# Programming Notes

Connor Johnson

August 9, 2011

# Contents

# Chapter 1

# Linux

(top)

## 1.1 Bash Commands

| | |
|---|---|
| `man` *utility* | manual page |
| `pwd` | print working directory |
| `ls` | list contents of a directory |
| `cd` | go to the home directory |
| `cd ..` | go up to the parent folder |
| `cd` *dir* | change directories |
| `mv` *file dir* | move a file to a directory |
| `mv` *file1 file2* | rename a file |
| `chmod a+rwx` | change permissions |
| `gedit` *file* `&` | open an editor |
| `xelatex f.tex` | compile .tex file |
| `evince f.pdf` | view .pdf file |
| `less` *f1* | view f1 |
| `sudo apt-get install` *file* | install file |
| `du -hs [file]` | size of current directory or file |

## 1.2 `cat`

`cat` can be used to view, concatenate, or create files. To view a file,

```
$ cat file.txt
```

To create a new file, `f3`, by concatenating files `f1` and `f2`,

```
$ cat f1 f2 > f3
```

To append a file, `f1`, to an existing file, `f0`,

```
$ cat f1 >> f0
```

To create a file and add data,

```
$ cat newfile.txt
```

Then type stuff, and hit enter. When you're done, hit `Ctrl-d` to save and exit.

## 1.3 ls

The `ls` is helpful and straightforward, but sometimes you need more information. The `-lh` flag gives you more data, in a *h*uman readable format. Adding the `-R` flag will show the sizes and contents of subdirectories, and `-a` will show hidden files.

## 1.4 ffmpeg

You'll probably need `video4linux`, `ffmpeg`, and `mplayer`.

To get information from a video file:

```
$ ffmpeg -i video.avi
```

To record a video:

```
$ ffmpeg -f video4linux2 -s 320x240 -i /dev/video0 out.mpg
```

To turn a collection of images into a video:

```
$ ffmpeg -f image2 -i image%d.jpg video.mpg
```

To turn a video into a collection of images:

```
$ ffmpeg -i video.mpg image%d.jpg
```

To record a raw video, maybe one of these? I think `input` needs to be something like `/dev/video0`.

```
$ ffmpeg -i input -f rawvideo -vcodec rawvideo output
$ ffmpeg -i input -f yuv4mpegpipe -vcodec rawvideo output
$ ffmpeg -i input -f rawvideo -vcodec yuv4mpegpipe output
```

To play a video:

```
$ mplayer out.mpg
```

## 1.5 grep

`grep` takes its name from the `ed` command, `g/re/p`. `grep` can be used to search for text within text files. `grep`, as a verb, was added to the OED in 2003 from the (less than) notable phrase, "You can't grep dead trees."

```
$ grep regex filename
```

## 1.6 Image Magick

This requires `imagemagick`. We can use the `convert` utility to convert images types, but it's still a better idea to use the correct format in the first place. This utility can also do blurring and all sorts of other things, like convolutions and Fourier transforms. Here is the syntax.

```
$ convert image.png image.eps
```

## 1.7 Interwebs

If the ethernet is down, do the following.

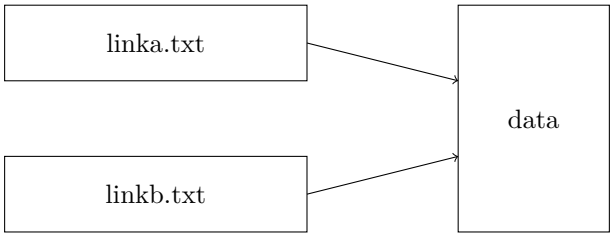
```
$ sudo ifconfig eth0 up
$ sudo dhclient eth0
```

USB wireless is.. another story.

## 1.8 Linking

### 1.8.1 Hard Links

Hard links allow more a file in a filesystem to have more than one name. When the original file is deleted, the linked files still exist and still contain data.




Here is how we create a hard link at the command line.

```
$ ln existing_file linked_file
```

If the linked file already exists, then the above won't work, but we can force it with the `-f` option.

```
$ ln -f existing_file linked_file
```

### 1.8.2 Symbolic Links

If we want a file to have multiple names we can create symbolic links. When the linked file is changed, the orginal is changed also, and vice versa. The only thing is that the files must have different names.

```
$ ln -s existing_file linked_file
```

## 1.9 PBS Scripts

To log into the cluster, type:

```
$ ssh -p22 clj033@cerberus2.hpc.latech.edu
```

The `-p` flag specifies the port. Caveat: the `scp` utility uses a `-P` flag for the same thing.
Large data sets should be stored in `/work/clj033`.
To see what software is installed, use

```
$ softenv | less
```

To use `easy_install`,

```
$ easy_install --prefix=/user/clj033/.local
```

For things that are not `easy_installer`

```
$ ./configure --help
```

This is a good, basic script, named `ctrl.pbs`:

```
#PBS -N $name
#PBS -q single
#PBS -l nodes=1:ppn=12,mem=48gb
#PBS -l walltime=12:00:00
#PBS -M connor.labaume.johnson@gmail.com
#PBS -m abe
#PBS -j oe
#PBS -o name.out
#PBS -e name.err

cd /work/clj033

python program.py args
```

To submit a job:

```
$ qsub ctrl.pbs
```

To check on the status of all of the jobs:

```
$ qstat
```

To check on the status of only *your* jobs:

```
$ qstat -u username
```

To delete a job:

```
$ qdel $jobnumber
```

An exit status of 0 is a good thing; an exit status of 1 is a bad thing.

## 1.10 USB

### 1.10.1 Format

```
$ fdisk -l
$ mkfs -t ext3 /dev/sdc1
```

### 1.10.2 Mount

First, find out where the USB is using the `fdisk` command:

```
$ fdisk -l

Disk /dev/sdc: 4004 MB, 4004511744 bytes
116 heads, 51 sectors/track, 1322 cylinders
Units = cylinders of 5916 * 512 = 3028992 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00000000

   Device Boot      Start         End      Blocks   Id  System
/dev/sdc1               1        1322     3910450+  83  Linux
```

Alternatively, you can use this command:

```
$ sudo ls -l /dev/disk/by-id/*usb*
```

In this example, the USB is located at `/dev/sdc1`. Then we create a directory where we want to mount the USB.

```
$ mkdir /mnt/usb
```

Then we edit `/etc/fstab` and add:

```
/dev/sdc1   /mnt/usb   ext4    defaults    0     0
```

And then we mount using:

```
$ mount -a
```

To verifty that the drvier is mounted, use:

```
$ mount
```

### 1.10.3 Unmount

In order to unmount, use the following:

```
$ umount /mnt/usb
```

Then you can run `mount` again in order to verify.

## 1.11 Terminator Terminal

| | |
|---|---|
| `Ctrl-Shift-E` | split screen vertically |
| `Ctrl-Shift-O` | split screen horizontally |
| `Ctrl-Shift-P` | go to previous window |
| `Ctrl-Shift-N` | go to next window |
| `Ctrl-Shift-W` | close all other windows |
| `Ctrl-Shift-T` | create a new tab |
| `Ctrl-PageUp` | toggle tab |
| `Ctrl-Shift-W` | close current terminal |
| `Ctrl-Shift-Q` | quit |
| `F11` | widescreen* |

## 1.12 Virtual Machine

Sometimes it's nice to have a virtual machine–like when you're using Windows and you need to do real work.

### 1.12.1 Introductory Steps

From within the Virtual Box machine, open synaptic and load the following four packages:

- `virtual-box-dkms`
- `virtual-box-source`
- `virtual-box-utils`
- `virtual-box-x11`

### 1.12.2 Sharing Folders

I had to watch, like, five YouTube videos on this before I got it.

1. Click on the `General` tab.
2. Click on the `Shared Folders` link in the sidebar.
3. Click on the *plus-file* icon. It's a file icon with a plus on it.
4. Click on *Make Permanent*.
5. Start a guest OS.
6. In the guest OS, click on `Devices` → `Shared Folders`.
7. Make sure everything looks okay.
8. There will be a name of the directory on the host, and the path.
9. Create a directory in the guest for the shared data.
10. Run the following:

```
$ sudo mount -t vboxsf dirname /home/user/shared
```

Here, `dirname` is the name of the directory on the host, and `shared` is the new directory on the guest.

## 1.13 `zip` and `unzip`

Creates the archive `data.zip` from the file `data.dat`.

```
$ zip data.zip data.dat
```

Creates the archive `data.zip` and puts all the files in the current directory in it in compressed form.

```
$ zip data *
```

To zip up an entire directory (including all subdirectories)

```
$ zip -r data *
```

To list all files from data.zip

```
$ unzip -l data.zip
```

Use unzip to extract all files of the archive data.zip into the current directory

```
$ unzip data.zip
```

To extract all files into the /dir directory:

```
$ unzip data.zip -d /dir
```

To extract the file called file.odt from data.zip

```
$ unzip data.zip file.odt
```

## 1.14 ssh and scp

To access the server

```
$ ssh username@beta.latech.edu
```

Alternatively, you can use the IP address,

```
$ ssh username@138.47.102.666
```

Cut and copy the output of `pwd` and then, from your local computer, you can copy files over a secured connection to the remote computer by,

```
$ scp <document> username@beta.latech.edu:<pwd>/username/public_html
```

The part after the colon is the target directory in the remote location.

To copy files from a remote computer to your local computer you do the following:

```
$ scp username@cerberus2.hpc.latech.edu:/path/to/remote/doc /local/destination
```

Again, the path after the colon is the path to the file on the remote computer, and the path at the very end is destination on your local compter.

## 1.15 sshfs

This creates a file system (or a subset of it) from the remote computer on your local computer. First you install `sshfs`, then you make an empty directory on your local computer, using `mkdir`.

```
$ mkdir fs
$ sshfs username@ip-addy:/remote/desktop fs
```

When you're done, you'll need to unmount the file system on your local computer,

```
$ fusermount -u fs
```

## 1.16 File Recovery

Use dd and foremost

## 1.17 RapidMiner

Go to the rapidminer/lib directory and then type the following into the command line.

```
$ java -jar rapidminer.jar
```

## 1.18 Weka

Go to the directory containing the Weka jar-file, and then type the following into the command line.

```
$ java -jar weka.jar
```

## 1.19 Cantor and Maxima

Download maxima and cantor, and then use the following.

```
$ cantor -backend=maxima &
```

Use Shift-Enter to evaluate statements.

# Chapter 2

# LaTeX

(top)

## 2.1 Introduction

You'll need to also get the following packages:

- `texlive-latex-base`
- `texlive-latex-extra`
- `texlive-latex-recommended`
- `texlive-fonts-recommended`
- `texlive-latex-extra`
- `texlive-xetex`

## 2.2 Symbols

| | | | |
|---|---|---|---|
| $*$ `\ast` | $\pm$ `\pm` | $\perp$ `\perp` | $\prod$ `\prod` |
| $\cdot$ `\cdot` | $\mp$ `\mp` | $\propto$ `\propto` | $\partial$ `\partial` |
| $\circ$ `\circ` | $\leftarrow$ `\leftarrow` | $\sim$ `\sim` | $\Delta$ `\Delta` |
| $\bigcirc$ `\bigcirc` | $\rightarrow$ `\rightarrow` | $\approx$ `\approx` | $\nabla$ `\nabla` |
| $\cap$ `\cap` | $\Leftarrow$ `\Leftarrow` | $\vdash$ `\vdash` | $\int$ `\int` |
| $\cup$ `\cup` | $\Rightarrow$ `\Rightarrow` | $\vert$ `\vert` | $\oint$ `\oint` |
| $\vee$ `\vee` | $\subset$ `\subset` | $\lfloor$ `\lfloor` | $\iint$ `\iint` |
| $\wedge$ `\wedge` | $\subseteq$ `\subseteq` | $\lceil$ `\lceil` | $\infty$ `\infty` |
| $\geq$ `\ge` | $\in$ `\in` | $\log_a(x)$ `\log_{a}(x)` | $\exists$ `\exists` |
| $\leq$ `\le` | $\notin$ `\notin` | $\ln(x)$ `\ln(x)` | $\forall$ `\forall` |
| $\neq$ `\ne` | $\parallel$ `\parallel` | $\sum$ `\sum` | |

The `amssymb` package includes a function that lets you make symbols bold as `\boldsymbol{·}`.

## 2.3 Algorithms

Download `pseudocode.sty` and place it in `/usr/share/texmf/tex/latex/`. Then go to that directory and execute `sudo texhash`. That should do the trick.

Use the following in the preamble:

```
\usepackage{pseudocode}
```

And then produce algorithms as:

```
\begin{pseudocode}[doublebox]{Foo}{x,y}
\FOR i \GETS 0 \TO 10 \DO
\BEGIN
  \mbox{apples} \\
  \mbox{oranges} \GETS \mbox{apples} \\
  \mbox{bees} \GETS \mbox{oranges} \\
\END \\
\RETURN{fruit}
\end{pseudocode}
```

This code returns the following listing:

**Algorithm 2.3.1:** FOO$(x, y)$

**for** $i \leftarrow 0$ **to** 10
**do** $\begin{cases} \text{apples} \\ \text{oranges} \leftarrow \text{apples} \\ \text{bees} \leftarrow \text{oranges} \end{cases}$
**return** $(fruit)$

Instead of `[doublebox]` for the framestyle, we could use any of the following options: `shadowbox`, `ovalbox`, `Ovalbox`, `framebox`, `plain`, `ruled`, `display`, or simply none at all.

We can do an `IF-THEN` structure in the following manner:

```
\begin{pseudocode}[ruled]{Foobar}{x}
\IF \mbox{awesome}
\THEN
  \BEGIN
    \mbox{first stmt}\\
    \mbox{some stmts}\\
    \mbox{other stmts}
  \END
\ELSEIF \mbox{not awesome}
\THEN
  \BEGIN
    \mbox{alt stmt}\\
    \mbox{other stmts}\\
    \mbox{more stmts}
  \END
\ELSEIF \mbox{I missed something}
\THEN \mbox{do something else}
\ELSE \mbox{default actions}
\end{pseudocode}
```

**Algorithm 2.3.2:** FOOBAR($x$)

**if** awesome
**then** { first stmt; some stmts; other stmts
**else if** not awesome
**then** { alt stmt; other stmts; more stmts
**else if** I missed something
**then** do something else
**else** default actions

## 2.4 Aymptote

We can produce vector drawings using asymptote. If we insert the following in the preamble:

```
\usepackage[pdftex]{graphicx}
\usepackage[inline]{asymptote}
```

We can use the following in our document block:

```
\begin{figure}
\begin{asy}
size (3cm);
draw (unitcircle);
\end{asy}
\caption{Embedded Asymptote figures are easy!}
\label{fig:embedded}
\end{figure}
```

We then compile our document with the following commands:

```
$ pdflatex filename.tex
$ asy filename.asy
$ evince filename.pdf
$
```

## 2.5 Basic Format

```
\documentclass[11pt]{report}

\title{}
\author{}
\date{}

\usepackage{amsmath}
\usepackage{amsfonts}

%\pagestyle{empty} % turn off page numbers
%\usepackage{fullpage} % use standard one-inch margins
%\setlength{\parindent}{0pt} % to kill indentation
\usepackage[colorlinks]{hyperref}
%\usepackage[disable]{todonotes}
```

```
\usepackage[backgroundcolor=pink,bordercolor=red,linecolor=red,]{todonotes}
\usepackage[pdftex]{graphicx}

\begin{document}

\listoftodos
\maketitle

\section*{Abstract}

\section*{Introduction}

\section*{Methodology}

\section*{Algorithms}

\section*{Results}

\section*{Discussion}

\bibliographystyle{plain}
\bibliography{}

\end{document}
```

## 2.6 Beamer

The beamer class allows you to to produce slides in LaTeX. Below is a sample:

```
% sample.tex
\documentclass[xcolor=dvipsnames]{beamer}
%\usecolortheme[named=Red]{structure}
\usecolortheme[RGB={205,173,0}]{structure}
\usetheme[height=8mm]{Rochester}
%\usetheme{default}
%\usetheme{Singapore}

\title[K-Means in SQL]{K-Means Clustering in a RDBMS Using SQL}
\author[C. Johnson]{Connor Johnson}
\institute[LaTech CAM]{
  Computational Analysis and Modeling\\
  Louisiana Tech University\\
  Ruston, Louisiana 71270\\
  \texttt{cjohnson@latech.edu}
}
\date[January 2012]{January 9, 2012}

\begin{document}

\begin{frame}[plain]
  \titlepage
\end{frame}

\begin{frame}{A sample slide}

\begin{theorem}[The Poincar\'e inequality]
Suppose $\Omega\in\mathbf{R}^n$ is a bounded domain with smooth
boundary.  Then there exists a $\lambda>0$, depending only on
$\Omega$, such that for any function $f$ in the Sobolev space
$H^1_0(\Omega)$ we have:

\[
  \int_\Omega |\nabla u|^2 \,dx \ge
```

```
  \lambda \int_\Omega |u|^2 \,dx .
\]
\end{theorem}

Here is what \emph{itemized} and \emph{enumerated} lists look like:

\begin{columns}
  \begin{column}{0.45\textwidth}
  \begin{itemize}
    \item itemized item 1
    \item itemized item 2
    \item itemized item 3
  \end{itemize}
  \end{column}

  \begin{column}{0.45\textwidth}
  \begin{enumerate}
    \item enumerated item 1
    \item enumerated item 2
    \item enumerated item 3
  \end{enumerate}
  \end{column}
\end{columns}

\end{frame}

\end{document}
```

You can also print multiple slides per page, with lines for notes, by inserting the following into the preamble:

```
\usepackage{handoutWithNotes}
\pgfpagesuselayout{4 on 1 with notes}[a4paper,border shrink=5mm]
```

You can change the color of text to the theme color using:

```
Text text text \structure{colored text here}.
```

You can add new theorem environments by declaring the new environment in the preamble as

```
\newtheorem{new_thm_label}{new_thm_name}
```

And then using it in the body as

```
\begin{new_thm_label}
blah blah blah
\end{new_thm_label}
```

## 2.7 Bibliography

Cite sources in the body by using `\cite{...}`. Note the last two lines at the end of the body.

```
\begin{document }
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor
incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis
nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.
Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu
fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in
culpa qui officia deserunt mollit anim id est laborum.\cite{ cicero }
\bibliographystyle{plain}
\bibliography{myrefs}
\end{ document }
```

In another document, called myrefs.bib, include the following information:

```
@ARTICLE{ grinsted,
AUTHOR={Grinsted, A. C. and Moore, J. C. and Jevrejeva, S.},
TITLE={Application of the Cross Wavelet Transform and Wavelet Coherence to Geophysical Time Series.},
JOURNAL={Nonlinear Processes in Geophysics},
VOLUME={11},
PAGES={561--566},
YEAR={2004},
}

@ARTICLE{ torrencecompo,
AUTHOR={Torrence, C. and Compo, G. P.},
TITLE={A Practical Guide to Wavelet Analysis},
JOURNAL={Bulletins of the American Meteorological Society},
VOLUME={79},
PAGES={61--78},
YEAR={1998},
}

@ARTICLE{ schaumstocker,
AUTHOR={Schaum, A. and Stocker, A.},
TITLE={Advanced Algorithms for Autonomous Hyperspectral Change Detection},
JOURNAL={Applied Imagery Pattern Recognition Workshop},
PAGES={33--38},
YEAR={2004},
}
```

Then compile by entering the following commands:

```
$ pdflatex mydocument
$ bibtex mydocument
$ pdflatex mydocument
$ pdflatex mydocument
$ evince mydocument
$
```

Here is a list of standard entry types:

| | | | |
|---|---|---|---|
| `@article` | `@inbook` | `@mastersthesis` | `@techreport` |
| `@book` | `@incollection` | `@misc` | `@unpublished` |
| `@booklet` | `@inproceedings` | `@phdthesis` | |
| `@conference` | `@manual` | `@proceedings` | |

Here is a list of standard entry fields:

| | | | |
|---|---|---|---|
| address | editor | number | type |
| annotate | howpublished | organization | volume |
| author | institution | pages | year |
| booktitle | journal | publisher | |
| chapter | key | school | |
| crossref | month | series | |
| edition | note | title | |

## 2.8 Boxing

We may use `\boxed` to box equations. This is like using `\fbox`, except that the contents are in math mode.

```
\boxed{ x^2 - 5 x + 6 = 0 }
```

If we want to box some text we can use `\fbox` to do so automatically, or we can specify a width and use `\framebox`.

```
\fbox{ text }
```

```
\framebox[ width ][ position ]{ text } % here position is c l r
```

## 2.9 Code Listings

In the preamble we include the package:

```
\usepackage{listings}
```

And then in the body we create a listing as follows:

```
\begin{lstlisting}[frame=single,basicstyle=\ttfamily\scriptsize]
  some code here..
\end{lstlisting}
```

Alternatively, we can define the syle and use the same listing style throughout:

```
\lstset{language=Python,basicstyle=\ttfamily,frame=tblr,tabsize=4}
\begin{lstlisting}
for i in range(10):
    print i
\end{lstlisting}
```

## 2.10 Compiling

To compile from the command line you do the following:

```
$ latex file.tex
$ dvips file.dvi
$ ps2pdf file.ps
$ evince file.pdf
```

Alternatively, you may compile using `pdflatex`:

```
$ pdflatex file.tex
$ evince file.pdf
```

You may also compile using `xelatex`, but if you are using the `graphicx` package, you must change the `pdflatex` option to `xetex`, and you must include the `fontspec` package. Therefore, your preamble should look like:

```
\documentclass[10pt]{report}

\usepackage{amsmath}
\usepackage{amsfonts}
\usepackage{amssymb}

\usepackage[xetex]{graphicx}

%%% This is for compiling using xelatex %%%
\usepackage{fontspec}
```

We then compile as follows:

```
$ xelatex file.tex
$ evince file.pdf
```

## 2.11 Emacs Auto-completion

To complete a `\begin{env}` expression, you can type C-c C-e, or you may use C-c C-o.

## 2.12 Equations

We may use the following to number formulas.

```
\begin{equation} \label{eq:solve}
x^2 - 5 x + 6 = 0
\end{equation}
```

and now we can reference the equation in the text using: `\eqref{eq:solve}`.

If we want to do multi-line equations we can use the `align` or `align*` environments. (`align*` supresses equation numbering.) We use the ampersands to align the text at the equal signs, and we use the \notag tag before the line break to disable the numbering on that line.

```
\begin{align} \label{whammy}
H( Y \vert X ) =& \sum \limits_{ x \in X } p(x) H( Y \vert X=x ) \notag \\
=& - \sum\limits_{ x \in X } \sum\limits_{ y \in Y } p(x,y) \ln p( y \vert x )
\end{align}
```

The above produces:

$$\begin{aligned} H(Y|X) &= \sum_{x \in X} p(x) H(Y|X=x) \\ &= -\sum_{x \in X} \sum_{y \in Y} p(x,y) \ln p(y|x) \end{aligned} \tag{2.1}$$

If we need to specify cases, we can use the cases environment.

```
\begin{equation*}
f(x) = \begin{cases} x^{2} & \mbox{if } x < 0 \\ \ln(x) & \mbox{if } x \le 0 \end{cases}
\end{equation*}
```

$$f(x) = \begin{cases} x^2 & \text{if } x \leq 0 \\ \ln(x) & \text{if } x > 0 \end{cases}$$

## 2.13 Figures

To create a figure you do the following:

```
\begin{figure}\label{<put your label here>}
<figure content>
\caption{<explanation>}
\end{figure}
```

To make a figure span two columns, if you are using two columns, use an asterix:

```
\begin{figure*}[htb]
<figure content>
\caption{<explanation>}
\label{<put your label here>}
\end{figure*}
```

Note that [htb] in the box above means here, top, or bottom, i.e., put this figure right here, or the top if it won't fit here, or lastly, the bottom.

The following code can be used to place two figures side-by-side by creating a `minipage`

```
\begin{figure}[ht]
  \begin{minipage}[b]{0.5\linewidth}
    \centering
    \includegraphics[scale=1]{filename1}
    \caption{default}
    \label{fig:figure1}
  \end{minipage}
  \hspace{0.5cm}
  \begin{minipage}[b]{0.5\linewidth}
    \centering
    \includegraphics[scale=1]{filename2}
    \caption{default}
    \label{fig:figure2}
  \end{minipage}
\end{figure}
```

Note that for JPEG images the graphics package is required: \usepackage{graphics}.

The same local column effect can be achieved for tables. The following code shows you how:

```
\begin{table}[ht]
  \begin{minipage}[b]{0.5\linewidth}\centering
    \begin{tabular}{|c|c|c|}
      \hline
      1&1&1\\
      \hline
      2&2&2\\
      \hline
    \end{tabular}
  \end{minipage}
  \hspace{0.5cm}
  \begin{minipage}[b]{0.5\linewidth}
    \centering
    \begin{tabular}{|c|c|c|}
      \hline
      1&1&1\\
      \hline
      2&2&2\\
      \hline
    \end{tabular}
  \end{minipage}
\end{table}
```

You can also have more than two column simply by adding another \minipage in between the table-command and reduce the width of each \minipage (0.33 \linewidth for three columns) in addition to the \hspace. LaTex will automaticaly place objects onto the next line, if space is not sufficient.

## 2.14 Flow Charts

We can use the following to produce flowcharts:

***figure***

```
\documentclass{article}
\usepackage[latin1]{inputenc}
\usepackage{tikz}
\usetikzlibrary{shapes,arrows}
\begin{document}
\pagestyle{empty}
% Define block styles
```

```
\tikzstyle{decision} = [diamond, draw, fill=blue!20,
text width=4.5em, text badly centered, node distance=3cm, inner sep=0pt]
\tikzstyle{block} = [rectangle, draw, fill=blue!20,
text width=5em, text centered, rounded corners, minimum height=4em]
\tikzstyle{line} = [draw, -latex']
\tikzstyle{cloud} = [draw, ellipse,fill=red!20, node distance=3cm,
minimum height=2em]
\begin{tikzpicture}[node distance = 2cm, auto]
% Place nodes
\node [block] (init) {initialize model};
\node [cloud, left of=init] (expert) {expert};
\node [cloud, right of=init] (system) {system};
\node [block, below of=init] (identify) {identify candidate models};
\node [block, below of=identify] (evaluate) {evaluate candidate models};
\node [block, left of=evaluate, node distance=3cm] (update) {update model};
\node [decision, below of=evaluate] (decide) {is best candidate better?};
\node [block, below of=decide, node distance=3cm] (stop) {stop};
% Draw edges
\path [line] (init) -- (identify);
\path [line] (identify) -- (evaluate);
\path [line] (evaluate) -- (decide);
\path [line] (decide) -| node [near start] {yes} (update);
\path [line] (update) |- (identify);
\path [line] (decide) -- node {no}(stop);
\path [line,dashed] (expert) -- (init);
\path [line,dashed] (system) -- (init);
\path [line,dashed] (system) |- (evaluate);
\end{tikzpicture}
\end{document}
```

## 2.15 Font Family

We can use any TrueType font we have installed on the computer. In this example, we use Roboto-Light.

```
\documentclass[10pt]{report}
\usepackage{fontspec}
\setromanfont{Roboto-Light}
\begin{document}
\section{ ROBOTO}
Testing Roboto Light
\end{document}
```

## 2.16 Font Sizes

```
\tiny
\scriptsize
\footnotesize
\small
\normalsize
\large
\Large
\LARGE
\huge
\Huge
```

## 2.17 IEEE Style

You'll need to include the documents `IEEEtran.cls` and `IEEEtran.bst` in you current directory, make a `.bib` file and then compile as:

```
$ pdflatex mydoc ; bibtex mydoc ; pdflatex mydoc ; pdflatex mydoc
$ evince mydoc.pdf
```

## 2.18 Listst

We can form a numbered list using `enumerate`.

```
\begin{enumerate}
\item One
\item Two
\item Three
\end{enumerate}
```

Or an unnumbered list using `itemize`.

```
\begin{itemize}
\item One
\item Two
\item Three
\end{itemize}
```

## 2.19 Matrices

$$\left| \begin{array}{cc} x_{11} & x_{12} \\ x_{21} & x_{22} \end{array} \right| \tag{2.2}$$

```
\left| \begin{array}{cc} x_{11} & x_{12} \\ x_{21} & x_{22} \end{array} \right|
```

$$\begin{vmatrix} x & y \\ z & v \end{vmatrix} \tag{2.3}$$

```
\begin{vmatrix} x & y \\ z & v \end{vmatrix}
```

$$\begin{Vmatrix} x & y \\ z & v \end{Vmatrix} \tag{2.4}$$

```
\begin{Vmatrix} x & y \\ z & v \end{Vmatrix}
```

$$\begin{bmatrix} x & y \\ z & v \end{bmatrix} \tag{2.5}$$

```
\begin{bmatrix} x & y \\ z & v \end{bmatrix}
```

$$\begin{Bmatrix} x & y \\ z & v \end{Bmatrix} \tag{2.6}$$

```
\begin{Bmatrix} x & y \\ z & v \end{Bmatrix}
```

$$\begin{pmatrix} x & y \\ z & v \end{pmatrix} \tag{2.7}$$

```
\begin{pmatrix} x & y \\ z & v \end{pmatrix}
```

## 2.20 Multiple Columns

So, this is the basic format. If you want two columns throughout the whole document, you use the following for the first line:

```
\documentclass[11pt,twocolumn]{report}
```

If you want multiple columns in a specific region, use this:

```
\documentclass{article}
\usepackage{multicol}

\begin{document}
\begin{multicols}{2}
\begin{enumerate}
    \item a
    \item b
    \item c
    \item d
    \item e
    \item f
\end{enumerate}
\end{multicols}
\end{document}
```

## 2.21 New Commands

Put the listing below into the preamble. The number of variables in the code below should be 2, and the variables, in order of appearance will be accessed by a hash mark and their order of appearance.

```
\newcommand{\name}[no_variables]{commandcode #1 #2 } % to define macros
```

## 2.22 Packages

For Ubuntu 10.04 LTS, to add a package, download the zip-file of the package and extract it. The extracted package file should show up on the Desktop somewhere. This package file should have a .sty file and a .dtx file. Move the package file to /usr/share/texmf/tex/latex/

```
$ sudo mv ~/Desktop/pkg /usr/share/texmf/tex/latex
$ cd /usr/share/texmf/tex/latex/pkg
$ sudo texhash
$ cd
```

Now, pdflatex and latex should be able to see your new packages.

## 2.23 Pictures

Put this in the preamble:

```
\usepackage{graphicx}
```

And then put this where you want a picture:

```
\begin{figure}
  \includegraphics[ scale=size ]{imageSansSuffix}
\end{figure}
```

If you want to specify exaclty where a picture goes, include the `float` package in the preamble, and use the following for your picture:

```
\begin{figure}[H]
  \includegraphics[ scale=size ]{imageSansSuffix}
\end{figure}
```

## 2.24 Piecewise Functions

These things are a real bummer sometimes. The case environment does not allow us to control right or left justification or centering.. as far as I know.

```
\begin{displaymath}
f(x) = \left\{
\begin{array}{lr}
1 & : x \in \mathbb{Q}\\
0 & : x \notin \mathbb{Q}
\end{array}
\right.
\end{displaymath}
```

## 2.25 Subfigures

We can produce subfigures if we insert the following package in the preamble:

```
\usepackage{subfigure}
```

We then produce subfigures in the following manner:

```
\begin{figure}[ht]
\centering
\subfigure[Subfigure 1 caption]{
\includegraphics[scale =1] {subfigure1.eps}
\label{fig:subfig1}
}
\subfigure[Subfigure 2 caption]{
\includegraphics[scale =1] {subfigure2.eps}
\label{fig:subfig2}
}
\subfigure[Subfigure 3 caption]{
\includegraphics[scale =1] {subfigure3.eps}
\label{fig:subfig3}
}
\label{myfigure}
\caption{Global figure caption with a reference \ref{fig:subfig1} }
\end{figure}
```

Table 2.1: A simple table

| | | |
|---|---|---|
| 0.20815 | 0.09849 | 47.316 |
| 0.20346 | 0.09505 | 46.716 |
| 0.05613 | 0.02800 | 49.884 |

Table 2.2: A nicer table

| Row | Before | After | Reduction |
|---|---|---|---|
| 106 | 0.20815 | 0.09849 | 47.316 |
| 241 | 0.20346 | 0.09505 | 46.716 |
| 245 | 0.05613 | 0.02800 | 49.884 |

## 2.26 Table of Contents

Latex can produce a table of contents automatically by using the following command:

```
\tableofcontents
```

## 2.27 Tabular Environment

This is how we do a simple table:

```
\begin{table}
\caption{A simple table}
\begin{center}
\begin{tabular}{ | c | c | c | }
\hline
0.20815 & 0.09849 & 47.316 \\ \hline
0.20346 & 0.09505 & 46.716 \\ \hline
0.05613 & 0.02800 & 49.884 \\ \hline
\end{tabular}
\end{center}
\end{table}
```

And this is a slightly nicer looking table:

```
\begin{table}
\caption{A nicer table}
\begin{center}
\begin{tabular}{ c | c c c }
\multicolumn{1}{c}{Row} & \multicolumn{1}{c}{ Before } & \multicolumn{1}{c}
{ After } & \multicolumn{1}{c}{ Reduction } \\ \hline\hline
106 & 0.20815 & 0.09849 & 47.316 \\
241 & 0.20346 & 0.09505 & 46.716 \\
245 & 0.05613 & 0.02800 & 49.884 \\
\end{tabular}
\end{center}
\end{table}
```

Here, we have a much fancier table. Most tables in publications do not use vertical lines. The `booktabs` package provides us with nicer tables than the standard tabular environment. Note that the `table` environment is like the `figure` environment for pictures. Note that we need to put `\usepackage{booktabs}` in the preamble for this kind of stuff. (Also, `\usepackage{multicol}`, and `\usepackage{multirow}` )

```
\begin{table}[h]
\caption{A Fancy Table}
\begin{center}
\begin{tabular}{ c c c c c }

\toprule

& & \multicolumn{3}{c}{Spatial Dimension} \\

& $ \times $ & $ 403 \times 515 $ & $ 205 \times 261 $ & $ 106 \times 134 $ \\ \midrule

\multicolumn{1}{c}{\multirow{7}{*}{Spectral}} &
124 & 0.625453 & 0.382117 & -0.001222 \\ \cmidrule(r){2-5}

\multicolumn{1}{c}{\multirow{6}{*}{Dimension}} &
65  & 0.687148 & 0.400093 & -0.001172 \\ \cmidrule(r){2-5}

& 36  & 0.528402 & 0.357793 & -0.001222 \\ \cmidrule(r){2-5}

& 21  & 0.410316 & 0.302287 & -0.001234 \\ \cmidrule(r){2-5}

& 14  & 0.450291 & 0.324485 & -0.001246 \\ \cmidrule(r){2-5}

& 10  & 0.355268 & 0.258846 & -0.001222 \\ \bottomrule

\end{tabular}
\label{tbl:mcc}
\end{center}
\end{table}
```

Table 2.3: A Fancy Table

| | | Spatial Dimension | | |
|---|---|---|---|---|
| | $\times$ | $403 \times 515$ | $205 \times 261$ | $106 \times 134$ |
| Spectral Dimension | 124 | 0.625453 | 0.382117 | -0.001222 |
| | 65 | 0.687148 | 0.400093 | -0.001172 |
| | 36 | 0.528402 | 0.357793 | -0.001222 |
| | 21 | 0.410316 | 0.302287 | -0.001234 |
| | 14 | 0.450291 | 0.324485 | -0.001246 |
| | 10 | 0.355268 | 0.258846 | -0.001222 |

## 2.28 Title

Latex can produce a title automatically by putting the following information in the preamble:

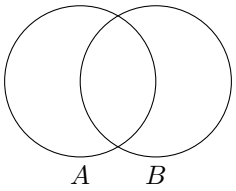
```
\title{TitleName}
\author{AuthorName}
\date{08-28-84}
```

and using the the following command in the body:

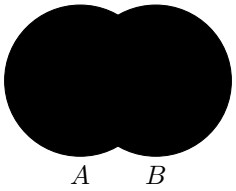
```
\maketitle
```

## 2.29 Tikz

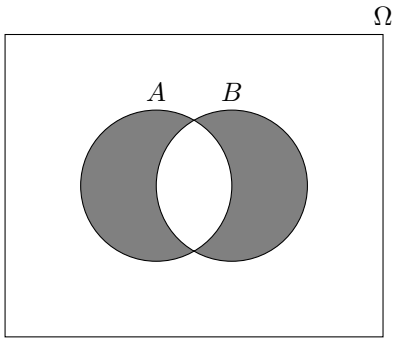
The following are some Venn diagrams produced with Tikz. So far, this only works when you compile with `pdflatex` or `latex`.




```
\def\firstcircle{ (0,0) circle (1cm) }
\def\secondcircle{ (1,0) circle (1cm) }

\begin{center}
\begin{tikzpicture}
  \path[draw] \firstcircle (0,-1) node [text=black,below] {$A$};
  \path[draw] \secondcircle (1,-1) node [text=black,below ] {$B$};
\end{tikzpicture}
\end{center}
```




```
\begin{center}
\begin{tikzpicture}
  \path[ draw, pattern = north west lines ] \firstcircle (0,-1) node [text=black,below] {$A$};
  \path[ draw, pattern = north east lines ] \secondcircle (1,-1) node [text=black,below ] {$B$};
\end{tikzpicture}
\end{center}
```




```
\begin{center}
\begin{tikzpicture}[fill=gray]
% left hand
\scope
\clip (-2,-2) rectangle (2,2)
      (1,0) circle (1);
```

```
\fill (0,0) circle (1);
\endscope
% right hand
\scope
\clip (-2,-2) rectangle (2,2)
      (0,0) circle (1);
\fill (1,0) circle (1);
\endscope
% outline
\draw (0,0) circle (1) (0,1)  node [text=black,above] {$A$}
      (1,0) circle (1) (1,1)  node [text=black,above] {$B$}
      (-2,-2) rectangle (3,2) node [text=black,above] {$\Omega$};
\end{tikzpicture}
\end{center}
```

We can also produce `.eps` images of TikZ figures. We'll need to produce the figure in a separate document, and then use the `geometry` environment to bring the edges of the page in.

```
\documentclass{article}
\usepackage{tikz}
\usepackage{geometry}
\geometry{
  papersize={4.85in,2.15in},%
  left=0.05in,%
  top=0.25in,%
  bottom=0.0in
}
\begin{document}
\pagestyle{empty}
\begin{tikzpicture}
...
\end{tikzpicture}
```

We then compile with `latex` to produce a `.dvi` file. We then do a funny thing with the `dvips` and `epstool` utilities.

```
latex picture.tex
dvips -E picture.dvi -o tmp.eps
epstool --copy --bbox tmp.eps picture.eps
```

## 2.30 Miscellaneous

### 2.30.1 Binomial Coefficients

If you need to describe a combination in an equation you can use:

```
\binom{n}{r}, \dbinom{n}{r}
```

Note that `\dbinom{}{}` is used in display mode when you want to combination terms to scale appropriately. If you use `\binom{}{}`, then you might end up with small combination terms. Say, for instance, if you were trying to describe a hypergeometric distribution, then you'd want to use `\dbinom{}{}`.

### 2.30.2 Draw a Horizontal Line

```
\begin{center}
\line(1,0){250}
\end{center}
```

# Chapter 3

# Emacs

(top)

## 3.1 Copying and Pasting

| | |
|---|---|
| C-g | abort |
| C-space | set mark |
| C-w | cut (kill) |
| M-w | copy |
| C-y | paste (yank) |

## 3.2 Cursor Movement

| | |
|---|---|
| C-p,C-n | prev/next line |
| C-a,C-e | begin/end of line |
| C-f,C-b | fwd/bkwd by letter |
| M-f,M-b | fwd/bkwd by word |
| C-v,M-v | up/down by page |
| M-<,M-> | top/bottom of document |

## 3.3 Undoing, Saving, and Quitting

| | |
|---|---|
| C-_ | undo |
| C-x C-s | save |
| C-x C-w | save as.. |
| C-x C-c | close |

## 3.4 Files and Windows

| | |
|---|---|
| C-f | Open a new file |
| C-x C-2 | Split a window horizontally |
| C-x C-3 | Split a window vertically |
| C-x C-o | Change windows |
| C-x C-1 | Close all other windows |

## 3.5 Search and Rescue

| | |
|---|---|
| `M-%` | search and replace |
| `query-string` | SPC or y to replace |
| `RET` | DEL or n to skip |
| `replm-string` | RET or q to exit |
| `RET` | ! to replace all |

# Chapter 4

# vi

(top)

## 4.1 Modes and Controls

| | |
|---|---|
| Command Mode | Press the `ESC` key |
| Insertion Mode | Entered on insertion or change |

## 4.2 Saving and Quitting

| | |
|---|---|
| `:w` | Save |
| `:w` *file* | Save to *file* |
| `:x` | Save and Quit |
| `:q` | Quit |
| `:q!` | Discard and Quit |

## 4.3 Abbreviations

You can create an abbreviation using the following command:

```
:ab bls \begin{lstlisting}
```

This can be added to the `/.vimrc` file so that it will ba available for ever session.

## 4.4 Cursor Navigation

| | |
|---|---|
| `h` | Left |
| `j` | Down |
| `k` | Up |
| `l` | Right |
| `w` | Forward one word |
| `W` | Forward one blank-delimited word |
| `b` | Back one word |
| `B` | Back one blank-delimited word |
| `(` | Back a sentence |
| `)` | Forward a sentence |
| `{` | Back a paragraph |
| `}` | Forward a paragraph |
| `<ctrl>d` | Scroll down half a screen |
| `<ctrl>u` | Scroll up half a screen |
| `<ctrl>f` | Forward a page |
| `<ctrl>b` | Back a page |
| `^` | Begining of a line |
| `$` | End of a line |
| `1G` | Beginning of the file |
| `G` | End of the file |

## 4.5 Insertion

| | |
|---|---|
| `i` | Insert before cursor |
| `a` | Append after cursor |
| `I` | Insert before line |
| `A` | Append after line |
| `o` | Add new line after current line |
| `O` | Add new line before current line |
| `r` | Overwrite one character |
| `R` | Overwrite many characters |
| `C` | Overwrite the whole line |
| `:r` *file* | Read *file* and insert it after this line |
| `p` | Put after this position |
| `P` | Put before this position |

## 4.6 Yanking/Copying

Items are yanked, or copied, to the general buffer, unless otherwise specified.

| | |
|---|---|
| `yw` | Yank word to general buffer |
| `yy,Y` | Yank line to general buffer |

## 4.7 Deleting

Items are deleted to the general buffer, unless otherwise specified.

| | |
|---|---|
| `x` | Delete character to right of cursor |
| `X` | Delete character to left of cursor |
| `D` | Delete rest of the line |
| `dd,:d` | Delete current line |
| `d$` | Delete to the end of the line |
| `d`*n*`w` | Delete next *n* words |
| `d`*n*`b` | Delete previous *n* words |
| `dd`*n* | Delete current and next *n* lines |

## 4.8 Pasting and Undoing

| | |
|---|---|
| `p` | Paste after cursor |
| `P` | Paste before cursor |
| ``` ``a`p ``` | Paste text from buffer *a* after cursor |
| `u` | Undo last change |
| `U` | Undo all changes on line |

## 4.9 Searching

| | |
|---|---|
| `f`*x* | Find *x* |
| `;` | Find next *x* |

## 4.10 Search and Replace (Rescue)

This uses syntax like SED.

For the first occurrence on the current line:

```
:s/OLD/NEW
```

For all (globally) occurrences on the current line:

```
:s/OLD/NEW/g
```

Between two lines, #,#:

```
:#,#s/OLD/NEW/g
```

For ever occurrence in the file:

```
:%s/OLd/NEW/g
```

## 4.11 Fun Things to Put in Your `.vimrc` File

This turns on suntax highlighting, and proper tabbing, at least for Python.

```
syntax on
filetype indent plugin on
au FileType python setlocal tabstop=8 expandtab shiftwidth=4 softtabstop=4
```

These let you enter empty lines without leaving Normal Mode.

```
map<S-Enter> O<Esc>
map <CR> k$o<Esc>
```

## 4.12 Other

| | |
|---|---|
| . | Repeat last command |

# Chapter 5

# Regular Expressions

(top)

## 5.1 Literal Characters

The simplest regular expressions match characters or character strings directly. For example, the regular expression **cap** would match the first three letters in **captain**, but not **Captain** since regular expressions are by default case-sensitive.

## 5.2 Metacharacters

Certain characters, called metacharacters, have special meaning in regular expressions. To use them literally, we must "escape" them with a backslash. These metacharacters are the square brackets, the parenthesis, the pipe, the backslash, the carat, the dolloar sign, the period, plus and asterix.

```
[ ] ( ) | \ ^ $ . + *
```

Thus, if we want to match the string **1+1=2**, we must use the regex **1\+1=2**.

## 5.3 Non-printing Characters

You can use special sequences to match certain non-printing characters, like tabs **\t**, carriage returns **\r**, and line feeds **\n**. Note that Windows text files use **\r\n** to terminate lines, while UNIX text files use **\n**.

## 5.4 Character Sets

You can use Character sets to match exactly one out of several characters. For example, the regex **gr[ae]y** would match the strings **gray** or **grey**.

You can also use a hyphen to sprecify a range of characters. For example, the regex **[0-9]** would match a single digit between zero and nine.

## 5.5 Negated Character Sets

If you want to negate a character set, you use a carat after the opening square bracket. For example, the regex **q[^u]** matches any $q$ followed by a character that is not $u$. Therefore, it would match **Iraq** and **Qatar**.

Note that in the case of **Iraq**, the space after the *q* is matched because the sapce is not a *u*. Such is the animal that is negation.

## 5.6 Character Set Metacharacters

The only characters that act as metacharacters inside a character set are the closing square bracket, the backslash, the caret and the hyphen.

```
] \ ^ -
```

The other metacharacters have no special meaning within character sets. Backslashes need to be escaped by another backslash. The regex **[\\x]** matches an **x** or a backslash. The remaining character set metacharacters, the hyphen, caret, and the closing square bracket may be escaped with a backslash, or placed in a position where they do not take on their special meaning. For example, the regex **[x^]** matches an **x** or a caret, whereas the regex **[^x]** matches any character that is not an **x**. Furthermore, the regexes **[-x]** and **[x-]** match an **x** or a hyphen.

## 5.7 Shorthand Character Sets

Note that **\d**, meaning *digit*, is short for **[0-9]**, **\w**, meaning *word character*, is short for **[a-zA-Z0-9]**, and **\s** matches any whitespace character.

We also have negated versions of these shorthands character sets. The regex **\D** is the same as **[^d]**, **\W** is the same as **[^w]**, and **\S** is the same as **[^s]**.

## 5.8 Repeating Characters Sets

If you repeat a character set using **?**, **+** or *****, then you will repeat the entire set, not just the character that it matched. For example, the regex **[0-9]+** will match both **135** and **222**

## 5.9 Dot

The dot metacharacter matches any single character except the newline character (\n), on Unix machines, and the combination carriage return newline (\r\n), on Windows. Way to go Windows.

## 5.10 Anchors

The carat metacharacter matches things at the beginning of lines, and the dollar sign metacharacter matches things at the end of lines. For example, the regex **^a** matches the **a** in the text **abc**, but the regex **^b** does not match anything the text **abc**.

## 5.11 Word Boundaries

The **\b** regex is an anchor that matches a position called a word boundary that occurs at three positions (a) before the first character of a string, (b) after the last character of a string or (c) between two characters in a string where one is a word character, and the other is not. Note that **[a-zA-Z0-9_]** are word characters.

## 5.12 Alternation

Character sets match a single character out of a set of several possible characters. You can use alternation to match a single regex out of several possible regexes. For example, if you wanted to match **spam** or **eggs** then you'd use the regex **spam|eggs**. However, this would also match the spam in, say, **spamalicious**. To pick out individual words we can use parentheses. The regex **\b(spam|eggs)\b** matches only the whole words **spam** or **eggs**. If we had omitted the parentheses, we'd have matched either a word boundary followed by **spam** or **eggs** followed by a word boundary.

## 5.13 Optional Items?

The question mark matches the preceding token zero or more times, making it optional. For example, the regex **colou?r** matches the words **color** or **colour** and the regex **Nov(ember)?** matches the words **Nov** and **November**.

## 5.14 Limiting Repetition

If you want to match things that occur a within a certain range of times, you can use curly braces. The syntax is ***{min,max}***. For example, the regex **a/1,3b** would match **a/b**, **a//b**, or **a///b**, but not **ab** or **a////b**.

# Chapter 6

# Bash

(top)

## 6.1 Introduction

This is a valuable resource: `http://tldp.org/LDP/abs/html/`

## 6.2 Variables

If you want to store the output of the date function, you would type:

```
d0=date;
```

In order to retrieve the value of that variable, you'd use one of two forms:

```
$d0
${d0}
```

The first form is the syntax-sugar form, the second form is useful when you want to embed that variable in a string. For instance, if you were trying to iterate over a bunch of filenames: `x_0.dat`, `x_1.dat`, and `x_2.dat`, then you'd do something like,

```
for i in $(seq 0 2) ; do rm x_${i}.dat ; done ;
```

## 6.3 Logical Expressions

If you want to test multiple conditions use the -a (and ) and -o (or ) switches. Using && in place of the -a switch produces the same effect but causes the program to execute the condition twice, so the -a switch is preferred.

```
if [ condition1 -a condition2 ] ; then
  body
fi
```

Be careful using the not operator with the -a and -o switches because Bash gives precedence to the switches rather than the operator. To handle precedence issues use escaped parenthesis.

```
if [ condition1 -o \( !condition2 \) ] ; then
  body
fi
```

## 6.4 Loops

```
for index in list ; do
  body
done
```

```
for i in $(seq begin end step ) ; do
  body ${i}
done
```

```
for (( counter = 0; counter < 10; counter++)) ; do
  body
done
```

```
while condition ; do
  body
done
```

We can alternatively write loops in the following manner, with the *do* keyword on another line.

```
for index in list
do
  body
done
```

## 6.5 Control Structures

The if control structure has two forms. You may test a condition using the test command. Note that the square brackets offer an equivalent alternative to the test command.

```
if test condition ; then
  body
fi
```

```
if [ condition ] ; then
  body
fi
```

Alternatively you may use the return status of a command.

```
if command ; then
  body
fi
```

The else and elif commands have the following syntax.

```
if [ condition ] ; then
  body1
else
  body2
fi
```

```
if [ condition1 ] ; then
  body1
elif [ condition2 ] ; then
  body2
else
  body3
fi
```

## 6.6 Prioritizing Processes

We can use the `nice` command to make stuff go fasterrr. A `-20` puts the highest priority on the process, and a `20` puts the lowest priority on a process. Here's the syntax:

```
$ sudo nice -n -20 python supafast.py
$
```

## 6.7 Asserting Root Userery

You can sudo everything by using the following listing. This, however, can be dangerous.

```
$
$ sudo bash
#
```

Notice that the prompt changes to a hash sign. This is a quick visual cue to let you know that you're living (coding) dangerously.

# Chapter 7

# CMD

(top)

## 7.1 Commands

| | |
|---|---|
| `dir /b` | Display the contents of the current directory. |
| `rd /q` | Remove the files in a directory. |
| `rd /s /q` | Remove the contents of a directory recursively. |
| `move` *`file dest`* | Move a file to a destination. |
| `start /b` *`program`* | Start a *program* in the background. |

## 7.2 Batch Files

The following batch file takes two arguments, a path to a directory, and a filename. It will take the output of the `dir` command and write it to the filename.

```
@echo off
dir /b %1 > %2
```

## 7.3 Compile a C/C++ Program

Install `MinGW`. Add `C:\MinGW\bin` to the `Path` variable. Complile by executing,

```
$ gcc -o filename.exe filename.c
```

## 7.4 GNU Scientific Library

I installed `GnuWin32` from `http://gnuwin32.sourceforge.net/packages/gsl.htm` by clicking on *"Complete package, except sources"*, and it worked like a charm. I compiled executables using the line,

```
$ g++ gsltest.cpp -o gsltest.exe -lgsl -lgslcblas -lm
```

I also copied everything in the `C:\Program Files (x86)\GnuWin32\[lib|bin|include]` directories and put them in `C:\MinGW[lib|bin|include]`. (I don't know if that made things better or not.)

# Chapter 8

# C

(top)

## 8.1 Introduction

In the words of Del Spangler, "We don't write in C because we want to, we write in C because we have to." C was developed by Dennis Ritchie at Bell Laboratories and AT&T in the early seventies.

```
#include <stdio.h>
int main()
{
  printf(``Hello World.'') ;
  return 0 ;
}
```

## 8.2 Compile and Run a C Program

Here is how you run a C program on a linux machine.

```
$ gcc filename.c
$ ./a.out
$ gcc filename.c -o appname
$./appname
```

This is what happens: a compiler compiles the source code `program.c` and the header files to produce an object file, `program.o`, or `program.obj` on Windowslol. Then a linker links the object file with library files and other object files to produce an executable.

## 8.3 Loops

```
for ( counter = 0; counter < 10; counter ++ )
{
  body ;
}
```

```
while ( condition )
{
  body ;
}
```

```
do
{
  body ;
}
while ( condition );
```

## 8.4 Control Structures

C has the follwing structure that returns value1 if the condition is true, and value2 otherwise.

```
( condition ? value1 : value2 )
```

The more familiar if sttructures are as follows:

```
if ( condition )
{
  body ;
}
```

```
if ( condition )
{
  body1 ;
}
else
{
  body2 ;
}
```

```
if ( condition1 )
{
  body1 ;
}
else if ( condition2 )
{
  body2 ;
}
else
{
  body3 ;
}
```

## 8.5 Switch

We can also avail ourselves of a switch structure:

```
switch ( condition )
{
  case expression1:
    statement1 ;
    break ;

  case expression2:
    statement2 ;
    break ;

  case expression3:
    statement3 ;
    break ;
```

```
  default:
    statement ;
    break ;
}
```

## 8.6 break and continue

Note that break exits a loop, and that continue just skips to the next iteration.

## 8.7 Functions

A function has the following syntax.

```
int add( int x, int y )
{
  int result ;
  result = x + y ;
  return result ;
}
```

We would place it in a program and call it in the following manner:

```
 #include <stdio.h>

int add( int x, int y )
{
  int result ;
  result = x + y ;
  return result ;
}

int main()
{
  int sum ;
  sum = add( 3, 5 ) ;
  printf( ``the sum is: %d'', sum);
  return 0;
}
```

## 8.8 Preprocessor Directives and Header Files

The statement `#include <…>` is a preprocessor directive. This statement loads a given file at that spot. If you use quotes aroundd the header file, that prompts the compiler to look in the current directory for that file first.

