Shanghai Jiao Tong University

Final year project

Bioinformatics Computer Lab Manual

Instructor:
Maoying Woo

November 20, 2009

Contents

Section I

High Performance Computing

1

Linux Cluster Construction

1.1 Introduction

Recently high performance computing (HPC) has become more and more prevalent in the field of bioinformatics, partly due to the adoption of open source software concepts and the introduction and refinement of clustering technology. Though the price of supercomputers is decreasing at a exponential speed, they are unaffordable for most of the users. However, with the help of high-speed switches, we can easily construct our own computer "cluster".

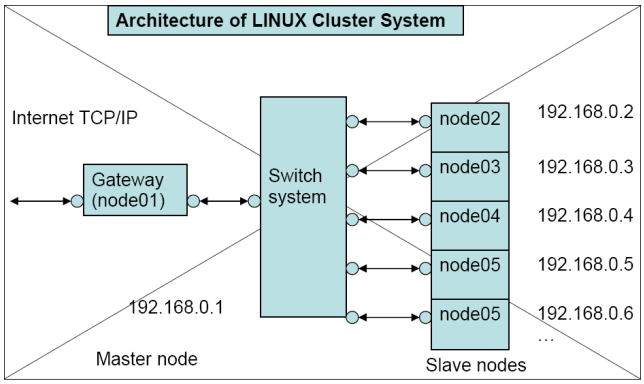

Figure 1.1: Architecture design of LINUX cluster

The basic requirement for building a LINUX cluster is addressed as follows. First, you need to collect multiple PCs and select one as the master node and others as slave nodes. The master serves as the gateway to Internet. It requires at least two

Ethernet network cards, with one serving as the gateway for outer connection, and the other as a medium for the inter-connection to other slave nodes, through a high-speed switch. Every slave node should have at least one Ethernet card, which connects the node to the switch.

The switch is utilized to group all computer nodes into a cluster so that the nodes can communicate with each other during parallel computing. Basically any fast-speed switch or hub can be served as an internal switch for a LINUX cluster.

The architecture of our simple cluster system is illustrated in figure 1.1.

1.2 Configuration Procedures

1.2.1 Install Linux and Configure Networks

At first, you can select any Linux operating system of your preference. However, we have finished the installation for you all. So what you need to do is to configure the network and also the NFS on the master node. Remember that on the master node the partition "*/cluster*" will be shared with other slave nodes via NFS. And the partition "*/cluster*" will be used to install MPICH, the Message Passing Utility software.

Now it is time for you all to configure the network for our Linux cluster:

```
1 ### For master node:
2 IP Address: 192.168.2.100
3 Host Name:  bioserv
4 ### For slave nodes:
5 IP Address: 192.168.2.X
6 Host Name:  bio0X
```

To configure the Ethernet card for the nodes, you should select the default option "*Activate on boot*".

The next step is to configure firewall security. On the master node we choose the default security level "*medium*". For the trusted device "*eth0*", we only selected the item "*SSH*".

Login on console window, you can use *vi*, or *emacs*, or any other editor to modify several configuration files.

(1) Edit the file "**/etc/hosts**" on each node. Make sure you are logging as the root user.

```
1 127.0.0.1             localhost
2 192.168.2.100         bioserv
3 192.168.2.2           bio02
4 192.168.2.3           bio03
5 192.168.2.4           bio04
6 ...                   ...
```

(2) After the above edition is completed, you can enter the command "**service network restart**" to restart the network system.

(3) In order to test whether you network work or not, enter the command "**ping $hostname**".

1.2.2 NFS Server Setup on the Master Node

After configuring the network system of your cluster correctly , you can set up the NFS. Let's show you how we set up NFS Server on our master node. The objective is to configure the master node so that it allows the slave nodes to share its file system.

(1) Log on the master node as a root user. Activate the following daemons: **"network", "nfs", "nfslock", "portmap", "rsh", "rlogin", "sshd"** and **"xinetd"**.

(2) Edit the "**/etc/exports**" to specify the file systems (to be shared), hosts (to be allowed) and the type of permissions (ro or rw). In this lab, our '**exports**' file has the following entry:

```
/home    192.168.2.1/24(rw, no_root_squash)
```

(3) Add a new user on the master node, then restart the NFS service in order to export the shared directory using the command:

```
# exportfs -a
# service nfs reload
```

1.2.3 NFS client Setup on the Slave Nodes

Similarly, we set up the NFS Client on each slave node. The objective is to enable each slave node to mount the file sharing system on the master node.

(1) Create the directory "**/home**" on each node. This directory serves as a mount point of the shared file system on the master node.

(2) Log on each slave node as root user. Activate the following daemons: "**network**", "**nfs**", "**portmap**", "**rsh**", "**rlogin**", "**sshd**", "**xinetd**".

(3) Edit the file "**/etc/fstab**". We need to append an extra line at the end of the file. This line tells the **/home** directory of the master node exactly mounts (links) to the one of current slave node. For example, we have the following line:

```
bioserv:/home    /home    nfs
```

This approach enables slave nodes to automatically and statistically mount the shared file system after rebooting your machines.

(4) Use command "**df**" to check if the NFS configuration is successful or not.

(5) Add a new user on each slave node with **username** consistent with the one on the master node, using the command "**adduser <username>**".

(6) Reboot the slave nodes.

1.2.4 Install Parallel Library MPI on Master Node

(1) Log on the master node as root user.

(2) Download the MPI parallel library and then compile and install on the master node. Note that we recommend that you install MPI in the directory "**/opt/cluster**".

(3) After generating the binary codes, it is time for you to configure the MPI environment. You can go to the directory **/opt/cluster/mpich-X.X.X/util/machines**. Edit the file **machines.LINUX** to specify the computer nodes, which are desirable to join the MPI empowered parallel computing in the cluster system. In this lab, we can use these entries in the file:

```
1 bioserv
2 bio02
3 bio03
4 bio04
5 ...
```

(4) Finally, you need to edit the file " **/.bash_profile**" to add the MPI to the environment variable "**$PATH**":

```
PATH=$PATH:$HOME/bin:/opt/cluster/mpich-X.X.X/bin\
        :/opt/cluster/mpich-X.X.X/util
LD_LIBRARY_PATH=/opt/cluster/mpich-X.X.X/lib
export PATH LD_LIBRARY_PATH
```

1.2.5 Test MPI Program

After installing MPI, you should now be able to test your parallel code enhanced by MPI library.

```
$ mpicc -o MPIcode MPIcode.c
$ mpirun -np 2 MPIcode
```

Congratulation! A simple but useful Linux cluster is now in your hand.

1.3 Install and Configure OpenPBS

OpenPBS is the original version of the Portable Batch System, a queuing system developed for NASA in the early 1990s.

1.3.1 Pre-install Configuration

(1) Configure RSH

rsh needs to be configured to allow passwordless communication between the master node and the slave nodes. This is done by creating **/etc/hosts.equiv** on the master and slave nodes.

```
1 // Head node /etc/hosts.equiv
2 ### a list of all of the slave nodes
3 bio02
4 bio03
```

```
5  ...
6  bioXX
7
8  // Slave node /etc/hosts.equiv
9  #### the master node
10 bioserv
```

(2) Enable rsh, rlogin, and rexec

Change the "disable=yes" to "disable=no" in each of their respective xinetd scripts located in /etc/xinetd.d and then run command "service xinetd reload" to restart the xinetd.

(3) Test rsh

From the master node, use rsh to login to a slave node as a non-root user using command "rsh bioXX"; from each slave node, use rsh to login to the master node as a non-root user; if a password prompt appears in either case, rsh is not configured correctly.

(4) Download OpenPBS

We recommend that you download the source package.

1.3.2 Installation

(1) Untar OpenPBS

Note that you should unpack OpenPBS in the directory "/home/cluster" so that it can be accessed from both the master node and slave node.

(2) Compile OpenPBS

Note that OpenPBS should be compiled twice, one for master node, and the other for slave nodes.

A. **Master Node**:

```
# mkdir /home/cluster/OpenPBS_x_x_x/
  master
# cd /home/cluster/OpenPBS_x_x_x/master
# ../configure --disable-gui --set-
  server-home=/var/spool/PBS \
--set-default-server=bioserv
# make
# make install
```

B. *This disables the GUI and sets the installation directory to /var/spool/PBS instead of /usr/spool/PBS. This also sets the default PBS server to bioserv, which should be the hostname of the master node. The source is then compiled and the binaries are then installed.

C. **Slave Nodes**:

```
# mkdir /home/cluster/OpenPBS_x_x_x/
  slave
# cd /home/cluster/OpenPBS_x_x_x/slave
# ../configure --disable-gui --set-
  server-home=/var/spool/PBS \
   --disable-server --set-default-server
     =bioserv --set-sched=no
# make
# rsh bioXX `cd /home/cluster/
  OpenPBS_x_x_x/slave; make install'
```

D. *After the head node is successfully installed, the configuration script is run again in a separate directory for the slave nodes. This disables the GUI and sets the installation directory to /var/spool/PBS instead of usr/spool/PBS. This also sets the default PBS server to bioserv, which should be the hostname of the head node. It also disables the server and scheduling processes of PBS on the slave nodes, since they are not necessary. The source is then compiled and installed using rsh.

(3) Install Document

```
# cd /home/cluster/OpenPBS_x_x_x/master/doc
# make install
```

1.3.3 Configure OpenPBS

(1) Create PBS node description file

On the master node, create the file /var/spool/PBS/server_priv/nodes.

```
1  # a list of all of the slave nodes
2  bioserv    np=1
3  bio02      np=1
4  ...        ...
5  bioXX      np=1
```

(2) Configure PBS mom

On the master node and on each slave node, create the file /var/spool/PBS/mom_priv/config.

```
1   #/var/spool/PBS/mom_priv/config
2   $logevent 0x0ff
3   $clienthost bioserv
```

```
4    $restricted bioserv
```

This causes all messages except for debugging messages to be logged and sets the primary server to bioserv. It also allows bioserv to monitor OpenPBS. On the master node and on each slave node, start PBS mom with command "pbs_mom".

(3) Configure PBS Server

On the master node, run the two commands:

```
# /usr/local/sbin/pbs_server -t create
# qmgr
```

In the "*qmgr*" console, enter the following commands:

```
1  > c q drug
2  > s q drug queue_type=execution
3  > s q drug enabled=true
4  > s q drug started=true
5  > s s default_queue=drug
6  > s s scheduling=true
7  > s s query_other_jobs=true
8  > s s node_pack=false
9  > s s log_events=511
10 > s s scheduler_iteration=600
11 > s s resources_default.neednodes=1
12 > s s resources_default.nodect=1
13 > s s resources_default.nodes=1
14 > quit
```

This creates an execution queue called drug that is enabled and started. It is then declared the default queue for the server. Logging and scheduling are enabled on the server and node_pack is set to false. The default number of nodes is set to 1.

