# Unix 102 Part 1 - awk is the Unix tool for delimited files

Concept: awk combines many of the features of other Unix tools, and is more flexible. As a result, it is also more complex; but an investment in learning it quickly pays off. One valuable feature is the ability to compare numeric values and perform calculations.

**Before we begin**

1. Download the file <https://research.nhgri.nih.gov/Training/unix/downloads/UnixClass_120516.tar.gz>
2. Double-click on the file to uncompress it. This will create a directory (folder) called *UnixClass*
3. Drag the *UnixClass* directory to the Desktop folder on your Mac.

|  |  |
| --- | --- |
| Awk searches like grep | |
| grep Ovar hgmd\_brca1.txt | records matching pattern Ovar |
| grep Ovar hgmd\_brca1.txt | wc -l | pipe to word count (lines) |
| grep -i Ovar hgmd\_brca1.txt | wc -l | case insensitive grep |
| awk '/Ovar/' hgmd\_brca1.txt | wc -l | records matching pattern Ovar; single quotes not always needed, but good to get in the habit |
| awk '/[oO]var/' hgmd\_brca1.txt | wc -l | case insensitive awk not so easy |
| awk '/(2009)/' hgmd\_brca1.txt | less | quotes are needed here, to stop shell from interpreting parens |
| Awk extracts columns like cut | |
| cut -f1,3,4 hgmd\_brca1.txt | head | extract columns 1, 3 and 4 |
| awk '{print $1, $3, $4}' hgmd\_brca1.txt | head | extract columns 1, 3 and 4; note curly brackets around action (default scope is all lines) |
| awk '{print $4, $1, $3}' hgmd\_brca1.txt | head | can reorder columns, unlike cut |
| awk '{print $1, $8}' hgmd\_brca1.txt |head | default delimiter is "whitespace", any combination of spaces and tabs; this is also how sort operates |
| awk -F"\t" '{print $1, $8}' hgmd\_brca1.txt  | head | Can specify input delimiter with -F option; '\t' stands for a tab. Note that output delimiter is still a space. |
| cut -f1,8 hgmd\_brca1.txt | head | Cut is simpler for this example, as the default input delimiter is tab |
| Combine selection of lines (scope) and selection of columns (action) | |
| awk '/X/' hgmd\_brca1.txt | head | all lines containing X (default action is to print whole line) |
| awk '/X/{print $4,$1}' hgmd\_brca1.txt | less | cols 4 and 1 for lines containing X |
| awk '$4~/X/{print $4,$1}' hgmd\_brca1.txt | less | cols 4 and 1 for lines where col 4 contains X |
| awk '$4!~/X/{print $4,$1}' hgmd\_brca1.txt | head | cols 4 and 1 for lines where col 4 does not contain X |
| awk '!/X/{print $4}' hgmd\_brca1.txt | head | col4 for lines that do not contain X |
| For each TF, count binding sites where score is greater than 500 | |
| head -1000 wgEncodeRegTfbs.txt > tfbs.bed | Use a portion of file, to make examples faster and clearer |
| awk '$5>500' tfbs.bed | head | records where score > 500 |
| awk '$5>500{print $4}' tfbs.bed | less | ...extract TF name |
| awk '$5>500{print $4}' tfbs.bed | sort | uniq ‑c | ...and count binding sites for each TF |
| Combining criteria (&& and ||) | |
| awk '$4=="JunD" && $5 > 500' tfbs.bed | find sites with TF name **equal to** (==) "JunD" **AND** score > 500. Use double quotes for literal strings; not needed for numbers. |
| awk '$4=="SIX5" || $4=="STAT2"' tfbs.bed | sites for TF name "SIX5" **OR** "STAT2" |
| awk '($4=="SIX5" || $4=="STAT2") && $5>300' tfbs.bed | "SIX5" or "STAT2" sites with scores > 300; use parentheses for grouping |
| awk '$5<35' tfbs.bed | Find low-scoring TFBS |
| awk '$5<35 && $4!="SREBP1"' tfbs.bed | …but not those for SREBP1 |
| awk '$5<35 && !($4=="SREBP1" || $4=="SREBP2")' tfbs.bed | …nor SREBP2 |
| Find sites in a region of interest | |
| awk '$1=="chr16" && $2 > 29000000  && $3 < 30000000' tfbs.bed | find sites **contained** in a region |
| awk '$1=="chr16" && $3 > 29000000  && $2 < 30000000' tfbs.bed | find sites **overlapping** a region |
| Create or replace columns | |
| awk '{print $3-$2, $0}' tfbs.bed | head | Awk can calculate! Add new length column at beginning; $0 stands for whole line. |
| awk '{$7=$3-$2; print $0}' tfbs.bed | head | add new length column at end; single = assigns a value. Note there are two commands, separated by a semicolon. What happens if you leave out "print $0"? |
| awk '{$6=$3-$2; print}' tfbs.bed | head | can replace an existing column. "print" by itself prints whole line. Note that in last two examples all output fields are separated by spaces. |

