Programming for Bioinformatics
BIOL 8803 B
September 14th, 2015

This week will be about getting/installing/running new programs. The commands and concepts for today are:

sudo – super-user do; do the command as a super user. Use with caution.

apt-get – package installation on Debian systems

rpm, yum – RedHat package manager

fink – Mac package manager

PATH – the environmental variable that the shell uses to search for executables

bin – typically the name for a folder that contains executables. Short for binary.

cc, gcc and icc – C compilers

configure – a script which builds the makefile

make – build the executables

install - put the executables where they belong, usually you don't call it yourself

git – a version control program, currently the hottest

cvs – older, but still good version control

Exercises

1.) sudo and installation, removal, etc.

sudo is very powerful, but dangerous. In short, it allows you to do anything on the system. On systems you don't own you probably won't have sudo privileges. On those where you do, please use it with care and be respectful of other people's data.

a.) People who have Ubuntu:

i) Use apt-get to install gcc without sudo. What happened?
apt-get install gcc
After executing this command permission is denied as root permissions are required for installing any software.

ii)Try again with sudo.
sudo apt-get install gcc
Now it downloads the latest version of gcc .

iii)Why does it work now?
It worked now as with sudo we get the root permissions to download and install the program.

iv) A lot of installation on Ubuntu can be handled with apt-get. Again: **Don't use it for the exercises here**; not everything has a neat little package and you need to know how to install things that don't.

b.) If you have a Mac and you are feeling bold, finish the other exercises and install fink. Try (a) above using fink instead of apt-get.

2.) The PATH to enlightenment

The PATH environmental variable tells the shell where to look for executables. When you use the ls command, it works because the shell knows to look for it on the PATH.

a)Print your PATH environmental variable to the screen. What are the directories in it? Some of them should look familiar at this point.
Print PATH environmental variable as follows :
echo $PATH
Directories in it are :
/usr/lib/lightdm/lightdm:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games

b.) Make a directory called 'bin' in your home directory and put a symbolic link to some already existing executable in the newly created bin directory. Call the link something different from the actual executable or this is all for naught.
tannishtha@tsom3:~$ mkdir ~/bin
tannishtha@tsom3:~$ cd bin

Now I create a symbolic link to cat command which is present in /bin folder and call it concat :
tannishtha@tsom3:bin$ ln -s /bin/cat concat

c)Add your bin directory to your PATH environmental variable. Keep in mind what character separates the different directories in the PATH environmental variable. If you mess up your PATH, then just leave the shell and make a new one.
tannishtha@tsom3:bin$ export PATH=$PATH:/home/tannishtha/bin
tannishtha@tsom3:bin$ echo $PATH
/usr/lib/lightdm/lightdm:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/home/tannishtha/bin
Thus we see that the newly created bin folder is there in my PATH environmental variable.

d)Try running the executable through the symbolic link. Did it work? Use which to see the alias.
Concat > abc.txt
Yes,it worked.
tannishtha@tsom3:~$ which concat
/home/tannishtha/bin/concat
Thus, which command gives me the path of the concat command to be in the bin folder in my home folder.

e.) Add your new bin directory to your PATH in your login file (you should know what this means). Make sure you didn't everything. If you did, get it right.
Added the following line to .bashrc file:
export PATH=/home/tannishtha/bin:$PATH

f.) Congratulations! You now have a centralized folder where you can install executables and will be on your PATH every time you log in. If you don't have permission to install things, install them here!

g.) If you're on a system with multiple users, what might be an advantage of installing an executable in one of the directories on the default PATH?
An advantage of installing an executable in one of the directories on the default PATH is that updates made to the executable by the system administrator are inherited by all the users and thus that saves time and work.

3.) The C compilers

There are various C compilers available today. They all pretty much do the same thing, turn C code into a program, but they do it with widely varying efficiency.

a.) Find out what your cc command really goes to, i.e. what is your default C compiler?

tannishtha@tsom3:~$ which cc
/usr/bin/cc
tannishtha@tsom3:~$ ls -l /usr/bin/cc
lrwxrwxrwx 1 root root 20 May 28 2012 /usr/bin/cc -> /etc/alternatives/cc

tannishtha@tsom3:~$ ls -l /etc/alternatives/cc
lrwxrwxrwx 1 root root 12 May 28 2012 /etc/alternatives/cc -> /usr/bin/gcc
tannishtha@tsom3:~$ ls -l /usr/bin/gcc
lrwxrwxrwx 1 root root 7 May 28 2012 /usr/bin/gcc -> gcc-4.6
tannishtha@tsom3:~$ ls -l /usr/bin/gcc-4.6
-rwxr-xr-x 1 root root 353216 Apr 15 2012 /usr/bin/gcc-4.6
Thus my default C compiler is gcc-4.6

b.) Compile the haveSomeNumbers.c file from T-Square and run the result (you're going to need the –o flag. Read about it before using it). Add the result to your PATH and run it that way.
tannishtha@tsom3:~$ gcc -o haveSomeNumbers haveSomeNumbers.c
tannishtha@tsom3:~$ haveSomeNumbers
bash: /home/tannishtha/bin/haveSomeNumbers: No such file or directory

- This executable doesn't run as it is not in my PATH.