It is useful to backup the PBS server configuration file with

```
# qmgr -c "print server" > /var/spool/PBS/qmgr.conf
```

so that the PBS server could be restored with

```
# qmgr < /var/spool/PBS/qmgr.conf
```

(4) Start PBS Scheduler

On the master node, PBS scheduling needs to be started after the server configuration is complete. This is done by

```
#/usr/local/sbin/pbs_sched
```

(5) Enable PBS on startup

On the master node and on each slave node, create the script /etc/init.d/pbs. Then set PBS to automatically restart when the computer boots

```
# chkconfig pbs on
```

(6) Restart PBS

On the master node and on each slave node, manually restart OpenPBS for all the configuration changes to take effect using the command:

```
#/sbin/service pbs restart
```

1.3.4 Testing PBS

After installing and configuring OpenPBS, testing should be done to verify that everything is working properly. In order to do this, a simple test script is created:

```
1 #!/bin/sh
2 #testpbs
3 echo This is a test
4 echo Today is `date`
5 echo This is `hostname`
6 echo The current working directory is `pwd`
7 ls -alF /home
8 uptime
```

and then submitted to PBS using the qsub command as a non-root user:

```
$ qsub testpbs
```

After the job is executed, the output is stored in the directory from which the job was submitted. If errors occur or output is not received, check the log files for messages about the job (especially the precise name used by PBS for the master node in the server_logs).

```
# more /var/spool/PBS/*_logs/*
```

Section II

Sequence Analysis

2

BioPerl Progamming

2.1 Bioperl Introduction

Bioperl is a collection of modules to carry out bioinformatics tasks, which can be accessed in the wiki `http://www.bioperl.org`. These modules are especially useful for sequence analysis.

2.2 Programming Exercises

EX2-1 Needleman-Wunsch algorithm and aligning small subunit ribosomal RNAs from different organisms.

This exercise is an implementation of the standard Needleman-Wunsch algorithm and application of the algorithm to the alignment of ribosomal RNAs from different organism.

The standard Needleman-Wunsch algorithm is described in the lecture and the rRNA sequences are provided below in the dataset section. The small subunit ribosomal RNAs are a family of well-conserved RNAs across different organisms. Use your algorithm to test how similar they are, even after millions of years of evolution.

Dataset(24 sequences total):
`http://202.120.45.17/course/final/lab2/sequences.tar.gz`

Input and Output Format:
You can hardcode your substitution matrix and gap penalty values (-3 for nucleotide mismatch, 1 for a match, -2 for a gap; ignore the affine gaps). The command line for calling your program should be of the form: *./program_name seq1.fasta seq2.fasta*. Output should be both the alignment score for this pair of sequences and the actual alignment itself printed with gaps. **Treat any special characters (such as R and N) the same as AUCG, i.e. use the same match and mismatch cost**.

EX2-2 Build a web server using your program with perl-CGI so that y ou can upload files to the server and run standard Needleman-Wunsch. Results will return to the web page.

EX2-3 Write a script to parse the BLAST result using Bio::Tools::BPlite; Extract the following columns and saved in a text file, the delimiter is a tab "";

query, score, bits, percent, P, EXP, match, postive,length, seq_id

EX2-4 For a set of sequences stored in a single file, do pairwise blast of these sequences and store the E-values in a table. Based on this matrix, build a UPGMA tree.

If you have any question with perl/bioperl, please refer to `http://202.120.45.17/course/bioperl/`

3

Computational Motif Discovery

3.1 Objective

This lab is to guide you through the comparison of existing most popular algorithms for computational motif finding.

3.2 Introduction

Transcription factors regulate the gene expression by activating or inhibiting the transcription machinery. Understanding the mechanisms that regulate gene expression is a major challenge in molecular biology. Thus, Identifying regulatory elements, especially the binding site in genomic sequence for transcription factors becomes an essential task. In essence, this is a pattern discovery problem. This problem can be simplified as follows:

> Given a set of DNA sequences, find an unknown pattern that occurs frequently.

A DNA motif is defined as a nucleic acid sequence pattern that has some biological significance. Normally, the pattern is fairly short (5 to

20 bp long) and is known to recur in different genes or several times within a gene.

Algorithms to find regulatory elements can be divided into two groups: (1) methods based on word counting and (2) methods based on probabilistic sequence models.

The word counting methods analyze the frequency of oligonucleotides in the upstream region and use intelligent strategies to speed up counting and to detect significantly over-represented motifs. These methods then compile a common motif by grouping similar words. Word counting methods lead to a global solution as compared to the probabilistic methods and are appropriate for short motifs and are therefore useful for motif finding in eukaryotic genome where motifs are generally shorter than prokaryotes. These algorithms are a good choice for finding totally constrained motifs, i.e., all instances are identical. However, for typical TF motifs that often have several weakly conserved positions, word-based methods can be problematic and results often needs to be post-processed. Word-based methods also suffer from the problem of producing too many spurious motifs.

In probabilistic approach, the model parameters are often estimated using ML principle or Bayesian inference. Many of the algorithms developed from the probabilistic approach are designed to find longer or more general motifs than are required for transcription factor binding sites. Therefore, they are more suitable for prokaryotic motif discovery. However, these algorithms do not guarantee to find global optima, since some local search methods are employed, such as Gibbs sampling, EM or greedy algorithms that may converge to a local optima.

Table (3.1) listed the current known motif finding algorithms:

EX3-1 Implement the motif finding algorithm Naïve_CONSENSUS.

The detailed description of Naïve_CONSENSUS and the required input and output for your program. Print out the output of your program as the written answer. The three input data files required

Table 3.1: **Some motif discovery algorithms**

Algorithm	Operating principles	Classification	Reference
by Gala *et al.*	Enumeration	Word-based	Galas *et al.*
by Mengeritsky	Enumeration	Word-based	Mengritsky
by Staden	Enumeration	Word-based	Staden
EM	EM	PSM	Lawrence and Reilly
WordUP	Enumeration	Word-based	Pesole *et al.*
Gibbs sampler	Gibbs sampling	PSM	Lawrence *et al.*
MACAW	Gibbs sampling	PSM	Liu
MEME	EM	PSM	Bailey and Elkan
AlignACE	Gibbs sampling	PSM	Roth *et al.*
Oligo-Analysis	Enumeration	Word-based	van Helden *et al.*
Consensus	Weight matrix	PSM	Hertz and Stormo
Dyad-Analysis	Enumeration	Word-based	van Helden *et al.*
WINNOWER	Graph	Other	Pevzner and Sze
ANN-Spec	Gibbs sampling	PSM	Workman and Stormo
SMILE	Suffix tree	Word-based	Marsan and Sagot
Verbumculus	Suffix tree	Word-based	Apostolico *et al.*
MobyDick	Dictionary	Word-based	Bussemaker *et al.*
YMF	Enumeration	Word-based	Sinha and Tompa
Bioprospector	Gibbs sampling	PSM	Liu *et al.*
Co-Bind	Gibbs sampling	PSM	Thakurta and Stormo
ITB	Enumeration	Word-based	Kielbasa *et al.*
Weeder	Enumeration	Word-based	Pavesi *et al.*
MotifSampler	Gibbs sampling	PSM	Thijs *et al.*
MITRA	Prefix tree/Graph	Word-based	Eskin and Pevzne
MDScan	Greedy algorithm	other	Liu *et al.*
Projection	Hashing	Other	Buhler and Tompa
Footprinter	DP	Other	Blanchette and Tompa
PhyloGibbs	Gibbs sampling	PSM	Siddharthan *et al.*
GIMF	EM	PSM	Qi *et al.*
WordSpy	Dictionary	Word-based	Wang *et al.*
MaMF	Enumeration	Word-based	Hon and Jain
EMD	Clustering-based	Other	Hu *et al.*
GibbsST	Gibbs sampling	PSM	Shida
MUSA	Biclustering	Other	Mendes *et al.*
GAME	Genetic algorithm	Other	Wei and Jensen
ALSE	EM	PSM	Leung and Chin
MotifSeeker	Data fusion and ranking	Other	Peng *et al.*
PhyloScan	Scanning	PF	Carmack *et al.*
PhyME	EM	PSM	Sinha *et al.*
OrthoMEME	EM	PSM	Prakash *et al.*
LOGOS	EM	PSM	Xing *et al.*
EC	GA	Other	Fogel *et al.*
GLAM	Gibbs sampling	PSM	Frith *et al.*

for your program are:
`http://202.120.45.17/course/final/lab3/bicoid.fa`
`http://202.120.45.17/course/final/lab3/hunchback.fa`
`http://202.120.45.17/course/final/lab3/kruppel.fa`

This algorithm finds a position weight matrix (PWM) motif in an input of DNA sequence.

1. The input is:
   - a set of sequences $S = S_1, S_2, ..., S_N$ in "FASTA" format.
   - disired motif length k.
   - Number of motifs to report, T.
2. The output is:
   - a set of "motifs" $M = m_1, m_2, ..., m_T$. Each motif m_i is a multi-set of k-mers, one taken from each input sequence S_i.
3. The score of a motif is the information content of the PWM formed by the k-mers in the motif.
4. The algorithm Naïve_CONSENSUS works as follows:

   Consider any arbitrary ordering $S_1, S_2, ..., S_N$ of the sequences in set S, e.g., the order in which the sequence were included in the input FASTA file.

   Iteration 1: For every pair of k-mer x_1 (substring of S_1), and x_2 (substring of S_2), compute the information content of the corresponding PWM. Store the T highest scoring pairs.

   Iteration t: For every k-mer x_{t+1} (substring of S_{t+1}), add x_{t+1} to m to obtain a new motif, and compute the information content of the corresponding PWM. Store the T highest scoring new motifs from this iteration.

   The algorithm terminates when each $S_i \in S$ has been handled, i.e., after iteration $(N - 1)$.

3.3 Exercise

Problem A Write pseudo-code for the above Naive-Consensus algorithm.

Problem B Implement a function that creates a PWM out of a given motif, and computes its information content. The formula for information content $I(W)$ of a PWM is:

$$I(W) = \sum_{k} \sum_{\beta \in A,C,G,T} W_{\beta k} log \frac{W_{\beta k}}{q_\beta} \tag{3.1}$$

where $W_{\beta k}$ is the entry for base β in column k of W. The background probabilities of the bases are denoted by q_β. Assume a background probability distribution $A = T = 0.35, C = G = 0.15$. If $W_{\beta k}$ is 0, assume $W_{\beta k} log W_{\beta k} = 0$.

Problem C Implement the algorithm Naïve_CONSENSUS, as described above. The input sequences are given in FASTA format. The input k (motif length) and T (number of motifs to report) are provided in command-line.

