# Linux: Beyond the Basics

Bingbing Yuan
BaRC Hot Topics – Oct. 2021
Bioinformatics and Research Computing
Whitehead Institute

http://barc.wi.mit.edu/hot_topics/

Whitehead Institute

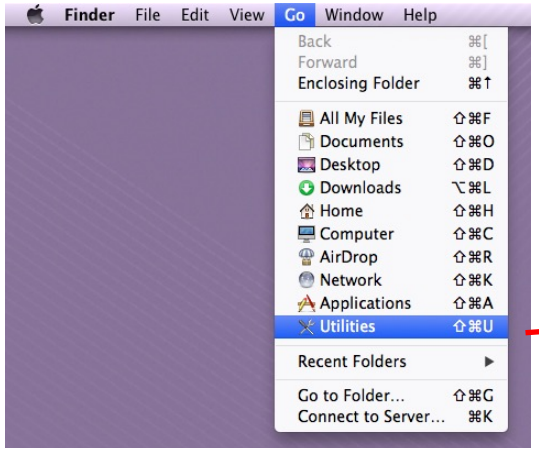
# Logging in to our Linux server

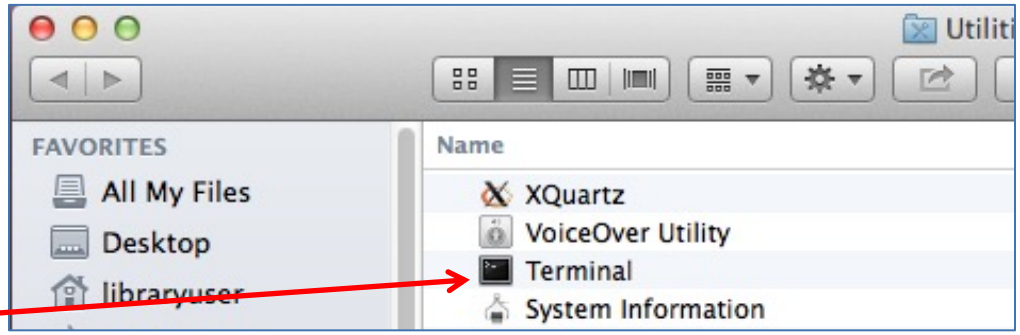
- Our main Unix/Linux server is called tak
- Request a tak account:
  https://element.wi.mit.edu/am/?f=unixRequest

- Connecting to tak:
  http://bioinfo.wi.mit.edu/bio/software/unix/serverConnect.php

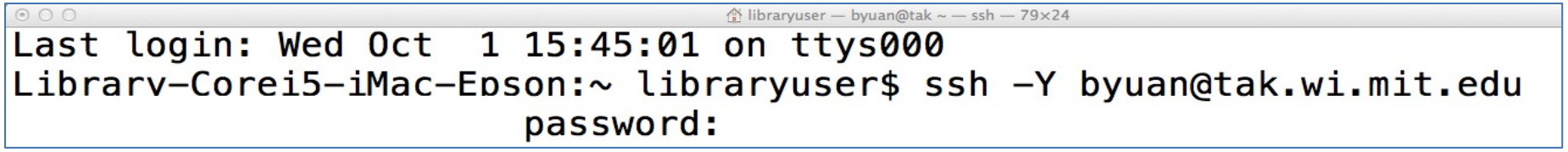
  – Windows:
    - MobaXterm
  – Mac
    - Access through Terminal