Now I add this into my PATH (in the bin folder in my home directory) and run the result as follows:
tannishtha@tsom3:~$ mv haveSomeNumbers ~/bin
tannishtha@tsom3:~$ haveSomeNumbers

- It runs properly now as it is added to the path.

c)C is beyond the scope of this course, but try and figure out what the haveSomeNumbers.c file does anyway. This program is way simple. Many, many bioinformatics programs are written in C.
This program prints the numbers from 1 to 100.

d.) Look up comparisons of the gcc and icc online. What are the advantages and disadvantages of each? What other C compilers exist?

Advantages of gcc : Open source
Produces extremely portable code
Disadvantages of gcc : Not very good in optimizing code

Advantages of icc : High-level optimizations for computer systems using processors that support Intel architectures.
Produces faster code than GCC
Disadvantages of icc : Because the code is optimized only for systems with Intel architecture, the code is not portable.

Othe C compilers which exist are Turbo C, Visual C++ Express, Clang LLVM.

I assume you'll have some basic knowledge on what compilers are and what it means to compile a program no matter, how big or how small, at this point. In the following exercises we are going to do something that will be very pertinent to all of you regardless of the area you work in (or will be working in). Although in the following exercises, you'll be doing a specific task (calling variants and changing the ref version), the basic procedure of how you accomplished the task remains the same across the fields.

Basic Concept: The field is moving from assembling genomes from scratch to mapping to an already assembled genome with high level of annotations. This saves enormous amount of time, computational resources as well as human resources. Is you work with a eukaryotic genome with an assembled reference, you will be working this way. If you work with prokaryotes, at some point you will do this too.

The whole process starts with mapping the sequencing reads to a reference genome. The output of the process is a SAM (Sequence Alignment Map) file. This file is converted to a BAM file (Binary Alignment Map), the compressed brother of SAM. That's your starting point today. The following steps are sorting the BAM file, converting it to MPILEUP format, calling the variants.

The additional wrinkle today is that the version of the reference genome you mapped your file to, is old now and there is a new version out. You already have done variant calling and a set of analysis and would not like to repeat the whole analysis. Is there a way you can translate your coordinates of your initial variant calling to match the new reference genome version? Yes – the tool is part of Kent source, a large repository of tools written primarily by Jim Kent (a.k.a. the god) to do a lot of operations on genomic data. Try googling what you can do with it.

Now let's do the whole thing as an exercise here.

Basic Workflow: In order to achieve this, we'll have to:

1) Sort the BAM file
2) Convert BAM to MPILEUP
3) Call variants
4) Lift the coordinates to a different ref genome version

In order to do this, we have the following tools at our disposals:

1) Sort the BAM file – samtools sort
2) Convert BAM to MPILEUP – samtools mpileup
3) Call variants – bcftools
4) Lift the coordinates to a different ref genome version – liftOver from Kent Source

So we need to install samtools and Kent source. Samtools is straightforward but Kent has multiple dependencies. MySQL is one of them. So we'll start by installing samtools, MySQL and Kent in order, from scratch. For any other dependencies – feel free to use sudo apt-get for this exercise.

4.) Compiling binaries.

Samtools is a program for mapping second-generation short read sequencing data to a genome. It is very, very fast and is great for mapping sequencing data.

a.) Download the latest version of the samtools program source code (http://www.htslib.org/), not the pre-compiled binaries.
Downloaded samtools-1.2 and bcftools-1.2 from the website.

b.) Unpack the source browse the Makefile, some parts should look vaguely familiar. What does a Makefile do?
Unpacking the source -
tar -xvjf samtools-1.2.tar.bz2
tar -xvjf bcftools-1.2.tar.bz2
j is used to extract .tar.bz2 files.

A Makefile tells make how to compile and link a program. The Makefile associates short names called target with a series of commands which make uses to compile the program.

c)Build samtools and its associated programs with make.
Initially executing make command gave an error as curses.h library was missing. Installed it by executing the following line:
sudo apt-get install libncurses-dev
Then installation of samtools was succesful.
For compiling bcftools we go to the unzipped folder and type make and installation is successful.

d)Put the created binaries somewhere on your PATH, e.g. in your bin directory.
The created binary is called samtool. Copied it to my bin directory
tannishtha@tsom3:samtools-1.2$ cp samtools ../../bin/
In the bcftools folder, the executables are called bcftools and vcfutils.pl. These are copied to my ~/bin folder.

e.) Run them to see what happens. You should just get some help information or errors as you didn't give them any input.
Ran it from my home folder as follows:
tannishtha@tsom3:~$ samtools
tannishtha@tsom3:~$ bcftools
tannishtha@tsom3:~$ vcfutils.pl
The execution gave some help options.

