# Bioinformatics course on Genome Assembly & Transcriptomics

Aug 18-22nd 2019, KAUST

# Team BcGAT 2019

DAY 1

Unix Basic programming,

Preparation for Day 2,3,4,5

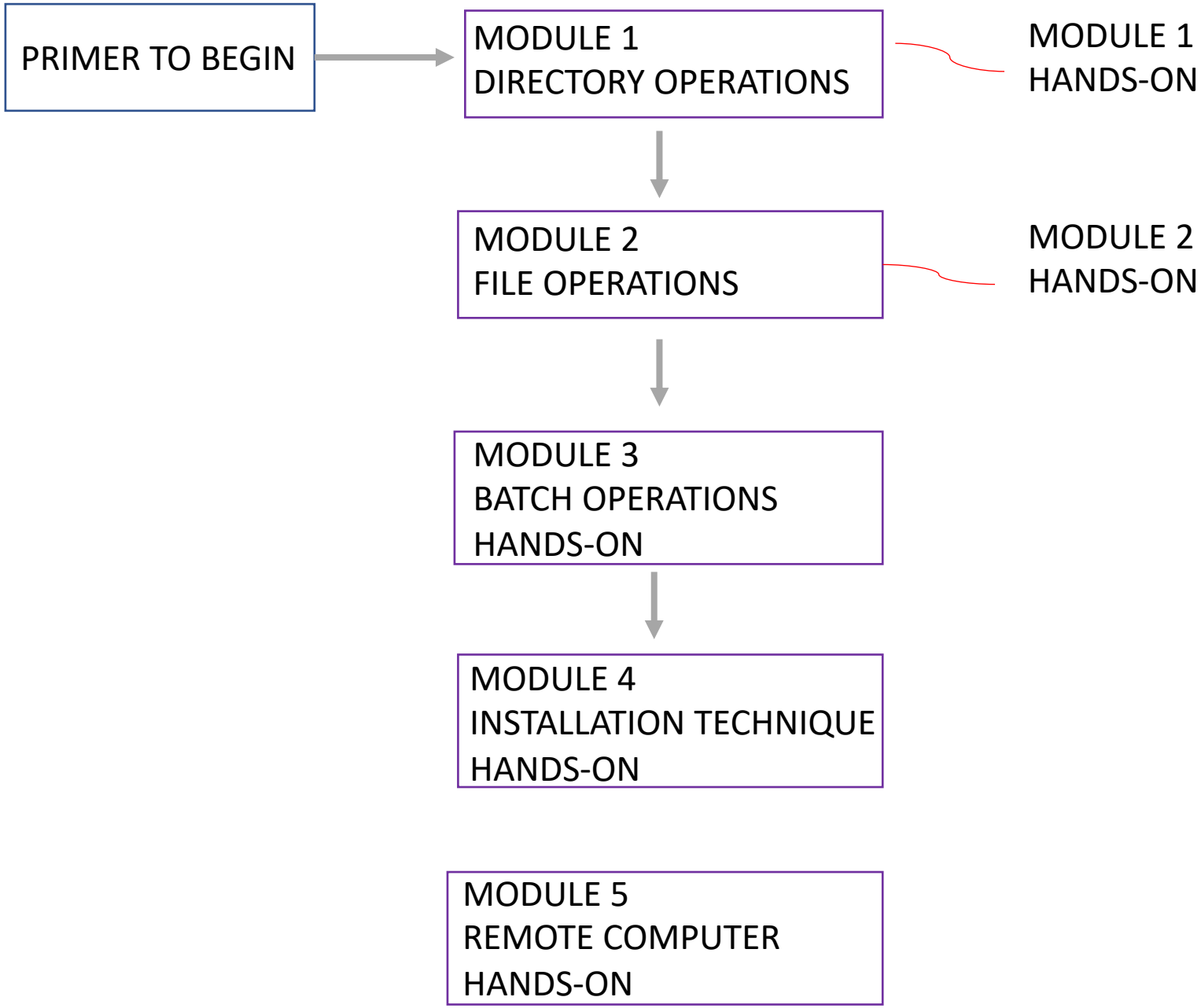
DAY 2 & 3

Genome Assembly

DAY 4 & 5

Transcriptomics

# PROGRAMMING BASICS FOR BIOLOGISTS


PRIMER TO BEGIN
MODULE 1
DIRECTORY OPERATIONS
MODULE 1
HANDS-ON
MODULE 2
FILE OPERATIONS
MODULE 2
HANDS-ON
MODULE 3
BATCH OPERATIONS
HANDS-ON
MODULE 4
INSTALLATION TECHNIQUE
HANDS-ON
MODULE 5
REMOTE COMPUTER
HANDS-ON

Markowetz recently wrote, "Computational biologists are just biologists using a different tool" [17]. If you are a traditionally trained biologist, we intend these 10 simple rules as instruction (and pep talk) to learn a new, powerful, and exciting tool. The learning curve can be steep; however, the effort will pay dividends. Computational experience will make you more marketable as a scientist (see "Top N Reasons To Do A Ph.D. or Post-Doc in Bioinformatics/Computational Biology" [18]). Computational research has fewer overhead costs and reduces the barrier to entry in transitioning fields [19], opening career doors to interested researchers. Perhaps most importantly, programming skills will make you better able to implement and interpret your own analyses and understand and respect analytical biases, making you a better experimentalist as well. Therefore, the time you spend at your computer is valuable. Acquiring programming expertise will make you a better biologist.

Carey MA, Papin JA (2018) Ten simple rules for biologists learning to program. PLoS Comput Biol 14(1): e1005871. https://doi.org/ 10.1371/journal.pcbi.1005871

# 10 COMMANDMENTS BEFORE YOU BEGIN

- RULE 1: Begin with the end in mind – there is no universal “best” language

- RULE 2: Baby step are steps – Breaking a large problem into modular tasks
- RULE 3: Immersion is the best learning tool - if a job can be done in one language or environment- do it all there

- RULE 4: Phone a friend – stackoverflow, biostars etc.,


**Fig 6. "How to exit the vim editor?" (or: We all get stuck at some point).** Now viewed >1.33 million times; see: http://stackoverflow.com/questions/11828270/how-to-exit-the-vim-editor.