Problem D Run your program on each of the three provided files in FASTA format, with $k = 9$.

EX3-2 Run another free motif discovery program with the same data set as input, and compare the result with yours.

EX3-3 Do you think there exist some defects in the above approach? If you are asked to improve the algorithm, what would you do?

4

Phylogenetic Reinference

4.1 Introduction

The main task for phylogenetic research is to reconstruct evolutionary history. Some clues are provided by fossils, but unfortunately it is often the case that such evidence is absent, and in these cases conclusions must be drawn using only the features that are observable in extant organisms. The only way to go about in this situation is to use similarities/distances between organisms.

Before the boost of molecular data, phylogenetic analysis was usually depending upon visible physical features. Nowadays the starting point of the analysis is mainly sequence information - either DNA/RNA or protein sequences.

Several approaches have been introduced on the class. And during this lab you will continue carrying out phylogenetic studies from either the distance-based methods, parsimony methods, or likelihood methods.

4.2 Distance-based approaches

Both UPGMA and neighbor-joining methods take as the input a distance matrix, in which all pairwise distance between genes/species are listed. A *greedy algorithm* is then used to pairwise join the two closest neighbors until the result is a binary tree. No model of evolution is needed.

4.2.1 Programming exercise 1

Write a program (preferably in c++) to execute the UPGMA approach; Remember that you should output the matrix for each step; but you are not required to output the tree. (**Hint**: have a look of the input file and you will understand that how the input data is called ultrametric.)

Input file: `http://202.120.45.17/course/final/lab4/dist1.txt`.

4.2.2 Programming exercise 2

Write a program (preferably in c++) to carry out the NJ method; Remember that you are asked to output only the matrix for each step. (**Hint**: it is important to determine the pair of sequences with minimum distance.)

Input file: `http://202.120.45.17/course/final/lab4/dist2.txt`.

4.2.3 Answer the following questions:

- What do you believe are the main differences between UPGMA and Neighbor-Joining?

- What are the pitfalls of the distance-based methods when compared to ML method?

4.3 Minimum evolution approach

The idea behind *maximum parsimony* is to construct a tree in which the sum of all mutations along all branches is minimized. No model of sequence evolution is needed.

4.4 Likelihood approach

Maximum likelihood is a very robust approach. It tries to find the tree for which the likelihood is maximized, among all possible tree topologies and parameters. A model of sequence evolution is needed. Due to the vast number of available trees, exhaustive approach is not feasible even for a modes number of species. Hence the very search is usually done using a hill-climbing algorithm, always looking for changes to the tree that will increase the likelihood. The search is often started using a tree computed by other simpler methods, such as NJ.

4.5 Bootstrapping

All above methods will give as a result a single phylogenetic tree (the best estimate). Normally you want a measure of confidence too. How confident are we that a certain group of taxa belong to the same branch?

In order to establish such confidence limits *bootstrapping* is used. In bootstrapping a number of pseudoreplicates are constructed by random sampling with replacement,from our aligned sequences. The pseudoreplicates are of the same length as the original aligned sequences.

The same tree-building method as earlier is then used on these pseudoreplicates, and from the set of trees confidence limits can be computed.

4.5.1 Programming exercise

Write a program to carry out the bootstrap resampling from the alignment in the following file.

Input file: `http://202.120.45.17/course/final/lab4/boot.txt`.

4.6 Using Mega for phylogenetic inference

In this section the exercises are built upon real science, and you can download the corresponding scientific papers. If you have time and interest, you can have a look of the figures and conclusions/discussions.

4.6.1 General information

Before you start the exercises, you need to download and install MEGA4.1 program on your computer. If you are working with linux, download and install the RPM version.

MEGA can either open MEGA-formatted files with extension .meg, or it can open FASTA format files in the alignment explorer window. You can take advantage of ClustalW to align the sequences in the FASTA file directly in the explorer.

4.6.2 MEGA exercise 1

This exercise is based upon a research article by Fredsted et.al.

During sequencing of mitochondrial DNA from 385 Danish hares(*Lepus europaeus*), we discovered a weird haplotype (counting 16 hares from Sealand with this very divergent haplotype). A BLAST search showed that it looked like a different species, namely the snow hare (*Lepus timidus*).

The author have compiled sequences from the two species from all over Europe and all the Danish hare haplotype. (The Danish hares are labeled "h#_number_of_individuals", i.e. "h2_15" means haplotype number 2, 15 individuals.)

Retrieve and align sequences

- Download the "hare_haplotype.fasta"to your hard disk;
- Launch MEGA;
- Choose Alignment - Alignment Explorer, and 'retrieve sequences from a file' and choose the sequence file;
- The sequences are unaligned, so align them using default parameters (Alignment - clustalw).

Crop the alignment to remove the gap filled ends of the alignment

- Select unwanted columns and delete the columns. Do both ends of the alignment;
- Now export the alignment to MEGA format and save it with a sensible name (*Data* - *Export*). **REMEMBER NOT** choose "Protein coding".

Do a phylogenetic analysis of the sequences

- Have a look at the alignment that "*pops up*". You can choose to

color alignment columns that are **Constant**, **Variable**, **Parsimony informative (Pi)**, and **Singleton (S)**;

- What is the difference between **Pi** and **S**? Do you think $Pi \subseteq V$?
- Now choose *Phylogeny - Construct Phylogeny*
  - Do a phylogenetic analysis using **NJ** and **UPGMA**;
  - Are there any differences in the resulting trees?
  - Which substitution model did you choose?
  - What is the difference between Jukes-Cantor and Kimura substitution model?
  - Which model is most realistic for this data set? Repeat the analysis but with "test of phylogeny" using 500 bootstrap replicates. Have a look at the resulting tree.
- Are the nodes of the current tree well supported by the bootstrap values?

Finally answer the biological relevant questions

- Are there really snow hare DNA in the Danish hares?
- How sure are you about this?
- Where do you think it came from?
- How sure are you about this?

When looking at a phylogeny in tree explorer try using the tool buttons to the left, you can specify the root (if you have an outgroup), you can rotate subtrees and so on. Try it out.

5

Phylogenetic Reinference using PAUP4.0

5.1 Introduction

Download the PAUP4 and large_angio.nex data set.

5.2 Command line in PAUP4.0

The first step when using PAUP is to load your data into the program. To do so, you can use the menu of the graphical interface, and select the "execute" menu.

The commands used by PAUP have to be entered through a command line. Here is a list of commands that you will need for this practice. Do not use them right now, they are given here for reference only. For the full list of commands, refer to the PAUP manual or type "help" in the command line.

```
1 set criterion=parsimony;
2 bandb;
3 hsearch start=stepwise addseq=simple nreps=1
     nchuck=1 chuckscore=1 swap=none multrees=
```

```
  yes steepest=no;
savetrees file=mytree.tre format=altnex
  brlens=no;
bootstrap nreps=100 search=heuristic
  treefile=bootstrap.tre format=phylip/
  start=stepwise addseq=random nreps=10 swap
  =nn1;
```

A quick explanation on the commands:

set set the optimality criterion to parsimony
possible options: parsimony, likelihood, distance.

lset set the maximum likelihood model

nset number of free parameters in the model

tratio sets the tratio value if nst=2

rmatrix sets the rmatrix if nst=6

rates how to account for rate heterogeneity among sites

shape sets the gamma shape value

bandb performs a branch and bound search

hsearch performs a heuristic search

start on which tree to start the swapping step of the search
possible options: stepwise (need the addseq option below), nj, current (need a tree in memory)

addseq type of taxa addition sequence
possible options: simple, closest, asis, random, furthest

nreps number of addition sequence to try, useful only with random

nchuck, chuckscore no more than the number of trees specified by the nchuck option of score greater than or equal to the score specified by the chuckscore option will be retained in a search (or in a random-addition-sequence replicate)

swap type of swapping algorithm
possible options: none, nni, spr, tbr

steepest allow suboptimal trees to be kept in memory during swapping (yes or no, usually no)

savetrees save the best tree(s) found in a file

file name of the file to use

format format of the tree
possible options: nexus (include a translation table), altnexus (without translation table), phylip, hennig

brlens save tree(s) with branch lengths (yes or no)

bootstrap performs a bootstrap analysis, the options for the search during the bootstrap are indicated after the "/"

nreps number of replicates of bootstrap (not the same as the "nreps" in "hsearch")

search type of searches to do on each bootstrap replicates
possible options: heuristic, bandb, nj, faststep

treefile file where to save all the bootstrap trees

format the format of these trees

contree build a consensus tree from all trees in computer's memory using the type of consensus given in the options (strict, semistrict and/or majority rule), then save it in the file given after "treefile"

5.3 Branch and bound

1. After having executed the data file, either `bandb;` in the command line
2. Then save the trees in the file of your choice with thecommand `savetrees;`
3. Explain the type of search just done, how many trees were found? What were their lengths?
4. Open the tree saved in the program `Mega`
5. Root the tree using *Welwitschia, Pinus, Gnetum, Gingko*

5.4 Heuristic search

1. Do a heuristic search, command `hsearch`, with one replicate of simple stepwise addition sequence, no swapping, keeping only one tree during the search
2. Then save the tree(s) in the file of your choice with the command `savetrees`
3. Explain the type of search just done, how many trees were found, what were their lengths
4. Open the tree saved in the program Mega
5. Repeat the search by changing the `swap` options to `nni, spr` and `tbr` respectively, saving the trees from each search in different files.
6. Explain the difference between these searches. Do they take the same time to complete? Do you obtain the same number of trees at the end? Are the lengths of the best trees the same for each search?

7. Repeat once again the searches with `nni` and `tbr`, but this time change the following options (saving the trees from each search in different files too):
   - `addseq=random`
   - `nreps=100`
   - `nchuck=1000`
   - `chuckscore=1`
8. Explain the differences between these two searches and the previous ones. Do you get the same tree?
9. Take the best trees found by all these strategies and root it in `TreeView`. Compare it with the one found by branch and bound. Did you find the same tree? Why?
10. Describe the evolutionary history of the organism in your data set.

5.5 bootstrap

So far we donąŕt know which tree obtained is the best estimate of the evolutionary history of our taxa. To get an assessment of how much confidence we can have in a tree, we need to use resampling techniques like the bootstrap.

1. Do a boostrap, command `bootstrap`, with 100 resampling, a heuristic search done on each resampled matrix, each with 10 random stepwise addition sequence with NNI swapping. Save the bootstrap trees in a file in phylip format
2. Open the bootstrap trees in Mega and compute the consensus
3. Repeat the analysis by doing 1000 bootstrap replicates instead of 100

4. Repeat the analysis by changing the following options:
   - `swap=tbr`
   - `nreps=100` (for the heuristic search,not for bootstrap)
5. Do you observe a change in the bootstrap percentage obtained? Is it useful to perform thorough heuristic searches for each bootstrap resampling?

