Week 5, Lecture 10

István Albert
Bioinformatics Consulting Center
Penn State
Some Programming Required

• Existing software tools can rarely do all steps

• Source data in different files may be formatted differently

• We need to bridge the differences with simple transformations

• Some programming ability is usually necessary for every project
Files for this presentation:
lecture-10.zip → SGD_features.tab obtained from SGD
The first scripting language: **awk**

- appeared in 1977, strong UNIX (1972) roots
- it is the precursor of languages such as **Perl** (1987) and **Python** (1989)
- Has fallen into disuse for a while; the large text have datasets lead to a resurgence of the language
The structure of an **awk** program

```
awk '{ pattern { action } }'
```

- **Line oriented**
  - Acts on each line of a stream
  - And produces another line
- If there is a match → do something with each line
- Default match = match everything
Awk *automatically splits* the input by whitespace (spaces and tabs) and assigns names to them:

- $0$ the entire line
- $1$ first field
- $2$ second field
- ...
- **NF** the number of fields
- **NR** the number of the current line
The whitespace curse: spaces and tabs

• Many tools will autosplit by whitespace → this was thought to be convenient but is also the source of extremely subtle errors → column shift in a tab file if a field contains spaces

• **Always specify the character to be split by!**

• This refers to programming languages as well! Don’t use `split()` methods with their default behavior! (Python, Perl etc)
Rename awk to have a split character

```
# set the default split for awk
alias awk='awk -F "\t"
```

Tip: you can add this to the .bashrc file so that it is always activated (see lecture 6, page 2)
A awk program: replacement for cut

```bash
# make the name shorter
mv SGD_features.tab SGD.tab

# cut the first column
cat SGD.tab | cut -f 1 | head -2
S000028864
S000002143

# cut the first column with awk

cat SGD.tab | awk '{ print $1 }' | head -2
S000028864
S000002143
```
Reorder columns

```bash
$ # reorder columns
$ cat SGD.tab | awk ' { print $4, $2 }' | head -4
TEL01L-TR telomeric_repeat
YAL069W ORF
CDS
TEL01L-XR X_element_combinatorial_repeats
$
$ # filter and reorder
$ cat SGD.tab | awk ' $2=="ORF" { print $4, $2 }' | head -4
YAL069W ORF
YAL068W-A ORF
YAL068C ORF
YAL067W-A ORF
$```

matching

action
Store/run it as a separate program

```perl
$2=="ORF" { print $4, $2 }
```

```bash
$ cat SGD.tab | awk -f program1.txt | head -3
YAL069W ORF
YAL068W-A ORF
YAL068C ORF
$
Interval lengths for genes

```
$2=="ORF" { print $4, $2, $11 - $10 }
```

```
$ cat SGD.tab | awk -f program1.txt | head -5
YAL069W ORF 314
YAL068W-A ORF 254
YAL068C ORF -362
YAL067W-A ORF 227
YAL067C ORF -1781
$`
```
Operators

- `+ - * /` → for numerical context
- `%` → modulo division (remainder of division)
- `<space>` → string concatenation
- `==, !=` → equal, not equal
- `~, !~` → match, no match (regular expressions)
Special patterns

- **BEGIN** → before the stream starts
- **END** → after the stream ends

```bash
$ cat SGD.tab | cut -f 2 | grep "ORF" | wc -l
6607
$ cat SGD.tab | awk -f counter1.txt
6607
```
Compute more values at a time

```plaintext
1 #
2 # computes the number of all and dubious genes
3 #
4 BEGIN { num = 0; dub=0 }
5 $2 == "ORF" { num = num + 1 }
6 $3 == "Dubious" { dub = dub + 1 }
7 END { print num, dub }
```

```
$ cat SGD.tab | awk -f counter1.txt
6607 809
$ 
```
Advanced awk

- conditionals: `if`
- loops: `for`, `while`
- `break`, `continue`
- associative data structures (hash, dictionary)

Though it is probably better to learn Python if one finds that they would be needing these.

But you can do a lot with just basic awk!

Awk’s power comes from its simplicity – one liners!
A few awk resources

Large number of resources, the quirky name makes it very searchable → **How to do X with awk?**

- **How to use awk:**
  [http://sparky.rice.edu/awk.html](http://sparky.rice.edu/awk.html)
1. Write an awk/perl/python program that generates a minimal GFF file from the SGD.tab file

2. Minimal GFF $\rightarrow$ chromosome, start, end, strand, replace the other fields with period (.)

(Note: due to the quirks in the file you may note be able to create a valid GFF – in that case state how close you got to the GFF specification and describe in what way does your file not conform the GFF standard)