https://doi.org/10.1371/journal.pcbi.1005871.g006

RULE 5: Learn how to ask questions - debugging

RULE 6: Don’t re-invent the wheel – Provide credit if appropriate

RULE 7: Develop good habits early-on - Labnotebook

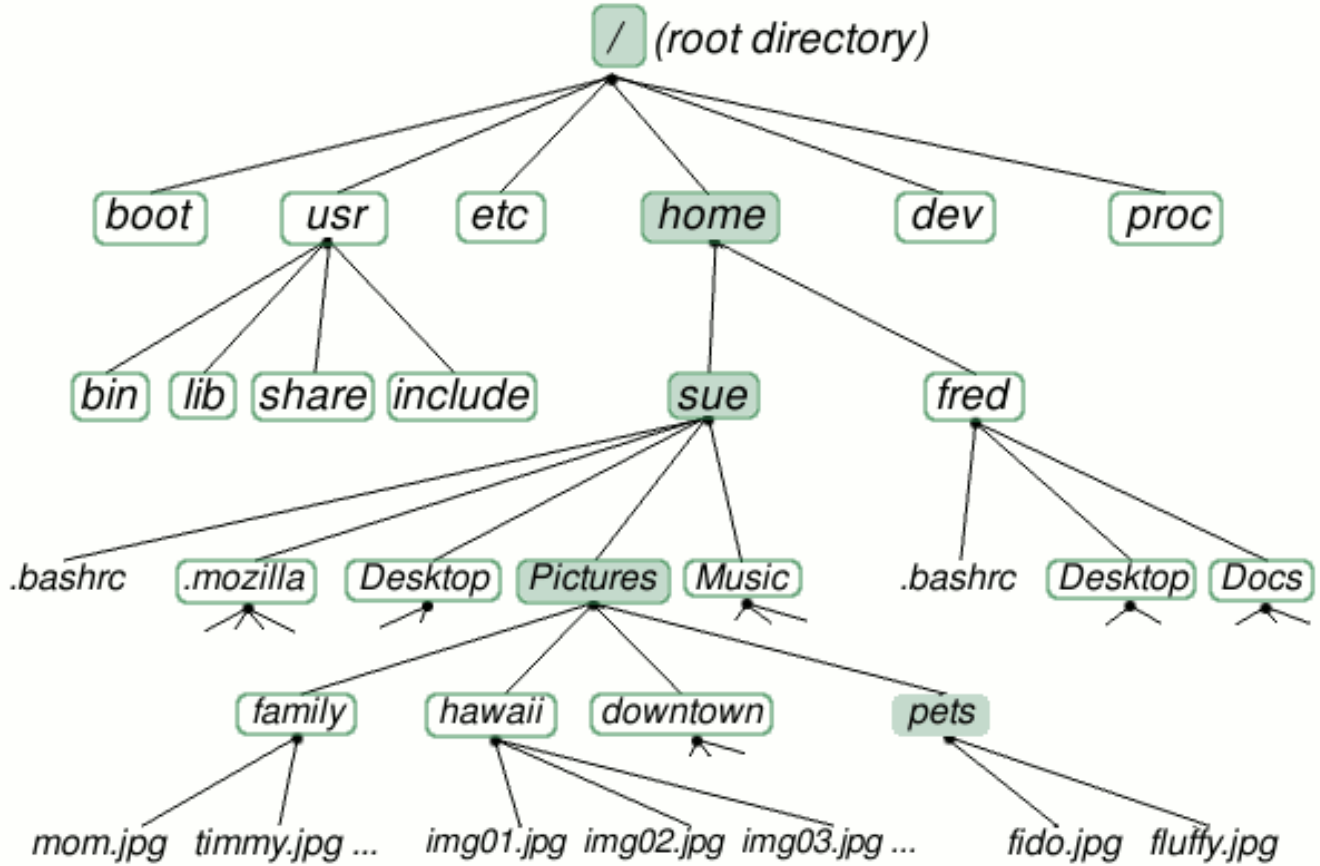
RULE 8: Practice makes perfect – generate toy datasets as your data

RULE 9: Teach yourself

RULE 10: Just do it – just start coding. You can’t edit a blank page

# Widely practiced Unix Philosophy

- TMWTD → There are many ways to do it
- K.I.S.S -> Keep it small and simple *aka* Keep it simple, stupid ! – A command should be designed to do only one task and do it well
- Understand the unix hierarchy – Everything on unix OS is interpreted as (file, dir, device). Directories are structured in hierarchical nature (like a family tree)

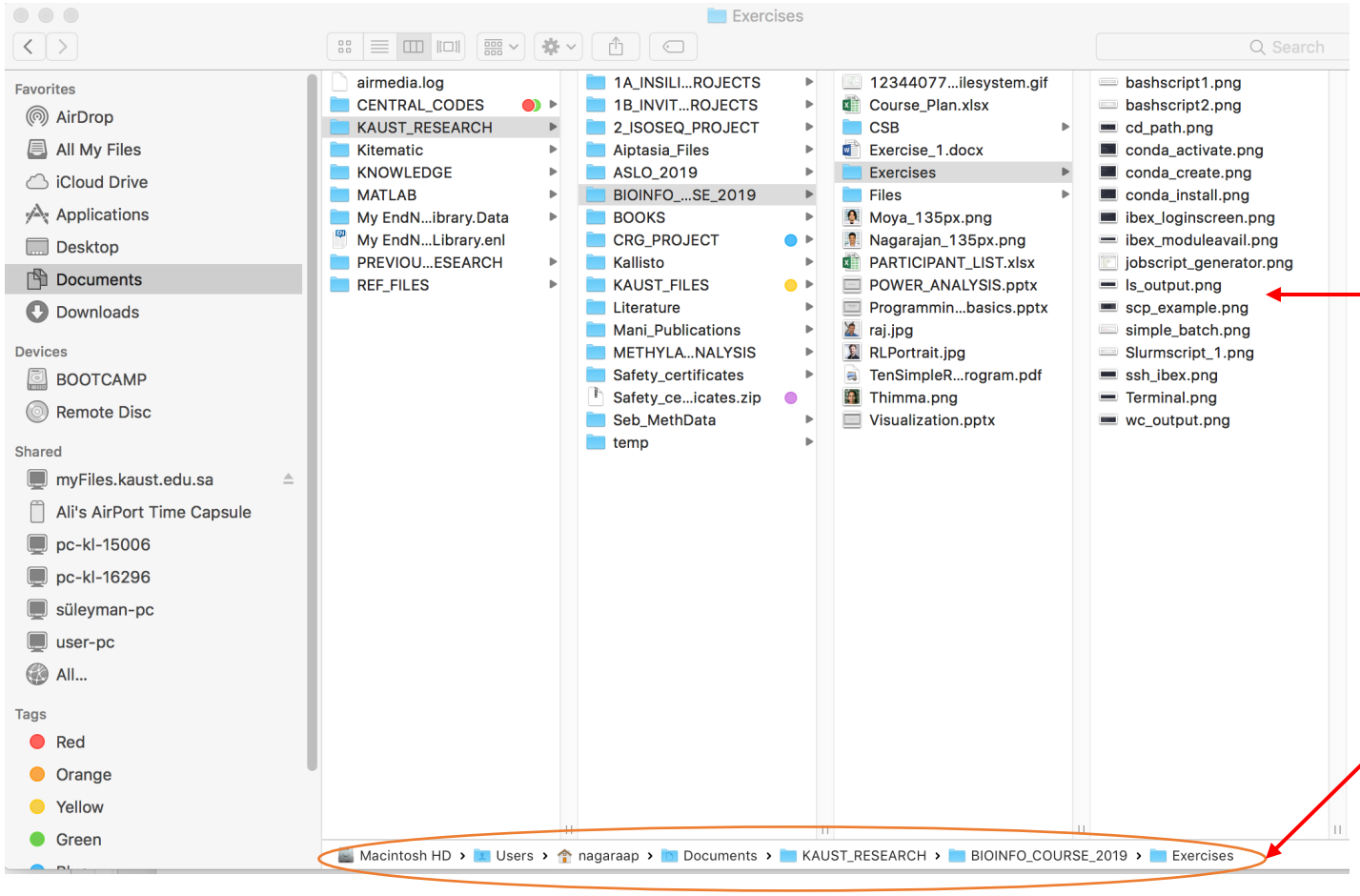


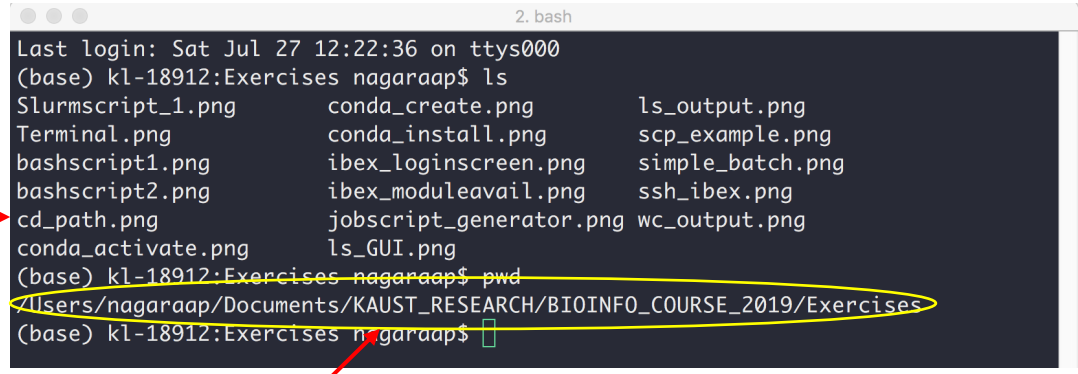

Path for timmy.jpg:  /home/sue/Pitcures/family/timmy.jpg

# Module 1 – DIRECTORY OPERATIONS

- Ditch the GUI – Graphical User Interface
- Use command-line to do anything and everything !
- Learn how to view the content of directory
- Copy
- Move
- Rename
- Create directory
- Delete a directory