5.6 Model comparison

The usually used criteria for selecting the best fitting model are AIC, BIC and LRT. Here we will use them to test which one of the following models is best fitting the angiosperm data set:

- JC69
- JC69 + Γ
- HKY85
- GTR
- HKY85 + Γ
- GTR + Γ

In order to use either the AIC or LRT, we need to have the same topology for each model (otherwise the number of parameters will be different between topologies, and it is impossible to quantify how many of them are different). Therefore

1. Use PAUP (maximum likelihood) with a good enough model to build a topology.

2. Use the fixed topology to estimate the likelihood (and parameter estimates) for each model
3. Calculate, by hand, the AIC for each model and the LRT for all possible nested models
4. Which model is best fitting the data set? Is it the same for AIC and LRT? If not, which one would you trust more?

Section III

Population Genetics

6

Genetic Association Studies

Objective

In this lab you will learn more aspects in statistical genetics, and grasp the technology of association analysis. Genome-wide approaches will be introduced, however, you may not spend much time in this topic.

6.1 Association Analysis

Genetic linkage is the tendency of short chromosomal segments to be inherited intact from parents to offsprings. As a result, some combinations of alleles, i.e. haplotypes, on these short segments may be preserved over a large number of generations. This co-segregation of alleles is more pronounced the shorter the genetics distance is between the corresponding loci. The excessive co-occurrence of certain haplotypes, because of tight linkage or for other reasons, is known as *allelic association*. **Linkage analysis** can be used to perform a genome-wide search for the existence of trait loci using a relatively small number of markers. **Association analysis**, on the other hand, is often used in an attempt to confirm the involvement of a suspected allele thought to be of importance for a trait of interest, or of an associated allele at

a closely linked locus.

6.2 Case-control Association Analysis

The simplest and oldest association analysis method is the **case-control study**. Two random samples are collected, one of persons with a particular disease (**case**), the other of persons without that disease (**control**). We can then test for whether a particular marker allele is more common among the case group than the control group. If the disease arose as a mutation (e.g. SNP) on a chromosome bearing that particular marker allele, then a (statistically) significant association could be due to linkage disequilibrium between the marker and disease loci.

6.2.1 R package

David Clayton's R package dgc.genetics can be found in `http://202.120.45.17/course/final/lab6/DGCgenetics_1.0.zip`.

6.2.2 Data for analysis

The data for this section concern a population-based case-control study of the association between a disease and four closely linked single nucleotide polymorphisms (SNPs). You can load and print a brief summary of the dataframe contents as following:

```
library(dgc.genetics)
data(popn)
summary(popn)
```

Each of the loci A, B, C and D are held as genotype variables. The dataframe also contains variables coding sex and case/control status.

We'll first attach this dataframe and do some tabulations:

```
data(popn)
table(affected)
table(sex)
```

6.2.3 Instructions

The following command does a chi-squared test for association between disease status and sex

```
chisq.test(sex, affected)
```

You should find a highly significant association between disease and sex (Pearson χ^2-test is the "score" test for association). Alternatively, we can create a table of disease status by sex and apply the $chisq.test()$ function to the table:

```
sbyd <- table(sex, affected)
sbyd
chisq.test(sbyd)
```

Genotype counting and allele counting There are two simple ways of testing for association between disease and a genetic variant. The first is to simply count genotypes in cases and controls and then compare them using a chi-squared test. For marker A,

```
abyd <- table(A, affected)
chisq.test(abyd)
```

The association is highly significant. Note that the test with degree of freedom 2 reflecting the fact that there are two dimensions in which the genotype distribution could differ.

Another commonly used analysis counts alleles (or chromosomes) rather than genotypes (people). The function $allele.table()$ expects its first argument to be a genotype variable and counts alleles — otherwise it is the same as $table()$:

```
abyd <- allele.table(A, affected)
chisq.test(abyd))
```

The chi-squared test now has one degree of freedom. This test is powerful against more restrictive alternative hypothesis in which the affect (broadly defined) on risk of genotype 1/2 vs 1/1 is the same as for genotype 2/2 vs 1/2 – the model of *generalized additive* effects of alleles. This strategy of counting alleles and treating them as independent samples from a population also assumes **Hardy-Weinberg equilibrium (HWE)**. This seems an appropriate point to mention that you could test for HWE by:

```
HWE.chisq(A)
```

However, it would usually be more appropriate to test for HWE only in controls. This brings us to an important mechanism for selecting *subsets* of data. We first generate ab object, $control$, which contains either TRUE or FALSE according to whether or not the subject is a control. We then use square brackets, as in $A[control]$, to select the genotypes for controls only:

```
control <- (affected=="Control")
table(control)
HWE.chisq(A[control])
```

Ex6.1 You might like to try some of these commands on the other marker loci in these data.

Logistic regression Analysis at the genotype (person level) is safer, since there is no need to assume HWE. A flexible way of carrying out

such tests is by use of logistic regression. An additional bonus of this approach is that it provides estimates of the size of genotype effects.

The general approach is to carry out a logistic regression which (rather counter-intuitively) treats disease status as the outcome variable and genotype as an explanatory variable. However, there are several ways in which the genotype can be entered into the regression. This can be controlled by setting an *attribute* of the genotype variable:

```
gcontrasts(A) <- "additive"
```

This attribute causes the genotype variable A to be entered into the logistic regression as a single indicator variable code 0, 1, 2 (the number of copies of allele "2"). This is the model of generalized additive allelic effects and, for the logistic model, corresponds to a multiplicative model for the odds ratio case:control. In a case/control study the measure of effect is then equivalent to the relative risk for each copy of allele "2". To fit this model,

```
logit(affected ~ A)
```

The relative risk bestowed by each copy of the "2" allele is 0.69.

In order to test for deviation, we need to include a "dominance" indicator in the model. Its effect here is to allow the risk for the 1/2 genotype to differ from the (geometric) mean of that for the two homozygous genotypes (1/1 and 2/2). We do this as follows:

```
gcontrasts(A) <- "dominance"
logit(affected ~ A)
```

There is now an extra coefficient (A:d:1:2) which tests this. It is not significantly different from its value under the null hypothesis (1.0). However, if it had been significant,this parametrisation would have been hard to interpret. Instead we might prefer to report the genotype relative risks. To obtain this,

```
gcontrasts(A) <- "genotype"
logit(affected ~ A)
```

This output gives the genotype relative risks with the most common genotype as baseline. If we want to use a different baseline, for example, "2/2",

```
gcontrasts(A, base="2/2") <- "genotype"
logit(affected ~ A)
```

EX6.2 You might test the other 3 loci and discuss with the given results.

6.3 Family-based Association Analysis

Case-control studies work perfect in most situations. However, if the sample comes from a heterogeneous population made up of two or more strata (i.e. sub-populations), and the strata differ with respect to their joint disease-marker distribution, this can by itself cause an overall disease-marker association, even if there is no such association in any of the separate strata. This kind of situation can be shown in following example:

N	A_i	B_j	A_iB_j
1000	0.3	0.5	0.15
2000	0.2	0.4	0.08
10000	0.05	0.1	0.005

Table 6.1: **Allele and haplotype relative frequenciesin three populations**

Example.1 Spurious associations Consider three populations that have reached gametic phase equilibrium with respect to allele A_i and B_j. Suppose that the population size (N), the relative frequency of allele A_i, the relative frequency of allele B_j, and the

relative frequency of the haplotype A_iB_j are as in Table (6.1). If these three subpopulations are merged what will be the allele and haplotype relative frequencies before any interbreeding takes place?

$$P(A_i) = [0.3 \times 1000 + 0.2 \times 2000 + 0.05 \times 10000]/13000 = 0.0923$$
$$P(B_j) = [0.5 \times 1000 + 0.4 \times 2000 + 0.1 \times 10000]/13000 = 0.1770$$
$$P(A_iB_j) = [0.15 \times 1000 + 0.08 \times 2000 + 0.005 \times 10000]/13000 = 0.0277$$

The equilibrium relative frequency of A_iB_j is $0.0923 \times 0.1770 = 0.0163$, which deviates from 0.0277. Alleles A_i and B_j are therefore associated in the merged population.

Although this type of association is of no biological interest, it is a true population association, caused merely by heterogeneity in the population.

There are three main ways to avoid an association due merely to heterogeneity. The first is to sample from a *homogeneous population*, but this may be difficult to achieve in practice. The second way is to include *appropriate covariates*, such as e.g. ethnicity, in the analysis, but this is not possible if the appropriate covariates, whether genetic or environmental, are not known. The third way is to use *matched controls*. Matching for ethnicity is necessary if other genetic factors could be causing an association, and one way to do this is to use family-based controls.

The most commonly used family-based association method is the transmission/disequilibrium test (TDT). TDT is used to evaluate departures from random assorment of alleles across families.

Here we will use the TDT and case/pseudocontrol approaches. The tests will be performed in the statistical analysis package R. As well as using standard functions implemented in R, we will also make use of special functions designed for genetic analysis that have been downloaded as add-in R libraries.

6.3.1 Data used for analysis

We will use family data consisting of a number of trio families with an affected diabetic child plus parents (of unknown disease status) all of whom are typed at 5 polymorphisms in the HLA region.

You can easily down the data file to your computer from `http://202.120.45.17/course/final/lab6/fiveloci.Rped`.

The data file is in standard pedigree file format, with columns corresponding to family id, subject id (within family), paternal id, maternal id, gender (1=male, 2=female), affection status (1=unaffected, 2=affected) and one column for each allele for each locus's genotype. The pedigree file used for the analysis in R differs from a standard pedigree file. It has a header line describing the different columns, and it uses R's own missing value code "NA" instead of zero.

6.3.2 Instructions

Your are now working within the R package, dgc.genetics. To begin with, what your do first is to install and read in the necessary libraries.

```
library(dgc.genetics)
```

To read your data into a dataframe called "family", type

```
family <- read.table("fiveloci.Rped", header=T)
```

And now you can look at the dataframe "family" by typing

```
family
```

or (better) by typing

```
fix(family)
```

Each variable can be accessed by using the name of the dataframe followed by a $ sign, then followed by the variable name, e.g.

```
family$pedigree
```

To make convenient, you can tell R to automatically look at variables in the "family" dataframe by typing

```
attach(family)
pedigree
```

To perform association analysis, we need to convert the variables corresponding to the two alleles at each locus into a genotype variable for each locus. This can be done e.g. for locus 1 to locus 5by

```
g1 <- genotype(loc1_1, loc1_2)
g2 <- genotype(loc2_1, loc2_2)
g3 <- genotype(loc3_1, loc3_2)
g4 <- genotype(loc4_1, loc4_2)
g5 <- genotype(loc5_1, loc5_2)
```

To perform a TDT analysis on these loci, type

```
tdt(g1)
```

Repeat the analysis for loci 2-5. As with the analysis in UNPHASED, you should find highly significant associations for loci 1, 2 and 5, and less significant associations at loci 3 and 4.

