CMG-bitools manual

Contents

1	VirtualBox and CMG-Biotools								
	1.1	Virtual computer setup	3						
	1.2	Create a virtual computer on your local hard-drive	3						
	1.3	Setting up a shared folder between the host and virtual computer	5						
2	Inti	Introduction to Unix							
	2.1	Some useful concepts	6						
	2.2	A brief overview of the command line shell $\ldots \ldots $	6						
	2.3	Directories and the file system	7						
	2.4	Working with files	7						
	2.5	Reading the contents of text files	8						
	2.6	Invoking executables	9						
	2.7	Redirection and pipes	9						
	2.8	File system permissions	10						
3	Inti	roduction	11						
	3.1	Command syntax	11						
	3.2	Example data	11						
	3.3	GenBank files	11						
	3.4	FASTA files	11						
	3.5	for loops	12						
4	Download genomes								
	4.1	Re-name GenBank files from numbers to organism names	13						
5	Ger	nome atlas	13						
6	\mathbf{Ext}	ract DNA from GenBank	15						
7	Calculate basic statistics								
8	8 Identify rRNA sequences in DNA								

9	Multiple sequence alignment of selected 16S rRNA sequences							
	9.1	Construct phylogenetic tree from 16S rRNA alignment.	18					
10	Pro	teomes	18					
	10.1	Extract genes and proteins from GenBank	18					
	10.2	Genefinding	19					
11	Ami	ino acid and codon usage	19					
	11.1	Comparing amino acid and codon usage	20					
12	Pro	tein BLAST matrix	21					
13	Pan	- and core-genomes	22					
	13.1	Pan- and core-genomes plot	22					
	13.2	Extract subset genes from pan and core genome plot	22					
	13.3	Gene frequency plot from a pan-core genome output	24					
14	Tips	s and Tricks	24					
	14.1	Remove FASTA entries with no sequence	24					
	14.2	Modify names in BLAST matrix *.ps file	24					
		14.2.1 HASH lines	24					
		14.2.2 Bounding box	25					
	14.3	Convert *.ps file to *.pdf	25					
	14.4	Rename files	25					
	14.5	Delete empty files	26					

1 VirtualBox and CMG-Biotools

- NOTE! It is possible to use the system on Note/Netbooks, but it is not recommended!
- NOTE! Your computer should have a minimum of 4 GB memory free!
- NOTE! If xubuntu at any point asks you to update, do NOT update xubuntu!

1.1 Virtual computer setup

You will install a program that will allow you to run this virtual computer. You will import a setup for the computer, a so-called *virtual hard-disk file*, that will hold all the tools you will need for a basic comparative genomics analysis.

1.2 Create a virtual computer on your local hard-drive.

- Download the program *virtualbox* from webpage (select the version that fits your computer): http://www.virtualbox.org/wiki/Downloads.
- Install the program VirtualBox following the installation steps.
- $\bullet~$ Start the VirtualBox
- Click new (the blue icon on the left of the tool bar).
- Type CMG biotools in the VM name dialog box
- Select linux (Operating system) and ubuntu (Version) in the Create New Virtual Machine dialog.
- Leave the *Memory* as default, if you are expecting to do very heavy computations set this as high as allowed (within the green area of the bar)
- Create a new hard disk
- Leave virtual disk creation wizard settings unchanged: file type = VDI
- Set memory allocation as dynamic
- Click create in the *Summary* dialog

The virtual computer has now been created and can access part of the physical computer. The next step is to read the disc image, this process is equivalent to inserting a CD into the physical computer.

- Download the latest version of CMG-biotools from the webpage: http://www.cbs.dtu.dk/staff/dave/CMGtools/
- Click the *Settings* tab of the CMG-biotools computer
- Click the tab *Storage*
- Part of this window is dedicated to the *Storage Tree* in which there will be a branch called *IDE Controller* Under this branch there should be a CD icon and the text *Empty*
- Select the *Empty* branch and click the CD icon to the right of the *CD/DVD Drive*(Figure 1(a))
- Click the "Choose a virtual CD/DVD disk file..." and find the folder where the CMG-biotools-version.iso is stored
- Click OK

The computer now knows which file to read when booting. Figure 1(b) shows how the computer is represented in the VirtualBox window To open the computer, click the *Start* button. When given the option, select *'live* - *boot the Live System*''. When the desktop appears, click the *Install* icon on the desktop and follow the instructions. Create a user and a password that you can remember. Restart the computer when asked and then you should be done.



(a) VirtualBox reading in the disc image

(b) VirtualBox representation of computer



Figure 1: Create a virtual computer on your local hard-drive using *VirtualBox*. Installing CMG-biotools on the virtual computer.

1.3 Setting up a shared folder between the host and virtual computer

A shared folder allows the user to access files from the host system from inside the virtual system. In this setup, files are shared over a network, in other words, you access remote files. For a virtual machine, the network between host and guest is virtual since they are on the same real machine. In order to share folders, **Guest Additions** must be installed. Guest Additions provide additional capability to a guest virtual machine, including file sharing. A version of Guest Additions is already installed on CMG-biotools.

Prepare host

- Create a folder on your normal (host) system that you want to use as the share folder, do not use default folders like *Documents* or *Desktop*. For the purpose of this tutorial the folder is named *share*
- When the virtual computer is turned off, go to the *Settings* tab and select *Shared Folders*.
- In the window you will se a *Folders List* with two lines, *Machine Folders* and *Transient Folders*. To the right you should see a folder icon with a green plus sign (add folder). Click this icon
- Select your host folder in the *Folder Path* field
- Put a tick-mark in both Auto-mount and Make Permanent
- The folder should now show up under Machine Folders
- Close the window and start the virtual computer

Prepare guest (CMG-biotools)

If the client is Linux, you have to mount and connect it to a directory. The following bash commands (in the client) would setup a correct mount (and creates a link from your desktop) Note: you should not use spaces in the share name. The shared folder on the CMG-biotools guest system is called Vboxshare. The shared folder on the guest system, like we defined above, is called share You will need your password to set up the shared folder.

```
# User name is "student"
sudo mkdir /home/student/Vboxshare
sudo chmod 777 /home/student/Vboxshare
sudo mount -t vboxsf -o uid=1000,gid=1000 share /home/student/Vboxshare
ln -s /home/student/Vboxshare $HOME/Desktop/Vboxshare
# NOTE: If you get the following error, change the vboxsf to vboxfs
mount: unknown filesystem type 'vboxsf'
```

To not run the mount command every time you need the shared folder, put the command in the file /etc/rc.local. Run the following command which will add the desired line at the right position in the *rc.local* file, right after the line with the text "By default this script does nothing.". That line is part of the original *rc.local* file.

```
# The following line should be typed on one single commandline
# User name is "student" and shared folder is called "Vboxshare"
sudo sed '/By default this script does nothing./a sudo mount -t vboxsf -o uid=1000,gid=1000 share /home/
student/Vboxshare' /etc/rc.local
```

Additional help: https://help.ubuntu.com/community/VirtualBox/SharedFolders

2 Introduction to Unix

This is a very simplified and rough introduction to using a terminal on a unix machine. The unix command line interface is a very powerful environment and there is much more to it than described here.

