

# Data Oriented Genomics Techniques

Zhihua Hua

ASK18	GDILSKIIEY	AKMHVNEPSE	EEAKKNLDSW	DAKFMKLDL	ETIFKIILAA	NYLNFEGLLG	FASQTV----
AlySkp08	GDILSKIIEY	AKKHVVEP-D	EEAKKKLDSW	DAKFVEKLDL	ETIFKIILAA	NYLNFEGLLG	FASQTV----
AhaSkp07	GDILSKIIEY	AKKHVVEP-D	EEAKKKLDSW	DAKFVEKLDL	ETIFKIILAA	NYLNFEGLLG	FASQTV----
ASK19	GEILAKVIEY	CKKHVED--D	DDKKEKLNEW	DAKFMKDFDI	KTIFDIILAA	NYLNVQGLFD	LCSKTI----
ASK15	GNILSIVLEY	CKKHVDDVVD	DEAKONLDAW	DAEFMKNIDM	ETIFKLILAA	NYLNVGELLG	LTCQTV----
AhaSkp02	GKTLSMVLEY	CKKHVDDVVD	DEAKKKLDAW	DAEFMKDLNM	ETIFNIILAA	NYLNVKGLLD	LTCQTV----
AlySkp03	GKTLSMVLEY	CKKHVDDVVA	DEAKKKLDAW	DAKFMKDLNT	ETIFSIILAA	NYLNVKGLLD	LTSQTV----
ASK16	GNILALVIEY	CKKHVLDVDD	DEAKNELRTW	DAEFMKFEDM	ETVMKLILAV	NYLNVQDLLG	LTCQTV----
ASK17	GKILAIIEY	CKKHVDDV--	-EAKNEFVTW	DAEFVKNIDM	DTLFLKLLDAA	DYLVIVIGLKN	LIAQAI----
AlySkp07	GKILAIIEY	CKKHVDD--V	DAKNELVMTW	DAEFMKNIDM	ETVFKLLNAA	DYLVNVKGLLD	LTSNTI----
AhaSkp08	GKILAMVIEY	CKKHVNDVDS	DEAKNELVAW	DAEFMKNIDM	DTIFNLLLLAA	NYLNVKGLLD	LTSQTI----
AlySkp12	GKILAVVIEY	CKKHVNDVDD	SEAKKELVTW	DAEFMKDIDM	ETMFQLLLLAA	NYLNVKSLLD	LTSQTI----
ASK14	GKILSMVVEY	CKKHVVD---	-EESDEFKTW	DEEFMKKFDQ	PTVFQLLLLAA	NYLNIKGLLD	LSAQTV----
AlySkp15	GDILSMVIEY	CKTHVDE--E	EEAOTKLKTW	DEEFMKKFDI	KTLLQIILAA	NYLNVKGLLD	LVSQTI----
AhaSkp10	GDILSMVIEY	CKEHVDE--E	EEAOTKLKTW	DDEFTKRFDL	QTLKILILAA	NYLNVKGLLD	LVSQTV----
ASK6	SKILLLVIEY	CKKHVVE--S	--KEEDLKKW	DAEFMKKMEQ	SILFDVMMAA	NYLNIQSLLD	LTFSSNC----
AlySkp02	SKILLLVIEY	CKKHVVE--N	-EEEEYLKKW	DTEFMKKMEQ	SIVFDVMMAA	NYLNIQSLID	LTCQTV----
AhaSkp11	SKILLLVIEY	CKKHVVE--S	-NEEEDLKKW	DTEFMKKMEQ	SIVFDVMMAA	NYLNIQSLID	LTCQTV----
ASK5	SKILKIVIDY	CEKHVKS---	-KEEEDLKEW	DADFMKTIET	TILFDVMMAA	NYLNIQSLID	LTCQTVSDLL
AhaSkp09	SKILKIVIAIY	CKKHVES---	-NEEEDLMEW	DADFMKKIEP	SILFDVMMAA	NYLCTQTLID	LTCQTVAAALL
AlySkp10	SKILKIVIAIY	CKKHVES---	-NEEEDLKEW	DADFMKKIEP	SILFDVMIAA	NYLNIQSLID	LTCQTVAAALL
AlySkp14	-----	-----	-----M	GRKVME-KDQ	LTFLDLINAA	SYLDIQSLLD	LACQTA----
AlySkp16	GKILAMVLEY	CKKHVVD--D	ASTDEDLKKW	DEKFME-KDQ	LTFLDLINAA	SYLDIQSLLD	LACQTA----
AhaSkp05	SKILSMVVEY	IKKHVVD--A	ASTDEDLKKW	DAEFMQ-IDQ	STIFDLIMVA	NHLIKSLID	LTCQTV----
AlySkp09	GKILSMVVEY	LNKHHVG--D	ASTDEDLKKW	DAEFMQ-IDQ	STIFDLIMAA	NHLNIKSLTD	LTCQTV----
ASK10	GKILEMVIEW	CNKHHVD--A	ACSDDEDLKKW	DKEFME-KYQ	STIFDLIMAA	NYLNIKSLLD	LACQTV----
ASK9	GKILAMVIEY	CNKHHVD--A	ACSDDDLKKW	DKEFME-KDT	STIFDLIKAA	NYLNIKSLFD	LACQTV----
ASK7	SEILEMVIEW	CNKHHVD--A	ACSDDEDLKKW	DKEFME-KDQ	YTIFHLMNAA	YDLHIKSLLA	LAYQTV----
ASK8	SEILEMVIEW	CNKHHVD--A	ACSDDDLKWK	DKEFME-KDK	STIFALTNA	NFLNNKSLH	LAGQTV----
ASK13	GVILSKVIEY	CKKHVVS--D	SESKDELKKW	DAEFMKALEQ	STLFDVMLAA	NYLNIKDLID	LGCQTV----
AhaSkp06	GATLSKVIEY	CKKHVVA--E	SESKDELKKW	DAEFMKAMEQ	STLFHVILAA	NYLNIKDLID	LGCQTV----
AlySkp13	GATLSKVIEY	CKKHVVA--A	EEEWDELKKW	DAEFMKAMEQ	STLFHVILAA	NYLNIKDLFD	LGCQTV----
AlySkp17	GGILAKVIEC	CKKHVET--A	ATENKELKAW	DADVFQ-VDQ	PILFDLILVA	NYLNNSGLLD	LTCQTV----
AlySkp11	GAILAKVIEY	CKKHVEA--A	ATENDELKAW	DNDFVK-VDQ	PTLFDLILAA	NYLNIQSLLD	LTCQTV----
AlySkp04	GAILAKVIEY	CKKHVEA--A	ATENDELKAW	DNDFVK-VDQ	PTLFDLILAA	NYLNIQSLLD	LTCQTV----
AhaSkp03	GAILAKVVEY	CKKHVEA--A	ATENDELKAW	DNDFVK-VDQ	PTLFDLILAA	NYLNIQSLLD	LTCQTV----
ASK3	GAILAKVIEY	CKKHVEA--A	ATENHELKTW	DNDFVK-VDH	PTLFDLILAA	NYLNIQSLLD	LTCQTV----
ASK4	GAILAKVIEY	CKKHVEA--A	AAENDELKNW	DSEFVK-VDQ	PTLFDLILAA	NYLNIQSLLD	LTCQTV----
AlySkp06	SKTLAKVIEY	CKKHVVD--E	AISEDELKKW	DTEFME-TDQ	STIFDLILAA	NYLNIKSLLD	LTCQTI----
ASK12	SKILVKVIEY	CKKYHVD--E	AISEEDLNKW	DEKFMD-LEQ	STIFELILAA	NYLNIKSLFD	LTCQTV----
ASK11	SKILVKVIEY	CKKHVVD--E	AISEEDLNNW	DEKFMD-LEQ	STIFELILAA	NYLNIKSLLD	LTCQTV----
ASK2	SKILSKVIEY	CKRHVEA--A	ESSDELKKTW	DSEFIK-VDQ	GTLFDLILAA	NYLNIKGLLD	LTCQTV----
AhaSkp04	SKILSKVIEY	CKKHVEA--A	ASSDEDLKTW	DSEFIK-VDQ	GTLFDLILAA	NYLNIKGLLD	LTCQTV----
AlySkp05	SKILSKVIEY	CKKHVEA--A	ASSDEDLKTW	DSEFIK-VDQ	GTLFDLILAA	NYLNIKGLLD	LTCQTV----
ASK1	SKILAKVIEY	CKRHVEA--A	ATSDDDLKAW	DADFMK-IDQ	ATLFEILILAA	NYLNIKSLLD	LTCQTV----
AhaSkp01	SKILAKVIEY	CKKHVEV--A	ATSDDELKAW	DTEFMK-IDQ	ATLFEILILAA	NYLNIKSLLD	LTCQTV----
AlySkp01	SKILAKVIEY	CKKHVEV--A	ATSDDELKAW	DTEFMK-IDQ	ATLFEILILAA	NYLNIKSLLD	LTCQTV----



# Data Oriented Genomics Techniques

**Zhihua Hua**

Department of Environmental & Plant Biology  
Ohio University  
Athens, OH 45701



**OHIO**  
UNIVERSITY



**Perl**





**Copyright © 2018 Zhihua Hua**

## **Creative Commons License**

The contents in this book are licensed under the Creative Commons Attribution-ShareAlike 4.0 International License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-sa/4.0/> or send a letter to Creative Commons, PO Box 1866, Mountain View, CA 94042, USA.





# Table of Contents

<b>Preface</b>	<b>ix</b>
<b>Acknowledgments</b>	<b>xi</b>
<b>Lab 1. Open Source Programming 1</b>	<b>1</b>
Part 1: General UNIX Commands	2
Part 2: A Real-Time Task	3
<b>Lab 2. Open Source Programming 2</b>	<b>5</b>
Part 3: Compile Your BLAST Standalone Program	5
Part 4: Executable Functions	5
<b>Lab 3. BLAST</b>	<b>9</b>
Part 1: Make a BLASTP Database	11
Part 2: BLASTP Search	11
Part 3: hallo.pl	12
Part 4: blast_parse.pl	13
<b>Lab 4. Bioperl 1</b>	<b>19</b>
Part 1: Fasta Sequence File	20
Part 2: Bio::DB::Fasta Bioperl Module	21
<b>Lab 5. Bioperl 2</b>	<b>25</b>
Part 1: Protein Sequence Retrieval	25
Part 2: Transcriptome and Proteome Comparison	26
<b>Lab 6. Molecular Phylogenetics</b>	<b>31</b>
Part 1: Multiple Sequence Alignment	31
Part 2: Sequence Alignment Format	33
Part 3: PHYLIP Phylogenetic Analysis	33
<b>Lab 7. Molecular Cloning</b>	<b>37</b>
Part 1: Polymerase Chain Reaction	39
<b>Lab 8. Ligation</b>	<b>43</b>
Part 1: Gel Electrophoresis	45
Part 2: Gel Extraction	45
Part 3: Set Up Ligation	46

<b>Lab 9. Bacterial Transformation &amp; Genomic DNA Extraction</b>	<b>49</b>
Part 1: Bacterial Transformation	50
Part 2: Genomic DNA Extraction	51
<b>Lab 10. Genotyping</b>	<b>55</b>
Part 1: Count Transformation and Ligation Efficiencies	56
Part 2: Colony PCR	56
<b>Lab 11. Gene Amplification</b>	<b>59</b>
Part 1: Make an Agarose Gel	60
Part 2: DNA Gel electrophoresis of PCR Products	60
<b>Lab 12. Plasmid Isolation</b>	<b>63</b>
Part 1: Plasmid MiniPrep	64
Part 2: Plasmid Yield	64
<b>Lab 13. Restriction Fragment Length Polymorphism</b>	<b>67</b>
Part 1: DNA Digestion	68
Part 2: Make an Agarose Gel	68
Part 3: Gel Electrophoresis of the DNA Digestion Products	68
<b>Lab 14. Protein Gel Electrophoresis</b>	<b>71</b>
Part 1: Total Plant Protein Extraction	72
Part 2: Protein Gel Electrophoresis	72
Part 3: Wet Protein Transfer	72
<b>Lab 15. Yeast Two-hybrid</b>	<b>75</b>
Part 1: Yeast Mating Assay	77
<b>Lab 16. Protein Immuno-Blotting</b>	<b>81</b>
Part 1: Immuno-blotting	82
Part 2: HRP Display	83
<b>Lab 17. Semi Quantitative Y2H Interaction Assay</b>	<b>85</b>
Part 1: Prepare Plates for Y2H Interaction Assay	85
Part 2: Equalize Cell Concentration	86
Part 3: Serial Dilution	86
Part 4: Spotting Inoculation	87



<b>Lab 18. R Programming</b>	<b>89</b>
Part 1: Open R	89
Part 2: Regular R Commands	90
Part 3: Microarray Data Retrieval and Visualization	91
<b>Lab 19. Processing RNA-Seq Raw Data</b>	<b>97</b>
Part 1: Evaluate Data Quality	98
Part 2: Trim Sequence Data	98
Part 3: Align Cleaned Reads to the Reference Genome	98
<b>Lab 20. Differential Gene Expression Analysis</b>	<b>101</b>
Part 1: edgeR based DEG Analysis	101
<b>Lab 21. RNA Isolation</b>	<b>107</b>
Part 1: RNA Isolation	108
Part 2: RNA Yield	109
<b>Lab 22. RNA Quality Examination</b>	<b>111</b>
Part 1: Make an Agarose Gel	112
Part 2: RNA Agarose Gel Electrophoresis	112
Part 3: Bioanalyzer Analysis	112
<b>Lab 23. cDNA Synthesis and RT-PCR</b>	<b>115</b>
Part 1: cDNA Synthesis	116
Part 2: RT-PCR Setup	117
<b>Lab 24. Continuance of cDNA Synthesis and RT-PCR</b>	<b>121</b>
Part 1: Make an Agarose Gel	121
Part 2: Gel Electrophoresis of RT-PCR Products	121
<b>APPENDIXES</b>	<b>123</b>
Appendix 1. Making agarose gels	123
Appendix 2. Making Escherichia coli growth media	124
Appendix 3. Plant DNA Extraction Buffer	125
Appendix 4. Making Protein Gels	126
Appendix 5. Making Yeast Media	129
Appendix 6. Yeast Transformation	130
Appendix 7. To Do List for Teaching Assistants	131
Appendix 8. Lab Supply	139
Appendix 9. Unix-Linux Command Reference	147
Appendix 10. Vim Cheat Sheet	149
Appendix 11. Perl Cheat Sheet	153
Appendix 12. R Cheat Sheet	155



## Preface

Since the invention of next generation sequencing technology, biology has become the No. 1 scientific field that rapidly produces data information in an exponential growth pace. The ever-increasing accumulation of massive genomic data has been challenging our biological research in many areas, from molecular and cellular biology, biochemistry, systematics, evolutionary biology, ecology genomics, clinical genomics, to high throughput phenomics. The incapability of manipulating enormous genomic data hinders the discovery of novel biological mechanisms and development of new hypotheses. This adds more learning challenges than ever to the new generation of graduate students and undergraduates in biology. Many are struggling with gigabyte and even terabyte-level sequencing data with no clue. Some with high motivation may spend a lot of time to take additional computational programming classes and bioinformatics courses. However, our students' time is very precious and, to many of them, computational programming or genomics is not their major or career path. Although massive genomic data could be helpful, many students **DO NOT** have a strong background to take additional computer science courses. They **do not have time** either because their thesis research requires them to spend enormous effort on pipetting in test tubes rather than typing on keyboards. In opposite, those who have gained a lot of computational programming skills could be also lost in sequencing data without the capability in finding meaningful biological information. Only verified in life system would any information drawn from the genomic data be conclusive. Obviously, pipetting is still the hardware in biological sciences. Therefore, it would be extremely important and also challenging to develop a course that can provide a concise but comprehensive genomics toolkit for biology and bioinformatics students that can allow them to be free of the stresses of data analysis and pipetting.

Using a **project-oriented design**, I developed this book for teaching class PBIO/MCB 5280 (4280) at Ohio University. The book begins with a comparative genomics study of a gene superfamily among 10 genomes through an *in silico* deep sequence data analysis, which will help students to learn how to develop biological hypothesis from big data analysis. To test their hypotheses, the students will experience a series of hands on wet-bench genomics techniques, including molecular cloning, genomic DNA analysis, protein biochemistry, and RNA expression analysis at both single gene and transcriptomic levels. Upon finishing the book, the students are expected to acquire a comprehensive set of skills in functional genomics studies.

The audience for this course could include graduate students and up-level undergraduate students in any biological disciplines, preferentially in molecular and cellular biology. Students in bioinformatics and genomics field are also encouraged. Although this course does not require programming experience, molecular biology or related biological courses are an essential prerequisite. All the program code in the book can be found and downloaded from the companion website for this book at <https://github.com/hua-lab/PBIO45280.git>.



## Acknowledgment

I would like to thank my faculty mentor, Dr. Sarah Wyatt, for her support and enormous encouragement in the process of developing this course. Thanks also to Drs. Morgan Vis-Chiasson and Robert Colvin for their support in developing this course when they served as the Chair of Department of Environmental and Plant Biology and the director of Interdisciplinary Program in Molecular and Cellular Biology, respectively, at Ohio University.

I am deeply thankful to both the Department of Environmental and Plant Biology and the Interdisciplinary Program in Molecular and Cellular Biology for the continuous support to this course. Many thanks also go to the comments and feedback from previous graduate and undergraduate students who had taken this class when it was in the developing stage. The accomplishment of this book has also benefited from two previous graduate students, Emily Keil and Colin P.S. Kruse, for their teaching assistance. Part of this work is supported by Ohio University Startup Grant (SU-1007172) and a National Science Foundation CAREER award (MCB-1750361).

Finally, I'd like to thank my mom and my family members in China, my wife Hongying, and my son Kyle for their support and encouragement.



## Lab 1: Open Source Programming 1 (2 hours)

### Introduction

The ever-increasing amount of data has overwhelmed many biologists when applying them for making conclusions, uncovering biological patterns/mechanisms, or developing new hypotheses. Many of these data can be retrieved from open-source databases. For example, DNA Data Bank of Japan, Mishima, Japan (DDBJ), EMBL-EBI, European Nucleotide Archive, Cambridge, UK (EMBL), and GenBank, NCBI, Bethesda, MD, USA (GenBank) were the earliest three large international biological sequence data banks. In addition, many other open source sequence databases have also rapidly increased for specific biological research purposes, such as the Universal Protein Resource (UniProt), the Pfam protein family databases, the Protein Data Bank archive (PDB). Species-specific databases, such as WormBase, FlyBase, Human Protein Reference Database, The Arabidopsis Information Resource, Phytozome, etc., have been also available to the public. According to the statistical data released from the NCBI, GenBank has collected >260 billion bases and the whole genome sequencing projects have accumulated more than 3 trillion bases as of Aug, 2018. The biology has become the field that generates information the fastest in the world.

To face these enormous data resources, what can you do? Will you be able to retrieve the information you need for your experimental design? Are you also generating a large amount of sequence information? How does your big data experiment lead to the biological meaningful conclusion? All of these questions require you to know how to handle the endless amount of biological information. To solve this problem, the powerful computer programming approaches have been introduced in the 1960s with the efforts of Margaret O. Dayhoff, Walter M. Fitch, Russell F. Doolittle and others. Since the invention of next-generation sequencing technology early this century, a fully developed discipline, called bioinformatics, has been matured and exponentially expanded.

Now, bioinformatics has become an important part of many areas in biology. Bioinformatics programming will lead you to easily extract useful results from a large amount of repeats generated through image and signal processing. Without bioinformatics techniques, you will completely lose in the genome sequences and will never be able to find genes and mutations in which you are interested. Bioinformatics modeling can also help you generate new hypothesis quickly by analyzing gene expression regulation and protein-protein interactions. In addition, bioinformatics has also played a key role in text mining of biological literature and development of biological and gene ontologies. In structural biology, it aids in simulation and modeling of DNA, RNA, proteins as well as biomolecular interactions.

It is not surprising that bioinformatics is an interdisciplinary field that develops methods and computer programming tools for understanding biological questions. Without biological relevance, any sequencing data are meaningless. Therefore, as an interdisciplinary field of science, bioinformatics integrates the knowledge from computer science, statistics, mathematics, physics, engineering and biology. It is a rapid approach applying computation to analyze biological queries using mathematical and statistical techniques. Common uses of bioinformatics include the identification of candidate genes and mutations (e.g., SNPs) in genomics. Often, such identification is aiming to better understanding the genetic basis of diseases, evolutionary adaptations, desirable traits in agriculture, or variations between populations.

## Lab 1: Open Source Programming 1 (2 hours)

---

Open-source software (OSS) is computer software with its source code made available for public use under a license provided by the copyright holder. It may also be developed collaboratively through public contributions characteristic of open-source development. Nearly all bioinformatics programs are written using open-source software programming, making the growth of this field even more rapidly and diverse. You may find many open-source resources of bioinformatics programs, such as Bioinformatics.Org, SourceForge.net, GitHub, and many bioinformatics and genomics journals.

### **Objectives**

1. Be able to list 5 types of biological databases and use 10 UNIX commands
2. Be able to locate, copy, move files between server and client computers using commands

### **Software**

CentOS  
Putty on Windows  
Terminal on Mac

### **Equipment**

Dell PowerEdge R710 Server X5670 2.93GHz 24-Cores 64GB 9-TB (RAID 5)  
Personal Computers

### **Procedure**

**Part 1: General UNIX Commands (In this manual, terminal commands are italicized and the function of each command is described in the parenthesis with plain font) (60 min)**

1. SSH to your user account (ID: your OU ID, PW:PBIO5280) through Putty (Win) or Terminal (Mac)

*ssh -X your\_OU\_ID@host*  
# host is the IP address of the server

2. Lists the files and directories in the folder when you log in (home folder)

*ls -l*  
# lists your files in 'long format'

*ls -all*  
# lists all files, including hidden files

3. Make a new directory

*mkdir programs*  
# make a new folder, named *programs*, for compiling local programs

4. Path of a file

*pwd*



## Lab 1: Open Source Programming 1 (2 hours)

---

# tells you where you currently are

5. Change directory

```
cd dirname  
cd
```

# only `cd`, go to your home directory

6. Copy a file from your personal computer (client) to a folder of your account on the server (terminal of your MAC or WinSCP of Windows)

```
scp path_to_file_in_your_computer your\_OU\_ID@host:/path to the directory
```

7. Copy a file

```
cp path_to_file1 path-to_file2
```

8. Change the file/folder name

```
mv old_file new_file
```

9. Delete a file (be careful, you cannot undo any commands in the terminal!)

```
rm ./old_file
```

10. Uncompress a file

```
gunzip filename.gz  
tar -xvzf filename.tar.gz  
bzip2 -d filename.bz2
```

11. Find the user name

```
whoami
```

### Part 2: A Real-Time Task (60 min)

Use your own computer (client) to download BLAST standalone program from NCBI and copy it into the program folder of your user account in the server.

### Homework

Practice and get familiar with the general Unix commands

**(Notes)**

## Lab 2: Open Source Programming 2 (2 hours)

### Introduction

See Lab 1

### Objectives

1. Be able to list 3 paths where executable variables are saved
2. Be able to use *vim* to write a *.bash\_profile* file

### Software and databases

CentOS  
Putty on Windows  
Terminal on Mac

### Equipment

Dell PowerEdge R710 Server X5670 2.93GHz 24-Cores 64GB 9-TB (RAID 5)  
Personal Computers

### Procedure

#### **Part 1: Compile your BLAST standalone program in your account on the server (30 min)**

1. Decompress the file and find where the BLAST executable functions are using the commands in Part 1.
2. Once you find where BLAST executable functions are, in the same directory, type the following commands and record the results.

- 1) *./blastp*
- 2) *./blastn*
- 3) *./blastx*
- 4) *./tblastn*
- 6) *./tblastx*

3. Go to your home folder, and type the same commands and record the results by taking screen pictures. What is the difference between Step 2 and Step 3? Why?

#### **Part 2: Executable functions of Open Source Programming and *vim* editor (45 min)**

PATH is an environmental variable in open-source operating systems that tells the shell which directories to find executable functions (i.e., ready-to-run programs) in response to commands. It increases both the convenience and the safety and is widely considered to be the single most important environmental variable. To avoid the conflicts with other users on the server, you have to build your own PATH and tell the server the commands you entered. This can be done by the following codes:

```
PATH=$PATH:path_to_the_directory_of_your_executable_files
export PATH
```

## Lab 2: Open Source Programming 2 (2 hours)

---

Unfortunately, you have to do this every time you log in. To make this more convenient, your PATH variable can be made permanently by adding it to your *.bash\_profile* file. *.bash\_profile* is a hidden file in each user's home directory that defines any specific environmental variables and startup programs for that user. A hidden file is a file whose name begins with a dot (i.e., a period) and which is normally not visible.

1. Log into your user account

2. Go to your home directory by type the command below

```
cd
```

3. Find whether you have a *.bash\_profile* file in your home folder using one of the command in Part 1.

4. Use *vim* to edit a program

5. Open *.bash\_profile* via *vim*

```
vim .bash_profile
```

6. Enter the editing model of *vim* by typing

```
i
```

7. If the *.bash\_profile* does not exist, enter the following code to establish the head lines of *.bash\_profile*

```
# .bash_profile
# get the aliases and functions
if [ -f ~/.basgrc ]; then
. ~/.bashrc
fi
#user specific environment and startup programs
```

8. Add a PATH to your *.bash\_profile*

```
PATH=$PATH:path_to_the_directory_of_your_executable_files
export PATH
```

9. Save your *.bash\_profile* with the commands below

```
esc
:wq
```

10. Log out your account

```
exit
```

11. Log into your account

## Lab 2: Open Source Programming 2 (2 hours)

---

*ssh -X your\_OU\_ID@host*

12. Go to your home folder and record the results by typing the following commands

- 1) *blastp*
- 2) *blastn*
- 3) *blastx*
- 4) *tblastn*
- 6) *tblastx*

13. Go to any another directory and record the results by typing the same commands in Step 12.

### **Homework (20 points)**

1. Download the right MUSCLE program at <http://www.drive5.com/muscle/downloads.htm> and compile it into your user account on the server. Run the program and print out the output.

**(Notes)**

## Lab 3: BLAST (3 hours)

### Introduction

BLAST is an abbreviation of Basic Local Alignment Search Tool. As the name stands for, BLAST is a sequence alignment tool for aligning one target sequence, termed “query”, with another, called “subject”. However, unlike to many other sequence alignment tools, in which both query and subject sequences are known, the subject sequences of BLAST are hidden, and sometimes are not existed, within a set of sequences, called database. Therefore, the process of BLAST is often named as BLAST search because the subject sequences need to be found first before the generation of a pairwise local alignment.

The initial BLAST was designed to search a subject sequence in a protein sequence database that is similar to a protein query sequence. Later, the BLAST program has been expanded for nucleotide to nucleotide sequence search and even for cross-comparisons between nucleotide and protein sequences. These different searches are now carried out separately by different programs, such as BLASTP, BLASTN, BLASTX, tBLASTN and tBLASTX, that are integrated in a BLAST suite package available as standalone and web versions on the National Center for Biotechnology Information ([www.ncbi.nlm.nih.gov](http://www.ncbi.nlm.nih.gov)). You should know how to compile the standalone BLAST package that we have discussed in Lab 2.

### 1.1 FASTA Sequence Format

Bioinformatics often deals with large datasets. If everyone were to apply his/her own style to write sequences, it would be very tedious and might be not possible to compare different sequence databases since these various styles adds more complications. Therefore, the bioinformatics society has developed several standard formats to write sequences or expression data. One important sequence format is called FASTA. A FASTA-formatted sequence is basically separated by two parts, a definition part and a sequence part. The definition part is one single line beginning with a “>” symbol followed by an identifier and sometimes with other descriptive information of the sequence. The sequence part could be one single line or multiple lines. If the sequence is written in multiple lines, all except the last line should contain the same number of characters. All BLAST sequences are written in a FASTA format.

Although the sequences in a BLAST database are written in a FASTA format, a searchable database needs to be reformatted to produce a mixture of binary- and ascii-encoded files that contain the sequences and indexing information used during the BLAST search.

### 1.2. Scoring of Alignments and Substitution Matrices

Similar to pairwise local sequence alignment, in a BLAST alignment every letter in one sequence is paired in position with a letter that matches or a gap if there is no match in the other sequence. To determine whether two letters match each other, a value to each aligned pair of letters is given. As such, an alignment score can be computed in theory over the length of an alignment of any two sequences. However, only alignments with good scores would be meaningful in biology.

Due to different biochemical properties between nucleotides and amino acids, the values assigned to pairs of nucleotides and amino acids are different. Scores for all possible amino acid pairs are given in a “substitution matrix”, which was developed based on the frequencies of amino acid substitutions in related protein sequences. One of the most commonly used series

of amino acid substitution matrix is called “blosum62” matrix. BLOSUM62 was created from protein sequences showing approximately 62% identity. This is also the matrix that has been adapted in BLAST protein sequence alignment. To align nucleotide sequences, BLAST assigns +2 for aligned pairs of identical letters and -3 for each nonidentical aligned pair. During sequence alignment, to avoid too many negative penalty scores generated by unidentical and/or unreplaceable pairs of letters, a gap can be introduced with a negative “gap-creation” penalty. However, an extension of a preexisting gap reduces a lesser penalty.

### 1.3. Algorithm

BLAST algorithm can be simplified as “rapidly word search” and “word length extension”. It first breaks down a “query” sequence into a number of “word” strings with a certain length, each of which is indexed with their starting position in the query. In general, three amino acid and 11 nucleotide letters are used as a word size for protein and nucleotide sequences, respectively.

BLAST then searches for matches of the query “words” to subject sequences within a database. For nucleotide-to-nucleotide sequence searches, a 100% match is required; for protein-to-protein sequence searches, the matches are determined based on whether an amino acid substitution score exceeds a user-defined threshold. The query words from the good matches are used to extend their sizes both forward and backward by adding new letters. This extension continues until the alignment score drops by a critical value, called “dropoff” (default = 0), owing to mismatch penalties. The resulting pair of query and subject sequences is called High Scoring Pair (HSP). If two HSPs are away in a certain region (default < 40 letters) in a query sequence, they are combined along with the middle region through a full dynamic sequence alignment analysis between the query and the subject sequences to yield a new HSP. As a consequence, one hit could have one or multiple HSPs and one query sequence could result in one or multiple hits in a BLAST search.

### 1.4. Statistical Significance

The score of each alignment computed from BLAST is also known as “bit score”, and is assigned with a statistical value, called the “Expect Value” or “e-value”. The “e-value” is the number of times that an alignment would be expected to occur by chance to be as good or better than the one found by BLAST, given the size of the database searched. In general, an e-value below  $1e-3$  ( $1e-5$  in many cases) is commonly used to produce BLAST alignment with high quality.