# 

# Hands-On Exercises #1

Answer the following exercises using the file tfbs.bed. You will likely use sort and wc as well as awk.

1. How many binding sites are there for transcription factor SREBP1?
2. On how many different chromosomes is there an NFKB binding site with score greater than 300?
3. What is the longest binding site in this file? The length of each binding site is the distance between its start and stop coordinates.
4. What are the names of all the transcription factors with binding sites overlapping the 1Mb interval chr3:57,000,000-58,000,000?

# Unix 102 Part 2: More awk-wardness

|  |  |
| --- | --- |
| Change output delimiter | |
| awk '{print $3-$2, $0}' tfbs.bed | head | Saw this earlier. Note that first field is separated from others by a space. |
| awk -v OFS="\t" '{print $3-$2, $0}' tfbs.bed | head | If want a tab-delimited file, set OFS--the **output field separator** to "\t", shorthand for a tab character |
| awk '{print $4, $1 ":" $2 "-" $3}' tfbs.bed | head | Create customized output |
| Getting the right input delimiter | |
| awk '{print $1, $8}' hgmd\_brca1.txt | head | awk divides a file at "whitespace", any combination of spaces and tabs; this is also how sort operates |
| awk '{print NF}' hgmd\_brca1.txt | sort -u | NF is the number of fields (columns) on a line. Normally, we expect the same number of columns on every line. This is an easy way to tell if breaking on whitespace will work; in this case, it doesn't. |
| awk -F"\t" '{print NF}' hgmd\_brca1.txt | sort -u | When we use a tab delimiter, we see every line has 8 columns. |
| Include/Exclude header line using NR (row number) | |
| awk -F"\t" '{print NR,$0}' hgmd\_brca1.txt  | head | NR adds line numbering |
| awk -F"\t" '$4~/X/{print $4,$6}' hgmd\_brca1.txt  | head | cols 4 and 6 for lines where col 4 contains X |
| awk -F"\t" 'NR==1 || $4~/X/ {print $4,$6}' hgmd\_brca1.txt | head | How to do something different with the first line, like retain header |
| cut -f6 hgmd\_brca1.txt | sort | uniq -c | How many mutations for each cancer type? |
| awk -F"\t" 'NR>1 {print $6}' hgmd\_brca1.txt | sort | uniq -c | How many mutations for each cancer type, excluding header line |
| Identify columns by number | |
| head -1 hgmd\_brca1.txt | How can we tell what each column contains without counting by hand? |
| head -1 hgmd\_brca1.txt | tr "\t" "\n" | tr translates every instance of a character to another; in this case, tabs to newlines. |
| head -1 hgmd\_brca1.txt | tr "\t" "\n" | awk '{print NR, $0}' | Number each of the column headers |
| Identify duplicate values | |
| cut -f4 hgmd\_brca1.txt | sort | uniq -c | less | can count for each value |
| cut -f4 hgmd\_brca1.txt | sort | uniq -c | awk '$1>1' | so ask awk to find ones where count is greater than one |
| cut -f4 hgmd\_brca1.txt | sort | uniq -c | awk '$1>1{print $2}' > dup.list.txt | redirect list of just the duplicated values to a file |
| grep -f dup.list.txt hgmd\_brca1.txt | cut -f4,5,6 | can use grep with -f option to read patterns from a file, and pull out lines of the original file that match any of these patterns |

# Hands-On Exercises #2

Answer the following exercises using the file refFlat.txt. Apply what you've learned about awk to answer these questions (you could answer some of these questions by manually inspecting the file--but make the computer do the work for you).

