

# Lecture 05

## text processing



Course: Practical Bioinformatics (BIOL 4220)  
Instructor: Michael Landis  
Email: [michael.landis@wustl.edu](mailto:michael.landis@wustl.edu)



# Lecture 05 outline

Last time: text, pipes, redirects

This time: text processing pipelines

- text formats

- text processing commands

- pipeline examples

# Text strings

Programs share input/output as text

Program A can't necessarily produce output in a *format* that can be read by Program B

Pipelines often reformat text to pass it between programs

# Human-readable format

```
~$ cat format_1.txt
```

```
This file contains DNA sequences downloaded from GenBank  
for multiple species (incl. human, mouse, cow) and multiple  
genes (incl. C02 and cytB). The first sequence, for the C02  
gene in Mus musculus, is interesting, in part, because its  
nucleotide site positions 31, 239, and 594 are G, C, and T,  
respectively, whereas C02 for all other species described  
in this document report nucleotides A, C, and T, at those  
...
```

Plain text description

# Some computer-readable formats

```
~$ cat format_2.txt
species, gene, sequence
Mus_musculus, C02, ACGTCAGGGCATT...
Homo_sapiens, C02, ACGTCACCGCATT...
Bos_taurus, C02, ACGTCACTGCATCAT...
Mus_musculus, cytB, CGGCAAGATGCC...
Homo_sapiens, cytB, CTGCAAGTTGCC...
Bos_taurus, cytB, CAGCAGGATGCCTT...
...
```

Comma-separated  
values (CSV) format

```
~$ cat format_4.txt
> Mus_musculus_C02
ACGTCAGGGCATTTCATCGTGCGATC...
> Homo_sapiens_C02
ACGTCACCGCATTGCTCGTGCGATC...
> Bos_taurus_C02
ACGTCACTGCATCATTTCTCGTGCGATC...
> Mus_musculus_cytB
CGGCAAGATGCCGATCTCGTGCGATC...
> Homo_sapiens_cytB
CTGCAAGTTGCCTGACTCGTGCGATC...
> Bos_taurus_cytB
CAGCAGGATGCCTTTCTCGTGCGATC...
...
```

FASTA format

```
~$ cat format_4.txt
{
  'Mus_musculus' : {
    'genes': [
      'C02': 'ACGTCAGGGCATT...',
      'cytB' : 'CGGCAAGATGCC...'
    ]
  },
  'Homo_sapiens' : {
    'genes': [
      'C02': 'ACGTCACCGCATT...',
      'cytB' : 'CTGCAAGTTGCC...'
    ]
  },
  ...
}
```

JSON format

# No format is perfect for all imaginable 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.



SOON:

SITUATION:  
THERE ARE  
15 COMPETING  
STANDARDS.

# .csv format

The ***comma-separated value*** format is flexible and easy to work with (parse)

- text between commas are columns
- each line is a row
- all rows have the same number of columns

```
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. Each sequence is described by two adjacent sets of rows.

- First row begins with “>” and gives the sequence name
- Following rows report the sequence data (e.g. ACGT) for that named sequence, until the next “>” row

```
> Mus_musculus_C02
ACGTCAGGGCATTTCATCGTGCGATC
CGATCAACGCTCATGGCATTACTCAG
...
> Homo_sapiens_C02
ACGTCACCGCATTGCTCGTGCGATC
CTGTCAATGCTCATGCTATTACTCAG
...
> Bos_taurus_C02
ACGTCAGTGCATCATTTCTGTGCGATC
CGGTCAGCGCTCATGCTACTACTCAG
...
```



# .sam format

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

- header lines (optional) begin with “@”
- following rows contain 11 columns
- columns identify mapped read by name, position, sequence, identity, etc.

```
@HD VN:1.6 S0: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
```

(example, don't memorize)

# .vcf format

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

- header lines (“##”) report file metadata
- following rows report each variant, its position, its type (e.g. SNP, microsat), and how the variant differs from the reference

```
##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
```

(example, don't memorize)

# Common forms of text processing

- **sort** text (e.g.) alphabetically
- filter out **duplicate** data entries
- **parse** or **tokenize** text strings into fields by a delimiter token
- **join** or **paste** multiple text strings together
- **translate** a set of characters into a new set of characters (e.g. lowercase to uppercase)
- **cut** relevant text out of a data table
- **find** all text that matches a search pattern

# Text format determines how that text is best processed

```
# search csv file for C02
$ grep C02 sequences.csv
Mus_musculus,C02,ACGTCAGGGCATT...
Homo_sapiens,C02,ACGTCACCGCATT...
Bos_taurus,C02,ACGTCACTGCATCAT...
```

```
# search fasta file for C02
$ grep C02 sequences.fasta
> Mus_musculus_C02
> Homo_sapiens_C02
> Bos_taurus_C02
```

# 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*, pattern searcher

*grep* returns lines that match a pattern;  
options explained in manual

```
$ cat seq1.csv
Viburnum_molle,ACAGTAGGTAGACACAGTA
Viburnum_acerifolium,ACGGTAGGTATACACAGTA
Viburnum_nudum,ACCGTAGATATACACAGTA
Viburnum_molle,ACAGTAGGTAGACACAGTA
# ignore case
$ 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*