### **Objectives**

1. Be capable of applying a short PERL script (10-20 statements) to acquire output.
2. Be capable of performing a standalone BLAST search;
3. Be capable of identifying and retrieving the hit list of a BLAST search;

### **Software and databases**

CentOS

Putty on Windows

Terminal on Mac

- 10 Proteomes: 1) *Homo sapiens* (hs)  
2) *Mus musculus* (mm)  
3) *Drosophila melanogaster* (dm)



## Lab 3: BLAST (3 hours)

---

- 4) *Saccharomyces cerevisiae* (sc)
- 5) *Caenorhabditis elegans* (ce)
- 6) *Chlamydomonas reinhardtii* (cr)
- 7) *Physcomitrella patens* (pp)
- 8) *Escherichia coli* (ec)
- 9) *Arabidopsis thaliana* (at)
- 10) *Oryza sativa* (os)

SKP1 Seed Database

### **Equipment**

Dell PowerEdge R710 Server X5670 2.93GHz 24-Cores 64GB 9TB  
Personal Computers

### **Procedure**

#### **Part 1: Make a BLASTP database (30 min)**

1. Log into your user account on server.
2. Make a directory named ***proteome\_databases*** (be sure to use the underscore line, no space, no hyphen in programming). If you don't know how to make a new directory, it means that you have not done after class practice. You may review Lab 1 to find the solution.
3. Copy 10 proteome files into your *proteome\_databases* directory from the target directory at */usr/local/databases/proteomes/*. Review Lab 1 to finish the code.
4. Make BLASTP databases for each proteome by the following code.

```
makeblastdb -in proteome_file_name -dbtype prot -out proteome_file_name_db
```

*-in* points one of the proteome files you just copied; *-dbtype* could be *prot* (for proteins) or *nucl* for nucleotide sequences; *-out* writes the output of the blastdb file into the name you give. I usually just attach *db* to a proteome file.

#### **Part 2: BLASTP search (1 hour)**

1. Make a new directory, named ***blastp\_queries***, in your home folder.
2. Use your local computer to find the SKP1 seed sequences at <http://pfam.xfam.org/family/SKP1#tabview=tab3>.
3. Download SKP1 seed sequences. Be sure to use a FASTA format as described in the Introduction (1.1).
4. Copy your SKP1 seed-sequence file into the ***blastp\_queries*** in your user account on the server.
5. Go back to your home folder. If you don't know how, review Lab 1.
6. Make a new directory, named ***blastp\_outputs***, in your home folder.

7. Enter *blastp\_outputs*. Again, review Lab 1 to do so if you don't know how.

8. Use your SKP1 seed sequences as query to search one of the 10 BLASTP database you created in Part 1 by the following code.

```
blastp -in your_SKP1_file_with_path -db one_of_the_10_BLASTP_database_with_path  
-out your_SKP1_BLASTP_output_file_name
```

Make sure each file has a clear path that *blastp* can read in and write out. If you don't know the format of this code, type *blastp -h* for the detail of each function.

9. Read and record part of the output of BLASTP by typing

```
more your_SKP1_BLASTP_output_file_name
```

*more* is a reading tool in terminal. Use *man more* to read the function of *more*. Type *q* to close *man more*.

10. Use your SKP1 seed sequences as query to search the same BLASTP database as in Step 7 by a modified code as follow.

```
blastp -in your_SKP1_file_with_path -db one_of_the_10_BLASTP_database_with_path  
-outfmt 6 evalute 1e-5 -out your_SKP1_BLASTP_output_file_name
```

Make sure each file has a clear path that *blastp* can read in and write out. If you don't know the format of this code, type *blastp -h* for the detail of each function.

11. Use the same code as in Step 8 to read and record part of the new output of BLASTP.

12. Use the same code as in Step 9 to finish BLASTP your SKP1 sequences as query against all the remaining 9 BLASTP databases. Make sure you save the output files.

### Part 3: *hallo.pl* (30 min)

This may be the first Perl code for many of you. From now on, you will learn some basic Perl codes one by one. Each Perl syntax and statement is described within the code following a # pound sign. Perl scripts are often written in a file with a extension file name ".pl". The main body of the script contains 3 types of variables, *string* (symbol: \$), *array* (symbol: @), and *hash* (symbol: %). A string is basically a line of characters, symbols, numbers, and any mixtures. An array is a list of strings. A hash is a pairwise relationship of strings within two arrays, *keys* and *values*. The strings within *keys* are unique but those in *values* can be redundant. This is much like the user IDs and passwords assigned in this class. Your user IDs are unique while you all use the same password to log into our class server.

1: Go to your home folder and use vim to write a Perl script, named the script "*hello.pl*". Review Lab 2 for how to use vim to write a file by the command as follow.

```
vim hello.pl
```

**Code for *hallo.pl* (Perl codes are italicized and highlighted with blue color in this manual)**

```
#!/usr/bin/perl
```

# Tells the computer where the perl executable variable is.

```
use warnings;
```

# This will allow the Perl to print out any errors in your Perl code. All Perl statement should end up with semi colon symbol except the first sentence. All pound # signs except the one in the first sentence are used for comments (not executable) in most programming languages, including Perl.

```
use strict;
```

# This restricts all variables in the code should be defined by *my* when they first appear in your code which can avoid the mixture of different variables with the same name.

```
my $hello="Hello World";
```

# A string is much of a line and it is marked with a dollar \$ sign. Again, any variable should be defined with *my* when it appears at the first time in the code.

```
print $hello,"n";
```

# Here, you don't give *my* to *\$hello* because it has been defined with *my* as a sentence string of *Hello World*. Note that the sentence is quoted by a double prime symbol.

```
exit;
```

# *exit* turns off the program.

2. Save the file by typing

```
Esc  
:wq
```

3. Execute the Perl file you wrote by typing the code below and record what happens

```
perl hello.pl
```

A Perl script is called by a *perl* command in the terminal.

4. Execute the same Perl file by typing the code below and compare the difference with the output from Step 3

```
perl hello.pl>hello.txt
```

5. Use the knowledge from Lab 1 to find where the file *hello.txt* is. Read the file and compare the result with the output from Step 3.

**Part 4: *blast\_parse.pl* (1 hour)**

Because BLAST searches the database by indexing all character strings of a certain length within the “query” by their starting position in the query. A target, named “hit”, might be found several times with different strings, called High-scoring Segment Pair (HSP)s, from one query sequence. In addition, in your query file, because there are multiple sequences (often it could be thousands of query sequences which can never been done without programming!), different query sequences might hit the same subject sequence. Therefore, in your BLASTP search output, the hit IDs are heavily redundant. You want to parse your BLASTP output file to remove all redundant IDs.

Here we will also introduce the remaining two Perl variables. An *array* is a collection of *strings*. *blast\_parse.pl* first reads the hit result from the BLASTP output file one by one, save all hit IDs as *strings*, including redundant ones, into an array.

In Perl, *hash* is a variable pointing the pairwise relationship of two arrays, named *key* and *value*. If we take all your IDs as the *key* array, we can use your unique *key* variables (IDs) to log into our server, which has the same IP address and can be assigned as the *value* variables. Therefore, applying the unique function of *key* variables in a *hash*, we will remove all redundant hit IDs.

1. Make a new directory, named ***blastp\_parse\_results***, in your home folder.
2. Go to ***.blastp\_parse\_results*** and use *vim* to write a Perl script, named the script “*blast\_parse.pl*”. Review Part 3 for how.

### Code for *blast\_parse.pl*

```
#!/usr/bin/perl
use strict;
use warnings;

# You should be able to the functions of these three statements now.

# Review Part 2 for the –outfmt 6 function, which writes out the BLASTP results as a tab file.
# The first column is the query IDs and the second column shows the hit IDs.

# Below, we will ask Perl to open a blastp output file, find all hit ids (the second column), and
# save the hit ids into an array variable, @blastp_parse_ids.

    open BLASTP_OUT, "< your_blastp_output_file ";

# open is a Perl syntax to read in a file

    my @blastp_parse_ids;

# my defines a new array

    while(my $parse=<BLASTP_OUT>){

# my defines a new string. While is a Perl syntax to read each line in the file one by one
```

```
chomp $parse;
```

# *chomp* removes the hidden new line symbol “\n” or its cousins linefeeds and carriage returns. Keep in mind, in programming, any symbols, including those hidden ones could make you very frustrated because they are all meaningful and should be carefully treated.

```
my @pares=split /\t,$parse;
```

# This is a useful statement which converts a *string*, *\$parse*, into, an *array* by splitting the *string* into pieces at each tab symbol, \t. Remember, the BLASTP output file is a tab file.

```
my $hit_id=$pares[1];
```

# Again, the hit ids are in the second column of each line. Because in most programming languages, counting begins 0. Thus, the second is counted as 1.

```
push(@blastp_parse_ids,$hit);
```

# Save all hit IDs into the array and many are duplicated

```
}
```

```
close BLASTP;
```

# After finish reading the file, the file should be closed.

# The codes below will apply the unique variable property of the *key* variables in a *hash* to retrieve unique HIT IDS

```
my %hash;
```

# my define a new hash

```
foreach my $hit_id(@blastp_parse_ids){
```

# *foreach* is a syntax to read each *string* of an array one by one

```
if (!$hash{$hit_id}) {
```

```
    push (@unique, $hit_id);  
    $hash{$hit_id} = "true";
```

```
}
```

# If a string (*key*) is not assigned (the syntax “!” means “no” in Perl) to a *value* (here is an artificial string “true”), we will assign it to “true”. *\$hash{\$hit\_id} = “true”* is a structure to write a *hash*. Here, *\$hash* calls *%hash* in and asks *%hash* to assign a “true” *value* to the *key* variable, *\$hit\_id*. However, if a *key* (*\$hit\_id* here) variable has already been assigned as a “true” *value* in a previous string read in by *foreach*, it will be memorized in the hash *%hash* and will be ignored by the *if* syntax. Again, “!” means “no”.

```
}
```

### Lab 3: BLAST (3 hours)

---

# Each loop needs to be separated by brackets. Thus the functions in the loop will finish first.

```
my @unique_hit_ids=keys %hash;
```

# keys reads all key variables of a hash and saves it into an array.

```
foreach my $unique_id(@unique_hit_ids){
```

```
    print $unique_id, "\n";
```

```
}
```

#print out the result and each id will be printed in one line

```
exit;
```

# exit turns off the program

#### **Homework (20 points)**

1. Use *blast\_parse.pl* to parse all your 10 BLASTP results and save the unique IDs in separated files under a ***./blastp\_parse\_results*** directory in your account on the class server for Lab 4. For each *blast\_parse* output file, you should know which proteome, i.e. species, was used as the database in your BLASTP search. I would recommend to add the initial of the species name into your output file name. Print out one page for each output and write down how many SKP1 proteins you discovered in each proteome.

**(Notes)**

**(Notes)**



## **Lab 4: Bioperl 1 (2 hours)**

### **Introduction**

Bioperl is an open-source project that are contributed and written by many pioneer bioinformaticians and now it is still constantly being improved by a number of volunteers in the field. It is a collection of >500 Perl modules that facilitate the development of Perl scripts for bioinformatics applications. The Bioperl modules cover various areas of bioinformatics, including sequence similarity search, sequence retrieval and manipulation, sequence alignment, sequence database transformation, gene annotation, phylogenetic analysis, population genetics, etc. It is not surprising that Bioperl has played an integral role in the Human Genome Project and the most recently Human ENCODE project.

Tim Hubbard and Jong Bhak at Medical Research Council (United Kingdom) MRC Centre Cambridge developed the first set of Perl codes of Bioperl. MRC Centre was one of the hubs and birth places of modern bioinformatics as it had large amount of DNA sequences and 3D protein structures and it was the place where the first genome sequencing was carried out by Fred Sanger. The name birth of Bioperl took a long period through the debate on whether Perl is superior to C for bioinformatics at the MRC Centre among its pioneer developers, including Jong Bhak and Steve Brenner, who are the Ph.D. student of Tim Hubbard and Cyrus Chothia, respectively. However, later another Cambridge student, Ewan Birney, became the main contributor for the expansion of Bioperl after another long debate on if Perl was superior to C for bioinformatics. Ewan and many other people were very active in developing Bioperl. Ewan is one of the most well-known bioinformaticians and genomicists in the world. I am a big fan of Ewan. He is one of the founders of the Ensembl genome browser and other databases, and has played a key role in many large-scale genomics projects, notably the sequencing of the Human Genome in 2000 and the human ENCODE project.

The first stable release of Bioperl was on 11 June 2002 and the most recent stable version releases on April 14, 2011 is 1.6.9. Now Bioperl has become an important tool for daily use in bioinformatics field, including the bioinformatics projects carried out in my lab.

### **Objectives:**

1. Be able to retrieve sequences from the database in batch
2. Be able to retrieve sequence ID and other annotation information in batch
3. Be able to manipulate sequences in batch
4. Be able to organize and write out a sequence report in batch

### **Software and databases**

CentOS  
Putty on Windows  
Terminal on Mac  
10 Proteomes  
Your balstp\_parse results  
Bioperl

### **Equipment**

Dell PowerEdge R710 Server X5670 2.93GHz 24-Cores 64GB 9-TB (RAID 5)

### Personal Computers

#### Procedure

##### **Part 1: Write and Format a Fasta Sequence File (*fasta\_format.pl*) (60 min)**

From Lab 2, you may have known that the first line of a Fasta format sequence starts with a ">" symbol followed by the unique sequence ID and often with other descriptive information, such as the function of the corresponding gene, the coordinates of the gene in the genome, etc. After the first line is the sequence. The sequence may be written in many lines or just by a single very long line (sometimes could be thousands of characters), which often is not human readable and may also cause programming error. *fasta\_format.pl* is to standardize the format of any sequence databases.

1. Make a new directory *./formatted\_proteomes* under your home folder.
2. Go to your new *./formatted\_proteomes* directory and use *vim* to write a new perl program and save it as *fasta\_format.pl* (*vim fasta\_format.pl*)

##### **Code for *fasta\_format.pl***

```
#!/usr/bin/perl
use warnings;
use strict;

open FILE,"<one_of_your_selected_proteomes_with_path";
my @file=<FILE>;
close FILE;
```

# Like in *blast\_parse.pl*, we use *open* to read in the content of a file. However, here we do not use *while* loop to read each line. We use an *array* to read in the entire file, including "\n" hidden return symbols. Each line is a *string* of the array.

```
my $file=join ("",@file);
```

# *join* is a Perl syntax which has an opposite function to *split*. Here, it adds all *strings* in the *@file* array into a long *string*.

```
my @fasta_seqs=split /\>/,$file;
```

# Each *fasta* sequence starts with a ">" symbol. Thus, each string in the new *array* *@fasta\_seqs* starts with the identification line followed by sequence lines but without the ">" symbol. .

```
foreach my $fasta_seq(@fasta_seqs){
```

# Again, *foreach* is a syntax to read each *string* of an array one by one

```
my @lines=split /\n/,$fasta_seq;
my $head_line=shift @lines;
```

## Lab 4: Bioperl 1 (2 hours)

---

# Here is a tricky part that is always used in programming. In order to retrieve a specific fragment of a string, you need to look for the pattern. You may use a *regular expression* or like the one here to retrieve the fragment. Here, we know each *fasta* sequence contains at least two lines, the first is the identification line, and the remaining is the sequence. We use a *shift* function to retrieve the first string in the array. *shift* always takes away the first *string* of an *array*.

```
print ">",$head_line,"\n";
```

# We know the first line is the identification of a sequence and we know a *fasta* sequence format begins with ">" symbol.

```
my $seq=join ("",@lines);
```

# In some *fasta* files, its sequence lines may have already been formatted. However, in order to make your own format, we need to remove the original format by joining all sequence lines into one line.

```
foreach my $reformat_line (unpack("(A30)*",$seq) ){  
    print $reformat_line,"\n";  
}
```

# *unpack* is the key syntax to reformat a string and print it out in a fixed length. Here, we unpack the joined one line *\$seq* into lines *\$reformat\_line* with 30 characters (*A30*) per line followed by any remaining characters \* whose length is < 30 characters. We then print each *\$reformat\_line* followed by a hidden "\n";

```
    }  
exit;
```

3. Run *fasta\_format.pl* and use *more* to compare the format differences between your output file and the original proteome file.

### Part 2: Bio::DB::Fasta Bioperl Module (60 min)

Bio::DB::Fasta is a very useful Bioperl module for retrieving and editing *fasta* formatted sequence databases. We will use it to retrieve the number of total annotated protein coding genes, sequence ids, and the length of ONE proteome.

1. Make a new directory ***./proteome\_statistics*** under your home folder.
2. Go to your new ***./proteome\_statistics*** directory and use *vim* to write a new perl program and save it as *biodbfasta.pl*.

#### Code for *biodbfasta\_id\_retrieval.pl*

```
#!/usr/bin/perl  
use warnings;  
use strict;
```

# You should know what these three lines mean. If you don't know, review Lab 2 Part 3.

## Lab 4: Bioperl 1 (2 hours)

---

```
use Bio::DB::Fasta;
```

# This is how we call a Perl module in to your program. How a Perl module is developed edited, and installed is out of the scope of this lab class

```
my $pep_db=Bio::DB::Fasta->new("path_to_your_proteome_file");
```

# This is an index statement that allows *Bio::DB::Fasta* to read in your *fasta*-formatted sequence database and save it into a sting, named *\$pep\_db*. An *arrow* symbol is always used in *Objective-oriented (OO) programming* to call *methods*. *OO programming* is also out of the scope of this lab.

```
my @ids=$pep_db->ids;
```

# Now, you can see how powerful a Bioperl module is. You just need one statement to get all sequence IDs and save them into an *@ids* array.

```
my @sort_ids=sort {$a cmp $b}@ids;
```

# You know what sort means. Here it has the exactly function to *sort* your ids alphabetically by *cmp* function and save the sorted IDs into a new array.

```
my $count=0;
```

# We will use a count function to record how many protein-coding genes that are annotated in a proteome file. Before we start counting, we should define and reset the counting number to 0.

```
foreach my $id(@sort_ids){
```

```
    ++$count;
```

# Here *++\$count* equals *\$count=\$count+1*. You may easily understand the latter. However, in programming, it sometimes adds more jargons into it to shorten the code, very much like many acronyms in biology. You can also use *\$count++* or *\$count += 1* for incrementing counts.

```
my $length=$pep_db->length($id);
```

# This is the embedded function of *Bio::DB::Fasta* module. It calculates the length of the sequence.

```
    print $id,"\t",$count,"\t",$length,"\n";
```

```
}
```

# We use *print* to write out the string variables that are separated by a tab symbol *"\t"*.

```
exit;
```

3. Use *biodbfasta\_id\_retrieval.pl* to get the total number of annotated protein coding genes and compare the size difference of one proteome.

**Homework (10 points)**

1. Calculate and compare the average sequence length of the annotated protein sequences among the 10 proteomes.

**(Notes)**

## Lab 5: Bioperl 2 (3 hours)

### Introduction

See Lab 3

### Objectives:

1. Be able to manipulate sequences in batch
2. Be able to organize and write out a sequence report in batch

### Software and databases

CentOS  
Putty on Windows  
Terminal on Mac  
10 Proteomes  
Your blastp\_parse results  
Bioperl

### Equipment

Dell PowerEdge R710 Server X5670 2.93GHz 24-Cores 64GB 9-TB (RAID 5)  
Personal Computers

### Procedure

#### **Part 1: Use Bio::DB::Fasta to Retrieve the Protein Sequences of Your BLAST Parse Result.**

In Lab 2, we used SKP1 seed sequences as a query and obtained a number of homologous sequences in each of the 10 proteome databases. However, we have only obtained a list of homologous protein IDs so far from each proteome database. We do not know how they are close to each other and how similar they are to a target SKP1 sequence, for example, the Arabidopsis SKP1-LIKE 1 (ASK1, AT1G75950). To address these questions, it is important to obtain the sequence of each homologous protein. We have learned to use Bio::DB::Fasta module to retrieve the sequence IDs and the length of each sequence in Part 2. Indeed, the powerful function of Bio::DB::Fasta module is for sequence retrieval and manipulation. *blast\_parse\_retrieval.pl* will guide you through the process.

1. Make a new directory *./blastp\_parse\_pep* under your home folder.
2. Go to your new *./blastp\_parse\_pep* directory and use *vim* to write a new Perl program and save it as *blast\_parse\_pep\_retrieval.pl*

#### **Code for *blast\_parse\_pep\_retrieval.pl***

```
#!/usr/bin/perl  
use warnings;  
use strict;  
use Bio::DB::Fasta;
```

```
my $pep_db=Bio::DB::Fasta->new("path_to_your_target_proteome_file");  
open BLAST_PARSE,"<one_of_your_blast_parse_file";  
while(my $parse_id=< BLAST_PARSE>){
```

```
    chomp $parse_id;
```

# You should know the functions of all the above statements now

```
    my $pep_obj=$pep_db->get_Seq_by_id($parse_id);
```

# You may have already known that all Bioperl modules are *OO programming*. One function of *OO programming* is to call an *objective*, which is composed of ###. You may just need to know now that in order to retrieve a sequence, we call an objective first using the sequence ID.

```
    my $pep=$pep_obj->seq;
```

# *OO programming* allows to retrieve the sequence of a target ID through its *objective*, *\$seq\_obj*, and a *method*, *seq*.

```
    print ">",$parse_id,"\n";  
    print $pep,"\n";
```

# For each homologous protein, we print out its sequence in a *fasta* format.

```
    }
```

# Use a *while* loop to get all homologous sequence in a targeted proteome database.

```
close BLAST_PARSE;
```

# After finish reading the file, the file should be closed.

```
exit;
```

3. Use ***blast\_parse\_pep\_retrieval.pl*** to retrieve the protein sequences of your BLASTP hits from one proteome

## Part 2: Use Bio::PrimarySeq to Compare the Predicted Transcript Sequences and the Protein Sequences of Your BLAST Parse Result.

The attaining of a protein sequence is not sufficient enough for you to manipulate the corresponding encoding gene. For example, so far we have not discovered the complementary DNA (cDNA) sequences of your BLAST hits yet, which are essential for us to design polymerase chain reaction (PCR) primers to clone them from a cDNA library. In this section, we will use Bio::DB::Fasta to retrieve the cDNAs (often coding sequence) of your BLAST hits from a predicted transcriptome.



In Bioinformatics, it is always a good idea to check the consistency of two different datasets. Here, one is a collection of protein sequences and the other is the corresponding coding sequences. You may ask whether the translation of each coding sequence is exactly the same as the protein sequence that has been retrieved from the originally annotated proteome file. Thus, we will apply another Bioperl module, `Bio::PrimarySeq`, to translate a coding sequence and then compare the result with the original annotation. If the two protein sequences are not the same, it will suggest that either your program has a problem or the original annotation was wrong.

List of transcriptome files (including path):

```
/usr/local/database/transcriptomes/hs_cds.fa
/usr/local/database/transcriptomes/mm_cds.fa
/usr/local/database/transcriptomes/dm_cds.fa
/usr/local/database/transcriptomes/sc_cds.fa
/usr/local/database/transcriptomes/ce_cds.fa
/usr/local/database/transcriptomes/cr_cds.fa
/usr/local/database/transcriptomes/pp_cds.fa
/usr/local/database/transcriptomes/ec_cds.fa
/usr/local/database/transcriptomes/at_cds.fa
/usr/local/database/transcriptomes/os_cds.fa
```

1. Make a new directory **`./blastp_parse_cds`** under your home folder.
2. Go to your new **`./blastp_parse_cds`** directory and use `vim` to write a new Perl program and save it as **`blast_parse_cds_retrieval.pl`**

**Code for `blast_parse_cds_retrieval.pl`**

```
#!/usr/bin/perl
use warnings;
use strict;
use Bio::DB::Fasta;
use Bio::PrimarySeq;
```

# Any new module should be incorporated into your program by a `use` function.

```
my $pep_db=Bio::DB::Fasta->new("path_to_your_target_proteome_file");
my $cds_db=Bio::DB::Fasta->new("path_to_your_target_transcriptome_file");

open BLAST_PARSE,"<one_of_your_blast_parse_file";

while(my $parse_id=< BLAST_PARSE>){

    chomp $parse_id;

    my $pep_obj=$pep_db->get_Seq_by_id($parse_id);
    my $pep=$pep_obj->seq;

    my $cds_obj=$cds_db->get_Seq_by_id($parse_id);
```

```
my $cds=$cds_obj->seq;
```

# Similar to retrieving a protein sequence, we use Bio::DB::Fasta to get a CDS for a target hit.

```
my $translate_obj=Bio::PrimarySeq->new(  
    -seq=>$cds  
);
```

# This is a typical structure to setup an objective in Object-oriented (OO) programming. You may just remember this structure for how to setup an objective for *Bio::PrimarySeq* module

```
my $translate_pep=$translate_obj->translate->seq();
```

# Another OO programming for introducing two methods, translate and seq, to get a translated peptide sequence from a CDS.

```
die $parse_id unless($pep eq $translate_pep);
```

# Here, we introduced three Perl syntaxes. As the name states, *die* means terminate the program. However, it also has a similar function to *print*. In this statement, it will *print* out *\$parse\_id* if *\$pep* is not the same as *\$translate\_pep*. In other word, it will not stop and *print* out *\$parse\_id* if *\$pep* is the same (eq) as *\$translate\_pep*. This is the proofread function often used in programming.

```
print ">",$parse_id,"\n";  
print $cds,"\n";
```

# print out the CDS if *\$cds* passes the proof.

```
}
```

```
close BLAST_PARSE;
```

# After finish reading the file, the file should be closed.

```
exit;
```

3. Use ***blast\_parse\_cds\_retrieval.pl*** to retrieve the coding sequences of your BLASTP hits from one transcriptome.

### **Homework (30 points, 10 points / question)**

1. Use ***blast\_parse\_pep\_retrieval.pl*** to retrieve the protein sequences of your BLASTP hits from LAB 3. Save the output files into your ***./blastp\_parse\_pep*** directory in your account on the class server for Lab 5.

2. Use ***blast\_parse\_cds\_retrieval.pl*** to retrieve the coding sequence of your BLASTP hits from LAB 3. Save the output files into your ***./blastp\_parse\_cds*** directory in your account on the class server for Lab 5.
3. Are the coding sequences of *SKP1* genes consistent with the protein sequences? Write a short Perl script to demonstrate your approach for this comparison.

**(Notes)**

## Lab 6: Molecular Phylogenetics (2 hr)

### **Introduction**

One important biological phenomenon is change of the heritable characteristics of populations over generations. This phenomenon is termed evolution. There are two fundamental questions in evolution: 1) how are the organisms related to each other through evolution? and 2) in which mechanisms do they evolve?. Similar to all other disciplines in life sciences, studies of evolution have been extraordinarily improved by the discovery of DNA structure, the growth of molecular biology, and the advance of genome sequencing technologies. Now, nearly all biologists apply the molecular evidence of sequences to address the two aforementioned questions. The interpretive framework, the principles, and the analytical methods of evolutionary biology has become critically important in all biological research fields, not only at the molecular level, such as molecular, genomic, cell, and developmental biology, but also at the organismal level, including paleobiology, ecology, functional morphology and physiology.

Another important factor that promoted the evolutionary studies is the advance in statistical methods and computational powers for data analysis. In this lab, we will use two programs, MUSCLE and Phylip, to analyze the relationship (phylogeny) of *SKP1* genes based on their encoded protein sequences obtained from your previous 4 labs. We will use your discovery to end up our first section, Bioinformatics in Genomic Studies. Certainly, I cannot incorporate many bioinformatics approaches in this section. I hope these 5 labs will guide you into the field of bioinformatics and help you recognize the power of computational biology in all disciplines of life sciences and the rapid evolving process in this field.

### **Objectives:**

1. Be able to perform multiple sequence alignment locally
2. Be able to program molecular phylogenetic analysis

### **Software and databases**

CentOS  
Putty on Windows  
Terminal on Mac  
10 Proteomes  
Your balstp\_parse results  
MUSCLE  
Phylip

### **Equipment**

Dell PowerEdge R710 Server X5670 2.93GHz 24-Cores 64GB 9-TB (RAID 5)  
Personal Computers

### **Procedure**

#### **Part 1: Multiple Sequence Alignment**

MUSCLE is one of the best-performing multiple sequence alignment programs considering both accuracy and speed. In Lab 1, you have download and compile MUSCLE locally in your user

## Lab 6: Molecular Phylogenetics (2 hr)

---

account on the class server. Before we start the data analysis, you need to make sure the program functions properly.

1. Go to your home folder and make a new directory, ***./skp1\_phylogeny***
2. Enter ***./skp1\_phylogeny*** and type the following code to learn how MUSCLE works.

```
muscle
```

If you get an error message, it will suggest that you have not compiled your program correctly. Solve this problem with the TA's help until you get a message similar to follows.

```
MUSCLE v3.8.31 by Robert C. Edgar
```

```
http://www.drive5.com/muscle
```

```
This software is donated to the public domain.
```

```
Please cite: Edgar, R.C. Nucleic Acids Res 32(5), 1792-97.
```

```
Basic usage
```

```
muscle -in <inputfile> -out <outputfile>
```

```
Common options (for a complete list please see the User Guide):
```

```
-in <inputfile>   Input file in FASTA format (default stdin)  
-out <outputfile> Output alignment in FASTA format (default stdout)  
-diags           Find diagonals (faster for similar sequences)  
-maxiters <n>     Maximum number of iterations (integer, default 16)  
-maxhours <h>    Maximum time to iterate in hours (default no limit)  
-html            Write output in HTML format (default FASTA)  
-msf            Write output in GCG MSF format (default FASTA)  
-clw            Write output in CLUSTALW format (default FASTA)  
-clwstrict      As -clw, with 'CLUSTAL W (1.81)' header  
-log[a] <logfile> Log to file (append if -loga, overwrite if -log)  
-quiet          Do not write progress messages to stderr  
-version        Display version information and exit
```

```
Without refinement (very fast, avg accuracy similar to T-Coffee): -maxiters 2
```

```
Fastest possible (amino acids): -maxiters 1 -diags -sv -distance1 kbit20_3
```

```
Fastest possible (nucleotides): -maxiters 1 -diags
```

2. We will use the basic function of MUSCLE to align your SKP1 protein sequences. But you may wonder whether you will do 10 times of alignment or just once. If you do 10 times, you cannot compare SKP1 sequences between species. Now, we will use a new UNIX command, *cat*, to combine all your 10 SKP1 sequence files into one file. You need to give a clear path to each file.

```
Cat file1 file2 file3 ... file10>skp1_peps_in_10_species.fa
```

3. Do MUSCLE alignment

```
muscle -in skp1_peps_in_10_species.fa -out skp1_peps_in_10_species_muscle.aln
```

4. Visualize your MUSCLE alignment result using another program (graphical user interface (GUI) script), *seqview*, that has been installed on the server. Type *seaview*, a GUI will pop out.

*seaview*

### Part 2: Sequence Alignment Format

You may have noticed that there are many jargons and formats in bioinformatics. Unfortunately, most of the time, you have to understand the mean of each jargon and know how different formats are generated and converted. For example, in this lab, the default output of a MUSCLE alignment is a fasta format. However, in the PHYLIP phylogenetic analysis we need a phylip alignment format as an input file. If you directly take the fasta format file as an input file, it will not work. Here, we use another alignment tool, *t\_coffee*, which is good for the alignment analysis of a small number of sequences, to convert a fasta format alignment to a phylip alignment. Again, *t\_coffee* has been already installed on our server.

```
t_coffee -other_pg seq_reformat -in your_fasta_muscle_alignment.aln -output phylip > infile
```

The input file name of PHYLIP phylogenetic analysis is always named as *infile*.