# Same operations in different ways



# Essential built-in commands

| Command | Function | Usage |
|---|---|---|
| mkdir | Make a new directory | mkdir “Unix_Files” |
| rmdir | Removes an empty directory | rmdir “Unix_Files” |
| cp | copy a file or directory | cp –r folder1 /topath/ |
| mv | move or rename a file or directory | mv folder1 new_filename |
| pwd | present working directory | Pwd |
| cd | change directory | cd . \| cd .. \| cd ../folder/path/../ |
| . | present directory | ls . |
| .. | previous directory | cd .. |

Proceed to 1. Directory operations exercise : 30-45 minutes

Complete the Quiz 

# Module 2 – FILE OPERATIONS

- Use command-line to do anything and everything !
- Learn how to create files
- Copy file
- Move a file
- View a file
- Scroll through file
- Delete a file
- Add lines in file

Go to school ! Memorize the next 3 slides !
The alphabets of command-line tools

# Essential built-in commands – File operations

| Command | Function | Usage |
|---|---|---|
| ls | list | ls –alth, ls *.txt |
| cat | concatenate and print files | cat file.txt, cat 1.txt 2.txt |
| more | scroll files page by page | more file.txt |
| less | opposite of more | less file.txt |
| head | view first few lines of the file | head –n 5 file.txt |
| tail | view last few lines of the file | tail –n -5 file.txt |
| wc | word count | wc –l file.txt – counts # of lines |
| echo | display anything | echo “ha ha…what a joke” |
| grep | globally search a regular expression and print | grep “Antartica” countries.txt |
| touch | create new files | touch testfile.txt |
| cut | cut fields based on delimiters | cut –d “,” –f2 file |

head

# Ten simple rules for biologists learning to program

**Maureen A. Carey**[1], **Jason A. Papin**[2]*

**1** Department of Microbiology, Immunology, and Cancer Biology, University of Virginia School of Medicine, Charlottesville, Virginia, United States of America, **2** Department of Biomedical Engineering, University of Virginia, Charlottesville, Virginia, United States of America

* papin@virginia.edu

## Introduction

As big data and multi-omics analyses are becoming mainstream, computational proficiency and literacy are essential skills in a biologist's tool kit. All "omics" studies require computational biology: the implementation of analyses requires programming skills, while experimental design and interpretation require a solid understanding of the analytical approach. While academic cores, commercial services, and collaborations can aid in the implementation of analyses, the computational literacy required to design and interpret omics studies cannot be replaced or supplemented. However, many biologists are only trained in experimental techniques. We write these 10 simple rules for traditionally trained biologists, particularly graduate students interested in acquiring a computational skill set.

## Rule 1: Begin with the end in mind

When picking your first language, focus on your goal. Do you want to become a programmer? Do you want to design bioinformatic tools? Do you want to implement tools? Do you want to just get these data analyzed already? Pick an approach and language that fits your long- and short-term goals.

Languages vary in intent and usage. Each language and package was created to solve a particular problem, so there is no universal "best" language (Fig 1). Pick the right tool for the job by choosing a language that is well suited for the biological questions you want to ask. If many people in your field use a language, it likely works well for the types of problems you will encounter. If people in your field use a variety of languages, you have options. To evaluate ease of use, consider how much community support a language has and how many resources that community has created, such as prevalence of user development, package support (documentation and tutorials), and the language's "presence" on help pages. Practically, languages vary in cost for academic and commercial use. Free languages are more amenable to open source work (i.e., sharing your analyses or packages). See Table 1 for a brief discussion of several programming languages, their key features, and where to learn more.

## Rule 2: Baby steps are steps

Once you've begun, focus on one task at a time and apply your critical thinking and problem solving skills. This requires breaking a problem down into steps. Analyzing omics data may sound challenging, but the individual steps do not: e.g., read your data, decide how to interpret missing values, scale as needed, identify comparison conditions, divide to calculate fold change, calculate significance, correct for multiple testing. Break a large problem into modular

tail

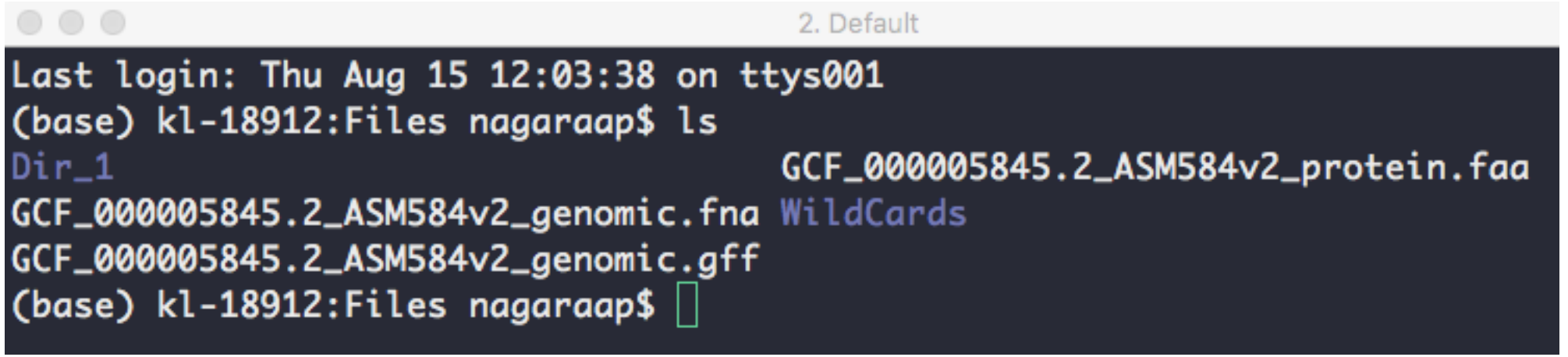
less

more

cat
wc

# Let us learn to use them with example !

Typing ls gives the following output




Typing ls –a : Notice the . and ..

ls –al : More information

```
(base) kl-18912:Files nagaraap$ ls -al
total 18192
drwxr-xr-x@  8 nagaraap  KAUST\Domain Users      272 Jul 23 14:28 .
drwxr-xr-x  29 nagaraap  KAUST\Domain Users      986 Aug 15 11:45 ..
-rw-r--r--@  1 nagaraap  KAUST\Domain Users     6148 Jul 18 15:19 .DS_Store
drwxr-xr-x  23 nagaraap  KAUST\Domain Users      782 Jul 28 11:43 Dir_1
-rw-r--r--   1 nagaraap  KAUST\Domain Users  4699745 Jul 17 10:36 GCF_000005845.2_ASM584v2_genomic.fna
-rw-r--r--@  1 nagaraap  KAUST\Domain Users  2855685 Jul 17 10:36 GCF_000005845.2_ASM584v2_genomic.gff
-rw-r--r--   1 nagaraap  KAUST\Domain Users  1742988 Jul 17 10:36 GCF_000005845.2_ASM584v2_protein.faa
drwxr-xr-x  21 nagaraap  KAUST\Domain Users      714 Jul 23 15:31 WildCards
(base) kl-18912:Files nagaraap$
```

ls –alt : sorted by time modified

```
(base) kl-18912:Files nagaraap$ ls -alt
total 18192
drwxr-xr-x  29 nagaraap  KAUST\Domain Users      986 Aug 15 11:45 ..
drwxr-xr-x  23 nagaraap  KAUST\Domain Users      782 Jul 28 11:43 Dir_1
drwxr-xr-x  21 nagaraap  KAUST\Domain Users      714 Jul 23 15:31 WildCards
drwxr-xr-x@  8 nagaraap  KAUST\Domain Users      272 Jul 23 14:28 .
-rw-r--r--@  1 nagaraap  KAUST\Domain Users     6148 Jul 18 15:19 .DS_Store
-rw-r--r--   1 nagaraap  KAUST\Domain Users  1742988 Jul 17 10:36 GCF_000005845.2_ASM584v2_protein.faa
-rw-r--r--   1 nagaraap  KAUST\Domain Users  4699745 Jul 17 10:36 GCF_000005845.2_ASM584v2_genomic.fna
-rw-r--r--@  1 nagaraap  KAUST\Domain Users  2855685 Jul 17 10:36 GCF_000005845.2_ASM584v2_genomic.gff
```