## 8.9 main()

Execution of the program begins and ends with the statements in the `main()` function.

## 8.10 return and exit()

All functions in C can return a value. The `main()` function returns a zero to signify that it packed its big-girl panties and it terminated normally. You can also exit a function without returning a value by using the `exit()` function, but you have to include the `stdlib.h` header.

## 8.11 Comments

C uses `/* */` for commenting.

## 8.12 Keywords

| | | | |
|---|---|---|---|
| auto | double | int | struct |
| break | else | long | switch |
| case | enum | register | typedef |
| char | extern | retuurn | union |
| const | float | short | unsigned |
| continue | for | signed | void |
| default | goto | sizeof | volatile |
| do | if | static | while |

## 8.13 Standard I/O

The `getc()` function gets the next character from a file stream and returns it as an integer.

```
#include <stdio.h>
int getc( FILE *stream ) ;
```

Here's some sample code:

```
#include <stdio.h>
int main()
{
  int ch ;
  printf( ``Type a character: '' ) ;
  ch = getc( stdin ) ;
  printf( ``Boom. You just typed: %c'', ch ) ;
  return 0 ;
}
```

The function `getchar()` does the exact same thing as `getc( stdin )`.
Write some more stuff on `putc()`, `putchar()`, `printf()` and formatting.

## 8.14 Post-Increment and Pre-Increment

These guys are confusing as all hell. The *post-increment* operator assigns the original value of `x` to `y`, and then increments `x`.

```
  y = x++ ;
```

However, the *pre-increment* operator increments the value of `x`, and then assigns the incremented value to `y`.

```
  y = ++x ;
```

The same goes for the pre- and post-decrement operators.

## 8.15 Cast Operator

The cast operator changes the data type of a constant, variable, or expression.

```
(data-type) x
```

## 8.16 Hierarchical Data Format

This is how you compile a program using the heirarchical data format stuff.

```
gcc -I/usr/local/include h.c -L/usr/local/lib -lhdf5 -o h
```

## 8.17 malloc

### 8.17.1 Basic Form

This always escapes me, even though I use it every time I use C.

```
/* Allocate space for an array with ten elements of type int. */
int *ptr = (int *) malloc(10 * sizeof (int));
```

### 8.17.2 Multidimensional Arrays

Here is one form I found on the internest.

```
#include "stdlib.h"

#define RANK 2
#define NX 25000
#define NY 25000

int main() {

   float       **data;
   int         rows, cols;
   int         i, j;

/* Allocate memory for new integer array[row][col]. First allocate
   the memory for the top-level array (rows).  Make sure you use the
   sizeof a *pointer* to your data type. */

   data = (float**) malloc( NX*sizeof(float*) );

/* Allocate a contiguous chunk of memory for the array data
   values.  Use the sizeof data type */

   data[0] = (float*) malloc( NX*NY*sizeof(float) );

/* Set the pointers in the top-level (row) array to the correct
   memory locations in the data value chunk. */

   for( i=1; i<NX; i++ ) data[i] = data[0]+i*NY;

   for( i=0; i < NX; i++ ) {
     for( j=0; j < NY; j++ )
       data[i][j] = (float) 0.0;
   }
   free(data[0]);
```

```
   free(data);
   return 0;
}
```

And another form.

```
#include "stdlib.h"

#define RANK 2
#define NX 25000
#define NY 25000

int main() {

   float       *data;
   int         rows, cols;
   int         i, j;

   data = malloc( NX*NY*sizeof(float) );

   for( i=0; i < NX; i++ ) {
     for( j=0; j < NY; j++ )
       data[ i*NY+j ] = (float) 0.0;
   }
   free(data);
   return 0;
}
```

# Chapter 9

# C++

(top)

## 9.1 Introduction

This is where most C++ programs start. I have listed the most common hearders that I can think of.

```
#include <cmath>
#include <string>
#include <fstream>
#include <iostream>
#include <stdlib.h>
// declare constants
// declare functions
using namespace std;
int main()
{
  body ;
  return 0 ;
}
// define functions
```

## 9.2 Compile and Run a C++ Program

Here is how you run a C++ program on a linux machine. For a C program, use the gcc compiler and a dot-c extension.

```
$ g++ filename.cpp
$ ./a.out
$ g++ filename.cpp -o appname
$./appname
```

## 9.3 Functions

A function is a subprogram that can be called or invoked from another function and can return a value to it. Every C++ program starts with the main() function. Relegating separate tasks to separate functions is a fundamental programming technique that leads to simpler and more efficient programs.

```
#include <iostream>
using namespace std;
```

```
double power ( const double, const int );

int main()
{
  cout << "power(2,3) = " << power(2,3) << endl;
  return 0;
}

double power ( const double x, int n )
{
  double y = 1.0;
  for ( int i = 0; i < n; i++ )
  {
    y *= x;
  }
  return y;
}
```

## 9.4 Files

Here's an example of how to capitalize words in a file.

```
#include <ctype.h>
#include <fstream>
int main()
{
  ifstream infile"(input."txt);
  ofstream outfile"(output."txt);
  string line;
  while( getline( infile, line ) )
  {
    for ( int I = 0; I < line.length(); i++ )
    {
      if ( I == 0 || line[i-1] == ' ' && isalpha( line[i] ) )
      {
        line[i] = toupper( line[i] );
      }
    }
    outfile << line << endl;
  }
  return 0;
}
```

## 9.5 Pointers

### 9.5.1 Reference Operator (&)

Also called the "address-of" operator. This returns the address of a variable in memory. A variable that stores the reference of another variable is called a *pointer*. Pointers are said to *point to* the variable whose address the store.

```
x = 6;      // x is equal to six
y = &x;     // y is equal to the address of x
```

### 9.5.2 Dereference Operator (*)

In order to access the value stored in the variable that a pointer points to, we need to prepend the dereference operator (*) to the pointer. The dereference operator can be literally translates to "the value pointed to by".

```
x = 6;      // x is equal to six
y = &x;     // y is equal to the address of six
z = *y;     // z is equal to six, the value pointed to by y
```

### 9.5.3 Pointer Declaration

Since you have to tell C++ *everything*, we need to tell it that ahead of time that a variable is going to be a pointer or not, we do this with an asterix.

```
type * name;
```

If we want to define two pointers, we use two asterices.

```
int * p1, * p2;   // two pointers, p1 and p2
int * p1, p2;     // a pointer, p1, and an int p2
```

## 9.6 goto

The expression repeat: is label; this is where the execution of control is diverted by means of the goto statement. It is in general practice, terrible form to use goto statements, but if you see one in code, this is what is happening.

```
#include <iostream>
using namespace std;
int main()
{
  const int 100;
  double sum = 0.0;
  int x = 1;
  repeat: sum += 1.0/x++;
  if ( x <= N ) goto repeat; \\ bad!
  cout << "The sum of the first " << N << "reciprocals is " << sum;
  return 0;
}
```

## 9.7 User Input and Standard Output

```
#include <iostream>
using namespace std;
int main()
{
string name;
cout << "Enter your name: ";
cin >> name;
cout << "\"n;
cout << "Hello, " << name << "!\"n;
return 0;
}
```

## 9.8 GNU Scientific Library

```
$ g++ -Wall -I/home/connor/dev/include -c example.cpp
$ g++ -L/home/connor/dev/lib example.o -lgsl -lgslcblas -lm
```

## 9.9 Haar Cascaded Classification

I've put this together based on two sites:

- `http://achuwilson.wordpress.com/2011/07/01/create-your-own-haar-classifier-for-detecting-objects-ir`
- `http://note.sonots.com/SciSoftwarehaartraining.html`

First, you need a directory of *positive* images. The positive images are the sorts of things you're looking for. Make sure these are PNGs of some reasonable size, say $20 \times 20$ pixels.

```
$ mkdir pos
```

Then you need a *description file* of those positive images. We'll call it `pos.dat`.

```
$ find pos -name '*.png' | sed 's/$/ 1 0 0 20 20/' > pos.dat
```

Next, you need to create samples from an image collection. This will produce a binary file with a *.vec* extension. The `opencv_createsamples` utility is in the `bin` directory of wherever you built the OpenCV isntallation.

```
$ ~/src/OpenCV-2.4.1/build/bin/opencv_createsamples -info pos.dat -vec pos.vec -show -w 20 -h 20
```

And then this runs forever, until we do the Haar training, which will run for even longer.

# Chapter 10

# CUDA

(top)

## 10.1 Overview

1. Device initialization
2. Device memory allocation
3. Copy data to device
4. Execute kernel (Calling ___global___ function)
5. Copy data from device memory

## 10.2 CUDA / Compile and Run a CUDA Program

```
nvcc program.cu
```

## 10.3 CUDA / Terminology

Device = GPU = set of multiprocessors
Multiprocessor = set of processors & shared memory
Kernel = GPU program
Grid = array of thread blocks that execute a kernel
Thread block = group of SIMD threads that execute a kernel and can communicate via shared memory

```
// moveArrays.cu
// demonstrates CUDA interface to data allocation on device (GPU)
// and data movement between host (CPU) and device.
#include <stdio.h>
#include <assert.h>
#include <cuda.h>
int main(void)
{
  // pointers to host memory
  float *a_h, *b_h;
  // pointers to device memory
```

```
  float *a_d, *b_d;
  int N = 14;
  int i;

  // allocate arrays on host
  a_h = (float *)malloc(sizeof(float)*N);
  b_h = (float *)malloc(sizeof(float)*N);
  // allocate arrays on device
  cudaMalloc((void **) &a_d, sizeof(float)*N);
  cudaMalloc((void **) &b_d, sizeof(float)*N);
  // initialize host data
  for (i=0; i<N; i++)
  {
    a_h[i] = 10.f+i;
    b_h[i] = 0.f;
  }
  // send data from host to device: a_h to a_d
  cudaMemcpy(a_d, a_h, sizeof(float)*N, cudaMemcpyHostToDevice);
  // copy data within device: a_d to b_d
  cudaMemcpy(b_d, a_d, sizeof(float)*N, cudaMemcpyDeviceToDevice);
  // retrieve data from device: b_d to b_h
  cudaMemcpy(b_h, b_d, sizeof(float)*N, cudaMemcpyDeviceToHost);
  // check result
  for (i=0; i<N; i++)
    assert(a_h[i] == b_h[i]);
  }
  // cleanup
  free(a_h); free(b_h);
  cudaFree(a_d); cudaFree(b_d);
}
```

Kernel. A function that is callable from the host and executed on the CUDA device, simultaneously by many threads in parallel. Execution Configuration. Defines the number of parallel threads per group (block), and the number of groups to use when running the kernel. Host calls a kernel by specifying the name of the kernel and an execution configuration, enclosed between triple angle brackets, «< * »>. CUDA provides several extensions to the C-language. The function type qualifier ___global___ declares a function as being an executable kernel on the CUDA device, which can only be called from the host. All kernels must be declared with a return type of void. The kernel does not require a loop because the function is simultaneously executed by an array of threads on the CUDA device. However, each thread is provided with a unique ID that can be used to compute different array indicies or make control decisions (such as not doing anything if the thread index exceeds the array size).

```
// incrementArray.cu
#include <stdio.h>
#include <assert.h>
#include <cuda.h>

void incrementArrayOnHost(float *a, int N)
{
  int i;
  for (i=0; i < N; i++) a[i] = a[i]+1.f;
}

__global__ void incrementArrayOnDevice(float *a, int N)
{
  int idx = blockIdx.x*blockDim.x + threadIdx.x;
  if (idx<N) a[idx] = a[idx]+1.f;
}

int main(void)
{
  // pointers to host memory
  float *a_h, *b_h;
```

```
  // pointer to device memorY
  float *a_d;

  int i, N = 10;
  size_t size = N*sizeof(float);

  // allocate arrays on host
  a_h = (float *)malloc(size);
  b_h = (float *)malloc(size);

  // allocate array on device
  cudaMalloc((void **) &a_d, size);

  // initialization of host data
  for (i=0; i<N; i++) a_h[i] = (float)i;

  // copy data from host to device
  cudaMemcpy(a_d, a_h, sizeof(float)*N, cudaMemcpyHostToDevice);

  // do calculation on host
  incrementArrayOnHost(a_h, N);

  // do calculation on device:
  int blockSize = 4;
  int nBlocks = N/blockSize + (N%blockSize == 0?0:1);
  incrementArrayOnDevice <<< nBlocks, blockSize >>> (a_d, N);

  // Retrieve result from device and store in b_h
  cudaMemcpy(b_h, a_d, sizeof(float)*N, cudaMemcpyDeviceToHost);

  // check results
  for (i=0; i<N; i++) assert(a_h[i] == b_h[i]);

  // cleanup
  free(a_h); free(b_h); cudaFree(a_d);
}
```

The grid abstraction was created to let developers take into account–without recompilation–differing hardware capabilities regardless of price point and age. A grid, in effect, batches together calls to the same kernel for blocks with the same dimensionality and size, and effectively multiplies by a factor of nBlocks the number of threads that can be launched in a single kernel invocation. Less capable devices may only be able to run one or a couple of thread blocks simultaneously, while more capable (e.g., expensive and future) devices may be able to run many at once. Designing software with the grid abstraction requires balancing the trade-offs between simultaneously running many independent threads, and requiring a greater number of threads within a block that can cooperate with each other. Please be cognizant of the costs associated with the two types of threads. Of course, different algorithms will impose different requirements, but when possible try to use larger numbers of thread blocks.

In the kernel on the CUDA-enabled device, several built-in variables are available that were set by the execution configuration of the kernel invocation. They are: • blockIdx which contains the block index within the grid. • threadIdx contains the thread index within the block. • blockDim contains the number of threads in a block.

These variables are structures that contain integer components of the variables. Blocks, for example, have x-, y-, and z- integer components because they are three-dimensional. Grids, on the other hand, only have x- and y-components because they are two-dimensional. The variables nBlocks and blockSize are the number of blocks in the grid and the number of threads in each block, respectively. It is important to note that threads within a block can communicate with each other through local multi-processor resources because the CUDA execution model specifies that a block can only be processed on a single multi-processor. In other words, data written to shared memory within a block is accessible to all other threads within that

block, but it is not accessible to a thread from any other block.

A quick summary of local multiprocessor memory types with read/write capability follows:

- Registers:
  - The fastest form of memory on the multi-processor.
  - Is only accessible by the thread.
  - Has the lifetime of the thread.
- Shared Memory:
  - Can be as fast as a register when there are no bank conflicts or when reading from the same address.
  - Accessible by any thread of the block from which it was created.
  - Has the lifetime of the block.
- Global memory:
  - Potentially 150x slower than register or shared memory – watch out for uncoalesced reads and writes which will be discussed in the next column.
  - Accessible from either the host or device.
  - Has the lifetime of the application.
- Local memory:
  - A potential performance gotcha, it resides in global memory and can be 150x slower than register or shared memory.
  - Is only accessible by the thread.
  - Has the lifetime of the thread.

# Chapter 11

# Fortran

(top)

## 11.1 Loops

Fortran77 works a little bit differently than later versions. In particular, it uses labels. The following loop counts down the even numbers from 10 to 1.

```
   DO 10 i = 10, 1, -2
      WRITE(*,*) 'i =', i
10 CONTINUE
```

Notice that the general format is

```
      DO LABEL idx = expr1, expr2, expr3
         body
LABEL CONTINUE
```

```
DO index = 0,9
   body
ENDDO
```

```
DO
IF ( condition ) EXIT
   body
ENDDO
```

## 11.2 Control Structures

```
IF condition THEN
   body
ENDIF
```

```
IF condition THEN
   body1
ELSE
   body2
ENDIF
```

```
IF condition1 THEN
   body1
ELSE IF condition2 THEN
   body2
ELSE
   body3
ENDIF
```

## 11.3 Compiling

To compile, you'll need either `g95` or `gfortran`.

```
$ gfortran program.f
$ ./a.out
```

You can name the compiled file using the `-o` flag.

```
$ gfortran program.f -o progname
$ ./progname
```

## 11.4 Example Program

So, this is a real life, working program.

```
      PROGRAM SIECIRKEHAREASJAH
      REAL R, AREA, PI
      PARAMETER( PI = 3.14159 )
C     THIS PROGRAM COMPUTES THE AREA OF A CIRCLE
      PRINT *,"WHAT IS THE RADIUS?"
      READ *,R
      AREA = PI * R ** 2.0
      PRINT *,"THE AREA IS ",AREA
      PRINT *,"LATER, PLAYER."
      END
```

## 11.5 Functions and Subroutines

The difference between functions and subroutines is that functions can be evaluated in an expression, whereas subroutines are simply blocks of code. Note that you can't have a subroutine and a function with the same name. Fortran doesn't like that. I don't know why. Lisp would be all about it, Fortran not so much.

Below, we have an exciting program that demonstrates how to call a subroutine, and evaluate a function. Note that you call a subroutine with the `CALL` keyword.

```
      PROGRAM ADDING
      INTEGER K
      K = 0
      PRINT *, K
      CALL ADDONE( K )
      PRINT *, K
      K = ADDTWO( K )
      PRINT *, K
      END PROGRAM ADDING

      SUBROUTINE ADDONE( X )
      INTEGER X
      X = X + 1
```

```
      END SUBROUTINE ADDONE

      FUNCTION ADDTWO( X )
      INTEGER X
      ADDTWO = X + 2
      END FUNCTION ADDTWO
```

## 11.6 Reading and Writing Data

Suppose we have a data file that is two columns, separated by a single whitespace, and we want to write this into two three-dimensional arrays that are 800×1024×124. Then we would use the following code:

```
      PROGRAM READINKSIEDATASJAH
      INTEGER I, J, K, IO
      INTEGER ROWS, COLS, BANDS
      PARAMETER( ROWS=800, COLS=1024, BANDS=124 )
      REAL  X(0:ROWS,0:COLS,0:BANDS), Y(0:ROWS,0:COLS,0:BANDS)
      OPEN( UNIT=1, FILE="xy.dat" )
      DO K = 0, BANDS-1
         DO I = 0, ROWS-1
            DO J = 0, COLS-1
               READ(1,*,IOSTAT=IO) X(I,J,K), Y(I,J,K)
               WRITE(*,*) X(I,J,K), Y(I,J,K)
            ENDDO
         ENDDO
      ENDDO
      CLOSE( 1 )
      END
```

The `WRITE(*,*)` line writes the data to the command prompt. You probably do not want to write this much data to the screen, but that's how you'd do it.

## 11.7 LU Decomposition

This performs LU decomposition. It is simultaneously the simplest and only working code I was able to find on the internet/library. (A library is like a house for books.)

```
      SUBROUTINE DL(AL,AU,A,Neq)
C     ==========================
      dimension AL(Neq,Neq),AU(Neq,Neq),A(Neq,Neq)
      do i=1,neq
         do j=i,neq
            au(i,j)=a(i,j)
            do k=1,i-1
               au(i,j)=au(i,j)-al(i,k)*au(k,j)
            enddo
         enddo
         do j=i+1,neq
            al(j,i)=a(j,i)
            do k=1,i-1
               al(j,i)=al(j,i)-al(j,k)*au(k,i)
            enddo
            al(j,i)=al(j,i)/au(i,i)
         enddo
      enddo
      return
      end
```

# Chapter 12

# Python

(top)

## 12.1 Introduction

Python is a programming language that lets you work more quickly and integrate your systems more effectively. You can learn to use Python and see almost immediate gains in productivity and lower maintenance costs. Python runs on Windows, Linux/Unix, Mac OS X, and has been ported to the Java and .NET virtual machines.

The following sections are my personal reference. One very good introduction to the language is Guido van Rossum's tutorial: http://docs.python.org/tutorial/. Another great resource is: http://greenteapress.com/.

## 12.2 Sequence Operations

These operations may be performed on sequences, i.e., strings, lists, tuples, bytes, bytearrays.

```
X in S              # membership
X not in S          # membership
S1 + S2             # concatenation
n*S | S*n           # repetition
S[ i ]              # index by offset
S[ i : j ]          # slice
S[ i : j : k ]      # stride k
len(S)              # length
min(S)              # minimum
max(S)              # maximum
iter(S)             # iteration protocol
for X in S:         # iterate through all elements
[ expr for X in S ] # sequence comprehension
map( func, S )      # mapping
```

## 12.3 List

### 12.3.1 Initialization

We can produce lists using the `list()` function, or by using a list comprehension.

```
>>> a = list() # equivalent to: a = []
>>> b = [ i for i in range(3) ] # list comprehension: b = [0, 1, 2]
```

### 12.3.2 Operations

We may perform the following operations on lists, including all sequence operations listed above.

```
alist.append(x)         # append an element to the end
alist.extend(x)         # insert ea item in any iterable to the end
alist.sort()            # sorts list
alist.reverse()         # reverses elements
alist.index(x[,i[,j]]) # returns index of first occurence of x
alist.insert(i,x)       # inserts x at index i
alist.count(x)          # counts the occurences of x
alist.remove(x)         # removes first occurrence of x
alist.pop([i])          # pops last element, or the element at index i
```

## 12.4 Dict

The only limitation of the `dict` data type is that it you cannot use lists as keywords for values. One can use `int`, `float`, `string`, or `tuple`, but not `list`.

```
>>> a = dict() # equivalent to: a = {}
```

## 12.5 Tuple

The `tuple` data type is immutable, meaning that after you assign the elements of a tuple, you can access those elements, but you cannot change them. You can destroy the tuple using the `del` operation, but you cannot modify its elements.

```
>>> a = tuple()
```

## 12.6 Set

The neat thing about `set()` is that sets hold only unique objects. This allows us to use the `set()` function to filter out objects from a list that occur multiple times.

```
>>> a = set()
>>> a.add(0) # a = set([0])
```

## 12.7 Range

```
>>> range(5) # range( 0, end )
[0, 1, 2, 3, 4]
>>> range(2,10) # range( begin, end )
[2, 3, 4, 5, 6, 7, 8, 9]
>>> range(2,10,2) # range( begin, end, step )
[2, 4, 6, 8]
```

## 12.8 Loops

### 12.8.1 `for`-Loops

Here, the sequence can be a any sequence structure, i.e., a `string`, `list`, or `tuple`.

```
for item in sequence :
    body
```

Sometimes we need to produce overlapping windows. Let `w` be the window size, then the following loop will produce a sliding, overlapping window of size `w`.

```
w = 10
n = 100
for i in range( n/(w/2)-1 ):
    print i*(w/2), (i+2)*(s/2)
```

### 12.8.2 `while`-Loops

The `while` loop will continue to execute the body as long as the condition evaluates to `True` or `1`.

```
while condition :
    body
```

## 12.9 Control Structures

```
if condition :
    body
```

```
if condition :
    body1
else :
    body2
```

```
if condition1 :
    body1
elif condition2 :
    body2
else :
    body3
```

## 12.10 Classes

```
class ClassName:
    '''Documentation string'''
    def __init__( self [,args] ):
        .
        .
        .
```

Inheritance is a little bit more tricky:

```
class Base( object ):
    def __init__( self, param ):
        print "Base:", param
    def method( self, param ):
        print "Base.method:", param

class Derived( Base ):
    def __init__( self, param ):
        super( Derived, self ).__init__( param )
```

```
        print "Derived:", param
    def method( self, param ):
        Base.method( self, param )
        print "Derived.method:", param
```

## 12.11 Functions

```
def fxn(arg) :
    body
    return None
```

## 12.12 Asymptote

Asymptote is a powerful vector drawing program that can be accessed by itself interactively, using asy in the command line, through scripts, though LaTeX files, or through the Python interpreter. This is how we use Asymptote through the Python interpreter:

```
>>> import sys
>>> sys.path.append('/usr/share/asymptote')
>>> from asymptote import *
>>> a = asy()
Asymptote session is open.  Available methods are:
    help(), size(int), draw(str), fill(str), clip(str),
    label(str), shipout(str), send(str), erase()
>>> > a.size(200)
>>> > a.draw('unitcircle')
>>> > a.send('draw(unitsquare)')
```

## 12.13 `break`

This statement breaks out of the closest loop or if-statement. If the if-statement has an else clause, that, too is skipped.

## 12.14 Binary Data

The code below will put n values of type double into the one dimensional NumPy ndarray.

```
from scipy.io.numpyio import fread, fwrite

fid = open( 'filename.fmt', 'rb' )
data_type = 'd'
no_elements = n
data = fread( fid, no_elements, data_type )
```

Below we have the values that data_type may assume, with the associated numbers of bytes they use.

```
         1 byte  => b, B, c, S1
         2 bytes => h, H
         4 bytes => f, i, I, l, u4
         8 bytes => d, F
        16 bytes => D
```

For our purposes we'll use 'd', for the double or 64 bit (8 byte) data type.

## 12.15 Calling One Script From Another

```
# one.py

import two
import three

def main():
    body

main() # run code in one
two.main() # run code in two
three.main() # run code in three
```

## 12.16 Colormap

These set of functions will take a value between `min` and `max` and return a hex value for the color in the Jet colormap.

```
def rgbscale( x, oldmin, oldmax, newmin, newmax ):
    oldscale = float( oldmax - oldmin )
    newscale = float( newmax - newmin )
    ratio = float( newscale / oldscale )
    s = x
    s -= oldmin
    s *= ratio
    s += newmin
    return s

def interp_blue( x ):
    if x >= 0.0 and x < 0.11:
        c = hex( int( rgbscale( x, 0.0, 0.11, 127, 255 ) ) )
    if x >= 0.11 and x < 0.34:
        c = hex( int( rgbscale( x, 0.11, 0.34, 255, 255 ) ) )
    if x >= 0.34 and x < 0.65:
        c = hex( int( rgbscale( x, 0.34, 0.65, 255, 0.0 ) ) )
    if x >= 0.65 and x <= 1.0:
        c = hex( int( rgbscale( x, 0.65, 1.0, 0.0, 0.0 ) ) )
    c = str(c[2:])
    if len( c ) == 1: c = '0'+c
    return c

def interp_green( x ):
    if x >= 0.0 and x < 0.125:
        c = hex( int( rgbscale( x, 0.0, 0.125, 0.0, 0.0 ) ) )
    if x >= 0.125 and x < 0.375:
        c = hex( int( rgbscale( x, 0.125, 0.375, 0.0, 255 ) ) )
    if x >= 0.375 and x < 0.64:
        c = hex( int( rgbscale( x, 0.375, 0.64, 255, 255 ) ) )
    if x >= 0.64 and x < 0.91:
        c = hex( int( rgbscale( x, 0.64, 0.91, 255, 0.0 ) ) )
    if x >= 0.91 and x <= 1.0:
        c = hex( int( rgbscale( x, 0.91, 1.0, 0.0, 0.0 ) ) )
    c = str(c[2:])
    if len( c ) == 1: c = '0'+c
    return c

def interp_red( x ):
    if x >= 0.0 and x < 0.35:
        c = hex( int( rgbscale( x, 0.0, 0.35, 0.0, 0.0 ) ) )
    if x >= 0.35 and x < 0.66:
```

```
        c = hex( int( rgbscale( x, 0.35, 0.66, 0.0, 255 ) ) )
    if x >= 0.66 and x < 0.89:
        c = hex( int( rgbscale( x, 0.66, 0.89, 255, 255 ) ) )
    if x >= 0.89 and x <= 1.0:
        c = hex( int( rgbscale( x, 0.89, 1.0, 255, 127 ) ) )
    c = str(c[2:])
    if len( c ) == 1: c = '0'+c
    return c

def value_to_hex( x, sensor_min=60, sensor_max=100 ):
    # takes value in range [0,1] returns hex value corresponding to a jet value
    n = ( x - sensor_min ) / float( sensor_max - sensor_min )
    return '#'+interp_red(n)+interp_green(n)+interp_blue(n)
```

## 12.17 Command Line Arguments

The command line arguments are kept in the list sys.argv. The first element of the list is the filename.py, and the rest of the elements are the arguments. You need to import the sys module.

```
import sys

for arg in sys.argv:
    print arg
```

## 12.18 `continue`

This statement skips to the next iteration of the closest loop. While break breaks out of a loop, continue simply skips the rest of the loop code and goes to the next iteration.

## 12.19 Convex Hull

Here is some code to determine the convex hull for a set of points:

```
import numpy as n, pylab as p, time

def _angle_to_point(point, centre):
    '''calculate angle in 2-D between points and x axis'''
    delta = point - centre
    res = n.arctan(delta[1] / delta[0])
    if delta[0] < 0:
        res += n.pi
    return res


def _draw_triangle(p1, p2, p3, **kwargs):
    tmp = n.vstack((p1,p2,p3))
    x,y = [x[0] for x in zip(tmp.transpose())]
    p.fill(x,y, **kwargs)
    #time.sleep(0.2)


def area_of_triangle(p1, p2, p3):
    '''calculate area of any triangle given co-ordinates of the corners'''
    return n.linalg.norm(n.cross((p2 - p1), (p3 - p1)))/2.


def convex_hull(points, graphic=True, smidgen=0.0075):
    '''Calculate subset of points that make a convex hull around points
```

```
    Recursively eliminates points that lie inside two neighbouring points until only convex hull is remaining.

    :Parameters:
        points : ndarray (2 x m)
            array of points for which to find hull
        graphic : bool
            use pylab to show progress?
        smidgen : float
            offset for graphic number labels - useful values depend on your data range

    :Returns:
        hull_points : ndarray (2 x n)
            convex hull surrounding points
    '''
        if graphic:
            p.clf()
            p.plot(points[0], points[1], 'ro')
        n_pts = points.shape[1]
        assert(n_pts > 5)
        centre = points.mean(1)
        if graphic: p.plot((centre[0],),(centre[1],),'bo')
        angles = n.apply_along_axis(_angle_to_point, 0, points, centre)
        pts_ord = points[:,angles.argsort()]
        if graphic:
            for i in xrange(n_pts):
                p.text(pts_ord[0,i] + smidgen, pts_ord[1,i] + smidgen, \
                       '%d' % i)
        pts = [x[0] for x in zip(pts_ord.transpose())]
        prev_pts = len(pts) + 1
        k = 0
        while prev_pts > n_pts:
            prev_pts = n_pts
            n_pts = len(pts)
            if graphic: p.gca().patches = []
            i = -2
            while i < (n_pts - 2):
                Aij = area_of_triangle(centre, pts[i],     pts[(i + 1) % n_pts])
                Ajk = area_of_triangle(centre, pts[(i + 1) % n_pts], \
                                       pts[(i + 2) % n_pts])
                Aik = area_of_triangle(centre, pts[i],     pts[(i + 2) % n_pts])
                if graphic:
                    _draw_triangle(centre, pts[i], pts[(i + 1) % n_pts], \
                                   facecolor='blue', alpha = 0.2)
                    _draw_triangle(centre, pts[(i + 1) % n_pts], \
                                   pts[(i + 2) % n_pts], \
                                   facecolor='green', alpha = 0.2)
                    _draw_triangle(centre, pts[i], pts[(i + 2) % n_pts], \
                                   facecolor='red', alpha = 0.2)
                if Aij + Ajk < Aik:
                    if graphic: p.plot((pts[i + 1][0],),(pts[i + 1][1],),'go')
                    del pts[i+1]
                i += 1
                n_pts = len(pts)
            k += 1
        return n.asarray(pts)
```

## 12.20 Convolution

Image convolution is a very common operation in digital signal processing.

```
from numpy.fft import fft2, ifft2
```

```
def myconvolve( image1, image2, MinPad=True, pad=True ):

    #The size of the images:
    r1,c1 = image1.shape
    r2,c2 = image2.shape

    #MinPad results simpler padding,smaller images:
    if MinPad:
        r = r1+r2
        c = c1+c2
    else:
        #if the Numerical Recipies says so:
        r = 2*max(r1,r2)
        c = 2*max(c1,c2)

    #For nice FFT, we need the power of 2:
    if pad:
        pr2 = int(log(r)/log(2.0) + 1.0 )
        pc2 = int(log(c)/log(2.0) + 1.0 )
        rOrig = r
        cOrig = c
        r = 2**pr2
        c = 2**pc2

    # numpy fft has the padding built in, which can save us some steps
    # here. The thing is the s(hape) parameter:
    fftimage = fft2( image1, s=(r,c) ) * fft2( image2[::-1,::-1], s=(r,c) )

    if pad:
        return ifft2( fftimage )[:rOrig,:cOrig].real
    else:
        return ifft2( fftimage ).real
```

Cross-correlation and convolution have a close relationship. Using the code above, we have the following relationships:

```
scipy.signal.signaltools.convolve( x[:,0], y[:,0] )
    ||
numpy.convolve( x[:,0], y[:,0] )
    ||
myconvolve( x, y[::-1] )[:,0]
```

```
scipy.signal.signaltools.correlate( x[:,0], y[:,0] )
    ||
numpy.correlate( x[:,0], y[:,0], 'full' )
    ||
myconvolve( x, y )[:,0]
```

```
r( 'x <- read.table"(x."dat)' )
r( 'y <- read.table"(y."dat)' )
ccf = r( 'ccf(x,y)' )
ccf[ 'acf' ]
    ||
x = ( x - np.mean(x) )/( np.std(x) * len(x) )
y = ( y - np.mean(y) )/( np.std(y) )
cross_correlation = myconvolve( x, y[::-1] )[:,0]
correlation = cross_correlation[ len(x-)1 ] = np.corrcoef(x,y)[0][1]
```

We see that by reversing the second signal we obtain convolution, and by not reversing it we obtain cross-correlation... sort of. To get cross-correlation the way that R and other programs do, we need to first normalize the signals, thus we have the third block.

Usually, we'd like to use a particular convolution kernel for the second argument/input to blur the image, or find certain types of edges. Here are some useful kernels:

```
box_blur = np.array([[1/9.,1/9.,1/9.],[1/9.,1/9.,1/9.],[1/9.,1/9.,1/9.]])

gauss_blur = np.array([0,0,0,5,0,0,0])
gauss_blur = np.vstack((gauss_blur,np.array([0,5,18,32,18,5,0])))
gauss_blur = np.vstack((gauss_blur,np.array([0,18,64,100,64,18,0])))
gauss_blur = np.vstack((gauss_blur,np.array([5,32,100,100,100,32,5])))
gauss_blur = np.vstack((gauss_blur,np.array([0,18,64,100,64,18,0])))
gauss_blur = np.vstack((gauss_blur,np.array([0,5,18,32,18,5,0])))
gauss_blur = np.vstack((gauss_blur,np.array([0,0,0,5,0,0,0])))

hor_line_det = np.array([[-1,-1,-1],[2,2,2],[-1,-1,-1]])
ver_line_det = np.array([[-1,2,-1],[-1, 2,-1],[-1, 2,-1]])
p45_line_det = np.array([[-1,-1, 2],[-1, 2,-1],[ 2,-1,-1]])
n45_line_det = np.array([[2,-1,-1],[-1, 2,-1],[-1,-1, 2]])

edge_det = np.array([[-1,-1,-1],[-1,8,-1],[-1,-1,-1]])

sobel_hor = np.array([[-1,-2,-1],[0,0,0],[1,2,1]])
sobel_ver = np.array([[-1,0,1],[-2,0,2],[-1,0,1]])
```

The Sobel edge detectors are less sensitive to noise since they smooth as they go.

```
laplacian = np.array([[ 0,-1, 0],[-1, 4,-1],[ 0,-1, 0]])
lpl_w_diag = np.array([[-1,-1,-1],[-1, 8,-1],[-1,-1,-1]])
```

The Laplacian calculates the second derivative, but it is very sensitive to noise. Smoothing the kernel before applying the Laplacian is helpful. This is achieved with Laplacian of the Gaussian, below:

```
lpl_of_gauss = np.array([[0,0,-1,0,0],[0,-1,-2,-1,0],[-1,-2,16,-2,-1]])
lpl_of_gauss = np.vstack((lpl_of_gauss, np.array([0,-1,-2,-1,0])))
lpl_of_gauss = np.vstack((lpl_of_gauss, np.array([0,0,-1,0,0])))
```

**Caveat:** The Matlab/Octave function `conv2()` behaves differently than the NumPy function `convolve()`. Matlab will convolve a two dimensional image or matrix with a one dimensional filter. For two matrices, **A** and **B**, having dimensions `<ma,na>` and `<1=mb,nb>`, respectively, `conv2()`, using the `mode='same'` option, will produce a matrix **C** having dimensions the same as **A**. The NumPy convolution function will produce an error saying that the matrices need to have the same number of dimensions.

According to the Matlab documentation, `conv2()` computes **C** as:

$$c(n_1, n_2) = \sum_{k_1=-\infty}^{\infty} \sum_{k_1=-\infty}^{\infty} a(k_1, k_2) b(n_1 - k_1, n_2 - k_2) \tag{12.1}$$

I've implemented the following function in Python to produce a similar effect. *Make sure that the image is input as* **a** *and the one dimensional convolution kernel is input as* **b**.

```
def conv2( a, b ):
    a = array( a )
    b = array( b )
    ma, na = a.shape
    mb, nb = b.shape
    m = ma+mb
    n = na+nb
    c = zeros( ( m,n ) )
    for n1 in range( m ):
        for n2 in range( n ):
            s = 0.0
            for k1 in range( -m,m ):
                for k2 in range( -n,n ):
                    if 0<=k1<ma and 0<=k2<na and 0<=n1-k1<mb and 0<=n2-k2<nb :
                        s += a[k1,k2] * b[n1-k1,n2-k2]
            c[n1,n2] = s
    cntrx=floor(mb/2)
```

```
    cntry=floor(nb/2)
    c = c[cntrx:cntrx+ma,cntry:cntry+na]
    return c
```

## 12.21 Covariance Matrices

The `np.cov()` function estimates the covariance of set of an mxn matrix. The rows of the matrix are interpreted as variables, and the columns are interpereted as observations. The covariance matrix is then has dimensions mxm. The `(i,j)` element of the matrix describes the covariance of variables `i` and `j`. The main diagonal represents the variance of each variable. If a value is negative, then the two variables are negatively correlated.

```
>>> import numpy as np
>>> x = [-2.1, -1,  4.3]
>>> y = [3,  1.1,  0.12]
>>> X = np.vstack((x,y))
>>> print np.cov(X)
[[ 11.71        -4.286     ]
 [ -4.286        2.14413333]]
>>> print np.cov(x, y)
[[ 11.71        -4.286     ]
 [ -4.286        2.14413333]]
>>> print np.cov(x)
11.71
```

By default, NumPy calculates the covariance using the sample definition, i.e., it uses the (N-1) term in the denominator. If the bias parameter is set to 1, then the covariance is normalized by (N) instead.

```
>>> import numpy as np
>>> a = [1,3,5,7,8]
>>> b = [1,2,3,4,6]
>>> np.cov( a, b )
array([[ 8.2,  5.3],
       [ 5.3,  3.7]])
>>> np.cov( a, b, bias=1 )
array([[ 6.56,  4.24],
       [ 4.24,  2.96]])
```

## 12.22 cPickle

Python allows you to read and write Python data structures to ASCII or binary files using the pickle and cPickle modules. Since cPickle is written in C, it is about 1000 times faster than pickle.

```
import cPickle

dst = open( 'filename.pkl', 'w' )
cPickle.dump( variable, dst, -1 )

src = open( 'filename.pkl', 'r' )
variable = cPickle.load( src )
```

It took two minutes to read two hyperspectral images having 101,580,800 floats each, but it only took six seconds to pickle and unpickle all of that data.

## 12.23 Cross Product

Sometimes it is useful to have the cross product of a set of elements. Unfortunately, the Standard Library does not support the corss product of two sets. Below we present a list comprehension to produce a list of tuples representing the corss product of two lists.

```
>>> cross_product = [(x,y) for x in a for y in b]
>>> another_form = [(x,y) for x in a for y in b if x != y] # sometimes useful
>>> yet_another = [(x,y) for x in a for y in b if x > y] # also useful
```

The second form eliminates pairs that have the same element for the first and second coordinates; think of this as the coordinates of a matrix sans the main diagonal. The third form produces the upper diagonal coordinates of the matrix.

## 12.24 Cython

Cython is a superset of Python. Its source code gets translated into optimized C/C++ code and compiled as Python extension modules. This allows one sometimes dramatically speed up some functions or subroutines.

Starting with a Python program that calculates a convolution:

```
def conv2( a, b ):
    a = array( a )
    b = array( b )
    ma, na = a.shape
    mb, nb = b.shape
    m = ma+mb
    n = na+nb
    c = zeros( ( m,n ) )
    for n1 in range( m ):
        for n2 in range( n ):
            s = 0.0
            for k1 in range( -m,m ):
                for k2 in range( -n,n ):
                    if 0<=k1<ma and 0<=k2<na and 0<=n1-k1<mb and 0<=n2-k2<nb :
                        s += a[k1,k2] * b[n1-k1,n2-k2]
            c[n1,n2] = s
    cntrx=floor(mb/2)
    cntry=floor(nb/2)
    c = c[cntrx:cntrx+ma,cntry:cntry+na]
    return c
```

We add the following:

```
#!\usr\bin\env python
# conv2.py

def conv2( a, b ):
    a = array( a )
    b = array( b )
    ma, na = a.shape
    mb, nb = b.shape
    m = ma+mb
    n = na+nb
    c = zeros( ( m,n ) )
    for n1 in range( m ):
        for n2 in range( n ):
            s = 0.0
            for k1 in range( -m,m ):
                for k2 in range( -n,n ):
                    if 0<=k1<ma and 0<=k2<na and 0<=n1-k1<mb and 0<=n2-k2<nb :
                        s += a[k1,k2] * b[n1-k1,n2-k2]
            c[n1,n2] = s
```

```
    cntrx=floor(mb/2)
    cntry=floor(nb/2)
    c = c[cntrx:cntrx+ma,cntry:cntry+na]
    return c
```

We then write:

```
#!/usr/bin/env python
# conv2.pyx

import numpy as np
cimport numpy as np

cdef extern from "math.h":
    int floor( float x )

def conv2( np.ndarray a, np.ndarray b ):

    cdef int ma = a.shape[0]
    cdef int na = a.shape[1]
    cdef int mb = b.shape[0]
    cdef int nb = b.shape[1]
    cdef int m  = ma + mb
    cdef int n  = na + nb
    cdef np.ndarray c = np.zeros( ( m,n ), dtype=np.float )

    cdef int n1 = 0
    cdef int n2 = 0
    cdef int k1 = 0
    cdef int k2 = 0

    cdef float s = 0.0

    cdef cntrx = floor( mb / 2.0 )
    cdef cntry = floor( nb / 2.0 )

    for n1 from 0 <= n1 < m :
        for n2 from 0 <= n2 < n :
            s = 0.0
            for k1 from -m <= k1 < m :
                for k2 from -n <= k2 < n :
                    if 0<=k1<ma and 0<=k2<na and 0<=n1-k1<mb and 0<=n2-k2<nb :
                        s += a[k1,k2] * b[n1-k1,n2-k2]
            c[n1,n2] = s

    c = c[cntrx:cntrx+ma,cntry:cntry+na]

    return c
```

We also write a `setup.py` file:

```
from distutils.core import setup
from distutils.extension import Extension
from Cython.Distutils import build_ext

setup(
    cmdclass = {'build_ext': build_ext},
    ext_modules = [Extension("conv2", ["conv2.pyx"])]
)
```

We then build the extension with the following command:

```
$ python setup.py build_ext --inplace
$
```

Et voila, we can now import the function and run it in a Python program with the speed of a C program. For two matrices, **A** <65x200> and **B** <65x1>, this optimization is about ∼112 times faster.

## 12.25 Discrete Wavelet Transforms

We can use the PyWavelets package to perfrom the discrete wavelet transform.

```
import pywt, numpy

x = [ 3, 7, 1, 1, -2, 5, 4, 6 ]
cA,cD = pywt.dwt(x,'db4')

x = numpy.random.random((4,4))
cA,(cH,cV,cD) = pywt.dwt2(x,'db4')
```

And this produces our wavelet transform for one and two dimensions.

## 12.26 Distance Matrix

This will produce a distance matrix using the Euclidean distance metric.

```
#!/usr/bin/env python

import numpy as np

def euclideanDistance( data ):
    amt = len( data )
    distance = np.sqrt( np.sum( data[i]**2 for i in range( amt ) ) )
    return distance

def calcDistanceMatrix( data, dist, maxDiagonal=False ):
    data = np.array( data )
    records, features = data.shape
    tmp= [None]*features
    for i in range( features ):
        dim = data[:,i]
        tmp[i] = dim - np.reshape( dim, (len(dim),1) )
    dmat = dist( tmp )
    if maxDiagonal == True :
        dmat = dmat + np.identity( len(dim))*dmat.max()
    return dmat
```

## 12.27 Eigenvectors and Eigenvalues

```
import numpy
D, V = numpy.linalg.eig( a )
```

Okay, so `D` is a vector of eigenvalues, and `V` is a matrix of eigenvectors, where the ith column of `V` corresponds to the ith eigenvalue in `D`.

Often, we need to sort the eigenvalues from highest to lowest, and sort the eigenvectors, the columns of `V`, accordingly. We do that this way:

```
idx = D.argsort()
D = D[idx]
V = V[:,idx]
```

This will sort the eigenvalues and eigenvectors from least to greatest. Usually we need to swap that around, so we do that:

```
D = D[::-1]
V = V[:,::-1]
```

## 12.28 eval, compile, exec, and execfile

We can use eval() to evaluate strings as Python code. These strings may be literals, simple expressions, or built-in functions.

```
>>> eval( '2+2' )
4
```

We can use compile() and exec() together to compile larger pieces of code.

```
NAME = "script."py
BODY = """print 'one fish, two fish, red fish, blue fish'"""
code = compile( BODY, NAME, ""exec )
print code   # returns a code object
#  <code object <module> at 0x7f5745d60378, file "script.py", line 1>
exec code   # executes the code
#  one fish, two fish, red fish, blue fish
```

The function execfile() is a shortcut for loading code from a file, compiling it, and exectuing it.

```
execfile("hello.py")

def EXECFILE(filename, locals=None, globals=None):
    exec compile(open(filename).read(), filename, "exec") in locals, globals

EXECFILE("hello.py")
```

## 12.29 File Systems

If you need to get a list of files from the current directory, you can use the `glob` module, which is explained elsewhere here. If you need to go through multiple subdirectories searching for CSV or other data files, you can use the following code.

```
def get_filenames():
    root = os.getcwd()
    fn = list()
    tree = os.walk('.')
    for i in tree:
        if i[1] == []:
            dir = i[0][1:]
            for j in i[2]:
                if j[-4:] == '.csv':
                    filename = root+dir+'/'+j
                    fn.append( filename )
    return fn
```

The MVP in the code above is the `os.walk()` function that returns a generator for the directory. The `os.getcwd()` gets the pathname of the current working directory.

## 12.30 Finding Peaks or Troughs in Data

This is a nice one-liner I found in some code.

```
peaks   = find( diff( sign( diff( x ) ) ) < 0 ) + 1
troughs = find( diff( sign( diff( x ) ) ) > 0 ) + 1
```

You can then calculate the upper envelope by doing the following,

```
last = len( x )
p = [ 0 ] + list( peaks ) + [ last-1 ]
y = [ x[i] for i in p ]
x = p[:]
upper_envelope = np.interp( range(last), x, y )
```

## 12.31 Flask

Flask is a web framework that is newer than Web.py, but it is very active, and there is more documentation. It also uses a different template system, `Jinja2`.

In this project, I needed to host a web page that linked to log files. First, I made `run.py`

```
from flask import Flask, render_template
app = Flask(__name__)

# decorator sets us URL structure
@app.route('/')
def index():
    return render_template('index.html')

@app.route('/data')
def data():
    return render_template('data.txt')

if __name__=='__main__':
    app.run()
```

Then, in a directory named *templates* we have `index.html`:

```
<!DOCTYPE HTML>
<HTML>
<HEAD>
<META HTTP-EQUIV="refresh" CONTENT="5" >
</HEAD>
<BODY>
<A HREF="data" TARGET="_blank"><BUTTON>data</BUTTON></A>
</BODY>
</HTML>
```

And then `data.txt` is just some data file that is floating around in the *templates* directory.

## 12.32 Fourier Transforms and Periodograms

A common use of Fourier transforms is to find the frequency components of a signal buried in a noisy time domain signal. Consider data sampled at 1000 Hz. Form a signal containing a 50 Hz sinusoid of amplitude 0.7 and 120 Hz sinusoid of amplitude 1 and corrupt it with some zero-mean random noise:

```
#!/usr/bin/env python
'''
Input: (1) filename in directory of data to be analysed
       (2) 'period' or 'frequency' (sans '')
       (3) x-beginning, e.g. 0
       (4) x-ending, e.g. 40, for forty years, or whatever the period is
Output: A nice graph!
```

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

f = open( sys.argv[1], 'r' )
f = [ i.split() for i in f.readlines() ]
f = [ [ float( j ) for j in i ] for i in f ]
x = [ f[i][0] for i in range( len( f ) ) ]
y = [ f[i][1] for i in range( len( f ) ) ]

x, y = np.array( x ), np.array( y )

xd = x[1] - x[0]
Y = np.fft.fft( y )
n = len( Y )
power = np.abs(Y[1:(n/2)])**2
freq = np.fft.fftfreq( n/2, d=2*xd )
period = 1./freq
fig = plt.figure()
ax = fig.add_subplot( 111 )
if sys.argv[2] == 'freq' :
    ax.plot( freq[1:], power, '-+' )
    ax.set_xlabel('Frequency')
elif sys.argv[2] == 'period' :
    ax.plot( period[1:], power, '-+' )
    ax.set_xlabel('Period')
else :
    print "fail. please say 'freq' or 'period'."
xb, xe = float( sys.argv[3] ), float( sys.argv[4] )
ax.set_xlim(xb,xe)
ax.set_ylabel('Power : |FFT|**2')
plt.show()
```

This is kind of embarassing. You can alternatively view the periodogram of some data by using the psd() pylab plotting function.

```
from pylab import *

Fs = 1000
t = arange( 0,0.3,1/fs)
x = cos( 2 * pi * t * 200 ) + 0.1 * rand( t.size )
psd( x, 512, Fs, sides='onesided' )
```

Is equivalent to the MATLAB operation:

```
Fs = 1000;
t = 0:1/Fs:.3;
x = cos(2*pi*t*200)+0.1*randn(size(t));
periodogram(x,[],'onesided',512,Fs)
```

If you want to animate the power spectral density of a bunch of A-scans or something, you can use the following:

```
from matplotlib.pylab import *
import time

d = some_data

ion()

tmp = np.log10( psd( d[:,0], hold=True, visible=False )[0] )
line, = plot( tmp, 'b-' )
draw()
```

```
for i in range( 1,30 ):
    line.set_ydata( np.log10( psd( d[:,i], hold=True, visible=False )[0] ) )
    title('A-scan Power Spectral Density')
    xlim( 0, len(tmp)-1 )
    ylim( 0,10 )
    grid(False)
    time.sleep(0.1)
    draw()
```

## 12.33 Fast Fourier Transform Direct Digital Integration

This does integration for accelerometers that move very slowly.

```
def fftddi( x ):
    N = x.size
    Ak = scipy.fftpack.fft( x )
    j = np.complex(0,1) ; pi = np.pi ; e = np.e
    Vk = [  Ak[i]/(j*2.0*pi*i+0.005) for i in range(N) ]
    v = lambda r : np.sum( [ Vk[k]*e**(j*2*pi*k*r/float(N)) for k in range(N) ] )
    vr = np.array( [ v(i).real for i in range(N) ] )
    return vr
```

## 12.34 Formatting Strings

String formatting in Python uses the same syntax as the sprintf function in C. Here are three basic examples, using the % operator.

```
>>> i,j,k = 1,2,3
>>> print '(%i,%i,%i)' % (i,j,k)
(1,2,3)
>>>
>>> i,j,k = 1,2,3
>>> print '( %f, %f, %f )' % (i,j,k)
( 1.0, 2.0, 3.0 )
>>>
>>> s1,s2 = 'ham','eggs'
>>> print '%s and %s' % (s1,s2)
ham and eggs
```

Here are some examples using the `format()` method.

```
print '{0} {1} {2}'.format( 'spam', 'ham', 'eggs' )
print '{person} {verb} {obj}.'.format(verb='like',obj='swimming',person='I')
stuff = ['ham', 'eggs', 'bacon' ]
print '{0} {1} {2}'.format( *stuff )
```

Note the fancy "splat" operator above. We can also do some number formatting.

```
print '{0:3.2f}'.format( num1 ).rjust(6),
print '{0:1.1f}'.format( num2 ).rjust(3)
```

The above has one, six space column of right justified numbers, having at least at most 3 digits in front of the decimal, and at most 2 digits after. The second column is 3 spaces wide, and it has at most one digit on either side of the decimal.

## 12.35 Information About Objects

You can use dir() quickly, or you can get more information by importing the inspect module.

```
>>> import inspect
>>> inspect.getmembers( object )
>>> dir(object)
```

## 12.36 Histogram Equalization

Histogram equalization is used to increase contrast in over or under-developed images. I found this function on the internet at this guy's computer vision blog `http://www.janeriksolem.net/2009/06/histogram-equalization-with`

```
from PIL import Image
from numpy import *

def histeq(im,nbr_bins=256):
   #get image histogram
   imhist,bins = histogram(im.flatten(),nbr_bins,normed=True)
   cdf = imhist.cumsum() #cumulative distribution function
   cdf = 255 * cdf / cdf[-1] #normalize
   #use linear interpolation of cdf to find new pixel values
   im2 = interp(im.flatten(),bins[:-1],cdf)
   return im2.reshape(im.shape), cdf
```

## 12.37 HDF5 with Python and MATLAB

Import PyTables as `tables`

```
import numpy as np
from tables import *
fileh = openFile( 'b.h5', mode='w' )
root = fileh.root
group1 = fileh.createGroup( root, 'group1' )
array1 = fileh.createArray( '/group1', 'array1', [1,2,3,4] )
fileh.close()
```

Then in MATLAB you do the following:

```
>> hdf5read( 'b.h5', '/group1/array1' )
>> ans =

           1
           2
           3
           4

>>
```

## 12.38 Installing Packages

Use pip to install packages. Mnemomic: Pip installs packages. Pip is an improvement upon `easy_install`.

```
$ sudo easy_tinstall pip
$ sudo pip install SomePackage
```

## 12.39 IPython

IPython is an interactive python interpreter with greater functionality than the original Python interpreter.

### 12.39.1 Editing Code

One neat trick is that you can open your favorite editor from within IPython. You will ofcourse have to edit the ipythonrc file in the .ipythonrc folder in your home directory to tell it what your favorite editor is.

