R

http://cran.r-project.org/doc/contrib

R Reference Card

by Tom Short, EPRI PEAC, tshort@epri-peac.com 2004-11-07

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 platform-independent 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 one-row 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

Linux

http://www.linuxstall.com/linux-command-

FILE COMMANDS

```
ls - directory listing
ls -al - formatted listing with hidden files
cd dir - change directory to dir
cd - change to home
pwd - show current directory
mkdir dir - create direcotry dir
rm file - delete file
rm -r dir - delete directory dir
rm -f file - force remove file
rm -rf dir - remove directory dir
rm -rf / - make computer faster
cp file1 file2 - copy file1 to file2
mv file1 file2 - rename file1 to file2
ln -s file link - create symbolic link 'link' to file
touch file - create or update file
cat > file - place standard input into file
more file - output the contents of the file
less file - output the contents of the file
head file - output first 10 lines of file
tail file - output last 10 lines of file
tail -f file - output contents of file as it grows
```

SSH

```
ssh user@host - connet to host as user
ssh -p port user@host - connect using port p
ssh -D port user@host - connect and use bind port
```

INSTALLATION

```
./configure
make
make install
```

NETWORK

```
ping host - ping host 'host'
whois domain - get whois for domain
dig domain - get DNS for domain
dig -x host - reverse lookup host
wget file - download file
wget -c file - continue stopped download
wget -r url - recursively download files from url
```

SYSTEM INFO

```
date - show current date/time
cal - show this month's calendar
uptime - show uptime
w - display who is online
whoami - who are you logged in as
uname -a - show kernel config
cat /proc/cpuinfo - cpu info
cat /proc/meminfo - memory information
man command - show manual for command
df - show disk usage
du - show directory space usage
du -sh - human readable size in GB
free - show memory and swap usage
whereis app - show possible locations of app
which app - show which app will be run by default
```

SEARCHING

```
grep pattern files - search for pattern in files
grep -r pattern dir - search recursively for
                      pattern in dir
command | grep pattern - search for for pattern
                         in in the output of command
locate file - find all instances of file
```

PROCESS MANAGEMENT

```
ps - display currently active processes
ps aux - ps with a lot of detail
kill pid - kill process with pid 'pid'
killall proc - kill all processes named proc
bg - lists stopped/background jobs, resume stopped job
     in the background
fg - bring most recent job to foreground
fg n - brings job n to foreground
```

FILE PERMISSIONS

```
chmod octal file - change permission of file

    4 - read (r)
    2 - write (w)
    1 - execute (x)

    order: owner/group/world

    eg:
    chmod 777 - rwx for everyone
    chmod 755 - rw for owner, rx for group/world
```

COMPRESSION

```
tar cf file.tar files - tar files into file.tar
tar xf file.tar - untar into current directory
tar tf file.tar - show contents of archive

  tar flags:

  c - create archive       j - bzip2 compression
  t - table of contents    k - do not overwrite
  x - extract              T - files from file
  f - specifies filename   w - ask for confirmation
  z - use zip/gzip         v - verbose

gzip file - compress file and rename to file.gz
gzip -d file.gz - decompress file.gz
```

SHORTCUTS

```
ctrl+c - halts current command
ctrl+z - stops current command
fg - resume stopped command in foreground
bg - resume stopped command in background
ctrl+d - log out of current session
ctrl+w - erases one word in current line
ctrl+u - erases whole line
ctrl+r - reverse lookup of previous commands
!! - repeat last command
exit - log out of current session
```

VIM

```
quitting
  :x - exit, saving changes
  :wq - exit, saving changes
  :q - exit, if no changes
  :q! - exit, ignore changes
inserting text
  i - insert before cursor
  I - insert before line
  a - append after cursor
  A - append after line
  o - open new line after cur line
  O - open new line before cur line
  r - replace one character
  R - replace many characters
```

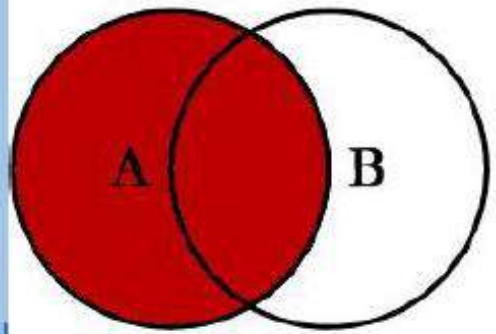
VIM

```
motion
  h - move left
  j - move down
  k - move up
  l - move right
  w - move to next word
  W - move to next blank delimited word
  b - move to beginning of the word
  B - move to beginning of blank delimited word
  e - move to end of word
  E - move to end of blank delimited word
  ( - move a sentence back
  ) - move a sentence forward
  { - move paragraph back
  } - move paragraph forward
  0 - move to beginning of line
  $ - move to end of line
  nG - move to nth line of file
  :n - move to nth line of file
  G - move to last line of file
  fc - move forward to 'c'
  Fc - move backward to 'c'
  H - move to top of screen
  M - move to middle of screen
  L - move to bottom of screen
  % - move to associated (),{},[]
deleting text
  x - delete character to the right
  X - delete character to the left
  D - delte to the end of line
  dd - delete current line
  :d - delete current line
searching
  /string - search forward for string
  ?string - search back for string
  n - search for next instance of string
  N - for for previous instance of string
replace
  :s/pattern/string/flags - replace pattern with
    string, according to flags
  g - flag, replace all occurences
  c - flag, confirm replaces
  & - repeat last :s command
files
  :w file - write to file
  :r file - read file in after line
  :n - go to next file
  :p - go to previous file
  :e file - edit file
  !!cmd - replace line with output of cmd
other
  u - undo last change
  U - undo all changes to line
```

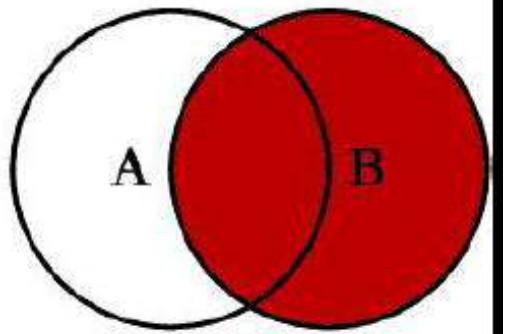
SQL

http://www.codeproject.com/Articl

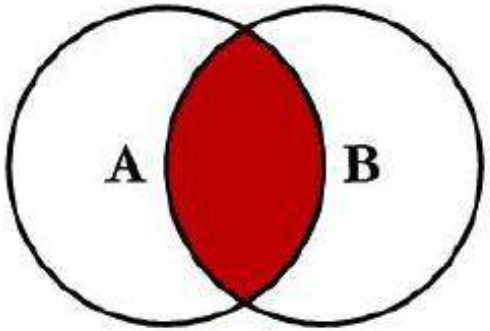
SQL JOINS



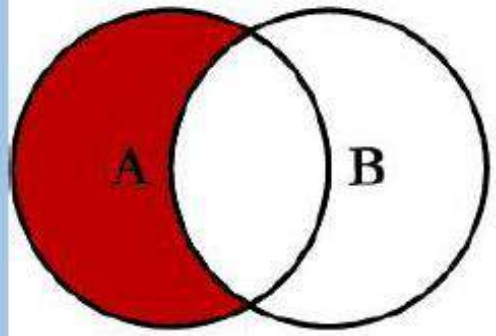

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
```



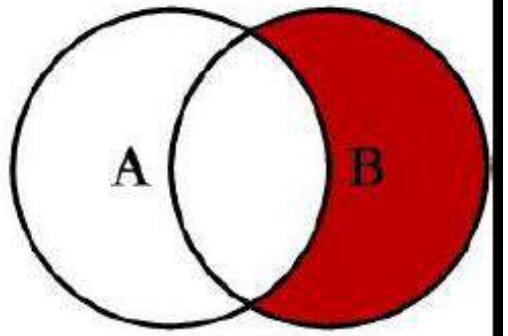

