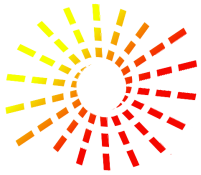
Introduction to PERL II

March 11, 2014

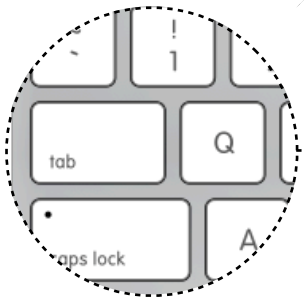
Noe Fernandez & Lukas Mueller



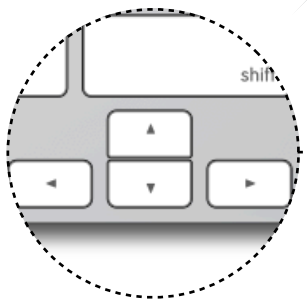
Class Content

- File I/O
- Arguments
- Regular expressions
- Tab-delimited files parsing
- Subroutines
- Modules
- BioPerl
- Two files comparison

Unix review



Use tab key to autocomplete names



Use up and down arrows to navigate the command history

http://btiplantbioinfocourse.wordpress.com

FASTA format

A sequence in FASTA format begins with a single-line description, followed by lines of sequence data. The description line is distinguished from the sequence data by a greater-than (">") symbol at the beginning.

http://www.ncbi.nlm.nih.gov/

description line

sequence data

```
>sequence_ID1 description
ATGCGCGCGCGCGCGCGGGTAGCAGATGACGACACAGAGCGAGGATGCGCTGAGAGTA
GTGTGACGACGATGACGGAAAATCAGATGGACCCGATGACAGCATGACGATGGGACGGGA
AAGATTGGACCAGGACAGGACCAGGACCAGGACCAGGGATTAGA
>sequence_ID2 description
ATGGGGGGGACGACGATGGACACAGAGACAGAGACGACGACAGCAGACAGATTTACCTTA
GACGAGATAGGAGAGACGACAGATATATATATATAGCAGACAGACAGACATTTAGACGAG
ACGACGATAGACGATaaaaataa
```

Tab-delimited text files

Tab-delimited files are a very common format in scientific data. They consist in columns of text separated by tabs. Other file formats could have different delimiters.

Query	Subject	id %	length	mismatch	gaps	qstart	qend	sstart	send	evalue	score
ATCG00500.1	PACid:23047568	64.88	299	64	2	220	477	112	410	5e-131	388
ATCG00500.1	PACid:23052247	58.88	321	69	3	220	477	381	701	3e-117	361
ATCG00890.1	PACid:16418828	90.60	117	11	0	18	134	1	117	1e-71	220
ATCG00890.1	PACid:16412855	90.48	147	14	2	41	387	27	173	1e-68	214
ATCG00280.1	PACid:24129717	95.99	474	19	0	1	474	1	474	0.0	847
ATCG00280.1	PACid:24095593	95.36	474	22	0	1	474	1	474	0.0	840
ATCG00280.1	PACid:20871697	94.94	474	24	0	1	474	1	474	0.0	837

Tabular blast output example

Blast, SAM (mapping), BED, VCF (SNPs), GTF, GFF ...

