Workshop 1: **Introduction to UNIX command-line**

Day 3

Peter Scott, PhD | pscott17@ucla.edu

CBI Fellow

“Swiss Army knife” set of tools

Summary of Day 2

file permissions

cat

wc

>, >>, <

pipeline

ln –s

grep

regex

sed

awk

- awk is both a
  - programming language
  - text processor

whitespace (spaces, tabs, etc.)
to separate fields

- parsing and manipulating **tabular** data
  - iterates through the entire file line-by-line

```
awk ‘{action_to_take}’ <file_to_parse>
```

awk : Simple Uses

action_to_take

```
awk '{print}’ <file_to_parse>
```

```
awk '{print $1}’ <file_to_parse>
```

column number

```
awk '{print $1”\t”$3}’ <file_to_parse>
```

delimiter

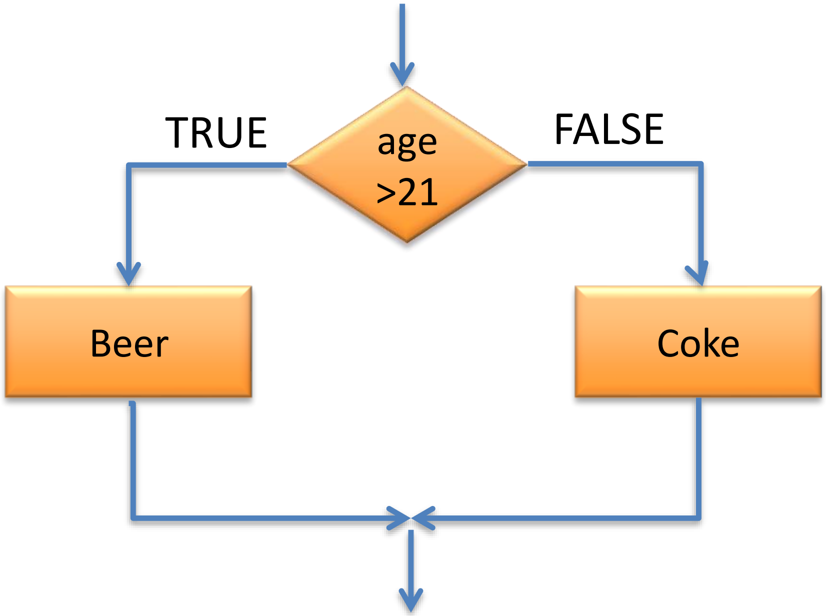
Let’s practice

```
awk '{print}' hg19.gtf
awk '{print $1}’ hg19.gtf
awk '{print $4}' hg19.gtf
awk '{print $1"\t"$4} hg19.gtf
awk '{print $1"\t"$4*$5}' hg19.gtf
```

awk : If statement

condition

```
if(age>21) then
    print("Beer Please")
```



awk : If statement

Print those employees who actually worked

```
Beth      4.00    0
Dan       3.75    0
Kathy    4.00    10
Mark     5.00    20
Mary     5.50    22
Susie     4.25    18
```

tabular data

```
awk '{if($3>0) print}' emp.txt
```

Data type

- Numbers
- **Text**

```
1           clothing        3141
1           computers       9161
1           textbooks       21312
2           clothing        3252
2           computers       12321
2           supplies        2242
2           textbooks       15462
```

Print information about computers only

```
awk '{if($2=="computers") print}' sales.txt
```

use double quote

Text

Let’s practice!

```
awk '{if($3>0) print}' emp.txt
awk '{if($3>0) print $1"\t"$2*$3}' emp.txt
```

```
Beth     4.00    0
Dan      3.75    0
Kathy   4.00    10
Mark    5.00    20
Mary    5.50    22
Susie    4.25    18
```

Let’s practice!

```
awk '{if($4>50) print}’ hg19.gtf
awk '{if($1=="chr2") print}’ hg19.gtf
awk '{if($1=="chr2") print $1"\t"$3}’ hg19.gtf
```

```
chr2    hg18_knownGene_GnfAtlas2    exon    237538  237602  0.000000    -   .   gene_id "204019_s_at"; transcript_id "uc002qvu.1";
chr2    hg18_knownGene_GnfAtlas2    exon    239731  239852  0.000000    -   .   gene_id "204019_s_at"; transcript_id "uc002qvu.1";
```

What If I only want genes larger than 500bp?
How about >500 on chr2?