2.1 Some useful concepts

This is a a brief overview of some useful concepts in unix.

- The Shell. Although Unix has a graphical interface called X Windows, it is often easier and quicker to run programs by typing commands into a terminal window. Access to unix from other operating systems is usually conducted through a terminal client e.g. Putty for windows.
- Users. All programs are run as a specific user, so you have to log into the system as that user with a password.
- Files and processes. Everything is a file or a process and the input and output from files and processes can be sent to each other (see pipes and redirection).
- Permissions. All files, directories and programs have access permissions. A user cannot see the contents of a file or run a program unless the permissions allow.

2.2 A brief overview of the command line shell

This is what is run when you open a terminal window. It provides a lot of information and tools to help you run programs.

The command prompt: When you open a terminal, the text at the bottom of the screen next to the cursor will look something like this:

interaction[maq]:/home/projects/MicrobialGenomicsGroup>

This is useful because it tells you who you are and where you are. It can be configured in different ways but the example above shows the machine, the username in square brackets '[]' and the current directory after the colon ':'.

- Command line history and auto-completion: Previously entered commands can be edited or executed again using the up arrow key. Filenames can be auto-completed using the tab key.
- Environment variables: When you login to a machine or terminal, a set of variables, collectively called the environment are created. These variables do things like telling the shell where to look for programs. The printenv command will list all the environment variables. The list can be quite long.

```
      printenv SHELL
      #
      prints the executable for the current shell

      #
      Unix offers a selection of many shells all of which are subtly different

      printenv HOME
      #
      This command prints the location of the users home directory

      printenv PATH
      #
      The PATH variable is a list of directories which are searched when a command it typed

      #
      It is useful to add directories where executables are stored to the PATH variable
```

Getting help: Many unix commands have one or more manpages (manual page).

\mathtt{man}	printenv	#	Use	'q'	to	exit	the	manpage	
----------------	----------	---	-----	-----	----	------	-----	---------	--

2.3 Directories and the file system

The shell logs you into a directory in the file system. There are some rules for about the file system.

- Files, directories and executables are case sensitive: This means that x.txt and X.txt are two different files.
- Path delimiter: The unix shell uses the forward slash '/' to separate files and directories NOT the backslash '\' like MsDOS. There are some special characters which are used when defining the location of a file, directory or program.
- The root directory: The root directory is defined by the single slash '/' and represents to the first node of the directory tree. It is similar to 'C:' on a windows machine.
- The '.' directory: The '.' character is used to define the current directory when it is part of a file path.
- The home directory: Most users are assigned a home directory where files can be created. This can be referenced using the '~' character.

3

3

```
# Examples: Absolute versus relative paths
/usr/bin/perl # Absolute paths are defined from the root directory e.g.
./script.pl # Relative paths can be defined from the current directory e.g.
~/script.pl # Relative paths can be defined from the home directory e.g.
```

Here are some useful command to assist getting around the filesystem:

2.4 Working with files

Files reside in directories and can contain text or binary information. Files can be created, copied, moved, renamed and deleted with the following commands.

• 1s: lists the contents of directories. Run just as '1s' the command lists all the contents without any other information. More information can be gained by supplying some arguments.

ls	-1	#	list showing permissions, user, group. size, modification date and filename
ls	-lh	#	as above but file sizes are printed in human readable form
ls	-lhS	#	as above but sort the results by file size in descending order
ls	-lhSr	#	as above but in ascending order
ls	-1R	#	list the contents of all directories recursively from the current directory
ls	-lt	#	list files in ascending order of last modified
ls	-ltr	#	as above but in descending order

• touch: used with a filename. If the file does not already exist, a new one is created. Otherwise the date of the file is changed to the current time.

touch filename

• cp: copies filename1 to filename2 or into a directory and leaves the original file untouched. It can also be used to copy directories.

```
cp original_file new_file
cp file directory/
cp -r directory new_directory
```

• mv: moves a file from one to another and deletes the original file. It is used for renaming files. It can also be used to recursively move directories.

```
mv original_file new_file
mv file directory/
mv dir1 dir2
```

• rm: deletes a file, can also be used to delete a directory and the contents. use with care.

```
rm file
rm -r dir
```

• File name advice: It is best not to use spaces or special characters such as " ' < > @ in filenames. Underscores '_' and hypens '-' are fine.

2.5 Reading the contents of text files

The contents of text files (but not binary files) can be read quite easily through the terminal.

• cat: appends the contents of one file into another

```
cat file1 file2
```

• more: shows the contents of a text file. Press 'q' to return to prompt

```
more filename
```

• less: better than more because the up and down keys can be used to scroll up and down.

```
# Some useful key commands are:
# space: scroll forward one screen
# b: scroll backward one screen
# g: scroll to the top of the file
# G: scroll to the end of the file
# q: quit to the prompt
```

• /text: searches the file for the word "text"

less filename

• head: shows the first few lines of the given file(s). A hyphen and number can be passed to determine how much of the file is shown

```
head filename
head -5 filename
```

• tail: shows the last few lines of the given file(s). A hyphen and number can be passed to determine how much of the file is shown

tail filename tail -5 filename

• grep: Searches through text files for a search term and print matching lines. grep is a complex command and has many options. Try looking at the manual page for grep (man grep).

```
grep searchword file1 file2 # to search two files for a searchword
grep -v searchterm file1 file2 # to search two files to print lines excluding a searchword
grep -c searchterm file1 file2 # To count the number of matches per file
```

• wc: counts the number of words or lines a file contains

```
wc -w file.txt # print the wordcount of a file
wc -l file.txt # print the line count of a file
```

• wildcards: A number of characters are interpreted by the Unix shell before any other action takes place. These characters are known as wildcard characters. Usually these characters are used in place of filenames or directory names.

```
# An asterisk matches any number of characters in a filename, including none
? # The question mark matches any single character
[] # Brackets enclose a set of characters, any one of which may match a single character
- # A hyphen used within [] denotes a range of characters.
~ # A tilde at the beginning of a word expands to the home directory
# If another user name is appended to the character, it refers to that user's home directory
```