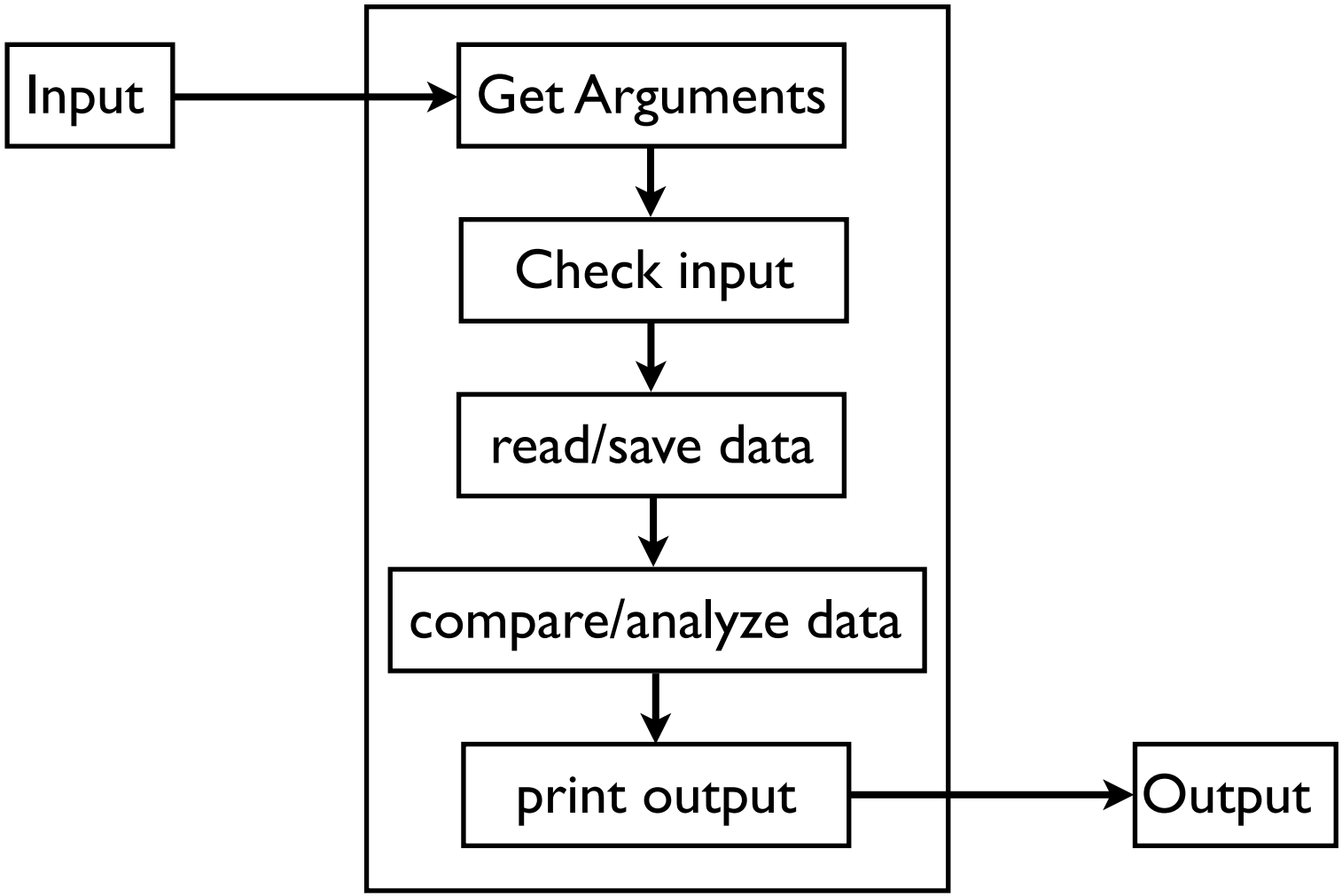
Class material

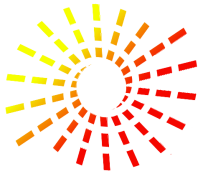
Exercise:

- write a script able to extract from a blast tabular result file the lines that belong to a list of query genes
- Input files: *blast_sample.txt* and *gene_list.txt*

ftp://ftp.solgenomics.net/bioinfo_class/2014/perl_class_files.zip

Script structure example





Hashes

BTI

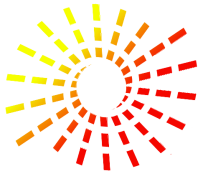
```
#!/usr/bin/perl

use strict;
use warnings;

# declare hash
my %last_names_h = ();

# fill hash data
$last_names_h{"Noe"} = "Fernandez";
$last_names_h{"Lukas"} = "Mueller";


# print hash elements
print "$last_names_h{Lukas}\n";
```



File I/O, Reading a file line by line

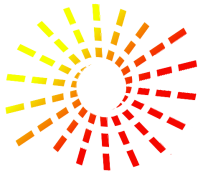
BTI

```
#!/usr/bin/perl

use strict;
use warnings;

# open file to read the input
open (my $fh, "<", "input.txt") || die ("\nERROR: the
file input.txt could not be found\n");

# read input file line by line
while (my $line = <$fh>) {
  chomp($line);
  print “$line\n”;
}
```



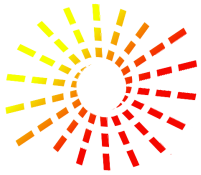
File I/O, Writing to output file

```
#!/usr/bin/perl

use strict;
use warnings;

# open file to write the output
open (my $out_fh, ">", "output.txt");

print $out_fh "Hello world\n";
```