Let’s practice!

```
awk '{if($4>50) print}’ hg19.gtf
awk '{if($1=="chr2") print}’ hg19.gtf
awk '{if($1=="chr2") print $1"\t"$3}’ hg19.gtf
```

```
chr2   hg18_knownGene_GnfAtlas2     exon   237538  237602  0.000000     -  .    gene_id "204019_s_at"; transcript_id "uc002qvu.1";
chr2   hg18_knownGene_GnfAtlas2     exon   239731  239852  0.000000     -  .    gene_id "204019_s_at"; transcript_id "uc002qvu.1";
```

What If I only want genes larger than 500bp?

```
awk '{if($5-$4>500) print}' hg19.gtf
```

How about >500 on chr2?

```
awk '{if($1=="chr2" && $5-$4>500) print}' hg19.gtf
grep '^chr2\s' hg19.gtf | awk '{if($5-$4>500) print}'
```

awk : sum

- Calculate sum of a particular column

```
awk '{sum+=$2} END { print sum}' emp.txt
```

New variable

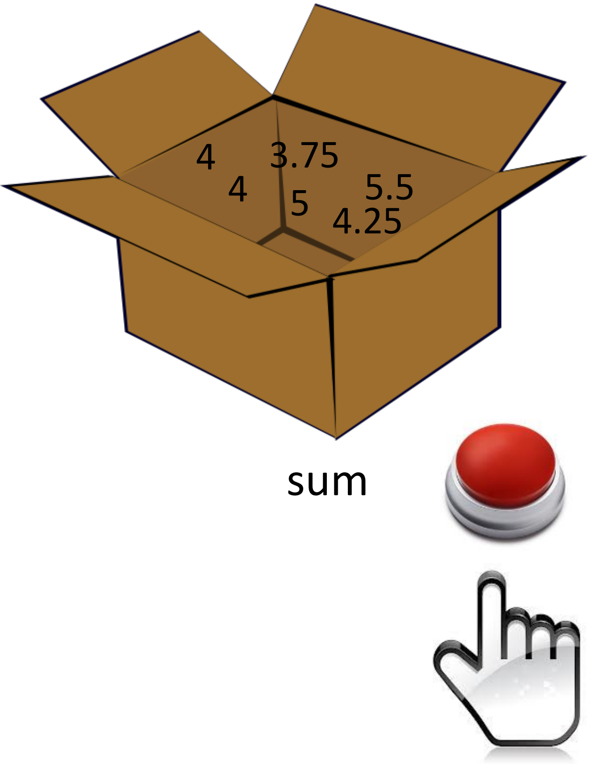
"+=" means to add the value to the variable sum

Do it until the end of the file

A variable is like a **box** where we can store a value and reuse this same value multiple times in our program.

Variable is like a box

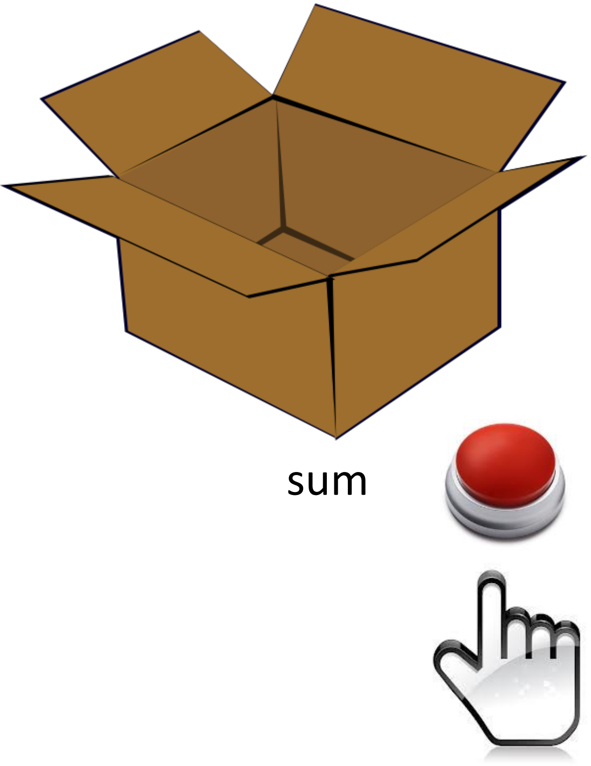
```
awk '{sum+=$2} END { print sum}' emp.txt
```



```
Beth      4.00    0
Dan       3.75    0
Kathy    4.00    10
Mark     5.00    20
Mary     5.50    22
Susie     4.25    18
```