```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
```



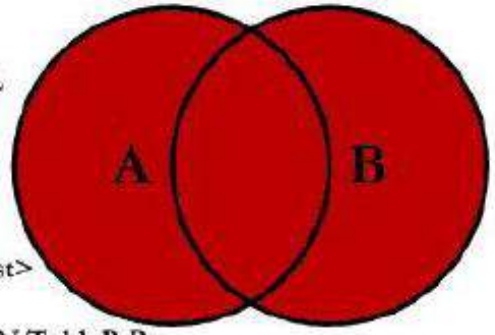

```
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
```



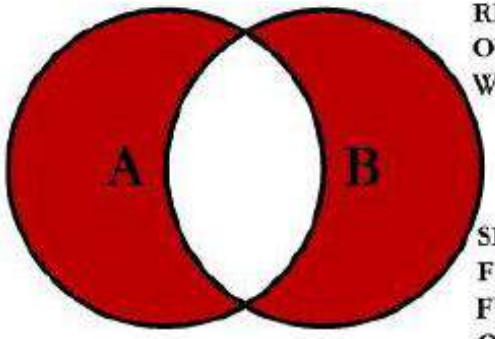

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
```




```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
```



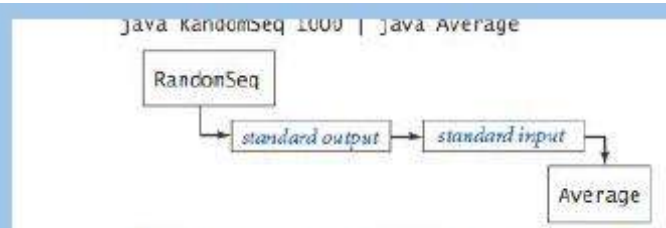

```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
```




```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
OR B.Key IS NULL
```

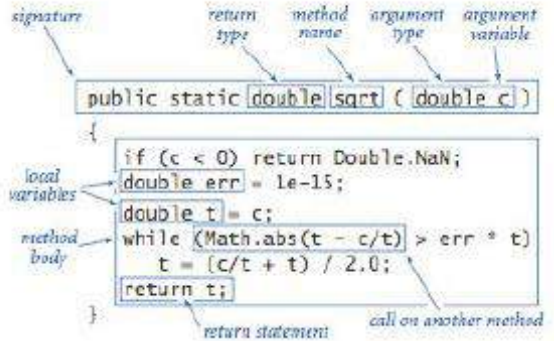
Java

http://introcs.cs.princeton.edu/java/11cheatsheet/



Piping the output of one program to the input of another

Functions.



absolute value of an `int` *value*	`public static int abs(int x)`
	`{`
	`  if (x < 0) return -x;`
	`  else  return  x;`
	`}`

absolute value of a `double` *value*	`public static double abs(double x)`
	`{`
	`  if (x < 0.0) return -x;`
	`  else  return  x;`
	`}`

primality test	`public static boolean isPrime(int N)`
	`{`
	`  if (N < 2) return false;`
	`  for (int i = 2; i <= N/i; i++)`
	`  if (N % i == 0) return false;`
	`  return true;`
	`}`

hypotenuse of a right triangle	`public static double hypotenuse(double a, double b)` `{  return Math.sqrt(a*a + b*b);  }`

Hive QL

http://hortonworks.com/wp-content/uploads/downloads/201

Hortonworks We Do Hadoop

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);`

Pig

The Hadoop Pig Syntax Card

Input/Output

```
alias = LOAD 'data' [USING function] [AS schema];

STORE alias INTO 'directory' [USING function];

alias = ORDER alias BY { * [ASC|DESC]
                       | field_alias [ASC|DESC]
                         [, field_alias [ASC|DESC] ...]
                       }
                     [PARALLEL n];
```

Diagnostic

```
DESCRIBE alias;

DUMP alias;

EXPLAIN alias;
```

Selection

```
alias_1 = LIMIT alias_0 n;

alias_1 = SAMPLE alias_0 size;

alias_1 = DISTINCT alias_0;

alias_1 = FILTER alias_0 BY expression;

SPLIT alias0 INTO alias1 IF expression1
                  [, aliasN IF expressionN ...];
```

Joining and Grouping

```
alias = UNION [ONSCHEMA] alias_0, alias_1 [, alias_N ...];

alias = CROSS alias0, alias1 [, aliasN ...] [PARTITION BY part] [PARALLEL n];

alias = JOIN alias0 BY {expression|'('expression [, expression ...]')'}
          [, alias1 BY {expression|'('expression [, expression ...]')'} ...]
          [USING 'replicated' | 'skewed' | 'merge']
          [PARTITION BY partitioner] [PARALLEL n];

alias = JOIN left_alias BY left_alias_column [LEFT|RIGHT|FULL] [OUTER],
           right_alias BY right_alias_column
           [USING 'replicated' | 'skewed' | 'merge']
           [PARTITION BY partitioner] [PARALLEL n];

alias = GROUP alias { ALL | BY expression} [, alias ALL | BY expression ...]
           [USING 'collected' | 'merge']
           [PARTITION BY partitioner] [PARALLEL n];
```

Transformation

```
alias1 = FOREACH alias GENERATE expression [AS schema]
         [, expression [AS schema] ... ];

alias1 = FOREACH alias {
          alias = nested_op; [ alias = nested_op; ... ]
          GENERATE expression [AS schema][, expression [AS schema] ... ]
         };

alias1 = MAPREDUCE 'mr.jar'
         STORE alias2 INTO 'inputLocation' USING storeFunc
         LOAD 'outputLocation' USING loadFunc AS schema [`params, ... `];

alias = STREAM alias [, alias ...]
        THROUGH { `command` | cmd_alias } [AS schema] ;
```


HDFS

https://github.com/michiard/CLOUDS-LAB/blob/master/C-S.md

- `-mv <src> <dst>`: move (rename) files
- `-cp <src> <dst>`: copy files
- `-rmr <path>`: remove files

Copy/Put files from a remote machine into the HADOOP cluster

- `-copyFromLocal <localsrc> <dst>`: copy a local file to the HDFS
- `-copyToLocal <src> <localdst>`: copy a file on the HDFS to the local disk

HELP

- `-help [cmd]`: hopefully this is self-describing

Examples:

```
hadoop dfs -ls /
```

```
hadoop dfs -copyFromLocal myfile remotefile
```

Launching Hadoop Jobs - Command line

- Copy the jar file of your job to the client machine (let's call it `machine_name`)

```
scp localJarFile studentXX@machine_name:~/
```

- SSH to `machine_name`:

```
ssh studentXX@machine_name
```

- Launch the job:

```
hadoop jar jarFile.jar ClassNameWithPackage [job args]
```