### Part 3: PHYLIP Phylogenetic Analysis

PHYLIP package has been widely-distributed and cited since it was first introduced in Oct, 1980. It is the oldest phylogeny package that allows to do batch programming analysis. It is the sixth most frequently cited phylogeny package, after MrBayes, PAUP\*, RAXML, PhymI, and MEGA. We will use SEQBOOT, PROTDIST, NEIGHBOR, and CONSENSUS, 4 methods in the PHYLIP package to construct a neighbor-joining phylogenetic tree of SKP1 protein sequence from 10 species. Comparing to other statistical methods, such maximum likelihood used in RAXML and PAUP\* and Bayesian in MrBayes, neighbor-joining method generates a phylogenetic tree the fastest and relatively as accurate as the others.

1. Type *seqboot*, the information below will pop up in the terminal.

Bootstrapping algorithm, version 3.696

Settings for this run:

```
D  Sequence, Morph, Rest., Gene Freqs? Molecular sequences
J  Bootstrap, Jackknife, Permute, Rewrite? Bootstrap
%  Regular or altered sampling fraction? regular
B  Block size for block-bootstrapping? 1 (regular bootstrap)
R  How many replicates? 100
W  Read weights of characters? No
C  Read categories of sites? No
S  Write out data sets or just weights? Data sets
I  Input sequences interleaved? Yes
0  Terminal type (IBM PC, ANSI, none)? ANSI
1  Print out the data at start of run No
2  Print indications of progress of run Yes
```

Y to accept these or type the letter for one to change

There are 4 different statistical methods available in Option J. Bootstrap is the default and often applied. If you want to change the method, type J and select the option you want. We keep Bootstrap in our analysis. The most important number you need to change is the number of bootstraps. Here, we use 10 bootstraps. Type R and enter 10. However, in most

phylogenetic analyses, 1,000 bootstraps are in general needed. Type Y to accept the parameters you selected and keep other parameters in default values. In order to generate the 100 random sequence alignment datasets, in each of which some alignment positions have been randomly switched (sampled), you also need to give the program a random number. In general, a number that can be divided by 4 with remainder 1 will lead the program to calculate the result more quickly. The output file is saved in a file, named *outfile*.

2. As I mentioned early, all PHYLIP methods take *infile* as the input file name. In the second step of PHYLIP phylogenetic analysis, we will calculate the pairwise distances of proteins in each sequence alignment dataset using PROTDIST. Try *protldist* in terminal to see what happens,

3. If you go through the process as instructed in the terminal, you will only calculate the pairwise distances of proteins in the original sequence alignment dataset. Oops, this must be wrong! There will be no statistics you can apply because it does not have repeats.

4. Here is how PHYLIP introduces the random sampling approaches. It applies *seqboot* to generate a set of random sampled sequence alignments through bootstrap. You may think if you use *protldist* to calculate the sequence distance of each protein to others in each bootstrap alignment, you will get a set of distance matrix which will allow you to cluster which protein sequences are more related to each other than others. Now, we will first convert the *outfile* containing 100 bootstrap alignments into *infile* for *protldist*.

```
mv outfile infile
```

You should know this function from Lab 1.

Type *protldist*, the following information will appear.

```
Protein distance algorithm, version 3.696
```

```
Settings for this run:
```

```
P Use JTT, PMB, PAM, Kimura, categories model? Jones-Taylor-Thornton matrix
G Gamma distribution of rates among positions? No
C One category of substitution rates? Yes
W Use weights for positions? No
M Analyze multiple data sets? No
I Input sequences interleaved? Yes
O Terminal type (IBM PC, ANSI)? ANSI
1 Print out the data at start of run No
2 Print indications of progress of run Yes
```

```
Are these settings correct? (type Y or the letter for one to change)
```

As in *seqboot*, the main character you need to modify is M. Type M then enter D (means datasets) and 100 (you have 100 data sets from *seqboot*). Type Y and enter. The program will start to calculate the distance matrix in each alignment dataset..

5. Convert *outfile* to *infile* and type *neighbor* to calculate a neighbor-joining tree from each bootstrap alignment.



6. Use *consensus* to generate a consensus phylogenetic tree and the bootstrap number of two branches, which represents the probability of how they are clustered together, are also calculated.

**Homework (20 points)**

1. Finish the phylogenetic tree with 1,000 bootstraps and copy the tree file (outtree) into your local computer. Use Figtree (<http://tree.bio.ed.ac.uk/software/figtree/>) to edit and print out your tree figure as a PDF file.

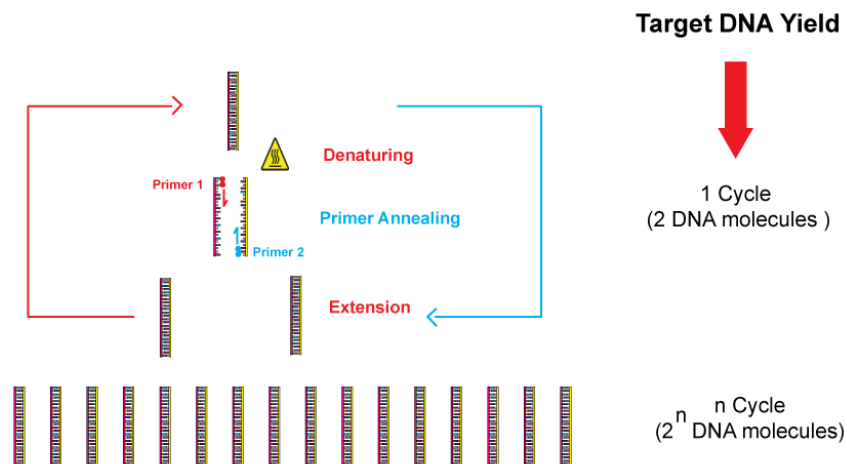
**(Notes)**

## Lab 7: Molecular Cloning (3 hours)

### Introduction

The polymerase chain reaction (PCR) is a revolutionized technique in molecular biology. Since PCR can rapidly increase the number of a target DNA molecule exponentially in a specific manner, its discovery has freed molecular biologists to get enough amount of DNA nearly from any genome within few hours, including those isolated from fossils. Moreover, its theory of DNA amplification has been adopted in many modern molecular biology techniques, such as Sanger and next generation DNA sequencing. In addition, many specific PCR methods, such as inverse PCR, reverse transcription PCR, Thermal Asymmetric Interlaced PCR (TAIL-PCR), etc. have been invented. I believe new PCR-base methods will be further developed in molecular biology.

A typical PCR reaction consists of a template (often DNA), a mixture of four nucleotides that include dATP, dTTP, dGTP and dCTP, reaction buffer, thermostable DNA polymerase, and primers. The primers are short oligonucleotides often with 20~40 nucleotides that are complimentary to two 3' ends of a target DNA fragment (Figure 4-1).



**Figure 4-1.** A diagram showing the process of DNA amplification by PCR.  $2^n$  target DNA molecules are synthesized after  $n$  cycles of PCR.

PCR is temporally separated into three steps: **denaturing, annealing, and extension (Figure 4-1)**. **Denaturing** separates double-strand DNA molecules into single strands by simply heating the reaction mixture to a high temperature such as 94°C. After DNA molecules are denatures, an **Annealing** step is applied. Modern thermocyclers can quickly drop the reaction temperature down to annealing temperature within few seconds. The selection of annealing temperature is primarily determined by the melting temperature of your primer pair and the DNA polymerase. For regular *Taq* DNA polymerase, annealing temperature is usually selected as the low  $T_m$  - 5°C. The annealing time also depends on the primers. The longer the primer length, the longer the annealing time. Usually, 30 sec is plenty. We all should know that the function of a DNA polymerase requires a primer to initiate its DNA synthesis activity. So does the thermostable PCR DNA polymerase. When the two primers are annealed to the 3'-end of two single strand

## Lab 7: Molecular Cloning (3 hours)

---

DNA molecules, an **extension** step is followed to allow the DNA polymerase to extend a daughter strand from 5'-end to 3'-end. A normal *Taq* DNA polymerase only has a 5' to 3' DNA polymerase function. Because DNA synthesis is so rapid in a PCR reaction, a wrong nucleotide is occasionally introduced to cause point mutation, which can cause trouble in our molecular cloning experiments. To increase the accuracy of DNA synthesis, some types of PCR polymerases have been either engineered or mixed with another enzyme to acquire a 3' to 5' exonuclease activity. During polymerization of a deoxyribonucleotide to the 3'-end hydroxyl group of a DNA strand, the 3' to 5' exonuclease activity proofreads the base pairing between the added nucleotide and the one on the template strand. If they are not paired, the mismatched nucleotide will be cleaved off. By adding this extra proofreading step, those PCR polymerases give high accuracy in DNA synthesis. The payoff is that the yield could be reduced and that the DNA molecules synthesized are blunt ended.

After accomplishing one cycle, the amount of DNA molecules in the reaction tube is doubled. With multiple cycles, usually 30-40 cycles, millions of DNA molecules are synthesized so that the PCR product can be directly separated and visualized by DNA agarose gel electrophoresis. Since each cycle heats the DNA polymerase once, the heat tolerance of a DNA polymerase is also an importance factor that can influence the PCR yield.

### **Objective**

1. Be able to amplify *Arabidopsis SKP1-Like 1 (ASK1)* coding sequence fragments from a plasmid.

### **Reagents**

Distilled water, sterile  
10x Reaction Buffer  
dNTP mixture, 10 mM (this consists of dATP, dCTP, dGTP, and dTTP)  
Forward primer (10  $\mu$ M)  
Reverse primer (10  $\mu$ M)  
Target DNA (pGBK-ASK1, 2 ng/ $\mu$ L)  
*DreamTaq* DNA polymerase (1 unit/ $\mu$ L)

### **Equipment**

0.2-mL PCR tubes  
1.5 mL microfuge tubes  
Microcentrifuge tube rack  
PCR tube rack  
Black sharpie (ultra-fine)  
Microcentrifuge  
Pipettes  
Pipette Tips  
Ice Bucket  
Thermocycler

## Procedure

### Part 1: Polymerase Chain Reaction

1. Label PCR tubes with your initials and gene names on top side of each tube. Please do not label on the cap because the ink on the cap can stain the heating block of a thermocycler. Two tubes for one student (5 min).

**Tube 1-2:** initial + ASK1;

2. Make Master Mix solution (2 people in one group)

**Table 6.1 Components in regular PCR reactions**

COMPONENTS	1 REACTION	5 REACTIONS
Distilled H <sub>2</sub> O, sterile	24.8 µL	124 µL
5x reaction buffer	10.0 µL	50 µL
2.5 mM dNTPs	4.0 µL	20 µL
Primer 1	2.5 µL	12.5 µL
Primer 2	2.5 µL	12.5 µL
<i>DreamTaq</i> DNA polymerase	0.2 µL	1 µL

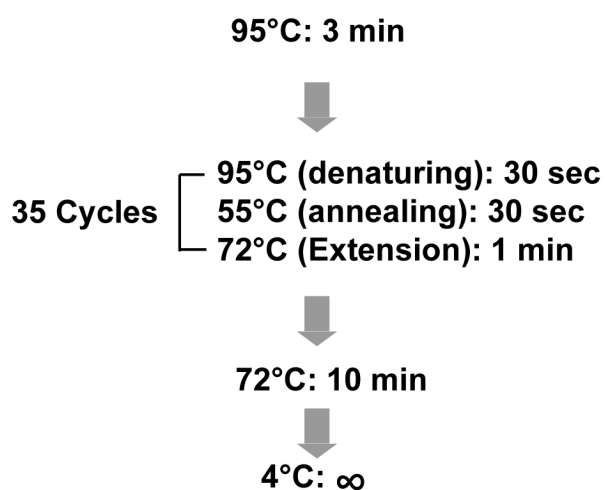
Note: to make enough Master Mix solution (e.g. 5 reactions), multiply by the amount of PCR components used in 1 reaction (15 minutes)

3. Add 44 µL of a Master Mix solution without the DNA template in one PCR tube.

2. Add 6 µL of DNA template (2 ng/µL). Mix reaction by pipetting the reaction up and down 5-6 times. Change tips before mixing next sample.

3. Close cap and place the tube on ice. Keep in mind, your tubes are well labeled on top side of the tube. Your TA will collect all the PCR tubes of the class and run the PCR program together.

4. A typical PCR program is setup as shown in Figure 4-2 and usually takes 2 hours to finish.



**Figure 4-2.** Typical setup of a regular PCR. The annealing temperature depends on the nucleotide composition in your two primers, the PCR buffer and DNA polymerase.

## Lab 7: Molecular Cloning (3 hours)

---

Usually, you need to design the two primers with close melting temperatures ( $T_m$ ). Primer  $T_m$  can be simply estimated using the formula  $T_m = 2 \times (A+T) + 4 \times (G+C) - 2$ . However, you may find it is more convenient using many online tools, such as

<http://www6.appliedbiosystems.com/support/techtools/calc/>

<https://www.thermofisher.com/us/en/home/brands/thermo-scientific/molecular-biology/molecular-biology-learning-center/molecular-biology-resource-library/thermo-scientific-web-tools/tm-calculator.html>

<http://tmcalculator.neb.com/#/>

<https://www.idtdna.com/calc/analyzer>

Extension time depends on the length of the target DNA template and DNA polymerase.

Usually, it requires 1 min per 1 kb DNA. Use 1 min if your template is < 1 kb

### **Questions**

1. What is the role of the *Taq* enzyme in PCR reactions?
2. What is the difference between a *Taq* DNA polymerase and an *E. coli* DNA polymerase?
2. What are the three steps in a PCR Reaction?
3. List three applications of PCR in biology?

**(Notes)**

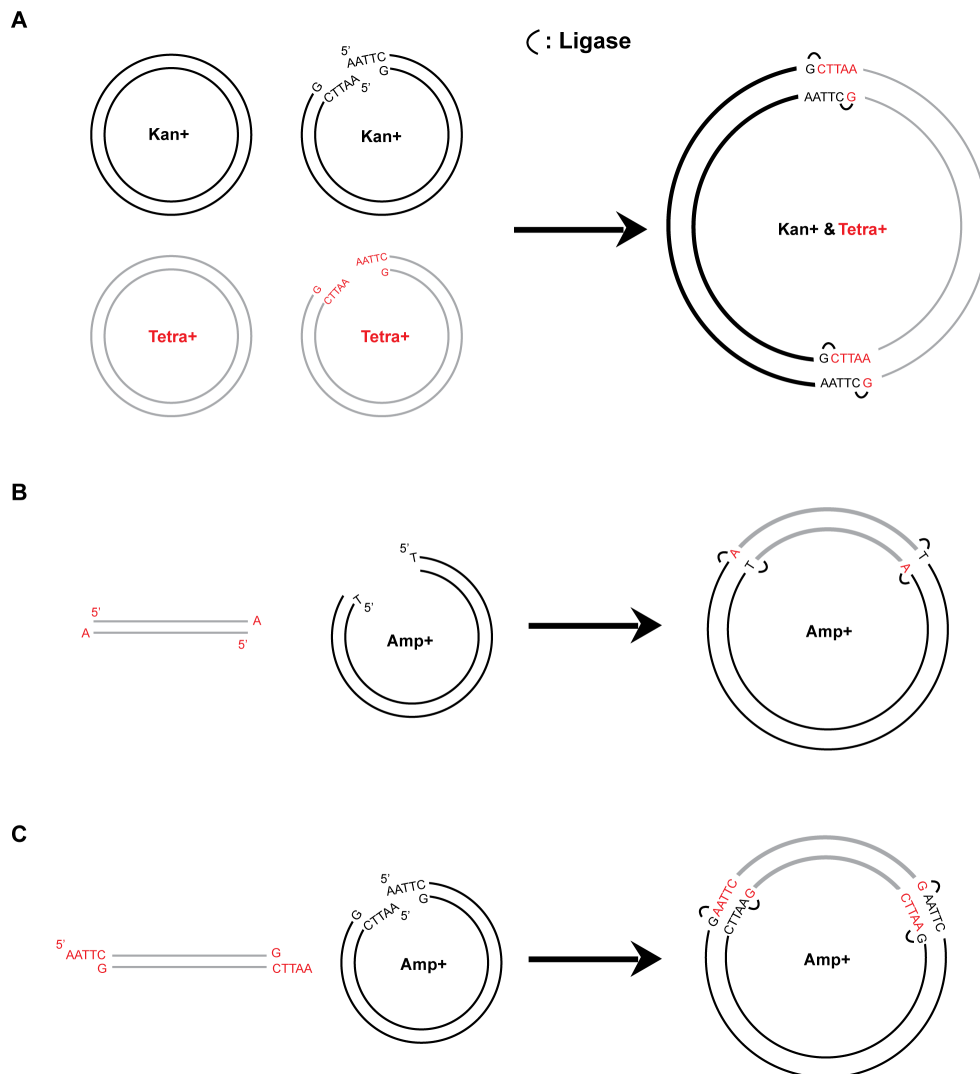
**(Notes)**



## Lab 8: Ligation (2 hours)

### Introduction

In this experiment, we will ligate your PCR product into a plasmid. Plasmids are circular DNA molecules that are naturally present in bacterial cells to help bacteria to combat antibiotics by expressing antibiotic resistant genes. Since some regions in a plasmid are not crucial and can be removed by molecular scissors, restriction enzymes, Herbert Boyer and Stanley N. Cohen first developed precise genetic engineering in 1973 [Cohen et al., PNAS 1973(70): 3240-3244]. They first applied a restriction enzyme, EcoRI, to cut one plasmid containing a Kanamycin resistant gene and another containing tetracycline resistant gene. Because linearized DNA strand after being cut by restriction enzymes exposes a sticky (cohesive) end that is complement to the other, two DNA strands will be attached together by these two cohesive ends through base pairing and can be further ligated into one DNA molecule by ligase (Figure 8-1A). Since then, a number of recombinant DNA technologies and applications have appeared.



**Figure 8-1.** A schematic diagram showing the flow of three types of recombinant DNA engineering

## Lab 8: Ligation (2 hours)

---

In our case, the original plasmid template DNA molecules are still present in your PCR products and its number is more than the ligation products. If you simply mix your PCR products with plasmid DNAs with cohesive ends, you may not be able to get recombinant plasmids containing your PCR products because they are extremely diluted by your original template plasmids. To solve this problem, we can apply DNA agarose gel electrophoresis to separate your PCR products from the template plasmid DNA molecules.

The agarose is a porous particle that allows DNA molecules to migrate through it. In soluble status, DNA molecules are charged and often are negatively charged if the solution pH is higher than its isoelectronic point (pI) value. Since the pH of DNA electrophoresis buffer, such as TAE (Tris-acetate-EDTA), is higher than the pI value of DNA molecules, DNA always migrate from negative end (anode) towards the positively charged end (cathode) in the gel if electric current is applied. However, the evenly distributed agarose matrix physically slows down this migration. The larger the DNA fragments the stronger the effect. Therefore, DNA molecules with different sizes move in the agarose gel with different migration rates so that they are separated after electrophoresis.

Practically, DNA molecules need to be dissolved in a loading buffer containing glycerol (or sucrose) and dye (e.g., bromophenol blue). Because the physical density of glycerol is heavier than water, a glycerol-containing DNA sample allows the DNA to be settled down in the loading well without floating away. The dye also migrates from anode to cathode and its migration rate is similar to a certain size of DNA molecules, which is used to estimate the migration status of DNA molecules. A DNA ladder containing a mixture of a number of DNA fragments with known sizes is always used to run together with DNA samples. Thus, it serves as a “ruler” to help estimate the size of a target DNA fragment. In order to visualize DNA molecules, another dye (e.g., ethidium bromide and GreenGlo), which specifically binds to DNA, is pre-mixed in the agarose gel. The dye-associated DNA molecules will illuminate under UV light allowing them to be detected and recorded by UV-imaging. Furthermore, the DNA fragments of interest can also be observed through UV light and isolated from the DNA gel using a DNA extraction kit.

Ligases catalyze the formation of a phosphodiester bond between the 5' phosphate group of a nucleotide and the 3' hydroxyl group of another. In normal cells, ligases are used for DNA nick repair. The bacteriophage T4 ligase is the ligase most-commonly used in laboratory research. Like in Cohen-Boyer's experiment, the attachment of complementary DNA segments, which contain sticky or cohesive ends (Figure 8-2 B and C), will produce a DNA nick making the ligation much more effective by preventing the dissociation of two DNA molecules during the reaction. In our case, the PCR products contain two adenine (A) sticky ends due to the activity of *Taq* DNA polymerase. These sticky ends are complementary to the T ends of a linearized pGEM-T easy vector, allowing the DNA insert and the vector to make a relatively stable intermediate. The catalysis activity of T4 ligase repairs the nick and yields a recombinant pGEM-T-ASK1 plasmid,

### **Objectives**

1. Be able to list the components in a ligation reaction.
2. Be able to pipette and mix a small amount of reaction.
3. Be able to perform gel electrophoresis and extract DNA from a gel slice.

### **Reagents**

Agarose (powder)  
1x TAE buffer  
GreenGlo™ Safe DNA Dye  
DNA molecular marker, 1 kb ladder  
6x Gel loading dye  
Distilled water, sterile

Macherey-Nagel gel extraction kit  
2x Rapid Ligation Buffer  
T4 DNA Ligase  
pGEM®-T Easy Vector  
PCR product

### **Equipment**

Glass flask  
Microwave  
Gel trays  
Gel comb  
Gel electrophoresis apparatus  
UV illumination source  
Razor blades  
50° C waterbath  
0.5 mL microcentrifuge tubes  
1.5 mL tubes

### **Procedure**

#### **Part 1: Gel electrophoresis**

1. Add 10 µL of 6 x loading dye to each 50 µL PCR product. (5 minutes)
2. Load 15 µL of the PCR product into each well. In total, 30 µL of each PCR product needs to load into the gel, i.e. two lanes per each PCR reaction from Lab 4. (30 minutes)

**NOTE:** Write down the order of your samples in the gel!

3. Run the gel for 20 ~ 30 minutes at 100 volts.

#### **Part 2: Gel Extraction**

1. **Before the end of gel electrophoresis:** Label one 1.5 mL tube and record the weight.
2. Take your gel to the UV illuminator and cut your DNA samples from the gel with a razor blade. Place the excised sample into the pre-weighed tube. For the same gene, place the bands from both lanes into the same tube.
3. Weigh the tube with the excised sample and record the weight. Take the difference of the tube weight with and without the sample to determine the weight of the gel,

## Lab 8: Ligation (2 hours)

---

$$\text{i.e Gel Weight} = \text{Weight (tube + gel)} - \text{Weight (tube)}$$

4. Add 300 $\mu$ L NTI buffer for every 100 mg of gel,

$$\text{i.e NTI volume} = 3 \times (\text{Gel Weight mg})/100.$$

5. Heat at 50° C for 5-10 minutes until the gel is completely melted.

6. Load the melted gel and buffer mixture onto a spin column and 2 mL collection tube (provided with the kit) and centrifuge at 11,000 x g for 30 s at room temperature (RT). This binds the DNA to the spin column. Discard the flow through.

7. Add 700  $\mu$ L NT3 buffer. Centrifuge at 11,000 x g for 30 s at RT. Remove the flow through.

8. Repeat step 7 once.

9. Following the second wash, centrifuge at 11,000 x g for 2 minutes. This dries the column.

10. Transfer the column into a new 1.5 ml eppendorf tube without disturb the waste liquid collected at the bottom of the 2 ml collection tube.

11. Add 30  $\mu$ L pre-heated (65° C ) distilled H<sub>2</sub>O onto the column and incubate the column at RT for 1 minute. Centrifuge at 11,000 x g for 1 minute to elute your PCR product.

### Part 3: Set up Ligation

1. Set up ligation master mix following the table below (by TAs). (30 min)

**Note:** Vortex 2x Rapid Ligation buffer vigorously before each use.

**Table 7.1 Components in a regular ligation reaction**

COMPONENTS	STANDARD REACTION
2x Rapid Ligation Buffer, T4 DNA Ligase	5 $\mu$ L
pGEM®-T Easy Vector	0.5 $\mu$ L
PCR product	3.5 $\mu$ L
T4 DNA Ligase	1 $\mu$ L
Nuclease-free water to a final volume of	<b>10 <math>\mu</math>L</b>

2. In a clean PCR tube, aliquot 3.5  $\mu$ L of your cleaned PCR product. When ligation master mix is ready, add 6.5  $\mu$ L of the mixture into your PCR product. Mix the reactions by slowly pipetting.

In general, the mole ratio of insert to the vector is 3:1. However, my experience shows that this ratio is not always crucial. Often it is due to the insufficient quantity and/or the low quality of your inserts (e.g. the damage of the sticky ends by vigorous pipetting) that causes the failure of ligation.

**Note:** Avoid air bubbles, sticky ends are sensitive.

4. Incubate overnight at 16° C.

### Questions

1. Why is it important to pay attention to which way you set up your electrophoresis gel?
2. Which PCR products are more effective to be cloned into a pGEM-T easy vector?

- A) PCR products amplified by DreamTaq DNA polymerase
  - B) PCR products amplified by high-fidelity Phusion DNA polymerase
  - C) EcoRI digested PCR products
  - D) Blunt-end restriction enzyme-treated PCR products
3. What is the function of DNA loading buffer and why can it help settle down the DNA sample into the sample wells of the gel?

**(Notes)**

## Lab 9: Bacterial Transformation & Genomic DNA Extraction (3 hours)

### Introduction

Bacterial transformation was first invented by Frederick Griffith for studying the infectious activity of smooth and rough *Streptococcus pneumoniae* in mouse in 1928. Since then, it has been widely applied as one of the best ways for DNA amplification. It can also serve as an efficient way to screen the positive ligation clones. For example, the insertion of a target DNA into the  $\alpha$ -peptide coding region of the enzyme  $\beta$ -galactosidase will result in the dysfunction of the enzyme. As a consequence, the negative and positive clones will have functional and non-functional  $\beta$ -galactosidases, and producing blue and white signals, respectively, inside the transformed *Escherichia coli* (*E. coli*) after incubating with the enzyme substrate,  $\beta$ -galactosidase.

The pGEM<sup>®</sup>-T Easy Vectors contain T7 and SP6 RNA polymerase promoters flanking a multiple cloning site region within the  $\alpha$ -peptide coding sequence of  $\beta$ -galactosidase. Insertional inactivation of the  $\alpha$ -peptide prevents  $\beta$ -galactosidase from being produced. The transformed bacterial cells are spread on Luria-Bertani (LB) broth plates with ampicillin, Isopropyl  $\beta$ -D-1-thiogalactopyranoside (IPTG), 5-bromo-4-chloro-3-indolyl- $\beta$ -D-galactopyranoside (X- $\beta$ -Gal). If the  $\beta$ -galactosidase is produced, X- $\beta$ -Gal is hydrolyzed into 5-bromo-4-chloro-indoxyl, which then dimerizes to produce a blue pigment. If  $\beta$ -galactosidase is not produced, X- $\beta$ -Gal is not hydrolyzed and will not form the blue pigment. Therefore, if a bacterial colony is white, the insert is present due to the inactivation of  $\beta$ -galactosidase in the cell. If a colony turns into blue, the insert is not present. Simply based on pigmentation of bacterial colonies, this white-blue screening approach provides an easy way to determine the success of cloning in a pGEM<sup>®</sup>-T Easy Vector.

This lab will also introduce Genomic DNA extraction. Genomic DNA has broad applications in biology and biotechnology, including Southern Blotting, molecular cloning, PCR, genetic mapping, etc. This lab will use cetyltrimethylammonium bromide (CTAB) to isolate genomic DNA from wild-type and ask1 mutant leave tissues. A number of methods has been developed to extract genomic DNA. Essentially all methods developed adopt a way to suppress DNA degradation and remove contaminants. In this lab we use 65°C incubation to inhibit nuclease activity and apply CTAB to remove glycoproteins and polysaccharides that co-precipitate with DNA during the extraction process.

### Objectives

1. Be able to transform plasmids into *E. coli*.
2. Be able to extract genomic DNA.

### Reagents

#### 1) Bacterial Transformation

Ligation reactions from the previous lab  
LB medium  
JM109 High Efficiency Competent Cells  
70% Ethanol

#### 2) Genomic DNA Extraction

CTAB buffer

## Lab 9: Bacterial Transformation & Genomic DNA Extraction (3 hours)

---

Liquid Nitrogen  
100% Ethanol (ice cold)  
70 % Ethanol (ice cold)  
7.5 M Ammonium Acetate  
55°C water bath  
Phenol:chloroform:isoamyl-alcohol (25:24:1)  
Water (sterile)

### **Equipment**

LB plates with 100 µg/L ampicillin+IPTG+ X-Gal  
1.5 mL microcentrifuge tube  
Ice water bath  
Bucket of Ice  
42°C water bath  
55°C water bath  
37°C Incubator, shaking  
Ethanol Lamp  
Cell spreader

### **Procedure**

#### **Part 1: Bacterial Transformation**

1. Prepare and label two LB/ampicillin/IPTG/X-Gal plates for each ligation reaction. Equilibrate the plates to room temperature.
2. Centrifuge the tubes containing the ligation reactions to collect contents at the bottom. Label and add 10µL of each ligation reaction to one sterile 1.5mL microcentrifuge tube on ice. Label and add 10µL of diluted intact plasmids (0.001 ng in total, by TA) into another sterile 1.5mL microcentrifuge tube on ice.
3. Gently flick tube of JM109 High Efficiency Competent Cells. Carefully transfer 50µL of cells into each tube prepared in Step 2.
4. Gently flick the tubes to mix and place them on ice for 30 minutes. (Go to Part 2 and finish Step 1 to Step 3)
5. Heat-shock the cells for 45-50 seconds in a water bath at exactly 42°C. **Do not shake!**
6. Immediately return the tubes to ice-water for 2 minutes.
7. Add 950 µL room-temperature LB medium.
8. Incubate for 1 hr at 37°C water bath. (Go to Part 2 and finish Step 5 to Step 10).
9. Spin down the cells with 8,000rpm for 1 min at RT.
10. Gently dump the supernatant into the sink and keep the remaining ~100 uL LB and the cell pellet.



## Lab 9: Bacterial Transformation & Genomic DNA Extraction (3 hours)

---

11. Resuspend the cell pellet with the remaining ~100  $\mu$ L LB and plate the cell culture onto a LB+ampicillin+IPTG+X-Gal plates. To do this, dip cell spreader in ethanol and sterilize with ethanol lamp, do this twice. Let the cell spreader cool before spreading the cultures. Then using the sterilized spreader spread the 100  $\mu$ L transformation cultures across the plate evenly. **Sterilize before working on the next plate!!!**

12. Place the plates upside down and incubate them overnight (16-24 hours) at 37°C.

### Part 2: Genomic DNA Extraction

1. Label 2 1.5 mL eppendorf tubes, 1 with wild-type (WT) and your initial and another with mutant (MU) and your initial.

2. Harvest 1 Arabidopsis Col-0 WT green rosette leaf in a 1.5 mL eppendorf tube and 1 MU green rosette leaf in another.

3. Close the cap and drop the tubes in liquid nitrogen container.

4. Use tweezers to take one tube at one time and place the tube on a eppendorf tube rack. Open the cap gently (do not break the cap) and use a blue pestle to grind leaf tissue directly in eppendorf tube. The tissue should turn a fine powder in the tube.