5.) A harder install with MySQL

MySQL is a relational database management system. If that doesn't mean anything to you right now that's okay, but databases are extremely useful in bioinformatics. I recommend relational databases (taught in CS 4400) for everybody. I have a mountain of SQL books if you are interested. MySQL is also a good example for typical compilation/installation.

a.) Download the latest source code for MySQL (http://www.mysql.com/downloads/mysql/), not the precompiled binaries.
Downloaded mysql-5.6.26.tar.gz

b.) Unpack the source and run cmake . in the just created directory. You won't have cmake in your system, get it from apt-get.
Unpacked the source as follows:
tannishtha@tsom3:Downloads$ tar -xvzf mysql-5.6.26.tar.gz
tannishtha@tsom3:Downloads$ cd mysql-5.6.26/
tannishtha@tsom3:mysql-5.6.26$ cmake .

c)Build the MySQL executables with make
tannishtha@tsom3:mysql-5.6.26$ make

d)Try to install them with make install
tannishtha@tsom3:mysql-5.6.26$ make install

e.) That should have failed. Why?
Yes it failed. Because cmake needs to create directory: /usr/local/mysql/docs which it can't as it does not have administrative privileges.

f.) How would you get around this with sudo, and how would you get around this with the cmake (hint: you have to tell cmake where YOUR bin directory is. Run cmake –help)?

Typing 'sudo make install' installs the program.

Doing the same thing by cmake requires using the CMAKE_INSTALL_PREFIX option at configuration time which mentions the source directory for installing the software. This is done as follows:

```
cmake -DCMAKE_INSTALL_PREFIX=/home/tannishtha/bin
```

6).Version control with git and cvs

When you're working on a big project you need a way to track changes in your code and share code. cvs and git are utilities which do just that. It doesn't really matter to me which one you use, though git is very popular these days. For right now, we're just going to use them to download existing code.

a.) Install them both with apt-get.
sudo apt-get install cvs
sudo apt-get install git

7.) Ultimate installation challenge: The Kent source tree

The Kent source tree is a huge bundle of bioinformatics utilities named for James Kent, the brain behind the UCSC Genome Browser. He looks like a hobo. The Source Tree has come in handy many times for many people. Although you can get binaries nowadays, you wouldn't learn how to install things which have similar dependencies as not everything is available precompiled. So: **don't download the precompiled binaries for this exercise**.

a.) Obtain the source code for the Kent Source Tree using git as described at http://genome.ucsc.edu/admin/git.html
Source code can be obtained from git as follows :
git clone git://genome-source.cse.ucsc.edu/kent.git

b.) The README file, or some variation thereof is often very useful for installation. Read it to figure out what you need to do to install the source tree. If you aren't setting your architecture type, you aren't doing it right.

First of all we need to check that the environment variable MACHTYPE exists on our system. The default MACHTYPE is often a long string like "x86_64-pc-linux-gnu" for my system which will not function correctly in this build environment. It needs to be without any alpha characters such as: - . 'uname -m' or 'uname -p' or 'uname -a' tells us how my system reports itself which is x86_64 in my case. The MACHTYPE variable needs to be set to this. This is set by the followiing command:

export MACHTYPE="x86_64"

Then me create the directory ~/bin/$MACHTYPE which is where the (non-web) executables will go and add this directory to our path. Then we do a make in the src/lib directory and then src/jkOwnLib directory. Then we do a make for the application we want to run.

This is how kent is installed from the source.

c.) Attempt to compile the source tree. It *should* work. If you get stuck, how can you resolve the issue? (Perhaps look into the environment variables). You don't need to install the whole thing, which is a huge pain. Individual programs can be built by going into their directories and using make to compile the programs as you need them.

Since this a challenge and a huge pain if you aren't doing it right – come to me once you've exhausted every possibility. It's fun once you get it right.

HINT: This exercise is all about dependencies and environment variables.

Firstly we go to kent/src/lib directory for compiling.
tannishtha@tsom3:lib$ make
Executing it for the first time gave an error :
udc.c:39:25: fatal error: openssl/sha.h: No such file or directory
compilation terminated.
make: *** [udc.o] Error 1
So installed openssl header files by the following command:
tannishtha@tsom3:lib$ sudo apt-get install libssl-dev
Now typing make compiles the folder.

