` Isopam clustering algorithm (*isopam*)
`LLAhclust()` hierarchical clustering based on likelihood linkage analysis (*LLAhclust*)
`flashClust()` optimal hierarchical clustering (*flashClust*)
`fastcluster()` fast hierarchical clustering (*fastcluster*)
`cutreeDynamic()`, `cutreeHybrid()` detection of clusters in hierarchical clustering dendrograms (*dynamicTreeCut*)
`HierarchicalSparseCluster()` hierarchical sparse clustering (*sparcl*)

Model based Clustering

`Mclust()` model-based clustering (*mclust*)
`HDDC()` a model-based method for high dimensional data clustering (*HDclassif*)
`fixmahal()` Mahalanobis Fixed Point Clustering (*fpc*)
`fixreg()` Regression Fixed Point Clustering (*fpc*)
`mergenormals()` clustering by merging Gaussian mixture components (*fpc*)

Density based Clustering

generate clusters by connecting dense regions
`dbscan(data,eps,MinPts,...)` generate a density based clustering of arbitrary shapes, with neighborhood radius set as `eps` and density threshold as `MinPts` (*fpc*)
`pdfCluster()` clustering via kernel density estimation (*pdfCluster*)

Other Clustering Techniques

`mixer()` random graph clustering (*mixer*)
`nncluster()` fast clustering with restarted minimum spanning tree (*nnclust*)
`orclus()` ORCLUS subspace clustering (*orclus*)

Plotting Clustering Solutions

`plotcluster()` visualisation of a clustering or grouping in data (*fpc*)
`bannerplot()` a horizontal barplot visualizing a hierarchical clustering (*cluster*)

RDM

Cluster Validation

`silhouette()` compute or extract silhouette information (*cluster*)
`cluster.stats()` compute several cluster validity statistics from a clustering and a dissimilarity matrix (*fpc*)
`clValid()` calculate validation measures for a given set of clustering algorithms and number of clusters (*clValid*)
`clustIndex()` calculate the values of several clustering indexes, which can be independently used to determine the number of clusters existing in a data set (*cclust*)
`NbClust()` provide 30 indices for cluster validation and determining the number of clusters (*NbClust*)

Packages

cluster cluster analysis
fpc various methods for clustering and cluster validation
mclust model-based clustering and normal mixture modeling
birch clustering very large datasets using the BIRCH algorithm
pvclust hierarchical clustering with p-values
apcluster Affinity Propagation Clustering
cclust Convex Clustering methods, including k-means algorithm, On-line Update algorithm and Neural Gas algorithm and calculation of indexes for finding the number of clusters in a data set
cba Clustering for Business Analytics, including clustering techniques such as Proximus and Rock
bclust Bayesian clustering using spike-and-slab hierarchical model, suitable for clustering high-dimensional data
biclust algorithms to find bi-clusters in two-dimensional data
clue cluster ensembles
clues clustering method based on local shrinking
clValid validation of clustering results
clv cluster validation techniques, contains popular internal and external cluster validation methods for outputs produced by package *cluster*
bayesclust tests/searches for significant clusters in genetic data
clustvarsel variable selection for model-based clustering
clustsig significant cluster analysis, tests to see which (if any) clusters are statistically different
clusterfly explore clustering interactively
clusterSim search for optimal clustering procedure for a data set
clusterGeneration random cluster generation
clusterCons calculate the consensus clustering result from re-sampled clustering experiments with the option of using multiple algorithms and parameter
gcExplorer graphical cluster explorer
hybridHclust hybrid hierarchical clustering via mutual clusters
Modalclust hierarchical modal Clustering
iCluster integrative clustering of multiple genomic data types
EMCC evolutionary Monte Carlo (EMC) methods for clustering
rEMM extensible Markov Model (EMM) for data stream clustering

Outlier Detection

Functions

`boxplot.stats()$out` list data points lying beyond the extremes of the whiskers
`lofactor()` calculate local outlier factors using the LOF algorithm (*DMwR* or *dprep*)
`lof()` a parallel implementation of the LOF algorithm (*Rlof*)

Packages

extremevalues detect extreme values in one-dimensional data
mvoutlier multivariate outlier detection based on robust methods
outliers some tests commonly used for identifying outliers
Rlof a parallel implementation of the LOF algorithm

Time Series Analysis

Construction & Plot

`ts()` create time-series objects (*stats*)
`plot.ts()` plot time-series objects (*stats*)
`smoothts()` time series smoothing (*ast*)
`sfilter()` remove seasonal fluctuation using moving average (*ast*)

Decomposition

`decomp()` time series decomposition by square-root filter (*timsac*)
`decompose()` classical seasonal decomposition by moving averages (*stats*)
`stl()` seasonal decomposition of time series by `loess` (*stats*)
`tsr()` time series decomposition (*ast*)
`ardec()` time series autoregressive decomposition (*ArDec*)

Forecasting

`arima()` fit an ARIMA model to a univariate time series (*stats*)
`predict.Arima()` forecast from models fitted by `arima` (*stats*)
`auto.arima()` fit best ARIMA model to univariate time series (*forecast*)
`forecast.stl(), forecast.ets(), forecast.Arima()` forecast time series using `stl`, `ets` and `arima` models (*forecast*)

Packages

forecast displaying and analysing univariate time series forecasts
timsac time series analysis and control program
ast time series analysis
ArDec time series autoregressive-based decomposition
ares a toolbox for time series analyses using generalized additive models
dse tools for multivariate, linear, time-invariant, time series models

Text Mining

Functions

`Corpus()` build a corpus, which is a collection of text documents (*tm*)
`tm_map()` transform text documents, e.g., stemming, stopword removal (*tm*)
`tm_filter()` filtering out documents (*tm*)
`TermDocumentMatrix(), DocumentTermMatrix()` construct a term-document matrix or a document-term matrix (*tm*)
`Dictionary()` construct a dictionary from a character vector or a term-document matrix (*tm*)
`findAssocs()` find associations in a term-document matrix (*tm*)
`findFreqTerms()` find frequent terms in a term-document matrix (*tm*)
`stemDocument()` stem words in a text document (*tm*)
`stemCompletion()` complete stemmed words (*tm*)
`termFreq()` generate a term frequency vector from a text document (*tm*)
`stopwords(language)` return stopwords in different languages (*tm*)
`removeNumbers(), removePunctuation(), removeWords()` remove numbers, punctuation marks, or a set of words from a text document (*tm*)
`removeSparseTerms()` remove sparse terms from a term-document matrix (*tm*)
`textcat()` n-gram based text categorization (*textcat*)
`SnowballStemmer()` Snowball word stemmers (*Snowball*)
`LDA()` fit a LDA (latent Dirichlet allocation) model (*topicmodels*)
`CTM()` fit a CTM (correlated topics model) model (*topicmodels*)
`terms()` extract the most likely terms for each topic (*topicmodels*)
`topics()` extract the most likely topics for each document (*topicmodels*)

Packages

tm a framework for text mining applications
lda fit topic models with LDA
topicmodels fit topic models with LDA and CTM
RTextTools automatic text classification via supervised learning
tm.plugin.dc a plug-in for package *tm* to support distributed text mining
tm.plugin.mail a plug-in for package *tm* to handle mail
RcmdrPlugin.TextMining GUI for demonstration of text mining concepts and *tm* package
textir a suite of tools for inference about text documents and associated sentiment
tau utilities for text analysis
textcat n-gram based text categorization
Yjdn.Jlp Japanese text analysis by Yahoo! Japan Developer Network