Case-Pseudocontrol Analysis

To create a case/pseudocontrol set for performing analysis at locus 5, for example, type:

```
psccloc5 <- pseudocc(g5, data=family)
```

This creates a new dataframe called "psccloc5" which contains cases each with 3 matched pseudocontrols.

Note that within the "psccloc5" dataframe, the 1/0 case/control indicator variable is called *cc* and the genotype variable is called *g5*. The *set* variable indicates which cases and pseudocontrols are in the same matched set. Besides, you need to clear the old "family" dataframe and old genotype variables from the memory and then read in "psccloc5", type:

```
detach(family)
rm(g1)
...
attach(psccloc5)
```

To analyze using conditional logistic regression, assuming either a 2df (genotype) test or a 1df (allele) test, type:

```
gcontrast(g5) <- "genotype"
clogit(cc ~ g5 + strata(set))

gcontrast(g5) <-  "additive"
clogit(cc ~ g5 + strata(set))
```

This will perform the analysis on the *g5* variable, also known as *psccloc5$g5*. The *strata(set)* option indicates that the *set* variable labels which cases and pseudocontrols are in the same matched set.

The results should be very similar to what you found in your UNPHASED and TDT analysis of locus 5, a highly significant likelihood ratio test.

The relative risk parameters labelled **exp(coef)** correspond to the risks of disease relative to the 1/2 genotype: we find values of 0.421 and 2.176 for the 1/1 and 2/2 genotype relative to the 1/2 genotype. So, if we wanted to calculate risks relative to the 1/1 genotype, we would

have values of $1.0/0.421 = 2.38$ for genotype $1/2$, and $2.176/0.421 = 5.17$ for genotype $2/2$, respectively.

For analysis at more than one locus (e.g. loci 4 and 5) we need to clear the memory and get back to our original "family" dataframe:

```
detach(psccloc5)
attach(family)
```

Now we need to create the relevant genotype variables and case/pseudocontrol datasets. We will create two different case/pseudocontrol datasets, one in which we do not keep track of phase information between the two loci, and one in which we condition on phase being known (in order to fit models where the disease risk depends on phase)

```
g4 <- genotype(loc4_1, loc4_2)
g5 <- genotype(loc5_1, loc5_2)
psccphase <- pseudocc(g4, g5, phase=TRUE, data=family)
psccnophase <- pseudocc(g4, g5, phase=FALSE, data=family)
```

For the "psccphase" dataframe, you should find that some sets of matched cases and pseudocontrols consist of one case and 3 pseudocontrols, other sets have only one pseudocontrol, and some families have been discarded entirely. This is because we the method has discarded pseudocontrols and families for which phase is not inferrable.

In the "psccphase" dataframe, you should also find that, as well as the genotype data at the two loci individually, a new two-locus phased genotype variable called g4.g5 has been created.

For the "psccnophase" data, when phase information is not kept, all families are used but only 1 pseudocontrol is generated per case. This is not in fact the most powerful approach, as Cordell and Clayton (2002) show that it is sometimes possible to generate 3 pseudocontrols per case in this situation. However, the R functions for case/pseudocontrol analysis are not yet fully developed, and so the more powerful creation

of 3 pseudocontrols per case in this situation is not yet implemented. (It is, however, implemented in David Clayton's Stata routines - as the "pseudocc" command in the "genassoc" package - so you might like to consider using these routines instead.

To analyse the "psccnophase" data, first clear the old "family" dataframe and associated genotype variables from the memory and read in the new dataframe as default:

```
detach(family)
rm(g4)
rm(g5)
attach(psccnophase)
```

To analyse each locus individually with a 2df test, type

```
gcontrasts(g4) <- "genotype"
clogit(cc ~ g4 + strata(set))
gcontrasts(g5) <- "genotype"
clogit(cc ~ g5 + strata(set))
```

To see whether locus 4 is significant once locus 5 is in the regression equation, use the following sequence of commands:

```
gcontrasts(g4) <- "genotype"
gcontrasts(g5) <- "genotype"
fullmodel<-clogit(cc ~ g5 + g4 + strata(set))
restrictedmodel<-clogit(cc ~ g5 + strata(set))
anova(restrictedmodel,fullmodel)
```

You should find a difference between the models reported as a deviance of 17.78 on 2df. To find the significance of this, use:

```
1-pchisq(17.78,2)
```

which gives you a p-value of around 0.00014. This suggests that locus 5 is not sufficient to account for all the association in the region: locus 4 still adds significantly to the regression model.

To see whether locus 5 is significant once locus 4 is in the regression equation, use the following sequence of commands:

```
gcontrasts(g4) <- "genotype"
gcontrasts(g5) <- "genotype"
fullmodel<-clogit(cc ~ g5 + g4 + strata(set))
restrictedmodel<-clogit(cc ~ g4 + strata(set))
anova(restrictedmodel,fullmodel)
```

You should find a difference between the models reported as a deviance of 48.33 on 2df. To find the significance of this, use:

```
1-pchisq(48.33,2)
```

which gives you a p value of around $3.2e-11$. This suggests that locus 4 is certainly not sufficient to account for all the association in the region: locus 5 still adds very significantly to the regression model.

Theoretically, one can compare these sorts of models for each pair of the loci in turn to see which loci might be able to account for most of the association. However, it turns out that the large number of alleles for loci 1 and 2 make this analysis a bit complicated: ideally one would want to group together alleles based on their frequency or on some biological criteria, or else drop the alleles that are very rare.

To fit a model for phase-known haplotypes at loci 4 and 5, discard the "psccnophase" dataframe from the memory, and read in the file "psccphase":

```
detach(psccnophase)
attach(psccphase)
```

The "psccphase" dataframe contains a two-locus phased genotype variable called g4.g5 . To fit a multiplicative model (equivalent to an additive model on the log odds scale) for the haplotypes type:

```
gcontrasts(g4.g5) <- "additive"
clogit(cc ~ g4.g5 + strata(set))
```

This gives a highly significant global test of 67.3 on 3df ($p = 1.63e-14$) for the effects of the 3 haplotypes (relative to the 1:2 haplotype). The individual haplotype relative risks are given under in column marked "exp(coeff)". It is seen that the 1:1 haplotype has the lowest risk and the 2:2 haplotype the highest, just as we found in our UNPHASED analysis.

Section IV

Microarray Data Analysis

7

Elementary Permutation Tests

7.1 Introduction

DNA microarray is one of the high-throughput biotechnologies that allow highly parallel and simultaneous monitoring of the whole genome. Increasingly it is employed to detect genes expressed differentially under diverse conditions. Typically two steps are used to identify differentially expressed (DE) genes: first, one computes the summary or test statistic (e.g. the mean) for each gene and rank the genes in order of their statistic; second, one chooses a threshold for the test statistics and call the genes with those above the threshold "significant" ones.

T-test is perhaps the most common and easiest approach for inference about means based on a single sample, matched pairs, or two independent samples, which rests on the assumption of normal distribution for data. However, no data are exactly normal. In these situations, inference about spread based on normal distributions is not robust and is therefore of little use in practice. Then, what should we do?

Researchers have proposed a variety of methods to deal with the difficulties. Of them, two state-of-the-art strategies — bootstrap confidence intervals and permutation tests — apply computing power to relax some of the conditions needed for traditional inference. We have

applied bootstrapping in previous lab to assess the reliability of the inferred phylogenic branches. In this lab we will go on discussing the permutation tests.

Similar to bootstrapping, permutation tests are conceptually simpler to understand than any other usual inference method: the sampling distribution that shows what would happen if we took very many samples under the same conditions. Although these methods do have some limitations, their effectiveness and range of use are so great that they are rapidly becoming the preferred way to do statistical inference.

7.2 Methods

7.2.1 Test statistics

For the purpose of clarity and simplicity, we only consider one-sample comparisons here, though extensions to two-sample comparisons and other more general settings are straightforward. Suppose after preprocessing, we have observed gene expression levels in the format of log ratios of the two-channel intensities in cDNA arrays, $X_{i1}, X_{i2}, \cdots, X_{ip}$ for gene i, $i = 1, 2, \cdots, n$ from p arrays.

For differential expression analysis (DEA), the null hypotheses are

$$\mathrm{H}_{i0} : E(X_{ij}) = 0 \text{ for } i = 1, 2, \cdots, n$$

Here we will consider three commonly used statistics.

The first one is the SAM-statistic, abbreviated as S-statistic.

$$S_i = \frac{\bar{X}_i}{(V_i + V_0)/\sqrt{p}}$$

where $\bar{X}_i = \sum_{j=1}^{p} X_{ij}/k$ and $V_i^2 = \sum_{j=1}^{p}(X_{ij} - \bar{X}_i)^2/(k-1)$ are the sample mean and sample variance of the expression levels for

gene i, and V_0 is a constant used to stabilize the denominator of the test statistic. V_0 can be chosen in different ways; one is $V_0 = median(V_1, \cdots, V_n)$.

The second is the mean statistic: $M_i = \bar{X}_i$, which corresponds to the early practice of simply using log fold change as a significance indicator.

The third one is the Student's t-statistic, $t_i = \bar{X}_i / V_i$, which is a standardized mean statistic.

7.2.2 Permutation test

A usual permutation test consists of 5 steps:

Firstly, one should analyze the problem: what is the hypothesis? What are the alternatives? What kind of distribution is the data drawn from? What losses are associated with bad decisions?

Secondly, one needs to choose the test statistic which will distinguish the hypothesis from the alternative.

Thirdly, one now can compute the test statistic for the original labeling of the observations.

Fourthly, the test statistic should be computed for all possible permutations (rearrangements) of labels of the observations.

The final step is to make a decision on whether to reject the hypothesis and accept the alternative based upon the original test statistic value and the permutation distribution of the statistic.

Permutation: Example

Assume we have two samples labeled as "X" and "Y" respectively. Now we want to compare the mean of the two samples. Intuitively, we can use Student's t-test to test the hypothesis, right?