3. Add 0.5 ml of CTAB extraction buffer (don't forget the  $\beta$ -mercaptoethanol) in the hood. Grind at room temp for another two min in hood. (At this point you should have a dark green liquid)

4. Incubate at 55-60°C for about 30 min. (Go to Part 1 to finish Step 5 to Step 7).

5. Extract the DNA with 0.5ml of 25:24:1 phenol:chloroform:isoamyl-alcohol. Tightly cap the tube and mix vigorously by hand (wear gloves). Incubate the tube for 5 min on a rocking shaker at room temperature.

6. Centrifuge at 12,000g for 5 minutes. Transfer ~500  $\mu$ L aqueous layer (should be the top layer) to a new 1.5 mL eppendorf tube.

7. Precipitate with ~500  $\mu$ L of cold isopropanol and invert to gently mix. Precipitate might be visible. Spin at 12,000g for 5 minutes to pellet the nucleic acids.

9. Wash with 1ml of cold 70% ethanol, and spin at 12,000g for 1 min. Discard the 70% ethanol and spin another time at 12,000g for 1 min. Suck any remaining residue liquid with P200 pipette. Leave the cap open on a eppendorf tube rack and dry the pellet in the fume hood for 10 min.

Do not completely dry the pellet or it will become difficult to resuspend. Pellet should become somewhat clear and glassy.

10. Dissolve DNA with 100 $\mu$ L warm water (65°C) by vortexing. Freeze the DNA sample at -20°C for Lab 9.

Avoid breaking up tan-colored pellets, most of which are polysaccharides and are hard to dissolve. They may retard your DNA in the sample well during gel electrophoresis.

**Questions**

1. Why is it necessary to heat shock the cells in bacterial transformation?
2. Why do *E.coli* colonies without the DNA insert turn blue?
3. What does CTAB do in genomic DNA extraction?

**(Notes)**

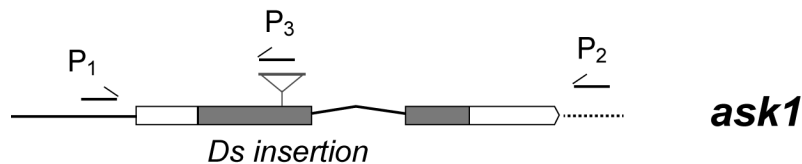
**(Notes)**

## Lab 10: Genotyping (2 hours)

### Introduction

Polymerase Chain Reaction (PCR), as discussed previously in Lab 7, can be used to further screen the presence of a target DNA sequence in a recombinant vector. Due to time limit, this lab is only allowed to setup PCR. Lab 11 will examine the PCR products by DNA gel electrophoresis. The PCR uses a pair of primers that are flanking to cloning site where the insert is placed. If the *E. coli* colony, which is propagated from one single cell from 16-hour culture in Lab 9, carries a positive recombinant vector, a larger DNA fragment will be detected on the gel than that detected from colonies carrying a vector without the insert.

From prior functional genomic studies supported by the National Science Foundation Arabidopsis 2010 project, T-DNA or transposon insertion mutations have been made in many nuclear genes in *Arabidopsis thaliana* genome, primarily in Col-0 accession. This information can be retrieved from the Arabidopsis Information Resource (TAIR, <https://www.arabidopsis.org>) and many T-DNA mutants can be ordered from the Arabidopsis Biological Resource Center (<https://abrc.osu.edu>). In *ASK1* gene, one Ds transposon insertion mutation can be found and it is present in mutant genomic DNA you extracted in Lab 9. The genomic structure of this mutant allele can be seen in Figure 10-1.



**Figure 10-1.** Genomic structure of *ASK1* gene. The triangle identifies a Ds insertion site in *ask1* mutant. The positions of primers for genotyping are also shown. Solid boxes: coding sequence; Open boxes: untranslated region; Solid straight line: promoter; Solid folded line: intron; Dash line: flanking DNA. Primers: P1, ASK1\_t\_gpcr\_for; P2, ASK1\_t\_gpcr\_rev; P3, ask1\_tdna.

### Objectives

1. Be able to sample *E. coli* colonies.
2. Be able to setup PCR master solution and perform PCR.

### Reagents

Distilled water  
10x Reaction Buffer  
dNTP mixture, 10 mM (this consists of dATP, dCTP, dGTP, and dTTP)  
T7: TAATACGACTCACTATAGGG  
SP6: TATTTAGGTGACACTATAG  
ask1\_tdna: CGTTCCGTTTTTCGTTTTTTACC  
ASK1\_t\_gpcr\_rev: TGTCATGGTAACGAGTTAGCG  
ASK1\_t\_gpcr\_for: AACAAATCGACCGACTTATC  
*DreamTaq* DNA polymerase, 1 unit/ $\mu$ L stock concentration  
6x Loading Buffer  
1x TAE solution  
Agarose gel electrophoresis system Ethidium Bromide solution

### **Equipment**

0.2 mL PCR tubes  
1.5 mL microcentrifuge tubes  
Ice bucket  
Thermocycler  
Mortar and pestle  
25° C Incubator, shaking  
65°C water bath  
Centrifuge

### **Procedure**

#### **Part 1: Count Transformation and Ligation Efficiencies**

1. Use a red mark pen to highlight white colonies while counting the total number. Write down the number in Table 9.1
2. Use a black mark pen to highlight blue colonies while counting the total number. Write down the number in Table 9.1
3. Calculate the transformation (Plate 1, control insert DNA) and ligation (Plate 2) efficiencies using the formula below.

$$200\text{cfu} / 0.001\text{ng DNA} = 2 \times 10^5 \text{ cfu/ng} = 2 \times 10^8 \text{ cfu/}\mu\text{g DNA}$$

**Table 9.1 Transformation and ligation efficiencies**

Plate	White Colonies	Blue Colonies	Total Colonies	# of effective PCR	# of Positive PCR	Efficiency
1						
2						

#### **Part 2: Colony PCR**

1. Label 8 1.5 mL microcentrifuge tubes with initials, gene, and replicate colonies.
2. Add 50  $\mu\text{L}$  sterile  $\text{H}_2\text{O}$  in each tube.
3. Pick up 1 colony with a pipet tip and suspend it in the 50  $\mu\text{L}$  of sterile water in each microcentrifuge tube. **Make sure to label properly!**
4. Do this for a total of 8 colonies from your ligation plate.
5. Make master mix following the table below.
6. Place 20  $\mu\text{L}$  of master mix into 0.2 mL PCR tubes.
7. Add 5  $\mu\text{L}$  of bacterial suspension into each PCR tube with the master mix.

8. Run PCR for about 2 hours.

**Table 9.2 Components of colony PCR Master Reaction Solution**

COMPONENTS	1 REACTION	2 REACTIONS	10 REACTIONS
10 x Buffer	2.5 µL	5 µL	25.0 µL
2.5 mM dNTP	2 µL	4 µL	20 µL
Forward Primer	1.25 µL	2.5 µL	12.5 µL
Reverse Primer	1.25 µL	2.5 µL	12.5 µL
DreamTaq	0.05 µL	0.1 µL	0.5 µL
Distilled H <sub>2</sub> O, sterile	12.95 µL	25.9 µL	129.5 µL
<b>Total</b>	<b>20.0 µL</b>	<b>40.0 µL</b>	<b>200.0 µL</b>

### Part 3: Characterization of T-DNA homozygous mutant

Follow Table 9.2 to setup two PCR reactions for primer pairs, P1 + P2 and P1 + P3, to amplify wild type fragment of *ASK1* and Ds transposon fragment in *ask1*, respectively. The former should be amplified when you use wild-type but not the *ask1* genomic DNA as template and vice versa. However, both will be amplified in a heterozygous insertion mutant.

### Questions

1. Why do we need to do both colony PCR and White/Blue screens for the pGEMT-Easy Vector System transformation? Be precise to explain the rationale.
2. Why are both wild-type and mutant fragments amplified in a heterozygous T-DNA or transposon insertion mutant?
3. List some applications of genomic DNA product in molecular biology.

**(Notes)**



## Lab 11: Gene Amplification (3 hours)

### **Introduction**

As mentioned in Lab 10, PCR screen of bacterial colonies is an effective way to examine transformants containing a positive recombinant vector with the insert of interest. The PCR can be performed with a pair of primers flanking to the cloning sites or with a combination of a primer specific to the insert and the other to the vector. If the colony has a positive recombinant vector, a larger band will be detected than colonies without it in the PCR products using vector primers. This step is important because it truthfully demonstrates the success of positive cloning. It can further verify the white-blue screening results. In some cases, the white-blue screening approach may cause false negative results. For example, if the insert does not change the coding frame of  $\alpha$ -peptide gene. The product of this recombinant gene may maintain the activity of  $\beta$ -galactosidase. The white-blue screening approach can also cause false positive errors. For example, X- $\beta$ -Gal is not uptaken well by the bacterial cells, X- $\beta$ -Gal is defective, or IPTG fails to induce the expression of  $\beta$ -galactosidase, etc.

Once the colonies have been screened and positive ones are chosen, we will propagate the recombinant plasmids by simply growing more bacterial cells containing the positive clone. In many cases, we also want make a glycerol stock of the positive clone for long-term storage. This is done by inoculating the positive colony in 5 mL LB liquid medium containing an appropriate antibiotic for selection. In our case, 100  $\mu$ g/L ampicillin is used to allow bacterial cells carrying a plasmid with ampicillin resistant gene to grow. Without the selection of antibiotics, the cells may lose the plasmids during cell propagation.

### **Reagents**

Agarose (powder)  
1x TAE buffer  
GreenGlo™ Safe DNA Dye  
DNA molecular marker, 1 kb ladder  
6x Gel loading dye  
Distilled water, sterile  
Ampicillin

### **Equipment**

Glass flask  
Microwave  
Gel trays  
Gel comb  
Gel electrophoresis apparatus  
15 mL culture tube  
Shaker

### **Objectives**

1. Be familiar with PCR set up
2. Be familiar with DNA gel electrophoresis.
3. Be able to perform sterile operation

## **Procedure**

### **Part 1: Make an Agarose Gel (2 students)**

1. Make agarose gel by adding 0.4 g of agarose powder to 50 mL of TAE buffer. Heat the mixture in the microwave for 2-3 minutes bringing the solution to boil, making sure to swirl the mixture every few seconds. (10 ~ 20 minutes)
2. Once the mixture is cooled down to ~60° C (able to hold the flask with your naked hand), add 3 µL of GreenGlo™ Safe DNA Dye and pour into gel tray. Insert the gel comb and allow the gel to solidify for approximately 30 minutes.
3. Place the gel into the electrophoresis apparatus. Make sure the gel is oriented in the right direction. Remove the combs and fill the apparatus with TAE buffer until it covers the gel.

### **Part 2: Gel electrophoresis of the colony PCR products**

1. Mix 6µL of loading dye to each 25 µL PCR product. (5 minutes)
2. Load 10 µL of the PCR product into each well. For each PCR product reaction, load into two wells. (30 minutes)

**NOTE:** Write down the order the gel was loaded!

3. Run the gel for 30 minutes at 100 volts.
4. Take image of gels.

### **Part 3: Inoculation of Positive Clones**

1. Label the PCR tubes with positive PCR band detected (2 minutes)
2. Align the PCR tubes to the 1.5 ml tubes containing the bacterial suspension from Lab 7.
4. Aliquot 5 mL LB + 100 ug/L ampicillin into a 10 ml culture tube.
5. Inoculate 10 µL of bacterial suspension producing a positive PCR product into the culture tube and incubate the cells overnight at 37 ° C with 250 RPM shake.

## **Questions**

1. Draw a plasmid map to show and explain why PCR screen can help detect the real transformant.
2. What is the purpose of the ampicillin during the bacterial amplification? Why is it ampicillin but not kanamycin?
3. Why may the cells lose the recombinant plasmids if ampicillin is not added in the LB medium for plasmid propagation in our experiment?

**(Notes)**

**(Notes)**

## **Lab 12: Plasmid Isolation (2 hours)**

### **Introduction**

Plasmids are important molecular biology tools that can accept, carry, and replicate (clone) target DNA fragments, e.g., coding sequence DNA from a gene of interest. Therefore, plasmid DNAs often need to be isolated and purified from bacterial cells. Many biotechnology companies have developed simple kits for this purpose to significantly reduced the time and exposure of nasty chemicals, such as chloroform, in plasmid purification.

In our experiment, as small as 3 mL of overnight bacterial culture is harvested for purifying pGEMT-ASK1 recombinant plasmids that you cloned from Labs 10 and 11. The bacterial cells are pelleted and re suspended in Buffer A1 containing ribonuclease A, followed by lysis in SDS/alkaline buffer (Buffer A2). Cells are disrupted, and RNA are degraded. Buffer A3 is then added to neutralize the lysate, which precipitates out proteins and bacterial chromosomal DNAs. The clarified supernatant containing plasmid DNA molecules is subsequently applied onto a silica column. After spinning, the plasmid DNA binds the column through salt bridge interaction. The retained contaminants, such as salts, small metabolites, and soluble macromolecular cellular components are removed by using an ethanol wash (Buffer A4). The cleaned plasmid DNAs are then eluted with warm distilled water.

This purified plasmid DNA can then be sequenced using an Sanger sequencer, such as Applied Biosystems 3130xL Genetic Analyzer in the Ohio University Genomic Facility located on the fifth floor of Porter Hall. The sequencing result (chromatogram) can be visualized and analyzed to determine an unknown gene cloned or to detect any mutations introduced during PCR cloning step in Lab 10.

### **Reagents**

Amplified cells from Lab 10  
DNA from Arabidopsis from Lab 8  
MACHERY-NAGEL Plasmid DNA Purification Kit  
    A1 buffer  
    A2 buffer  
    A3 buffer  
    A4 buffer  
Distilled water

### **Equipment**

1.5 mL microcentrifuge tubes  
Centrifuge  
Ice bucket  
37° C incubator

### **Objectives**

1. Be familiar with pipette operation
2. Be able to isolate plasmid DNA
3. Be able to operate nanodrop spectrometer

## **Procedure**

### **Part 1: Plasmid MiniPrep**

1. Take the culture pellet from Lab 10 and add 250  $\mu$ L Buffer A1 to suspend the cells.
2. Transfer the cell suspension into a 1.5 mL tube.
3. Add 250  $\mu$ L Buffer A2 and shake vigorously right away. Then open lid to aerate for 4 to 5 minutes.
4. Add 300  $\mu$ L Buffer A3 and gently shake via inverting (~8-10 inversions or until the blue sample is colorless) then let rest for 3 minutes.
5. Spin down at 11,000 g for 10 minutes.
6. Transfer supernatant for to a spin column by adding 400  $\mu$ L of supernatant and spin at 11,000 x g for 30 seconds. Do this twice.
7. Add 700  $\mu$ L Buffer A4, spin 11,000 x g for 30 seconds.
8. Dry the column by spinning with 2 minutes at 11,000 x g.
9. Place the spin column into a new 1.5 mL microcentrifuge tube and add 30  $\mu$ L hot distilled H<sub>2</sub>O. Incubate at RT for 1 min.
10. Elute the plasmid DNA with a full speed for 1 minute.

### **Part 2: Plasmid Yield**

1. Lab heads up to the Genomics Facility on 5<sup>th</sup> floor of Porter Hall
2. Bring purified plasmid DNA from Part 1 and genomic DNA from Lab 8 on ice.
3. Set Nanodrop to "Nucleic Acid".
4. When program first opens, load 1.5  $\mu$ L distilled water onto the pedestal and hit "OK". This initializes the instrument
5. Wipe off the pedestal and load 1.5  $\mu$ L distilled water onto the pedestal again. Hit "Blank". In the Sample # box, put your initials.
6. Wipe off the pedestal and load 1.5  $\mu$ L of your sample onto the pedestal. (1 minute)  
Click "Measure" (1 minute)
7. Record the number which appears in the green box with the label "ng/ $\mu$ L". Also record the number in the 260/230 box, this determines how much contamination there is.
8. Calculate the DNA concentration and yield.

**Table 11.1 Concentrations of DNA samples purified**

Sample	Genomic DNA (WT)	Genomic DNA (MU)	Plasmid 1
Concentration ( $\mu\text{g}/\mu\text{L}$ )			
OD260/ OD280			
OD260/ OD230			

**Questions**

1. Please describe the steps of plasmid purification and the purpose of each step?
2. What do OD230, OD260, and OD280 measure?

**(Notes)**



## **Lab 13: Restriction Fragment Length Polymorphism (3 hours)**

### **Introduction**

Restriction enzymes are commonly used in molecular biology laboratories. They are enzymes that cut a DNA molecule at a specific place, called restriction site, which usually contains four to eight nucleotides that form a palindromic structure. A particular restriction enzyme cuts the sequence between two nucleotides within its recognition site, or somewhere nearby. For example, the common restriction enzyme EcoRI recognizes the palindromic sequence GAATTC and cuts between the G and the A on both the top and bottom strands, leaving an overhang (an end-portion of a DNA strand with no attached complement) known as a sticky end on each end of AATT. This overhang can then be used to ligate in a piece of DNA with a complementary overhang (another EcoRI-cut piece). You may review the instruction of Lab 8 to get better understanding of restriction enzyme in molecular cloning.

Another application of restriction enzyme is to produce restriction fragment length polymorphism (RFLP) after cutting a DNA molecule. This technique can differentiate homologous genes with different restriction sites between different genomes. If two homologous genes with different restriction site pattern, after restriction enzyme digestion, the corresponding genome DNA will yield DNA fragments with different lengths, which is called RFLP. After gel electrophoresis, a subsequent Southern Blotting analysis, which uses a specific DNA/RNA probe from the gene of interest to hybridize the DNA fragments transferred on a nylon membrane, is applied to visualize the polymorphisms of homologous genes in different genomes. RFLP is not only used to detect the genetic diversity but can also be used for genetic mapping, paternity analysis, DNA fingerprinting, etc. Applying this technology, we can also detect the presence and the insertion copy number of a transgene when using the transgene DNA as a probe. In this lab, we will apply this approach to 1) get familiar with the pattern of BamHI-digested genomic DNA purified in Lab 9, and 2) examine the presence of ASK1 insert in pGEM-T easy vector.

### **Reagents**

Genomic DNA from Lab 9  
Plasmid DNA from Lab 12  
Agarose (powder)  
1x TAE buffer  
GreenGlo™ Safe DNA Dye  
DNA molecular marker, 1 kb ladder  
6x Gel loading dye

### **Equipment**

Glass flask  
Microwave  
Gel trays  
Gel comb  
Gel electrophoresis apparatus

### **Objectives**

1. Be familiar with the pattern of restriction enzyme digested genomic DNA
2. Be able to setup DNA digestion by restriction enzyme

3. Be able to analyze DNA digestion result

### **Procedure**

#### **Part 1: DNA Digestion**

1. 10 units of the restriction enzyme, BamHI, is needed to digest 1 µg of DNA at 37°C for 2 hours. We need approximately 2 µg of genomic DNA and 1 µg plasmid DNA. According to Table 12.1 to setup digestion. Add enzyme at the last step of mixture.

2. Once restriction enzyme is added, the samples are mixed with gentle vortex and placed in a 37°C water bath for 2 hours. For complete digestion of genomic DNA, you need an overnight incubation.

**Table 12.1 DNA Digestion Setup**

Components	Stock Concentration	Final Concentration	Volume (µL)		
			Genomic DNA (10 µg)	Plasmid 1 (1 µg)	Plasmid 2 (1 µg)
DNA					
CutSmart Buffer (10x)					
BamHI HF					0 µL
H <sub>2</sub> O					
Total Volume (µL)					

#### **Part 2: Make an Agarose Gel (2 students)**

1. Make agarose gel by adding 0.4 g of agarose powder to 50 mL of TAE buffer. Heat the mixture in the microwave for 2-3 minutes bringing the solution to boil, making sure to swirl the mixture every few seconds. (10 ~ 20 minutes)

2. Once the mixture is cooled down to ~60°C (able to hold the flask with your naked hand), add 3 µL of GreenGlo™ Safe DNA Dye and pour into gel tray. Insert the gel comb and allow the gel to solidify for approximately 30 minutes.

3. Place the gel into the electrophoresis apparatus. Make sure the gel is oriented in the right direction. Remove the combs and fill the apparatus with TAE buffer until it covers the gel.

#### **Part 3: Gel electrophoresis of the DNA Digestion products**

1. Mix 6 µL of loading dye to each 25 µL digestion product.

2. Load 20 µL of the digestion product into each well.

**NOTE:** Write down the order the gel was loaded!

3. Run the gel for 30 minutes at 100 volts.
4. Take image of gels.

**Questions**

1. What are the differences of DNA digestion profiles among the three samples and why?
2. Why does genomic DNA take a long time to finish digestion than plasmid DNA?
3. How many bands will you see for undigested plasmid DNA molecules purified from single transformant in a DNA agarose electrophoresis gel?
4. List three applications of RFLP.

**(Notes)**

## **Lab 14: Protein Gel Electrophoresis (3 hours)**

### **Introduction**

Like DNA molecules, a protein mixture can also be separated using gel electrophoresis based on their electrical charge and/or molecular weight. Sodium dodecyl sulfate polyacrylamide gel electrophoresis (SDS-PAGE) is a widely used method for separating proteins based on their sizes. Here, SDS is an anionic detergent, which keeps net negative charged when dissolved in aqueous solution. Negative charged SDS particles binds to unfolded protein peptide during denaturing process, preventing refolding of denatured proteins and allowing protein migrate from cathode to anode. Since the number of SDS molecules bound to a peptide is in proportion to the size of a protein, the ratio of electrical charge-to-mass remains the same among all SDS-denatured polypeptides. Therefore, final separation of proteins in an SDS-PAGE is only dependent on their differences in molecular mass.

### **Reagents**

5x SDS Sample Buffer  
1x SDS Sample Buffer  
Coomassie Brilliant Blue Stain  
Transfer Buffer  
Membrane  
Filter paper  
Fiber Pad  
Methanol

### **Equipment**

1.5 mL microcentrifuge tubes  
Micropipette  
Micropipette tips  
Centrifuge  
Gel Cassette (SDS gel)  
    Glass Cassette  
    Casting Frame  
    Spacer Plate  
    Short Plate  
Mini-PROTEAN 3 Electrophoresis Module  
Mini-Trans-Blot Cell  
95°C hot water bath

### **Objectives**

1. Be able to handle low temperature (liquid nitrogen) samples safely
2. Be able to perform protein electrophoresis
3. Be able to describe the protein transfer procedure

## **Procedure**

### **Part 1: Total Plant Protein Extraction**

1. Label two 1.5 mL eppendorf tubes, one with wild-type (WT) and your initial and the other with mutant (MU) and your initial. Weigh the tubes and record the weight of each empty tube.
2. Harvest one Arabidopsis Col-0 WT rosette leaf in a 1.5 mL eppendorf tube and one MU yellow-green chimeric rosette leaf in another.
3. Weigh the tube with the leaf and record the weight. Take the difference of the tube weight with and without the sample to determine the weight of the leaf sample,

$$\text{i.e Leaf Weight} = \text{Weight (tube + leaf)} - \text{Weight (tube)}$$

4. Drop the tube in liquid nitrogen to quickly freeze the sample
5. Use tweezers to take one tube at one time and place the tube on an eppendorf tube rack. Open the cap gently (do not break the cap) and use a blue pestle to grind leaf tissue directly in eppendorf tube. The tissue should turn a fine powder in the tube.
6. Add 250  $\mu\text{L}$  2 x SDS-Sample buffer for 100 mg leaf tissue and grind the tissue for another 1 min.

$$\text{i.e SDS-Sample volume} = 250 \times (\text{Gel Weight mg}) / 100 (\mu\text{L}).$$

7. Freeze the sample in liquid nitrogen.
8. Prepare your next sample.
9. If you have done your two samples, use tweezers to take one tube at one time and place them on an eppendorf tube rack. Gently open the cap to release the air pressure.
10. Slightly close your tubes and place them on a 100°C heating block for 5 min to denature the protein.
11. After denature, leave your sample at room temperature to cool down. Do not put on ice.

### **Part 2: Protein Gel Electrophoresis**

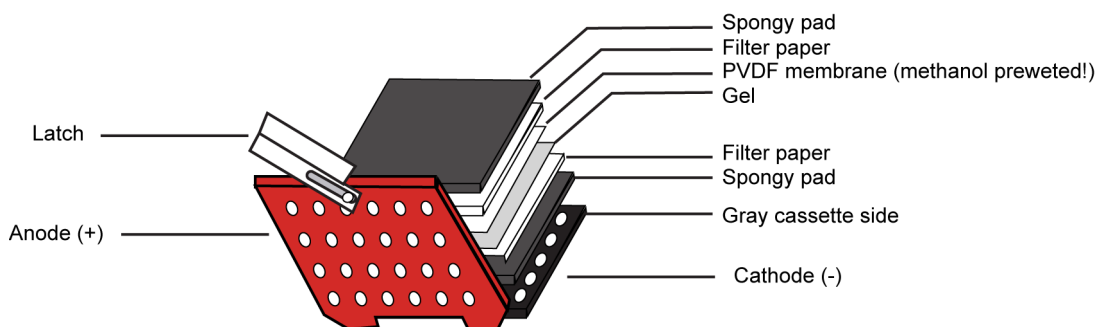
1. Load 10  $\mu\text{L}$  for each sample for two gels.
2. Run the gel at 200 volts for 40-45 minutes.
3. Place Gel 1 into a staining dish with Coomassie Blue staining solution for 24 hours.

### **Part 3: Wet Protein Transfer**

1. Cut the membrane and filter paper to the dimensions of the gel.

2. Pre-wet membrane in methanol for 10 seconds
3. Equilibrate the gel and soak membrane, filter paper, and bifer pads in transfer buffer (15-20 minutes). **Remember to wear gloves when handling the gel!**
4. Prepare the gel sandwich as shown in Figure14-1.

Place the cassette, with the gray side down, on a clean surface.  
Place on pre-wetted fiber pad on the gray side of the cassette.  
Place a sheet of filter paper on the spongy pad.  
Place the equilibrated gel on the filter paper.  
Place the pre-wetted membrane on the gel.  
Complete the sandwich by placing a piece of filter paper on the membrane.  
Add the last spongy pad.



**Figure14-1.** Setup order of a wet protein transfer unit

5. Close the cassette gently, being careful not to move the gel and filter paper sandwich. Lock the cassette with the white latch.
6. Place the cassette in module, red to anode and gray to cathode. Do not get confused!
7. Run the blot at 100 volts for 30 minutes.
8. Dye the membrane with Ponceau S staining solution for 5 min at room temperature.
9. Place the membrane in the center of 4 pieces of white filter paper on bench. Gently apply pressure with your palm to suck away Ponceau S residue.
10. Quickly take out membrane and leave it on top of a piece of clean filter paper until it is semidried. You may scan the membrane to record the electrophoresis of total protein at this time. Otherwise, keep the membrane in clean area at room temperature for the next lab.

### Questions

1. How does SDS-PAGE separate proteins based on their molecular sizes?

**(Notes)**

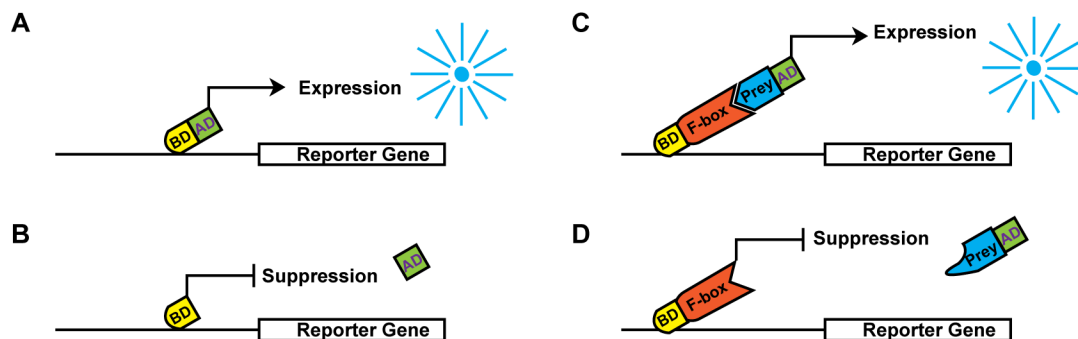


## Lab 15: Yeast Two-hybrid (Y2H) (2 hours)

### Introduction

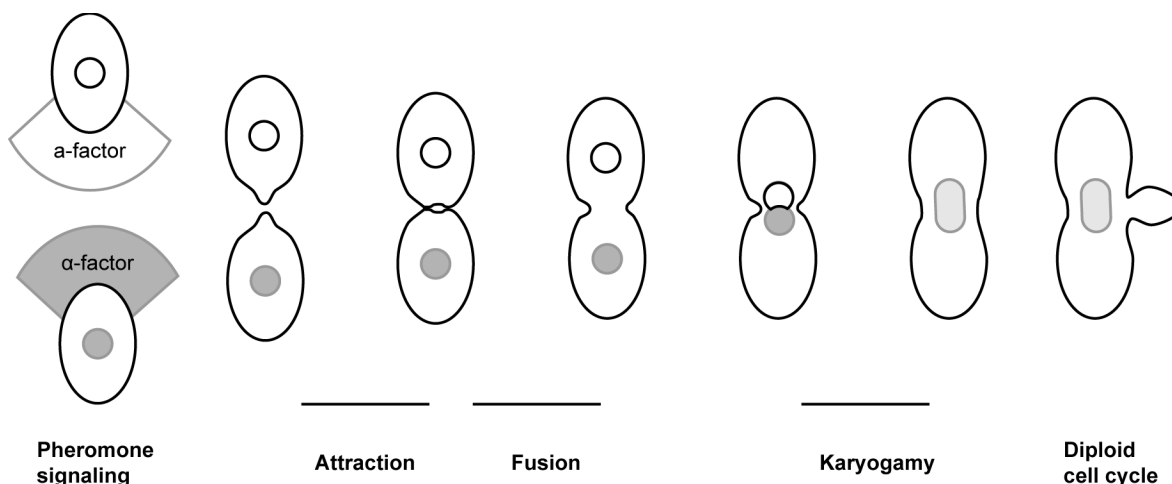
#### *Principle of Yeast Two-Hybrid*

A number of biochemical methods have been developed to examine protein-protein interactions, such as co-immunoprecipitation, pull down, protein binding assay, fluorescence resonance emission transduction (FRET), bimolecular fluorescence componentry (BiFC), gel filtration, yeast two-hybrid, etc. Y2H has been by far one of the simplest way to detect protein-protein interaction in yeast. In a Matchmaker GAL4-based Y2H assay, a bait protein is expressed as a fusion to the Gal4 DNA-binding domain (BD), while prey proteins are expressed as fusions to the Gal4 activation domain (AD). When bait and prey fusion proteins interact, the BD and AD are brought into proximity to activate transcription of four independent reporter genes (*AUR1-C*, *ADE2*, *HIS3*, and *MEL1*). In a Y2H Yeast strain, activation of these reporters (*ADE2*, *HIS3*, and *MEL1*) only occurs in a cell that contains proteins, which interact and bind to the Gal4-responsive promoter.