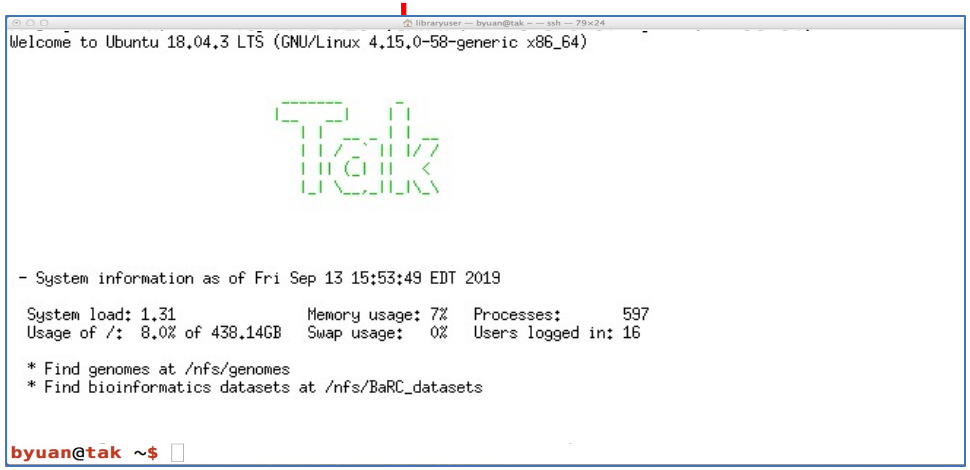
# Log in to tak for Mac




ssh –Y username@tak.wi.mit.edu

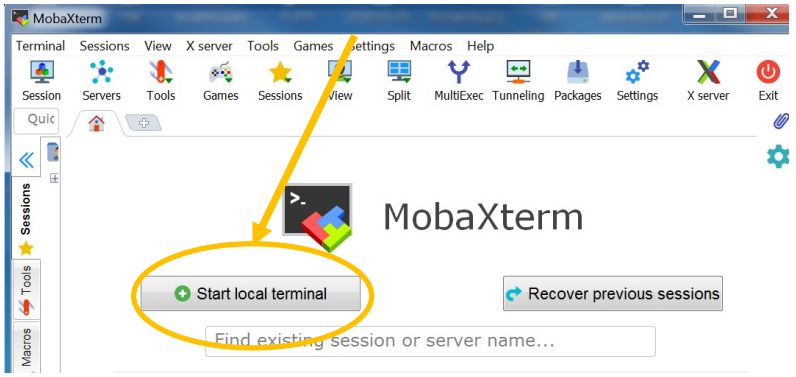


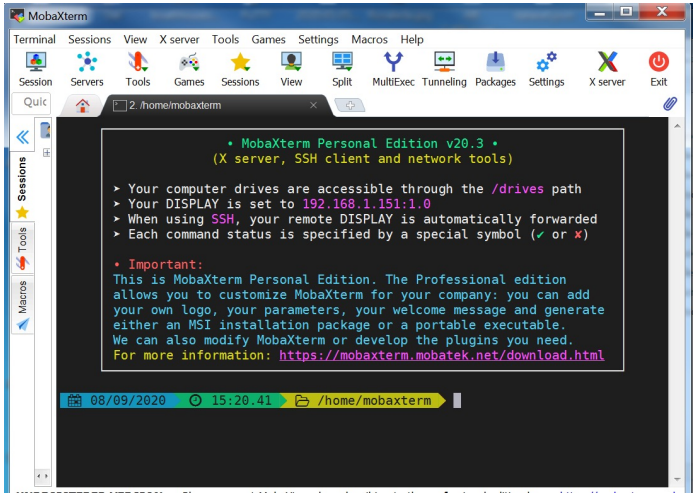



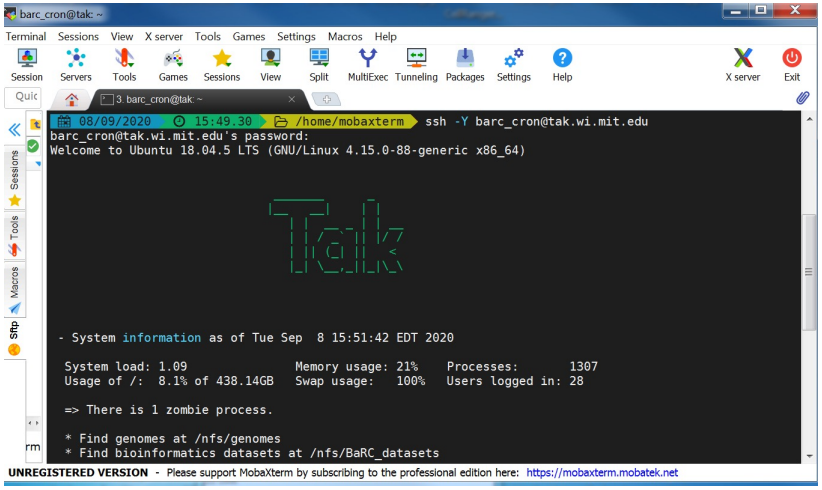
# Connecting to tak from Windows

1) Open MobaXterm

2) Click on the ”Start local terminal” button




3) Type:

```
ssh -Y username@tak.wi.mit.edu
```

*Note:*

When you write the password you won’t see any characters being typed.