Variable is like a box

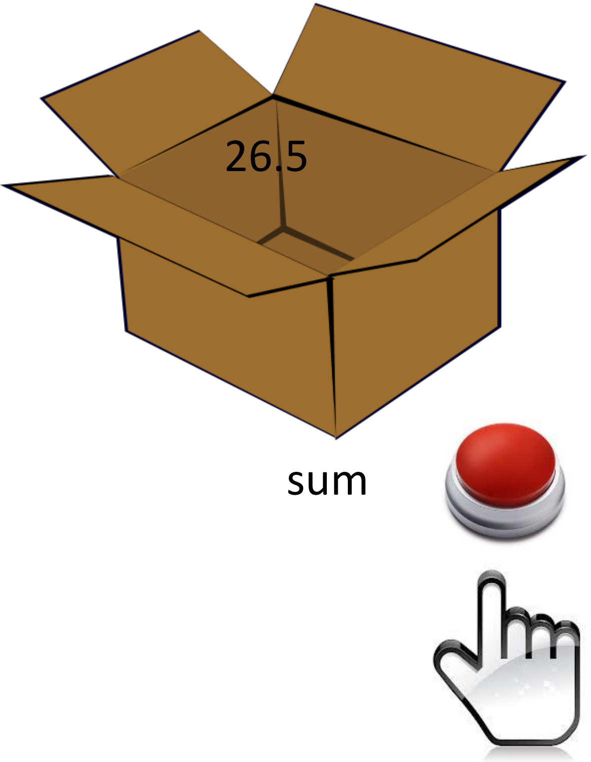
```
awk '{sum+=$2} END { print sum}' emp.txt
```



```
Beth      4.00    0
Dan       3.75    0
Kathy    4.00    10
Mark     5.00    20
Mary     5.50    22
Susie     4.25    18
```

Variable is like a box

```
awk '{sum+=$2} END { print sum}' emp.txt
```



```
Beth       4.00     0
Dan        3.75     0
Kathy    4.00      10
Mark     5.00      20
Mary     5.50      22
Susie      4.25     18
```

Let’s practice!

```
awk '{sum+=$4} END {print sum}’ hg19.gtf

awk ' END {print NR}' hg19.gtf
wc –l hg19.gtf
```

NR=number of records (here that is number of lines)

```
awk '{sum+=$4} END {print sum/3000}’ hg19.gtf
awk '{sum+=$4} END {print sum/NR}’ hg19.gtf
awk '{if($1=="chr2") sum+=$4} END {print sum}’ hg19.gtf

awk '{sum+=$4} END {printf "%8.3f\n", sum/NR}' hg19.gtf
```

Sort

- will rearrange the lines in a text file so that they are sorted, numerically and alphabetically.

```
sort [OPTION]... [FILE]...
```

Options :

- -n - numerical ordering
- -k - sort by a particular column

Sort a file

regex.txt

```
beat
brat
boat
bat
banana
```

```
sort regex.txt
```

Sort a text file

```
sort -n regex.txt
```

Sort a file with numbers

Sort by column

```
sort -k 2 sales.dat
```

Column number

```
sort -k 3n sales.dat
```

sales.dat

```
1     clothing      3141
1     computers      9161
1     textbooks     21312
2   clothing      3252
3   …
```

Uniq

- Removes duplicate lines from a file*

```
uniq [OPTION]... [INPUT]
```

Options :

- -c - how many times each line occurred
- -d - print only duplicated lines

*assumes that the file is sorted

Let’s practice

```
$ sort -n numbers.txt >numbers_sort.txt
$ uniq numbers_sort.txt
$ sort numbers.txt | uniq
$ uniq -d numbers_sort.txt
$ uniq -c numbers_sort.txt
```

```
2 7
```

a line of the file

How many times it occurs

numbers.txt

```
3
4
5
7
2
1
6
7
```

Shell scripts

- If you have a bunch of commands you’d like to automate, you can put them on separate lines of a file.

My first shell script!

Following steps are required to write shell script:

- Use any **text** editor like to write shell script.
- After writing shell script set execute permission for your script

My first shell script!

```
nano script.sh
```

```
#!/bin/bash
echo "My first script”
```

How to run the script :

```
chmod u+x script.sh
./script.sh
```

More scripting

- Let’s create a bash script which will split <gtf> into files corresponding to every chr (2,3,21), save every file in separate directory called chr${i}_gtf.