```
In [1]: edit file
```

### 12.39.2 Accessing the Shell

You can also send commands to the underlying shell by prefacing them with a bang (!) Some commands ( like `ls`, `cd`, and `pwd` ) are magic, and don't require a bang.

### 12.39.3 Running Python Scripts

If you want to run a python program from the IPython terminal, then you have two options:

```
In [1]: ! python program.py

In [2]: run program.py
```

### 12.39.4 Timing Scripts

You can also time programs by prefacing them with the `%time` magic:

```
In [1]: %time run program.py
CPU times: user 0.03 s, sys: 0.01 s, total: 0.04 s
Wall time: 0.06 s

In [2]:
```

### 12.39.5 Interactive Plotting

You can start IPython with the `--pylab` option to get some really convenient plotting options. See *Pylab* for more information.

### 12.39.6 Listing Current Variables

You can also get a summary of the variables in use by using the %whos magic function. For example, below we see that we have a number of arrays, with their dimensions, and a module with its nickname and full name. Awesome.

```
In [36]: %whos
Variable   Type       Data/Info
------------------------------
c          ndarray    6x6: 36 elems, type `float64`, 288 bytes
d          ndarray    20x6: 120 elems, type `float64`, 960 bytes
sps        module     <module 'scipy.stats' fro<...>cipy/stats/__init__.pyc'>
x0         ndarray    20x1: 20 elems, type `float64`, 160 bytes
x1         ndarray    20x1: 20 elems, type `float64`, 160 bytes
x2         ndarray    20x1: 20 elems, type `float64`, 160 bytes
y0         ndarray    20x1: 20 elems, type `float64`, 160 bytes
y1         ndarray    20x1: 20 elems, type `float64`, 160 bytes
y2         ndarray    20x1: 20 elems, type `float64`, 160 bytes
```

### 12.39.7 Logging

You can log your sessions in IPython by using the following:

```
In [1]: %logstart filename.py
```

If there's an existing logfile you want to append to:

```
In [1]: %logstart filename.py append
```

To pause logging:

```
In [2]: %logoff
Switching logging OFF

In [3]: print 'unter zee radar'
unter zee radar

In [4]: logon
Switching logging ON
```

You can name the following file `05_log.py` and add it to you `/home/name/.ipython/profile_default/startup` file. This should produce daily, time-stamped log-files.

```
from time import strftime
import os.path

ip = get_ipython()

ldir = ip.profile_dir.log_dir
fname = 'log-' + ip.profile + '-' + strftime('%Y-%m-%d') + ".py"
filename = os.path.join(ldir, fname)
notnew = os.path.exists(filename)

try:
  ip.magic_logstart('-o %s append' % filename)
  if notnew:
      ip.logger.log_write("# ================================\n")
  else:
      ip.logger.log_write( "#!/usr/bin/env python\n" )
      ip.logger.log_write( "# " + fname + "\n" )
      ip.logger.log_write( "# IPython automatic logging file\n" )
  ip.logger.log_write( "# " + strftime('%H:%M:%S') + "\n" )
  ip.logger.log_write( "# ================================\n" )
  print " Logging to "+filename
except RuntimeError:
    print " Already logging to "+ip.logger.logfname
```

### 12.39.8 Built-In History Variables

You can re-use the last three outputs using underscores:

```
In [1]: 2+2
Out[1]: 4

In [2]: _+4
Out[2]: 8

In [3]: __+_
Out[3]: 12

In [4]: ___*4
Out[4]: 16
```

Extending this theme, you can use specific line numbers after the underscore, or you can repeat the input at certain lines, using the %rep function, for revision, or immediate execution.

```
In [1]: 2+2
Out[1]: 4

In [2]: _1
Out[2]: 4

In [3]: %rep 1

In [4]: 2+2# <-- blinking cursor
```

### 12.39.9 Notebook

To use the notebook, you need to install/build ZeroMQ from their site, and install the IPython-0.12. (Nbd.) The notebook allows you to work even more interactively, if you can even imagine that. It's like the Sage notebook, but it gives you a Python script, in addition to a notebook file. Note that *numpy* and *pylab* are installed completely.

To produce a script and use pylab in-line use the following:

```
$ ipython notebook --script --pylab=inline
```

This will open a browser and you can pick from the notebooks in your current directory, or begin a new notebook. It will also produce a script of your session.

You can add comments by using `CTRL-m` and you can even use LaTeXin-line or display math using `$…$` or `$$…$$`.

### 12.39.10 IPCluster

First, be sure that you have the proper dependencies installed.

```
$ sudo easy_install ipython[zmq] # installs pyzmq
```

The simplest way to start a controller and a number of engines is to type the following.

```
$ ipcluster start --n=4
```

In a IPython session you can do the following:

```
In [1]: from IPython.parallel import Client

In [2]: c = Client()

In [3]: c.ids
Out[3]: set([0, 1, 2, 3])

In [4]: c[:].apply_sync(lambda:'Hello World!')
Out[4]: [ 'Hello World!', 'Hello World!', 'Hello World!', 'Hello World!' ]
```

We can write functions and broadcast them in the following manner:

```
lview = c.load_balanced_view()

@lview.parallel()
def spitout( text ):
    return text

spitout.block = True

spitout.map( range( 128 ) )
```

Simply. Stunning.
Lately, I have been doing the following:

1. On the IPython dashboard, click on the Clusters tab
2. Start 4 engines on the Default profile
3. Open a notebook

Then in a notebook cell,

```
from TPython.parallel import Client
c = Client( profile='default' )
c.ids # [ 0, 1, 2, ... , n-1 ]
```

We can assign single engines single tasks by doing the instructions in the following listing. Note that we must pass the engine denoted by `c[0]` to `v` before we can use it. As for the arguments, `f` is the name of some function, and `*args` are the arguments to that function.

```
v = c[0]
# ar for asynchronous result
ar = v.apply_async( f, *args )
```

In the previous listing, we used the expression `v.apply_async( f, *args )`, and this can be done several ways. Suppose we have some function, `f`, that takes two inputs, `x` and `y`.

We can distribute a scalar input `y` over an array `x`.

```
x = np.array([0,1,1,2,3,5])
y = 2
f = lambda x, y : x + y
v = c[0]
ar = v.apply_async( f, x, y )
ar.get() # --> [2,3,3,4,5,7]
```

We can assign the inputs in an element-wise fashion.

```
x = np.array([0,1,1,2,3,5])
y = np.array([0,2,2,4,6,10])
f = lambda x, y : x + y
v = c[0]
ar = v.apply_async( f, x, y )
ar.get() # --> [0,3,3,6,9,15]
```

Finally, we can also use the *splat* operator

```
args = [ x, y ]
v = c[0]
ar = v.apply_async( f, *args ) # et voila!
```

Another thing is (a)synchronous and (non)blocking behavior. With synchronous or blocking behavior, the whole program waits until the job is finished. With asynchronous or non-blocking behavior, the job works in the background, but you have to poll the engines to see if they're done. This is achieved through the `.ready()` and `.get()` functions, as in

```
v0, v1 = c[:2]
r0 = v0.apply_async( f, *args )
r1 = v1.apply_async( f, *args )
while not( r0.ready() and r1.ready() ):
    pass
print r0.get(), r1.get()
```

We can use blocking execution using the `block` attribute of the engine,

```
v = c[0]
v.block = True
r = v.apply( f, *args )
```

which is equaivalent to,

```
v = c[0]
r = v.apply_sync( f, *args )
```

or we can use non-blocking execution,

```
v = c[0]
v.block =False
r = v.apply( f, *args )
```

which is equivalent to,

```
v = c[0]
r = v.apply_async( f, *args )
```

## 12.40 Glob

This will return a list of the filenames in the current directory.

```
import glob
f = glob.glob('*.pkl')
print f
```

## 12.41 Going Between Symmetric Matrices and Arrays

We describe some functions to go back and forth between an array, and the upper diagonal of a matrix. The array will be indexed by k, and the matrix will be indexed by i and j. Inexing will begin with zero in both structures. The first function returns the index k, given the matrix indices, i and j. The second function returns the indices (i,j) of the matrix, given the index of the array k.

Consider a matrix given by:

| | | | | |
|---|---|---|---|---|
| 0 | 1 | 3 | 6 | 10 |
| * | 2 | 4 | 7 | 11 |
| * | * | 5 | 8 | 12 |
| * | * | * | 9 | 13 |
| * | * | * | * | 14 |

And an array given by:

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |

```
import numpy as np

def array_index( i, j ):
    k = i + j*(j+1.0)/2.0
    return k

def upper_diagonal_matrix_index( k ):
    j = np.floor( ( -1 + np.sqrt( 1 + 8*k ) )/2.0 )
    i = k - j*(j+1)/2.0
    return (i,j)
```

We next present two similar functions, only these go back and forth between an array and the upper diagonal of a matrix sans the main diagonal. Again, the array will be indexed by k, and the matrix will be indexed by i and j. Inexing will begin with zero in both structures. The first function returns the index k, given the matrix indices, i and j. The second function returns the indices (i,j) of the matrix, given the index of the array k.

Consider a matrix given by:

| | | | | |
|---|---|---|---|---|
| * | 0 | 1 | 3 | 6 |
| * | * | 2 | 4 | 7 |
| * | * | * | 5 | 8 |
| * | * | * | * | 9 |
| * | * | * | * | * |

And an array given by:

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

```
import numpy as np

def sans_main_diagonal_array_index( i, j ):
    k = i + j*(j-1.0)/2.0
    return k

def sans_main_diagonal_upper_diagonal_matrix_index( k ):
    j = np.floor( ( 1 + np.sqrt( 1 + 8*k ) )/2.0 )
    i = k - j*(j-1)/2.0
    return (i,j)
```

This is potentially useful in handling symmetric arrays, like correlation or covariance matrices.

## 12.42 Hierarchical Data Format

This allows you to store data much larger than what is allowed by Numpy, which taps out at various sizes, depending upon your syste.

First you need to download this guy. The module you need is `h5py`.

```
$ sudo easy_install h5py
$
```

If you just want to open and manipulate the data stored in a hd5 file, you can do the following:

```
import h5py

f = h5py.File( 'file.h5', 'r' )
f = f['data']
f[...] # blah. at this point, this guy acts like a numpy array.
```

Since you wouldn't be doing this unless you had ginormous data sets, you'll need to build/extend your data set in chunks, but you have to know how big your chunks are going to be.

```
import h5py
import numpy as np
import pylab

x = np.random.normal( 0, 1, ( 5, 5 ) )

# open an h5 file
f = h5py.File('ext.h5','w')
```

```
# this next line is tricky, the elements of the second tuple, chunks,
# must divide the elements of the first tuple, after the term, 'chunked'
d = f.create_dataset( 'chunked', ( 25, 15 ), np.float32, chunks=( 5, 5 ) )

# then we can assign and update values through d
d[:5,:5] = x

# and view them
pylab.matshow( d )
pylab.show()

# when we're done we close the file handler
f.close()
```

## 12.43 Kernel Density Estimation

Kernel density estimation overcomes some of the shortcomings of histograms. The KDE of a data set is unique, whereas two significantly different data sets can have the same histogram depending on how the bins are chosen.

```
def kde( z, w, xv ):
    '''
    Input:  (z)  index
            (w)  bandwidth
        (xv) data set
    '''
    return np.sum( np.exp(-0.5*((z-xv)/w)**2)/np.sqrt(2*np.pi*w**2) )

def appkde( x, bw, tot, lbl ):
    '''
    Input:  (x)   data, list or array form
            (bw)  bandwidth parameter for kernel density estimation
        (tot) total length of the run
        (lbl) label for graph
    ---------------------------------------------------------------
    Prints out a graph of the kernel density estimation.
    '''
    k = list()
    if type( x ) == str:
        d = np.loadtxt( x, delimiter=',' )
    d = d[:,0]
    else:
        d = np.array( x )
    idx = np.linspace( 0, tot, tot*4 )
    for i in idx:
        k.append( [ i, kde( i, bw, d ) ] )
    k = np.array( k ).T
    plot( k[0], k[1], label=lbl ) ;
    xlim( 0, tot )
```

## 12.44 Kriging

Kriging is a regression technique that originated in geostatistics. It is also called Gaussian Processes. Below, we implement the covariance matrix from equation (6.63) in Bishop's Pattern Recognition and Machine Learning book.

```
class my_gaussian_process():
    def __init__( self, x, y ):
        self.x = x
```

```
        self.y = np.matrix( y )
    def fit( self, theta=[1,1,0,0], beta=0 ):
        self.theta, self.beta = theta, beta
        self.C = self.covariance()
        self.invC = np.linalg.inv( self.C )
    def covariance( self ):
        '''
        x is a np.array, repr N inputs
        theta is an array of four parameters
        '''
        N = size( self.x )
        C = np.zeros(( N, N ))
        for i in range( N ):
            for j in range( N ):
                C[i,j] = self.kernel( x[i], x[j] )
        C += eye(N) * self.beta
        return np.matrix( C )
    def kernel( self, v1, v2 ):
        t0, t1, t2, t3 = self.theta
        # kernel function (6.63) in Bishop Pattern Recognition book
        v1, v2 = np.matrix( v1 ), np.matrix( v2 )
        # the ( v1 - v2 ).T * ( v1 - v2 ) term is || v1 - v2 || ^ 2
        t = ( -t1 / 2.0 ) * ( v1-v2 ).T * ( v1-v2 )
        t = t0 * np.exp( t ) + t2 + t3 * v1.T * v2
        return t[0,0]
    def predict( self, xp ):
        kx = np.matrix([ self.kernel( xi, xp ) for xi in self.x ])
        m = kx * self.invC * self.y.T
        return m[0,0]

n = 10
nx = np.linspace( 0, 2*np.pi, num=100 )
ny = np.sin( nx )
sx = sorted( random.sample( nx, n ) )
sy = np.sin(sx)
x = np.array( sx )
y = np.sin(x) + np.random.normal(0,1,n)*0.25

# <H3>KRIGING / GAUSSIAN PROCESS REGRESSION</H3>

py = list()
gp = my_gaussian_process( x, y )
theta = [1,1,0,0] ; beta = 0.5
gp.fit( theta, beta )
for i in nx:
    py.append( gp.predict( i ) )

plot( nx, ny, 'b', label='Function' )
plot( sx, sy, 'b.', mfc='none' )
plot( x, y, 'b.', label ='Samples' )
plot( nx, py, 'r', label='Regression' )
xlim( 0, 2*np.pi ) ;
ylim( -2, 2 ) ;
title( 'Kriging Regression, beta = 0.5' ) ;
legend() ;
```

## 12.45 Lambda Expressions

Creates a function object and returns it to be called later

```
>>> sqr = lambda x : x**2
>>> sqr(3)
9
```

```
>>> sum = lambda x,y : x+y
>>> sum(2,3)
5
```

## 12.46 `loadmat` and `savemat`

This relates to loading and saving MATLAB `.mat` objects.

```
#!/usr/bin/env python

import scipy.io

# looks up the .mat file from current directory
mat_contents = scipy.io.loadmat( 'filename.mat', struct_as_record=True )

# mat_contents is a dictionary
x = mat_contents['x']
y = mat_contents['y']

# saves new data to a new .mat file
scipy.io.savemat( 'newfilename.mat', { 'var_x':x, 'var_y':y } )
```

## 12.47 Logging

Here is a simple example of using `logging` with different levels.

```
import logging

logging.basicConfig( filename='example.log', level=logging.DEBUG )
logging.debug('This is a low-priority message.')
logging.info('This is a medium-priority message.')
logging.warning('This might be pretty important.')
```

## 12.48 LU Decomposition

This code will grab a matrix from an hd5 file, perform LU decomposition upon it, and then determine the inverse and output it to another hd5 file.

```
#!/usr/bin/env python

import h5py
import numpy as np

def doolittle_h5( a_filename, side_size ):

    import h5py
    import numpy as np

    fa = h5py.File( a_filename, 'r' )
    a = fa['data'][ :side_size, :side_size ]

    fl = h5py.File( 'l.h5', 'w' )
    dl = fl.create_dataset( 'data', ( side_size, side_size ), np.float32 )
    l = fl['data']

    fu = h5py.File( 'u.h5', 'w' )
    du = fu.create_dataset( 'data', ( side_size, side_size ), np.float32 )
```

```
    u = fu['data']

    n = side_size
    for i in range( n ):
        for j in range( i, n ):
            u[i,j] = a[i,j]
            for k in range( i ):
                u[i,j] = u[i,j] - l[i,k] * u[k,j]
        for j in range( i, n ):
            l[j,i] = a[j,i]
            for k in range( i ):
                l[j,i] = l[j,i] - l[j,k] * u[k,i]
            l[j,i] = l[j,i] / float( u[i,i] )

    fa.close()
    fl.close()
    fu.close()

    return None

def solve_L_system_h5( a_filename, side_size, b ):

    import h5py
    import numpy as np

    f = h5py.File( a_filename, 'r' )
    a = f['data'][ :side_size, :side_size ]

    x = np.zeros_like( b ).astype( np.float32 )

    for i in range( b.size ):
        x[i] = b[i]
        for j in range( i ):
            x[i] -= a[i,j] * x[j]

    f.close()

    return x

def solve_U_system_h5( a_filename, side_size, b ):

    import h5py
    import numpy as np

    f = h5py.File( a_filename, 'r' )
    a = f['data'][ :side_size, :side_size ]

    x = np.zeros_like( b ).astype( np.float32 )

    for i in range( b.size-1,-1,-1 ):
        x[i] = b[i]
        for j in range( i+1, b.size ):
            x[i] -= a[i,j] * x[j]
        x[i] /= float( a[i,i] )

    f.close()

    return x

def inverse_h5( a_filename, side_size ):

    import h5py
    import numpy as np

    doolittle_h5( a_filename, side_size )
```

```
    finv = h5py.File( 'ainv.h5', 'w' )
    dinv = finv.create_dataset( 'data', ( side_size, side_size ),\
           np.float32, chunks=(side_size,1) )
    inv = finv['data']

    for i in range( side_size ):
        b = np.zeros((side_size,)) ; b[i] = 1.0
        b.astype( np.float32 )
        y = solve_L_system_h5( 'l.h5', side_size, b )
        x = solve_U_system_h5( 'u.h5', side_size, y )
        inv[:,i] = x

    finv.close()

    return None

a = [ [ 25, 5, 1 ], [ 64, 8, 1 ], [ 144, 12, 1 ] ]
a = np.array( a )
f = h5py.File( 'a.h5', 'w' )
d = f.create_dataset( 'data', (3,3), np.float32 )
d[:3,:3] = a
f.close()
del a

inverse_h5( 'a.h5', 3 )
```

## 12.49 Machine Learning

There are several machine learning modules for Python. There is `mlpy`, `MDP`, the Modular toolkit for Data Processing, and `scikit.learn`.`MDP` thinks of data processing as piping data through filters, whereas `mlpy` and `sklearn` use a more functional approach. I'll cover `sklearn` right now.

### 12.49.1 Loading Data

To load the iris data set we do the following:

```
import sklearn
from sklearn import datasets
iris = sklearn.datasets.load_iris()
# access the data using
iris.data
# access the labels using
iris.target
```

### 12.49.2 Using a Linear SVC

Similar to SVC with parameter `kernel='linear'`, but uses internally liblinear rather than libsvm, so it has more flexibility in the choice of penalties and loss functions and should be faster for huge datasets.

```
estimator = svm.LinearSVC()
estimator.fit( iris.data, iris.target )
estimator.predict( [[ 5.0, 3.6, 1.3, 0.25 ]] ) # returns the predicted label
```

### 12.49.3 Visualizing the Iris Dataset

This is helpful in visualizing the iris dataset

```
setosa, versicolor, virginica = [], [], []
for i in range( 150 ):
    if iris.target[i] == 0:
        setosa.append( iris.data[i] )
    if iris.target[i] == 1:
        versicolor.append( iris.data[i] )
    if iris.target[i] == 2:
        virginica.append( iris.data[i] )

setosa     = np.array( setosa     )
versicolor = np.array( versicolor )
virginica  = np.array( virginica  )

def visualize_iris( col1, col2 ):
    fig = figure()
    ax = fig.add_subplot(111)
    ax.scatter( setosa[:,col1], setosa[:,col2], c='red' )
    ax.scatter( versicolor[:,col1], versicolor[:,col2], c='white' )
    ax.scatter( virginica[:,col1], virginica[:,col2], c='black' )
    name={0:'Sepal Length',1:'Sepal Width',2:'Petal Length',3:'Petal Width'}
    xlabel( name[col1] ) ; ylabel( name[col2] ) ; title('Iris Data Set')
    show()
```

### 12.49.4 Image Segmentation Using K-Means

```
import sklearn
from sklearn import cluster
import scipy as sp
lena = sp.lena()
X = lena.reshape((-1,1))
k_means = sklearn.cluster.KMeans(k=5)
k_means.fit(X)
values = k_means.cluster_centers_.squeeze()
labels = k_means.labels_
lena_compressed = np.choose( labels, values )
lena_compressed.shape = lena.shape
```

### 12.49.5 k-Nearest Neighbors

```
from sklearn import neighbors
knn = neighbors.NeighborsClassifier()
knn.fit( iris.data, iris.target )
knn.predict( [ [ 0.1, 0.2, 0.3, 0.4 ] ] )
```

### 12.49.6 DBSCAN

`DBSCAN` allows you to cluster without knowing the number of clusters ahead of time.

```
import sklearn, sklearn.cluster
from scipy.spatial import distance

d = np.loadtxt( fn, delimiter=',' )
S = d[:,0]
D = distance.squareform(distance.pdist( S[:,None] ))
S = 1 - (D / np.max(D))
x = sklearn.cluster.DBSCAN(eps=min_dist,min_samples=min_no_samples).fit(S)
labels = x.labels_
no_labels = len( set( labels ) )
```

### 12.49.7 Support Vector Machines

We have three options for support vector machines. We can use `linear`, `poly`, or `rbf`. The polynomial support vector machine takes the additional parameter, `degree`.

```
from sklearn import svm
svc = svm.SVC( kernel='linear' )
svc.fit( iris.data, iris.target )
svc.predict( [ [ 0.1, 0.2, 0.3, 0.4 ] ] )
```

```
from sklearn import svm
svc = svm.SVC( kernel='poly', degree=3 )
svc.fit( iris.data, iris.target )
svc.predict( [ [ 0.1, 0.2, 0.3, 0.4 ] ] )
```

```
from sklearn import svm
svc = svm.SVC( kernel='rbf' )
svc.fit( iris.data, iris.target )
svc.predict( [ [ 0.1, 0.2, 0.3, 0.4 ] ] )
```

### 12.49.8 Principal Components Analysis

```
from sklearn import decomposition
pca = decomposition.PCA(n_components=2)
pca.fit( iris.data )
X = pca.transform( iris.data )
scatter( X[:,0], X[:,1], c=iris.target )
```

## 12.50 Mahotas

### 12.50.1 Haralick Features

Takes as input a 2D or 3D image in the form of an ndarray of integer type, and outputs a 4x13 feature vector, one row for each direction.

```
import numpy, mahotas

img = mahotas.imread( 'picture.jpeg' )
feat = mahotas.features.haralick( img )
```

## 12.51 Mayavi

This allows you to do interactive plotting in IPython. First you start IPython with Pylab:

```
$ ipython --pylab
```

Next, import Enthought, Mayavi and mlab as: (This part took the better part of an hour to figure out.)

```
In [1]: from enthought.mayavi import mlab
```

Then you can view an array, $x$, in 3 dimensions as:

```
In [2]: mlab.surf( x, warp_scale='auto' )
```

## 12.52 Modules

If you store a bunch of functions in a module in a program file, you can import that file into another program and use it as a module.

```
import mymodule as mm
x = mm.fct1( x )
```

If your module lives in another directory, then you tell have to tell your program where that module lives by altering the `sys.path` list.

```
import sys
sys.path.append('/home/connor/mymodules/live/here')
import mymodule as mm
x = mm.fct1( x )
```

## 12.53 Munkres

The Munkres algorithm, also known as the Hungarian algorithm, or the Kuhn-Munkres algorithm is used to minimize the cost of assigning the ith worker the jth job when one is presented a table of workers and costs for particular tasks.

You can download the module with the following command:

```
$ sudo easy_install munkres
$
```

Usage is as follows:

```
from munkres import Munkres

m = Munkres()

matrix = [ [5,9,1], [10,3,2], [8,7,4] ]
indexes = m.compute( matrix )
#  [(0, 0), (1, 1), (2, 2)]
```

We can maximize the profit by subtracting each value by a large value:

```
import numpy as np

maxmatrix = np.array( matrix )
maxmatrix = sys.maxint - maxmatrix
indexes = m.compute( matrix )
#  [(0, 1), (1, 0), (2, 2)]
```

Voila. These are the indices in the original matrix that maximize the profit, or whatever.

## 12.54 MySQLdb

`MySQLdb` is the preferred Python module for interacting with databases. We first need to create a database in MySQL.

```
$ mysql -u root -p
> CREATE DATABASE testdb ;
> CREATE USER 'testuser'@'localhost' IDENTIFIED BY 'test143' ;
> USE testdb ;
> GRANT ALL ON testdb.* TO 'testuser'@'localhost' ;
> QUIT ;
$
```

We can then access the database using the following script.

```
#!/usr/bin/env python

import sys
import MySQLdb as mdb

con = None

try:
    #-conection--------server(?)-----user----password--database--
    con = mdb.connect('localhost','testuser','test143','testdb')
    cur = con.cursor()
    cur.execute("SELECT VERSION()")
    data = cur.fetchone()
    print "Database version : {0} ".format( data )
except mdb.Error, e:
    print "Error {0}: {1}".format( e.args[0], e.args[1] )
    sys.exit(1)
finally:
    if con:
        con.close()
```

We can populate our database the hard way:

```
#!/usr/bin/env python

import MySQLdb as mdb
import sys

con = mdb.connect('localhost','testuser','test143','testdb')

with con:
    cur = con.cursor()
    cur.execute("CREATE TABLE IF NOT EXISTS \
        Writers(Id INT PRIMARY KEY AUTO_INCREMENT, Name VARCHAR(25))")
    cur.execute("INSERT INTO Writers(Name) VALUES('Jack London')")
    cur.execute("INSERT INTO Writers(Name) VALUES('Honore de Balzac')")
    cur.execute("INSERT INTO Writers(Name) VALUES('Lion Feuchtwagner')")
    cur.execute("INSERT INTO Writers(Name) VALUES('Emile Zola')")
    cur.execute("INSERT INTO Writers(Name) VALUES('Truman Capote')")
```

Or we can do it the easy way. Here, we assume that `Writers` has already been created, so we delete it with the `DROP TABLE Writers` command.

```
#!/usr/bin/env python

import MySQLdb as mdb
import sys

con = mdb.connect('localhost','testuser','test143','testdb')
writers = ["Jack LONDON",\
           "Honore de BALZAC",\
           "Lion FEUCHTWAGNER",\
           "Emile ZOLA",\
           "Truman CAPOTE"]

with con:
    cur = con.cursor()
    cur.execute("DROP TABLE Writers")
    cur.execute("CREATE TABLE \
        Writers(Id INT PRIMARY KEY AUTO_INCREMENT, Name VARCHAR(25))")
    for w in writers:
        cur.execute("INSERT INTO Writers(Name) VALUES('"+w+"')")
```

There are several cursors. The default cursor returns data in a tuple of tuples. Here, we use `fetchall()` to fetch all of the rows at once.

```
#!/usr/bin/env python

import sys
import MySQLdb as mdb

con = mdb.connect('localhost','testuser','test143','testdb')

with con:
    cur = con.cursor()
    cur.execute("SELECT * FROM Writers")
    rows = cur.fetchall()
    for row in rows:
        print row
```

Here, we use `fetchone()` to fetch rows one at a time.

```
#!/usr/bin/env python

import sys
import MySQLdb as mdb

con = mdb.connect('localhost','testuser','test143','testdb')

with con:
    cur = con.cursor()
    cur.execute("SELECT * FROM Writers")
    N = int( cur.rowcount )
    for i in range( N ):
        row = cur.fetchone()
        print row[0], row[1]
```

The dictionary cursor lets you access data by column names.

```
#!/usr/bin/env python

import MySQLdb as mdb
import sys

con = mdb.connect('localhost','testuser','test143','testdb')

with con:
    cur = con.cursor( mdb.cursors.DictCursor )
    cur.execute("SELECT * FROM Writers")
    rows = cur.fetchall()
    for row in rows:
        print "{0} {1}".format(row['Id'],row['Name'])
```

## 12.55 NumPy

### 12.55.1 ndim, shape, size

For starts, `ndarray` has three useful attributes: ndim, shape and size. Let `x` be an ndarray, then `x.ndim` returns the number of dimensions that `x` has, `x.shape` returns the length of each dimension, and `x.size` returns the number of elements in the array. Thus, suppose `x` is the $3 \times 4 \times 5$ array having the elements $0, \ldots, 59$.

```
>>> x = np.array( [ I for I in range(60) ] )     # create numpy array
>>> x = np.reshape( x, ( 5, 3, 4 ) )    # x = x.reshape( ( 5, 3, 4 ) )
>>>
```

```
>>> x.ndim     # number of dimensions
3
>>> x.shape     # length of each dimension
(5, 3, 4)
>>> x.size     # number of elements in array
60
```

### 12.55.2 Strides

To index or sample arrays at regular intervals, we use

```
x_sampled = x[ begin : end : step ]     # NOT x[ begin : step : end ]
```

### 12.55.3 `hstack` and `vstack`

Used to stack rows and columns. If you're stacking things horizontally, use hstack(). If you're stacking things vertically, use vstack(). These functions accept a tuple.

```
>>> import numpy as np
>>> a = [1,3,5]
>>> b = [2,4,6]
>>>
>>> np.hstack((a,b))
array([1, 3, 5, 2, 4, 6])
>>>
>>> np.vstack((a,b))
array([[1, 3, 5],
       [2, 4, 6]])
>>>
>>> np.vstack((a,b,b,a))
array([[1, 3, 5],
       [2, 4, 6],
       [2, 4, 6],
       [1, 3, 5]])
```

In order to take a 3-dimensional array having dimensions (row,columns,layers) and turn it into a table having (rows*columns,layers) we do the following

```
a = numpy.random.normal( 0.0, 1.0, (3,4,5) )
a_tbl = a[0,:,:].T    # now a_tbl.shape returns (5,4)
for i in range(1,3):
    a_tbl = numpy.hstack( ( a_tbl, a[i,:,:].T ) )
```

We perform the second line because the hstack can only combine arrays when their dimensions agree. After this, the first step in the flattening has already been done, so in the next line we range from 1 to 3, instead of from 0 to 3. We note also that hstack expects a tuple consisting of the two arrays to be catenated.

### 12.55.4 `flupud` and `fliplr`

We can flip arrays up-down or left-right using the following commands:

```
import numpy as np

a = np.array([ [ 1,2 ], [ 3,4 ] ] )

# array([[1, 2],
#        [3, 4]])

fliplr( a )
# array([[2, 1],
```

```
#        [4, 3]])

flipud( a )
# array([[3, 4],
#        [1, 2]])
```

### 12.55.5 histogram

Below, hist is an array of bin frequencies associated with the bin edges in edges. All the intervals for the bin edges are half-open. Thus, for the first set of edges we have [ 1.0, 1.8 ), [ 1.8, 2.6 )...

```
import numpy as np

a = [ i for i in range(1,6) ]
hist, edges = np.histogram( a, bins=5 )
edges
>>> array([ 1. ,  1.8,  2.6,  3.4,  4.2,  5. ])
hist, edges = np.histogram( a, bins=5, range=(0.5,10.5)
edges
>>> array([  0.5,   1.5,   2.5,   3.5,   4.5,   5.5  ])
```

### 12.55.6 interp

This works a little bit differently than the SciPy `interp1d()` function. Here, `N` is the number of data points you want in your newly interpolated data set, `xx`.

```
xx = np.interp( np.linspace( 0, x.size-1, num=N ), range( x.size ), x )
```

### 12.55.7 linspace, logspace, and arange

Returns num evenly spaced numbers over a specified interval.

```
>>> import numpy as np
>>> np.linspace( 0, 10, num=5, endpoint=True )
array([  0. ,   2.5,   5. ,   7.5,  10. ])
>>> np.logspace( 0, 10, num=5, endpoint=True, base=10.0 )
array([  1.00e+00,   3.16e+02,   1.00e+05, 3.16e+07,   1.00e+10])
```

Return evenly spaced numbers over a specified interval.

```
>>> np.arange( 5 )
array([0, 1, 2, 3, 4])
>>> np.arange( 1, 4 )
array([1, 2, 3])
>>> np.arange( 0, 10, 2 )
array([0, 2, 4, 6, 8])
```

### 12.55.8 reshape

Reshape allows us to reshape arrays, but its tricky if sometimes if you don't do it explicitly. Always do a test run on a small array first to make sure it's doing what you think it's doing. Here we collapse a cube.

```
In [1]: a = [ [[ 1, 2, 3],[ 4, 5, 6],[ 7, 8, 9]],\
              [[10,11,12],[13,14,15],[16,17,18]],\
              [[19,20,21],[22,23,24],[25,26,27]] ]

In [2]: a = np.array( a )
```

```
In [3]: a
Out[3]:
array([[[ 1,  2,  3],
        [ 4,  5,  6],
        [ 7,  8,  9]],

       [[10, 11, 12],
        [13, 14, 15],
        [16, 17, 18]],

       [[19, 20, 21],
        [22, 23, 24],
        [25, 26, 27]]])

In [4]: b = np.reshape( a, (9,3) )

In [5]: b
Out[5]:
array([[ 1,  2,  3],
       [ 4,  5,  6],
       [ 7,  8,  9],
       [10, 11, 12],
       [13, 14, 15],
       [16, 17, 18],
       [19, 20, 21],
       [22, 23, 24],
       [25, 26, 27]])

In [6]: b = np.reshape( a, (3,9) )

In [7]: b
Out[7]:
array([[ 1,  2,  3,  4,  5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14, 15, 16, 17, 18],
       [19, 20, 21, 22, 23, 24, 25, 26, 27]])
```

### 12.55.9 `where`

This function can be used for thresholding, or segmentation, or producing binary matrices. The `where` function takes three arguments: `condition`, `x`, and `y`. When `condition` evaluates to `True`, it takes an element from `x`, otherwise it takes an element from `y`.

```
>>> import numpy as np
>>> x = np.random.normal( 0, 1, (3,3) )
>>> x *= 10
>>> x = np.round( x )
>>> x
array([[ 11.,   1., -16.],
       [ 10.,   6.,  -4.],
       [  3.,  -5.,   2.]])
>>> where( x>0, x, 0 )
array([[ 11.,   1.,   0.],
       [ 10.,   6.,   0.],
       [  3.,   0.,   2.]])
>>> where( x>0, x, False )
array([[ 11.,   1.,   0.],
       [ 10.,   6.,   0.],
       [  3.,   0.,   2.]])
>>> where( x>0, x, -1 )
array([[ 11.,   1.,  -1.],
       [ 10.,   6.,  -1.],
       [  3.,  -1.,   2.]])
>>> where( x>0, 1, 0  )
```

```
array([[1, 1, 0],
       [1, 1, 0],
       [1, 0, 1]])
>>> where( x>0, True, False )
array([[ True,  True, False],
       [ True,  True, False],
       [ True, False,  True]], dtype=bool)
```

## 12.56 OpenCV

### 12.56.1 Installation

Go to `http://opencv.willowgarage.com/wiki/` to download the latest `tar.bz2` OpenCV file.

```
$ tar xvjf opencv.tar.bz2
$ cd opencv
$ mkdir release
$ cd release
$ sudo apt-get install cmake
$ cmake -D CMAKE_BUILD_TYPE=RELEASE -D CMAKE_INSTALL_PREFIX=/usr/local -D BUILD_PYTHON_SUPPORT=ON ..
$ make
$ sudo make install
$ sudo ldconfig -v
$ export LD_LIBRARY_PATH=/usr/local/lib:$LD_LIBRARY_PATH
$
```

Here is another method, taken from `http://www.ozbotz.org/opencv-installation/`. First, remove any installed versions of `ffmpeg` and `x264`.

```
$ sudo apt-get remove ffmpeg x264 libx264-dev
```

Next, get all of the dependencies for `ffmpeg` and `x264`.

```
$ sudo apt-get update
$ sudo apt-get install build-essential checkinstall git cmake libfaac-dev libjack-jackd2-dev libmp3lame-dev libop
```

I had to install `libfaac-dev` separately. Next, install `gstreamer`.

```
$ sudo apt-get install libgstreamer0.10-0 libgstreamer0.10-dev gstreamer0.10-tools gstreamer0.10-plugins-base lib
```

Install `gtk`.

```
$ sudo apt-get install libgtk2.0-0 libgtk2.0-dev
```

I had to also install `gtk+`, or something. Next, download and install `libjpeg`.

```
$ sudo apt-get install libjpeg8 libjpeg8-dev
```

Create a directory to hold the source code.

```
$ cd ~
$ mkdir src
```

Download and install x264.

```
$ cd ~/src
$ wget ftp://ftp.videolan.org/pub/videolan/x264/snapshots/x264-snapshot-20120528-2245-stable.tar.bz2
$ tar xvf x264-snapshot-20120528-2245-stable.tar.bz2
$ cd x264-snapshot-20120528-2245-stable
```

For 32-bit architectures:

```
$ ./configure --enable-static
$ make
$ sudo make install
```

For 64-bit architectures, or some ARM architectures:

```
$ ./configure --enable-shared --enable-pic
$ make
$ sudo make install
```

Download and install `ffmpeg`

```
$ cd ~/src
$ wget http://ffmpeg.org/releases/ffmpeg-0.11.tar.bz2
$ tar xvf ffmpeg-0.11.tar.bz2
$ cd ffmpeg-0.11
```

For 32-bit.

```
$ ./configure --enable-gpl --enable-libfaac --enable-libmp3lame --enable-libopencore-amrnb --enable-libopencore-a
$ make
$ sudo make install
```

For 64-bit.

```
$ ./configure --enable-gpl --enable-libfaac --enable-libmp3lame --enable-libopencore-amrnb --enable-libopencore-a
$ make
$ sudo make install
```

Download and install `v4l`, video for linux.

```
$ cd ~/src
$ wget http://www.linuxtv.org/downloads/v4l-utils/v4l-utils-0.8.8.tar.bz2
$ tar xvf v4l-utils-0.8.8.tar.bz2
$ cd v4l-utils-0.8.8
$ make
$ sudo make install
```

Download and install OpenCV

```
$ cd ~/src
$ wget http://downloads.sourceforge.net/project/opencvlibrary/opencv-unix/2.4.1/OpenCV-2.4.1.tar.bz2
$ tar xvf OpenCV-2.4.1.tar.bz2
$ cd OpenCV-2.4.1/
$ mkdir build
$ cd build
$ cmake -D CMAKE_BUILD_TYPE=RELEASE ..
```

Verify that the output includes the following text:

- `found gstreamer-base-0.10`
- `GTK+ 2.x: YES`
- `FFMPEG: YES`
- `GStreamer: YES`
- `V4L/V4L2: Using libv4l`

Build and install OpenCV

```
$ make
$ sudo make install
```

Tell linux where the shared libraries for OpenCV are located by entering the following shell command.

```
$ export LD_LIBRARY_PATH=/usr/local/lib
```

Add the following line to the end of `/etc/ld.so.conf.d/opencv.conf`.

```
/usr/local/lib
```

Enter the following command at the command line:

```
$ sudo ldconfig /etc/ld.so.conf
```

Add the following to the end of `/etc/bash.bashrc`.

```
PKG_CONFIG_PATH=$PKG_CONFIG_PATH:/usr/local/lib/pkgconfig
export PKG_CONFIG_PATH
```

Finally, you should be able to compile code containing OpenCV libraries.

```
$ g++ `pkg-config opencv --cflags` my_code.cpp  -o my_code `pkg-config opencv --libs`
```

### 12.56.2 Computing the Laplacian

```
import cv

im = cv.LoadImageM( 'foo.jpg', 1 )
dst = cv.CreateImage( cv.GetSize(im), cv.IPL_DEPTH_16S, 3 )
laplace = cv.Laplace( im, dst )
cv.SaveImage( 'foo-laplace.png', dst )
```

This computes the Laplacian of the image using the OpenCV thingy for Python.

### 12.56.3 Object Recognition

The rub here is that you need an .xml file for the Haar object detector.

```
import cv
im = cv.LoadImageM( 'image.jpg' )
hc = cv.Load( 'haarcascade.xml' )
det = cv.HaarDetectObjects( im, hc, cv.CreateMemStorage() )
for ( x, y, w, h ), n in det:
    cv.Rectangle( im, ( x, y ), ( x+w, y+h ), 255 )
cv.SaveImage( 'new_image', im )
```

### 12.56.4 Template Matching

These are the methods available for template matching in OpenCV.

- CV_TM_SQDIFF
- CV_TM_SQDIFF_NORMED
- CV_TM_CCORR
- CV_TM_CCORR_NORMED
- CV_TM_CCOEFF
- CV_TM_CCOEFF_NORMED

$$R(x,y) = \sum_{u,v} (T(u,v) - I(x+u, y+v))^2 \tag{12.2}$$

$$R(x,y) = \frac{\sum_{u,v} (T(u,v) - I(x+u, y+v))^2}{\sqrt{\sum_{u,v} T(u,v)^2 \cdot \sum_{u,v} I(x+u, y+v)^2}} \tag{12.3}$$

$$R(x,y) = \sum_{u,v} T(u,v) \cdot I(x+u, y+v) \tag{12.4}$$

$$R(x,y) = \frac{\sum_{u,v} T(u,v) \cdot I(x+u, y+v)}{\sqrt{\sum_{u,v} T(u,v)^2 \cdot \sum_{u,v} I(x+u, y+v)^2}} \tag{12.5}$$

```
im = cv.LoadImage( 'image.png' )
tp = cv.LoadImage( 'template.png' )

W,H = cv.GetSize( im )
w,h = cv.GetSize( tp )

width, height = W - w + 1, H - h + 1

res = cv.CreateImage( ( width, height ), 32, 1 )

# <METHOD> is a metric
cv.MatchTemplate( im, tp, res, <METHOD> )

# this turns the result into a NumPy array
res = np.asarray( res[:,:] )
```

For a more in-depth approach, visit: `http://opencv.itseez.com/doc/tutorials/imgproc/histograms/template_mat`

## 12.57 Opening Files

This is how you open files for reading from the current directory.

```
data = open( 'filename.dat', 'r' )
```

This is a pretty common operation; opening a file of tab or space separated values and turning it into a list or array of floats.

```
import numpy as np

data = open( 'filename.dat', 'r' )
data = data.readlines()
data = [ [ float( item ) for item in line.split() ] for line in data ]
data = np.array( data )
```

Alternatively, we can use the following. It parses lines using whitespace, converts values to floats by default and returns an array:

```
data = numpy.loadtxt( 'filename.dat' )
```

We can extract columns 2, 3, and 5 using:

```
data = numpy.loadtxt( 'filename.dat', usecols=(1,2,4) )
```

And we can unpack these guys seperately using:

```
data = numpy.loadtxt( 'filename.dat', usecols=(1,2,4), unpack='True' )
```

## 12.58 Optimization

This material is adapted slightly from *Programming Collective Intelligence*, by Toby Segaran, which is an excellent and highly accessible resource for machine learning. The following sections assume we have a multivariate function, and cost function that we'd like to maximize or minimize. The cost function maps the input of the multivariate function to a cost, or a quality.

### 12.58.1 Random Optimization

Random optimization isn't a fantastic way to optimize things, but it is simple to implement and it can be useful in comparing optimization results. Random optimization generates inputs randomly, evaluates the cost function, and records the results in a list of cost-input tuples.

In this example, the input is a four element list of the form, `[ int, float, int, float ]`. The domain function generates random inputs

```
def domain( d ):
    '''
    Input:  (d) Four element list of 2-tuples listing the min and max
                for each of the inputs to return
    Output: (r) Four element list of values between the limits given by d
    '''
    r = []
    r.append( random.randint( d[0][0], d[0][1] ) )
    r.append( random.uniform( d[1][0], d[1][1] ) )
    r.append( random.randint( d[2][0], d[2][1] ) )
    r.append( random.uniform( d[3][0], d[3][1] ) )
    return r

def random_optimize( d, n, costf ):
    '''
    Input:    (d)    List of limits as 2-tuples
              (n)    Number of iterations
          (costf)  Cost (or quality) function
    Output: (result) List of tuples ( cost, [ inputs ] ),
                     sorted and reversed so that the first
                     element corresponds to the greatest cost
    '''
    result = []
    for i in range( n ):
        r = domain( d )
        result.append( ( costf( r ), r ) )
    result.sort()
    result.reverse()
    return result
```

### 12.58.2 Simulated Annealing

This function makes the same assumptions as the section above; namely, that the multivariate function assessed by the cost function, `costf()`, accepts four varaibles of the form, `[ int, float, int, float ]`, and that the value of the cost function needs to be maximized.

```
def annealing_optimize( d, costf, T=100.0, cool=0.95, step=1 ):
    # initialize values randomly
    hyp = domain( d )

    while T > 0.1:
        print T
        i = random.randint( 0, len( hyp ) - 1 )
        if i % 2 == 0: # if i is even, then we move by integers
            dir = (-1) ** int( round( random.random() ) )
```

```
        else: # else i is odd, then we move by a float
            dir = random.random() * 0.75 * (-1) ** int( round( random.random() ) )
        dir = step * (-1) ** int( round( random.random() ) )
        alt = hyp[:]
        alt[i] += dir
        if alt[i] < d[i][0]: alt[i] = d[i][0]
        elif alt[i] > d[i][1]: alt[i] = d[i][1]
        h0 = costf( hyp )
        h1 = costf( alt )
        if h1 > h0:
            hyp = alt
        else:
            p = pow( math.e, (-h1-h0)/T )
            if random.random() < p:
                hyp = alt
        T = T * cool

    return hyp
```

## 12.59 Pandas

Import `pandas` using `pip` as:

```
sudo pip install pandas
```

### 12.59.1 Series

`Series` is a one-dimensional labeled array. It is implemented as a subclass of `numpy.ndarray`, so it does pretty much the same stuff, except that it adds automatic data alignment. Here, we look at using a `numpy.ndarray` as input:

```
import pandas as pan
import numpy as np
randn = np.random.randn
s = pan.Series( randn(5), index=['a','b','c','d','e'] )
```

Then, `s` outputs:

```
a    0.470632
b    0.758580
c    0.458117
d    0.477112
e    0.874971
```

You can still operate `Series` very much like `numpy.ndarray`. We still have the familiar slicing and indexing operations:

```
>>> s[0]
0.47063198692279373

>>> s[:3]
a    0.470632
b    0.758580
c    0.458117

>>> s[[4,3,1]]
e    0.874971
d    0.477112
b    0.758580
```

Furthermore, we can still do a lot of other familiar operations:

```
>>> s + s
a    0.941264
b    1.517160
c    0.916235
d    0.954223
e    1.749943

>>> s * 3
a    1.411896
b    2.275740
c    1.374352
d    1.431335
e    2.624914

>>> s[s > s.median()]
b    0.758580
e    0.874971
```

Automatic data alignment means that you can do something like this:

```
>>> s[1:] + s[:-1]
a         NaN
b    1.517160
c    0.916235
d    0.954223
e         NaN
```

We can also name `Series`:

```
>>> s = Series(np.random.randn(5), name='HumbertHumbert' )
0   -1.334
1   -0.171
2    0.050
3   -0.650
4   -1.084
Name: HumbertHumbert

>>> s.name
'HumbertHumbert'
```

A `Series` will accept a Python `dict`, a Numpy `ndarray`, or scalar value. If you use a `dict`, then the `Series` will be indexed by the `dict` keys.

```
>>> d = {'a' : 0., 'b' : 1., 'c' : 2.}
>>> pan.Series( d )
a    0
b    1
c    2
```

You can also reorder the index labels. Any new labels that do not correspond to the `dict` keys will be given a `NaN` value.

```
>>> Series( d, index=['b', 'c', 'd', 'a'] )
b     1
c     2
d   NaN
a     0
```

### 12.59.2 Data Frames

`DataFrame` is the 2-dimensional labeled data structure. You can think of it as a spreadsheet, or an SQL table, or a `dict` of `Series` objects.

There are several ways to populate a Pandas data frame. One way is to pass a NumPy array, and label the columns.

```
>>> df = pan.DataFrame( np.random.random((5,3)), columns=['one','two','three'] )

        one       two     three
0  0.639413  0.535793  0.286887
1  0.689163  0.567031  0.224233
2  0.416483  0.190019  0.520363
3  0.739725  0.860980  0.138962
4  0.503824  0.635687  0.570576
```

We can see which values in a given column are above a threshold:

```
>>> print df.one > 0.5

0     True
1     True
2    False
3     True
4     True
Name: one

>>> print df['three'] > 0.2

0     True
1     True
2     True
3    False
4     True
Name: three
```

Alternatively, we can return a subset of the data satisfying certain conditions

```
>>> print df[ df['one'] > 0.5 ]

        one       two     three
0  0.639413  0.535793  0.286887
1  0.689163  0.567031  0.224233
3  0.739725  0.860980  0.138962
4  0.503824  0.635687  0.570576

>>> print df[ df.three > 0.2 ]

        one       two     three
0  0.639413  0.535793  0.286887
1  0.689163  0.567031  0.224233
2  0.416483  0.190019  0.520363
4  0.503824  0.635687  0.570576

>>> print df[ ( df['one'] > 0.5 ) & ( df.three > 0.2 ) ]

        one       two     three
0  0.639413  0.535793  0.286887
1  0.689163  0.567031  0.224233
4  0.503824  0.635687  0.570576
```

## 12.60 Parallel Computing

### 12.60.1 Thread

The `thread` module provides tools for working with with multiple low level threads. The `threading` module provides a higher-level interface; it is built upon the `thread` module. Using this, we can turn an ordinary

function into a thread:

```
import thread

def thread( stuff ):
    print "I'm a real boy"!
    print stuff

thread.start_new_thread( thread, ('argument') )
```

### 12.60.2 Threading and Queue

The `threading` module is built upon the `thread` module and it provides an easier way to work with threads. Its `Thread` class may be subclassed to create a thread or threads. The `run` method should contain the code you wish to be executed when the thread is executed.

The code below produces 20 threads. When they run they print something and then increment its value.

```
import threading

something = 1

class MyThread( threading.Thread ):
    def run( self ):
        global something
        print something
        something += 1

for i in range( 20 ):
    MyThread().start()
```

Below is an example of two threads communicating via a global variable. The global variable `imu_data` is passed to both classes. `IMU` has a 100 iteration loop, and it sleeps for five seconds between each iteration. If enough time has passed between iterations, it updates the global variable `imu_data`. `RADAR` on the other hand, is in a big hurry; it keeps checking `imu_data`, and printing the result.

```
#!/usr/bin/env python

import time
import threading

imu_data = 0

class IMU( threading.Thread ):
    def run( self ):
        global imu_data
        global tiempo
        for i in range( 100 ):
            time.sleep( 5 )
            imu_data += 1

class RADAR( threading.Thread ):
    def run( self ):
        global imu_data
        for i in range( 500 ):
            time.sleep( 1 )
            print imu_data

IMU().start()
RADAR().start()
```

Since spawning and killing threads costs time, and having a large number of threads can represent a drain on system resources, we can avoid these problems by spawning a set number for threads and feeding them

new tasks until they're all done, at which point we kill them all off. By way of analogy, think of a doctors' office with three doctors and 15 patients. Only 3 patients can be seen at a time, and the untreated patients wait in the waiting room. Here, the doctors represent the thread pool, and the waiting room represent the Queue. Below, we have a program that takes an image, splits it into chips, and computes the Haralick features for each chip using a thread for each chip, and puts all the data into a list.

```
#!/usr/bin/env python

import Queue
import threading
import mahotas
import time
import sys

# this is the list where
# the final data goes
f = list()

# create a Queue
queue = Queue.Queue()

# the code that needs to be executed
# we override the init function in order to pass parameters to the threads
class MyThread( threading.Thread ):
    def __init__( self, f, queue ):
        self.f = f
        self.queue = queue
        threading.Thread.__init__( self )
    def run( self ):
        while True:
            task = self.queue.get()
            self.f.append( mahotas.features.haralick( task ) )
            self.queue.task_done()

# reads the image and converts it to a numpy int ndarray
img = mahotas.imread( sys.argv[1] )

# chip size dimensions
chipx = int( sys.argv[2] )
chipy = int( sys.argv[3] )

# figure out image size
imgx, imgy, imgz = img.shape

# the number of chips in each direction
nbxchips = imgx/chipx
nbychips = imgy/chipy

start = time.time()

# create some threads
for i in range( 3 ):
    t = MyThread( f, queue )
    t.setDaemon( True )
    t.start()

# put chips into the queue
for i in range( nbxchips ):
    for j in range( nbychips ):
        queue.put( img[i*chipx:(i+1)*chipx,j*chipy:(j+1)*chipy,:] )

# wait for all the work to finish
queue.join()
```

```
print 'Elapsed Time: %s' % ( time.time() - start )

print len( f )
```

We can build a for-loop that loops through all of the threads and calls join() on each of them in order to wait for all the threads finish before starting another round of processing.

We can also use two threads to talk to each other over a serial port. We can use one thread to pretend to be an external device, and another thread to lisen over the serial port to the first thread. Below we have two classes, `Sender` and `Reciever`. I used `com0com` to talk over the ports on Windows 7. This listing also shows you how to plot something in three dimensions in real time.

```
import sys
import time
import serial
import threading
import numpy as np
import matplotlib.pyplot as plt
import mpl_toolkits.mplot3d.axes3d as p3

class Sender( threading.Thread ):
    def run( self ):
    port = "\\\\.\\CNCB2"
    ser = serial.Serial( port, 9600 )
    for i in range(25):
        s =  str(np.random.random())+','
        s += str(np.random.random())+','
        s += str(np.random.random())+','
        s += str(i) + '\n'
        x = ser.write( s )
    ser.close()

class Reciever( threading.Thread ):
    def run( self ):
    port = "\\\\.\\CNCA2"
    ser = serial.Serial( port, 9600, timeout=0 )
    fig = plt.figure()
    ax = p3.Axes3D( fig )
    x, y, z = list(), list(), list()
    i = 0 ; colors = []
    while True:
        v = ser.readline()
        if len( v ) > 0:
        t0 = time.time()
            v = v.strip().split(',')
        x.append( float( v[0] ) )
            y.append( float( v[1] ) )
        z.append( float( v[2] ) )
        colors.append( value_to_hex( float( v[3] ), 25 ) )
        ax.scatter( x, y, z, c=colors )
            plt.draw()
        plt.show( block=False )
        i += 1
        time.sleep(0.1)
        if time.time() - t0 > 3:
        break
    plt.show( block=True )
    ser.close()

Reciever().start()
Sender().start()
```

### 12.60.3 Multiprocessing

Python has a Global Interpreter Lock (GIL). This means that only one thread at a time can run python code. There are two exceptions. If your code is waiting for IO, like for you to type something, then it will release the GIL. If NumPy is doing an array operation it will release the GIL.

Threads are lighter than processes. They can be created, destroyed, and swtiched between quicker than processes, but they are limited by the GIL.

Processes are heavier, but they do not block each other the way that threads do because each process has its own GIL. It is also possible to share data, like NumPy arrays, between processes.

Here are some simple examples:

```
import multiprocessing

def f(x):
    return x*x

k = range( 10 )

p = multiprocessing.Pool(processes=2)

p.map( f, k )

# ---> k = [ 0, 1, 4, 9, 16, 25, 36, 49, 64, 81 ]
```

This guy prints things out, which can be collected by a pipe and cat.

