Mon, Sep 28

Lecture 3A:
Working with text

Hamamelis sp.
© Matilda Adams/
Missouri Botanical Garden

Practical Bioinformatics (Biol 4220)
Instructor: Michael Landis
Email: michael.landis@wustl.edu

Lecture 3A outline

1. Text formatting
2. Text processing
3. Example pipelines

Text strings

Pipelines sharing input/output as text

Programs expect input in a variety of formats

Pipelines often require intermediate stages that reformat or filter output from one stage before passing it as input to the next stage

Example file formats

```
~$ cat format_1.txt
This file contains DNA sequences downloaded from GenBank
for multiple species (incl. human, mouse, cow) and multiple
genes (incl. CO2 and cytB). The first sequence, for the CO2
gene in Mus musculus, is interesting, in part, because it
nucleotide site positions 31, 239, and 594 are G, C, and T,
respectively, whereas CO2 for all other species described
in this document report nucleotides A, C, and T, at those
...
```

plain text
(.txt)

```
~$ cat format_2.txt
species,gene,sequence
Mus_musculus,CO2,ACGTCAGGGCATT...
Homo_sapiens,CO2,ACGTCACCGCATT...
Bos_taurus,CO2,ACGTCACTGCATCAT...
Mus_musculus,cytB,CGGCAAGATGCC...
Homo_sapiens,cytB,CTGCAAGTTGCC...
Bos_taurus,cytB,CAGCAGGATGCCTT...
...
```

comma-separated values
(.csv)

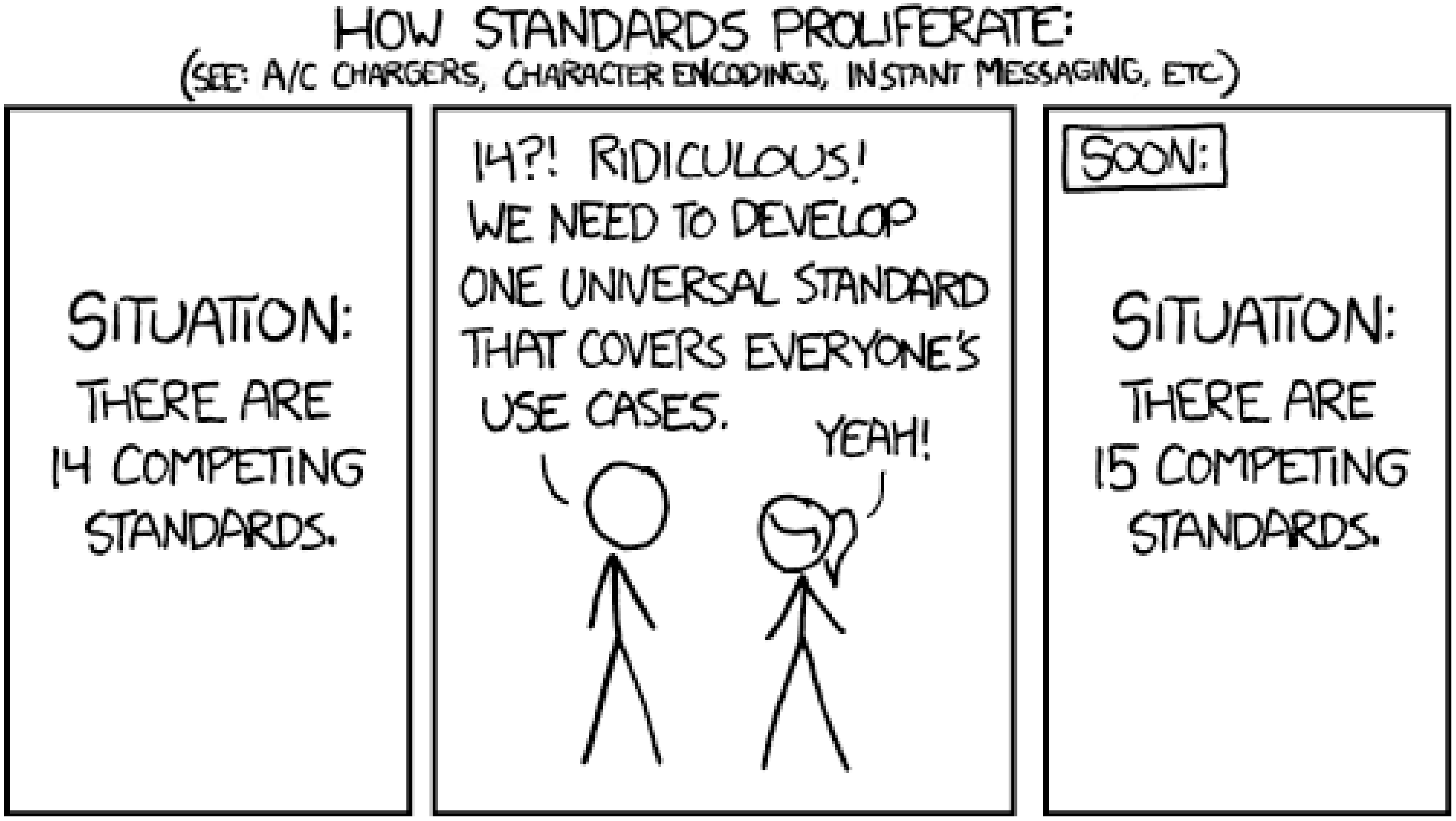
```
~$ cat format_4.txt

> Mus_musculus_CO2
ACGTCAGGGCATTTCATCGTGCGATC...
> Homo_sapiens_CO2
ACGTCACCGCATTTGCTCGTGCGATC...
> Bos_taurus_CO2
ACGTCACTGCATCATTTCGTGCGATC...
> Mus_musculus_cytB
CGGCAAGATGCCGATCTCGTGCGATC...
> Homo_sapiens_cytB
CTGCAAGTTGCCTGACTCGTGCGATC...
> Bos_taurus_cytB
CAGCAGGATGCCTTTCTCGTGCGATC...
...
```