More scripting

nano script.sh

```
#!/bin/bash
echo "My first script"

mkdir chr2_gtf
mkdir chr3_gtf
mkdir chr21_gtf

grep "chr2\s" $1 >chr2_gtf/chr2.gtf
grep "chr3\s" $1 >chr3_gtf/chr3.gtf
grep "chr21\s" $1 >chr21_gtf/chr21.gtf
```

How to run the script :

```
./script.sh hg19.gtf
```

How to call a program on Hoffman:

- Install locally and call the program.
  - `java -Xmx4g -jar \ /u/home/m/myname/GATK.jar …..`
- Use the module pre-loaded on Hoffman.
  - `module load gatk`
  - Can also go at the top of the shell script
- Make a profile to load local programs
  - `source gatk_profile`

Environmental modules

- set you environment to be able to run bioinformatics tools
- List of software: https://www.hoffman2.idre.ucla.edu/software/
- `module available`

```
module load <modulefile>
```

```
bowtie
BWA
samtools
tophat
```

Let’s practice

```
[pscott17@login1 test]$ bwa
-bash: bwa: command not found
[pscott17@login1 test]$ module load bwa
[pscott17@login1 test]$ bwa
```

How to run a program on Hoffman

- In an interactive shell:

- Submit to the queue:

  -
    ```
    qsub  -cwd -V -m bea -N <jobname> -l h_data=4G,h_rt=24:00:00 myshell.sh
    ```

  - qsub = submit a job
  - -cwd = run from this current working directory(relative paths)
  - -V = keep these environmental variables
  - -m bea = email at **b**eginning, **e**nd, and **a**bort of job
  - -N <jobname> = names the job if you don’t want the default #
  - -l h_data-4G,h_rt=24:00:00 = requested memory/time

amount of memory
(in megabytes M, or
gigabytes G) that
your job will require
4GB-32GB

How to run a program on Hoffman

- Check the queue:
  - `qstat` = lists all jobs
  - `qstat -u <userName>` = list only your jobs
- Check job usage (once done):
  - `qacct -j job_ID`

How to run a program on Hoffman

- HEEEEEELLLLLPPPP!!!!!:
  - https://www.hoffman2.idre.ucla.edu/faq/
  - https://www.ccn.ucla.edu/wiki/index.php/Hoffman2:Submitting_Jobs
  - https://www.hoffman2.idre.ucla.edu/computing/running/

Let’s practice!

