

Day 13: Recipes II

(Mostly Numeric) Data

Suggested Reading:
Perl Cookbook (2nd Ed.)

Chapter 2: Numbers

Chapter 8: File Contents *(if not read already)*

CS 368, Fall 2011 • 8 Weeks, 24 Oct – 15 Dec

- Section 2
 - Introduction to Python
 - Nick LeRoy
 - Tuesdays & Thursdays, 9:55–10:45 a.m.
 - Computer Sciences 1263
- Section 4
 - Introduction to Scripting for CHTC **NEW!**
 - Tim Cartwright
 - Tuesdays & Thursdays, 1:20–2:10 p.m.
 - Computer Sciences 1257

Homework Review

Homework Preview

AO&SS RIG Data

- AO&SS building, Rooftop Instrument Group
- One record every 5 seconds or so
- One day \cong 17K lines, 2.7 MB file
- Comma-separated values (CSV) — no quoting!
- Weird date and time formats

```
1,2011,210,759,56.9,979.31,5.8577,30.092,20.648,980,587  
32,48.394,19.375,19.631,20.826,38.257,3.3178,5.358,141.  
8,78.06,20.075,16.226,142,19.885,.13195,0,0,30.096
```

```
1,2011,210,800,1.9,979.35,5.8577,30.092,20.652,979.96,5  
8732,48.394,19.372,19.633,20.826,38.251,2.9901,2.1385,1  
41.8,78.215,20.068,16.25,142.2,19.879,.06598,0,0,30.095
```

Data Files

Fixed-Width Text Data Files

```
00000000000111111111122222222222
012345678901234567890123456789
Livny          Miron    4367    20856
Cartwright    Tim       4265    24002
LeRoy         Nick      4289    55761
De Smet       Alan      4247    53151
```

```
while (my $line = <INPUT>) {
    my $lastname = substr($line, 0, 12);
    my $firstname = substr($line, 12, 7);
    ...
}

# E.g., $lastname = 'Livny          ';
```

String Trimming

- Input strings may have leading, trailing whitespace
- User input, data files, configuration, arguments
- Can mess up REs, comparisons, etc.

```
sub trim {  
    my $string = shift;  
    $string =~ s/^\s+//;  
    $string =~ s/\s+$//;  
    return $string;  
}  
  
trim("\t oops \t\n"); => 'oops'
```


Tab-Delimited Text Data Files

```
Livny\tMiron\t4367\t20856\nCartwright\tTim\t4265\t24002\nLeRoy\tNick\t4289\t55761\nDe Smet \tAlan\t4247\t 53151 \n
```

```
while (my $line = <INPUT>) {  
    my @parts = split(/\t/, $line);  
    my $lastname = trim($parts[0]);  
    my $firstname = trim($parts[1]);  
    ...  
}
```

Comma-Separated Values (CSV) Data Files

```
Livny,Miron,4367,20856  
Cartwright,Tim,4265,24002  
LeRoy,Nick,4289,55761  
De Smet ,Alan,4247, 53151
```

```
while (my $line = <INPUT>) {  
    my @parts = split(/,/ , $line);  
    my $lastname = trim($parts[0]);  
    my $firstname = trim($parts[1]);  
    ...  
}
```

CSV Challenges

- What if data contain commas?

```
"Livny, Miron",4367,20856,"fishing"  
"Cartwright, Tim",4265,24002,"gaming,biking"  
"LeRoy, Nick",4289,55761,"hunting,painting"  
"De Smet, Alan",4247, 53151,"LARPing"
```

- What if data contain commas *and* quotes?

```
"Cartwright, Tim",4265,24002,"gaming,biking",  
"CS 368-1 2009 Summer, ""Introduction to  
Scripting""; CS 368-1 2011 Summer,  
""Introduction to Perl"""
```

Complex CSV Solution

- Don't reinvent the wheel!
- Use CPAN's **Text::CSV**

```
use Text::CSV;
my @rows;
my $csv = Text::CSV->new();
open(INPUT, $filename) or die "die: $!\n";
while (my $row = $csv->getline(INPUT)) {
    $row->[2] =~ m/pattern/ or next; # grep
    push @rows, $row;
}
close(INPUT);
```