**Figure 15-1. Diagrammatic representation of yeast two-hybrid system in identifying protein-protein interactions.** A) The activity of a full-length transcription factor binds to DNA through BD and activates the expression of the reporter gene by AD. B) The split of BD and AD repress the expression of the reporter gene. C) Engineered BD fusion and AD fusion are brought in close proximity through the interaction between two test proteins, F-box and Prey, resembling a functional transcription factor and the reporter gene is expressed. D) The lack of interaction between an F-box protein and a prey protein retains the separation of BD and AD. The reporter gene remains silent.

Unlike many other organisms preferentially grow in either haploid or diploid status, yeast cells can exist in both haploid and diploid statuses. There are two types of haploid yeast cells with different mating types (MAT): *MATa* and *MATα*. When these two different mating types of yeast cells meet, one will secrete a pheromone that binds to a receptor exclusively expressed on another, resulting in the two haploid cells to grow towards each other by activating the G protein/mitogen-activated protein (MAP) kinase signal pathway. Eventually, new cell wall and plasma membrane are formed to wrap two haploid cells into one diploid cell. This provides a convenient way to mix two different genes, which are pretransformed in two different mating types of haploid yeast cells, together into one diploid cell. If one haploid expresses a bait protein and the other expresses a prey protein, we can then test the bait-prey interaction in a mated diploid yeast cell.



**Figure 15-2.** Yeast mating process

In this experiment, six mated yeast diploid cells (A1~ A3 and B1 ~ B3) will be given. Since bait construct contains a tryptophan synthase gene and prey construct carries a leucine synthesis gene, we can select the mated diploid cells by growing the cells on -Trp-Leu synthetic dropout (SD) medium. Only diploid cells expression tryptophan and leucine synthase enzymes would the cell grow because yeast is auxotrophic for tryptophan and leucine.

In this lab, the bait constructs are pGBKT7-DIF, pGBKT7-DIFΔFbox (truncated Fbox), and pGBKT7, which are randomly assigned to a numeric number from 1 to 3. Two prey constructs, pGADT7-Ask1 and pGADT7, are randomly assigned to a letter (A or B). Based on the prior knowledge and your final Y2H results obtained in Lab 17, you need to analyze and figure out which cells contain what vectors.

### **Reagents**

SD-Trp-Leu plates (3/person)  
Sterile water

### **Equipment**

Pipette Tips  
Micropipette  
1.5 mL centrifuge tubes  
Centrifuge

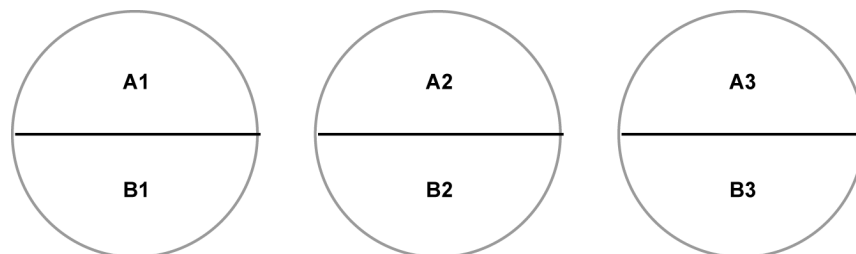
### **Objectives**

1. Be able to describe the principle of Y2H for protein-protein interaction assay
2. Be able to perform sterile operation

## Procedure

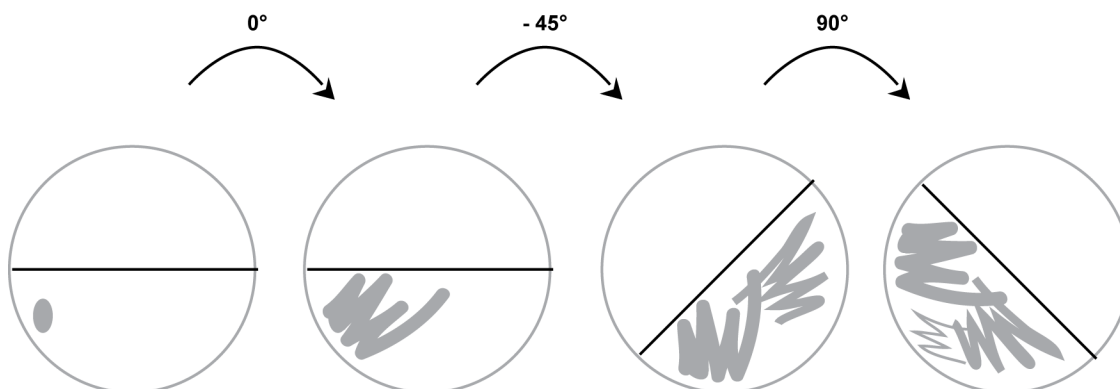
### Part 1: Yeast Mating Assay

1. Each person should have 3 SD-TRP-LEU plates.
2. Draw a line directly down the middle of the bottom of the plates (the side with the agar on it).
3. Each combination of bait and prey needs to be spread on a plate. Label each plate as shown in Figure 15-3.



**Figure 15-3.** Culture arrangement of mated cells on SD-W-L plates

4. Streak 50  $\mu$ L of each bait and prey combination (See Figure 15-4). 0°



**Figure 15-4.** Cell suspension streak diagram.

- 4.1. Bend the 5 mm tip end of a sterile 200  $\mu$ L tip by gently pushing it on the tip rack. Use this tip to pipette the mated cells prepared by your TA to mix and aliquot 50  $\mu$ L to one end of the designated area of SD-W-L medium prepared in Step 3.
- 4.2. Apply the same bent tip to streak (draw) 5-6 lines of cell suspension on the designated medium area from one end to the other.
- 4.3. Lift the tip and touch it to the end of the last line you just streak. Make a 45 degree turn and further streak 5-6 non-overlapped lines of cell suspension.

4.4. Repeat 4.3 once.

**Make sure to keep the mated cell suspension on its designated area and cells should be streaked from high density to low density.**

**Questions**

1. Why are mated yeast cells cultured on SD-Trp-Leu media?
2. What are other ways to determine protein-protein interactions other than yeast two-hybrid assays?

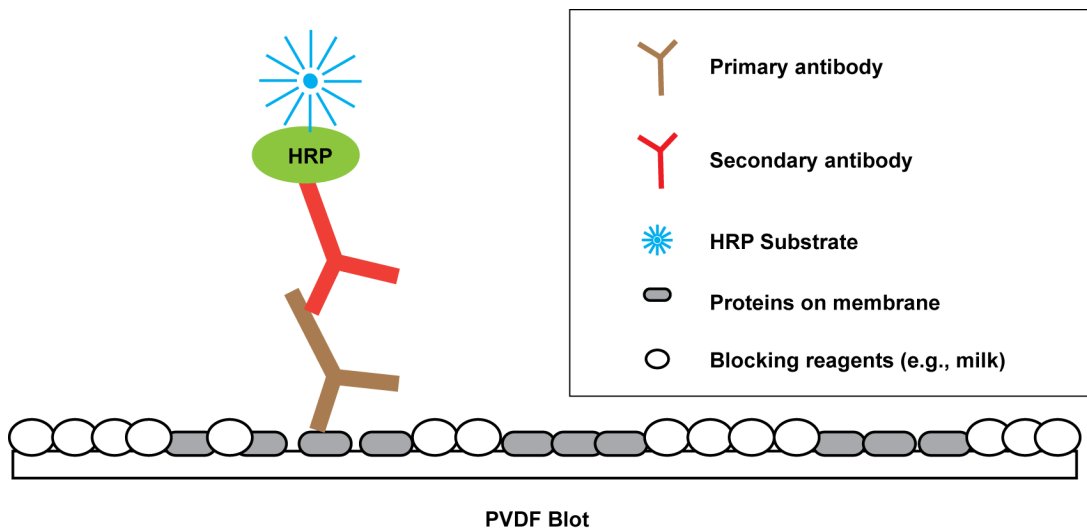
**(Notes)**

**(Notes)**

## Lab 16: Immuno-Blotting (3 hours)

### Introduction

Western blotting is often used to identify a specific protein in a mixture of proteins. In the last lab, we separated the total plant protein in a SDS-PAGE gel based on protein size. The resolved proteins were further transferred on a piece of PVDF membrane. However, the total protein electrophoresis patterns shown by Ponceau S staining does not tell us any difference between DAL co suppression mutant, DAL overexpression transgenic plant, and wild type plants. In order to examine the expression pattern of DAL proteins, a protein immunoblotting assay needs to be done. In protein immunoblotting assay (Figure 16-1), a primary antibody specific to a target protein is first incubated with the transferred protein membrane in TBST (Tris-buffered saline, 0.1% Tween 20). Specific recognition of primary antibody with antigens on the target protein is the major factor that determines the specificity and sensitivity of immunoblotting assay. Therefore, primary antibody is expensive. Some primary antibodies are directly associated with horse radish peroxidase, serving as a reporter enzyme to display the position where antibody and antigen interact on the membrane. However, in many cases, the recognition signal of primary antibodies is further amplified by a secondary antibody, which is raised to specifically react to immune globulin chains of the primary antibody. HRP is conjugated with the secondary antibody for displaying the signal. In some case, HRP is replaced with alkaline phosphatases for reporting the immunoblotting signal.



**Figure 16-1.** A schematic diagram demonstrating the detection of a target protein by immune blotting analysis

### Reagents

1x TBST buffer  
3% BSA (5% milk) TBST buffer  
Antibody  
Methanol  
X-ray paper

## Lab 16: Immuno-Blotting (3 hours)

---

Silver solution  
Membrane from Protein Transfer (previous lab)  
Film fixing solution  
Yeast mating plates  
X- $\alpha$ -Gal indicator plates

### **Equipment**

Pipette  
Pipette Tips  
Shaker  
Glassware for membranes  
10 mL Graduated cylinders

### **Objectives**

1. Be able to describe the principle of immunoblotting assay
2. Be able to handle antibodies
3. Be able to use dark room

### **Procedure**

#### **Part 1: Immuno-blotting**

1. Rinse membrane with methanol (~ 10 seconds)
2. Shake membrane in 10 mL TBST buffer for 2 minutes. Dispose of buffer.
3. Repeat step 2.
4. Shake membrane with 3% milk TBST buffer for about 40 minutes (**Blocking**)
5. Dump blocking buffer and add in diluted primary antibody. Shake for 1 hour.  
  
To dilute antibody (1:2500 dilution):  
Add 6 mL 5% milk to 4 mL TBST buffer  
Add 4  $\mu$ L concentrated antibody to 3% TBST buffer
6. Dispose of primary antibody solution and wash with TBST buffer for 10 minutes. Dispose of buffer.
7. Do the TBST buffer wash a second time for 10 minutes. Dispose of buffer.
8. Add 5% BSA TBST buffer blocking for about 15 minutes. Dispose of blocking buffer.
9. Add secondary antibody and shake for about 40 minutes.  
  
To dilute antibody (1:10000 dilution):  
Add 6 mL 5% milk to 4 mL TBST buffer  
Add 1  $\mu$ L concentrated antibody to 3% TBST buffer
10. Dump the secondary antibody, wash with TBST buffer 3 times, at 5 minutes each.



11. Keep the membrane in 1 x TBST at 4C if you run out of time.

**Part 2: HRP Display**

1. Wash the membrane twice more with 1 x TBST at room temperature.
2. Mix with 3 mL SuperSignal West Pico Chemiluminescent Substrates (freshly prepared by TA) for 5 min at room temperature.
3. Take the membrane out with tweezers and place it between two pieces of transparency film.
4. Head to the dark room on the fifth floor with the membranes.
5. Expose the membrane to X-ray film membrane for about 2 seconds.
6. Make sure to mark the bottom right corner to be sure of the orientation
7. Soak the X-ray film in a silver solution until development occurs (about 1-2 minutes)
8. Dip the X-ray film into water to rinse off the silver solution (10-15 seconds)
9. Set X-ray film into the film development solution

**(Notes)**

## Lab 17: Semi quantitative Y2H interaction assay (2 hours)

### Introduction

See Lab 15.

### Reagents

Yeast mating plates  
SD-A-H-W-L selection media

### Equipment

Pipette  
Pipette Tips  
1.5 mL tubes  
Spectrometer

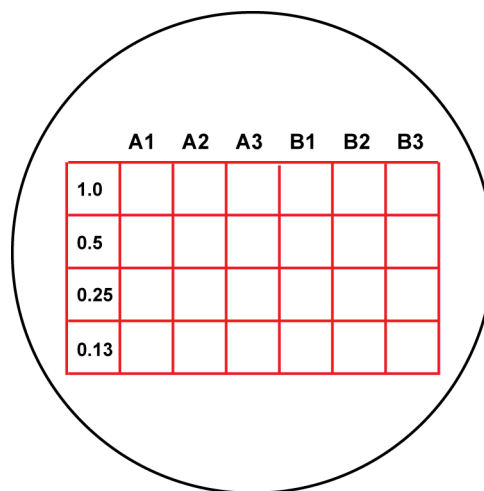
### Objectives

See Lab 15.

### Procedure

#### Part 1: Prepare plates for Y2H Interaction Assay

1. Check yeast colonies formulated on the mating plates. You should see a yeast cell lawn area and individual colonies when the cells are diluted by streak.
2. Each student will receive one SD-A-H-W-L plate. Use a mark pen to draw 4 (rows) x 6 (columns) = 24 grids at the bottom of the plate. I would recommend drawing 1 cm x 1 cm = 1 cm<sup>2</sup> for each grid (Figure 17-1).



**Figure 17-1.** Grid module for semi Y2H interaction assay

### Part 2: Make yeast cell suspensions with the same concentration

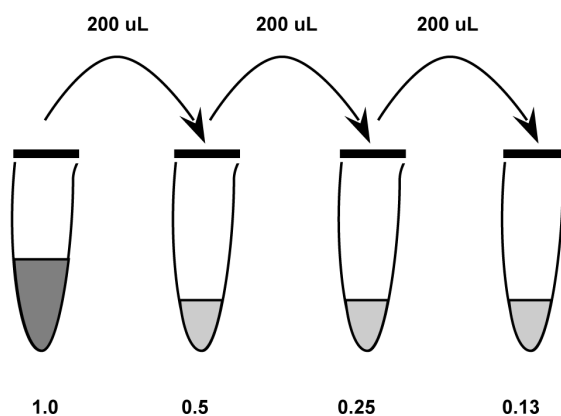
1. Using a sterile pipette tip to collect ~5 colonies and suspend the cells in 1 mL sterile water in 1.5 ml tube for one bait-prey pair.
2. Aliquot 500  $\mu$ L cell suspension into a second 1.5 mL tube and keep the remaining cells in a sterile condition.
3. Add 500  $\mu$ L H<sub>2</sub>O into the second 1.5 mL tube containing the aliquot yeast cells.
4. Repeat 1-3 for the rest 5 pairs of mated yeast cells.
5. Record the optical density (OD) value for each of your 1 mL cell suspension under 600 nm using a spectrometer in Table 17-1.
6. Calculate the dilution factor (DF) for each pair of mated cells to bring down the yeast concentration to OD<sub>600</sub> = 1.
7. Calculate the amount of H<sub>2</sub>O and the original yeast cell suspension (keep in mind, Tube 1) to make 500  $\mu$ L cell suspension with the final concentration of OD<sub>600</sub> = 1.

**Table 17-1. Equalize the number of mated cells**

Mated cell	Tube 1 OD <sub>600</sub>	DF	500 $\mu$ L cell suspension	
			Tube 1 Cell ( $\mu$ L)	Sterile H <sub>2</sub> O ( $\mu$ L)
A1				
A2				
A3				
B1				
B2				
B3				

### Part 3: Make serial dilution of yeast cells

1. Prepare 3 tubes for each pair of mated cells. Label each tube with the cell name and one of the four dilutions (0.5, 0.25, 0.13) as seen in Figure 17-2.



**Figure 17-2.** A schematic diagram showing the process of serial dilution of cell suspension

2. Add 200  $\mu$ L sterile H<sub>2</sub>O in each tube.
3. For each 500  $\mu$ L cell suspension prepared in Part 2, resuspend and aliquot 200  $\mu$ L cells to mix with 200  $\mu$ L sterile H<sub>2</sub>O in the 1.5 mL tube labeled with 0.5 dilution. Keep the remaining cell suspension uncontaminated!
4. Mix the diluted cells thoroughly and aliquot 200  $\mu$ L cells to mix with 200  $\mu$ L sterile H<sub>2</sub>O in the 1.5 mL tube labeled with 0.25 dilution.
5. Repeat 4 to finish all the dilutions (Figure 17-1).
6. Repeat 3-5 to finish the 3 dilutions for the rest 5 pairs of mated cells.

#### **Part 4: Spot yeast cells on selection media**

1. For the plate prepared in Part 1, label the columns as the name of 6 pairs of mated cells and label the rows as the 4 concentrations of cells (OD<sub>600</sub> = 1, 0.5, 0.25, 0.125) prepared in Part 3.
2. For each column (i.e., each pair of mated cells), from low concentration (OD<sub>600</sub> = 0.125) to high concentration (OD<sub>600</sub> = 1), spot 5  $\mu$ L cell suspension in each corresponding grid.
3. After you finish spotting, leave the cell on plate and make sure the liquid is completely absorbed into the media.
4. Culture the cells at 30°C for 2-4 days with the plate upside down to avoid accumulation of condensation water.

#### **Questions**

1. Which cells are mated from pGBKT7-F-box and pGADT7-ASK1? Why?

**(Notes)**

## **Lab 18: R Programming (3 hours)**

### **Introduction**

R is a derivative of the S programming language, an early statistical language developed at Bell Labs for data analysis, statistical modeling, simulation and graphics. However, due to its license limit, only very few people can access S. Since R is a GNU (an operating system that is free software) package and freely accessible under the GNU General Public License, it has been broadly accepted in academia. In addition to its source code, several pre-compiled binary versions are also available for different operating systems, including Windows, Mac and Linux.

R was created by Ross Ihaka and Robert Gentleman at the University of Auckland, New Zealand. It is so named because of the first two R authors and functioning similarly to S language. Since it was released in 1994, R has been widely distributed as an effective tool in data analysis, graphics, machine learning, and many other statistical packages because of its open source programming system. However, it has also its shortcoming. It consumes memory quickly, i.e., it requires a large memory to handle big dataset (like sequences datasets we have learned in our first bioinformatics section at the beginning of this class).

In this section, we will learn some basic R programming, most of which you may have already learned from your statistics classes, and learn how to use a bioconductor package, a free, open source and open development software project for the analysis and comprehension of big biological data.

### **Objectives**

- 1) Be capable of applying a short R script (10-20 statements) to acquire output.
- 2) Be capable of install an R and Bioconductor package

### **Software and databases**

CentOS  
Putty on Windows  
Terminal on Mac

### **Equipment**

Dell PowerEdge R710 Server X5670 2.93GHz 24-Cores 64GB 9-TB (RAID 5)  
Personal Computers

### **Part 1: Open R**

Many of you may know R-studio, a graphical front-end interface of R. However, I prefer to use a command line interface. If you feel more comfortable with R-studio, you have to use your own personal computer.

1. Log into your user account on the server.
2. Type R to enter the command line interface of R, which first prints out the current version of R followed by a command line prompt, >, in the terminal (see below).

R version 3.2.3 (2015-12-10) -- "Wooden Christmas-Tree"  
Copyright (C) 2015 The R Foundation for Statistical Computing  
Platform: x86\_64-apple-darwin15.3.0 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.  
You are welcome to redistribute it under certain conditions.  
Type 'license()' or 'licence()' for distribution details.

Natural language support but running in an English locale

R is a collaborative project with many contributors.  
Type 'contributors()' for more information and  
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or  
'help.start()' for an HTML browser interface to help.  
Type 'q()' to quit R.

>

Now, it is ready for you to start R coding.

## Part 2: Regular R Commands

### 1. R as a calculator

You may enter R commands or even expressions. If R can understand your codes, it will print out the result instantly. Thus, at some point, R can be used as a calculator.

Type the codes as follows following by enter to see what the results R produces. Again, similar to Perl programming in this manual, R codes are italicized and are highlighted with blue color. Comments for the code are following a pound sign and are in plain font.

<code>&gt;1+2-5</code>	<code>#Math function</code>
<code>&gt;1*2*3*4*8</code>	<code>#Math function</code>
<code>&gt;100/(10*2)</code>	<code>#Math function</code>
<code>&gt;sqrt(100)</code>	<code>#the square root of 10</code>
<code>&gt;pi</code>	<code>#Constant number</code>

### 2. plot

R is a great tool for graphics. Here I will introduce a simple graphic function, called two-dimensional plot.

```
>species<-c(1,2,3,4,5,6,7,8,9,10)
>gene_number<-c(21000,23000,33000,35000,45000,55000,65000,66000,67000,68000)
```

# <- gives the values on the right to the variable on the left. Different to Perl, R does not have declared types of variables, such as string, array, and hash in Perl. It gets the variables assigned by <- and can be changed dynamically.

```
>ls()
```



# Similar to the UNIX `ls` command, `ls()` lists all the variables currently available in the workspace.

```
>plot(gene_number ~ species, pch=16)
```

# You will see a two-dimensional graph popped up on screen in your local X-Windows in PC or XQuartz in MAC are active. If not, re install them in your local computer. `pch` is a plotting character. `pch=16` tells the plot to print out a solid black dot. If you want to know more detail about any R codes, you can type `?` followed by the code in the terminal. For example, type the code shown below to see what happens.

```
>?pch
```

#Type `q` to exit the help page in R.

# Now you may wonder how you will get the graph for your publication. You will use a new function to print out the graph into a graphic file. I always use PDF because it is editable in Adobe Illustrator, the best graphic editing software I like.

```
> pdf('gene_number_across_10_species.pdf',family='Times',height=10,width=10)
>plot(gene_number ~ species, pch=16)
>dev.off()
```

# `dev.off()` means close the device (graphic). Now, you do not see any new input in your X-Windows or XQuartz. Why? Because the file has been printed into a PDF file, named `'gene_number_across_10_species.pdf'`, in the same folder where you entered R. Type the following code to get where you are in R.

```
>getwd()
```

### 3. Quit R by typing the command below

```
>q()
Save workspace image? [y/n/c]: n # I do not save the workspace
```

# Wait a second, why should I quit R? No worries. You are a programmer now. You can enter R easily whenever you want. But you may want to learn how to switch the programs in the terminal. You may turn on many programs at the same time, but this is out of the scope of this class. The reason I want you to quit R is that you need to go back to the UNIX command lines to get the file you need.

### 4. UNIX Command Review

Use the 10 UNIX Commands you have learned to find and copy the `'gene_number_across_10_species.pdf'` file from the server to your local computer.

## Part 3: Microarray Data Retrieval and Visualization

The merit of R is that R can manipulate a large data matrix and perform a number of math and statistical analysis. Here we will use a microarray as an example to read, extract, and do some math and clustering analysis. You will touch several R packages.

**Step 1. Make a directory “*microarray\_data*” under your home directory.**

```
mkdir microarray_data
```

**Step 2. Enter the directory of “*microarray\_data*”**

```
cd microarray_data
```

**Step 3. Go to R**

```
R
```

Step 4. Data retrieval. Many of you may have known next-generation sequencing-based transcriptomic analysis, i.e., RNA-Seq. You may argue why bother microarray data since this technology is expired. This is indeed not true. On the perspective of bioinformatics, much information is yet not discovered from the previous microarray data. Here, we will apply an R package “GEOquery” to download a series record of microarray data, GSE5632, from the National Center for Biotechnology Information (NCBI) Gene Expression Omnibus (GEO) (<https://www.ncbi.nlm.nih.gov/geo/>). GSE5632 recorded 66 transcriptomic profiles of *Arabidopsis thaliana* flowers and pollen grains at different developmental stages.

GEO is a genomic data repository publicly available to all scientists and readers worldwide. You may find enormous high-throughput functional genomics data, including those obtained by microarray, next-generation sequencing, mass-spectrometry proteomic sequencing, and other techniques. If you generate high-throughput functional genomics data, you may also want dump your data on GEO before you publish your data.

```
library(GEOquery)
```

#Like Perl, R uses this format to call a Comprehensive R Archive Network (CRAN) package module

```
gse <- getGEO("GSE5632",GSEMatrix=FALSE)
```

# Download the dataset, GSE5632, and save the data in variable, *gse*

```
names(GSMList(gse))
```

# List the samples saved in this series of expression data

**Step 5. Make a data matrix**

```
probesets <- Table(GPLList(gse)[[1]])$ID
data.matrix<-do.call('cbind',
                    lapply(GSMList(gse),
                          function(x) {tab <- Table(x)
                                         mymatch <- match(probesets,tab$ID_REF)
                                         return(tab$VALUE[mymatch])
                                       }
                          )
                    )

data.matrix <- apply(data.matrix, 2, function(x) {as.numeric ( as.character(x) ) })
```

```
data.matrix <- log2(data.matrix)
rownames(data.matrix) <- probesets
colnames(data.matrix) <- names(GSMList(gse))
```

You may get frustrated from so many commands. However, as an effective learner in molecular biology, you may just take this as a recipe to retrieve expression data from a *gse* dataset. If you forget how to do this in the future, you can always come back to find this recipe here 😊. You may completely understand the trick of this recipe someday in the future if you keep using the tools you learned today.

### Step 6. Heatmap.2

Many of you may have seen many colorful heatmap graphs in many publications. Some of you may wonder how these graphs are made. Here, we will first retrieve a small dataset (the first 50 rows, i.e., loci) from the data matrix obtained in Step 5. We will then apply **Heatmap.2** in an R package, *gplots*, to cluster the dataset based on the expression level and print out a color-coded data matrix into a PDF file.

```
library(gplots)

# call the R package

dataset<- data.matrix[1:50,]

# Retrieve the dataset

exp_range<-quantile(dataset, probs = c(30,60)/100)
lowest_exp<-min(dataset)
max_exp<-max(dataset)

# We split the expression data into three blocks, lowest to 30 percentile, 30 percentile to 60
percentile, and 60 percentile to the highest expression

my_palette<-colorRampPalette(c("blue","yellow","red"))(n=29)
col_breaks=c(seq(lowest_exp, exp_range[[1]],length=10),
              seq(exp_range[[1]]+0.01,exp_range[[2]],length=10),
              seq(exp_range[[2]]+0.01,max_exp,length=10))

# We assigned three colors, "blue","yellow","red", to the three blocks of expression data and the
color is gradually transited from one to the other according to the expression level.

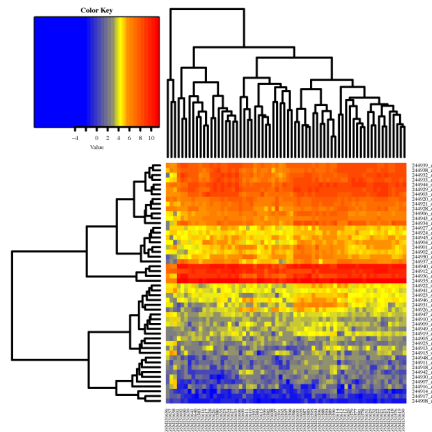
pdf("color-coded data matrix.pdf",family="Times",height=10,width=10)

par(mar=c(2.75,2.75,0.5,0.5),mgp=c(1.7,.7,0))

heatmap.2(dataset,
           col=my_palette,
           breaks=col_breaks,
           keysize=2,
           trace=("none"),
           margins=c(3,5),
           density.info=c("none"))
```

```
)  
  
dev.off()
```

# The dataset is clustered by *hclust* function in heatmap.2 and printed in a PDF file, *color-coded data matrix.pdf*. The function needs to shut off by *dev.off()* in order to accomplish the job. Now you should see a PDF file in your folder. Use UNIX Commands to find and copy the file from the server to your local computer as in Part 2. Check whether your result is the same as that shown in Figure 18-1.



**Figure 18-1.** Output of heatmap analysis of partial dataset of a serial microarray data, GSE5632.

### **Homework (practice, practice, and practices)**

1. I know some of you may have been stressed out. Please take it easy and practice what we have learned. Hope these codes will help your research in one day in the future. If you want more practices, please read one paper in my lab that has been published in the Plant Journal [Hua et al., (2018) 95, 296–311]. We have published a number of useful R-scripts in that paper along with the corresponding datasets. If you see any error I made, please do let me know. Please also cite the paper in your publication if you like it 😊.

**(Notes)**

**(Notes)**

## **Lab 19: Processing RNA-Seq Raw Data (2 hours)**

### **Introduction**

Next-generation sequencing technologies have transformed molecular biology over the past two decades. These technologies can be roughly divided into two categories—long read and short read. Long read sequencing technologies are exemplified by Pacific Biosciences SMRT sequencing (often referred to as PacBio Sequencing) and Oxford Nanopore's MinION sequencing. Both can sequence a long strand of DNA with the length that matches or even exceeds the best read-length detected by regular Sanger sequencing technology. Due to their high throughput long read sequencing capacity, these technologies have enabled superior genome assemblies. For example, they can sequence across long repeat regions in genomes, which is very challenging to assemble. The long sequencing results from these technologies have also demonstrate a far greater number of alternatively spliced gene forms. Albeit the advantages of long read sequencing technologies, next-generation sequencing projects are dominated by short read sequencing technologies that are epitomized by Illumina Hiseq and Miseq platforms. With an unrivaled cost per base sequenced and the ability to capture nearly all nucleotide sequences in a biological sample, NGS has empowered DNase hypersensitivity sequencing, Chromatin immunoprecipitation sequencing (ChIP-seq), whole genome Bisulfite-Seq, and most commonly thus far, RNA-Seq. RNA-Seq has been demonstrated to have a vastly reduced bias and error rate compared to other high throughput gene expression assays, such as microarray, Serial Analysis of Gene Expression (SAGE), and Cap Analysis of Gene Expression (CAGE). In this lab, we will learn how to process RAN-Seq raw data, which is recorded as FASTQ format, using *ask1* RNA-seq experiment as an example. In the next lab, we will further learn RNA-Seq data analysis using R to obtain a list of differential expressed genes (DEGs)

A raw RAN-Seq data contains erroneous reads and adapters ligated to the sequences to enable sample separation and sequencing. These two factors must be cleaned before any downstream analysis. We will use FastQC to evaluate sequence quality and trimmomatic to trim adapters and remove low quality reads. Once the reads are cleaned, a fast-short read alignment tool, STAR, is used to align the reads to the Arabidopsis reference genome. We finally apply HTSeq to count the number of reads aligned to each individual gene.

### **Objectives**

- 1) Be able to process NGS data typical of an RNA-seq analysis.
- 2) Understand the parameters, outputs and data types involved in an RNA-seq analysis.

### **Software and databases**

CentOS (with Trimmomatic, FastQC, STAR, Python, and HTSeq preconfigured)  
Putty on Windows  
Terminal on Mac

### **Equipment**

Dell PowerEdge R710 Server X5670 2.93GHz 24-Cores 64GB 9-TB (RAID 5)  
Personal Computers

### Part 1: Evaluate Data Quality

Log into the server under your username and locate the RNAseq directory. Navigate into the directory and choose one of the sequences.

Run FastQC on the chosen sequence files:

```
Fastqc name_of_your_file.fastq
```

When the program is finished, navigate into the new directory that has been created and type:

```
cat summary.txt
```

What does it mean?