```
. /u/local/Modules/default/init/modules.sh
module load bwa
bwa mem toy.ref.fasta toy.reads.fastq>toy.reads.bwa.sam
```

nano bwa.sh

- `qsub -cwd -V -N testBWA -l h_data=2G,time=1:00:00 bwa.sh`
- `qstat -u pscott17`
- `qstat | grep pscott17`

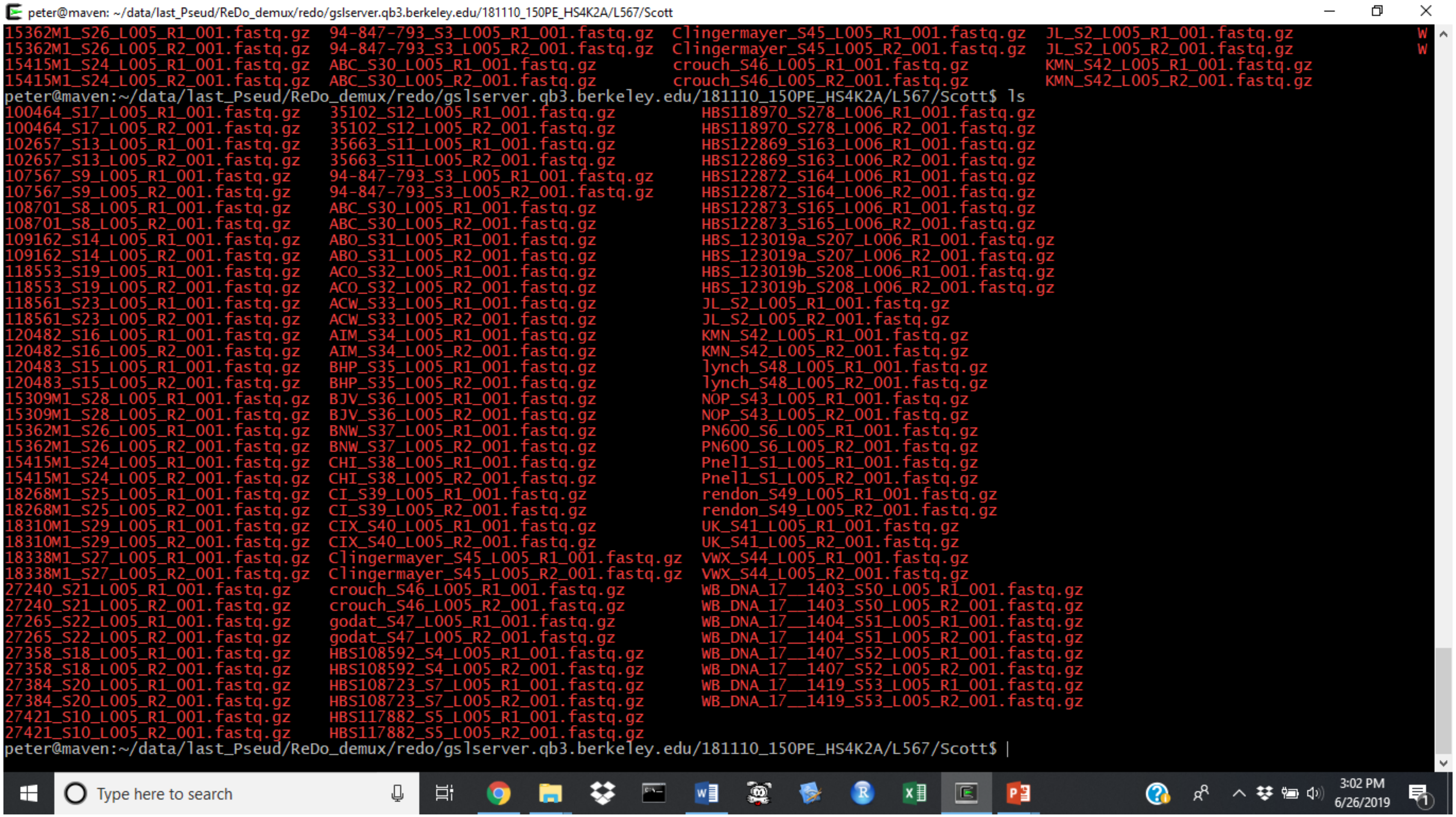
Displays all the jobs which are running on hoffman2

toy.reads.bwa.sam:

```
r1      0       ref     9       60      30M  *        0       0
        ACTGGGGACTGGGGTTTTTTTGGACTGG
        ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~    NM:i:0
        MD:Z:30    AS:i:30XS:i:0
```

After the job is done, toy.reads.bwa.sam will be created

But I have DATA…



peter@maven: ~/data/last_Pseud/ReDo_demux/redo/gslserver.qb3.berkeley.edu/181110_150PE_HS4K2A/L567/Scott