Social Network Analysis and Graph Mining

Functions

`graph(), graph.edgelist(), graph.adjacency(), graph.incidence()` create graph objects respectively from edges, an edge list, an adjacency matrix and an incidence matrix (*igraph*)
`plot(), tkplot()` static and interactive plotting of graphs (*igraph*)
`gplot(), gplot3d()` plot graphs (*sna*)
`V(), E()` vertex/edge sequence of igraph (*igraph*)
`are.connected()` check whether two nodes are connected (*igraph*)
`degree(), betweenness(), closeness()` various centrality scores (*igraph, sna*)
`add.edges(), add.vertices(), delete.edges(), delete.vertices()` add and delete edges and vertices (*igraph*)
`neighborhood()` neighborhood of graph vertices (*igraph, sna*)
`get.adjlist()` adjacency lists for edges or vertices (*igraph*)
`nei(), adj(), from(), to()` vertex/edge sequence indexing (*igraph*)
`cliques()` find cliques, ie. complete subgraphs (*igraph*)
`clusters()` maximal connected components of a graph (*igraph*)
`%->%, %<-%, %--%` edge sequence indexing (*igraph*)
`get.edgelist()` return an edge list in a two-column matrix (*igraph*)
`read.graph(), write.graph()` read and writ graphs from and to files (*igraph*)

Packages

sna social network analysis
igraph network analysis and visualization
statnet a set of tools for the representation, visualization, analysis and simulation of network data
egonet ego-centric measures in social network analysis
snort social network-analysis on relational tables
network tools to create and modify network objects
bipartite visualising bipartite networks and calculating some (ecological) indices
blockmodeling generalized and classical blockmodeling of valued networks
diagram visualising simple graphs (networks), plotting flow diagrams
NetCluster clustering for networks
NetData network data for McFarland's SNA R labs

NetIndices estimating network indices, including trophic structure of foodwebs in R
NetworkAnalysis statistical inference on populations of weighted or unweighted networks
tnet analysis of weighted, two-mode, and longitudinal networks
triads triad census for networks

Spatial Data Analysis

Functions

`geocode()` geocodes a location using Google Maps (*ggmap*)
`qmap()` quick map plot (*ggmap*)
`get_map()` queries the Google Maps, OpenStreetMap, or Stamen Maps server for a map at a certain location (*ggmap*)
`gvisGeoChart()`, `gvisGeoMap()`, `gvisIntensityMap()`, `gvisMap()` Google geo charts and maps (*googleVis*)
`GetMap()` download a static map from the Google server (*RgoogleMaps*)
`ColorMap()` plot levels of a variable in a colour-coded map (*RgoogleMaps*)
`PlotOnStaticMap()` overlay plot on background image of map tile (*RgoogleMaps*)
`TextOnStaticMap()` plot text on map (*RgoogleMaps*)

Packages

plotGoogleMaps plot spatial data as HTML map mushup over Google Maps
RgoogleMaps overlay on Google map tiles in R
plotKML visualization of spatial and spatio-temporal objects in Google Earth
ggmap Spatial visualization with Google Maps and OpenStreetMap
clustTool GUI for clustering data with spatial information
SGCS Spatial Graph based Clustering Summaries for spatial point patterns
spdep spatial dependence: weighting schemes, statistics and models

Statistics

Summarization

`summary()` summarize data
`describe()` concise statistical description of data (*Hmisc*)
`boxplot.stats()` box plot statistics

Analysis of Variance

`aov()` fit an analysis of variance model (*stats*)
`anova()` compute analysis of variance (or deviance) tables for one or more fitted model objects (*stats*)

Statistical Test

`t.test()` student's t-test (*stats*)
`prop.test()` test of equal or given proportions (*stats*)
`binom.test()` exact binomial test (*stats*)

Mixed Effects Models

`lme()` fit a linear mixed-effects model (*nlme*)
`nlme()` fit a nonlinear mixed-effects model (*nlme*)

Principal Components and Factor Analysis

`princomp()` principal components analysis (*stats*)
`prcomp()` principal components analysis (*stats*)

Other Functions

`var()`, `cov()`, `cor()` variance, covariance, and correlation (*stats*)
`density()` compute kernel density estimates (*stats*)

Packages

nlme linear and nonlinear mixed effects models

Graphics

Functions

`plot()` generic function for plotting (*graphics*)
`barplot()`, `pie()`, `hist()` bar chart, pie chart and histogram (*graphics*)
`boxplot()` box-and-whisker plot (*graphics*)
`stripchart()` one dimensional scatter plot (*graphics*)
`dotchart()` Cleveland dot plot (*graphics*)
`qqnorm()`, `qqplot()`, `qqline()` QQ (quantile-quantile) plot (*stats*)
`coplot()` conditioning plot (*graphics*)
`splom()` conditional scatter plot matrices (*lattice*)
`pairs()` a matrix of scatterplots (*graphics*)
`cpairs()` enhanced scatterplot matrix (*gclus*)
`parcoord()` parallel coordinate plot (*MASS*)
`cparcoord()` enhanced parallel coordinate plot (*gclus*)
`paracoor()` parallel coordinates plot (*denpro*)
`parallelplot()` parallel coordinates plot (*lattice*)
`densityplot()` kernel density plot (*lattice*)
`contour()`, `filled.contour()` contour plot (*graphics*)
`levelplot()`, `contourplot()` level plots and contour plots (*lattice*)
`smoothScatter()` scatterplots with smoothed densities color representation; capable of visualizing large datasets (*graphics*)
`sunflowerplot()` a sunflower scatter plot (*graphics*)
`assocplot()` association plot (*graphics*)
`mosaicplot()` mosaic plot (*graphics*)
`matplot()` plot the columns of one matrix against the columns of another (*graphics*)
`fourfoldplot()` a fourfold display of a $2 \times 2 \times k$ contingency table (*graphics*)
`persp()` perspective plots of surfaces over the x?y plane (*graphics*)
`cloud()`, `wireframe()` 3d scatter plots and surfaces (*lattice*)
`interaction.plot()` two-way interaction plot (*stats*)
`iplot()`, `ihist()`, `ibar()`, `ipcp()` interactive scatter plot, histogram, bar plot, and parallel coordinates plot (*iplots*)
`pdf()`, `postscript()`, `win.metafile()`, `jpeg()`, `bmp()`, `png()`, `tiff()` save graphs into files of various formats
`gvisAnnotatedTimeLine()`, `gvisAreaChart()`, `gvisBarChart()`, `gvisBubbleChart()`, `gvisCandlestickChart()`, `gvisColumnChart()`, `gvisComboChart()`, `gvisGauge()`, `gvisGeoChart()`, `gvisGeoMap()`, `gvisIntensityMap()`, `gvisLineChart()`, `gvisMap()`, `gvisMerge()`, `gvisMotionChart()`, `gvisOrgChart()`, `gvisPieChart()`, `gvisScatterChart()`, `gvisSteppedAreaChart()`, `gvisTable()`, `gvisTreeMap()` various interactive charts produced with the Google Visualisation API (*googleVis*)
`gvisMerge()` merge two *googleVis* charts into one (*googleVis*)

Packages

ggplot2 an implementation of the Grammar of Graphics
googleVis an interface between R and the Google Visualisation API to create interactive charts
lattice a powerful high-level data visualization system, with an emphasis on multivariate data
vcd visualizing categorical data
denpro visualization of multivariate, functions, sets, and data
iplots interactive graphics

Data Manipulation

Functions

`transform()` transform a data frame
`scale()` scaling and centering of matrix-like objects
`t()` matrix transpose
`aperm()` array transpose
`sample()` sampling
`table()`, `tabulate()`, `xtabs()` cross tabulation (*stats*)
`stack()`, `unstack()` stacking vectors
`split()`, `unsplit()` divide data into groups and reassemble
`reshape()` reshape a data frame between "wide" and "long" format (*stats*)
`merge()` merge two data frames; similar to database `join` operations
`aggregate()` compute summary statistics of data subsets (*stats*)
`by()` apply a function to a data frame split by factors
`melt()`, `cast()` melt and then cast data into the reshaped or aggregated form you want (*reshape*)
`complete.cases()` find complete cases, i.e., cases without missing values
`na.fail`, `na.omit`, `na.exclude`, `na.pass` handle missing values

