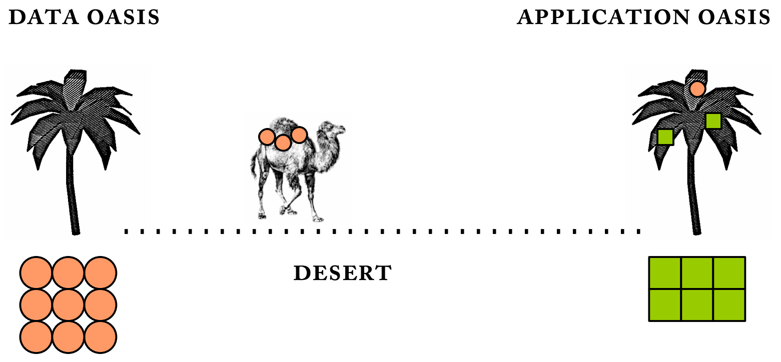
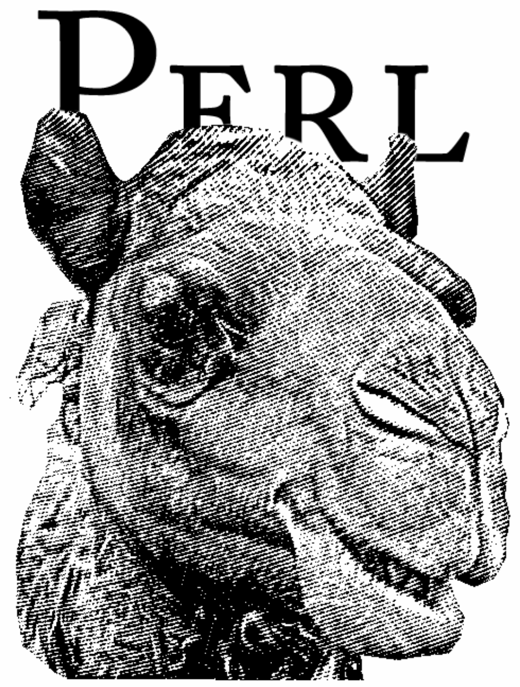
Perl Panacea?

- The camel represents the desirable features of Perl
  - O'Reilly colophon
- Why is the camel successful?
  - adapted itself to desert environment
    - low water needs (gets around with what's around)
    - elegant from a distance
    - still comfortable
  - not cute - until you get to know the camel

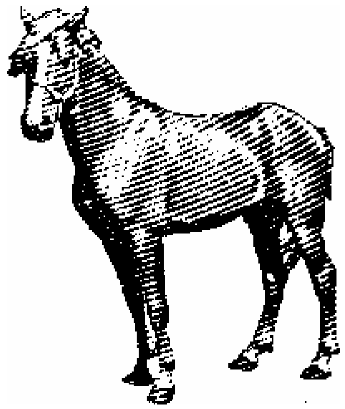
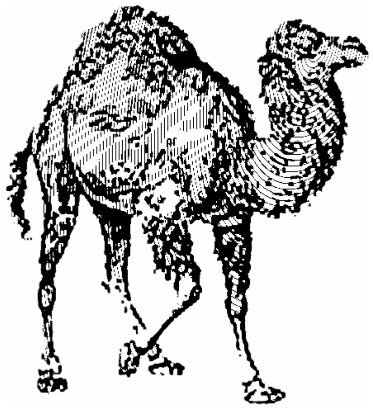
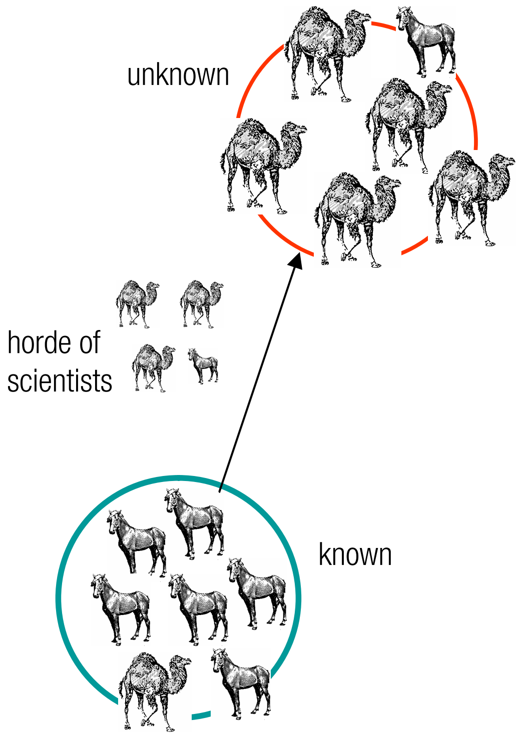