# Hot Topics website: http://barc.wi.mit.edu/education/hot_topics/

- After you login to tak, create a directory for the exercises within your home directory, and use it as your working directory

  $ mkdir Hot_Topics

  $ cd Hot_Topics

- Copy all files into your working directory

  $ cp -r /nfs/BaRC_training/Linux_Beyond/data_files/* .

- You should have the files below in your working directory:
- **datasets**/ Ensembl_info.txt exercise.txt foo.txt Gene_exp.txt HumanGenes PlusMinus3kb.bed peaks.bed sample1.txt exercise.pdf
  - You can check they're there with the 'ls' command

# Linux Review: Commands

- **`command [arg1 arg2 … ] [input1 input2 … ]`**

$ sort -k2,3nr foo.tab

start end

**-n** or **-g**: -n is recommended, except for scientific notation or a leading '+'

**-r**: reverse order

$ cut -f1,5 foo.tab

$ cut -f1-5 foo.tab

-f: select only these fields
-f1,5: select 1st and 5th fields
-f1-5: select 1st, 2nd, 3rd, 4th, and 5th fields

$ wc -l foo.txt

How many lines are in this file?

# Linux Review: Common Mistakes

- Case sensitive

**`cd /nfs/Barc_Public`** is different from **`cd /nfs/BaRC_Public`**

```
-bash: cd: /nfs/Barc_Public: No such file or directory
```

- Spaces may matter!

  **`rm –f myFiles*`** **vs** **`rm –f myFiles *`**

- Office applications can convert text to special characters that Linux won't understand
  - Smart quotes, dashes
  - Carriage return from DOS
    - Use fromdos to remove carriage return

# Linux Review: Pipes

- Stream output of one command/program as input for another
  - Avoid intermediate file(s)
  - Merge multiple commands in to one long command

➢ $ cut -f1 myFile.txt | sort | uniq -c > uniqCounts.txt

↗ pipes ↗

# What we will discuss today

- Aliases (to reduce typing)
- sed (for file manipulation)
- awk(to filter by column)
- join (merge files)
- groupBy and intersect from bedtools (not typical Linux)
- loops (one-line and with shell scripts)
- scripting (to streamline commands)

# Aliases

- Add a one-word link to a longer command
- To get current aliases (from ~/.bashrc)

```
alias
```

- Create a new alias (two examples)

```
alias sp='cd /lab/solexa_public/Reddien'
alias picard='java -jar /usr/local/share/picard-tools/picard.jar'
```

- Make an alias permanent
  - Paste command(s) in ~/.bashrc

# sed:

## stream editor for filtering and transforming text

- Print lines 10 - 15:

```
$ sed -n '10,15p' bigFile > selectedLines.txt
```

- Delete 5 header lines at the beginning of a file:

```
$ sed '1,5d' file > fileNoHeader
```

- Remove all version numbers (eg: '.1') from the end of a list of sequence accessions: eg. NM_000035.2

```
$ sed 's/\.[0-9]\+//g' accsWithVersion > accsOnly
```

*s: substitute* *g: global modifier (change all)*

- Get good examples from “sed cheat sheet”:
  - https://www.pement.org/sed/sed1line.txt

# Join files together

## With Linux join

```
$ join -1 1 -2 2 --nocheck-order -t $'\t' sorted_File1 sortedFile2
```

Join files on the 1st field of FILE1 with the 2nd field of FILE2,

only showing the common lines.

FILE1 and FILE2 must be sorted on the join fields before running join

-t $'\t' : Both input and output use tab as separator

--nocheck-order: good for sorted files with header line.

Sorted sample tables to join:

| Symbol | Heart | Skeletal Muscle | Skin | Smooth Muscle | Spinal cord |
|---|---|---|---|---|---|
| HHAT | 8.15 | 7.7 | 5 | 6.55 | 6.4 |
| INPP5D | 19.65 | 5.95 | 4.55 | 5.25 | 14.5 |
| NDUFA10 | 441.8 | 160.2 | 24.9 | 188.85 | 158.75 |
| RPS6KA1 | 85.2 | 47.75 | 46.45 | 35.85 | 44.55 |
| RYBP | 20.45 | 13.05 | 11.95 | 20.7 | 17.75 |
| SLC16A1 | 15.45 | 20.45 | 12.2 | 248.35 | 27.15 |

| Ensembl Gene ID | Symbol |
|---|---|
| ENSG00000280680 | HHAT |
| ENSG00000280820 | LCN1P1 |
| ENSG00000280584 | OBP2B |
| ENSG00000280775 | RNA5SP136 |
| ENSG00000252303 | RNU6-280P |
| ENSG00000280963 | SERTAD4-AS1 |

| Symbol | Heart | Skeletal Muscle | Skin | Smooth Muscle | Spinal cord | Ensembl Gene ID |
|---|---|---|---|---|---|---|
| HHAT | 8.15 | 7.7 | 5 | 6.55 | 6.4 | ENSG00000280680 |

