MOLB – 4485/5485
Computers in Biology

Week 5

Analysis of Human Genomic Variation through NGS

Vikram E. Chhatre & Nicolas Blouin
University of Wyoming

vchhatre@uwyo.edu
nblouin@uwyo.edu

November 2, 2016

Contents

1 Data for Today's Exercise

Once you log on to Mt. Moran, copy the data for today's exercise as follows:

```
[inbre@mtmoran ~]$ mkdir /project/inbre-train/inbre/LastName_Week5
[inbre@mtmoran ~]$ cd /project/inbre-train/LastName_Week5/
[inbre@mtmoran ~]$ cp -r /project/inbre-train/Week5Data/* .
[inbre@mtmoran ~]$ ls -othr
-rw-rw-r-- 1 vchhatre  16M Oct 30 09:51 chr7.vcf
-rw-rw-r-- 1 vchhatre  813 Oct 30 10:59 popdata
drwxrwsr-x 2 vchhatre 128K Oct 30 20:12 popfiles
```

2 Variant Call Format (VCF)

This is a standard for storing information on polymorphism in genes. It is a `[n x n]` matrix of individuals and genes. Columns refer to individuals, whereas rows refer to genes. Analogous to the `FASTQ` format, `VCF` also stores quality information. Files formatted to store variant calls always end in `.vcf` extension. For example: `input.vcf`. The `VCF` format is described in detail in the user manual available at http://samtools.github.io.

2.1 VCFTools

`VCFTools` is a program that can report information on and manipulate the `.vcf` files. A comprehensive manual on `VCFTools` usage is available at http://vcftools.sourceforge.net/man_latest.html. Before we can use this program, we must load it in memory as a module.

```
[inbre@mtmoran ~]$ module spider vcftools
[inbre@mtmoran ~]$ module load intel/16.3 perl vcftools
[inbre@mtmoran ~]$ vcftools

VCFtools (0.1.15)
 Adam Auton and Anthony Marcketta 2009
```

2.2 VCF File

We have provided you with an example `.vcf` file for today's exercise: `chr7.vcf`. As the name suggests, this file contains data from *Homo sapiens* chromosome 7. The data was collected as part of the `1000 Genomes` project whose aim is to understand human genomic variation across the globe. This data is from the final phase of the project and contains genotypes of 2500 individuals from 26 populations distributed globally genotyped at more than 4.7 million SNPs.

To make it more manageable, we have subset the original data to only include information on 1000 SNPs. You will be further subsetting this file to obtain data on one specific SNP that we are interested in. But for now, open the file in text editor and take a look at the contents. Then close

the file without saving.

```
[inbre@mtmoran ~]$ vim chr7.vcf

[vim ]$ :set nowrap
[vim ]$ :q
```

2.3 VCF Data Summary

`VCFTools` can provide a quick summary of the data. Try it out.

```
[inbre@mtmoran ~]$ vcftools --vcf chr7.vcf

VCFtools - 0.1.15
(C) Adam Auton and Anthony Marcketta 2009

Parameters as interpreted:
        --vcf chr7.vcf

After filtering, kept 2504 out of 2504 Individuals
After filtering, kept 1000 out of a possible 1000 Sites
Run Time = 0.00 seconds
```

When running without any options, `VCFTools` outputs summary statistics but does not make any changes to the file nor does it produce any output files.

2.4 SNP `rs776746`

For today's exercise, we are interested in only one SNP: `rs776746`. Every documented SNP from the human genome has a `rs` designation by which it is identified. Note that the number following `rs`, is not the physical position of this SNP on Chromosome 7, but simply a code. You can look up the physical position of this SNP in the VCF file. For example:

```
[inbre@mtmoran ~]$ grep 'rs776746' chr7.vcf | cut -f1-5
7       99270539        rs776746        C       T
```

Why did we pass the output of `grep` to `cut` via a `|`? It's because when `grep` matches a pattern, it prints the entire line to the screen. Our lines here have more than 2500 columns and we are only interested in the first few columns as displayed above.