Here are some examples:

cat	с*	#	Displays any file whose name begins with c including the file c, if it exists.	1
ls	*.c	#	Lists all files that have a .c extension.	2
cp	/rmt?.	#	Copies files in parent directory to working directory if file matches:	3
		#	four characters long and begins with rmt.	4
ls	rmt [34567]	#	Lists every file that begins with rmt and has a 3, 4, 5, 6, or 7 at the end.	5
ls	rmt[3-7]	#	Does exactly the same thing as the previous example.	6
ls	~	#	Lists your home directory.	7
ls	~hessen	#	Lists the home directory of the guy1 with the user id hessen.	8

2.6 Invoking executables

Some files can be marked as executable which means they can run as programs and perform tasks. Executables residing in the PATH directories can be invoked without including the path to the executable. If a program called **perl** is found in the folder /usr/bin and the folder /usr/bin is part of PATH, the program can be called without giving the absolute or relative path to the program. Non-PATH program must be called giving an absolute or relative path.

```
perl myscript.pl # Program found in PATH
/usr/bin/perl myscript.pl # Non-PATH program
```

2.7 Redirection and pipes

Most unix processes write their output to the standard output (the terminal screen) and take their input from standard input (they keyboard). There is also a standard error which is usually printed to the terminal

screen. These inputs and outputs can be redirected to files. The standard output can be redirected using the '>' character

3

```
ls -l > list.txt  # Appending to a file
echo 'hello' >> list.txt  # Redirecting input
sort < filewithdata.txt  # Sorts lines in file
sort < input.txt > output.txt  # Both at the same time
```

Pipes ('|') allow processes to direct output directly to other programs. Pipes can be put together to allow complex "pipelines" of commands to be put together. The output of command 1 can be passed to command 2 as follows:

ls -1 | grep -v '*.txt' | grep -c '*.coli*'

This pipeline will first list the filenames in a directory, then exclude (grep -v) the once with the file extension (.txt) and then count (grep -c) the once with he extension (.coli).

2.8 File system permissions

File system permissions ensure that file contents or executables can only be examined or invoked by users with the correct authorization. They can often also be a source of problems when using data or programs created by others where you can't access a file or directory due to the permissions set. To view permissions type 'ls -l' in a directory containing some files, the output will look like this:

-rw-r---- 1 user1 cdrom 11802 Jul 9 10:02 file.txt

The 10 character string '-rw-r---' describes the permissions. The hyphen indicates that the permission has not been granted and 'r' indicates read permission, 'w' indicates write and 'x' indicates that the file can be executed. There are three sets of 'r', 'w', 'x' or '-' to control access by the user to whom the file belongs, members of the same group and anyone else. e.g. '-rwxrwxrwx' means anyone can read, write and execute the file

```
whoami # will display your username
groups # will display the groups you are a member of
chmod # file permissions can be changed using the chmod command
```

The chmod command takes a complex set of arguments. The user, group or other are represented by 'u', 'g' and 'o'. The letter 'a' is used to represent all whether permissions are granted or revoked is determined using '+' or '-' respectively read, write or execute permissions are represented by 'r', 'w' and 'x'.

```
chmod go-wx data.txt # remove write and execute permissions for the group and others try
chmod a+rw data.txt # to give everyone read and write permissions
```

Useful pages with Unix introductions include: http://www.ee.surrey.ac.uk/Teaching/Unix/

3 Introduction

3.1 Command syntax

Below is seen an example of how commands will be shown in these exercises. The example task is how to create a folder, GenBankDNA. Whenever a word is marked with <> it indicates that a word should be inserted here WITHOUT the <> signs. Below is shown the syntax that will be used in this document and what the command should look like when used.

```
syntax command: mkdir <directory_name>
typed command: mkdir GenBankDNA
```

3.2 Example data

To illustrate the different steps in this manual, a folder with example data is included in the CMGbiotools. The folder is called VeillonellaExample.tar.gz and is a compressed archive. Go to the folder /usr/bitools/ and click the VeillonellaExample.tar.gz file. An application will open, called Archive Manager. Click the option Extract files from archive, and then select where the files should be extracted/stored. The package holds GenBank, DNA FASTA, protein FASTA, RNA FASTA, codon and amino acid usage and genome atlas for a set of genomes from the genus Veillonella.

3.3 GenBank files

Locate the example GenBank files and investigate the content. Open the file in a text-editor (CMG-biotools has a text-editor called mousepad and another called gedit).

mousepad <name>.gbk</name>	-
gedit <name>.gbk</name>	4

In the beginning of the file is the metadata, names, publications, habitat and similar information. The next part is the annotations, genes and CDS (CoDing Sequences). In this section the genes are described by their location, direction, note, and translation.

3.4 FASTA files

Look at the FASTA formatted files created by Prodigal (use head, tail, cat, less or the texteditor). The wiki entry about FASTA says:

" In bioinformatics, FASTA format is a text-based format for representing either nucleotide sequences or peptide sequences, in which base pairs or amino acids are represented using single-letter codes. The format also allows for sequence names and comments to precede the sequences. The format originates from the FASTA software package, but has now become a standard in the field of bioinformatics. http://en.wikipedia.org/wiki/FASTA_format "

Underneath here is shown an example of a FASTA file, with a description header and an amino acid sequence. The sequence lines within a FASTA entry (header + sequence) make up the entire sequence of a gene.

```
>SEQUENCE_1
MTEITAAMVKELRESTGAGMMDCKNALSETNGDFDKAVQLLREKGLGKAAKKADRLAAEG
LVSVKVSDDFTIAAMRPSYLSYEDLDMTFVENEYKALVAELEKENEERRRLKDPNKPEHK
IPQFASRKQLSDAILKEAEEKIKEELKAQGKPEKIWDNIIPGKMNSFIADNSQLDSKLTL
MGQFYVMDDKKTVEQVIAEKEKEFGGKIKIVEFICFEVGEGLEKKTEDFAAEVAAQL
>SEQUENCE_2
SATVSEINSETDFVAKNDQFIALTKDTTAHIQSNSLQSVEELHSSTINGVKFEEYLKSQI
ATIGENLVVRRFATLKAGANGVVNGYIHTNGRVGVVIAAACDSAEVASKSRDLLRQICMH
```

From this point forward that the number of genes/proteins is equivalent to the number of headers. As such, counting the number of annotated genes in a GenBank file, can be done by counting the number of '>' signs in the corresponding FASTA file. A simple Unix tool for finding sequence patterns in a file is the program grep. Here we will use grep to count how many times the '>' sign is found in a given FASTA file. It is important to remember the " signs, otherwise the sign will be viewed as a output redirect. grep will only find the line, with the search-pattern. To count the number of times the pattern is found grep has a -c option, which will print the number on the screen.

```
grep ">" <name>.orf.fsa
grep -c ">" <name>.orf.fsa
```