FASTA format
(.fas or .fasta)

```
~$ cat format_4.txt
{
    'Mus_musculus' : {
        'genes': [
            'CO2': 'ACGTCAGGGCATT...
            'cytB : 'CGGCAAGATGCC...
        ]
    },
    'Homo_sapiens' : {
        'genes': [
            'CO2': 'ACGTCACCGCATT...
            'cytB : 'CTGCAAGTTGCC...
        ]
    },
    ...
```

Javascript Object Notation
(.json)

No formats are perfect for all use cases


HOW STANDARDS PROLIFERATE:
(SEE: A/C CHARGERS, CHARACTER ENCODINGS, INSTANT MESSAGING, ETC)
SITUATION: THERE ARE 14 COMPETING STANDARDS.
14?! RIDICULOUS! WE NEED TO DEVELOP ONE UNIVERSAL STANDARD THAT COVERS EVERYONE'S USE CASES.
YEAH!
SOON:
SITUATION: THERE ARE 15 COMPETING STANDARDS.

xkcd.com

.csv format

The ***comma-separated values*** format is flexible and easy to work with

- All rows have same number of columns
- Each line is a row
- Text between commas is a column

```
species,gene,sequence
Mus_musculus,CO2,ACGTCAGGGCATT...
Homo_sapiens,CO2,ACGTCACCGCATT...
Bos_taurus,CO2,ACGTCACTGCATCAT...
Mus_musculus,cytB,CGGCAAGATGCC...
Homo_sapiens,cytB,CTGCAAGTTGCC...
Bos_taurus,cytB,CAGCAGGATGCCTT...
...
```

.fasta format

FASTA is a popular format for molecular sequence data. Sequences are defined by two adjacent sets of rows:

- First row beginning with ">" provides sequence name
- Following rows report the sequence data (e.g. ACGT) for that named sequence, until the ">" row

```
> Mus_musculus_CO2
ACGTCAGGGCATTTCATCGTGCGATC
CGATCAACGCTCATGGCATTACTCAG
...
> Homo_sapiens_CO2
ACGTCACCGCATTTGCTCGTGCGATC
CTGTCAATGCTCATGCTATTACTCAG
...
> Bos_taurus_CO2
ACGTCACTGCATCATTTCGTGCGATC
CGGTCAGCGCTCATGCTACTACTCAG
...
```

.sam format

Sequence Alignment/Map format is a *tab-delimited* format used with for sequence alignment against a reference genome