2.5 Subset VCF

Let's go ahead and subset our file `chr7.vcf` for the SNP that we are interested in.

```
[inbre@mtmoran ~]$ vcftools --vcf chr7.vcf --snp rs776746 --recode --out mysnp

VCFtools - 0.1.15
(C) Adam Auton and Anthony Marcketta 2009

Parameters as interpreted:
        --vcf chr7.vcf
        --out mysnp
        --recode
        --snp rs776746

After filtering, kept 2504 out of 2504 Individuals
Outputting VCF file...
After filtering, kept 1 out of a possible 1000 Sites
Run Time = 0.00 seconds
```

In the above command, we extracted data for SNP `rs776746` from the file `chr7.vcf` and provided a prefix `mysnp` for the output file. Check the output:

```
[inbre@mtmoran ~]$ ls -othr
-rw-rw-r-- 1 vchhatre  16M Oct 29 16:21 chr7.vcf
-rw-rw-r-- 1 vchhatre  36K Oct 29 16:45 mysnp.recode.vcf
-rw-rw-r-- 1 vchhatre  296 Oct 29 16:45 mysnp.log

[inbre@mtmoran ~]$ mv mysnp.recode.vcf mysnp.vcf
```

The information that was printed to the screen was also populated inside the log file `mysnp.log`. Your main output file is `mysnp.recode.vcf`, but we changed its name to `mysnp.vcf` for convenience.

3 Gene Frequency

As we discussed earlier, allele frequency is a measure of the relative proportion of the two forms of a gene in a population. If a given SNP is not polymorphic, it will have only one form. Frequency of that form will be $f = 1.00$ or 100%. However, if it is polymorphic, the resulting gene forms (alleles) may either be present at equal proportion ($f(p) = f(q) = 0.5$) or at variable proportions such as $f(p) = 0.25$ and $f(q) = 0.75$. No matter their relative proportion, the frequencies of the two alleles will always add up to 1 in any population sample.

3.1 Frequency of rs776746 Alleles

We can use `VCFTools` to quickly estimate frequencies of the two alleles (`C/T`). But remember that allele frequency is a measure of population(s). During today's exercise, we want to calculate these frequencies in each of the 26 human populations that we have data for. However, the `vcf` files does not contain any information on which individuals belong to which populations.

Instead the population classification information is present in the folder `popfiles`:

```
[inbre@mtmoran ~]$ ls -l popfiles/
-rw-rw-r-- 1 vchhatre  768 Oct 29 14:46 ACB.pop
-rw-rw-r-- 1 vchhatre  488 Oct 29 14:46 ASW.pop
-rw-rw-r-- 1 vchhatre  688 Oct 29 14:46 BEB.pop
-rw-rw-r-- 1 vchhatre  744 Oct 29 14:46 CDX.pop
-rw-rw-r-- 1 vchhatre  792 Oct 29 14:46 CEU.pop
-rw-rw-r-- 1 vchhatre  824 Oct 29 14:46 CHB.pop
-rw-rw-r-- 1 vchhatre  840 Oct 29 14:46 CHS.pop
-rw-rw-r-- 1 vchhatre  752 Oct 29 14:46 CLM.pop
-rw-rw-r-- 1 vchhatre  792 Oct 29 14:46 ESN.pop
-rw-rw-r-- 1 vchhatre  792 Oct 29 14:46 FIN.pop
-rw-rw-r-- 1 vchhatre  728 Oct 29 14:46 GBR.pop
-rw-rw-r-- 1 vchhatre  824 Oct 29 14:46 GIH.pop
-rw-rw-r-- 1 vchhatre  904 Oct 29 14:46 GWD.pop
-rw-rw-r-- 1 vchhatre  856 Oct 29 14:46 IBS.pop
-rw-rw-r-- 1 vchhatre  816 Oct 29 14:46 ITU.pop
-rw-rw-r-- 1 vchhatre  832 Oct 29 14:46 JPT.pop
-rw-rw-r-- 1 vchhatre  792 Oct 29 14:46 KHV.pop
-rw-rw-r-- 1 vchhatre  792 Oct 29 14:46 LWK.pop
-rw-rw-r-- 1 vchhatre  680 Oct 29 14:46 MSL.pop
-rw-rw-r-- 1 vchhatre  512 Oct 29 14:46 MXL.pop
-rw-rw-r-- 1 vchhatre  680 Oct 29 14:46 PEL.pop
-rw-rw-r-- 1 vchhatre  768 Oct 29 14:46 PJL.pop
-rw-rw-r-- 1 vchhatre  832 Oct 29 14:46 PUR.pop
-rw-rw-r-- 1 vchhatre  816 Oct 29 14:46 STU.pop
-rw-rw-r-- 1 vchhatre  856 Oct 29 14:46 TSI.pop
-rw-rw-r-- 1 vchhatre  864 Oct 29 14:46 YRI.pop
```