ADAPTIVE, LOW-MAINTENANCE
EASY TO RIDE

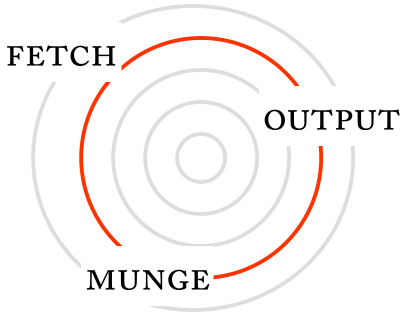
Perl as Explorer

Lots of camel mechanics in the desert, and we're in a desert

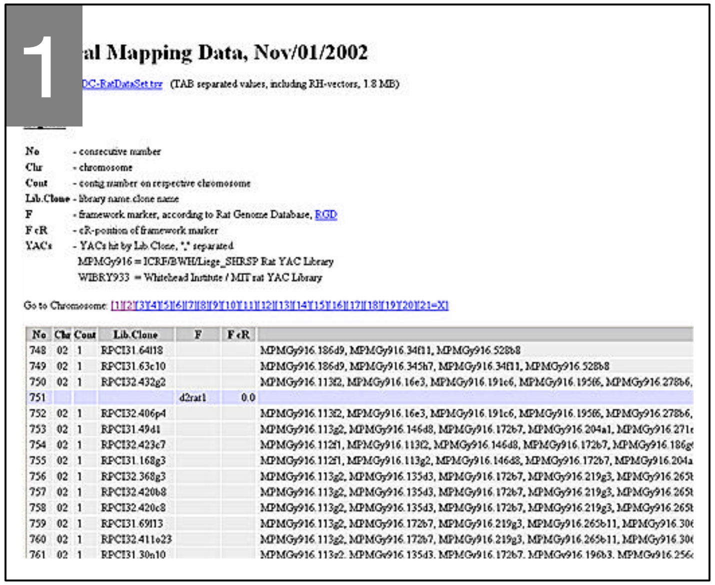
ANOTHER LEADING LANGUAGE	PERL	SIMPLIFIED EXPLORATION MODEL
		
~ mathematics beautiful & elegant rigorous, requires overhead fast on smooth ground slow in rough terrain ! killed by camel veterinarian	~ physics gets you there explores uninhabited terrain, cavalier average speed on smooth ground average speed in rough terrain * horse veterinarian OK	

Holy Triad of Analysis

- many types of analyses fall into this analysis triad
  - fetch from: file, user, pipe, http, ftp
  - munge: collate, sort, organize, count, enumerate
  - output: text, image, HTML, XML
- each step is made pleasant and easy with Perl



1



al Mapping Data, Nov/01/2002

DC-RatDataSet.txt (TAB separated values, including RH-vectors, 1.8 MB)

No	- consecutive number
Chr	- chromosome
Cont	- contig number on respective chromosome
Lib.Clone	- library name.clone name
F	- framework marker, according to Rat Genome Database, RGD
F cR	- cR-position of framework marker
YACs	- YACs hit by Lib.Clone, "," separated
	MPMGy916 = ICRF/BWH/Liege_SHRSP Rat YAC Library
	WIBRY933 = Whitehead Institute / MIT rat YAC Library