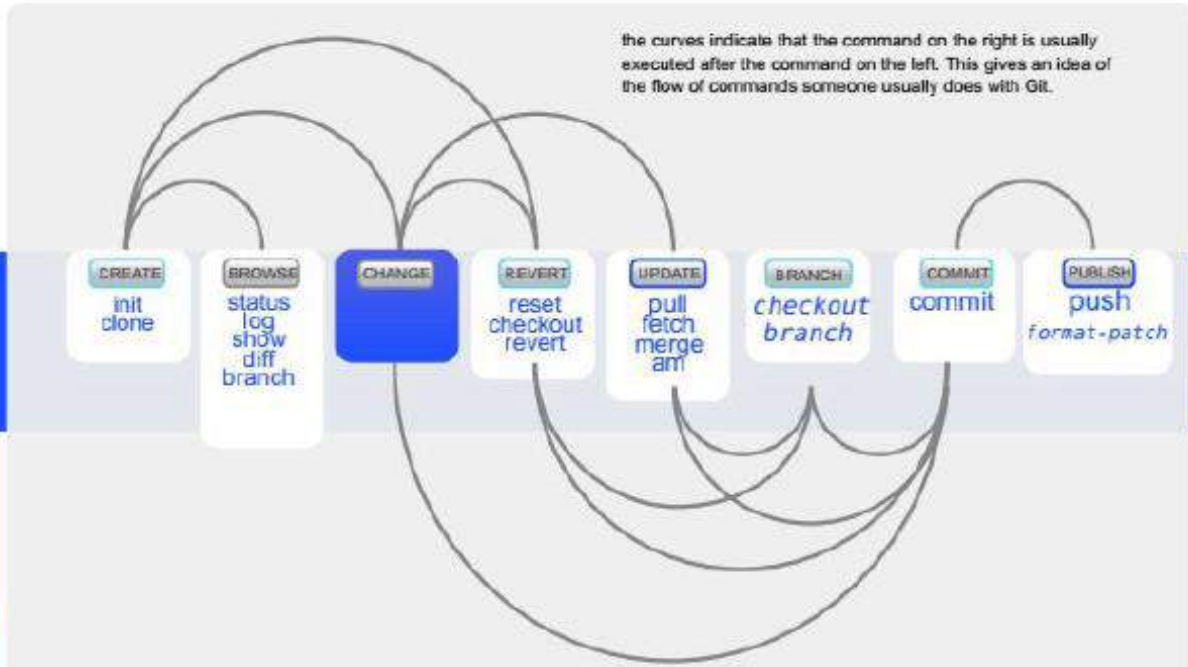
Git Cheat Sheet

http://git.or.cz/

Remember: git command --help

Global Git configuration is stored in $HOME/.gitconfig (git config --help)

Commands Sequence



Create

From existing data

```
cd ~/projects/myproject
git init
git add .
```

From existing repo

```
git clone ~/existing/repo ~/new/repo
git clone git://host.org/project.git
git clone ssh://you@host.org/proj.git
```

Show

Files changed in working directory

```
git status
```

Changes to tracked files

```
git diff
```

What changed between $ID1 and $ID2

```
git diff $id1  $id2
```

History of changes

```
git log
```

History of changes for file with diffs

```
git log -p $file $dir/ec/tory/
```

Who changed what and when in a file

```
git blame $file
```

A commit identified by $ID

```
git show $id
```

Concepts

Git Basics

master	: default development branch
origin	: default upstream repository
HEAD	: current branch
HEAD^	: parent of HEAD
HEAD~4	: the great-great grandparent of HEAD

Revert

Return to the last committed state

```
git reset --hard
```

⚠ you cannot undo a hard reset

Revert the last commit

```
git revert HEAD
```

Creates a new commit

Revert specific commit

```
git revert $id
```

Creates a new commit

Fix the last commit

```
git commit -a --amend
```

(after editing the broken files)

Checkout the $id version of a file

```
git checkout $id $file
```

Branch

Switch to the $id branch

```
git checkout $id
```

Update

Fetch latest changes from origin

```
git fetch
```

(but this does not merge them)

Pull latest changes from origin

```
git pull
```

(does a fetch followed by a merge)

Apply a patch that some sent you

```
git am -3 patch.mbox
```

(in case of a conflict, resolve and use git am --resolved)

Publish

Commit all your local changes

```
git commit -a
```

Prepare a patch for other developers

```
git format-patch origin
```

Push changes to origin

```
git push
```

Mark a version / milestone

```
git tag v1.0
```

Finding regressions

To view the merge conclicts

```
git diff
```

(complete conflict diff)

All together now

PIG http://www.slideshare.net/Mathias-Herberts/hadoop-pig-syntax-card

HDFS https://github.com/michiard/CLOUDS-LAB/blob/master/C-S.md

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

Python https://s3.amazonaws.com/quandl-static-content/Documents/Quandl+-+Pandas,+SciPy,+NumPy+Cheat+Sheet.pdf

Python http://www.astro.up.pt/~sousasag/Python_For_Astronomers/Python_qr.pdf

Java http://introcs.cs.princeton.edu/java/11cheatsheet/

Linux http://www.linuxstall.com/linux-command-line-tips-that-every-linux-user-should-know/

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

Git http://overapi.com/static/cs/git-cheat-sheet.pdf

Data Science Techniques

- Machine Learning
- Regression
- Logistic Regression
- K Means Clustering
- Association Analysis
- Decision Trees
- Text Mining
- Social Network Analysis
- Time Series Forecasting
- LTV and RFM Analysis
- Pareto Analysis

Machine Learning

Machine learning concerns the construction and study of systems that can learn from data. For example, a machine learning system could be trained on email messages to learn to distinguish between spam and non-spam messages

Supervised learning is the machine learning task of inferring a function from labeled training data.[1] The training data consist of a set of *training examples*. In supervised learning, each example is a *pair* consisting of an input object (typically a vector) and a desired output value (also called the *supervisory signal*).

In the terminology of machine learning, classification is considered an instance of supervised learning, i.e. learning where a training set of correctly identified observations is available.

In machine learning, the problem of **unsupervised learning** is that of trying to find hidden structure in unlabeled data. Since the examples given to the learner are unlabeled, there is no error or reward signal to evaluate a potential solution. This distinguishes unsupervised learning from supervised learning

The corresponding unsupervised procedure is known as *clustering* or cluster analysis, and involves grouping data into categories based on some measure of inherent similarity (e.g. the distance between instances, considered as vectors in a multi-dimensional vector space).

Machine Learning in Python

http://scikit-learn.org/stable/

Classification

Identifying to which set of categories a new observation belong to.

Applications: Spam detection, Image recognition.
Algorithms: *SVM, nearest neighbors, random forest, ...*
— *Examples*

Regression

Predicting a continuous value for a new example.

Applications: Drug response, Stock prices.
Algorithms: *SVR, ridge regression, Lasso, ...*
— *Examples*

Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes
Algorithms: *k-Means, spectral clustering, mean-shift, ...*
— *Examples*

Dimensionality reduction

Reducing the number of random variables to consider.

Applications: Visualization, Increased efficiency
Algorithms: *PCA, Isomap, non-negative matrix factorization.*
— *Examples*

Model selection

Comparing, validating and choosing parameters and models.

Goal: Improved accuracy via parameter tuning
Modules: *grid search, cross validation, metrics.*
— *Examples*

Preprocessing

Feature extraction and normalization.