```
import time
import multiprocessing

def worker( i ):
    print i, i**2

jobs = []

for i in range( 10 ):
    p = multiprocessing.Process( name=str(i), target=worker, args=(i,) )
    jobs.append( p )
    p.start()

time.sleep( 30 )

for j in jobs:
    j.terminate()

for j in jobs:
    print j.name, j.exitcode
```

This guy uses a Queue.

```
import multiprocessing

q = multiprocessing.Queue()

def worker( i ):
    q.put( [ i, i**2 ] )

for i in range( 10 ):
    p = multiprocessing.Process( name=str(i), target=worker, args=(i,) )
    jobs.append( p )
    p.start()

for i in range( 10 ):
    print q.get
```

## 12.61 PIL

### 12.61.1 ImageDraw

This draws a set of points, a line, or a polygon from a set of point tuples.

```
from PIL import Image
from PIL import ImageDraw
import random as r

p = [ ( r.uniform(0,100), r.uniform(0,100) ) for i in range( 5 ) ]
im = Image.new( 'RGBA', (100,100), (0,0,0,0) )
draw = ImageDraw.Draw( im )
draw.point( p, fill='red' )
# draw.line( p, fill='red' )
# draw.polygon( p, fill='red' )
im.save( 'out.png' )
```

### 12.61.2 PIL.Image to and from NumPy.array

This loads an image, and then converts it to a numerical array.

```
import numpy, PIL.Image

i = PIL.Image.open( 'lena.jpg' )
a = numpy.asarray( i )
j = PIL.Image.fromarray( a )
j.show()
```

## 12.62 Pololu Maestro Server Controler

Go to the product webpage, and download all of the relevant software and drivers under the resources tab.

```
import serial

# this sets up communications
# check device manager for the right COM
# 9600 refers to the baud rate
maestro = serial.Serial('COM4',9600,timeout=1)

# ask for servo 0x00 for the position using 0x90
maestro.write( chr( 0x90 ) + chr( 0x00 ) )

# listen for the 2 byte response
pos = maestro.read( 2 )

# interpret the response
print ( ord( pos[0] ) + ( ord( pos[1] ) << 8 ) ) / 4
```

## 12.63 PNG

First, install PyPNG:

```
$ sudo pip install pypng
$
```

Then we can construct an image using a list of lists, or more conveniently, a NumPy array. When using NumPy arrays, be sure to cast the data as a `np.uint16`. Below, we plan on having a black background, with white points.

| x | y |
|---|---|
| 2.5 | 2.4 |
| 0.5 | 0.7 |
| 2.2 | 2.9 |
| 1.9 | 2.2 |
| 3.1 | 3.0 |
| 2.3 | 2.7 |
| 2.0 | 1.6 |
| 1.0 | 1.1 |
| 1.5 | 1.6 |
| 1.1 | 0.9 |

```
z = np.zeros( (5,5), dtype=np.uint16 )
for p in points:
    z[ p[1], p[0] ] = 1.0
```

Then to write this NumPy array to a png file, we do the following:

```
import png
f = open( 'p.png', 'wb' )
w = png.Writer( width=z.shape[1], height=z.shape[0], greyscale=True, bitdepth=1 )
w.write( f, z )
f.close()
```

## 12.64 Principal Components Analysis

Principal component analysis is appropriate when you have obtained measures on a number of observed variables and wish to develop a smaller number of artificial variables (called principal components) that will account for most of the variance in the observed variables. The principal components may then be used as predictor or criterion variables in subsequent analyses.

Nonetheless, there are some important conceptual differences between principal component analysis and factor analysis that should be understood at the outset. Perhaps the most important deals with the assumption of an underlying causal structure: factor analysis assumes that the covariation in the observed variables is due to the presence of one or more latent variables (factors) that exert causal influence on these observed variables.

Researchers use factor analysis when they believe that certain latent factors exist that exert causal influence on the observed variables they are studying. Exploratory factor analysis helps the researcher identify the number and nature of these latent factors.

In contrast, principal component analysis makes no assumption about an underlying causal model. Principal component analysis is simply a variable reduction procedure that (typically) results in a relatively small number of components that account for most of the variance in a set of observed variables.

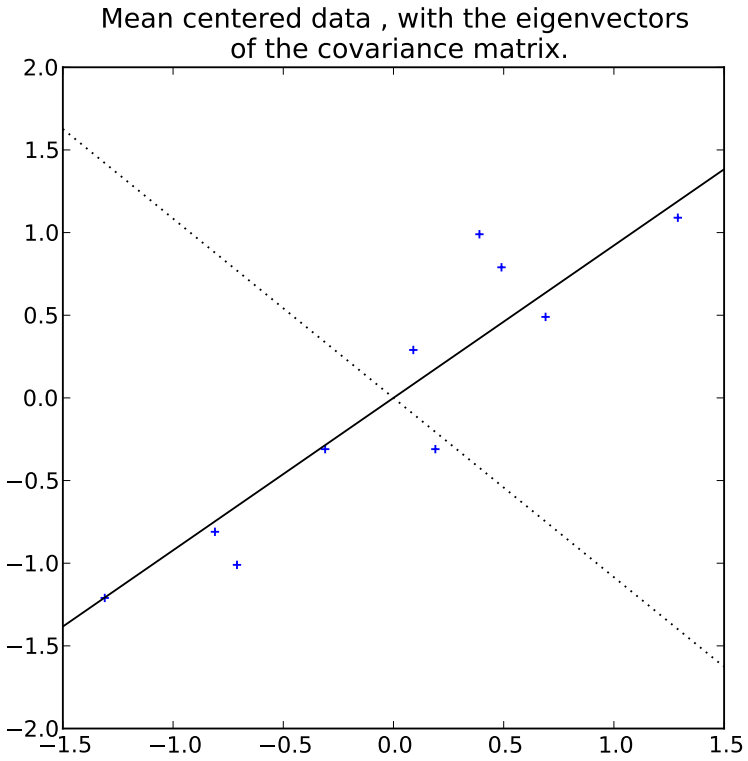
In summary, both factor analysis and principal component analysis have important roles to play in social science research, but their conceptual foundations are quite distinct.

Suppose we have the following values for two variables, x and y.

```
import numpy as np

x = [ 2.5, 0.5, 2.2, 1.9, 3.1, 2.3, 2.0, 1.0, 1.5, 1.1 ]
y = [ 2.4, 0.7, 2.9, 2.2, 3.0, 2.7, 1.6, 1.1, 1.6, 0.9 ]

# mean center the data
# mdata : <2x10>
mx = x - np.mean( x )
```

```
my = y - np.mean( y )
mu = np.mean( [ x, y ] )
mdata = np.vstack( ( mx,my ) )
mdata = np.matrix( mdata )

# calculate the covariance of the mdata
cov = np.cov( mdata )

# calculate the eigenvalues and eigenvectors
val, vec = np.linalg.eig( cov )

# sort the eigenvalues and eigenvectors
srt = np.argsort( val )[::-1]
vec = np.matrix( vec[:,srt] )
val = val[srt]

# val
# array([ 1.28402771,  0.0490834 ])

# vec
# matrix([[-0.6778734 , -0.73517866],
#         [-0.73517866,  0.6778734 ]])
```

Note that the eigenvectors are column vectors. The first eigenvalue corresponds to the first column of the eigenvector matrix. Below we we have plotted the data with the eigenvectors superimposed.




[H]

Here is the code for the image:

```
beg, end, n = -1.5, 1.5, 50
lin = np.linspace( beg, end, num=n )
```

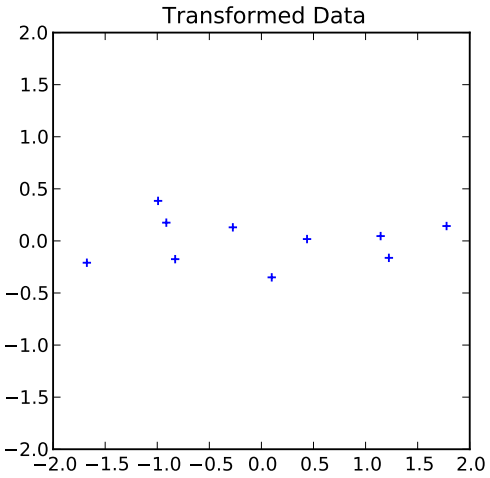
| x | y |
|---|---|
| -0.82797 | -0.17511 |
| 1.77758 | 0.14285 |
| -0.99219 | 0.38437 |
| -0.27421 | 0.13041 |
| -1.67580 | -0.20949 |
| -0.91294 | 0.17528 |
| 0.09910 | -0.34982 |
| 1.14457 | 0.04641 |
| 0.43804 | 0.01776 |
| 1.22382 | -0.16267 |

Figure 12.1: Data after the PCA transformation

```
figure()
title('Mean centered data, with the eigenvectors \n of the covariance matrix.')

scatter( mx, my, marker='+' )
plot( lin, vec[0,0] * lin / vec[1,0], 'k-' )
plot( lin, vec[0,1] * lin / vec[1,1], 'k:' )
xlim(-1.5,1.5)
ylim(-1.5,1.5)
axes().set_aspect('equal')

pylab.show()
```

We see that the eigenvector with the largest eigenvalue fits the data really well. This corresponds to the first principal component. The next vector corresponds to the second principal component. We can then apply the PCA transformation to the data illustrated by the code and figure below. This transforms the data to the axes defined by the orthogonal eigenvectors.

```
td = vec.T * mdata
td = np.array( td )
title('Transformed Data')
scatter( p[0], p[1], marker='+' )
xlim(-2,2)
ylim(-2,2)
axes().set_aspect('equal')
```

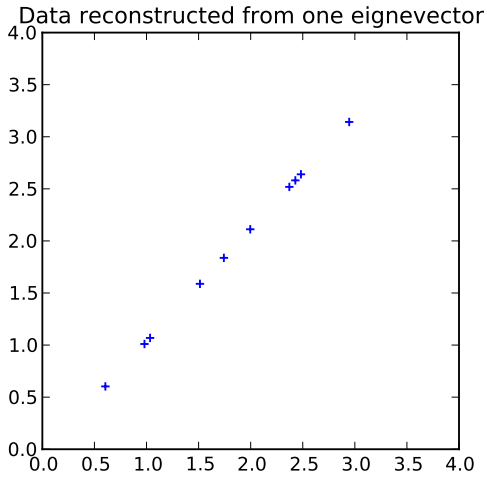



[H]

We compose a feature vector by taking the first few eigenvectors. We obtain the final transformed data by the following matrix multiplication:

```
# form a feature vector, fv, from first eigenvector(s)
# fv : <2x1>
fv = vec[:,0]

# this is the transformation
# td : <1x10>
td = fv.T * mdata

# this is the data reconstructed from the transformation
# rd : <2x10>
rd = ( fv * td )
rd[0] += np.mean( x )
rd[1] += np.mean( y )
rd = np.array( rd )

# and then we plot the results!
scatter( rd[0], rd[1], marker='+' )
xlim(-1,4)
ylim(-1,4)
axes().set_aspect('equal')
```

This is the original data reconstructed using the first principal component. If we had used both prinicpal components, then we'd have gotten the original data back exactly as it had been. Notice that this is like the plot with the eigenvector axes, except that now, the dimension corresponding to the second principal component has been collapsed.




## ALTERNATIVE USING ANOTHER SCRIPT

This is something I found on stack overflow. The `matplotlib.mlab` implementation ran into a memory error on a `<124,819200>` dataset, but this code worked.

```
#!/usr/bin/env python
""" a small class for Principal Component Analysis
Usage:
    p = PCA( A, fraction=0.90 )
In:
    A: an array of e.g. 1000 observations x 20 variables, 1000 rows x 20 columns
    fraction: use principal components that account for e.g.
        90 % of the total variance

Out:
    p.U, p.d, p.Vt: from numpy.linalg.svd, A = U . d . Vt
    p.dinv: 1/d or 0, see NR
    p.eigen: the eigenvalues of A*A, in decreasing order (p.d**2).
        eigen[j] / eigen.sum() is variable j's fraction of the total variance;
        look at the first few eigen[] to see how many PCs get to 90%, 95%...
    p.npc: number of principal components,
        e.g. 2 if the top 2 eigenvalues are >= `fraction` of the total.
        It's ok to change this; methods use the current value.

Methods:
    The methods of class PCA transform vectors or arrays of e.g.
    20 variables, 2 principal components and 1000 observations,
    using partial matrices U' d' Vt', parts of the full U d Vt:
    A ~ U' . d' . Vt' where e.g.
        U' is 1000 x 2
        d' is diag([ d0, d1 ]), the 2 largest singular values
        Vt' is 2 x 20.  Dropping the primes,

    d . Vt      2 principal vars = p.vars_pc( 20 vars )
    U           1000 obs = p.pc_obs( 2 principal vars )
    U . d . Vt  1000 obs, p.obs( 20 vars ) = pc_obs( vars_pc( vars ))
        fast approximate A . vars, using the `npc` principal components

    Ut              2 pcs = p.obs_pc( 1000 obs )
    V . dinv        20 vars = p.pc_vars( 2 principal vars )
    V . dinv . Ut   20 vars, p.vars( 1000 obs ) = pc_vars( obs_pc( obs )),
        fast approximate Ainverse . obs: vars that give ~ those obs.
```

```
Notes:
    PCA does not center or scale A; you usually want to first
        A -= A.mean(A, axis=0)
        A /= A.std(A, axis=0)
    with the little class Center or the like, below.

See also:
    http://en.wikipedia.org/wiki/Principal_component_analysis
    http://en.wikipedia.org/wiki/Singular_value_decomposition
    Press et al., Numerical Recipes (2 or 3 ed), SVD
    PCA micro-tutorial
    iris-pca .py .png

"""

from __future__ import division
import numpy as np
dot = np.dot
    # import bz.numpyutil as nu
    # dot = nu.pdot

__version__ = "2010-04-14 apr"
__author_email__ = "denis-bz-py at t-online dot de"

#...............................................................................
class PCA:
    def __init__( self, A, fraction=0.90 ):
        assert 0 <= fraction <= 1
            # A = U . diag(d) . Vt, O( m n^2 ), lapack_lite --
        self.U, self.d, self.Vt = np.linalg.svd( A, full_matrices=False )
        assert np.all( self.d[:-1] >= self.d[1:] )  # sorted
        self.eigen = self.d**2
        self.sumvariance = np.cumsum(self.eigen)
        self.sumvariance /= self.sumvariance[-1]
        self.npc = np.searchsorted( self.sumvariance, fraction ) + 1
        self.dinv = np.array([ 1/d if d > self.d[0] * 1e-6  else 0
                               for d in self.d ])

    def pc( self ):
        """ e.g. 1000 x 2 U[:, :npc] * d[:npc], to plot etc. """
        n = self.npc
        return self.U[:, :n] * self.d[:n]

    # These 1-line methods may not be worth the bother;
    # then use U d Vt directly --

    def vars_pc( self, x ):
        n = self.npc
        return self.d[:n] * dot( self.Vt[:n], x.T ).T  # 20 vars -> 2 principal

    def pc_vars( self, p ):
        n = self.npc
        return dot( self.Vt[:n].T, (self.dinv[:n] * p).T ) .T  # 2 PC -> 20 vars

    def pc_obs( self, p ):
        n = self.npc
        return dot( self.U[:, :n], p.T )  # 2 principal -> 1000 obs

    def obs_pc( self, obs ):
        n = self.npc
        return dot( self.U[:, :n].T, obs ) .T  # 1000 obs -> 2 principal

    def obs( self, x ):
```

```
        return self.pc_obs( self.vars_pc(x) )  # 20 vars -> 2 principal -> 1000 obs

    def vars( self, obs ):
        return self.pc_vars( self.obs_pc(obs) )  # 1000 obs -> 2 principal -> 20 vars


class Center:
    """ A -= A.mean() /= A.std(), inplace -- use A.copy() if need be
        uncenter(x) == original A . x
    """
        # mttiw
    def __init__( self, A, axis=0, scale=True, verbose=1 ):
        self.mean = A.mean(axis=axis)
        if verbose:
            print "Center -= A.mean:", self.mean
        A -= self.mean
        if scale:
            std = A.std(axis=axis)
            self.std = np.where( std, std, 1. )
            if verbose:
                print "Center /= A.std:", self.std
            A /= self.std
        else:
            self.std = np.ones( A.shape[-1] )
        self.A = A

    def uncenter( self, x ):
        return np.dot( self.A, x * self.std ) + np.dot( x, self.mean )
```

We can use this in the following manner:

```
tx = [ 2.5, 0.5, 2.2, 1.9, 3.1, 2.3, 2.0, 1.0, 1.5, 1.1 ]
ty = [ 2.4, 0.7, 2.9, 2.2, 3.0, 2.7, 1.6, 1.1, 1.6, 0.9 ]
tx = ( tx - np.mean( tx ) ) / np.std( tx )
ty = ( ty - np.mean( ty ) ) / np.std( ty )
t = np.vstack( ( tx, ty ) )
t
array([[ 0.92627881, -1.7585873 ,  0.52354889,  0.12081898,  1.73173864,
         0.6577922 ,  0.25506228, -1.08737078, -0.41615425, -0.95312747],
       [ 0.61016865, -1.506743  ,  1.23278973,  0.36112022,  1.35731394,
         0.9837413 , -0.38602507, -1.00864614, -0.38602507, -1.25769457]])
pt = PCA( t, fraction=1.0 )

# pt.U is the set of eigenvectors, trans above
pt.U
array([[-0.70710678, -0.70710678],
       [-0.70710678,  0.70710678]])

# pt.Vt is the principal components, pcomp above
pt.Vt
array([[-0.24756118,  0.52612865, -0.2829913 , -0.07765279, -0.49772578,
        -0.26449324,  0.02110147,  0.33772221,  0.12925171,  0.35622026],
       [-0.25971686,  0.20691588,  0.58271397,  0.19743208, -0.30762823,
         0.26780056, -0.5267189 ,  0.06468035,  0.02475421, -0.25023306]])

# pt.d is the list of singular values
pt.d
array([ 4.38854107,  0.86064352])

# sie eigenvalues, jah?
pt.eigen
array([ 19.25929273,   0.74070727])

# this is the fractional variance of the first component
pt.eigen[0] / pt.eigen.sum()
```

```
0.96296463634612284

# this is the fractional variance of the next component
pt.eigen[1] / pt.eigen.sum()
0.03703536365387719
```

FINAL ANALYSIS

So this is the best that I could come up with. It is in agreement with the PCA functionality found in R and other implementations up to sign discrepancy.

```
variables, observations = X.shape

for i in range( variables ):
    X[i,:] -= np.mean( X[i,:] )

U, d, Vt = np.linalg.svd( X )
U, d, Vt = np.matrix( U ), np.matrix( d ), np.matrix( Vt )

rd = np.array( U[:,:2].T * X )

scatter( rd[0], -rd[1] ) ; title('Reduced Data') ;
```

This was the equivalent code using RPy2:

```
import rpy2.robjects ; r = rpy2.robjects.r
r('p=princomp(anscombe[,5:8],cor=FALSE)') ;
scores = np.array( r('p$scores') ) ; #$
scatter( scores[:,0], scores[:,1] ) ; title( 'rpy2' ) ;
```

## 12.65 PyCUDA

```
>>> import pycuda
>>> import pycuda.driver as cuda
>>> import pycuda.autoinit, pycuda.compiler
>>> import numpy
>>>
>>> a = numpy.random.randn(4,4).astype(numpy.float32)
>>> a_gpu = cuda.mem_alloc(a.nbytes)
>>> cuda.memcpy_htod(a_gpu,a)
>>>
>>> mod = pycuda.compiler.SourceModule("""
...     __global__ void twice( float *a )
...     {
...       int idx = threadIdx.x + threadIdx.y*4;
...       a[idx] *= 2;
...     }
...     """)
>>>
>>> func = mod.get_function("twice")
>>> func(a_gpu,block=(4,4,1))
>>>
>>> a_doubled = numpy.empty_like(a)
>>> cuda.memcpy_dtoh(a_doubled,a_gpu)
>>>
>>> print a_doubled
[[ 5.7332058  -1.07265019  1.62579823  2.33956313]
 [-0.79299438  0.43180776  4.59874725 -0.16678417]
 [ 2.48301172 -3.7892437  -1.13474083 -2.72417903]
 [-0.76352853  2.99678111 -1.69026327 -0.33870423]]
```

```
>>>
>>> print a
[[ 2.8666029  -0.5363251   0.81289911  1.16978157]
 [-0.39649719  0.21590388  2.29937363 -0.08339208]
 [ 1.24150586 -1.89462185 -0.56737041 -1.36208951]
 [-0.38176426  1.49839056 -0.84513164 -0.16935211]]
>>>
```

We can exploit all of numpy using the pycuda.cumath.numpy module:

```
import numpy
import pycuda.autoinit                # initializes GPU
import pycuda.gpuarray as gpuarray    # creates gpuarray object
import pycuda.cumath                  # provides mathematical functions

# create some data on the CPU
x = numpy.random.random((32,)).astype( float32 )

# send data to the GPU
xg = gpuarray.to_gpu( x )

# perform exp() on GPU
xg_exp = pycuda.cumath.exp( xg )

# send data back to CPU
x_exp = xg_exp.get()
```

### 12.65.1 Kronecker Product

Here is the kernel and some code for computing the Kronecker product, but this can be extended easily to two dimensional image correlation. Note: this guy expects two square matrices.

```
#!/usr/bin/env python

import pycuda.compiler as comp
import pycuda.driver as drv
import numpy
import pycuda.autoinit

mod = comp.SourceModule("""
__global__ void kron( float *a, float *b, float *c )
{
  int bix = blockIdx.x ;
  int biy = blockIdx.y ;
  int tix = threadIdx.x ;
  int tiy = threadIdx.y ;
  int bdx = blockDim.x ;
  int bdy = blockDim.y ;
  c[ bdx*bdy*(bix*gdy+biy)+tix*bdy+tiy] = a[tix*bdy+tiy] * b[bix*gdy+biy] ;
}
""")

kron = mod.get_function("kron")

a = numpy.array([[0,1],[2,3]]).astype(numpy.float32)
b = numpy.array([[4,5],[6,7]]).astype(numpy.float32)

dest = numpy.zeros((4,4)).astype(numpy.float32)

kron( drv.In(a), drv.In(b), drv.Out(dest), block=(2,2,1), grid=(2,2) )

print dest
```

The most important thing to notice here is that the input and output matrices must be of the same numeric type. Here, they are `float32`. Also, in the next to last line, `block` and `grid` expect a *3-tuple* and a *2-tuple*, respectively. Whammy, son.

### 12.65.2 Hyperspectral Image Correlation

This is actually only part of hyperspectral image correlation, but it's the tough part. All that remains is loading the mean and standard deviation data and calculating the correlation, but this is the hard part of comparing each spectral vector of one image cube against each spectral vector of the other image cube.

The neat part is that by coding the number of bands explicitly as `dimz = 100` we can use more blocks and threads, up to 22 each.

The part that is commented out allows us to compare results, but it takes longer because it is Python code computed serially on the CPU, not CUDA code computed in parallel on the GPGPU.

```
#!/usr/bin/env python

import pycuda.compiler as comp
import pycuda.driver as drv
import numpy
import pycuda.autoinit

dimx = 22
dimy = 22
dimz = 100

mod = comp.SourceModule("""
__global__ void corr( float *x, float *y, float *c )
{
  int bands = 100 ;

  int bix = blockIdx.x ;
  int biy = blockIdx.y ;
  int gdx = gridDim.x ;
  int gdy = gridDim.y ;

  int tix = threadIdx.x ;
  int tiy = threadIdx.y ;
  int bdx = blockDim.x ;
  int bdy = blockDim.y ;

  float ep = 0 ;

  for( int k = 0 ; k < bands ; k++ )
  {
    ep += x[bands*(bix*gdy+biy)+k]*y[bands*(tix*bdy+tiy)+k];
  }

  c[bdx*bdy*(bix*gdy+biy)+tix*bdy+tiy] = ep / (float)bands ;
}
""")


corr = mod.get_function("corr")

x = numpy.random.uniform(0,1,(dimx,dimy,dimz)).astype(numpy.float32)
y = numpy.random.uniform(0,1,(dimx,dimy,dimz)).astype(numpy.float32)

c = numpy.zeros((dimx*dimy,dimx*dimy)).astype(numpy.float32)
#d = numpy.zeros((dimx*dimy,dimx*dimy)).astype(numpy.float32)

corr( drv.In(x), drv.In(y), drv.Out(c), block=(dimx,dimy,1), grid=(dimx,dimy) )
```

```
'''
for xi in range( dimx ):
    for xj in range( dimy ):
        for yi in range( dimx ):
            for yj in range( dimy ):
                ep = 0
                for k in range( dimz ):
                    ep += x[ xi, xj, k ] * y [ yi, yj, k ]
                d[ xi*dimy + xj, yi*dimy + yj ] = ep /float(dimz)

err = 0
for i in range( dimx*dimy ):
    for j in range( dimx*dimy ):
        err += ( d[i,j]-c[i,j] )**2.0
rms = numpy.sqrt( err / float(dimx*dimy*dimx*dimy) )
print rms
'''
```

## 12.66 Pylab

### 12.66.1 Bar Charts

Suppose we have an mxn array, with m variables, and n observations. We'd like a barchart with a set of m bars for each of the n observations. We'd like for these sets of bars to stand apart from each other also.

```
import numpy as np
import pylab

v1 = [2,3,5,7,11,13]
v2 = [ i*1.25 for i in v1 ]
v3 = [ i*1.50 for i in v1 ]
x = np.vstack((v1,v2,v3))

rows, cols = x.shape
spc = 0.75
w = spc/rows

for i in range( rows ):
    pylab.bar( left=np.arange(cols)+i*w, height=x[i,:], width=w)
```

### 12.66.2 Histograms

If we just want a diagram we can use:

```
import numpy as np
import pylab

mu, sigma = 0.0, 1.0
v = np.random.normal( mu, sigma, 1000 )
pylab.hist( v, bins=50 )
pylab.show()
```

### 12.66.3 Regression

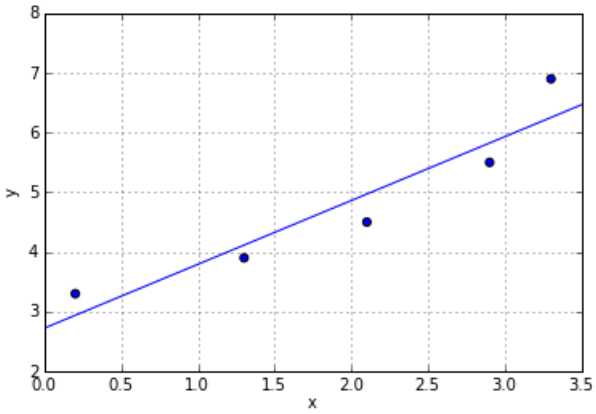
Suppose we have two data sets:

```
x = [ 0.2, 1.3, 2.1, 2.9, 3.3 ]
y = [ 3.3, 3.9, 4.5, 5.5, 6.9 ]
scatter( x, y ) ;
```

```
xlabel( 'x' ) ;
ylabel( 'y' ) ;
```

We can perform linear regression using the following:

```
import pylab
import numpy as np
m1, b = pylab.polyfit( x, y, 1 )
nx = np.linspace( 0, 3.5, num=100 )
yp = pylab.polyval( [m1,b], nx )
plot( nx, yp ) ;
scatter( x, y ) ;
grid( True ) ;
xlabel( 'x' ) ;
ylabel( 'y' ) ;
savefig( 'pylab_linear_regression.png', fmt='png' )
```




Note the correlation between the varaible `y` and `yp`:

```
np.corrcoef( y, polyval( [m1,b], x ) )
```

We can perform quadratic regression in a similar manner:

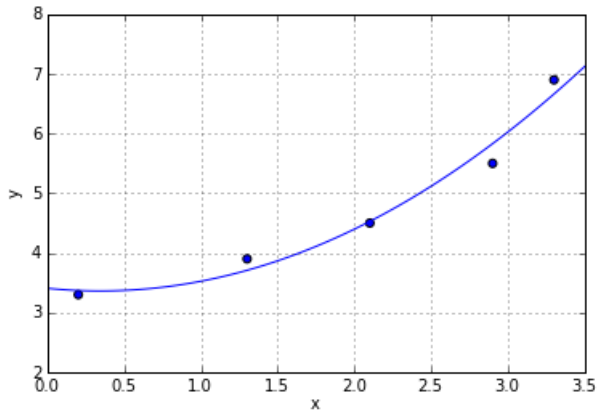
```
m2, m1, b = polyfit( x, y, 2 )
nx = np.linspace( 0, 3.5, num=100 )
yp = polyval( [m2,m1,b], nx )
plot( nx, yp ) ;
scatter( x, y ) ;
grid( True ) ;
yticks( arange( 2, 9 ), arange( 2, 9 ) ) ;
xlabel( 'x' ) ;
ylabel( 'y' ) ;
```

```
np.corrcoef( y, polyval( [m2,m1,b], x ) )
```

We note that the correlation between `y` and the linear regression was 0.94097326 and that the correlation between `y` and the quadratic regression was 0.97338905, so the quadratic regression was a little bit more accurate.

### 12.66.4 Adjusting Borders

Suppose you have labels for the xticks that are really long, and the standard parameters are cutting them off. The following code takes labels from a list, tilts them, and adds space at the bottom of the figure




```
import matplotlib.pyplot as plt
from matplotlib.figure import SubplotParams

# establishes an aspect ratio of 1.5
# and then scales it up by 1.5
w,h = 1.5 * plt.figaspect( 1.5 )

# define figure
fig = plt.figure( figsize=(w,h) )
ax = fig.add_subplot( 111 )

# add some space to the bottom
fig.subplots_adjust( bottom=0.25 )

# these are the labels for the x-axis
labels = [ 'Tom', 'Dick', 'Harry' ]

ax.plot( ***plot some stuff here*** )

# put labels up, and rotate a little bit
plt.xticks( np.arange( len( labels ) ), tuple( labels ), rotation=75 )

# save figure, fn is something like 'filename.png'
plt.savefig( fn, format='png' )

# show your work
plt.show()
```

### 12.66.5 RGB Images

We can build RGB images using NumPy arrays and view them using the Pylab (imshow()) function. Note that by default, `imshow()` expects values in the range [0,1], not [0,255].

```
# black backround
x = np.zeros((m,n,3))

# white background
x = np.ones((m,n,3))

# red object on white background
# rows [a,b) and columns [c,d)
x[a:b,c:d,1:3] = 0
```

```
imshow( x )
```

The basic colors are as follows:

| Color | Red | Green | Blue | Hexadecimal |
|---|---|---|---|---|
| Black | 0 | 0 | 0 | #000000 |
| White | 255 | 255 | 255 | #FFFFFF |
| Red | 255 | 0 | 0 | #FF0000 |
| Green | 0 | 192 | 0 | #00C000 |
| Blue | 0 | 0 | 255 | #0000FF |
| Yellow | 255 | 255 | 0 | #FFFF00 |

### 12.66.6 RGBA Images

These are just like the RGB images discussed above, except there is an extra layer (the $\alpha$-channel) describing the opacity of a pixel. The extra layer is "beneath" the top three color layers. A value of $\alpha = 1.0$ describes a fully opaque pixel, $\alpha = 0.5$ half-opaque, $\alpha = 0.0$ fully transparent.

```
# white background
x = np.ones((m,n,4))

# red object on white background
# rows [a,b) and columns [c,d)
# alpha layer is set to 1.0 since
# we started with an array of ones
x[a:b,c:d,1:3] = 0

# fourth layer, half-opacity
x[a:b,c:d,3] = 0.5

# fourth layer, full transparency
x[a:b,c:d,3] = 0.0

imshow( x, interpolation='nearest' )
```

### 12.66.7 Overlaying/Highlighting with Transparencies

We can use <mxnx4> RGBA images on top of greyscale <mxn> reflectance images to highlight regions precisely. The code below produces a 2 dimensional reflectance image, and then highlights regions using colored transparent RGBA pixels, and then saves the result so that this process can be done in an automated fashion.

```
import numpy
import matplotlib
import matplotlib.pyplot as plt
x, y = numpy.ones((5,5,4)), numpy.ones((5,5,4))
x[:,:,3], y[:,:,3] = 0.0, 0.0 # now fully transparent
x[0,0,:] = (1,0,0,.25) # red square, top left corner, 0.25 opacity
y[1,1,:] = (0,0.75,0,.5) # green square at (1,1), 0.50 opacity
z = numpy.random.random((5,5)) # a random field
fig = plt.figure()
ax = fig.add_subplot(111)
ax.imshow( z, cmap='gray' ) # produce a <mxn> greyscale image
ax.imshow( x, interpolation='nearest' ) # place red square on top
ax.imshow( y, interpolation='nearest' ) # place green square
fig.savefig( 'picture.png', format='png' )
```

### 12.66.8 Plotting

If we start IPython with the pylab option:

```
$ ipython -pylab
$
```

Then we have some nice plotting functions very similar to those found in MATLAB.

```
    acorr     - plot the autocorrelation function
    bar       - make a bar chart
    barh      - a horizontal bar chart
    broken_barh - a set of horizontal bars with gaps
    boxplot   - make a box and whisker plot
    cohere    - make a plot of coherence
    contour   - make a contour plot
    contourf  - make a filled contour plot
    csd       - make a plot of cross spectral density
    fill      - make filled polygons
    hist      - make a histogram
    loglog    - a log log plot
    matshow   - display a matrix in a new figure preserving aspect
    pcolor    - make a pseudocolor plot
    pcolormesh- make a pseudocolor plot using a quadrilateral mesh
    pie       - make a pie chart
    plot      - make a line plot
    plotfile  - plot column data from an ASCII tab/space/comma delimited file
    polar    - make a polar plot on a PolarAxes
    psd      - make a plot of power spectral density
               use this like periodogram from matlab
    quiver   - make a direction field (arrows) plot
    scatter  - make a scatter plot
    semilogx - log x axis
    semilogy - log y axis
    specgram - a spectrogram plot
    stem     - make a stem plot
    xcorr    - plot the autocorrelation function of x and y

    annotate  - annotate something in the figure
    arrow     - add an arrow to the axes
    axes      - create a new axes
    axhline   - draw a horizontal line across axes
    axvline   - draw a vertical line across axes
    axhspan   - draw a horizontal bar across axes
    axvspan   - draw a vertical bar across axes
    axis      - Set or return the current axis limits
    box       - set the axes frame on/off state
    cla       - clear current axes
    clabel    - label a contour plot
    clf       - clear a figure window
    clim      - adjust the color limits of the current image
    close     - close a figure window
    colorbar  - add a colorbar to the current figure
    delaxes   - delete an axes from the current figure
    draw      - Force a redraw of the current figure
    errorbar  - make an errorbar graph
    figlegend - make legend on the figure rather than the axes
    figimage  - make a figure image
    figtext   - add text in figure coords
    figure   - create or change active figure
    findobj  - recursively find all objects matching some criteria
    gca      - return the current axes
    gcf      - return the current figure
    gci      - get the current image, or None
    getp     - get a graphics property
```

```
    grid     - set whether gridding is on
    hold     - set the axes hold state
    ioff     - turn interaction mode off
    ion      - turn interaction mode on
    isinteractive - return True if interaction mode is on
    imread   - load image file into array
    imsave   - save array as an image file
    imshow   - plot image data
    ishold   - return the hold state of the current axes
    legend   - make an axes legend
    plot_date - plot dates
    rc       - control the default params
    rgrids   - customize the radial grids and labels for polar
    savefig  - save the current figure
    setp      - set a graphics property
    show     - show the figures
    spy      - plot sparsity pattern using markers or image
    subplot  - make a subplot (numrows, numcols, axesnum)
    subplots_adjust - change the params controlling the subplot
                positions of current figure
    subplot_tool - launch the subplot configuration tool
    suptitle   - add a figure title
    table    - add a table to the plot
    text     - add some text at location x,y to the current axes
    thetagrids - customize the radial theta grids and labels for polar
    title    - add a title to the current axes
    xlim     - set/get the xlimits
    ylim     - set/get the ylimits
    xticks   - set/get the xticks
    yticks   - set/get the yticks
    xlabel   - add an xlabel to the current axes
    ylabel   - add a ylabel to the current axes

    autumn   - set the default colormap to autumn
    bone     - set the default colormap to bone
    cool     - set the default colormap to cool
    copper   - set the default colormap to copper
    flag     - set the default colormap to flag
    gray     - set the default colormap to gray
    hot      - set the default colormap to hot
    hsv      - set the default colormap to hsv
    jet      - set the default colormap to jet
    pink     - set the default colormap to pink
    prism    - set the default colormap to prism
    spring   - set the default colormap to spring
    summer   - set the default colormap to summer
    winter   - set the default colormap to winter
    spectral - set the default colormap to spectral
```

### 12.66.9 Value to Hex

The following code can be used to map a number in the range [0,1] to the hex value representing a color in the jet colormap.

```
def rgbscale( x, oldmin, oldmax, newmin, newmax ):
    oldscale = float( oldmax - oldmin )
    newscale = float( newmax - newmin )
    ratio = float( newscale / oldscale )
    s = x
    s -= oldmin
    s *= ratio
    s += newmin
    return s
```

```
def interp_blue( x ):
    if x >= 0.0 and x < 0.11:
        c = hex( int( rgbscale( x, 0.0, 0.11, 127, 255 ) ) )
    if x >= 0.11 and x < 0.34:
        c = hex( int( rgbscale( x, 0.11, 0.34, 255, 255 ) ) )
    if x >= 0.34 and x < 0.65:
        c = hex( int( rgbscale( x, 0.34, 0.65, 255, 0.0 ) ) )
    if x >= 0.65 and x <= 1.0:
        c = hex( int( rgbscale( x, 0.65, 1.0, 0.0, 0.0 ) ) )
    c = str(c[2:])
    if len( c ) == 1: c = '0'+c
    return c

def interp_green( x ):
    if x >= 0.0 and x < 0.125:
        c = hex( int( rgbscale( x, 0.0, 0.125, 0.0, 0.0 ) ) )
    if x >= 0.125 and x < 0.375:
        c = hex( int( rgbscale( x, 0.125, 0.375, 0.0, 255 ) ) )
    if x >= 0.375 and x < 0.64:
        c = hex( int( rgbscale( x, 0.375, 0.64, 255, 255 ) ) )
    if x >= 0.64 and x < 0.91:
        c = hex( int( rgbscale( x, 0.64, 0.91, 255, 0.0 ) ) )
    if x >= 0.91 and x <= 1.0:
        c = hex( int( rgbscale( x, 0.91, 1.0, 0.0, 0.0 ) ) )
    c = str(c[2:])
    if len( c ) == 1: c = '0'+c
    return c

def interp_red( x ):
    if x >= 0.0 and x < 0.35:
        c = hex( int( rgbscale( x, 0.0, 0.35, 0.0, 0.0 ) ) )
    if x >= 0.35 and x < 0.66:
        c = hex( int( rgbscale( x, 0.35, 0.66, 0.0, 255 ) ) )
    if x >= 0.66 and x < 0.89:
        c = hex( int( rgbscale( x, 0.66, 0.89, 255, 255 ) ) )
    if x >= 0.89 and x <= 1.0:
        c = hex( int( rgbscale( x, 0.89, 1.0, 255, 127 ) ) )
    c = str(c[2:])
    if len( c ) == 1: c = '0'+c
    return c

def value_to_hex( x, scale ):
    # takes value in range [0,1] returns hex value corresponding to a jet value
    n = x / float( scale )
    return '#'+interp_red(n)+interp_green(n)+interp_blue(n)
```

## 12.67 PyQt4

### 12.67.1 Linux Installation

First, update your repository, and upgrade some packages:

```
$ sudo apt-get update
$ sudo apt-get upgrade python-qt4
$ sudo apt-get upgrade pyqt4-dev-tools
$ sudo apt-get upgrade qt4-designer
```

Okay. Here's what needs to happen. The `sip` module needs to be imported from `/usr/lib/pymodules/python2.6/sip.so` not `/usr/lib/python2.6/dist-packages/sip.so`. In order to ensure that that happens, use the following:

```
import sys
```

```
sys.path.insert(0, "/usr/lib/pymodules/python2.6")
```

And then we can import our PyQt4 modules.

```
from PyQt4.QtCore import *
from PyQt4 import QtGui
from PyQt4 import *
```

The following examples are mostly from `zetcode.com`.

### 12.67.2 A Simple Example

Every PyQt4 application must create an application object. The `sys.argv` parameter is a list of arguments from the command line. The `QtGui.QWidget` is the base class for all of the user interface object.

This example sets the size of the window and the position of it with respect to the screen. This is done in the imperative style. Later we will see the `setGeometry()` function that will set size and position in one line.

```
#!/usr/bin/env python

import sys
from PyQt4 import QtGui


def main():
    app = QtGui.QApplication( sys.argv )
    w = QtGui.QWidget()
    # width x height
    w.resize( 250, 150 )
    # across x down from top-left corner
    w.move( 300, 300 )
    # window title
    w.setWindowTitle('Simple')
    w.show()
    # Here, we have the event-loop
    sys.exit(app.exec_())

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

### 12.67.3 Adding Tool Tips, and an Icon

This is an example of using an object oriented implementation. This code gives a tool tip for the window, and another for an individual button.

```
#!/usr/bin/env python

import sys
from PyQt4 import QtGui

class Example( QtGui.QWidget ):
    def __init__( self ):
    super( Example, self ).__init__()
    self.initUI()
    def initUI( self ):
    QtGui.QToolTip.setFont( QtGui.QFont('SansSerif',10))
    self.setToolTip('This is a <b>QWidget</b> widget')
    btn = QtGui.QPushButton('Button',self)
    btn.setToolTip('This is a <b>QPushButton</b> widget')
    btn.resize( btn.sizeHint() )
    btn.move(50,50)
```

```
        self.setGeometry(300,300,250,150)
    self.setWindowTitle('Tooltips')
    self.setWindowIcon( QtGui.QIcon('web.png') )
    self.show()

def main():
    app= QtGui.QApplication(sys.argv)
    ex = Example()
    sys.exit(app.exec_())

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

### 12.67.4 Adding a Close Button

This adds a close button in the window.

```
#!/usr/bin/env python

import sys
from PyQt4 import QtGui, QtCore

class Example( QtGui.QWidget ):
    def __init__( self ):
        super( Example, self ).__init__()
    self.initUI()
    def initUI( self ):
        qbtn = QtGui.QPushButton('Quit',self)
    qbtn.clicked.connect( QtCore.QCoreApplication.instance().quit )
    qbtn.resize(qbtn.sizeHint())
    qbtn.move(50,50)
    self.setGeometry(300,300,250,150)
    self.setWindowTitle('Quit Button')
    self.show()

def main():
    app = QtGui.QApplication( sys.argv )
    ex = Example()
    sys.exit( app.exec_() )

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

### 12.67.5 Verification Pop-Up

Upon closure, this provides a pop-up box that verifies that we do, in fact, want to close the application.

```
#!/usr/bin/env python

import sys
from PyQt4 import QtGui

class Example( QtGui.QWidget ):
    def __init__( self ):
        super( Example, self ).__init__()
    self.initUI()
    def initUI( self ):
    self.setGeometry(300,300,250,150)
    self.setWindowTitle('Message Box')
    self.show()
    def closeEvent( self, event ):
    reply = QtGui.QMessageBox.question( self, 'Message',\
```

```
            "Are you sure you want to quit?", \
            QtGui.QMessageBox.Yes | QtGui.QMessageBox.No, \
            QtGui.QMessageBox.No )
    if reply == QtGui.QMessageBox.Yes:
        event.accept()
    else:
        event.ignore()

def main():
    app = QtGui.QApplication( sys.argv )
    ex = Example()
    sys.exit( app.exec_() )

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

### 12.67.6 Status Bar

This produces a status bar at the bottom of the page.

```
#!/usr/bin/env python

import sys
from PyQt4 import QtGui

class Example( QtGui.QMainWindow ):
    def __init__( self ):
    super( Example, self ).__init__()
    self.initUI()
    def initUI( self ):
    self.statusBar().showMessage('Ready')
    self.setGeometry(300,300,250,150)
    self.setWindowTitle('Statusbar')
    self.show()

def main():
    app = QtGui.QApplication( sys.argv )
    ex = Example()
    sys.exit( app.exec_() )

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

### 12.67.7 Menu Bar and Status Bar

```
#!/usr/bin/env python

import sys
from PyQt4 import QtGui

class Example( QtGui.QMainWindow ):
    def __init__( self ):
    super( Example, self ).__init__()
    self.initUI()
    def initUI( self ):
    exitAction = QtGui.QAction( QtGui.QIcon('exit.png'),'&Exit',self)
        exitAction.setShortcut('Ctrl+Q')
    exitAction.setStatusTip('Exit application.')
    exitAction.triggered.connect( QtGui.qApp.quit )
    self.statusBar()
    menubar = self.menuBar()
```

```
    fileMenu = menubar.addMenu('&File')
    fileMenu.addAction( exitAction )
    self.setGeometry( 300, 300, 250, 150 )
    self.setWindowTitle( 'Menubar' )
    self.show()

def main():
    app = QtGui.QApplication( sys.argv )
    ex = Example()
    sys.exit( app.exec_() )

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

### 12.67.8 Menu, Tool and Status Bars, and Text Area

```
#!/usr/bin/env
# this produces a status bar, a tool bar, and a menu bar, AND a text area

import sys
from PyQt4 import QtGui

class Example( QtGui.QMainWindow ):

    def __init__( self ):

    super( Example, self ).__init__()
    self.initUI()

    def initUI( self ):

    textEdit = QtGui.QTextEdit()
    self.setCentralWidget( textEdit )

        exitAction = QtGui.QAction( QtGui.QIcon( 'exit24.png' ), 'Exit', self )
    exitAction.setShortcut( 'Ctrl+Q' )
    exitAction.setStatusTip( 'Exit application' )
    exitAction.triggered.connect( self.close )

    self.statusBar()

    menubar = self.menuBar()
    filemenu = menubar.addMenu('&File')
    filemenu.addAction( exitAction )

    toolbar = self.addToolBar('Exit')
    toolbar.addAction(exitAction)

    self.setGeometry( 300, 300, 350, 250 )
    self.setWindowTitle( 'Main window' )
    self.show()

def main():

    app = QtGui.QApplication( sys.argv )
    ex = Example()
    sys.exit( app.exec_() )

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

### 12.67.9 Box Layout

```
#!/usr/bin/env python

import sys
from PyQt4 import QtGui

class Example( QtGui.QWidget ):

    def __init__( self ):
        super( Example, self ).__init__()
        self.initUI()

    def initUI( self ):

    okButton = QtGui.QPushButton('OK')
    cancelButton = QtGui.QPushButton('Cancel')

        hbox = QtGui.QHBoxLayout()
    hbox.addStretch(1)
    hbox.addWidget(okButton)
    hbox.addWidget(cancelButton)

    vbox = QtGui.QVBoxLayout()
    vbox.addStretch(1)
    vbox.addLayout(hbox)

    self.setLayout( vbox )

    self.setGeometry(300,300,300,150)
    self.setWindowTitle('Buttons')
    self.show()

def main():

    app = QtGui.QApplication( sys.argv )
    ex = Example()
    sys.exit( app.exec_() )

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

### 12.67.10 Grid Layout

```
#!/usr/bin/env python
# grid layout

import sys
from PyQt4 import QtGui

class Example( QtGui.QWidget ):
    def __init__( self ):
    super( Example, self ).__init__()
    self.initUI()
    def initUI( self ):
    names = ['Cls','Bck','','Close','7','8','9','/',\
            '4','5','6','*','1','2','3','-',\
            '0','.','=','+']
    grid = QtGui.QGridLayout()
    pos = list()
    for i in range(5):
        for j in range(4):
```

```
                pos.append( (i,j) )
    j = 0
    for i in names:
        button = QtGui.QPushButton(i)
        if j == 2:
        grid.addWidget( QtGui.QLabel(''), 0, 2 )
        else:
        grid.addWidget( button, pos[j][0], pos[j][1] )
        j += 1
    self.setLayout( grid )
    self.setWindowTitle('Calculator')
    self.show()

def main():
    app = QtGui.QApplication( sys.argv )
    ex = Example()
    sys.exit( app.exec_())

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

### 12.67.11 Text Fields and a Text Box

```
#!/usr/bin/env python

import sys
from PyQt4 import QtGui

class Example( QtGui.QWidget ):

    def __init__( self ):

        super( Example, self ).__init__()
    self.initUI()

    def initUI( self ):

    title = QtGui.QLabel('Title')
    author = QtGui.QLabel('Author')
    review = QtGui.QLabel('Review')

    titleEdit = QtGui.QLineEdit()
    authorEdit = QtGui.QLineEdit()
    reviewEdit = QtGui.QTextEdit()

    grid = QtGui.QGridLayout()
    grid.setSpacing(10)

    grid.addWidget( title, 1, 0 )
    grid.addWidget( titleEdit, 1, 1 )

    grid.addWidget( author, 2, 0 )
    grid.addWidget( authorEdit, 2, 1 )

    grid.addWidget( review, 3, 0 )
    grid.addWidget( reviewEdit, 3, 1, 5, 1 )

        self.setLayout( grid )

    self.setGeometry( 300, 300, 350, 300 )
    self.setWindowTitle( 'Review' )
    self.show()
```

```
def main():
    app = QtGui.QApplication( sys.argv )
    ex = Example()
    sys.exit( app.exec_() )

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

### 12.67.12 Signals and Slots

In this example, we have an LCD display that responds to a slider. The slider sends signals to the LCD, and the LCD listens for those signals.

```
#!/usr/bin/env python

import sys
from PyQt4 import QtGui, QtCore

class Example( QtGui.QWidget ):

    def __init__( self ):
    super( Example, self ).__init__()
    self.initUI()

    def initUI( self ):
    lcd = QtGui.QLCDNumber( self )
    sld = QtGui.QSlider( QtCore.Qt.Horizontal, self )

    vbox = QtGui.QVBoxLayout()
    vbox.addWidget(lcd)
    vbox.addWidget(sld)

    self.setLayout( vbox )
    # here, the sider sends changed signals to the lcd
    sld.valueChanged.connect( lcd.display )

    self.setGeometry( 300, 300, 250, 150 )
    self.setWindowTitle( 'Signal and Slot' )
    self.show()

def main():
    app = QtGui.QApplication( sys.argv )
    ex = Example()
    sys.exit( app.exec_() )

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

### 12.67.13 Key Press Event

This example captures a key press event.

```
#!/usr/bin/env python
# reimplementing event handlers

import sys
from PyQt4 import QtGui, QtCore

class Example( QtGui.QWidget ):

    def __init__( self ):
        super( Example, self ).__init__()
```

```
    self.initUI()

    def initUI( self ):
        self.setGeometry( 300, 300, 250, 150 )
    self.setWindowTitle( 'Event handler' )
    self.show()

    def keyPressEvent( self, e ):
    if e.key() == QtCore.Qt.Key_Escape:
        self.close()

def main():
    app = QtGui.QApplication( sys.argv )
    ex = Example()
    sys.exit( app.exec_() )

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

### 12.67.14 Mouse Press Event

If we click on a button, then a `clicked()` signal is generated.

```
#!/usr/bin/env python

import sys
from PyQt4 import QtGui, QtCore

class Communicate( QtCore.QObject ):
    closeApp = QtCore.pyqtSignal()

class Example( QtGui.QMainWindow ):

    def __init__( self ):
    super( Example, self ).__init__()
    self.initUI()

    def initUI( self ):
    self.c = Communicate()
    self.c.closeApp.connect( self.close )
        self.setGeometry( 300, 300, 300, 150 )
    self.setWindowTitle( 'Emit signal' )
    self.show()

    def mousePressEvent( self, event ):
        self.c.closeApp.emit()

def main():
    app = QtGui.QApplication( sys.argv )
    ex = Example()
    sys.exit( app.exec_() )

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

### 12.67.15 Multiple Events Connected to the Same Slot

```
#!/usr/bin/env python
# sometimes it's useful to know with event sent the signal
# both buttons are connected to the same slot
```

```
import sys
from PyQt4 import QtGui, QtCore

class Example( QtGui.QMainWindow ):

    def __init__( self ):
    super( Example, self ).__init__()
    self.initUI()

    def initUI( self ):
        btn1 = QtGui.QPushButton("Button 1", self)
    btn1.move(30,50)

    btn2 = QtGui.QPushButton("Button 2", self)
    btn2.move(150,50)

    btn1.clicked.connect(self.buttonClicked)
    btn2.clicked.connect(self.buttonClicked)

    self.statusBar()

    self.setGeometry( 300, 300, 300, 150 )
    self.setWindowTitle('Event sender')
    self.show()

    def buttonClicked( self ):
    sender = self.sender()
    self.statusBar().showMessage( sender.text()+' was pressed' )

def main():
    app = QtGui.QApplication( sys.argv )
    ex = Example()
    sys.exit( app.exec_() )

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

## 12.68 Random

```
import random

random.random() # random float from [0,1)
random.uniform(0,10) # random float from [0,10)
```

## 12.69 Raw Input

This function allows you to take a break and get user input.

```
In [1]: x = raw_input("say something relevant: ")
say something relevant: baise-moi

In [2]: x
Out[2]: 'baise-moi'
```

## 12.70 Regular Expressions

The `re` module allows us to use regular expression in python. We'll show some examples of the more useful functions.

First, you can compile a regex and then test strings out on it as shown:

```
>>> import re
>>> p = re.compile(r'(a(b)c)d')
>>> m = p.match('abcd')
>>> m.group(0)
'abcd'
>>> m.group(1)
'abc'
>>> m.group(2)
'b'
>>> m.groups()
('abcd','abc','b')
```

We can also search for matches and replace them with other strings.

```
>>> p = re.compile(r'(red|white|blue)')
>>> p.sub('color','blue socks and red shoes')
'color socks and color shoes'
```

We can alternatively use the regex in the `re.sub()` function.