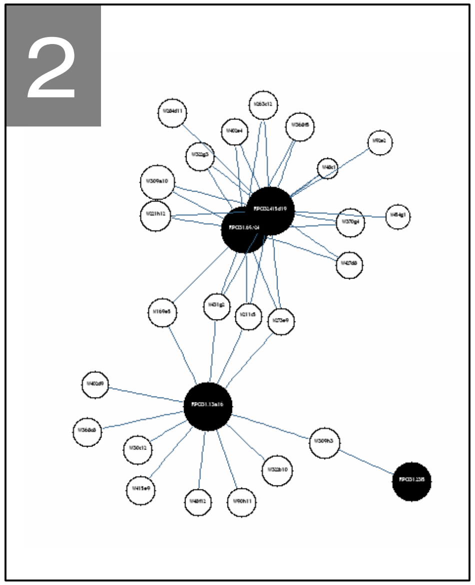
Go to Chromosome: [1][2][3][4][5][6][7][8][9][10][11][12][13][14][15][16][17][18][19][20][21=X]

No	Chr	Cont	Lib.Clone	F	F cR	
748	02	1	RPCI31.64l18			MPMGy916.186d9, MPMGy916.34f11, MPMGy916.528b8
749	02	1	RPCI31.63c10			MPMGy916.186d9, MPMGy916.345h7, MPMGy916.34f11, MPMGy916.528b8
750	02	1	RPCI32.432g2			MPMGy916.113f2, MPMGy916.16e3, MPMGy916.191c6, MPMGy916.195f6, MPMGy916.278b6,
751				d2rat1	0.0	
752	02	1	RPCI32.406p4			MPMGy916.113f2, MPMGy916.16e3, MPMGy916.191c6, MPMGy916.195f6, MPMGy916.278b6,
753	02	1	RPCI31.49d1			MPMGy916.113g2, MPMGy916.146d8, MPMGy916.172b7, MPMGy916.204a1, MPMGy916.271
754	02	1	RPCI32.423c7			MPMGy916.112f1, MPMGy916.113f2, MPMGy916.146d8, MPMGy916.172b7, MPMGy916.186g
755	02	1	RPCI31.168g3			MPMGy916.112f1, MPMGy916.113g2, MPMGy916.146d8, MPMGy916.172b7, MPMGy916.204a
756	02	1	RPCI32.368g3			MPMGy916.113g2, MPMGy916.135d3, MPMGy916.172b7, MPMGy916.219g3, MPMGy916.265
757	02	1	RPCI32.420b8			MPMGy916.113g2, MPMGy916.135d3, MPMGy916.172b7, MPMGy916.219g3, MPMGy916.265
758	02	1	RPCI32.420c8			MPMGy916.113g2, MPMGy916.135d3, MPMGy916.172b7, MPMGy916.219g3, MPMGy916.265
759	02	1	RPCI31.69l13			MPMGy916.113g2, MPMGy916.172b7, MPMGy916.219g3, MPMGy916.265b11, MPMGy916.30
760	02	1	RPCI32.411o23			MPMGy916.113g2, MPMGy916.172b7, MPMGy916.219g3, MPMGy916.265b11, MPMGy916.30
761	02	1	RPCI31.30n10			MPMGy916.113g2, MPMGy916.135d3, MPMGy916.172b7, MPMGy916.196b3, MPMGy916.256

! data we want is in a web table (Very Bad Thing™)