Arguments

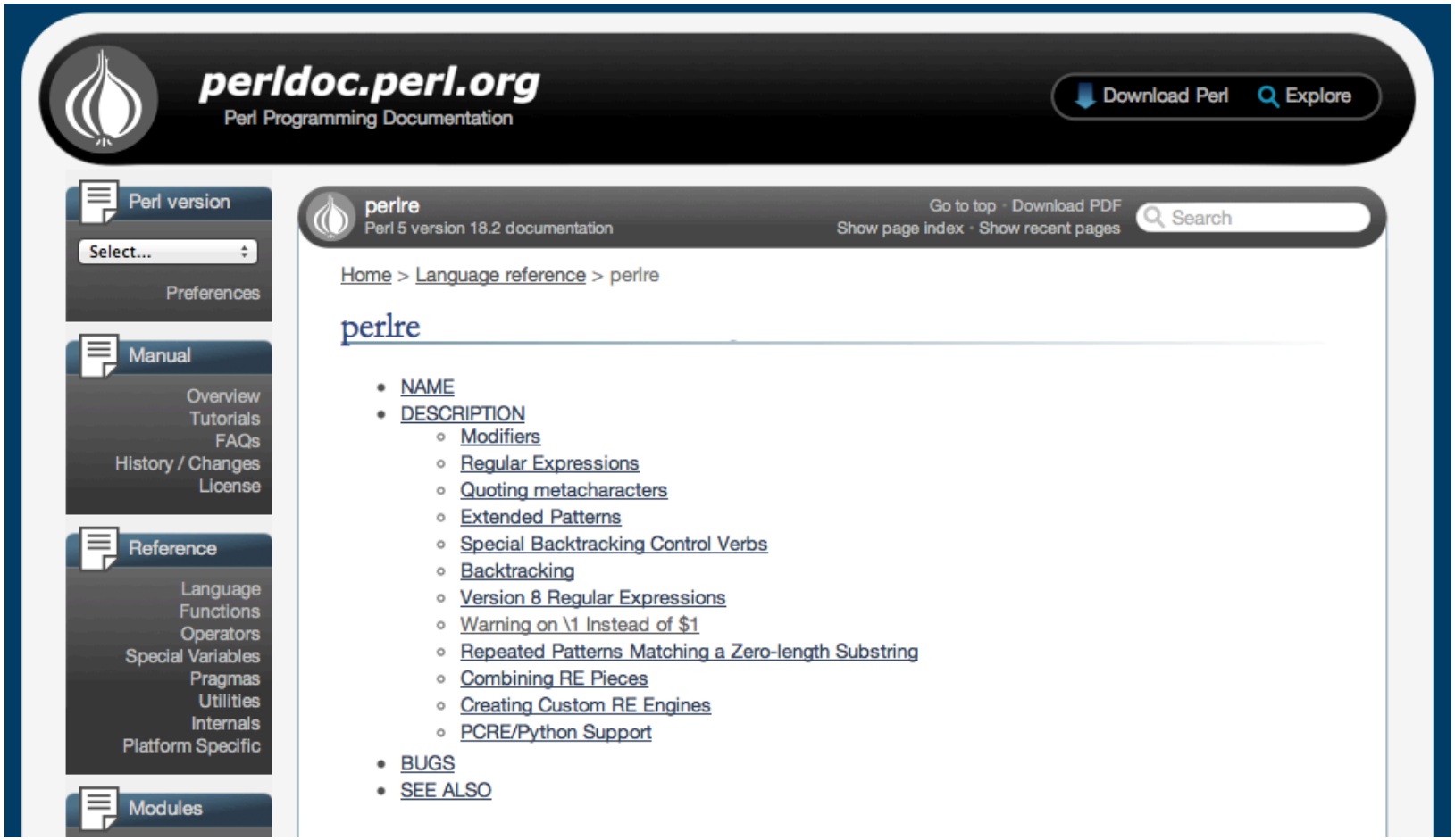
```
#!/usr/bin/perl

use strict;
use warnings;

# check arguments and print usage
if (scalar(@ARGV) != 2) {
  print "Usage: perl script.pl <file.txt> <min_id>\n";
  exit;
}

# save arguments in variables
my ($file, $min_id) = @ARGV;
```

