

An Open Source Multi-platform GIS Toolbox

An introduction to:

LandSerf, MeshLab, SketchUp
Quantum GIS, GRASS, GMT,
R/RStudio, iPython,
Google Earth, Paraview ...

... to own your computational processes.

NB: work in progress

A/Prof Patrice F Rey

Welcome to the World of UNIX

Unix is one of the oldest computer operating systems (OS). It is made of a collection of programs, the development of which started at Bell Laboratories in the 60's. UNIX is now at the core of many modern OS including Sun Solaris, Linux and MacOS X, each with its own Graphic User Interface (or GUI).

Unix versatility is such that a GUI is not enough to tap into Unix power. This is where the Shell - also called Terminal Window - comes about. While most users will be happy with their favorite UNIX GUI, computer scientists cannot live without a UNIX Shell.

Many Unix tutorials can be found on the Internet. Here we attach *Unix Tutorial for Beginners*, licensed under a *Creative in Common* license. This tutorial is very short, but it covers all that is necessary to start using UNIX programs.

<http://www.ee.surrey.ac.uk/Teaching/Unix/index.html>

UNIX INTRODUCTION

This session concerns UNIX, which is a common operating system. By operating system, we mean the suite of programs which make the computer work. UNIX is used by the workstations and multi-user servers within the school.

On X terminals and the workstations, X Windows provides a graphical interface between the user and UNIX. However, knowledge of UNIX is required for operations which aren't covered by a graphical program, or for when there is no X windows system, for example, in a telnet session.

The UNIX operating system

The UNIX operating system is made up of three parts; the kernel, the shell and the programs.

The kernel

The kernel of UNIX is the hub of the operating system: it allocates time and memory to programs and handles the filestore and communications in response to system calls.

As an illustration of the way that the shell and the kernel work together, suppose a user types `rm myfile` (which has the effect of removing the file **myfile**). The shell searches the filestore for the file containing the program `rm`, and then requests the kernel, through system calls, to execute the program `rm` on `myfile`. When the process `rm myfile` has finished running, the shell then returns the UNIX prompt `%` to the user, indicating that it is waiting for further commands.

The shell

The shell acts as an interface between the user and the kernel. When a user logs in, the login program checks the username and password, and then starts another program called the shell. The shell is a command line interpreter (CLI). It interprets the commands the user types in and arranges for them to be carried out. The commands are themselves programs: when they terminate, the shell gives the user another prompt (`%` on our systems).

The adept user can customise his/her own shell, and users can use different shells on the same machine. Staff and students in the school have the `tcsh` shell by default.

The `tcsh` shell has certain features to help the user inputting commands.

Filename Completion - By typing part of the name of a command, filename or directory and pressing the `[Tab]` key, the `tcsh` shell will complete the rest of the name automatically. If the shell finds more than one name beginning with those letters you have typed, it will beep, prompting you to type a few more letters before pressing the tab key again.

History - The shell keeps a list of the commands you have typed in. If you need to repeat a command, use the cursor keys to scroll up and down the list or type history for a list of previous commands.

Files and processes

Everything in UNIX is either a file or a process.

A process is an executing program identified by a unique PID (process identifier).

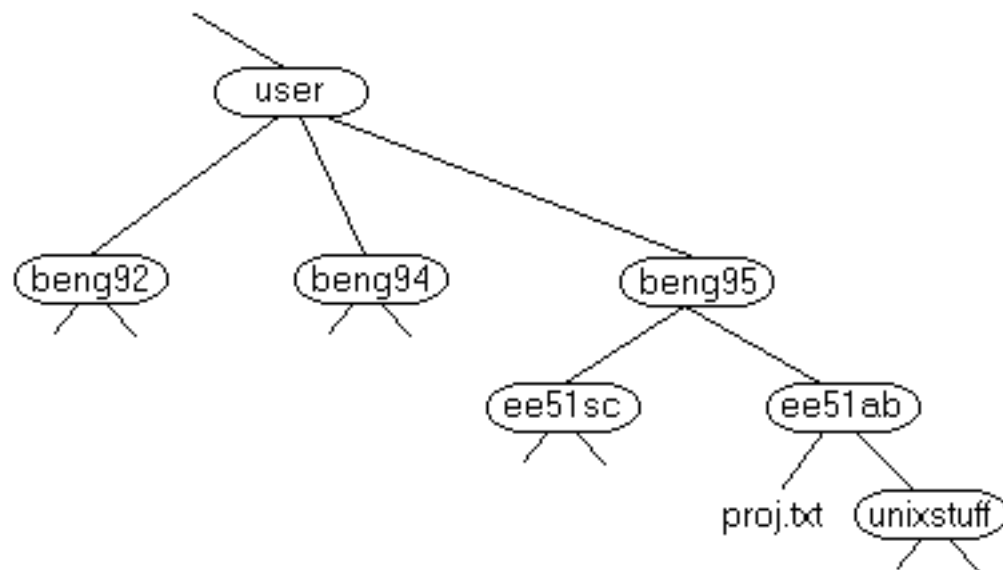
A file is a collection of data. They are created by users using text editors, running compilers etc.

Examples of files:

- a document (report, essay etc.)
- the text of a program written in some high-level programming language
- instructions comprehensible directly to the machine and incomprehensible to a casual user, for example, a collection of binary digits (an executable or binary file);
- a directory, containing information about its contents, which may be a mixture of other directories (subdirectories) and ordinary files.

The Directory Structure

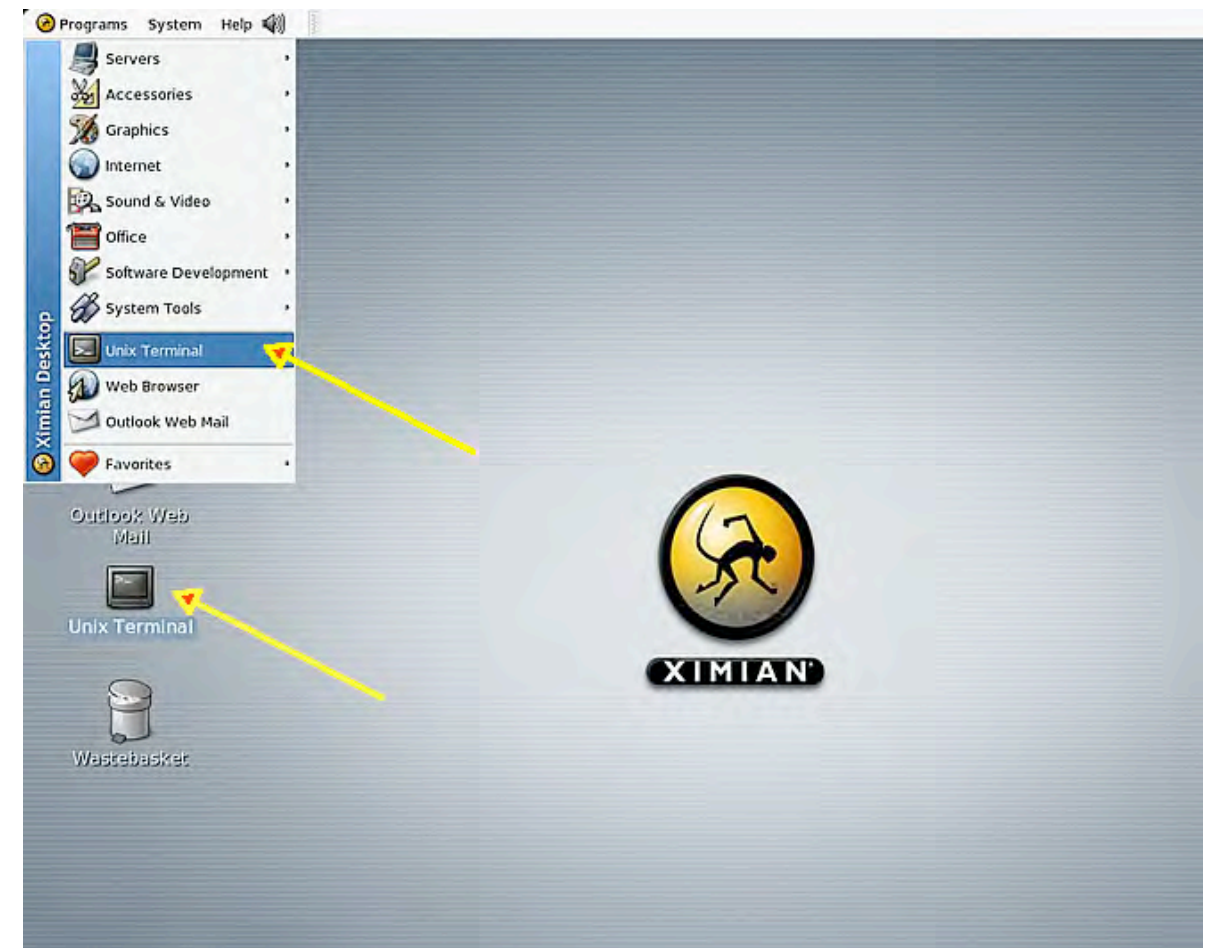
All the files are grouped together in the directory structure. The file-system is arranged in a hierarchical structure, like an inverted tree. The top of the hierarchy is traditionally called **root**.



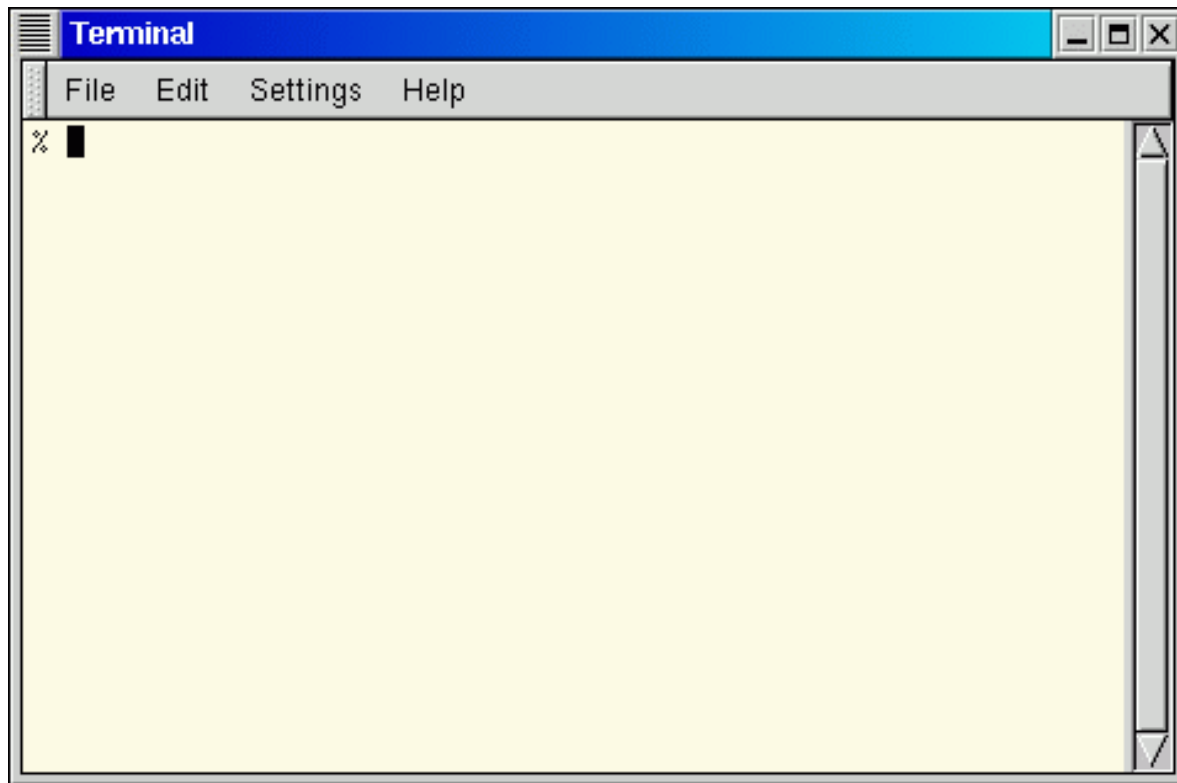
In the diagram above, we see that the directory ee51ab contains the subdirectory unixstuff and a file proj.txt

Starting an Xterminal session

To start an Xterm session, click on the Unix Terminal icon on your desktop, or from the drop-down menus



An Xterminal window will appear with a Unix prompt, waiting for you to start entering commands.



seperate files. Beware if copying files to a PC, since DOS and Windows do not make this distinction.

Typographical conventions

In what follows, we shall use the following typographical conventions:

- Characters written in **bold typewriter font** are commands to be typed into the computer as they stand.
- Characters written in *italic typewriter font* indicate non-specific file or directory names.
- Words inserted within square brackets [Ctrl] indicate keys to be pressed.

So, for example,

% ls *anydirectory* [Enter]

means "at the UNIX prompt %, type ls followed by the name of some directory, then press the key marked Enter"

Don't forget to press the [Enter] key: commands are not sent to the computer until this is done.

Note: UNIX is case-sensitive, so LS is not the same as ls.

The same applies to filenames, so myfile.txt, MyFile.txt and MYFILE.TXT are three

UNIX TUTORIAL ONE

1.1 Listing files and directories

ls (list)

When you first login, your current working directory is your home directory. Your home directory has the same name as your user-name, for example, ee91ab, and it is where your personal files and subdirectories are saved.

To find out what is in your home directory, type

```
% ls (short for list)
```

The `ls` command lists the contents of your current working directory.

There may be no files visible in your home directory, in which case, the UNIX prompt will be returned. Alternatively, there may already be some files inserted by the System Administrator when your account was created.

`ls` does not, in fact, cause all the files in your home directory to be listed, but only those ones whose name does not begin with a dot (`.`). Files beginning with a dot (`.`) are known as hidden files and usually contain important program configuration information. They are hidden because you should not change them unless you are very familiar with UNIX!!!

To list all files in your home directory including those whose names begin with a dot, type

```
% ls -a
```

`ls` is an example of a command which can take options: `-a` is an example of an option. The options change the behaviour of the command. There are online manual pages that tell you which options a particular command can take, and how each option modifies the behaviour of the command. (See later in this tutorial)

1.2 Making Directories

mkdir (make directory)

We will now make a subdirectory in your home directory to hold the files you will be creating and using in the course of this tutorial. To make a subdirectory called `unixstuff` in your current working directory type

```
% mkdir unixstuff
```

To see the directory you have just created, type

```
% ls
```

1.3 Changing to a different directory

cd (change directory)

The command `cd directory` means change the current working directory to '*directory*'. The current working directory may be thought of as the directory you are in, i.e. your current position in the file-system tree.

To change to the directory you have just made, type

```
% cd unixstuff
```

Type `ls` to see the contents (which should be empty)

Exercise 1a

Make another directory inside the `unixstuff` directory called **backups**

1.4 The directories `.` and `..`

Still in the `unixstuff` directory, type

```
% ls -a
```

As you can see, in the **unixstuff** directory (and in all other directories), there are two special directories called **(.)** and **(..)**

In UNIX, **(.)** means the current directory, so typing

```
% cd .
```

NOTE: there is a space between **cd** and the dot this means stay where you are (the **unixstuff** directory).

This may not seem very useful at first, but using **(.)** as the name of the current directory will save a lot of typing, as we shall see later in the tutorial.

(..) means the parent of the current directory, so typing

```
% cd ..
```

will take you one directory up the hierarchy (back to your home directory). Try it now.

Note: typing **cd** with no argument always returns you to your home directory. This is very useful if you are lost in the file system.

1.5 Pathnames

pwd (print working directory)

Pathnames enable you to work out where you are in relation to the whole file-system. For example, to find out the absolute pathname of your home-directory, type **cd** to get back to your home-directory and then type

```
% pwd
```

The full pathname will look something like this -

```
/a/fservb/fservb/fservb22/eebeng99/ee91ab
```

which means that **ee91ab** (your home directory) is in the directory **eebeng99** (the group directory), which is located on the **fservb** file-server.

Note:

```
/a/fservb/fservb/fservb22/eebeng99/ee91ab
```

can be shortened to

```
/user/eebeng99/ee91ab
```

Exercise 1b

Use the commands **ls**, **pwd** and **cd** to explore the file system.

(Remember, if you get lost, type **cd** by itself to return to your home-directory)

1.6 More about home directories and pathnames

Understanding pathnames

First type **cd** to get back to your home-directory, then type

```
% ls unixstuff
```

to list the contents of your **unixstuff** directory.

Now type

```
% ls backups
```

You will get a message like this -

```
backups: No such file or directory
```

The reason for this is, **backups** is not in your current working directory. To use a command on a file (or directory) not in the current working directory (the directory you are currently in), you must either **cd** to the correct directory, or specify its full pathname. To list the contents of your **backups** directory, you must type

```
% ls unixstuff/backups
```

~ (your home directory)

Home directories can also be referred to by the tilde **~** character. It can be used to specify paths starting at your home directory. So typing

```
% ls ~/unixstuff
```

will list the contents of your **unixstuff** directory, no matter where you currently are in the file system.

What do you think

```
% ls ~
```

would list?

What do you think

% ls ~/.

would list?

Summary

ls
list files and directories
ls -a
list all files and directories
mkdir
make a directory
cd *directory*
change to named directory
cd
change to home-directory
cd ~
change to home-directory
cd ..
change to parent directory
pwd
display the path of the current directory



M.Stonebank@surrey.ac.uk, © 9th October 2000

UNIX TUTORIAL TWO

2.1 Copying Files

cp (copy)

`cp file1 file2` is the command which makes a copy of **file1** in the current working directory and calls it **file2**

What we are going to do now, is to take a file stored in an open access area of the file system, and use the `cp` command to copy it to your `unixstuff` directory.

First, `cd` to your `unixstuff` directory.

```
% cd ~/unixstuff
```

Then at the UNIX prompt, type,

```
% cp /vol/examples/tutorial/science.txt .
```

(Note: Don't forget the dot (.) at the end. Remember, in UNIX, the dot means the current directory.)

The above command means copy the file **science.txt** to the current directory, keeping the name the same.

(Note: The directory `/vol/examples/tutorial/` is an area to which everyone in the department has read and copy access. If you are from outside the University, you can grab a copy of the file [here](#). Use 'File/Save As..' from the menu bar to save it into your **unixstuff** directory.)

Exercise 2a

Create a backup of your **science.txt** file by copying it to a file called **science.bak**

2.2 Moving files

mv (move)

`mv file1 file2` moves (or renames) **file1** to **file2**

To move a file from one place to another, use the `mv` command. This has the effect of moving rather than copying the file, so you end up with only one file rather than two.

It can also be used to rename a file, by moving the file to the same directory, but giving it a different name.

We are now going to move the file `science.bak` to your backup directory.

First, change directories to your `unixstuff` directory (can you remember how?).

Then, inside the **unixstuff** directory, type

```
% mv science.bak backups/
```

Type `ls` and `ls backups` to see if it has worked.

2.3 Removing files and directories

rm (remove), rmdir (remove directory)

To delete (remove) a file, use the `rm` command. As an example, we are going to create a copy of the **science.txt** file then delete it.

Inside your **unixstuff** directory, type

```
% cp science.txt tempfile.txt
```

```
% ls (to check if it has created the file)
```

```
% rm tempfile.txt
```

```
% ls (to check if it has deleted the file)
```

You can use the `rmdir` command to remove a directory (make sure it is empty first). Try to remove the **backups** directory. You will not be able to since UNIX will not let you remove a non-empty directory.

Exercise 2b

Create a directory called **tempstuff** using `mkdir` , then remove it using the `rmdir` command.

2.4 Displaying the contents of a file on the screen

clear (clear screen)

Before you start the next section, you may like to clear the terminal window of the previous commands so the output of the following commands can be clearly understood.

At the prompt, type

```
% clear
```

This will clear all text and leave you with the % prompt at the top of the window.

cat (concatenate)

The command `cat` can be used to display the contents of a file on the screen. Type:

```
% cat science.txt
```

As you can see, the file is longer than the size of the window, so it scrolls past making it unreadable.

less

The command `less` writes the contents of a file onto the screen a page at a time. Type

```
% less science.txt
```

Press the [space-bar] if you want to see another page, type [q] if you want to quit reading. As you can see, `less` is used in preference to `cat` for long files.

head

The `head` command writes the first ten lines of a file to the screen.

First clear the screen then type

```
% head science.txt
```

Then type

```
% head -5 science.txt
```

What difference did the -5 do to the head command?

tail

The `tail` command writes the last ten lines of a file to the screen.

Clear the screen and type

```
% tail science.txt
```

How can you view the last 15 lines of the file?

2.5 Searching the contents of a file

Simple searching using less

Using `less`, you can search through a text file for a keyword (pattern). For example, to search through **science.txt** for the word 'science', type

```
% less science.txt
```

then, still in `less` (i.e. don't press [q] to quit), type a forward slash [/] followed by the word to search

```
/science
```

As you can see, `less` finds and highlights the keyword. Type [n] to search for the next occurrence of the word.

grep (don't ask why it is called grep)

`grep` is one of many standard UNIX utilities. It searches files for specified words or patterns. First clear the screen, then type

```
% grep science science.txt
```

As you can see, `grep` has printed out each line containing the word science.

Or has it????

Try typing

```
% grep Science science.txt
```

The `grep` command is case sensitive; it distinguishes between Science and science.

To ignore upper/lower case distinctions, use the `-i` option, i.e. type

```
% grep -i science science.txt
```

To search for a phrase or pattern, you must enclose it in single quotes (the apostrophe symbol). For example to search for spinning top, type

```
% grep -i 'spinning top' science.txt
```

Some of the other options of `grep` are:

- v display those lines that do NOT match

- n precede each matching line with the line number

- c print only the total count of matched lines

Try some of them and see the different results. Don't forget, you can use more than one option at a time, for example, the number of lines without the words science or Science is

```
% grep -ivc science science.txt
```

display the first few lines of a file

```
tail file
```

display the last few lines of a file

```
grep 'keyword' file
```

search a file for keywords

```
wc file
```

count number of lines/words/characters in file

wc (word count)

A handy little utility is the `wc` command, short for word count. To do a word count on **science.txt**, type

```
% wc -w science.txt
```

To find out how many lines the file has, type

```
% wc -l science.txt
```

Summary

```
cp file1 file2
```

copy file1 and call it file2

```
mv file1 file2
```

move or rename file1 to file2

```
rm file
```

remove a file

```
rmdir directory
```

remove a directory

```
cat file
```

display a file

```
more file
```

display a file a page at a time

```
head file
```

UNIX TUTORIAL THREE

3.1 Redirection

Most processes initiated by UNIX commands write to the standard output (that is, they write to the terminal screen), and many take their input from the standard input (that is, they read it from the keyboard). There is also the standard error, where processes write their error messages, by default, to the terminal screen.

We have already seen one use of the `cat` command to write the contents of a file to the screen.

Now type `cat` without specifying a file to read

```
% cat
```

Then type a few words on the keyboard and press the [Return] key.

Finally hold the [Ctrl] key down and press [d] (written as ^D for short) to end the input.

What has happened?

If you run the `cat` command without specifying a file to read, it reads the standard input (the keyboard), and on receiving the 'end of file' (^D), copies it to the standard output (the screen).

In UNIX, we can redirect both the input and the output of commands.

3.2 Redirecting the Output

We use the `>` symbol to redirect the output of a command. For example, to create a file called **list1** containing a list of fruit, type

```
% cat > list1
```

Then type in the names of some fruit. Press [Return] after each one.

```
pear  
banana  
apple  
^D (Control D to stop)
```

What happens is the `cat` command reads the standard input (the keyboard) and the `>` redirects the output, which normally goes to the screen, into a file called **list1**

To read the contents of the file, type

```
% cat list1
```

Exercise 3a

Using the above method, create another file called **list2** containing the following fruit: orange, plum, mango, grapefruit. Read the contents of **list2**

The form `>>` appends standard output to a file. So to add more items to the file **list1**, type

```
% cat >> list1
```

Then type in the names of more fruit

```
peach  
grape  
orange  
^D (Control D to stop)
```

To read the contents of the file, type

```
% cat list1
```

You should now have two files. One contains six fruit, the other contains four fruit.

We will now use the `cat` command to join (concatenate) **list1** and **list2** into a new file called **biglist**. Type

```
% cat list1 list2 > biglist
```

What this is doing is reading the contents of **list1** and **list2** in turn, then outputting the text to the file **biglist**

To read the contents of the new file, type

```
% cat biglist
```

3.3 Redirecting the Input

We use the `<` symbol to redirect the input of a command.

The command `sort` alphabetically or numerically sorts a list. Type

```
% sort
```

Then type in the names of some vegetables. Press [Return] after each one.

```
carrot
```

```
beetroot
```

```
artichoke
```

```
^D (control d to stop)
```

The output will be

```
artichoke
```

```
beetroot
```

```
carrot
```

Using `<` you can redirect the input to come from a file rather than the keyboard. For example, to sort the list of fruit, type

```
% sort < biglist
```

and the sorted list will be output to the screen.

To output the sorted list to a file, type,

```
% sort < biglist > slist
```

Use `cat` to read the contents of the file **slist**

3.4 Pipes

To see who is on the system with you, type

```
% who
```

One method to get a sorted list of names is to type,

```
% who > names.txt
```

```
% sort < names.txt
```

This is a bit slow and you have to remember to remove the temporary file called `names` when you have finished. What you really want to do is connect the output of the `who` command directly to the input of the `sort` command. This is exactly what pipes do. The symbol for a pipe is the vertical bar `|`

For example, typing

```
% who | sort
```

will give the same result as above, but quicker and cleaner.

To find out how many users are logged on, type

```
% who | wc -l
```

Exercise 3b

`a2ps -Phockney textfile` is the command to print a postscript file to the printer hockney.

Using pipes, print all lines of **list1** and **list2** containing the letter 'p', sort the result, and print to the printer hockney.

[Answer available here](#)

Summary

```
command > file
```

redirect standard output to a file

```
command >> file
```

append standard output to a file

```
command < file
```

redirect standard input from a file

```
command1 | command2
```

pipe the output of command1 to the input of command2

```
cat file1 file2 > file0
```

concatenate file1 and file2 to file0

```
sort
```

sort data

```
who
```

list users currently logged in

```
a2ps -Pprinter textfile
```

print text file to named printer

```
lpr -Pprinter psfile
```

print postscript file to named printer

Exercise 3b

`a2ps -Phockney textfile` is the command to print a postscript file to the printer hockney.

Using pipes, print all lines of `list1` and `list2` containing the letter 'p', sort the result, and print to the printer hockney.

Answer

```
% cat list1 list2 | grep p | sort | a2ps -Phockney
```

UNIX TUTORIAL FOUR

4.1 Wildcards

The characters * and ?

The character `*` is called a wildcard, and will match against none or more character(s) in a file (or directory) name. For example, in your **unixstuff** directory, type

```
% ls list*
```

This will list all files in the current directory starting with **list....**

Try typing

```
% ls *list
```

This will list all files in the current directory ending with **....list**

The character `?` will match exactly one character.

So `ls ?ouse` will match files like **house** and **mouse**, but not **grouse**.

Try typing

```
% ls ?list
```

4.2 Filename conventions

We should note here that a directory is merely a special type of file. So the rules and conventions for naming files apply also to directories.

In naming files, characters with special meanings such as `/ * & %`, should be avoided. Also, avoid using spaces within names. The safest way to name a file is to use only alphanumeric characters, that is, letters and numbers, together with `_` (underscore) and `.` (dot).

File names conventionally start with a lower-case letter, and may end with a dot followed by a group of letters indicating the contents of the file. For example, all files consisting of C code may be named with the ending `.c`, for example, `prog1.c`. Then in order to list all files containing C code in your home directory, you need only type `ls *.c` in that directory.

Beware: some applications give the same name to all the output files they generate.

For example, some compilers, unless given the appropriate option, produce compiled files named **a.out**. Should you forget to use that option, you are advised to rename the compiled file immediately, otherwise the next such file will overwrite it and it will be lost.

4.3 Getting Help

On-line Manuals

There are on-line manuals which gives information about most commands. The manual pages tell you which options a particular command can take, and how each option modifies the behaviour of the command. Type `man` command to read the manual page for a particular command.

For example, to find out more about the `wc` (word count) command, type

```
% man wc
```

Alternatively

```
% whatis wc
```

gives a one-line description of the command, but omits any information about options etc.

Apropos

When you are not sure of the exact name of a command,

```
% apropos keyword
```

will give you the commands with keyword in their manual page header. For example, try typing

```
% apropos copy
```

Summary

*

match any number of characters

?

match one character

`man command`

read the online manual page for a command

`whatis command`

brief description of a command

`apropos keyword`

match commands with keyword in their man pages

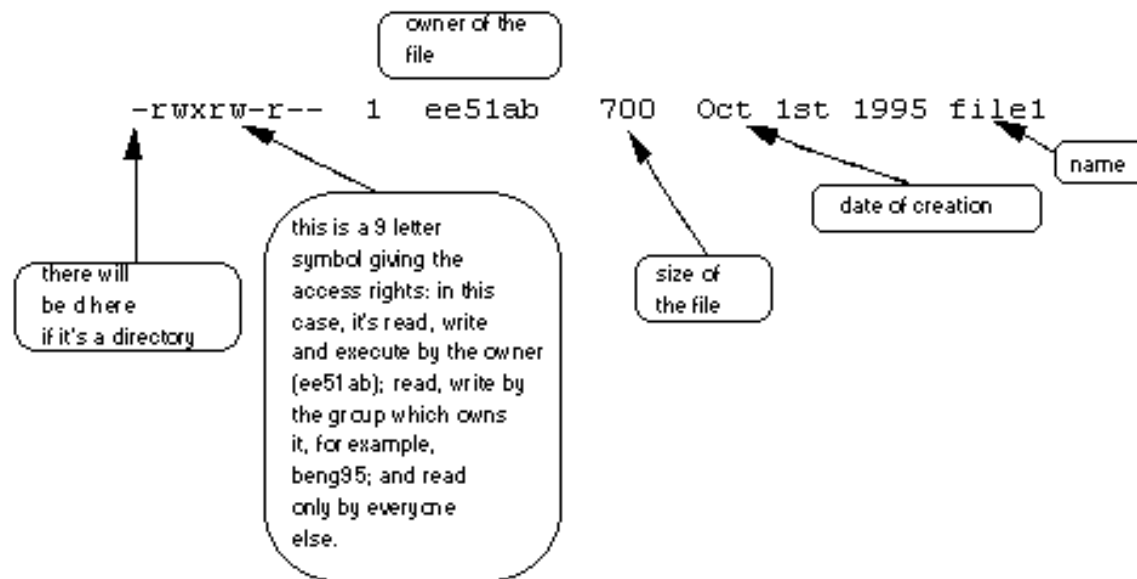
UNIX TUTORIAL FIVE

5.1 File system security (access rights)

In your unixstuff directory, type

```
% ls -l (l for long listing!)
```

You will see that you now get lots of details about the contents of your directory, similar to the example below.



Each file (and directory) has associated access rights, which may be found by typing `ls -l`. Also, `ls -lg` gives additional information as to which group owns the file (beng95 in the following example):

```
-rwxrw-r-- 1 ee51ab beng95 2450 Sept29 11:52 file1
```

In the left-hand column is a 10 symbol string consisting of the symbols d, r, w, x, -, and, occasionally, s or S. If d is present, it will be at the left hand end of the string, and indicates a directory: otherwise - will be the starting symbol of the string.

The 9 remaining symbols indicate the permissions, or access rights, and are taken as three groups of 3.

- The left group of 3 gives the file permissions for the user that owns the file (or directory) (ee51ab in the above example);
- the middle group gives the permissions for the group of people to whom the file (or directory) belongs (eebeng95 in the above example);
- the rightmost group gives the permissions for all others.

The symbols r, w, etc., have slightly different meanings depending on whether they refer to a simple file or to a directory.

Access rights on files.

- r (or -), indicates read permission (or otherwise), that is, the presence or absence of permission to read and copy the file
- w (or -), indicates write permission (or otherwise), that is, the permission (or otherwise) to change a file
- x (or -), indicates execution permission (or otherwise), that is, the permission to execute a file, where appropriate

Access rights on directories.

- r allows users to list files in the directory;
- w means that users may delete files from the directory or move files into it;
- x means the right to access files in the directory. This implies that you may read files in the directory provided you have read permission on the individual files.

So, in order to read a file, you must have execute permission on the directory containing that file, and hence on any directory containing that directory as a subdirectory, and so on, up the tree.

Some examples

```
-rwxrwxrwx
```

a file that everyone can read, write and execute (and delete).

```
-rw-----
```

a file that only the owner can read and write - no-one else can read or write and no-one has execution rights (e.g. your mailbox file).

5.2 Changing access rights

chmod (changing a file mode)

Only the owner of a file can use `chmod` to change the permissions of a file. The options of `chmod` are as follows

	Symbol Meaning
user	u
group	g
other	o
all	a
read	r
write (and delete)	w
execute (and access directory)	x
add permission	+
take away permission	-

For example, to remove read write and execute permissions on the file **biglist** for the group and others, type

```
% chmod go-rwx biglist
```

This will leave the other permissions unaffected.

To give read and write permissions on the file **biglist** to all,

```
% chmod a+rw biglist
```

Exercise 5a

Try changing access permissions on the file **science.txt** and on the directory **backups**

Use `ls -l` to check that the permissions have changed.

5.3 Processes and Jobs

A process is an executing program identified by a unique PID (process identifier). To see information about your processes, with their associated PID and status, type

```
% ps
```

A process may be in the foreground, in the background, or be suspended. In general the shell does not return the UNIX prompt until the current process has finished executing.

Some processes take a long time to run and hold up the terminal. Backgrounding a long process has the effect that the UNIX prompt is returned immediately, and other tasks can be carried out while the original process continues executing.

Running background processes

To background a process, type an **&** at the end of the command line. For example, the command `sleep` waits a given number of seconds before continuing. Type

```
% sleep 10
```

This will wait 10 seconds before returning the command prompt `%`. Until the command prompt is returned, you can do nothing except wait.

To run `sleep` in the background, type

```
% sleep 10 &
```

```
[1] 6259
```

The **&** runs the job in the background and returns the prompt straight away, allowing you to run other programs while waiting for that one to finish.

The first line in the above example is typed in by the user; the next line, indicating job number and PID, is returned by the machine. The user is notified of a job number (numbered from 1) enclosed in square brackets, together with a PID and is notified when a background process is finished. Backgrounding is useful for jobs which will take a long time to complete.

Backgrounding a current foreground process

At the prompt, type

```
% sleep 100
```

You can suspend the process running in the foreground by holding down the `[control]` key and typing `[z]` (written as **^Z**) Then to put it in the background, type

% bg

Note: do not background programs that require user interaction e.g. pine

5.4 Listing suspended and background processes

When a process is running, backgrounded or suspended, it will be entered onto a list along with a job number. To examine this list, type

% jobs

An example of a job list could be

[1] Suspended sleep 100

[2] Running netscape

[3] Running nedit

To restart (foreground) a suspended processes, type

% fg %jobnumber

For example, to restart `sleep 100`, type

% fg %1

Typing `fg` with no job number foregrounds the last suspended process.

5.5 Killing a process

kill (terminate or signal a process)

It is sometimes necessary to kill a process (for example, when an executing program is in an infinite loop)

To kill a job running in the foreground, type `^C` (control c). For example, run

% sleep 100

^C

To kill a suspended or background process, type

% kill %jobnumber

For example, run

% sleep 100 &

% jobs

If it is job number 4, type

% kill %4

To check whether this has worked, examine the job list again to see if the process has been removed.

ps (process status)

Alternatively, processes can be killed by finding their process numbers (PIDs) and using `kill PID_number`

% sleep 100 &

% ps

```
PID TT S TIME COMMAND
20077 pts/5 S 0:05 sleep 100
21563 pts/5 T 0:00 netscape
21873 pts/5 S 0:25 nedit
```

To kill off the process `sleep 100`, type

% kill 20077

and then type `ps` again to see if it has been removed from the list.

If a process refuses to be killed, uses the **-9** option, i.e. type

% kill -9 20077

Note: It is not possible to kill off other users' processes !!!

Summary

`ls -lag`

list access rights for all files

`chmod [options] file`

change access rights for named file

`command &`

run command in background

^C

kill the job running in the foreground

^Z

suspend the job running in the foreground

bg

background the suspended job

jobs

list current jobs

fg %1

foreground job number 1

```
kill %1  
kill job number 1  
ps  
list current processes  
kill 26152  
kill process number 26152
```



UNIX TUTORIAL SIX

Other useful UNIX commands

quota

All students are allocated a certain amount of disk space on the file system for their personal files, usually about 100Mb. If you go over your quota, you are given 7 days to remove excess files.

To check your current quota and how much of it you have used, type

```
% quota -v
```

df

The df command reports on the space left on the file system. For example, to find out how much space is left on the fileserver, type

```
% df .
```

du

The du command outputs the number of kilobytes used by each subdirectory. Useful if you have gone over quota and you want to find out which directory has the most files. In your home-directory, type

```
% du
```

compress

This reduces the size of a file, thus freeing valuable disk space. For example, type

```
% ls -l science.txt
```

and note the size of the file. Then to compress science.txt, type

```
% compress science.txt
```

This will compress the file and place it in a file called science.txt.Z

To see the change in size, type ls -l again.

To uncompress the file, use the uncompress command.

```
% uncompress science.txt.Z
```

gzip

This also compresses a file, and is more efficient than compress. For example, to zip science.txt, type

```
% gzip science.txt
```

This will zip the file and place it in a file called science.txt.gz

To unzip the file, use the gunzip command.

```
% gunzip science.txt.gz
```

file

file classifies the named files according to the type of data they contain, for example ascii (text), pictures, compressed data, etc.. To report on all files in your home directory, type

```
% file *
```

history

The C shell keeps an ordered list of all the commands that you have entered. Each command is given a number according to the order it was entered.

```
% history (show command history list)
```

If you are using the C shell, you can use the exclamation character (!) to recall commands easily.

```
% !! (recall last command)
```

```
% !-3 (recall third most recent command)
```

```
% !5 (recall 5th command in list)
```

```
% !grep (recall last command starting with grep)
```

You can increase the size of the history buffer by typing

```
% set history=100
```

UNIX TUTORIAL SEVEN

7.1 Compiling UNIX software packages

We have many public domain and commercial software packages installed on our systems, which are available to all users. However, students are allowed to download and install small software packages in their own home directory, software usually only useful to them personally.

There are a number of steps needed to install the software.

- Locate and download the source code (which is usually compressed)
- Unpack the source code
- Compile the code
- Install the resulting executable
- Set paths to the installation directory

Of the above steps, probably the most difficult is the compilation stage.

Compiling Source Code

All high-level language code must be converted into a form the computer understands. For example, C language source code is converted into a lower-level language called assembly language. The assembly language code made by the previous stage is then converted into object code which are fragments of code which the computer understands directly. The final stage in compiling a program involves linking the object code to code libraries which contain certain built-in functions. This final stage produces an executable program.

To do all these steps by hand is complicated and beyond the capability of the ordinary user. A number of utilities and tools have been developed for programmers and end-users to simplify these steps.

make and the Makefile

The `make` command allows programmers to manage large programs or groups of programs. It aids in developing large programs by keeping track of which portions

of the entire program have been changed, compiling only those parts of the program which have changed since the last compile.

The `make` program gets its set of compile rules from a text file called **Makefile** which resides in the same directory as the source files. It contains information on how to compile the software, e.g. the optimisation level, whether to include debugging info in the executable. It also contains information on where to install the finished compiled binaries (executables), manual pages, data files, dependent library files, configuration files, etc.

Some packages require you to edit the Makefile by hand to set the final installation directory and any other parameters. However, many packages are now being distributed with the GNU configure utility.

configure

As the number of UNIX variants increased, it became harder to write programs which could run on all variants. Developers frequently did not have access to every system, and the characteristics of some systems changed from version to version. The GNU configure and build system simplifies the building of programs distributed as source code. All programs are built using a simple, standardised, two step process. The program builder need not install any special tools in order to build the program.

The `configure` shell script attempts to guess correct values for various system-dependent variables used during compilation. It uses those values to create a **Makefile** in each directory of the package.

The simplest way to compile a package is:

1. `cd` to the directory containing the package's source code.
2. Type `./configure` to configure the package for your system.
3. Type `make` to compile the package.
4. Optionally, type `make check` to run any self-tests that come with the package.

5. Type `make install` to install the programs and any data files and documentation.
6. Optionally, type `make clean` to remove the program binaries and object files from the source code directory

The configure utility supports a wide variety of options. You can usually use the `--help` option to get a list of interesting options for a particular configure script.

The only generic options you are likely to use are the `--prefix` and `--exec-prefix` options. These options are used to specify the installation directories.

The directory named by the `--prefix` option will hold machine independent files such as documentation, data and configuration files.

The directory named by the `--exec-prefix` option, (which is normally a subdirectory of the `--prefix` directory), will hold machine dependent files such as executables.

7.2 Downloading source code

For this example, we will download a piece of free software that converts between different units of measurements.

First create a download directory

```
% mkdir download
```

[Download the software here](#) and save it to your new download directory.

7.3 Extracting the source code

Go into your **download** directory and list the contents.

```
% cd download
```

```
% ls -l
```

As you can see, the filename ends in `tar.gz`. The `tar` command turns several files and directories into one single tar file. This is then compressed using the `gzip` command (to create a `tar.gz` file).

First unzip the file using the `gunzip` command. This will create a `.tar` file.

```
% gunzip units-1.74.tar.gz
```

Then extract the contents of the tar file.

```
% tar -xvf units-1.74.tar
```

Again, list the contents of the **download** directory, then go to the **units-1.74** sub-directory.

```
% cd units-1.74
```

7.4 Configuring and creating the Makefile

The first thing to do is carefully read the **README** and **INSTALL** text files (use the `less` command). These contain important information on how to compile and run the software.

The units package uses the GNU configure system to compile the source code. We will need to specify the installation directory, since the default will be the main system area which you will not have write permissions for. We need to create an install directory in your home directory.

```
% mkdir ~/units174
```

Then run the configure utility setting the installation path to this.

```
% ./configure --prefix=$HOME/units174
```

NOTE:

The **\$HOME** variable is an example of an environment variable. The value of **\$HOME** is the path to your home directory. Just type

```
% echo $HOME
```

to show the contents of this variable. We will learn more about environment variables in a later chapter.

If configure has run correctly, it will have created a Makefile with all necessary options. You can view the Makefile if you wish (use the `less` command), but do not edit the contents of this.

7.5 Building the package

Now you can go ahead and build the package by running the `make` command.

```
% make
```

After a minute or two (depending on the speed of the computer), the executables will be created. You can check to see everything compiled successfully by typing

```
% make check
```

If everything is okay, you can now install the package.

```
% make install
```

This will install the files into the `~/units174` directory you created earlier.

7.6 Running the software

You are now ready to run the software (assuming everything worked).

```
% cd ~/units174
```

If you list the contents of the `units` directory, you will see a number of subdirectories.

```
bin
```

The binary executables

```
info
```

GNU info formatted documentation

```
man
```

Man pages

```
share
```

Shared data files

To run the program, change to the **bin** directory and type

```
% ./units
```

As an example, convert 6 feet to metres.

You have: 6 feet

You want: metres

```
* 1.8288
```

If you get the answer 1.8288, congratulations, it worked.

To view what units it can convert between, view the data file in the `share` directory (the list is quite comprehensive).

To read the full documentation, change into the **info** directory and type

```
% info --file=units.info
```

7.7 Stripping unnecessary code

When a piece of software is being developed, it is useful for the programmer to include debugging information into the resulting executable. This way, if there are

problems encountered when running the executable, the programmer can load the executable into a debugging software package and track down any software bugs.