Then we go to src/jkOwnLib folder and type make :
tannishtha@tsom3:lib$ cd ../jkOwnLib/
tannishtha@tsom3:jkOwnLib$ make
This compiled in one go without giving any error.

Next if we want to install BLAT we execute make in each of the following directories:src/gfServer, src/gfClient, src/blat, src/utils/faToNib.

In the same way, we can compile whichever programs we want to by going to their directories and executing make.

8.) Lifting variants from one version to another.

This is a real, bona-fide task that is done quite often by many people. Reference genomes are updated continually. The coordinates that you have today might or might not change in the next release. It doesn't make sense to keep repeating the analysis every time a new version is out. That would be horrible to do. So let's be a little smarter here.

a.) Download (wget/curl) the BAM file from http://jordan.biology.gatech.edu/biol8803b/. This BAM file contains reads mapped to chromosome 21 of human genome version 18 (hg18). Again, a BAM is a binary version of SAM file. Which means you can't read it as it is (unless you are some sort of cyborg). But if you want to know what kind of information is there in the file, check here: http://samtools.github.io/hts-specs/SAMv1.pdf

The chr21 file for hg18 is located on the UCSC genome browser's download page. It can also be reached directly from this link http://hgdownload.soe.ucsc.edu/goldenPath/hg18/chromosomes/

Downloaded the BAM file as follows and kept it in ~/Downloads folder :
wget http://jordan.biology.gatech.edu/biol8803b/bamExample.bam
Downloaded chr21 file as follows :
wget http://hgdownload.soe.ucsc.edu/goldenPath/hg18/chromosomes/chr21.fa.gz
Unzipped it as follows:

tannishtha@tsom3:Downloads$ gunzip chr21.fa.gz

b.) Now let's sort the file. Whenever you map reads to a reference, they will come out in the order they went. But when you call variants, you want the order to be according to the genomic coordinate as you walk through the genome to perform variant calling.

The command is simple, see here: http://samtools.sourceforge.net/samtools.shtml.
samtools sort bamExample.bam bamExample_sorted

c.) Great! Give me some variants now!

samtools mpileup -uf ref.fa aln.bam | bcftools view -bvcg - > var.raw.bcf

bcftools view var.raw.bcf | vcfutils.pl varFilter -D100 > var.flt.vcf

NOTE: DON'T COPY PASTE. IT WILL NOT WORK. The files ref.fa and aln.bam are your reference genome file and sorted alignment file respectively. The files var.raw.bcf and var.flt.vcf are the output file which can be named anything. Keep the extensions (bcf and vcf) so that you know what they are.

Source: http://samtools.sourceforge.net/mpileup.shtml. Try to understand what happened with the above commands.

NOTE 2: With GATK in picture, things have complicated a bit but in the spirit of not making this exercise harder, we'll stick to the two lines of code in the link

For the latest release of samtools and bcftools the above commands in the following way:
samtools mpileup -uf chr21.fa bamExample_sorted.bam | bcftools view > var.raw.bcf
bcftools view var.raw.bcf | vcfutils.pl varFilter -D100 > var.flt.vcf

The new VCF file is called var.flt.vcf.

Here samtools collects summary information in the input BAMs (Example_sorted.bam), computes the likelihood of data given each possible genotype and stores the likelihoods in the BCF format. It does not call variants.

Bcftools applies the prior and does the actual calling. vcfutils.pl converts the BCF file to a VCF file.

d.) In order to use liftover, we will require to convert the VCF file to BED format. This can be done by the following command:

awk '{if($1 !~ /^#/){print $1, $2, $2+1}}' var.flt.vcf > in.bed

Of course change var.flt.vcf to whatever name you gave. What did I just do in that command?

Here from the var.flt.vcf file the lines which don't have '#' in the beginning are extracted. In those extracted lines the first two columns are picked up which gives the chromosome number and position ID. The two columns are printed on in.bed file with another column which adds one to the value of the second column. Thus the three columns mean the chromosome number, start position and end position.

e.) Stay with me, we are almost through. Give the var.bed file that you just created to liftover and use appropriate mapping files.

The mapping files can be found here: http://hgdownload.soe.ucsc.edu/downloads.html. Scroll down to hg18 (cause that's what the data was mapped against), click on LiftOver files and select hg18Tohg19.over.chain.gz

To know how to run liftOver program, just run it without any arguments. The usage instructions is in the first 5 lines. That's how it typically is. It's not hard at all.

Downloaded hg18Tohg19.over.chain.gz mapping file.
Downloaded liftOver program from UCSC genome browser source.
Ran the program as follows :
liftOver in.bed hg18ToHg19.over.chain hg19.bed hg19unmapped.bed
The new file is called hg19.bed

f.) Congratulations! You learned how to move around in different versions of reference genomes. **Rename** the new bed file you just created as **hg19.bed, and submit** the new bed file on T-square.