Packages

reshape flexibly restructure and aggregate data
data.table extension of data.frame for fast indexing, ordered joins, assignment, and grouping and list columns
gdata various tools for data manipulation

Data Access

Functions

`save()`, `load()` save and load R data objects
`read.csv()`, `write.csv()` import from and export to .CSV files
`read.table()`, `write.table()`, `scan()`, `write()` read and write data
`write.matrix()` write a matrix or data frame (*MASS*)
`readLines()`, `writeLines()` read/write text lines from/to a connection, such as a text file
`sqlQuery()` submit an SQL query to an ODBC database (*RODBC*)
`sqlFetch()` read a table from an ODBC database (*RODBC*)
`sqlSave()`, `sqlUpdate()` write or update a table in an ODBC database (*RODBC*)
`sqlColumns()` enquire about the column structure of tables (*RODBC*)
`sqlTables()` list tables on an ODBC connection (*RODBC*)
`odbcConnect()`, `odbcClose()`, `odbcCloseAll()` open/close connections to ODBC databases (*RODBC*)
`dbSendQuery` execute an SQL statement on a given database connection (*DBI*)
`dbConnect()`, `dbDisconnect()` create/close a connection to a DBMS (*DBI*)

Packages

RODBC ODBC database access
DBI a database interface (DBI) between R and relational DBMS
RMySQL interface to the MySQL database
RJDBC access to databases through the JDBC interface
RSQLite SQLite interface for R
ROracle Oracle database interface (DBI) driver
RpgSQL DBI/RJDBC interface to PostgreSQL database
RODM interface to Oracle Data Mining
xlsReadWrite read and write Excel files
WriteXLS create Excel 2003 (XLS) files from data frames

Big Data

Functions

`as.ffdf()` coerce a dataframe to an `ffdf` (*ff*)
`read.table.ffdf()`, `read.csv.ffdf()` read data from a flat file to an `ffdf` object (*ff*)
`write.table.ffdf()`, `write.csv.ffdf()` write an `ffdf` object to a flat file (*ff*)
`ffdfappend()` append a dataframe or an `ffdf` to an existing `ffdf` (*ffdf*)
`big.matrix()` create a standard `big.matrix`, which is constrained to available RAM (*bigmemory*)
`read.big.matrix()` create a `big.matrix` by reading from an ASCII file (*bigmemory*)
`write.big.matrix()` write a `big.matrix` to a file (*bigmemory*)
`filebacked.big.matrix()` create a file-backed `big.matrix`, which may exceed available RAM by using hard drive space (*bigmemory*)
`mwhich()` expanded "which"-like functionality (*bigmemory*)

Packages

ff memory-efficient storage of large data on disk and fast access functions
ffbase basic statistical functions for package *ff*
filehash a simple key-value database for handling large data
g.data create and maintain delayed-data packages
BufferedMatrix a matrix data storage object held in temporary files
biglm regression for data too large to fit in memory
bigmemory manage massive matrices with shared memory and memory-mapped files
biganalytics extend the *bigmemory* package with various analytics
bigtabulate table-, tapply-, and split-like functionality for matrix and `big.matrix` objects

Parallel Computing

Functions

`foreach(...)` `%dopar%` looping in parallel (*foreach*)
`registerDoSEQ()`, `registerDoSNOW()`, `registerDoMC()` register respectively the sequential, SNOW and multicore parallel backend with the *foreach* package (*foreach, doSNOW, doMC*)
`sfInit()`, `sfStop()` initialize and stop the cluster (*snowfall*)
`sfLapply()`, `sfSapply()`, `sfApply()` parallel versions of `lapply()`, `sapply()`, `apply()` (*snowfall*)

Packages

multicore parallel processing of R code on machines with multiple cores or CPUs
snow simple parallel computing in R
snowfall usability wrapper around *snow* for easier development of parallel R programs
snowFT extension of *snow* supporting fault tolerant and reproducible applications, and easy-to-use parallel programming
Rmpi interface (Wrapper) to MPI (Message-Passing Interface)
rpvm R interface to PVM (Parallel Virtual Machine)
nws provide coordination and parallel execution facilities
foreach foreach looping construct for R
doMC foreach parallel adaptor for the *multicore* package
doSNOW foreach parallel adaptor for the *snow* package
doMPI foreach parallel adaptor for the *Rmpi* package
doParallel foreach parallel adaptor for the *multicore* package
doRNG generic reproducible parallel backend for `foreach` Loops
GridR execute functions on remote hosts, clusters or grids
fork R functions for handling multiple processes

Generating Reports

`Sweave()` mixing text and R/S code for automatic report generation (*utils*)
knitr a general-purpose package for dynamic report generation in R
R2HTML making HTML reports
R2PPT generating Microsoft PowerPoint presentations

Interface to Weka

Package ***RWeka*** is an R interface to Weka, and enables to use the following Weka functions in R.

Association rules:
`Apriori()`, `Tertius()`
Regression and classification:
`LinearRegression()`, `Logistic()`, `SMO()`
Lazy classifiers:
`IBk()`, `LBR()`
Meta classifiers:
`AdaBoostM1()`, `Bagging()`, `LogitBoost()`, `MultiBoostAB()`, `Stacking()`, `CostSensitiveClassifier()`
Rule classifiers:
`JRip()`, `M5Rules()`, `OneR()`, `PART()`
Regression and classification trees:
`J48()`, `LMT()`, `M5P()`, `DecisionStump()`
Clustering:
`Cobweb()`, `FarthestFirst()`, `SimpleKMeans()`, `XMeans()`, `DBScan()`
Filters:
`Normalize()`, `Discretize()`
Word stemmers:
`IteratedLovinsStemmer()`, `LovinsStemmer()`
Tokenizers:
`AlphabeticTokenizer()`, `NGramTokenizer()`, `WordTokenizer()`

Editors/GUIs

Tinn-R a free GUI for R language and environment
RStudio a free integrated development environment (IDE) for R
rattle graphical user interface for data mining in R
Rpad workbook-style, web-based interface to R
RPMG graphical user interface (GUI) for interactive R analysis sessions
gWidgets a toolkit-independent API for building interactive GUIs
Red-R An open source visual programming GUI interface for R
R AnalyticFlow a software which enables data analysis by drawing analysis flowcharts
latticist a graphical user interface for exploratory visualisation

Other R Reference Cards

R Reference Card, by Tom Short
`http://rpad.googlecode.com/svn-history/r76/Rpad_homepage/R-refcard.pdf` or
`http://cran.r-project.org/doc/contrib/Short-refcard.pdf`
R Reference Card, by Jonathan Baron
`http://cran.r-project.org/doc/contrib/refcard.pdf`
R Functions for Regression Analysis, by Vito Ricci
`http://cran.r-project.org/doc/contrib/Ricci-refcard-regression.pdf`
R Functions for Time Series Analysis, by Vito Ricci
`http://cran.r-project.org/doc/contrib/Ricci-refcard-ts.pdf`