This is useful for the programmer, but unnecessary for the user. We can assume that the package, once finished and available for download has already been tested and debugged. However, when we compiled the software above, debugging information was still compiled into the final executable. Since it is unlikely that we are going to need this debugging information, we can strip it out of the final executable. One of the advantages of this is a much smaller executable, which should run slightly faster.

What we are going to do is look at the before and after size of the binary file. First change into the **bin** directory of the units installation directory.

```
% cd ~/units174/bin
```

```
% ls -l
```

As you can see, the file is over 100 kbytes in size. You can get more information on the type of file by using the `file` command.

```
% file units
```

units: ELF 32-bit LSB executable, Intel 80386, version 1, dynamically linked (uses shared libs), not stripped

To strip all the debug and line numbering information out of the binary file, use the `strip` command

```
% strip units
```

```
% ls -l
```

As you can see, the file is now 36 kbytes - a third of its original size. Two thirds of the binary file was debug code !!!

Check the file information again.

```
% file units
```

units: ELF 32-bit LSB executable, Intel 80386, version 1, dynamically linked (uses shared libs), stripped

HINT: You can use the `make` command to install pre-stripped copies of all the binary files when you install the package.

Instead of typing `make install`, simply type `make install-strip`

UNIX TUTORIAL EIGHT

8.1 UNIX Variables

Variables are a way of passing information from the shell to programs when you run them. Programs look "in the environment" for particular variables and if they are found will use the values stored. Some are set by the system, others by you, yet others by the shell, or any program that loads another program.

Standard UNIX variables are split into two categories, environment variables and shell variables. In broad terms, shell variables apply only to the current instance of the shell and are used to set short-term working conditions; environment variables have a farther reaching significance, and those set at login are valid for the duration of the session. By convention, environment variables have UPPER CASE and shell variables have lower case names.

8.2 Environment Variables

An example of an environment variable is the `OSTYPE` variable. The value of this is the current operating system you are using. Type

```
% echo $OSTYPE
```

More examples of environment variables are

- `USER` (your login name)
- `HOME` (the path name of your home directory)
- `HOST` (the name of the computer you are using)
- `ARCH` (the architecture of the computers processor)
- `DISPLAY` (the name of the computer screen to display X windows)
- `PRINTER` (the default printer to send print jobs)
- `PATH` (the directories the shell should search to find a command)

Finding out the current values of these variables.

ENVIRONMENT variables are set using the `setenv` command, displayed using the `printenv` or `env` commands, and unset using the `unsetenv` command.

To show all values of these variables, type

```
% printenv | less
```

8.3 Shell Variables

An example of a shell variable is the history variable. The value of this is how many shell commands to save, allow the user to scroll back through all the commands they have previously entered. Type

```
% echo $history
```

More examples of shell variables are

- `cwd` (your current working directory)
- `home` (the path name of your home directory)
- `path` (the directories the shell should search to find a command)
- `prompt` (the text string used to prompt for interactive commands shell your login shell)

Finding out the current values of these variables.

SHELL variables are both set and displayed using the `set` command. They can be unset by using the `unset` command.

To show all values of these variables, type

```
% set | less
```

So what is the difference between `PATH` and `path` ?

In general, environment and shell variables that have the same name (apart from the case) are distinct and independent, except for possibly having the same initial values. There are, however, exceptions.

Each time the shell variables `home`, `user` and `term` are changed, the corresponding environment variables `HOME`, `USER` and `TERM` receive the same values. However, altering the environment variables has no effect on the corresponding shell variables.

`PATH` and `path` specify directories to search for commands and programs. Both variables always represent the same directory list, and altering either automatically causes the other to be changed.

8.4 Using and setting variables

Each time you login to a UNIX host, the system looks in your home directory for initialisation files. Information in these files is used to set up your working environment. The C and TC shells use two files called `.login` and `.cshrc` (note that both file names begin with a dot).

At login the C shell first reads `.cshrc` followed by `.login`.

.login is to set conditions which will apply to the whole session and to perform actions that are relevant only at login.

.cshrc is used to set conditions and perform actions specific to the shell and to each invocation of it.

The guidelines are to set `ENVIRONMENT` variables in the `.login` file and `SHELL` variables in the `.cshrc` file.

WARNING: NEVER put commands that run graphical displays (e.g. a web browser) in your `.cshrc` or `.login` file.

8.5 Setting shell variables in the .cshrc file

For example, to change the number of shell commands saved in the history list, you need to set the shell variable `history`. It is set to 100 by default, but you can increase this if you wish.

```
% set history = 200
```

Check this has worked by typing

```
% echo $history
```

However, this has only set the variable for the lifetime of the current shell. If you open a new xterm window, it will only have the default history value set. To PERMANENTLY set the value of `history`, you will need to add the `set` command to the `.cshrc` file.

First open the `.cshrc` file in a text editor. An easy, user-friendly editor to use is `nedit`.

```
% nedit ~/.cshrc
```

Add the following line AFTER the list of other commands.

```
set history = 200
```

Save the file and force the shell to reread its `.cshrc` file by using the shell source command.

```
% source .cshrc
```

Check this has worked by typing

```
% echo $history
```

8.6 Setting the path

When you type a command, your `path` (or `PATH`) variable defines in which directories the shell will look to find the command you typed. If the system returns a message saying "command: Command not found", this indicates that either the command doesn't exist at all on the system or it is simply not in your `path`.

For example, to run `units`, you either need to directly specify the `units` path (`~/units174/bin/units`), or you need to have the directory `~/units174/bin` in your `path`.

You can add it to the end of your existing `path` (the `$path` represents this) by issuing the command:

```
% set path = ($path ~/units174/bin)
```

Test that this worked by trying to run `units` in any directory other than where `units` is actually located.

```
% cd; units
```

HINT: You can run multiple commands on one line by separating them with a semicolon.

To add this `path` PERMANENTLY, add the following line to your `.cshrc` AFTER the list of other commands.

```
set path = ($path ~/units174/bin)
```



LandSerf MeshLab SketchUp

The accurate representation of geo-referenced 3D landscape models and their underlying geology is made possible by the availability of i/ free, easily available global digital elevation data, and ii/ portable (i.e. multi-platform) extensible freeware to combine, sample, process and visualize data, and translate them into various formats.

This chapter explains how to download and manipulate high-resolution digital elevation data (SRTM3) in preparation to build 3D landscape and geological models. We will use a workflow involving a suite of portable (Windows, Mac OS X, Linux), freeware: LandSerf, MeshLab, SketchUp and later on Google Earth.



SRTM digital elevation data

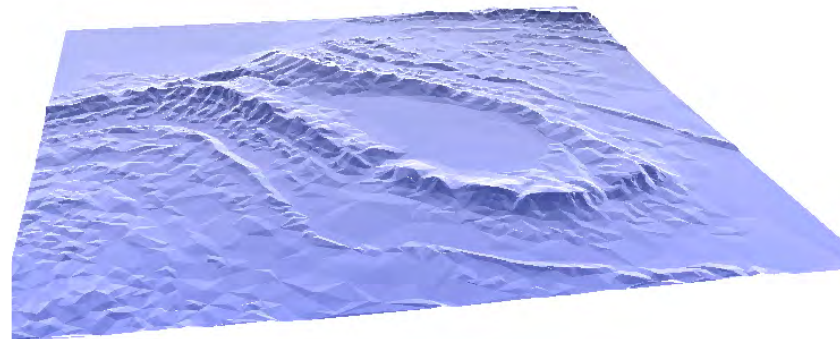
READ AT THE SOURCE ...

1. <http://www2.jpl.nasa.gov/srtm/>

Read about SRMT3 version 2.1:

2. http://dds.cr.usgs.gov/srtm/version2_1/SRTM3/

3. <http://srtm.csi.cgiar.org/>



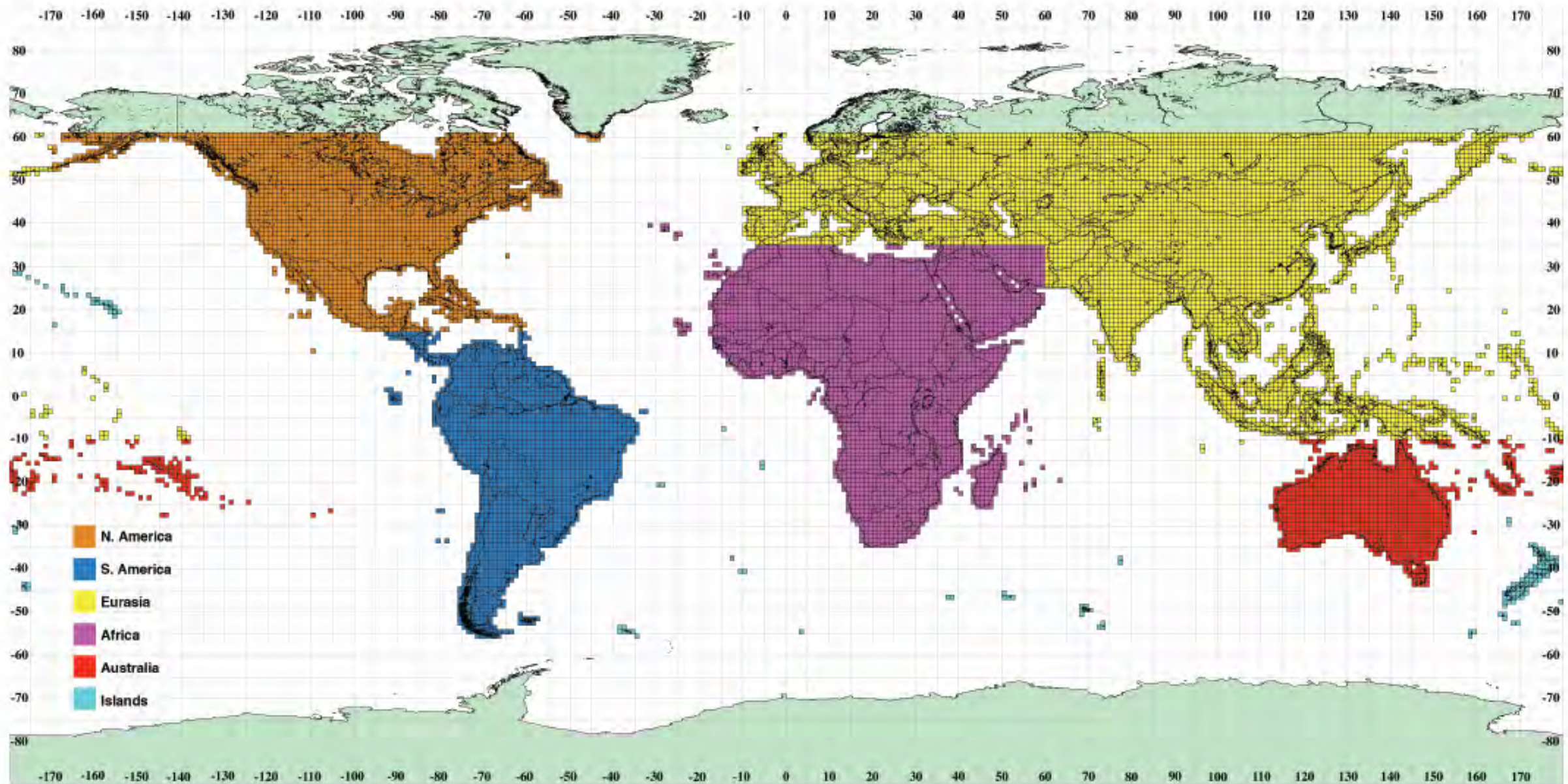
Over 11 days in 2000, the Shuttle Radar Topography Mission (SRTM) gathered high resolution elevation data on a near global scale. Since then, a number of versions of the dataset have been released involving different resolution and levels of processing.

SRMT1 has a resolution of 1 arc second covering approximately 30 m at the equator. At this resolution only the US dataset is publicly available (http://dds.cr.usgs.gov/srtm/version2_1/SRTM1/).

SRMT3 has a resolution 3 arc second covering approximately 90 m at the equator. In Version 2.1, SRMT3 data has been reprocessed using full resolution data (1x1 arc second) averaged over 3x3 arc second tiles. SRMT3 is publicly available at: http://dds.cr.usgs.gov/srtm/version2_1/SRTM3/

In SRTM30, SRTM3 data are averaged over 30 arc-second and combined with GTOPO30 data to extend the global DEM beyond the 60.25 degrees north latitude. This dataset can be accessed here: http://dds.cr.usgs.gov/srtm/version2_1/. SRTM30 Plus is a version including the bathymetry: ftp://topex.ucsd.edu/pub/srtm30_plus/

SRTM data are delivered in 1°x1° tile. The name of SRTM tiles (eg. S22E118) refer to the longitude and latitude of the southwest corner, which is centered on the 90X90 m data sample. Elevations are in meters and lat-long referenced to the WGS84/EGM96 geoid.



SRMT3 dataset. Each tile covers 1x1 degree. Source: <http://www2.jpl.nasa.gov/srtm/>

SRMT3 dataset covering Australia can be downloaded here: http://dds.cr.usgs.gov/srtm/version2_1/SRTM3/Australia/

NB: ASTER GDEM offers the world at 30 m resolution. <http://gdem.ersdac.jspacesystems.or.jp/search.jsp>

LandSerf

LANDSERF

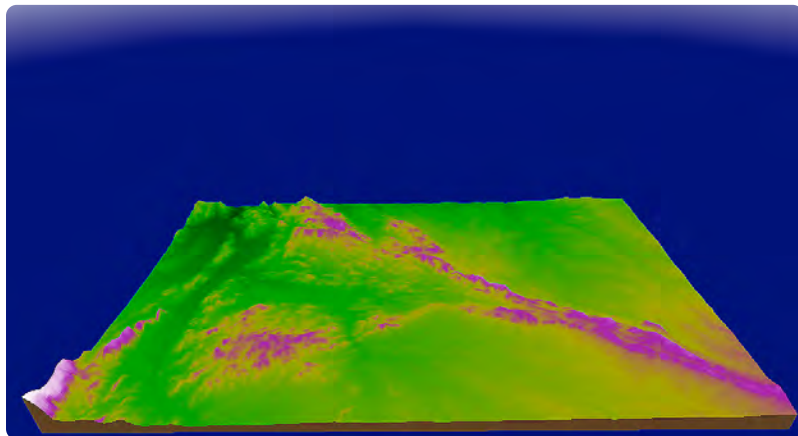
1. LandSerf Tutorial:

<http://www.soi.city.ac.uk/~jwo/landserf/landserf230/doc/tutorial/index.html>

2. LandSerf import-export format:

<http://www.soi.city.ac.uk/~jwo/landserf/landserf230InfoVis/doc/howto/fileformats.html>

LandSerf images



SRTM3 data from the Marble Bar township, East Pilbara, Western Australia

• • • • •

LandSerf can easily display, combine, sub-sample, process and re-format SRTM3 data.

Developed by Prof Jo Wood (City University London), LandSerf is a freely available Geographical Information System (GIS) for the visualization and analysis of surfaces. Applications include visualization of landscapes; geomorphological analysis; GIS file conversion; map output; surface modeling and many others. It runs on any platform that supports the Java Runtime Environment (Windows, MacOSX, Unix, Linux etc.). LandSerf software and documentation can be downloaded for free at LandSerf.org.

1/ unzip one of your SRTM3.hgt.zip file. You can download some of these files at the following link: http://dds.cr.usgs.gov/srtm/version2_1/SRTM3/

2/ In LandSerf open a srtm3.hgt file. It will load automatically into LandSerf.

3/ To extract a smaller surface area from the SRTM file click on the **Edit** menu item and select **Edit raster**, move the edit raster window on the side if necessary, and click-and-drag to select a portion of the raster. This will update the lat-long coordinates in the Edit raster window. In the Edit raster window, click on the field **Extract subset** and then click on **OK** at the bottom of the window. A new raster thumbnail will appear on the left. At this stage you can close the original raster by clicking on it and selecting **Close raster** in the **File** menu item.

The SRTM data are georeferenced using latitude and longitude values. However, much of the surface processing and 3D modeling is done using a planar and homogeneous metric coordinate system (the coordinates x, y, z and in meters). Therefore, we need to re-project the SRTM data in a Universal Transverse Mercator (UTM) projection, which is a metric georeference framework.

3/ In LandSerf, to translate lat-long into UTM coordinates click: **Transform > Reproject**, a new window will appear. In the **New Projection** drop down menu select **UTM** and click **OK**; a new window appears, which will allow you to enter the spatial resolution of your data. Since the resolution of SRTM3 data is 90 m, in the **Resolution** section of the **Set raster dimensions** window enter 90 in both E-W Res and N-S Res fields, and click OK to validate. A new raster with a UTM projection will be created.

4/ Click on the thumbnail and save this file in a .wrl format. This is a format readable in MeshLab.

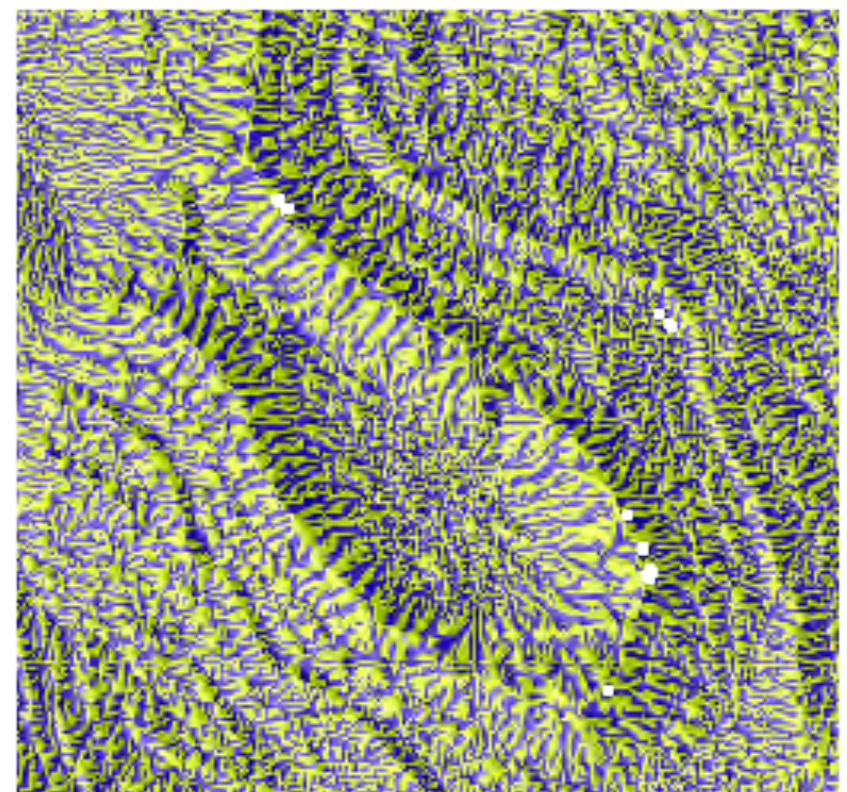
You may want to spend a bit more time to explore LandSerf:

In **Edit**, play with the coloring scheme.

In **Transform**, play with the option **DEM to contours**.

In **Display**, select **Relief**, then in **Configure** play with **Shaded relief**.

Images on the right represent a landscape in the Flinders Ranges (South Australia). The first image shows the relief while the second shows surface features extracted in LandSerf. The grey areas (very few on this image) represent planar regions, blue areas represent channels and yellow ridges.

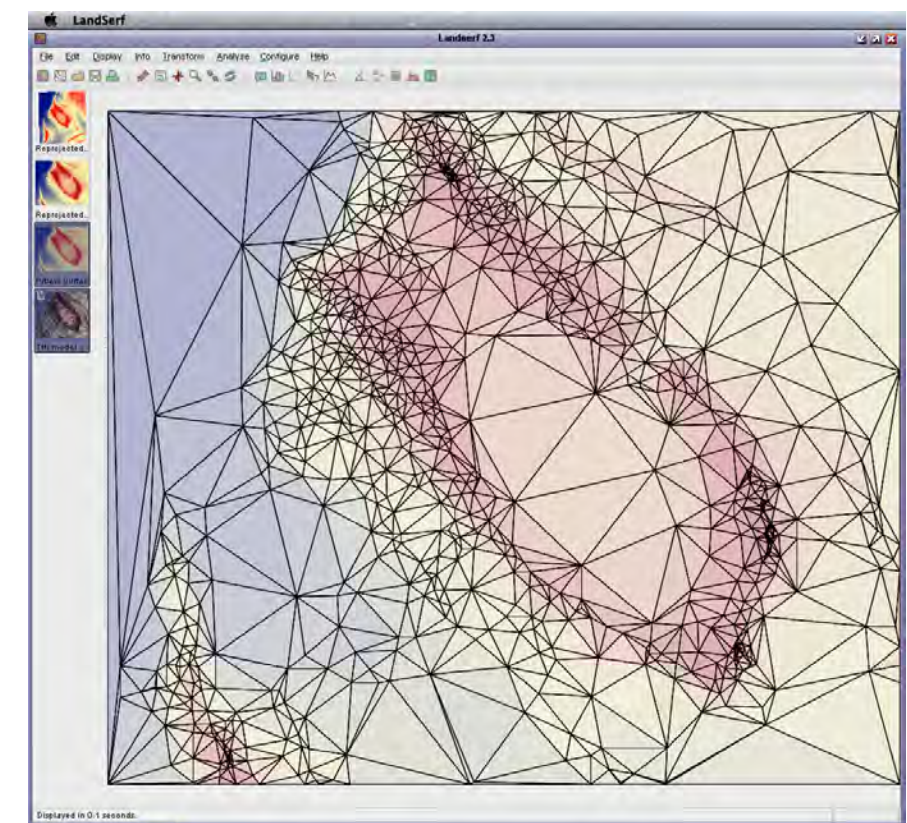
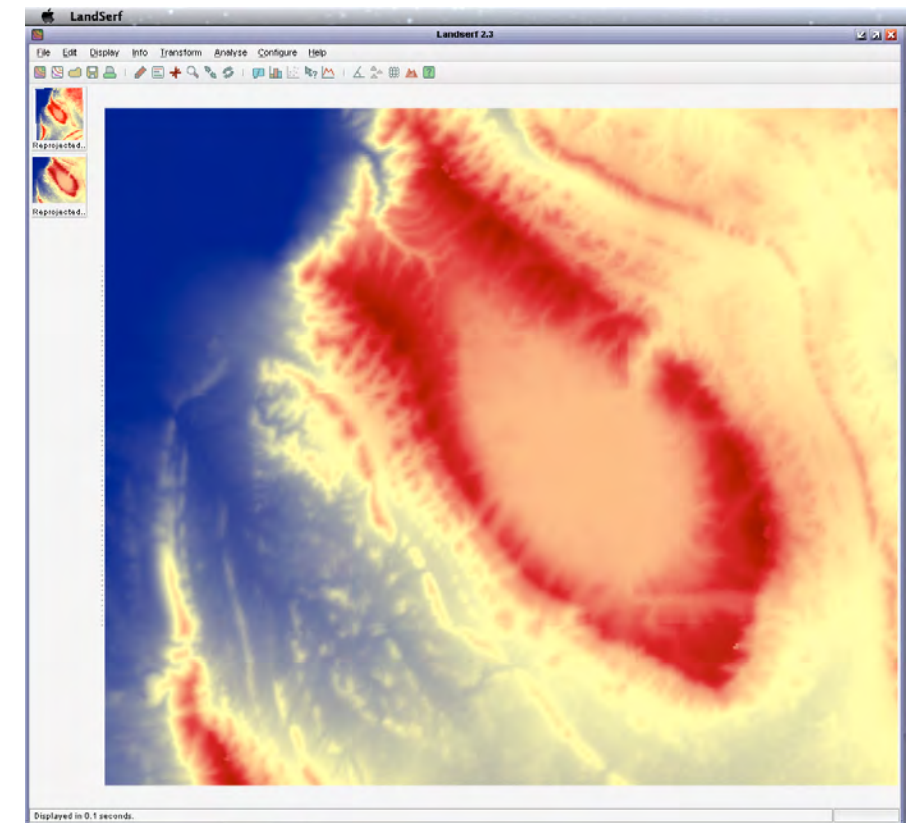
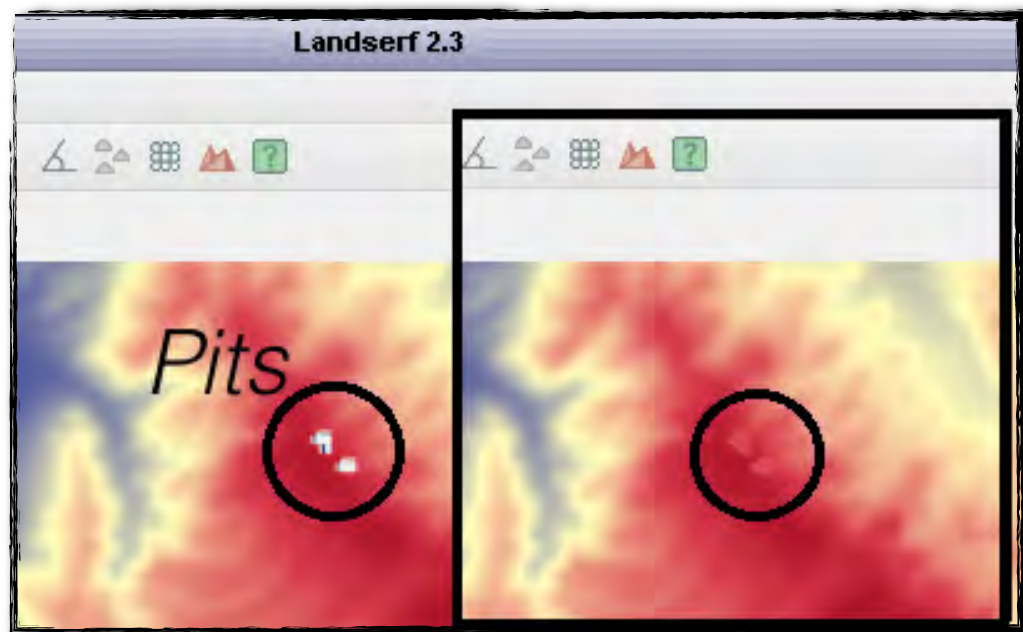


Before going further, we need to clean the “pits” that may exist in the SRTM dataset. A pit is a cell with no recorded elevation (see below for before and after pits removal).

5/ Select the UTM raster and click **Analyse > Pit removal**, and on the new window select **Infilling**. Two new rasters will appear on the thumbnail section of LandSerf window. One is the pitless raster and the other shows the removed pits.

To control the resolution of our .wrl file we generate a TIN mesh (Triangulated Irregular Network also known as Delaunay mesh) see image on the bottom right ...

6/ Select the pitless raster and click **Transform > DEM to TIN**, a window called **DEM to TIN conversion** opens. Enter the number of cells (typically few hundred to a few thousand), the higher the number the better the resolution. Save the TIN vector map in a .wrl format for further processing into MeshLab.



MeshLab

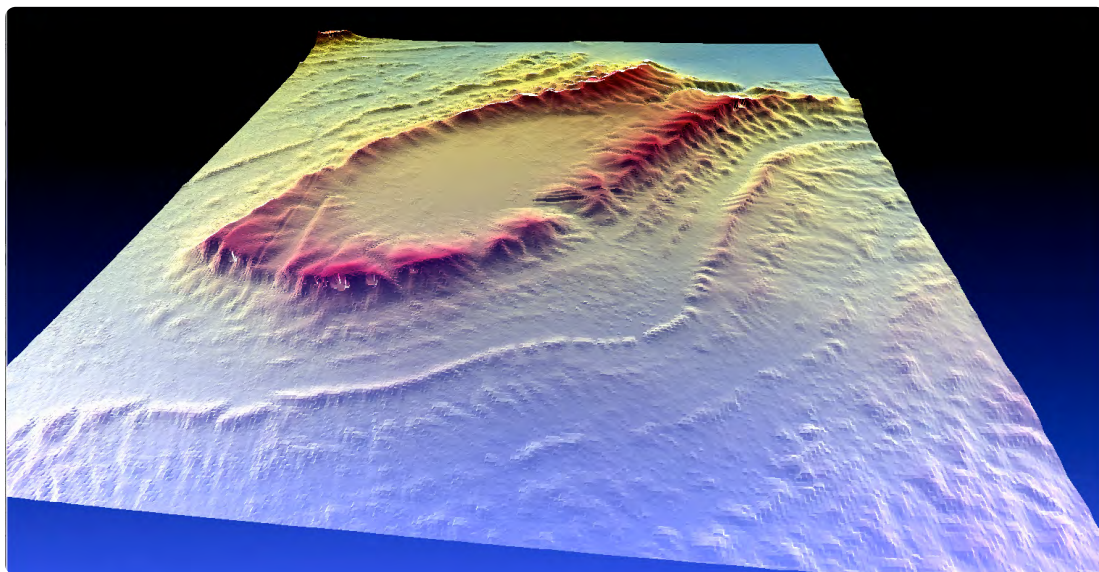
MESHLAB

1. Official website:

<http://meshlab.sourceforge.net>

2. Tutorial:

<http://sourceforge.net/apps/mediawiki/meshlab/index.php?>



Because LandSerf can't export UTM data to Google kml format (which requires lat-long), or any format compatible with SketchUp, we will use MeshLab to translate LandSerf UTM files. MeshLab is an open source, portable and extensible (via plugins) system for the processing and editing of unstructured 3D triangular meshes. It is developed at the Visual Computing Lab of Institute of the National Research Council of Italy. MeshLab can import and export 3D meshes in a large number of formats. We will use MeshLab to transform LandSerf outputs into input files that can be imported into SketchUp.

1/ In MeshLab, File > Import Mesh ... and open a LandSerf.wrl file.

2/ Then File > Export Mesh ... and save this file in a .3ds format, compatible with the standard (free) version of SketchUp.

NB: The image on the left is a high-resolution TIN mesh produced in LandSerf and rendered in MeshLab using **render > shaders > xray.gdp**

SketchUp

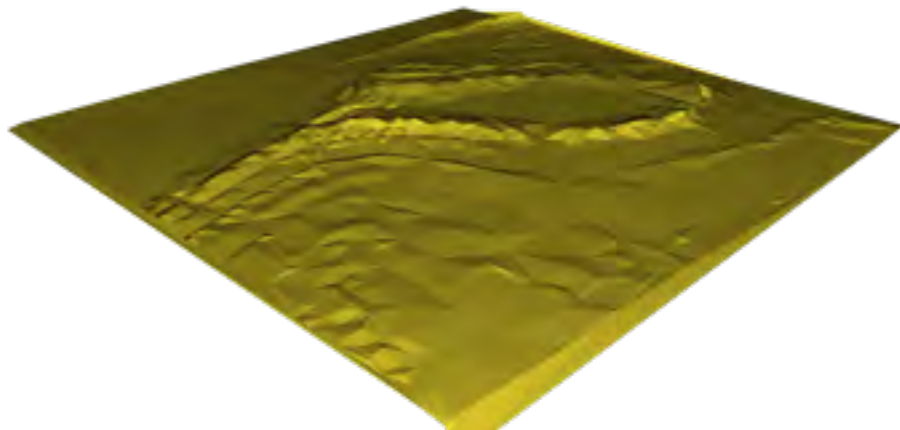
SKETCHUP

1. Official website: <http://www.sketchup.com/>

2. 3D Wharehouse: A repository of SU models

<http://sketchup.google.com/3dwarehouse/?hl=en&ct=lc>

3. SketchUp education: <http://sketchucation.com/>



Interactive SketchUp 3d model

SketchUp is a multi-platform (Windows, OS X and Linux via Wine), extensible 3D drawing software initially developed for architects. Its capabilities are such that SketchUp is now used by civil engineers, geographers, geologists, filmmakers, game developers etc. SketchUp comes as freeware, but also exists in a “professional” version (about \$400).

SketchUp was born in 2000 and initially developed by a startup company called @LastSoftware. It was then bought by Google in 2006, and in April 2012 SketchUp was sold to Trimble.

One of the features that appeals to geoscientists is the capability of SketchUp to georeference models using Google kml format so SketchUp models can be ported into Google Earth. Another feature is the possibility to very easily and interactively extract 2D sections through the model, and design 3D animations.

SketchUp can also export its 3D models in a .dae format (digital asset exchange), a format that allows interactive 3D models to be embedded into pdf files, ePubs and iBooks.



SketchUp basic principles

- Draw a simple 3D surface (view movie then read on).

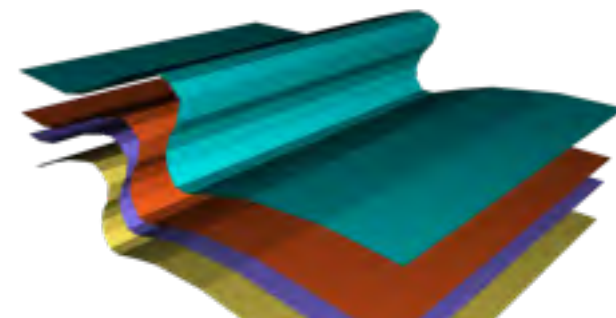
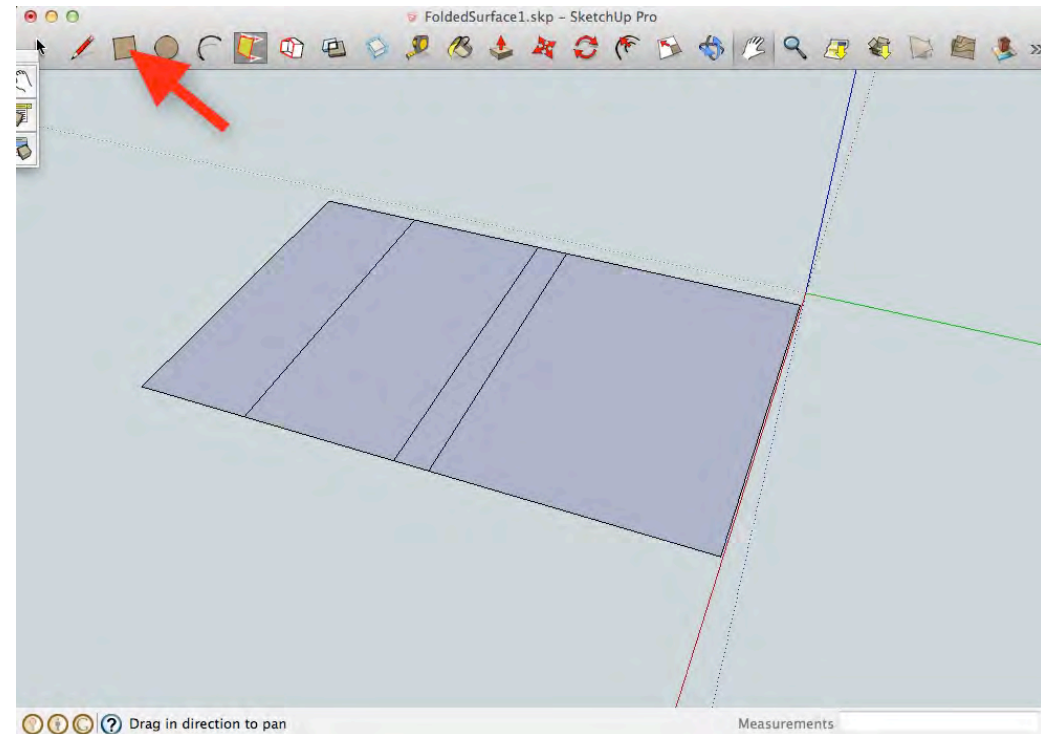
Once the surface is finished select **Edit > Make Group**

 + **<click option>** to duplicate the surface

Select a group and color the surface using 

In menu item **Window** select **Style**, click on **Edit** and deselect **Edges**.

Movie 2.1 Drawing simple surfaces in SketchUp



Interactive

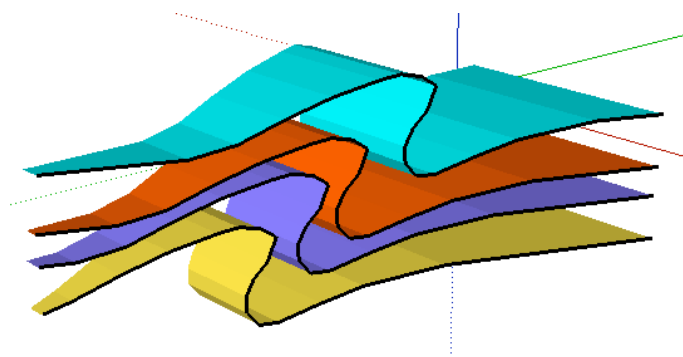


SketchUp basic principles

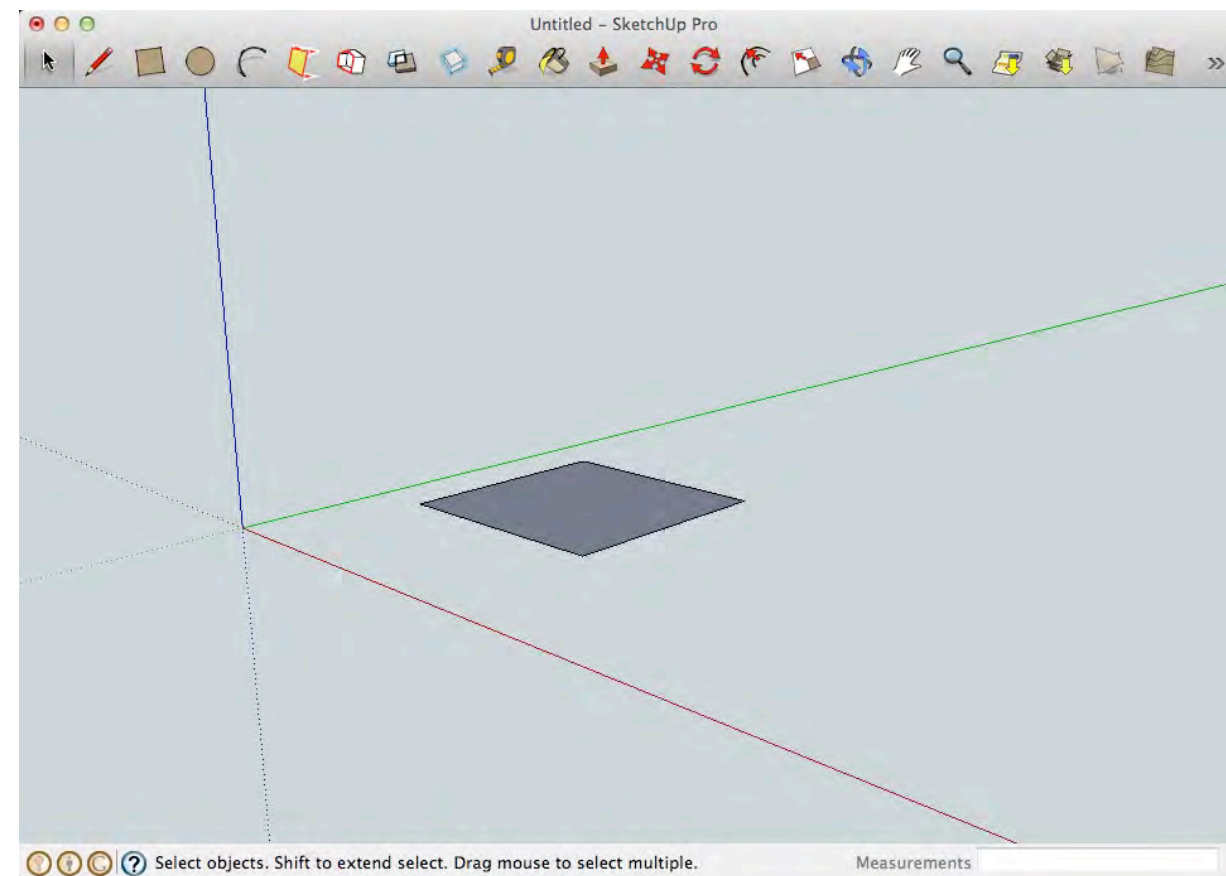
- 3D through Push-pull
- 2D Section

Volumes can easily be created by applying the Push-Pull tool on 2D surfaces. These volumes can be explored through 2D sections using the **Section Plane** tool. A 2D slice of the model can be created by *control-clicking* on the section plane and selecting **Create Group from Slice**.

This offers geologists the capacity to create 3D block diagrams and to extract cross-sections of any orientation.