Each of these popfiles contains a list of individuals that belong to that specific population. Try using the linux command `cat` on some of the files to see what's inside. Next, we will be writing a simple shell script to automate the estimation of allele frequencies using these population files and `VCFTools`. This script consumes very little computational resources, so we will not run it using `sbatch`. Thus you can exclude the scheduler commands that begin with `#SBATCH`.

```
[inbre@mtmoran ~]$ vim makefreq.sh

#!/bin/bash

## Make folder to store frequency output
mkdir freq

## Navigate to the folder 'popfiles/'
cd popfiles/

## For loop
for p in *.pop
do
  vcftools --vcf ../mysnp.vcf --keep $p --freq --out ../freq/$p
done
```

This script demonstrates a simple `for loop` which iterates over popfiles shown above and executes vcftools (to estimate allele frequencies) for only the individuals in the population file currently under iteration. Go ahead and execute it. Then look at the results.

```
[inbre@mtmoran ~]$ bash makefreq.sh
[inbre@mtmoran ~]$ pwd
/project/inbre-train/vchhatre/Chhatre_Week5/
[inbre@mtmoran ~]$ cd freq
[inbre@mtmoran ~]$ ls -l

-rw-rw-r-- 1 vchhatre   71 Oct 29 14:55 ACB.pop.frq
-rw-rw-r-- 1 vchhatre  350 Oct 29 14:55 ACB.pop.log
-rw-rw-r-- 1 vchhatre   79 Oct 29 14:55 ASW.pop.frq
-rw-rw-r-- 1 vchhatre  350 Oct 29 14:55 ASW.pop.log
```

Above, we have truncated the output of the `ls` command after the first four lines. You should be seeing 52 files altogether, of which half are freq files and the other half are log files from `VCFTools`. Let's look inside one of the population allele frequency files.

```
[inbre@mtmoran ~]$ cat ACB.pop.frq
CHROM        POS        N_ALLELES        N_CHR        {ALLELE:FREQ}
7        99270539        2        192        C:0.25        T:0.75
```

The output is self explanatory. The SNP we are studying (`rs977946`) is located on Chromosome 7 at base position 99270539. It has two alleles. In population ACB, there are 192 chromosomes altogether (2 per individual @ 96 individuals). Finally, frequencies of both alleles are provided. Notice that the frequencies add up to 1.0 as per expectation.

In order to visualize these allele frequencies for all 26 populations, we first need to combine infor-

mation from all 26 output files. Simplest way would be to use `cat command`.

```
[inbre@mtmoran ~]$ cat *.pop.frq > all.freq
[inbre@mtmoran ~]$ head all.freq -n 6
CHROM        POS        N_ALLELES        N_CHR        {ALLELE:FREQ}
7        99270539        2        192        C:0.25                T:0.75
CHROM        POS        N_ALLELES        N_CHR        {ALLELE:FREQ}
7        99270539        2        122        C:0.311475        T:0.688525
CHROM        POS        N_ALLELES        N_CHR        {ALLELE:FREQ}
7        99270539        2        172        C:0.633721        T:0.366279
```

3.2 Formatting the Frequency File

For our downstream visualization analysis, we need to get this file into correct format. You may have noticed that each allele frequency estimate has a header line. We do not need this redundant information for downstream analysis. Only one instance of the header line should be sufficient. We can selectively extract only those lines that we care about from this file. Let's grab these lines.

```
[inbre@mtmoran ~]$ sed -n '1p;0~2p' all.freq > master.freq
[inbre@mtmoran ~]$ head master.freq -n 5
CHROM        POS        N_ALLELES        N_CHR        {ALLELE:FREQ}
7        99270539        2        192        C:0.25                T:0.75
7        99270539        2        122        C:0.311475        T:0.688525
7        99270539        2        172        C:0.633721        T:0.366279
7        99270539        2        186        C:0.688172        T:0.311828
```

The `sed` command above extracts line 1 (i.e. the first header line), and then every 2nd line from `all.freq` and outputs them to `master.freq`. The output contains 27 lines (header + 1 line per population). We do not have population names in this file, but their order is alphabetical. So we should still be able to discern what frequency belongs to which population during the analysis.