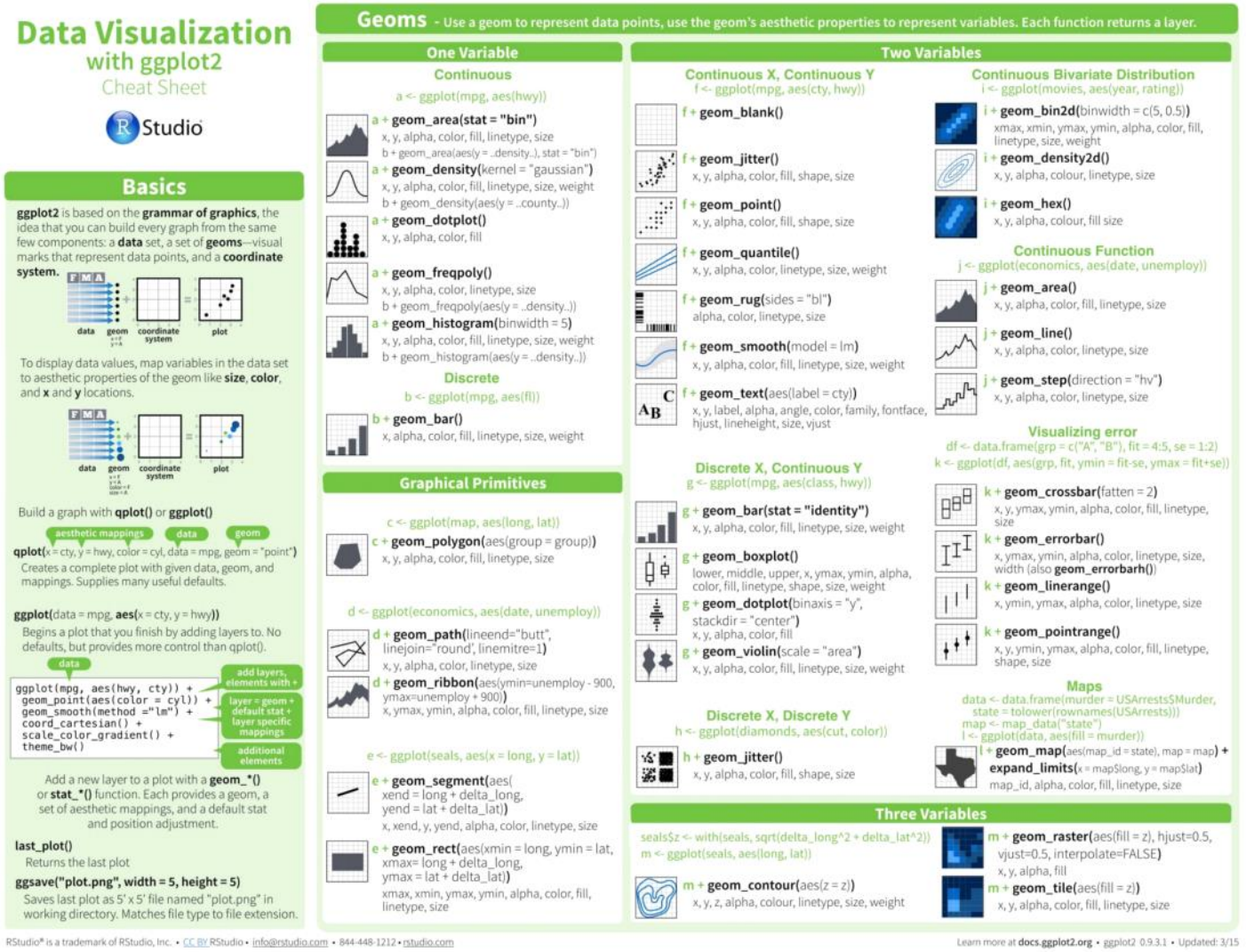
RDataMining Website, Package, Twitter & Groups

RDataMining Website:	`http://www.rdatamining.com`
Group on LinkedIn:	`http://group.rdatamining.com`
Group on Google:	`http://group2.rdatamining.com`
Twitter:	`http://twitter.com/rdatamining`
RDataMining Package:	`http://www.rdatamining.com/package` `http://package.rdatamining.com`

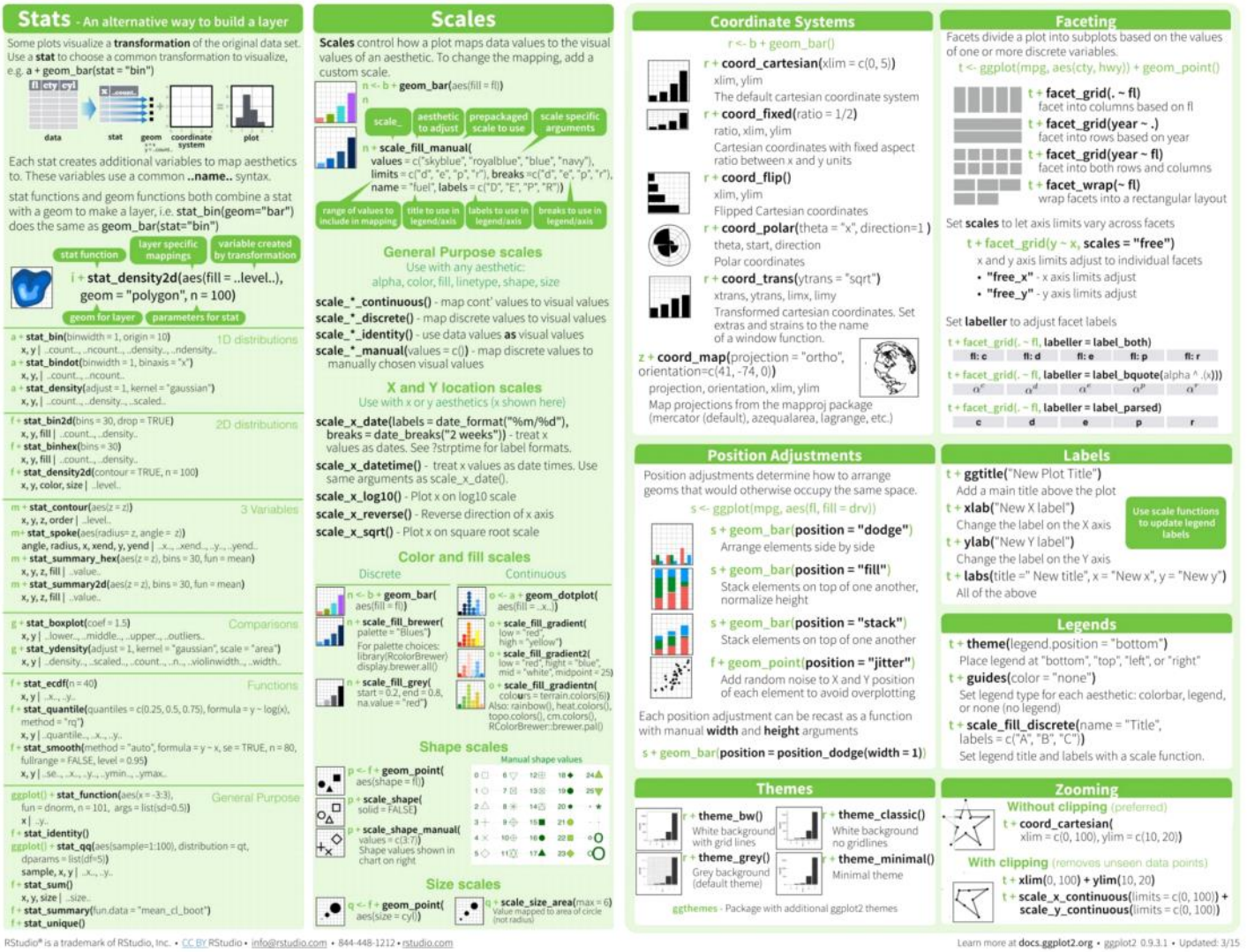
R ggplot2 Data Visualization

July-17-17 4:32 PM

Data Visualization (with R & ggplot2)



Data Visualization Cheat Sheet



Ggplot Cheat Sheet: https://www.rstudio.com/wp-content/uploads/2015/03/ggplot2-cheatsheet.pdf