Movie 2.2 Section through volume

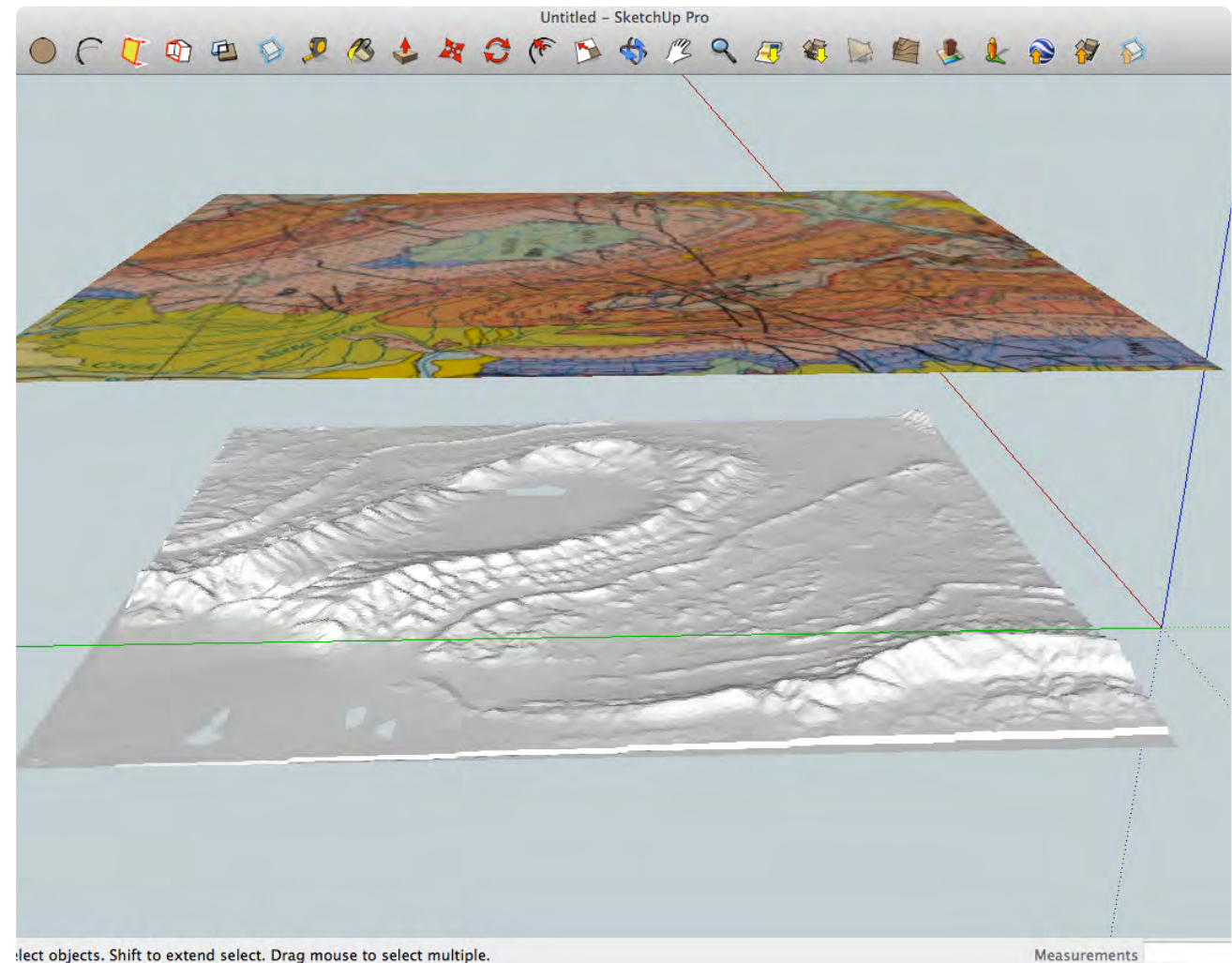




SketchUp basic principles

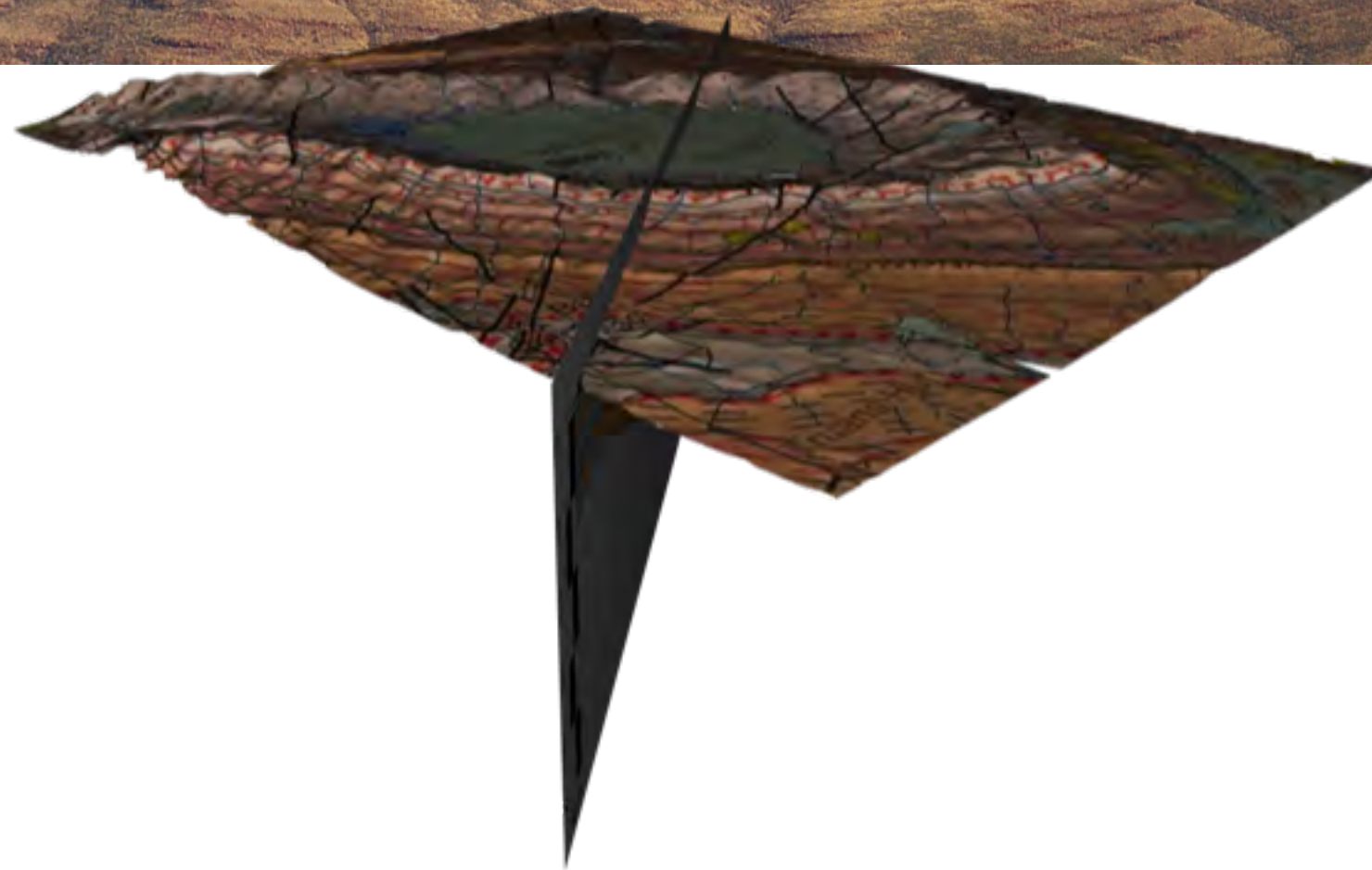
- Draping an image onto a 3D surface

- 1/ Import a .3ds file and an image file (.jpeg or .png)
- 2/ Scale them so they cover the same lat-long
- 3/ Window > Style > Edit > unselect Edges
- 4/ Explode the image file
- 5/ Eyedrop (alt-bucket or command-bucket) onto the image to copy its texture
- 6/ Select the 3D surface Group and Edit > Component > Edit Component and triple click into the 3D surface. All triangles of the mesh should turn blue.
- 7/ Use the Bucket tool to drop the image texture onto the 3D surface, et voilà.



1/ Import a .3ds and an image and scale them so they cover the same lat.long, and unselect Edges.





Quantum GIS GRASS

Geo-referenced data informs decision makers in infrastructure planning, insurance policies, environment protection etc. Driven by remote sensing, information technologies and the Internet, GIS allows for the processing, analysis, visualization and distribution of local to global geo-referenced data.

Quantum GIS (also known as QGIS) and GRASS (Geographic Resources Analysis Support System) belong to a small family of free, open source and extensible (via plugins) GIS softwares that can run on Windows, Linux and Mac platforms. In this chapter we learn the fundamentals of GIS by using QGIS as well as GRASS tools accessible directly in QGIS.



What is GIS ?

GIS

1. Video

With and historical perspective (10 minutes)

<http://www.youtube.com/watch?v=j5WmvTxQF5w>

<http://www.youtube.com/watch?v=kEaMzPo1Q7Q>

The world of GIS is expanding very fast. This expansion is driven by technological progresses (satellites and remote sensing techniques, the Internet, high-performance computers) allowing us to collect geo-referenced data in unprecedented volumes, and to process, analyse and visualize them.

It is also driven by our increasing capacity to probe social networks (hosting the digital life of a large fraction of the Earth's population), national and international open digital databases, and to deploy global surveillance networks around the planet to monitor weather patterns, the spread of epidemic diseases, earthquake activities, tsunamis and flooding, migratory flux, population changes etc.

This is part of the “Big Data” revolution (enter “big data” on Google Trends, and see for yourself). This revolution is affecting all parts of economic, social and natural sciences, and there is a growing need for scientists able to work with Big Data.

Data is “big” when it challenges our technological ability to handle, process and visualize it. Big Data in Geosciences is indeed very BIG because of the global (volume), multidimensional (variety), and often real-time (flux) nature of Geosciences data. To start the learning process, check these two youtube videos on Big Data:

http://www.youtube.com/watch?feature=player_detailpage&v=eEpxN0htRKI

http://www.youtube.com/watch?feature=player_detailpage&v=7D1CQ_LOizA

What is a GIS problem?

Example 1: There is a plan to build rescue centers across a city, which is regularly exposed to flooding. These centers should be 1/ safe from flooding, 2/ reachable by anyone within 10 to 15 mins, 3/ accessible to 90% of its inhabitants. The first outcome requires the mapping of all available sites well above the highest historic flooding. In case of flooding 10 cm higher or lower can be the difference between safety and disaster, High-resolution Digital Elevation Models are therefore paramount here. The second outcome requires the mapping of “path of least resistance” from any point in the city to the closest rescue center. Such a path should minimize crossing bridges, major intersections including high-way and train lines, narrow streets, and low elevation spots. The third outcome requires to superimpose the potential rescue center sites to a map showing the present and projected population density to maximize access to 90% of the population.

Example 2: In seismically active regions, regional development requires to pay particular attention to earthquake risks. The objective here is to avoid developing major infrastructures (airports, train lines and highways, dams, nuclear plants, hospitals, schools, rescue centers) in places exposed to earthquake hazards (ground shaking, liquefaction, landslides, ...). This requires the mapping of all exposed sites. For this, one needs to know i/ the distribution of existing faults in the region, ii/ a map of Coulomb stress changes over the past century to

assess future earthquake potential, iii/ the susceptibility to ground shaking, liquefaction and landslides (a function of the local geology and topography).

Both problems require working with geo-referenced data, maps combining raster images (i.e. gridded field data: data distributed on a grid covering the region of interest, data can be elevation, population density, shaking potential etc), as well as vector images (i.e. points such as position of rescue centers, lines such as roads, and surfaces such as lakes, geological exposures etc) . These raster and vector images (often referred to as geo-referenced data) come from various sources using different grids (lat-long, UTM with various datum, and various projection system such as Mercator, ...), they may cover different surfaces, and use different symbols. Hence, a fundamental part of building a synthetic GIS map is to homogeneize the various images. GIS is not restricted to combining existing maps. It also involves the creation of new maps via the processing of geo-referenced data. For instance, from a Digital Elevation Model giving the elevation on a grid, one can create another map showing the *Relief*. This can be done by running a program returning at each node N of the grid the difference $MaxElev - MinElev$ over a region centered of N and covering the surrounding cells directly adjacent to N (closest neighbors).

QGIS/GRASS

QUANTUM GIS

1. Source of QGIS:

<http://www.qgis.org>

<http://plugins.qgis.org/plugins/plugins.xml>

2. QGIS Tutorials

<http://qgis.spatialthoughts.com/?m=1>

<http://planet.qgis.org/planet/>

http://docs.qgis.org/html/en/docs/user_manual/introduction/qgis_gui.html

3. Web map server

<http://www.geoscience.gov.au/wms.html>

http://grasswiki.osgeo.org/wiki/Global_datasets

4. QGIS Mapping tool

<http://www.niwa.co.nz/software/quantum-map>



Quantum GIS: The development of QGIS began in 2002. Because QGIS is less demanding on hardware (less ram and less processing power), and because it is multi-platform and has a simple Graphic User Interface, QGIS offers an easy access to the world of GIS. Nevertheless, under the hood QGIS provides integration with web map servers (wms), and other GIS packages including GDAL, PostGIS, and remarkably to GRASS via plugins.

GRASS is a collection of over 350 programs (tools) accessible through a terminal (command line) or through a Graphic User Interface. GRASS tools can also be loaded into QGIS via a plugin. As a standalone application, GRASS follows a rather tight files architecture which helps collaborative work (i.e. many people working on the same project editing and working on the the same set of files, maps etc), and allows projects to be portable across various GIS. To keep things simple we will access GRASS tools through QGIS. QGIS can be seen as a Graphic User Interface (GUI) for GRASS.

QGIS will be our entrance door to the world of GIS. Installing QGIS on your machine will require installing some frameworks and other packages (order is important here, <http://www.qgis.org/>): First GSDAL (a Geospatial Data Abstraction Library), then GSL (a numerical library for C and C++ programmer). Then install GRASS, then Python, and then install QGIS. *Quantum GIS:*

What follows is a set of 5 to 10 minutes video tutorials on QGIS.
These tutorials will guide you through the basics.

QGIS Video Tutorials

Basic features:

<http://www.youtube.com/watch?v=3kuakfQFq-o&lr=1>

<http://www.youtube.com/watch?v=59Oer-i6nVc>

http://www.youtube.com/watch?v=AsC_AEqRRRI

Plugins:

<http://www.youtube.com/watch?v=XCuFK-0Ckyg>

Layer properties:

<http://www.youtube.com/watch?v=9tkOeRM0OXY>

<http://www.youtube.com/watch?v=ZbnCrfoWnNk>

Georeferencing & vectorization:

<http://www.youtube.com/watch?v=ffPL5h4mJf4>

<http://www.youtube.com/watch?v=xcqzEpoRuok>

Projection:

<http://www.youtube.com/watch?v=kcGW2YHGNTM>

<http://www.youtube.com/watch?v=hx-lASR7WHk>

Symbology and labeling:

http://www.youtube.com/watch?v=gPnp7o_Qcwg

<http://www.youtube.com/watch?v=duuYMufA-RU>

Vector analysis:

<http://www.youtube.com/watch?v=9HTvinfugAg>

Map composer:

<http://www.youtube.com/watch?v=nQVnVJea8AQ>

Raster analysis with GRASS tools:

<http://www.youtube.com/watch?v=iffRz7M2L2U>

Importing GPS data:

<http://www.youtube.com/watch?v=QoXNQuETPSg>

Importing Google / kml data:

<http://www.youtube.com/watch?v=-Ze1IP1kyW8nb>:

nb: The next one requires Google API

<http://www.youtube.com/watch?v=-ujt3C06Org>

Importing QGIS into Google Earth:

http://www.youtube.com/watch?v=p9Egl_RbXBU

Importing from wms server (One Geology):

http://www.onegeology.org/wmscookbook/1_4_7.html

Creating a DEM from contours:

<http://linfiniti.com/2010/12/3d-visualisation-and-dem-creation-in-qgis-with-the-grass-plugin/>

Creating a DEM from vector data:

<http://wiki.awf.forst.uni-goettingen.de/wiki/index.php/>

[Creating a DEM from vector data](#)

Creating Maps from SRTM Data:

<http://developmentseed.org/blog/2009/jul/30/using-open-source-tools-make-elevation-maps-afghanistan-and-pakistan/>

Inverse Distance Weighting (IDW) Interpolation using QGIS:

<http://www.gistutor.com/quantum-gis/20-intermediate-quantum-gis-tutorials/51-inverse-distance-weighting-idw-interpolation-using-qgis.html>

A workflow for creating beautiful relief shaded dems using GDAL:

<http://linfiniti.com/2010/12/a-workflow-for-creating-beautiful-relief-shaded-dems-using-gdal/>

1/ Using SRTMImport plugin import and save as a shape file the Sydney basin srtm tile.

2/ Follow <http://underdark.wordpress.com/2012/06/07/mapping-open-data-with-open-gis/> to produce a hillshade map showing the river network.

3/ ASTER GDEM offers the world at 30 m resolution. Download the Sydney basin <http://gdem.ersdac.jspacesystems.or.jp/search.jsp>

4/ To upload the GDEM dataset into QGIS and make a contour map follow this procedure: <http://planet.qgis.org/planet/tag/dem/>

- ° Click on Add Raster Layer
- ° Select the GDEM .tiff file and press Open
- ° Right click (or option click) on the image in the Table of Contents and Select Properties

° Select the Style Tab and under Contrast enhancement, Change the Current pulldown from Default No Stretch, to Stretch to MinMax

Multispectral imagery by Anthony Beck

Satellite 1: http://www.youtube.com/watch?feature=player_detailpage&v=SheQQkZ5NYk

Satellite 2: http://www.youtube.com/watch?feature=player_detailpage&v=4OcQYB7RPUA

DEM1: http://www.youtube.com/watch?feature=player_detailpage&v=ZI2sBWEQ7Ok

QGIS and Google Map 1: http://www.youtube.com/watch?feature=player_detailpage&v=GS3n_zBk_tE

QGIS and Google Map 2: http://www.youtube.com/watch?feature=player_detailpage&v=SRPkLQbxNmk

QGIS and Raster: http://www.youtube.com/watch?feature=player_detailpage&v=6XH0qINm5UE

First map in QGIS from scratch

1/ Fetching and loading data

ASTER GDEM offers the topography of the world at 30 m spatial resolution. Download the four tiles covering the Sydney basin from:

<http://gdem.ersdac.jspacesystems.or.jp/search.jsp>

You have to register and get a username and password (send via email) to be able to download the ASTER data.

Unzip the four ASTER tiles and load the four _dem.tiff into QGIS (via Add raster layer).

2/ Merging rasters

Raster > Miscellaneous > Build Virtual Raster and select the following options:

i/ Use Visible raster layers for input

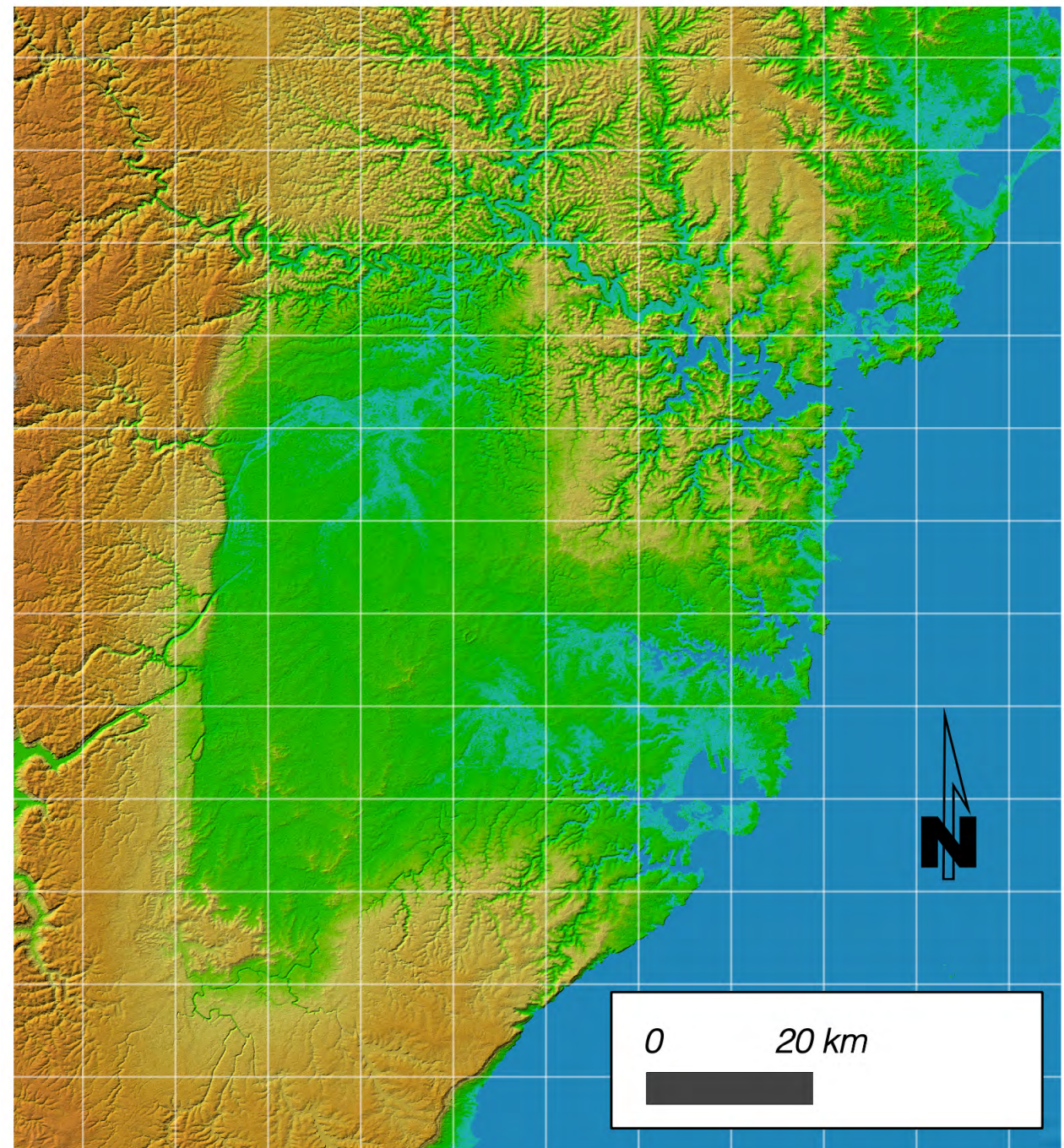
ii/ In Output File choose a name for the merged file and a location where to store it.

iii/ Load into canvas when finished

Click OK to create a file in which the four ASTER tiles have been merged.

Remove the four ASTER tiles, as they are no longer needed.

Topographic map of Sydney basin (NSW)



In QGIS canvas, the merged raster looks grey because “style” has yet to be applied. Double click into the merged layer to open the Layer Properties window. In the “Load min / max values from band” choose *Actual*. In “Contrast enhancement” choose *stretch to MinMax*. Click *Apply* then *OK* to stretch the grey levels (0 to 255) over the lowest and highest elevation of your map. This will produce a map whose grey shading represents the topography.

3/ **Subsampling**

Lets extract a smaller subsample of this image:

Raster > Extraction > Clipper . Choose round lat & long values (e.g., -33.2 rather than -33.163, adding a lat-long grid on your project will be easier.)

In QGIS canvas, select a rectangle covering the Sydney Basin.

Back in the Clipper window choose a name and location of the output file.

Activate *Load into canvas when finished*.

Close the merged file and keep the subsample region.

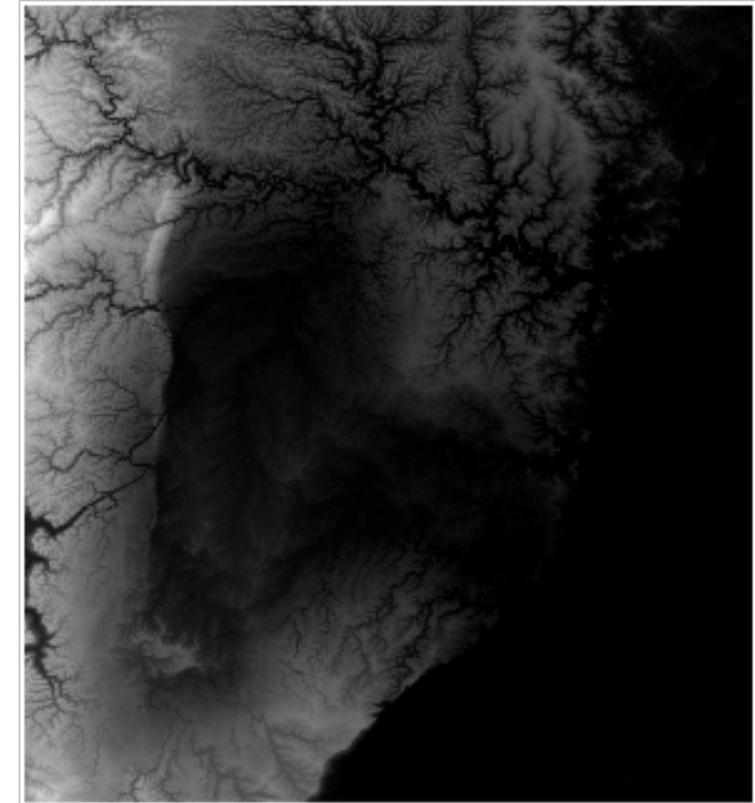
Double click on the subsample region to open the *Layer Properties* window and stretch the colour response to *Actual Min and Max*.

4/ **Make topographic contours**

Raster > Extraction > Contours

Choose an Output name and select the following options (100, ELEV, Load into canvas). Click OK ... Be patient contouring can take a while.

NB: To see these contours draped onto the Google Earth landscape: Select the contours vector layer in the Layer window then > Layer > Save As option “Keyhole Markup Language”. This file can be opened into Google Earth.



5/ Lightning the landscape

Another way to visualize the topography is by lightning the landscape to produce a *hillshade* image.

To do this: Raster > Analysis > Dem

Select the Input image and choose an Output name, etc.

You end up with a rather granular and dark image. This is because x and y (i.e. longitude and latitude) are in degrees, and z (height) is in meters.

To change the vertical to horizontal ratio (-s) knowing that at Sydney latitude 1 degree = 111120 m. For this we edit the gdaldem command (click on the edit icon) and replace the option “-s 1” by “-s 111120”. This gives a much paler and correct *hillshade* map. This factor can also be entered in the Hillshade window.

6/ What about colour?

Various methods exist, one uses the Raster Properties. Another option is to create a simple colour scheme to apply to your topographic map. In your favorite Text Editor enter the following and save this text file as “color100.txt” into your working directory:

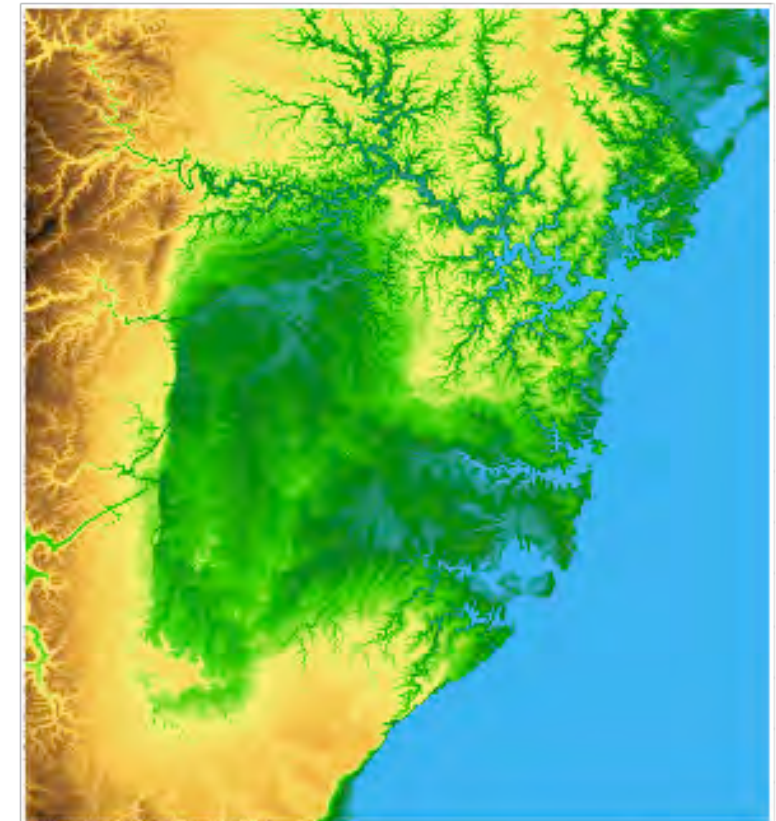
```
# Elev R G B
0 60 180 240
20 5 124 20
100 51 204 0
200 244 240 113
400 244 189 69
600 153 100 43
940 10 10 10
```

The first column of the file shows the elevation at which a given colour starts. The three adjacent columns refer to the RGB colour.

Use *colour relief* (in Raster > Analysis > Dem) to apply this color scheme to the to grey-scale raster (not the *hillshade*). Find a way to map all areas at elevation from 0 to 10 m.

Tip 1: Regularly save your project: File > Save Project.

Tip 2: Need more colour schemes? Try <http://colorbrewer2.org/>.



7/ Rasters fusion

A simple way to fuse rasters is to use Transparency. Alternatively, fusing *Color relief* (gdaldem color-relief) and *Shaded relief* (gdaldem hillshade) rasters produces visually pleasing maps. One way to do this is to use the *hsv_merge.py* program from Frank Warmerdam. You will need Python as well as NumPy librairies (check Python Modules in QGIS.org). This script combines the hue and saturation from the color bands of the color relief image, with the panchromatic band of the shaded relief. One can run this script from the Python Console available in QGIS or simply from a Terminal window.

Download *hsv_merge.py* on your working directory and enter the following command:

```
./hsv_merge.py path_to_ColorRelief.tif path_to_Color_Shading.tif fused.tif
```

NB: Be careful here: In many instances, the order of your input images may be important, feel free to experiment.

Tip3 : More technique using ImageMagick : <http://dirkraffel.com/2011/07/05/best-way-to-merge-color-relief-with-shaded-relief-map/>

More tips here: <http://linfiniti.com/2010/12/a-workflow-for-creating-beautiful-relief-shaded-dems-using-gdal/>

8/ No map without graticules and a scale bar

A proper map should always include information about its geographic coordinate system (via graticules), a scale bar and the north direction.

Add a grid (first option): Vector > Research tools > Vector grid

Add a grid (second option): Install fTools plugin (via the *Plugins Manager*) then Research Tools > Vector grid. Click on *Update extents from layer*. This will populate the *XMin XMax, YMin and YMax* fields.

In both options, tidy things up by rounding the extend of the grid, and choose appropriate grid increments. Choose *Output grid as lines*. Finally choose a name and a destination for your grid file. This will create a shape file with a grid. You can change the grid's style and add more lines via the Layer Properties window.

To add a scale in *km* you need first to switch to a UTM projection. Go to File > Project Properties > Project Reference System Choose Enable "on the fly" CRS transformation and select WGS 84 / UTM zone.

Then View > Decoration and setup the scale bar. In Decoration you can also add a North direction.

Adding the scale bar and other decoration in QGIS is neither the best nor the easiest strategy. It's better to finalize your map in another app such as inkscape.

SYDNEY BASIN TOPOGRAPHIC MAP

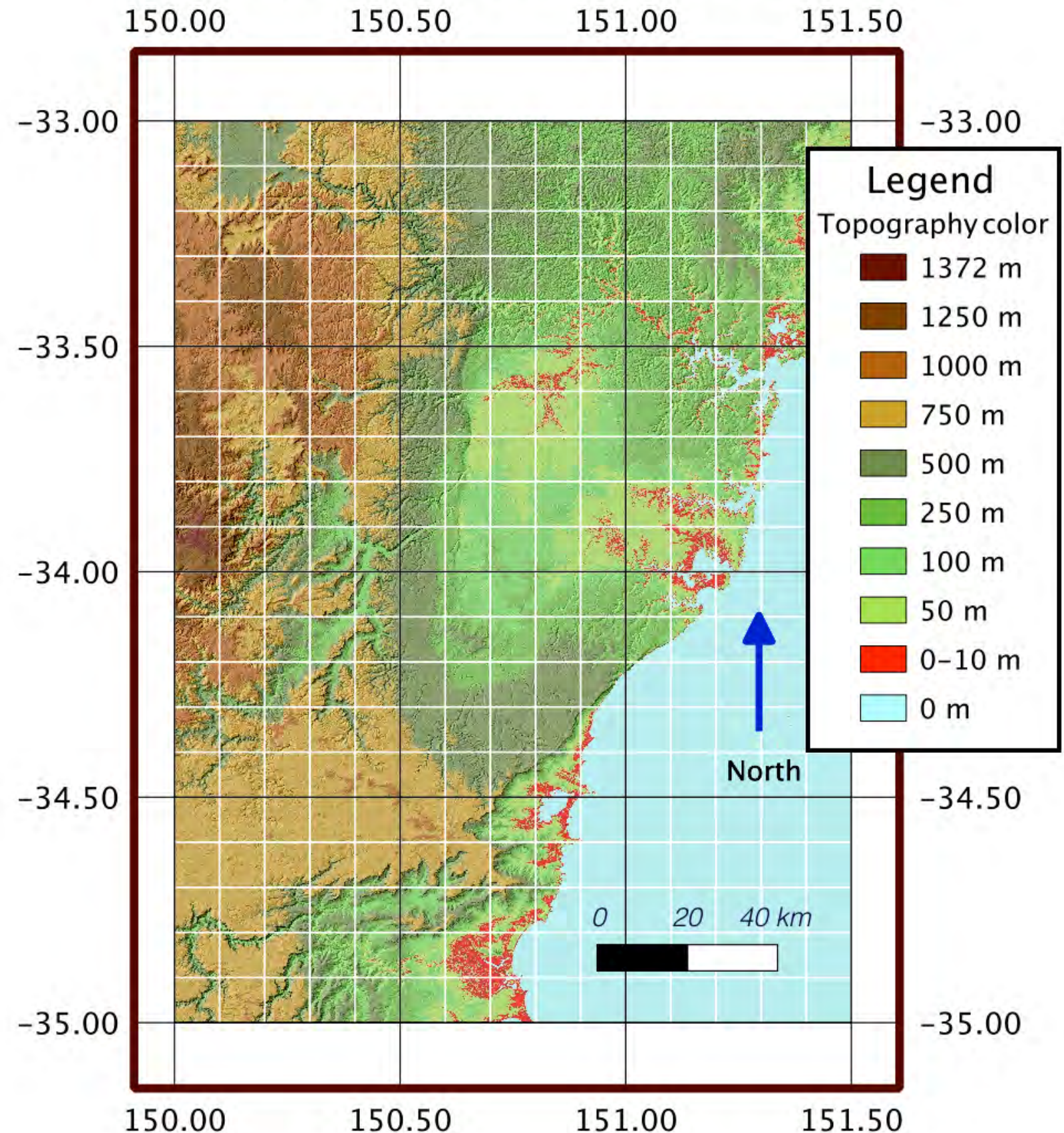
9/ Your first topographic map

Your topographic map can be assembled in QGIS Print Composer. Check this tutorial:

<http://qgis.spatialthoughts.com/2012/06/making-maps-for-print-using-qgis.html>

<http://gis.stackexchange.com/questions/28870/add-utm-labels-to-grid-in-qgis>

However, nothing prevents you from using your favorite drawing package (try inkscape an open source, multi-platform vector graphics apps).



10/ Let's add a bit of geology

Go to: <http://www.resources.nsw.gov.au/geological/geological-maps/1-250-001>

Download the ESRI Shape file for the geological maps (1/250000) of Sydney and Wollongong. Also download the most recent scanned version of the maps. UnZip and put into your working directory.

Tip 4: Should you need to georeference a scanned document then follow this tutorial:

<http://qgis.spatialthoughts.com/2012/02/tutorial-georeferencing-topo-sheets.html>

In QGIS open the two RockUnit shape files. To merge both Vector maps into one: Vector > Data Management Tools > Merge shapefiles into one, OR Vector > Geoprocessing Tools > Union, OR alternatively: Plugins > mmqgis > Transfer > Merge Layers.

The merged vector map shows one color only ... not helpful. To properly colorize your map: Layer Properties > Style Categorized – LETT_SYMB – Classify – Apply. The color of each Rock Unit can be changed (double click on the symbol). Look at the geological maps (scanned versions) and group the rock units in a few categories (ages + one category for unconsolidated sediments).

Sydney Sheet	Wollongong Sheet
Unconsolidated sediment: Qa, Ts, Lat	Unconsolidated sediment: Qal, T
Tertiary: Tv	Tertiary: T##
Triassic sandstone and shale: R##	Triassic sandstone and shale: R##
Permian sandstone and shale: P##	Permian sandstone and shale: P##
Paleozoic rocks: S#, D##, C##	Paleozoic rocks: O#, S#, D##, C##

Tip 5: To make sure that all layers use the same projection go to: File > Project Properties and choose a Coordinate Reference System (CRS) for the project (WGS84).

11/ Importing Comma Separated Variable (csv) datasets

Go to <http://www.ga.gov.au/earthquakes/searchQuake.do> and get all earthquakes available in the database (magnitude 0 – 9.9; 1/1/55 to today, depth 0-1000 km, all earthquakes, then search; in the following window keep the default options except for Approximate Location and Solution Finalized -not needed- and click on Export).

If it is not already installed, install via *Plugins Manager* the “*Add Delimited Text Layer*” plugin, then: Layer > Add Delimited Text Layer

In the window “Create a Layer from a Delimited Text File” select the appropriate options and keep an eye on the “*Sample text*” window to check how the spreadsheet responds to your options (i.e. make sure that headers and corresponding columns are properly aligned).

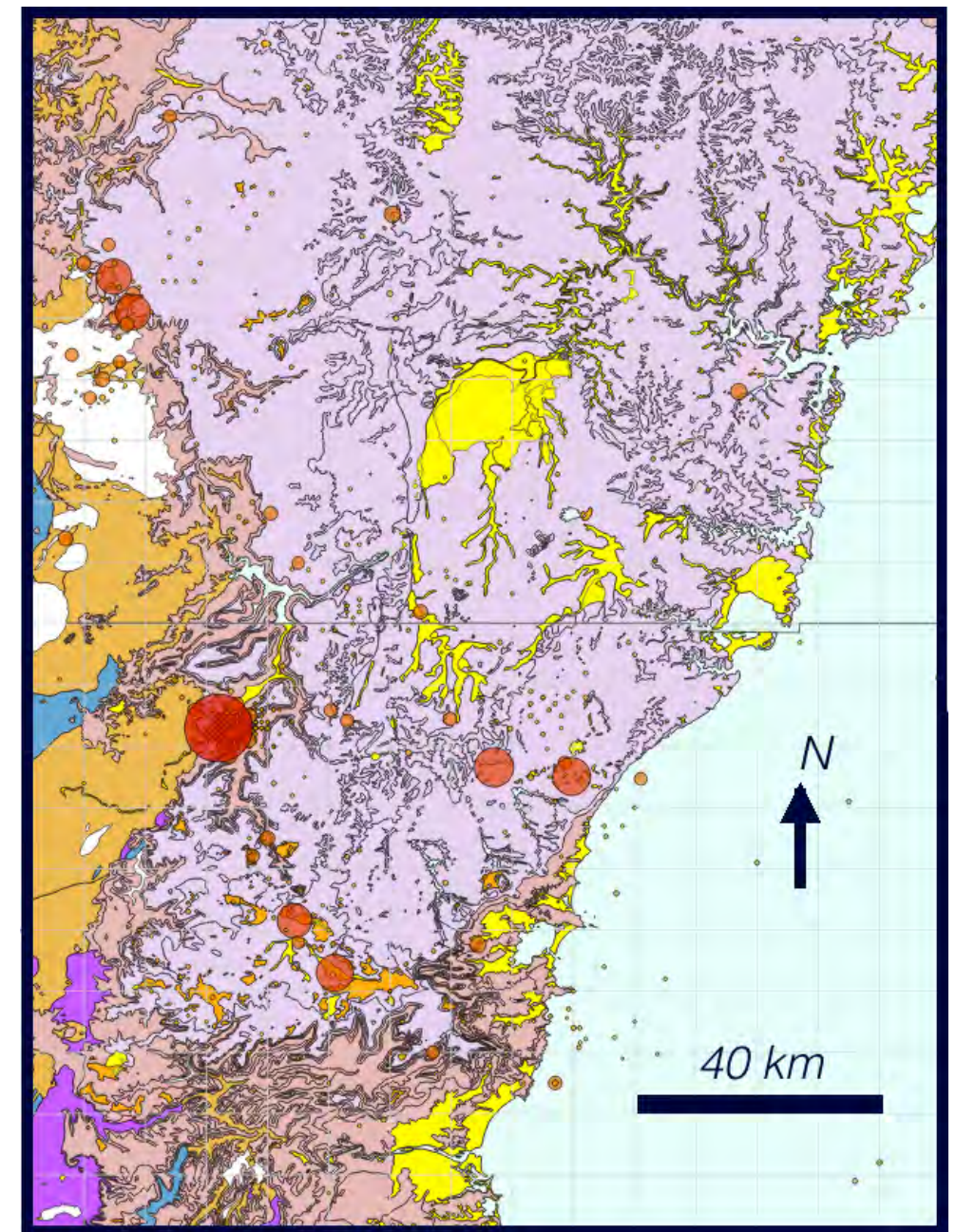
Click OK to load the data into a layer. To select the epicenter in the vicinity of Sydney there are two methods:

1/ Right click (Mac: Control-Click) on the data layer and open the Attribute Table. Click in the *Advanced* search and in the “*search query builder*” enter the following query:

Longitude>149.5 AND Longitude<153 AND Latitude<-33.0 AND Latitude>-35.5.

Click OK to return to the Attribute Table and Close. Click on the Earthquake layer again and activate *Save Selection As*. Save your selection as an ESRI shape file. You can remove the Earthquake database and keep only the earthquake for the region of interest. Now is a good time for Tip 5 !!!

2/ Select features by rectangle tool, stretch the selection box over the region of interest, right (or control) click on the earthquake layer and Save selection as.



Build your map of the Sydney basin in a PDF format. It should include graticules, scale, a north arrow. It should show earthquake epicenters with a color function of the magnitude. The background should show a simplified geological map with unconsolidated sediments clearly visible. Bonus points for map showing the relief.

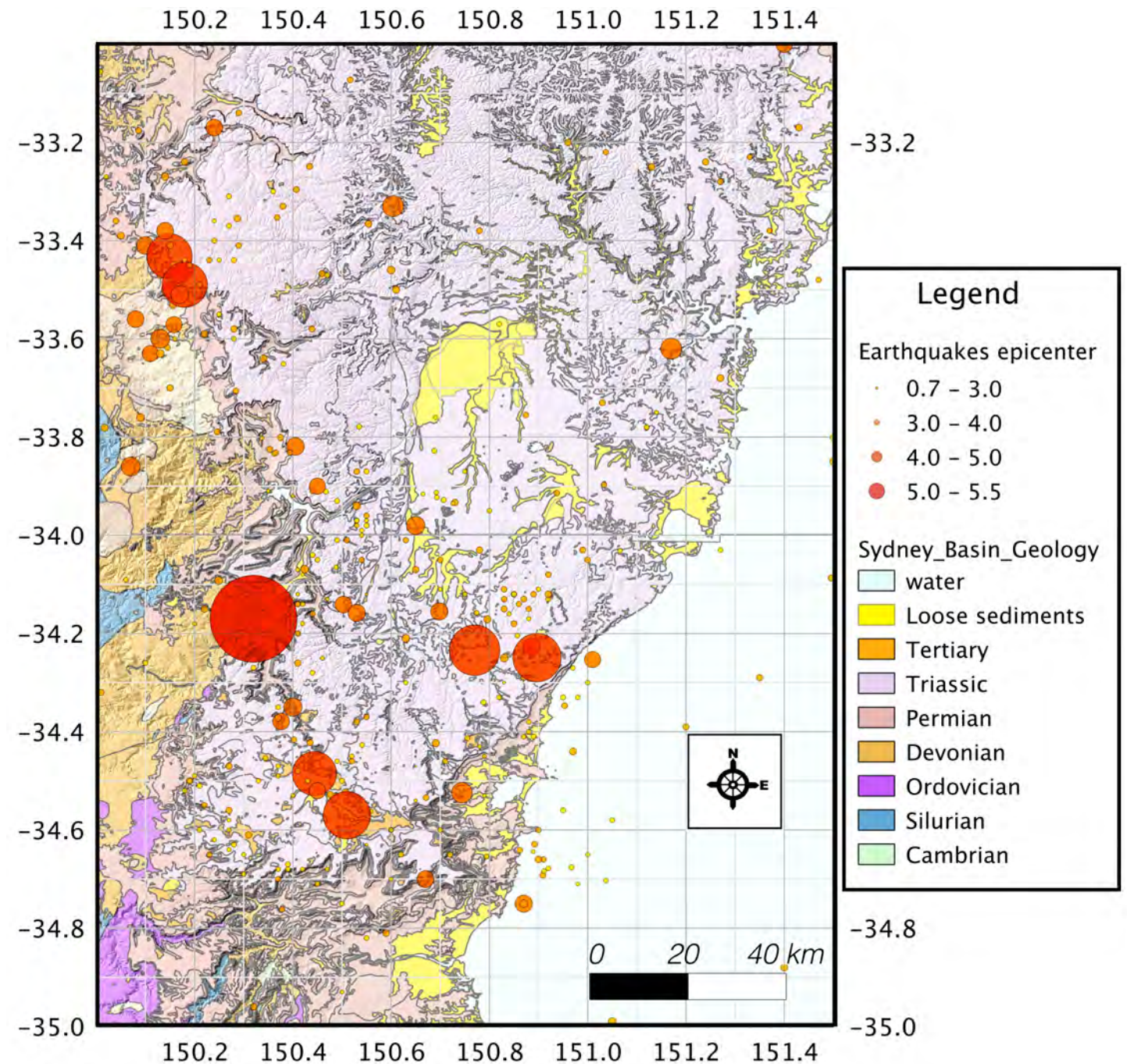
A Final Note: Like any other apps QGIS suffers from occasional bugs. Unlike proprietary softwares, these bugs – tracked and documented by users - usually find a quick fix thanks to the rapid response of the users community.