- Header lines (optional) begin with "@"
- Alignment lines follow; each row contains 11 columns that identify each mapped read by its name, position, sequence identity, quality score, etc.

```
@HD VN:1.6 SO:coordinate @SQ SN:ref LN:45
@SQ SN:ref LN:45
r001   99 ref  7 30 8M2I4M1D3M = 37  39 TTAGATAAAGGATACTG *
r002    0 ref  9 30 3S6M1P1T4M *  0   0 AAAAGATAAGGATA    *
r003    0 ref  9 30 5S6M       *  0   0 GCCTAAGCTAA       * SA:Z:ref,29,-,6H5M,17,0;
r004    0 ref 16 30 6M14N5M    *  0   0 ATAGCTTCAGC       *
r003 2064 ref 29 17 6H5M       *  0   0 TAGGC             * SA:Z:ref,9,+,5S6M,30,1;
r001  147 ref 37 30 9M         =  7 -39 CAGCGGCAT         * NM:i:1
```

.vcf format

Variant Call Format is a tab-delimited format that reports genomic variants

- Header lines ("##") report file metadata
- Following rows reports each variant, and its chromosome, position, its variant type (e.g. SNP, microsat), and how the variant differs from a reference sequence

```
##fileformat=VCFv4.3
##fileDate=20090805
##source=myImputationProgramV3.1
##
##  ---> OMITTED LARGE PART OF HEADER FOR BREVITY <---
##
##FORMAT=<ID=DP,Number=1,Type=Integer,Description="Read Depth">
##FORMAT=<ID=HQ,Number=2,Type=Integer,Description="Haplotype Quality">
#CHROM POS      ID         REF   ALT    QUAL  FILTER   INFO                             FORMAT
20     14370    rs6054257  G     A      29    PASS    NS=3;DP=14;AF=0.5;DB;H2           GT:GQ:D
20     17330    .          T     A      3     q10     NS=3;DP=11;AF=0.017               GT:GQ:D
20     1110696  rs6040355  A     G,T    67    PASS    NS=2;DP=10;AF=0.333,0.667;AA=T;DB GT:GQ:D
20     1230237  .          T     .      47    PASS    NS=3;DP=13;AA=T                   GT:GQ:D
20     1234567  microsat1  GTC   G,GTCT 50    PASS    NS=3;DP=9;AA=G                    GT:GQ:D
```

Common forms of text processing

- ***sort*** the text (e.g.) alphabetically
- filter out ***duplicate*** data entries
- ***parse*** or ***tokenize*** text strings into fields by a delimiter
- ***join*** or ***paste*** multiple text files into a single file
- ***translate*** a set of text characters into a new set of characters (e.g. lowercase to uppercase)
- ***cut*** relevant text out of a data table
- find all lines and/or files that match a ***search pattern***

Text format determines how it that text is best processed

```
# search csv file for CO2
$ grep CO2 sequences.csv
Mus_musculus,CO2,ACGTCAGGGCATT...
Homo_sapiens,CO2,ACGTCACCGCATT...
Bos_taurus,CO2,ACGTCACTGCATCAT...
```

```
# search fasta file for CO2
$ grep CO2 sequences.fasta
> Mus_musculus_CO2
> Homo_sapiens_CO2
> Bos_taurus_CO2
```

Formats for filesystem organization

How you name and organize filesystems determines how they can be processed

```
$ tree
.
├── U3392125.fasta
├── U3392126.fasta
├── U3392127.fasta
└── U3392128.fasta
```

```
$ tree
.
├── species_1.gene_1.fasta
├── species_2.gene_1.fasta
├── species_1.gene_2.fasta
└── species_2.gene_2.fasta
```

```
$ tree
.
├── gene_1
│   ├── species_1.fasta
│   └── species_2.fasta
└── gene_2
    ├── species_1.fasta
    └── species_2.fasta
```

```
$ tree
.
├── species_1
│   ├── gene_1.fasta
│   └── gene_2.fasta
└── species_2
    ├── gene_1.fasta
    └── gene_2.fasta
```