X			Y		
A	B	C	D	E	F
121	118	110	34	12	22
$\bar{x_n} = 116.33$			$\bar{y_n} = 22.67$		

The null hypothesis and alternative are as follows:

$$\mathrm{H}_0 : \mu_x = \mu_y$$
$$\mathrm{H}_\mathrm{A} : \mu_x \neq \mu_y$$

The test statistic can be computed using the following equation

$$T = \frac{\bar{X} - \bar{Y}}{\sqrt{\frac{(n_x-1)S_x^2+(n_y-1)S_y^2}{n_x+n_y-2}}} \cdot \sqrt{\frac{n_x \cdot n_y}{n_x + n_y}}$$

If H_0 holds, T value follows t-distribution with degree of freedom of $n_x + n_y - 2$:

$$T \sim t_{n_x+n_y-2}$$

We thus can compute the p-value for the observed value t of test statistic T:

$$\begin{aligned} p &= 1 - P(|T| \leq |t||H_0) \\ &= 2[1 - P(T \leq |t||H_0)] \\ &= 2[1 - F_{t,n_x+n_y-2}(|t|)] \end{aligned}$$

If $p \leq \alpha$, then we can reject H_0. For the above example, we can easily get: $t = 13.0875$, two-sided $p = 0.0002$.

We can achieve with the other approach — permutation test. To illustrate how permutation works, we still use the above example.

After one permutation, we can get a relabeled table:

X			Y		
A	B	D	C	E	F
121	118	34	110	12	22
$\bar{x_n} = 91$			$\bar{y_n} = 48$		

Obviously there exist $C_6^3 = \frac{6!}{3!3!} = 20$ permutations. For each permutation, the corresponding T-value can be computed according to the same equation. Thus, we can see that in two of cases overall the absolute value of the test statistic t is greater than or equal to the absolute value of the original one. Therefore, we can obtain the exact p-value: $p = 2/20 = 0.1$.

Since the two samples have equal size, so only half of the permutations is really needed (symmetry) and 0.1 should be the smallest p-value we can get for comparing two groups of size 3.

For two unbalanced samples test with size m and $n-m$, we have $C_n^m = \frac{n!}{m!(n-m)!}$. For large sample size, we can't do all the permutations, instead we use Monte Carlo sampling to approximate the permutation test.

7.2.3 Simulation studies

Suppose for gene i, its observed gene expression level X_{ij} on array j has mean μ_i and variance σ_i^2; $\mu_i = 0$ if it is an equally expressed (EE) gene, and $\mu_i \neq 0$ differentially expressed (DE) gene. Then how to do permutation for the one-sample data?

Stop here for a few minutes, can you figure out an approach? Hint: note that this is the two-channel expression data, which represents the log-ratio of two channels.

Define Bernoulli random variable Y_{ij} as: $Y_{ij} = 1$ (corresponding to keeping the sign of X_{ij}) with probability $\pi = 0.5$ and $Y_{ij} = -1$ (corresponding to flipping the sign of X_{ij}) with probability $1 - \pi =$

0.5, and assume that X_{ij} and Y_{ij} are independent. Then the random variable $W_{ij} = Y_{ij}X_{ij}$ represents the permuted gene expression level for gene i at array j. It is so simple to verify that $E(Y_{ij}) = 2\pi - 1 = 0$, and then

$$\begin{aligned} E(W_{ij}) &= E(Y_{ij}E(X_{ij})) = (2\pi - 1)\mu_i = 0 \\ Var(W_{ij}) &= E(Var(Y_{ij}X_{ij}|Y_{ij})) + Var(E(Y_{ij}X_{ij}|Y_{ij})) \\ &= E(Y_{ij}^2\sigma_i^2) + Var(Y_{ij}\mu_i) \\ &= \sigma_i^2 + \mu_i^2 \end{aligned}$$

To facilitate discussion, we suppose that gene i is a DE gene, and rewrite $X_{ij} = X_{ij}^* + \mu_i$; X_ij^* can be regarded as the expression level of gene i if gene i were equally expressed.

The mean statistic

The null statistic for DE gene i is

$$\begin{aligned} m_i &= \sum_{j=1}^{p} \frac{W_{ij}}{p} \\ &= \sum_{j=1}^{p} \frac{Y_{ij}X_{ij}^*}{p} + \mu_i \sum_{j=1}^{p} \frac{Y_{ij}}{p} \end{aligned}$$

If gene i were an EE gene, $\mu_i = 0$ and its null statistic would become

$$m_i^* = \sum_{j=1}^{p} \frac{Y_{ij}X_{ij}^*}{p}$$

Because $\mu_i \neq 0$, it can be shown that $Var(m_i) = Var(m_i^*) + \mu_i^2/k$. Therefore, the distribution of the null statistic of a DE gene has heavier tails than that of an EE gene. In other words, because of the presence of both DE and EE genes, the distribution of the null statistics of all genes, as adopted in the standard permutation methods, has heavier tails than that of only EE genes. Note that the difference between

$Var(m_i)$ and $Var(m_i^*)$ depends on both μ_i and p, the difference will get smaller when p increases.

The t-statistic

The null statistic for DE gene i is

$$t_i = \frac{\sum_{j=1}^{p} Y_{ij} X_{ij}^*/p + \mu_i \sum_{j=1}^{p} Y_{ij}/p}{V(Y_{ij} X_{ij}^* + \mu_i Y_{ij})/\sqrt{p}}$$

In contrast, if gene i were an EE gene, its null statistic would be

$$t_i^* = \frac{\sum_{j=1}^{p} Y_{ij} X_{ij}^*/p}{V(Y_{ij} X_{ij}^*)/\sqrt{p}}$$

where $V(R_{ij})$ is the sample standard deviation of $R_{i1}, \cdots, R_{ip}$. Now can we here draw a conclusion that $Var(t_i) > Var(t_i^*)$?

Then we will use simulation to compare the variance of t_i and t_i^*, m_i and m_i^* under the hypothesis that X_{ij} has a normal distribution.

We first simulate X_{ij}^* from a standard normal distribution (i.e. with mean 0 and variance 1), and Y_{ij} from a Bernoulli distribution specified above, with $i = 1, \cdots, 100000$ and $j = 1, \cdots, p$. With $\mu_i = 2$ and $\mu_i = 0.5$, $p = 3 - 6$, calculate each m_i, m_i^*, t_i and t_i^*.

EX7-1 Please Summarize the variance for the four statistic into following table

Table 7.1: Variances of the null statistics for a DE gene and a corresponding EE gene with various replicates p and true difference of the means between EE and DE (μ_i)

μ_i	p	3	4	5	6
2	$Var(m_i)$				
	$Var(m_i^*)$				
	Relative difference				
	$Var(t_i)$				
	$Var(t_i^*)$				
	Relative difference				
0.5	$Var(m_i)$				
	$Var(m_i^*)$				
	Relative difference				
	$Var(t_i)$				
	$Var(t_i^*)$				
	Relative difference				

7.3 Multiple comparison problems

Due to the instability of variance with few replicates, t-test is not so powerful in differential analysis. Say we are interested in statistical inference, we need to define statistical significance. If we are ranking we might need to define a cutoff that defines interesting enough. The naïve answer to determine a cutoff is the p-values. However, are they appropriate?

Notice that if you were looking at 10,000 genes for which the null is true, you expect to see 500 attain p-values of 0.05. Thus for large number of simultaneous testing, such a cut-off is not appropriate, which is called the multiple comparison problem. Another popular solution is to report the FDR instead.

Consider the example in the following table, what happens if we called all genes significant with $p < 0.05$:

	Called Significant	Not Called Significant	Total
Null TRUE	V	$m_0 - V$	m_0
Alter. TRUE	S	$m_1 - S$	m_1
Total	R	$m - R$	m

EX7-2 As in the previous table, what is called Type I error? Type I error rate?

EX7-2 What is called Type II error? Type II error rate?

There are some definitions of the different error rates:

Per-comparison error rate (PCER) :
The expected value of the number of Type I errors over the number of hypotheses, that is

Per-family error rate (PFER) :
The expected number of Type I errors, that is

Family-wise error rate (FWER) :
The probability of at least one Type I error, that is

False Discovery Rate (FDR) :
rate that false discoveries occur, that is

$$FDR = E(V/R; R > 0) = E(V/R|R > 0)Pr(R > 0)$$

Positive false discovery rate (pFDR) :
rate that discoveries are false, that is

$$pFDR = E(V/R|R > 0)$$

8

Affymetrix Microarray Data Analysis with R and Bioconductor

This tutorial requires some basic knowledge of R, which we have discussed in the previous class.

8.1 R Review

As were reminded in previous class, there is a short description of how to get help in R and how to look at the variables in the workspace. You should use the functions extensively throughout the tutorial to understand the commands and follow what is going on.

Getting help There are many ways to get help from R. Find out what the function `library()` does by using the commands `help(library)` or `?library` results in a list of R-packages that are already loaded and can be used by you.

Online help Running `help.start()` launches a web browser that allows the help pages to be browsed with hyperlinks.

With `ls()` or `objects()` you get on overview of the objects in your workspace. Single objects can be removed by `rm()`. To clear your

whole workspace use `rm(list=ls())`.

Let's have a closer look at `x`. `summary()` gives you an overview of an object. The output depends on what type of object it is. For vectors you can get information on the distribution of values in it.

```
> x <- matrix(1:8, 2, 4)
> x
> summary(x)
> length(x)
> mode(x)
> class(x)
> dim(x)
```

Here `x` is a numeric matrix with 2 rows and 4 columns.

8.2 Bioconductor training

In the following example data from the `estrogen` package is loaded to demonstrate normalization and quality control functions for Affymetrix data in Bioconductor.

1). Install the required packages.

```
> source("http://bioconductor.org/biocLite.R")
> biocLite()
### additional packages
> pkgs <- c("affy","estrogen", "SpikeInSubset", "KEGG",
            "keggorth", "MLInterfaces", "XML", "sma",
            "scatterplot3d", "randomForest")
> biocLite(pkgs=pkgs)
```

2). Find the directory where the example cel files are, which should end in "extdata".

```
> library(affy)
> library(estrogen)
> library(vsn)
> cwd <- getwd()
> datadir <- system.file("extdata", package='estrogen')
> datadir
> dir(datadir)
> setwd(datadir)
```

The function `system.file()` here is used to find the subdirectory extdata of the `estrogen` package on your computer harddisk. To use your own data, set datadir to he appropriate path instead.

3). The file `estrogen.txt` contains information on the samples that were hybridized onto the arrays. Look at it in a text editor. Load it into a `phenoData` object with

```
> pd <- read.AnnotatedDataFrame("estrogen.txt", header=T,
          row.names="filename")
> pData(pd)
```

`phenoData` objects are where the Bioconductor package store s information about samples, for example, treatment conditions in a cell line experiment or clinical or histopathological characteristics of tissue biopsies. The `header` option lets the `read.phenoData` function know that the first line in the file contains column headings, and the `row.names` option indicates that the first column of the file contains the row names.

4). Now load the data from the CEL files as well as the `pheno` data into an `AffyBatch` object.

```
> a <- ReadAffy(filenames = rownames(pData(pd)),
          phenoData=pd, verbose=TRUE)
> a
```

5). Let's have a look at the CEL file images. The `image` function allows us to look at the spatial distribution of the intensities on a chip. This can be very useful for quality control. Fortunately, all of the 8 celfiles that we have just loaded do not show any remarkable spatial artifact.

```
> image(a[,1])
```

We also have the "bad" example:

```
> badc <- ReadAffy('bad.cel')
> image(badc)
```

Note that in these images, row 1 is at the bottom.