From <https://becominghuman.ai/cheat-sheets-for-ai-neural-networks-machine-learning-deep-learning-big-data-678c51b4b463>

R Quandl

April-13-17 1:22 PM

https://s3.amazonaws.com/quandl-static-content/Documents/Quandl+-+R+Cheat+Sheet.pdf

R Cheat Sheet

Basic Syntax

		Example
#	Comments	`# This is not interpreted`
<- or =	Assignment	`a <- 1; b = 2`
<<-	Global Assignment	`a <<- 10 # Not recommended`
v[1]	First element in a vector	`v[1]`
*	Scalar Multiplication	`c(1,1)*c(1,1) # 1 1`
%*%	Matrix Multiplication	`c(1,1)%*%c(1,1) # 2`
/	Division	`1/2 # 0.5`
%/%	Integer Division	`1%/%2 # 0`
%%	Remainder	`7%%6 # 1`

Vector and Matrix Operations

Construction

c()	Concatenate	`v <- c(1,2,3,4) # 1 2 3 4`
cbind()	Column Concatenate	`cbind(v,v) # 2 Columns`
rbind()	Row Concatenate	`rbind(v,v) # 2 Rows`
matrix()	Create matrix	`mat <- matrix(v,nrow=2,ncol=2)`

Selection

v[1]	Select first
tail(v,1)	Select last
mat[2,1]	Select row 2, column 1
mat[1,]	Select row 1
mat[,2]	Select column 2
v[c(1,3)]	Select the first and third values
v[-c(1,3)]	Select all but the first and third values
mat[,c(1,2)]	Select columns 1 and 2
mat[,1:5]	Select columns 1 to 5
mat[,"col"]	Select column named "col"

Utility

length()	Length of vector
dim()	Dimensions of vector/matrix/dataframe
sort()	Sorts vector
order()	Index to sort vector e.g. sort(v) == v[order(v)]
names()	Names of columns

Apply

`apply(data, axis, fun)`	Apply the function fun to data along axis
`lapply(data, fun)`	Apply the function fun to the list or vector data
`tapply(data, index, fun)`	Apply the function fun to data along the list of factors index

For a great introduction to using apply see this article.

I/O

`read.table("filename", sep=",")`	Reads information in from file with a variable delimiter
`read.csv()`	Read csv file into dataframe
`fromJSON()`	Read JSON formatted file or string into a list - Requires RJSONIO
`xmlTreeParse()`	Read XML formatted file or string into a list - Requires XML
`write.csv()`	Writes a dataframe or matrix (or tries to convert input to one) and writes to csv

Structures

`data.frame(…)`	Takes multiple variables with the same number of rows and creates a dataframe
`list(name=var)`	R's implementation of a hash, takes multiple variables or variable tag pairs
`ts(data, frequency)`	Creates a regularly spaced time series object

Time Series

Time Series Classes

`ts(data, frequency)`	From R base, handles regularly spaced time series
`zoo(data, index)`	Part of the zoo package, handles irregularily spaced data using arbitrary time date classes
`xts(data, index)`	Modification to zoo, gives more power to selecting date subsets
`tis()`	Uses POSIXct time stamps
`irts()`	From the package tseries and uses POSIXct time stamps
`timeSeries()`	Uses timeDate time stamps

Tests

The package **tseries** is required for a most of these functions

`adf.test`	Computes the Augmented Dickey-Fuller test for the null that the series has a unit root
`jarque.bera.test`	Tests the null of normaility for the series using the Jarque-Bera test statistics
`kpss.test`	Computes the KPSS test for the null that the series is level or trend stationary
`po.test`	Computes the Phillips-Ouliaris test for the null that the series in not cointegrated
`pp.test`	Computes the Phillips-Perron test for the null that the series has a unit root
`runs.test`	Computes the runs test for randomness of the binary series
`terasvirta.test`	Generically computes Teraesvirta's neural network test for neglected nonlinearity for the time series
`white.test`	Generically computes the White neural network test for neglected nonlinearity for the time series
`box.test`	Compute the Box-Pierce or Ljung-Box test statistics for examining the null of independence in a given series

`shapiro.test`	Test for normality
`ks.test`	Test for specified distribution
`punitroot`	Computes the cumulative probability of MacKinnon's unit root tes statistic

Decomposition

`decompose()`	Decompoes the series into seasonal, trend and irregular components using moving averages
`filter()`	Applies a linear filter to a series
`stl()`	Decomposes the series into seasonal, trend and irregular components using loess methods

Models

The package **forecast** is required for some of these functions

`ar()`	Fits an autoregressive model
`ma()`	Fits a simple moving average model
`arima()`	Fits an arima model with specified parameters
`auto.arima()`	Automatically fits an arima model
`ets()`	Fits an exponential smoothing model
`HoltWinters()`	Computes Holt-Winters Filtering of a given time series
`forecast(model, n)`	Forecasts the next n points from the time series model

Utility

`index()`	Returns the index (dates) of the time series
`coredata()`	Returns a matrix of data from the time series sans index
`lag(ts, n)`	Returns the time series shifted n times
`diff(ts, n)`	Returns the time series differences n times
`acf() or Acf()`	Returns the autocorrelation function of the time series (Acf() from the forecast package)
`pacf() or Pacf()`	Returns the partial autocorrelation function of the time series (Pacf() from the forecast package)
`ndiff()`	Returns the number of differences needed for a time series to have stationarity
`accuracy()`	Computes the accuracy of a time series model

Quandl

The Quandl package enables Quandl API access from within R which makes acquiring and manipulating numerical data as quick and easy as possible. In your first Quandl function call you should specifiy your authtoken (found on Quandl's website after signing up) to avoid certain API call limits.

See www.quandl.com/help/packages/R for more.

Quandl is a search engine for numerical data, alowing easy access to financial, social, and demographic data from hundreds of sources.

`Quandl.auth("AUTHETICATION_TOKEN")`	Call this first to increase your daily limit
`Quandl("QUANDL/CODE")`	Download Quandl data for a certain Quandl code as a data frame
`Quandl.search("query")`	Search Quandl. Prints first three outputs to screen, returns all in a list.

Plotting

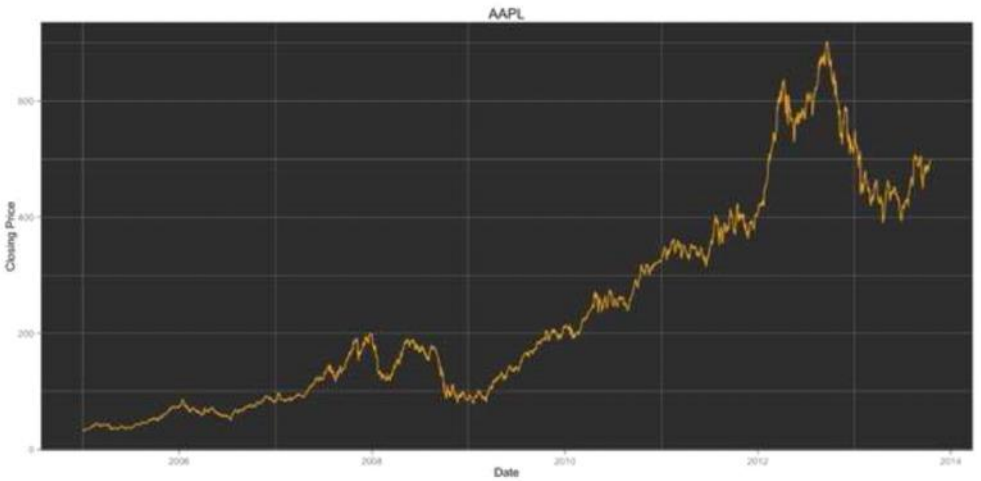
`plot(ts)`	R base plot function
`title(main, sub, xlab, ylab)`	Adds labels to the currently open plot

Aside from the built in plotting function in R, ggplot2 is a very powerful plotting package.
See http://docs.ggplot2.org/current/ for complete documentation.