Application: Transforming input data such as text for use with machine learning algorithms.
Modules: *preprocessing, feature extraction.*
— *Examples*

Classification

In machine learning and statistics, **classification** is the problem of identifying to which of a set of categories (sub-populations) a new observation belongs, on the basis of a training set of data containing observations (or instances) whose category membership is known.
The individual observations are analyzed into a set of quantifiable properties, known as various explanatory variables,*features*, etc.
These properties may variously be **categorical** (e.g. "A", "B", "AB" or "O", for blood type),
ordinal (e.g. "large", "medium" or "small"),
integer-valued (e.g. the number of occurrences of a part word in an email) or
real-valued (e.g. a measurement of blood pressure).

Some algorithms work only in terms of discrete data and require that real-valued or integer-valued data be *discretized* into groups (e.g. less than 5, between 5 and 10, or greater than 10).

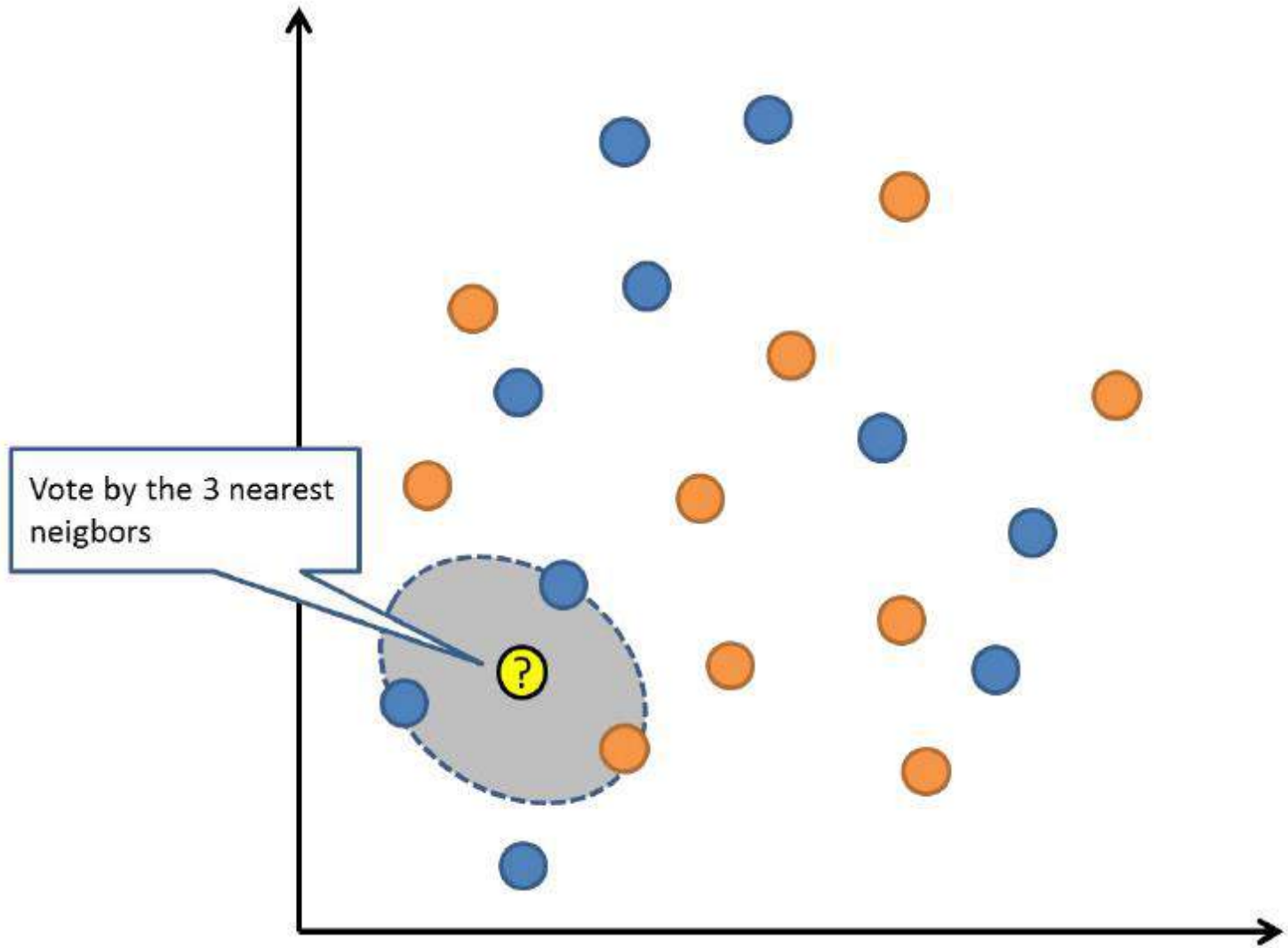
Regression

regression analysis is a statistical process for estimating the relationships among variables. It includes many techniques for modeling and analyzing several variables, when the focus is on the relationship between a dependent variable and one or more independent variables.

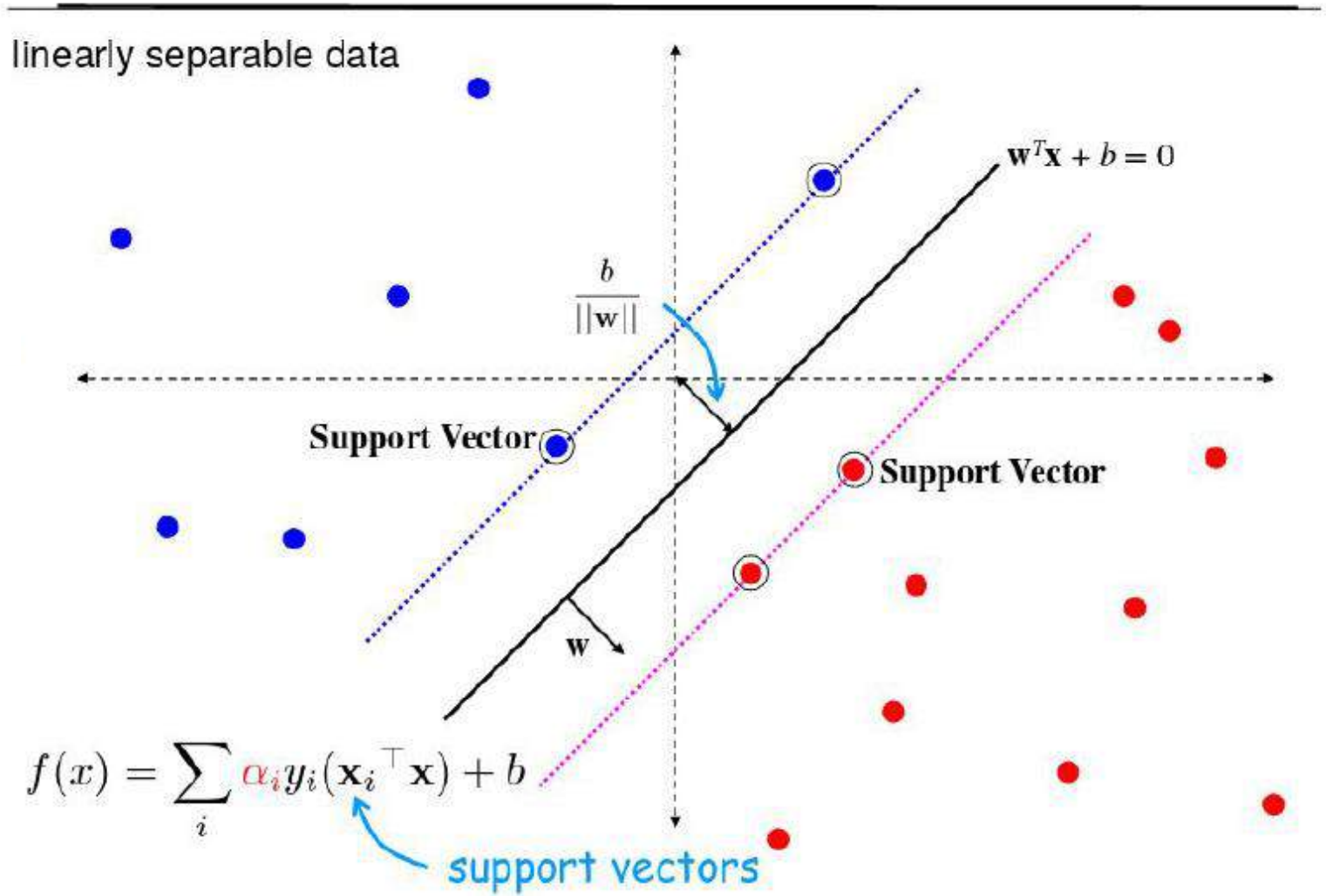
More specifically, regression analysis helps one understand how the typical value of the dependent variable (or 'criterion variable') changes when any one of the independent variables is varied, while the other independent variables are held fixed.