3.5 for loops

When working with comparative genomics, it is often necessary to run the same analysis on multiple files. This can be done by constructing loops using the program for. A for loop will perform the same command on all files in a list. Take some time to play with these loops before you go into advances things, as you might accidentally overwrite your file or similar. To wrap the grep command from above it is necessary to add an extra command that will show the filename (which is the organism name). The command is called echo. The for loop will look like this:

```
for x in <name1> <name2> <name3> <name4>
do
echo $x
grep -c ">" <name>.orf.fsa
done
```

The following will allow you to run a command on all files with the extension .orf.fsa without specifying the names of the files. The counts will be written to the screen along with the name of the file.

```
for x in *orf.fsa;
do
echo $x
grep -c ">" $x
done
```

4 Download genomes

Obtaining genome data (metadata, DNA and annotations) is not a straight forward process. This is largely due to the reliance on public databases that get updated and modified over time. The analysis presented here works on unpublished data as long as the file formats are as required. Bellow is shown a procedure for downloading genome data from the National Center for Biotechnology Information database (NCBI).

The BioProject database of NCBI (http://www.ncbi.nlm.nih.gov/genome/browse/) allows the user to browse genome projects based on organism, status, or other attributes. From this list it is possible to obtain so-called International Nucleotide Sequence Database Collaboration (INSDC) and Whole Genome Sequence (WGS) numbers. These numbers can be used as input to the program textttgetgbk that downloads the corresponding genome data as GenBank file formats. The WGO numbers are a little tricky because the refer to a large number of data entries, all connected to one genome project. In order to get the records for the entire project the WGS number must be modified slightly, see the example bollow.

Organism name					WGS	1
Mycobacterium	avium	subsp.	paratuberculosis	S397	AFIF01	2
Mycobacterium	avium	subsp.	paratuberculosis	Pt139	AFPC01	З
Mycobacterium	avium	subsp.	paratuberculosis	Pt144	AFPD01	4

The root of the WGS number is the first 4 characters (AFIF, AFPC and AFPD). In order to access the entire project, remove the '01' and add '00000000'. The resulting numbers have 12 characters (AFIF00000000, AFPC00000000 and AFPD00000000). The output from the program is a GenBank file equivalent to the files found on the webpage. The program option is -a which reads the input as an NCBI INSDC or WGS number. The syntax of the program is shown below. Note the Unix usage of the '>' sign, which is a redirection of the output into a file. If this is not included (getgbk -a <INSDC>), the program will write the output, the GenBank file, to the screen.

getgbk -a <INSDC> > <INSDC>.gbk
getgbk -a <modifiedWGS> > <modifiedWGS>.gbk

4.1 Re-name GenBank files from numbers to organism names

To make it easier to recognize files, they will now be renamed so they are called an organism name instead of a GPID number. From this point on, <INSDC> will be replaced with <name> and will refer to the organism name the file is given.

extractname <INSDC>.gbk

Note that the files are not moved, but rather, they are copied into a new file. Delete the numbered files using the command 'rm'. The new files will from here on be referred to as <name>.gbk in the command syntax.

5 Genome atlas

A genome atlas is a visual representation of genome properties, genes/proteins and patterns in DNA associated with DNA structures, helix, repeats and so on. A genome atlas is made from a GenBank file and requires a continuos piece of DNA. It is important to have only one replicon in your GenBank file (count number of LOCUS if you are not sure). The program is made to only run for one DNA string, a decission made because of the nature of the figure. Creating a circular atlas for a in-complete genome would not make biological sense. If you do wish to construct a atlas for a multi-contig genome, it is possible to "glue" the DNA strings together with in a FASTAA file. In order to construct an atlas, the DNA sequence is scanned for all kinds of patterns. This means that it takes time to prepare the files necessary for a genome atlas.

3

13

14

15

16

8

12

13

14

```
mkdir GenomeAtlas <acc>
                                    # Create folder to hold the files for the genome atlas
cp <name>.gbk GenomeAtlas<acc>
cd GenomeAtlas<acc>
                                    # Enter the atlas specific folder
# Replace the pattern 'XXXX' with the name/acc in the file called 'genomeAtlas'
sed s/XXXX/<name>/g /usr/biotools/genomeAtlas > <name>.genomeatlas.sh
# Example
# In the current working directory you have a file called A_sp_SE50_ID_CP003170.gbk
sed s/XXXX/A_sp_SE50_ID_CP003170/g /usr/biotools/genomeAtlas > A_sp_SE50_ID_CP003170.genomeatlas.sh
# NOTE: it is possible to make a atlas from local genefinding or published proteomes,
# but it is not possible to make an atlas for a whole genome sequence
chmod +x <name>.genomeatlas.sh
                                    # Make file executable
./<name>.genomeatlas.sh
                                    # Execute file
```

The following example shows how to create six atlases for six different genomes with prodigalrunner annotations. The pipeline is the same for a normal GenBank file with published annotations. The pipeline runs in two steps, first it create a folder per atlas and then copies the relevant GenBank file into that folder. Next, it create the .genomeatlas.sh file and executes it.

```
for x in A_fermentans_DSM_20731_ID_CP001859_prodigal A_intestini_RyC-MR95_ID_CP003058_prodigal
    M_hypermegale_ART12_1_ID_FP929048_prodigal M_elsdenii_DSM_20460_ID_HE576794_prodigal
    S_sputigena_ATCC_35185_ID_CP002637_prodigal V_parvula_DSM_2008_ID_CP001820_prodigal
do
mkdir GenomeAtlas_$x
cp $x.gbk GenomeAtlas_$x
done
for x in A_fermentans_DSM_20731_ID_CP001859_prodigal A_intestini_RyC-MR95_ID_CP003058_prodigal
    M_hypermegale_ART12_1_ID_FP929048_prodigal M_elsdenii_DSM_20460_ID_HE576794_prodigal
    S_sputigena_ATCC_35185_ID_CP002637_prodigal V_parvula_DSM_2008_ID_CP001820_prodigal
do
cd GenomeAtlas $x
sed s/XXXX/$x/g /usr/biotools/genomeAtlas > $x.genomeatlas.sh
chmod +x $x.genomeatlas.sh
./$x.genomeatlas.sh
cd ..
done
```

To do a zoom of a specific region, open the file called <name>.genomeatlas.cf and add the line described bellow. Note that this procedure should be run in the same directory as the not-zoomed atlas, as it uses the same files.

```
mousepad <name>.genomeatlas.cf # Open genome atlas file
circlesection <start> <end>; # Syntax
circlesection 515000 535000; # Example
circletics auto; # This line should be added under the line that looks like this
<name>.zoom.genomeatlas.cf # Save the file with new name
genewiz -p <name>.zoom.genomeatlas.ps <name>.zoom.genomeatlas.cf # Now re-run the atlas picture
```