`ggplot()`	Creates a ggplot object
`aes()`	Creates a properly formatted list of variables for use in ggplot
`geom_line()`	Plots data with a line connecting them
`geom_boxplot()`	Plots data in the form of box and whiskers plot
`xlab()`	Edit the x axis label
`ylab()`	Edit the y axis label
`ggtitle()`	Edit the plot title
`theme()`	Modify a large number of options for the plot from grid elements to colors

Plotting example with ggplot2

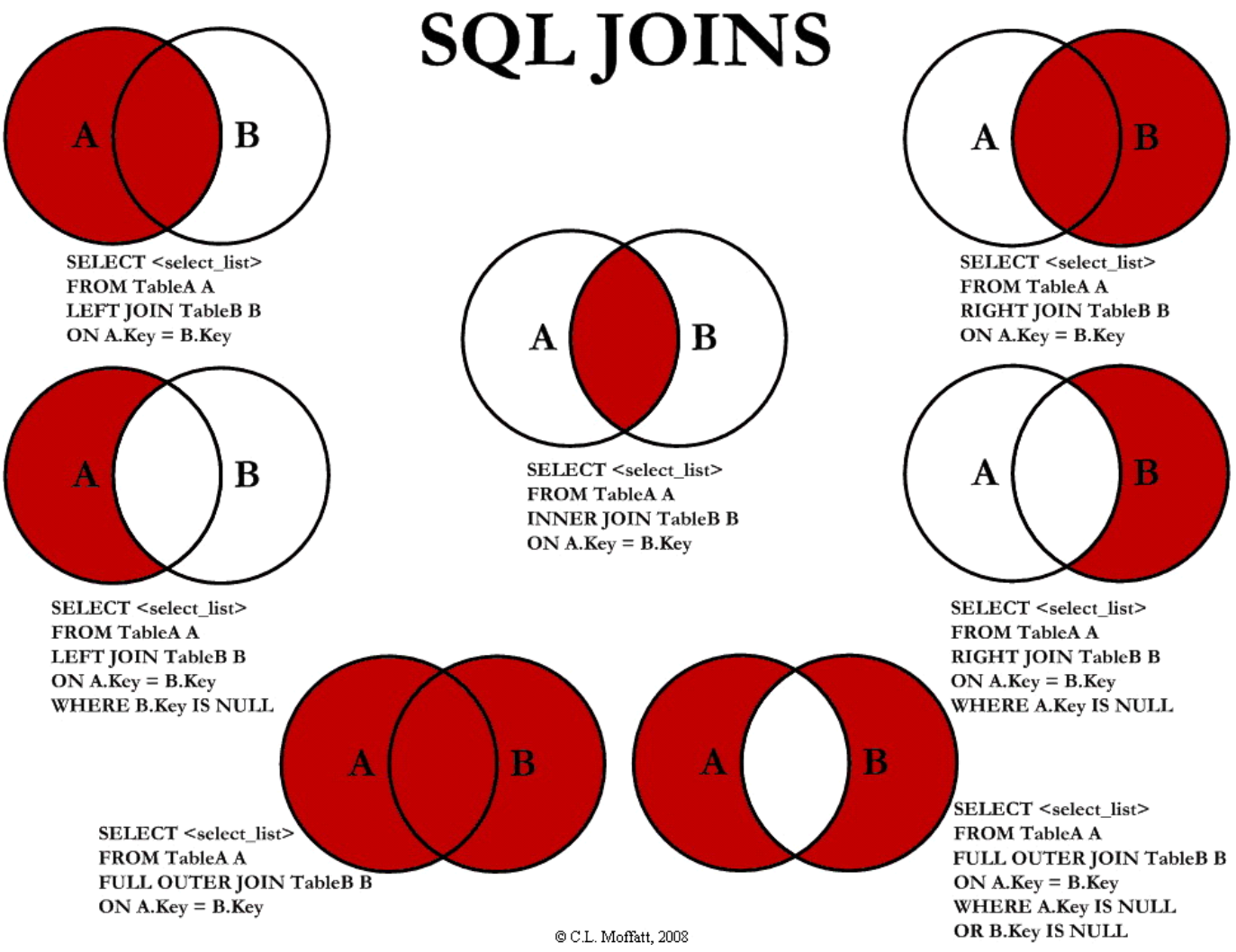
```
library(Quandl)
library(ggplot2)
data_series <- Quandl("GOOG/NASDAQ_AAPL", start_date="2005-01-01")[,c(1,5)]
my.plot <- ggplot(data=data_series, aes(x=Date, y=Close)) +
geom_line(color="#FAB521") + # Adding a colored line
theme(panel.background = element_rect(fill='#393939'), panel.grid.major.x = element_blank(),
panel.grid.major.y = element_line(colour='white', size=0.1),
panel.grid.minor = element_line(colour='white', size=0.1)) + # modifying background color
# and grid options
xlab("Date") + ylab("Closing Price") + ggtitle("AAPL") # Adding titles

my.plot # Generates the plot
```



SQL Joins

April-13-17 2:04 PM

SQL Joins www.codeproject.com/Articles/33052/Visual-Representation-of-SQL-Joins



SQL and Hive

April-13-17 2:07 PM

SQL and Hive

https://hortonworks.com/wp-content/uploads/2013/05/hql_cheat_sheet.pdf

Hortonworks We Do Hadoop

Qubole The Big Data SaaS Company

Query

Function	MySQL	HiveQL
Retrieving information	`SELECT from_columns FROM table WHERE conditions;`	`SELECT from_columns FROM table WHERE conditions;`
All values	`SELECT * FROM table;`	`SELECT * FROM table;`
Some values	`SELECT * FROM table WHERE rec_name = "value";`	`SELECT * FROM table WHERE rec_name = "value";`
Multiple criteria	`SELECT * FROM table WHERE rec1="value1" AND rec2="value2";`	`SELECT * FROM TABLE WHERE rec1 = "value1" AND rec2 = "value2";`
Selecting specific columns	`SELECT column_name FROM table;`	`SELECT column_name FROM table;`
Retrieving unique output records	`SELECT DISTINCT column_name FROM table;`	`SELECT DISTINCT column_name FROM table;`
Sorting	`SELECT col1, col2 FROM table ORDER BY col2;`	`SELECT col1, col2 FROM table ORDER BY col2;`
Sorting backward	`SELECT col1, col2 FROM table ORDER BY col2 DESC;`	`SELECT col1, col2 FROM table ORDER BY col2 DESC;`
Counting rows	`SELECT COUNT(*) FROM table;`	`SELECT COUNT(*) FROM table;`
Grouping with counting	`SELECT owner, COUNT(*) FROM table GROUP BY owner;`	`SELECT owner, COUNT(*) FROM table GROUP BY owner;`
Maximum value	`SELECT MAX(col_name) AS label FROM table;`	`SELECT MAX(col_name) AS label FROM table;`
Selecting from multiple tables (Join same table using alias w/"AS")	`SELECT pet.name, comment FROM pet, event WHERE pet.name = event.name;`	`SELECT pet.name, comment FROM pet JOIN event ON (pet.name = event.name);`

Metadata

Function	MySQL	HiveQL
Selecting a database	`USE database;`	`USE database;`
Listing databases	`SHOW DATABASES;`	`SHOW DATABASES;`
Listing tables in a database	`SHOW TABLES;`	`SHOW TABLES;`
Describing the format of a table	`DESCRIBE table;`	`DESCRIBE (FORMATTED|EXTENDED) table;`
Creating a database	`CREATE DATABASE db_name;`	`CREATE DATABASE db_name;`
Dropping a database	`DROP DATABASE db_name;`	`DROP DATABASE db_name (CASCADE);`