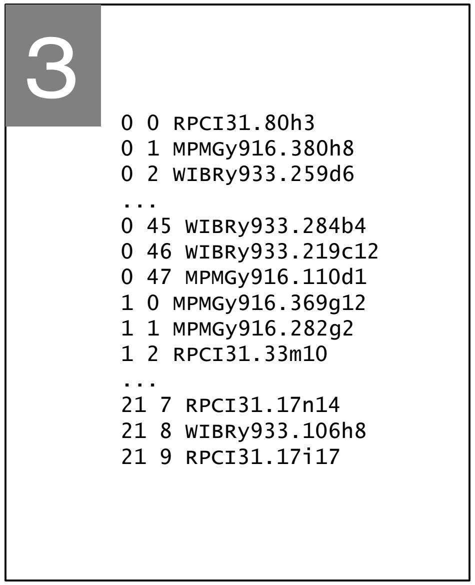
BAD USAGE

2



* visualize the relationships for sanity

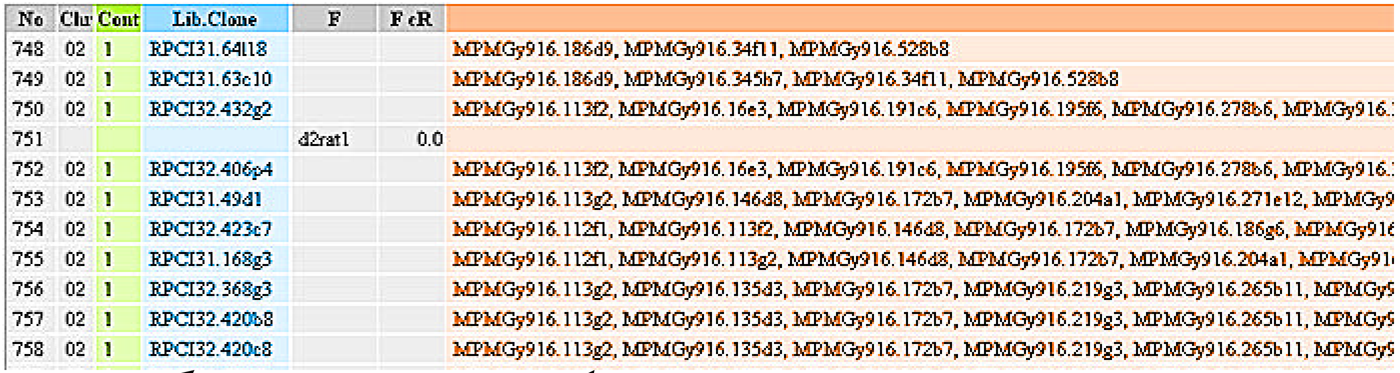
3



```
0 0 RPCI31.80h3
0 1 MPMGy916.380h8
0 2 WIBRy933.259d6
...
0 45 WIBRy933.284b4
0 46 WIBRy933.219c12
0 47 MPMGy916.110d1
1 0 MPMGy916.369g12
1 1 MPMGy916.282g2
1 2 RPCI31.33m10
...
21 7 RPCI31.17n14
21 8 WIBRy933.106h8
21 9 RPCI31.17i17
```

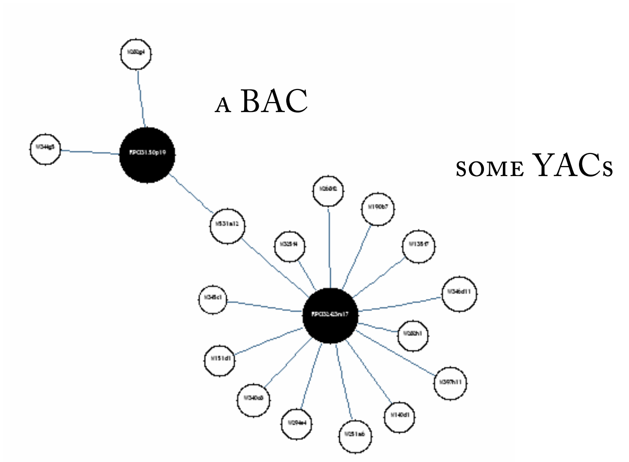
* format data to STDOUT

Step 1 – Fetch – Perl Makes it Fun



No	Clu	Cont	Lib.Clone	F	F cR	
748	02	1	RPCI31.64l18			MPMGy916.186d9, MPMGy916.34f11, MPMGy916.528b8
749	02	1	RPCI31.63c10			MPMGy916.186d9, MPMGy916.345h7, MPMGy916.34f11, MPMGy916.528b8
750	02	1	RPCI32.432g2			MPMGy916.113f2, MPMGy916.16e3, MPMGy916.191c6, MPMGy916.195f6, MPMGy916.278b6, MPMGy916.
751				d2rat1	0.0	
752	02	1	RPCI32.406p4			MPMGy916.113f2, MPMGy916.16e3, MPMGy916.191c6, MPMGy916.195f6, MPMGy916.278b6, MPMGy916.
753	02	1	RPCI31.49d1			MPMGy916.113g2, MPMGy916.146d8, MPMGy916.172b7, MPMGy916.204a1, MPMGy916.271e12, MPMGy9
754	02	1	RPCI32.423c7			MPMGy916.112f1, MPMGy916.113f2, MPMGy916.146d8, MPMGy916.172b7, MPMGy916.186g6, MPMGy916
755	02	1	RPCI31.168g3			MPMGy916.112f1, MPMGy916.113g2, MPMGy916.146d8, MPMGy916.172b7, MPMGy916.204a1, MPMGy91
756	02	1	RPCI32.368g3			MPMGy916.113g2, MPMGy916.135d3, MPMGy916.172b7, MPMGy916.219g3, MPMGy916.265b11, MPMGy9
757	02	1	RPCI32.420b8			MPMGy916.113g2, MPMGy916.135d3, MPMGy916.172b7, MPMGy916.219g3, MPMGy916.265b11, MPMGy9
758	02	1	RPCI32.420c8			MPMGy916.113g2, MPMGy916.135d3, MPMGy916.172b7, MPMGy916.219g3, MPMGy916.265b11, MPMGy9