Less organized
More organized

grep, file pattern searcher

Many options are available to modify the behavior of ***grep***.
Execute *man grep* for more examples. Some below.

```
$ cat
Viburnum_molle,ACAGTAGGTAGACACAGTA
Viburnum_acerifolium,ACGGTAGGTATACACAGTA
Viburnum_nudum,ACCGTAGATATACACAGTA
# ignore case
Viburnum_molle,ACAGTAGGTAGACACAGTA
$ grep -i MOLLE seq1.csv
# print row number with match
$ grep -n nudum seq1.csv
3:Viburnum_nudum,ACCGTAGATATACACAGTA
# find lines that contain AAT, CAT, GAT, or TAT
$ grep '[ACGT]AT' seq1.csv
Viburnum_acerifolium,ACGGTAGGTATACACAGTA
Viburnum_nudum,ACCGTAGATATACACAGTA
# find lines that contain AAT, BAT, ... ZAT
#                      or aAT, bAT, ... zAT
$ grep '[A-Za-z]AT' seq1.csv
Viburnum_acerifolium,ACGGTAGGTATACACAGTA
Viburnum_nudum,ACCGTAGATATACACAGTA
```

sort, sorts file by line

Sorts each line in a file alpanumerically;

delimit files (-t) to sort against specific fields (-k)

```
$ cat seq1.csv
Viburnum_molle,ACAGTAGGTAGACACAGTA
Viburnum_acerifolium,ACGGTAGGTATACACAGTA
Viburnum_nudum,ACCGTAGATATACACAGTA
$ cat seq2.csv
Viburnum_lantana,ACGGTAGGTATACGCAGTA
Viburnum_tinus,ACGGTAGGTCTACACTGTA
Viburnum_clemensiae,AGGGTCAGTCTACACTGTA
# sort two files
$ sort seq1.csv seq2.csv
Viburnum_acerifolium,ACGGTAGGTATACACAGTA
Viburnum_clemensiae,AGGGTCAGTCTACACTGTA
Viburnum_lantana,ACGGTAGGTATACGCAGTA
Viburnum_molle,ACAGTAGGTAGACACAGTA
Viburnum_nudum,ACCGTAGATATACACAGTA
Viburnum_tinus,ACGGTAGGTCTACACTGTA
# sort by the 2nd field with "," is delimiter
$ sort -t ',' -k2,2 seq1.csv
Viburnum_molle,ACAGTAGGTAGACACAGTA
Viburnum_nudum,ACCGTAGATATACACAGTA
Viburnum_acerifolium,ACGGTAGGTATACACAGTA
```

uniq, filters repeated lines

Filters out any line that is identical to the previous line, then prints filtered text to standard output

```
$ cat seq3.seq
Viburnum_molle,ACAGTAGGTAGACACAGTA
Viburnum_molle,ACAGTAGGTAGACACAGTA
Viburnum_acerifolium,ACGGTAGGTATACACAGTA
Viburnum_acerifolium,ACGGTAGGTATACACAGTA
Viburnum_nudum,ACCGTAGATATACACAGTA
Viburnum_molle,ACAGTAGGTAGACACAGTA
$ uniq seq3.csv
Viburnum_molle,ACAGTAGGTAGACACAGTA
Viburnum_acerifolium,ACGGTAGGTATACACAGTA
Viburnum_nudum,ACCGTAGATATACACAGTA
Viburnum_molle,ACAGTAGGTAGACACAGTA
```

tr, translate characters

Expects a text stream as input; translates all characters in the first pattern into the second pattern

```
# convert from DNA to RNA
$ echo "aGcttAcGCaTaC" | tr "t" "u" | tr "T" "U"
aGcuuAcGCaUaC
# needs input redirect to operate on file
$ cat seq.txt
aGcttAcGCaTaC
$ tr "t" "u" < seq.txt | tr "T" "U"
# change case, then convert DNA to RNA
$ echo "aGcttAcGCaTaC" | tr "[:lower:]" "[:upper:]" | tr "T" "U"
AGCUUACGCAUAC
# delete spaces
$ echo "AGC UUAC  G CAUAC" | tr -d " "
AGCUUACGCAUAC
# squeeze all repeated U's
$ echo "AUUUUGUAAAAC" | tr -s "U" "U"
AUGUAAAAC
```

rev, reverse text for each line

Prints characters for each line in reverse order (left-to-right) to standard output

```
# example csv for Darwin's finches
$ cat finch.csv
name,wingL,tarsusL
magnirostris,4.404200,3.038950
conirostris,4.349867,2.984200
# reverse text for each line
$ rev finch.csv
Lsusrat,Lgniw,eman
059830.3,002404.4,sirtsoringam
002489.2,768943.4,sirtsorinoc
```