6 Extract DNA from GenBank

The GenBank file format contains raw DNA sequence for a given genome project. Here the DNA will be extracted and searched for patterns similar to rRNA sequences. The extraction of DNA from GenBank could in theory be done manually, but there are much faster ways. Here it is done using a small extraction program called saco_convert which takes a GenBank file as input and stores the DNA in a FASTA format.

```
saco_convert -I genbank -O fasta <name>.gbk > <name>.fna
# The above can be wrapped in for-loop to make it faster
for x in *gbk
do
saco_convert -I genbank -O fasta $x > $x.fna
done
```

Look at the file and make sure that it contains DNA in FASTA format (use head, tail, cat, less or the text-editor). Number of '>' FASTA headers should be equal to the number of replicons (chromosomes or plasmids). Count the number of replicons using grep:

```
grep -c '>' <name>.fna
# The above can be wrapped in for-loop to make it faster
for x in *fna
do
echo $x >> proteinCounts.txt
grep -c ">" $x >> proteinCounts.txt
done
```

If you need to run this loop again, delete the proteinCounts.txt file first.

7 Calculate basic statistics

The next part is an analysis performed on a FASTA formatted file containing complete genomic DNA (<name.fna>), not genes or proteins (<name> proteins.fsa). Calculate the AT content (Per.AT), number of replicons (ContigCount), deviation of AT across replicons (StDevAT), percentage of unknown bases (Per.Unknowns) and total size in bp (TotalBases).

genomeStatistics <name>.fna

Output is by default written to the screen. You can copy the output from the screen window into a spreadsheet. The command line can be used in a for loop, but the amount of data written to the screen might be overwhelming. In that case the result can be redirected to an output file.

```
for x in *fna
do
genomeStatistics $x >> genomeStats.all
done
```

Note the '> >' signs, this means "append" instead of redirect. When using redirect, '>', a file is created if not already found or overwritten if already found. When appending, the output is added if the file is not found or appended if the file is already found. Copy the genome statistics into a spreadsheet (gnumeric).

Bellow is shown an example of a genomeStats.all file for three genomes. It is seen that the complete genome of *Actinoplanes* consists of 1 contig and as such, the largest sequence is 100% of the total DNA. For the remaining genomes, the data consist of 1146 and 176 contigs with very small sequences, 1 and 2.8% of the total DNA content.

3

3

```
Filename
            TotalBases: Per.AT: StDevAT:
                                            ContigCount:
                                                             Per.Unknowns:
                                                                             Per.LargestSeq
A_sp_SE50_ID_CP003170.fna
                           9239851 28.6828 0.0 1
                                                    0 100.0
            TotalBases: Per.AT: StDevAT:
                                            ContigCount:
                                                            Per.Unknowns:
Filename
                                                                             Per.LargestSeq
M_avium_ID_AFNS00000000.fna 4610244 31.0383 0.0255 1146
                                                             0
                                                               1.0841
            TotalBases: Per.AT: StDevAT:
                                                             Per.Unknowns:
Filename
                                            ContigCount:
                                                                             Per.LargestSeg
M_avium_ID_AFIF00000000.fna 4813711 30.6866 0.0269
                                                    176 0
                                                             2.8546
```

8 Identify rRNA sequences in DNA

For identifying rRNA sequences in DNA we will use **rnammer**, a program that implements an algorithm designed to find rRNA sequences in DNA¹. The program was made by modeling a large number of known rRNA sequences and making them into generalized patterns. These patterns are then used as search models in a new DNA sequence. If a part of the DNA matches the model, the sequence is extracted as a likely rRNA sequence. The help page for **rnammer** has the following description for the program:

" RNAmmer predicts ribosomal RNA genes in full genome sequences by utilising two levels of Hidden Markov Models: An initial spotter model searches both strands. The spotter model is constructed from highly conserved loci within a structural alignment of known rRNA sequences. Once the spotter model detects an approximate position of a gene, flanking regions are extracted and parsed to the full model which matches the entire gene. By enabling a two-level approach it is avoided to run a full model through an entire genome sequence allowing faster predictions.

RNAmmer consists of two components: A core Perl program, 'core-rnammer', and a wrapper, 'rnammer'. The wrapper sets up the search by writing on or more temporary configuration(s). The wrapper requires the super kingdom of the input sequence (bacterial, archaeal, or eukaryotic) and the molecule type (5/8, 16/17s, and 23/28s) to search for. When the configuration files are written, they are parsed in parallel to individual instances of the core program. Each instance of the core program will in parallel search both strands, so a maximum of 3x2 hmmsearch processes will run simultaneously. The input sequences are read from sequence and must be in Pearson FASTA format. "

The program is run as follows, specifying the taxonomical kingdom (bac) and the type of molecules to search for (tsu for 5/8s rRNA, ssu for 16/18s rRNA, lsu for 23/28s rRNA). The parameter -f specifies the name of the output file.

rnammer -S bac -m ssu -f <name>.rrna.fsa <name>.fna

Wrapping all your genomes in a single for loop for rnammer might, therefore, not a good idea. Make 2-3 loops and wait for each of them to finish before you start the next one.

```
# Loop for running a list of files
for x in <name1> <name2> <name3> <name4>
do
rnammer -S bac -m ssu -f $x.rrna.fna $x.fna
done
```

¹Lagesen, K. et al. (2007). RNAmmer: consistent and rapid annotation of ribosomal RNA genes. Nucleic Acids Research

```
# Loop for running all files in directory with extension *fna
for x in *fna
do
rnammer -S bac -m ssu -f $x.rrna $x
done
```

9 Multiple sequence alignment of selected 16S rRNA sequences

The way of comparing the 16 sRNA sequences is to do a multiple sequence alignment. CMG-biotools comes with the alignment program clustal. The greater the distance between two sequences the greater the difference between the organisms from which the sequences came. The **rnammer** program finds all possible rRNA sequences in a genome. Some, and indeed many, genomes have more than one copy of this gene. In order to do a 16S rRNA tree you should pick one of these sequences. Here we choose the one that has the highest score according to the **rnammer** models. The program **extractseqs** takes all files in a directory name <**something**.**rrna.fsa** and selects the best sequence within each file (each organism). For some organisms, **rnammer** might not find a sequence with a good enough score. This means that the model does not find a sequence that looks sufficiently like a rRNA sequence. These sequences, and organisms, will be excluded from the analysis.