- 1 BAC associated with many YACs
- want to extract the list of YACs associated with each BAC
  - BACa -> YAC1,YAC2,YAC3,…,YACm
  - BACb -> YAC2,YAC3,YAC5,…,YACn
- examine linking relationships



RELATIONSHIP BETWEEN OUR DATA

LWP::Simple

- It's very easy to grab a remote web page.

10 GOOD THING

```
use LWP::Simple;
my $url  = "http://www.mdc-berlin.de/ratgenome/data/MDC-Map-15.html";
my $html = get($url);
```

- `$html` now contains the HTML content of the web page

```
HTML><HEAD><TITLE>MDC-Rat-Data</TITLE></HEAD>
<BODY scroll=yes>
<H1>Physical Mapping Data, Nov/01/2002</H1>
<P>Download: <A href="http://flipper.molgen.mpg.de:10085/mdcRATdata/MDC-
RatDataSet.tsv">MDC-RatDataSet.tsv</A>&nbsp;&nbsp;&nbsp;(TAB separated values,
including RH-vectors, 1.8 MB)</P><BR>
<H3><U>Legend:</U></H3>
<TABLE border=0>
<TBODY>
<TR>
<TD><B>No </B></TD>
<TD>- consecutive number<BR></TD></TR>
<TR>
```

Parsing HTML – HTML::TreeBuilder

- Never parse HTML with your own code, unless you have a good reason. Use existing parser modules.

10 GOOD THING

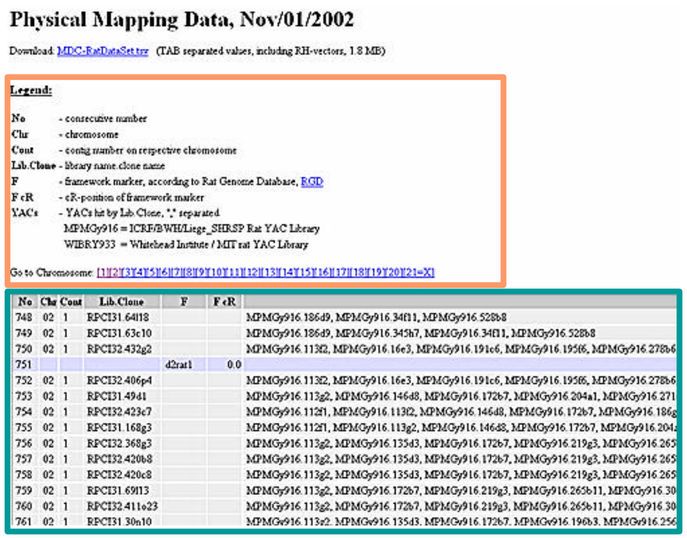
```
use HTML::TreeBuilder;
my $tree = HTML::TreeBuilder->new_from_content($html);
```

- `$tree` is an object which you can traverse

- you have to know what you're looking for

Examine HTML – Brittle!

```
<TABLE border=0>
<TBODY>
<TR>
<TD><B>No </B></TD>
<TD>- consecutive number<BR></TD></TR>
<TR>
<TD><B>Chr </B></TD>
<TD>- chromosome</TD></TR>
…
</TD></TR></TBODY></TABLE>
…
<TABLE rules=none border=1><FONT size=-1>
…
<TR bgColor=#eeeeee>
<TD>&nbsp;748</TD>
<TD>&nbsp;02</TD>
<TD>&nbsp;1</TD>
<TD>&nbsp;RPCI31.64l18</TD>
<TD>&nbsp;</TD>
<TD>&nbsp;</TD>
<TD>&nbsp;MPMGy916.186d9,&nbsp;MPMGy916.34f11…
```