cut, extract columns from file

Prints selected text to standard out. Text can be selected by a delimited field (-d) or by character position (-c)

```
# example csv for Darwin's finches
$ cat finch.csv
name,wingL,tarsusL
magnirostris,4.404200,3.038950
conirostris,4.349867,2.984200
# cut columns for name (1) and tarsus length (3)
$ cut -f1,3 -d "," finch.csv
name,tarsusL
magnirostris,3.038950
conirostris,2.984200
# extract the first five character columns
$ cut -c1-5 finch.csv
name,
magni
conir
```

paste, merge lines of files

Prints interleaved lines from each file to standard output; use "-d" to specify what character is used to join lines

```
# file 1 contains sequence names
$ cat seq_names.txt
> Viburnum_molle
> Viburnum_acerifolium
# file 2 contains sequence data
$ cat seq_data.txt
ACAGTAGGTAGACACAGTA
ACGGTAGGTATACACAGTA
# interleave sequence names and data
$ paste -d "\n" seq_names.txt seq_data.txt
> Viburnum_molle
ACAGTAGGTAGACACAGTA
> Viburnum_acerifolium
ACGGTAGGTATACACAGTA
```

join, merge two data tables

Joins lines of two files that share a common field, then writes joined text to standard output

```
# what do the two tables contain?
$ cat dat1.txt
index,name,size
1,dog,24
2,whale,523
$ cat dat2.txt
name,appetite
dog,102
whale,1405
# join two tables against shared "name" field
$ join -1 2 -2 1 -t , dat1.txt dat2.txt
name,index,size,appetite
dog,1,24,102
whale,2,523,1405
```

find, search file tree

Finds all filesystem objects that match search criteria, then prints the path to each file/directory to stdout

```
# list contents of directory
$ ls data
file1.txt  file2.txt  old_files
# find all contents of directory
$ find data
data
data/file2.txt
data/file1.txt
data/old_files
data/old_files/file3.txt
data/old_files/older_files
data/old_files/older_files/file1.txt
# find all contents that match pattern
$ find data -name "file1*"
data/file1.txt
data/old_files/older_files/file1.txt
# find all directories
$ find data -type d
data
data/old_files
data/old_files/older_files
```

Testing pipelines during design

Don't expect a pipeline will work correctly on your first design attempt

1. Create simplified versions of expected input so that problems can be easily spotted
2. If the pipeline modifies your filesystem, make a copy of the affected files
3. Add commands one-at-a-time to the pipeline

Worked pipeline example

How would you extract a sorted data table that lists the scientific name and adult body mass for all species in order Monotremata in tab-delimited format, then save that output to monotreme_mass.tsv?

```
$ cat data/mammal_data.csv
Order;Scientific_name;AdultBodyMass_g;Max_longevity_d
Rodentia;Eligmodontia typus;17.37;292
Rodentia;Microtus oregoni;20.35;456.25
Rodentia;Peromyscus gossypinus;27.68;471.45833335
Macroscelidea;Elephantulus myurus;59.51;401.5
Rodentia;Peromyscus boylii;23.9;547.5
Rodentia;Phodopus campbelli;27.06;653.95833335
Rodentia;Myodes gapperi;19.83;608.33333335
Eulipotyphla;Sorex palustris;13.07;547.5
Rodentia;Reithrodontomys humulis;8.25;817.90416665
```

Worked pipeline example

How would you extract a sorted data table that lists the scientific name and adult body mass for all species in order Monotremata in tab-delimited format, then save that output to monotreme_mass.tsv?

```
$ grep Monotremata mammal_data.csv
Monotremata;Tachyglossus aculeatus;4499.97;18158.75
Monotremata;Zaglossus bruijnii;7500;13176.5
Monotremata;Zaglossus attenboroughi;2500;no information
Monotremata;Zaglossus bartoni;6500;no information
Monotremata;Ornithorhynchus anatinus;1484.25;8139.5
```