Floating-Point Numbers

Math:

$$0.725 = 725 / 1000$$

Perl:

0.725 \neq 725 / 1000

Rounding Floats

- How to round 2.5? 3.5? -2.5? -3.5?
- Pick a method: `sprintf`, `int`, `floor`, `ceil`
- Different tradeoffs

<code>raw</code>	<code>sprintf</code>	<code>int</code>	<code>floor</code>	<code>ceil</code>
-4.5	-4	-4	-5	-4
-3.5	-4	-3	-4	-3
-2.5	-2	-2	-3	-2
-1.5	-2	-1	-2	-1
-0.5	-0	0	-1	0
0.5	0	0	0	1
1.5	2	1	1	2
2.5	2	2	2	3
3.5	4	3	3	4
4.5	4	4	4	5

Fixed-Decimal Floats

- Example: Currency (\$23.45)
- Trick: Convert all fixed-decimal floats to integers
- Do all math as integers
- Convert back to float only to print

```
my $dollars = '$1234.56';  
$dollars =~ /^\$(\d+)\.(\d{2})$/;  
my $d = ($1 * 100) + $2;  
  
for (my $i = 0; $i < 10; $i++) { $d += 12; }  
printf('$%.2f', $d / 100);    # => $1235.76
```

Numeric Computations

(More) Trigonometric Functions

- Perl includes only 3 trig functions: **sin**, **cos**, **atan2**
- Options:
 - Derive others from these... *NOT!*
 - Use **POSIX** or **Math::Trig** (both built-in)

```
use Math::Trig;
```

```
my $half_pi = pi / 2;  
my $x = tan(0.9);  
my $y = acos(3.7);  
my $z = asin(3.4);
```

Logarithms in Other Bases

- Perl includes only natural log, $\log_e()$: `log()`
- Options:
 - Derive others from it: `log($n) / log($base)`
 - Use POSIX for `log10()`

```
sub log_x { log($_[1]) / log($_[0]) }  
my $x = log_x(2, $n); # log2(n)
```

```
use POSIX qw/log10/;  
my $x = log10($n); # log10(n)
```

Complex Numbers

- Don't reinvent the wheel!
- Use **Math::Complex**

```
use Math::Complex;
```

```
my $z = Math::Complex->make(5, 6);
```

```
my $t = 4 - 3*i + $z;
```

```
print Re($t) . "+" . Im($t) . "i\n";
```

```
=> 9+3i
```

Matrix Algebra

- Don't reinvent the wheel!
- Within Perl: Perl Data Language (**PDL**) modules
- Or, run Octave (free!) or Matlab (\$\$\$) from Perl

Homework

AO&SS RIG Data

- Instruments on top of AO&SS building
- One record every 5 seconds or so
- One day \cong 17K lines, 2.7 MB file
- Comma-separated values (CSV) — no quoting
- Weird date and time formats

```
1,2011,210,759,56.9,979.31,5.8577,30.092,20.648,980,587  
32,48.394,19.375,19.631,20.826,38.257,3.3178,5.358,141.  
8,78.06,20.075,16.226,142,19.885,.13195,0,0,30.096
```

```
1,2011,210,800,1.9,979.35,5.8577,30.092,20.652,979.96,5  
8732,48.394,19.372,19.633,20.826,38.251,2.9901,2.1385,1  
41.8,78.215,20.068,16.25,142.2,19.879,.06598,0,0,30.095
```

Weather Analysis, Part II

- New script: Get, parse, condense, and save RIG data
- Take command-line argument(s) to specify date
- **DO NOT DOWNLOAD TOO MUCH!!!!**
 - Save as, e.g., **rig-2011-08-01.txt**
 - Only download if file does not exist already
 - Even then, only download the days you need
 - Code already written to do this...
- Save date, hour, min, and max temperatures (°F)
- One download => one saved data file