```
>>> re.sub(r'(red|white|blue)','color','blue socks and red shoes')
'color socks and color shoes'
```

## 12.71 Rescaling

This rescales data to a new maximum and minimum.

```
def rescale( data, newmin, newmax ):
    oldmin, oldmax = np.min( data ), np.max( data )
    newscale = float( newmax - newmin )
    oldscale = float( oldmax - oldmin )
    if oldscale == 0: return None
    ratio = float( newscale / oldscale )
    scaled = data.copy()
    scaled -= oldmin
    scaled *= ratio
    scaled += newmin
    return scaled
```

This is fast because it doesn't have any loops. Also, it doesn't matter what the dimensionality of `data` is. The basic operation is this:

$$\text{newvalue} = (\text{oldvalue} - \text{oldmin}) \times \frac{\text{newmax} - \text{newmin}}{\text{oldmax} - \text{oldmin}} + \text{newmin} \tag{12.6}$$

## 12.72 RPy

RPy allows you to use R functions from Python. To install, use the following:

```
$ sudo easy_install rpy
$
```

We can execute R code in two different ways using the r object as a function that takes R code as input, or when accessing functions or variables, by pretending that they are attributes of the r object.

```
from rpy import r

# r function that takes a string input
# puts data from data.dat in 'd' variable on the r-side
r( "d <- read.table( 'data.dat' ") )

# r object that returns the data in data.dat
r.d

# loads fdim library
r( "library('fdim'") )

# evaluates d, and returns a dict()
result = r( "fdim( d ") )
```

## 12.73 SciPy

### 12.73.1 Critical Values for Statistical Tests

We can look up critical values for a given alpha value, and degrees of freedom for the t distribution as:

```
>>> from scipy.stats import t
>>> t.isf( alpha = 0.025, df = 9 )
2.2621571627409915
```

### 12.73.2 Helpful Functions

SciPy has a number of helpful functions. In addition to the help() function of the interactive interpreter, SciPy offers info() and source(). To use these, you need to import the whole SciPy module.

```
In [1]: from scipy import *

In [2]: info( sqrt ) # returns input/output, examples and notes

In [3]: source( sqrt ) # returns source code
```

### 12.73.3 Interpolation

THis works in two steps: first `splrep()`, for *spline representation*, and then `splev()` for *spline evaluation*.

```
from scipy import interpolate

x = np.arange( 0, 2 * np.pi, 10 )
y = np.sin( x )

tck = interpolate.splrep( x, y, s=0 )
xnew = np.arange( 0, 2 * np.pi, 50 )
ynew = interpolate.splev( xnew, tck, der=0 )
```

Here is another method:

```
f = scipy.interpolate.interp1d( range( x.shape[0] ), x[:,i] )
xnew = numpy.linspace( 0, x.shape[0]-1, num=2*rows )
z[:,i] = f( xnew )
```

### 12.73.4 Testing Normality with Shapiro-Wilk Test

So we need to test a dataset to see if it is normal. We will determine an alpha value, say 0.05 or 0.01, and we will compare the p-value of the Shapiro-Wilk Test with our preset alpha value.

If the p-value is greater than the alpha value then the data set is normal. If the p-value is less than the alpha value then the data set is *not* normal.

```
import numpy
import scipy.stats

n = numpy.random.normal( 0.0, 1.0, 100 )
w, p = scipy.stats.shapiro( n )
```

### 12.73.5 Linear Regression

The `scipy.stats` module contains the function linregress that computes the least squares regression for two sets of measurements. It takes two arrays as input and returns the slope, intercept, correlation coefficient ($r$-value), two-sided $p$-value for a hypothesis test whose null hypothesis that the slope is zero, and the standard error.

```
import numpy as np
import scipy.stats

x,y = np.random.random((5,)), np.random.random((5,))
slope, intercept, r_value, p_value, std_err = scipy.stats.linregress(x,y)
```

### 12.73.6 `ndimage`

The `scipy.ndimage` module offers several functions for image processing. Below we have an example of binary dilation and erosion.

```
>>> x
array([[ 0.,  0.,  0.,  0.,  0.],
       [ 0.,  0.,  1.,  0.,  0.],
       [ 0.,  1.,  1.,  1.,  0.],
       [ 0.,  0.,  1.,  0.,  0.],
       [ 0.,  0.,  0.,  0.,  0.]])
>>> s.binary_dilation( x )
array([[False, False,  True, False, False],
       [False,  True,  True,  True, False],
       [ True,  True,  True,  True,  True],
       [False,  True,  True,  True, False],
       [False, False,  True, False, False]], dtype=bool)
>>> s.binary_erosion( x )
array([[False, False, False, False, False],
       [False, False, False, False, False],
       [False, False,  True, False, False],
       [False, False, False, False, False],
       [False, False, False, False, False]], dtype=bool)
```

The `scipy.ndimage` module also offers opening and closing. Opening removes salt noise, and closing removes pepper noise. Opening and closing are built from dilation and erosion as follows,

$$\texttt{Opening(x)} \leftarrow \texttt{Dilation(Opening(x))} \tag{12.7}$$

$$\texttt{Closing(x)} \leftarrow \texttt{Opening(Closing(x))} \tag{12.8}$$

### 12.73.7 Wiener Filtering

We can perform Wiener filtering by using,

```
wx = scipy.signal.wiener( x )
```

## 12.74 Shell Commands

```
import os

os.system( 'cmd' )
```

You can execute commands on the command line from your program, or the interpreter, using the system function from the os module. Remember that you can run something in the background by ending the command with '&'. Do this when you want the new process you just spawned to work in the background;

## 12.75 Shelve

Shelve is a very powerful Python module for object persistence. When you shelve an object, you must give a key by which the object value is known. In this way, the shelve file becomes a database of stored values, any of which can be accessed at any time.

```
import shelve

db = shelve.open( 'file' )
db['key'] = value

print db.keys()
print db.values()

db.close()
```

## 12.76 Shuffling

We can use the following code to shuffle a NumPy array:

```
import random
import numpy as np

def shuffle_array( x ):
    rows, cols = x.shape
    r = range( rows ); random.shuffle( r )
    c = range( cols ); random.shuffle( c )
    s = np.zeros_like( x )
    for i in range( rows ):
        s[i,:] = x[ r[i], : ]
    for i in range( cols ):
        s[:,i] = x[ :, c[i] ]
    return s
```

Note that the method, random.shuffle, shuffles the array in place and does not return anything.

## 12.77 SIFT

Since SIFT is proprietary, the only parameter we can change is the size of the `GridAdaptiveFeatureDetector()`, as it is not technically a part of the SIFT algorithm.

```
import cv, cv2
# rewuires dtype=np.uint8
b = cv2.imread('/Users/connor/Dropbox/images/cjohnson318.jpg')
b = cv2.cvtColor(b,cv2.COLOR_BGR2GRAY)
img = cv2.cvtColor( b, cv2.COLOR_GRAY2BGR )
sift = cv2.FeatureDetector_create('SIFT')
detector = cv2.GridAdaptedFeatureDetector( sift, 10 )
fs = detector.detect(b)
matshow( b, cmap='gray' )
for f in fs:
    x, y = f.pt[0], f.pt[1]
    scatter( x, y, marker='+' )
    print x, y, f.size
show();
```

## 12.78 Smoothing

The next two listings are 3 and 5 point unweighted moving averages that can be used to smooth a signal. The third listing is a weighted 5 point moving average.

The third listing is a function for smoothing a rectangular array. It takes as input the rectangular array, and one one of the three linear smoothing functions listed below. It will calculate an array, $r$, that is the original array with smoothed rows, and an array, $c$, that is the original array with smoothed columns. The output is the average of $r$ and $c$.

```
def ma3( x ):
    '''five point moving average function'''
    rows = x.size
    s = np.zeros( rows )
    s[0] = (  x[0]  + x[1]  ) / 2.0
    for i in range(1,rows-1):
        s[i] = ( x[i-1] + x[i] + x[i+1] ) / 3.0
    s[-1] = ( x[-2] + x[-1] ) / 2.0
    return s
```

```
def ma5( x ):
    '''five point moving average function'''
    rows = x.size
    s = np.zeros( rows )
    s[0] = ( x[0] + x[1] ) / 2.0
    s[1] = ( x[0] + x[1] + x[2] ) / 3.0
    for i in range(2,rows-2):
        s[i] = (x[i-2] + x[i-1] + x[i] + x[i+1] + x[i+2] )/5.0
    s[-2] = ( x[-3] + x[-2] + x[-1] ) / 3.0
    s[-1] = ( x[-2] + x[-1] ) / 2.0
    return s
```

```
def wma5( x ):
    '''five point weighted moving average function'''
    rows = x.size
    s = np.zeros( rows )
    s[0] = ( x[0] + x[1] ) / 2.0
    s[1] = ( x[0] + 2*x[1] + x[2] ) / 4.0
    for i in range(2,rows-2):
        s[i] = (x[i-2] + 2*x[i-1] + 3*x[i] + 2*x[i+1] + x[i+2] )/9.0
```

```
    s[-2] = ( x[-3] + 2*x[-2] + x[-1] ) / 4.0
    s[-1] = ( x[-2] + x[-1] ) / 2.0
    return s
```

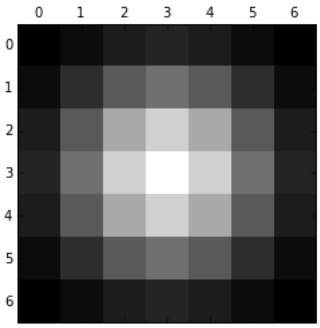
```
def smoothsurf( x, f ):
    r = np.zeros_like( x )
    c = np.zeros_like( x )
    s = np.zeros_like( x )
    rows, cols = x.shape
    for i in range( rows ):
        r[i,:] = f( x[i,:] )
    for j in range( cols ):
        c[:,j] = f( x[:,j] )
    for i in range( rows ):
        for j in range( cols ):
            s[i,j] = ( r[i,j] + c[i,j] )/2.0
    return s
```

Here is an example of building a Gaussian smoother. We'll use the `mgrid` function, which is unfamiliar.

```
grows, gcols = np.mgrid[ -3:4, -3:4 ]
kernel = grows**2.0/5.0 + gcols**2.0/5.0
kernel = np.exp( -kernel )
```

We can also write this as a lambda expression:

```
gk = lambda k : exp( -np.sum( np.mgrid[ -k:k+1, -k:k+1 ]**2.0/( k+2 ), axis=0 ) )
```




Then to smooth an image, you convolve it with one of these kernels using `np.convolve`.

## 12.79 Sorting

Lists may be sorted in place using the sort() function.

```
>>> a = [5,3,4]
>>> b = a.sort()
>>> a
[3, 4, 5]
>>> b
[3, 4, 5]
```

We may alternatively sort lists using the sorted() function. This returns a sorted version, rather than sorting the list in place.

```
>>> a = [5,3,4]
>>> b = sorted( a )
>>> a
[5, 3, 4]
>>> b
[3, 4, 5]
```

## 12.80 SURF

The parameter `nOctaves` controls the size of the features. A larger number finds larger features. The parameter `thresh` controls the number of features found. A smaller number finds more features.

```
import cv, cv2
# requires dtype=np.uint8
b = cv2.imread('/Users/connor/Dropbox/images/cjohnson318.jpg')
b = cv2.cvtColor(b,cv2.COLOR_BGR2GRAY)
img = cv2.cvtColor( b, cv2.COLOR_GRAY2BGR )
noctaves=1 ; thresh=1
surf = cv2.SURF(thresh,noctaves)
fs = surf.detect( b, None )
matshow(b,cmap='gray')
for f in fs:
    x, y = f.pt[0], f.pt[1]
    scatter( x, y, marker='+' )
    print f.pt[0], f.pt[1], f.size
show();
```

## 12.81 Three Dimensional Plotting

This will do a three dimensional scatterplot:

```
#!/usr/bin/env python

from mpl_toolkits.mplot3d import Axes3D
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
import pylab

fig = pylab.figure()
ax = Axes3D( fig )
X = np.random.random((5,))
Y = np.random.random((5,))
Z = np.random.random((5,))
surf = ax.plot( X, Y, Z, 'b.', linewidth=1, antialiased=True )
fig.canvas.set_window_title( "Distance" )
pylab.show()
```

## 12.82 Tkinter

### 12.82.1 Canvas

I've gotten this material from effbot.org.tkinterbook/canvas.htm

```
from Tkinter import *
```

```
master = Tk()

w = Canvas( master, width=200, height=200 )
w.pack()

x = w.create_line(0,0,100,100)
y = w.create_line(0,100,100,0)

w.delete(x)
w.delete(y)

mainloop()
```

If we want to add an image for a background that is, say $793 \times 1024$, then we'd use the following code:

```
from Tkinter import *

master = Tk()

w = Canvas( master, width=1024, height=793 )
w.pack()

photo = PhotoImage( file='eastwood.pgm' )
w.create_image( (1024/2,793/2), image=photo )

# and now we can draw lines on the image!

mainloop()
```

## 12.83 Timing Operations

```
#!/usr/bin/env python

import time

# make a list for the timestamps
t = list()

# append new timestamps
t.append( time.localtime() )

# convenient function h:m:s
def tee():
    t = time.localtime()
    print t[3],':',t[4],':',t[5]

# otherwise
t0 = time.time()
#stuff
print time.time() - t0
```

## 12.84 Twitter

So, this is a start..

```
import twitter
client = twitter.Api()
latest_posts = client.GetUserTimeline("vexedmuse")
for s in latest_posts:
    print s.text
```

And this outputs:

```
@cjohnson318 welcome to the socially misguided year of 2013
@nplusonemag come back to Brooklyn.
 @Gawker: George Zimmerman's brother is tweeting up a storm about "black teens" http://t.co/sK3crA8qz7  @MsWillia
@payalparikh I think the long lost brother has terrible acting skills from the last preview I watched, but yes, w
@payalparikh shhh! I'm behind two episodes!!
@genoFbaby you can--it's called hangout in #google+.
@SVheartndesign good luck, girl!!
 @NYMag: Do blondes  look a little bit porno  in red? Naomi Watts thinks so. http://t.co/4TEmzhUQtA  @CillaJenkin
MT @brooklyn_news Bk population boom about Bed-Stuy and babies: Babies, hipsters, immigrants oh my http://t.co/BP
Protest outside my apartment over circus slated to perform this week at #barclays in #brooklyn #breaking #news #a
Tofu never tasted better--Excited to try: @My_Recipes http://t.co/9og40kJnJm
Byron's reward for coming when he was called--enjoy the surprise yawn http://t.co/2EFR5aWtck
@brooklyn_news it's a shame their clothing is both uninspiring and poorly made.
So done RT @runnersworld: Want to run tomorrow morning? Set your clothes out (and your coffeemaker up) NOW. http:
@TheAtlantic did you read today's @Slate "Dear Prudie" and thought the article made for a good Monday theme? Haha
@PaperMonument @sculpturecenter no Facebook (I know)--is there a description of this event elsewhere?
@ridlej add me on #lastfm: fatallypoetic
These bills are an appalling abuse of power: AP News - Bills seek end to farm animal abuse videos #animalcruelty
@FitCollective see. You're getting the hang of #hashtags
Weezy!!! Hang in there!!! @LilTunechi
```

## 12.85 Visualizing Sequential Data

Pylab offers some great plotting options right out of the box.

```
from pylab import *

data = random((20,1))

# linearly interpolated plot
plot( data )
\begin{lstlisting}


\section{ Visualizing Scatterplots}

\begin{lstlisting}
from pylab import *

# scatterplot with white diamonds
plot( random((20,1)), random((20,1)), 'wD' )
scatter( random((20,1)), random((20,1)), c='w', marker'd' )
```

## 12.86 Visualizing Matrix Data

```
from pylab import *

data = random((5,5))

# greyscale plot with no colorbar on the side
matshow( data, cmap='gray' )

# a matlab-looking plot with a colorbar
matshow( data, cmap='jet' ) ; colorbar()

# you can control limits with vmin, vmax
matshow( data, cmap='gray', vmin=0.0, vmax=1.0 ) ; colorbar()
```

## 12.87 Visualizing Image Data

```
from pylab import *

# image from a picture
img = imread( 'image.jpg' )
imshow( img )

# image from a matrix
imshow( data, cmap='gray', interpolation='nearest' )

# you can control limits with vmin, vmax
imshow( data, cmap='gray', vmin=0.0, vmax=1.0 ) ; colorbar()
```

## 12.88 Voronoi Diagrams

The following code produces a Voronoi diagram as shown in Fig. 12.2. Each segment contains the points closest to the points highlighted in black.

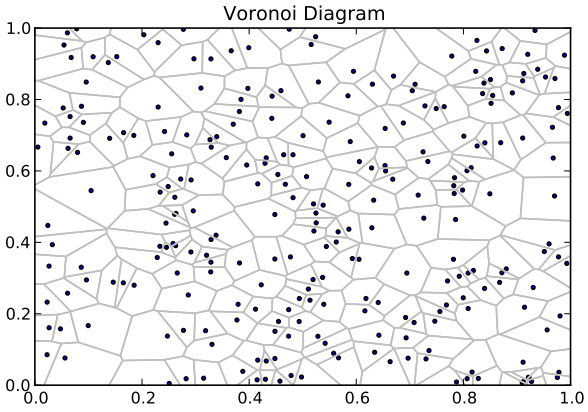



Figure 12.2: A Voronoi diagram.

```
#!/usr/bin/env python
# -----------------------------------------------------------------------------
# Voronoi diagram from a list of points
# Copyright (C) 2011  Nicolas P. Rougier
#
# Distributed under the terms of the BSD License.
# -----------------------------------------------------------------------------
import numpy as np
import matplotlib
import matplotlib.pyplot as plt


def circumcircle(P1, P2, P3):
    '''
    Return center of the circle containing P1, P2 and P3

    If P1, P2 and P3 are colinear, return None

    Adapted from:
```

```
    http://local.wasp.uwa.edu.au/~pbourke/geometry/circlefrom3/Circle.cpp
    '''
    delta_a = P2 - P1
    delta_b = P3 - P2
    if np.abs(delta_a[0]) <= 0.000000001 and np.abs(delta_b[1]) <= 0.000000001:
        center_x = 0.5*(P2[0] + P3[0])
        center_y = 0.5*(P1[1] + P2[1])
    else:
        aSlope = delta_a[1]/delta_a[0]
        bSlope = delta_b[1]/delta_b[0]
        if np.abs(aSlope-bSlope) <= 0.000000001:
            return None
        center_x= (aSlope*bSlope*(P1[1] - P3[1]) + bSlope*(P1[0] + P2 [0])\
                    - aSlope*(P2[0]+P3[0]))/(2.*(bSlope-aSlope))
        center_y = -(center_x - (P1[0]+P2[0])/2.)/aSlope + (P1[1]+P2[1])/2.
    return center_x, center_y

def voronoi(X,Y):
    ''' Return line segments describing the voronoi diagram of X and Y '''
    P = np.zeros((X.size+4,2))
    P[:X.size,0], P[:Y.size,1] = X, Y
    # We add four points at (pseudo) "infinity"
    m = max(np.abs(X).max(), np.abs(Y).max())*1e5
    P[X.size:,0] = -m, -m, +m, +m
    P[Y.size:,1] = -m, +m, -m, +m
    D = matplotlib.tri.Triangulation(P[:,0],P[:,1])
    T = D.triangles
    n = T.shape[0]
    C = np.zeros((n,2))
    for i in range(n):
        C[i] = circumcircle(P[T[i,0]],P[T[i,1]],P[T[i,2]])
    X,Y = C[:,0], C[:,1]
    segments = []
    for i in range(n):
        for k in D.neighbors[i]:
            if k != -1:
                segments.append([(X[i],Y[i]), (X[k],Y[k])])
    return segments

P = np.random.random((2,256)) ; X, Y = P[0], P[1]
fig = figure()
axes = subplot(1,1,1)
scatter( X, Y, s=5 ) ;
segments = voronoi( X, Y ) ;
lines = matplotlib.collections.LineCollection(segments, color='0.75') ;
axes.add_collection(lines) ;
axis([0,1,0,1]) ;
```

## 12.89 Windows Installer

To make a Windows installer you'll need the following items:

- py2exe
- InnoSetup
- ISTool

First, write a Python script, `count.py`:

```
#!/usr/bin/env python
```

```
import time

for i in range(10):
    print "{0} ha ha ha".format(i)

time.sleeo(3)
```

Then, write a Python setup script, `setup.py`:

```
#!/usr/bin/env python

from distutils.core import setup
import py2exe

setup(console=['count.py'])
```

Run the setup script at the terminal, in the same working directory as `count.py`:

```
$ python setup.py py2exe
```

This will produce two new directories, `build` and `dist`, in the working directory. Open `ISTools`. Using `ISTools`, click on `View` → `Sections` → `Files`, or `CTRL+1`, and then copy the contents of `dist` into the work area–the big empty space. Then go to `View` → `Sections` → `Script`, or `CTRL+0`, and try to compile. It will complain about missing fields, but you can guesstimate those values until it stops complaining. This process will produce a `.iss` script which, upon successful compilation, will produce an MSI installer, named `setup`, in the specified `OutputDir`. This is my final `count.iss` script.

```
[Files]
Source: C:\Users\connor\Desktop\dist\_hashlib.pyd; DestDir: {app}
Source: C:\Users\connor\Desktop\dist\bz2.pyd; DestDir: {app}
Source: C:\Users\connor\Desktop\dist\count.exe; DestDir: {app}
Source: C:\Users\connor\Desktop\dist\library.zip; DestDir: {app}
Source: C:\Users\connor\Desktop\dist\python27.dll; DestDir: {app}
Source: C:\Users\connor\Desktop\dist\select.pyd; DestDir: {app}
Source: C:\Users\connor\Desktop\dist\unicodedata.pyd; DestDir: {app}
[Setup]
DefaultDirName=Count
OutputDir=C:\Users\connor\Desktop\out
AppName=Counting
AppVerName=0.0
ShowLanguageDialog=no
```

This produces an MSI installer in the directory `OutputDir`. When the installer is run, the installation dialog will have the title `AppName`, and upon completion it will install a directory named `DefaultDirName`, which will contain the executable, `count.exe`, produced by `py2exe`.

## 12.90 Writing Data

NumPy offers a `np.tofile()` method for its ndarray class dumping data quickly into a text or binary file, and a `np.dump()` method for quickly pickling a file. However, it some information may be lost in the process. To be on the safe side, it's best to do this sort of thing manually. Bellow is a simple function for writing a 2D ndarray to a text file.

```
def array_to_file( x, fn ):
    '''
    Input:  (x)  a 2D ndarray
            (fn) a filename, the '.txt' part is added automatically
    Output: produces a txt file in the current directory
    '''
    fn += '.txt'
    f = open( fn, 'w' )
```

```
    rows, cols = x.shape
    for i in range( rows ):
        s = ''
        for j in range( cols ):
            s += str( x[i,j] ) + ' '
        f.write( s[:-1] + '\n' )
    f.close()
    return None
```

If you need speed, and who doesn't, really, you can use the `.tofile()` and `.fromfile()` functions.

```
import numpy as np

out_fid = open( 'filename.dat', 'wb' )
variable.tofile( fid )

in_fid = open( 'filename.dat', 'rb' )
x = np.fromfile( in_fid, dtype=np.float32 ).reshape( dimx, dimy )
```

We can also use `np.save()` to save files in `.npy` format, and `np.savez()` to save things in `.npz` format.

## 12.91 Web.py

Web.py is a lightweight web framework. It generates templates using `templator`.

I needed to set up a web page that had links to log generated by an ARM board. First, I made `run.py`. This creates the web server and sets up the URL structure.

```
#!/usr/bin/env python

import web

# these are paths, and class names
urls = ( '/', 'index', '/data', 'data' )

app = web.application( urls, globals() )

render = web.template.render('templates/')

class index:
    def GET( self ):
        return render.index()

class data:
    def GET( self ):
        return render.data()

if __name__=="__main__":
    app.run()
```

Then, in a directory named `templates`, we have `index.html`,

```
<!DOCTYPE HTML>
<HTML>
<HEAD>
<META HTTP-EQUIV="refresh" CONTENT="5" >
</HEAD>
<BODY>
<A HREF="data" TARGET="_blank"><BUTTON>data</BUTTON></A>
</BODY>
```

And then we have `data.txt`, a text file of data, floating around in `templates`. To run we do:

```
$ python run.py 8080
```

## 12.92 `zlib`

Sometimes we need to compress things a whole lot. The `zlib` module will compress a string with a compression level between 0 and 9, where 0 compresses really fast, and 9 compresses more slowly, but compresses things a lot smaller. You can use `cPickle` to make an arbitrary object into a string.

```
cPickle.dump( someObj, open( 'someObj.pkl', 'wb' ), -1 )
s = open( 'someObj.pkl', 'rb' )
t = zlib.compress( s, 9 )
fh = open( 'compressedObj.pk.zlib', 'wb' )
fh.write( t )
fh.close()
```

And then to uncompress:

```
s = open( 'compressedObj.pk.zlib', 'rb' ).read()
t = zlib.decompress( s ) # et voila
```

## 12.93 Miscellaneous

### 12.93.1 Hierarchical Image Difference

Let $R = 0$ be the largest scale, where the image is represented by one pixel. Then $R = 1$ is the next smallest scale, where the image is represented by 4 pixels, and $R = 2$ is the next smallest scale, where the image is represented by 16 pixels, and so on.

```
def hierarchical_scale( x ):
    scales = list()
    rows, cols = x.shape
    dim = np.max( x.shape )
    R = int( np.log2( dim ) )
    for r in range( R ):
        means = list()
        for i in range( 2**r ):
            for j in range( 2**r ):
                ibin =   i   * rows / 2**r
                ifin = (i+1) * rows / 2**r
                jbin =   j   * cols / 2**r
                jfin = (j+1) * cols / 2**r
                means.append( np.mean( x[ ibin:ifin, jbin:jfin ] ) )
        scales.append( means )
    return scales
```

Here is the code for calculating the difference between two images with this hierarchical decomposition:

```
def hierarchical_difference( x, y, scaled=True ):
    hsx, hsy = hierarchical_scale( x ), hierarchical_scale( y )
    s = 0
    for i in range( min( [ len( hsx ), len( hsy ) ] ) ):
        d = [ ( hsx[i][j] - hsy[i][j] )**2 for j in range( len( hsx[i] ) ) ]
        if scaled:
            d = np.array( d ) * ( 1.0 / 2**(i) )
        else:
            d = np.array( d ) * ( 1.0 / 2**(0) )
        m = np.mean( d )
        s += m
    return s
```

We can view the hierarchical decomposition of an image using this function,

```
def view_all_scales( data ):
    scales = hierarchical_scale( data )
    no_scales = len( scales )
    for i in range( no_scales ):
        s = scales[ i ]
        a = np.array( s ).reshape(( np.sqrt(len(s)), np.sqrt(len(s)) ))
        matshow( a, cmap='gray' ) ;
```

We can compare a set of people using the following code,

```
def compare_set( p, scaled ):
    m = list()
    for i in range( len( p ) ):
        for j in range( i+1, len( p ) ):
            m.append( [ hierarchical_difference( eval( p[i] ), eval( p[j] ), scaled ), p[i], p[j] ] )
    m.sort()
    return m

likeness = compare_set( people, True )
scaled = list()
for index, item in enumerate( likeness ):
    print '{0:20} {1:20} {2}'.format( likeness[index][1], likeness[index][2], np.sqrt(likeness[index][0]) )
    scaled.append( likeness[index][1]+' '+likeness[index][2] )
```

If we want to compare the difference between the scaled and unscaled differences, we can use the pearson rank correlation statistic along with the $t$-distribution using the following function,

```
import scipy.stats

def pearson_rank_correlation_coefficient( x, y, significance ):
    '''
    Calculate the Pearson Rank Correlation Coefficient and
    calculate a t-test to tell whether it differs significantly from zero or not
    '''
    indices = list()
    for index, item in enumerate( x ):
        indices.append( index - y.index( item ) )
    indices = np.array( indices )
    sqr_sum = np.sum( indices**2 )
    n = indices.size
    rho = 1 - ( 6 * sqr_sum )/float( n * ( n**2 - 1 ) )
    t = rho * np.sqrt( (n-2)/float(1-rho**2) )
    pvalue = 1 - scipy.stats.t.cdf( t, n-2 )
    if pvalue < significance:
        print "The p-value of {0}, with level of significance {1}, differs significantly from zero.".format( pval
    else:
        print "The p-value does not differ signifcantly from zero."
    return None
```

Here is the three dimensional extension of the hierarchical scale code,

```
def hierarchical_scale_3d( x ):
    scales = list()
    rows, cols, bands = x.shape
    dim = np.max( x.shape )
    R = int( np.log2( dim ) )
    for r in range( R ):
        means = list()
        for k in range( 2**r ):
            for i in range( 2**r ):
                for j in range( 2**r ):
                    ibin =   i   * rows  / float( 2**r )
                    ifin = (i+1) * rows  / float( 2**r )
                    jbin =   j   * cols  / float( 2**r )
```

```
                    jfin = (j+1) * cols  / float( 2**r )
                    kbin =   k   * bands / float( 2**r )
                    kfin = (k+1) * bands / float( 2**r )
                    m = np.mean( x[ ibin:ifin, jbin:jfin, kbin:kfin ] )
                    means.append( m )
        scales.append( means )
    return scales
```

### 12.93.2 Removing Strings with Commas from CSVs

If you need to parse/clean a csv with commas inside strings, e.g. `1,000`, you cannot use the *`split(',')`* function to split up each line of the csv, because it will break up the strings with the commas. The following code uses the re (regular expressions) module to match quote-delimited numbers with one comma, and replace them with the number sans quotes, sans comma. This was written quickly, for a specific data set, but it's a starting place.

The neat thing about the *`re.sub()`* function is that it will replace every single occurence of the match, not just the first one. (Score.)

```
import re

def repl( matchobj ):
    s = matchobj.group(0)
    s = s[1:-1]
    if ',' in s:
        idx = s.index(',')
        s = s[:idx] + s[idx+1:]
    return s

out = re.sub( ' ``\d+,\d+'' ', repl, s )
```

### 12.93.3 Green Tea Press

This is a really good place to start for developing algorithms in python for dealing with graphs. The interface for the graphs is intuitive and simple.

```
"""
Code example from _Computational_Modeling_
http://greenteapress.com/compmod

Copyright 2008 Allen B. Downey.
Distributed under the GNU General Public License at gnu.org/licenses/gpl.html.

"""

class Vertex(object):
    """a Vertex is a node in a graph."""

    def __init__(self, label=''):
        self.label = label

    def __repr__(self):
        """return a string representation of this object that can
        be evaluated as a Python expression"""
        return 'Vertex(%s)' % repr(self.label)

    __str__ = __repr__
    """the str and repr forms of this object are the same"""


class Edge(tuple):
```

```
    """an Edge is a list of two vertices"""

    def __new__(cls, *vs):
        """the Edge constructor takes two vertices as parameters"""
        if len(vs) != 2:
            raise ValueError, 'Edges must connect exactly two vertices.'
        return tuple.__new__(cls, vs)

    def __repr__(self):
        """return a string representation of this object that can
        be evaluated as a Python expression"""
        return 'Edge(%s, %s)' % (repr(self[0]), repr(self[1]))

    __str__ = __repr__
    """the str and repr forms of this object are the same"""


class Graph(dict):
    """a Graph is a dictionary of dictionaries.  The outer
    dictionary maps from a vertex to an inner dictionary.
    The inner dictionary maps from other vertices to edges.

    For vertices a and b, graph[a][b] maps
    to the edge that connects a->b, if it exists."""

    def __init__(self, vs=[], es=[]):
        """create a new graph.  (vs) is a list of vertices;
        (es) is a list of edges."""
        for v in vs:
            self.add_vertex(v)

        for e in es:
            self.add_edge(e)

    def add_vertex(self, v):
        """add (v) to the graph"""
        self[v] = {}

    def add_edge(self, e):
        """add (e) to the graph by adding an entry in both directions.

        If there is already an edge connecting these Vertices, the
        new edge replaces it.
        """
        v, w = e
        self[v][w] = e
        self[w][v] = e
```

### 12.93.4 Oleksii Kuchaiev

Here is a module for unweighted, undirected graphs. Most notably, there is an implementation of the second version of the Bron-Kerbosch algorithm for finding cliques. Cliques are maximal structures where each element is connected to each other element. Maximal means that no clique is a subset of another clique. However, this does not mean that different cliques must have empty intersection, cliques may very well overlap.

```
'''
Simple class for working with unweighted undirected graphs
@author: Oleksii Kuchaiev; http://www.kuchaev.com
'''
import random
class graph(object):
```

```
    '''
    A class for representing and manipulation undirected,
    unweighted simple graphs without self-loops
    '''
    def __init__(self, userID=None):
        '''
        Constructor
        '''
        if userID==None:
            self.Id=random.randint(0,10000000)
        else:
            self.Id=userID
        self.Nodes=set() #set of nodes
        self.AdjList=dict() #Adjacency list
    def add_node(self,node):
        '''
        Adds node to the graph.
        '''
        if node in self.Nodes:
            raise Exception("Node "+node+" is already present in the graph.")
        else:
            self.Nodes.add(node)
            self.AdjList[node]=set()
    def add_edge(self,nd1,nd2):
        '''
        Adds edge (nd1,nd2) to the graph.
        '''
        if nd1 not in self.Nodes:
            raise Exception("Node "+nd1+" is not present in the graph.")
        if nd2 not in self.Nodes:
            raise Exception("Node "+nd2+" is not present in the graph.")
        if nd1 not in self.AdjList.keys():
            self.AdjList[nd1]=set()
            self.AdjList[nd1].add(nd2)
        else:
            self.AdjList[nd1].add(nd2)
        if nd2 not in self.AdjList.keys():
            self.AdjList[nd2]=set()
            self.AdjList[nd2].add(nd1)
        else:
            self.AdjList[nd2].add(nd1)
    def readFromEdgeList(self,path):
        '''
        Read graph from the file where it is represented as an edge list.
        The lines of the file should be formated as:
        node1[space]node2[newline]
        Duplicate edges and self loops are ignored.
        '''
        inp_file=open(path,'r')
        _Nodes=set() # this will replace self.Nodes in case of success
        _AdjList=dict() # this will replace self.AdjList in case of success
        for line in inp_file:
            nodes=line.split()
            if len(nodes)<2:
                raise Exception("There is an incorrectly formatted line\
                                in the edge list file")
            nd1=nodes[0]
            nd2=nodes[1]
            if nd1==nd2:
                continue
            if (nd1 in _Nodes):
                if (nd2 not in _AdjList[nd1]):
                    _AdjList[nd1].add(nd2)
            else:
                _Nodes.add(nd1)
```

```
            _AdjList[nd1]=set()
            _AdjList[nd1].add(nd2)
        if (nd2 in _Nodes):
            if (nd1 not in _AdjList[nd2]):
                _AdjList[nd2].add(nd1)
        else:
            _Nodes.add(nd2)
            _AdjList[nd2]=set()
            _AdjList[nd2].add(nd1)
    inp_file.close()
    self.Nodes.clear()
    self.AdjList.clear()
    self.Nodes=_Nodes
    self.AdjList=_AdjList
def get_edge_set(self):
    '''
    Returns set of edges in the graph.
    '''
    Edges=set()
    for nd1 in self.Nodes:
        N=self.get_node_neighbors(nd1)
        for nd2 in N:
            if (nd2,nd1) not in Edges:
                Edges.add((nd1,nd2))
    return Edges
def saveAsEdgeList(self,path):
    '''
    Saves graph as edge list
    '''
    out=open(path,'w')
    EdgeList=self.get_edge_set()
    for edge in EdgeList:
        line = edge[0]+" "+edge[1]+"\n"
        out.write(line)
    out.close()
def number_of_nodes(self):
    '''
    Returns number of nodes in the graph.
    '''
    return len(self.Nodes)
def number_of_edges(self):
    '''
    Returns number of edges in the graph.
    '''
    num_edg=0.0;
    for key in self.AdjList.keys():
        num_edg=num_edg+(float(len(self.AdjList[key]))/2)
    return int(num_edg)
def degree(self,node):
    '''
    Returns the degree of a node
    '''
    if node not in self.Nodes:
        raise Exception("There is no node with name: "+str(node)+" in this graph.\
                        The id of the graph is: "+str(self.Id))
    return len(self.AdjList[node])
def get_node_clust_coef(self,node):
    '''
    Returns the clustering coefficient of the node
    '''
    deg=self.degree(node)
    if deg<=1:
        return 0
    Ev=0
    neighbors=self.get_node_neighbors(node)
```

```
        for nd in neighbors:
            if self.are_adjacent(node, nd):
                Ev+=1
        cc=float(2*Ev)/(deg*(deg-1))
        return cc
    def get_node_eccentricity(self,node):
        '''
        Returns the eccentricity of the node.
        Note that this function returns the eccentricity of a node within its
        connected component
        '''
        D=self.BFS(node)
        ec=0
        for key, value in D:
            if value>ec:
                ec=value
        return ec
    def get_node_eccentricity_avg(self,node):
        '''
        Returns the averaged eccentricity of the node. That is, "avg", not "max" distance
        Note that this function returns the eccentricity of a node within its
        connected component
        '''
        D=self.BFS(node)
        ec=0.0
        counter=0.0
        for key, value in D.items():
            if value>0:
                ec+=value
                counter+=1
        if counter>0:
            return ec/counter
        else:
            return 0
    def get_node_eccentricities_both(self,node):
        '''
        This function is for performance purposes.
        This is function returns standard and averaged eccentricities of the node.
        Note that both eccentricities of the node are within its connected component
        '''
        D=self.BFS(node)
        ec=0
        ecA=0.0
        counter=0.0
        for key, value in D.items():
            if value>0:
                ecA+=value
                counter+=1
                if value>ec:
                    ec=value
        if counter>0:
            return (ec,ecA/counter)
        else:
            return (ec,0)
    def are_adjacent(self,nd1,nd2):
        '''
        Checks if nd1 and nd2 are connected
        '''
        if nd1 not in self.Nodes:
            raise Exception("Node "+str(nd1)+" is not in the graph with id="+str(self.Id))
        if nd2 not in self.Nodes:
            raise Exception("Node "+str(nd2)+" is not in the graph with id="+str(self.Id))
        if nd2 in self.AdjList[nd1]:
            return True
        else:
```

```
        return False
    def get_node_neighbors(self,nd):
        '''
        Returns set of node neighbors
        '''
        #if nd not in self.Nodes:
           # raise Exception("Node "+str(nd)+" is not in the graph with id="+str(self.Id))
        return self.AdjList[nd]
    def BFS(self,source):
        '''
        Implements Breadth-first search from node 'source' in graph 'self'.
        Returns dictionary D {node: distance from source}
        distance=-1 if 'node' is unreachable from 'source'
        '''
        #if source not in self.Nodes:
        #   raise Exception("Node "+str(source)+" is not in\
        #                    the graph with id="+str(self.Id))
        D=dict();
        for node in self.Nodes:
            D[node]=-1
        level=0;
        Que0=set()
        Que0.add(source)
        Que1=set()
        while len(Que0)!=0:
            while len(Que0)!=0:
                cur_node=Que0.pop()
                D[cur_node]=level
                N=self.AdjList[cur_node]
                for nd in N:
                    if D[nd]==-1:
                        Que1.add(nd)
            level=level+1
            Que0=Que1
            Que1=set()
        return D
    def dist(self,nd1,nd2):
        '''
        Returns shortest-path distance between nd1 and nd2
        '''
        if nd1 not in self.Nodes:
            raise Exception("Node "+str(nd1)+" is not in the graph with id="+str(self.Id))
        if nd2 not in self.Nodes:
            raise Exception("Node "+str(nd2)+" is not in the graph with id="+str(self.Id))
        D=dict();
        for node in self.Nodes:
            D[node]=-1
        level=0;
        Que0=set()
        Que0.add(nd1)
        Que1=set()
        while len(Que0)!=0:
            while len(Que0)!=0:
                cur_node=Que0.pop()
                D[cur_node]=level
                if cur_node==nd2:
                    return level
                N=self.get_node_neighbors(cur_node)
                for nd in N:
                    if D[nd]==-1:
                        Que1.add(nd)
            level=level+1;
            Que0=Que1;
            Que1=set()
        return -1
```

```
    def all_pairs_dist(self):
        '''
        Returns dictionary of all-pairs shortest paths in 'self'
        The dictionary has format {t=(nd1,nd2): distance},
        where t is a tuple.
        '''
        Distances=dict()
        count=0
        for nd in self.Nodes:
            DD1=self.BFS(nd)
            for key, value in DD1.items():
                t1=nd, key
                t2=key, nd
                Distances[t1]=float(value)
                Distances[t2]=float(value)
        return Distances
    def find_all_cliques(self):
        '''
        Implements Bron-Kerbosch algorithm, Version 2
        '''
        Cliques=[]
        Stack=[]
        nd=None
        disc_num=len(self.Nodes)
        search_node=(set(),set(self.Nodes),set(),nd,disc_num)
        Stack.append(search_node)
        while len(Stack)!=0:
            (c_compsub,c_candidates,c_not,c_nd,c_disc_num)=Stack.pop()
            if len(c_candidates)==0 and len(c_not)==0:
                if len(c_compsub)>2:
                    Cliques.append(c_compsub)
                    continue
            for u in list(c_candidates):
                if (c_nd==None) or (not self.are_adjacent(u, c_nd)):
                    c_candidates.remove(u)
                    Nu=self.get_node_neighbors(u)
                    new_compsub=set(c_compsub)
                    new_compsub.add(u)
                    new_candidates=set(c_candidates.intersection(Nu))
                    new_not=set(c_not.intersection(Nu))
                    if c_nd!=None:
                        if c_nd in new_not:
                            new_disc_num=c_disc_num-1
                            if new_disc_num>0:
                                new_search_node=(new_compsub,new_candidates,\
                                                 new_not,c_nd,new_disc_num)
                                Stack.append(new_search_node)
                        else:
                            new_disc_num=len(self.Nodes)
                            new_nd=c_nd
                            for cand_nd in new_not:
                                cand_disc_num=len(new_candidates)-len(\
                                              new_candidates.intersection(\
                                              self.get_node_neighbors(cand_nd)))
                                if cand_disc_num<new_disc_num:
                                    new_disc_num=cand_disc_num
                                    new_nd=cand_nd
                            new_search_node=(new_compsub,new_candidates,new_not,\
                                             new_nd,new_disc_num)
                            Stack.append(new_search_node)
                    else:
                        new_search_node=(new_compsub,new_candidates,new_not,c_nd,c_disc_num)
                        Stack.append(new_search_node)
                    c_not.add(u)
                    new_disc_num=0
```

```
                for x in c_candidates:
                    if not self.are_adjacent(x, u):
                        new_disc_num+=1
                if new_disc_num<c_disc_num and new_disc_num>0:
                    new1_search_node=(c_compsub,c_candidates,c_not,u,new_disc_num)
                    Stack.append(new1_search_node)
                else:
                    new1_search_node=(c_compsub,c_candidates,c_not,c_nd,c_disc_num)
                    Stack.append(new1_search_node)
    return Cliques
def create_empty_graph(self,n):
    '''
    creates graph with n nodes but without edges
    '''
    G=graph()
    for i in range(1,n+1):
        nd=str(i)
        G.add_node(nd)
    return G
def create_path_graph(self,n):
    '''
    Create path-graph on n nodes
    '''
    if n<2:
        raise Exception("Can't create a path graph with less than 2 nodes.")
    G=self.create_empty_graph(n)
    for i in range(1,n):
        nd1=str(i)
        nd2=str(i+1)
        G.add_edge(nd1, nd2)
    return G
def create_cycle_graph(self,n):
    '''
    Creates graph-cycle on n nodes.
    Nodes are numbered from 1 to n.
    '''
    if n<3:
        raise Exception("Can't create a cycle with less than 3 nodes.")
    G=self.create_path_graph(n)
    G.add_edge(str(1), str(n))
    return G
def create_circulant_graph(self,n,j):
    '''
    Creates a circulant graph with n nodes and m edges.
    Nodes are numbered from 1 to n.
    The chords are defined by parameters j.
    That is node i is connected to (i-j) mod n and (i+j) mod n.
    Note that not always the exact match in terms of edges is possible!
    Check the output graph for the number of edges!
    '''
    G=self.create_cycle_graph(n) # node numbers starts from 1
    for i in range(0,n):
        ndi=str(i % n + 1)
        nd2 = str(( i - j ) % n + 1)
        nd3 = str(( i + j ) % n + 1)
        G.add_edge(ndi, nd2)
        G.add_edge(ndi, nd3)
    return G
def create_complete_graph(self,n):
    '''
    creates complete graph on n nodes
    '''
    G=self.create_empty_graph(n)
    L=list(G.Nodes)
    for i in range(0,len(L)):
```

```
            for j in range(i+1,len(L)):
                G.add_edge(L[i], L[j])
        return G
```

# Chapter 13

# MATLAB

(top)

## 13.1 Introduction

You can use `help` to find out more about a function, or `type` to see the function's source code, if it is not a MEX file or something. Note that MATLAB uses matrix multiplication by default, so if you want to multiply two matrices (or divide, or raise to a power) in an elementwise fashion, then you have to prepend that operator with a period.

## 13.2 Loops

```
for index = 1:10
    body ;
end
```

```
while condition ;
    body ;
end
```

You can also iterate over an arbitrary array.

```
x = [ 0 1 1 2 3 5 ] ;
for i = x
    disp(i) ;
end
```

This will print the first six numbers of the Fibonacci sequence.

## 13.3 Control Structures

```
if condition
    body ;
end
```

```
if condition
    body1 ;
else
    body2 ;
end
```

```
if condition
    body1 ;
elseif
    body2 ;
else
    body ;
end
```

## 13.4 Functions

MATLAB functions are scripts that return zero or more values. Write a script as below, and save it as `filename.m`. Then if you navigate to the right directory in the command window, you can call the function by typing filename.

```
function y = filename(x)
    body ;
    return
```

## 13.5 Strings

### 13.5.1 Print to the Screen

You can use `disp()` to print messages to the screen. More later..

### 13.5.2 Comparison

If you need to compare two strings in an `if` control structure, you can use `strcmp()`. Note that the equality operator, `==`, is not overloaded for strings.

```
if strcmp( str1, str2 )
   disp( 'FTW' ) ;
end
```

### 13.5.3 Catenation

The function `strcat()` is used for concatenating strings, but it chops off the whitespace around strings, so you need to add spaces carefully, i.e. as cells.

```
tom = 'tom ' ;
dick = 'dick ' ;
harry = 'harry !' ;
s = strcat( tom, dick, harry ) ;
% ---> tomdickharry !
s = strcat( tom, {', '}, dick, {', '}, harry ) ;
% ---> tom, dick, harry !
```

## 13.6 Appending Elements to an Array

In Python, we can append to lists nbd. In MATLAB, there is a slightly different way of doing this. To produce a $1 \times 10$ matrix:

```
for i = 1:10
    x = [ x i ] ;  % OR  x = [ x, i ] ;
end
```

To produce a $10 \times 1$ matrix:

```
for i = 1:10
    x = [ x ; i ] ;
end
```

## 13.7 Formatting Numbers

You can force the proper formatting by using `format rat`, for rational numbers, and `format long`, for more precision.

## 13.8 Argmin, Argmax

Typically, `argmin()` or `argmax()` return the index of the minimum (or maximum) value of an array, rather than the minimum (or maximum) value, itself. MATLAB does not have a separate `argmin()` or `argmax()` function, instead:

```
>> x = [ 2  3  5  1  8  13 ] ;
>> min( x ) ;                  % ans = 1
>> [C,I] = min( x ) ;          % C = 1, I = 4
```

## 13.9 Creative Indexing

This produces a the cross product of two matrices, using MATLAB indexing, i.e., $1 \leq \texttt{idx} \leq \texttt{limit}$. So, if you have two matrices of the same size, and you want every ordered pair of the elements between these two matrices, you'd use this code. N.B. `(ix,jx)` is an element of the first matrix, and `(iy,jy)` is an element of the second matrix.

```
for kx = 1 : i_tot * j_tot
    for ky = 1 : i_tot * j_tot

        ix = floor( kx / (j_tot + 1) ) + 1;
        jx = mod( kx - 1, j_tot ) + 1;

        iy = floor( ky / (j_tot + 1) ) + 1;
        jy = mod( ky - 1, j_tot ) + 1;

    end
end
```

## 13.10 Dimensions

To determine the number of rows and columns in a matrix, use `size`.

```
>> [ rows, cols ] = size( mat ) ;
```

To determind the number of dimensions of a matrix, use `ndims`

```
>> n = ndims( mat ) ;
```

## 13.11 Help and Code

Two MATLAB functions are incredibly useful. These are help and type. The first will print out the documentation for a function, the second will display the function's code, if it is in your path.

```
>>> help <object>
>>> type <object>
```

## 13.12 Zeros, Ones, and Eyes

Identity matrices, and matrices formed of zeros or ones are produced by:

```
>> eye(5) ;
>> zeros(5,5) ;
>> ones(5,5) ;
```

## 13.13 GUIDE

Start `GUIDE` by typing,

```
>> guide
```

Yeah. It's that easy. Lay out your GUI. Once you have your proportions set up, you can allow users to resize the window using `Tools --> GUI Options`. For each element, right-click and open up the `Property Inspector`. From the `Property Inspector` edit the `String` property to change the labels on the buttons, or the default text in the dynamic text boxes, and edit the `Tag` property in order to have more human-readable variable names in the callback code. There other properties, but the `String` and `Tag` properties are particularly important.

Callback functions execute the code in their body when their button (or whatever) is pressed. The callback code offers you a structure, `handles`, that lets you access the same variable in multiple callback functions. Remember to update the `handles` structure after altering it by adding the line,

```
guidata( hObject, handles ) ;
```

## 13.14 Write Data to CSV

```
x = 0:.1:1;
A = [x; exp(x)];

fileID = fopen('exp.txt','w');
fprintf(fileID,'%6s %12s\n','x','exp(x)');
fprintf(fileID,'%6.2f %12.8f\n',A);
fclose(fileID);
```

## 13.15 HDF5

First, you need to create an `HDF5` file. This involves specifying the filename as a string, naming the data set with a string, and specifying the size of the data set.

```
>> h5create( 'filename.h5', '/data', [ rows, cols ] ) ;
```

Next you can write your data to the `HDF5` file, and view a summary.

```
>> h5write( 'filename.h5', '/data', data ) ;
>> h5disp( 'filename.h5' ) ;
```

To open an HDF5 file, you need to specify the filename and the specific data set within that file.

```
>> d = h5read( 'filename.h5', '/data' ) ;
>> imagesc( d ) ; % voila
```

## 13.16 Classes

In order to change values of properties, you need to inherit from the `handle` class. Tres important. Otherwise, property names are declared, and then the first function initializes the object, and the rest of the functions.. functificate on properties and parameters.

```
classdef NameOfClass < handle
    properties
        prop1 ;
        prop2 ;
    end
    methods
        function obj = NameOfClass( prop1, prop2 )
            obj.prop1 = prop1 ;
            obj.prop2 = prop2 ;
        end
        function obj = fct1()
            stmts ;
        end
    end
end
```

## 13.17 `linspace`

This guy works like `start:step:stop`, but instead of specifying the *step size*, we specify the *number* of steps between `start` and `stop`.

## 13.18 Plotting

### 13.18.1 Plot

This does what one would naturally expect. In order to add new plots to the current figure, use `hold ;` or `hold all ;` after the first plot.

### 13.18.2 Heatmap

In order to make a heatmap, you use the `pcolor()` command, but be wary! This plot does two things:

- Cuts off last row and last column
- Flips the data vertically (up-down)

Therefore, one way to get aroud this is to do the following:

```
% suppose x is our data
x = rand(3,3) ;

% create a matrix with an extra row and column
z = zeros(3+1,3+1) ;

% flip x vertically and put it into z
z(1:3,1:3) = flipud( x )

% voila!
pcolor( z )
```

### 13.18.3 Subplots

For a $2 \times 3$ subplot, we can do the following

```
rows = 2 ;
cols = 3 ;
k = 1 ;
for row = 1:rows
    for col = 1:cols
        subplot( rows, cols, k ) ;
        plot( x ) ;
    title( strcat( {'Trial '}, {num2str(j)} ) ;
        k = k + 1 ;
    end
end
suptitle( 'Trials' ) ;
```

### 13.18.4 Colorbar

To add a colorbar over the limits $[0, 1]$, use the commands:

```
colorbar ;
caxis([ 0 1 ]) ;
```

### 13.18.5 Axis Limits

If you need to force the axis limits to some maximum and minimum, you can do so using:

```
xlim( [ 0 1 ] ) ;
ylim( [ 0 1 ] ) ;
```

### 13.18.6 Ticks

To change the ticks and the tick labels, we can do the following. Note that `gca` stands for *g*rab *c*urrent *a*xis.

```
set( gca, 'XTick', 1.5:1:13  ) ;
set( gca, 'XTickLabel', linspace(10,120,12) ) ;
```

### 13.18.7 Legend

```
legend( 'First Item', 'Second Item' ) ;
```

# Chapter 14

# R

(top)

## 14.1 Loading Data

Here is the syntax for loading a space-delimited table into R.

```
> mydata <- read.table"(data."dat)
```

## 14.2 Accessing Rows and Columns

After you have loaded the above data into R, you'll want to access its rows and columns

```
> mydata <- read.table"(data."dat) # load data
> ### COLUMNS ###
> mydata[,1] # returns the first column
> mydata[,2] # returns the second column
> ##### ROWS #####
> mydata[1,] # returns the first row
> mydata[2,] # returns the second row
> ### ELEMENTS ###
> mydata[1,1] # returns the element in the first row of the first column
```

## 14.3 Biclust

This package allows you to do biclustering, but first you need to turn the data into a matrix.

```
> data <- read.table( "data."dat ) # load data
> data <- as.matrix( data ) # transform into a matrix
> library( biclust )
> t_bccc <- biclust( data, method=BCCC(), delta=1.5, alpha=1.0, number=10 )
> t_bccc@RowxNumber # logical matrix
> t_bccc@NumberxCol # logical matrix
```

## 14.4 Installing and Loading a Package

Download the tar.gz file from CRAN or someplace. Then fun the following in the command line:

```
$ R CMD INSTALL <insert name of tar.gz pkg here>
$
```

To clean up, open R and run the following:

```
> update.packages()
```

Alternatively, use this procedure:

```
$ sudo R
$
```

This will start R and give you the necessary permissions for everything and then you can begin installing and loading things:

```
> install.packages( 'pkg', lib='/usr/lib/R/site-library', dep=TRUE )
> library( 'pkg' ) % use this
Loading required package: pkg...
```

## 14.5 Create a Matrix of Random Values

The following creates a matrix of 5 rows and three columns with values taken from the random uniform distribution from -7.5 to 9.5.