```
15362M1_S26_L005_R1_001.fastq.gz  94-847-793_S3_L005_R1_001.fastq.gz  Clingermayer_S45_L005_R1_001.fastq.gz  JL_S2_L005_R1_001.fastq.gz             W
15362M1_S26_L005_R2_001.fastq.gz  94-847-793_S3_L005_R2_001.fastq.gz  Clingermayer_S45_L005_R2_001.fastq.gz  JL_S2_L005_R2_001.fastq.gz             W
15415M1_S24_L005_R1_001.fastq.gz  ABC_S30_L005_R1_001.fastq.gz        crouch_S46_L005_R1_001.fastq.gz        KMN_S42_L005_R1_001.fastq.gz
15415M1_S24_L005_R2_001.fastq.gz  ABC_S30_L005_R2_001.fastq.gz        crouch_S46_L005_R2_001.fastq.gz        KMN_S42_L005_R2_001.fastq.gz
peter@maven:~/data/last_Pseud/ReDo_demux/redo/gslserver.qb3.berkeley.edu/181110_150PE_HS4K2A/L567/Scott$ ls
100464_S17_L005_R1_001.fastq.gz   35102_S12_L005_R1_001.fastq.gz         HBS118970_S278_L006_R1_001.fastq.gz
100464_S17_L005_R2_001.fastq.gz   35102_S12_L005_R2_001.fastq.gz         HBS118970_S278_L006_R2_001.fastq.gz
102657_S13_L005_R1_001.fastq.gz   35663_S11_L005_R1_001.fastq.gz         HBS122869_S163_L006_R1_001.fastq.gz
102657_S13_L005_R2_001.fastq.gz   35663_S11_L005_R2_001.fastq.gz         HBS122869_S163_L006_R2_001.fastq.gz
107567_S9_L005_R1_001.fastq.gz    94-847-793_S3_L005_R1_001.fastq.gz     HBS122872_S164_L006_R1_001.fastq.gz
107567_S9_L005_R2_001.fastq.gz    94-847-793_S3_L005_R2_001.fastq.gz     HBS122872_S164_L006_R2_001.fastq.gz
108701_S8_L005_R1_001.fastq.gz    ABC_S30_L005_R1_001.fastq.gz           HBS122873_S165_L006_R1_001.fastq.gz
108701_S8_L005_R2_001.fastq.gz    ABC_S30_L005_R2_001.fastq.gz           HBS122873_S165_L006_R2_001.fastq.gz
109162_S14_L005_R1_001.fastq.gz   ABO_S31_L005_R1_001.fastq.gz           HBS_123019a_S207_L006_R1_001.fastq.gz
109162_S14_L005_R2_001.fastq.gz   ABO_S31_L005_R2_001.fastq.gz           HBS_123019a_S207_L006_R2_001.fastq.gz
118553_S19_L005_R1_001.fastq.gz   ACO_S32_L005_R1_001.fastq.gz           HBS_123019b_S208_L006_R1_001.fastq.gz
118553_S19_L005_R2_001.fastq.gz   ACO_S32_L005_R2_001.fastq.gz           HBS_123019b_S208_L006_R2_001.fastq.gz
118561_S23_L005_R1_001.fastq.gz   ACW_S33_L005_R1_001.fastq.gz           JL_S2_L005_R1_001.fastq.gz
118561_S23_L005_R2_001.fastq.gz   ACW_S33_L005_R2_001.fastq.gz           JL_S2_L005_R2_001.fastq.gz
120482_S16_L005_R1_001.fastq.gz   AIM_S34_L005_R1_001.fastq.gz           KMN_S42_L005_R1_001.fastq.gz
120482_S16_L005_R2_001.fastq.gz   AIM_S34_L005_R2_001.fastq.gz           KMN_S42_L005_R2_001.fastq.gz
120483_S15_L005_R1_001.fastq.gz   BHP_S35_L005_R1_001.fastq.gz           lynch_S48_L005_R1_001.fastq.gz