Physical Mapping Data, Nov/01/2002

Download: MDC-RatDataSet.txt (TAB separated values, including RH-vectors, 1.8 MB)

Legend:

No	- consecutive number
Chr	- chromosome
Cont	- contig number on respective chromosome
Lib.Clone	- library name.clone name
F	- framework marker, according to Rat Genome Database, RGD
F cR	- cR-position of framework marker

YAC's	- YACs hit by Lib.Clone, "," separated
	MPMGy916 = ICRF/BWH/Liege_SHRSP Rat YAC Library
	WIBRY933 = Whitehead Institute / MIT rat YAC Library

Go to Chromosome: [1][2][3][4][5][6][7][8][9][10][11][12][13][14][15][16][17][18][19][20][21=X]

No	Chr	Cont	Lib.Clone	F	F cR	
748	02	1	RPCI31.64l18			MPMGy916.186d9, MPMGy916.34f11, MPMGy916.528b8
749	02	1	RPCI31.63c10			MPMGy916.186d9, MPMGy916.345h7, MPMGy916.34f11, MPMGy916.528b8
750	02	1	RPCI32.432g2			MPMGy916.113f2, MPMGy916.16e3, MPMGy916.191c6, MPMGy916.195f6, MPMGy916.278b6
751				d2rat1	0.0	
752	02	1	RPCI32.406p4			MPMGy916.113f2, MPMGy916.16e3, MPMGy916.191c6, MPMGy916.195f6, MPMGy916.278b6
753	02	1	RPCI31.49d1			MPMGy916.113g2, MPMGy916.146d8, MPMGy916.172b7, MPMGy916.204a1, MPMGy916.271
754	02	1	RPCI32.423c7			MPMGy916.112f1, MPMGy916.113f2, MPMGy916.146d8, MPMGy916.172b7, MPMGy916.186g
755	02	1	RPCI31.168g3			MPMGy916.112f1, MPMGy916.113g2, MPMGy916.146d8, MPMGy916.172b7, MPMGy916.204
756	02	1	RPCI32.368g3			MPMGy916.113g2, MPMGy916.135d3, MPMGy916.172b7, MPMGy916.219g3, MPMGy916.265
757	02	1	RPCI32.420b8			MPMGy916.113g2, MPMGy916.135d3, MPMGy916.172b7, MPMGy916.219g3, MPMGy916.265
758	02	1	RPCI32.420c8			MPMGy916.113g2, MPMGy916.135d3, MPMGy916.172b7, MPMGy916.219g3, MPMGy916.265
759	02	1	RPCI31.69f13			MPMGy916.113g2, MPMGy916.172b7, MPMGy916.219g3, MPMGy916.265b11, MPMGy916.30
760	02	1	RPCI32.411o23			MPMGy916.113g2, MPMGy916.172b7, MPMGy916.219g3, MPMGy916.265b11, MPMGy916.30
761	02	1	RPCI31.30n10			MPMGy916.113g2, MPMGy916.135d3, MPMGy916.172b7, MPMGy916.196b3, MPMGy916.256

Fetch Columns from Second Table

Columns 2, 3, 6 contain data we want. Extract data and save in memory.

grep(?,@x)