### Part 2: Trim sequence data

Using the same sequence you had examined before, use the command below to trim you sequence data:

```
java -classpath /auto/rcf-proj/sa1/software/Trimmomatic-0.32/trimmomatic-0.32.jar PE  
-threads 16 -phred33 Input_file_R1.fastq Input_file_R2.fastq file_R1_trimmed.fastq  
file_R1_trimmed_unpaired.fastq file_R2_trimmed.fastq file_R2_trimmed_unpaired.fastq  
ILLUMINACLIP:TrueSeq.fa:2:30:10 LEADING:3 TRAILING:3 HEADCROP:10  
SLIDINGWINDOW:4:20 MINLEN:30
```

Now, run FastQC again and see what has changed.

### Part 3: Align cleaned reads to the references genome

To use STAR quick mapping, will need to generate a STAR formatted genome. This is required and is part of how STAR is able to run so quickly and accurately (this has been done):

```
STAR --runMode genomeGenerate --genomeDir TAIR_STARgenome/ --  
genomeFastaFiles TAIR10.fasta --runThreadN 2 --sjdbGTFfile TAIR10.gtf --  
genomeChrBinNbits 16
```

Once this is complete we will align the reads (pair end here) to the genome:

```
STAR --genomeDir TAIR_STARgenome/ --readFilesIn file_R1_trimmed.fastq.gz  
file_R2_trimmed.fastq.gz --runThreadN 2 --sjdbGTFfile TAIR10.gtf --outFilterType  
BySJout --outFilterMultimapNmax 10 --alignSJDBoverhangMin 1 --  
outFilterMismatchNmax 999 --alignIntronMin 150 --alignIntronMax 20000
```

```
# Oct 25 18:28:30 ..... Started STAR run  
# Oct 25 18:28:30 ..... Loading genome  
# Oct 25 18:28:30 ..... Processing annotations GTF  
# Oct 25 18:28:33 ..... Inserting junctions into the genome indices  
# Oct 25 18:28:38 ..... Started mapping  
# Oct 25 18:33:25 ..... Finished successfully
```



# This may take a little while.... When the alignment is complete, navigate into the new directory and identify the alignment file (a “.sam” or sequence alignment map file). These files are HUGE, unwieldy, and are not particularly handy for our purposes.

```
samtools view -Sb Aligned.out.sam > AlignmentwithReplicatename.bam
```

# *samtools* converts a *sam* file into a more manageable *binary sequence alignment map (bam)* file

```
samtools sort -n AlignmentwithReplicatename.bam AlignmentwithReplicatename.bam
```

```
# [bam_sort_core] merging from 13 files...
```

# sort the alignments so that HTSeq can process the alignment correctly.

```
htseq-count -f bam -m union -s no -q AlignmentwithReplicatename.bam TAIR10.gtf >  
ReplicatenameCounts.tab
```

# *htseq-count* counts the real number of counts to each locus. This is the digital expression for each gene!

Use *more* function to examine your result. Since the process is time and space consuming, we only use one RNA-Seq dataset as an example. If you like, you may practice more after the lab.

### **Homework (keep practicing):**

Get familiar with the data processing of RNA-Seq data.

**(Notes)**

## Lab 20: Differential Gene Expression Analysis (3 hours)

### Introduction

In this section, we continue to analyze the RNA-Seq data. In Lab 19, we obtained the digital expression value for each transcript. However, this is not our ultimate goal. What more interesting is to figure out the transcriptomic changes across samples. To do this, we will apply edgeR, a highly cited Bioconductor R package for statistical analysis of genomic data, to obtain a list of differential expressed genes (DEGs).

Bioconductor is a free, open source and open development software project. The packages in Bioconductor is primarily written in R and many are very useful for genomic data analysis. edgeR stands for “Empirical Analysis of Digital Gene Expression Data in R”. This package implements a range of statistical methods based on the negative binomial distributions, including empirical Bayes estimation, exact tests, generalized linear models and quasi-likelihood tests [Robinson et al., Bioinformatics, 2010 (26), 139-140]. It can be used for differential signal analysis of genomic data obtained by RNA-Seq, ChIP-seq, Bisulfite-seq, SAGE and CAGE.\

### Objectives

- 1) Understand the basic methods of running R-scripts
- 3) Be able to analyze transcript sequence read counts to determine differential expression.
- 4) Be able to export differential expression data as a “final product.”

### Software and databases

CentOS  
Putty on Windows  
Terminal on Mac

### Equipment

Dell PowerEdge R710 Server X5670 2.93GHz 24-Cores 64GB 9-TB (RAID 5)  
Personal Computers

### Part 1: edgeR based DEG Analysis

Like Perl, a # at the beginning of a line designates a comment. Keep in mind that the greatest strength of programming, like Perl and R, is the versatility provided by numerous packages/modules. R with only default packages is essentially a super-powered version of excel with a steeper learning curve. By installing packages, R can perform highly specific and powerful analysis (such as edgeR).

```
library(edgeR)
library(gtools)
library(gplots)
library(RColorBrewer)
```

## Lab 20: Differential Gene Expression Analysis (3 hours)

---

# Load R package that have been installed in our server. There are two ways to install R packages: `install.packages()` and `biocLite()`. Compiling programs could be nontrivial. Fortunately, you don't need to do this. It is beyond the scope of this class. All the packages you need have been already compiled in our server.

```
wt_r1<-read.table("wt_flower_r1_count.tab",header=FALSE)
wt_r2<-read.table("wt_flower_r2_count.tab",header=FALSE)
ask1_r1<-read.table("ask1_flower_r1_count.tab",header=FALSE)
ask1_r2<-read.table("ask1_flower_r2_count.tab",header=FALSE)
```

# Read count files processed in Lab 20

```
count_list<-list(wt_r1,wt_r2,ask1_r1,ask1_r2)
probesets<-wt_r1[,1]
count.matrix<-do.call('cbind',
                      lapply(count_list,
                             function(x)
                             {
                               mymatch <- match(probesets,x[,1])
                               return(x[,2][mymatch])
                             }
                      )
count.matrix <- apply(count.matrix,2,function(x) {as.numeric ( as.character(x) ) })
rownames(count.matrix) <- probesets
colnames(count.matrix) <- c("wt_r1","wt_r2","ask1_r1","ask1_r2")
rawdata<-read.table("gene_counts_file.tab",header = TRUE)
```

# Combine the processed transcriptomic data of four samples into one data matrix, *count.matrix* , and saved it as *rawdata*

```
rawdata<-na.omit(rawdata)
```

# omits loci with missing count

```
keep<-rowSums(cpm(rawdata)>2)>=3
rawdata<-rawdata[keep,]
table(keep)
```

```
#### output of table(keep)
# keep
# FALSE TRUE
# 16287 17320
```

# keep the expression level > 2 cpm (count per million reads)

```
group<-factor(c(1,1,2,2))
y<-DGEList(counts = rawdata,group=group)
design<-model.matrix(~group,data=y$samples)
design
```

```
#### output of design
```

```
#      (Intercept) group2
# wt_r1           1      0
# wt_r2           1      0
```

## Lab 20: Differential Gene Expression Analysis (3 hours)

---

```
# ask1_r1      1    1
# ask1_r2      1    1
# attr(,"assign")
# [1] 0 1
# attr(,"contrasts")
# attr(,"contrasts")$group
# [1] "contr.treatment"

#####

y<-calcNormFactors(y)
y$samples

#### output of y$samples
#      group lib.size norm.factors
# wt_r1      1 17752578  0.9614987
# wt_r2      1 16821827  0.9685758
# ask1_r1     2 17405658  1.0327091
# ask1_r2     2 19576785  1.0397758

#####

y<-estimateGLMCommonDisp(y,design,verbose=TRUE)
y<-estimateGLMTrendedDisp(y,design)
y<-estimateGLMTagwiseDisp(y,design,prior.df=20)

# statistical analysis process by edgeR

corraw_y_counts<-cor(y$counts,use="na.or.complete",method="pearson")

heatmap.2(corraw_y_counts,
          col=brewer.pal(10,"Set3"),
          keysize=2,
          margins=c(10,10),
          trace=c("none"),
          dendrogram="column",
          density.info=c("none"),
          cexRow=1,
          cexCol=1)

# You may recall what this does from Lab 18.

fit<-glmFit(y,design)
lrt.1vs2<-glmLRT(fit,coef=2)

FDR<-p.adjust(lrt.1vs2$table$PValue,method="BH")
sum(FDR < 0.05)

#Differential Gene count

diff_genes<-topTags(lrt.1vs2,n=sum(FDR<0.05),adjust.method="BH",
                    sort.by="p.value")

diff_genes[1:5,]

#### output of diff_genes[1:5,]
# Coefficient: group2
#      logFC logCPM   LR   PValue   FDR
```

## Lab 20: Differential Gene Expression Analysis (3 hours)

---

```
# AT1G75870 -5.605026 6.135436 864.7183 4.589976e-190 7.949839e-186
# AT4G21895 -5.414704 5.730719 843.7974 1.621963e-185 1.404620e-181
# AT1G51250 -5.123334 5.916828 817.1419 1.011939e-179 4.741866e-176
# AT5G43300 -5.182856 5.634825 816.9841 1.095119e-179 4.741866e-176
# AT2G44560 -5.244938 5.794482 799.7842 6.011357e-176 2.082334e-172
```

```
summary(dt<-decideTestsDGE(lrt.1vs2), adjust.method="BH", p.value=0.05, lfc=1)
```

```
#### output of summary
```

```
#      group2
# Down   2007
# NotSig 13731
# Up     1582
```

# adjust.method: character string specifying p-value adjustment method. Possible values are "none", "BH", "fdr" (equivalent to "BH"), "BY" and "holm". See 'p.adjust' for details.

# p.value: numeric value between 0 and 1 giving the required family-wise error rate or false discovery rate.

# lfc: numeric, minimum absolute log2-fold-change required.

```
DEgenes <-dim(lrt.1vs2$table)[1]
```

```
# DEgenes list
```

```
DIFF_Exp_List = topTags(lrt.1vs2, n = DEgenes)$table
```

```
DIFF_SIG <- DIFF_Exp_List [DIFF_Exp_List$PValue <= 0.05, ]
```

```
DIFF_FDR <- DIFF_Exp_List [DIFF_Exp_List$FDR <= 0.05, ]
```

# Keep rows with Pvalue or FDR less than or equal to .05 respectively. In general, this can be considered as the list of Differentially Expressed Genes (DEG)

```
write.csv(DIFF_FDR, file="ask1_wt_DEG_FDR.csv")
```

```
#Exporting Data
```

### **Homework (100 pts):**

You now have the differential expression data generated from an RNA-Seq experiment. If your lab plans to do this experiment, it likely needs to cost thousands of dollars. Currently, the best price for sequencing one RNA sample needs ~\$200. With bioinformatics service, it needs more than that. Please tell me how you will do with the list of DEGs? One approach would be to do gene ontology (GO) enrichment analysis. Please use the data you obtained to compare the GO differences, if any, between two lists of DEGs that are upregulated and down regulated in the mutant. You may find that AmiGO2 (<http://amigo.geneontology.org/amigo/landing>) is very helpful to search GO enrichment in a list of genes. You may also tell me other cool analyses if you can. Keep in mind, there is no absolute answer in this homework.

**(Notes)**

**(Notes)**



## Lab 21: RNA Isolation (3 hours)

### **Introduction**

There are three major types of RNA: ribosomal RNA (rRNA), transfer RNA (tRNA) and messenger RNA (mRNA). Only mRNA is able to translate a protein. A eukaryotic gene itself includes coding and non-coding regions. The coding region of a gene comprises one to multiple DNA fragments that are transcribed into the final mRNA products. These DNA fragments are called exons. The non-coding regions that are synthesized in pre-matured RNA and later removed from the final mRNA products are called introns.

The mRNA is transported into cytoplasm where it is associated with ribosome to assemble protein synthesis machinery to guide protein synthesis, called translation. As a consequence, the number of mRNA transcripts of a gene generally determines the abundance of its protein product. Because mRNA can be reverse transcribed into complementary DNA (cDNA), which is more stable than mRNA, mRNA molecules are usually converted into cDNAs for downstream studies, including molecular cloning and gene expression analysis.

In this lab, we will isolate total inflorescence RNAs from Arabidopsis wild type and *ask1* mutant plants. Each student will be assigned to extract one RNA sample, which is harvested by your TA from either wild type or the *ask1* mutant. In your final result obtained from this lab through the next two labs, you will determine to which genotype your RNA sample belongs.

### **Objectives**

1. Be able to describe the steps that can significantly reduce the yield of RNA (i.e. increase RNA degradation).
2. Be able to isolate total RNA with high quality

### **Reagents**

Liquid nitrogen  
Buffer RLT  
Ethanol (96-100%)  
Buffer RW1  
Buffer RPE  
RNase-free water

### **Equipment**

1.5 mL microcentrifuge tubes  
Small, blue pestles  
Vortex  
56°C hot water bath  
Razor blades  
Centrifuge  
QIAshredder spin column  
2 mL collection tubes  
RNeasy mini spin column

## **Procedure**

### **Part 1 RNA Isolation (Adapted from the manufacture manual)**

1. Grind ~100 mg sample completely in 1.5 mL centrifuge tube taken from liquid nitrogen using the blue pestle. **Do not let sample thaw and move immediately to step 2.**

2. Add 350  $\mu$ L Buffer RA1 and 3.5  $\mu$ L  $\beta$ -mercaptoethanol ( $\beta$ -ME) to 100 mg tissue and vortex vigorously.

3. Reduce viscosity and clear the lysate by filtration through NucleoSpin®Filter (violet ring): Place NucleoSpin® Filter in a Collection Tube (2 mL), apply the mixture, and centrifuge for 1 min at 11,000 x g. Transfer the filtrate to a new 1.5 mL microcentrifuge tube.

**Do not disturb the pellet of cell debris at the bottom of the collecting tube, which may be visible after centrifugation.**

4. Add 350  $\mu$ L ethanol (70 %) to the homogenized lysate and mix by vortexing (2 x 5 s).

After addition of ethanol a stringy precipitate may become visible which will not affect the RNA isolation. Be sure to disaggregate any precipitate by mixing and load all of the precipitate on the column as described in step 5. Do not centrifuge the ethanolic lysate before loading it onto the column in order to avoid pelleting the precipitate.

5. For each preparation take one NucleoSpin® RNA Plant Column (light blue ring) placed in a Collection Tube and load the lysate. Centrifuge for 30 s at 11,000 x g. Place the column in a new Collection Tube (2 mL).

Maximum loading capacity of NucleoSpin®RNAPlant Columns is 750  $\mu$ L. Repeat the procedure if larger volumes are to be processed.

6. Add 350 $\mu$ L MDB (Membrane Desalting Buffer) and centrifuge at 11,000 x g for 1 min to dry the membrane.

Salt removal will make the following rDNase digest much more effective. If the column outlet has come into contact with the flow-through for any reason, discard the flow-through and centrifuge again for 30 s at 11,000 x g.

7. Prepare DNase reaction mixture in a sterile 1.5 mL microcentrifuge tube: For each isolation, add 10  $\mu$ L reconstituted rDNase to 90  $\mu$ L Reaction Buffer for rDNase. Mix by flicking the tube.

Apply 95  $\mu$ L DNase reaction mixture directly onto the center of the silica membrane of the column. Incubate at room temperature for 15 min.

8. Wash and dry silica membrane.

8.1. Add 200  $\mu$ L Buffer RAW2 to the NucleoSpin® RNA Plant Column. Centrifuge for 30 s at 11,000 x g. Place the column into a new Collection Tube (2 mL).

8.2. Add 600  $\mu$ L Buffer RA3 to the NucleoSpin® RNA Plant Column. Centrifuge for 30 s at 11,000 x g. Discard flow-through and place the column back into the Collection Tube.

**Note: Make sure that residual buffer from the previous steps is washed away with Buffer RA3**, especially if the lysate has been in contact with the inner rim of the column during loading of the lysate onto the column. For efficient washing of the inner rim, flush it with Buffer RA3.

8.3. Add 250  $\mu$ L Buffer RA3 to the NucleoSpin® RNA Plant Column. Centrifuge for 2 min at 11,000 x g to dry the membrane completely. Place the column into a nuclease- free Collection Tube (1.5 mL, supplied with the kit).

If for any reason, the liquid level in the Collection Tube has reached the NucleoSpin® RNA Plant Column after centrifugation, discard flow-through, and centrifuge again.

9. Add 60  $\mu$ L RNase-free H<sub>2</sub>O (supplied with the kit) in the center of the column and centrifuge at 11,000 x g for 1 min.

10. Leave the RNA-containing tube on ice.

## Part 2: RNA Yield

1. Lab heads up to the Genomics Facility on 5<sup>th</sup> floor of Porter Hall

2. Bring isolated RNA from Part 1 on ice.

3. Set Nanodrop to “Nucleic Acid” and “RNA” mode.

4. When program first opens, load 1.5  $\mu$ L distilled water onto the pedestal and hit “OK”. This initializes the instrument

5. Wipe off the pedestal and load 1.5  $\mu$ L distilled water onto the pedestal again. Hit “Blank”. In the Sample # box, put your initials.

6. Wipe off the pedestal and load 1.5  $\mu$ L of your sample onto the pedestal. (1 minute)  
Click “Measure” (1 minute)

7. Record the number which appears in the green box with the label “ng/ $\mu$ L”. Also record the number in the 260/230 box, this determines how much contamination there is.

8. Calculate the RNA concentration and yield.

**Table 19.1 Concentrations of RNA samples purified**

Sample	RNA
Concentration ( $\mu$ g/ $\mu$ L)	
OD260/ OD280	
OD260/ OD230	

## Questions

1. How will you distinguish the RNA samples that are isolated from wild type and *ask1* mutants?

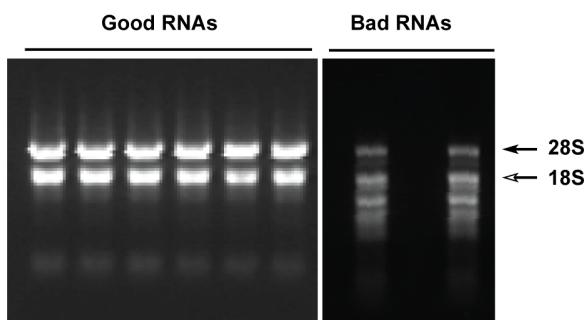
**(Notes)**

## Lab 22: RNA Quality Examination (2 hours)

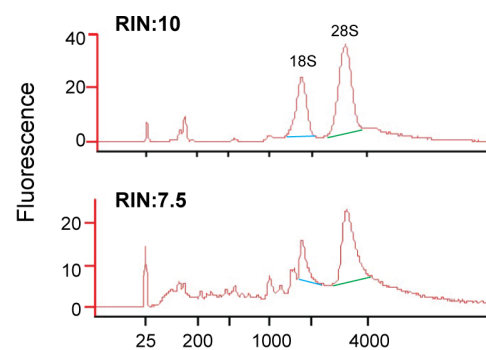
### Introduction

Since the downstream gene expression analysis is expensive, RNA samples should always be of high quality. Before gene expression analysis, the purity and integrity of total RNA molecules should be carefully checked. DNA contamination and RNA degradation are two common scenarios that can significantly reduce the quality of your sample. The RNA quality check generally includes yield detection, DNA contamination, and RNA Integrity. RNA yield can be determined based on RNA absorbance detected in a nanodrop spectrometer. Although the OD260/280 and OD260/230 ratios can roughly determine the contamination of proteins and small carbohydrates, it cannot rule out the contamination from DNA molecules and verify the integrity of RNA molecules. RNA electrophoresis gel can roughly determine the both integrity and DNA contamination. Since 28S RNAs are more sensitive to degradation, the ratio of 28S over 18S bands is always used to suggest the integrity of RNA molecules (Figure 21-1). This ratio is also known as RNA integrity number (RIN). If a high molecular weight band is present in your RNA electrophoresis gel, it tells a genomic DNA contamination.

**A**



**B**



**Figure 21-1.** An example of A) RNA gel electrophoresis on 1% agarose gel (Courteously Provided by Madhura Yapa from Hua Lab), and B) Bioanalyzer chromatograms of RNAs (Hua Lab)

Since agarose gel electrophoresis cannot sufficiently prevent post-harvest RNA degradation and is also hard to quantify, the RNA samples can be loaded onto a Bioanalyzer chip to measure the quality of RNA samples more precisely (Figure 21-1 B).

### Objectives

1. Be able to determine the quality of RNA and recognize the three major types of rRNA.

### Reagents

RNA extracts  
1x TAE buffer  
Agarose powder  
GreenGlo™ Safe DNA Dye  
DNA molecular marker, 1 kb ladder  
6x Gel loading dye

### **Equipment**

Ice Bucket  
Pipettes  
Pipette tips  
Glass flask  
Microwave  
Gel trays  
Gel comb  
Gel electrophoresis apparatus

### **Procedure**

#### **Part 1: Make an Agarose Gel (3 students one gel)**

1. Make agarose gel by adding 0.4 g of agarose powder to 50 mL of TAE buffer. Heat the mixture in the microwave for 2-3 minutes bringing the solution to boil, making sure to swirl the mixture every few seconds. (10 ~ 20 minutes)
2. Once the mixture is cooled down to ~60° C (able to hold the flask with your naked hand), add 3 µL of GreenGlo™ Safe DNA Dye and pour into gel tray. Insert the gel comb and allow the gel to solidify for approximately 30 minutes. (Go to Part 2)
3. Place the gel into the electrophoresis apparatus. Make sure the gel is oriented in the right direction. Remove the combs and fill the apparatus with TAE buffer until it covers the gel.

#### **Part 2: RNA Agarose Gel Electrophoresis**

1. Mix 2 µg RNA (scale up the total volume to 10 µL + 2 µL 6x loading dye).
2. Heat the sample at 65°C to break down the secondary structure of RNA molecules.
3. Load samples.
4. Run the gel for 30 minutes at 100 v.
5. Image the gel picture under Bio-Rad gel doc 2000 system in Hua lab.

#### **Part 3: Bioanalyzer Analysis**

1. Take samples up to the Genomics Facility
2. The genomic facility will set up the Bioanalyzer for class to watch.

### **Questions**

1. Looking at the RIN score of our RNA samples, which sample has the best quality? Why?
2. What are the three peaks in an RNA chromatogram from bioanalyzer analysis?



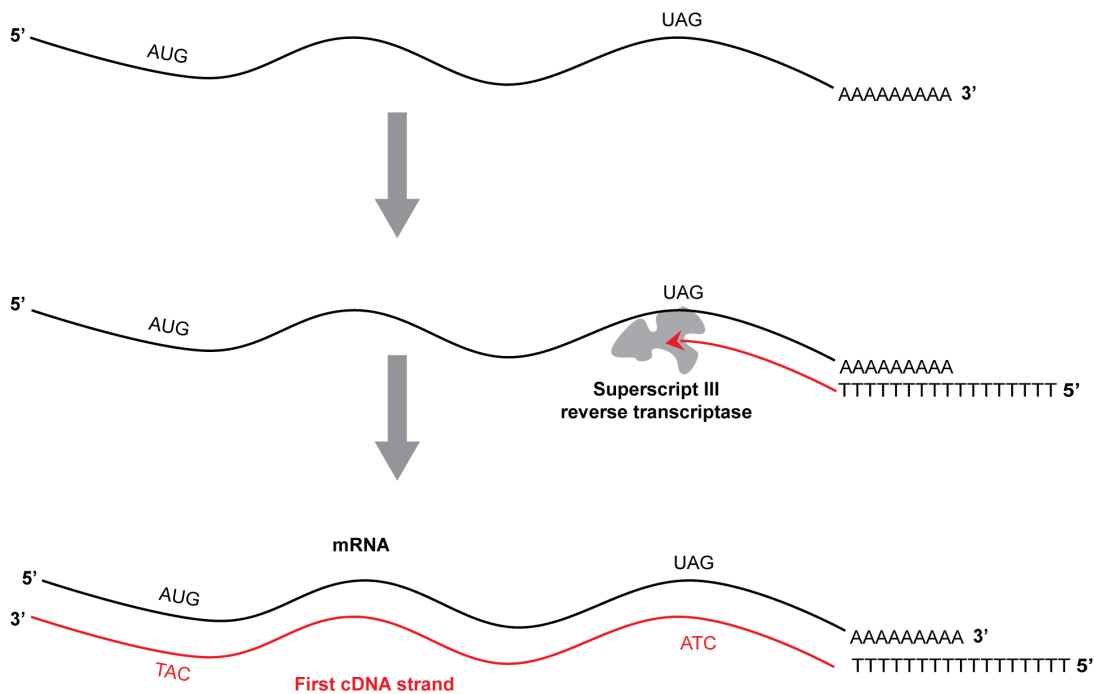
**(Notes)**



## Lab 23: cDNA Synthesis and RT-PCR (3 hours)

### Introduction

As mentioned previously, RNA can be used to determine gene expression. There are many methods to do this. Conventional approaches include northern blotting, reverse transcription polymerase chain reaction (RT-PCR), suppression subtractive hybridization (SSH), differential display PCR (DD-PCR), promoter-fusion analysis, in-situ RNA hybridization. To study the entire transcriptome, high-throughput transcriptome analyses include microarray, Serial Analysis of Gene Expression (SAGE), Cap Analysis of Gene Expression (CAGE), Expression Sequence Tags (ESTs), and next-generation sequencing-based RNA-Seq, etc. In this lab, we are going to create cDNA libraries by reverse converting mRNA strands into cDNAs using **Reverse Transcriptase**. The benefit of having cDNAs is that it is more stable than RNA and PCR analysis can be conducted on cDNAs.



**Figure 23-1.** A diagram showing the synthesis process of the first cDNA strand

### Objectives

1. Be able to handle RNA sample appropriately.
2. Be able to perform the cDNA synthesis with a team member.
3. Be able to set up PCR independently.

### **Reagents**

RNA samples (2 µg of RNA brought to 8 µL)  
10X DNase I Reaction Buffer  
DNase I, Amp Grade, 1 U/µL H<sub>2</sub>O  
25 mM EDTA  
Oligo(dT) 12-18 (500 ng/µL)  
10 mM dNTP  
Distilled H<sub>2</sub>O  
5X First-strand buffer  
0.1M DTT  
RNaseOUT™ RNase inhibitor  
SuperScript™ III  
10x buffer  
2.5 mM dNTP  
*ASK1* forward primer  
*ASK1* reverse primer  
*Actin II* forward primer  
*Actin II* reverse primer

### **Equipment**

Pipettes  
Pipette tips  
70°C hot water bath  
65 °C hot water bath  
50°C water bath  
Ice bucket  
Centrifuge  
Thermocycler

### **Procedure**

#### **Part 1: cDNA synthesis**

- 1) Add 1 µL 10X DNase I Reaction Buffer to the 8 µL RNA given by the TA
- 2) Add 1 µL DNase I, Amp Grade, 1 U/µL H<sub>2</sub>O to RNA sample
- 3) Tap and spin down.
- 4) Incubate tube for 15 minutes at room temperature.
- 5) Inactivate the DNase I by the addition of 1 µL of 25 mM EDTA solution to the reaction mixture.
- 6) Heat for 10 minutes at 65°C.
  - a. This denatures and removes contaminant DNA
- 7) Add the following to the 11 µL RNA sample tube:
  - a. 1 µL Oligo (dT) 12-18 (500ng/µL)
  - b. 1 µL 10 mM dNTP
  - c. 1 µL H<sub>2</sub>O
- 8) Heat at 65 °C for 5 minutes.
  - a. Denatures RNA

- 9) **Immediately** place sample on ice-water for 2 minutes.
  - a. By quickly putting in the ice, this prevents the secondary structures of RNA from forming.
- 10) Briefly centrifuge.
- 11) Add the following to RNA mixture:
  - a. 4  $\mu$ L 5X First-strand buffer
  - b. 1  $\mu$ L 0.1 M DTT
  - c. 1  $\mu$ L RNaseOUT™ RNase inhibitor (If have less than 50 ng RNA)
  - d. 1  $\mu$ L SuperScript™ III
- 12) Gently mix by pipetting gently up and down
- 13) Place in 50°C water bath for 50 minutes.
- 14) Place in 70° C water bath for 15 minutes (this stops the reaction).
- 15) Dilute your cDNA product 30 times (4  $\mu$ L cDNA + 116  $\mu$ L H<sub>2</sub>O).
- 16) Label and aliquot 14  $\mu$ L of the diluted cDNAs in 8 tubes

## Part 2: RT-PCR setup

1. Each cDNA template will be assigned to replicate 1 or replicate 2 according to the sheet handed out (TBA)
2. Each student will be assigned to do one gene replicate for each cDNA template. A list table will be provided (TBA).
3. Set up PCR master mix. Each student will need to make two master mixes, one for the assigned gene and one for the Actin II control. Make the master mix following Tables 23-1.

**Tables 23-1.** PCR master mix recipe ( $\mu$ L)

10 Reaction Master Mix for selected gene ( $\mu$ L)	
10x Buffer	25
2.5 mM dNTP	20
Gene forward primer	12.5
Gene reverse primer	12.5
Taq	10
H <sub>2</sub> O	110
<b>Total</b>	<b>190</b>

10 Reaction Master Mix for Actin II ( $\mu$ L)	
10x Buffer	25
2.5 mM dNTP	20
Actin II forward primer	12.5
Actin II reverse primer	12.5
Taq	10
H <sub>2</sub> O	110
<b>Total</b>	<b>190</b>

4. Write down the sequence of your samples for each tube of your strip tube according to the sample-spread sheet. Label one strip for your assigned gene and the other strip for the Actin II control.

5. Aliquot 19  $\mu\text{L}$  of master mix into each tube and then add 6  $\mu\text{L}$  of diluted cDNA for one PCR reaction.
6. Place the samples on a thermo cycler and perform PCR. 28 and 35 cycles will be used to amplify *Actin II* and *ASK1* genes, respectively.

**Questions**

1. Will RT-PCR be able to examine the expression level of rDNA genes? Why?

**(Notes)**



## **Lab 24: Continuance of cDNA Synthesis and RT-PCR (2 hours)**

### **Introduction**

See Lab 21

### **Objectives**

See Lab 21

### **Reagents**

See Lab 21

### **Equipment**

See Lab 21

### **Procedure**

#### **Part 1: Make an Agarose Gel (2 students one gel)**

1. Make agarose gel by adding 0.4 g of agarose powder to 50 mL of TAE buffer. Heat the mixture in the microwave for 2-3 minutes bringing the solution to boil, making sure to swirl the mixture every few seconds. (10 ~ 20 minutes)
2. Once the mixture is cooled down to ~60° C (able to hold the flask with your naked hand), add 3 µL of GreenGlo™ Safe DNA Dye and pour into gel tray. Insert the gel comb and allow the gel to solidify for approximately 30 minutes. (Go to Part 2)
3. Place the gel into the electrophoresis apparatus. Make sure the gel is oriented in the right direction. Remove the combs and fill the apparatus with TAE buffer until it covers the gel.