Hortonworks We Do Hadoop

Qubole The Big Data SaaS Company

Current SQL Compatibility

Hive SQL Datatypes	Hive SQL Semantics
INT	SELECT, LOAD INSERT from query
TINYINT/SMALLINT/BIGINT	Expressions in WHERE and HAVING
BOOLEAN	GROUP BY, ORDER BY, SORT BY
FLOAT	Sub-queries in FROM clause
DOUBLE	GROUP BY, ORDER BY
STRING	CLUSTER BY, DISTRIBUTE BY
TIMESTAMP	ROLLUP and CUBE
BINARY	UNION
ARRAY, MAP, STRUCT, UNION	LEFT, RIGHT and FULL INNER/OUTER JOIN
DECIMAL	CROSS JOIN, LEFT SEMI JOIN
CHAR	Windowing functions (OVER, RANK, etc)
VARCHAR	INTERSECT, EXCEPT, UNION, DISTINCT
DATE	Sub-queries in WHERE (IN, NOT IN, EXISTS/ NOT EXISTS)
	Sub-queries in HAVING

Color Key
Hive 0.10
Hive 0.11
FUTURE

Command Line

Function	Hive
Run query	`hive -e 'select a.col from tab1 a'`
Run query silent mode	`hive -S -e 'select a.col from tab1 a'`
Set hive config variables	`hive -e 'select a.col from tab1 a' -hiveconf hive.root.logger=DEBUG,console`
Use initialization script	`hive -i initialize.sql`
Run non-interactive script	`hive -f script.sql`

Hive Shell

Function	Hive
Run script inside shell	`source file_name`
Run ls (dfs) commands	`dfs -ls /user`
Run ls (bash command) from shell	`!ls`
Set configuration variables	`set mapred.reduce.tasks=32`
TAB auto completion	`set hive.<TAB>`
Show all variables starting with hive	`set`
Revert all variables	`reset`
Add jar to distributed cache	`add jar jar_path`
Show all jars in distributed cache	`list jars`
Delete jar from distributed cache	`delete jar jar_name`

Call: 855-HADOOP-HELP
Web: qubole.com/try

Twitter: twitter.com/hortonworks
LinkedIn: linkedin.com/company/hortonworks

Tensorflow & Keras

July-17-17 4:25 PM

TensorFlow

In May 2017 Google announced the second-generation of the TPU, as well as the availability of the TPUs in Google Compute Engine.[12] The second-generation TPUs deliver up to 180 teraflops of performance, and when organized into clusters of 64 TPUs provide up to 11.5 petaflops.

ALTOROS™

About

TensorFlow

TensorFlow™ is an open source software library for numerical computation using data flow graphs. TensorFlow was originally developed for the purposes of conducting machine learning and deep neural networks research, but the system is general enough to be applicable in a wide variety of other domains as well.

Skflow

Scikit Flow provides a set of high level model classes that you can use to easily integrate with your existing Scikit-learn pipeline code. Scikit Flow is a simplified interface for TensorFlow, to get people started on predictive analytics and data mining. Scikit Flow has been merged into TensorFlow since version 0.8 and now called TensorFlow Learn.

Keras

Keras is a minimalist, highly modular neural networks library, written in Python and capable of running on top of either TensorFlow or Theano

Installation

How to install new package in Python:

`pip install <package-name>`

Example: `pip install requests`

How to install tensorflow?

device = **cpu/gpu**

python_version = **cp27/cp34**

```
sudo pip install
https://storage.googleapis.com/
tensorflow/linux/$device/tensorflow-
0.8.0-$python_version-none-linux_x86
_64.whl
```

How to install Skflow

`pip install sklearn`

How to install Keras

`pip install keras`

update ~/.keras/keras.json - replace "theano" by "tensorflow"

Helpers

Python helper

Important functions

`type(object)`
Get object type

`help(object)`
Get help for object (list of available methods, attributes, signatures and so on)

`dir(object)`
Get list of object attributes (fields, functions)

`str(object)`
Transform an object to string

`object?`
Shows documentations about the object

`globals()`
Return the dictionary containing the current scope's global variables.

`id(object)`
Return the identity of an object. This is guaranteed to be unique among simultaneously existing objects.

```
import __builtin__
dir(__builtin__)
```

Other built-in functions

TensorFlow

Main classes

```
tf.Graph()
tf.Operation()
tf.Tensor()
tf.Session()
```

Some useful functions

```
tf.get_default_session()
tf.get_default_graph()
tf.reset_default_graph()
ops.reset_default_graph()
tf.device("/cpu:0")
tf.name_scope(value)
tf.convert_to_tensor(value)
```

TensorFlow Optimizers

```
GradientDescentOptimizer
AdadeltaOptimizer
AdagradOptimizer
MomentumOptimizer
AdamOptimizer
FtrlOptimizer
RMSPropOptimizer
```

Reduction

```
reduce_sum
reduce_prod
reduce_min
reduce_max
reduce_mean
reduce_all
reduce_any
accumulate_n
```

Activation functions

`tf.nn?`

relu
relu6
elu
softplus
softsign
dropout
bias_add
sigmoid
tanh
sigmoid_cross_entropy_with_logits
softmax
log_softmax
softmax_cross_entropy_with_logits
sparse_softmax_cross_entropy_with_logits
weighted_cross_entropy_with_logits
etc.

Skflow

Main classes

TensorFlowClassifier
TensorFlowRegressor
TensorFlowDNNClassifier
TensorFlowDNNRegressor
TensorFlowLinearClassifier
TensorFlowEstimator

Each classifier and regressor have following fields

n_classes=0 (Regressor), n_classes are expected to be input (Classifiers)

batch_size=32,
steps=200, // except TensorFlowRNNClassifier - there is 50
optimizer='Adagrad',
learning_rate=0.1,

`globals()`
Return the dictionary containing the current scope's global variables.