120483_S15_L005_R2_001.fastq.gz   BHP_S35_L005_R2_001.fastq.gz           lynch_S48_L005_R2_001.fastq.gz
15309M1_S28_L005_R1_001.fastq.gz  BJV_S36_L005_R1_001.fastq.gz           NOP_S43_L005_R1_001.fastq.gz
15309M1_S28_L005_R2_001.fastq.gz  BJV_S36_L005_R2_001.fastq.gz           NOP_S43_L005_R2_001.fastq.gz
15362M1_S26_L005_R1_001.fastq.gz  BNW_S37_L005_R1_001.fastq.gz           PN600_S6_L005_R1_001.fastq.gz
15362M1_S26_L005_R2_001.fastq.gz  BNW_S37_L005_R2_001.fastq.gz           PN600_S6_L005_R2_001.fastq.gz
15415M1_S24_L005_R1_001.fastq.gz  CHI_S38_L005_R1_001.fastq.gz           Pnel1_S1_L005_R1_001.fastq.gz
15415M1_S24_L005_R2_001.fastq.gz  CHI_S38_L005_R2_001.fastq.gz           Pnel1_S1_L005_R2_001.fastq.gz
18268M1_S25_L005_R1_001.fastq.gz  CI_S39_L005_R1_001.fastq.gz            rendon_S49_L005_R1_001.fastq.gz
18268M1_S25_L005_R2_001.fastq.gz  CI_S39_L005_R2_001.fastq.gz            rendon_S49_L005_R2_001.fastq.gz
18310M1_S29_L005_R1_001.fastq.gz  CIX_S40_L005_R1_001.fastq.gz           UK_S41_L005_R1_001.fastq.gz
18310M1_S29_L005_R2_001.fastq.gz  CIX_S40_L005_R2_001.fastq.gz           UK_S41_L005_R2_001.fastq.gz
18338M1_S27_L005_R1_001.fastq.gz  Clingermayer_S45_L005_R1_001.fastq.gz  VWX_S44_L005_R1_001.fastq.gz
18338M1_S27_L005_R2_001.fastq.gz  Clingermayer_S45_L005_R2_001.fastq.gz  VWX_S44_L005_R2_001.fastq.gz
27240_S21_L005_R1_001.fastq.gz    crouch_S46_L005_R1_001.fastq.gz        WB_DNA_17__1403_S50_L005_R1_001.fastq.gz
27240_S21_L005_R2_001.fastq.gz    crouch_S46_L005_R2_001.fastq.gz        WB_DNA_17__1403_S50_L005_R2_001.fastq.gz
27265_S22_L005_R1_001.fastq.gz    godat_S47_L005_R1_001.fastq.gz         WB_DNA_17__1404_S51_L005_R1_001.fastq.gz
27265_S22_L005_R2_001.fastq.gz    godat_S47_L005_R2_001.fastq.gz         WB_DNA_17__1404_S51_L005_R2_001.fastq.gz
27358_S18_L005_R1_001.fastq.gz    HBS108592_S4_L005_R1_001.fastq.gz      WB_DNA_17__1407_S52_L005_R1_001.fastq.gz
27358_S18_L005_R2_001.fastq.gz    HBS108592_S4_L005_R2_001.fastq.gz      WB_DNA_17__1407_S52_L005_R2_001.fastq.gz
27384_S20_L005_R1_001.fastq.gz    HBS108723_S7_L005_R1_001.fastq.gz      WB_DNA_17__1419_S53_L005_R1_001.fastq.gz
27384_S20_L005_R2_001.fastq.gz    HBS108723_S7_L005_R2_001.fastq.gz      WB_DNA_17__1419_S53_L005_R2_001.fastq.gz
27421_S10_L005_R1_001.fastq.gz    HBS117882_S5_L005_R1_001.fastq.gz
27421_S10_L005_R2_001.fastq.gz    HBS117882_S5_L005_R2_001.fastq.gz
peter@maven:~/data/last_Pseud/ReDo_demux/redo/gslserver.qb3.berkeley.edu/181110_150PE_HS4K2A/L567/Scott$ |
```

Making shells with loops

- Great for reproducibility and repeatability without issue