The sequences will now be evaluated based on fixed criteria for a 16S rRNA sequence. These criteria include length and fitness score to the **rnammer** models. This extraction procedure will only work for 16s rRNA sequences that fulfill the criteria.

extractseqs all.rrna # The code works on all files in the current working directory

The output is a FASTA formatted file with rRNA genes in DNA code. Count the number of sequences in this FASTA file (using grep). Try using grep without the -c option and look at what the headers of this FASTA file looks like. Look at the header lines for the selected sequences. The alignment is performed using clustalw and the file holding one rRNA sequence from each organism. Here we use the commandline version of clustal but the graphical version is also installed, clustalx.

clustalw all.rrna

Next we create a distance tree for the alignment, using a bootstrap value of 1000. Wiki about bootstrapping in statistics:

" In statistics, bootstrapping is a computer-based method for assigning measures of accuracy to sample estimates (Efron and Tibshirani 1994). This technique allows estimation of the sample distribution of almost any statistic using only very simple methods (Varian 2005). Generally, it falls in the broader class of resampling methods. Bootstrapping is the practice of estimating properties of an estimator (such as its variance) by measuring those properties when sampling from an approximating distribution. One standard choice for an approximating distribution is the empirical distribution of the observed data. In the case where a set of observations can be assumed to be from an independent and identically distributed population, this can be implemented by constructing a number of resamples of the observed dataset (and of equal size to the observed dataset), each of which is obtained by random sampling with replacement from the original dataset. "

When using bootstrapping, different versions of the distance tree will be constructed and each time a branching point will be recorded. With a bootstrap value of 1000, the number on the tree will indicate how many, out of 1000 trees, have this branching. The closer to 1000 the more sure we are of that branch.

clustalw all.rrna -outputtree=nexus -bootstrap

Output is two different files, *phb and *treb.

9.1 Construct phylogenetic tree from 16S rRNA alignment.

The tree construction is simply a drawing program called njplot. Open the bootstrap tree file *.phb and tick the Display setting by clicking Bootstrap values.

njplot all.phb

Try to change the settings for the tree, rooted/unrooted and so on.

10 Proteomes

Constructing a proteome for a given genome project is often done by the people who have published the genome. It is therefore possible to use the published proteomes or do a local genefinding. How you choose to select your data is up to you. You can use published proteomes where they exist and combine with locally calculated proteomes where published once are not available. You can also choose to run local genefinding on all the data and disregard the published proteomes.

10.1 Extract genes and proteins from GenBank

This procedure extracts translated genes from GenBank files in the case where these have been annotated by the publisher of the genomes. To extract annotated proteins from GenBank files, use the following loop. NOTE, some genome project do not have annotated proteins but might still have DNA sequence.

```
for x in *gbk
do
echo $x
gbkExtract $x
done
```

The output files are automatically generated and are called <name>.gbk.fsa (amino acids) and <name>.gbk.fna (nucleotides). As seen from the examples bellow, some GenBank files have DNA sequences but no proteins. It is also seen that unfinished genomes can also have published annotations.

```
# Counting number of FASTA headers in DNA FASTA files
grep -c ">" *fna
Actinoplanes_sp_SE50_110_ID_CP003170.fna : 1
Mycobacterium_avium_subsp_paratuberculosis_CLIJ361_ID_AFNS00000000.fna : 1146
Mycobacterium_avium_subsp_paratuberculosis_S397_ID_AFIF00000000.fna : 176

# Counting number of FASTA headers in protein FASTA files
grep -c ">" *fsa
Actinoplanes_sp_SE50_110_ID_CP003170.gbk.proteins.fsa : 8247
```

8

9

10

3

8

9 10

12

13

14 15

16

17

18

19

: 0

:

4619

10.2 Genefinding

Now we will run our own genefinding algorithm on the DNA sequence of the genome. This is very often the same that the publishers of the genome has done. The good thing about running one algorithm on all the genomes is that the results will be standardized which is not the case with published annotations.

```
prodigalrunner <name>.fna
for x in *fna
do
prodigalrunner $x
done
```

Output files include:

11 Amino acid and codon usage

These basic statistics for each genome are visualized in a PDF summery with three plots and a text summery. The calculations have been implemented in a program called basicgenomeanalysis and takes gene FASTA files as input. The FASTA file can be obtained from GenBank extraction or genefinding. Output is four PDF files with visual representations of codon and amino acid usage along with base bias in third codon position. The file named *all.pdf is a A4 representation of all the figures combined. Along with the PDF is created a text file with the numbers used to draw the figures (see example bellow).

```
basicgenomeanalysis <name>.orf.fna /usr/bin/gnuplot
# Example run
basicgenomeanalysis Actinoplanes_sp_SE50_110_ID_CP003170_prodigal.orf.fna /usr/bin/gnuplot
#==== Roseplot of Codon Usage finished/attempted
#==== Roseplot of Amino acid usage finished/attempted
#==== Bias in third position plot, Gnuplot
#==== Made pdf from ps files
[1] Wrote 1 pages, 190972 bytes
# Output textfile (first and alst 5 lines)
head -n 5 Actinoplanes_sp_SE50_110_ID_CP003170_prodigal.fna.CodonAaUsage
Actinoplanes_sp_SE50_110_ID_CP003170_prodigal.orf.fna TotalBases: 8189487 PerAT: 28.23 StDevAT: 0.04
codon
      AAA 0.26687 7285
codon
       CAA 0.18463 5040
codon
      GAA 0.75983 20742
codon TAA 0.01960 535
tail -n 5 Actinoplanes_sp_SE50_110_ID_CP003170_prodigal.fna.CodonAaUsage
aa S 4.8983
```

```
    aa T 6.4059
    aa M 1.5833
    aa C 0.7422
    aa P 6.1556
```

> 11 12

> 13 14

g

10 11

12 13

For fast viewing of PDF and PS (post script) files you can either double click the file in the file browser window or use the commandline tool ghostview.

11.1 Comparing amino acid and codon usage

Comparing the amino acid and codon usage between many genomes also usually involves clustering genomes with similar usage. It was therefor found useful to compare these numbers using a so-called heatmap constructed in R (The R Project for Statistical Computing).