# Regular Expressions

- A sequence of characters defining a search pattern
- Powerful, but syntax is often non-intuitive
- Examples

  List all txt files: `ls *.txt`

  Replace CHR with Chr at the beginning of each line:

  ```
  $ sed 's/^CHR/Chr/' myFile.txt
  ```

  Delete a dot followed by one or more numbers

  ```
  $ sed 's/\.[0-9]\+//g' myFile.txt
  ```

| | Matches |
|---|---|
| . | All characters |
| * | Zero or more; wildcard |
| + | One or more |
| ? | One |
| ^ | Beginning of a line |
| $ | End of a line |
| [ab] | Any character in brackets |

- Note: regular expression syntax may slightly differ between sed, awk, Linux shell, and Perl
  - Ex: \+ in sed is equivalent to + in Perl

# awk

- A simple programing language to process files
- Good for filtering and manipulating multiple-column files
- “awk” comes from the original authors:

  Alfred V. Aho, Peter J. Weinberger, Brian W. Kernighan

# awk

- By default, awk splits each line by spaces

- Print the 2nd and 1st fields of the file:

$ awk ' { print $2"\t"$1 } ' foo.tab

- Convert sequences from tab delimited format to fasta format:

```
$ head -1 foo.tab
Seq1    ACTGCATCAC
$ awk ' { print ">" $1 "\n" $2 }' foo.tab  >  foo.fa
$ head -2 foo.fa
>Seq1
ACGCATCAC
```

# awk: field separator

- Issues with default separator (white space)
  - one field is gene description with multiple words
  - consecutive empty cells
- To use tab as the separator:

```
$ awk -F "\t" '{ print NF }' foo.txt
```

or

```
$ awk 'BEGIN {FS="\t"} { print NF }' foo.txt
```

*BEGIN:* action before read input
*NF:* number of fields in the current record
*FS:* input field separator
*OFS:* output field separator
*END*: action after read input

| Character | Description |
|---|---|
| \n | newline |
| \r | carriage return |
| \t | horizontal tab |

# awk: arithmetic operations

Add average values of 4th and 5th fields to the file:
$ awk '{ print $0 "\t" ($4+$5)/2 }' foo.tab

$0: all fields

| Operator | Description |
|---|---|
| + | Addition |
| - | Subtraction |
| * | Multiplication |
| / | Division |
| % | Modulo |
| ^ | Exponentiation |
| ** | Exponentiation |

# awk: making comparisons

Print out records if values in 4th or 5th field are above 4:

$ awk '{ if( $4>4 || $5>4 ) print $0 } ' foo.tab

| Sequence | Description |
|---|---|
| > | Greater than |
| < | Less than |
| <= | Less than or equal to |
| >= | Greater than or equal to |
| == | Equal to |
| != | Not equal to |
| ~ | Matches |
| !~ | Does not match |
| \|\| | Logical OR |
| && | Logical AND |

# More awk examples

- Conditional statements:

Display expression levels for the gene NANOG:
$ awk '{ if(/NANOG/) print $0 }'  foo.txt
or
$ awk '/NANOG/ { print $0 } ' foo.txt
or
$ awk '/NANOG/' foo.txt