ls –alth :  sorted by time modified

```
(base) kl-18912:Files nagaraap$ ls -alth
total 18192
drwxr-xr-x  29 nagaraap  KAUST\Domain Users   986B Aug 15 11:45 ..
drwxr-xr-x  23 nagaraap  KAUST\Domain Users   782B Jul 28 11:43 Dir_1
drwxr-xr-x  21 nagaraap  KAUST\Domain Users   714B Jul 23 15:31 WildCards
drwxr-xr-x@  8 nagaraap  KAUST\Domain Users   272B Jul 23 14:28 .
-rw-r--r--@  1 nagaraap  KAUST\Domain Users   6.0K Jul 18 15:19 .DS_Store
-rw-r--r--   1 nagaraap  KAUST\Domain Users   1.7M Jul 17 10:36 GCF_000005845.2_ASM584v2_protein.faa
-rw-r--r--   1 nagaraap  KAUST\Domain Users   4.5M Jul 17 10:36 GCF_000005845.2_ASM584v2_genomic.fna
-rw-r--r--@  1 nagaraap  KAUST\Domain Users   2.7M Jul 17 10:36 GCF_000005845.2_ASM584v2_genomic.gff
```

ls –R :  list all files in subdirectories

```
(base) kl-18912:Files nagaraap$ ls -R
Dir_1                                GCF_000005845.2_ASM584v2_genomic.gff WildCards
GCF_000005845.2_ASM584v2_genomic.fna GCF_000005845.2_ASM584v2_protein.faa

./Dir_1:
1_E.coli_genes.gff                E.coli_genome.fa                  lipoproteins.txt
2_E.coli_genes_identifiercut.out  E.coli_genomic.gff                one-liner.csv
3_E.coli_genes_onlyIDs.out        E.coli_protein.fa                 oneliner_genelist.csv
4_genelist.out                    Slurmscript_1.sh                  protein_headers.txt
5_genelist.sorted                 Test_E.coli_genelist.out          sorted_genelist.csv
6.1_E.coli_genelist.out           bashscript_1.sh                   uniq_genes.csv
6_genelist.sorted.uniq.out        header_trimmed_protein.fa

./WildCards:
1.txt         AA.txt         b.txt         cat_2.txt     rat_10_c.txt  rat_20_c.txt  rat_30_c.txt
2.txt         a.txt          c.txt         rat_10_a.txt  rat_20_a.txt  rat_30_a.txt
3.txt         add_a_text.sh  cat_1.txt     rat_10_b.txt  rat_20_b.txt  rat_30_b.txt
```

# List - ls

ls –alth → You quiet often see commands followed by notations like this. What do they mean ?

**man** ls

- `-l` long format, displaying Unix file types, permissions, number of hard links, owner, group, size, last-modified date and filename
- `-f` do not sort. Useful for directories containing large numbers of files.
- `-F` appends a character revealing the nature of a file, for example, `*` for an executable, or `/` for a directory. Regular files have no suffix.
- `-a` lists all files in the given directory, including those whose names start with "." (which are hidden files in Unix). By default, these files are excluded from the list.
- `-R` recursively lists subdirectories. The command `ls -R /` would therefore list all files.
- `-d` shows information about a symbolic link or directory, rather than about the link's target or listing the contents of a directory.
- `-t` sort the list of files by modification time.
- `-h` print sizes in human readable format. (e.g., 1K, 234M, 2G, etc.) This option is not part of the POSIX standard, although implemented in several systems, e.g., GNU coreutils in 1997,[8] FreeBSD 4.5 in 2002,[9] and Solaris 9 in 2002.[10]

Expand ls –alth now:

List – **a**ll files in given directory, in **l**ong format, sorted by modification **t**ime and print size in **h**uman readable format

Can you now expand: **cut –d “,” –f 1** ?

# Essential built-in commands – Utilities

| Command | Function | Usage |
|---|---|---|
| > | Redirect stdout | cat file1 file2 > file3 |
| < | Redirect stdin (take input from file) | a.py < data |
| >> | Redirect stdout and append | cat file1 >> file2 |
| ; | Separate commands on same line | cal;date;time |
| () | Group commands on same line | (date; uname; cal) > group.txt |
| $0 | Special variable – scritpt itself | |
| $1..$n | Special variable -- parameters | ./Script.sh param1 |
| \| | Pipe symbol | cat file \|cut \| grep “pattern”\|sort \|uniq > output file |

# One-liners are super-useful !

Combine multiple commands to create-one liners !!

Mini workflows

Assemble commands in order to achieve a task

Direct output of one command as input to other

Find how many unique genes are present in the table

```
1,gene1,basal,wildtype
2,gene2,DEG,knockout
3,gene2,basal,wildtype
4,gene1,DEG,knockout
```

1. Extract second column
2. Sort them
3. Remove duplicates

1. Extract second column: cut –d “,” –f1 file.csv > genelist.csv

```
gene1
gene2
gene2
gene1
```

2. sort: sort genelist.csv > sorted_genelist.csv

```
gene1
gene1
gene2
gene2
```

3. Remove duplicates: uniq sorted_genelist.csv > uniq_list.csv

```
gene1
gene2
```

Answer is 2

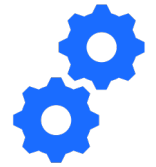
Now combine them all !!

cut –d “,” –f1 file.csv | sort | uniq | wc –l  Answer is 2

See the flow of information !

Proceed to 2. File operations exercise : 45 minutes → Complete the Quiz

# Module 3 – BATCH OPERATIONS



Processing 1 file is ok but, what do you do when you have to deal with 100 files at once ?

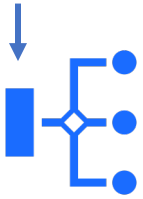
Answer: Perform batch operations

Bash file is nothing but soft-coded command lines stitched together to achieve an operation

First line of bash file

```
1  #!/bin/bash
2
3  # In this exercise, we will write a small loop to write a small text in all
4  # the empty files
5
6  for filelist in $(ls *.txt)
7  do
8      echo "Opening file: $filelist"
9      echo 'A line in now written !' > $filelist
10     echo "$filelist is filled with a line"
11     echo
12 done
```

Here, filelist is called a variable. i.e. its value keeps change after every iteration



# UNIX WILDCARDS [NOT] REGULAR EXPRESSION WILDCARDS

| Special character | Meaning |
|---|---|
| * | Zero or more characters except a leading dot |
| ? | Any single character except a leading dot |
| [] | Class of characters. [- for range, ! To exclude] |

| | | |
|---|---|---|
| [abc]?? | 3 character filename beginning with "a", "b", or "c". | cat.txt, b_01.txt |
| [1-9][A-Z] | 2 character filename starting with a number, and ending with an uppercase letter. | 1A.py, 8Z.c |
| [!A-Z]?? | 3 character filename that *does not* begin with an uppercase letter. | a10.txt, b10.txt, 1AF.txt |
| *t[0-9]e | any file ending with ”t", a single number, and ”e". | Cart1e.txt |

Proceed to 3. Batch operations exercise : 45 minutes;  Complete the Quiz

# Module 4 – EASY INSTALLATION

Installation of softwares in windows is a cake-walk, but most computationally intensive jobs are performed in shell based systems

Ok-ish in Mac

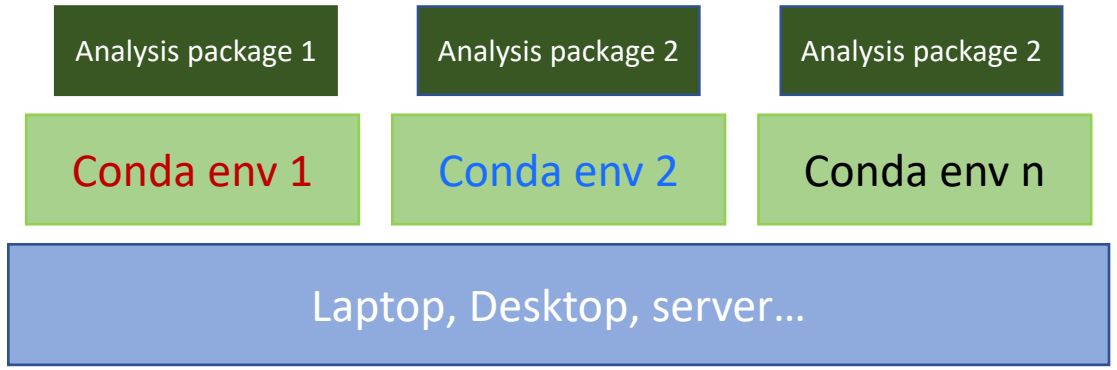
Linux

BUT THIS FRIENDLY ANACONDA MIGHT HAVE AN ANSWER !!