#### **Part 2: Gel Electrophoresis of RT-PCR Product**

1. Mix 6 µL with 25 µL **RT-PCR Product directly in the PCR tubes.**
2. Load 10 µL sample in each well and record the loading sequence
4. Run the gel for 30 minutes at 100 v.
5. Image the gel picture under Bio-Rad gel doc 2000 system in Hua lab.

### **Questions**

1. What do you see the differential expression pattern of both *ASK1* and *Actin2* genes? What does your result suggest? Which genotype was your RNA sample extracted from?

**(Notes)**



## **APPENDIXES**

<b>Appendix 1. Making agarose gels</b>	<b>123</b>
<b>Appendix 2. Making <i>Escherichia coli</i> growth media</b>	<b>124</b>
<b>Appendix 3. Plant DNA Extraction Buffer</b>	<b>125</b>
<b>Appendix 4. Making Protein Gel using Bio-Rad Mini-Protein-3 System</b>	<b>126</b>
<b>Appendix 5. Making Yeast Media</b>	<b>129</b>
<b>Appendix 6. Yeast Transformation</b>	<b>130</b>
<b>Appendix 7. To Do List for Teaching Assistants</b>	<b>131</b>
<b>Appendix 8. Lab Supply</b>	<b>139</b>
<b>Appendix 9. Unix-Linux Command Reference</b>	<b>147</b>
<b>Appendix 10. Vim Cheat Sheet</b>	<b>149</b>
<b>Appendix 11. Perl Cheat Sheet</b>	<b>153</b>
<b>Appendix 12. R Cheat Sheet</b>	<b>155</b>

**(Intentionally Left Blank)**

## **Appendix 1. Making agarose gels**

1. Make 1 x TAE running buffer by diluting 20 mL 50XTAE stock solution into 1,000 mL distilled water.
2. Aliquot 50 mL of TAE buffer in a 250 mL flask.
3. Add 0.4 g of agarose powder into 50 mL of TAE buffer. Heat the mixture in microwave for 1 to 2 minutes while paying attention to the solution. Stop heating when the solution is just about to boil.
4. Take out the solution with a HOT HAND. Gently swirl the mixture. Heat the solution again with ~10 sec.
5. Repeat Step 4 2-3 times until the agarose granules are completely melted.
6. After cooling down the solution on bench to ~60° C (able to hold the flask with your naked hand), add 3 µL of GreenGlo™ Safe DNA Dye and gently swirl to mix
7. Pour the melted gel solution into a sealed gel tray. Insert an electrophoresis comb and let the gel to solidify for approximately 40 minutes.
8. Place the gel into an electrophoresis apparatus. Make sure the gel is oriented in the right direction with the loading well close to the cathode side. Gently remove the comb and fill the apparatus with TAE buffer until it just covers the gel for approximately 5 mm.

**Appendix 2. Making *Escherichia coli* growth media****1) *Luria broth (LB) liquid medium* (100 ml)**

Dissolve 2.5 g LB-Broth directly in 100 ml H<sub>2</sub>O in a 125 ml Wheaton medium bottle. Close the cap and then turn back one quarter of turn to lose the cap. Autoclave the medium for 20 min at 121°C. After cooling down the medium at room temperature (leave the medium on bench overnight), close the cap and store the medium on bench shelf.

**2) *LB plates with ampicillin* (100µg/ml)**

Mix 20 g LB-agar to 500 ml H<sub>2</sub>O in a one-liter flask with a stirrer bar. Prepare two flasks with in total one-liter LB-agar medium. Autoclave the medium after LB is dissolved and the agar is well suspended. Allow the medium to cool down to 50~60°C (able to hold the flask with your palms) before adding ampicillin to a final concentration of 100 µg/ml (500 µL of 100 µg/µL ampicillin stock solution). If you don't have time to wait the medium to cool down, you can leave it in a 50° C incubator for several hours.

Pour ~20-25 mL of medium into one 100 mm petri dish. In total, make about 40-50 plates. Let the agar harden on bench over night to remove condense water. Wrap the plates with the original sleeve bag and store them at 4°C for up to 1 month or at room temperature for up to 1 week.

**3) *LB plates + ampicillin/IPTG/X-Gal* (1 hour)**

Take LB plates (1 plate for one student) with ampicillin from 4°C cold room and warm them up to room temperature. Add 100 µL of 100mM IPTG and 20 µL of 50mg/ml X-β-Gal in the center of the medium, spread them with a spreader evenly over the surface of a LB-ampicillin plate. Let the solution to be completely absorbed for 30 minutes at 37°C prior to use.

**Appendix 3. Plant DNA Extraction Buffer***1. CTAB buffer 100ml*

Mix or dissolve the following chemicals except for  $\beta$  - Mercaptoethanol in a 150 mL glass beaker in order. After it is done, store the solution in a 125 ml Wheaton medium bottle.

40.0 mL	H <sub>2</sub> O
10.0 mL	1M TrisHCl (pH8.0)
4.0 mL	0.5M EDTA-Na <sub>2</sub>
28.0 mL	5M NaCl
2.0 g	CTAB (Hexadecyl trimethyl-ammonium bromide)
200 $\mu$ L	$\beta$ - Mercaptoethanol added just prior to use

*2. 1 M TrisHCl (pH 8.0)*

Dissolve 12.1 g of Tris base in 80 ml of H<sub>2</sub>O. Adjust pH to 8.0 by adding ~4 ml of concentrated HCL. Adjust the volume to 100 mL with H<sub>2</sub>O. Sterilize by autoclave.

## Appendix 4. Making Protein Gel using Bio-Rad Mini-Protein-3 System

### 1. Gel Cassette-Casting Frame Preparation (All glass plates should be clean and dry)

- 1.1. Place a Casting Frame (green color) upright on a flat surface with the pressure cams in an open position.
- 1.2. Select an appropriate Spacer Plate suitable for the desired gel thickness and place a Short Plate on top of it.
- 1.3. Slide the two glass plates into the Casting Frame while keeping the Short Plate towards you and the label on Spacer Plate up. Make sure both plates are at the same level at the bottom. Otherwise, leaking might happen during gel casting.
- 1.4. When the glass plates are in place, apply the pressure cams to lock the glass cassette assembly in the Casting Frame.
- 1.5. Place the Casting Frame into the Casting Stand. The bottom of the two plates should be tightly sealed with the rubber gasket on the Casting Frame after engaging the spring-loaded lever of the Casting Stand onto the Space Plate

Repeat steps B-E for the second gel

### 2. Casting Gel

- 2.1. Place one comb completely into the assembled gel cassette. Mark the short plate ~0.5-1 cm below the comb teeth as the top position that the resolving gel reaches.
- 2.2. For making two 10% resolving gels, mix the components in Table S1 in order in a 50 mL flask. Change pipette tips when pipette a different solution.

**Table S1.** Recipe for two 10% mini resolving gels

Reagents	Volume
H <sub>2</sub> O	5 mL
4X Resolving solution	2.5 ML
40% Acrylamide/Bis	2.5 mL
10% APS	50 µL
TEMED	10 µL
Total volume (mL)	10 mL

- 2.3. Using a 10 mL serology pipette to aliquot 5 mL resolving solution and add into the middle space of one Gel Cassette prepared in Step A. The level of resolving solution should not pass above the mark on the short plate labeled on Step 2.1. Gently add ~100-200 µL isopropanol to overlay the resolving solution.
- 2.4. Allow the gel to polymerize for 20 to 30 minutes.
- 2.5. Making two Stacking gel by mixing the components in Table S2 in order in a 50 mL flask. Change pipette tips when pipette a different solution.

**Table S2.** Recipe for two mini stacking gels

Reagents	Volume
H <sub>2</sub> O	2.5 mL
4X Stacking sol	1 mL
40% Acrylamide/Bis	0.5 mL
10% APS	30 $\mu$ L
TEMED	5 $\mu$ L
Total volume	4 mL

2.5 Take out the Gel Cassette-Casting Frame assembly, dump off the isopropanol overlay, add 2 mL stacking gel solution, and insert the gel comb with the appropriate thickness.

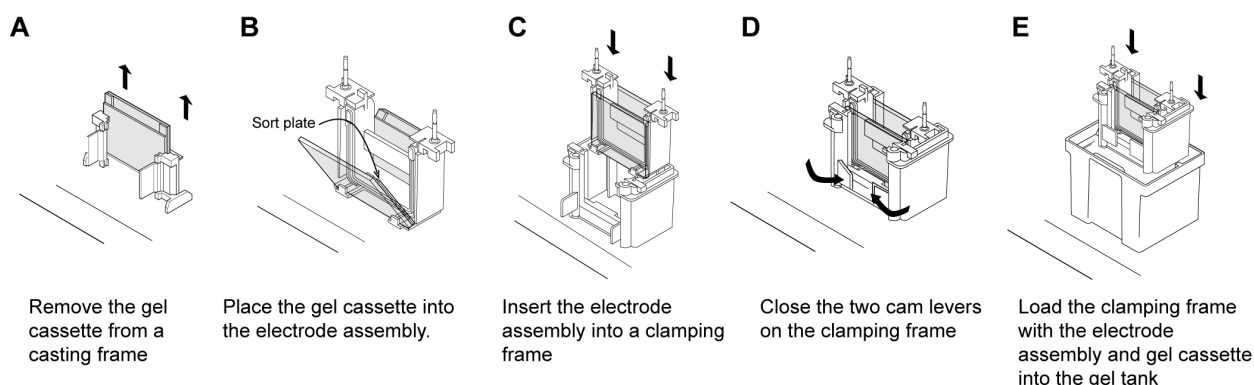
2.6 Allow the stacking gel to polymerize for 20 to 30 minutes.

2.7. Take out the Gel Cassette-Casting Frame assembly. Open the cams of the Casting Frame to release the Gel Cassette. If you don't need the gel right away, you may seal the Gel Cassette with the gel with Saran Wrap and store it at 4°C for about one week.

### 3. Assemble Mini-PROTEAN3 Electrophoresis Module

3.1. Take out the Gel Cassette, remove the comb gently, and rinse the wells quickly with distilled water

3.2. Follow Figure S1 to assemble a Mini-PROTEAN3 electrophoresis module. Add running buffer to the electrode assembly (top buffer chamber) until the sample wells are covered and ~200 mL running buffer to the Mini Tank (lower buffer chamber). If any leakage happens in the top buffer chamber, fill the running buffer in the lower buffer chamber to the same level as it is in the top buffer chamber.



**Figure S1.** Assemble a Mini-PROTEAN3 electrophoresis module (Adapted from Bio-Rad Mini-PROTEAN® 3 Cell Instruction Manual)

#### 4. Reagent Preparation

##### 4.1. 10% (w/v) SDS

Dissolve 10 g SDS in 90 mL water with gentle stirring and bring to 100 mL with deionized water

##### 4.2. 1.5 M Tris-HCl, pH 8.8

27.23 g Tris base (18.15 g/100 mL)

80 mL deionized water

Adjust to pH 8.8 with 6 N HCl. Bring total volume to 150 mL with deionized water and store at 4°C

##### 4.3. 0.5 M Tris-HCl, pH 6.8

6 g Tris base

60 mL deionized water

Adjust to pH 6.8 with 6 N HCl. Bring total volume to 100 mL with deionized water and store at 4°C.

##### 4.4. Sample Buffer (SDS Reducing Buffer)

3.55 mL deionized water

1.25 mL 0.5 M Tris-HCl

2.5 mL glycerol

2.0 mL 10% SDS

0.2 mL 0.5% (w/v) bromophenol blue

Total volume = 9.5 mL

Store at room temperature.

Add 50  $\mu$ L  $\beta$ -Mercaptoethanol to 950  $\mu$ L sample buffer prior to use.

##### 4.5. 10x Electrophoresis (Running) Buffer, pH 8.3 (makes 1L)

Dilute 50 mL of 10x stock Tris/Glycine/SDS with 450 mL deionized water for each electrophoresis run. Mix thoroughly before use.

##### 4.6. 10% APS (fresh daily)

100 mg ammonium persulfate

Dissolved in 1 mL of deionized water.

##### 4.7. Coomassie Blue dye

0.1% Coomassie Blue dye

50% methanol

10% glacial acetic acid

##### 4.8. Transfer Buffer

100 mL 10 x Tris Glycine transfer buffer

860 mL distilled H<sub>2</sub>O

40 mL methanol



## Appendix 5. Making Yeast Media

Prepare YPDA, SD-W (100 mL), SD-L (100 mL), SD-W-L (1 liter), SD-W-L-H-A+X- $\alpha$ -gal (500mL). Keep all media at 4°C when they are left on clean bench for 2 days at room temperature.

### YPDA liquid medium

Dissolve 1.5 g YPDA in 30 ml H<sub>2</sub>O.  
Autoclave for 30 min at 121 °C

### SD-W plates

Dissolve 1/5 SD-Trp-Leu broth pouch into 100 mL ddH<sub>2</sub>O in a 125 mL flask  
Add 1 mL 100 X Adenine  
Add 1 mL 100 X Leucine  
Add 1.8 g agar  
Autoclave for 30 min at 121 °C  
Cool down to 50 °C by storing the medium in 50 °C incubator for ~3 hours  
Stir to mix  
pour the medium into 5 plates

### SD-L plates

Dissolve 1/5 SD-Trp-Leu broth pouch into 100 mL ddH<sub>2</sub>O in a 125 mL flask  
Add 1 mL 100 X Adenine  
Add 1 mL 100 X Tryptophan  
Add 1.8 g agar  
Autoclave for 30 min at 121 °C  
Cool down to 50 °C by storing the medium in 50 °C incubator for ~3 hours  
Stir to mix  
pour the medium into 5 plates

### SD-W-L plates

Dissolve one SD-Trp-Leu broth pouch into 500 mL ddH<sub>2</sub>O in a one-liter flask  
Add 5 mL 100 X Adenine  
Add 9 g agar  
Autoclave for 30 min at 121 °C  
Cool down to 50 °C by storing the medium in 50 °C incubator for ~3 hours  
Stir to mix  
pour the medium into 20 plates  
Repeat this two more times to make in total 60 SD-W-L plates

### SD-A-H-W-L (500mL) plates

Dissolve one SD/-Ade-His-Trp-Leu broth pouch into 500 mL ddH<sub>2</sub>O in a one-liter flask  
Add 9 g agar  
Autoclave for 30 min at 121 °C  
Cool down to 50 °C by storing the medium in 50 °C incubator for ~3 hours  
Stir to mix  
pour the medium into 20 plates

## **Appendix 6. Yeast Transformation**

1. Inoculate 15 mL of YPDA medium in one 125 mL flask with several 2-3 mm colonies of a desired yeast strain. Grow the cells overnight at 30°C with 250rpm shake.
2. Next day, place cells in 50 mL tubes and centrifuge at 1,000 x g for 5 minutes at room temperature.
3. Discard the supernatant and resuspend cell pellets by vortexing in 25-50 mL of sterile H<sub>2</sub>O
4. Centrifuge cells at 1,000 x g for 5 minutes at room temperature
5. Decant the supernatant
6. Resuspend the cell pellet in 1.5 mL of freshly prepared sterile 1 X Te/Liac
7. Prepare 1 X PEG/LiAc solution
8. In a 1.5 mL tube, add and mix the following (Look at figure below for combinations):
  - 0.1 µg DNA-BD/bait
  - 0.1 µg AD
  - 0.1 mg shortened Salmon Sperm DNA
9. Add 0.1 mL of yeast competent cells as follows and mix well by vortexing.
10. Add 0.6 mL of sterile 1 X PEG/LiAc solution and vortex at high speed to mix
11. Incubate at 30°C for 30 minutes with shaking (200rpm)
12. Add 70 µL of DMSO and mix well by gentle inversion or swirling. Do not vortex!
13. Heat shock for 15 minutes in a 42°C water bath.
14. Chill cells on ice for 1-2 minutes
15. Centrifuge cells for 5 seconds at room temperature at 14,000 rpm
16. Remove the supernatant
17. Suspend cells in 0.5 mL sterile H<sub>2</sub>O
18. Spread 0.1 mL transformants on an appropriate selection yeast medium.
19. Grow cells for 3 to 5 days. Positive colonies appear after ~2 to 3 days when cultured at 30°C.

## **Appendix 7. To Do List for Teaching Assistants**

**(Intentionally Left Blank)**

**(Intentionally Left Blank)**

### **Lab 1: Open Source Programming 1**

1. Setup 2 additional PC computers as a backup for the class
2. Setup student user accounts on the class server
3. Pre-run codes on the server
4. Assist the instructor to help students troubleshoot problems
5. Clean up after the lab
6. Office hour

### **Lab 2: Open Source Programming 2**

1. Setup 2 additional PC computers as a backup for the class
2. Pre-run codes on the server
3. Assist the instructor to help students troubleshoot problems
4. Clean up after the lab
5. Office hour
6. Grading

### **Lab 3: BLAST**

1. Setup 2 additional PC computers as a backup for the class
2. Download and organize 10 proteomes into the class server
3. Edit, pre-run and save Perl codes on the server
4. Assist the instructor to help students troubleshoot problems
5. Clean up after the lab
6. Office hour
7. Grading

### **Lab 4: Bioperl 1**

1. Setup 2 additional PC computers as a backup for the class
2. Assist the instructor to help students troubleshoot problems
3. Edit, pre-run and save Perl codes on the server
4. Assist the instructor to help students troubleshoot problems
5. Clean up after the lab
6. Office hour
7. Grading

### **Lab 5: Bioperl 2**

1. Setup 2 additional PC computers as a backup for the class
2. Download and organize 10 transcriptomes into the class server
3. Edit, pre-run and save Perl codes on the server
4. Assist the instructor to help students troubleshoot problems
5. Clean up after the lab
6. Office hour
7. Grading

### **Lab 6: Molecular Phylogenetics**

1. Setup 2 additional PC computers as a backup for the class
2. Edit, pre-run phylogenetic analysis and save the result on the server
3. Assist the instructor to help students troubleshoot problems
4. Clean up after the lab
5. Office hour
6. Grading

### **Lab 7: Molecular Cloning**

1. Pre-run PCR
2. Assist the instructor to help students troubleshoot problems
3. Have pipettes, tips, microcentrifuge tubes, PCR tubes at each bench
4. Prepare sterile distilled water, load tips, autoclave tips and tubes
5. Place Ice bucket with primers, 5 x buffer, dNTPs, and diluted DNA templates (2 ng/ $\mu$ L, 20  $\mu$ L per template for 2 students) at each bench
6. Have bucket in front of class with *DreamTaq*
7. Have thermocycler cycle set ahead of time
8. Collect PCR product for the class
9. Clean up after the lab
10. Office hour

### **Lab 8: Ligation**

1. Pre-run ligation and transformation
2. Assist the instructor to help students troubleshoot problems
3. Have pipettes, tips, and microcentrifuge tubes at each bench
4. Prepare sterile distilled water, load tips, autoclave tips and tubes
5. Place bucket with PCR products from previous lab in front of class
6. Thaw and make ligation master mix
7. Preheat distilled water and incubate the water at 65° C, Preheat water bath to 50° C
8. Make 4 agarose gels
9. Clean up the lab after the class
10. Office hour

## **Lab 9: Bacterial Transformation & Genomic DNA Extraction**

### **1 month before lab**

Grow 1 tray of Col-0 and 1 tray of ask1 +/- mutant plants

### **2 Days before lab**

Pre-run genomic DNA extraction

### **Day of lab**

1. Assist the instructor to help students troubleshoot problems
2. Have pipettes, tips, and microcentrifuge tubes at each bench
3. Prepare sterile distilled water, LB plates, buffers
4. Dilute control intact plasmid DNA to 0.0001 ng/ $\mu$ L
5. Preheat 42°C water bath, at exactly that temperature
6. Preheat 55°C water bath, at exactly that temperature
7. Prepare 65°C warm water
8. Place cell spreader and ethanol lamp at each bench
9. Remove and thaw frozen JM109 High Efficiency Competent Cells from storage and aliquot 100  $\mu$ L cells in each 1.5 mL tube. Keep tubes on ice.
10. Clean up the lab after the class
11. Office hour

## **Lab 10: Genotyping**

1. Pre-run colony PCR
2. Assist the instructor to help students troubleshoot problems
3. Have pipettes, tips, and microcentrifuge tubes at each bench
4. Prepare sterile distilled water
5. Take care bacterial culture in Lab 6
6. Clean up the lab after the class
7. Office hour

## **Lab 11: Gene Amplification**

1. Pre-run Lab 11
2. Assist the instructor to help students troubleshoot problems
3. Have pipettes, tips, and microcentrifuge tubes at each bench
4. Prepare ampicillin antibiotics and LB+ ampicillin medium
5. Take care bacterial culture from Lab 8
6. Clean up the lab after the class
7. Office hour
8. Harvest cell culture next day

### **Lab 12: Plasmid Isolation**

1. Pre-run Lab 12
2. Assist the instructor to help students troubleshoot problems
3. Have pipettes, tips, and microcentrifuge tubes at each bench
4. Contact the Genomic Facility for Nanodrop spectrometer use
5. Set up 65°C water bath
6. Prepare 100% and 75% ethanol
7. Clean up the lab after the class
8. Office hour

### **Lab 13: Restriction Fragment Length Polymorphism**

1. Pre-run Lab 13
2. Assist the instructor to help students troubleshoot problems
3. Have pipettes, tips, and microcentrifuge tubes at each bench
4. Set up 37°C water bath
5. Clean up the lab after the class
6. Office hour

### **Lab 14: Protein Gel Electrophoresis**

#### **1 month before lab**

Grow 1 tray of Col-0 and 1 tray of DAL-YFP Co suppression line

#### **1 Day before lab**

Make 8 mini protein 3 gels following the instruction described in Appendix 4.

#### **Day of Lab**

- 1) Assist the instructor to help students troubleshoot problems
- 2) Have pipettes, tips, and microcentrifuge tubes at each bench
- 3) Clean up the lab after the class
- 4) Office hour



**Lab 15: Yeast Two-hybrid (Y2H)****2 Weeks prior to lab**

Prepare YPDA, SD-W (100 ML), SD-L (100 mL), SD-W-L (1.5 liter), and SD-W-L-H-A (500mL) using the instruction in Appendix 5. Keep all media at 4°C when they are left on clean bench for 2 days at room temperature.

**1 Week prior to lab**

Streak bait and prey yeast cells on SD-W and SD-L plates, respectively, and grown them at 30°C for 4-5 days.

**1 Day prior to lab**

Take SD-W-L plates out of the cold room and warm them up at room temperature overnight

Perform yeast mating according to Table S3.

**Table S3. Mating Pairs for Y2H Assay**

		Bait		
		1	2	3
Prey	A	A1	A2	A3
	B	B1	B2	B3

**Day of Lab**

1. Assist the instructor to help students troubleshoot problems
2. Have pipettes, tips, and microcentrifuge tubes at each bench
3. Clean up the lab after the class
4. Office hour

### **Lab 16: Immuno-Blotting**

1. Pre-run Lab 16
2. Assist the instructor to help students troubleshoot problems
3. Have pipettes, tips, and microcentrifuge tubes at each bench
4. Prepare 1x TBST, 1 x TBST with 5% milk, and 1 x TBST with 3% milk
5. Prepare developing and fixing solutions
6. Clean up the lab after the class
7. Office hour

### **Lab 17: Semi quantitative Y2H interaction assay**

1. Pre-run Lab 17
2. Take SD-A-H-W-L plates out of the cold room and warm them up at room temperature on a clean bench overnight.
3. Assist the instructor to help students troubleshoot problems
4. Have pipettes, tips, and microcentrifuge tubes at each bench
5. Prepare developing and fixing solutions
6. Clean up the lab after the class
7. Office hour

### **Lab 18: R Programming (3 hours)**

1. Setup 2 additional PC computers as a backup for the class
2. Pre-run codes on the server
3. Assist the instructor to help students troubleshoot problems
4. Clean up after the lab
5. Office hour

### **Lab 19: Processing RNA-Seq Data**

1. Setup 2 additional PC computers as a backup for the class
2. Pre-run codes on the server
3. Assist the instructor to help students troubleshoot problems
4. Clean up after the lab
5. Office hour

### **Lab 20: Processing RNA-Seq Data**

1. Setup 2 additional PC computers as a backup for the class
2. Pre-run codes on the server
3. Assist the instructor to help students troubleshoot problems
4. Clean up after the lab
5. Office hour

### **Lab 21: RNA Isolation (3 hours)**

#### **6 weeks before Lab**

Grow 1 tray of Col-0 and 1 tray of *ask1* mutant seedlings.

#### **1 Day before Lab**

Autoclave tips, tubes, and water for RNA Lab

#### **Day of lab**

1. Assist the instructor to help students troubleshoot problems
2. Have pipettes, tips, and microcentrifuge tubes at each bench
3. Clean up the lab after the class
4. Office hour

### **Lab 22: RNA Quality Examination (2 hours)**

#### **2 Weeks before lab**

1. Contact genomic facility to schedule bionalyzer service

#### **Day of Lab**

1. Assist the instructor to help students troubleshoot problems
2. Have pipettes, tips, and microcentrifuge tubes at each bench
3. Clean up the lab after the class
4. Office hour

### **Lab 23: cDNA Synthesis and RT-PCR**

1. Assist the instructor to help students troubleshoot problems
2. Have pipettes, tips, and microcentrifuge tubes at each bench
3. Clean up the lab after the class
4. Office hour

### **Lab 24: Continuance to cDNA Synthesis and RT-PCR**

1. Assist the instructor to help students troubleshoot problems
2. Have pipettes, tips, and microcentrifuge tubes at each bench
3. Clean up the lab after the class
4. Office hour

**(Intentionally Left Blank)**

## **Appendix 8. Lab Supply**

**(Intentionally Left Blank)**

**(Intentionally Left Blank)**

### Lab 1 ~ 6: Bioinformatics

PBIO4/5280 Server: Dell PowerEdge R710 Server X5670 2.93GHz 24-Cores 64GB 9-TB (RAID 5) (Provided by PBIO and Hua Lab)

### Lab 7: Molecular Cloning

1. Primers (IDT)  
  
ER1-ASK1\_For: GAATTCATGTCTGCGAAGAAGATTGTG  
BH1-ASK1\_Rev: GGATCCTCATTCAAAAGCCCATTGG
2. *Thermo Scientific™ DreamTaq Green DNA Polymerase* (Fisher, Cat # FEREP0712)
3. dNTP mixture, 10 mM (this consists of dATP, dCTP, dGTP, and dTTP) (VWR, Cat # 95057-676)
4. Gloves (Large, Medium, Small) (VWR, Cat # 89038-272, 89038-270, 89038-268, 2 Packs each)
5. Pipette Tips (1 mL, Fisher, Cat# 02-707-507 ( 3 PKs) )
6. Pipette Tips (1-200 µL, Fisher, Cat # 02-707-500 (3 PKs) )
7. 1.5 mL microfuge tubes (VWR, Cat # 89404-824 (5 PKs))
8. Individual 0.2 mL PCR tubes (Applied Biosystems, Cat # N8010540)

### Lab 8: Ligation

1. pGEM™-T Easy Vector Systems (Fisher, Cat # PRA1360)
2. NucleoSpin® Gel and PCR Clean-up (Fisher, NC9233808)
3. Agarose (<http://lab-express.com/agarose.htm>, Cat # 2001-100g)
4. 50X TAE BUFFER 1L (Fisher, Cat# FERB49)
5. Thermo Scientific GENERULER 1 KB 5X50UG (Fisher, Cat # FERSM0311)
6. GreenGlo™ Safe DNA Dye (Thomas Scientific, Cat # C788T73)
7. Individual 0.2 mL PCR tubes (Applied Biosystems, Cat # N8010540, Lab 7)
8. Pipette Tips (1 mL, Spring 2016, Fisher, Cat# 02-707-5076(PK), Lab 7)
9. Pipette Tips (1-200 µL, Fisher, Cat # 02-707-500 (PK), Lab 7)

10. 1.5 mL microfuge tubes (VWR, Cat # 89404-824 (PK), Lab 7)
11. Gloves (Large, Medium, Small) (VWR, Cat # 89038-272, 89038-270, 89038-268, Lab 7)

### **Lab 9: Bacterial Transformation & Genomic DNA Extraction**

1. LB Broth Mix, Miller (Fisher, Cat # BP1426-500)
2. LB Agar Mix, Miller (Fisher, Cat # BP1425-500)
3. Phenol:Chloroform:Isoamyl Alcohol 25:24:1 Saturated with 10 mM Tris, pH 8.0, 1 mM EDTA (Sigma, Cat # P2069-100ML)
4. EDTA-Na<sub>2</sub> (Sigma, Cat # E5134-50G)
5.  $\beta$  - Mercaptoethanol (Sigma, Cat # M3148-25ML)
6. CTAB (Fisher Cat# ICN19400480, 100G)
7. Tris-Base (Fisher Cat # BP152-500, 500G)
8. JM109 competent cells (Fisher 10 x 100 ul, Cat # 50-136-7453 (RPI: ZT3003) )
9. NaCl (Sigma, Cat # S7653-1KG)
10. Petri-dishes (VWR, Cat # 89038-970 (CS))
11. 12N HCl (Fisher Cat # A144-500)
12. Pipette Tips (1 mL, Spring 2016, Fisher, Cat# 02-707-5076(PK), Lab 7)
13. Pipette Tips (1-200  $\mu$ L, Fisher, Cat # 02-707-500 (PK), Lab 7)
14. 1.5 mL microfuge tubes (VWR, Cat # 89404-824 (PK), Lab 7)
15. Gloves (Large, Medium, Small) (VWR, Cat # 89038-272, 89038-270, 89038-268, Lab 7)
16. Control Plasmid DNA (Hua Lab)



### Lab 10: Genotyping

1. Primers (IDT)  
T7: TAATACGACTCACTATAGGG
2. *Thermo Scientific™ DreamTaq Green DNA Polymerase* (Fisher, Cat # FEREP0712, Lab 7)
3. *Thermo Scientific™ 0.2 mL Individual Tubes* (Fisher, Cat # AB-0337R, Lab 7)
4. dNTP mixture, 10 mM (this consists of dATP, dCTP, dGTP, and dTTP) (Lab 7)
5. Pipette Tips (1 mL, Spring 2016, Fisher, Cat# 02-707-5076(PK), Lab 7)
6. Pipette Tips (1-200 µL, Fisher, Cat # 02-707-500 (PK), Lab 7)
7. 1.5 mL microfuge tubes (VWR, Cat # 89404-824 (PK), Lab 7)
8. Gloves (Large, Medium, Small) (VWR, Cat # 89038-272, 89038-270, 89038-268, Lab 7)

### Lab 11: Gene Amplification

1. Fisherbrand™ Sterile Plastic Culture Tubes (Fisher Cat # 14-956-3A)
2. Pipette Tips (1 mL, Spring 2016, Fisher, Cat# 02-707-5076(PK), Lab 7)
3. Pipette Tips (1-200 µL, Fisher, Cat # 02-707-500 (PK), Lab 7)
4. 1.5 mL microfuge tubes (VWR, Cat # 89404-824 (PK), Lab 7)
5. Gloves (Large, Medium, Small) (VWR, Cat # 89038-272, 89038-270, 89038-268, Lab 7)