*Sort* each line in a file alphanumerically;  
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,ACGGTAGGTCTACTACTGTA
Viburnum_clemensiae,AGGGTCAGTCTACTACTGTA
# sort two files
$ sort seq1.csv seq2.csv
Viburnum_acerifolium,ACGGTAGGTATACACAGTA
Viburnum_clemensiae,AGGGTCAGTCTACTACTGTA
Viburnum_lantana,ACGGTAGGTATACGCAGTA
Viburnum_molle,ACAGTAGGTAGACACAGTA
Viburnum_nudum,ACCGTAGATATACACAGTA
Viburnum_tinus,ACGGTAGGTCTACTACTGTA
# 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 out any line that is identical to the previous line,  
then prints *unique* text to stdout

```
$ cat seq3.csv
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*

Match input text stream against the first pattern, then *translate* that text using 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*

Print each line in *reverse* order to stdout

```
# example csv for Darwin's finches
$ cat finch.csv
name,wingL,tarsusL
magirostris,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*

*Cut* selected text and print to stdout;  
select text using a delimited field (-d)  
or character position (-c)

```
# example csv for Darwin's finches
$ cat finch.csv
name,wingL,tarsusL
magirostris,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
magirostris,3.038950
conirostris,2.984200
# extract the first five character columns
$ cut -c1-5 finch.csv
name,
magi
conir
```

# *paste*

*Paste* interleaved lines from multiple files to stdout;  
use `-d` to specify token when lines join

```
# 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*

*Join* lines from two files by common field  
then write joined text to stdout

```
# 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*

*Find* all filesystem objects that match search criteria,  
then print each path 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
```

# Designing a pipeline

Don't expect that a pipeline will work correctly on the first design attempt!

1. create simplified versions of input files to easily identify problems
2. if the pipeline modifies the filesystem, test against a copy of the filesystem
3. add and test commands one-by-one

# Worked pipeline example

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

```
$ head 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 create and sort a data table that lists the scientific name and adult body mass for **all species in order Monotremata** in tab-delimited format, and 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 create and sort a data table that lists the scientific name and adult body mass for all species in order Monotremata in tab-delimited format, and 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 create and **sort** a data table that **lists** the **scientific name and adult body mass** for **all species in order Monotremata** in tab-delimited format, and 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 create and **sort** a data table that **lists** the **scientific name** and **adult body mass** for **all species** in order **Monotremata** in **tab-delimited format**, and 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
```

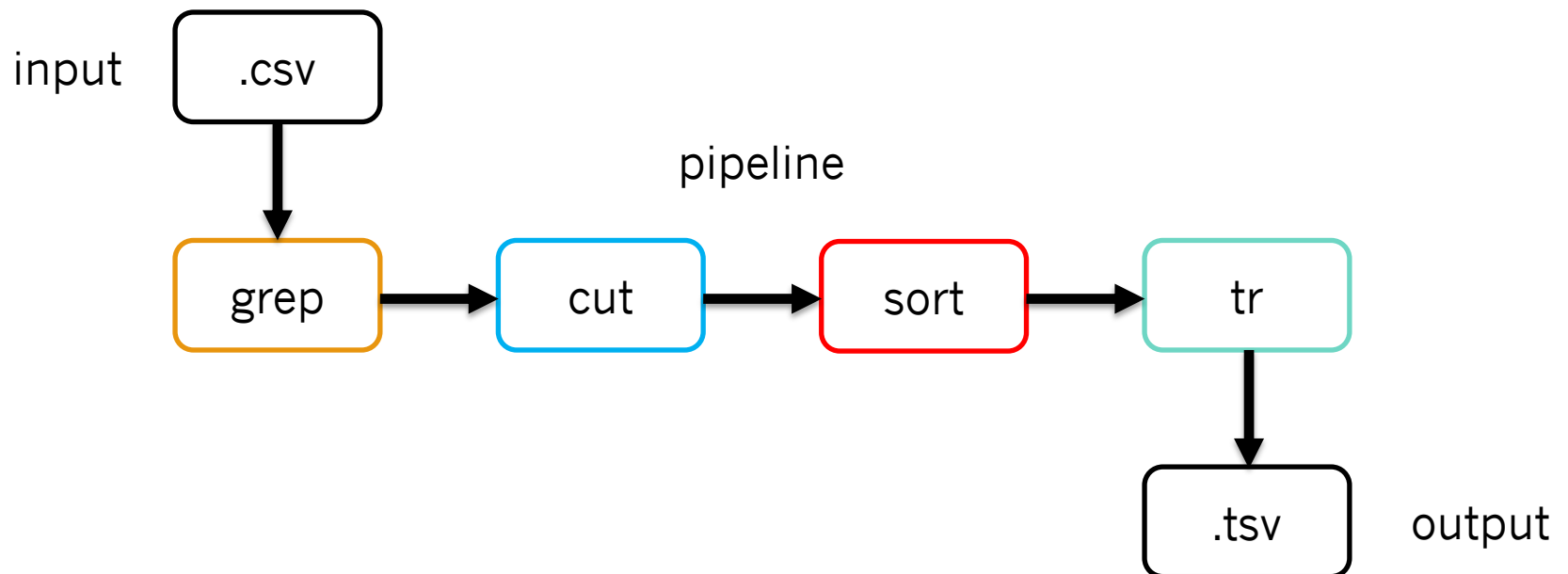
# Worked pipeline example

How would you create and **sort** a data table that **lists** the **scientific name** and **adult body mass** for **all species** in order **Monotremata** in **tab-delimited format**, and 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
```

# Worked pipeline example

How would you create and **sort** a data table that **lists** the **scientific name and adult body mass** for **all species in order Monotremata** in **tab-delimited format**, and then save that output to **monotreme\_mass.tsv**?



# Example pipeline 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?*

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

# Overview for Lab 05