1. How many columns does the file have? Can we use whitespace as the delimiter?
2. Which columns have the start and end position of each transcript?
3. By convention, non-coding transcripts have cdsStart equal to cdsEnd. How many non‑coding transcripts are there?
4. How many coding genes have transcripts on more than one chromosome? Note that the genes are named by the gene symbol in column 1 of refFlat.txt, while the transcripts are named by the accession in column 2.
5. If you exclude transcripts on alternate haplotypes (e.g. chr6\_apd\_hap1, chr6\_cox\_hap2, etc.), how many coding genes have transcripts on more than one chromosome?
6. What is the distribution of the number of transcripts for each gene, i.e. how many genes have a single transcript, how many have two, etc.?

# Unix 102 Part 3 - Join two files on a common column

Concept: when you have two files with different, but related information in each, you can combine them into a single file, as long as they have one column in common (preferably a unique identifier).

|  |  |
| --- | --- |
| Join files on a common column | |
| less intensity.txt | Data from an experiment; no gene names or positions, just accessions |
| wc -l intensity.txt | Number of rows |
| less -S refFlat.txt | Information about transcripts |
| sort -k1,1 intensity.txt > intensity.by\_acc.txt | **Sort on common column** |
| cut -f1-6 refFlat.txt | sort -k2,2  > refFlat.by\_acc.txt | Select desired fields and **sort on common column** |
| join -1 2 -2 1 refFlat.by\_acc.txt intensity.by\_acc.txt > intensity\_with\_pos.txt | Join on common column (col 2 of first file, col 1 of second file) |
| less intensity\_with\_pos.txt | Only one copy of common column, and it now appears **first**. All other columns are retained, in order, separated by spaces. |
| wc -l intensity\_with\_pos.txt | always a good idea to check number of lines after join |
| Values seen in one file, not in the other | |
| join -1 2 -2 1 -v 2 refFlat.by\_acc.txt intensity.by\_acc.txt | With “-v 2” option, join reports lines that are **not** matched in the 2nd file |
| Use BBEdit to modify intensity.by\_acc.txt | Change NM\_000063.2 to NM\_000063 |
| join -1 2 -2 1 refFlat.by\_acc.txt intensity.by\_acc.txt > intensity\_with\_pos.txt | Regular join again, without -v |
| wc -l intensity\_with\_pos.txt | Can end up with more lines, as well as fewer |
| less intensity\_with\_pos.txt | When the value in the join column appears more than once, join outputs one row for each pairing |
| Can use join to find all matching values, or remove them | |
| sort -k4,4 hgmd\_brca1.txt > hgmd\_brca1.by\_aa.txt | Sort by join column |
| join -1 1 -2 4 dup.list.txt hgmd\_brca1.by\_aa.txt | Find matches |
| join -1 1 -2 4 -t $'\t' dup.list.txt hgmd\_brca1.by\_aa.txt | Retain tabs as separators (input and output); note that join doesn't treat '\t' as a special value, so we have to make the shell interpret it. |
| join -1 1 -2 4 -t $'\t' -v 2 dup.list.txt hgmd\_brca1.by\_aa.txt | wc -l | Keep only those that don't match |

# Hands-On Exercises #3

Answer the following exercises using the files refFlat.txt and intensity.txt.