Print Composer Issue (QGIS 1.8, Mac 10.7.6, <http://hub.qgis.org/issues/6125>): Should the right panel and Tool Bar disappear in the Print Composer window here is what to do:

In the Plugins > Python Console execute the following script:

```
from PyQt4.QtGui import *
actcs = qgis.utils.iface.activeComposers()
for actc in actcs:
    cw = actc.composerWindow()
    mb = cw.menuBar()
    wm = mb.findChild(QMenu, 'wmenu')
    if not wm:
        wm = mb.addMenu('Window')
        wm.setObjectName('wmenu')
    wm.addAction(cw.createPopupMenu().actions())
```

All Composer windows now have a menu named 'Window' which lists the toolbar and dock widgets (same as contextual menu on tool bar). You can execute the code as many times as needed. It will not keep adding new 'Window' menus.



A Bushfire mitigation example using QGIS and GRASS >> :

www.qgis.org/en/community/qgis-case-studies/queensland-australia.html

R - GIS Statistics & Graphics

R is an open-source, cross-platform, extensible-by-nature, toolbox that bridges three disciplines: GIS, Statistics and Infography.

R is the Swiss Army Knife of all scientists and engineers. It can replace Matlab, and run scripts written in other languages including Matlab, Python, C etc. In short, if you do not have R in your pc or mac, you should. This chapter shows you why.



An Overview

MORE INFO...

1. To get R, Packages, Manual:

<http://lib.stat.cmu.edu/R/CRAN>
<http://www.r-project.org>

2. R Mailing Lists

<http://www.R-project.org/mail.html>

3. R community websites

<http://www.r-bloggers.com>
<http://spatial-analyst.net>
<http://www.statmethods.net>

4. You Tube Intro (4 parts - total 30 mn):

http://www.youtube.com/watch?feature=player_detailpage&v=M2u7kbcXI_k
http://www.youtube.com/watch?feature=player_detailpage&v=6sr6i62YdxM
http://www.youtube.com/watch?feature=player_detailpage&v=NoV7VrE90LA
http://www.youtube.com/watch?feature=player_detailpage&v=MBwNWANSb4



What is R? R is a collection of functions (i.e. *package* also called *libraries*) written in S (S is a programming language) aiming at reading, processing and visualizing data. R has a strong emphasis on statistical analyses. R is also a programming environment for data analyses and graphics.

What does R bring to GIS? R is a very useful tool to have in your GIS toolbox as it provides users with the computational power to perform sophisticated statistical analysis on your GIS data. The statistics of interest here is the one concerned with the characterization and understanding of “spatial point patterns”, in simple terms: the distribution of an apparently random set of points (e.g. earthquake epicenter), and “line segment patterns” (cyclone tracks, tsunamis).

Typical “GIS” questions that can be answered with R: What is the intensity of the point pattern (i.e. number of points per unit area)? What is the average distance from a point to its nearest neighbour? Is there an azimuth dependence of this average distance? Is the point pattern the result of a “Poisson process” i.e. a uniform random process? Is there a dependency between the point pattern and other variables? (i.e geographic coordinates, elevation, slope, surface geology, proximity of faults, annual rainfall etc).

In what follows, we will explore a number of problems that can be solved using R.

Installing R

Download R from cran.r-project.org and follow the instructions to install. Once R is installed, install *RStudio*, an open source, multi-platform integrated development Environment for R: <http://www.rstudio.com>. *RStudio* works on top of R. When both packages are installed click on *RStudio* icon to start your R environment.

Downloading and Installing R Packages

R is extensible, i.e. new modules (i.e. packages or libraries) can be added. Downloading R packages can be done from cran.r-project.org or from *RStudio* via `install.packages(name_of_package)`. Once installed these packages are selectively loaded into *RStudio* via the command: `library(name_of_package)`.

Tip1: Packages continuously evolve, hence check for updates.

Tip2: In the *R* console enter `library()` to see all loaded packages.

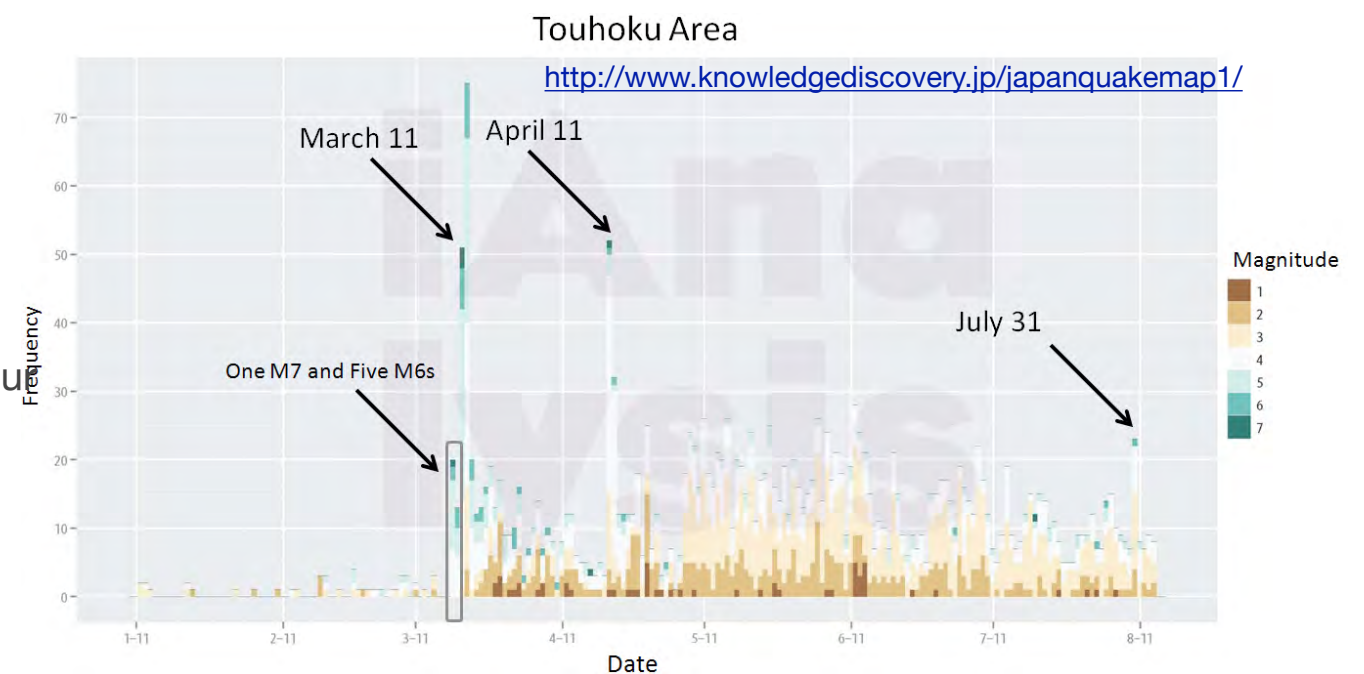
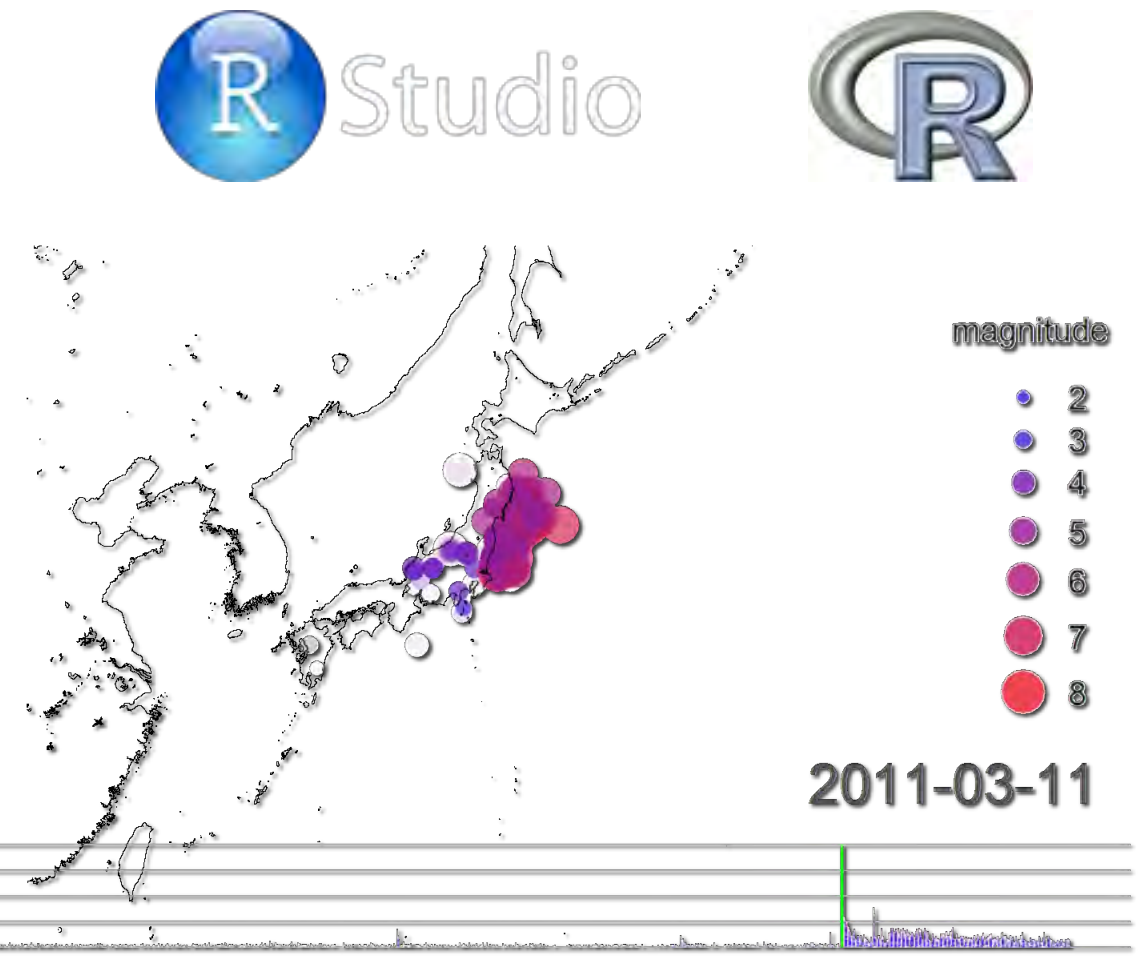
Tip3: Built a directory for each separate project.

Tip4: R is case sensitive: GIS, Gis, gis are different objects.

Tip5: Enter the command: `?name_of_the_R_function` to get help

Workflow ... Start a *R* session, load the libraries required in your project (`require(lib1, lib2, ...)`, or `library(lib1, lib2, ...)`), load data, process data, visualise data, have fun.

NB: If a command that includes quotation marks (" ", "") doesn't work when you copy and paste from ebook to *R*, type the command in *R* instead.



Downloading earthquake data and loading them into R

Go to: <http://www.ga.gov.au/earthquakes/searchQuake.do> and select all Australian's earthquakes of past few decades down to a depth of 1000 km. i.e. Select *Location*: Australia / Select *Magnitude*: 0-9 / Select *Time*: 10 years from today / Select *Depth* 0-1000 km / Select *All Earthquake* / A new page opens, click on *Export Data* (bottom right) / A new page opens, keep all default options and click on *Export* to download a csv (comma separated variable) file into your computer. Move this file into a folder and give that folder a descriptive name. In a text editor, change the column headers, get rid of unwanted data columns etc and save in a .csv format.

You may want to check this You Tube tutorial on loading data into R :

http://www.youtube.com/watch?feature=player_detailpage&v=VLtazaiYo-c

To load your data follow these steps:

Start *RStudio*, in the console enter the following commands in which “qak” is the name chosen for our dataset:

qak=read.csv("filepath", header=T) #nb: header=T(rue) the table has a header

Alternatively look for the “Import dataset” tab and simply follow the instructions.

What if your data are not in a cvs format? No drama:

For space separated columns: *qak=read.table("filepath", header=T, sep=" ")*

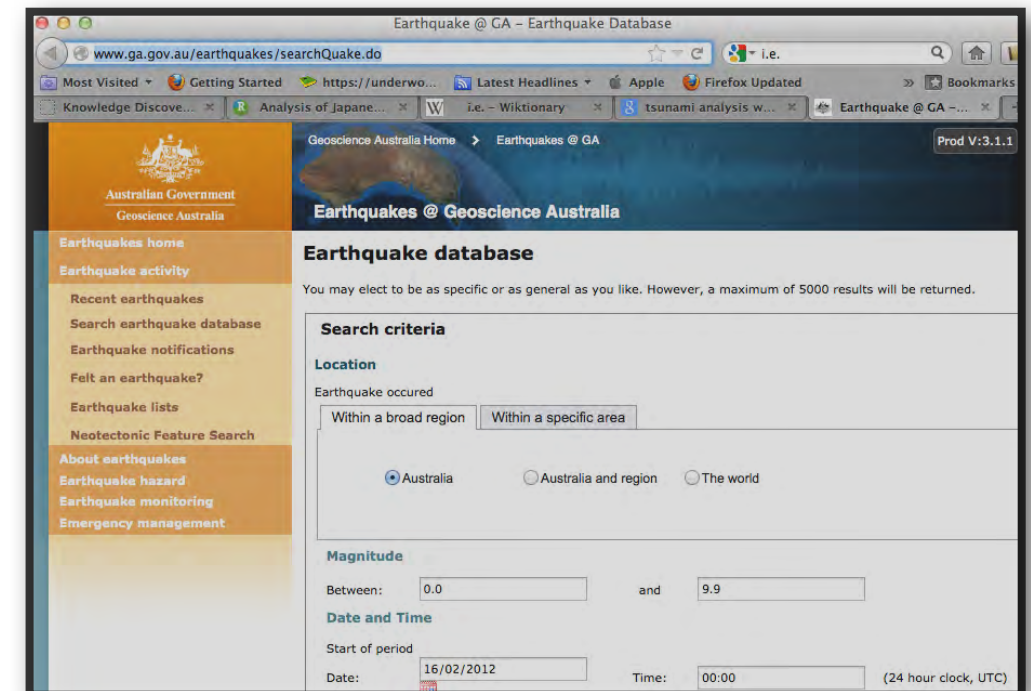
For comma separated columns: *qak=read.table("filepath", header=T, sep=",")*

For tab separated column: *qak=read.table("filepath", header=T, sep="\t")*

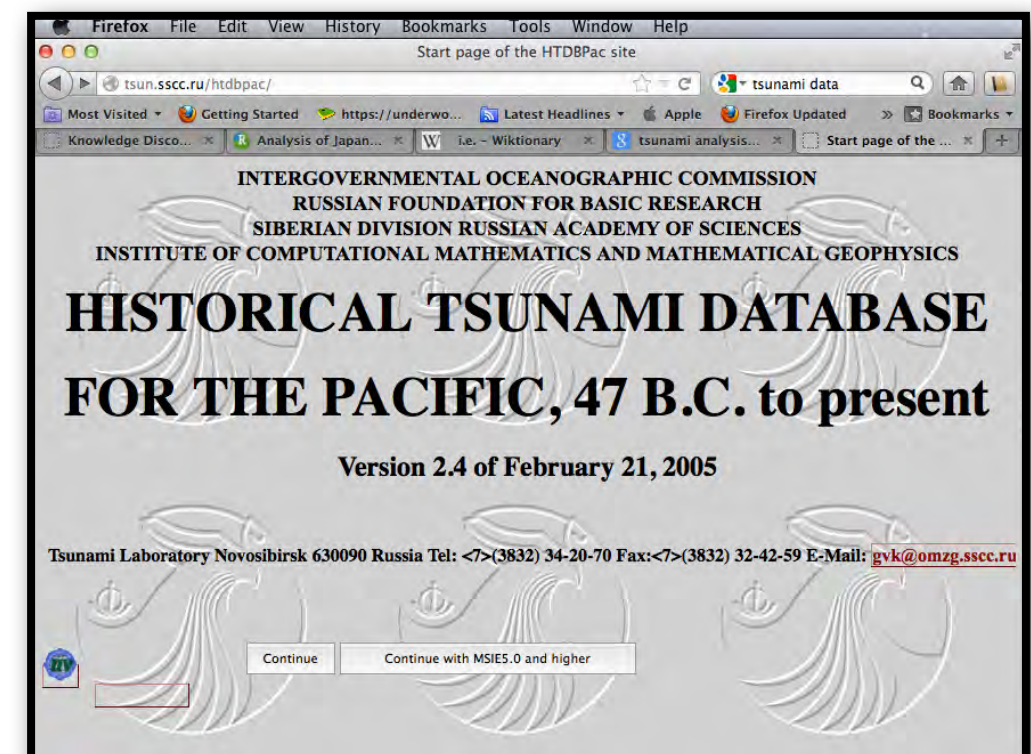
Check that your dataset is loaded by running one of these commands:

str(qak), *summary(qak)*, *fix(qak)*

Your raw data are now loaded, its time for some manipulation...



Geoscience Australia's Earthquake database provides data on thousands of earthquakes in Australia and around the world.



The Historical Tsunami Database for the Pacific (Novosibirsk Tsunami Laboratory) provides data on 1490 tsunami from 47 BC.

Manipulating data in R

Check out:

http://www.youtube.com/watch?feature=player_detailpage&v=7BXHI31Hars

A few commands to know:

1/ To check a file data header enter: `names(qak)`

2/ To replace the header of the x^{th} column's: `names(qak)[x] <- "NewHeader"`

3/ To select a column: `qak$ColumnHeader`

4/ To select the y^{th} row: `qak[y, ]` (nb: note the comma)

5/ To select the z^{th} column: `qak[, z]` (nb: note the comma)

6/ To select the y^{th} data of the z^{th} column: `qak[y, z]`

7/ To select a dataset subset, here all earthquakes > 5 : `qak[qak$Mag > 5]`

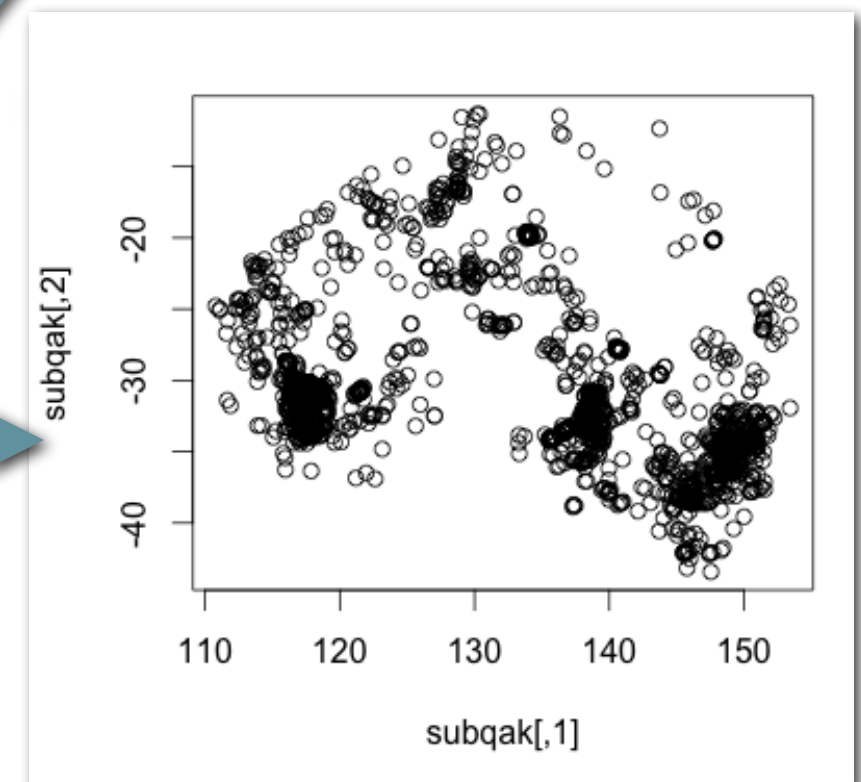
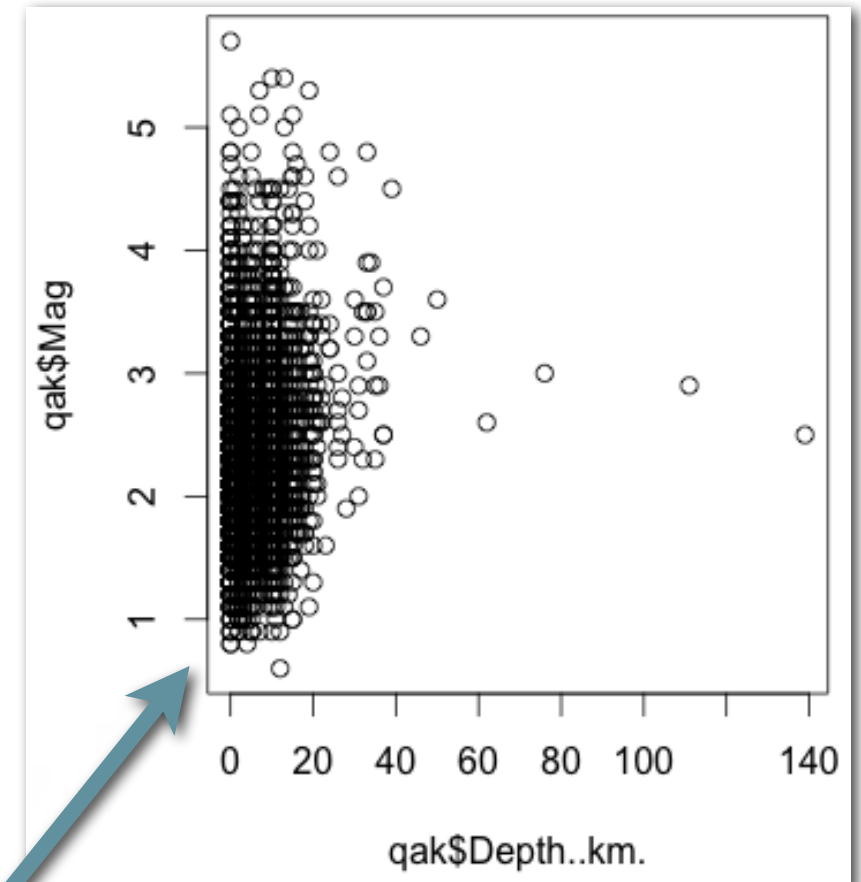
8/ To select from the column *Depth* all earthquakes deeper than 50 km: `qak$Depth[qak$Depth > 50]`

9/ To plot the column *Mag*(nitude) as a function of *Depth*: `plot(qak$Depth, qak$Mag)`

10/ To combine columns *Long*, *Lat* and *Mag* into a table: `subqak <- cbind(qak$Long, qak$Lat, qak$Mag)`; use `rbind` to combine rows.

11/ Then plot: `plot(subqak)`

Nice but we can do better ...



12/ Colored version

The function `col` returns a colour defined by a number. By defining the third argument of the vector as `col=data$Nb` then epicenters have a magnitude-dependent colour:

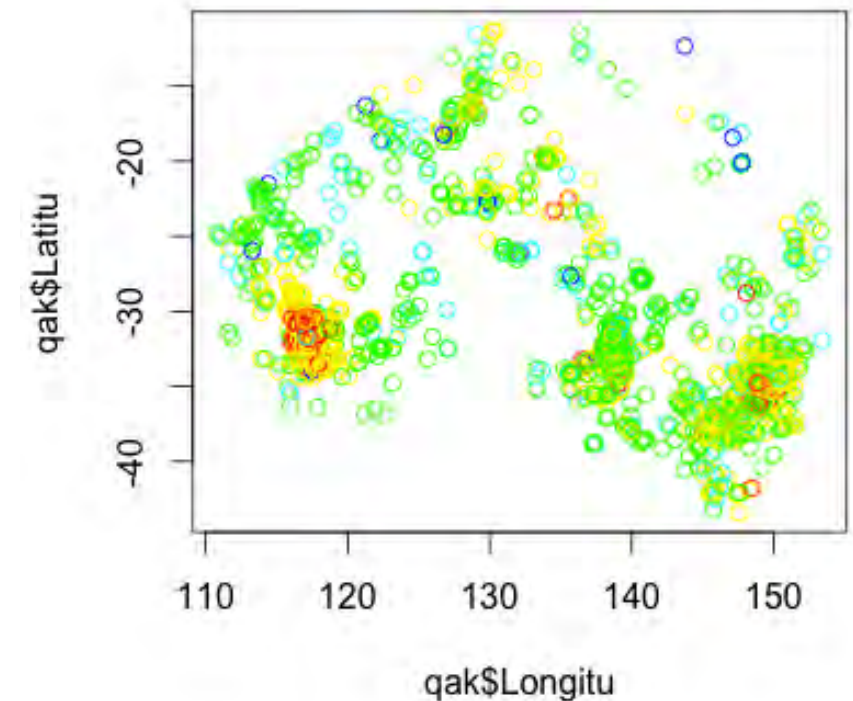
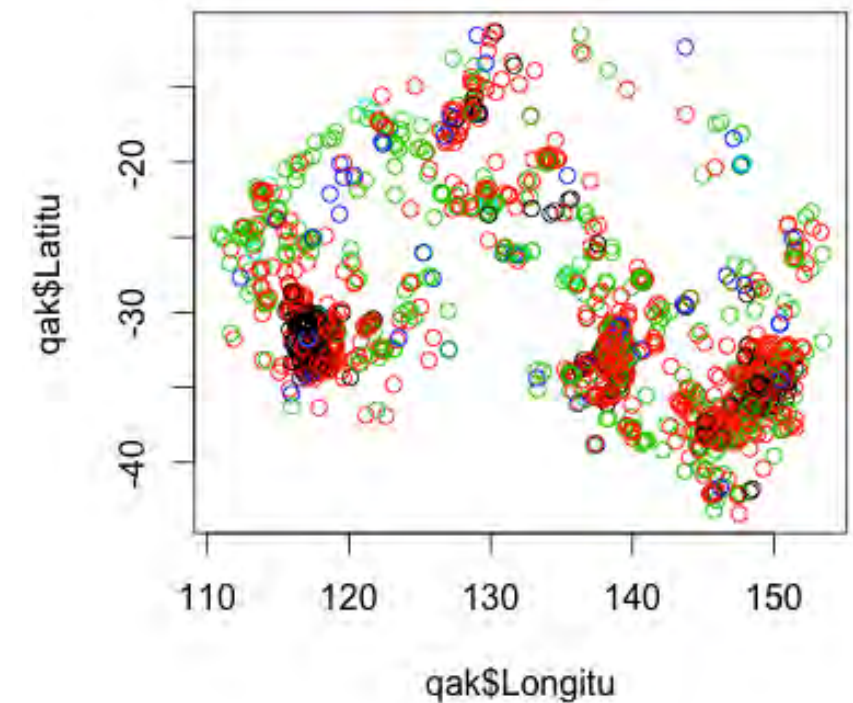
```
plot(qak$Long, qak$Lat, col=qak$Mag)
```

We can do better by defining a Rainbow colour proportional to the magnitude:

```
col<- rainbow(255,end=5/6)
colid <- function( x, range=NULL, depth=255 )
{
  if ( is.null( range ) )
    y <- as.integer(x-min(x))/(max(x)-min(x))*depth+1
  else
  {
    y <- as.integer(x-range[1])/(range[2]-range[1])*depth+1
    y[ which( y < range[1] ) ] <- 1
    y[ which( y > range[2] ) ] <- depth
  }
  y
}
plot(qak$Long, qak$Lat, col=col[colid(qak$Mag)])
```

OZ earthquakes: Code for Graphs 1 to 4

```
qak=read.csv('path_to_file', header=T) # ATTN: in R rewrite the " around path_to_file
plot(qak$Depth, qak$Mag) # Graph Magnitude vs Depth
subqak<-cbind(qak$Long, qak$Lat, qak$Mag) # Sub sampling of data
plot(subqak) # Epicenters on map
plot(qak$Long, qak$Lat, col=qak$Mag) # Coloured version
qak$Mag[is.na(qak$Mag)] <- 0 # Replace NA values by 0
```



Australia seismicity

```
# Loading libraries
```

```
library (ggplot2) # Plotting system
```

```
library (maps) # Draw country boundaries and states
```

```
library (mapproj) # Draw a grid on an existing map
```

```
library (maptools) # Draw shoreline, kml etc
```

```
# Reading data of earthquakes and map.
```

```
qak <- read.csv('/path_to_/OZ-Data_2003-2013.txt',  
as.is=T, header=T)
```

```
# Setting coordinate of plot region.
```

```
Longi <- c (min(qak$Long)-5, max(qak$Long)+5)
```

```
Lati <- c (min(qak$Lat)-5, max(qak$Lat)+5)
```

```
map <- data.frame(map(xlim = Longi, ylim = Lati)
```

```
[c("x","y")])
```

```
# Creating image with ggplot. In what follow: Long, Lat, and
```

```
# Mag are the names of the columns in you qak.csv file
```

```
p <- ggplot(qak, aes(Long, Lat))
```

```
p + geom_path(aes (x, y), map) +
```

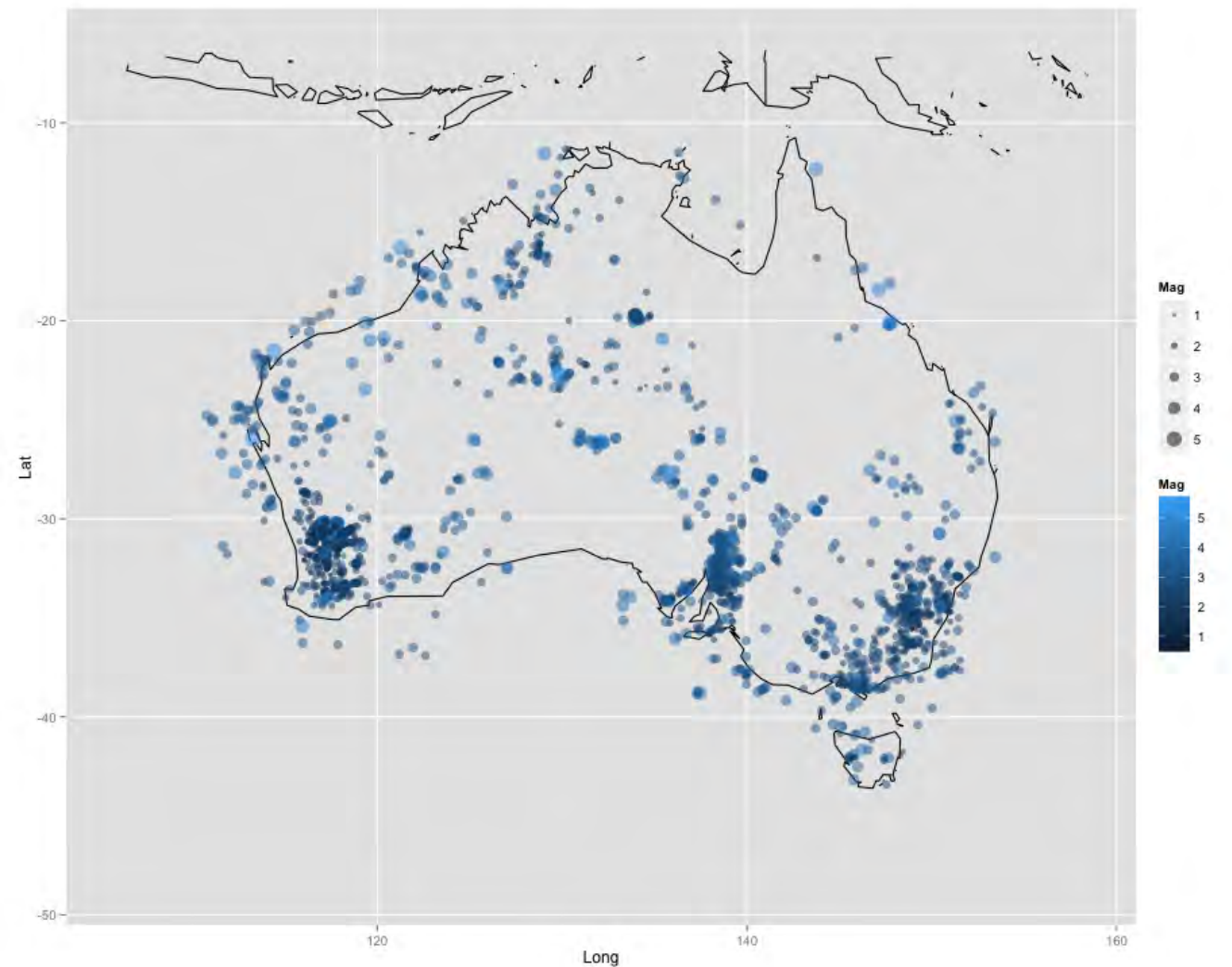
```
  geom_point(aes(size = Mag, colour = Mag), alpha = 1/2) +
```

```
xlim(Longi) + ylim(Lati)
```

```
#Explanations
```

```
geom_path(aes (x, y), map) # Australia contours
```

```
aes(size = Mag, colour = Mag) # Colored points and cool legend
```



Earthquake epicenter and magnitude from 15-2-2003 to 1-02-2013

Recipe from: <http://www.knowledgediscovery.jp/japanquakemap1/>

http://procomun.wordpress.com/2012/02/18/maps_with_r_1/

http://procomun.wordpress.com/2012/02/20/maps_with_r_2/

Loading Tsunamigenic events in the Pacific region

Goto: <http://tsun.sccc.ru/htdbpac/>

Click on *continue* (stay clear from MSIE5).

Choose *Event data*, on the next page choose *Magnitude* as selection criteria, and select magnitudes from 0 to 9.9 (default values), click *OK* and on the right panel click on *Search* to get all tsunamigenic events (142) available in the database (i.e. 65S to 65N and 80E to 50W).

Successively copy and paste the 5 pages of data (use navigation arrows from the top right panel) into your favorite text editor (for me TextMate). The headers are explained by clicking on *Legend* on the top left panel (\$Int Intensity; \$C Cause: T tectonic, V volcanic, L landslide, M meteorological, S seiches, E explosion, I impact, U unknow; \$V Validity: 4 definite, 3 probable, 2 questionable, 1 very doubtful, 0 false).

In your text editor replace all commas “,” by points “.”, and save this dataset using a descriptive name.

Australia's earthquake database

<http://www.ga.gov.au/earthquakes/searchQuake.do>

Japan earthquake database

http://www.jma.go.jp/en/quake/quake_local_index.html

New Zealand earthquake database

<http://magma.geonet.org.nz/resources/quakesearch/>

Top 10 tips to get started with R

<http://www.r-bloggers.com/top-10-tips-to-get-started-with-r/>

#Introduction to data ming with R

http://www.youtube.com/watch?feature=player_detailpage&v=6jT6Rit_5EQ

Importing Data Into R from Different Sources

From Wesley (Posted in Applied Statistics, R) , December 6, 2012

Local Column Delimited Files

```
1.file <- "c:\\my_folder\\my_file.txt"
2.raw_data <- read.csv(file, sep=","); ##'sep' can be a number of options including \t for tab delimited
3.names(raw_data) <- c("VAR1","VAR2","RESPONSE1")
```

Text File From the Internet

Data sources from the National Data Buoy Center. This example pulls data from buoy #44025 off the coast of New Jersey.

```
1.file <- "<a href='http://www.ndbc.noaa.gov/view_text_file.php?filename=44025h2011.txt.gz&dir=data/historical/stdmet/'>http://
www.ndbc.noaa.gov/view_text_file.php?filename=44025h2011.txt.gz&dir=data/historical/stdmet/</a>"
2.raw_data <- read.csv(file, header=T, skip=1)
```

Files From Other Software

From SPSS

```
1.library(foreign)
2.file <- "C:\\my_folder\\my_file.sav"
3.raw <- as.data.frame(read.spss(file))
```

From Microsoft Excel

```
1.library(XLConnect)
2.file <- "C:\\my_folder\\my_file.xlsx"
3.raw_wb <- loadWorkbook(file, create=F)
4.raw <- as.data.frame( readWorksheet(raw_wb, sheet='Sheet1'))
```

Data From Relational Databases

This example works on any SQL database. You just need to make sure you set up an ODBC connection call (in this example) MY_DATABASE.

```
1.library(RODBC)
2.channel <- odbcConnect("MY_DATABASE",
uid="username", pwd="password")
3.raw <- sqlQuery(channel, "SELECT * FROM Table1");
```

Copied and Pasted Text

Copied and Pasted Text

```
01.raw_txt <- "  
02.STATE READY TOTAL  
03.AL 36 36  
04.AK 5 8  
05.AZ 15 16  
... # many more lines here ...  
49.WI 122 125  
50.WY 12 14  
51."  
52.raw_data <- textConnection(raw_txt)  
53.raw <- read.table(raw_data, header=TRUE,  
comment.char="#", sep="")  
54.close.connection(raw_data)  
55.  
56.raw  
57.  
58.###Or the following line can be used  
59.  
60.raw <- read.table(header=TRUE, text=raw_txt
```

Structured Local or Remote Data

R can read through HTML and import from a Web site the table that you want. This example uses the XML library and pulls down the population by country in the world.

```
1.library(XML)  
2.url <- "http://en.wikipedia.org/wiki/  
List\_of\_countries\_by\_population"  
3.population = readHTMLTable(url, which=3)  
4.population
```

The source:

<http://www.r-bloggers.com/importing-data-into-r-from-different-sources/>

Processing Raster

```
library(raster)
```

```
library(rasterVis)
```

```
library(colorspace)
```

```
library(ggplot2)
```

```
ext <- extent(65, 135, 5, 55) # geographic extend to be analysed
```

From <http://neo.sci.gsfc.nasa.gov/Search.html?group=64>

download the world population map (chose GeoTIFF format),
rename and load into R. In R read the data, select a subset, and
replace the 99999 with NA.

```
pop <- raster('path_to_world_popul.TIFF')
```

```
pop <- crop(pop, ext)
```

```
pop[pop==99999] <- NA
```

```
pTotal <- levelplot(pop, zscaleLog=10, par.settings=BTCTheme)
```

```
pTotal
```

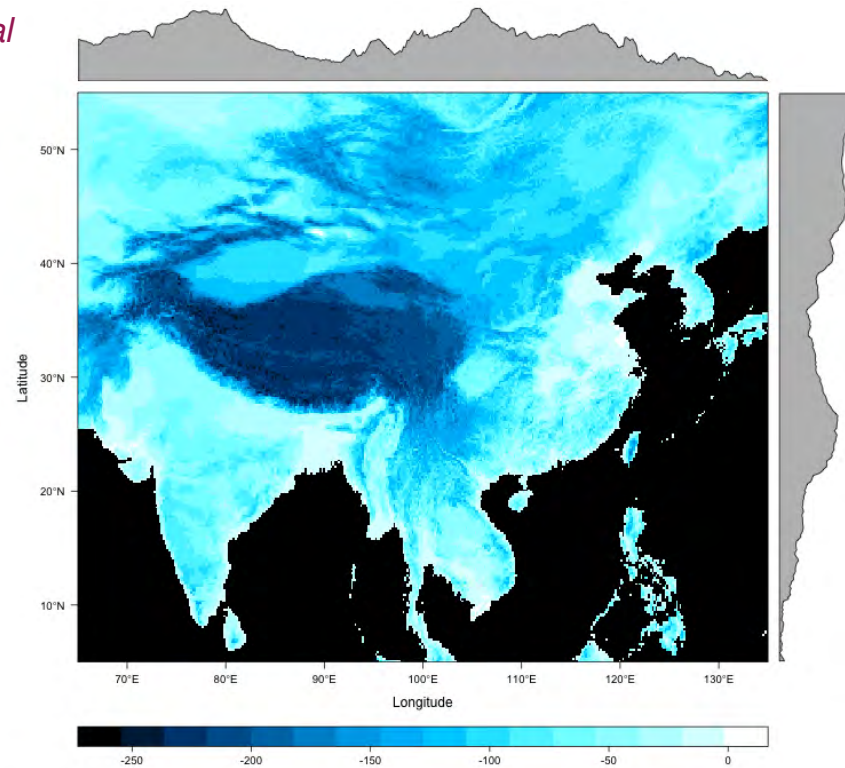
From <http://neo.sci.gsfc.nasa.gov/Search.html?group=29>

download the world topography map (GeoTIFF format), rename
and load in R.

```

topo <- raster('path_to_world_topog.TIFF')
topo <- crop(topo, ext)
topo[topo %in% c(0, 254)] <- NA
topoTotal <- levelplot(topo, par.settings=BTCTheme)
topoTotal

```



To bin topo data and change colour scheme:

```

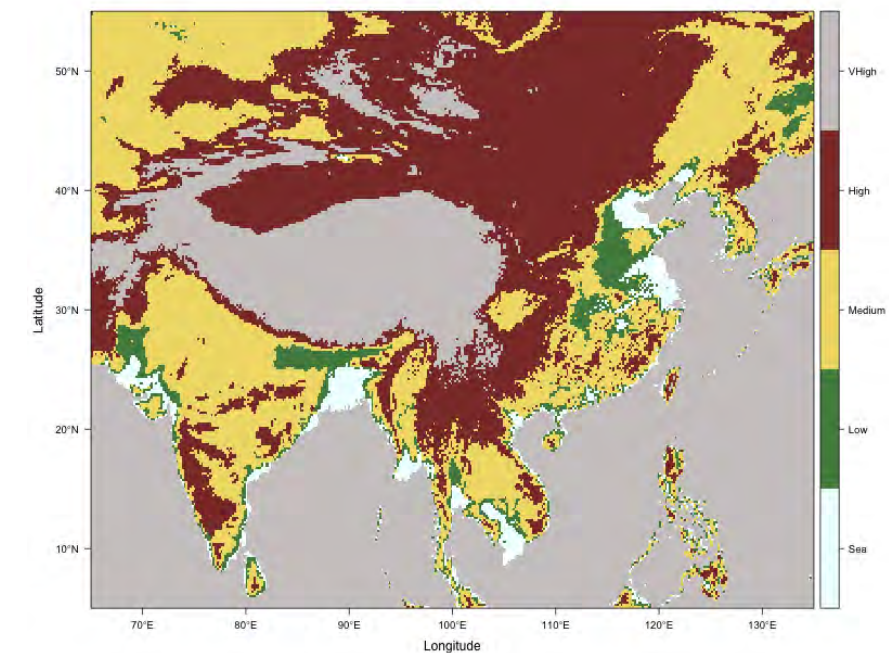
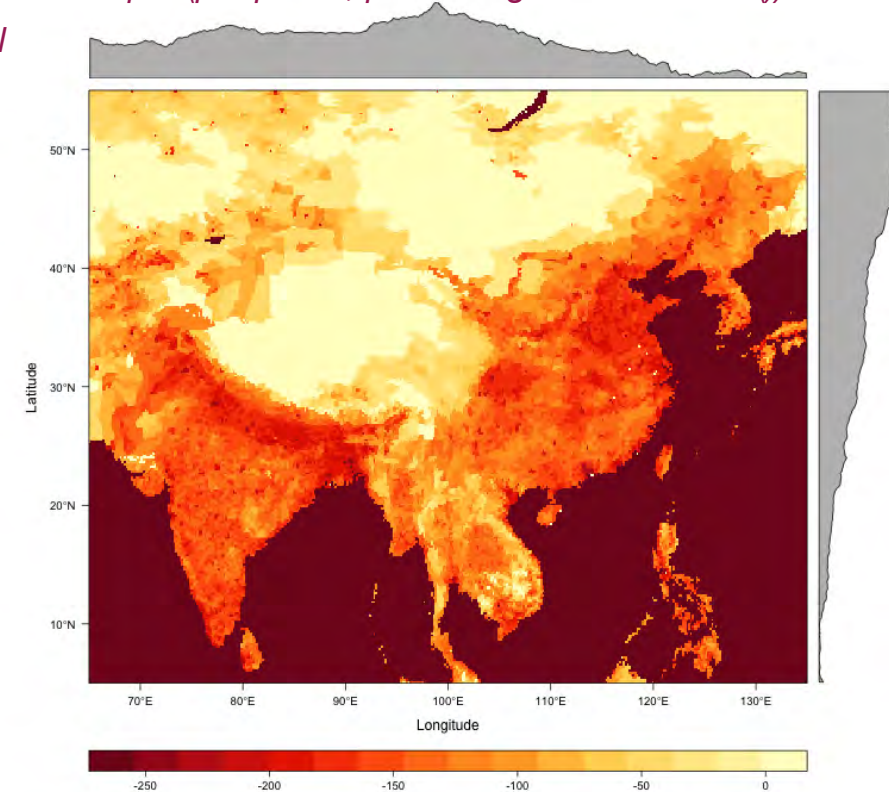
topoBin <- cut(topo, c(0, 14, 28, 72, 144, 255))
classes <- c('Sea', 'Low', 'Medium', 'High', 'VHigh')
pal <- c('azure1', 'palegreen4', 'lightgoldenrod', 'indianred4', 'snow3')
nClasses <- length(classes)
rng <- c(minValue(topoBin), maxValue(topoBin)) ## breaks of the color key
my.at <- seq(rng[1]-1, rng[2]) ## the labels vertical centered
my.labs.at <- seq(rng[1], rng[2])-0.5
topoBinImage <- levelplot(topoBin, at=my.at, margin=FALSE,
col.regions=pal,colorkey=list(labels=list(labels=classes,at=my.labs.at)))
topoBinImage