6). Histogram. Another way to visualize what is going on on a chip is to look at the histogram of its intensity distribution. Because of the large dynamic range ($O(10^4)$), it is useful to look at the log-transformed values.

```
> hist(log2(intensity(a[,4])), breaks=100, col='blue')
```

7). Normalization

Before comparing data from different arrays the probe-level data has to be summarized to represent expression levels per gene and intensities have to be normalized between different arrays. We can use the function `expresso` to choose between different methods to normalize the data and calculate expression values.

```
> x <- expresso(a, bg.correct=F, normalize.method='vsn',
       normalize.param=list(subsample=1000),
       pmcorrect.method='pmonly',
       summary.method='medianpolish')
```

The parameter subsample determines the time consumption, as well as the prediction of the calibration. The default (if you leave away the parameter `normalize.param=list(subsample=1000)`) is

20000; here we choose a smaller value for the sake of demonstration.

8). Boxplot

To compare the intensity distribution across several chips, we can look at the boxplots, both of the raw intensities a and the normalized probe set values x:

```
> boxplot(a, col='red')
> boxplot(data.frame(exprs(x)), col='blue')
```

In the command above, note the different syntax: a is an object of type `AffyBatch`, and the `boxplot` function has been programmed to know automatically what to do with it. `exprs(x)` is an object of type `matrix`. What happens if you do `boxplot(exprs(x))`?

```
> class(x)
> class(exprs(x))
```

9). Scatterplot

The `scatterplot` is a visualization that is useful for assessing the variation (or reproducibility depending on how you look at it) between chips. We can look at all probes, the perfect match probes only, the mismatch probes only, and of course also at the normalized, probe-set-summarized data. Distinguish between probes that are supposed to represent genes (you can access these, e.g. through the function `pm()`) and control probes.

```
> plot(exprs(a)[,1:2], log='xy', pch='.', main='all')
> plot(pm(a)[,1:2], log='xy', pch='.', main='pm')
> plot(mm(a)[,1:2], log='xy', pch='.', main='mm')
> plot(exprs(x)[,1:2], pch='.', main='x')
```

10). ALL data

The data used for these exercises come from a study of Chiaretti et al. on acute lymphoblastic leukemia (ALL), which was conducted with HGU95Av2 Affymetrix arrays. They are used to demonstrate the functions to find differential genes. The data package `ALL` contains an `exprSet` object called `ALL`, which contains the expression data that were normalized with `rma` (intensities are on the log2-scale), and annotations of the samples.

a. Load the `ALL` package. What is the dimension of the expression data matrix?

b. Use the function `show` to get an overview of the `exprSet` object. What are the variables describing the samples stored in the `pData` slot?

```
> library(ALL)
> library(hgu95av2)
> library(annotate)
> data(ALL)
> show(ALL)
> dim(exprs(ALL))
> print(summary(pData(ALL)))
```

11). B-cell ALL

We want to look at the B-cell ALL samples (they can be identified by the column BT of the `pData` of the `exprSet ALL`). Of particular interest is the comparison of samples with the BCR/ABL fusion gene resulting from a translocation of the chromosome 9 and 22 (labeled BCR/ABL in the column mol), with samples that are cytogenetically normal (labeled NEG).

a. Define an `exprSet` object containing only the data from the B-cell ALL samples. How many samples belong to the cytogenetically defined groups?

```
> pdat <- pData(ALL)
> table(pdat$BT)
> table(pdat$mol.biol)
> subset <- intersect(grep("^B", as.character(pdat$BT)),
      which(as.character(pdat$mol.biol) %in%
      c("BCR", "NEG")))
> eset <- ALL[,subset]
> table(eset$mol.biol)
```

12). Non-specific filtering

Many of the genes on the chip won't be expressed in the B-cell lymphozytes studied here, or might have only small variability across the samples.

a. We try to remove these genes (more precisely, the corresponding probe sets) with an intensity filter (the intensity of a gene should be above 100 in at least 25 percent of the samples), and a variance filter (the interquartile range of log2-intensities should be at least 0.5). We create a new `exprSet` containing only the probe sets which passed our filter. How many probe sets do we get?

```
> library(genefilter)
> f1 <- pOverA(.25, log2(100))
> f2 <- function(x) (IQR(x) > .5)
> ff <- filterfun(f1, f2)
> selected <- genefilter(eset, ff)
> sum(selected)
> esetSub <- eset[selected,]
```

13). Differential expression

Now we are ready to examine the selected genes for differential expression between BCR/ABL samples and the cytogenetically normal ones.

a. Use the two-sample t-test to identify genes that are differentially expressed between the two groups. The function `mt.teststat` from the `multtest` package allows to compute several commonly used test statistics for all rows of a data matrix - study its help page. First, we calculate the nominal p-value — the function `pt` gives the distribution function of the t-distribution. We can get an impression of the amount of differential gene expression by looking at a histogram of the p-value distribution.

```
> library(multtest)
> cl <- as.numeric(esetSub$mol.biol == "BCR/ABL")
> t <- mt.teststat(exprs(esetSub), classlabel=cl,
      test="t.equalvar")
> pt <- 2 * pt(-abs(t), df=ncol(exprs(esetSub))-2)
> hist(pt, 50)
```

b. The function `p.adjust` contains different multiple testing procedures. Look at the help page of this function. For p-value adjustment in terms of the FDR, we use the method of Benjamini and Hochberg. How many genes do you get when imposing an FDR of 0.1?

```
> pa <- p.adjust(pt, method='BH')
> sum(pa < 0.1)
```

c. Plot the p-value against the log-ratios (differences of mean log-intensities within the two groups) in a volcano plot. Note the asymmetry of the volcano plot.

```
> logRatio <- rowMeans(exprs(esetSub)[, cl==1])
                - rowMeans(exprs(esetSub)[, cl==0])
> plot(logRatio, -log10(pt),
      xlab='log-ratio', ylab='-log10(p)')
```

14). Limma

A `t-test` analysis can also be conducted with function of the `limma` package.

a. First, we have to define the design matrix. One possibility is to use an intercept term that represents the mean log-intensity of a gene across all samples (first column consisting of 1's), and to encode the difference between the two classes in the second column.

```
> library(limma)
> design <- cbind(mean=1, diff=cl)
```

b. A linear model is fitted for every gene by the function `lmFit`, and Empirical Bayes moderation of the standard errors is done by the function `eBayes`.

```
> fit <- lmFit(esetSub, design)
> fit2 <- eBayes(fit)
> topTable(fit2, coef='diff',
           adjust.method='fdr')
```

c. when you compare the resulting p-value with those from the parametric t-test, you will see that they are almost identical. Because of the large number of samples, the Empirical Bayes moderation is not so relevant in this data set — the gene-specific variance can well be estimated from the data of each gene.

```
> plot(log10(pt), log10(fit2$p.value[,'diff']),
       xlab="two-sample t-test",
       ylab='limma')
> abline(c(0, 1), col='Red')
```

15). Annotation

a. Now we want to see which genes are the most significant ones, and look at their raw and adjusted p-values from the different methods. Gene symbols are provided in the annotation package `hgu95av2`.

```
> diff <- ((1:length(pa))[order(pa)])[1:10]
> genesymbols <- mget(geneNames(esetSub)[diff],
```

```
                hgu95av2SYMBOL)
> pvalues <- cbind(pt, pa)[diff, ]
> rownames(pvalues) <- genesymbols
> print(pvalues)
```

b. The top 3 probe sets represent the ABL1 gene, which is affected by the translocation characterizing the BCR/ABL samples. Now we want to see whether there are further probe sets representing this gene, and whether these have been selected by our nonÍC-specific filtering.

```
> geneSymbols = mget(geneNames(ALL), hgu95av2SYMBOL)
> ABL1probes <- which(geneSymbols == "ABL1")
> selected[ABL1probes]
```

16). Gene Ontology.

a. Many of the effects due to the BCR/ABL translocation are mediated by tyrosine kinase activity. Let's look at the probe sets that are annotated at the GO term protein-tyrosine kinase activity, which has the identifier GO:0004713.

```
> gN <- geneNames(esetSub)
> tykin <- unique(lookUp("GO:0004713",
            "hgu95av2", "GO2ALLPROBES"))
> str(tykin)
> sel <- (gN %in% unlist(tykin))
```

b. We can now check whether there are more differentially expressed genes among the tyrosine kinases than among the other genes. Fisher's exact test for contingency tables is used to check whether the proportions of differentially expressed genes are significantly different in the two gene groups.

```
> tab <- table(pt < 0.05, sel, dnn = c("p < 0.05", "tykin"))
> print(tab)
> fisher.test(tab)
```

17). ROC curve screening

a. We want to find marker genes that are specifically expressed in leukemias with the BCR/ABLÍCtranslocation. At different cut-off levels we can determine how well the expression levels of the genes are seperated between the two classes and calculate specificity and sensitivity for each gene. At a specificity of at least 0.9, we would like to identify the genes with the best sensitivity for the BCR/ABL phenotype. This can be expressed by the partial area under the ROC curve (pAUC, we choose t0 = 0.1). To limit the computation time, we compute the pAUCÍCstatistic only for the first 100 probe sets.

```
> library(ROC)
> mypauc1 <- function(x) {
   pAUC(rocdemo.sca(truth = cl, data = x,
       rule = dxrule.sca),
   t0 = 0.1)
}
> pAUC1s <- esApply(esetSub[1:100, ], 1, mypauc1)
```

b. Select the 2 probe sets with the maximal value of our pAUC-statistic, and plot the corresponding ROC curves. Look for a comparison at the t-test p-values for these genes.

```
> best <- order(pAUC1s, decreasing = T)[1:2]
> x11()
> par(mfrow = c(1, 2))
> for (pS in best) {
       RC <- rocdemo.sca(truth = cl,
           data = exprs(esetSub)[pS, ],
           rule = dxrule.sca)
       plot(RC, main = geneNames(esetSub)[pS])
> }
> print(pt[best])
```

Section V

Computational Systems Biology

9

Protein-Protein Interaction Network

Objective

This lab is to introduce the network biology, which play an essential part in systems biology.

9.1 Biological Networks

Network of interaction are fundamental to all biological processes; for example, the cell can be described as a complex network of chemicals connected by chemical reactions. Cellular processes are controlled by various types of biological network: metabolic network, protein-protein interaction network, and gene regulatory network.

The last few years has witnessed the great progress in analyzing biological networks using the statistical mechanics of random network approach. The random network approach is becoming a powerful tool for investigating different biological systems,such as the yeast protein interaction network, food web and metabolic metabolic network. Many studies indicate that there are underlying global structures of those bi-

ological networks.