```
#!/bin/bash
. /u/local/Modules/default/init/modules.sh
module load bwa
for reads in *.reads.fastq; do
   echo $reads
   out=`echo $R1 | sed ‘s/.reads.fastq/_rawMap/’`
   echo $out
   bwa mem -M -t 8 /u/home/……/toy.ref.fasta \
   $reads > $out.sam
done
```

Making shells with loops

- Great for reproducibility and repeatability without issue

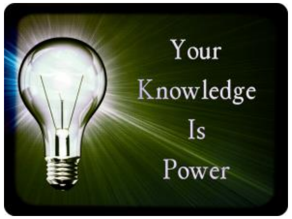
```
#!/bin/bash
. /u/local/Modules/default/init/modules.sh
module load bwa
module load samtoos
for reads in *.reads.fastq; do
   echo $reads
   out=`echo $R1 | sed ‘s/.reads.fastq/_rawMap/’`
   echo $out
   bwa mem –M –t 8 toy.ref.fasta \
   $reads | samtools view –Sb > $out.bam
done
```

History

- history | grep awk
- history >history_w1.txt

All UNIX commands from
the workshop in one file

Do Biologists have to become Programmers?



+

Google

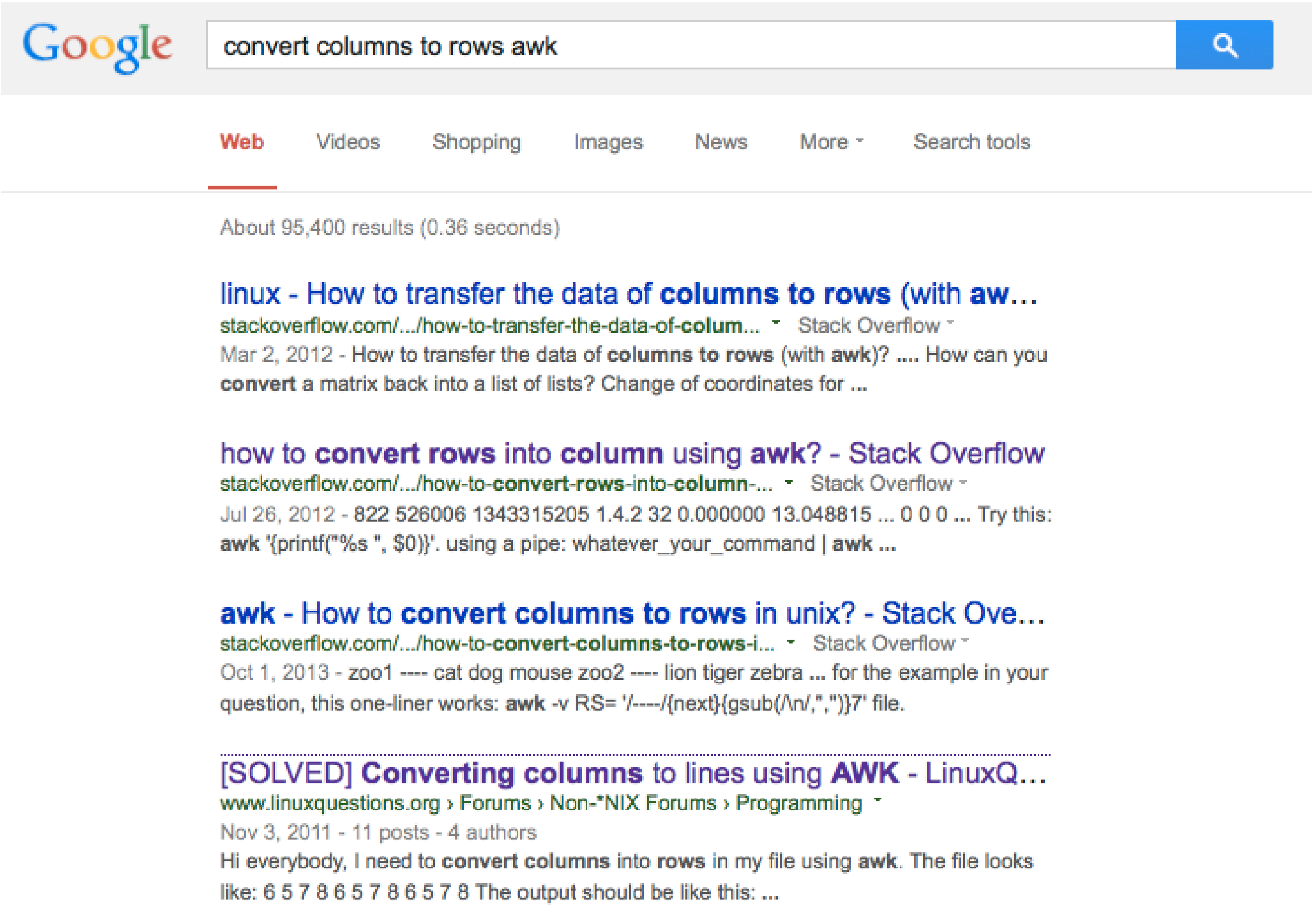
stackoverflow

SEQanswers
the next generation sequencing community

=

*provided in the class

*free and easy to use



convert columns to rows awk

About 95,400 results (0.36 seconds)

linux - How to transfer the data of **columns to rows** (with **aw**…
stackoverflow.com/.../how-to-transfer-the-data-of-colum... Stack Overflow
Mar 2, 2012 - How to transfer the data of **columns to rows** (with **awk**)? .... How can you **convert** a matrix back into a list of lists? Change of coordinates for ...

how to **convert rows** into **column** using **awk**? - Stack Overflow
stackoverflow.com/.../how-to-**convert-rows**-into-**column**-... Stack Overflow
Jul 26, 2012 - 822 526006 1343315205 1.4.2 32 0.000000 13.048815 ... 0 0 0 ... Try this: **awk** '{printf("%s ", $0)}'. using a pipe: whatever_your_command | **awk** ...

awk - How to **convert columns to rows** in unix? - Stack Ove…
stackoverflow.com/.../how-to-**convert-columns-to-rows**-i... Stack Overflow
Oct 1, 2013 - zoo1 ---- cat dog mouse zoo2 ---- lion tiger zebra ... for the example in your question, this one-liner works: **awk** -v RS= '/----/{next}{gsub(/\n/,",")}7' file.

[SOLVED] **Converting columns** to lines using **AWK** - LinuxQ…
www.linuxquestions.org › Forums › Non-*NIX Forums › Programming
Nov 3, 2011 - 11 posts - 4 authors
Hi everybody, I need to **convert columns** into **rows** in my file using **awk**. The file looks like: 6 5 7 8 6 5 7 8 6 5 7 8 The output should be like this: ...

http://www.linuxquestions.org/questions/programming-9/converting-columns-to-lines-using-awk-911677/

Thanks!

- Please take a few minutes to fill the survey