First we prepare the data from the ***CodonAaUsage** files and collect them into one file. You can of course choose to only include some of your file if you wish to do a smaller comparison.

```
grep aa *CodonAaUsage > aaUsage.all
grep Total *CodonAaUsage > statistics.all
cut -f2,3,4,5,6,7,8 statistics.all > tmp.all
mv tmp.all statistics.all
grep codon *CodonAaUsage > codonUsage.all
```

Install the gplots package:

```
install.packages("gplots")
```

Drawing heatmaps of bias in third codon position and amino acid and codon usage

```
library(gplots)
codon <- read.table("codonUsage.all")
colnames(codon) <- c( 'Name', 'codon', 'score', 'count')
codon <- codon[1:3]
test <- reshape(codon, idvar="Name", timevar="codon", direction="wide")
codonMatrix <- data.matrix(test[2:length(test)])
rownames(codonMatrix) <- test$Name
# R allows you to run one long command like the following
codon_heatmap <- heatmap.2(codonMatrix, scale="none", main="Codon usage",
xlab="Codon fraction", ylab="Organism", trace="none", margins=c(8, 25)) # Command finished
dev.print(postscript, "codonUsage.ps", width = 25, height=25)
dev.off()
```

```
library(gplots)
aa <- read.table("aaUsage.all")
colnames(aa) <- c( 'Name', 'aa', 'score')
test <- reshape(aa, idvar="Name", timevar="aa", direction="wide")
aaMatrix <- data.matrix(test[2:length(test)])
rownames(aaMatrix) <- test$Name
# R allows you to run one long command like the following
stat_heatmap <- heatmap.2(aaMatrix, scale="none", main="Amino acid usage", xlab="Amino acid fraction",
ylab="Organism", trace="none", margins=c(8, 25), col = cm.colors(256)) # Command finished
dev.print(postscript, "aaUsage.ps", width = 25, height=25)
dev.off()</pre>
```

12 Protein BLAST matrix

A BLAST matrix is a comparison of proteomes (proteins from a genome) used to estimate how many proteins is found in common between two genomes. We will construct a matrix from the protein FASTA files created by prodigal or extracted from GenBank. The BLAST matrix algorithm has been implemented in a program called blastmatrix. First we will construct an input file for this program. The input file must be of the format XML which is a nice computer-reading format but not very friendly to human eyes. A small program called makebmdest construct a XML file from all the protein FASTA files in a directory. Make sure that you have the right files in the current working directory.

```
# Create BLAST matrix XML file from the protein FASTA files within the current working directory
# NOTE: files MUST have the extension *fsa
```

```
makebmdest . > matrix.xml
```

The '.' indicates that the path to the files is the path to the current working directory. If this dot is not included the code will create a XML file with no references to the actual protein files. the BLAST matrix is then generated using the blastmatrix program with the file matrix.xml as input.

blastmatrix matrix.xml > matrix.ps

While the matrix is running you will encounter an error message about a MySQL database. This error is related to the original usage of the script, where a CBS database was searched to see if the calculation had been made before. Because the biotools-xubuntu system does not have a common database the program complains that it can not find it.

Error messages related to illegal division by zero can be due to faulty genefinding (prodigal). In some cases, the genefinder will locate a gene but then afterwards not include that gene for other reasons. This will cause the program to make a header for a gene but no sequence. If you encounter this problem, run the following commands in the directory where you have the amino acid FASTA files.

```
for x in *.proteins.fsa
do saco_convert -I Fasta -0 tab $x > $x.tab
sed -r '/\t\t\t/d' $x.tab > $x.temp.tab
saco_convert -I tab -0 Fasta $x.temp.tab > $x
rm *tab
end
```

13 Pan- and core-genomes

A method of comparing genomes looks at the cumulative set of all genes, shared across genomes (pan-genome) and the conserved set of gene families across all genomes (core-genome). The pan- and core-genomes are theoretical representations of a collective protein pool and a conserved protein pool, respectively. When a protein type is found in all genomes in a collection, it is called a core gene of this collection. Here this is implemented in a pan- and core-genome plot (Figure 7) where sequences are compared using BLAST and the 50/50% cutoff described above. As the clusters grow to more than two members, single linkage clustering is used to assign a new sequence to a group. The program performing this analysis is called **pancoreplot** and

the input is a tab separated text file representing a number of FASTA files containing amino acid sequences. For the first genome, the pan and core are identical, and the core becomes smaller with the addition of a second genome, as genes in this pool now need to be found in both genomes. If a gene from the core is not found in a new genome it is removed from the core, and is then only part of the pan-genome. The pan-genome is the entire gene pool and as such includes the core genome. The order of the genomes can change the course of the graph, but the final shared gene pool (core and pan-genome) will be the same.

13.1 Pan- and core-genomes plot

The algorithm is dependent on BLAST and has been implemented in the program pancoreplot. First we construct an input file for the pancoreplot program, note that is is the number one in the ls command and a pipe character before the gawk command.

ls -1 *.fsa | gawk '{print \$1 "\t" \$1}' > pancore.list

Take a look at the file and note that the look of it (cat, less, head, tail). The first column will become the name in the plot image. You can freely edit this column using the texteditor (mousepad). The second column, separated by tab, is the filename and must NOT be edited. This program runs a series of BLAST comparisons using the blastp program and will take time.

pancoreplot -keep blastOutPut pancore.list > pancoreplot.ps

13.2 Extract subset genes from pan and core genome plot

This procedure outputs the genes/gene families in common or complementary between genomes in a coreplot. The directory argument must be of the type created as temporary directory by the coregenome script. If -o is unspecified (the default), the program will print one representative for each gene family, selected somewhat randomly. Due to speed issues, the program tries to extract the genes from as few genomes as possible. If -o is used, all members of the relevant gene families will be extracted from the genomes specified. Genomes must be specified by their number in the order displayed on the coreplot (because that is how they are named in the coregenome script). Individual values can be separated by commas, while ranges can be specified using ':' or '-'. The final value in a range can be omitted, letting the range terminate at the last genome in the set (or, if -1 was specified, whatever genome was given there). The various options can be specified several times and will be interpreted as a request for several core genomes simultaneously, one from each subset indicated. If neither -u nor -i is specified, the default is to select the intersection (i.e. core gene families) of all genomes up to and including the last genome in the set (or, if -1 was specified, whatever genome was given there).

- -c or -cu or -complementary: If set, output only from gene families which are complementary to (i.e. not present in) the union of these genomes.
- -ci or –compinter: If set, output only from gene families which are complementary to the intersection of these genomes - i.e. not present in the core of these genomes.

- -d or –dispensable: If set, reverses the outputting behavior so that genes formerly to be outputted are discarded and the genes which would normally have been discarded are instead outputted. The option is named for its ability to output the dispensable/auxiliary genes instead of the core genes.
- -i or --intersection: Find intersecting (i.e. common) gene families between these genomes.
 Specifying both -i and -u is valid, but if the sets overlap, then those genes obeying both restrictions will be extracted twice. This is considered a feature and not a bug :-).
- -l or -last: The index of the last organism from the core genome plot to include in the calculation of the core genome. Defaults to the last organism in the plot.
- -o or -organisms: The index of the organisms for which to output the genes. If specified the program outputs all genes from selected gene families and specified organism(s).
 If unspecified, the program will print one representative for each gene family, selected somewhat randomly. Due to speed issues, the program tries to extract the genes from as few genomes as possible.
- -s or -slack: Number of genomes a gene is allowed to be missing from and still be considered part of the core genome. This would normally only be set if the slack option was used with the pancoreplot script when the input directory was created, and then only to the value used there.
- -u or -union Find the union of gene families (i.e. all genes) from these genomes. Specifying both -i and
 -u is valid, but if the sets overlap, then those genes obeying both restrictions will be extracted twice.
 This is considered a feature and not a bug :-).