We need to make two more changes to `master.freq`. Both require opening it in `vim`.

1. This file has 6 columns, but the column header has only 5 names. Delete the `ALLELE:FREQ` and replace it with `Allele1` tab `Allele2`. You will need to open the file in vim and then activate `INSERT` mode to make this change, i.e. press `i`
2. Delete the allele names `C:` and `T:` as follows. This should be done using `COMMAND` mode, i.e. press `ESC`.

```
[inbre@mtmoran ~]$ vim master.freq
[vim ~]$ :%s/C://g
[vim ~]$ :%s/T://g
[vim ~]$ :wq
```

Then save, close and check `head` of the file to make sure it looks right.

```
[inbre@mtmoran ~]$ head master.freq -n 5
CHROM        POS        N_ALLELES        N_CHR        Allele1        Allele2
7        99270539        2        192        0.25                0.75
7        99270539        2        122        0.311475        0.688525
7        99270539        2        172        0.633721        0.366279
7        99270539        2        186        0.688172        0.311828
```

3.3 `popdata` File

At the base of today's working directory (`/project/inbre-train/inbre/inbre_Week5/`), there is a file called `popdata`. It contains alphabetical list of population names whose lines correspond to those in allele frequency file. Check to make sure the file exists. This file also contains information on latitudes (i.e. distance from equator) for each of the 26 human populations we are studying. The populations are grouped into 5 super populations.

```
[inbre@mtmoran ~]$ pwd
/project/inbre-train/inbre/inbre_Week5/freq

[inbre@mtmoran ~]$ cd .. && ls -l
-rw-rw-r-- 1 vchhatre  813 Oct 30 10:59 popdata
```

3.4 Join Files

For further analysis, we need to merge the files, `popdata` and master.freq together. The simplest tool to accomplish this is `paste`.

```
[inbre@mtmoran ~]$ paste popdata freq/master.freq > freq.df
[inbre@mtmoran ~]$ head freq.df
```

3.5 Download File to Local Workstation

Recall that in order to copy files from server to local workstation, you will first need to **open a new terminal** window. Notice the different linux prompts above and below:

```
[train@localhost ~]$ mkdir /home/train/Week5
[train@localhost ~]$ scp user@server:/path/to/freq.df ~/Week5/
```

4 Working with R

R (http://www.r-project.org) is an open source, statistical data analysis language. Our goal today is to become familiar with the basic functionality included in R and then plot a figure of gene frequency differences among human populations with respect to their distance from equator.

4.1 Launch R Session

R can run both within the terminal shell or as a standalone GUI application (e.g. R-Studio). Today we will be using it via a terminal. Launch a new R session as follows:

```
[train@localhost ~]$ cd /home/train/Week5/
[train@localhost ~]$ R

R version 3.3.1 (2016-06-21) -- "Bug in Your Hair"
Copyright (C) 2016 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin13.4.0 (64-bit)

>
```

Notice that the R prompt is `>` as opposed to Linux where the prompt is `$`.

4.2 R as Calculator

R can be used a calculator. Try the following exercises:

```
> 2223482 + 98979723

> 55/279

> 1e-3

> 10^-20

> 4.58*2000
```

4.3 Importing Data into R

R has specific functions for reading text files as data frames. It can import a variety of text files (comma or tab separated). We will now import the gene frequency data that we created earlier (`freq.df`). Note that our data is in tab delimited format.

```
> getwd()
/home/train/Week5
> options(width=150)

> freq <- read.table('freq.df', header=TRUE)

> head(freq)
> tail(freq)
```

Notice that R formats the data so that's all columns are properly spaced making it easy to read. First checked our current location in the directory tree. Then we set the display width to 150 pts. Finally we imported our data and stored it into the variable `freq`. Now try the following functions:

```
> names(freq)

> dim(freq)
```

The above two commands tell you the names of columns and the dimensions of the data frame. Do the dimensions match your expectations? This can be a sanity check for the files you read in. If something is off, you will be able to notice it.

