

# Day 13: Recipes III

*(Mostly) Numeric Data*

Suggested Reading:  
*Perl Cookbook* (2nd Ed.)

Chapter 2: Numbers

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

# Homework Review

# Homework Preview

## 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

```
1,2010,210,759,56.9,979.31,5.8577,30.092,20.648,980,587  
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
```

```
1,2010,210,800,1.9,979.35,5.8577,30.092,20.652,979.96,5  
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 $lastname = trim(substr($line, 0, 12));
    my $firstname = trim(substr($line, 12, 7));
    ...
}
```

## Tab-Delimited Text Data Files

```
"Livny\tMiron\t4367\t20856\n",  
"Cartwright\tTim\t4265\t24002\n",  
"LeRoy\tNick\t4289\t55761\n",  
"De 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 2010 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

## The Problem With Floats

$$0.725 \neq 725/1000$$

```
print 0.625; # == 1/2 + 1/8
=> 0.625

printf('%.39g', 0.625);
=> 0.625

print 0.725;
=> 0.725

printf('%.39g', 0.725);
=> 0.7249999999999999999977795539507496869191527
```

## Rounding Floats

- Pick a method: `sprintf`, `int`, `ceil`, `floor`
- Different tradeoffs

raw	<code>sprintf</code>	<code>int</code>	<code>ceil</code>	<code>floor</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

## Comparing Floats

```
$n = 0;
for ($i = 0; $i < 10; $i++) { $n += 0.1; }
printf('%.39g', $n);
=> 0.99999999999999999999888977697537484345957637
```

So, convert to fixed-decimal strings and compare:

```
sub fp_equal {
    my ($left, $right, $dp) = @_;
    return sprintf("%.${dp}g", $left) eq
           sprintf("%.${dp}g", $right);
}
```

## Fixed-Decimal Floats

- Trick: Convert all fixed-decimal floats to integers
- Do 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++) { $d += 12; }  
printf('$%.2g', $d / 100);
```

# 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**

```
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

```
1,2010,210,759,56.9,979.31,5.8577,30.092,20.648,980,587  
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
```

```
1,2010,210,800,1.9,979.35,5.8577,30.092,20.652,979.96,5  
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 III

- Download, parse, and reduce RIG data
- **DO NOT DOWNLOAD TOO MUCH!!!!**
  - Save and reuse one 1 download until code works
  - Even then, only download the days you need
- What data will you save? How often?
- Preserve older data and do not duplicate
- Can be separate from forecast script