```

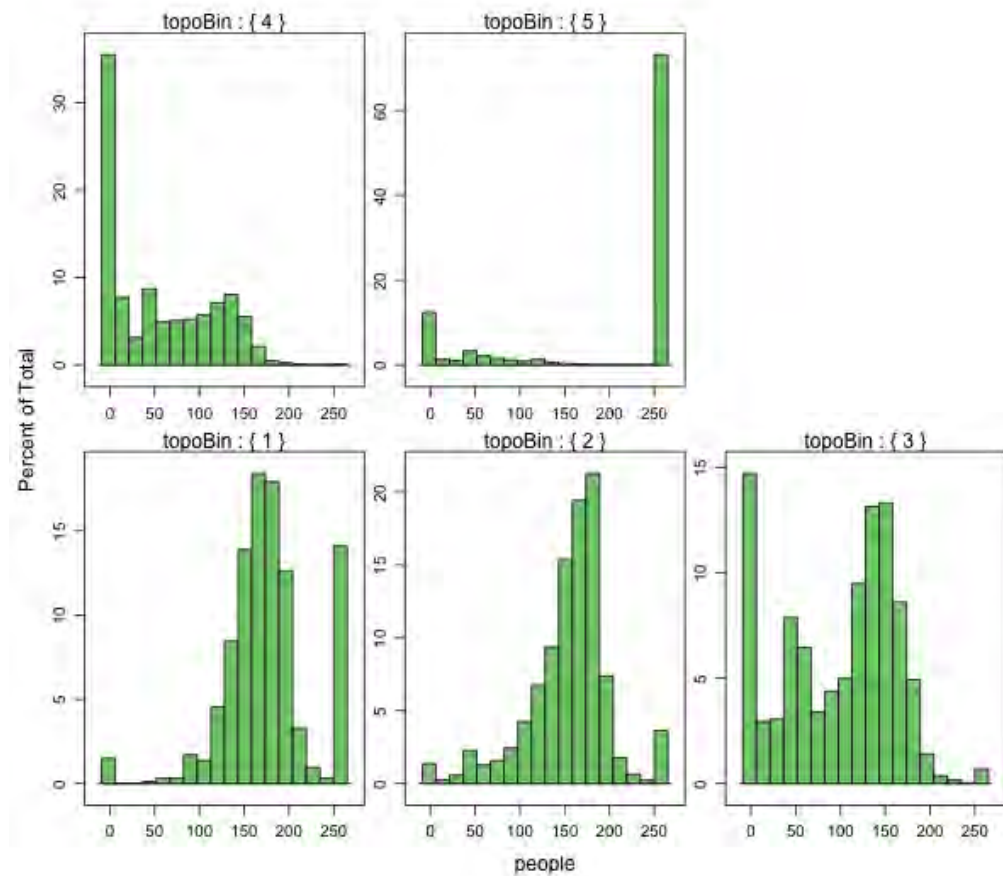
```

people <- raster('path_to_world_popul.TIFF')
people <- crop(people, ext)
people[people==99999] <- NA
peopleTotal <- levelplot(people*-1, par.settings=rasterTheme())
peopleTotal

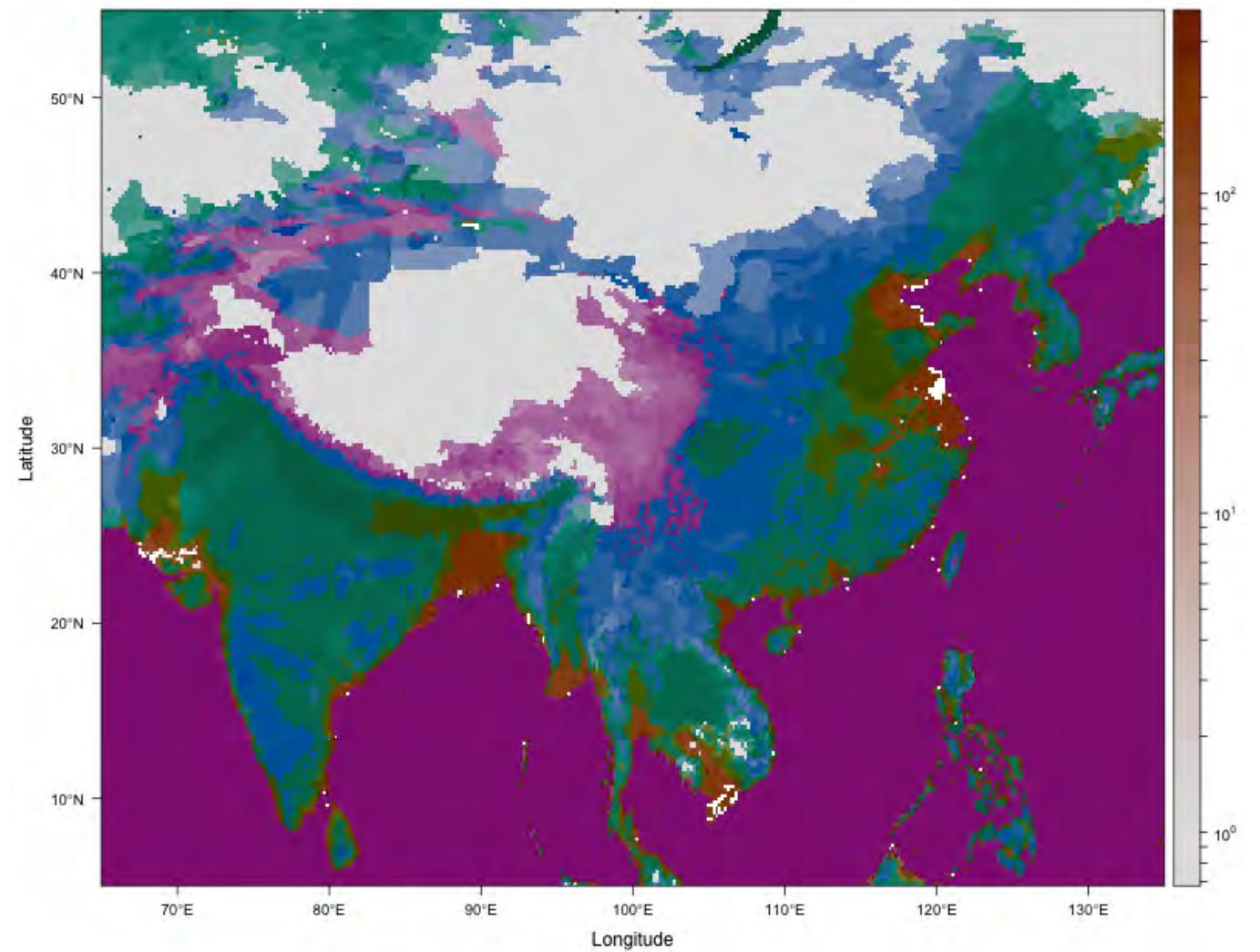
```



```
s <- stack(people, topoBin)
names(s) <- c('people', 'topoBin')
histogram(~people|topoBin, data=s,
scales=list(relation='free'),strip=strip.custom(strip.levels=TRUE))
```



```
pList <- lapply(1:nClasses, function(i){landSub <- topoBin
landSub[!(topoBin==i)] <- NA
popSub <- mask(people, landSub)
step <- 360/nClasses ## distance between hues
pal <- rev(sequential_hcl(16, h = (360 + step*(i-1))%%360))
pClass <- levelplot(popSub, zscaleLog=10, at=at, col.regions=pal,
margin=FALSE))
p <- Reduce('+', pList)
p
```



```

> addTitle <- function(legend, title){
+   titleGrob <- textGrob(title, gp=gpar(fontsize=8), hjust=1, vjust=1)
+   legendGrob <- eval(as.call(c(as.symbol(legend$fun), legend$args)))
+   ly <- grid.layout(ncol=1, nrow=2, widths=unit(0.9, 'grobwidth',
data=legendGrob))
+   fg <- frameGrob(ly, name=paste('legendTitle', title, sep='_'))
+   pg <- packGrob(fg, titleGrob, row=2)
+   pg <- packGrob(pg, legendGrob, row=1)
+ }
> for (i in seq_along(classes)){
+   lg <- pList[[i]]$legend$right
+   lg$args$key$labels$cx=ifelse(i==nClasses, 0.8, 0)
+   pList[[i]]$legend$right <- list(fun='addTitle',
+                                   args=list(legend=lg, title=classes[i]))
+ }
> legendList <- lapply(pList, function(x){
+   lg <- x$legend$right
+   clKey <- eval(as.call(c(as.symbol(lg$fun), lg$args)))
+   clKey
+ })
> packLegend <- function(legendList){
+   N <- length(legendList)
+   ly <- grid.layout(nrow = 1, ncol = N)
+   g <- frameGrob(layout = ly, name = "mergedLegend")
+   for (i in 1:N) g <- packGrob(g, legendList[[i]], col = i)
+   g
+ }
> p$legend$right <- list(fun = 'packLegend', args = list(legendList =
legendList))
> p

```

```

#####
## This piece of code is adapted from the following source
## Displaying time series, spatial and space-time data with R:
## By by Oscar Perpiñán. Ed Chapman&Hall/CRC.
#####
#####
## Raster maps:
## From http://neo.sci.gsfc.nasa.gov download the world population map
## and world topographic map (chose GeoTIFF format).
## Rename to world_popul.TIFF and world_topog.TIFF
## and drop them in a folder.
#####

#####
## Diverging palettes: The following defines the colour palettes used in this
## project.
#####

pdf(file="/Users/patricerey/Documents/Teaching/2013/2111-GIS/
World_Maps/figs/leveplotSISavOrig.pdf")
library(colorspace)
library(maps)
library(maptools)
library(classInt)
library(sp)
library(maptools)
library(rgdal)
library(raster)
library(rasterVis)
library(classInt)

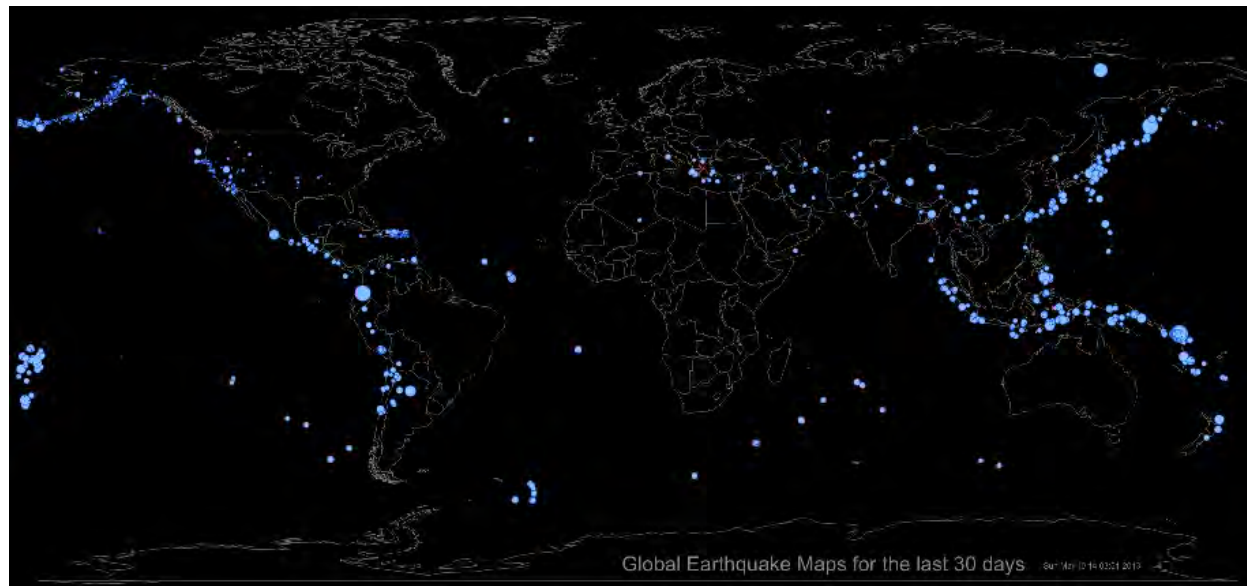
SISav <- raster('data/SISav')
levelplot(SISav)
dev.off()

meanRad <- cellStats(SISav, 'mean')
SISav <- SISav - meanRad

```

<https://github.com/oscarperpinan/spacetime-vis/blob/master/code/thematicMaps.R>

Global Earthquake Map of past 30 days Earthquake of Mag>2.5



```
# From Sean Mulcahy:
# http://www.r-bloggers.com/the-global-earthquake-desktop/

# load the maps library
library(maps)

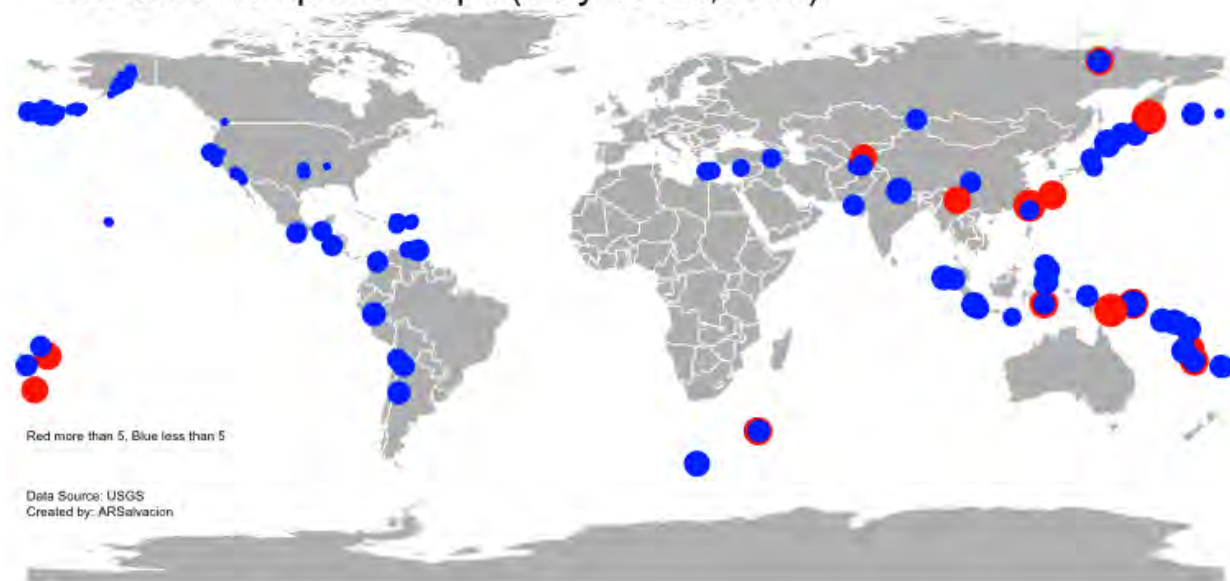
# get the earthquake data from the USGS
#http://earthquake.usgs.gov/earthquakes/feed/csv/2.5/month.txt

eq <- read.csv("/Users/patricerey/Documents/Teaching/2013/2111-GIS/Pract_Christchurch/month.csv", sep = ",", header = TRUE)

# size the earthquake symbol areas according to magnitude
radiuses <- 100 * exp($Magnitude)
```

Global Earthquake Map another map.

Global Earthquake Maps (May 15-23,2012)



```
# From Arsalvacion:
#http://www.r-bloggers.com/r-nold-2012-05-23-054800/

usgseq<-"http://earthquake.usgs.gov/earthquakes/recenteqsww/Quakes/quakes\_all.html"

weq1 = readHTMLTable(usgseq, header=T,
which=1,stringsAsFactors=F)
weq2 = readHTMLTable(usgseq, header=T,
which=2,stringsAsFactors=F)
weq3= readHTMLTable(usgseq, header=T, which=3,stringsAsFactors=F)
weq4 = readHTMLTable(usgseq, header=T,
which=4,stringsAsFactors=F)
weq5 = readHTMLTable(usgseq, header=T,
which=5,stringsAsFactors=F)
weq6 = readHTMLTable(usgseq, header=T,
```

GEOmap for Geology

GEOMAP INFO ...

1. Website

<http://www.unc.edu/~leesj/index.html>

2. Source of data and more

<http://www.ruf.rice.edu/~ben/gmt.html>

<http://rgm3.lab.nig.ac.jp/RGM/>

GEOmap is an R package developed by Jonathan M. Lees at the University of North Carolina. *GEOmap* overlaps somehow with other package such as *maps*. However, *GEOmap* has a number of attributes making the process of drawing geological maps easier. *GEOmap* works if *geomapdata*, which must be loaded independently.

GEOmap offers 7 different projections

Fom J. Lees, 2012: GEOMap, mapping and geology in R
<http://www.unc.edu/~leesj/index.html>

```
library(GEOmap)
require('geomapdata')
```

```
# Set some options
options(continue = " ")
kliuLL = c(56.056000, 160.640000)
PROJ =setPROJ(type=2, LAT0=kliuLL[1], LON0= kliuLL[2] ,
LATS=NULL, LONS=NULL)
```

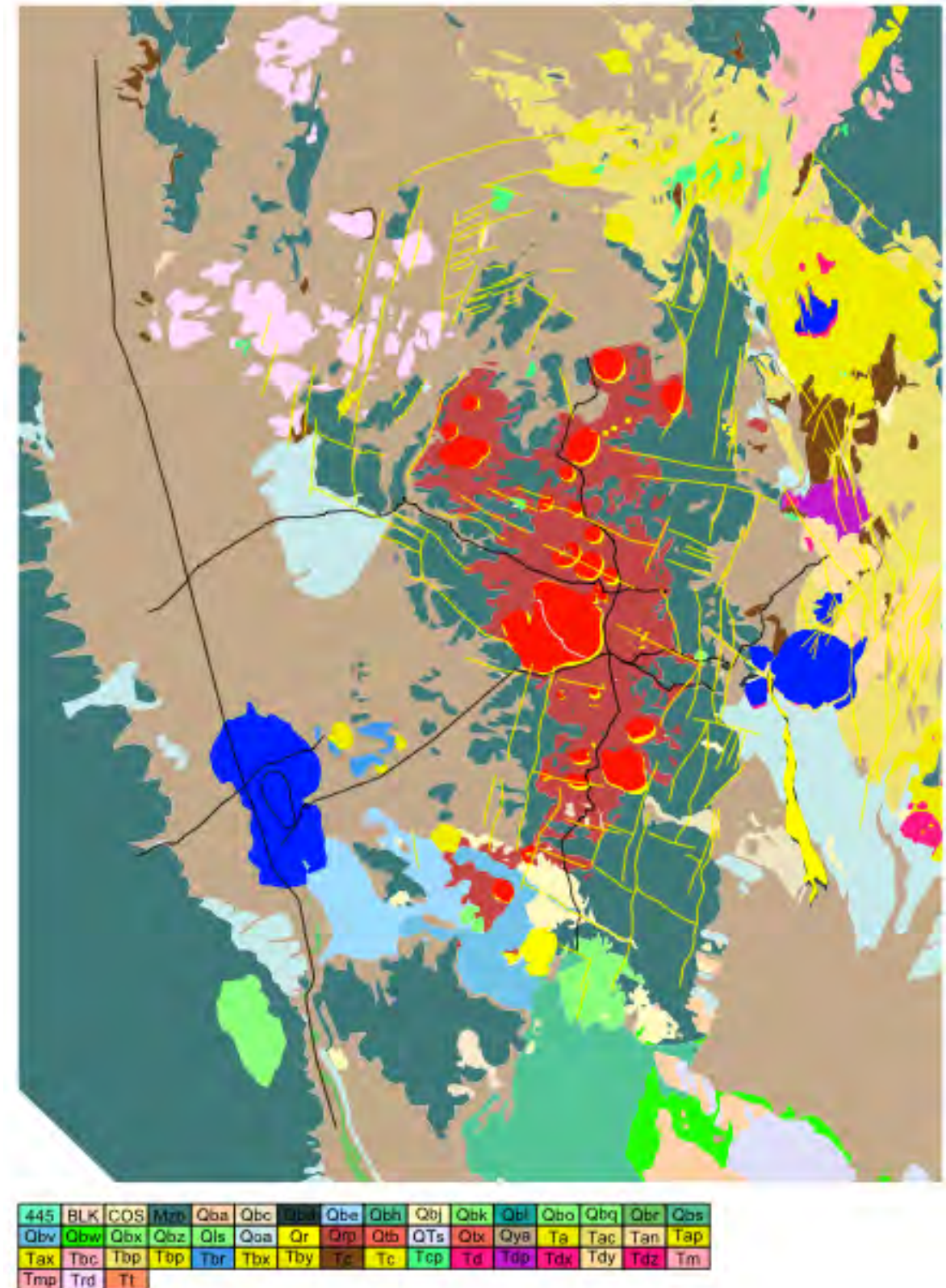
```
# Load data and plot with no projection
data(kammap)
plotGEOmap(kammap, add=FALSE, asp=1)
```

```
# With set projection (2=mercator spherical)
plotGEOmapXY(kammap, PROJ=PROJ, add=FALSE, xlab="km",
ylab="km")
```

```
# Load data
data(cosomap)
data(faults)
data(hiways)
data(owens)
data(cosogeol)

## cosocolnumbers = cosogeol$STROKES$col+1
```

```
# Successively plot features
proj = cosomap$PROJ
plotGEOmapXY(cosomap, PROJ=proj, add=FALSE, ann=FALSE,
axes=FALSE)
```



```
# From J. Lees, 2012: GEOmap, mapping and geology in R
```

```
library(GEOmap)
require('geomapdata')
```

```
# Set region of interest and projection system
```

```
options(continue = " ")
```

```
kliuLL = c(56.056000, 160.640000)
```

```
PROJ = setPROJ(type=2, LAT0=kliuLL[1], LON0= kliuLL[2] ,
LATS=NULL, LONS=NULL)
```

```
# Read data
```

```
eqs = read.csv('/Users/patricerey/Documents/Teaching/2013/2111-
GIS/Volc_LLZ_sm.csv', header=T)
```

```
ifuji = grep('Fuji', eqs$name)
```

```
PROJ = setPROJ(type=2, LAT0=eqs$lat[ifuji], LON0=eqs$lon[ifuji])
```

```
# Define an inset box
```

```
LL = XY.GLOB(c(-150, 150), c(-150,150), PROJ =PROJ)
```

```
FUJIAREA = c(LL$lon[1], LL$lat[1], LL$lon[2], LL$lat[2])
```

```
# First map
```

```
data("japmap", package="geomapdata")
```

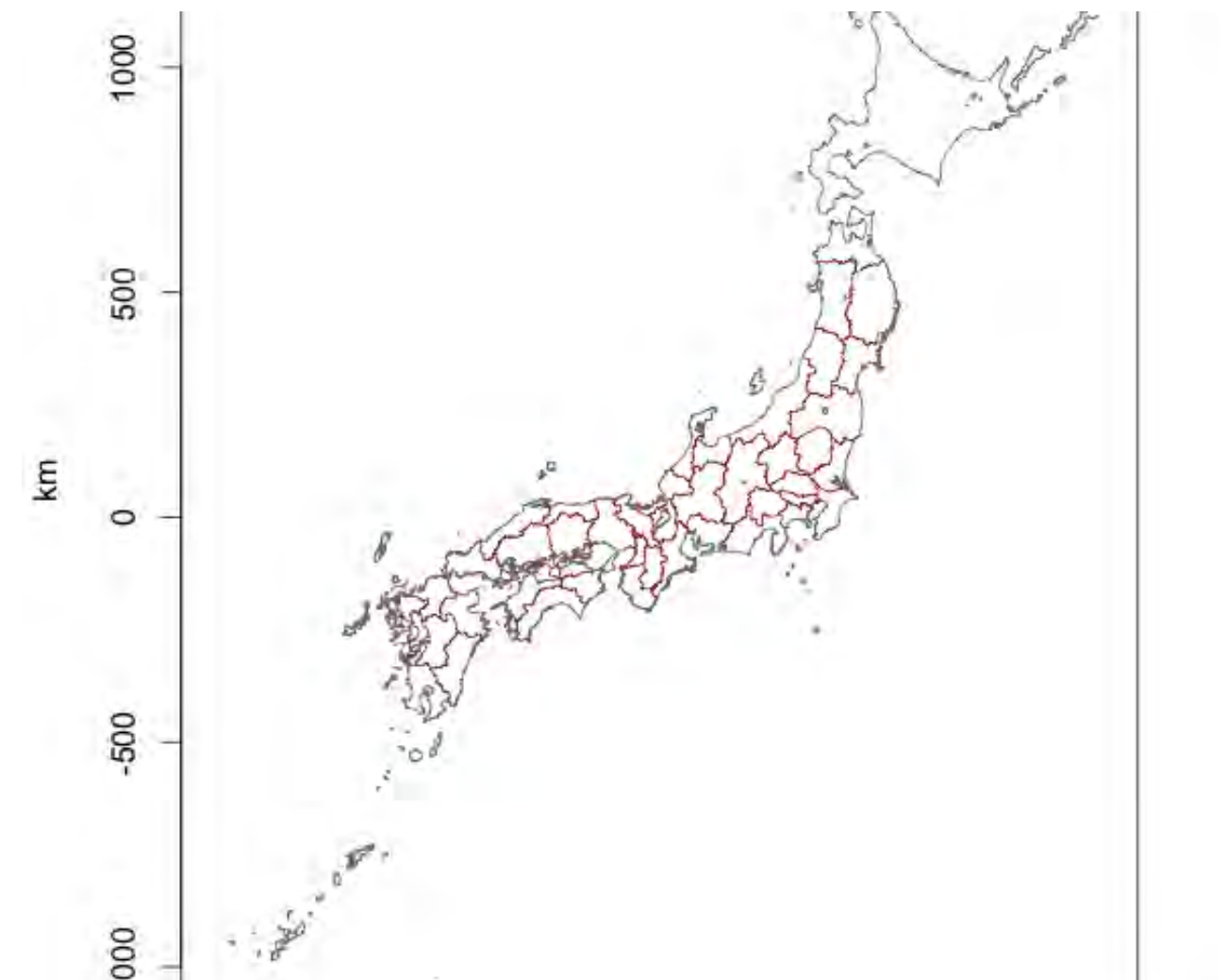
```
plotGEOmapXY(japmap, PROJ=PROJ, xlab="km", ylab="km" )
```

```
# Add volcanoes and box
```

```
pointsGEOmapXY(eqs$lat, eqs$lon, PROJ=PROJ, col='red', pch=2,
cex=.5)
```

```
rect(-150, -150, 150,150)
```

Gallery 4.1 GEOmap



• • • • •

R scripts: To copy, paste and run in the R console

Map 1: Using GEOmap

```
library(GEOmap)
require('geomapdata')
data("japmap", package="geomapdata")

# Set region of interest and projection system (here mercator shp)
options(continue = " ")
Akai = c(37.5, 140) #coordinates of the map center
```

Map 2: Some cosmetic modifications

```
library(GEOmap)
require('geomapdata')
data("worldmap", package="geomapdata")

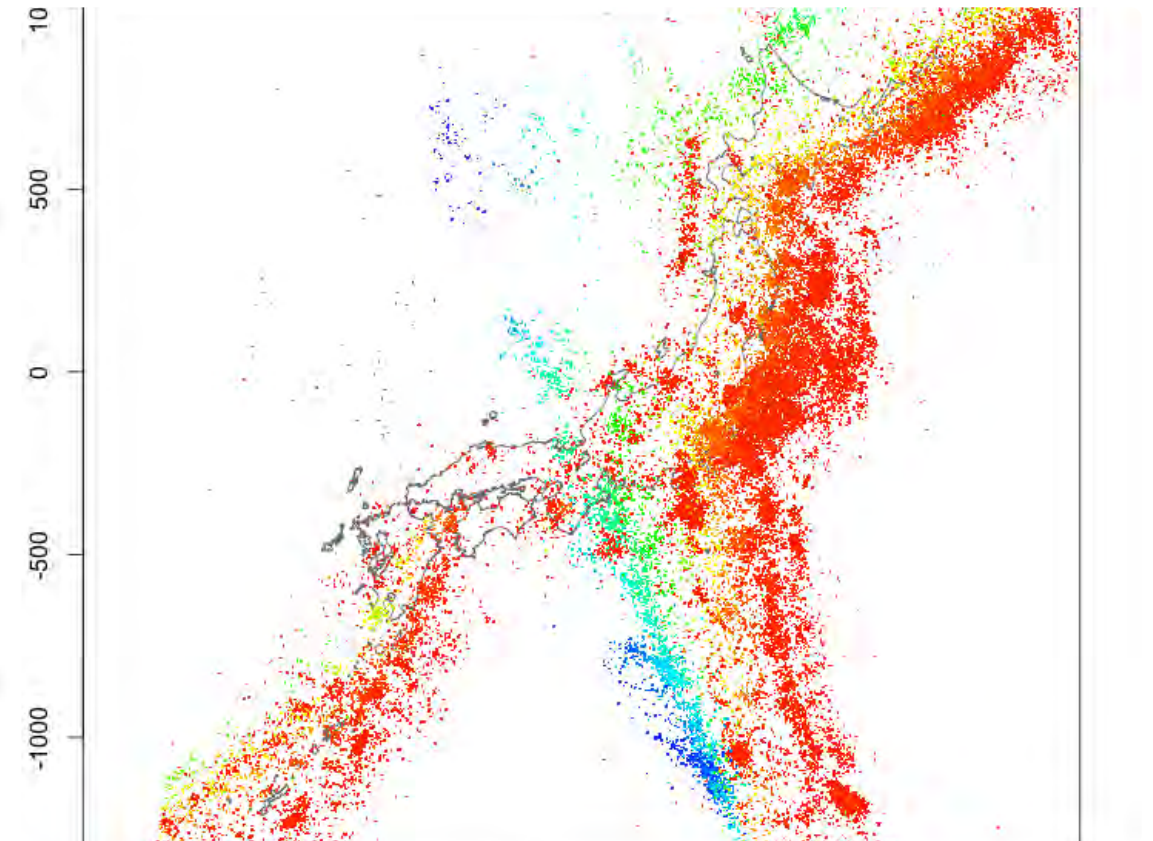
# Set region of interest and projection system
options(continue = " ")
Akai = c(37.5, 140)
```

Map 3: Some more cosmetic modifications

```
library(GEOmap)
require('geomapdata')
data("worldmap", package="geomapdata")

# Set region of interest and projection system
options(continue = " ")
Akai = c(37.5, 140)
```

Gallery 4.2 GEOmap



Map 1

• • • •

Map 4: Some cosmetic modifications

```
library(GEOmap)
require('geomapdata')
data("worldmap", package="geomapdata")

# Set region of interest and projection system
options(continue = " ")
Akai = c(37.5, 140)
```

Map 5: Australia seismicity

```
require('geomapdata')
data("worldmap", package="geomapdata")

# Set region of interest and projection system
options(continue = " ")
Akai = c(-23.7, 133.87)
PROJ = setPROJ(type=2, LAT0=Akai[1], LON0= Akai[2],
               LAT5=NULL, LONS=NULL)

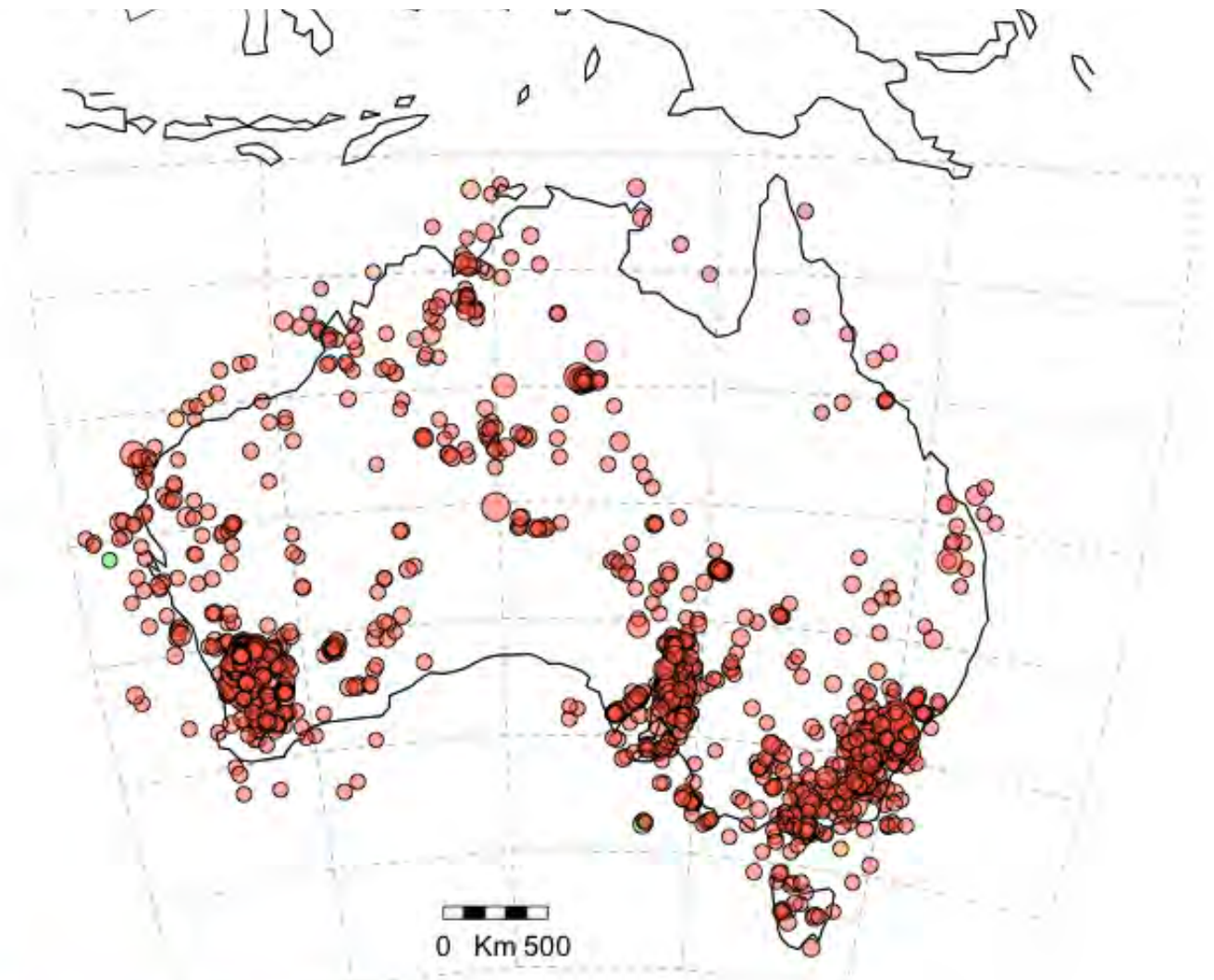
#Coordinates of the clip to be applied to worldmap
LL = XY.GLOB(c(-2500, 2500), c(-2500, 2500), PROJ =
             PROJ)
OZREA = c(LL$lon[1], LL$lat[1], LL$lon[2], LL$lat[2])
eqs = read.csv('/Users/patricerey/Documents/Teaching/
2013/2111-GIS/OZ-Data_2003-2013.txt', header=T)

eqs$Depth[eqs$Depth>150] <- 150

# Earthquakes colour function of depth
rcol = rainbow(120)
ecol = 1 + floor(99 * (eqs$Depth - min(eqs$Depth))/(70 -
min(eqs$Depth)))

# Earthquake size (polygone) function of magnitude
EXY = GLOB.XY(eqs$Lat, eqs$Long, PROJ)
eqs$Mag[is.na(eqs$Mag)] <- 0
esiz = exp(eqs$Mag)
rsiz = RESCALE(esiz, 0.04, 0.2, min(esiz), max(esiz))
ordsiz = order(rsiz, decreasing = TRUE)
```

Gallery 4.3 GEOMap



Map 5

Animated ...

How cool is that: <http://www.vizworld.com/tag/earthquake/>

Map 7: New Zealand seismicity

Data from <http://magma.geonet.org.nz/resources/quakesearch/>

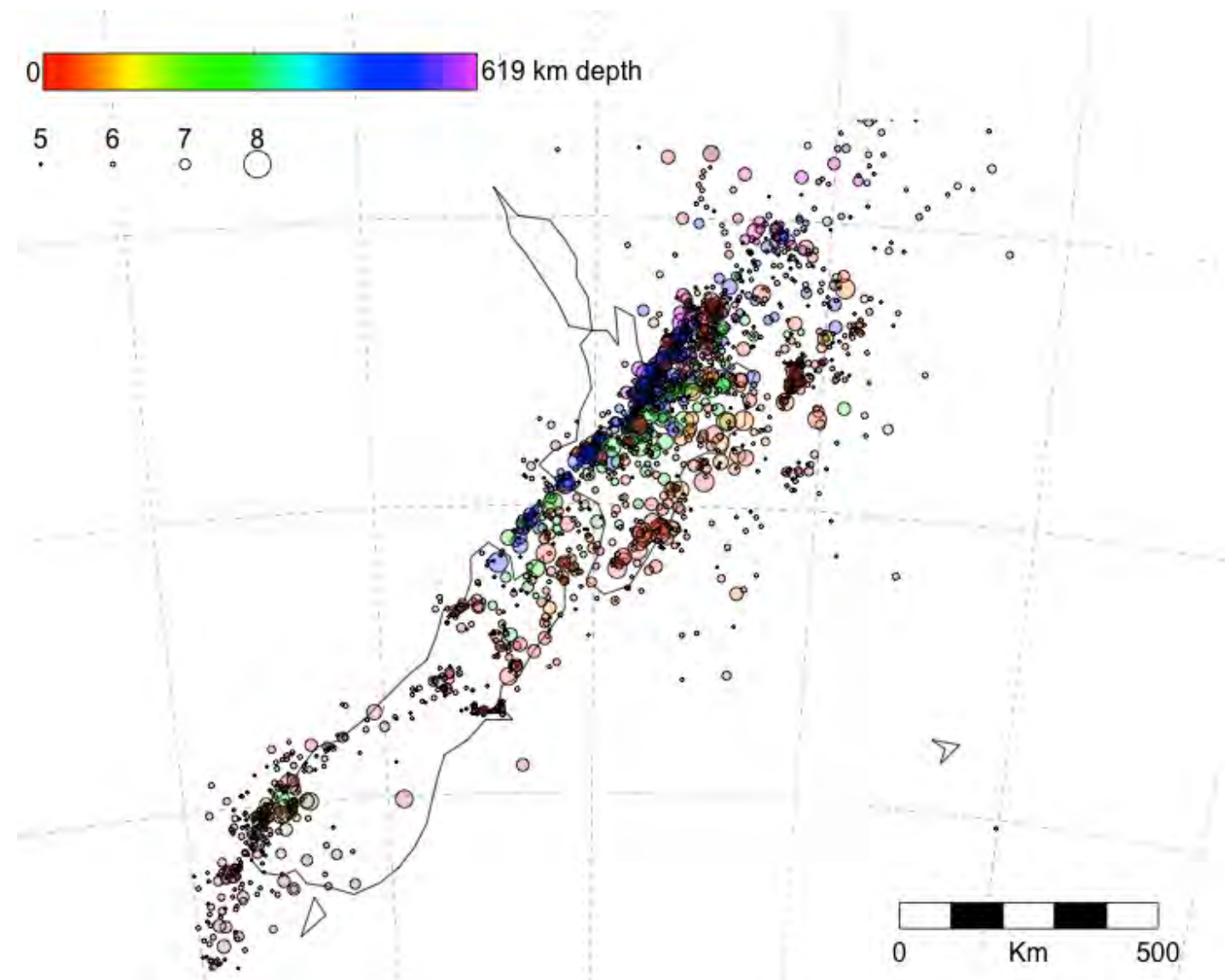
```
library(GEOmap)
require('geomapdata')
data("worldmap", package="geomapdata")

# Set region of interest and projection system
options(continue = " ")
PigBay = c(-41.1, 174.3)
PROJ = setPROJ(type=2, LAT0=PigBay[1], LON0= PigBay[2],
LATS=NULL, LONS=NULL)

#Coordinates of the clip to be applied to worldmap
LL = XY.GLOB(c(-750, 800), c(-800, 800), PROJ = PROJ)
NZAREA = c(LL$lon[1], LL$lat[1], LL$lon[2], LL$lat[2])
eqs = read.csv("/Users/patricerey/Documents/Teaching/
2013/2111-GIS/Pract_Christchurch/NZ-1913-2013.csv",
header=T)

# Earthquakes colour function of depth
rcol = rainbow(120)
ecol = 1 + floor(99 * (eqs$DEPTH - min(eqs$DEPTH))/(max(eqs
$DEPTH) - min(eqs$DEPTH)))

# Earthquake size (polygone) function of magnitude
# EXY = GLOB.XY(eqs$LAT, eqs$LONG, PROJ)
# Extract only earthquake >= 5.0
eqs <- subset(eqs, eqs$MAG >= 5 & eqs$MAG < 8)
eqs$MAG[is.na(eqs$MAG)] <- 0
```



All seisms > 5, in the region of New Zealand, over the past 100 years.
Size of epicenters proportional to magnitude, their color is function of their hypocenter depth.

Modify the script to plot the earthquakes from 40 and 100 km depth.