# Anaconda makes your life easy !!

One command installation of softwares and their dependencies

Avoids version conflicts, given the virtual environment setups




## CONDA 4.6 CHEAT SHEET

CONDA®

Take a conda test drive at **bit.ly/tryconda**

Windows, macOS, Linux:
Same commands for all platforms.

For full documentation of any command, add `--help` to the command.

EXAMPLE: `conda create --help`

| Getting Started | |
|---|---|
| Verify Conda is installed, check version number | `conda info` |
| Update Conda to the current version | `conda update -n base conda` |
| Update all packages to the latest version of Anaconda. Will install stable and compatible versions, not necessarily the very latest. | `conda update anaconda` |

| Working with Environments | |
|---|---|
| Create a new environment named ENVNAME with specific version of Python and packages installed. | `conda create --name ENVNAME python=3.6 "PKG1>7.6" PKG2` |
| Activate a named Conda environment | `conda activate ENVNAME` |
| Activate a Conda environment at a particular location on disk | `conda activate /path/to/environment-dir` |
| Deactivate current environment | `conda deactivate` |
| List all packages and versions in the active environment | `conda list` |
| List all packages and versions in a named environment | `conda list --name ENVNAME` |
| List all revisions made within the active environment | `conda list --revisions` |
| List all revisions made in a specified environment | `conda list --name ENVNAME --revisions` |
| Restore an environment to a previous revision | `conda install --name ENVNAME --revision REV_NUMBER` |
| Delete an entire environment | `conda remove --name ENVNAME --all` |

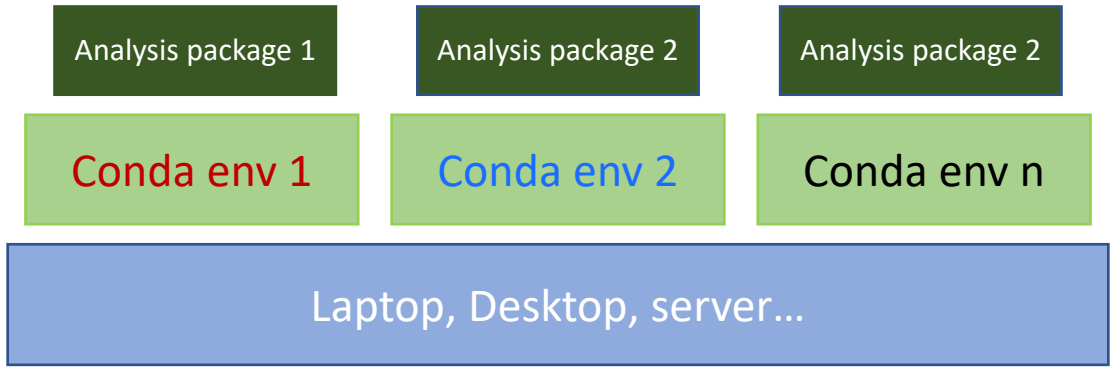
TIP: Anaconda Navigator is a desktop graphical user interface to manage packages and environments with Conda. With Navigator you do not need to use a terminal to run Conda commands, Jupyter Notebooks, JupyterLab, Spyder, and other tools. Navigator is installed with Anaconda, and may be added with Miniconda.

| Sharing Environments | |
|---|---|
| Make an exact copy of an environment | `conda create --clone ENVNAME --name NEWENV` |
| Export an environment to a YAML file that can be read on Windows, macOS, and Linux | `conda env export --name ENVNAME > envname.yml` |
| Create an environment from YAML file | `conda env create --file envname.yml` |
| Create an environment from the file named environment.yml in the current directory | `conda env create` |
| Export an environment with exact package versions for one OS | `conda list --explicit > pkgs.txt` |
| Create an environment based on exact package versions | `conda create --name NEWENV --file pkgs.txt` |

ANACONDA.

Continued on back ➜

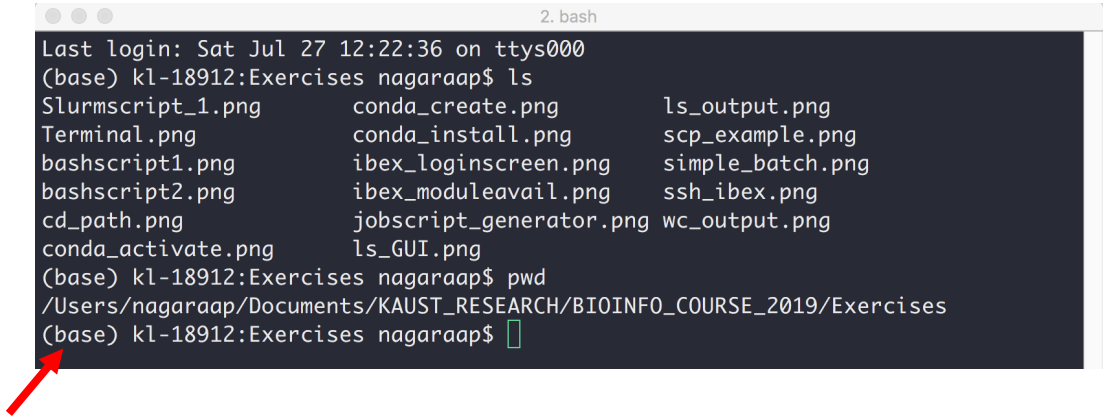
# Anaconda makes your life easy !!