Examples:

```
ProgName -i 1:3,5:7 [Directory]
   Gives you the core gene families of genomes 1, 2, 3, 5, 6, and 7.
                                                                                                            2
                                                                                                            3
ProgName -i 1: [Directory]
   Gives the core gene families of all genomes in the set. This is actually the default, used if no
    options are given.
                                                                                                            6
ProgName -i 1,3:5 -c 6: [Directory]
   Gives the core gene families of genomes 1, 3, 4 and 5 which are not present in any of the genomes
                                                                                                            8
    from 6 and on to the last.
    In this command, genome 2 is not considered at all.
                                                                                                            g
ProgName -i 1:3 -i 5:7 [Directory]
                                                                                                            11
   Gives the core genome of organisms 1, 2 and 3 as well as the core genome of 5, 6 and 7.
                                                                                                            12
   This is a larger set different from '-i 1:3,5:7' above.
                                                                                                            13
                                                                                                            14
ProgName -u 1:5 -ci 7:9 -ci 8:10 [Directory]
                                                                                                            15
   Gives the part of the pan-genome of organisms 1 through 5 which is neither in the core genome of 7, 8
                                                                                                            16
     and 9 or in the core genome of 8, 9 and 10. Yes, subsets can overlap.
```

13.3 Gene frequency plot from a pan-core genome output

Before performing this analysis you must read section 13.1 on pan- and core-genome construction. Find BLAST output reports from pan-core genome plot and locate the last group file $(group_lastX.dat)$. Run following command in the Terminal:

Syntax

 $\begin{array}{l} gawk \ '\{print \ NF-2\}' \ group_<lastX>.dat \ | \ gawk \ '\{arr\,[\$1]++\} \ END \ \{for\,(i \ in \ arr\,) \ print \ i, arr\,[i]\}' \ | \ sort \ -n > \ frec.txt \\ \# \ Example \\ gawk \ '\{print \ NF-2\}' \ group_3.dat \ | \ gawk \ '\{arr\,[\$1]++\} \ END \ \{for\,(i \ in \ arr\,) \ print \ i, arr\,[i]\}' \ | \ sort \ -n > \ frec.txt \\ \end{array}$

Start R

Read in file:

```
frec <- read.table("/username/path/frec.txt", sep=" ", dec=",")  # Syntax
frec <- read.table("/home/student/Desktop/frec.txt", sep=" ", dec=",")  # Example</pre>
```

Create plot:

```
# R allows you to run one long command like the following
barplot(frec$V2, main="Gene frequency",
xlab="Gene occurrence count, genes can be found multiple times in one genome",
ylab="Frequency of gene count, how often does a gene get counted x times",
ylim=c(0,max(frec$V2)*1.1)) # Command finished
```

Save plot:

```
dev.print(pdf, file = "/username/path/frec.pdf", width = 15, height=8)  # Syntax
dev.print(pdf, file = "/home/student/Desktop/frec.pdf", width = 15, height=8)  # Example
```

14 Tips and Tricks

14.1 Remove FASTA entries with no sequence

```
for x in *.proteins.fsa
do saco_convert -I Fasta -O tab \texttt{x > \texttt{x.tab
sed -r '/\t\t\t/d' \texttt{x.tab > \texttt{x.temp.tab
saco_convert -I tab -O Fasta \texttt{x.temp.tab > \texttt{x
rm *tab
end
```

14.2 Modify names in BLAST matrix *.ps file

14.2.1 HASH lines

If the matrix has a line for each organism saying something like

```
(HASH\(0x954dba8\)) -45 rotate dup stringwidth pop neg 0 rmoveto show 45 rotate
```

A Post script file is actually a text file that gets interpreted as a picture. You can open the *.ps file in the text editor an see what it looks like. To remove the HASH lines from the picture, we must remove a part of the text in the file. The entire block must be deleted for each genome for the lines to disappear.

```
183.497474683058 ux 111.639610306789 uy moveto
(HASH\(0x954dba8\)) -45 rotate dup stringwidth pop neg 0 rmoveto show 45 rotate
newpath
```

Fortunately, one commandline can do this for us. This can be done for all entries in the ***.ps** file as follows:

2

awk '/HASH/{c=1;next} c--<0 && p{print p} {p=\$0} END{print p} ' test.ps > new.ps

14.2.2 Bounding box

If the names of the organisms reach outside the border of the picture, change the numbers in the BOUNDING BOX field of the *.ps file. Each of these numbers correspond to different points in a coordinate system:

%%BoundingBox: 0 0 2212 1110 %%BoundingBox: llx lly urx ury

The numbers correspond to the size of the picture, and are coordinates llx lly urx ury (lower left corner (x,y), upper right corner (x,y)). Change the coordinates, save the *.ps file and click the file again.

2

3

2

3

4

14.3 Convert *.ps file to *.pdf

To use your figures in a paper or presentation, it is handy to have the picture as a *.pdf. For this you can use the program ps2pdf. The syntax is simple:

ps2pdf -dEPSCrop file.ps file.pdf

You might experience that the *.pdf only contains part of the figure. If this is the case, right click the *.ps file and open it in mousepad. Find a line that looks something like %%BoundingBox: 0 0 2212 1110. The numbers correspond to the size of the picture, and are coordinates llx lly urx ury (lower left corner (x,y), upper right corner (x,y)). Change the coordinates, save the *.ps file and run the ps2pdf again. It takes a couple of tries the first time, but then it is easy enough to set the size of the picture.

14.4 Rename files

Files with wrong ending can be renamed the following type of loop:

```
for x in *gbk.fna
do
newx=`echo $x | sed "s/.gbk.fna/.fna/"`
mv $x $newx
done
```

Or another example:

```
for x in *gbk.proteins.fsa
do
newx=`echo $x | sed "s/.gbk.proteins.fsa/.proteins.fsa/"`
mv $x $newx
done
```

14.5 Delete empty files

Always list the files before using the removing command so you do not delete fields unintentionally.

```
find . -type f -empty # Display empty files
find . -type f -empty -exec rm {} \; # Remove empty files
```