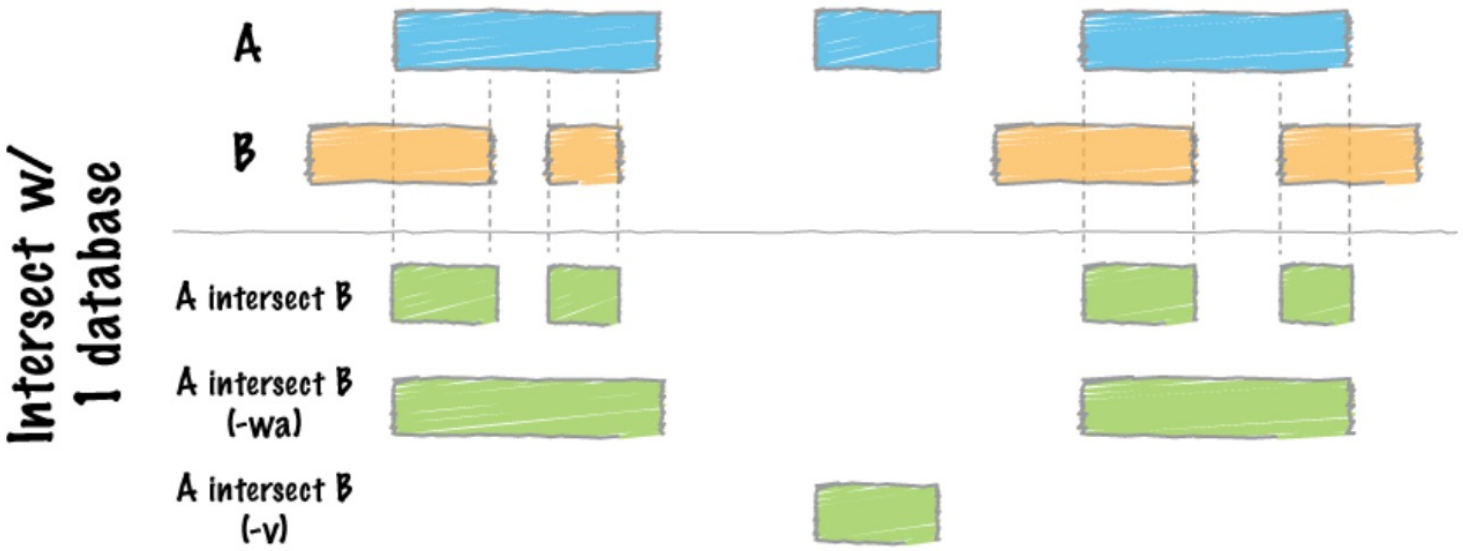
Add line number to the above output:
$ awk '/NANOG/ { print NR"\t"$0 }'  foo.txt
*NR: line number of the current row*

- Looping:

Calculate the average expression (4th, 5th and 6th fields in this case) for each transcript
$ awk '{ total= $4 + $5 + $6; avg=total/3; print $0"\t"avg}' foo.txt
or
$ awk '{ total=0; for (i=4; i<=6; i++) total=total+$i; avg=total/3; print $0"\t"avg }' foo.txt

# intersect from bedtools ( intersectBed )

- Find overlaps between two sets of genomic features




https://bedtools.readthedocs.io/en/latest/content/tools/intersect.html

# intersectBed: Examples

$ head -2 HumanGenesPlusMinus3kb.bed

| | | | |
|---|---|---|---|
| chr1 | 50899700 | 50905978 | ENSG00000271782_RP5-850O15.4 |
| chr1 | 103814769 | 103831355 | ENSG00000232753_RP11-347K2.1 |

$ head -2 peaks.bed

| | | | | |
|---|---|---|---|---|
| chr1 | 19921 | 20016 | MACS_peak_1 | 50 |
| chr1 | 20025 | 20890 | MACS_peak_2 | 568 |

$ intersectBed -wa -wb -a HumanGenesPlusMinus3kb.bed -b peaks.bed | head -2

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| chr1 | 45956538 | 45968751 | ENSG00000236624_CCDC163P | chr1 | 45955389 | 45956863 | MACS_peak_3385 | 1192.24 |
| chr1 | 45956538 | 45968751 | ENSG00000236624_CCDC163P | chr1 | 45957202 | 45957380 | MACS_peak_3386 | 121.87 |

$ intersectBed -a HumanGenesPlusMinus3kb.bed -b peaks.bed | head -2

| | | | |
|---|---|---|---|
| chr1 | 45956538 | 45956863 | ENSG00000236624_CCDC163P |
| chr1 | 45957202 | 45957380 | ENSG00000236624_CCDC163P |

$ intersectBed -a HumanGenesPlusMinus3kb.bed -b peaks.bed | cut -f4 |sort -u |head -2

| |
|---|
| ENSG00000000003_TSPAN6 |
| ENSG00000000419_DPM1 |

# Summarize by Columns: groupBy (from bedtools)

Input file must be pre-sorted by grouping column(s)!