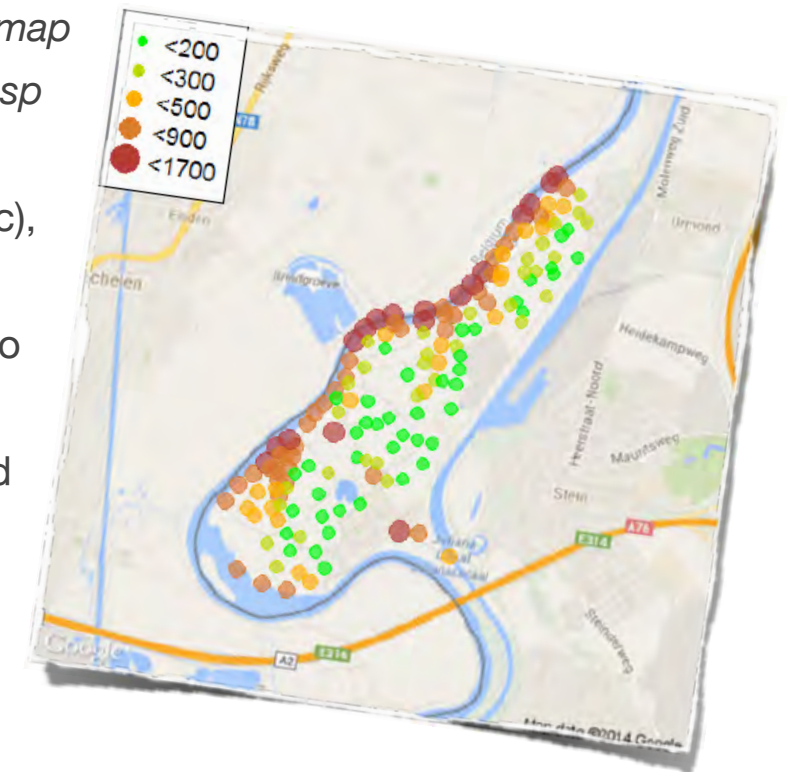
Web cartography with R and Google map

INFO & KEY REFERENCES

1. **RgoogleMaps:** A package to plot data onto maps from Google as well as *OpenStreetMap* servers for static maps in the form of PNGs. Markus Loecher and Karl Ropkins, 2015. *RgoogleMaps and loa: Unleashing R Graphics Power on Map Tiles. Journal of Statistical Software*, 63, issue2.
2. **plotGoogleMaps:** A package to plot data onto interactive web maps from Google. This package is based on the Google Maps Application Programming Interface (API) the html file with Cascading Style Sheet (CSS) styling and Java Script functionality. *plotGoogleMaps* is developed by Milan Kilibarda and Branislav Bajat, from the University of Belgrade.
http://e-science.amres.ac.rs/TP36035/wp-content/uploads/2012/06/PLOTGOOGLEMAPS_full.pdf
3. **ggmap:** A package which combines *RgoogleMaps* and *ggplot2*. From David Kahle and Hardley Wickham.
<http://stat405.had.co.nz/ggmap.pdf>

Colleagues of mine were putting together a practical exercise for intermediate chemistry students, which consisted of measuring the copper levels in samples of drinking water collected at various locations on the Darlington campus of the University of Sydney. They wanted students to pool their data together and produce a GIS map, perhaps using Google maps. A quick dive into the R ecosystem leads to *RgoogleMaps*, *plotGoogleMaps* and *ggmap*, three little gems to plot and process spatial data on base maps supplied by Google.

RgoogleMaps, *plotGoogleMaps* and *ggmap* build on top of other packages such as *sp* (handle spatial data), *rgdal* (handle GIS data, Coordinate Reference Systems etc), so make sure to turn on “*install dependencies*” when installing these two packages. These packages create map overlays, whose parameters are passed via HTML to Google map API, returning the overlays over Google base maps, or, in the case of *plotGoogleMap*, an interactive Google map on a web browser.



plotGoogleMaps in 6 lines of R.

Let's first create a synthetic dataset; a simple column-based file containing the longitude and latitude of 100 samples and corresponding chemical analyses for zinc, lead, uranium and polonium. The samples were collected on the Darlington campus of The University of Sydney.

To create a grid covering Sydney Uni Darlington campus (using decimal degree):

```
latitude<-runif(100, -33.892500, -33.883600)
longitude<-runif(100, 151.178000, 151.195000)
```

To create random chemical data (ppm)

```
Zinc<-runif(100, 0.5, 50)
Lead<-runif(100, 0.0015, 0.15)
Uranium<-runif(100, 0.3, 30)
Polonium<-runif(100, 0.0001, 0.01)
```

Let's put grid and data together into a file, and save it in a comma separated value (csv) format.

```
SyntheticChemDataUSyd <- cbind(latitude, longitude, Zinc, Lead,
Uranium, Polonium)
write.csv(SyntheticChemDataUSyd, file =
"SyntheticChemDataUSyd.csv", row.names=FALSE)
```

Let's plot the data onto an interactive Google map:

Set your working directory:

```
setwd('/path_to_data_folder/datafolder/')
```

Load the *plotGoogleMaps* package:

```
require(plotGoogleMaps)
```

Read the dataset:

```
ChemDataDecilatlon<-read.csv("SyntheticChemDataUSyd.csv",
header=T, sep=",")
```

Point the names of the columns holding the coordinates

```
coordinates(ChemDataDecilatlon)<--~longitude+latitude
```

Assign geographic projection of the dataset, here decimal lat long:

```
proj4string(ChemDataDecilatlon) <- CRS("+init=epsg:4326")
```

Convert lat long into NSW coordinate (epsg:3308) to use in Google map

```
ChemData <- spTransform(ChemDataDecilatlon, CRS("+init=epsg:
3308"))
```

... and plot in Google Map. For more info: ?bubbleGoogleMaps

```
m1<-bubbleGoogleMaps(ChemData, zcol='Zinc', layerName='USyd
water: zinc', filename='myZincMap.htm', key.entries =
quantile(ChemData$Zinc, (1:8)/8), zoom=16, shape='c',
max.radius=20, strokeColor='blue')
```

The Google map for zinc:

... Try these:

```
m2<-bubbleGoogleMaps(ChemData, zcol='Lead',
layerName='USyd water: lead',
filename='myLeadMap.htm', key.entries =
quantile(ChemData$Lead, (1:6)/6), zoom=16,
shape='t',max.radius=20, strokeColor='red',
add=FALSE)
```

```
m3<-bubbleGoogleMaps(ChemData, zcol='Polonium',
layerName='USyd water: polonium',
filename='myPoloniumMap.htm', key.entries =
quantile(ChemData$Polonium, (1:4)/4), zoom=16,
shape='q',max.radius=20, strokeColor='red',
add=FALSE)
```

```
m4<-bubbleGoogleMaps(ChemData,
zcol='Uranium', layerName='USyd water: uranium',
filename='myUraniumLead.htm', key.entries =
quantile(ChemData$Uranium, (1:6)/6), zoom=16,
max.radius=20, strokeColor='red', add=FALSE)
```

Something a bit different ...

```
m5<-segmentGoogleMaps(ChemData,
zcol=c('Lead','Zinc','Polonium','Uranium'),
mapTypeId='ROADMAP', filename='myChemMap.htm',
max.radius=20, colPalette=c('#E41A1C','#377EB8', '#B3B3B3',
'#66C2A5'), strokeColor='black')
```



Statistical Analyses and Infography

R FOR STATS

1. Communicating through data

A 20 minutes talk from David Candless

http://www.ted.com/talks/david_mccandless_the_beauty_of_data_visualization.html

A 20 minutes talk from Hans Rosling

http://www.ted.com/talks/hans_rosling_shows_the_best_stats_you_ve_ever_seen.html

2. GoogleVis

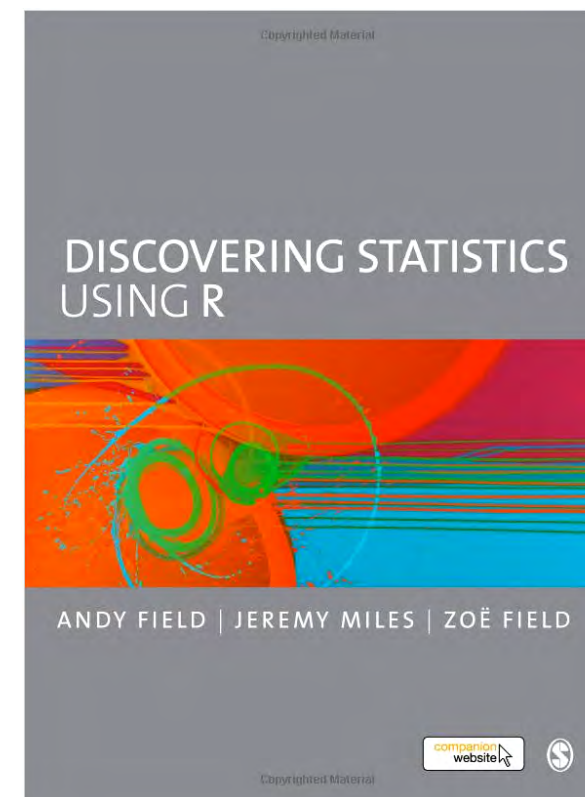
<http://code.google.com/p/google-motion-charts-with-r/>

Others

<http://geostat-course.org/Software>

An R Introduction to Statistics:

<http://www.r-tutor.com/elementary-statistics>



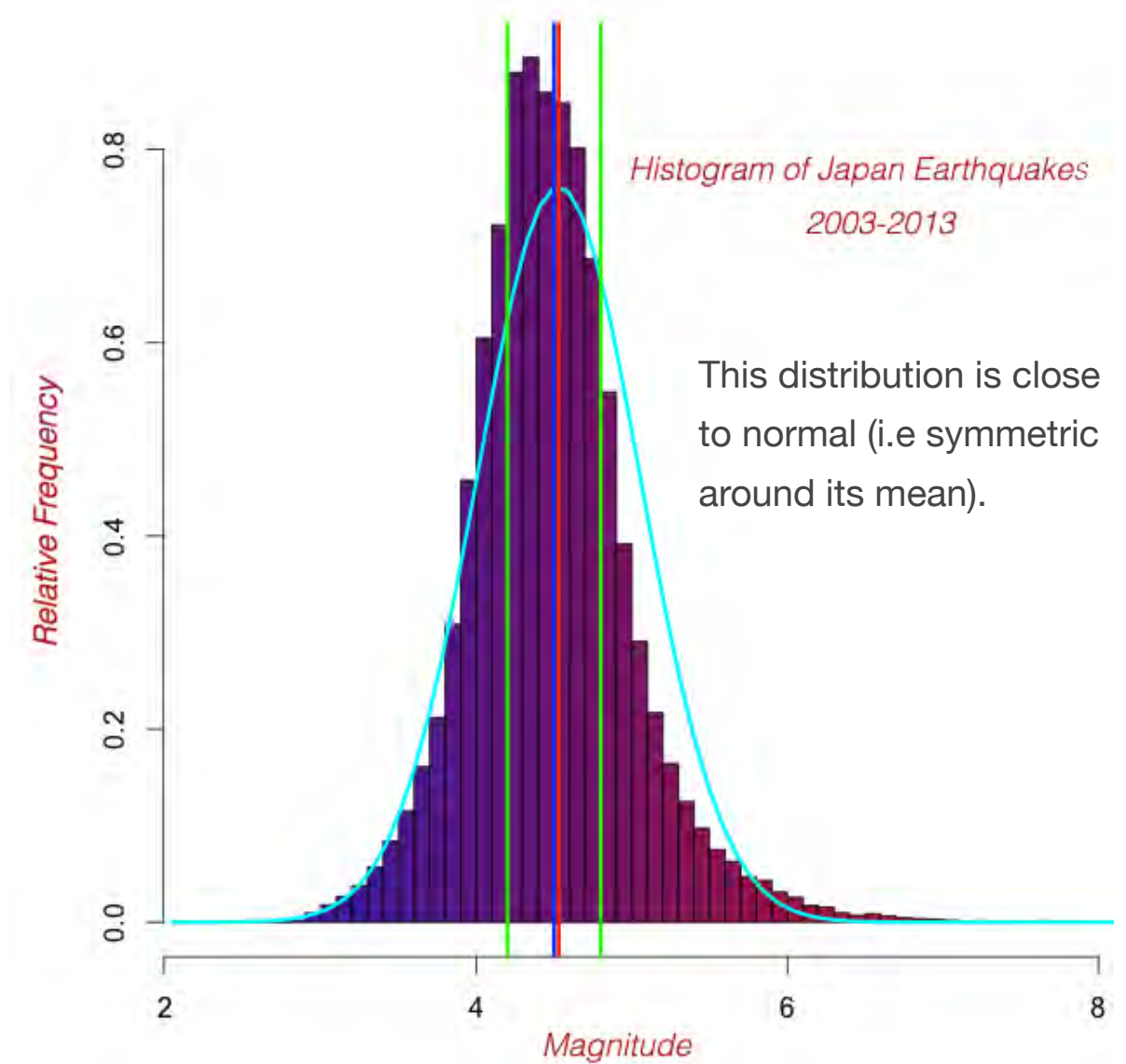
Exploring data? http://ktmaps.blogspot.com.au/2010_07_01_archive.html

Heat map: <http://qgis.spatialthoughts.com/2012/07/tutorial-making-heatmaps-using-qgis-and.html>

Frequency Graphs

Because data in a spreadsheet doesn't talk, distributing data into graphs is the best way to start data analysis. Earthquake data typically includes information about geographic coordinates (Lat, Long, Depth), time coordinates (Date and Time) and intensity (Mag). Frequency graphs allow investigating the distribution of data over their magnitude. For instance we may want to know how magnitudes are distributed over a magnitude scale that goes from 0 to 9.5. We may also want to know whether the depth of earthquakes are homogeneously distributed over the depth range. Importantly for earthquake forecasting, we want to know whether the timing of earthquakes is random or not.

The graph on the right shows a histogram of the magnitude of Japan earthquakes from 2003 to 2013. The earthquakes are distributed over magnitude "bins" of size 0.1. The number of earthquakes in each bin has been divided by the total number of earthquakes over that period (40263). This leads to the relative frequency (sometimes called density).



This histogram shows that there are more earthquakes of magnitude 4.3 to 4.4 than any other magnitude. At first glance the distribution of magnitudes is symmetric about the maximum frequency with roughly as many magnitudes > 4.4 than magnitudes < 4.4 . However, these qualitative assessments can be properly investigated.

Range, Quartile, Median, Percentile, Means, Mode. These descriptors refer to the distribution of data in frequency graphs. The range is simply: (maximum magnitude - minimum magnitude). The first quartile returns the value that cut off the first 25% of the dataset (here 4.2, first green line). The second quartile is by definition the median (here 4.5, in blue). The 3rd quartile ... (4.8, second green line). The nth percentile (or quantile) is the value that cuts off the first n% of the dataset when it is sorted in ascending order. The mean is the averaged magnitude (here, 4.53, in red), whereas the mode is the peak frequency (here 4.3-4.4).

R to the Rescue

Lets load some data in R. Here 427253 earthquake data from New Zealand (earthquake > 1 since 1913).

```
qk <- read.csv("/Path_to_Data/NZ-1913-2013.csv", header=T)
```

`summary(qk)` returns the Min, Max, Quartile, Median and Mean (average) of your dataset. While the command `range(qk)` returns the range.

Histogram of magnitude in a few easy steps

```
# This creates magnitude bin from 0 to 9 with size 0.1.
```

```
MagBin <- seq(0, 9, 0.1)
```

```
#To produce the histogram on this page:
```

```
colfunc <- colorRampPalette(c("yellow", "red"))
```

```
colfunc(length(MagBin))
```

```
hist(qk$MAG, breaks=MagBin, freq=FALSE, col=colfunc(length(MagBin)))
```

```
#To add a normal distribution curve (Gaussian model) using
```

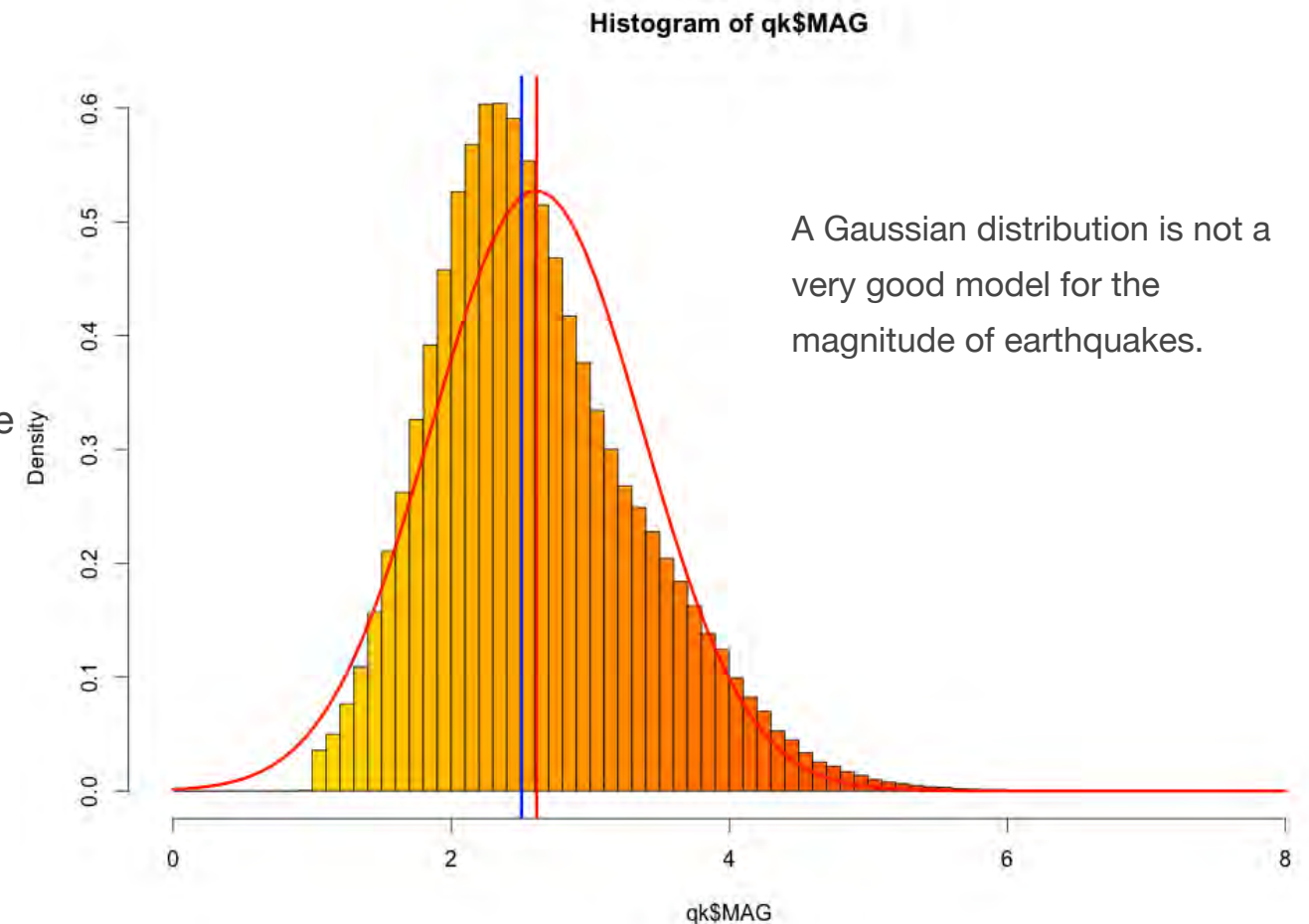
```
curve(dnorm(x, mean(qk$MAG, na.rm=T), sd(qk$MAG, na.rm=T)),  
add=TRUE, col="red", lwd=3)
```

```
#These lines add the median, mean, quantiles to the graph
```

```
abline(v=median(qk$MAG, na.rm=T), lwd=3, col="blue")
```

```
abline(v=mean(qk$MAG, na.rm=T), lwd=3, col="red")
```

```
abline(v=quantile(subqkJP[, 'Mag'], c(.25, .75), na.rm=T),  
lwd=3, col="green")
```



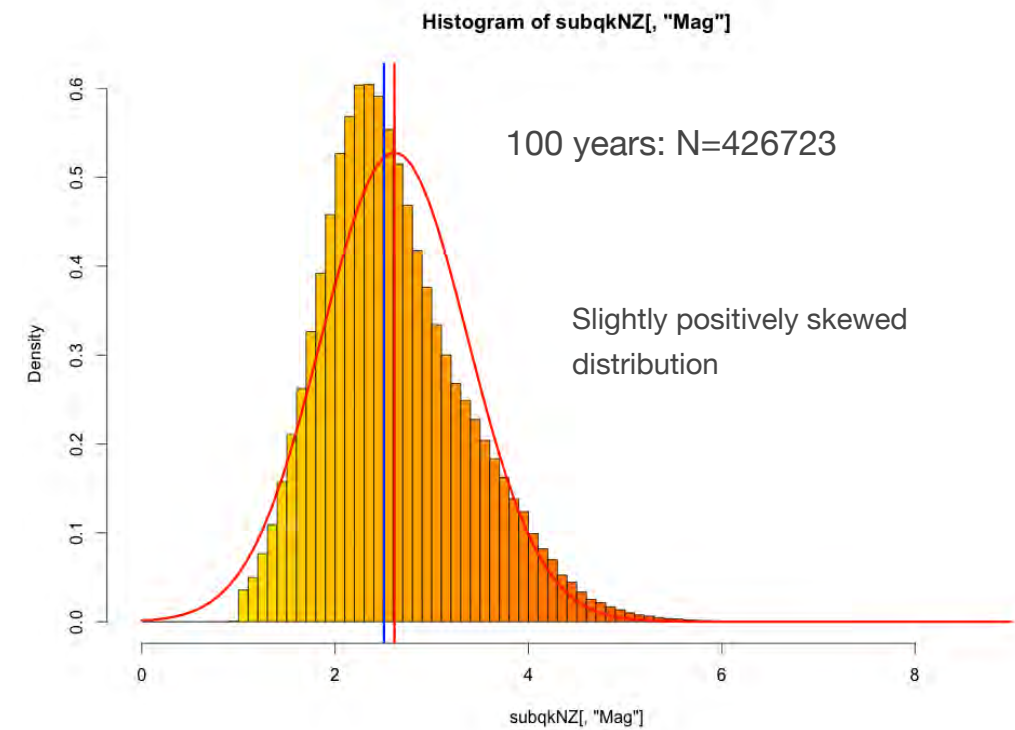
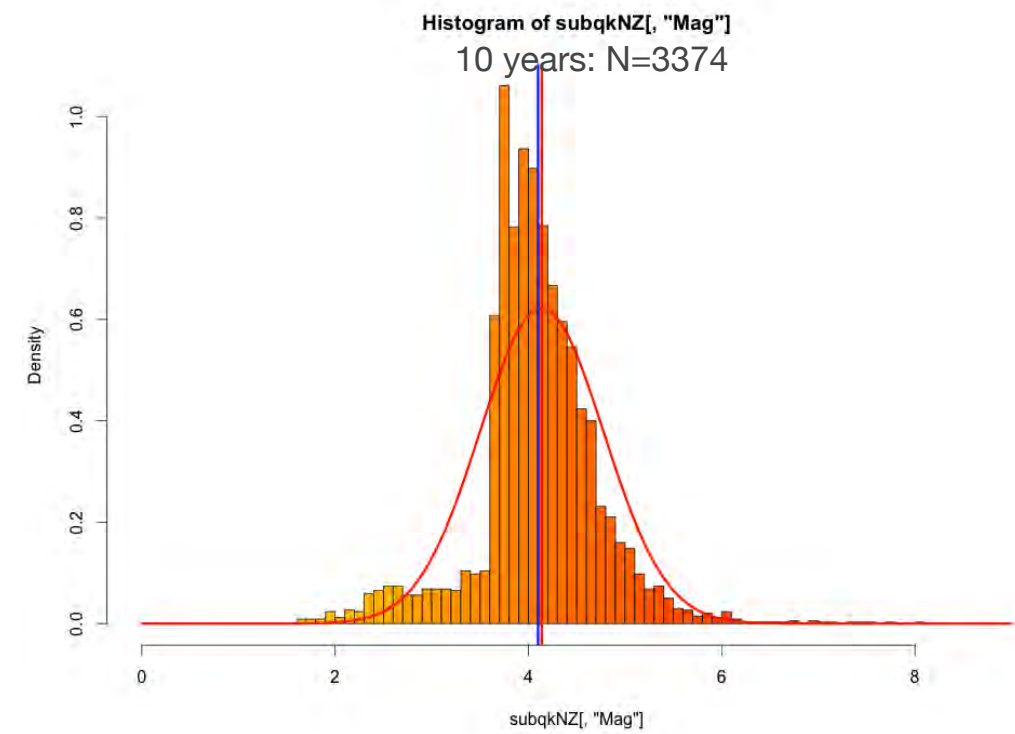
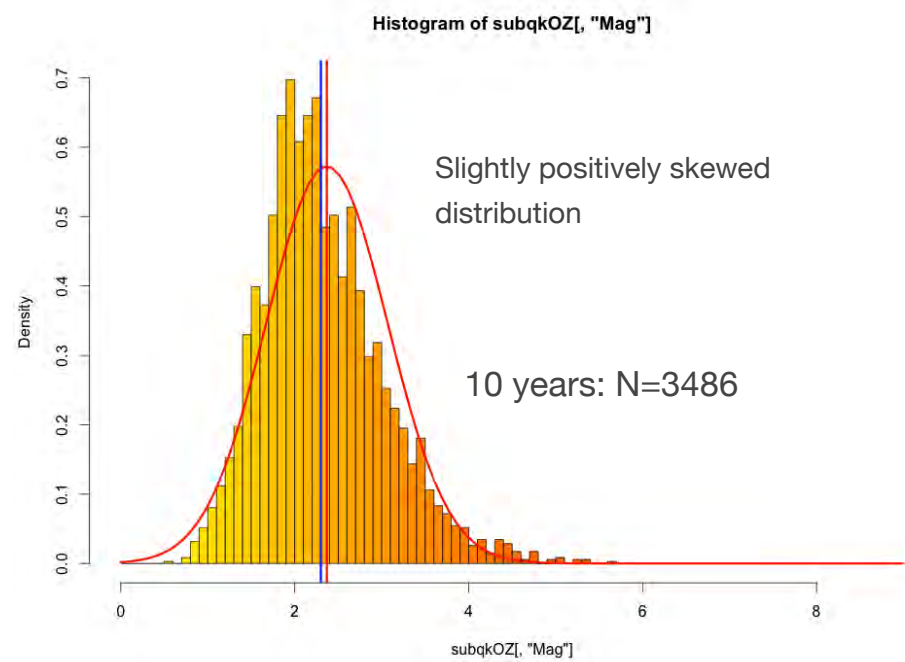
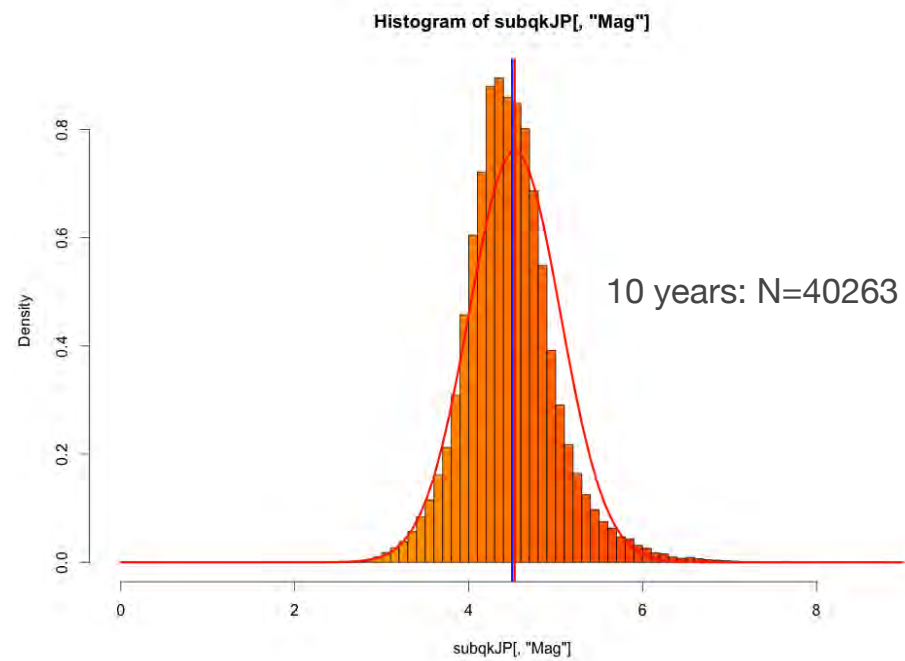
NB: The Gaussian curve (red curve on the graph above) is derived from the mean and the standard deviation. The standard deviation captures the spread of the data around the mean.

Histograms can be read in terms of probability of occurrence of a particular category of earthquake. The overall probability for earthquakes > 6 over a decade corresponds to the relative surface area covered by the histogram.

Some examples of earthquake frequency distributions

A decade of Earthquake in New Zealand

```
qkNZ <- read.csv("/Users/patricerey/Documents/Teaching/2013/2111-GIS/Data/
```



More on Frequency Graphs

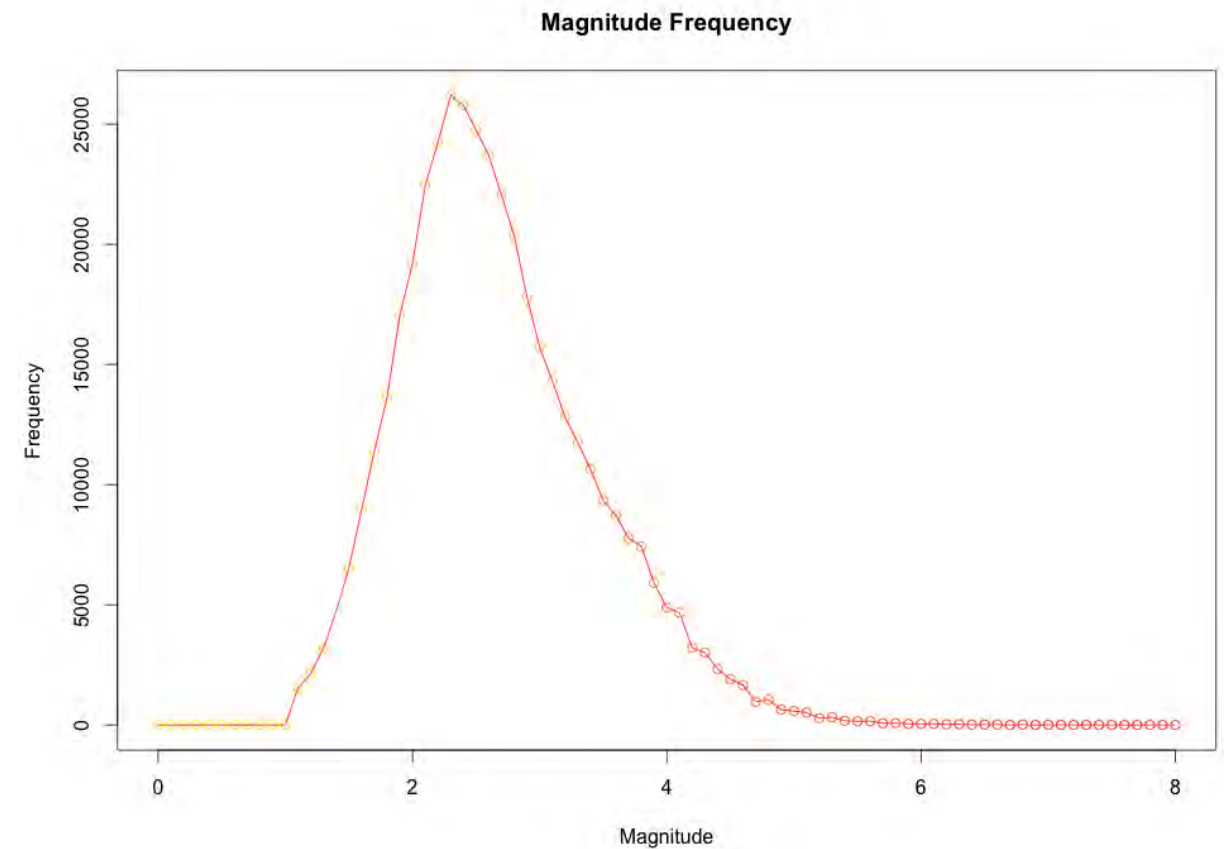
The frequency distribution of a variable describes its occurrence in a collection of non-overlapping categories. This is best represented in a histogram, but alternatives - such as the frequency polygon (right) - exist.

```
Magnitude = qk$MAG
```

```
Magnitude.cut = cut(Magnitude, MagBin, right=FALSE)
```

```
Magnitude.freq = table(Magnitude.cut)
```

```
plot(MagBin, c(0, Magnitude.freq), col=colfunc(length(MagBin)),  
main="Magnitude Frequency", xlab="Magnitude", ylab="Frequency")  
lines(MagBin, c(0, Magnitude.freq), col="red")
```



Cumulative Frequency Graph

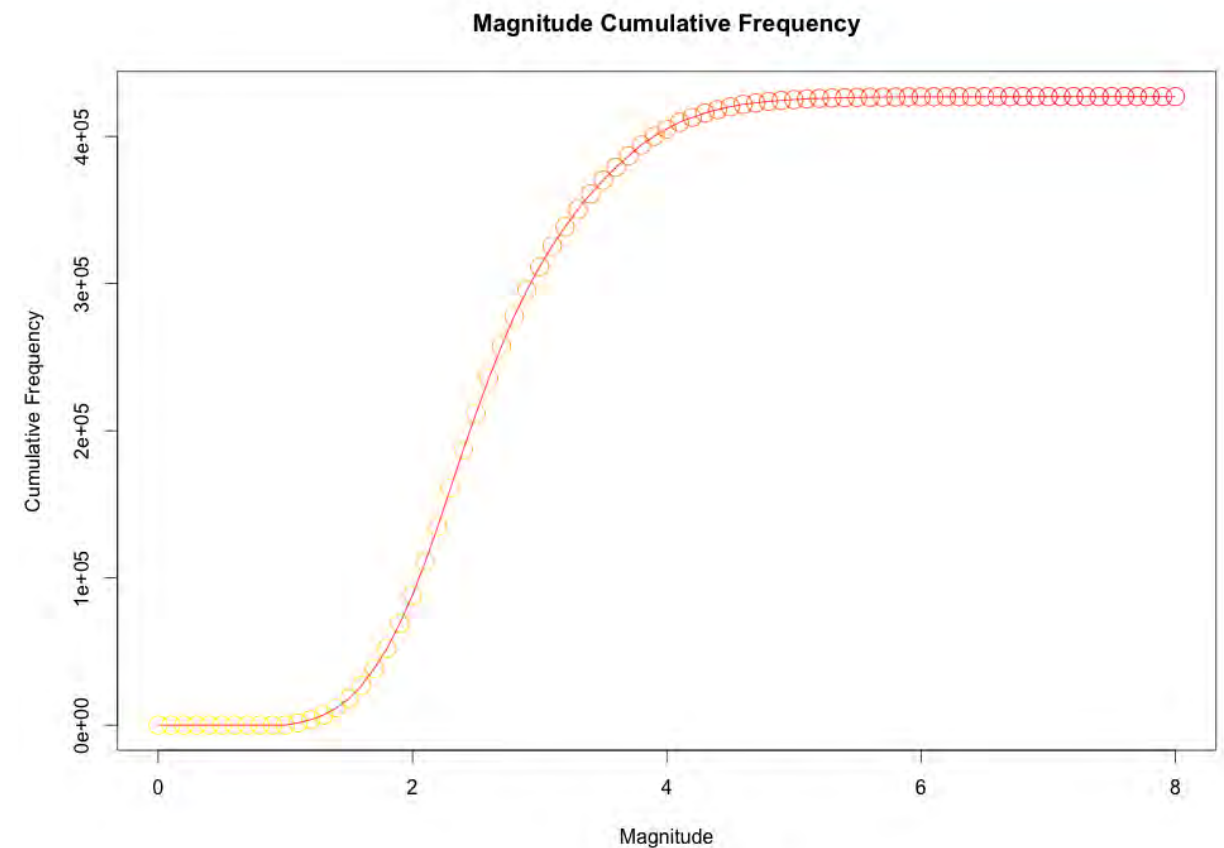
It determines the proportion of data lower or higher than a given threshold.

```
cumfreq0 = c(0, cumsum(Magnitude.freq))
```

```
cumfreq0
```

```
plot(MagBin, cumfreq0, col=colfunc(length(MagBin)), cex = 2,  
main="Magnitude Cumulative Frequency", xlab="Magnitude",  
ylab="Cumulative Frequency")
```

```
lines(MagBin, cumfreq0, col="red")
```



nb: Cumulative Relative Frequency = Relative Frequency / Sample Size

Variance, covariance and standard deviation

The variance describe how the dataset of size n is dispersed around the mean.

$$\sigma^2 = \frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^2$$

```
var(qk$MAG, na.rm=T) #return 0.572
```

The standard deviation is the square root of the variance. It is the average distance to the mean:

```
sd(qk$MAG, na.rm=T) #return 0.756
```

The covariance of two variables x and y (for instance magnitude and depth) in a dataset measures how the two are linearly related. The absolute value is not easy to interpret, but the sign tells whether they are proportional or inversely proportional.

$$S = \frac{1}{n} \sum_{i=1}^n (x_i - \bar{x}) \cdot (y_i - \bar{y})$$

The correlation coefficient of two variables in a data sample is their covariance divided by the product of their individual standard deviations (e.i = normalized covariance).

$$r = \frac{S}{\sigma_x \cdot \sigma_y}$$

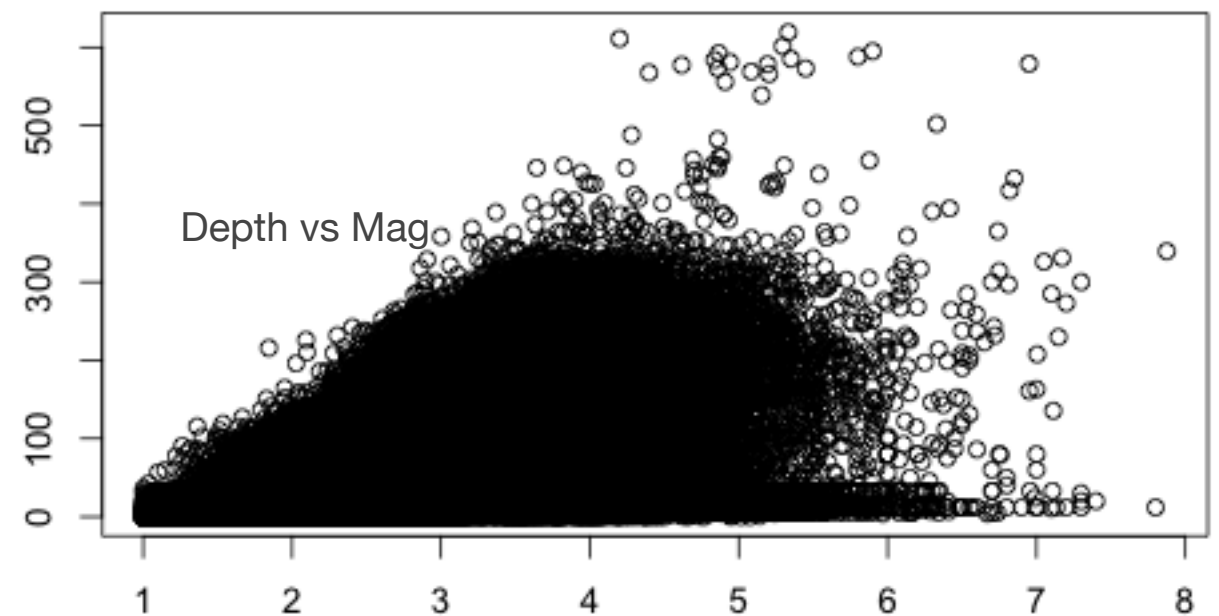
```
quake <- read.csv("/Path_to_data/NZ-1913-2013.csv", header=T)
```

```
#Lets take a subset of our dataset (here MAG and DEPTH)
subqak2<-cbind(Mag=quake$MAG, Depth=quake$DEPTH)
```

```
#Lets remove all rows containing NA values
subqak3<-na.omit(subqak2)
```

```
#Lets visually assess the correlation between Mag and Depth
pairs(subqak3)
```

```
#Lets calculate the covariance and coefficient of correlation:
cov(subqak3) # this returns a covariance of 18.02
cor(subqak3) # this returns a correlation of 0.44 (Pearson's
coefficient of corrleation).
```



More on coefficient of correlation

In a dataset involving several variables, the command `pairs(qak)` creates a matrix of scatter plots in which each column of the dataset is plotted against each others. This is a convenient way to quickly assess possible correlations.

First build a sub-dataset by combining a few columns:

```
subqak2<-cbind(qak$Long, qak$Lat, qak$Mag, qak$Depth, qak$Mb)
```

Then: `pairs(subqak2)` to produce the matrix on the right.

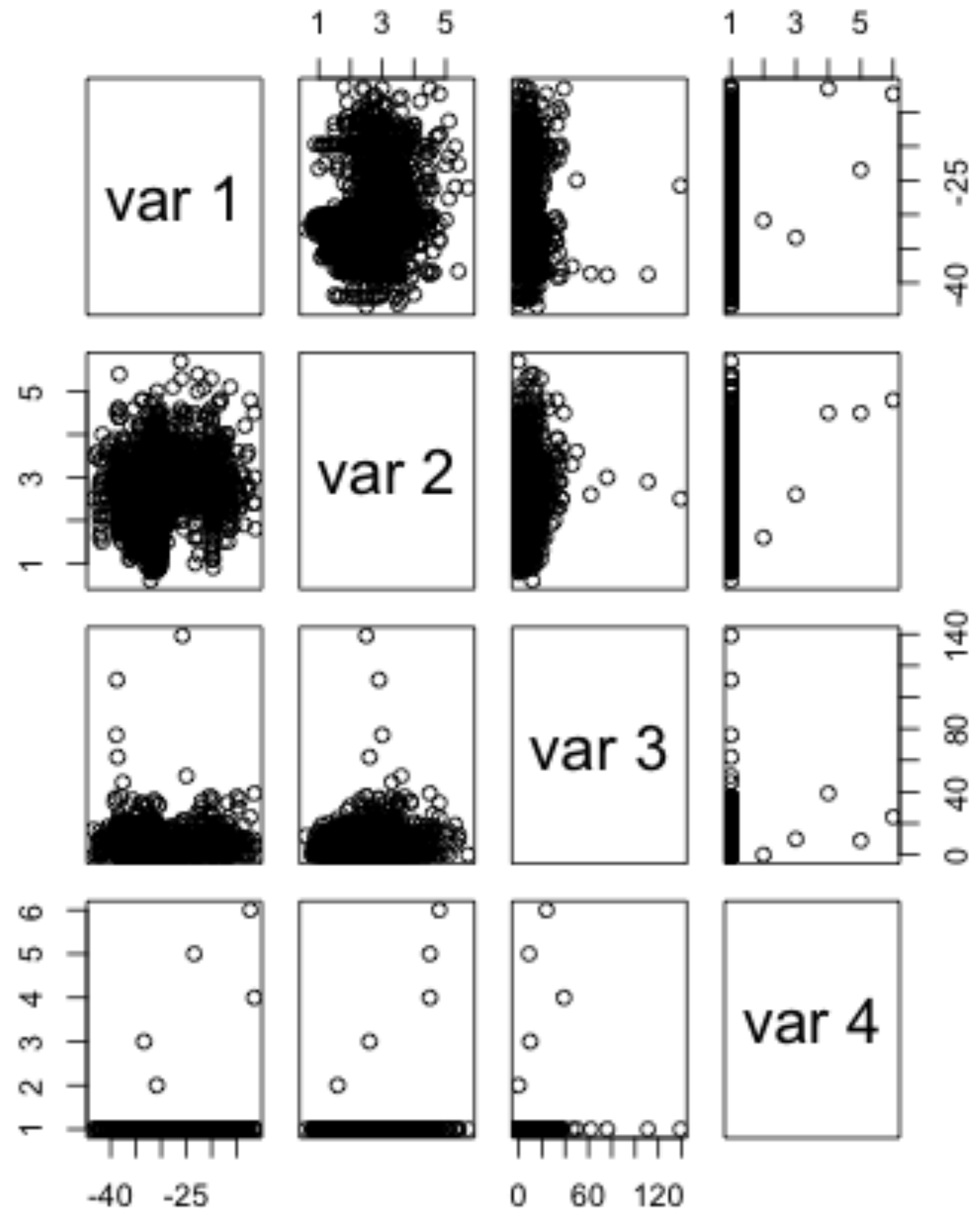
The Pearson's coefficient of correlation for each plot of the matrix can be calculated with the command

```
cor(subqak2)
```

Other coefficients of correlation exist:

```
cor(qak, method="kendall")
```

```
cor(qak, method="spearman")
```



Use of frequency graphs to compare datasets: Student's t-test:

Problem: Is the seismicity in Japan from 2000 to 2010 anomalous with respect to that of the past century? Both datasets are normally distributed, they have unequal size and similar variance (spread). One way to assess this is by comparing the difference of their means relative to their combined standard deviations (means difference relative to spread). This is the parameter t . The smaller this parameter the larger your confidence that the decadal seismicity is not anomalous. This is called the Null hypothesis, and t measures the deviation from the Null hypothesis. This method is grounded in probability theory for assessing whether the dataset represents one of many ($n!$) possible arrangements of a subsample from a much larger population. By definition, the Null Hypothesis is valid when $t=0$. The exact mathematic expression of t changes to account for equal or unequal sample sizes (m and n) and pooled variances S of both populations. For unequal size, same variance:

$$t = \frac{x_1 - x_2}{S\sqrt{\frac{1}{n} + \frac{1}{m}}} \quad \text{with} \quad S = \sqrt{\frac{(m-1)\sigma_x^2 + (n-1)\sigma_y^2}{m+n-2}} \quad \text{and} \quad \sigma_x^2 = \frac{\sum_{i=1}^m (x_i - \bar{x})^2}{m}$$

Should the Null hypothesis be correct, then we expect t to be zero, but this is statistically unlikely, as the parameter t is sensitive to the size of the dataset. The smaller the dataset, the smaller the confidence that parameter t can be used to assess the validity of the Null Hypothesis. So the real question is: What is the probability α that one will wrongly reject the Null hypothesis? Another way to put it: To which level of confidence ($1 - \alpha$) can t allow one to validate the Null Hypothesis.