4.4 Accessing Rows and Columns

R keeps track of the order of each row and column in data frame by assigning them serial numbers. Thus you can access specific rows and columns of the data frames very quickly and easily. See the following example:

```
> freq[1:5,]

> freq[,1:5]

> freq[1:5,1:5]
```

As these examples show, you can access specific columns and rows in a data frame. In the first example, we asked R to print rows 1 through 5 to the screen. We indicated rows by printing a comma at the end of our argument. Likewise, columns can be specified by prefixing a comma to the argument. In the last example, we limited the output to the first 5 rows and 5 columns by providing both arguments separated by a comma. **Try out more examples on your own**.

4.5 Getting Summary Statistics

Suppose you want to calculate basic summary statistics on columns with numerical data. In our data frame, we have three numerical columns: (1) Population location latitudes, (2) Frequency of

Allele C and (3) Frequency of Allele T. Let's work on these columns. While doing so, let's learn another way to access columns in R, by using their names.

```
> range(freq$dist)
> summary(freq$Allele1)
> summary(freq$N_CHR)
```

We tested two R functions above. First function `range` does exactly what its name suggests. It finds the range over which values of a numerical variable are distributed. In other words, it reports the minimum and maximum values. Thus, by looking at the output you can tell that our data comes from human populations that range 70 degrees in latitudinal distribution.

Second function, `summary` indeed summarizes the numerical data – in this case, frequency of allele C. It shows `mean`, `median`, and the quantile distribution of the values. Notice that frequency of the allele varies a lot. It goes from 0.1 all the way to 0.9.

4.6 Histograms

Another way of looking at the distribution of values for a variable is to plot a histogram. R makes it very easy to quickly put together a histogram. Try following commands:

```
> hist(freq$Allele1)
> hist(freq$Allele1, breaks=20)
```

4.7 Scatterplots

When you want to compare the relationships of two variables with each other, a scatterplot comes handy. How would you make a scatterplot in R? Below are a series of R commands each building up the plot we desire, bit by bit. Type each command in increments and observe the resulting plot before typing the next command.

```
> plot(freq$Allele1, freq$dist)
```

```
> plot(freq$Allele1, freq$dist, ylab='Latitude', xlab='Frequency of Allele C',
  pch=16)
```

```
> plot(freq$Allele1, freq$dist, ylab='Latitude', xlab='Frequency of Allele C',
  pch=16, cex=0.8, col=freq$superpop, xlim=c(0,1))
```

```
> legend('topleft', c('African', 'Admixed American', 'East Asian', 'European',
  'South Asian'), cex=0.7, col=c(1:5), pch=16, inset=0.02)
```

```
> title(main='Latitudinal Variation in Allele Frequency at Locus rs776746',
  cex.main=0.8)
```

4.8 Saving Plots and Writing Scripts

Often, it will save you time if you wrote your plot code inside an R script. That way you do not have to type all the code again and again. Let's transfer our code into an R script file. Open a new text file `snp_plot.R`.

```
[train@localhost ~]$ vim snp_plot.R
```

Type the script that appears below in your newly created vim file `snp_plot.R`. Do not copy/paste; it won't work and you will spend more time troubleshooting the problems than it will take you to type it out. Once you are done, save and close the file with `:wq`.

Note: This script has 6 lines of code. Below the text is wrapped for 3 lines. You must type that code on one contiguous line each.

```
freq <- read.table('freq.df', header=T)

pdf('rs776746.pdf', width=8, height=8)

plot(freq$Allele1, freq$dist, ylab='Latitude', xlab='Frequency of Allele C',
  pch=16, cex=0.8, col=freq$superpop, xlim=c(0,1))

legend('topleft', c('African', 'Admixed American', 'East Asian', 'European',
  'South Asian'), cex=0.7, col=c(1:5), pch=16, inset=0.02)

title(main='Latitudinal Variation in Allele Frequency at Locus rs776746',
  cex.main=0.8)

dev.off()
```

We first import the original data frame into R and store it in variable `freq`. Then we open a new PDF device and specify it's dimensions. The `plot()` command tells R to use the two variables, `freq$Allele1` and `freq$dist` to make the scatterplot. Rest of the options are geared towards placing labels on the axes (`xlab, ylab` etc.), determining the limits of axes (`xlim`), plotting symbol type (`pch`) and size (`cex`), and finally we color the symbols using their classification into one of the 5 super populations (based on continents).