Metabolic Network

Metabolism comprises the network of interaction that provides energy and building blocks for cells and organisms. In many of the chemical reactions in living cells, enzymes act as catalysts in the conversion of certain compounds (substrates) into other compounds (products). Comparative analyses of the metabolic pathways formed by such reactions give important information on their evolution and on pharmacological targets. Recently, the large-scale organization of the metabolic networks of 43 organisms are investigated and it is found that they all have the feature of scale-free small-world network, i.e. $P(k)\ k^g$, where k is the number of links, and the diameter of the metabolic pathway is the same for the 43 organisms.

Protein-Protein Interaction Network

Proteins perform distinct and well-defined functions, but little is known about how interactions among them are structured at the cellular level. Recently, it was reported that in the yeast (a total of 3728 proteins by the Y2H method measurement), the protein interactions are not random, but well organized. It was found that, most of the neighbors of highly connected proteins have few neighbors, which implies that highly connected proteins are unlikely to interact directly with each other.

Gene Transcription Regulatory Network

A genetic regulatory network consists of a set of genes and their mutual regulatory interactions. The interactions arises from the fact that genes code for proteins that may control the expression of other genes, for

instance, by activating or inhibiting DNA transcription. Recently, it was reported that in the yeast organism, there is a hierarchical and combinatorial organization of transcriptional activity pattern.

To gain more knowledge on properties of complex network, you can review the article below:

> *Newman M E J. The structure and function of complex networks. SIAM Review, 2003, 45: 167-256.*

9.2 Materials and Methods

9.2.1 Database of Interacting Protein (DIP)

There are thousands of different proteins active in a cell at any given time. Many proteins act as enzymes, catalyzing the chemical reactions of metabolism. In our analysis we will use the database DIP (`http://dip.doe-mbi.ucla.edu`) as the input data. DIP is a database that documents experimentally determined protein-protein interactions (a binary relation). We analyze the latest version of the DIP database (Nov, 2007), for six different species, *S. cerevisiae, H. pylori, E. coli, H. sapiens, M. musculus* and *D. melanogaster*.

In order to minimize experimental uncertainty, we employed the CORE subset or DIP, which contains the pairs of interacting proteins identified in the budding yeast, *S. cerevisiae*, that were validated according to the gold standard.

EX7-1 First fill in the table 9.1 below according to the statistics of the number of proteins, and number of interactions in the six corresponding species.

Table 9.1: **DIP statistic for the six species studied**

Organism	Proteins	Interactions
S. cerevisiae (CORE)		
H. pylori		
E. coli		
H. sapiens		
M. musculus		
D. melanogaster		

9.2.2 Topological Properties of a Complex Network

The biological network mentioned above have a complex topology. A complex network can be characterized by certain topological measurements.

In the graph theory approach, each protein is represented as a node and interaction as an edge. By analyzing the DIP database on can construct an interaction matrix to represent the protein-protein interaction network. In the interaction matrix a value of one and infinity is assigned to represent direct interacting and non-interacting protein respectively.

EX7-2 Above all, it is required that you write a program (preferably in C++) to convert the DIP format file into pajek-format file (***.net**) so that the data can be loaded into pajek for further analysis.

Degree Distribution

The first topological feature of a complex network is the distribution of its degree of connectivity. From the pajek, one can obtain a histogram of k interactions for the network. And thus you can infer the distribution, $p(k)$. We know that in a random network, each edge is present or absent with equal probability, and hence the degree distribution is bino-

mial, or Poisson in the limit of large network size. Real-world networks are mostly found to be sharply different from the random network in their degree distribution. As reported, far from having a Poisson distribution, the degrees of the proteins are highly right-skewed, implying a long right tail.

An alternative way of presenting degree data is to make a plot of the cumulative distribution function

$$P(k) = \sum_{k'=k}^{\infty} p(k') \tag{9.1}$$

which is the probability that the degree is greater than or equal to k. In many real-world network, the degree distribution has no well-defined peak but has a power-law distribution, $P(k)\ k^{-r}$, where r is a constant. Such a network is known as scale-free network. The power-law form of the degree distribution implies that the networks are extremely inhomogeneous. In a scale-free network, there are many nodes with small degrees and a few nodes with large degrees. And the highly connected nodes ought to play a key role in the functioning of the network.

EX7-3 Draw both the density distribution $(p(k))$ and cumulative distribution $(P(k))$ of degrees for the protein interaction networks of the six species, respectively.

Interaction Path Length

Proteins can have direct or indirect interactions among themselves. Direct interactions such as binding interaction, including formation of protein complexes, covalent modifications of phosphorylation, glycosylation, and proteolytic processing of polypeptide chain. Indirect interaction refers to two proteins are interacted indirectly via successive chemical reactions. Among class of indirect protein-protein interaction

is gene regulation, where the message of one protein is transmitted to the next protein via the process of protein synthesis from the gene.

The second topological measurement is the distance between two nodes, which is given by the number of links along the shortest path. The number of links by which a node is connected to the other nodes varies from node to node. The diameter of the network, also known as the average path length, is the average of the distances between all pairs of nodes.

For all pairs of proteins, the shortest interaction path length, $L(j)$ (i.e. the smallest number of reactions by which one can reach protein 2 from protein 1) will be determined by using the Floyd's algorithm. Floyd's algorithm is an algorithm to find the shortest paths for each node in a graph. It does this by operating on a matrix representing the costs of edges between vertices. The diameter d is given by

$$d = \frac{\sum_j jL(j)}{\sum_j L(j)} \tag{9.2}$$

where j is the shortest path length and $L(j)$ is the frequency of nodes having path length j.

Robustness of the Network

In order to test whether the interaction network is robust against errors, we slightly perturbed the network randomly. First, we randomly select a pair of edges A-B and C-D. The two edges are then rewired in such a way that A connected to D, while B connect to C. Notice that this process will not change the degree of each node. A repeated sampling (100 times of random sampling) of the randomized networks allowed us to calculate the average diameter of the perturbed network d_{pert} and compare the perturbed results with the unperturbed network, i.e. $D = (d_{per} - d)/d \times 100\%$.

EX7-4 Generate 100 poisson random networks with the same proteins

and interactions for the previous six networks. Compare the real-world networks with the artificial random network. Fill in the table 9.2.

Table 9.2: **Maximum connectivity, average diameter of the six species**

Organism	k_{max}	d	d_{rand}	$d_{pert}(D)$
S. cerevisiae (CORE)				
H. pylori				
E. coli				
H. sapiens				
M. musculus				
D. melanogaster				

Here k_{max} is the maximum degree; d denotes the average diameter of the network; d_{rand} corresponds to the average diameter of the random network; d_{pert} refers to the average diameter of the perturbed network, and D can be called perturbation coefficient.

Transitivity or Clustering

A clear deviation from the behavior of the random network can be seen in the property of network transitivity, sometimes also called clustering, though the latter term also has another meaning and might cause confusion. In many networks it is found that if A is connected to B and B to C, then there is a heightened probability that A will also be connected to C. In the language of social networks, the friend of your friend is likely also to be your friend. In terms of network topology, transitivity means the presence of a heightened number of triangles in the network. It can be quantified by defining a clustering coefficient C thus:

$$C = \frac{3 \times number\ of\ triganles\ in\ the\ network}{number\ of\ connected\ triplets\ of\ vertices} \tag{9.3}$$

EX7-5 Show me how much the clustering coefficient of your network is deviated from the random network generated by pajek.

9.3 Conclusion and Discussion

To be filled later by the students

9.4 Appendix: Using Pajek

Pajek is a windows program that provides some integrated analysis tools and graph drawing capabilities for large-scale complex networks. It also can run nicely on Linux via Wine, a famous virtual machine under Linux.

Input file format

Pajek network files are in plain text, with a very strict format. If the file format is not correct, the file will not be loaded into Pajek. The format is as follows:

```
*Vertices <number of vertices>
1 "label1"
2 "label2"
...
*Edges
<vertex1> <vertex2>
<vertex3> <vertex4>
...
```

Here is an example a Pajek network file:

```
*Vertices 1788
1 "YAL005C"
2 "YAR002W"
3 "YLR310C"
4 "YPL240C"
5 "YAL021C"
...

*Edges
1 2
1 3
1 4
5 6
7 8
9 10
11 12
13 14
15 16
17 18
17 19
20 20
20 21
20 22
23 23
24 24
25 26
27 28
27 33
...
```

Note that the whitespace consists of SPACEs, not TABs.

(1) Start Pajek and load the file you created using "File -> Network

-> Read" from the menu;

(2) Visualize the network by selecting "Draw -> Draw" from the menu;

(3) The network layout can be altered either by moving vertices using the mouse, or by choosing a new layout from the "Layout" menu;

(4) Save your preferred layout as bitmap using "Export -> Bitmap", and paste it into your lab writeup;

(5) The connectivity of each node can be counted using "Net -> Partitions -> Degree -> All". Note that when you do this an entry appears in the Partition box of the Pajek interface. Click the save button to save the partition information file into your hard disk. You can use either Excel/Origin or R to draw the histogram. Inspect the results by using the "Edit Partition" button on the Pajek interface. More information can be gained using "Info -> Partition" from the menu system. Note that the numbers here are double what you see on the screen,since every edge is counted twice, as A -> B and as B -> A;

(6) Cluster coefficient is calculated using "Net -> Vector -> Clustering Coefficients -> CC1" (CC2 is a normalized cluster coefficient). This calculates the CC for each node. The CC of the network is the average of all the nodes in the network.

Section VI

Writing and Presentation Skills

10

Write an Essay on the Topic of Your Interest

Objective

This lab requires you all to select a topic and prepare an essay based on at least two papers on the topic. You should make sure that the resulting essays will be interesting and accessible to any non-expert in the given topic. Most of the topics have a strong algorithmic flavor, but some are more geared towards biology. Please sign up for the topics listed below, on the first-come first-serve basis.

10.1 Suggested Topics

1). Bioinformatics database

2). Regulatory motif finding

3). Protein structure prediction

4). Phylogenetic inference

5). Gene mapping (linkage and association analysis)

6). Protein multiple alignment

7). Comparative gene finding

8). DNA computation

9). Modeling regulatory networks

10). Metabolic network construction and analysis

11). Molecular dynamics simulation

10.2 Requirements

1). An explicit title should be given to your essay so that the reviewer can immediately gain the impression of your topic;

2). The essay should consist of the motivation, central idea, results and conclusion (if available);

3). Your own comments should also be included in the essay;

4). You should write at least 1000 words; specially, your own comments should be more than 200 words;

5). Prepare a talk (about 5 to 10 minutes) on the subject of your choice.

6). Hand in a poster for one of the research articles you read.