1. Identify the transcripts (accession) that appear more than once (i.e. mapped to more than one location). Separate the RefSeq transcripts into two files: those that are mapped to only a single location, and those that are mapped to multiple locations.
2. a) Attach the intensity from the intensity.txt file to the corresponding RefSeq transcript and save the output to a file called refFlat\_with\_intensity.txt.

b) Where there is no intensity data for a given accession, set the intensity to zero and save the output to a file called refFlat\_wo\_intensity.txt.

c) Combine the two files together using the append operator “>>”.

cat refFlat\_with\_intensity.txt refFlat\_wo\_intensity.txt >> refFlat\_plus\_intensity.txt

# Unix 102 additional topics

|  |  |
| --- | --- |
| Identify the last column | |
| awk -F"\t" '{print $NF}' hgmd\_brca1.txt | head | Can use NF as a column number; print the last column |
| awk -F"\t" '{print $(NF-1)}' hgmd\_brca1.txt | head | Need parens here; print the next-to-last column |
| Calculate sum and mean (using END) | |
| awk '$4=="SREBP1"{s=s+$5;print $0,s}' tfbs.bed | You can assign values to variables that persist from one line to the next |
| awk '$4=="SREBP1"{s=s+$5} END{print s}' tfbs.bed | When calculating sums, often you just want the final answer; use the END scope for an action that happens after all lines are processed |
| awk '$4=="SREBP1"{s=s+$5;c=c+1} END{print s/c}' tfbs.bed | If you track the count as well as the score, you can calculate the mean. |
| Remove duplicates: keep only the first | |
| sort -k4,4 -k1,1 hgmd\_brca1.txt | less | sort so that the one you want to keep is first |
| sort -k4,4 -k1,1 hgmd\_brca1.txt | awk '$4!=prev{print; prev=$4}' > hgmd1.txt | print whole line each time the protein change is different from what it was on the last line |
| wc -l hgmd\* |  |

# 

# Extra Credit Hands-On Exercises

Answer the following exercises using the file refFlat.txt.

1. What is the average number of exons per transcript? What is the average number of exons for non-coding transcripts?
2. Generate a file that retains only the longest transcript (txStart to txEnd) for each gene.
3. A BED6-formatted file has six columns of interest: chrom, transcription\_start, transcription\_end, name, score, strand. Produce a BED6 file that contains 1000bp upstream of each transcript. The “name” should be “gene:prom:accession” (e.g., WASH7P:prom:NR\_024540*)* and the score for each promoter is 0*.* *More challenging variation:* take account of the orientation of the transcript: for those on '-' strand, "upstream" will be to the right of the end of the transcript (larger chromosome positions), rather than to the left of the start position.

# Creating tar archives

|  |  |
| --- | --- |
| Creating tar archives ("tarball") | |
| cd .. | Go to parent directory |
| tar -czf UnixClass.tar.gz UnixClass | typical invocation to create a gzipped tar file |
| tar -xzf UnixClass.tar.gz | typical invocation to extract a gzipped tar file |

# Accessing Remote Computers

Concepts: Unix commands work exactly the same on Linux servers and HPC clusters (like biowulf) as they do on your Mac. To access these computers over the network, we use ssh (secure shell) and scp (secure copy).

|  |  |
| --- | --- |
| Log in to a remote computer | |
| ssh *studXX*@biowulf.nih.gov | "studXX" is your username on the remote computer. If username is the same on local and remote, can leave out "username@" |
| Password: | Enter password (case sensitive) |
| pwd | All commands that we know work here, e.g. Where are we? |
| ls -l | What's here? |
| exit | Log out |
| Copy files to remote server | |
| scp UnixClass.tar.gz *studXX*@biowulf.nih.gov: | Note colon after hostname. |
| Password: | Asks for password on remote (won't mention this again, happens every time we connect) |
| ssh *studXX*@biowulf.nih.gov |  |
| ls | The file is here, in your home directory, because we didn't provide a destination directory name |
| tar -xzf UnixClass.tar.gz | Untar gzipped archive |
| ls -l | created a new directory |
| ls -l UnixClass | Familiar files |
| exit |  |
| Copy files from remote server | |
| scp *studXX*@biowulf.nih.gov:UnixClass/hgmd\_brca1.txt . | Note dot to indicate that the destination is the current directory; always have to provide a destination for scp |