Regular expressions



http://perldoc.perl.org/perlre.html

Regular expressions

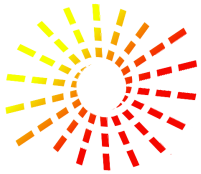
```
$var =~ m/pattern/;
$var =~ /pattern/i; # case insensitive

# value in the parenthesis are saved in $1, $2...
$var =~ /(match1).+(match2)/;
$match1 = $1;
$match2 = $2;


# Example, print description lines in fasta files:
if ($line =~ /^>/) {
   print "$line\n"
}
```

Result

```
>Solyc12g013590.1.1 Helicase sen1
>Solyc08g006880.2.1 ABC transporter
>Solyc10g081020.1.1
```



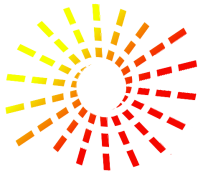
Regular expressions

[A-Z]	Match any uppercase letter
[a-z]	Match any lowercase letter
\w	Match any digit or word character (\w = [A-Za-z0-9])

\W	[3]	Match a non-"word" character
\s	[3]	Match a whitespace character
\S	[3]	Match a non-whitespace character
\d	[3]	Match a decimal digit character
\D	[3]	Match a non-digit character

1.	*	Match 0 or more times
2.	+	Match 1 or more times
3.	?	Match 1 or 0 times
4.	{n}	Match exactly n times
5.	{n,}	Match at least n times
6.	{n,m}	Match at least n but not more than m times

1.	\	Quote the next metacharacter
2.	^	Match the beginning of the line
3.	.	Match any character (except newline)
4.	$	Match the end of the line (or before newline at the end)
5.	|	Alternation
6.	()	Grouping
7.	[]	Bracketed Character class



Regular expressions

BTI

```
$var =~ s/pattern/new_value/; # substitution
$var =~ s/\s//; # remove first space from $var

$var =~ s/pattern/new_value/g; # global substitution
$var =~ s/\s//g; # remove all spaces from $var


$counts = ($seq =~ tr/ACGT/TGCA/); # translation
```

```
perl -e '$a="aaccttgg";$c = $a=~tr/acgt/tgca/;print "$a\nc: $c\n";'
```

Regular expressions

http://www.perlfect.com/articles/regextutor.shtml

/Solyc(\d+)g(\d+)[\.\d]+/ match Solyc09g072620.2.1

/AT(\d)G(\d+)\.\d/ match AT5G60300.1

/([a-z]+)(\d+)g(\d+)[\.\d]+/i match Both

/([A-Za-z]+)(\d+)[G|g](\d+)[\.\d]+/ match Both

[A-Z]	Match any uppercase letter
[a-z]	Match any lowercase letter
\d	Match a decimal digit character
+	Match one or more times
\	Quote the next metacharacter
[ ]	Bracketed Character class
()	Grouping, and save in $1, $2...
|	Alternation

Regular expressions

http://www.perlfect.com/articles/regextutor.shtml

AT5G60300.1 Solyc09g072620.2.1

(\S+)\s+(\S+)

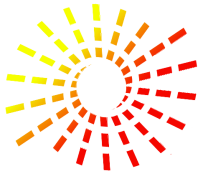
^(AT)(\d)G(\d{5})\.\d\s+(Solyc)(\d{2})g(\d{6})\.\d\.\d$

^([^G]+)G([^G]+)\s+([^g]+)g([^g]+)$

not greedily

^([^G]+)G([^G]+?)\s+([^g]+)g([^g]+)$

\S	Match a non-whitespace character
\s	Match a whitespace character
{n}	Match exactly n times
^	Match the beginning of line
$	Match the end of line



Tab-delimited files parsing

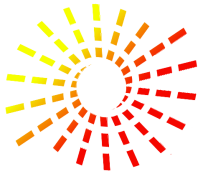
split lines by tabs getting each column value in a variable

```
#!/usr/bin/perl

use strict;
use warnings;

# open file to read the input
open (my $blast_fh, "<", "blast.txt");

# read input file line by line
while (my $line = <$blast_fh>) {
  chomp($line);
  my ($col1,$col2,$col3,$col4) = split("\t",$line);
  print “$col2\t$col4\n”;
}
```



Tab-delimited files parsing

split lines by tabs saving values in an array

```
#!/usr/bin/perl

use strict;
use warnings;

# open file to read the input
open (my $blast_fh, "<", "blast.txt");

# read input file line by line
while (my $line = <$blast_fh>) {
  chomp($line);
  my @line_array = split("\t",$line);
  print “$line_array[3]\t$line_array[4]\n”;
}
```