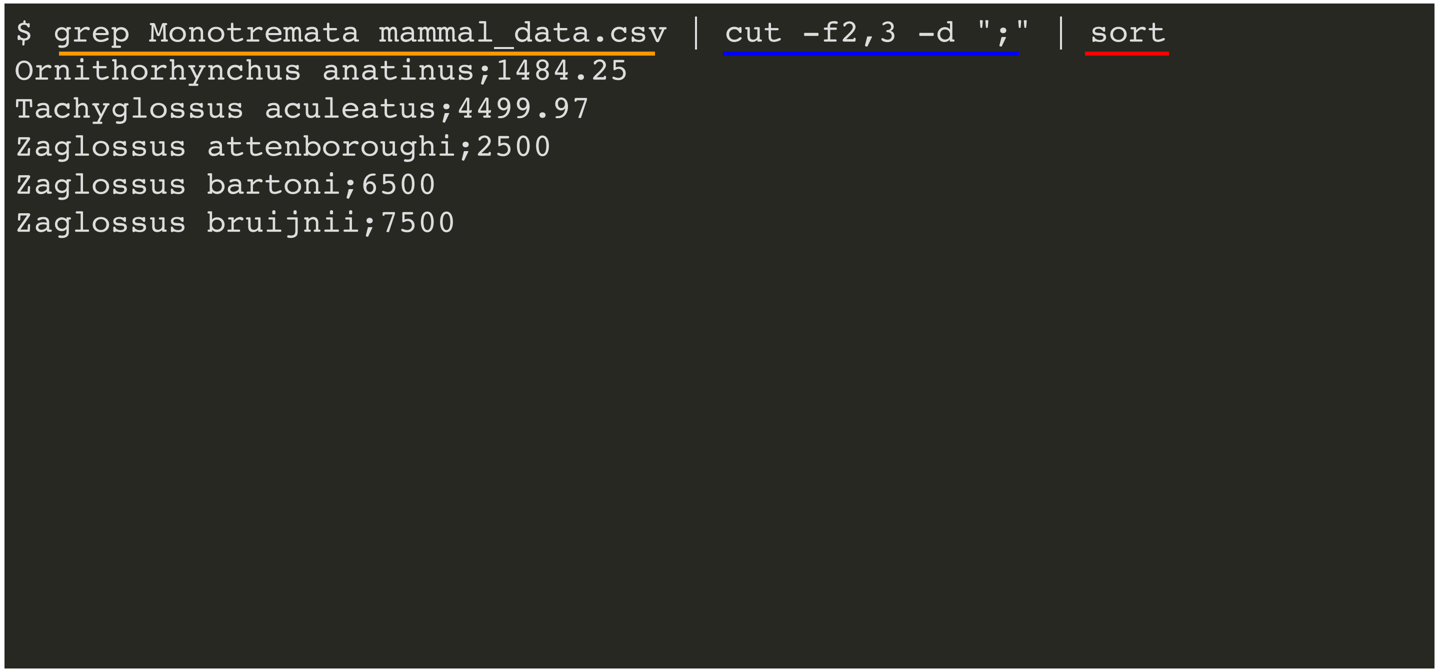
Worked pipeline example

How would you extract a sorted data table that lists the scientific name and adult body mass for all species in order Monotremata in tab-delimited format, then save that output to monotreme_mass.tsv?

```
$ grep Monotremata mammal_data.csv | cut -f2,3 -d ";"
Tachyglossus aculeatus;4499.97
Zaglossus bruijnii;7500
Zaglossus attenboroughi;2500
Zaglossus bartoni;6500
Ornithorhynchus anatinus;1484.25
```