*input*

| Ensembl Gene ID | Ensembl Transcript ID | Symbol |
|---|---|---|
| ENSG00000281518 | ENST00000627423 | FOXO6 |
| ENSG00000281518 | ENST00000630406 | FOXO6 |
| ENSG00000280680 | ENST00000625523 | HHAT |
| ENSG00000280680 | ENST00000627903 | HHAT |
| ENSG00000280680 | ENST00000626327 | HHAT |
| ENSG00000281614 | ENST00000629761 | INPP5D |
| ENSG00000281614 | ENST00000630338 | INPP5D |

| | |
|---|---|
| -g grpCols | column(s) for grouping |
| -c -opCols | column(s) to be summarized |
| -o | Operation(s) applied to opCol:<br><br>sum, count, min, max, mean, median, stdev, collapse (comma-sep list)<br>distinct (non-redundant comma-sep list) |

Print the gene ID (1st column), the gene symbol , and a list of transcript IDs (2nd field)

$ sort -k1,1 Ensembl_info.txt | groupBy -g 1 -c 3,2 -o distinct,collapse

*Partial output*

| Ensembl Gene ID | Symbol | Ensembl Transcript ID |
|---|---|---|
| ENSG00000281518 | FOXO6 | ENST00000627423,ENST00000630406 |
| ENSG00000280680 | HHAT | ENST00000625523,ENST00000626327,ENST00000627903 |

# Shell Flavors

- Syntax (for scripting) depends the shell

  echo $SHELL  # /bin/bash (on tak)

- bash is common and the default on tak.
- Some Linux shells (incomplete listing):

| Shell | Name |
|---|---|
| sh | Bourne |
| **bash** | **Bourne-Again** |
| ksh | Korn shell |
| csh | C shell |

# Shell script advantages

- Automation: avoid having to retype the same commands many times
- Ease of use and more efficient
- Outline of a script:

#!/bin/bash ⟵ shebang: interprets how to run the script

*commands…* ⟵ *set of* commands used in the script

#comments ⟵ write comments using "#"

- Commonly used extension for script is .sh (eg. foo.sh), file must have executable permission

# Bash Shell: ‘for’ loop

- Process multiple files with one command
- Reduce computational time with many cluster nodes

```
for mySam in `/bin/ls *.sam`
do
        bsub  wc -l $mySam
done
```

*When referring to a variable, $ is needed before the variable name ($mySam), but $ is not needed when defining it (mySam).*

Identical one-line command:

```
for samFile in `/bin/ls *.sam`; do bsub wc -l $samFile; done
```

# Shell script example

```
#!/bin/bash

# 1.  Take two arguments: the first one is a directory with all the datasets, the second one is for output
# 2. For each file, calculate average gene expression, and save the results in a file in the output directory

inDir=$1                    # 1st argument
outDir=$2                   # 2nd argument; outDir must already exist
                            # Define variables: no spaces on either side of the equal sign
for i in `/bin/ls $inDir `  # refer to variable with $
do
        # output file name
        outFileName="${i}_avg.txt"          # {}: $i_avg is not valid;
        # calculate average gene expression
        # NM_001039201    Hdhd2   5.0306    5.3309   5.4998
        bsub "sort -k2,2 $inDir/$i | groupBy -g 2 -c 3,4,5 -o mean,mean,mean >| $outDir/$name"
done
```

*# You can use editors such as nedit, gedit, emacs, or pico to create shell scripts.*
*# If you use Windows or Macs text editor, save as simple text format.*
*# Special hidden Windows characters can be be removed with fromdos command*

# Accessing Shared Resources at Whitehead

- Linux
  - **/nfs/BaRC_Public**
  - **/nfs/BaRC_training**
  - **/lab/solexa_public**
- Windows (access using *Start Menu* → *Search*)
  - **\\wi-files1\BaRC_Public**
  - **\\wi-files1\BaRC_training**
  - **\\wi-bigdata\solexa_public**
- Macs (access using *Go* → *Connect to Server…*)
  - **smb://wi-files1/BaRC_Public**
  - **smb://wi-files1/BaRC_training**
  - **smb://wi-bigdata/solexa_public**

Where's my lab's share?

- http://it.wi.mit.edu/systems/file-storage/lab-share-paths

# Further Reading

- BaRC one liners:
  - http://bioinfo.wi.mit.edu/bio/bioinfo/scripts/#unix
- Linux Info for Bioinfo:
  - http://bioinfo.wi.mit.edu/bio/education/unix_intro.php
- Bash Guide for Beginners:
  - http://tldp.org/LDP/Bash-Beginners-Guide/html/

# Upcoming Hot Topics

http://barc.wi.mit.edu/education/hot_topics/upcoming/

- Python: An Introduction - November
- Python: Advanced Topics - November
- Analytical project management [TBA]