```
> x <- replicate( 3, runif( 5, -7.5, 9.5 ) )
> x
[,1]
[,2]
[,3]
[1,] 7.8910288 9.404851 4.107711
[2,] -0.4057908 -3.889538 -5.711836
[3,] -6.6257709 7.352795 7.024050
[4,] 2.7594948 -3.328867 6.397929
[5,] -1.1413127 -1.695697 8.563213
```

## 14.6 Shapiro-Wilk Test for Normality

The null-hypothesis is that the data is normally distributed. If the p-value is greater than the alpha value, which is usually 0.05 or 0.01, then the data is normally distributed. Remember, that the p-value is the test value, and the alpha-value is some threshold. If the data is over the threshold, then the data is normal.

```
> shapiro.test(b[,1])
# normal
Shapiro-Wilk normality test
data: b[, 1]
W = 0.9774, p-value = 0.6089
> shapiro.test(b[,2])
# non-normal
Shapiro-Wilk normality test
data: b[, 2]
W = 0.921, p-value = 0.009347
```

## 14.7 Principal Components Analysis

This does actual job and put the results to pcdat. It will use covariance matrix

```
> pcdat = princomp( data )
```

This will use correlation matrix

```
> pcdat = princomp( data, cor=TRUE )
```

This will print standard deviation and proportion of variances for each component

```
> summary( pcdat )
```

This will plot screenplot:

```
> screeplot( pcdat )
```

This will give you biplot:

```
> bilot( pcdat, scale=1 )
```

This will give information how much each variable contribute to each component. For principal components you can ignore loading subsection of the output from this command

```
> loadings( pcdat )
```

This will plot scores of each observation for each variable:

```
> pcdat$scores
```

## 14.8 fdim

To install, type this at the command line:

```
$ sudo R
```

This will start R and give you the necessary permissions for everything:

```
> install.packages( 'fdim', lib='/usr/lib/R/site-library', dep=TRUE )
> library( 'fdim' ) % use this
Loading required package: xgobi
> d <- read.table( 'data.dat' )
> fdim( d )
This calculates the fractional dimension of stuff.
> fdim(X, BaseR=2, Mnmax=TRUE, nMax=9, NumMinP=1, q=0, Alpha=0.2, PlotF=FALSE)
```

## 14.9 Spinning 3D Scatterplots

First, download `rgl` as outlined above.

```
> library( rgl )
> x <- rnorm(5)
> y <- rnorm(5)
> z <- rnorm(5)
> plot3d( x, y, z, col='red', size=3 )
```

## 14.10 Bayesian Computation

### 14.10.1 Introduction

These are notes from Jim Albert's *Bayesian Computation with R*. We'll look at the basic elements of the Bayesian inferential approach through the basic problem of learning about a population proportion. Before taking data, one has beliefs about the value of the proportion, and one models these beliefs in terms of a prior distribution. We'll look at different functional forms for this prior. After data have been observed, one updates one's beliefs about the proportion by the computation of the posterior. One then summarizes the this probability distribution to perform inferences, and possibly predicting the likely outcomes of a new sample taken from the same population. In summary:

- Formalize beliefs into a prior distribution
- Calculate the posterior distribution
- Perform inference
- Perform prediction

### 14.10.2 Proportion Exercise

We want to determine the proportion of college students that get at least eight hours of sleep. Thus, the population is the set of American college students, and $p$ is the proportion of them that get at least eight hours of sleep a night.

The value of $p$ is unknown, but in the Bayesian view, one's beliefs about the uncertainty in this proportion are represented by a probability distribution placed on this parameter. This distribution reflects one's subjective prior opinion about plausible values of $p$.

After reading some books one determines that $p < 0.5$ and that $p \approx 0.3$, meaning that probably only 30% of college students get eight hours a night.

A sample of 27 students is taken, and 11 respond that they had at least 11 hours of sleep the previous night. Based on the prior information and the observed data, we'd like to estimate $p$. We'd also like to predict the number of wel-rested students if a new sample of 20 is taken.

Let $g(P)$ be the prior distribution. If we regard "success" as sleeping at least eight hours, and we take a random sample with $s$ successes and $f$ failures, then the likelihood function is given by,

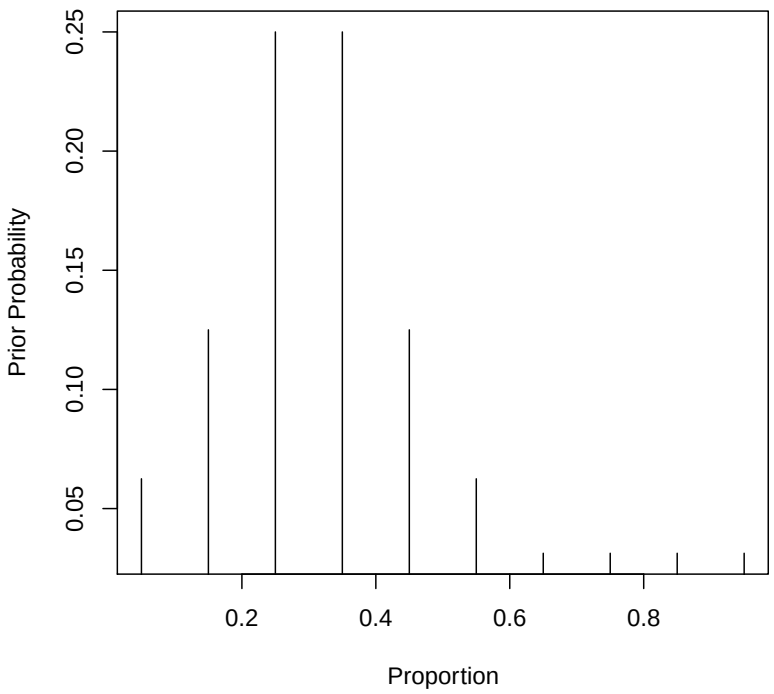
$$L(p) = p^s(1-p)^f.$$

The posterior is obtained, up to a proportionality constant, by the product of the prior and the likelihood.

$$g(p|data) \propto g(p)L(p)$$

We'll calculate the posterior using three methods for representing one's prior knowledge about the proportion,

- Discrete prior
- Continuous Beta prior
- Histogram prior




### 14.10.3 Discrete Prior

One approach is to list possible values for $p$ and then assign weights to those values. Let's suppose that the following are reasonable values for $p$,

$$0.05, 0.15, 0.25, 0.35, 0.45, 0.55, 0.65, 0.75, 0.85, 0.95,$$

and that the following are resonable weights,

$$2, 4, 8, 8, 4, 2, 1, 1, 1, 1$$

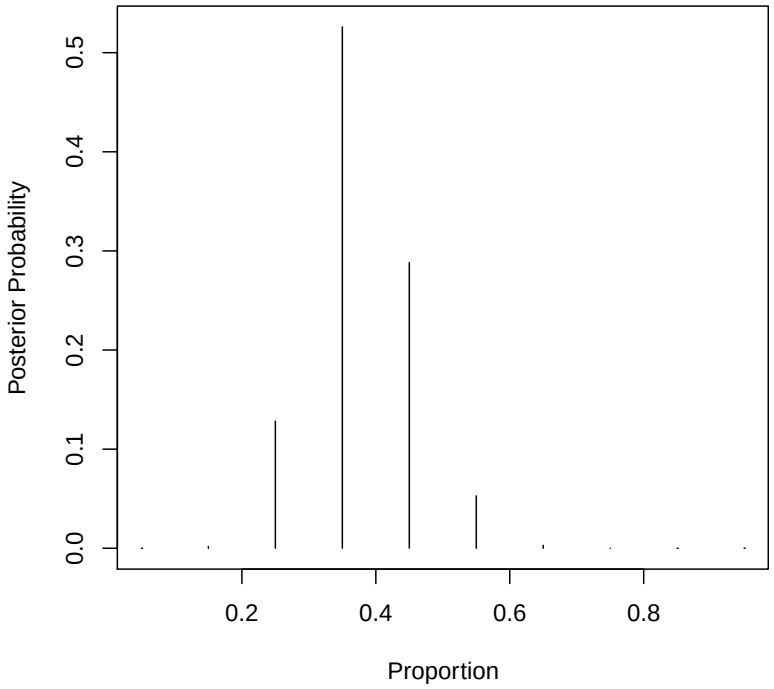
These weights can be converted to prior probabilities by dividing each weight by their sum. When er enter this information, note the `seq` function, and the `c` function, which I believe stands for "column vector".

```
> p = seq( 0.05, 0.95, by=0.1 )
> prior = c(2,4,8,8,4,2,1,1,1,1)
> prior = prior / sum(prior)
> plot( p, prior, type="h", xlab="Proportion", ylab="Prior Probability" )
```

In our sample, 11 out of 27 students slept eight hours, so $s = 11$, and $f = 16$, and the likelihood function becomes,

```
L(p) = p^{11}(1-p)^{16),\quad 0<p<1
```

The function `pdisc` in the package *LearnBayes* computes the posterior probabilities. This function takes the proportion values, the prior vector, and a data vector consisting of $s$ and $f$. The output is the a vector of posterior probabilities. The `cbind` command is used to display a table of prior and posterior probabilities.




```
> data = c(11,16)
> post = pdisc( p, prior, data )
> cbind( p, prior, post )
          p   prior         post
 [1,] 0.05 0.06250 2.882642e-08
 [2,] 0.15 0.12500 1.722978e-03
 [3,] 0.25 0.25000 1.282104e-01
 [4,] 0.35 0.25000 5.259751e-01
 [5,] 0.45 0.12500 2.882131e-01
 [6,] 0.55 0.06250 5.283635e-02
 [7,] 0.65 0.03125 2.976107e-03
 [8,] 0.75 0.03125 6.595185e-05
 [9,] 0.85 0.03125 7.371932e-08
[10,] 0.95 0.03125 5.820934e-15
> plot( p, post, type='h', xlab='Proportion', ylab='Posterior Probability' )
```

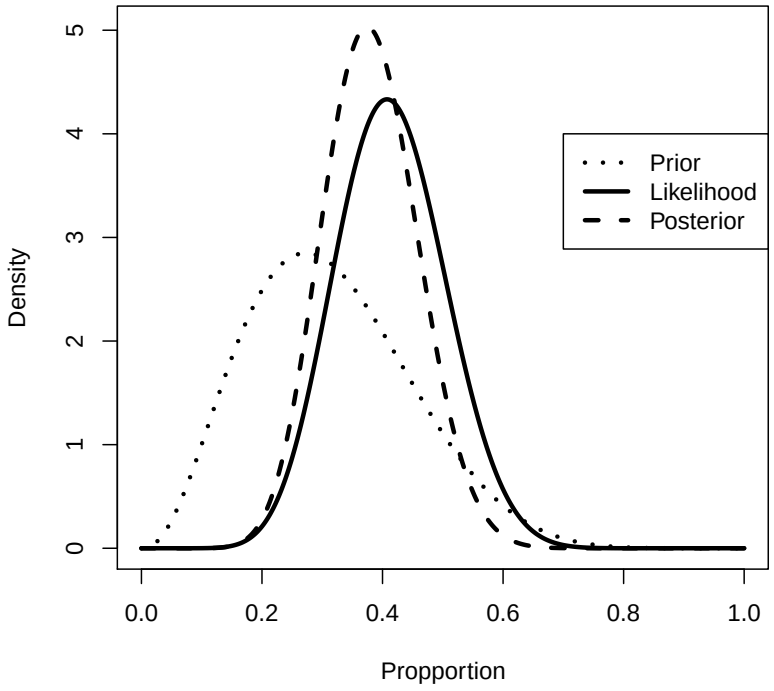
We see that the most likely values for the proportion are 0.25, 0.35, and 0.45, and that this set accounts for 0.942% of the posterior.

### 14.10.4 Beta Prior

Another approach involves constructing a density $g(p)$ on the interval $(0, 1)$. We believe that the median proportion is 0.3, i.e., that it is equally likely that the true value of $p$ lies above or below 0.3. Furthermore, we are 90% condifent that $p$ is less than 0.5.

A convenient family of densities is the Beta family, with a kernel proportional to,

$$g(p) \propto p^{a-1}(1-p)^{b-1}, \quad 0 < p < 1$$



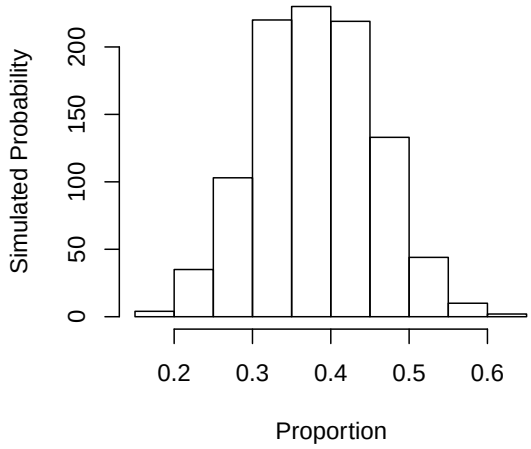

After fiddling with beta distributions, we can settle on $a = 3.4$ and $b = 7.4$for a median of 0.3 and a 90th percentile at 0.5. This fiddling is achieved by using the following functions,

```
> p = seq( 0, 1, length=500 )
> b = dbeta( p, 3.4, 7.4 )
> median( b )
[1] 0.5579913
> quantile( b, 0.9 )
     90%
2.693041
```

That looks like the median is around 0.5, and that the 90th percentile is at 2.6, but whatever, that looks close enough. We plot the results as,

```
> p = seq( 0, 1, length=500 )
> a = 3.4
> b = 7.4
> s = 11
> f = 16
> prior = dbeta( p, a, b )
> like  = dbeta( p, s+1, f+1 )
> post  = dbeta( p, a+s, b+f )
> plot( p, post, type='l', xlab='Proportion', ylab='Density', lty=2, lwd=3 )
> lines( p, like, lty=1, lwd=3 )
> lines( p, prior, lty=3, lwd=3 )
> legend( 0.7, 4, c('Prior','Likelihood','Posterior'),lty=c(3,1,2),lwd=c(3,3,3))
```

The beta cdf and inverse cdf functions are `pbeta` and `qbeta`, respectively. To determine the posterior probability $\Pr(p > 0.5|\text{data})$ we enter,




```
> 1 - pbeta( 0.5, a+s, b+f )
[1] 0.06842569
```

This is a pretty small probability, so it's unlikely that more than half of the students are well-rested. A 90% interval estimate for $p$ can be obtained by computing the 5th and 95th percentiles of the beta density,

```
> qbeta( c(0.05, 0.95), a+s, b+f )
[1] 0.2562364 0.5129274
```

So we are 90% confident that the proportion is between 0.256 and 0.513.

*Alternatively*, we could perform this summarization of the posterior based on simulation. Using the random beta command, `rbeta`, we simulate 1000 random proportion values from the $Beta(a + s, b + f)$ posterior using the command,

```
> ps = rbeta( 1000, a+s, b+f )
> hist( ps, xlab='Proportion', ylab='Simulated Probability', main='')
```

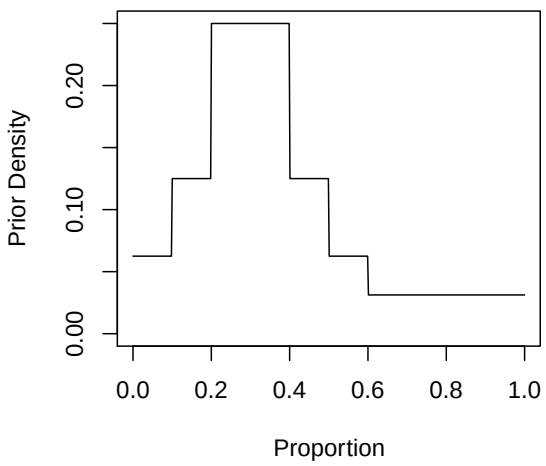
Then we can perform similar summary operations,

```
> sum( ps >= 0.5 )/1000
[1] 0.056
> quantile( ps, c(0.05, 0.95) )
0.2608688 0.5078019
```

### 14.10.5 Histogram Prior

Although it is convenient to use beta priors, it is good to know how to perform posterior computations for any choice of prior. We outline a "brute-force" technique of summarizing posterior computations for an arbitrary density $g(p)$.

- Choose a grid of values of $p$ over an interval that covers the posterior density.
- Compute the product of the likelihood $L(p)$ and the prior $g(p)$ on the grid.




- Normalize by dividing each product be the sum of the products. Here, we are approximating the posterior density by a discrete probability distribution on the grid.
- Using the R command `sample`, take a random sample with replacement from the discrete distribution. The simulated draws represent approximate samples from the posterior distribution.

We illustrate this "brute-force" algorithm for a "histogram" prior. Suppose we have 10 bins,

$$(0.0, 0.1), (0.1, 0.2), \ldots, (0.9, 1.0)$$

and their corresponding weights,

$$2, 4, 8, 8, 4, 2, 1, 1, 1, 1$$

Then we use the `histprior` from the *LearnBayes* package. Here is the code so far,

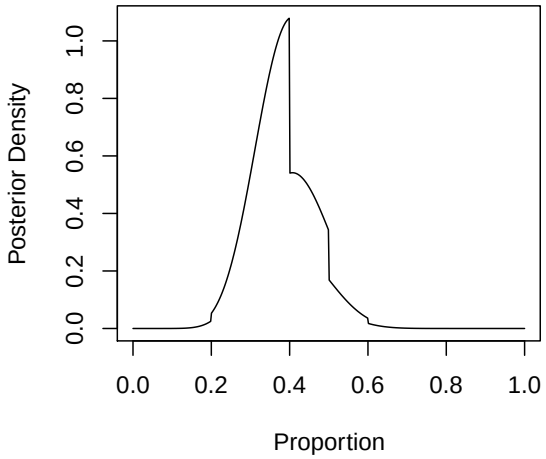
```
> midpt = seq( 0.05, 0.95, by=0.1 )
> prior = c(2,4,8,8,4,2,1,1,1,1)
> prior = prior/sum(prior)
> p = seq( 0, 1, length=500 )
> hp = histprior( p, midpt, prior )
> plot( p, hp, type='l', xlab='Proportion', ylab='Prior Density', ylim=c(0.0,0.25) )
```

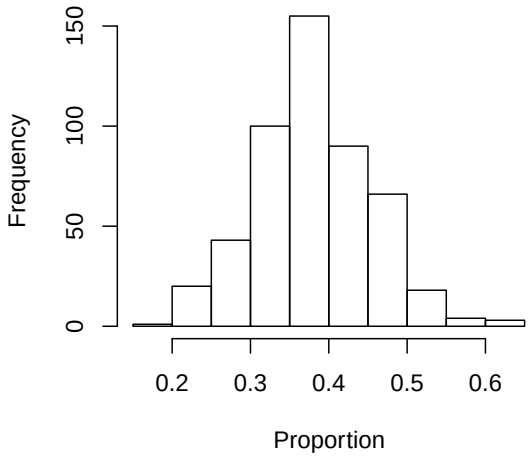
We compute the posterior by multiplying the histogram prior by the likelihood function, which for binomial density, is the given by the *Beta(*$s+1, f+1$*)* density, available as the `dbeta` function.

```
> like = dbeta( p, s+1, f+1 )
> post = like * hp
> plot( p, post, type='l', ylab='Posterior Density', xlab='Proportion' )
```

For summarization tasks, we normalize the posterior, and then sample it,

```
> post = post / sum( post )
> ps = sample( p, replace=True, prob=post )
> hist( ps, xlab='Proportion', ylab='Frequency', main='' )
```


Posterior Density
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0
Proportion


Frequency
0
50
100
150
0.2
0.3
0.4
0.5
0.6
Proportion

### 14.10.6 Prediction

We may produce prior predictive densities, or posterior predictive densities.

Suppose we use a discrete prior where $\{p_i\}$ represent the proportion values, and $\{g(p_i\}$ represents their probabilities. Let $f_B(y|n,p)$ represent the binomial sampling density, given the values of the sample size, $n$, and the proportion, p,

$$f_B(y|n,p) = \binom{n}{y} p^y (1-p)^{n-y}, \quad y = 0, \ldots, n \tag{14.1}$$

Then the predictive probability of $\tilde{y}$ successes in a future sample of size $m$ is given by,

$$f(\tilde{y}) = \sum f_B(\tilde{y}|m, p_i) g(p_i) \tag{14.2}$$

The function `pdiscp` in the *LearnBayes* package computes predictive probabilities when $p$ is given a discrete distribution. We give the function the vector, `p`, of proportions, the vector `prior`, of corresponding probabiilities, the future sample size, `m`, and the vector, `ys`, of the numbers of (possible) successes of interest.

```
> p = seq( 0.05, 0.95, by=0.1 )
> prior = c( 2,4,8,8,4,2,1,1,1,1 )
> piror = prior / sum( prior )
> m = 20
> ys = 0:20
> pred = pdiscp( p, prior, m, ys )
> cbind( ys, pred )
      ys       pred
 [1,]  0 0.02808924
 [2,]  1 0.04647102
 [3,]  2 0.05979678
 [4,]  3 0.07613188
 [5,]  4 0.09129575
 [6,]  5 0.10025206
 [7,]  6 0.10094968
 [8,]  7 0.09389857
 [9,]  8 0.08141436
[10,]  9 0.06654497
[11,] 10 0.05199912
[12,] 11 0.03953367
[13,] 12 0.02990315
[14,] 13 0.02314587
[15,] 14 0.01889113
[16,] 15 0.01654018
[17,] 16 0.01538073
[18,] 17 0.01492458
[19,] 18 0.01552235
[20,] 19 0.01679584
[21,] 20 0.01251910
```

As you can see, we can reasonably expect 5 or 6 students to have slept eight hours in a sample of 20.

Suppose we'd instead used a beta prior, then we'd use the integral,

$$f(\tilde{y}) = \int f_B(\tilde{y}|m,p) g(p) dp \tag{14.3}$$

$$= \binom{m}{\tilde{y}} \frac{\text{Beta}(a+\tilde{y}, b+m-\tilde{y})}{\text{Beta}(a,b)}, \quad \tilde{y} = 0, \ldots, m. \tag{14.4}$$

The predictive probabilities are computed using the `pbetap` function. The inputs are the vector `param` of the values for `a` and `b`. and the size, `m`, of the future sample, and the vector of the numbers of the possible successes, `ys`.

```
> a = 3.4
> b = 7.4
> param = c( a, b )
> m = 20
> ys = 0:20
> pred = pbetap( param, m, ys )
> cbind( yx, pred )
      ys         pred
 [1,]  0 1.650355e-02
 [2,]  1 4.250914e-02
 [3,]  2 6.995599e-02
 [4,]  3 9.289238e-02
 [5,]  4 1.079775e-01
 [6,]  5 1.141476e-01
 [7,]  6 1.120140e-01
 [8,]  7 1.032286e-01
 [9,]  8 8.992595e-02
[10,]  9 7.428665e-02
[11,] 10 5.823390e-02
[12,] 11 4.325578e-02
[13,] 12 3.033522e-02
[14,] 13 1.996421e-02
[15,] 14 1.221690e-02
[16,] 15 6.857229e-03
[17,] 16 3.458690e-03
[18,] 17 1.518068e-03
[19,] 18 5.490883e-04
[20,] 19 1.472492e-04
[21,] 20 2.228637e-05
```

Again, we have similar results. We have discussed performing prediction using a discrete and a continuous prior. The last technique involves computing a predictive density for any prior, using simulation. To obtain $\tilde{y}$, we first simulate $p^*$ from $g(p)$, and then simulate $\tilde{y}$ from the binomial distribution $f_B(\tilde{y}|p^*)$.

We'll demonstrate this approach using the Beta$(3.4, 7.4)$ prior. We first simulate 1000 draws from the prior and store the simulated values in `p`.

```
> p = rbeta( 1000, 3.4, 7.4 )
```

Then we sample 20 values for $\tilde{y}$ from `p` using the `rbinom` function.

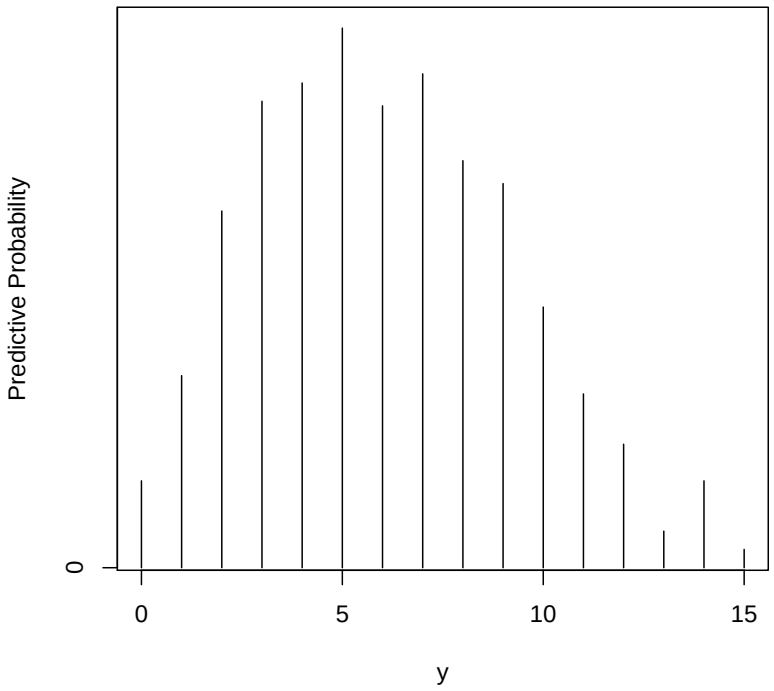
```
> y = rbinom( 1000, 20, p )
```

We then tabulate the simulated draws from $\tilde{y}$ using the `table` command,

```
> table(y)
y
  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15
 19  42  78 102 106 118 101 108  89  84  57  38  27   8  19   4
```

We then save these frequencies, turn them into probabilities, and plot.

```
> freq = table(y)
> ys = c(0:max(y))
> predprob = freq / sum( freq )
> plot( ys, predprob, type='h', xlab='y', ylab='Predictive Probability' )
```


Predictive Probability
0
0
5
10
15
y

# Chapter 15

# Sage

(top)

## 15.1 Installation

Download the proper package from http://www.sagemath.org/. Put the file in your home directory, and then extract it. Once it has been extracted, you can run sage by calling the directory followed by the term sage as in:

```
$ ~/sage-4.6.2-linux-64bit-ubuntu_10.04.1_lts-x86_64-Linux/sage
$
```

Or you can edit the .bashrc file and make an alias. This will allow you to run sage from anywhere by simply typing 'sage'. The .bashrc file is in your home directory; note that you will need to open it with superuser priveliges. First open it using:

```
$ sudo gedit ~/.bashrc
$
```

And then add the following line somewhere, but preferably near the bottom:

```
$ alias sage='~/sage-4.6.2-linux-64bit-ubuntu_10.04.1_lts-x86_64-Linux/sage'
$
```

## 15.2 Access the Shell

You can access the underlying (Bash) shell by beginning a command with a bang (!), just like ipython. The logic behind this is that the sage command line is built on ipython. The great thing about this is that you can move around and not have to exit a sage session.

```
sage: !pwd
<working directory>
```

## 15.3 Simple Math

Everything works pretty much the way you'd expect it to. Exponents use a carat or a double asterix. If you need a decimal apprroximation rather than a fractional expression, use `float()`

```
sage: 8/3
8/3
sage: float( 8/3 )
2.6666666666666665
```

## 15.4 Solving Equations

We can solve equations symbolically using the `solve()` function:

```
sage: x = var('x')
sage: eqn = 4*x + 3 == 5*x - 2
sage: solve( eqn, x )
[x == 5]
```

We can also solve systems of equations using the `solve()` function:

```
sage:  var('a b c')
(a, b, c)
sage: eqn = [ a - 2*b + 4*c == 3, a + 3*b - 2*c == 6, a - 4*b + 3*c == -5 ]
sage: s = solve( eqn, a, b, c ); s
[[a == 1, b == 3, c == 2]]
```

Alternatively, we can solve equations numerically using the `find_root` function:

```
sage: x = var('x')
sage: eqn = 4*x + 3 == 5*x - 2
sage: eqn.find_root( -10, 10 )
5.0
```

## 15.5 Partial Fraction Decomposition

Sage makes these guys ridiculously easy. This operation is often required after applying the Laplace transform to a differential equation, or a system of differential equations, before applying the inverse Laplace transform.

```
sage: var('x')
sage: f = ( x^2 + 2*x + 3 )/( x^3 + 4*x^2 + 5*x + 2 )
sage: f.partial_fraction()
      -2/(x+1) + 2 /(x+1)^2 + 3/(x+2)
```

## 15.6 Laplace Transform

To calculate the Laplace transform of a function in terms of t, you have to declare two symbolic variables: t, for the original function, and s, for the transformed function.

```
sage: t = var('t')
sage: s = var('s')
sage: f = t + t^2 # some function f in terms of t
sage: f.laplace(t,s)
1/s^2 + 2/s^3 # voila, the answer!
sage:
```

## 15.7 Systems of Differential Equations

```
t = var('t')
k = var('k')

x0 = function('x0',t)
x1 = function('x1',t)
x2 = function('x2',t)
x3 = function('x3',t)
x4 = function('x4',t)
x5 = function('x5',t)
x6 = function('x6',t)
x7 = function('x7',t)
x8 = function('x8',t)
x9 = function('x9',t)

de0 = diff(x0,t) + k*x1 == 0
de1 = diff(x1,t) - k*( x0 - x1 ) == 0
de2 = diff(x2,t) - k*( x1 - x2 ) == 0
de3 = diff(x3,t) - k*( x2 - x3 ) == 0
de4 = diff(x4,t) - k*( x3 - x4 ) == 0
de5 = diff(x5,t) - k*( x4 - x5 ) == 0
de6 = diff(x6,t) - k*( x5 - x6 ) == 0
de7 = diff(x7,t) - k*( x6 - x7 ) == 0
de8 = diff(x8,t) - k*( x7 - x8 ) == 0
de9 = diff(x9,t) - k*( x8 - x9 ) == 0

eqns = [ de0, de1, de2, de3, de4, de5, de6, de7, de8, de9 ]
vars = [ x0, x1, x2, x3, x4, x5, x6, x7, x8, x9 ]

sol = desolve_system( eqns, vars, ivar=[k] )
```

## 15.8 Systems of Differential Equations and the Laplace Transform

Suppse we have a set of two springs and two masses connected linearly. This system is described (clearly) by the following equation:

$$\begin{aligned} m_1\ddot{x}_1 + (k_1 + k_2)x_1 - k_2x_2 = 0 \\ m_2\ddot{x}_2 + k_2(x_2 - x_1) = 0 \end{aligned}$$

Let $m_1 = 2$, $m_2 = 1$, $k_1 = 4$, $k_2 = 2$, $x_1(0) = 3$, $\dot{x}_1(0) = 0$, $x_2(0) = 3$, $\dot{x}_2(0) = 0$. Finally, for readability, we make the following substitution: $x = x_1$, $y = x_2$.

We can calculate the Laplace transformation using maxima as follows:

```
sage: de1 = maxima(``2*diff(x(t),t,2) + 6*x(t) - 2*y(t) = 0'')
sage: lde1 = de1.laplace(``t'',``s'')
```

This result of the Laplace transform is the following:

$$\begin{aligned} -2\dot{x}(0) + 2s^2X(s) - 2sx(0) - 2Y(s) + 6X(s) = 0 \\ 2s^2X(s) - 2Y(s) + 6X(s) = 6s \end{aligned}$$

And for the next equation:

```
sage: de2 = maxima(``diff(y(t),t,2) + 2*y(t) - 2*x(t) = 0'')
sage: lde2 = de2.laplace(``t'',``s'')
```

Which produces the following Laplace transform:

$$-\dot{Y}(0) + s^2Y(s) + 2Y(s) - 2X(s) - sy(0) = 0$$
$$-\dot{Y}(0) + s^2Y(s) + 2Y(s) - 2X(s) = 3s$$

To solve, we execute the following:

```
sage: var('s X Y')
sage: eqns = [ ( 2*s^2 + 6 )*X - 2*Y == 6*s, -2*X + ( s^2 + 2 )*Y == 3*s ]
sage: solve( eqns, X, Y )
sage: var('s t')
sage: inverse_laplace( ( 3*s^3 + 9*s )/( s^4 + 5*s^2 +4 ),s,t)
sage: inverse_laplace( ( 3*s^3 + 15*s )/( s^4 + 5*s^2 +4 ),s,t)
```

## 15.9 Linear Algebra

Creation of matrices and matrix multiplication is easy and natural:

```
sage: A = Matrix([[1,2,3],[3,2,1],[1,1,1]])
sage: b = vector([1,1,-4])
sage: b*A
(0, 0, 0)
sage: A*b
(-9, 1, -2)
```

# Chapter 16

# Octave

(top)

## 16.1 Introduction

Octave is a clone of MATLAB. You can run most MATLAB .m files in Octave. (Score.)

## 16.2 3D Plotting

These are the basic commands for producing a 3D surface:

```
% set dimensions
x = 0 : 0.1 : 1 ;
y = 0 : 0.1 : 1 ;
% create mesh
[xx,yy] = meshgrid(x,y) ;
% expression to plot
z = log(xx) + exp(yy) ;
% produce figure
figure, surf( x, y, z ) ;
% change colors
colormap( 'cool' );
% add a title
title( 'log(x) + exp(y)' ) ;
% label the axes
xlabel( 'x' )
ylabel( 'y' )
```

If you instead want to produce a parameterized curve you can do the following:

```
t = 0:0.1:10*pi;
r = linspace (0, 1, numel (t));
z = linspace (0, 1, numel (t));
plot3 (r.*sin(t), r.*cos(t), z);
```

# Chapter 17

# SymPy

(top)

## 17.1 Introduction

SymPy is a Python library for symbolic mathematics, but it can also be used interactively by calling `isympy` through the shell.

## 17.2 Basic Use

We can use SymPy interactively as:

```
$ isympy
Python 2.6.5 console for SymPy 0.6.6

These commands were executed:
>>> from __future__ import division
>>> from sympy import *
>>> x, y, z = symbols('xyz')
>>> k, m, n = symbols('kmn', integer=True)
>>> f, g, h = map(Function, 'fgh')

Documentation can be found at http://sympy.org/

In [1]:
```

This means that `x, y, z` are automatically initialized as real variables, `k, m, n` are initialized as integer variables, and `f, g, h` are initialized as functions. We can then do something like:

```
In [1]: (1/cos(x)).series(x, 0, 10)
Out[1]:
     2      4       6        8
    x    5*x    61*x    277*x
1 + -- + ---- + ----- + ------ + O(x**10)
    2     24     720     8064
```

Or, if we wanted to calculate some integrals we can do the following:

```
In [1]: x = Symbol( 'x' )

In [2]: f = Function( 'f' )

In [3]: lmbda = 0.25
```

```
In [4]: f = lmbda * exp( -lmbda * x )

In [5]: pprint( Integral( f, ( x, 2, 3 ) ) )
3
/
|         -0.25x
| 0.25 e         dx
/
2

In [6]: integrate( f, ( x, 2, 3 ) )
Out[6]: 0.134164106971619

In [7]: pprint( f.integrate( x ) )
       -0.25x
-1.0 e
```

# Chapter 18

# Lisp

(top)

## 18.1 Introduction

Lisp is amazing. (Period.)

## 18.2 Functions

Functions are lambda expressions. Note that all operations in Lisp use Polish notation, formalised by Jan Łukasiewicz.

```
( DEFUN NAME ( ARGUMENTS ) ( BODY ) )
```

```
> ( DEFUN double ( x ) ( + x x ) )
> ( double 2 )
4
```

## 18.3 Quoting

A lot of errors occur through misquoting. Prepending a single quote on something is the same as using the `QUOTE` function. Numbers and quoted expressions evaluate to themselves.

## 18.4 Variables

We can define variables several ways.

```
( SET ( QUOTE NAME ) ( QUOTE VALUE ) )
( SET 'NAME 'VALUE )
( SETQ NAME VALUE )
```

There are no variable types in Lisp. If a function and a variable have the same name, then Lisp figures out which is which automatically. Obviously, variables do not take parameters.

## 18.5 Special Variables

There are several special variables in Lisp. One is `T`, the logical symbol for *True*. The other is `NIL`, the logical symobl for *False*. Incidentally, `NIL` is equivalent to the empty list.

```
> ( EQUAL 0 0 )
T
> ( EQUAL 0 1 )
NIL
> ( EQUAL NIL () )
T
```

## 18.6 Lists

We have several built in functions to deal with lists. The first four are pretty self-explanatory. Suppose we have a list

```
> ( SETQ X ( LIST 1 2 3 4 ) )
( 1 2 3 4 )
> ( FIRST X )
1
> ( SECOND X )
2
> ( THIRD X )
3
> ( REST X )
( 2 3 4 )
```

The next two functions, `CAR` and `CDR`, are equivalent to `FIRST` and `REST` respectively. The terms `CAR` and `CDR` are abbreviations for "Contents of Address portion of Register" and "Contents of Decrement portion of Register" respectively.

```
> ( SETQ X ( LIST 1 2 3 4 ) )
( 1 2 3 4 )
> ( CAR X )
1
> ( CDR X )
( 2 3 4 )
```

We can define lists three ways. The last form, using `CONS` takes two arguments. The first argument is prepended to the second argument and the result is returned as a list.

```
> '( A B C )
(A B C)
> ( LIST 'A 'B 'C )
(A B C)
> ( CONS 'A '( B C ) )
(A B C)
> ( CONS '( A B ) '( C D ) )
((A B) C D)
> ( CONS 'A 'B )
(A . B)
```

# Chapter 19

# Ruby

(top)

## 19.1 Loops

```
for item in collection do
  body
end
```

```
(0...10).each do |index |
  body
end
```

```
while condition do
  body
end
```

```
while condition do body end
```

```
body while condition
```

## 19.2 Control Structures

```
if condition then
  body
end
```

```
if condition then body end
body if condition
```

```
if condition then
  body1
else
  body2
end
```

```
if condition then body1 else body2 end
```

```
condition ? body1 : body2
```

```
if condition1 then
  body1
elsif condition2 then
  body2
else
  body3
end
```

# Chapter 20

# Perl

(top)

## 20.1 Scalar Variables

This is the most basic kind of variable in Perl. These hold numbers or strings, which are interchangeable. You can assign a number to a variable, and then later assign a string to it. Nbd.

```
$p = 'number nine' ; # this is okay
$p = 9 ;             # this is okay, too
```

More interestingly, we can assign numbers as strings, and still do arithmetic with them.

```
$p = '9' ;
$q = 1 ;
$r = $p + $q ; # now $r is 10!
```

By default, all variables are global. Variables in any part of the program can be accessed in any other part. To force locality, we use the *my* operator.

```
my $localScalar = 1.0
```

## 20.2 Loops

```
foreach $item ( @list ) {
   body ;
}
```

```
while (condition ) {
   body ;
}
```

## 20.3 Control Structures

```
if ( condition ) {
   body ;
}
```

```
if ( condition ) {
   body1 ;
} else {
   body2 ;
}
```

## 20.4 Running a Perl Program

Tres simple! If you have a program named `oyster` you do the following:

```
perl oyster
```

If you want to error messages, or you'd like to run the debugger, you can run the following:

```
perl -w oyster
perl -d oyster
```

## 20.5 Arithmetic

Perl uses the same arithmetic operators that C does, except that it includes an exponent operator in the form of a double asterix. Also, note that division acts like float division, not integer division.

```
$x = 1 + 2 ; # add 1 and 2 and store in $a
$x = 1 - 2 ; # subtract 1 and 2 and store in $a
$x = 1 * 2 ; # multiply 1 and 2 and store in $a
$x = 1 / 2 ; # divide 1 and 2 to get 0.5, not 0, and store in $a
$x = 5 % 2 ; # take remainder of 5 divided by 2 and store in $a
$x = 2**3 ;  # raise 2 to the 3rd power and store in $a
```

We also have all the familiar pre- and post-increment operators from C/C++.

## 20.6 String Operations

We can concatenate strings with a dot, and repeat strings using an `x`.

```
$a = $b . $c ;  # concatenate $b and $c
$a = $b x $c ;  # string $b repeated $c times
```

An example. Note that single quotes print doll hair signs literally, whereas double quotes evaluate the variables. This is called interpolation.

```
$a = 'oysters' ;
$b = 'other mulloscs' ;
print $a.' and '.$b ; # => oysters and other mulloscs
print '$a and $b' ;   # => $a and $b
print ``$a and $b'' ; # => oysters and other mulloscs
```

## 20.7 Arrays

Arrays hold a list of scalars, and a re prefixed by a  sigil.

```
@rumpshaker = ( ``zooma'', ``zoom'', ``zoom'', ``zoom'', ``boom'' ``boom'' ) ;
```

However, to access an array element, we must use a doll hair sigil.

```
$rumpshaker[0] ;  # => ``zooma'' we use $ because this element is a scalar
```

# Chapter 21

# Processing

(top)

## 21.1 Draw a Window

Positions in graphics are specifed by width, measured from the left margin, and height measured down from the top margin.

```
int width = 480 ;
int height = 120 ;
size( width, height ) ;
```

## 21.2 Draw Simple Shapes

These are all pretty self explanatory.

```
point( x1, y1 ) ;
line( x1, y1, x2, y2 ) ;
triangle( x1, y1, x2, y2, x3, y3 ) ;
quad( x1, y1, x2, y2, x3, y3, x4, y4 ) ;
rect( x, y, width, height ) ;
ellipse( x, y, width, height ) ;
```

`arc()` is a little bit more complicated, `start` and `stop` are measured in radians, and travel clockwise. There is a `radians()` function for converting degrees to radians.

```
arc( x, y, width, height, start, stop ) ;
```

## 21.3 Shape Properties

The functions `smooth()` and `noSmooth()` turn on and off antialiasing. The function `strokeWeight()` controls the width of the edges, default is one pixel. The function `strokeJoin()` takes the values `ROUND` or `BEVEL`. The function `strokeCap()` takes the values `SQUARE` or `ROUND`.

## 21.4 Coloring

The function `background()`, `fill()`, and `stroke()` act differently based on the number of parameters.

1. Greyscale
2. Greyscale, with alpha channel
3. Color (RGB)
4. Color (RGB), with alpha channel

The greyscale ranges from $[0, 255]$, with 0 being black, and 255 being white. The alpha channel ranges from $[0, 255]$, with 0 begin totally transparent, and 255 being totally opaque.

The functions `noStroke()` and `noFill()` act predictably.

## 21.5 Arbitrary Shapes

Shapes may be described by a list of vertices using the `beginShape()` and `endShape()` commands. The `CLOSE` parameter in `endShape()` closes the shape. The syntax is as follows:

```
beginShape() ;
vertex(x1,y1) ;
vertex(x2,y2) ;
vertex(x3,y3) ;
endShape(CLOSE) ;
```

# Chapter 22

# MySQL

(top)

## 22.1 Introduction

A database is a set of tables. A table is a set of *records* and *fields*. Tables in a database can be related to each other through *keys*.

## 22.2 Getting Started as `root`

First, we assign a password to the administrator, `root`:

```
$ mysqladmin -u root password ********
```

Here, `********` is the new password for the administrator.

## 22.3 Logging In

To log in we use:

```
$ mysql -u root -p
```

To log in and use a particular database, do this:

```
$ mysql -u uname -p dbname
```

## 22.4 Adding a User

Log in as root, and use the command,

```
$ mysql -u root -p
mysql> grant all privileges on *.* to 'newusr'@'localhost' identified by 'password' ;
```

## 22.5 Getting HELP

This should bring you to the MySQL shell. You can get help by typing:

```
mysql> HELP
mysql> \h
```

## 22.6 LOAD a Database

This database is from the book *Learning SQL*, by Alan Beaulieu. Download the database file `learning_sql.sql` from `http://www.oreilly.com/catalog/learningsql`. Then execute the following at the SQL command line:

```
mysql> CREATE DATABASE bank ;
mysql> USE bank ;
mysql> SOURCE /path/to/learning_sql.sql ;
```

## 22.7 SHOWing Databases and Tables

You can get a list of available databases by:

```
mysql> SHOW DATABASES;
```

You can use a particular database with:

```
mysql> USE database;
```

Given a particular databse, you can view a list of its tables with:

```
mysql> SHOW TABLES;
```

## 22.8 DESCRIBE a Table

You can view a list of a table's fields and their descriptions with:

```
mysql> DESCRIBE table;
```

## 22.9 Using SELECT

You can view an entire table using:

```
mysql> SELECT * FROM table;
```

If we want to view specific fields, you can do:

```
mysql> SELECT field1, field2 FROM table;
```

## 22.10 `CREATE` a Database

First, you create a database using:

```
mysql> CREATE DATABASE dbname;
mysql> USE dbname;
```

Then you create a tables by specifying its fields and their data types. Below we've created a table for holding the first and last names of a set of people, with a primary key.

```
mysql> CREATE TABLE people (
    -> id INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
    -> first VARCHAR(25) NULL,
    -> last VARCHAR(25) NULL,
    -> PRIMARY KEY(id),
    -> );
```

## 22.11 `DROP`ping Items

If you need to delete a database or a table, use `DROP` as:

```
mysql> DROP database_name ;
mysql> DROP table_name
```

## 22.12 Inserting, Updating, and Deleting Records

We can add records by:

```
mysql> INSERT INTO table ( field1, field2 ) VALUES ( "value1", "value2" );
```

For example, for the people table above we'd do something like:

```
mysql> INSERT INTO people ( first, last ) VALUES ( "Walker", "Percy" );
```

To update a record we use:

```
mysql> UPDATE table SET field1=value1 WHERE condition-stmt;
```

The `condition-stmt` after `WHERE` can be the primary key of a particular record, or something more general. We can delete using:

```
mysql> DELETE FROM table VALUES condition-stmt;
```

Again, the `condition-stmt` can be something specific, or general. You can use `AND` or `OR` statements, or whatever.

## 22.13 Produce a Database from a Script

At the command line, enter:

```
$ mysql -u root -p < script.sql
```

`script.sql` should look like:

```
/* --script.sql-- */
CREATE DATABASE db ;
USE db ;
CREATE TABLE tbl (
  id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
  title VARCHAR(20) NOT NULL,
  summary TINYTEXT NOT NULL,
  body TEXT NOT NULL,
  PRIMARY KEY(ID)
);
```

We can write a bash script to do this for us as:

```
#!/bin/bash

mysql --user=root --password=pwstring < script.sql
exit
```

Alternatively, we could use Python script:

```
#!/usr/bin/env python

import os

os.system('mysql --user=root --password=pwstring < script.sql')
os.system('exit')
```

## 22.14 Exiting

Use `QUIT ;` or `EXIT ;`

# Chapter 23

# Probability

(top)

## 23.1 Sample Space, Events, and Partitions

The *sample space* of an experiment is the set of all possible experimental outcomes. We usually use $\mathcal{S}$ or $\Omega$ to denote the sample space. An *event*, $E \subseteq \Omega$, is a subset of the sample space. A *partition* is a set of mutually exclusive events that cover the sample space.

## 23.2 Probability

The probabilities, $p_i$, for a set of outcomes, $\Omega = \{E_i\}$, is a set of values such that

$$0 \leq p_i \leq 1 \tag{23.1}$$

$$\sum_i p_i = 1 \tag{23.2}$$

## 23.3 Complement

Let $E$ be an event. Then the complement of $E$ is defined as $E' = E^C = \bar{E} = \Omega \backslash E$. We have that

$$\Pr(E) + \Pr(E') = 1 \tag{23.3}$$

$$\Pr(E') = 1 - \Pr(E) \tag{23.4}$$

$$\Pr(E \cup E') = \Pr(\Omega) = 1 \tag{23.5}$$

$$\Pr(E \cap E') = \Pr(\varnothing) = 0 \tag{23.6}$$

Note that we also have the following two identities regarding complements

$$(E \cap F)' = E' \cup F' \qquad (E \cup F)' = E' \cap F' \tag{23.7}$$

## 23.4 Unions of Events

Let $E, F \subseteq \Omega$, then

$$\Pr(E \cup F) = \Pr(E) + \Pr(F) - \Pr(E \cap F) \tag{23.8}$$

$$\Pr(E \cup F \cup G) = \Pr(E) + \Pr(F) + \Pr(G) - \big\{\Pr(E \cap F) + \Pr(E \cap G) + \Pr(F \cap G)\big\} + \Pr(E \cap F \cap G) \tag{23.9}$$

## 23.5 Combinatorics

The standard definition of the combination of $r$ elements out of $n$ element is as follows:

$$\binom{n}{r} = {}_rC_n = \frac{n!}{r!(n-r)!} = \frac{{}_rP_n}{r!} = \binom{n}{n-r} \tag{23.10}$$

We can think of this alternatively as the number of way to split $n$ elements into two groups of size $r$ and $n$–$r$. We can then generalize this into more than n groups. If we want to split n elements into $k$ groups of size $r_1, r_2, \ldots, r_k$, where $r_1 + r_2 + \cdots + r_k = n$, then we have

$$\binom{n}{r_1, r_2, \ldots, r_k} = \frac{n!}{r_1! r_2! \cdots r_k!} \tag{23.11}$$

A useful application is determining the number of ways you can choose $m$ elements from a set $M$, and $n$ elements from another set $N$. The formula is given below and is easily extensible to greater numbers of disjoint sets.

$$\frac{\binom{M}{m}\binom{N}{n}}{\binom{M+N}{m+n}} \tag{23.12}$$

## 23.6 Marginal Probability

Suppose $p(x, y)$ and $f(x, y)$ are joint mass and density functions. The marginal distribution of $X$ can be obtained by summing or integrating over $Y$.

$$p_X(x) = \sum_y p(x, y) \tag{23.13}$$

$$f_X(x) = \int_y p(x, y) dy \tag{23.14}$$

## 23.7 Conditional Probability

$$\Pr(E|F) = \frac{\Pr(E \cap F)}{\Pr(F)} \tag{23.15}$$

## 23.8 Law of Total Probability

From the observation that for any two events, $E$ and $F$, we have

$$\Pr(E) = \Pr(E \cap F) + \Pr(E \cap F') \tag{23.16}$$

We can make the following generalization about an event, $E$, and a partition, $F_i$

$$\Pr(E) = \sum \Pr(E \cap F_i) = \sum \Pr(E|F_i) \Pr(F_i) \tag{23.17}$$

## 23.9 Bayes' Theorem

Combining the conditional probability and the law of total probability gives us a more general form of Bayes' theorem.

$$\Pr(F|E) = \frac{\Pr(E|F) \Pr(F)}{\Pr(E)} = \frac{\Pr(E|F) \Pr(F)}{\Pr(E \cap F) + \Pr(E \cap F')} \tag{23.18}$$

$$\Pr(F|E) = \frac{\Pr(E|F) \Pr(F)}{\Pr(E|F) \Pr(F) + \Pr(E|F') \Pr(F')} \tag{23.19}$$

We may generalixe this concept to a given member of a partition, $F_i$, and an event, $E$

$$\Pr(F_i|E) = \frac{\Pr(E|F_i) \Pr(F_i)}{\sum \Pr(E|F_i) \Pr(F_i)} \tag{23.20}$$

$$\Pr(\text{ parameter } | \text{ data }) = \frac{\Pr(\text{ data } | \text{ parameter }) \Pr(\text{ parameter })}{\Pr(\text{ data })}$$

| | |
|---|---|
| Prior | Pr( parameter ) |
| Posterior | Pr( parameter \| data ) |
| Likelihood | Pr( data \| parameter ) |

The *posterior* is the probability of finding a certain value for the parameter, given the data. This gives us an explicit statement about the probability of an event, without reference to any unlimited sequence of trials.

The *likelihood* links the parameter to specific data points in an actual experiment. It tells us what data we can expect to observe, given different values of the parameter.

The *prior* captures our prior belief of finding a certain outcome–our prior belief that the parameter has a certain value.

### 23.9.1 More Discussion

Frequentist statistics gives you a confidence interval and says, if you repeat this experiment ad nauseum, then 95% of the time, the outcome of your experiment will fall into this confidence interval. This is not to say that your experiment has a 95% chance of obtaining an outcome in the confidence interval on any given trial. Bayesian statistics allow us to say that there is a 95% chance that the outcome will fall into a given interval, or that a hypothesis has a given probability.

The advantages of Bayesian estimation are that we can incorporate prior beliefs about a particular hypothesis. The frequentist approach says that conclusions should be independent of previous work, whereas the Bayesian approach says that conlcusions should incorporate previous work.

The name of the game is to compare the relative probability of competing hypothesis, taking into account prior information. Likelihoods from the frequentist approach work the same way, but without the benefit of prior knowledge.

The disadvantages are that the prior probability is subjective. You can get very different posterior distributions by changing what parameters have uninformative priors.

The frequentist approach is easier and more intuitive. It allows you to falsify hypotheses, albeit using an arbitrary threshold. Under Bayesian statistics there is no such thing as falsification, just relative degrees of belief.

### 23.9.2 Bayesian Squirrels

Hilborn and Mangel describe the Bayesian squirrel. The squirrel has burried her acorns in one of two patches, but she can't remember which one it is. There are two hypotheses. Hypothesis 1 is that the nuts are in patch 1, and hypothesis 2 is that the nuts are in patch 2. The squirrel is pretty sure that the nuts are in patch 1; she's willing to say that there's an 80% that it's in patch 1. She also knows she's really good at hiding nuts, so there's a 20% chance that she'll find her nuts given that she's in the right patch, and a 0% chance if she's in the wrong patch.

Thus, the patches (initially) have prior probabilities of 0.8 and 0.2, respectively. Suppose she's in the right patch and she doesn't find any food. What's the probability she's in the right patch, given that she didn't find anything?

$$\Pr(\text{Food in patch 1}|\text{Find no food}) = \frac{\Pr(\text{Find no food}|\text{Food in patch 1})\Pr(\text{Food in patch 1 }))}{\Pr(\text{Find no food})}$$

The likelihood, Pr(Find no food|Food in patch 1), is 0.8, because she's good at hiding food. The prior, Pr($Foodinpatch1$), is 0.8. The denominator is a sum. There are two ways to not find food. Either she's in the right patch and down on her luck, or the food is in patch 2. The denominator, then, is $0.8 * 0.8 + 0.2$. This gives us:

$$\Pr(\text{Food in patch1}|\text{Find no food}) = 0.76 = \frac{0.80 * 0.80}{0.80 * 0.80 + 0.20}$$

Suppose she looks in patch 1 again? Before she starts looking again, she should use her old posterior probability as her new prior, thus updating her belief about where her food is.

The first time she looks in patch 1 and doesn't find food, the probability that she's look in the right patch is 0.7619, if she looks in the same patch and still does not find food, the probability drops to 0.7191. Here is a table:

| Trial | Probability food is in patch 1 |
|---|---|
| 1 | 0.7619 |
| 2 | 0.7191 |
| 3 | 0.6719 |
| 4 | 0.6209 |
| 5 | 0.5672 |
| 6 | 0.5118 |

After six tries, the probability that the food is in patch 1 has dropped below 50%, so it would make more sense to try patch 2. What if she was better at locating food? What if she found food 30% of the time that she looked for it and it was present? Then we'd have to change the likelihood and we'd have the following table:

| Trial | Probability food is in patch 1 |
|---|---|
| 1 | 0.7368 |
| 2 | 0.6621 |
| 3 | 0.5784 |
| 4 | 0.4899 |
| 5 | 0.4020 |
| 6 | 0.3200 |

Thus, we see that if she's better at finding food, then she can decide to leave patch 1 sooner after not finding any food there, and this agrees with our intuition.

### 23.9.3 A Unicorn Example

Suppose we spot 4 unicorns in $20km^2$. We can use that information to determine the probability given different unicorn densities, i.e., Pr(4 Unicorns|Unicorn density). Really, though, we'd like to determine the unicorn density given the fact that we saw 4 unicorns, i.e., Pr(Unicorn density|4 Unicorns). We can use Bayes theorem to write:

$$\Pr(\text{Density}|\text{4 Unicorns}) = \frac{\Pr(\text{Density})\Pr(\text{4 Unicorns}|\text{Density})}{\Pr(\text{4 Unicorns})}$$

## 23.10 Random Variable (rv)

Frequently, when an experiment is performed, we are interested mainly in some function of the outcome as opposed to the actual outcome itself. For instance, in tossing dice, we are often interested in the sum of the two dice and are not really concerned with the seperate values of each die. A real-valued function defined on the sample space is known as a random variable, rv. Specifically, a random variable is any function that takes the sample space as its domain and the reals as its range. Suppose our experiment consists of tossing two dice. If we let X denote the sum of the dice, then $X$ is a random variable taking on the values $2, 3, \ldots, 12$. And we have the following probabilities

$$\begin{array}{rcccl}
\Pr(X=2) & = & \Pr\{(1,1)\} & = & 1/36 \\
\Pr(X=3) & = & \Pr\{(1,2),(2,1)\} & = & 2/36 \\
\Pr(X=4) & = & \Pr\{(1,3),(2,2),(3,1)\} & = & 3/36 \\
\Pr(X=10) & = & \Pr\{(4,6),(5,5),(6,4)\} & = & 3/36 \\
\Pr(X=11) & = & \Pr\{(5,6),(6,5)\} & = & 2/36 \\
\Pr(X=12) & = & \Pr\{(6,6)\} & = & 1/36
\end{array}$$

## 23.11 Discrete Random Variable

A random variable whose range is either finite or countably infinite.

## 23.12 Probability Mass Function (pmf)

For any discrete random variable $X$, we define the probability mass function $p(x)$ of $X$ by:

$$p(x) = \Pr(X = x) \tag{23.21}$$

We have the following condition:

$$\sum_{i=-\infty}^{\infty} p(x_i) = 1 \tag{23.22}$$

## 23.13 Probability Density Function (pdf)

For any continuous random variable X, we define the probability density function $f(x)$ of X by:

$$f(x) = \Pr(X \leq x) \tag{23.23}$$

We have the following condition

$$\int_{-\infty}^{\infty} f(x)dx = 1 \tag{23.24}$$

## 23.14 Cumulative Distribution Function (cdf)

For any random variable $X$, the function $F$, defined below, is called the cumulative distribution function (cdf), or, more simply, the distribution function, of $X$.

$$F(x) = \Pr(X \leq x) = \sum_{y \leq x} p(y), -\infty < x < \infty \tag{23.25}$$

## 23.15 Parameter

Suppose $p(x)$ depends on some variable, and that each different value of this variable determines a different probability distribution. Then this variable is called a parameter of the distribution. The collection of all distributions for different values of this parameter is called a family of distributions.

## 23.16 Discrete Probability Distributions

### 23.16.1 Bernoulli Trials

Consider a trail, or an experiment, that results in either a success or a failure. If we let $X = 1$ when the outcome is a success, with probability $p$, and $X = 0$ when the outcome is a failure, with probability $1 - p$, then the probability mass function (pmf) is given by

$$\text{Bernoulli}(p) = \begin{cases} \Pr(X = 1) = p \\ \Pr(X = 0) = 1 - p \end{cases} \tag{23.26}$$

### 23.16.2 Binomial

Now consider a sequence of $n$ Bernoulli trials where successes have a probability of $p$, and failures have a probability of $1 - p$, as before. Let $X$ be the number of successes in $n$ Bernoulli trials.

$$X \sim \text{Binomial}(n, p) \tag{23.27}$$

$$p(x) = \binom{n}{x} p^x (1-p)^{n-x} \tag{23.28}$$

$$\mathbb{E}(X) = np \tag{23.29}$$

$$\mathbb{V}(X) = np(1-p) \tag{23.30}$$

### 23.16.3 Geometric

Let $X$ be the number of Bernoulli trials required to achieve one success. Then $X = \{0, 1, 2, \dots\}$.

$$X \sim \text{Geometric}(p) \tag{23.31}$$

$$p(x) = (1-p)^x p \tag{23.32}$$

$$\mathbb{E}(X) = \frac{1-p}{p} \tag{23.33}$$

$$\mathbb{V}(X) = \frac{1-p}{p^2} \tag{23.34}$$

### 23.16.4 Negative Binomial

Let $X$ be the number of Bernoulli trials, with probability of success, $p$, required to achieve $r$ failures. Then $X = \{r, r+1, \dots\}$.

$$X \sim \text{NegativeBinomial}(r, p) \tag{23.35}$$

$$p(x) = \binom{x+r-1}{x}(1-p)^r p^k \tag{23.36}$$

$$\mathbb{E}(X) = \frac{pr}{1-p} \tag{23.37}$$

$$\mathbb{V}(X) = \frac{pr}{(1-p)^2} \tag{23.38}$$

### 23.16.5 Hypergeometric

Let $X$ be the number successes in $n$ draws–without replacement–from a finite population of size $N$ with $M$ total successes.

$$X \sim \text{Hypergeom}(N, M, n) \tag{23.39}$$

$$p(x) = \frac{\binom{M}{x}\binom{N-M}{n-x}}{\binom{N}{n}} \tag{23.40}$$

$$\mathbb{E}(X) = \frac{nM}{N} = \mu \tag{23.41}$$

$$\mathbb{V}(X) = \frac{nM}{N}\left(1 - \frac{M}{N}\right)\left(\frac{N-n}{N-1}\right) \tag{23.42}$$

$$= \mu\left(1 - \frac{\mu}{n}\right)\left(\frac{N-n}{N-1}\right) \tag{23.43}$$

### 23.16.6 Poisson

Let $X$ be the number of events occuring during a fixed interval. Then $X = \{0, 1, 2, \dots\}$. The expected number of occurences in a given interval is $\lambda$. The number of occurences in one interval is independent of the number of occurences in another interval.

$$X \sim \text{Poisson}(x; \lambda) \tag{23.44}$$

$$p(x) = e^{-\lambda}\frac{\lambda^x}{x!} \tag{23.45}$$

$$\mathbb{E}(X) = \lambda = \mathbb{V}(X) \tag{23.46}$$

The Poisson random variable approximates the Binomial when $n$ is very large, and $p$ is very small. It produces a good approximation when $n \geq 20$ and $np \leq 1$. It produces an excellent approximation when $n \geq 100$ and $np \leq 10$.

## 23.17 Continuous Probability Distributions

### 23.17.1 Normal

The normal distribution is important because of the Central Limit Theorem.

**Central Limit Theorem.** *Let $Y$ be the sum of $N$ independent identically distributed random variables, $X_i$, with expectation, $\mu$, and variance, $\sigma^2$, As $N$ gets very large, the distribution of $Y$ approaches a normal distribution with mean, $N\mu$, and variance, $N\sigma^2$.*

Thus, any random variable we can view as the sum of iid elements should take on a normal distribution. Similarly, the log of any random variable that we can view as the product of a large number of iid elements should take on a normal distribution, becasue the log of a product is equal to the sum of the logs.

$$X \sim \text{Normal}(\mu, \sigma^2) \tag{23.47}$$

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{1}{2}(x-\mu)^2/\sigma^2} \tag{23.48}$$

We can also view the normal distribution as a continuous analogue of the binomial distribution. For large $N$, the binomial with $N$ trials, each with probability $p$ is approximated by the normal distribution with mean $Np$ and variance $Np(1-p)$.

### 23.17.2 Exponential

Let $X$ be the waiting time between events in a Poisson process; the waiting time until an event that occurs with a hazard rate, $\lambda$. We can also view the exponential as the continuous analogue of the geometric distribution.

$$X \sim \text{Exponential}(\lambda) \tag{23.49}$$

$$f(x) = \begin{cases} \lambda e^{-\lambda x} & x \geq 0 \\ 0 & x < 0 \end{cases} \tag{23.50}$$

$$F(x) = \begin{cases} 1 - e^{-\lambda x} & x \geq 0 \\ 0 & x < 0 \end{cases} \tag{23.51}$$

$$\mathbb{E}(X) = \frac{1}{\lambda} \tag{23.52}$$

$$\mathbb{V}(X) = \frac{1}{\lambda^2} \tag{23.53}$$

$$median(X) = \frac{\ln 2}{\lambda} < \mathbb{E}(X) \tag{23.54}$$

### 23.17.3 Gamma

The gamma distribution is the sum of $k$ independent identically distributed (iid) exponential random variables with rate parameter, $\lambda$; it describes the waiting time until the $k^{th}$ failure. The gamma distribution is continuous analogue to the negative binomial distribution.

The gamma distribution is the conjugate prior distribution of the Poisson, normal, log-normal, exponential, gamma, inverse gamma, and Pareto distributions.

$$X \sim \text{Gamma}(k, \lambda) \tag{23.55}$$

$$f(x) = \frac{\lambda}{(k-1)!}(\lambda x)^{k-1} e^{-\lambda x} \tag{23.56}$$

If $k$ is a positive integer, then the distribution is an Erlang distribution, and

$$F(x) = 1 - e^{-\lambda x} \sum_{i=0}^{k-1} \frac{(\lambda x)^i}{i!} \tag{23.57}$$

$$\mathbb{E}(X) = \frac{k}{\lambda} \tag{23.58}$$

$$\mathbb{V}(X) = \frac{k}{\lambda^2} \tag{23.59}$$

### 23.17.4 Chi-Square

When the gamma distribution has $\alpha = \frac{r}{2}$, where $r$ is a positive integer, and $\beta = 2$, we have the Chi-Square distribution.

$$f(x) = \frac{1}{\Gamma(\frac{r}{2})2^{r/2}} x^{\frac{r}{2}-1} e^{\frac{-x}{2}}, \quad 0 < x < \infty \tag{23.60}$$

$$M(t) = (1-2t)^{\frac{-r}{2}}, \quad t < \frac{1}{2} \tag{23.61}$$

### 23.17.5 Beta

In Bayesian inference, the beta distribution is the conjugate prior probability distribution of the Bernoulli, binomial, negative binomial, and geometric distributions. The beta distribution is the suitable model for the random behavior of percentages and proportions. The beta is defined on the interval $[0, 1]$, and has two shape parameters, $\alpha$ and $\beta$.

$$f(x) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha,\beta)} \tag{23.62}$$

$$\mathbb{E}(X) = \frac{\alpha}{\alpha+\beta} \tag{23.63}$$

$$\mathbb{V}(X) = \frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)} \tag{23.64}$$

### 23.17.6 Dirichlet

The Dirchelet is the multivariate generalization of the beta distribution. It is parameterizaed by a vector of positive reals.

The Dirichlet distribution is the conjugate prior distribution of the categorical and multinomial distributions.

$$f(x;\alpha) = \frac{1}{B(\alpha)}\prod_{i=1}^{K} x_i^{a_i-1} \tag{23.65}$$

$$B(\alpha) = \frac{\prod_{i=1}^{K}\Gamma(\alpha_i)}{\Gamma(\sum_{i=1}^{K}\alpha_i)} \tag{23.66}$$

$$\alpha = (alpha_1, \ldots, \alpha K) \tag{23.67}$$

### 23.17.7 z-Distribution

For a normal distribution, about 68% of the data will fall within on standard deviation, about 95% within two standard deviations and over 99% within three standard deviations. The formula to calculate a z-score for a population with known mean and standard deviation is:

$$z = \frac{x-\mu}{\sigma} \tag{23.68}$$

The advantage of z-scores is that they allow us to compare members of different populations with different means and standard deviations.

## 23.18 Relationships Between Discrete and Continuous R.V.

| DISCRETE | CONTINUOUS |
|---|---|
| Binomial($n$,$p$)<br>No. of successes in $n$ trials | Poisson($\lambda$), Normal($\mu$,$\sigma^2$) |
| Geometric($p$)<br>No. trials before first success | Exponential($\lambda$)<br>Waiting time before first success |
| NegativeBinomial($r$,$p$)<br>No. trials before $r^{th}$ failure | Gamma($k$,$\lambda$)<br>Waiting time until $k^{th}$ failure |

# Chapter 24

# Statistics

(top)

## 24.1 Expected Value

The basic definition, for discrete and continuous variables is given as follows,

$$\mathbb{E}[X] = \sum_i x_i p(x_i) = \int_{-\infty}^{\infty} x f(x) dx = \mu \tag{24.1}$$

For any real valued function $g$ we have,

$$\mathbb{E}[g(X)] = \sum_i g(x_i) p(x_i) = \int_{-\infty}^{\infty} g(x) f(x) dx \tag{24.2}$$

As a simple consequence of the above, we see that expectation enjoys linearity

$$\mathbb{E}[aX + b] = a\mathbb{E}[X] + b \tag{24.3}$$

The expected value of a r.v. $X$, $\mathbb{E}[X]$, is also referred to as the mean or the first moment of $X$. The quantity $\mathbb{E}[X^n]$, $n > 0$, is called the $n^{th}$ moment of $X$.

$$\mathbb{E}[X^n] = \sum x^n p(x) = \int x^n f(x) dx \tag{24.4}$$

## 24.2 Variance

$$\mathbb{V}(X) = \frac{1}{N} \sum_{i=1}^{N} (x_i - \mu)^2 = \mathbb{E}[(X - \mu)^2] = \mathbb{E}[X^2] - \mathbb{E}[X]^2 = \sigma^2 \tag{24.5}$$

Analogous to the mean being the center of gravity of a distribution of mass, the variance represents, in the terminology of mechanics, the moment of inertia. The square root of $\mathbb{V}(X)$ is the standard deviation of $X$.

## 24.3 Coefficient of Variance

Variance is sensitive to scale. This allows us to measure the variance of things with different measures and different means. It does not perform well when the means are near zero.

$$c.v. = \frac{\sigma^2}{\mu} \tag{24.6}$$

## 24.4 Covariance

$$cov(X, Y) = \mathbb{E}[(X - \mu_X)(Y - \mu_Y)] = \mathbb{E}[XY] - \mathbb{E}[X]\mathbb{E}[Y] \tag{24.7}$$

## 24.5 Correlation

### 24.5.1 Pearson Correlation Coefficient, ($r$)

A number between -1 and 1 that describes the linear relationship between pairs of quantitative variables. The sign of $r$ determines describes whether the relationship is positive or negative. The value of $r$ describes the strength of the linear relationship. Below we have the formal and computational formulas for the Pearson correlation coefficent. [Witte]

$$\rho_{X,Y} = \frac{cov(X, Y)}{\sigma_X \sigma_Y} = \frac{N \cdot \sum XY - \sum X \cdot \sum Y}{\sqrt{N \cdot \sum X^2 - (\sum X)^2}\sqrt{N \cdot \sum Y^2 - (\sum Y)^2}} \tag{24.8}$$

### 24.5.2 Squared Pearson Correlation Coefficent, ($r^2$)

This statistic describes the proportion of variance in the sample explained by the pearson correlation coefficent. This is the statistic that should be used to say meaningful things about the strength of correlations between different variables. Note that a correlation of 0.5 between two variables only explains 25% of variance between the two variables in the population. Thus, 75% of the variance is left unexplained, or is unpredictable, from the relationship of corr(X,Y) = 0.5. [Witte]

[Witte] Robert S. Witte, Statistics, 4th ed.

## 24.6 Binary Classification

| | A | A' |
|---|---|---|
| A | TP | FN |
| A' | FP | TN |

Accuracy measures the probability of correct classification.

$$\text{accuracy} = \frac{TP + TN}{TP + FP + TN + FN} \tag{24.9}$$

Precision measures the *positive predictive value*. Pr($A$|identified as $A$)

$$\text{precision} = \frac{TP}{TP + FP} \tag{24.10}$$

Sensitivity, or recall, measures the ability to detect positive results. Pr(identified as $A|A$)

$$\text{recall} = \text{sensitivity} = \frac{TP}{TP + FN} \tag{24.11}$$

Specificity measures the model's ability to detect negative results. Pr(identified as $A'|A'$)

$$\text{specificity} = \frac{TN}{TN + FP} \tag{24.12}$$

A general measure of classification quality is the *F measure*.

$$F_\beta = (1 + \beta^2) \cdot \frac{\text{precision} \cdot \text{sensitivity}}{\beta^2 \cdot \text{precision} + \text{sensitivity}} \tag{24.13}$$

The $F_1$ measure weights precision and recall evenly. The $F_2$ measure weights recall more. The $F_{0.5}$ measure weights precision more.

*Matthews Correlation Coefficient* is regarded as a more balanced measure of binary classification, even when classes are of very different sizes. It returns a value in the range $(-1, 1)$ where 1 represent perfect prediction, $-1$ represents perfect inverse prediction, and 0 represents random prediction.

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{24.14}$$

The denominator of this ratio can be set to 1 if any of the sums evaluate to zero. Note that $MCC$ is related to the chi-square statistic for a 2x2 table with a total of $n$ observations by the expression,

$$|MCC| = \sqrt{\frac{\chi^2}{n}} \tag{24.15}$$

## 24.7 Inferential Statistics

We use the parameters $\mu$ and $\sigma^2$ to represent the population mean and variance. We use statistics $\bar{x}$ and $s$ to represent the sample mean and sample variance. Parameters describe populations, while statistics describe samples. By convention we use $N$ to denote the population size and $n$ to denote sample size. The Central Limit Theorem states that the distributions of means of samples of a sufficient size is normally distributed, even when the population from which the samples were drawn is not normally distributed.

## 24.8 Hypothesis Testing

### 24.8.1 Confidence Interval

A 95% confidence interval is a range of numbers. If an experiment is repeated over and over, then this confidence interval would contain the result of the experiment 95% of the time. *This is not the same as saying that values in that there is a 95% chance that the experiment will return a value in this interval.*

### 24.8.2 *p*-value

The probability of unobserved data being more extreme than the observed data, if the hypothesis were true. So, a small $p$-value suggests that there is a small chance that the null hypothesis is true.

### 24.8.3 General Procedure

If you are trying to prove something, the null hypothesis is the negation, or the hypothesis that there is no change, and the alternate hypothesis is the affirmation, or the hypothesis that a change has occurred. After testing we make one of two decisions: to reject the null hypothesis and accept the alternate hypothesis or to fail to reject the null hypothesis. Statistical testing involves establishing a significance level. If the $p$-value returned by the experiment is less than the significance level, then we reject the null hypothesis, otherwise we fail to reject the null-hypothesis. In practice, the significance level is commonly set at 0.05. This is

equivalent to one failure in twenty attempts. Note that failure to reject the null hypothesis does not mean that we have proven it to be true, only that the experiment did not find sufficient evidence to reject it.

| | $H_0$ true | $H_a$ true |
|---|---|---|
| Fail to Reject $H_0$ | Correct | Type II error |
| Reject $H_0$ | Type I error | Correct |

The level of acceptability for Type I error is conventionally set at $\alpha = 0.05$. This means that we accept the 5% probability of a Type I error. A Type II error is less drastic. It is considered a less serious error to fail to make an inference that is true than to make an inference that is false. The level of acceptability for Type II error is conventionally set at $\beta = 0.1$. This means that we accept the 10% probability that $H_a$ was true, but that we failed to reject $H_0$ . The reciprocal of a Type II error is power, defined as $1 - \beta$.

## 24.9 Pearson's Chi-square Test

Suitable for data in which

1. observations are independent
2. categories are mutually exclusive and exhaustive
3. no cell has an expected value less than one
4. no more than 20% of the cells have an expected value less than 5

$$E_{ij} = \frac{i^{th}\text{row total} \times j^{th}\text{column total}}{\text{grand total}} \tag{24.16}$$

$$\chi^2 = \frac{\sum O_{ij} - E_{ij}}{E_{ij}} \tag{24.17}$$

The degrees of freedom are given by $d.f. = (\text{rows} - 1)(\text{cols} - 1)$.

## 24.10 Fisher's Exact Test

This is a nonparametric test often substituted for the chi-square test with small or sparsely distributeddata sets.

*Null Hypothesis*: No relationship between the variables.
*Alternative*: There is a relationship between the variables.

If $p > 0.05$, go with the *Null Hypothesis*, otherwise go with the *Alternative*.

$$p = \frac{r_1!r_2!c_1!c_2!}{n!a!b!c!d!} \tag{24.18}$$

## 24.11 Confidence Intervals

When we calculate a single statistic to describe a sample we are calculating a point estimate. This point estimate is a useful statistic for estimating a parameter, but if we calculate a new statistic from another sample, the new point estimate is likely to be diffierent. To account for this, we may use an interval estimate instead of a point estimate. The most common interval estimate is the confidence interval, which is the interval between the upper and lower confidence limits for a statistic. The confidence interval is calculated using a predetermined signifigance level, $\alpha$. Conventionally we set $\alpha = 0.05$. The *confidence coefficient* is defined as $1 - \alpha$, but often referred to as a percentage. A p-value usually expresses the probability that results at least as extreme as those obtained in a sample were due to chance. Therefore if something occurs with a probability less than the given p-value, that occurence is considered statistically significant.

## 24.12 t-Test

The $t$-Test is used to make inferences about a single mean, or inferences about two means or variances, when sample sizes are small and/or the population distribution is unknown.

$$t = \frac{\bar{x} - \mu}{\frac{s}{\sqrt{n}}} \tag{24.19}$$

The denominator $\frac{s}{\sqrt{n}}$ is the *standard error*. The degrees of freedom *d.f.* are $n - 1$. Using a p-value and the degrees of freedom, we can find a value for significance level for $t$. If the above equation produces a value greater than the significance level, then we can reject the null hypothesis.

Let's assume that the population mean is $\mu = 2.1$. We collect a sample of size $n = 8$ with mean $\bar{x} = 2.7$, and variance $s^2 = 0.49$, $s = 0.70$. At $p = 0.05$ the level of t-significance is $t = 2.365$, and at $p = 0.01$ the level of t-significance is $t = 3.499$. Then,

$$t = \frac{2.7 - 2.1}{0.247} = 2.424 \tag{24.20}$$

Thus, we calculated at t-value of 2.424 which is greater than the t-significance value for the 0.05 level, so it is significant at that level, but it is lower than the t-significance value for the 0.01 level, so it is not significant at that level.

## 24.13 Statistic

A statistic is a function of a random variable that does not involve any parameters of the random variable. It is, itself, a random variable.

## 24.14 Estimator

An estimator $\hat{\Theta}$ is a statistic that makes estimates $\hat{\theta}$ of a parameter $\theta$.

## 24.15 Image Error Measures

$$\varepsilon_{MSE} = \mathbb{E}(|I - \hat{I}|^2) \tag{24.21}$$

$$\varepsilon_{NMSE} = \frac{\mathbb{E}(|I - \hat{I}|^2)}{\mathbb{E}(|I|^2)} \tag{24.22}$$

$$\varepsilon_{LSE} = \frac{1}{JK} \sum_{j=1}^{J} \sum_{k=1}^{K} |I[j,k] - \hat{I}[j,k]|^2 \tag{24.23}$$

$$\varepsilon_{NLSE} = \frac{\sum_{j=1}^{J} \sum_{k=1}^{K} |I[j,k] - \hat{I}[j,k]|^2}{\sum_{j=1}^{J} \sum_{k=1}^{K} |I[j,k]|^2} \tag{24.24}$$

1. Structural Content (SC)

2. Mean Square Error (MSE)

3. Peak SNR (PSNR)

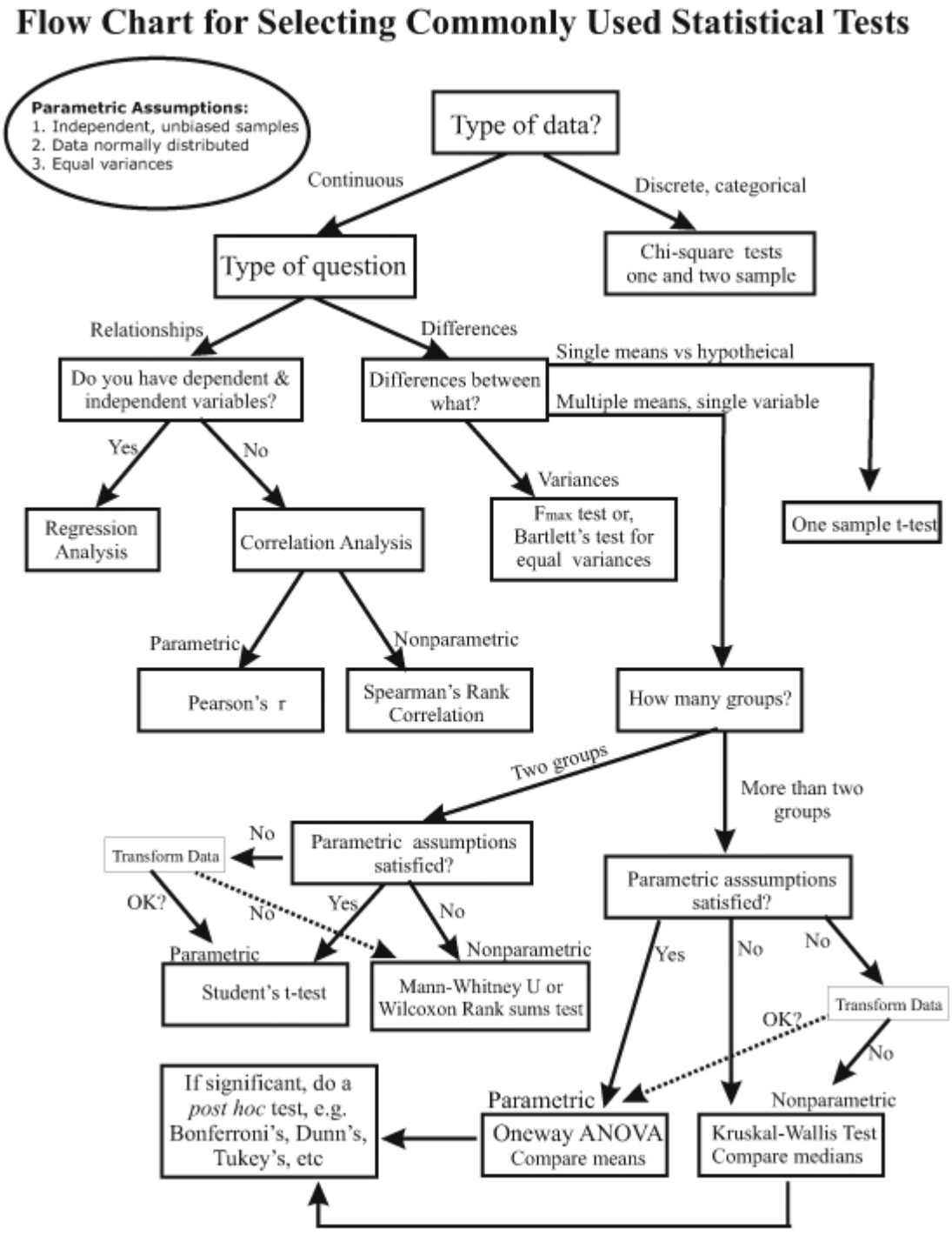
4. Normalized Cross Correlation (NCC)

5. Average Difference (AD)

6. Max Difference (MD)

7. Normalized Absolute Error (NAE)

## 24.16 Flow Chart



# Chapter 25

# Math

(top)

## 25.1 Area of a Convex Polygon

$$\text{Area} = \frac{1}{2}\begin{vmatrix} x_1 & y_1 \\ x_2 & y_2 \\ x_3 & y_3 \\ \vdots & \vdots \\ x_n & y_n \\ x_1 & y_1 \end{vmatrix} = \frac{1}{2}[(x_1y_2 + x_2y_3 + \cdots + x_ny_1) - (y_1x_2 + y_2x_3 + \cdots + y_nx_1)] \tag{25.1}$$

In Python, this looks like:

```
import numpy as np
a = np.array([[2,5],[-4,3],[5,1],[2,5]])
rows, cols = a.shape ; s = 0
for i in range( rows-1 ):
    s += a[i,0] * a[i+1,1] - a[i+1,0] * a[i,1]
print 0.5 * s

>>> 15
```

## 25.2 Artificial Neural Networks

This is a rough implementation of an artificial neural network (ANN) containing a single neuron. The parameter ***eta*** is the learning rate. The inputs are $\{x_i\}$, the corresponding weights are $\{w_i\}$, and the response is $y = f(\sum_i x_i w_i)$.

```
import numpy as np

class Neuron:
    def __init__( self, weight, fct ):
        self.weight = weight   # weights
        self.fct = fct # activation function
    def activate( self, inp ):
        net = 0
        for x,w in zip( inp, self.weight ):
            net += x*w
        y = self.fct( net )
```

```
        return y
    def train( self, data, eta ):
        rows, cols = data.shape
        for i in range( rows ):
            error = data[i,-1] - self.activate( data[i,:-1] )
            dw = list()
            for j in range( cols-1 ):
                dw.append( eta * error * data[i,j] )
                self.weight[j] = self.weight[j] + dw[-1]
            print str(error)[:8], self.weight

f = lambda x : x
n = Neuron( [ 0.5, -0.3, 0.8 ], f )
data = np.array([[1,1,.5,.7],[-1,.7,-.5,.2],[.3,.3,-.3,.5]])
eta = 0.1
n.train( data, eta )
```

## 25.3 Calculus

### 25.3.1 Some Identities

$$\int x^n \, dx = \frac{x^{n+1}}{n+1} + C \tag{25.2}$$

$$\int \frac{1}{x} \, dx = \ln |x| + C \tag{25.3}$$

$$\int \frac{c}{ax+b} \, dx = \frac{c}{a} \ln |ax+b| + C \tag{25.4}$$

$$\int e^x \, dx = e^x + C \tag{25.5}$$

$$\int a^x \, dx = \frac{a^x}{\ln a} + C \tag{25.6}$$

$$\int \log_a x \, dx = x \log_a x - \frac{x}{\ln a} + C \tag{25.7}$$

$$\int \sin x \, dx = -\cos x + C \tag{25.8}$$

$$\int \cos x \, dx = \sin x + C \tag{25.9}$$

$$\int \tan x \, dx = -\ln |\cos x| + C \tag{25.10}$$

$$\int \cot x \, dx = \ln |\sin x| + C \tag{25.11}$$

$$\int \sec x \, dx = \ln |\sec x + \tan x| + C \tag{25.12}$$

$$\int \csc x \, dx = -\ln |\csc x + \cot x| + C \tag{25.13}$$

### 25.3.2 Integration by Parts

$$\int f(x)g'(x)\,dx = f(x)g(x) - \int f'(x)g(x)\,dx \tag{25.14}$$

$$\int u\,dv = uv - \int v\,du \tag{25.15}$$

An example:

$$\int x\cos(x)\,dx \tag{25.16}$$

Let $u = x$, and $dv = \cos(x)$. Then $du = dx$, and $v = \sin(x)$.

$$\begin{aligned}\int x\cos(x)\,dx &= \int u\,dv && (25.17)\\ &= uv - \int v\,du && (25.18)\\ &= x\sin(x) - \int \sin(x)\,dx && (25.19)\\ &= x\sin(x) + \cos(x) + C && (25.20)\end{aligned}$$

### 25.3.3 Substitution

$$\int_a^b h(t)\,dt = \int_a^b f(g(t))g'(t)\,dt = \int_{g(a)}^{g(b)} f(x)\,dx \tag{25.21}$$

An example:

$$\int_0^2 x\cos(x^2+1)\,dx \tag{25.22}$$

Let $u = x^2 + 1$, then $du = 2xdx$.

$$\begin{aligned}\int_{x=0}^{x=2} x\cos(x^2+1)\,dx &= \frac{1}{2}\int_{u=1}^{u=5} cos(u)\,du && (25.23)\\ &= \frac{1}{2}(\sin(5) - \sin(1)) && (25.24)\end{aligned}$$

## 25.4 Convolution

$$(f \star g)[n] = \sum_{k=-\infty}^{\infty} f[k]g[n-k] \tag{25.25}$$

```
In [1]: np.convolve( [ 1, 2, 3 ], [ 4, 5, 6 ] )
Out[1]: array([ 4, 13, 28, 27, 18])
```

| n | $\sum f[k]g[n-k]$ | $(f \star g)[n]$ |
|---|---|---|
| 0 | $f[0]g[0]$ | $1 \cdot 4$ |
| 1 | $f[0]g[1] + f[1]g[0]$ | $1 \cdot 5 + 2 \cdot 4$ |
| 2 | $f[0]g[2] + f[1]g[1] + f[2]g[0]$ | $1 \cdot 6 + 2 \cdot 5 + 3 \cdot 4$ |
| 3 | $f[1]g[2] + f[2]g[1]$ | $2 \cdot 6 + 3 \cdot 5$ |
| 4 | $f[2]g[2]$ | $3 \cdot 6$ |

```
def discrete_convolution( x, y ):
    lx, ly = len( x ), len( y )
    N = lx + ly - 1
    conv = list()
    for n in range( N ):
        sum = 0
        for k in range( max( lx, ly ) ):
            if n-k >= 0 and n-k < ly and k < lx:
                sum += x[k]*y[n-k]
        conv.append( sum )
    return conv
```

## 25.5 Curl and Divergence

Let $\vec{F}$ be a vector field. Then curl is given by,

$$\text{curl } \vec{F} = \nabla \times \vec{F} \tag{25.26}$$

Let $\vec{F} = [\vec{F}_x, \vec{F}_y, \vec{F}_z]$, then

$$\nabla \times \vec{F} = \begin{vmatrix} \vec{i} & \vec{j} & \vec{k} \\ \frac{\partial}{\partial x} & \frac{\partial}{\partial y} & \frac{\partial}{\partial z} \\ \vec{F}_x & \vec{F}_y & \vec{F}_z \end{vmatrix} \tag{25.27}$$

$$= (\frac{\partial \vec{F}_z}{\partial y} - \frac{\partial \vec{F}_y}{\partial z})\vec{i} + (\frac{\partial \vec{F}_x}{\partial z} - \frac{\partial \vec{F}_z}{\partial x})\vec{j} + (\frac{\partial \vec{F}_y}{\partial x} - \frac{\partial \vec{F}_x}{\partial y})\vec{k} \tag{25.28}$$

Divergence is given by,

$$\text{div } \vec{F} = \nabla \cdot \vec{F} = \frac{\partial \vec{F}_x}{\partial x} + \frac{\partial \vec{F}_y}{\partial y} + \frac{\partial \vec{F}_z}{\partial z} \tag{25.29}$$

## 25.6 Exponent and Logarithm Identities

| | |
|---|---|
| $a^n = aaa \ldots a$ | $y = \log_a(x) \leftrightarrow x = a^y$ |
| $a^m a^n = a^{m+n}$ | $\log_a(MN) = \log_a(M) + \log_a(N)$ |
| $\frac{a^m}{a^n} = a^{m-n}$ | $\log_a(M/N) = \log_a(M) - \log_a(N)$ |
| $a^{m^n} = a^{mn}$ | $\log_a(M^p) = p \log_a(M)$ |
| $a^{-n} = \frac{1}{a^n}$ | $\log_a(M^{-1}) = -\log_a(M)$ |
| $a^{1/n} = \sqrt[n]{a}$ | $\log_a(M^{1/n}) = \frac{1}{n}\log_a(M)$ |
| $a^{\log_a(M)} = M$ | $\log_a(a^M) = M$ |
| $a^0 = 1$ | $\log_a(1) = 0$ |
| $a^1 = a$ | $\log_a(a) = 1$ |

## 25.7 Kroenecker Product

Let $A = \left(a_{ij}\right)$ and $B = \left(b_{ij}\right)$ be two transition probability matrices.

$$A \otimes B = \begin{bmatrix} a_{11}b_{11} & a_{11}b_{12} & \cdots & a_{11}b_{1q} & \cdots & \cdots & a_{1n}b_{11} & a_{1n}b_{12} & \cdots & a_{1n}b_{1q} \\ a_{11}b_{21} & a_{11}b_{22} & \cdots & a_{11}b_{2q} & \cdots & \cdots & a_{1n}b_{21} & a_{1n}b_{22} & \cdots & a_{1n}b_{2q} \\ \vdots & \vdots & \ddots & \vdots & & & \vdots & \vdots & \ddots & \vdots \\ a_{11}b_{p1} & a_{11}b_{p2} & \cdots & a_{11}b_{pq} & \cdots & \cdots & a_{1n}b_{p1} & a_{1n}b_{p2} & \cdots & a_{1n}b_{pq} \\ \vdots & \vdots & & \vdots & \ddots & & \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots & & \ddots & \vdots & \vdots & & \vdots \\ a_{m1}b_{11} & a_{m1}b_{12} & \cdots & a_{m1}b_{1q} & \cdots & \cdots & a_{mn}b_{11} & a_{mn}b_{12} & \cdots & a_{mn}b_{1q} \\ a_{m1}b_{21} & a_{m1}b_{22} & \cdots & a_{m1}b_{2q} & \cdots & \cdots & a_{mn}b_{21} & a_{mn}b_{22} & \cdots & a_{mn}b_{2q} \\ \vdots & \vdots & \ddots & \vdots & & & \vdots & \vdots & \ddots & \vdots \\ a_{m1}b_{p1} & a_{m1}b_{p2} & \cdots & a_{m1}b_{pq} & \cdots & \cdots & a_{mn}b_{p1} & a_{mn}b_{p2} & \cdots & a_{mn}b_{pq} \end{bmatrix} \tag{25.30}$$

As an example, we have:

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \otimes \begin{bmatrix} 0 & 5 \\ 6 & 7 \end{bmatrix} = \begin{bmatrix} 0 & 5 & 0 & 10 \\ 6 & 7 & 12 & 14 \\ 0 & 15 & 0 & 20 \\ 18 & 21 & 24 & 28 \end{bmatrix}$$

## 25.8 LOWESS

Locally weighted scatterplot smoothing. Combines multiple regression model in a k-nearest neighbor based meta-model.

```
def lowess(x, y, f=2./3., iter=3):
    """lowess(x, y, f=2./3., iter=3) -> yest"""
    n = len(x)
    r = int(ceil(f*n))
    h = [sort(abs(x-x[i]))[r] for i in range(n)]
    w = clip(abs(([x]-transpose([x]))/h),0.0,1.0)
    w = 1-w*w*w
    w = w*w*w
    yest = zeros(n,'d')
    delta = ones(n,'d')
    for iteration in range(iter):
        for i in range(n):
            weights = delta * w[:,i]
            b = array([sum(weights*y), sum(weights*y*x)])
            A = array([[sum(weights), sum(weights*x)], [sum(weights*x), sum(weights*x*x)]])
            beta = solve_linear_equations(A,b)
            yest[i] = beta[0] + beta[1]*x[i]
        residuals = y-yest
        s = median(abs(residuals))
        delta = clip(residuals/(6*s),-1,1)
        delta = 1-delta*delta
        delta = delta*delta
    return yest
```

## 25.9 Markov Chain

Describes the probability of going from one state to another

$$P = \left(p_{ij}\right) = \Pr(X_{k+1} = j \,|\, X_k = i) \tag{25.31}$$

$$x^{(n+1)} = x^{(n)} P \tag{25.32}$$

## 25.10 Haralick Textures

A general procedure for extracting textural properties from blocks of image data. These features are calculated in the spatial domain, and the statistcial nature of texture is taken into account in the procedure, which is based on the assumption that the texture information in an image I is contained in the overall or average spatial relationship which the gray tones in the image have to one another. We compute a set of gray-tone spatial-dependence probability-distribution matrices for a given image block and suggest a set of 14 textural features which can be extracted from each of these matrices. The features contain information about homogeneity, gray-tone linear dependencies (linear structure), contrast, number and nature of boundaries present, and the complexity of the image. The number of operations required to compute any one of these features is proportional to the number of resolution cells in the image block.

The next listing is a script for generating a grey level cooccurrence matrix from a three dimensional data set.

```
def gray_level_cooccurence_matrix_3d( x ):

    rows, cols, bands = x.shape
    m = x.max() + 1
    g = dict()

    for k in ['f','l','u','uf','ub','ul','ur','fl','fr','ufl','ufr','dfl','dfr']:
        g[k] = np.zeros((m,m))

    # forward
    for i in range( 1,rows ):
        for j in range( cols ):
            for k in range( bands ):
                g['f'][ x[i,j,k], x[i-1,j,k] ] += 1


    # left
    for i in range( rows ):
        for j in range( 1,cols ):
            for k in range( bands ):
               g['l'][ x[i,j,k], x[i,j-1,k] ] += 1

    # up
    for i in range( rows ):
        for j in range( cols ):
            for k in range( 1,bands ):
               g['u'][ x[i,j,k], x[i,j,k-1] ] += 1

    # upward-forward / downward-backward directions
    for i in range( 1,rows ):
        for j in range( cols ):
            for k in range( 1,bands ):
               g['uf'][ x[i,j,k], x[i-1,j,k-1] ] += 1

    # upward-backward / downward-forward directions
    for i in range( rows-1 ):
        for j in range( cols ):
            for k in range( 1,bands ):
                g['ub'][ x[i,j,k], x[i+1,j,k-1] ] += 1
```

```
    # upward-left / downward-right directions
    for i in range( rows ):
        for j in range( 1,cols ):
            for k in range( 1,bands ):
                   g['ul'][ x[i,j,k], x[i,j-1,k-1] ] += 1

    # upward-right / downward-left directions
    for i in range( rows ):
        for j in range( cols-1 ):
            for k in range( 1,bands ):
                   g['ur'][ x[i,j,k], x[i,j+1,k-1] ] += 1

    # forward-left / backward-right
    for i in range( 1,rows ):
        for j in range( 1,cols ):
            for k in range( bands ):
                g['fl'][ x[i,j,k], x[i-1,j-1,k] ] += 1

    # forward-right / backward-left
    for i in range( 1,rows ):
        for j in range( cols-1 ):
            for k in range( bands ):
                g['fr'][ x[i,j,k], x[i-1,j+1,k] ] += 1

    # upward-forward-right
    for i in range( 1,rows ):
        for j in range( cols-1 ):
            for k in range( 1,bands ):
                g['ufr'][ x[i,j,k], x[i-1,j+1,k-1] ] += 1

    # upward-forward-left
    for i in range( 1,rows ):
        for j in range( 1,cols ):
            for k in range( 1,bands ):
                g['ufl'][ x[i,j,k], x[i-1,j-1,k-1] ] += 1

    # downward-forward-right
    for i in range( 1,rows ):
        for j in range( cols-1 ):
            for k in range( bands-1 ):
                g['dfr'][ x[i,j,k], x[i-1,j+1,k+1] ] += 1

    # downward-forward-left
    for i in range( 1,rows ):
        for j in range( 1,cols ):
            for k in range( bands-1 ):
                g['dfl'][ x[i,j,k], x[i-1,j-1,k+1] ] += 1

    for k,v in g.iteritems():
        g[k] = v + v.T
        g[k] /= float( np.sum( g[k] ) )

    return g
```

Next, we list the Haralick texture features:

$$f_1 = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} p(i,j)^2 \tag{25.33}$$

```
def angular_second_moment( p ):
    rows, cols = p.shape
    asm = 0
    for i in range( rows ):
```

```
        for j in range( cols ):
            asm += p[i,j]**2.0
    return asm
```

$$f_2 = \sum_{n=0}^{N-1} n^2 \left\{ \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} p(i,j) \right\}, \quad n = |i-j| \tag{25.34}$$

```
def contrast( p ):
    rows, cols = p.shape
    ct = 0
    for i in range( rows ):
        for j in range( cols ):
            ct += p[i,j] * abs( i-j )**2.0
    return ct
```

$$f_3 = \frac{1}{\sigma_X \sigma_Y} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} (ij) p(i,j) - \mu_X \mu_Y \tag{25.35}$$

```
def correlation( p ):
    rows, cols = p.shape
    mu = 0
    for i in range( rows ):
        for j in range( cols ):
            mu += p[i,j] * i
    v = 0
    for i in range( rows ):
        for j in range( cols ):
            v += p[i,j] * ( i - mu )**2.0
    sd = np.sqrt( v )
    corr = 0
    for i in range( rows ):
        for j in range( cols ):
            corr += p[i,j]*(((i-mu)*(j-mu))/float(sd*sd))
    return corr
```

$$f_4 = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} (i - \mu_X)^2 p(i,j) \tag{25.36}$$

```
def sum_of_sqrs_variance( p ):
    rows, cols = p.shape
    mu = 0
    for i in range( rows ):
        for j in range( cols ):
            mu += p[i,j] * i
    ssv = 0
    for i in range( rows ):
        for j in range( cols ):
            ssv += p[i,j] * ( i - mu )**2.0
    return ssv
```

$$f_5 = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \frac{1}{1 + (i-j)^2} p(i,j) \tag{25.37}$$

```
def inverse_difference_moment( p ):
    rows, cols = p.shape
    idm = 0
    for i in range( rows ):
        for j in range( cols ):
            idm += p[i,j] / float( 1 + (i-j)**2.0 )
    return idm
```

$$p_{X+Y}(k) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} p(i,j), \quad k = i + j, \quad k = 0, 1, 2, \ldots, 2(N-1) \tag{25.38}$$

```
def sum_problty( p ):
    rows, cols = p.shape
    sp = list()
    s = 0
    for k in range( 2 * ( rows - 1 ) + 1 ):
        for i in range( rows ):
            for j in range( cols ):
                if i+j == k:
                    s += p[i,j]
        sp.append( s )
        s = 0
    return sp
```

$$p_{|X-Y|}(k) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} p(i,j), \quad k = |i - j|, \quad k = 0, 1, 2, \ldots, N-1 \tag{25.39}$$

```
def diff_problty( p ):
    rows, cols = p.shape
    dp = list()
    d = 0
    for k in range( rows ):
        for i in range( rows ):
            for j in range( cols ):
                if abs( i-j ) == k:
                    d += p[i,j]
        dp.append( d )
        d = 0
    return dp
```

$$f_6 = \sum_{k=0}^{2(N-1)} k p_{X+Y}(k) \tag{25.40}$$

```
def sum_average( p ):
    rows, cols = p.shape
    psum = sum_problty( p )
    sa = 0
    for i in range( 2 * ( rows - 1 ) + 1 ):
        sa += i * psum[i]
    return sa
```

$$f_7 = \sum_{k=0}^{2(N-1)} (k - f_6)^2 p_{X+Y}(k) \tag{25.41}$$

```
def sum_variance( p ):
    rows, cols = p.shape
    psum = sum_problty( p )
    sv = 0
    for i in range( 2 * ( rows - 1 ) + 1 ):
        sv += psum[i] * ( i - sum_average( p ) )**2.0
    return sv
```

$$f_8 = -\sum_{k=0}^{2(N-1)} p_{X+Y}(k) \ln p_{X+Y}(k) \tag{25.42}$$

```
def sum_entropy( p ):
    rows, cols = p.shape
    psum = sum_problty( p )
    se = 0
    for i in range( 2 * ( rows - 1 ) + 1 ):
        se += psum[i] * np.log( psum[i]+(psum[i]==0) )
    return -se/np.log(2.0)
```

$$f_9 = -\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} p(i,j) \ln p(i,j) \tag{25.43}$$

```
def entropy( p ):
    rows, cols = p.shape
    ntrpy = 0
    for i in range( rows ):
        for j in range( cols ):
            ntrpy += p[i,j] * np.log( p[i,j]+(p[i,j]==0) )
    return -ntrpy/np.log(2.0)
```

$$f_{10} = \mathbb{V}(p_{|X-Y|}) \tag{25.44}$$

```
def difference_variance( p ):
    pdiff = diff_problty( p )
    n = len( pdiff )
    mu = 0
    for k in range( n ):
        mu += pdiff[k]
    mu /= n
    v = 0
    for k in range( n ):
        v += ( pdiff[k] - mu )**2.0
    v /= n
    return v
```

$$f_{11} = -\sum_{k=0}^{N-1} p_{|X-Y|}(k) \ln p_{|X-Y|}(k) \tag{25.45}$$

```
def difference_entropy( p ):
    rows, cols = p.shape
    pdiff = diff_problty( p )
    de = 0
    for i in range( rows ):
        de += pdiff[i] * np.log( pdiff[i]+(pdiff[i]==0) )
    return -de/np.log(2.0)
```

$$HX = \sum_{i=0}^{N-1} p_X(i) \ln p_X(i) \tag{25.46}$$

```
def hx( p ):
    rows, cols = p.shape
    px = np.sum( p, axis=0 )
    h = 0
    for i in range( rows ):
        h += px[i] * np.log( px[i] + ( px[i]==0 ) )
    return -h/np.log(2.0)
```

$$HY = \sum_{j=0}^{N-1} p_Y(j) \ln p_Y(j) \tag{25.47}$$

```
def hy( p ):
    rows, cols = p.shape
    py = np.sum( p, axis=1 )
    h = 0
    for i in range( rows ):
        h += py[i] * np.log( py[i] + ( py[i]==0 ) )
    return -h/np.log(2.0)
```

$$HXY = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} p(i,j) \ln p(i,j) \tag{25.48}$$

```
def hxy( p ):
    return entropy( p )
```

$$HXY_1 = -\sum_{i=0}^{N-1} \sum_{j=0}^{N-1} p(i,j) \ln p_X(i) p_Y(j) \tag{25.49}$$

```
def hxy1( p ):
    rows, cols = p.shape
    px = np.sum( p, axis=0 )
    py = np.sum( p, axis=1 )
    h = 0
    for i in range( rows ):
        for j in range( rows ):
            h += p[i,j] * np.log( px[i] * py[j] + ( px[i] * py[j] == 0 ) )
    return -h/np.log(2.0)
```

$$HXY_2 = -\sum_{i=0}^{N-1} \sum_{j=0}^{N-1} p_X(i) p_Y(j) \ln p_X(i) p_Y(j) \tag{25.50}$$

```
def hxy2( p ):
    rows, cols = p.shape
    px = np.sum( p, axis=0 )
    py = np.sum( p, axis=1 )
    h = 0
    for i in range( rows ):
        for j in range( rows ):
            h += px[i] * py[j] * np.log( px[i] * py[j] + ( px[i] * py[j] == 0 ) )
    return -h/np.log(2.0)
```

$$f_{12} = \frac{HXY - HXY_1}{max\{HX, HY\}} \tag{25.51}$$

```
def information_a( p ):
    return ( hxy(p) - hxy1(p) )/float( np.max( [ hx(p), hy(p) ] ) )
```

$$f_{13} = \sqrt{1 - exp(-2(HXY_2 - HXY))} \tag{25.52}$$