Assuming a normal standard distribution, the t -table gives the maximum amount of mean difference one can expect for various sample sizes. From this, one can assess, typically with 95% or 99% confidence, whether the Null Hypothesis can be rejected. NB the level of confidence is the criterion used for rejecting the Null Hypothesis.

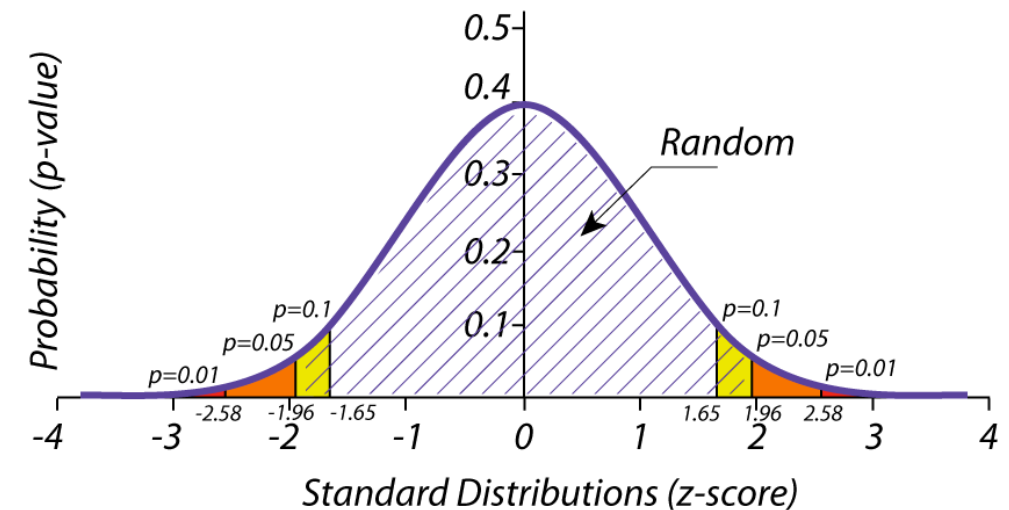
For legal reasons, Gosset published his work 1908 under the pseudonym 'Student' hence the name Student's t-test.

<http://www.r-bloggers.com/two-sample-students-t-test-1/> and <http://www.r-bloggers.com/two-sample-students-t-test-2/>

'Student' t-Test determines the probability that a population B is a subset of a larger population A . The standard 'Student's t-test assumes both populations are normally distributed (i.e symmetric cluster around a mean). In this case, the surface area limited by i/ the distribution curve and ii/ a category interval can easily be calculated and put into a table. Nevertheless, the t-test can be adapted to large samples of non-normal (skewed) distributions, in which case the mathematics slightly differ.

Student's Test workflow:

1/ Calculate the parameter t i.e. deviation from the mean, in standard deviation units. Statisticians call t the z-score. Assuming that the Null Hypothesis - B is a subset of A - is correct, what is the probability for a z-score larger by chance alone? (i.e chance to incorrectly invalidate the Null Hypothesis). To calculate the probability for a larger z-score than the one observed under the assumption of the Null Hypothesis, follow these two steps:



2/ Using the t -Table (this table is also called critical value for the z-test, or z-score chart), determine the surface area p under the two tails beyond the calculated z and $-z$ (i.e. colored areas in the graph). This represents the probability for a z-score larger than the one observed assuming the Null Hypothesis is correct (e.g. yellow 10%, orange 5%, red 1% chance B differs from A assuming the Null hypothesis is correct, with z-score of ± 1.65 , ± 1.96 , ± 2.56 respectively). A 5% level of confidence says that if the Null Hypothesis is correct we would have nevertheless 5% chance to get a z-score above the threshold.

3/ Compare this probability p to the chosen level of confidence α (1% more conservative, 5% less conservative). If $p \leq \alpha \Rightarrow$ The Null Hypothesis is rejected. Example: the z-score value when using a 95% confidence level is $-/+1.96$ standard deviations. The p-value associated with a 95% confidence level is 0.05. If $-1.96 \leq z\text{-score} \leq +1.96$, then its p-value is larger than 0.05, and the null hypothesis cannot be rejected; the populations are similar within expected statistical differences. If the z-score falls outside that range, the difference between the two populations is too large to be due to statistical error and the p-value will be small to reflect this. In this case, it is possible to reject the Null hypothesis. The next step is to investigate the origin of the difference.

Spatial statistics aims at characterizing gridded and field data distribution in a geographic space by documenting: clusters, outliers, spatial trends and trends over time, patterns, and covariate/multivariate relationships. The objective of spatial statistics is to gain insights into processes, and make predictions in relation to risk assessment.

- Spatial interpolation: Density analysis, spatial anisotropy, stochastic/deterministic pattern via “grid to field” interpolation aiming at estimating values of variables where no data is available. Interpolation methods include *inverse distance weighing* and the more sophisticated *Kriging*.
- Spatial autocorrelation: Measure the dependency between variables in a geographic space (Global Morans I, Geary’s C, Getis’s G, standard deviational ellipse). This requires measuring distance between neighbors, length of shared borders, shared directional anisotropy.
- Spatial clustering: Getis Ord Gi* Hotspot analysis. Determines statistically significant clusters (hot spot, and cold spot) using Student t-test to test the Null hypothesis: *There is no high/low cluster*.
- Spatial interaction: Estimate the degree of interaction/dependency between variables in a geographic space, with the idea that a variable depends on the value of its geographically, or topologically close neighbors.

Spatstat: Written by CSIRO Adrian Baddeley and Rolf Turner, spatstat is an R package dedicated to the analysis of spatial data. One of the largest R packages, spatstat includes over 1000 functions.

Point pattern, covariates, marks and multivariate.

Covariates are data potentially available at every cell of your GIS grid. Geographic coordinates (Lat-Long) are covariates. Topographic attributes, such as elevation and slope, and physical properties such as seismic velocities of surface waves are also a covariate. Covariate may involve the interpolation of gridded data known at a few sampling locations.

Covariate patterns can derive from processing of point patterns. For instance a map of geological faults is a covariate pattern. The distance of every location on a 2D domain to the nearest fault is also a covariate pattern.

A mark variable is a data attached to a point. For instance the magnitude of an earthquake. A mark is **multivariate** when it involves many data. For instance an earthquake could be marked by its magnitude, intensity and timing.

Loading Tsunamigenic events in the Pacific region

Goto: <http://tsun.sccc.ru/htdbpac/>

Click on *continue* (stay clear from MSIE5).

Choose *Event data*, on the next page choose *Magnitude* as selection criteria, and select magnitudes from 0 to 9.9 (default values), click *OK* and on the right panel click on *Search* to get all tsunamigenic events (142) available in the database (i.e. 65S to 65N and 80E to 50W).

Successively copy and paste the 5 pages of data (use navigation arrows from the top right panel) into your favorite text editor (for me TextMate). The headers are explained by clicking on *Legend* on the top left panel (\$Int Intensity; \$C Cause: T tectonic, V volcanic, L landslide, M meteorological, S seiches, E explosion, I impact, U unknow; \$V Validity: 4 definite, 3 probable, 2 questionable, 1 very doubtful, 0 false).

In your text editor replace all commas “,” by points “.”, and save this dataset using a descriptive name.

Time Series Analyses in R

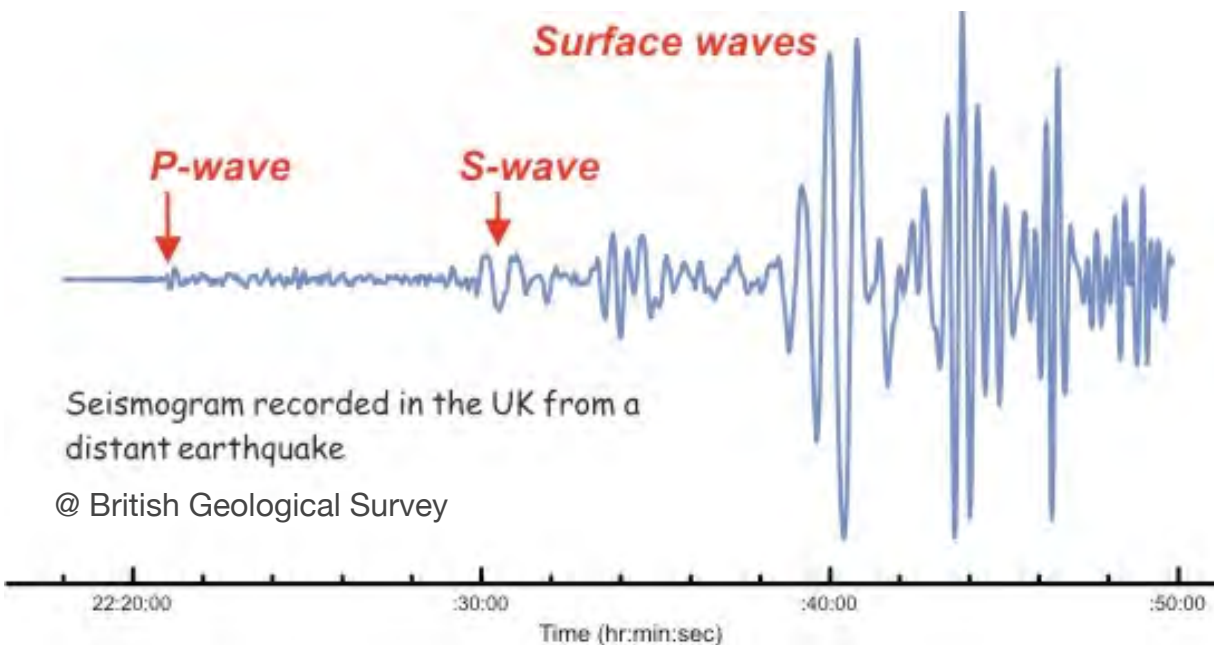
SUMMARY

1. What is it

<http://a-little-book-of-r-for-time-series.readthedocs.org/en/latest/>

<https://onlinecourses.science.psu.edu/stat510/?q=node/47>

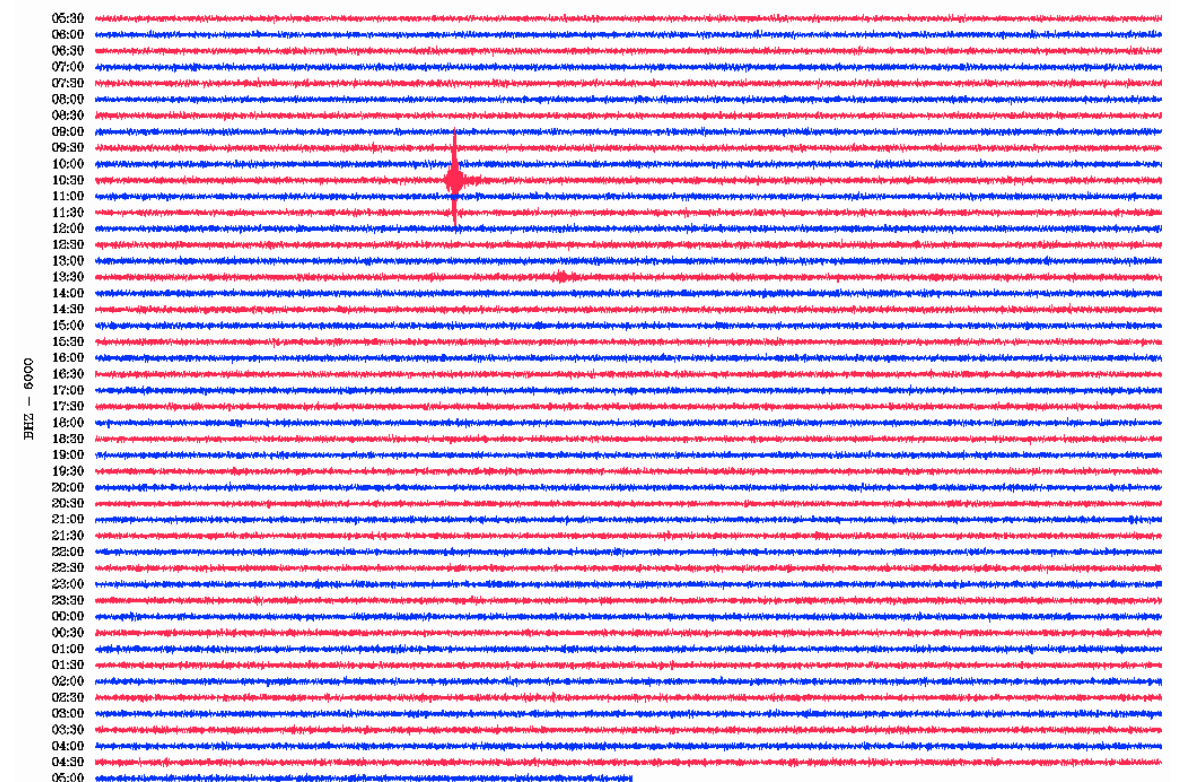
Time as a variable is important to Earthquake data. Seismograms record ground shaking over a time scale lasting few 10s of minutes, distance to the hypocenter is determined by the time difference between P and S waves arrivals, seismic station records earthquakes over a continuous time line. Earthquake recurrence is also key to earthquake forecasting. It is therefore natural that time series analysis is commonly performed to understand seismicity.



GEOFON station Walferdange, Luxembourg (49.68 N 6.15 E)

2013-03-16

Applied filter: WWSSN-SP



@ <http://www.ecgs.lu/geofon-live/index.html>

A time series is a list of data where the ordering is important; data at time t_n influences data at time $t_{m>n}$ with a decreasing level as m increases.

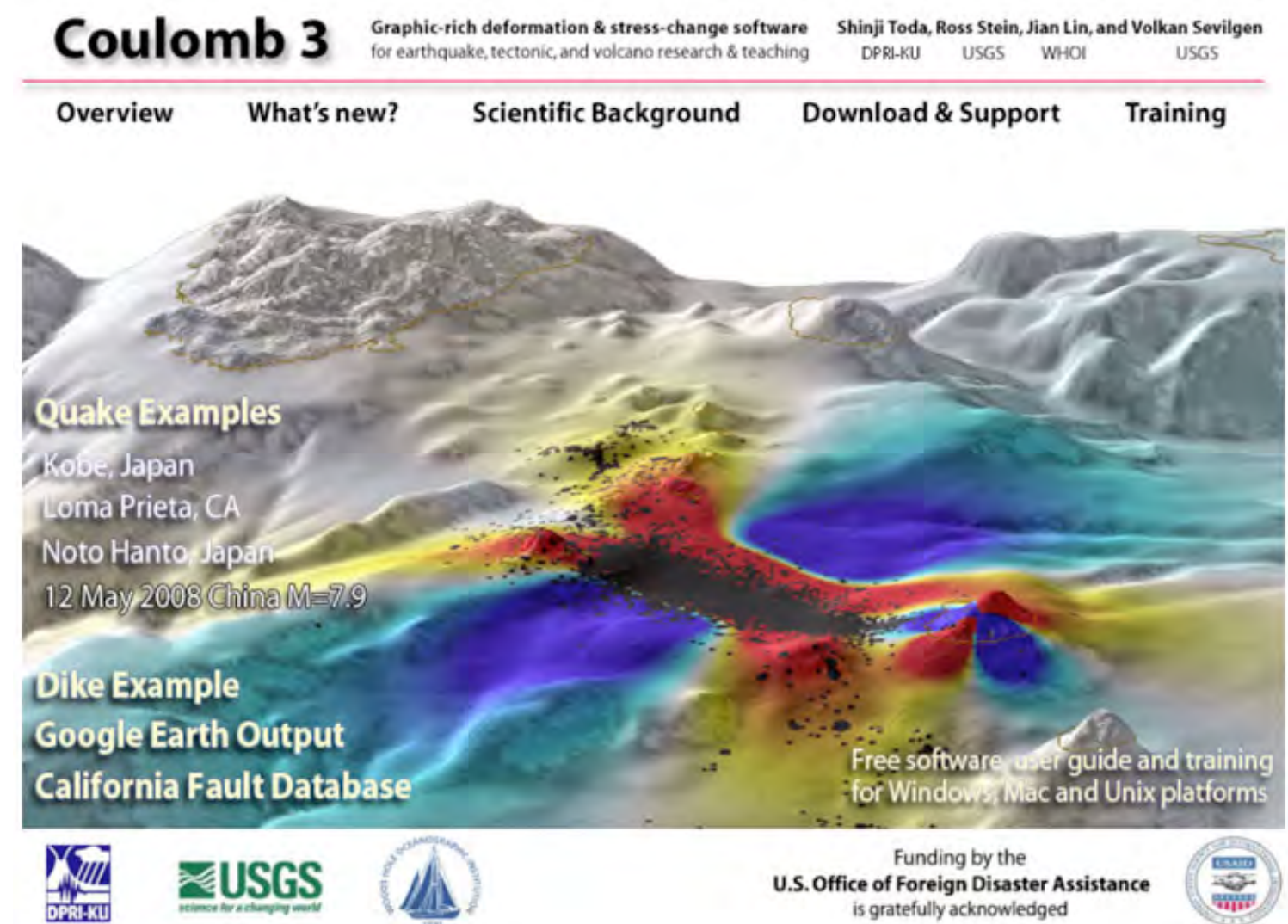
The objective of time series analysis is to produce a model to describe the pattern of the series with the aim at forecasting the future. In addition, time series analysis allows us to get a better understanding of the forces driving a time dependent process.

Earthquakes can be described as mechanical instabilities that release over a small amount of time, elastic energy accumulated continuously over a much longer time interval. Despite this simple physics, forecasting is very difficult because i/ earthquakes change the physical properties of faults, and ii/ because during earthquakes elastic energy is redistributed over a large region, bringing some faults closer or even past their rupture points (hence aftershocks), while releasing the stress acting on others. In addition, non-seismic slip also contributes to the transfer of elastic energy. Although large earthquakes are recurrent, it is difficult to predict accurately when and where they will occur.

The stress that drives earthquakes is called the Coulomb stress. Knowing the distribution of faults in a given region, one can calculate the change in

Coulomb Stress following a large earthquake. On the image above the Coulomb stress has increased in the red region, and has decreased in the blue regions. This can help to predict the region with an increased risk potential, and the region with a decreased risk potential.

Coulomb 3.3 is an open source application from USGS to calculate Coulomb stress changes.



Time series analyses aims at documenting trends, seasonalities or periodicities, constant variance, and identify any anomalous data or anomalous cluster of data (outliers). Let's have a look at seismicity around Christchurch (NZ) and in Japan.

Christchurch earthquake data can be retrieved from Geonet:
<http://magma.geonet.org.nz/resources/quakesearch/>

Region of Interest: Northern Latitude (-43.15), Southern Latitude (-43.90), Western Longitude (171.75), Eastern Longitude (173.35)

Verify the cvs file before running the R script on the right to show earthquakes of magnitude > 3.5 over an 18-month period (from Aug 2010 to Feb 2012).

```
#Example from: http://www.quantumforest.com/2012/01/plotting-earthquake-data/
```

```
setwd('/your_path_to_Data_Folder')
```

```
library(ggplot2)
```

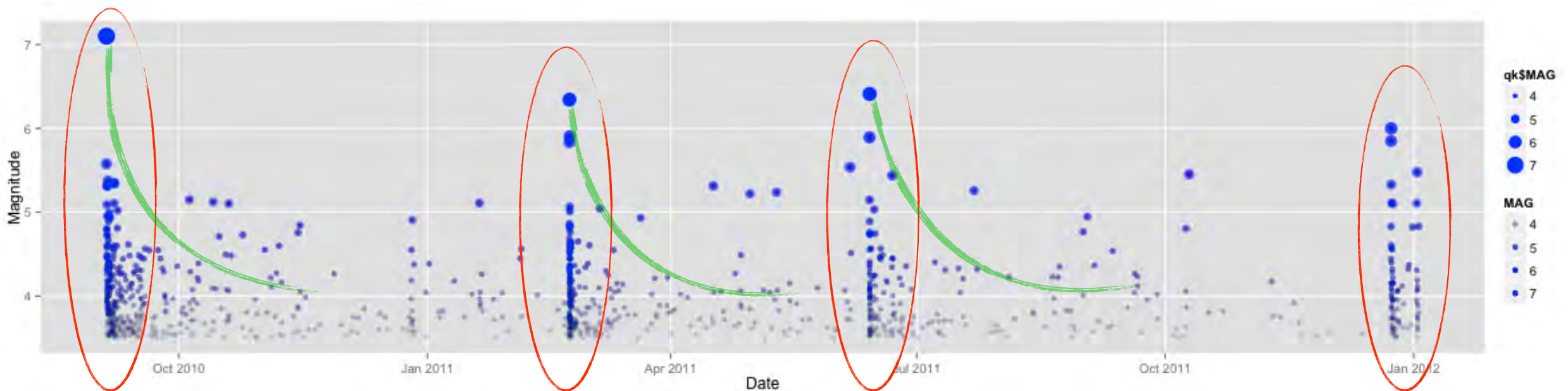
```
# Reading file and manipulating dates
```

```
qkb <- read.csv('earthquakes.csv', header=TRUE)
```

```
qk <- subset(qkb, qkb$MAG >= 3.5 & qkb$MAG < 9)
```

```
qk$DATEfmt = with(qk, paste(ORL YEAR '-', ORL MONTH '-', ORL DAY -
```

The time series reveals recurrent > 6 magnitude earthquakes, each belonging to an earthquake cluster. These clusters are followed by a swarm of earthquakes decreasing in number and magnitude. For readability, the size and transparency of the markers are functions of the magnitude.



Four decades of seismicity in Japan. The first graph shows earthquakes of magnitude > 3.5 , and the second all earthquakes > 5.5 .

If one focuses on the largest earthquakes, it seems that their magnitude has steadily increased over the past few decades. However, this trend is rather weak and could be fortuitous.

#Original example from: <http://www.quantumforest.com/2012/01/plotting-earthquake-data/>

```
setwd('/Path_to_data/Data/')
```

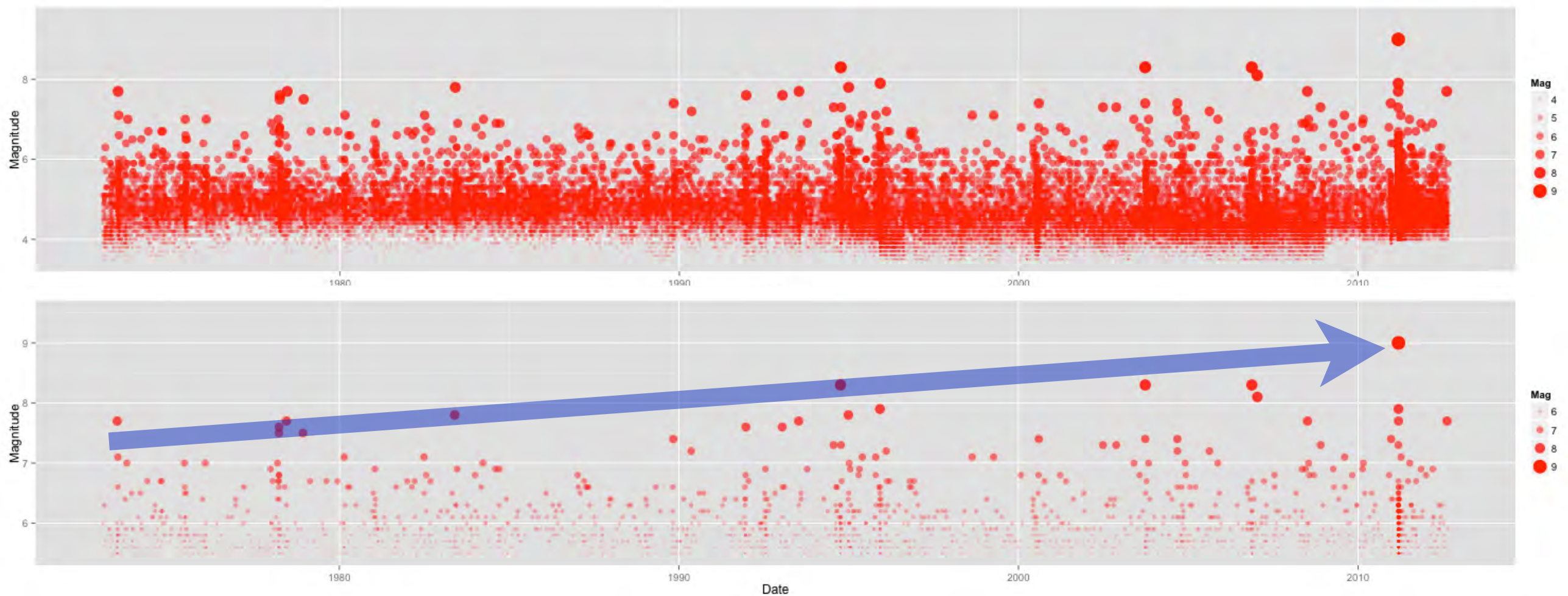
```
library(ggplot2)
```

```
# Reading file and manipulating dates
```

```
qkb <- read.csv('Japan_Qk_1973_2013.txt', header=TRUE)
```

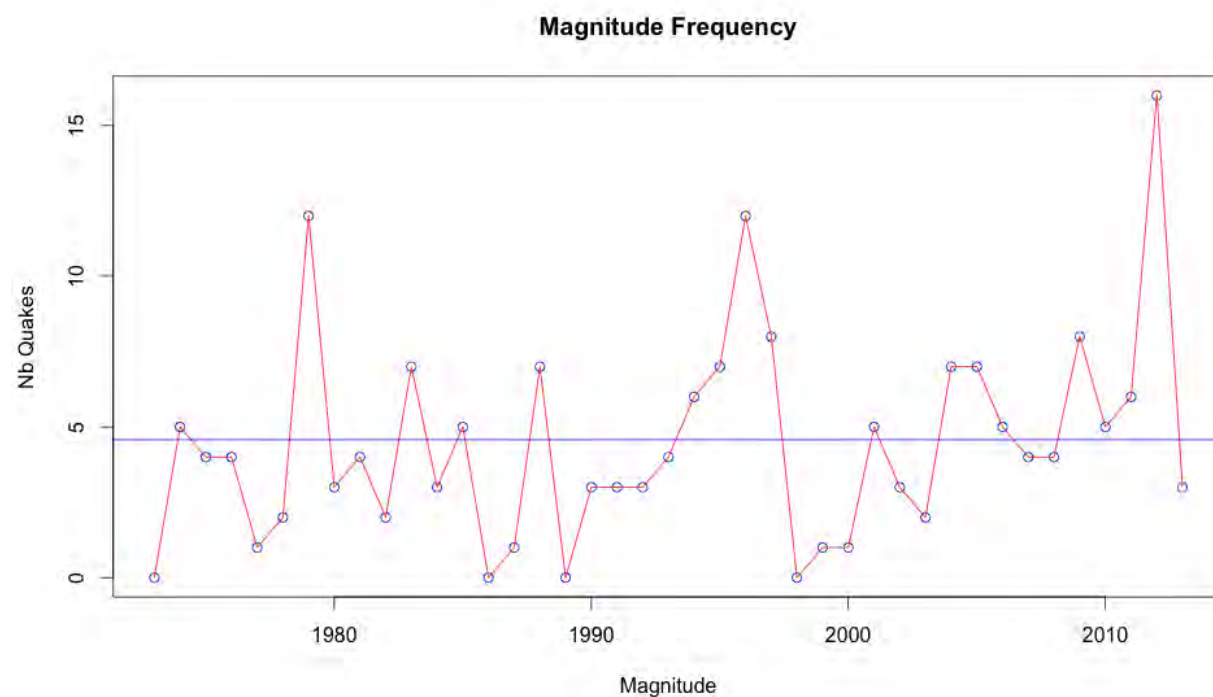
```
qk <- subset(qkb, qkb$Mag >= 3.5 & qkb$Mag < 9.5)
```

```
qk$DATEtxt = with(qk, paste(Year,'-',Month,'-', Day, sep=''))
```



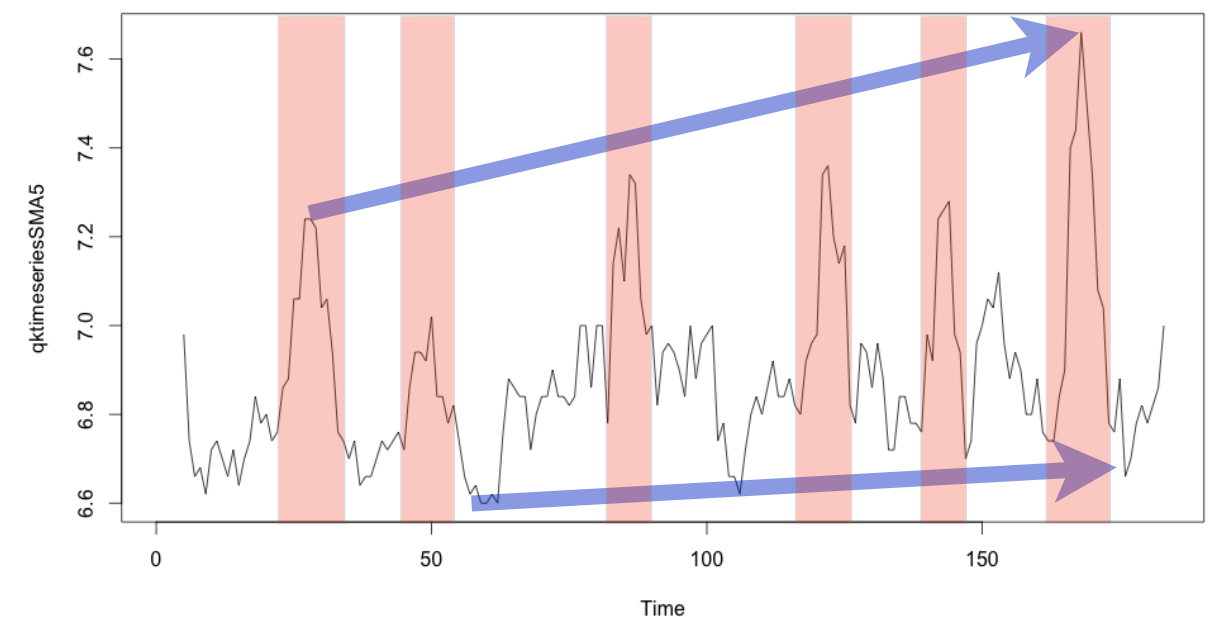
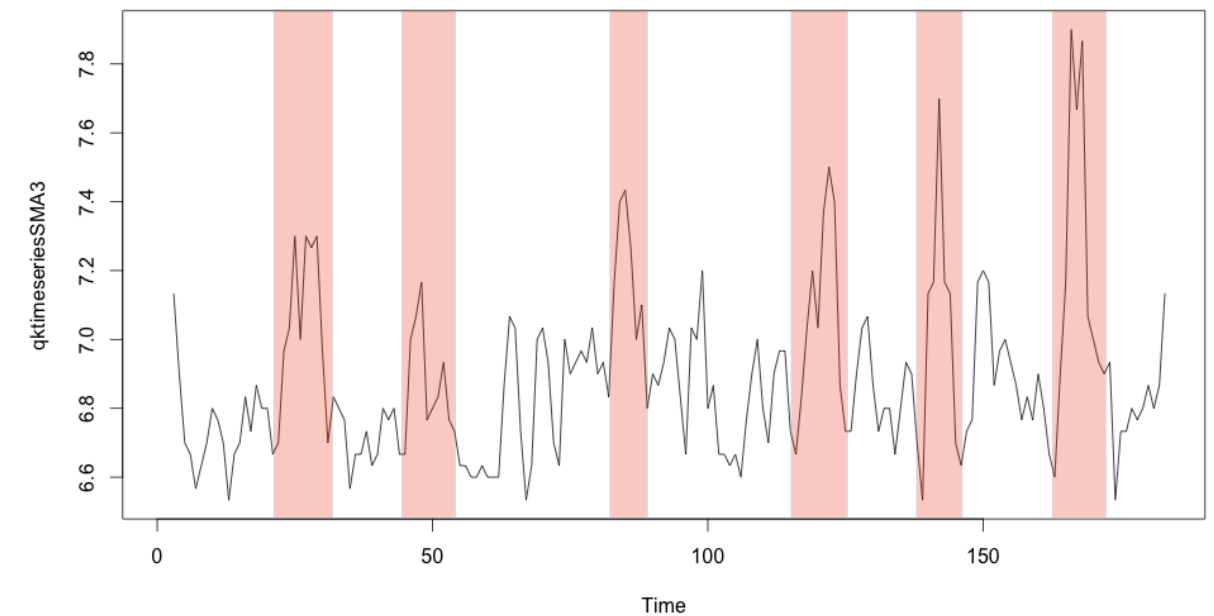
Here we look at the annual number of earthquakes in Japan with seismic magnitude over 6.5 since 1973. Over the past 40 years, Japan has had to deal with close to 5 annual earthquakes of magnitude 6.5 or higher.

```
# R script for the graph below.
setwd('/Users/patricerey/Documents/Teaching/2013/2111-GIS/Data/')
library(ggplot2)
# Reading file and manipulating dates
```



The graph on the top right shows the time series of the 183 earthquakes of magnitude > 6.5 shown above. A number of recurrent intensity picks stand out from a high-frequency near constant variance distributions. High-frequency variations can be attenuated through a “moving average” window of 5 years (bottom right). The slight upward trend in the intensity of the seismicity is also visible.

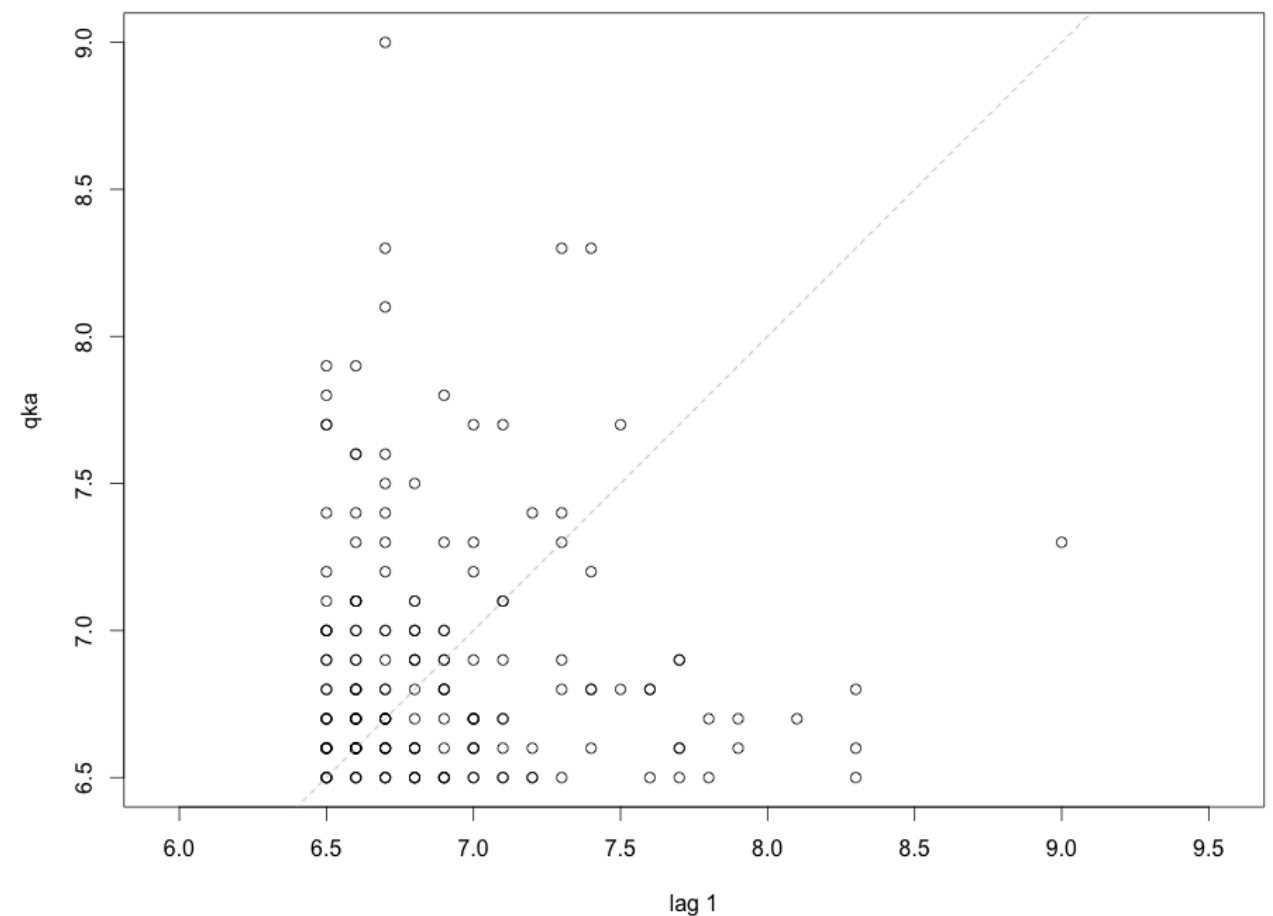
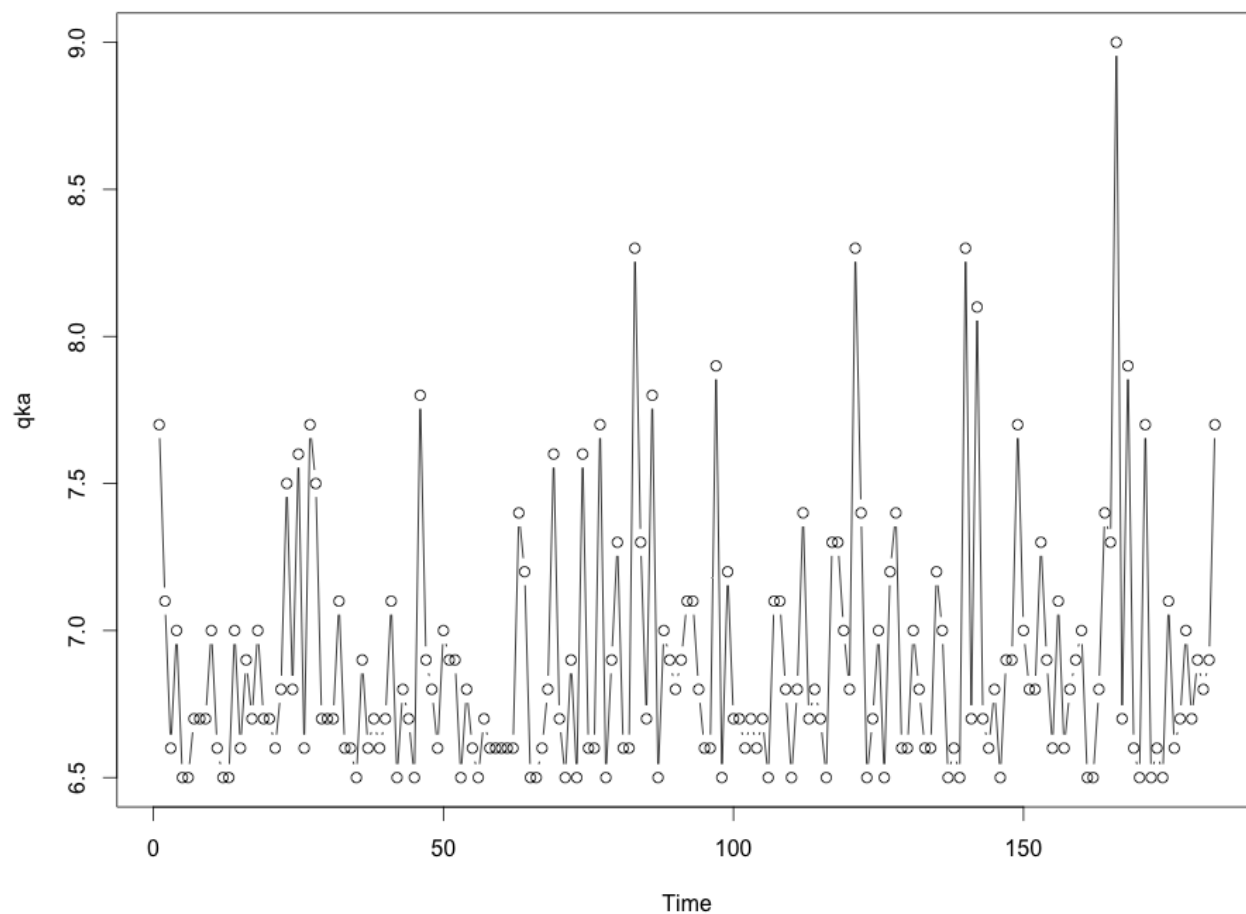
```
# R script for the two graphs below.
setwd('/Path_to_data/Data/')
library("TTR")
qkb <- read.csv('Japan_Qk_1973_2013.txt', header=TRUE)
```



The question arises whether the earthquake at rank i in the time series depends on the earthquakes at ranks $i-n$, where n indicates how many previous times is considered. This is the essence of autoregressive model of order n , in which the present can be explained by the recent past. This family of models are called ARIMA models (for Autoregressive Integrated Moving Average).