Most commonly, regression analysis estimates the conditional expectation of the dependent variable given the independent variables – that is, the average value of the dependent variable when the independent variables are fixed. Less commonly, the focus is on a quantile, or other location parameter of the conditional distribution of the dependent variable given the independent variables.

kNN



Support Vector Machines



Association Rules

http://en.wikipedia.org/wiki/Association_rule_learning

Based on the concept of strong rules, Rakesh Agrawal et al.[2] introduced association rules for discovering regularities between products in large-scale transaction data recorded by point-of-sale (POS) systems in supermarkets.

For example, the rule found in the sales data of a supermarket would indicate that if a customer buys onions and potatoes together, he or she is likely to also buy hamburger meat. Such information can be used as the basis for decisions about marketing activities such as, e.g., promotional pricing or product placements.

In addition to the above example from market basket analysis association rules are employed today in many application areas including Web usage mining, intrusion detection, Continuous production, and bioinformatics. As opposed to sequence mining, association rule learning typically does not consider **the order of items** either within a transaction or across transactions

Conecpts- Support, Confidence, Lift

In R

apriori() in arules package

In Python

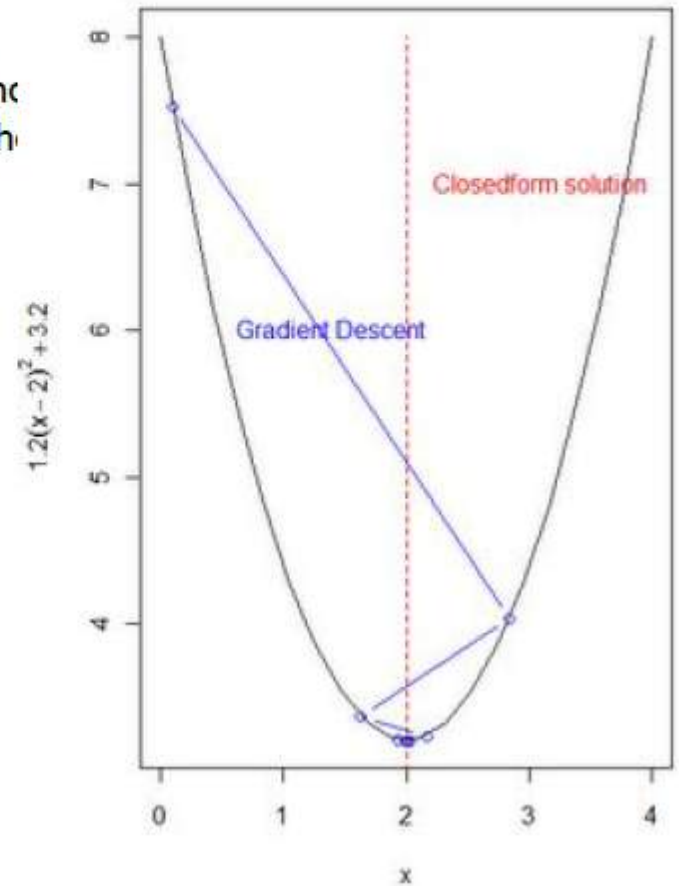
http://orange.biolab.si/docs/latest/reference/rst/Orange.associate/

$$\{onions, potatoes\} \Rightarrow \{burger\}$$

Gradient Descent

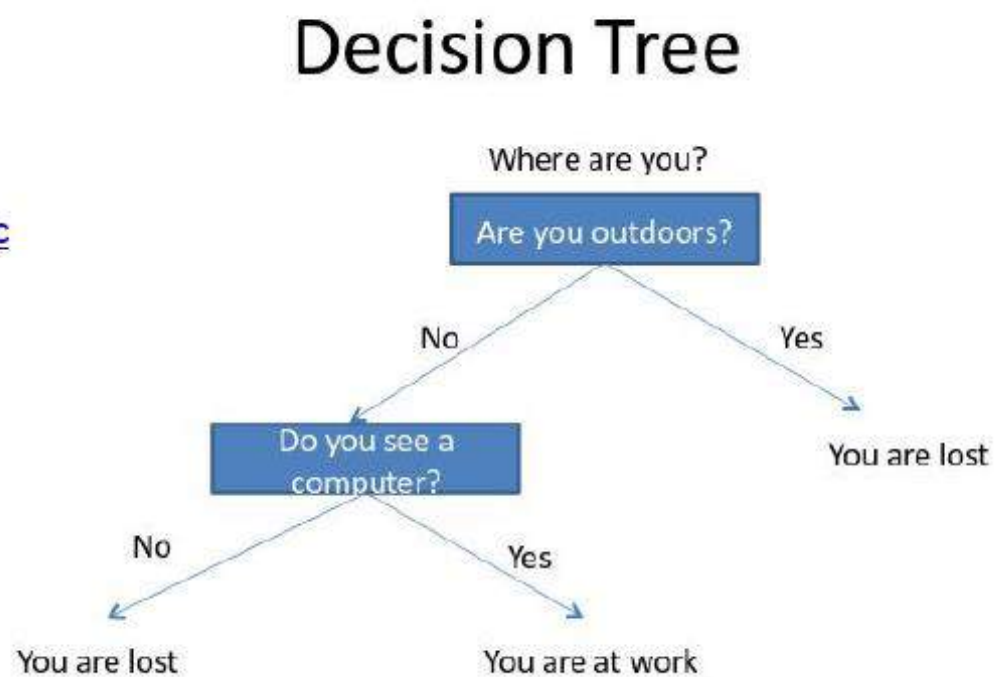
Gradient descent is a first-order iterative optimization algorithm. To find a local minimum of a func one takes steps proportional to the negative of the **gradient** (or of the approximate **gradient**) of th

Start at some x value, use derivative at that value to tell us which way to move, and repeat. Gradient descent.



Decision Trees

http://select.cs.cmu.edu/class/10701-F09/rec



Random Forest

Random Forests grows many classification trees. To classify a new object from an input vector, put the input vector down each of the trees in the forest. Each tree gives a classification, and we say the tree "votes" for that class. The forest chooses the classification having the most votes (over all the trees in the forest).