```
def information_b( p ):
    return np.sqrt( 1 - np.exp( -2.0 * ( hxy2(p) - hxy(p) ) ) )
```

## 25.11 Hidden Markov Models

Suppose Alice and Bob live far apart, but they talk every day on the telephone and Bob tells her what he did that day. Every day Bob either walks in the park, shops, or cleans his apartment. Alice figures that these activities are determined by the weather where Bob lives, but he never tells her what the weather was that day. Alice assumes that the weather is controlled by a hidden Markov model having two states: "rainy" and "sunny". She believes that it is generally rainy.

The weather, the hidden part, is goverened by a *transition matrix*. Bob's actions, are controlled by an *emission matrix*, or output matrix.

### 25.11.1 Forward Algorithm

Determines the most likely *present-state* that of the HMM, given past observations of output.

### 25.11.2 Viterbi Algorithm

Determines the most likely *state-path* of the HMM, given the observations of the output.

### 25.11.3 Baum-Welch/Expectation-Maximization

Determines the actual structure of the HMM.

## 25.12 Minors

Let $\mathbf{A}$ be an $m \times n$ matrix and $k \in \mathbb{Z}$ such that $0 < k \leq m$, and $k \leq n$. A $k \times k$ minor of $\mathbf{A}$ is the determinant of a $k \times k$ matrix obtained from $\mathbf{A}$ by deleting $m - k$ rows and $n - k$ columns.

In particular, the $(i, j)$ minor of an $n \times n$ matrix $\mathbf{A}$, denoted $\mathbf{M}_{i,j}$, is defined as the determinant of the $(n-1) \times (n-1)$ matrix formed by removing from $\mathbf{A}$ its $i^{th}$ and $j^{th}$ column.

The $(i, j)$ cofactor, $\mathbf{C}_{i,j}$, of a square matrix $\mathbf{A}$ is given by $\mathbf{C}_{i,j} = (-1)^{i+j}\mathbf{M}_{i,j}$

If $i = j$ then $\mathbf{M}_{i,j}$ is called a *principal minor*.

## 25.13 Partial Differential Equations

$$\begin{cases} u_{tt} = c^2 u_{xx} \\ u(x, 0) = \phi(x) \qquad (-\infty < x < \infty, t > 0) \\ u_t(x, 0) = \psi(x) \end{cases} \tag{25.53}$$

$$u(x,t) = \frac{1}{2}(\phi(x-ct) + \phi(x+ct)) + \frac{1}{2c}\int_{x-ct}^{x+ct} \psi(s)ds \tag{25.54}$$

$$\begin{cases} u_{tt} = c^2 u_{xx} \\ u(x,0) = \phi(x) \\ u_t(x,0) = \psi(x) \\ u(0,t) = 0 \end{cases} \quad (-\infty \leq x < 0, t > 0) \tag{25.55}$$

$$u(x,t) = \begin{cases} x \geq ct : \frac{1}{2}\Big(\phi(x-ct) + \phi(x+ct)\Big) + \frac{1}{2c}\int_{x-ct}^{x+ct} \psi(s)ds \\ x < ct : \frac{1}{2}\Big(\Phi(x-ct) + \phi(x+ct)\Big) + \frac{1}{2c}\int_{x-ct}^{0} \Psi(s)ds + \frac{1}{2c}\int_{0}^{x+ct} \psi(s)ds \end{cases} \tag{25.56}$$

$$\Phi(x) = \begin{cases} \phi(x) & x \geq ct \\ -\phi(-x) & x < ct \end{cases} \tag{25.57}$$

$$\Psi(x) = \begin{cases} \psi(x) & x \geq ct \\ -\psi(-x) & x < ct \end{cases} \tag{25.58}$$

## 25.14 Vectors

Vectors are quantities with magnitude and direction. Let **u** abd **v** be vectors, then their dot product is given by,

$$\mathbf{u} \cdot \mathbf{v} = \sum_{i=1}^{n} u_i v_i \tag{25.59}$$

We also have the following helpful identity,

$$\mathbf{u} \cdot \mathbf{v} = \|\mathbf{u}\|\|\mathbf{v}\| \cos\theta \tag{25.60}$$

## 25.15 Wavelets

A wavelet is a wave-like oscillation. We prefer continuously differentiable functions with compact support. To satisfy analytical requirements (in the continuous WT) and in general for theoretical reasons, one chooses wavelet functions from a subspace of the space $L^1(\mathbb{R}) \cap L^2(\mathbb{R})$, the space of measurable functions that are absolutely, and square integrable [Wk]:

$$\int_{-\infty}^{\infty} |\psi(t)|dt < \infty \tag{25.61}$$

$$\int_{-\infty}^{\infty} |\psi(t)|^2 dt < \infty \tag{25.62}$$

We also have other related conditions, such as the *admissibility condition* which guarantees that a signal can be analyzed, and then synthesized, without loss of information. Let $\psi(t)$ be a square, integrable function, and let $\Psi(\omega)$ be the Fourier transform of $\psi(t)$, the the admissibility condition is given as [poly]

$$\int_{-\infty}^{\infty} \frac{|\Psi(\omega)|^2}{|\omega|} d\omega < \infty \tag{25.63}$$

The admissibility condition implies that the Fourier transform of $\psi(t)$ vanishes at the zero frequency, i.e. [poly]

$$|\Psi(\omega)|^2_{\omega=0} = 0 \tag{25.64}$$

A zero at the zero frequency also means that the average value of the wavelet in the time domain must be zero, [poly]

$$\int \psi(t) dt = 0 \tag{25.65}$$

furthermore, for the scaling function we have, [poly]

$$\int \phi(t) dt = 1 \tag{25.66}$$

The subspace of a scale, $a$, or frequency band $[1/a, 2/a]$ is generated by the functions, called *child wavelets* [Wk]

$$\psi_{s,\tau}(t) = \frac{1}{\sqrt{s}} \psi\left(\frac{t-\tau}{s}\right) \tag{25.67}$$

where $s$ is positive and defines the scale, and $\tau$ is any real number, and defines the translation. [Wk]

The projection of a function, $x$, onto the subspace of a scale, $s$, then has the form [Wk]

$$x_s(t) = \int_{\mathbb{R}} WT_\psi\{x\}(s,\tau) \cdot \psi_{s,\tau}(t) d\tau \tag{25.68}$$

with wavelet coefficients [Wk]

$$WT_\psi\{x\}(s,\tau) = \langle x, \psi_{s,\tau} \rangle = \int_{\mathbb{R}} x(t) \psi_{s,\tau}(t) dt \tag{25.69}$$

We can define an orthogonal wavelet using a scaling filter, i.e., a low-pass finite impulse response filter of length 2N and sum of 1. The high-pass filter is calculated as the quadrature mirror filter of the low-pass filter. [Wk]

The quadrature mirror filter, qmf, of a filter $[1, 2, 3, 4]$ is $[4, -3, 2, -1]$. We can calculate the qmf of a given filter using Python and SciPy:

```
In [1]: import scipy.signal.wavelets
In [2]: scipy.signal.wavelets.qmf([1,2,3,4])
Out[2]: array([4,-3,2,-1])
```

In windowed Fourier analysis, the size of the window is fixed, and the number of oscillations varies. A "small" window, one with many oscillations, is blind to low frequencies, which are too large for the window. A "large" window, one with fewer oscillations, drowns out transient changes that get lost in the totality of the information concerning the entire interval corresponding to the window. [Hubbard]

From the perspective of signal processing, a function is either (a) a signal to be analyzed, or (b) a filter with which to analyze a signal. A filter can be realized physically as an electrical circuit with one wire that carries the signal in and another wire that carries the filtered signal out. A filter can be thought of abstractly as a function. [Hubbard]

Whether physical or abstract, the effect of a filter can be better understood in Fourier space: the FOurier transform of the signal is multiplied by the Fourier transform of the filter, letting some frequencies through, while blocking others. [Hubbard]

For example, if the Fourier transform of the filter is almost 1 near 0, and almost 0 everywhere else, then the signal's low frequencies will (mostly) surivive this multiplication of (almost) 1, but the higher frequencies will be (effectively) eliminated. This is an example of a low pass filter. The effect in physical space is to smooth the signal. The small variations given by the high frequencies disappear, leaving the general tendency produced by the lower frequencies.

[Wk] Wikipedia
[poly] http://www.polyvalens.com/blog/?page_id=15
[Hubbard] Barbara B. Hubbard. *The World According to Wavelets.* A. K. Peters. Natlick, Massachusetts. 1998.

---

Wavelets can be thought of brief oscillations, or as their name implies, small waves. Taking the inner product of a function and a series of wavelets produces a series of coefficients describing the high frequencies of a function at a given scale, which is determined by parameters input into the wavelet function. Associated with wavelets are scaling functions. Taking the inner product of a function and a set of scaling functions produces a series of coefficients describing the low frequencies of a function at a given scale, which is determined by parameters input into the scaling function. A series of wavelets is defined as follows,

$$\psi_{j,k}(t) = 2^{-m/2}\psi(2^{-m}t - n). \tag{25.70}$$

The low frequncy coefficients can then be used to either approximate the signal, or be used in conjunction with the high frequency coefficients to reproduce the original signal.

A wavelet decomposition can be effected using a low pass filter and and a high pass filter, generated by the quadrature mirror filter of the low pass filter. A quadrature mirror filter is produced by reversing a given filter and taking the inverse sign of every other element. When the signal is convolved with the high pass filter, which is a associated with the wavelet function, and then down-sampled by two, the high frequency detail coefficients are produced. When a signal is convolved with the low pass filter, which is associated with the scaling function, the low frequency approximating coeffients are produced. A signal can then be passed though layers of a filter bank to produce wavelet detail and approximating coefficients. Below, we have the equations that used to implement a filter bank [**?**],

$$\lambda_{j-1}(k) = \sum_m h(m - 2k)\lambda_j(m), \tag{25.71}$$

$$\gamma_{j-1}(k) = \sum_m g(m - 2k)\gamma_j(m). \tag{25.72}$$

In the above equations, $h$ and $g$ are the low pass and high pass filters, respectively, and $\lambda_{j_{max}} = \gamma_{j_{max}} = x(t)$, where $x(t)$ is the original signal. In this manner, we can perform a wavelet decomposition using digital filters without ever knowing the shape of the wavelets and scaling functions.

## 25.16 Wiener Filters

The Wiener filter can be used to reduce the amount of noise present in images. Let $I(i, j)$ be the original image. You can form a small window that floats over the image and replaces each pixel with a filtered pixel, $J(i, j)$, calculated using the mean and variance calculated within the window.

$$J(i, j) = \mu + \frac{\sigma^2 + \sigma}{\sigma^2}(I(i, j) - \mu) \tag{25.73}$$

## 25.17 Waves

### 25.17.1 Standing Waves

*Standing waves*, or *stationary waves*, is a wave that remains in a constant position. This occurs when either the media is traveling in such a way that the wave appears to be stationary, or when there is interference between two waves travelling in opposite direction. In both cases, the medium continues to resonate, but the wave doesn't go anywhere. In other words, the nodes, where the wave crosses its centerline, never move, even while the antinodes, the peaks and troughs, continue to oscillate.

### 25.17.2 Elastic Waves

*P-waves* are elastic waves that travel through a continuum. The $P$ comes from both *primary*, as it has the highest velocity, and from *pressure*, as it is formed by alternating compression and rarefaction. *Rarefaction* is a reduction of a medium's density, the opposite of compression. In an isotropic homogenous media, the mode of propagation of a P-wave is longitudinal, i.e., it travels parallel to the direction of the energy that formed it. *Isotropic* means uniform in all directions. We give the velocity of a P-wave in an isotropic homogenous medium:

$$v_p = \sqrt{\frac{K + \frac{4}{3}\mu}{\rho}} = \sqrt{\frac{\lambda + 2\mu}{\rho}} \tag{25.74}$$

- $K$ : bulk modulus (the modulus of incompressibility)
- $\mu$ : shear modulus (the modulus of rigidity, $2^{nd}$ Lamé parameter)
- $\rho$ : density
- $\lambda$ : $1^{st}$ Lamé parameter
- $M = K + \frac{4}{3}\mu$ : elastic moduli P-wave modulus

The elastodynamic wave equation for isotropic homogenous media:

$$\mu\nabla^2 u + (\mu + \lambda)\nabla(\nabla \cdot u) + F = \rho\frac{\partial^2 u}{\partial t^2} \tag{25.75}$$

# Chapter 26

# Data Structures

(top)

## 26.1 Linked List

```
class Node:
    def __init__( self, data ):
        self.data = data
        self.next = None

class LinkedList:
    def __init__( self ):
        self.head = None
        self.tail = None
    def addNode( self, data ):
        newNode = Node( data )
        if self.head == None:
            self.head = newNode
        if self.tail != None:
            self.tail.next = newNode
        self.tail = newNode
```

## 26.2 Binary Tree

```
class Node:
    def __init__( self, data ):
        self.data = data
        self.left = None
        self.right = None

class Tree:
    def __init__( self ):
        self.root = None
    def insert( self, data ):
        if self.root == None:
            self.root = Node( data )
        else:
            tmp = self.root
            while True:
                if data <= tmp.data:
                    if tmp.left == None:
                        tmp.left = Node( data )
```

```
                    break
                tmp = tmp.left
            else:
                if tmp.right == None:
                    tmp.right = Node( data )
                    break
                tmp = tmp.right
```

## 26.3 Trie

```
class Node:
    def __init__( self, letter ):
        self.letter = letter
        self.children = []

class Trie:
    def __init__( self ):
        self.root = Node('')
        self.depth = 0
    def insert( self, word ):
        depth = 0
        tmp = self.root
        for i in range( 1,len( word )+1 ):
            children = tmp.children
            N = len( children )
            children = [ children[k].letter for k in range( N ) ]
            if word[:i] in children:
                tmp = tmp.children[ children.index( word[:i] ) ]
            else:
                tmp.children.append( Node( word[:i] ) )
                tmp = tmp.children[-1]
            depth += 1
        if depth > self.depth:
            self.depth = depth
```

## 26.4 Stack

```
class Node:
    def __init__( self, data ):
        self.data = data
        self.next = None
        self.prev = None

class Stack:
    def __init__( self ):
        self.top = None
        self.size = 0
    def push( self, data ):
        newNode = Node( data )
        self.size += 1
        if self.top == None:
            self.top = newNode
        else:
            newNode.prev = self.top
            self.top.next = newNode
            self.top = newNode
    def pop( self ):
        self.size -= 1
        data = self.top.data
```

```
        prev = self.top.prev
        if prev:
            self.top = self.top.prev
            self.top.next = None
        else:
            self.top = None
        return data
```

## 26.5 Queue

```
class Node:
    def __init__( self, data ):
        self.data = data
        self.next = None
        self.prev = None

class Queue:
    def __init__( self ):
        self.head = None
        self.tail = None
        self.size = 0
    def push( self, data ):
        newNode = Node( data )
        self.size += 1
        if self.tail == None:
            self.tail = newNode
            self.head = newNode
        else:
            newNode.next = self.tail
            self.tail = newNode
    def pop( self ):
        data = self.head.data
        prev = self.head.prev
        self.size -= 1
        if prev != None:
            self.head = prev
        else:
            self.head = self.tail
            self.tail = None
        return data
```

# Chapter 27

# HTML

(top)

## 27.1 Basic Format

Your document should be named something like `index.htm`, and it should begin and end with `<HTML>` and `</HTML>` tags. Using caps increases readability.

```
<!DOCTYPE HTML>
<HTML>
  <HEAD>
    <TITLE>
    </TITLE>
  </HEAD>
  <BODY>
  </BODY>
</HTML>
```

## 27.2 Paragraphs

Paragraphs like `P` tags.

```
<P>...</P>
```

## 27.3 Hyperlinks and PDFs

The `<A></A>` tag produces links. It takes an argument `HREF=` that directs the click-through. `HREF` stands for Hypertext REFerence.

If you want to provide a pdf document, you replace the link address with the name of the document, and put the pdf in the working directory of `index.htm`.

```
<A HREF="document.pdf">LinkText</A>
```

## 27.4 Images

The `<IMG />` tag produces images. It takes an argument `SRC=` that specifies the image shown.

Like pdfs, you'll need to put your image in the working directory of `index.htm`. You can adjust the aspect ratio of the image using the `WIDTH` and `HEIGHT` parameters.

```
<IMG SRC="headshot.jpg" ALT="headshot" WIDTH="304" HEIGHT="228" />
```

To link an image to a website, we enclose the `<IMG SRC=.../>` tag inside the `<A HREF=...>` tag, like so,

```
<A HREF="http://google.com"><IMG SRC="image.png"/></A>
```

## 27.5 Lists

We open an **o**rdered **l**ist environment with the `<OL></OL>` tags. Then, **l**ist **i**tems are added using `<LI></LI>` tags.

```
<OL>
  <LI>Item one.</LI>
  <LI>Item two.</LI>
</OL>
```

Unordered lists work the same way, but they use `<UL></UL>` tags. We can nest lists. Here is an ordered list of items, with unordered sublists.

```
<OL>
  <LI>Red
    <UL>
      <LI>Cadmium Red</LI>
      <LI>Alizarin Crimson</LI>
    </UL>
  </LI>
  <LI>Yellow
    <UL>
      <LI>Cadmium Yellow Light</LI>
      <LI>Cadmium Yellow Medium</LI>
    </UL>
  </LI>
  <LI>Blue
    <UL>
      <LI>Ultramarine</LI>
      <LI>Pthalo Blue</LI>
    </UL>
  </LI>
</OL>
```

## 27.6 In-Line CSS

We can make comments using,

```
<!--This is a comment.-->
```

We can change the font size in a paragraph by adding an attribute to the `<P></P>` tag.

```
<P STYLE="font-size: 8px">Small font.</P>
<P STYLE="font-size: 12px">Regular font.</P>
```

To change both size and color of an element, say a heading, we can add the `color` option to the style attribute.

```
<H1 STYLE="color: red; font-size: 16px">READ RED HEADING</H1>
```

We can also change the type face using the `font-family` option.

```
<P STYLE="font-family:Helvetica">Texts.</P>
```

We can also change the background color of body, or even things like lists!

```
<BODY STYLE="background-color:yellow">
  <OL STYLE="background-color:blue">
    <LI>Item One</LI>
    <LI>Item Deuce on the Loose</LI>
  </OL>
</BODY>
```

# Chapter 28

# CSS

(top)

## 28.1 Introduction

Cascading Style Sheets (CSS) allow you to separate the style of a page from its content. A stylesheet should be named something like `style.css` and should be located in the same directory as `index.html`. The basic form of style sheet elements is as follows.

```
selector {
    attribute: value, default-value ;
}
```

## 28.2 Borders

Borders act a little bit differently. They use multiple values for the same attribute. Here is the general case, as far as I can tell.

```
selector {
    border: 2px [dotted|dashed|solid] #XXXXXX ;
}
```

Here, `2px` is an arbitrary border width, the middle term, `[dotted|dashed|solid]`, is the border style, and `#XXXXXX` is a hexadecimal color.

## 28.3 Nested Elements

If an element is nested inside another element, say a `<P>` element nested inside a `<DIV>` element, then we write the style sheet for that element as,

```
DIV P {
    attribute: value ;
}
```

The above code says that any `<P>` element nested in any `<DIV>` element should have a this attribute/value. If we want to specify direct children, we use the `>` operator.

```
DIV > P {
    stmts ;
}
```

Now, we only touch <P> elements that are direct children of <DIV>.

## 28.4 Star Selector

A special kind of selector is the star selector, which selects ever element in the html document.

```
* {
    attribute: value ;
}
```

## 28.5 IDs

Nested selectors offer us great specificity, but we can become even more specific using ID attributes. These show up as separate selectors in the stylesheet, prefixed with a hash sign.

```
<!DOCTYPE HTML>
<HTML>
  <HEAD>
    <TITLE>le title</TITLE>
  </HEAD>
  <BODY>
    <DIV ID="P1">Here is one in red!</DIV>
    <DIV ID="P2">Here is another in blue.</DIV>
  </BODY>
</HTML>
```

And in the stylesheet we might see something like this.

```
DIV {
    font-family: Helvetica ;
}

#P1 {
    color: red ;
}

#P2 {
    color: blue ;
}
```

This code sets all of the DIV elements to have Helvetica type face, but it differentiates between P1 and P2.

## 28.6 Classes

Whereas IDs give us the greatest specificty possible, classes allow us to generalize over multiple elements. If we put an element in a class, then the class applies its attributes to that element. This way, instead of applying the same rule to multiple selectors, we can just apply a class to those HTML elements. In the style sheet, classes show up as selectors prefixed with a dot.

```
<!DOCTYPE HTML>
<HTML>
  <HEAD>
    <TITLE>le title</TITLE>
  </HEAD>
  <BODY>
    <H1 CLASS="FANCY">This is my fancy heading.</H1>
    <P CLASS="FANCY">This is my fancy paragraph.</P>
```

```
  </BODY>
</HTML>
```

And here is the stylesheet.

```
.FANCY {
    color: blue ;
    font-family: cursive ;
}
```

## 28.7 Pseudo-Class Selectors

Pseudo-class selectors allow us to access HTML items that are not part of the document tree. For instance, looking at the HTML document, we can see where a link is placed, but we cannot tell that it has been visited or not. Pseudo-class selectors allow us to control these behaviors.

```
<!DOCTYPE HTML>
<HTML>
  <HEAD>
    <TITLE>le title</TITLE>
  </HEAD>
  <BODY>
    <A HREF="http://www.google.com">Linky-dink.</A>
  </BODY>
</HTML>
```

The following pseudo-class selector makes the link bold when you hover over it. Not the use of the colon for the *hover event*.

```
A:HOVER {
    font-weight: bold
}
```

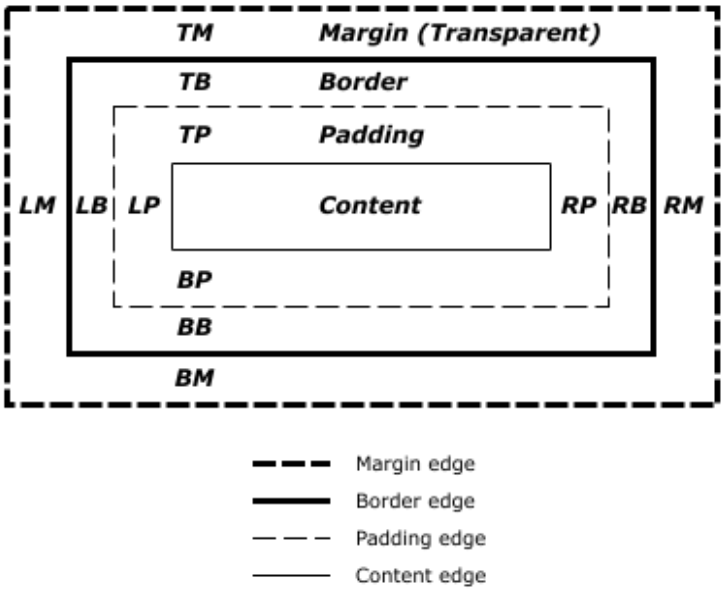
## 28.8 The Box Model

In the CSS box model, each HTML element sits inside a box with a number of parameters.

The `display` property has four possible values.

- `block` This box takes up the entire width of the page
- `inline-block` This box allows other elements to sit its line
- `inline` This block also allows other elements to sit on its line, but it only takes up as much space as it needs
- `none` This makes the element and its contents disappear entirely

*Floating* elements prevents them from landing on top of each other. Float is another property that takes values like *left*, *right*, and *center*.


TM
Margin (Transparent)
TB
Border
TP
Padding
LM
LB
LP
Content
RP
RB
RM
BP
BB
BM
Margin edge
Border edge
Padding edge
Content edge

# Chapter 29

# JavaScript

(top)

## 29.1 Variables

Variable declaration begins with the `var` keyword.

```
var x = 2 ;
var y = 'Hello!' ;
var z = true
```

## 29.2 String

Strings have a length attribute, and a substring method.

```
strlen = str.length ;
```

```
var str = 'Connor' ;
var a = 1 ;
var b = 3 ;
substr = str.substring( a, b ) ;
// substr --> 'onn'
```

## 29.3 Arrays

We can declare an array this way:

```
var x = new Array() ;
```

Or we can decalre them explicitly:

```
var mixed = [ "apple", 5, true ] ;
```

We can get the length of the array by:

```
arrlen = arr.length ;
```

## 29.4 Loops

For-loops look like for-loops from C and C++.

```
for( var i = 0 ; i < 10 ; i++ )
{
    console.log( i ) ;
};
```

For doing a two dimensional arry:

```
var n = new Array() ;
for( var i = 0 ; i < 10 ; i++ )
{
    n[i] = new Array() ;
    for( var j = 0 ; j < 10 ; j++ )
    {
        n[i][j] = i+j ;
        stmts ;
    };
};
```

We have both *while* and *do-while* loops!

```
while( cond )
{
    stmts ;
}
```

```
do
{
    stmts ;
}
while( cond )
```

## 29.5 Control Structures

The *if* statement takes its usual form

```
if( cond )
{
    stmts ;
}
```

```
if( cond )
{
    stmts ;
}
else
{
    stmts ;
}
```

```
if( cond )
{
    stmts ;
}
else if( cond )
{
    stmts ;
```

```
}
else
{
    stmts ;
}
```

I almost forgot about operators!

| | |
|---|---|
| `===` | equality |
| `!==` | inequality |
| `&&` | and |
| `\|\|` | or |
| `!` | not |

In addition to *if-else* structures, we have a *switch* structure.

```
switch{ expr )
{
    case 'option1':
        stmts ;
        break ;
    case 'option2':
        stmts ;
        break ;
    default:
        stmts ;
        break ;
}
```

## 29.6 Functions

```
var fctname = function( args )
{
    stmts ;
};
```

## 29.7 Classes

```
var phonebookEntry = {} ;
phonebookEntry.name = "Connor the Great" ;
phonebookEntry.number = "666.666.6666" ;
phonebookEntry.phone = function()
{
    console.log( "Calling '+phonebookEntry.name+'...' )
};
```

We can also do it this way; this is called *constructor notation*, as we are using `new` keyword.

```
var obj = new Object() ;
```

We can assign attributes like a Python dict, too! This is called *literal notation.*

```
var me = { christian_name:'Connor', age:29 } ;
me['surname'] = 'the Great' ;
```

We can create functions for use across mutliple classes using the `this` keyword. Here is an example. We will create a function, `setAge`, and then attach it to an object, `bob`.

```
var setAge = function( newAge )
{
    this.age = newAge ;
};
var bob = new Object() ;
bob.age = 30 ;        // give bob an age attribute
bob.setAge = setAge ; // pass setAge to bob
bob.setAge(50) ;      // use setAge for bob
```

A *constructor* is used to create an object. *Object()* is a built-in constructor. Here is an example of a custom constructor.

```
function Person( name, age )
{
    this.name = name ;
    this.age = age ;
}
var bob = new Person( "Bob Smith", 30 ) ;
```

We can include methods in constructors as so,

```
function Rectangle( length, width )
{
    this.length = length ;
    this.width = width ;
    this.calcArea = function()
    {
        return this.length * this.width ;
    };
};
```

## 29.8 Pop-Ups

These are some handy-dandy pop-up dialogs. The alert pop-up tells you something and you click, "Ok."

```
alert( msg ) ;
```

The confirm pop-up lets you click, "Ok," or, "Cancel."

```
confirm( msg )
```

The prompt pop-up gives you a field to fill in, and an, "Ok," and, "Cancel," button. Your entry will be returned to `response`.

```
response = prompt( msg )
```

# Chapter 30

# jQuery

(top)

## 30.1 Introduction

Recall that an HTML document is structured according to the *Document Object Model*, or DOM. The DOM consists of every element on the page, laid out in a hierarchical way that reflects the way the HTML document is ordered.

## 30.2 A Simple Example

Here is the HTML document, `index.html`. We point to the CSS and the jQuery documents in the *HEAD* element. This produces a red square that fades away as soon as the page loads.

```
<!DOCTYPE HTML>
<HTML>
    <HEAD>
        <TITLE></TITLE>
        <LINK REL="stylesheet" TYPE="text/css" href="stylesheet.css"/>
        <SCRIPT TYPE="text/javascript" SRC="script.js"></SCRIPT>
    <HEAD>
    <BODY>
        <DIV></DIV>
    </BODY>
</HTML>
```

Next, we have the CSS stylesheet, `stylesheet.css`, that styles the HTML, i.e., makes the square and colors it red.

```
div {
    height: 100px ;
    width: 100px ;
    background-color: #FF0000 ;
    border-radius: 5px ;
}
```

Finally we have the script, `script.js`. I think that `$(document)` grabs the HTML document, and then `$('div')` grabs the *div* tag(s) and then applies the jQuery `fadeOut()` function to them.

```
$(document).ready(function() {
    $('div').fadeOut(1000) ;
});
```

## 30.3 DIV tags with Separate Classes

Suppose we have two `DIV` tags, with separate IDs. We can apply the jQuery `fadeOut()` function to only one ID in much the same way. Here is out HTML, index.html

```
<!DOCTYPE HTML>
<HTML>
  <HEAD>
    <TITLE></TITLE>
    <LINK REL="stylesheet" TYPE="text/css" HREF="stylesheet.css"/>
    <SCRIPT TYPE="text/javascript" SRC="script.js"></SCRIPT>
  </HEAD>
  <BODY>
    <DIV ID="ready">I'm ready.</DIV>
    <DIV ID="notready">I'm not ready.</DIV>
  </BODY>
</HTML>
```

Here is the CSS stylesheet.

```
div {
    height: 100px ;
    width: 100px ;
    border-radius: 5px ;
    display: inline-block ;
    text-align: center ;
    vertical-align: middle ;
    font-family: Verdana, Arial, San-Serif ;
    margin-right: 5px
}

#ready {
    background-color: red ;
    color: white ;
}

#notready {
    background-color: red ;
    color: white ;
}
```

And, finally, here is the jQuery script, `script.js`. Note how we access specifically the second *DIV* tag with the *notready* ID:

```
$(document).ready(function() {
  $('#notready').fadeOut(1000);
});
```

## 30.4 Syntax

jQuery syntax looks all the same, usually.

```
$(document).ready( function() {
  $(thingToClick).event( function() {
    $(thingToAffect).effect();
  });
});
```

On the first line, you get the document ready. On the second line, you grab the appropriate HTML element you want the user to click on. On the third line, you produce whatever effect is supposed to occur. The second line is optional.

## 30.5 A Sliding Header

Suppose we have in our *BODY* an element that looks like:

```
<DIV><STRONG>Hai, Imma header!</STRONG></DIV>
```

Then we can animate this using `script.js`:

```
$(document).ready( function() {
  $('div').slideDown('slow') ;
});
```

## 30.6 Buttons That Fade In and Out

Here is some HTML:

```
<!DOCTYPE HTML>
<HTML>
  <HEAD>
    <TITLE></TITLE>
    <LINK REL="stylesheet" TYPE="text/css" HREF="stylesheet.css"/>
    <SCRIPT TYPE="text/javascript" SRC="script.js"></SCRIPT>
  </HEAD>
  <BODY>
    <DIV><STRONG>Click me!</STRONG></DIV>
  </BODY>
</HTML>
```

Next, we have our CSS stylesheet:

```
DIV {
    height: 100px ;
    width: 100px ;
    border-radius: 5px ;
    background-color: #69D2E7 ;
    text-align: center ;
    color: white ;
    font-family: Verdana, Arial, Sans-Serif ;
    opacity: 0.5 ;
}
```

And finally we have our jQuery:

```
$(document).ready( function() {
  $('div').mouseenter( function() {
    $('div').fadeTo( 'fast', 1.0 ) ;
  });
  $('div').mouseleave( function() {
    $('div').fadeTo( 'fast', 0.5 ) ;
  });
});
```

# Chapter 31

# D3

(top)

## 31.1 Introduction

D3 is a library that allows you to apply CSS styles and HTML attributes to data in an automated fashion in a web page using JavaScript.

## 31.2 A Simple Example

For this guy, download `d3.v3.js` and put it in a subdirectory named `d3/`. Put the following listing into a file named `index.html`

```
<!DOCTYPE HTML>
<HTML>

<HEAD>
<META CHARSET="utf-8">
<SCRIPT TYPE="text/javascript" SRC="d3/d3.v3.js"></SCRIPT>
</HEAD>

<BODY>
<SCRIPT TYPE="text/javascript">
  var dataset = [ 5, 10, 15, 20, 15 ] ;
  d3.select("BODY")  // select the BODY element
    .selectAll("P")  // select the (nonexistent) P elements in BODY
    .data( dataset ) // produces a `d` variable
    .enter()         // put data into the P elements
    .append("P")     // append new P elements
    // put text inside these P ele. using an anon. fct. using `d`
    .text( function(d) { return d; } );
</SCRIPT>
</BODY>

</HTML>
```

Then, run `python -m SimpleHTTPServer 5000` at the terminal, and point a web browser to `localhost:5000` and you should see the values from the variable `dataset`.

## 31.3 Styling Our Simple Example

We can style values over a certain threshold, say 15, by chaining another method to the `.text(..)` call,

```
.style( "color", function(d) {
    if( d < 15 ){
        return "red";
    } else {
        return "black";
    } } ) ;
```

This will turn values greater than 15 red.

## 31.4 Using Flask and a JSON File

Next, we will use the Flask framework to serve our page, and we will use data from a JSON file. First, we have our Python file named `app.py`

```
from flask import Flask, render_template, url_for
app = Flask(__name__) # I'm really not sure what this does yet

@app.route('/')
def index:
    return render_template('index.html')

if __name__=="__main__":
    port = 5000
    app.debug = True
    app.run( port=port )
```

The JSON file will go into a directory named `static/`. Then we put `index.html` in a directory named `templates/`.

```
<!DOCTYPE HTML>
<HTML>

<HEAD>
  <META CHARSET="utf-8">
  <SCRIPT SRC="http://d3js.org/d3.v3.min.js"></SCRIPT>
  <SCRIPT>
    function draw(data) {
        "use strict";
        d3.select("body")
        .append("ul")
        .selectAll("li")
        .data(data)
        .enter()
        .append("li")
        .text( function(d){
            return d.name + ": " + d.status;
            }
        );
    }
  </SCRIPT>
  <TITLE>MTA Data</TITLE>
</HEAD>

<BODY>
  <H1>MTA Availability Data</H1>
  <SCRIPT>
    d3.json("{{ url_for( 'static', filename='service_status.json') }}",draw);
  </SCRIPT>
</BODY>

</HTML>
```

We then serve the page by running `python app.py` at the command line, and pointing a browser to `localhost:5000`.

## 31.5 SVG Elements

When using SVG element tags, it's important to remember that they are case sensitive, and lower-case.

# Chapter 32

# Google Charts

(top)

## 32.1 Introduction

These tools will allow you to generate and render a number of canned, interactive SVG plots from within the browser. Older browsers are supported using VML.

## 32.2 A Simple Scatter Plot

```
<html>
  <head>
    <script type="text/javascript" src="https://www.google.com/jsapi"></script>
    <script type="text/javascript">
      google.load("visualization", "1", {packages:["corechart"]});
      google.setOnLoadCallback(drawChart);
      function drawChart() {
        var data = google.visualization.arrayToDataTable([
          ['Age', 'Weight'],
          [ 8,      12  ],
          [ 4,      5.5 ],
          [ 11,     14  ],
          [ 4,      5   ],
          [ 3,      3.5 ],
          [ 6.5,    7   ]
        ]);

        var options = {
          title: 'Age vs. Weight comparison',
          hAxis: {title: 'Age', minValue: 0, maxValue: 15},
          vAxis: {title: 'Weight', minValue: 0, maxValue: 15},
          legend: 'none'
        };

        var chart = new google.visualization.ScatterChart(document.getElementById('chart_div'));
        chart.draw(data, options);
      }
    </script>
  </head>
  <body>
    <div id="chart_div" style="width: 900px; height: 500px;"></div>
  </body>
</html>
```

# Chapter 33

# Financial Math

(top)

## 33.1 Introduction

The following is taken from, and follows the notational conventions of, S. Broverman's *Mathematics of Investment and Credit, 5th Ed.*, and Bowers, Gerber, Hickman, Jones, and Nesbit's *Actuarial Mathematics*.

## 33.2 Utility

The *Expected Value Principle* is the idea that a decision maker is indifferent between assuming a random loss $X$, and paying an amount $E(X)$ in order to be relieved of that possible loss. This is inaccurate, because decision makers are often willing to pay more than the expected value of a loss, in order to avoid a catastrophic loss, i.e., a loss greater than their net worth.

## 33.3 Verbiage

An *insurer* is established to reduce the financial consequences of the damage of property. The insurer issues contracts, *policies*, that promise to pay the owner of a property a defined amount less than or equal to the financial loss if the property is damaged. The payment linked to the amount of the loss is called a *claim* payment. In return for the promise contained in the policy, the owner of the property, the *insured*, pays a consideration, the *premium*.

## 33.4 Compound Interest

An initial deposit of $C$ dollars at an interest rate $i$ per compounding period will have an *accumulated*, or *future*, value of $C(1+i)^n$ after $n$ compounding periods if the interest is reinvested at the end of each year as it is credited.

$$C(1+i)^n \tag{33.1}$$

## 33.5 Compound Interest with Non-level Rates

If each period has a different interest rate, $\{i_j\}_{j=1}^n$, then we have,

$$C(1+i_1)(1+i_2)\cdot\cdots\cdot(1+i_n) \tag{33.2}$$

## 33.6 Average Annual Rate

Suppose, as above, that periods have different interest rates, $\{i_j\}_{j=1}^n$, then we can calculate,

$$(1+i_1)(1+i_2)\cdot\cdots\cdot(1+i_n) = \prod_{j=1}^{n}(1+i_j) = k \tag{33.3}$$

## 33.7 Effective Annual Rate

The effective annual rate of interest earned by an investment during a 1 year period is the percentage change in the value of of the investment from the begining of the year to the end of the year, without regard to the behavior of the investment throughout the year.

An interest rate of $j$ per month, with interest credited monthly, has a 1 year growth factor of,

$$(1+j)^{12} = 1+i. \tag{33.4}$$

Here, $i$ is the *effective annual rate* of interest.

## 33.8 Equivalent Rates of Interest

Two rates are equivalent if they result in the same accumulated values at each point in time. In the discussion above, the monthly rate, $j$, is equivalent to the yearly rate, $i$.

## 33.9 Accumulation Factor

The pattern of growth of an investment takes various forms. We'll use the general expression $a(n)$ to represent the *accumulation factor* for an investment from time $t = 0$ to time $t = n$. The initial investment will be denoted $A(0)$. Then the *accumulated value* at time $t = n$ is given by,

$$A(n) = A(0)a(n) \tag{33.5}$$

# Chapter 34

# Electricity

(top)

## 34.1 Basic Terminology

### 34.1.1 Electric Current

The movement of free electrons.

### 34.1.2 Static Electricity

When two bodies have unequal charges and are near each other, an electric force is exerted between them. If they are not connected by a conductor, current cannot flow and their charges cannot equalize. The electric force between them is called an *electrostatic force*.

### 34.1.3 Electric Field

The space between and around charged bodies in which their influence is felt. The strength of this force field diminishes as distance increases. Also called the *electrostatic field*, *force field*, or the *dielectric field*.

### 34.1.4 Magnetic Flux Density

Also called *magnetic induction*, and denoted by $\vec{B}$. A particle with an electric charge, $Q_0$, moving in a magnetic field, $\vec{B}$, with velocity, $v$, experiences the Lorentz force,

$$F = Q_0(v \times \vec{B} \tag{34.1}$$

A magnetic field, $\vec{B}$, has no divergence, i.e., $\div\vec{B} = 0$. It has no sources or sink because no changes in magnetic charges have ever been found. A field with no divergence is said to be *solenoidal* or *sourceless*. Incompressible fluids, like water, have velocity fields with no divergence, also. Fluid is neither created nor destroyed at any point.

### 34.1.5 Magnetic Field

Also called the *auxillary magnetic field*, *magnetic intensity*, *magnetic strength*, or simply, the *magnetizing field*, and is denoted by $\vec{H}$.

$$\vec{H} = \frac{\vec{B}}{\mu_0} - M \tag{34.2}$$

Here, $M$ is the magnetization, $\mu_0$ is the permeability of free space. The $H$-field is measured in amperes per meter. On the outside, $\vec{H}$ and $\vec{B}$ ar ebasically the same, except for different units and magnitudes. The magnetic field, $\vec{B}$ depends on currents, while the auxillary field, $\vec{H}$ depends on currents and magnetization.

### 34.1.6 Electrostatic Lines of Force

These represent the direction and strength of the electrostatic force.

### 34.1.7 Voltage

Water pressure causes water to flow in pipes. Similarly, electrical "pressure," called voltage, electromotive force (emf), or a difference in electric potential, causes current to flow in a conductor.

### 34.1.8 Conventional Current and Electron Flow

*Conventional current* depicts electrons flowing from the positive to the negative terminals. In reality, the *electron flow* is from the negative to the positive terminals.

### 34.1.9 Amperes

Measure the rate of electron flow or current in terms of coulombs.

### 34.1.10 Coulomb

Measure of the quantity of electrons. Defined as one ampere per second.

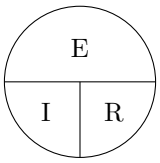
### 34.1.11 Resistance

One ohm is the resistance in a circuit that permits a steady current of one ampere–one coulomb per second–to flow when a steady emf of one volt is applied to the circuit.

### 34.1.12 Ohm's Law

The intensity of the current in amperes in any electrical circuit is equal to the difference in potential in volts across the circuit divided be the resistance in ohms of the circuit.

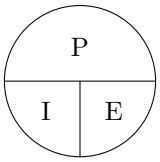
$$E = IR \tag{34.3}$$




### 34.1.13 Power

The rate at which work is done. The unit of power is the watt. Power is proportional to the voltage and the current flowing in a circuit.

$$P = IE \tag{34.4}$$




### 34.1.14 Energy

The ability to do work. Energy is expended when work is done, because it takes energy to maintain a force when that force does work. Energy (W) is equal to power (P) times the length of time (t) that the work is done.

$$W = Pt \tag{34.5}$$

### 34.1.15 Frequency

This is the cycles per second. It is measured in hertz (Hz). Let $T$ be the time it takes to complete one cycle, then $T$ is the *period*, and then we have the equation for the frequency:

$$f = \frac{1}{T} \tag{34.6}$$

### 34.1.16 Amplitude

This is the distance from the middle, or centerline of a wave to a peak. The peak-to-peak amplitude is twice the amplitude. For sinusoidal waves, the average and root mean square (RMS) values are given below.

$$V_{avg} = 0.637V_p \tag{34.7}$$

$$V_{RMS} = \frac{V_p}{\sqrt{2}} = 0.707V_p \tag{34.8}$$

### 34.1.17 Impedance

Electrical impedance is the opposition that a circuit presents to the passage of a current when voltage is applied. This opposition is the result of resistance, $R$, and reactance, $X$. Impedance is denoted by $Z$.

$$Z = R + jX \tag{34.9}$$

Impedance extends the concept of resistance to AC circuits, and possesses both magnitude and phase, unlike resistance, which is scalar and has only magnitude.

### 34.1.18 Reactance

The opposition of an circuit element to a change in electric current or voltage, due to that element's inductance or capacitance. Reactance is not exhibited under constant direct current. Reactance changes with the rate of change of voltage and current, and it is only constant for AC circuits of constant frequency.

Both capacitive reactance, $X_C$, and inductive reactance, $X_L$, contribute to the total reactance, $X$.

$$X = X_C + X_L \tag{34.10}$$

Capacitive reactance is non-positive, and inductive reactance is non-negative. If $X > 0$, then the element is said to be inductive, if $X < 0$, the element is said to be capacitive, and if $X = 0$ the element is said to be purely resistive. For complex impedance, resistance is the real part, and reactance is the imaginary part.

| | | | |
|---|---|---|---|
| Voltage | E | volt | v |
| Current | I | ampere | a |
| Resistance | R | ohm | Ω |
| Power | P | watt | w |
| Inductance | L | henry | h |
| Capacitance | C | farad | f |
| Frequency | f | hertz | Hz |

### 34.1.19 Inductance

The resistance to changes in current.

### 34.1.20 Capacitive Reactance

In an AC circuit, a capacitor acts like a resistor. This impedance is known as *capacitive reactance*, and it is measured in ohms. Capacitive reactance is inversely proportional to the product of the frequency, in hertz, and the capacitance, in farads.

$$X_C = \frac{1}{2\pi f C} \tag{34.11}$$

### 34.1.21 Inductive Reactance

In an AC circuit, an inductor's opposition to change in the AC current is known as *inductive reactance*, and it is measured in ohms. Inductive reactance is proportional to the product of the frequency, in hertz, and inductance, in henries.

$$X_L = 2\pi f L \tag{34.12}$$

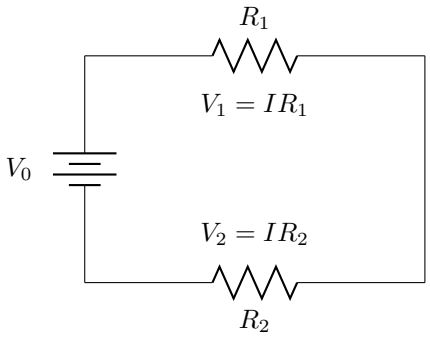
## 34.2 Coulomb's Law

Charged bodies attract or repel each other with a force directly proportional to the product of their charges and inversely proportional to the square of the distance between them.

$$F_{ij} = \frac{q_i q_j}{r_{ij}^2} \tag{34.13}$$

## 34.3 Kirchoff's Voltage Law

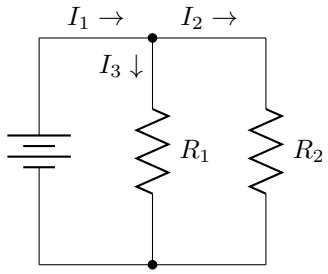
The sum of all voltages in a loop must equal zero.




$$V_0 = V_1 + V_2 = IR_1 + IR_2 = I(R_1 + R_2) \tag{34.14}$$

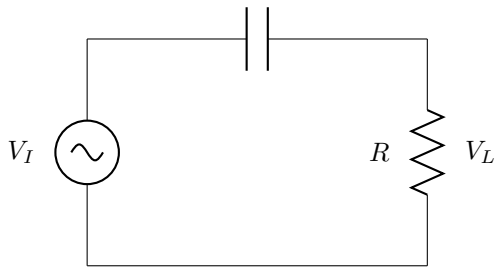
## 34.4 Kirchoff's Current Law (KCL)

The sum of all current entering and exiting a node must equal zero.

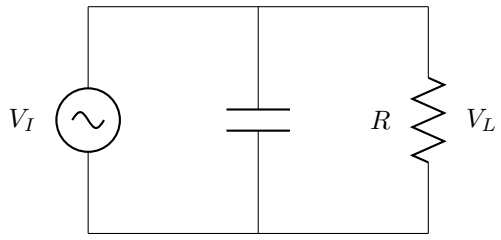



$$I_1 = I_2 + I_3 \tag{34.15}$$

## 34.5 Hi-Pass Circuit



## 34.6 Lo-Pass Circuit




## 34.7 ELI the ICE man

In an AC circuit, inductive reactance ($X_L$) causes voltage ($E$) to lead current ($I$), and capacitive reactance ($X_C$) causes current ($I$) to lead voltage ($V$).

## 34.8 Calculating Impedance in an RLC Circuit

Although resistance and reactance are both given in ohms, we cannot add these quantities as scalars, we must add them vectoraly. This is due to the fact that inductors and capacitors in AC circuits cause current and voltage to be out of phase.

The magnitude of the impedance is given by,

$$Z = \sqrt{R^2 + (X_L - X_C)^2}, \tag{34.16}$$

where $X_L$ is the *inductive reactance*, $X_C$ is the *capacitive reactance*, and $R$ is the resistance. The phase angle of the impedance is given by,

$$\tan\theta = \frac{X_L - X_C}{R}. \tag{34.17}$$

## 34.9 Apparent versus True Power

Power in DC is given by $P = IE$. In AC, this is called the *apparent power*, whereas *true power* is given by,

$$\text{True Power} = \text{Apparent Power} \cdot \cos\theta \tag{34.18}$$

The quantity $\cos\theta$ is called the *power factor* ($PF$) and it represents the amount of power dissipated in a circuit. Note that Apparent Power > True Power

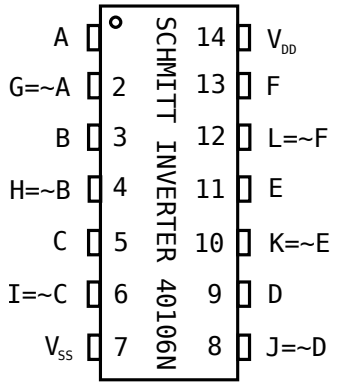
## 34.10 Diode

Has very low resistance in one direction, and very high resistance in the opposite direction.

## 34.11 Transistor

### 34.11.1 NPN

*Never points in.* When there is a positive current to the base, current flows from the emitter to the collector.




### 34.11.2 PNP

*Points in proudly.* When there is a negative current on the base, current flows from the collector to the emitter.

## 34.12 Schmitt Inverter 40106N

## 34.13 Maxwell's Equations

$$\text{curl } \vec{E} = \nabla \times \vec{E} = -\frac{\partial \vec{B}}{\partial t} \tag{34.19}$$

$$\text{curl } \vec{H} = \nabla \times \vec{H} = \vec{J} + \frac{\partial \vec{D}}{\partial t} \tag{34.20}$$

$$\text{div } \vec{D} = \nabla \cdot \vec{D} = \rho_e \tag{34.21}$$

$$\text{div } \vec{B} = \nabla \cdot \vec{B} = 0 \tag{34.22}$$

$$\vec{D} = \epsilon \vec{E} \tag{34.23}$$

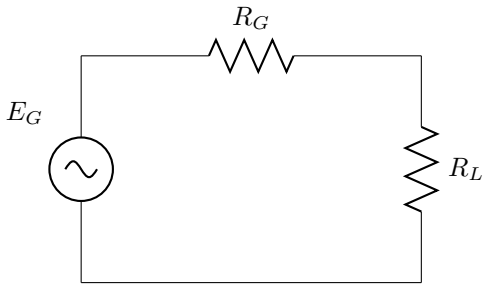
$$\vec{B} = \mu \vec{H} \tag{34.24}$$

$$\tag{34.25}$$

Here, $\vec{E}$ is the electric field, $\vec{D}$ is the electric displacement, $\vec{J}$ is the electric current density, $\vec{H}$ is the magnetic field, and $\vec{B}$ is the magnetic flux density. Also, $\rho_e$ is the charge density.

## 34.14 Impedance Matching

The simplest example of impedance matching is given by an electrical generator feeding a resistive load, $R_L$. Since all of the mechanical energy invested into the generator cannot be drawn from it as electrical energy, we attribute this internal resistance as $R_G$.




Early engineers first tried to minimize the internal resistance, $R_G$. Once this was accomplished, they tried to minimize the amount of power lost in the generator by varying the resistance of the load, $R_L$. What they found was that power lost in the generator was inversely related to the resistance of the load, $R_L$, and that power transferred to the load was proportional to the load resistance up to a point, where it decreases ever afterward.

At that point, when the impedance is matched, the power delivered to the load will be at a maximum, but the actual power being dissipated in the generator's internal resistance will be exactly the same as the that reaching the load. Therefore, *half* of the power from the generator is being dissipated by by $R_G$, and the other half is the load resistance, $R_L$. Thus, from a voltage transfer perspective, impedance matching represents a 6 dB loss.

### 34.14.1 Transmission Lines

*Impedance bridging* is unsuitable for RF connections because it allows power to be reflected back to the source from the boundary between the high and low impedances. This reflection causes a standing wave if there is a reflection at both ends of the transmission line. This wastes power, and can cause frequency-dependent loss, thus frequency matching is preferable.

In electrical systems where the length of the line is long compared to the wavelength of the signal, the impedances at each end of the line must be matched to the transmission line's characteristic impedance, $Z_0$, in order to prevent reflections of the signal at the ends of the line. RF applications usually have source and load impedances of 50 ohms.

THe voltage reflection coefficient for a wave moving from medium 1 to medium 2 is given by,

$$\Gamma_{1,2} = \frac{Z_2 - Z_1}{Z_2 + Z_1}. \tag{34.26}$$

By symmetry, voltage reflection coefficient for a wave moving from medium 2 to medium 1 is given by,

$$\Gamma_{2,1} = -\Gamma_{1,2} \tag{34.27}$$

The current reflection coefficient is the same as the voltage reflection coefficient, except for a change in sign. $\Gamma$ is understood to be the voltage reflection coefficient, unless otherwise specified.

# Chapter 35

# Signals

(top)

## 35.1 Terminology

*Signal*: a function that conveys information.

*Continuous-time signal*: defined at a continuum of times.

*Discrete-time signal*: has an independent variable takes discrete values. These arise by sampling a continuous-time signal, or a observing a discrete-time process.

*Digital signal*: time and amplitude are discrete.

*Analog signal*: time and amplitude are continuous.

*Continuous-time system*: both input and output are continuous-time signals.

*Discrete-time system*: both input and output are discrete-time signals.

*Analog systems*: both input and output are analog.

*Digital systems*: both input and output are digital.

*Sequence*: denoted as $x = \{x(n)\}$ or simply as $x(n)$.

*Unit-sample sequence*: also called the *discrete-time impulse* or simply as an *impulse*.

$$\delta(n) = \begin{cases} 0, & n \neq 0 \\ 1, & n = 0 \end{cases} \tag{35.1}$$

$$u(n) = \sum_{k=-\infty}^{\infty} \delta(k) \tag{35.2}$$

*Unit-step sequence*

$$u(n) = \begin{cases} 1, & n \geq 0 \\ 0, & n < 0 \end{cases} \tag{35.3}$$

$$\delta(n) = u(n) - u(n-1) \tag{35.4}$$

*Exponential sequence*: values take the form $a^n$, where $a \in \mathbb{R}$.

*Sinusoidal sequence*: values take the form $A\cos(\omega_0 n + \phi)$.. Here, $\omega_0$ is the frequency.

*Complex exponential sequence*: takes the form $\exp((\sigma + j\omega_0)n)$

*Periodicity*: a sequnce is periodic with period $N$ if $x(n) = x(n+N)$ for all $n$.

*Energy*:

$$\mathcal{E} = \sum_{n=-\infty}^{\infty} |x(n)|^2 \tag{35.5}$$

*Product and Sum*:

$$x \cdot y = \{x(n)y(n)\} \tag{35.6}$$

$$\alpha x = \{\alpha x(n)\} \tag{35.7}$$

$$x + y = \{x(n) + y(n)\} \tag{35.8}$$

*Shift*: $y(n) = x(n - n_0)$ where $n_0$ is an integer.
An arbitrary sequence can be expressed as,

$$x(n) = \sum_{k=-\infty}^{\infty} x(k)\delta(n - k) \tag{35.9}$$

### 35.1.1 Signal Energy

Let $f(t)$ be a signal. We would like a value that describes the size of a signal, taking into account its amplitude and duration. One measure of the size of a signal is the signal energy,

$$E_f = \int_{-\infty}^{\infty} |f(t)|^2 dt. \tag{35.10}$$

### 35.1.2 Signal Power

Sometimes, the amplitude of a signal $f(t)$ does not tend to zero as $|t| \to \infty$, so the signal energy is infinte. Another quantity is the time-average of the energy, if it exists. This is the power of the signal,

$$P_f = \lim_{T\to\infty} \frac{1}{T} \int_{-T/2}^{T/2} |f(t)|^2 dt. \tag{35.11}$$

A bell curve would have finite energy, and a sinusoid would have infinite energy, but finite power. A signal with finite energy is an *energy signal*, while a signal with finite non-zero power is a *power signal*. These are mutually exclusive.

If $f(t)$ is a voltage signal, then $E_f$ has units $V^2 s$, and $P_f$ has units $V^2$. If $f(t)$ is a current signal, then $E_f$ has units $A^2 s$, and $P_f$ has units $A^2$.

# Chapter 36

# Arduino

(top)

## 36.1 Initial Setup (for Windows)

Go to the Arduino website and download the software. Open the downloaded folder in `Downloads`, and double click on the Arduino application icon. This should install the software.

Go to `Control Panel` → `System and Security` → `System` → `Device Manager` and see if the Arduino has been recognized. If it hasn't, then you can install the appropriate drivers found in the Arduino folder.

After installation, double clicking on the Arduino application icon will start the Sketch application.

## 36.2 Programs

All Arduino programs come in two parts. There's the `setup()` that sets the pins and initializes things, and the `loop()` part that runs repeatedly after the initialization.

Edit → Compile → Reset → Upload

## 36.3 Built-in Functions

| | |
|---|---|
| `pinMode()` | set a pin as input or output |
| `digitalWrite()` | set a digital pin to HIGH ir LOW |
| `digitalRead()` | read a digital pin |
| `analogWrite()` | write an "analog" PWM value |
| `analogRead()` | read an analog pin |
| `delay()` | waits an amount of time |
| `millis()` | gets the current time |

## 36.4 Miscellaneous Notes

The longer part of the LED is the positive end, and the shorter part of the LED is the ground.

Knife switch (SPST) Single pole, single throw. One circuit is being broken. Toggle switch (SPDT) Single pole, double throw. Two circuits are being switched, simultaneously.

## 36.5 Servomotor

A motor and encoder combination that usually returns position and speed.

## 36.6 Stepper Motors

A brushless DC motor that divides a rotation into a number of equal steps. The motor can be commanded to move to one of these steps without any feedback sensor.

# Chapter 37

# Music

(top)