Command `legend()` prints a legend that describes the population groupings. Here, `'topleft'` designates the position of the legend, `'inset'` calls for it to be inset by a certain value, and labels are within quotes and separated by a comma. Finally, the `title()` command prints a title above the plot. You can adjust the size of this title label with `cex.main` option. Once you are done typing and saving the script, execute it as follows.

```
[train@localhost ~]$ R
> source('snp_plot.R')
```

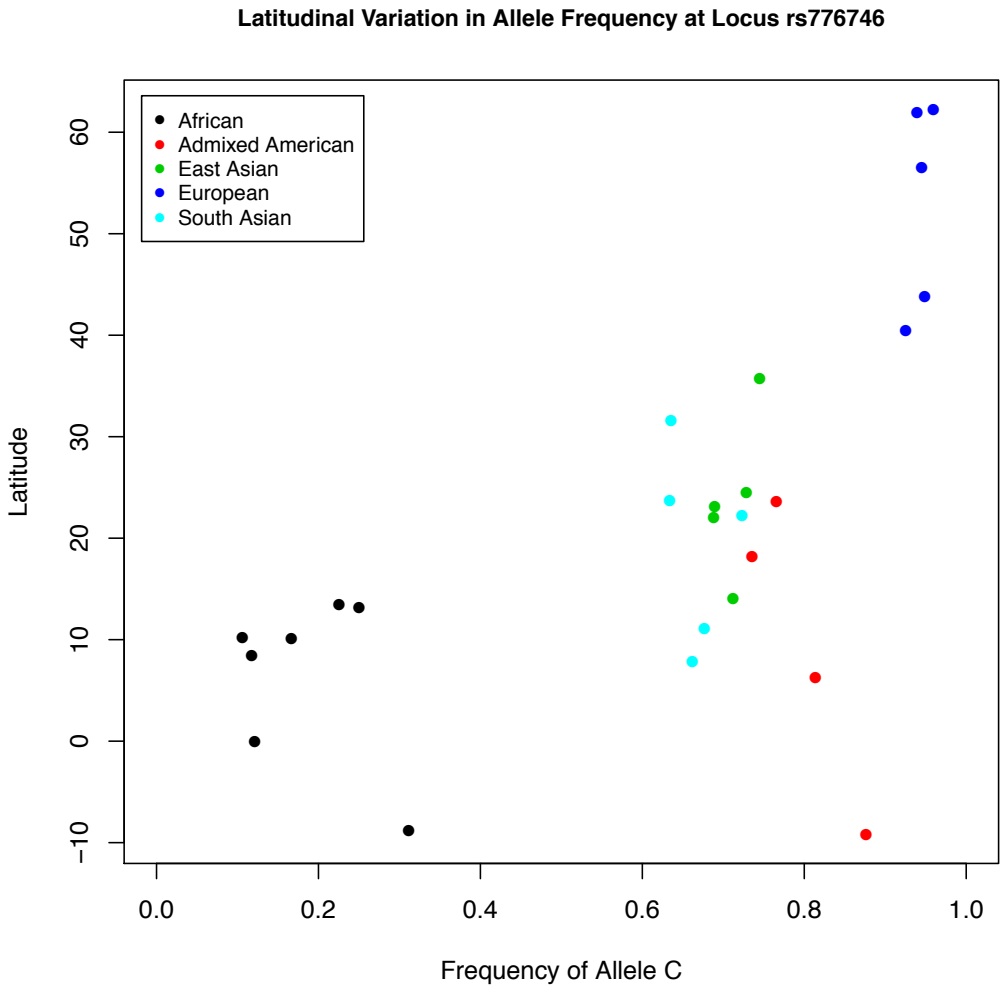

Figure 1: Frequency of allele C at SNP *CYP3A5*3* (rs776746) as a function of distance from equator in 26 human populations. Increase in the frequency of this allele reduces Sodium retention in humans.

5 Assignment: Save Activities Log

5.1 On Your Workstation (Virtualbox)

```
[train@localhost ~]$ cd /home/train/Week5/
[train@localhost ~]$ scp *.R *.pdf user@mtmoran.uwyo.edu:/path/to/week5/
```

5.2 On Mt. Moran

```
[inbre@mtmoran ~]$ cd /project/inbre-train/inbre/LastName_Week5/
[inbre@mtmoran ~]$ history > LastName_Week5_History.sh
```