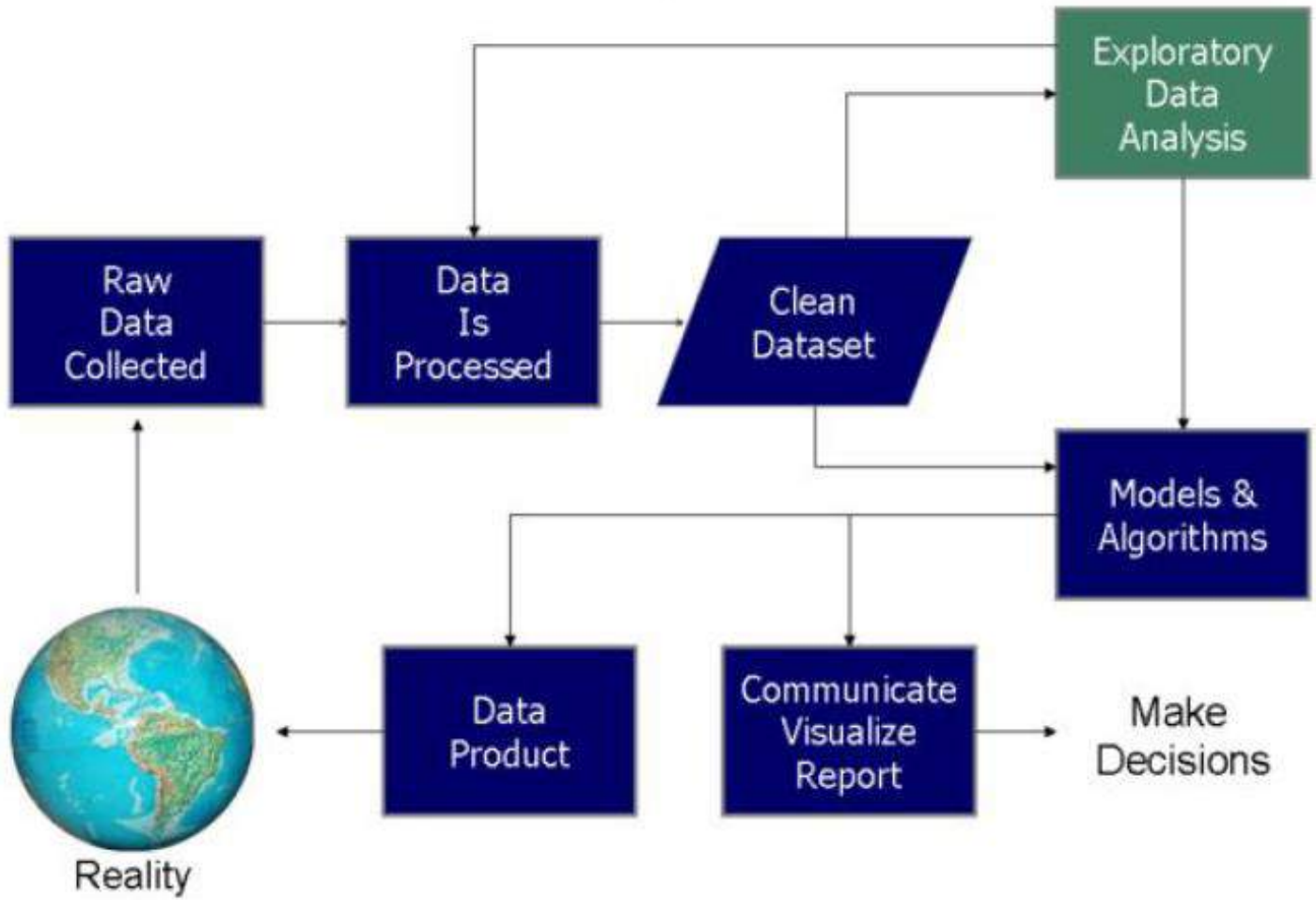
Each tree is grown as follows:

1. If the number of cases in the training set is N, sample N cases at random - but *with replacement*, from the original data. This sample will be the training set for growing the tree.
2. If there are M input variables, a number m<<M is specified such that at each node, m variables are selected at random out of the M and the best split on these m is used to split the node. The value of m is held constant during the forest growing.
3. Each tree is grown to the largest extent possible. There is no pruning.

In the original paper on random forests, it was shown that the forest error rate depends on two things:

- The *correlation* between any two trees in the forest. Increasing the correlation increases the forest error rate.
- The *strength* of each individual tree in the forest. A tree with a low error rate is a strong classifier. Increasing the strength of the individual trees decreases the forest error rate.

Data Science Process



Example Case Study

LTV Analytics

Breaking Down LTV Further

LTV WILL BE DIFFERENT FOR DIFFERENT KINDS OF CUSTOMERS

Step 2 in this graphic is intended to help you determine LTV as a total average (an average of all your customers). To do this, companies will typically average the data from randomly chosen customers (as shown in Step 1 above). Sometimes it's helpful to break down the average further and perform separate LTV calculations for different kinds of customers. Try and segment your customer base by total purchases over a long time period, and it will help you determine the LTV of a "good" customer versus an "average" one. This type of analysis will help you determine how much more you should pay in order to acquire a "good" customer. See chart below.

INVESTING IN "GOOD" CUSTOMERS

Companies should be worried about the lasting impact of "buying cheap customers." How likely are these customers to buy another product, or hang around for a few years? Sometimes it pays to invest in "good" customers. "Good" customers might cost more to acquire, but they'll likely be more profitable as well.

Let's say that the LTV of an "average" customer is $8,000, and the LTV of a "good" customer is $10,000. By subtracting the two LTVs, you can see that you might expect to pay $2,000 more to acquire "good" customers.

Customer Satisfaction Boosts LTV

One of the most effective ways to boost LTV is to increase customer satisfaction. Research has found that a 5% increase in customer retention can increase profits by 25% to 95%. The same study found that it costs six to seven times more to gain a new customer than to keep an existing one.

Life Time Value (LTV) will help us answer 3 fundamental questions:

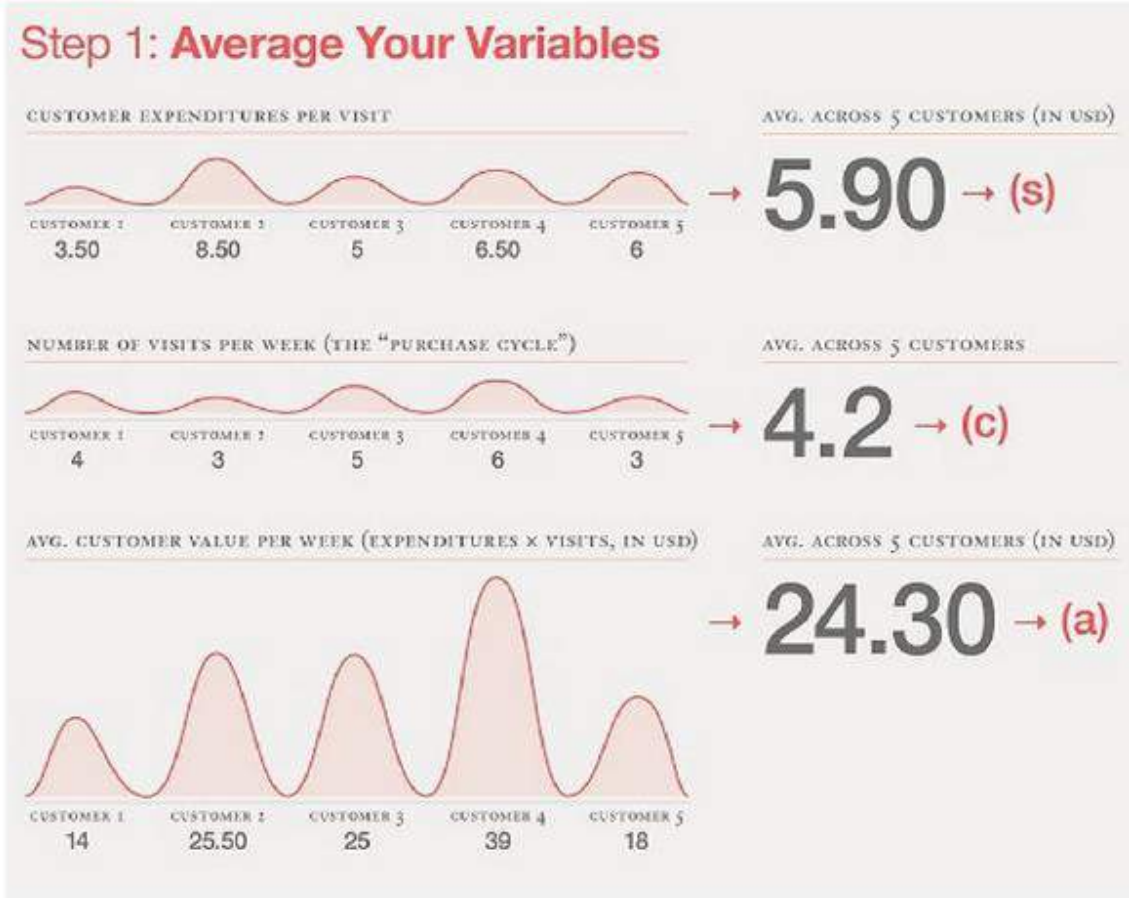
1. Did you pay enough to acquire customers from each marketing channel?

2. Did you acquire the best kind of customers?

3. How much could you spend on keeping them sweet with email and social media?

LTV Analytics :Case Study

https://blog.kissmetrics.com/how-to-calculate-lifetime-value/



LTV Analytics

https://blog.kissmetrics.com/how-to-calculate-lifetime-value/

Step 2: Calculate Lifetime Value (LTV)

CONSTANTS

t — **The Average Customer Lifespan** (how long someone remains a customer). In the case of Starbucks, the average customer lifespan is **20 years**.

r — **Customer Retention Rate.** The percentage of customers, who, over a given period of time, repurchase, when compared to an equal and preceding period of time. Starbucks: **75%**.

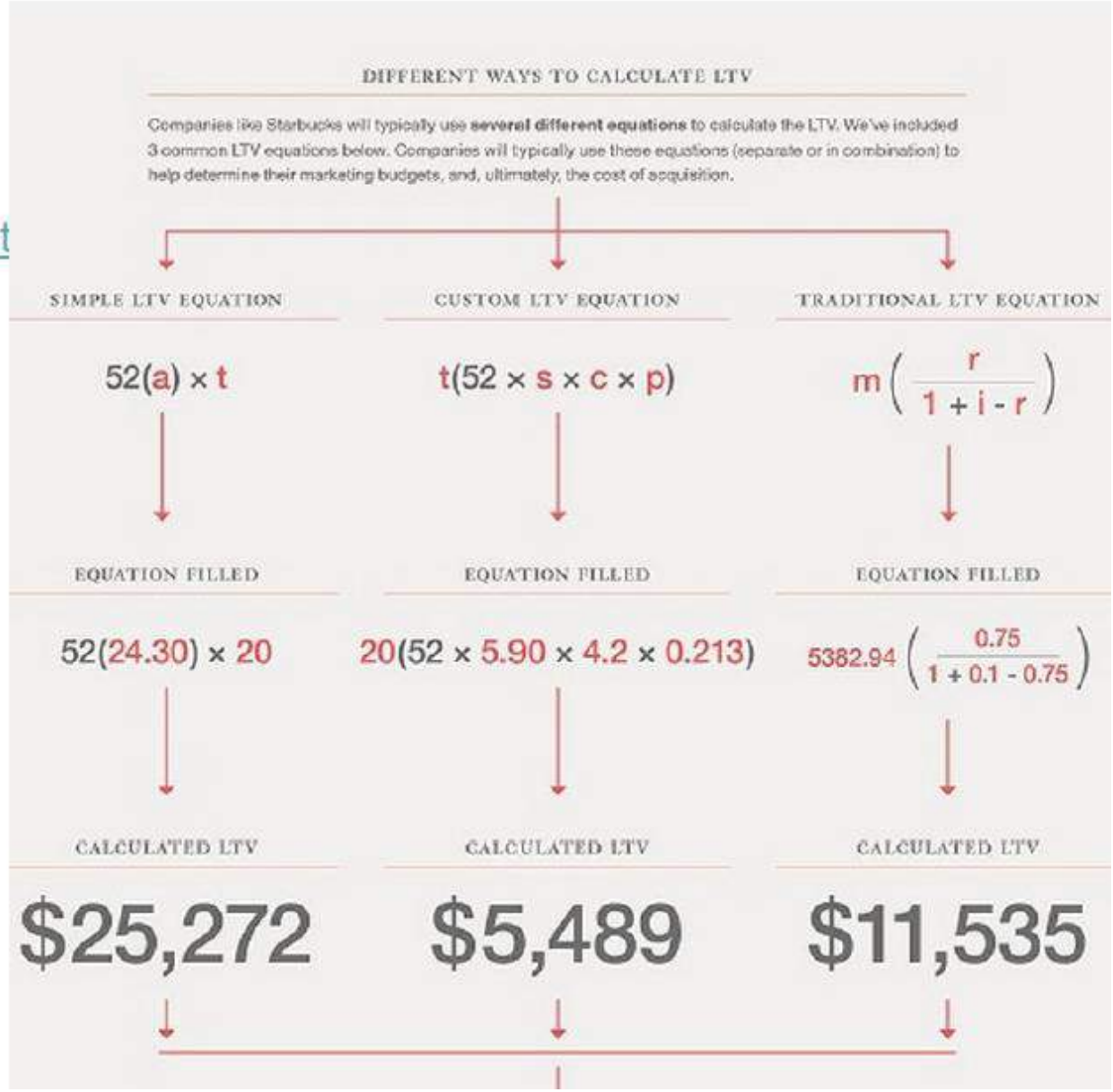
p — **Profit Margin per Customer.** Starbucks: **21.3%**.

i — **The Rate of Discount.** The "rate of discount" is the interest rate used in discounted cash flow analysis to determine the present value of future cash flows. Usually this number falls between 8% and 15%. Starbucks: **10%**.

m — **Avg. Gross Margin per Customer Lifespan.** Starbucks has a profit margin of 21.3% (see constant "p"). If the average customer spends $25,272 (see the "Simple LTV Equation" results below) during their time as a customer ("t"), Starbucks has gross margin per customer lifespan of **$5382.94**.