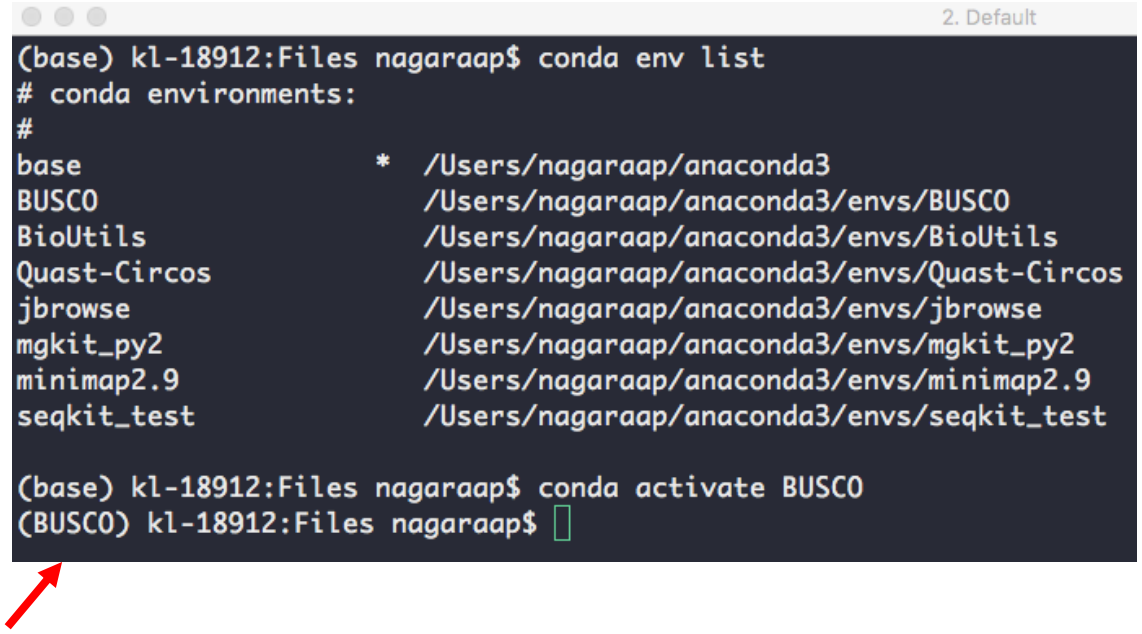

| Using Packages and Channels | |
|---|---|
| Search for a package in currently configured channels with version range >=3.1.0, <3.2" | `conda search PKGNAME=3.1 "PKGNAME [version='>=3.1.0,<3.2']"` |
| Find a package on all channels using the Anaconda Client | `anaconda search FUZZYNAME` |
| Install package from a specific channel | `conda install conda-forge::PKGNAME` |
| Install a package by exact version number (3.1.4) | `conda install PKGNAME==3.1.4` |
| Install one of the listed versions (OR) | `conda install "PKGNAME[version='3.1.2\|3.1.4']"` |
| Install following several constraints (AND) | `conda install "PKGNAME>2.5,<3.2"` |
| Add a channel to your Conda configuration | `conda config --add channels CHANNELNAME` |

| Additional Useful Hints | |
|---|---|
| Detailed information about package versions | `conda search PKGNAME --info` |
| Remove unused cached files including unused packages | `conda clean --all` |
| Remove a package from an environment | `conda uninstall PKGNAME --name ENVNAME` |
| Update all packages within an environment | `conda update --all --name ENVNAME` |
| Run most commands without requiring a user prompt. Useful for scripts. | `conda install --yes PKG1 PKG2` |
| Examine Conda configuration and configuration services | `conda config --show`<br>`conda config --show-sources` |

Install anaconda - ~650 Mb (prefer python 3 version)




Proceed to Hands-on 4. Easy installations : 20 minutes



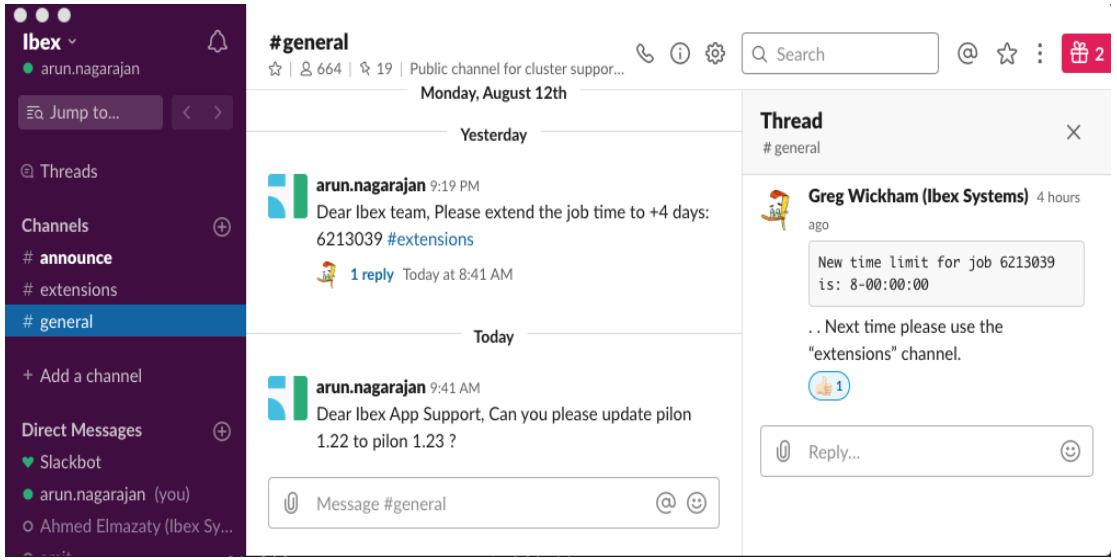
# Module 5 – WORK ON REMOTE SERVER

KAUST provides a centralized storage space, software installations and support for users

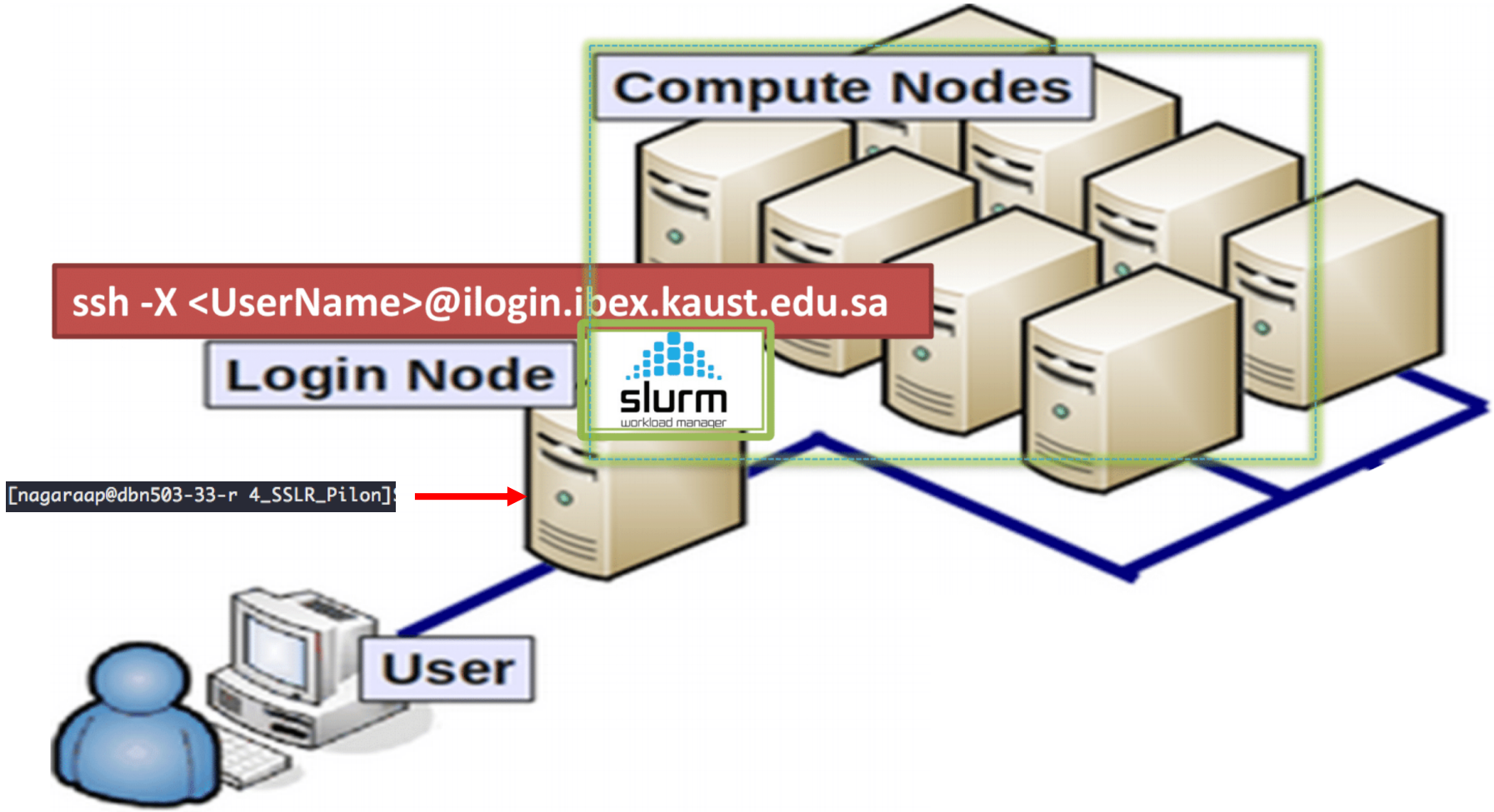
Perform computationally intensive tasks in remote server

```
1. nagaraap@dbn503-33-r:/ibex/scratch/nagaraap/amd/AIP_PBLR/CC7_PBLR/2_CC75cell_canu/2.1_pur
(base) [nagaraap@dbn503-33-r 4_SSLR_Pilon]$ module avail

---------------------------------------- /sw/csi/modulefiles/applications ----------
abaqus/2017                                  maestro/1.7.1
abyss/2.0.2                                  mafft/7.407-with-extensions
abyss/2.1.4                                  mafft/7.407-without-extensions
adf/2016.102/precompiled                     maker/2.31.10
adf/2016.106/precompiled                     marvel/6d68df1/el7.5_gnu6.4.0_python
amber/18/gnu-6.4.0                           mashtree/0.33/gnu6.4.0_perl5.26.1
amber/18/openmpi-3.0.0                       masurca/3.2.4
ambertools/18/anaconda3env                   materialstudio/2017R2
ambertools/18/openmpi_3.0.0                  mathematica/11.2.0
amphora2/8742a1                              matlab/R2016b
annotsv/2019                                 matlab/R2017b
annotsv/2.2                                  matlab/R2018a
ansys/18.1                                   mcce/3.0
ansys/19.1                                   mcl-edge/14.137
antismash/4.1/anaconda2-2.5.0                mecat/20170522
anvio/3.0.0                                  medaka/0.8.0/conda3env
anvio/5.2                                    medea/2.21
any2fasta/0.4.2                              meep/1.3/ompi211-intel1602
arcs/1.0.5/gnu-6.4.0                         megahit/1.1.4/gnu6.4.0
arks/1.0.2/gnu-6.4.0                         megan/6.12.3
asciidoc/8.6.10                              meme/5.0.3/gnu-6.4.0
atom/4.2.7-100/gnu-6.4.0                     meraculous/2.2.6/gnu-6.4.0
augustus/3.2.3/boost1.65.1_gnu6.4.0          mercurial/4.7.1
augustus/3.3/boost1.65.1_gnu6.4.0            meshclust2/2.3.0
```