Worked pipeline example

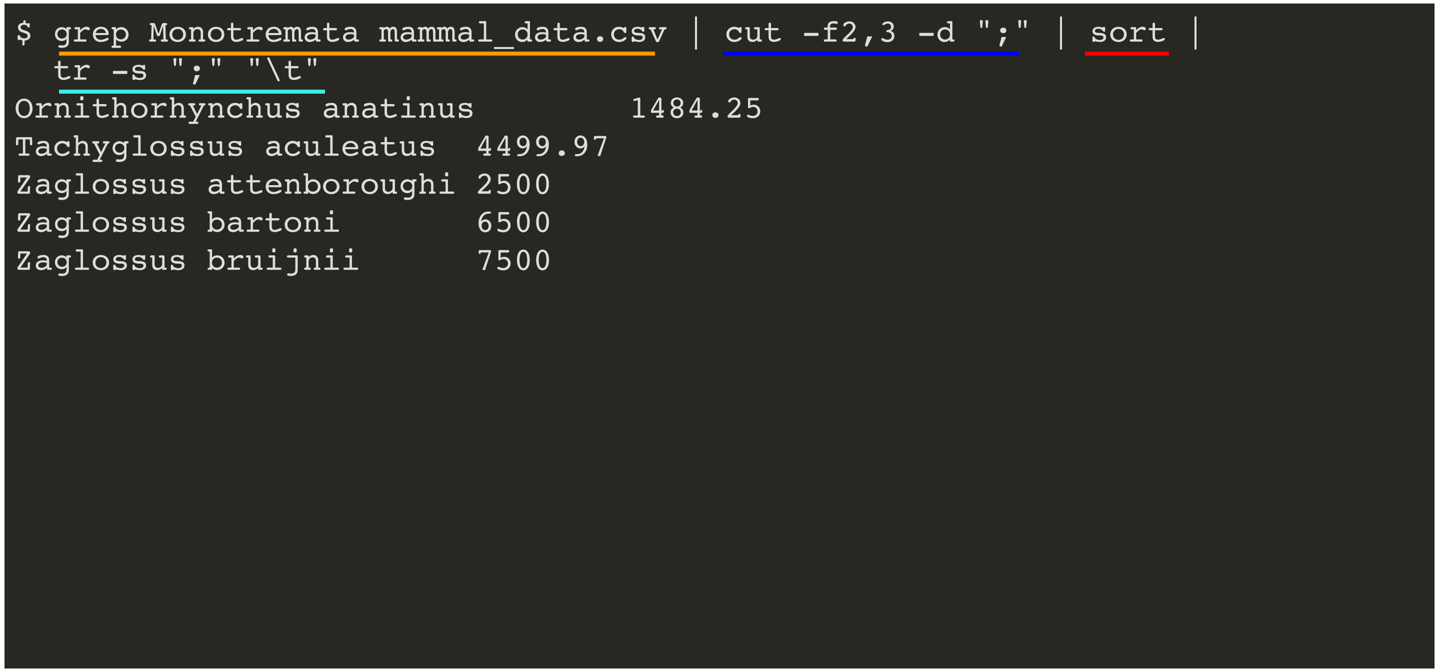
How would you extract a sorted data table that lists the scientific name and adult body mass for all species in order Monotremata in tab-delimited format, then save that output to monotreme_mass.tsv?



```
$ grep Monotremata mammal_data.csv | cut -f2,3 -d ";" | sort
Ornithorhynchus anatinus;1484.25
Tachyglossus aculeatus;4499.97
Zaglossus attenboroughi;2500
Zaglossus bartoni;6500
Zaglossus bruijnii;7500
```

Worked pipeline example

How would you extract a sorted data table that lists the scientific name and adult body mass for all species in order Monotremata in tab-delimited format, then save that output to monotreme_mass.tsv?



```
$ grep Monotremata mammal_data.csv | cut -f2,3 -d ";" | sort |
  tr -s ";" "\t"
Ornithorhynchus anatinus        1484.25
Tachyglossus aculeatus  4499.97
Zaglossus attenboroughi 2500
Zaglossus bartoni       6500
Zaglossus bruijnii      7500
```

Worked pipeline example

How would you extract a sorted data table that lists the scientific name and adult body mass for all species in order Monotremata in tab-delimited format, then save that output to monotreme_mass.tsv?

```
$ grep Monotremata mammal_data.csv | cut -f2,3 -d ";" | sort |
  tr -s ";" "\t" > monotreme_mass.tsv
$ cat monotreme_mass.tsv
Ornithorhynchus anatinus        1484.25
Tachyglossus aculeatus  4499.97
Zaglossus attenboroughi 2500
Zaglossus bartoni       6500
Zaglossus bruijnii      7500
```

Example problem

Write a pipeline to compute the number of uniquely named files in a directory that remain after applying various filters.

What code is needed for these steps?

1. find file paths for all .txt files in local directory (and subdirectories)
2. filter out all files whose names contain the text "ignore"
3. reverse each line in the text stream
4. extract the first column in the reversed text (i.e. the reversed file names)
5. sort the reversed file names
6. filter out duplicate (non-unique) file names
7. print the number of lines in the filtered text stream

Lab 3A

github.com/WUSTL-Biol4220/home/labs/lab_03A.md