The graph below shows >6.5 earthquakes from 1973 to 2013 in Japan. On the right a graph in which earthquake at time t is plotted as a function of the earthquake at $t-1$ (lag=1).

```
setwd('/Users/patricerey/Documents/Teaching/2013/2111-
GIS/Data/')
library(ggplot2)
# Reading file and manipulating dates
qkb <- read.csv('Japan_Qk_1973_2013.txt', header=TRUE)
qk <- subset(qkb, qkb$Mag >= 6.5 & qkb$Mag < 9.5)
qka=ts(qk$Mag)
lag.plot(qka, 1)
```



Scientific Computing with iPython

Scientific computing and data analysis is the backbone of science and engineering, for which iPython provides a compelling environment.

Fernando Pérez began the *iPython* project in 2001 with the aim to design a collaborative, *interactive* scientific computing environment. While the Python language is at its core, iPython is designed in a language-agnostic way to facilitate interactive computing in any language.

Open source, multi-platform and extensible, iPython is a framework which provides a friendly Integrated Development Environment for scientific computing, powerful enough to allow for flexible parallel computing.

The logo for iPython, featuring the letters 'IP' in a large, serif font, followed by '[y]' in a smaller, blue, sans-serif font. The background of the slide shows a view of Earth from space with a bright sun or moon in the upper right corner.

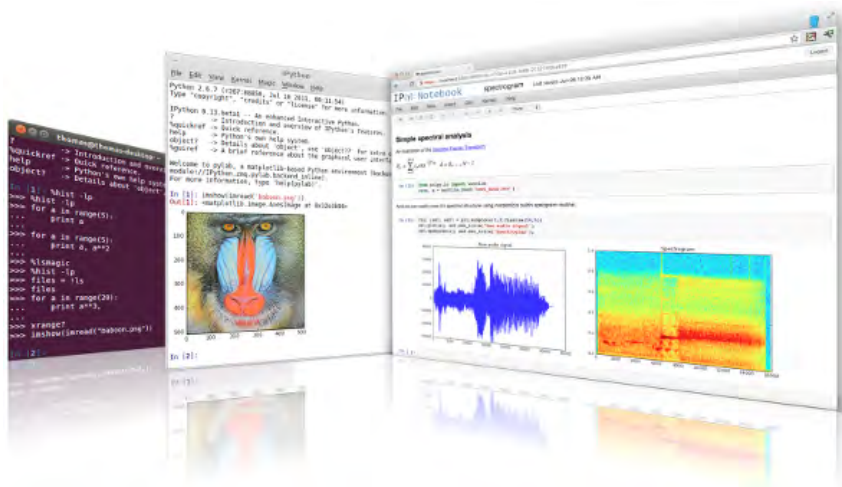
iPython can easily integrate python libraries such as Numpy, Matplotlib, SciPy, SymPy, SPy, as well as other language codes such as C, R, Julia, Octave, MatLab etc and scripting in Bash, Perl or Ruby.

iPython has a number of User Interface (Terminal-based e.g emacs, vim, pylab, QT console ...), including a web Notebook that runs on any web browser. Notebook is able to work with embedded formatted text, image and video in various format, webpage etc. Outputs can be produced in LaTeX, HTML, reST, svg, pdf, etc.

Installing iPython

IPYTHON

1. iPython home: <http://ipython.org/>
2. Introduction to iPython project by Fernando Pérez:
http://www.youtube.com/watch?feature=player_embedded&v=26wgEsg9Mcc



The image above shows from left to right iPython on Terminal, Qt-console and Notebook

Perhaps the easier way to install iPython (Mac, Windows or Linux), with all its components including dependencies and basic libraries for scientific computing and data analysis, is via the Anaconda installer.

Download the Anaconda installer <http://continuum.io/downloads>, open a terminal and navigate to the directory where the installer has been saved. From this directory, run the following command, replacing the <your_architecture> in the command with your version number.

```
bash Anaconda<your_architecture>.sh
```

In Windows double click on the installer application icon and follow the instruction.

Update iPython to current vs:

```
conda update conda  
conda update ipython
```

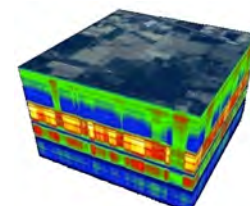
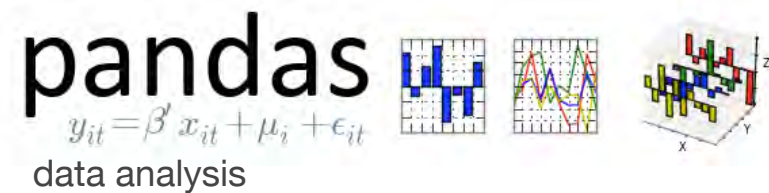
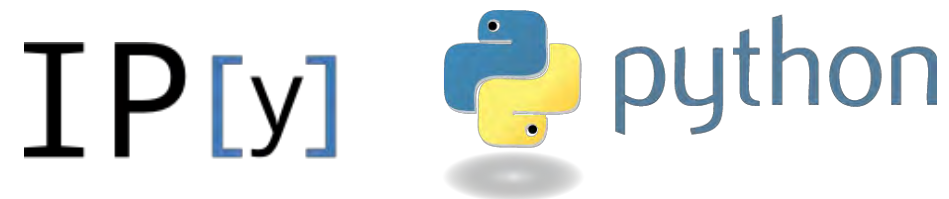
More info here: <http://docs.continuum.io/anaconda/install.html#windows-install>

Well we are ready to play with iPython. Remember to give credit to the iPython authors:

Fernando Pérez, Brian E. Granger, IPython: A System for Interactive Scientific Computing, Computing in Science and Engineering, vol. 9, no. 3, pp. 21-29, May/June 2007, doi:10.1109/MCSE.2007.53. URL: <http://ipython.org>

First step with iPython

Python-based scientific computing ecosystem ...



iPython is an easy entry door to the world of Python.

In what follows, we will use Notebook as our iPython User Interface. iPython Notebook is a web-based interactive computational environment in which one can combine code execution, formatted text, mathematics, plots and rich media into a single document. These Notebooks can be saved into HTML, LaTeX or PDF format and shared with the broader community using the iPython Notebook Viewer service which will render it as a static web page.

Let's start: In a Terminal execute the following commands:

`iPython --version`

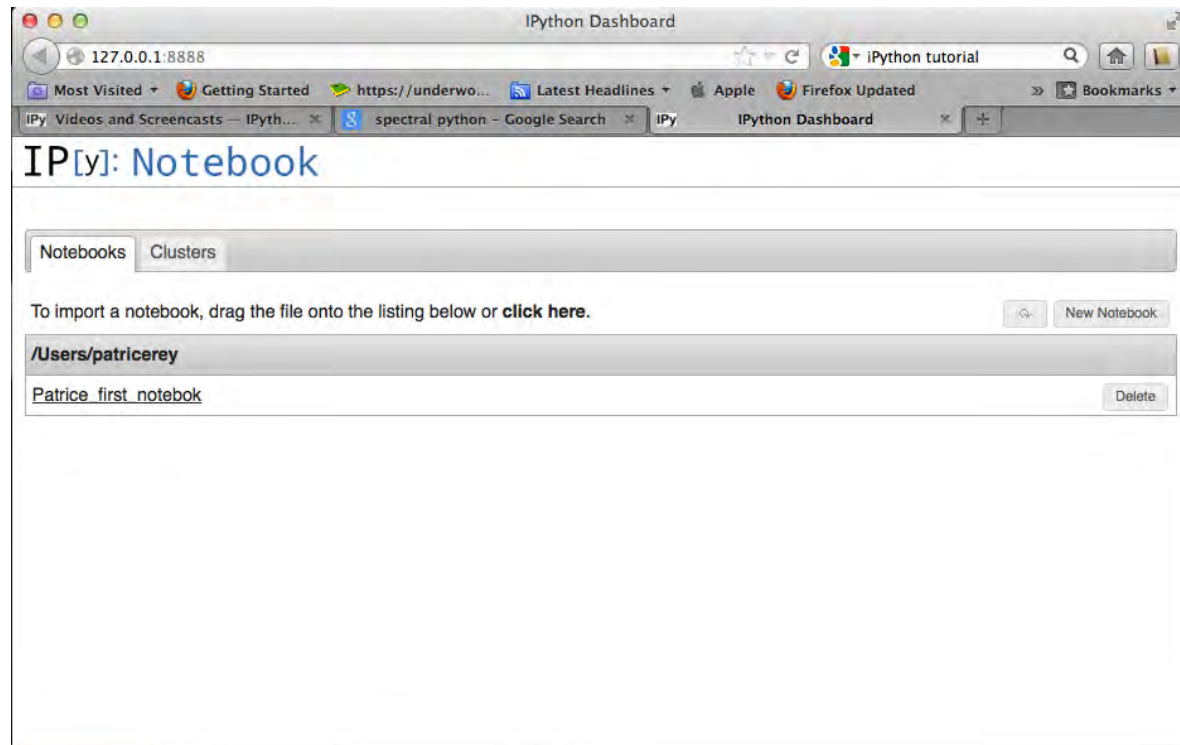
At this stage, if nothing happens then chances are that your iPython is not in your PATH. In your `.bash_profile` make sure that you have a line such as: **export PATH="/Users/jamesbond/anaconda/bin:\$PATH"**, then close and re-open a Terminal and retype the command `iPython --version`.

To open a Notebook execute the command:

`iPython notebook`

An easy, GIS oriented tutorial about Notebook by Richard Wareham (first 14 minutes some basic manipulations, then GIS application):

http://www.youtube.com/watch?v=r_fbS4t_Koc



You should open a new window in your Internet Browser (here Firefox). Click on *New Notebook* to create a new Notebook.

On the new Notebook window click on *Untitled0* to change its name. Then, explore the menu items (File, Edit, View, etc).

Shift-enter will execute command in the Notebook cells. These cells can be deleted, moved and edited at any time.

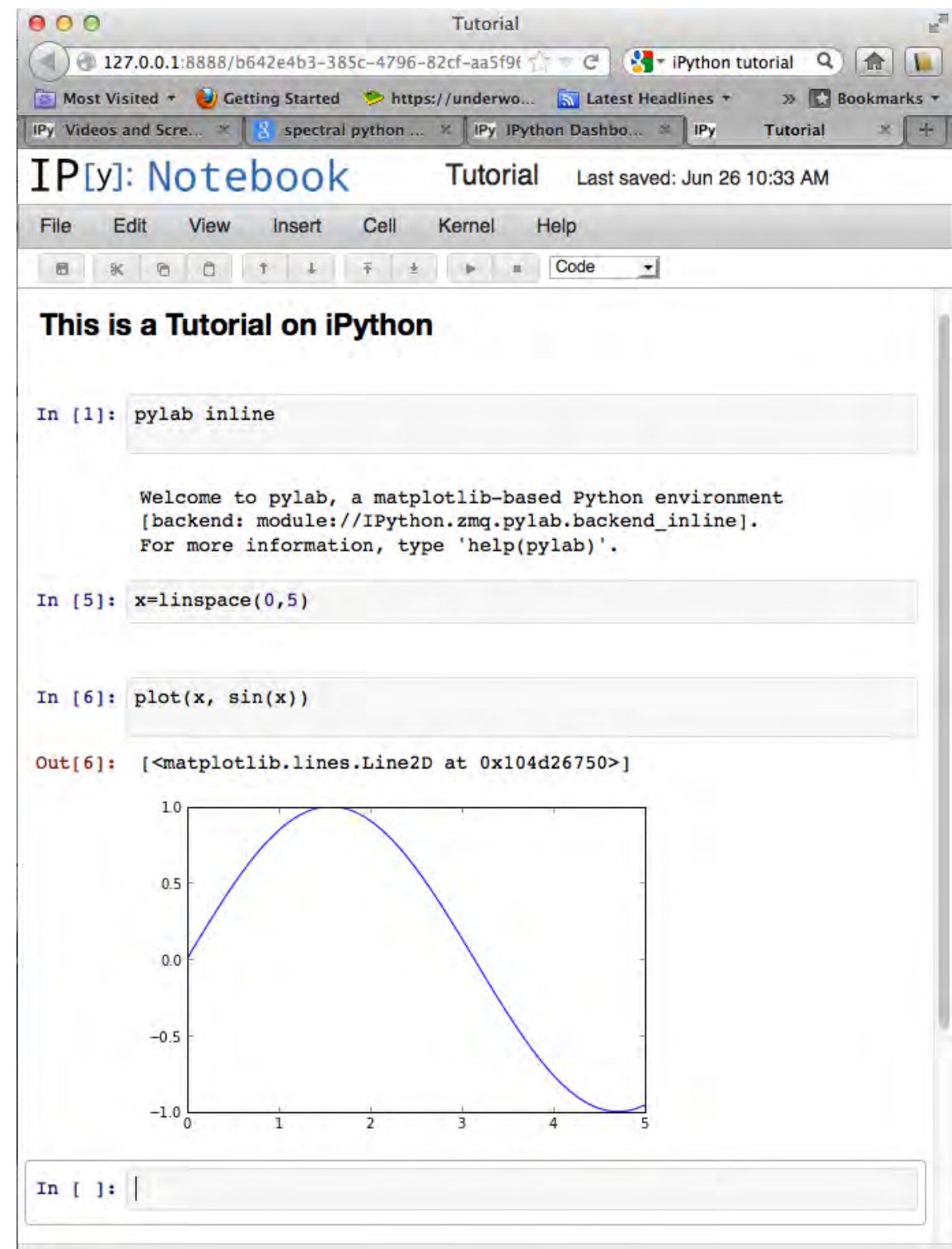
In pylab mode, plots can be displayed inline. To call the pylab mode enter in a Notebook cell: `pylab inline`

Then try these:

```
x=linspace(0,5)
```

```
plot(x, sin(x))
```

This should produce the result shown on the right.



Notebook can be saved. A notebook is a linear sequence of cells containing everything (text, code, plot, etc).

Need help? Integrated help is available at any time. Should you require some help with any command or an object just add a ? For instance try: plot?

A double question mark ?? will give more information. To get rid of the explanation click on the divider.

Entering %quickref will give info on python commands, and *int*? will give information on all objects containing the sequence *int*.

iPython -h (shows various customisation options)

An exclamation ! before a command means a system command (not a python command). Example !ls is equivalent to ls in a Terminal shell.

An overview of IPython (40 minutes) delivered by Fernando Perez. It combines a rapid overview of the IPython project with hands-on demos using the new HTML notebook interface.

http://www.youtube.com/watch?feature=player_embedded&v=26wgEsg9Mcc

A detailed tutorial (2:48h) presented by Fernando Perez, Brian Granger and Min Ragan-Kelley. (At 1.03.28 Brian Granger gives an intro on Notebook, the last speaker talks about configuring iPython environment).

http://www.youtube.com/watch?feature=player_embedded&v=bP8ydKBCZiY

Numpy: NumPy is the Python package for scientific computing. It contains among other things, 1/ a powerful N-dimensional array object, 2/ sophisticated functions, 3/ tools for integrating C/C++ and Fortran code, 4/ linear algebra, Fourier transform, etc.

SciPy: SciPy is a Python package that provides many user-friendly and efficient numerical routines such as routines for numerical integration and optimization.

SymPy: SymPy is a Python library for symbolic mathematics. It aims to become a full-featured computer algebra system (CAS).

Pandas: Pandas is a Python data analysis library providing high-performance, easy-to-use data structures and data analysis tools. It is based on data.frames.

matplotlib: matplotlib is a python 2D plotting library which produces publication quality figures in a variety of formats and interactive environments across platforms.

SPy (spectral python): Spy is package for processing hyperspectral image data.

All these libraries and more can be loaded into iPython to extend and align its capabilities to the users' needs. For a list of libraries

see <https://pypi.python.org/simple/>. These libraries can be installed from within iPython using: `!pip install name_of_libraries`

In a Notebook cell execute the following command.:

```
import numpy as np
import scipy as sp
import matplotlib.pyplot as pltPy
```

SPy is not part of the packages installed by default by iPython, nor is it available through the pypi.python.org website. Manually download SPy from <https://sourceforge.net/projects/spectralpython/files>, navigate to the directory then execute:

```
sudo python setup.py install
```

This will install SPy.

The standard means of opening and accessing a hyperspectral image file with SPy is via the `image` function, which returns an instance of a `SpyFile` object.

First Install wx

```
!pip install wx
```

If this doesn't work then try to include the full path:

```
!pip install https://pypi.python.org/packages/source/w/wx/
wx-1.0.0.tar.gz#md5=0f464e6f2f1e80adb8d1d42bb291ebf1
```

```
import wx
```

Manually download and install wxmPlot, via `sudo python setup.py install`

```
pylab inline
!ipython --pylab=WX
```

```
from spectral import *
img = open_image('pah_to_image.lan')
img.__class__
print img
w=view(img, [29, 19, 9])
save_rgb('rgb.jpg', img, [29, 19, 9])
```

or

```
arr = img.load()
arr.__class__
print arr.info()
arr.shape
```

nb:Since `spectral.ImageArray` uses 32-bit floating point values, the amount of memory consumed will be approximately $4 * \text{numRows} * \text{numCols} * \text{numBands}$ bytes.

Spectrum Plot: The image display windows provide a few interactive functions. If you create an image display with `view` and then double-click on a particular location in the window, a new window will be created with a 2D plot of the spectrum for the pixel that was clicked.

Note that the row/col of the double-clicked pixel is printed on the command prompt. Since there is no spectral band metadata in our sample image file, the spectral plot's axes are unlabeled and the pixel band values are plotted vs. band number, rather than wavelength. To have the data plotted vs. wavelength, we must first associate spectral band information with the image.

```
import spectral.io.aviris as aviris  
img.bands = aviris.read_aviris_bands('92AV3C.spc')
```

Now, close the image and spectral plot windows, call view again and click on a few locations in the image display. You will notice that the x-axis now shows the wavelengths associated with each band.

More at:

<http://spectralpython.sourceforge.net/graphics.html>

Chapter 6

Google Earth

GE is the most iconic App of the Internet era. It gives the ability to any user with a computer, a tablet or a smart phone to see the world from far above and zoom in down to an astonishing level of details.

GE uses a range of DEM, including SRTM data, over which a range of textures can be draped including aerial photographs, topographic contours maps as well as users own textures.

GE has evolved to be a Virtual Globe on which geo-referenced data can be attached for global distribution using kml (keyhole markup language). Many of these database are updated in real time to give up-to-date distributions of earthquakes, cyclonic low pressure systems, tsunami warning, bushfire progression, extend of flooding, etc.

In this Chapter we learn to geo-reference data using kml.

© 2012 Cnes/Spot Image
© 2012 Whereis® Sensis Pty Ltd
Image © 2012 GeoEye



1194 m

Geo-Referencing in GE

GOOGLE EARTH

1. Sources

<http://www.google.com/earth/index.html>

2. KML tutorial

<https://developers.google.com/kml/documentation/>

A little trick:

On a .kmz file replace the extension .kmz by .zip. Unzip the file (you may have to do it via a Terminal). This will extract a .kml script and the image(s) or .dae model(s).

Geo-referencing SU 3D models (dae) or images (png, jpeg) for Google Earth

1/ Open a 3D.dae, or an image (cross-section.png) in SketchUp

2/ **Camera > Standard Views > Top**

3/ **Camera > Parallel Projection**

4/ Move a reference point of the model on the SketchUp origin (Xo, Yo). This point could be a road intersection, a mountain peak, any feature that can be easily and accurately located on Google Earth.

5/ Measure a characteristic length in your model, it could be its length, or the distance between two prominent features.

6/ Save the file keeping the .dae or .png format.

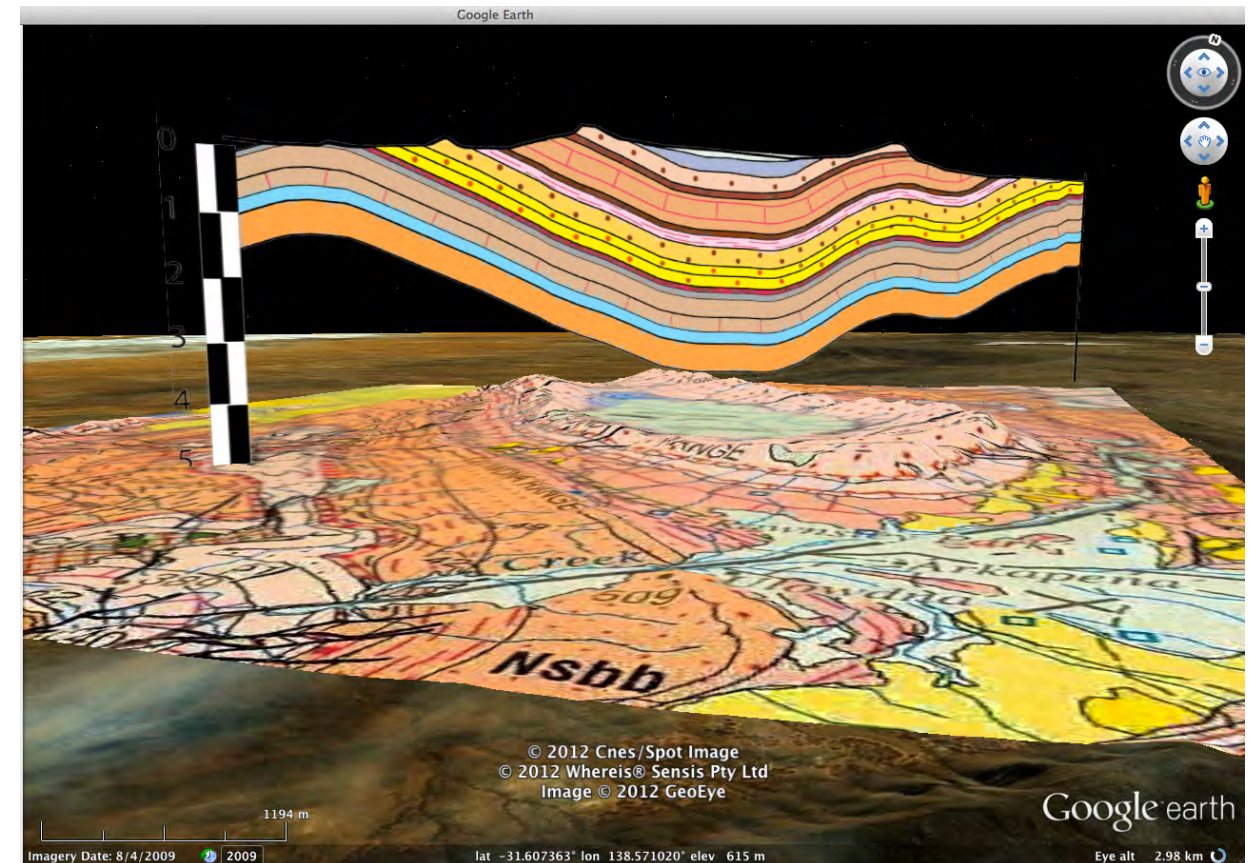
7/ Copy and save the kml script on the next page (with extension .kml) and update when necessary the red bits to fit your model and its location on GE.

8/ Save and drag and drop this file into GE.

To drape an image (map, air photo, satellite image ...) over the Earth: From GE, **Add > Image Overlay**. Once properly draped, this image can be selected from the **Places** menu on the left side of GE and save to automatically generate a .kmz file in which a kml script and the georeferenced image are embedded. This kmz file can be shared via email etc and drag-and-dropped into GE.



```
<kml xmlns="http://www.opengis.net/kml/2.2" xmlns:gx="http://www.google.com/kml/ext/2.2"
xmlns:kml="http://www.opengis.net/kml/2.2" xmlns:atom="http://www.w3.org/2005/Atom">
<Document>
<Folder><name> Flinders Ranges </name>
<Placemark>
<name> Wilpena Xsection </name>
<LookAt>                                <!-- Observer position upon loading -->
<longitude> 138.90 </longitude>          <!-- Decimal degrees, can be read from GE -->
<latitude> -31.81 </latitude>
<altitude> 22390 </altitude>
<heading> 316 </heading>                 <!-- Direction toward which the observer faces -->
<tilt> 90 </tilt>                        <!-- Observer tilt with respect to vertical: 90 is horizontal -->
<range> 30000 </range></LookAt>         <!-- Distance from target in meters -->
<Model id=" Xsec1 ">
<altitudeMode> relativeToGround </altitudeMode>
<Location>                               <!-- Lat-long of the dae model to load in GE -->
<longitude> 138.51 </longitude>          <!-- Longitude of SketchUp origin -->
<latitude> -31.63 </latitude>            <!-- Latitude of SketchUp origin -->
<altitude> 0 </altitude></Location>
<Orientation>                            <!-- of the 3D dae model with respect to EW -->
<heading> -44 </heading>                 <!-- East-West is 0, positive if clockwise -->
<tilt> -90 </tilt>                       <!-- Dip of the cross section, 90 is vertical -->
<roll> 0 </roll></Orientation>
<Scale>                                  <!-- Scaling factor of the 3D dae model determined from ... -->
<x> 7659.16 </x>                        <!-- ... the characteristic SketchUp length and ... -->
<y> 7659.16 </y>                        <!-- ... its corresponding length in GE. -->
<z> 7659.16 </z></Scale>                <!-- Homogeneous scaling: x = y = z -->
<Link>
<href> Wilpena_Cross_Section.dae </href></Link> <!-- Path to the model -->
</Model></Placemark></Folder></Document></kml>
```



[http://en.wikipedia.org/wiki/Geographic information system](http://en.wikipedia.org/wiki/Geographic_information_system)

<http://www.ga.gov.au/hazards.html>

[http://en.wikipedia.org/wiki/Comparison of GIS software](http://en.wikipedia.org/wiki/Comparison_of_GIS_software)

<http://earthquake.usgs.gov/research/data/>

<http://maps.unomaha.edu/maher/GEOL2300/week9/ex9ArcGIS.html>

<http://pubs.usgs.gov/tm/2005/12A01/>

<http://earthquake.usgs.gov/research/software/>

<http://www.opensha.org/>

<http://earthquake.usgs.gov/research/modeling/coulomb/>

<https://profile.usgs.gov/rstein>

http://www.ehow.com/how_7314751_create-terrain-google-sketchup.html

QuakeCaster: [ubs.usgs.gov/of/2011/1158/](http://pubs.usgs.gov/of/2011/1158/)

http://www.3dworldmag.com/2009/10/21/10_must_have_sketchup_plugin/

<http://www.gees.ac.uk/pubs/guides/eesguides.htm#fwgeosciguide>

Paraview

Paraview is a powerful open-source, multi-platform and extensible data analysis, processing and visualization application. Paraview brings 3D interactivity to GIS and non-GIS data, and works well on a simple laptop but also on high-performance computers to process extremely large datasets.

It is being used in a broad range of scientific and engineering disciplines. It has the convenience of point-and-click applications, with all the advantage of being scriptable via python.



Introduction

LOREM IPSUM

1. Website

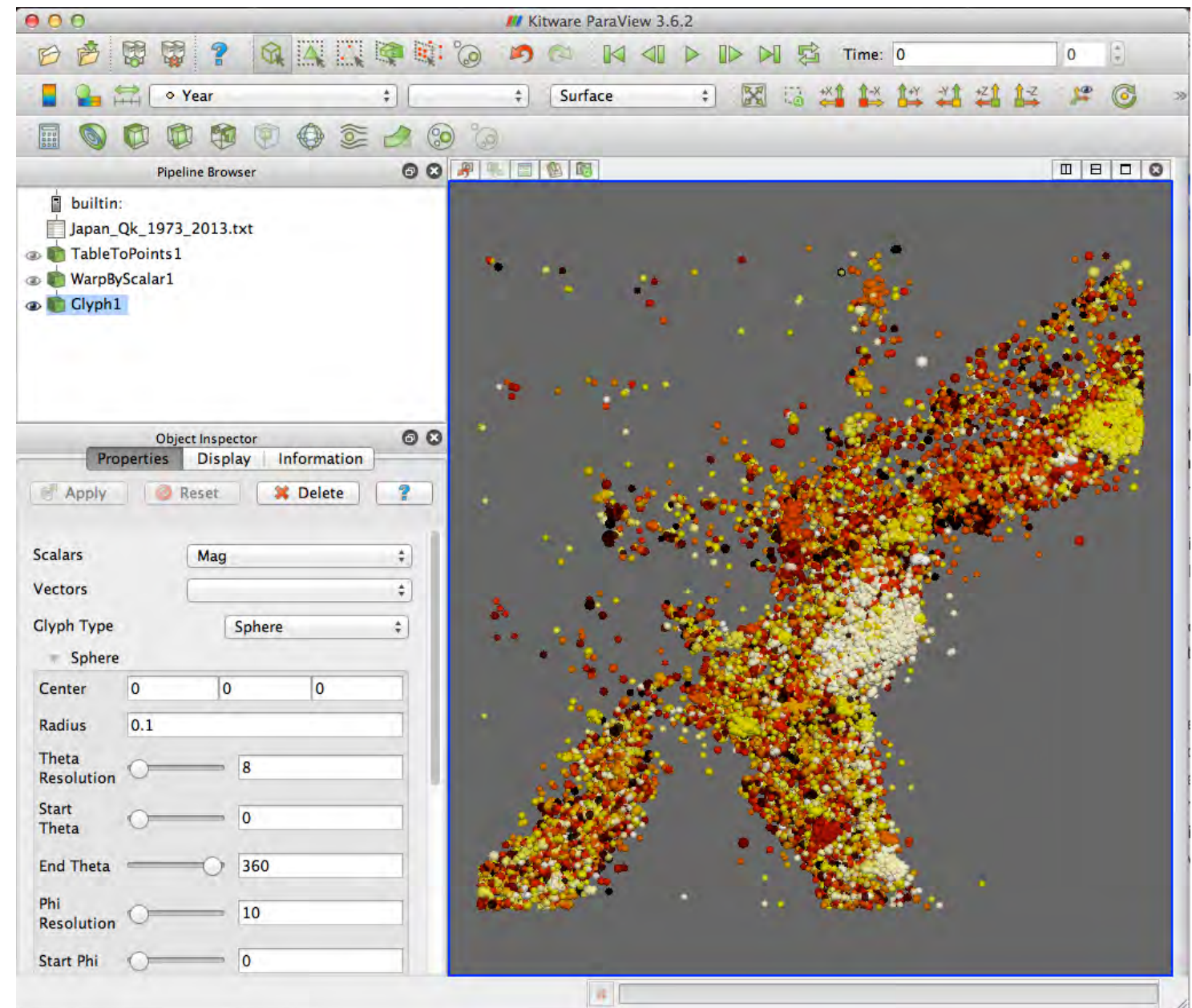
<http://www.paraview.org/>

2. Download

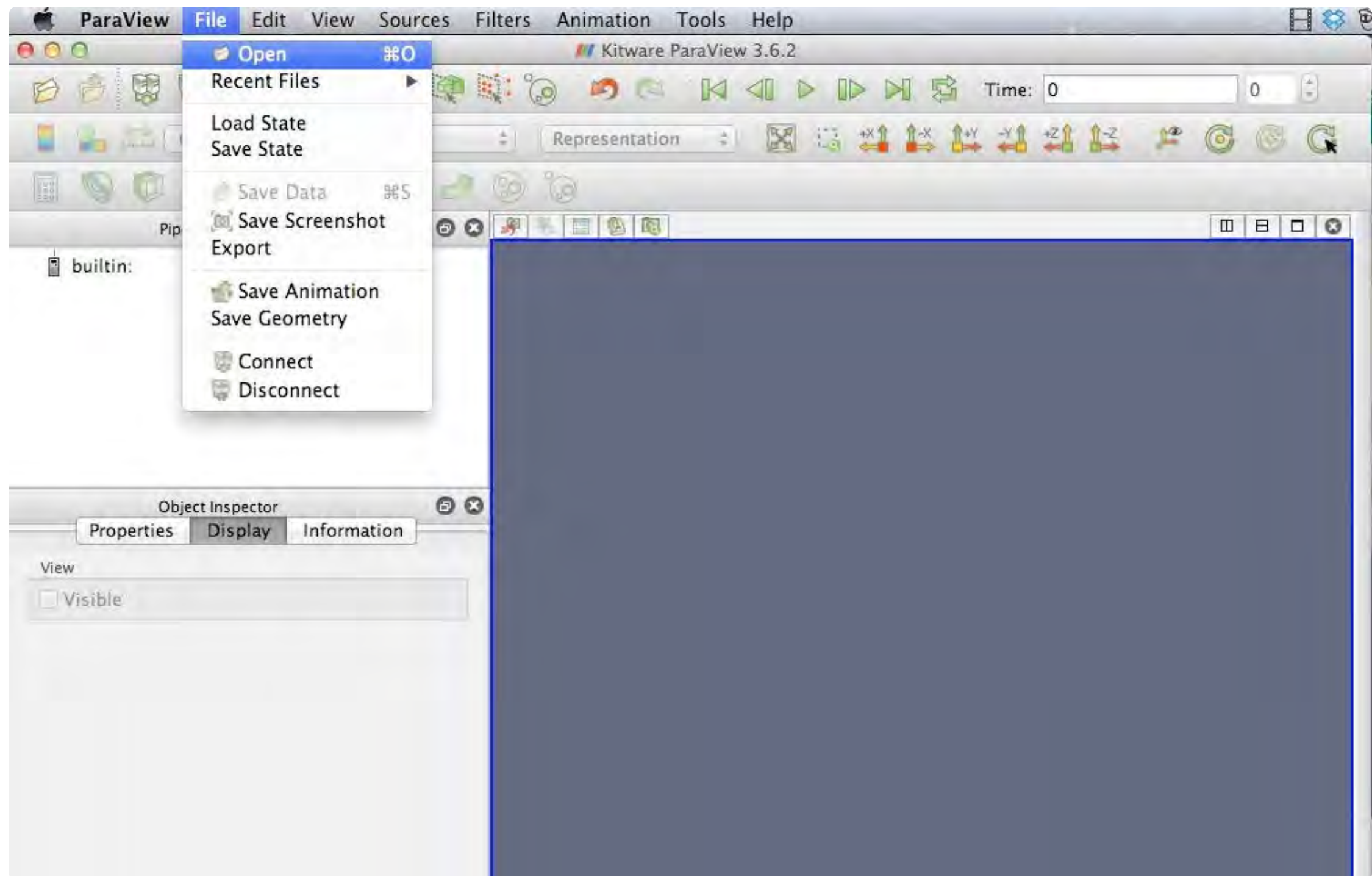
<http://paraview.org/paraview/resources/software.php>

Paraview allows the visualization of multidimensional dataset, including georeferenced data, hence Paraview has GIS capabilities.

The picture on the right shows a cloud of points representing 40 years of seismicity in Japan (from 1973 to 2013). It is the same dataset (Japan_Qk_1973_2013.txt) as the one used in the Section 2 of Chapter 3. Each earthquake is plotted as a bubble whose diameter is proportional to magnitude and colour represents the year of occurrence (dark is older, white is younger).



Gallery 7.1 Loading data into Paraview: slides 1 to 12. Visualizing data: slides 13 to 23



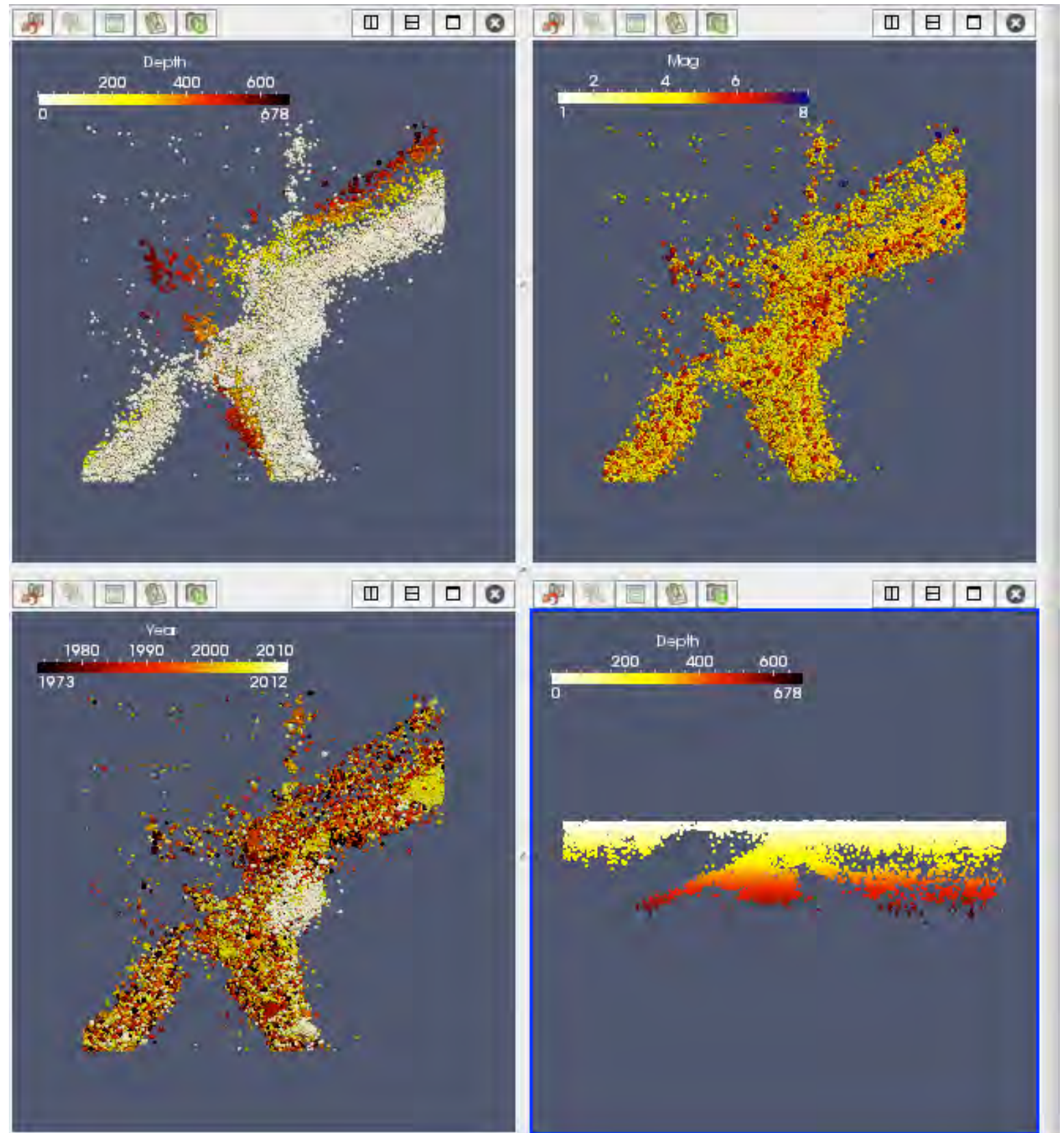
Interactive 3D visualization is one of the main advantages of Paraview over other GIS apps. Here our dataset is simultaneously visualized in 4 windows, each presenting the data in slightly different ways.

Top left: Map view, with depth-coloured earthquakes.

Top right: Map view, with magnitude-coloured earthquakes.

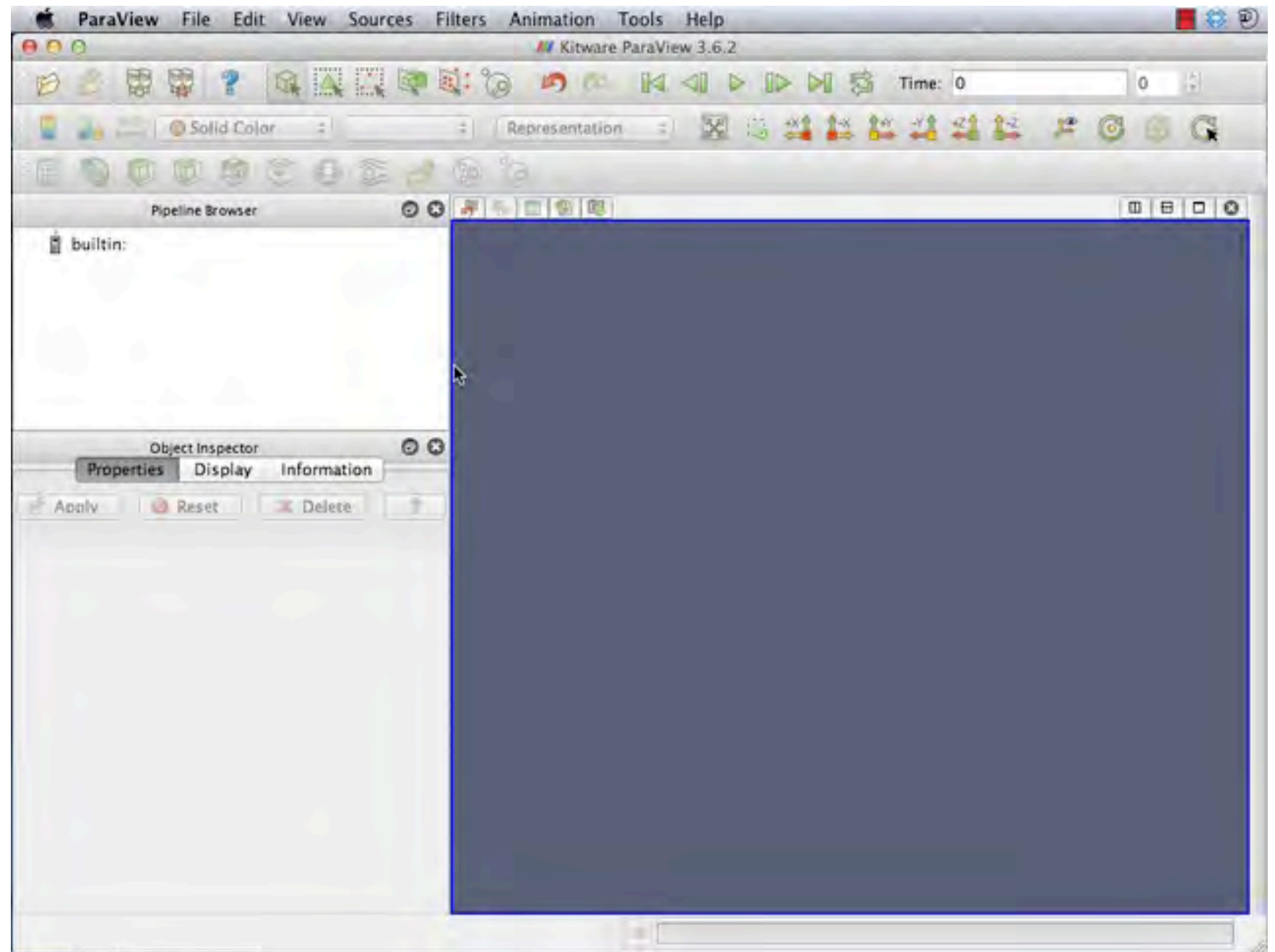
Bottom left: Map view, with year-coloured earthquakes.

Bottom right: Side view with depth-coloured earthquakes.



Statistical analysis - Paraview is also equipped to perform statistics. Cluster analysis, via k-means clustering, can be performed to partition the dataset into clusters. Paraview also offers multicorrelative statistics, temporal statistics as well as principal component analysis. The raw dataset (spreadsheet) is directly available within the Paraview environment, allowing for the direct analysis of data (minimum, maximum, standard deviation etc). ParaView is also fully scriptable using the simple but powerful Python language.

Movie 7.1 Cluster analysis in Paraview



3D Digital Elevation Model (DEM) in Paraview One can open a tiff raster colored for elevation and visualize this raster in 3D.

For this: 1/ load the tiff file, 2/ Filters > *Extract Surface*, 3/ Filters > *Tetrahedralize*, and 4/ Filters > *WarpByVector* using *Vector Tiff Scalars* and Scale Factor of 0.008999 (i.e. z values are in divided by 111.120). Et voilà, the image below shows a NW looking bird view of the Sydney Basin.