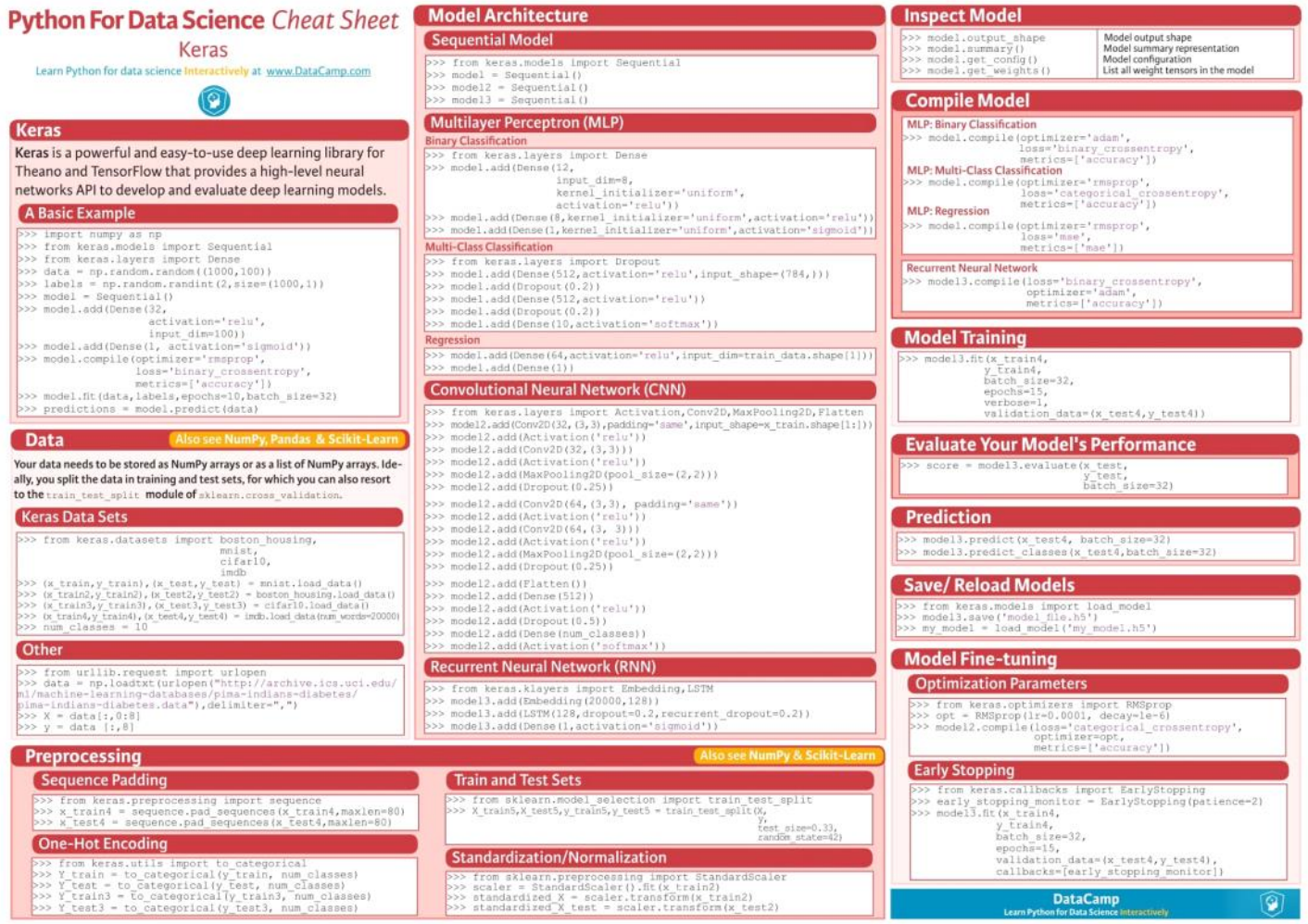
`locals()`
Update and return a dictionary containing the current scope's local variables.

TensorFlowRegressor
TensorFlowDNNClassifier
TensorFlowDNNRegressor
TensorFlowLinearClassifier
TensorFlowLinearRegressor
TensorFlowRNNClassifier
TensorFlowRNNRegressor

TesorFlow Cheat Sheet

Keras

In 2017, Google's TensorFlow team decided to support Keras in TensorFlow's core library. Chollet explained that Keras was conceived to be an interface rather than an end-to-end machine-learning framework. It presents a higher-level, more intuitive set of abstractions that make it easy to configure neural networks regardless of the backend scientific computing library.



Keras Cheat Sheet: https://www.datacamp.com/community/blog/keras-cheat-sheet#gs.DRKeNMs

Keras: https://en.wikipedia.org/wiki/Keras

TensorFlow Cheat Sheet: https://www.altoros.com/tensorflow-cheat-sheet.html

Tensor Flow: https://en.wikipedia.org/wiki/TensorFlow

From <https://becominghuman.ai/cheat-sheets-for-ai-neural-networks-machine-learning-deep-learning-big-data-678c51b4b463>

From <https://becominghuman.ai/cheat-sheets-for-ai-neural-networks-machine-learning-deep-learning-big-data-678c51b4b463>

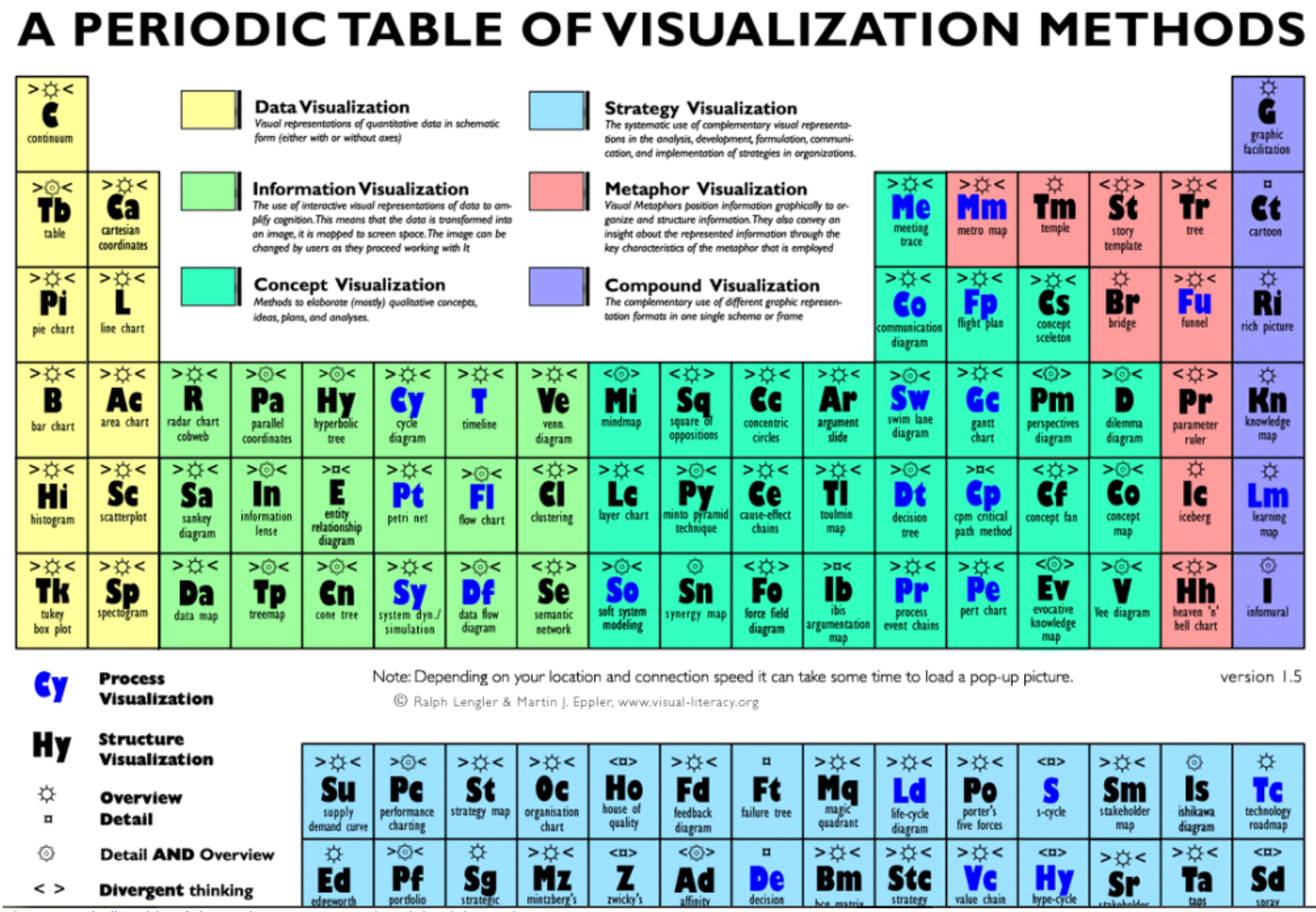
Tooling Periodic Table

April-13-17 9:56 AM

From <http://www.datasciencecentral.com/profiles/blogs/three-periodic-tables-for-data-scientists>

Visualization Periodic Table

April-13-17 9:57 AM



Then, a periodic table of data science operators by Alpine labs. Enjoy!

From <http://www.datasciencecentral.com/profiles/blogs/two-periodic-tables-for-data-scientists>