# Remote login




Source: https://wiki.rc.hms.harvard.edu/
Acknowledgements to Harvard university

# IBEX CHEAT SHEET

Ibex is a heterogeneous cluster with a mix of AMD, INTEL and NVIDIA GPUs.

**To Login:**

Intel nodes:

```
ssh -X <UserName>@ilogin.ibex.kaust.edu.sa
```

AMD nodes:

```
ssh -X <UserName>@alogin.ibex.kaust.edu.sa
```

GPU nodes:

```
ssh -X <UserName>@glogin.ibex.kaust.edu.sa
```

**Application installation :**

All compilers, libraries and applications are installed on each login node due to variation in the system architecture. Intel, AMD and GPU based architecture specific applications are available through modules.

**Application availability:**

```
$module avail
$module avail <ApplicationName>
```

**Application loading:**

```
$module load <ApplicationName>
$module load <ApplicationName>/<version>
```

**Job Submission (batch mode):**

To set memory requirement: `--mem=<in MB>`
To select architecture specific node type:
`--constraint=intel|amd`
`--gres=gpu`
To set number of nodes: `--nodes`
To set number of tasks (for parallel processing): `--ntasks`
To set the number of core per tasks : `--cpus-per-task`
To set wall clock time: `--time`
To set the node as dedicated for the job: `--exclusive`
To set the file name for standard err: `--error`
To set the file name for standard out: `--output`

**Tunable job script generator for IBEX is available in**:
https://www.hpc.kaust.edu.sa/ibex/job

**Example Job Script:**

```
#!/bin/bash
## SLURM Resource requirement:
#SBATCH --nodes=1
#SBATCH --cpus-per-task=8
#SBATCH --job-name=spades
#SBATCH -output=myjob.%J.out
#SBATCH --error=myjob.%J.err
#SBATCH --time=8:00:00

## Required software list:
  module load gaussian09/d.01/precompiled
## Run the application:
echo "This job ran on $SLURM_NODELIST dated `date`";
  srun g09 < testgau.inp > testgau.out
```

**Job Submission queues:**

There are 2 queues, the default batch is for production runs and the debug is for interactive debugging the jobs.

**To use debug queue (for example):**

```
salloc --time=5:00 --nodes=1 \
--partition=debug
```

**Other Slurm Commands:**

`sbatch`: to run jobs
`sinfo`: to check node availability
`squeue`: to check job status
`scancel job#`: to cancel jobs

**General Tips:**

- Do to run on the logins nodes, always submit your jobs through scripts.
- Logins are designed for compilations and edits.
- Always run your jobs from the scratch.
- Remember to clean up your scratch.

**Filesystem:**

- `/home/<UserName>`: *Home directory for important data backup.*
- Always use the `/scratch` filesystem to *submit jobs from amd/intel/gpu* nodes.
- Use `/fscratch` if your jobs require a high number of IOPS.

**Contact for Help/Support:**

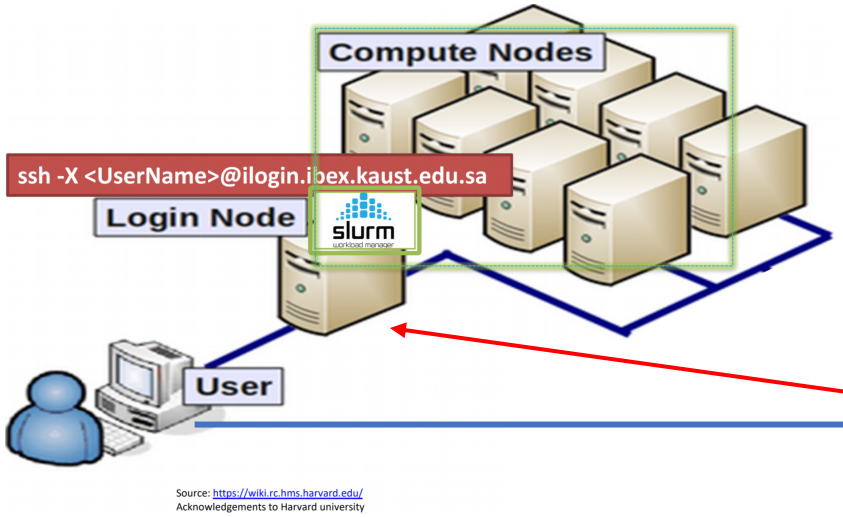
**Application installation/failure/support:**
cluster-apps@hpc.kaust.edu.sa
**System issues/failure/support:**
cluster-systems@hpc.kaust.edu.sa

**Our website:**
https://www.hpc.kaust.edu.sa/ibex

# Slurm Script

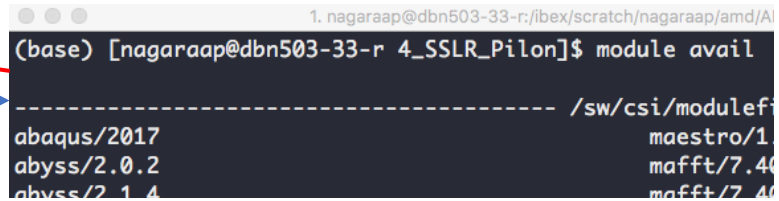
```
#!/bin/bash
#SBATCH -N 1
#SBATCH --partition=batch
#SBATCH -J genelist
#SBATCH -o genelist.%J.out
#SBATCH -e genelist.%J.err
#SBATCH --mail-user=arun.nagarajan@kaust.edu.sa
#SBATCH --mail-type=ALL
#SBATCH --time=00:10:00
#SBATCH --mem=100M

#run the application:
grep "\tgene" E.coli_genomic.gff |cut -f 9 |cut -d ';' -f1 |cut -d '=' -f2 |sort|uniq > Slurm_E.coli_genelist.out
```