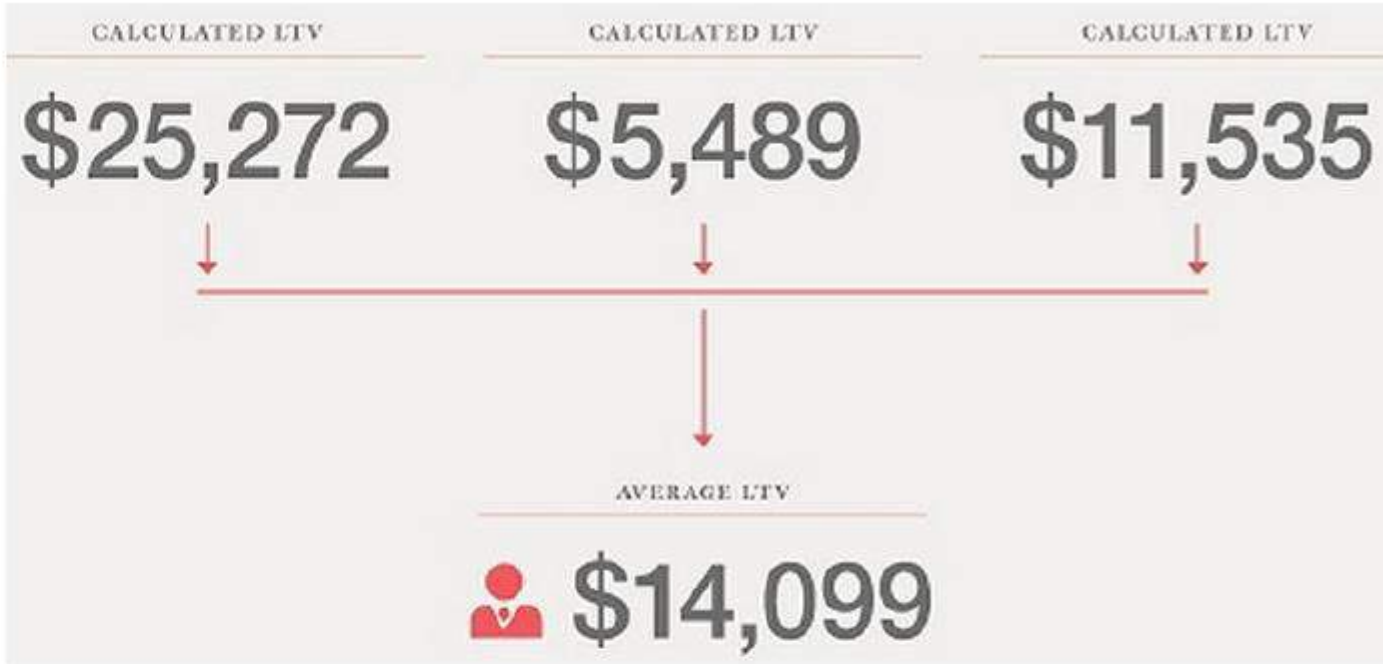
LTV Analytics

https://blog.kissmetrics.com/how-to-calculate-lifet



LTV Analytics

https://blog.kissmetrics.com/how-to-calculate-lifetime-value/



LTV Analytics

http://www.kaushik.net/avinash/analytics-tip-calculate-ltv-customer-lifetime-value/

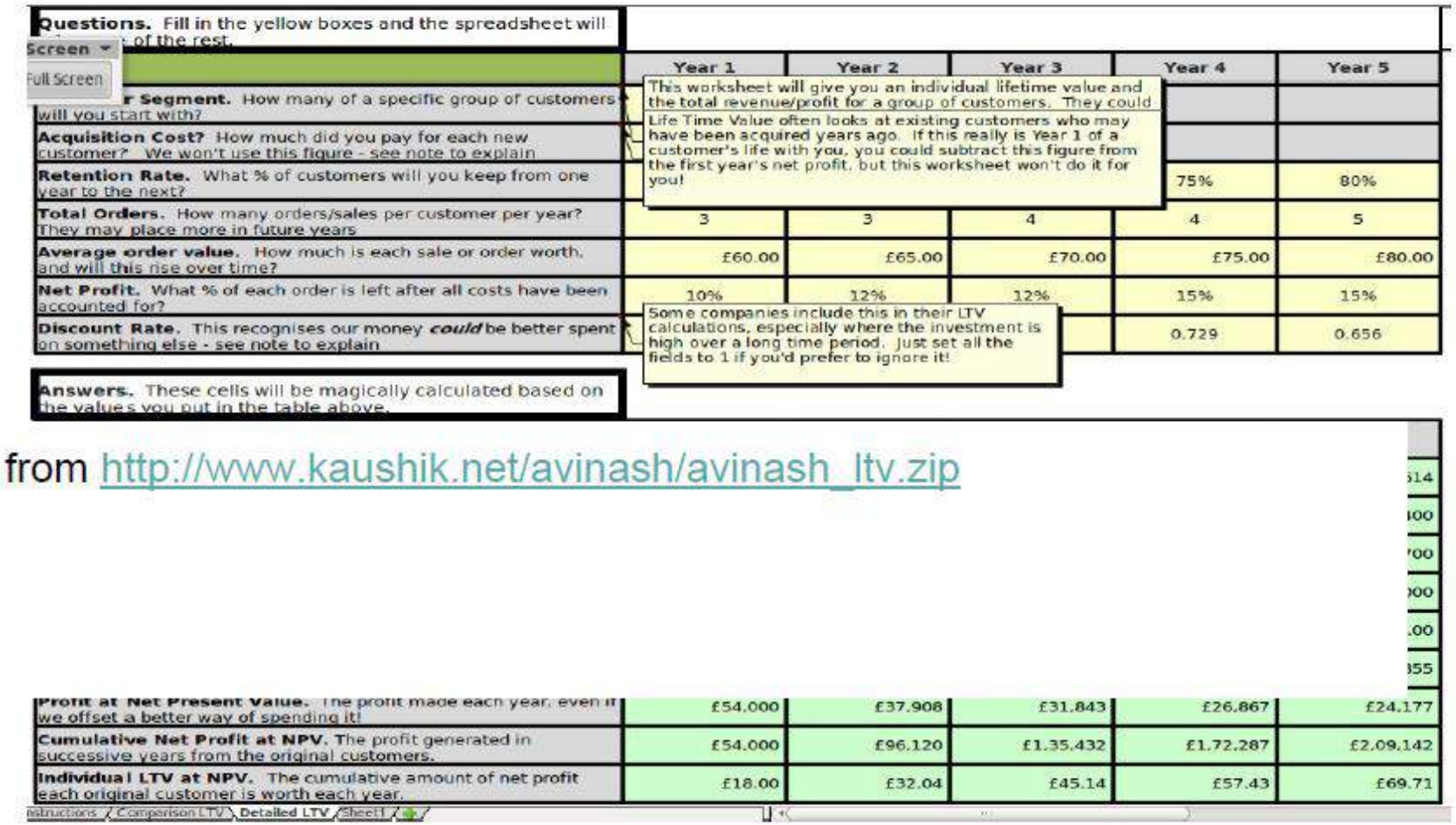
Questions. Fill in the yellow boxes and the spreadsheet will take care of the rest.

	Best Customers	Average Customers
Acquisition Cost .How much did you pay to acquire these customers?	£40.00	£12.00
Average order value. How much do they spend per order?	£92.00	£70.00
Orders per year? Quite simply, How many orders do they place per year?	5	2
Retention? How many years will they be customers for?	3	2
Net profit? What is the net profit percentage of goods sold?	10%	10%

Answers. These cells will be magically calculated based on the values you put in the table on the left

	Best Customers	Average Customers
Lifetime Gross Revenue	£1,380.00	£280.00
Life Time Net Profit	£98.00	£16.00

LTV



Download the zip file from http://www.kaushik.net/avinash/avinash_ltv.zip

Are you ready To use more Data Science


Thank
you!