### Lab 12: Plasmid Isolation

1. NucleoSpin Plasmid (50) ([www.mn-net.com](http://www.mn-net.com), Cat# NC0211709,)
2. Pipette Tips (1 mL, Spring 2016, Fisher, Cat# 02-707-5076(PK), Lab 7)
3. Pipette Tips (1-200 µL, Fisher, Cat # 02-707-500 (PK), Lab 7)
4. 1.5 mL microfuge tubes (VWR, Cat # 89404-824 (PK), Lab 7)
5. Gloves (Large, Medium, Small) (VWR, Cat # 89038-272, 89038-270, 89038-268, Lab 7)

### **Lab 13 Restriction Fragment Length Polymorphism**

1. BamHI (Fisher, Cat # NC9790794)
2. Pipette Tips (1 mL, Spring 2016, Fisher, Cat# 02-707-5076(PK), Lab 7)
3. Pipette Tips (1-200  $\mu$ L, Fisher, Cat # 02-707-500 (PK), Lab 7)
4. 1.5 mL microfuge tubes (VWR, Cat # 89404-824 (PK), Lab 7)
5. Gloves (Large, Medium, Small) (VWR, Cat # 89038-272, 89038-270, 89038-268, Lab 7)

### **Lab 14: Protein Gel Electrophoresis**

1. 10x stock Tris/Glycine/SDS (Bio-Rad, Cat # 161-0772)
2. 10 x Tris Glycine transfer buffer (Bio-Rad, Cat # 161-0771)
3. 30% Degassed Acrylamide/Bis (Bio-Rad, Cat # 161-0146,)
4. PVDF Membrane (Fisher, Cat # IPVH08100)
5. GE Healthcare Whatman 3MM chromatography paper (Fisher, Cat # 3030153)
6. Pipette Tips (1 mL, Spring 2016, Fisher, Cat # 02-707-5076(PK), Lab 7)
7. Pipette Tips (1-200  $\mu$ L, Fisher, Cat # 02-707-500 (PK), Lab 7)
8. 1.5 mL microfuge tubes (VWR, Cat # 89404-824 (PK), Lab 7)
9. Gloves (Large, Medium, Small) (VWR, Cat # 89038-272, 89038-270, 89038-268, Lab 7)
10. Other chemicals (Hua Lab)

### **Lab 15: Yeast Two-hybrid (Y2H)**

1. SD-LEU/-TRP Broth (Fisher, Cat # NC0928944)
2. SD-ADE/-HIS/LEU/-TRP (Fisher, Cat # NC0928955)
3. X- $\alpha$ -gal 25 mg (Fisher, Cat # NC9631658)
4. Pipette Tips (1 mL, Spring 2016, Fisher, Cat # 02-707-5076(PK), Lab 7)
5. Pipette Tips (1-200  $\mu$ L, Fisher, Cat # 02-707-500 (PK), Lab 7)
6. 1.5 mL microfuge tubes (VWR, Cat # 89404-824 (PK), Lab 7)

7. Gloves (Large, Medium, Small) (VWR, Cat # 89038-272, 89038-270, 89038-268, Lab 7)
8. Other chemicals (Hua Lab)

### **Lab 16: Immuno-Blotting**

1. 10 x TBST (Fisher, Cat # 50-674-86)
2. Milk powder (Walmart)
3. X-ray film (MidSci, Cat # BX57)
4. HRP-conjugated goat anti rabbit secondary antibodies (Fisher, Cat # 50-672-00)
5. Kodak fixer powder (Fisher Cat # 50-268-00)
6. Kodak DEKTOL developer powder (Fisher, Cat # 50-267-64)
7. Pipette Tips (1 mL, Spring 2016, Fisher, Cat # 02-707-5076(PK), Lab 7)
8. Pipette Tips (1-200  $\mu$ L, Fisher, Cat # 02-707-500 (PK), Lab 7)
9. 1.5 mL microfuge tubes (VWR, Cat # 89404-824 (PK), Lab 7)
10. Gloves (Large, Medium, Small) (VWR, Cat # 89038-272, 89038-270, 89038-268, Lab 7)
11. Rabbit anti DAL primary antibodies (Hua Lab)

### **Lab 17: Semi quantitative Y2H interaction assay**

Covered in Lab 15

### **Lab 18~20: R-Programming and RNA-Seq Data Analysis**

PBIO4/5280 Server: Dell PowerEdge R710 Server X5670 2.93GHz 24-Cores 64GB 9-TB (RAID 5) (Provided by PBIO and Hua Lab)

### **Lab 21: RNA Isolation**

1. NucleoSpin RNA Plant (50) (Fisher, Cat # NC0648387)
2. Pipette Tips (1 mL, Fisher, Cat # 02-707-5076(PK), Lab 7)
3. Pipette Tips (1-200  $\mu$ L, Fisher, Cat # 02-707-500 (PK), Lab 7)

4. 1.5 mL microfuge tubes (VWR, Cat # 89404-824 (PK), Lab 7)
5. Gloves (Large, Medium, Small) (VWR, Cat # 89038-272, 89038-270, 89038-268, Lab 7)

## **Lab 22: RNA Quality Examination**

1. Bioanalyzer service (Ohio University Genomic Facility)
2. Pipette Tips (1 mL, Fisher, Cat # 02-707-5076(PK), Lab 7)
3. Pipette Tips (1-200  $\mu$ L, Fisher, Cat # 02-707-500 (PK), Lab 7)
4. 1.5 mL microfuge tubes (VWR, Cat # 89404-824 (PK), Lab 7)
5. Gloves (Large, Medium, Small) (VWR, Cat # 89038-272, 89038-270, 89038-268, Lab 7)

## **Lab 23: cDNA Synthesis and RT-PCR**

1. SuperScript™ III Reverse Transcriptase (Fisher, Cat # 18080044)
2. Invitrogen™ RNaseOUT™ Recombinant Ribonuclease Inhibitor (Fisher, Cat # 10-777-019)
3. Invitrogen™ DNase I, Amplification Grade (Fisher, Cat # 18-068-015)
4. Primers: (IDT, \$15)  
Oligo(dT)<sub>25</sub>: TTTTTTTTTTTTTTTTTTTTTTTTTTTTV  
RT-ACT2\_For: GGCATCACACTTTCTACAATGAG  
RT-ACT2\_Rev: ACCCTCGTAGATTGGCACAG
6. Pipette Tips (1 mL, Fisher, Cat # 02-707-5076(PK), Lab 7)
7. Pipette Tips (1-200  $\mu$ L, Fisher, Cat # 02-707-500 (PK), Lab 7)
8. 1.5 mL microfuge tubes (VWR, Cat # 89404-824 (PK), Lab 7)
9. Gloves (Large, Medium, Small) (VWR, Cat # 89038-272, 89038-270, 89038-268, Lab 7)

**Lab 24: Continuance to cDNA Synthesis and RT-PCR**

1. Pipette Tips (1 mL, Fisher, Cat # 02-707-5076(PK), Lab 7)
2. Pipette Tips (1-200  $\mu$ L, Fisher, Cat # 02-707-500 (PK), Lab 7)
3. 1.5 mL microfuge tubes (VWR, Cat # 89404-824 (PK), Lab 7)
4. Gloves (Large, Medium, Small) (VWR, Cat # 89038-272, 89038-270, 89038-268, Lab 7)

**(Intentionally Left Blank)**

**Appendix 9. Unix/Linux Command Reference (Adapted from [www.fosswire.com](http://www.fosswire.com))**

(Creative Commons Licenses, <https://creativecommons.org/licenses/>)

**File Commands**

**ls** – directory listing  
**ls -al** – formatted listing with hidden files  
**cd dir** - change directory to dir  
**cd** – change to home  
**pwd** – show current directory  
**mkdir dir** – create a directory dir  
**rm file** – delete file  
**rm -r dir** – delete directory dir  
**rm -f file** – force remove file  
**rm -rf dir** – force remove directory dir \* (caution!)  
**cp file1 file2** – copy file1 to file2  
**cp -r dir1 dir2** – copy dir1 to dir2; create dir2 if it doesn't exist  
**mv file1 file2** – rename or move file1 to file2 if file2 is an existing directory, moves file1 into directory file2  
**ln -s file link** – create symbolic link link to file  
**touch file** – create or update file  
**cat > file** – places standard input into file  
**more file** – output the contents of file  
**head file** – output the first 10 lines of file  
**tail file** – output the last 10 lines of file  
**tail -f file** – output the contents of file as it grows, starting with the last 10 lines

**Process Management**

**ps** – display your currently active processes  
**top** – display all running processes  
**killall proc** – kill all processes named proc \*  
**kill pid** – kill process id pid  
**bg** – lists stopped or background jobs; resume a stopped job in the background  
**fg** – brings the most recent job to foreground  
**fg n** – brings job n to the foreground

**File Permissions**

**chmod octal file** – change the permissions of file to octal, which can be found separately for user, group, and world by adding:

- 4 – read (r)
- 2 – write (w)
- 1 – execute (x)

Examples (For more options, see man chmod)

**chmod 777** – read, write, execute for all

**chmod 755** – rwx for owner, rx for group and world

## SSH

**ssh user@host** – connect to *host* as *user*  
**ssh -p port user@host** – connect to *host* on port  
                                  *port* as *user*  
**ssh-copy-id user@host** – add your key to *host* for *user* to enable a keyed o  
                                  passwordless login

## Searching

**grep pattern files** – search for pattern in files  
**grep -r pattern dir** – search recursively for pattern in dir  
**command | grep pattern** – search for *pattern* in the output of command  
**locate file** – find all instances of file

## System Information

**date** – show the current date and time  
**cal** – show this month's calendar  
**uptime** – show current uptime  
**w** – display who is online  
**whoami** – who you are logged in as  
**finger user** – display information about user  
**uname -a** – show kernel information  
**cat /proc/cpuinfo** – cpu information  
**cat /proc/meminfo** – memory information  
**man command** – show the manual for command  
**df** – show disk usage  
**du** – show directory space usage  
**free** – show memory and swap usage  
**whereis app** – show possible locations of app  
**which app** – show which app will be run by default

## Compression

**tar cf file.tar files** – create a tar named file.tar containing files  
**tar xf file.tar** – extract the files from file.tar  
**tar czf file.tar.gz files** – create a tar with Gzip compression  
**tar xzf file.tar.gz** – extract a tar using Gzip  
**tar cjf file.tar.bz2** – create a tar with Bzip2 compression  
**tar xjf file.tar.bz2** – extract a tar using Bzip2  
**grype file** – compresses file and renames it to file.gz  
**gzip -d file.gz** – decompresses file.gz back to file



## Network

***ping host*** – ping host and output results  
***whois domain*** – get whois information for domain  
***dig domain*** – get DNS information for domain  
***dig -x host*** – reverse lookup host  
***wget file*** – download file  
***wget -c file*** – continue a stopped download

## Install from source

***./configure***  
***make***  
***make install***  
***dpkg -i pkg.deb*** – install a package (Debian)  
***rpm -Uvh pkg.rpm*** – install a package (RPM)

## Shotrcuts

***Ctrl+C*** – halts the current command  
***Ctrl+Z*** – stops the current command, resume with  
    ***fg*** in the foreground or ***bg*** in the background  
***Ctrl+D*** – log out of current session, similar to exit  
***Ctrl+W*** – erases one word in the current line  
***Ctrl+U*** – erases the whole line  
***Ctrl+R*** – type to bring up a recent command  
***!!*** - repeats the last command  
***exit*** – log out of current session

**(Intentionally Left Blank)**

## Appendix 10. Vim Cheat Sheet (Adapted from <https://github.com/rtorr/vim-cheat-sheet>)

(The MIT Licenses, <https://opensource.org/licenses/MIT>)

### Cursor movement

***h*** - move cursor left  
***j*** - move cursor down  
***k*** - move cursor up  
***l*** - move cursor right

***w*** - jump forwards to the start of a word  
***W*** - jump forwards to the start of a word (words can contain punctuation)  
***e*** - jump forwards to the end of a word  
***E*** - jump forwards to the end of a word (words can contain punctuation)  
***b*** - jump backwards to the start of a word  
***B*** - jump backwards to the start of a word (words can contain punctuation)  
***0*** - jump to the start of the line  
***^*** - jump to the first non-blank character of the line  
***\$*** - jump to the end of the line  
***G*** - go to the last line of the document  
***5G*** - go to line 5

Prefix a cursor movement command with a number to repeat it. For example, ***4j*** moves down 4 lines.

### Insert mode - inserting/appending text

***i*** - insert before the cursor  
***I*** - insert at the beginning of the line  
***a*** - insert (append) after the cursor  
***A*** - insert (append) at the end of the line  
***o*** - append (open) a new line below the current line  
***O*** - append (open) a new line above the current line  
***ea*** - insert (append) at the end of the word  
***Esc*** - exit insert mode

### Editing

***r*** - replace a single character  
***J*** - join line below to the current one  
***cc*** - change (replace) entire line  
***cw*** - change (replace) to the end of the word ***c\$*** - change (replace) to the end of the line  
***s*** - delete character and substitute text  
***S*** - delete line and substitute text (same as ***cc***) ***xp*** - transpose two letters (delete and paste)  
***u*** - undo  
***Ctrl + r*** - redo  
***.*** - repeat last command

### Marking text (visual mode)

**v** - start visual mode, mark lines, then do a command (like y-yank)  
**V** - start linewise visual mode  
**o** - move to other end of marked area  
**Ctrl + v** - start visual block mode  
**O** - move to other corner of block  
**aw** - mark a word  
**ab** - a block with ()  
**aB** - a block with {}  
**ib** - inner block with ()  
**iB** - inner block with {}  
**Esc** - exit visual mode

### Visual commands

**>** - shift text right  
**<** - shift text left  
**y** - yank (copy) marked text  
**d** - delete marked text  
**~** - switch case

### Cut and paste

**yy** - yank (copy) a line  
**2yy** - yank (copy) 2 lines  
**yw** - yank (copy) word  
**y\$** - yank (copy) to end of line  
**p** - put (paste) the clipboard after cursor  
**P** - put (paste) before cursor  
**dd** - delete (cut) a line  
**2dd** - delete (cut) 2 lines  
**dw** - delete (cut) word  
**D** - delete (cut) to the end of the line  
**d\$** - delete (cut) to the end of the line  
**x** - delete (cut) character

### Exiting

**:w** - write (save) the file, but don't exit  
**:wq** - write (save) and quit  
**:x** - write (save) and quit  
**:q** - quit (fails if there are unsaved changes)  
**:q!** - quit and throw away unsaved changes

### Search and replace

**/pattern** - search for pattern  
**?pattern** - search backward for pattern  
**n** - repeat search in same direction  
**N** - repeat search in opposite direction  
**:%s/old/new/g** - replace all old with new throughout file  
**:%s/old/new/gc** - replace all old with new throughout file with confirmations

### Working with multiple files

**:e filename** - edit a file in a new buffer  
**:bnext** or **:bn** - go to the next buffer  
**:bprev** or **:bp** - go to the previous buffer  
**:bd** - delete a buffer (close a file)  
**:sp filename** - open a file in a new buffer and split window  
**:vsp filename** - open a file in a new buffer and vertically split window  
**Ctrl + ws** - split window  
**Ctrl + ww** - switch windows  
**Ctrl + wq** - quit a window  
**Ctrl + wv** - split window vertically  
**Ctrl + wh** - move cursor to next buffer (right)  
**Ctrl + wl** - move cursor to previous buffer (left)

### Tabs

**:tabnew filename** or **:tabn filename** - open a file in a new tab  
**Ctrl + wt** - move the current split window into its own tab  
**gt** or **:tabnext** or **:tabn** - move to the next tab  
**gT** or **:tabprev** or **:tabp** - move to the previous tab  
**#gt** - move to tab number #  
**:tabmove #** - move current tab to the #th position (indexed from 0)  
**:tabclose** or **:tabc** - close the current tab and all its windows  
**:tabonly** or **:tabo** - close all tabs except for the current one

**(Intentionally Left Blank)**

## Appendix 11. Perl Cheat Sheet from

<https://rc.hms.harvard.edu/training/perl/Perl%20Cheat%20Sheet.pdf>

(The MIT Licenses, <https://opensource.org/licenses/MIT>)

### Functions

Get information on a function by typing, e.g., `perldoc -f chomp` at the command line.

<b>Scalar variables</b>	<code>while (defined (\$x=&lt;&gt;)) {code}</code>	False if variable has never been set (or when you try to read past the end of an input file)
	<code>length(\$x)</code>	Length of a string
	<code>chomp(\$line);</code>	Remove a newline at the end of a string
	<code>\$short=substr (\$long, 2, 5)</code>	Characters 3-7 of \$long (first char is 0!)
<b>Arrays</b>	<code>push @arr, \$x</code>	Add to end of array
	<code>\$x = pop @arr;</code>	Remove last element of array, put in \$x
	<code>shift @arr;</code> (See also <code>unshift</code> )	Remove first element of an array, put in \$x
	<code>\$size = scalar @arr;</code>	Number of things in the array
	See also: <b>split, join, splice, sort</b>	split string->array, join array->string, delete part of array, sort array in many ways
<b>Hashes</b>	<code>@key = keys %hash</code>	The lookup terms in the hash
	<code>if (exists \$hh{"Hsp"}) {...}</code>	See whether hash %hh has a value for key Hsp
<b>Input/Output and Files</b>	<code>open(HANDLE, "&gt;outfile")</code> or <code>die "Can't open \$outfile: \$!\n"</code>	Open outfile for writing, and associate it with filehandle HANDLE. Use "<infile" for reading
	<code>print \$x;</code> <code>print HANDLE \$x;</code>	Prints to standard output (screen), Print to filehandle HANDLE
	<code>warn "Something wrong\n";</code>	Prints to standard error (screen)
	<code>\$x=&lt;HANDLE&gt;</code>	Read a line from filehandle HANDLE, put in \$x
	<code>close(HANDLE);</code>	Stop reading/writing a previously opened file
<b>Exit</b>	<code>exit;</code>	Exits the program
	<code>die "Something broke!\n";</code>	Exits the program with error message

### Operators and Loops

<b>Assign value</b>	<code>\$x = 1</code>	Sets variable to a value. Don't confuse with <code>==</code> , which tests whether numerical values are equal
<b>Math</b>	<code>print 1 * (2 + 3/4)</code>	Regular math symbols
	<code>10%3==1; 12%3==0</code>	Modulus (remainder) operator
	<code>\$x += 4;</code>	Same as <code>\$x=\$x+4</code> ; Also <code>--</code> <code>*=</code> <code>/=</code>
	<code>\$x++;</code>	Same as <code>\$x=\$x+1</code> ;
<b>Conditions</b>	<code>if (.1 == 0.1) {print "same num"}</code>	Are <b>numbers</b> equal? Don't confuse with <code>=</code> or <code>eq</code>
	<code>if (1 != 2) {print "diff num"}</code>	Are <b>numbers</b> different?
	<code>&gt; &lt; &gt;= &lt;=</code>	Are <b>numbers</b> greater than, less than, etc.
	<code>if ("a" eq "a") {print "same text"}</code>	Does <b>text</b> have exactly the same letters?
	<code>if ("A" ne "a") {print "diff text"}</code>	Does <b>text</b> have different letters?
	<code>if ((\$x &gt; 1) &amp;&amp; (\$x &lt; 2)) {code}</code>	Logical AND (true if both sides are true)
	<code>if ((\$x &gt; 10)    (\$x &lt; -10)) {code}</code>	Logical OR (true if one or both sides are true)
	<code>=~ !~</code>	Test for a match: See Matching cheat sheet
<b>Loops</b>	<code>foreach my \$i (1 .. 100) {code}</code> ( <code>for</code> and <code>foreach</code> are equivalent)	Sets \$i to 1 and does <code>code</code> . Sets \$i to 2, ... up to (and including) 100
	<code>while (\$a &lt; \$b) {code}</code>	Does <code>code</code> while the condition is true (If condition is false, never enters the loop.)

**Matching and Regular Expressions**

<b>Test for Match</b>	<code>=~</code>	Test for match	<code>if (\$x =~ /abc/) { ...}</code>	Does \$x have the string "abc" anywhere in it?
	<code>!~</code>	Test for non-match	<code>if (\$x !~ /abc/) { ...}</code>	Does \$x NOT have the string "abc" anywhere in it?
	<code>\$_</code>	Default variable	<code>if (/abcd/) { s/bc/x/ }</code>	// and s/// work on \$_ by default, no =~ needed
<b>Substitute</b>	<code>s///</code>	Do a Substitution	<code>\$x =~ s/abc/def/;</code>	Replace (only) first occurrence of "abc" in \$x with def
<b>Match/ Sub Options</b>	<code>i</code>	Ignore case.	<code>/abc/i</code>	Matches abc, ABC, aBc, etc.
	<code>g</code>	Global substitution.	<code>s/a/c/g</code>	Replace ALL occurrences
<b>Special Match Items</b>	<code>.</code>	Any one character (except \n)	<code>/a.c/</code>	"arc", "a9c", but not "ac".
	<code>[ ]</code>	Any one of.	<code>/[abc]/</code>	Any one of "a", "b", or "c". [a-zA-Z] matches any letter
	<code>\d</code>	Digit (Same as [0-9])	<code>/\d:\d\d\d/</code>	"10:30" (but not "9:30")
	<code>\s</code>	Space, tab, or newline	<code>/^\s*\$/</code>	An empty line.
	<code>\</code>	Literally match special characters: + * ( ) / [ ] \   { } ^ \$ @	<code>/1+2/</code>	"1+2", not "1112". The backslash "quotes" or "escapes" the plus sign.
<b>Item Locations</b>	<code>^</code>	Beginning of a line	<code>/^a/</code>	"arginine" but not "valine".
	<code>\$</code>	End of a line	<code>/a\$/</code>	"beta" but not "beat".
<b>Item Repetitions</b>	<code>?</code>	An optional thing	<code>/ab?c/</code>	"ac" or "abc".
	<code>*</code>	Any number of copies OR nothing at all	<code>/a*/</code>	"", "a", "aaa".
	<code>+</code>	Any number of copies	<code>/a+b/</code>	"ab" or "aaab" but not "b".
	<code>{ }</code>	m to n copies	<code>/ab{2,4}c/</code>	"abbc", "abbbc", "abbbbc", but not "abc" or "abbbbbbc"
<b>Misc</b>	<code> </code>	One or the other	<code>/abc def/</code>	"abc" or "def"
	<code>( )</code>	Capture parts of match in numbered variables	<code>/a(b(..)e)f/</code>	"abcdef". This will also set \$1 to "bcde" and \$2 to "cd".
		AND group things together for repetition, etc.	<code>/a(bc)+d/</code>	"abcd" or "abcbcbcbcd"



## **Appendix 12. R Cheat Sheet**



# Base R

## Cheat Sheet

### Getting Help

Accessing the help files

**?mean**  
Get help of a particular function.  
**help.search('weighted mean')**  
Search the help files for a word or phrase.  
**help(package = 'dplyr')**  
Find help for a package.

More about an object

**str(iris)**  
Get a summary of an object's structure.  
**class(iris)**  
Find the class an object belongs to.

### Using Libraries

**install.packages('dplyr')**  
Download and install a package from CRAN.  
**library(dplyr)**  
Load the package into the session, making all its functions available to use.  
**dplyr::select**  
Use a particular function from a package.  
**data(iris)**  
Load a built-in dataset into the environment.

### Working Directory

**getwd()**  
Find the current working directory (where inputs are found and outputs are sent).  
**setwd('C://file/path')**  
Change the current working directory.  
**Use projects in RStudio to set the working directory to the folder you are working in.**

### Vectors

#### Creating Vectors

c(2, 4, 6)	2 4 6	Join elements into a vector
2:6	2 3 4 5 6	An integer sequence
seq(2, 3, by=0.5)	2.0 2.5 3.0	A complex sequence
rep(1:2, times=3)	1 2 1 2 1 2	Repeat a vector
rep(1:2, each=3)	1 1 1 2 2 2	Repeat elements of a vector

#### Vector Functions

<b>sort(x)</b> Return x sorted.	<b>rev(x)</b> Return x reversed.
<b>table(x)</b> See counts of values.	<b>unique(x)</b> See unique values.

#### Selecting Vector Elements

##### By Position

<b>x[4]</b>	The fourth element.
<b>x[-4]</b>	All but the fourth.
<b>x[2:4]</b>	Elements two to four.
<b>x[-(2:4)]</b>	All elements except two to four.
<b>x[c(1, 5)]</b>	Elements one and five.

##### By Value

<b>x[x == 10]</b>	Elements which are equal to 10.
<b>x[x &lt; 0]</b>	All elements less than zero.
<b>x[x %in% c(1, 2, 5)]</b>	Elements in the set 1, 2, 5.

##### Named Vectors

<b>x['apple']</b>	Element with name 'apple'.
-------------------	----------------------------

### Programming

#### For Loop

```
for (variable in sequence){  
  Do something  
}
```

##### Example

```
for (i in 1:4){  
  j <- i + 10  
  print(j)  
}
```

#### While Loop

```
while (condition){  
  Do something  
}
```

##### Example

```
while (i < 5){  
  print(i)  
  i <- i + 1  
}
```

#### If Statements

```
if (condition){  
  Do something  
} else {  
  Do something different  
}
```

##### Example

```
if (i > 3){  
  print('Yes')  
} else {  
  print('No')  
}
```

#### Functions

```
function_name <- function(var){  
  Do something  
  return(new_variable)  
}
```

##### Example

```
square <- function(x){  
  squared <- x*x  
  return(squared)  
}
```

### Reading and Writing Data

Input	Ouput	Description
df <- read.table('file.txt')	write.table(df, 'file.txt')	Read and write a delimited text file.
df <- read.csv('file.csv')	write.csv(df, 'file.csv')	Read and write a comma separated value file. This is a special case of read.table/write.table.
load('file.RData')	save(df, file = 'file.Rdata')	Read and write an R data file, a file type special for R.

Conditions	a == b	Are equal	a > b	Greater than	a >= b	Greater than or equal to	is.na(a)	Is missing
	a != b	Not equal	a < b	Less than	a <= b	Less than or equal to	is.null(a)	Is null

## Types

Converting between common data types in R. Can always go from a higher value in the table to a lower value.

<code>as.logical</code>	TRUE, FALSE, TRUE	Boolean values (TRUE or FALSE).
<code>as.numeric</code>	1, 0, 1	Integers or floating point numbers.
<code>as.character</code>	'1', '0', '1'	Character strings. Generally preferred to factors.
<code>as.factor</code>	'1', '0', '1', levels: '1', '0'	Character strings with preset levels. Needed for some statistical models.

## Maths Functions

<code>log(x)</code>	Natural log.	<code>sum(x)</code>	Sum.
<code>exp(x)</code>	Exponential.	<code>mean(x)</code>	Mean.
<code>max(x)</code>	Largest element.	<code>median(x)</code>	Median.
<code>min(x)</code>	Smallest element.	<code>quantile(x)</code>	Percentage quantiles.
<code>round(x, n)</code>	Round to n decimal places.	<code>rank(x)</code>	Rank of elements.
<code>signif(x, n)</code>	Round to n significant figures.	<code>var(x)</code>	The variance.
<code>cor(x, y)</code>	Correlation.	<code>sd(x)</code>	The standard deviation.

## Variable Assignment

```
> a <- 'apple'
> a
[1] 'apple'
```




## The Environment

<code>ls()</code>	List all variables in the environment.
<code>rm(x)</code>	Remove x from the environment.
<code>rm(list = ls())</code>	Remove all variables from the environment.

You can use the environment panel in RStudio to browse variables in your environment.

## Matrixes

```
m <- matrix(x, nrow = 3, ncol = 3)
Create a matrix from x.
```

 <code>m[2, ]</code> - Select a row	<code>t(m)</code> Transpose
 <code>m[, 1]</code> - Select a column	<code>m %*% n</code> Matrix Multiplication
 <code>m[2, 3]</code> - Select an element	<code>solve(m, n)</code> Find x in: $m \cdot x = n$

## Lists

```
l <- list(x = 1:5, y = c('a', 'b'))
A list is collection of elements which can be of different types.
```

<code>l[[2]]</code> Second element of l.	<code>l[1]</code> New list with only the first element.	<code>l\$x</code> Element named x.	<code>l['y']</code> New list with only element named y.
---	--	---------------------------------------	--


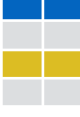
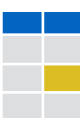
Also see the **dplyr** library.

## Data Frames



```
df <- data.frame(x = 1:3, y = c('a', 'b', 'c'))
A special case of a list where all elements are the same length.
```

x	y
1	a
2	b
3	c

### Matrix subsetting

<code>df[, 2]</code>	
<code>df[2, ]</code>	
<code>df[2, 2]</code>	

### List subsetting

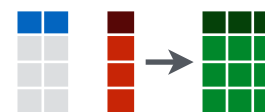
<code>df\$x</code>		<code>df[[2]]</code>	
<i>Understanding a data frame</i>			
<code>View(df)</code>	See the full data frame.		
<code>head(df)</code>	See the first 6 rows.		

`nrow(df)`  
Number of rows.

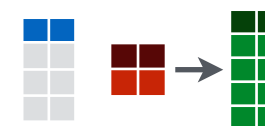
`ncol(df)`  
Number of columns.

`dim(df)`  
Number of columns and rows.

`cbind` - Bind columns.



`rbind` - Bind rows.



## Strings

Also see the **stringr** library.

<code>paste(x, y, sep = ' ')</code>	Join multiple vectors together.
<code>paste(x, collapse = ' ')</code>	Join elements of a vector together.
<code>grep(pattern, x)</code>	Find regular expression matches in x.
<code>gsub(pattern, replace, x)</code>	Replace matches in x with a string.
<code>toupper(x)</code>	Convert to uppercase.
<code>tolower(x)</code>	Convert to lowercase.
<code>nchar(x)</code>	Number of characters in a string.

## Factors

<code>factor(x)</code> Turn a vector into a factor. Can set the levels of the factor and the order.	<code>cut(x, breaks = 4)</code> Turn a numeric vector into a factor but 'cutting' into sections.
--	---

## Statistics

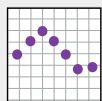
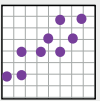
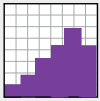
<code>lm(x ~ y, data=df)</code> Linear model.	<code>t.test(x, y)</code> Perform a t-test for difference between means.	<code>prop.test</code> Test for a difference between proportions.
<code>glm(x ~ y, data=df)</code> Generalised linear model.	<code>pairwise.t.test</code> Perform a t-test for paired data.	<code>aov</code> Analysis of variance.
<code>summary</code> Get more detailed information out a model.		

## Distributions

	Random Variates	Density Function	Cumulative Distribution	Quantile
Normal	<code>rnorm</code>	<code>dnorm</code>	<code>pnorm</code>	<code>qnorm</code>
Poisson	<code>rpois</code>	<code>dpois</code>	<code>ppois</code>	<code>qpois</code>
Binomial	<code>rbinom</code>	<code>dbinom</code>	<code>pbinom</code>	<code>qbinom</code>
Uniform	<code>runif</code>	<code>dunif</code>	<code>punif</code>	<code>qunif</code>

## Plotting

Also see the **ggplot2** library.

 <code>plot(x)</code> Values of x in order.	 <code>plot(x, y)</code> Values of x against y.	 <code>hist(x)</code> Histogram of x.
---	---	---

## Dates

See the **lubridate** library.