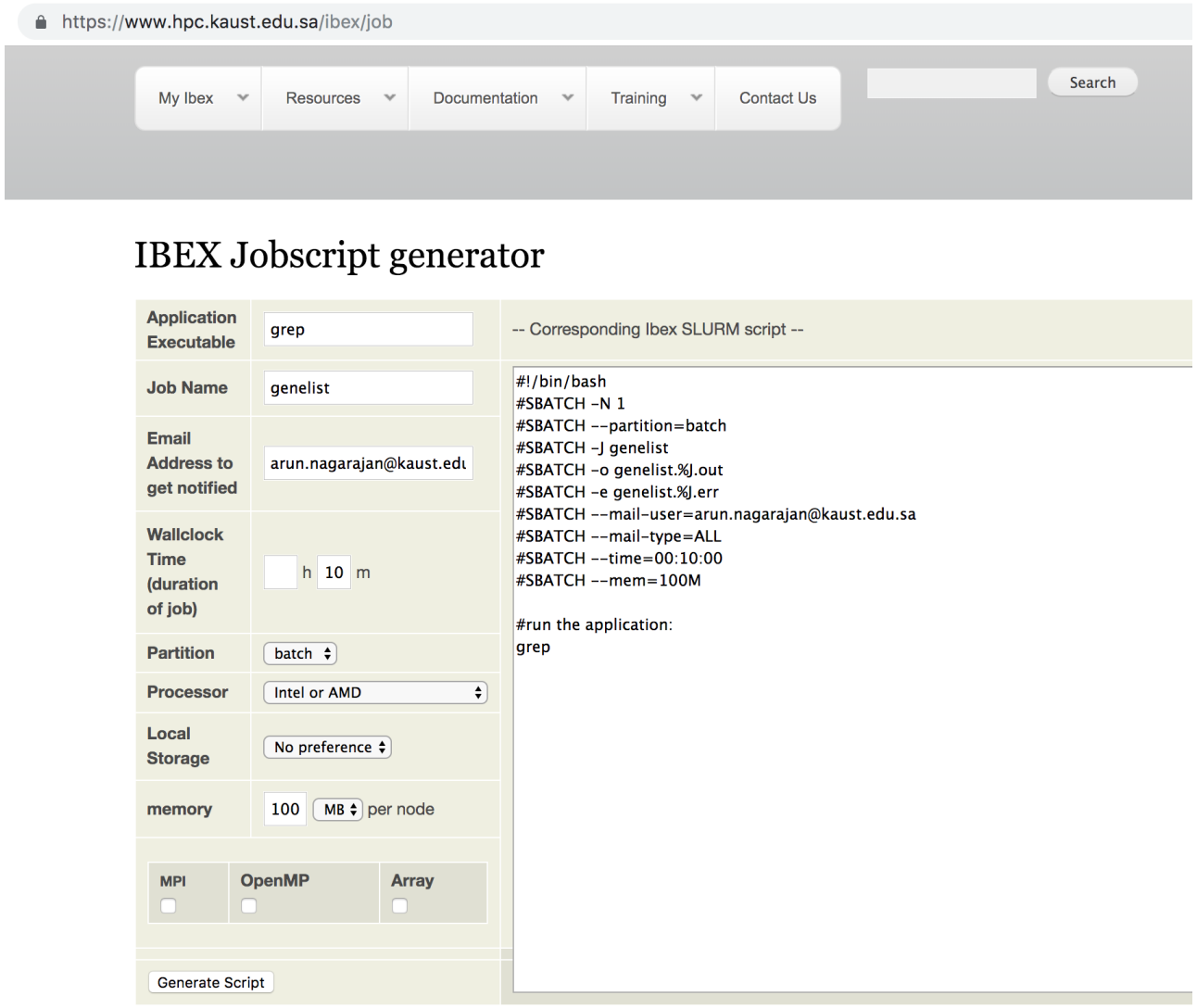


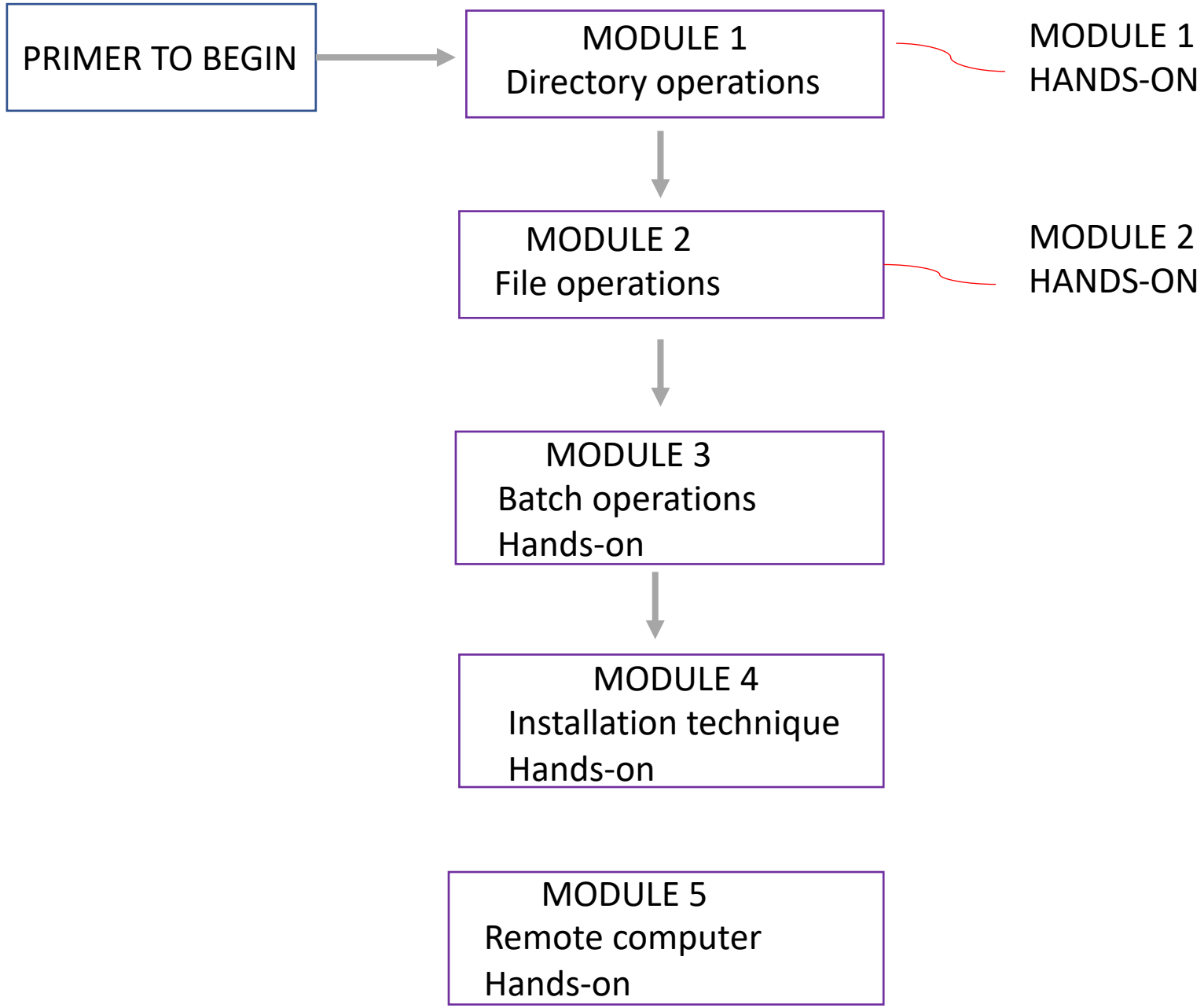

Source: https://wiki.rc.hms.harvard.edu/
Acknowledgements to Harvard university

#SBATCH –N 1 ➔ Request 1 node to do the task
#SBATCH –time → For 10 minutes
#SBATCH –mem → with 100 Mb memory



# Lazy way to create job script..




PRIMER TO BEGIN
MODULE 1
Directory operations
MODULE 1
HANDS-ON
MODULE 2
File operations
MODULE 2
HANDS-ON
MODULE 3
Batch operations
Hands-on
MODULE 4
Installation technique
Hands-on
MODULE 5
Remote computer
Hands-on