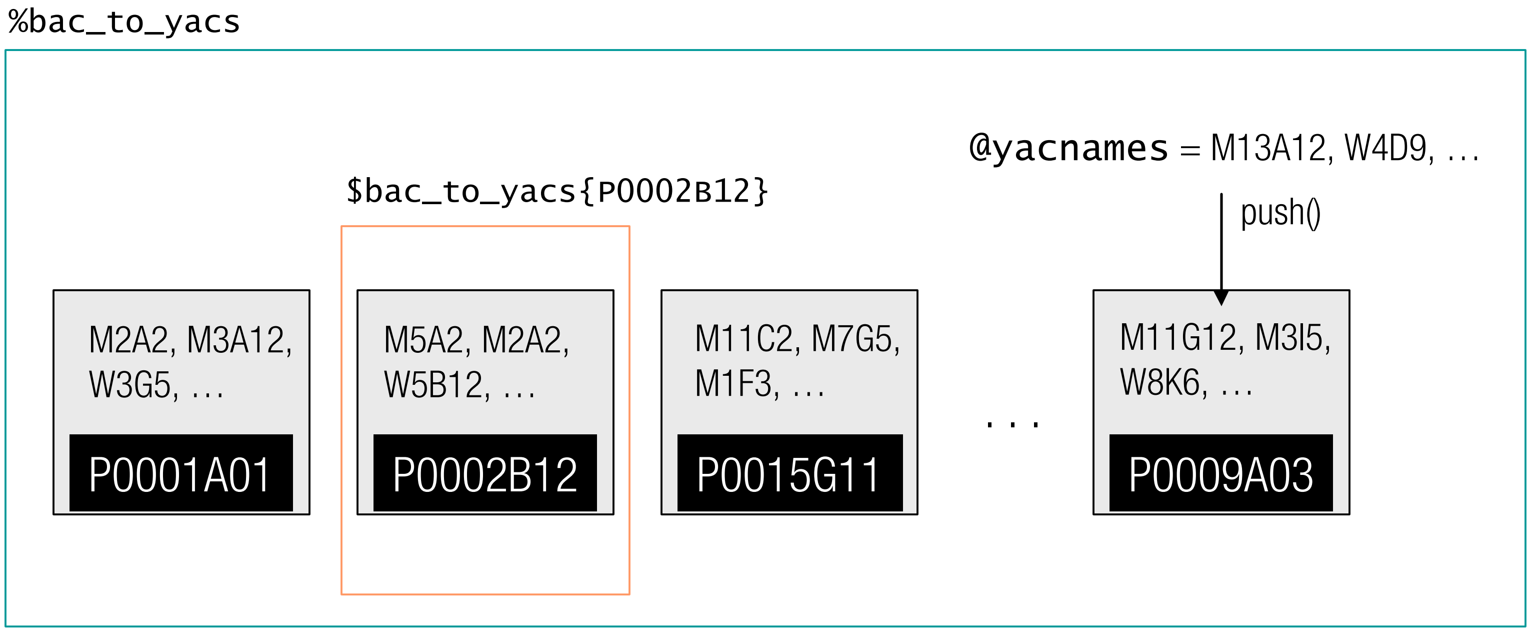
IDIOM

```
# fetch table
my ($table) = grep($_->attr("rules") eq "none", $tree->find_by_tag_name("table"));
# get all rows from table
my @rows    = $table->find_by_tag_name("tr");
# for each row…
ROW:
foreach my $row (@rows) {
  # get all columns
  my @cols = $row->find_by_tag_name("td");
  # some columns do not contain data we want
  next unless @cols == 7;
  # get data from columns 2,3,6
  my $contig   = $cols[2]->as_text;
  my $bacname  = $cols[3]->as_textl
  my $yacnames = $cols[6]->as_text;
  # split YAC names a,b,c,d -> (a b c d)
  my @yacnames = split(/,/,$yacnames);
  # save data in a hash of lists
  push ( @{$bac_to_yacs{$bacname}}, @yacnames );
}
```

Hashes and Arrays

```
my $bacname  = $cols[3]->as_text
my $yacnames  = $cols[6]->as_text;
my @yacnames = split(/,/,$yacnames);

push ( @{$bac_to_yacs{$bacname}}, @yacnames );
```



Step 2 – Munge - Perl Makes It Easy

Store data in a way that allows you to easily find needed relationships – choose wisely

- BAC -> list all associated YACs
  - `@list = @{$bac_to_yac{$bacname}}`
- BAC -> how many YACs?
  - `scalar ( @list )`
- how many total BACs?
  - `scalar ( keys %bac_to_yac )`
- how many total YACs?
  - `$num_yacs = scalar ( map { @{$bac_to_yac{$_}} keys %bac_to_yac )`
  - this sum doesn't take care of duplicates
- how many average YACs per BAC?
  - `use Math::VecStat qw(average);`
  - `average ( map { scalar ( @{$bac_to_yac{$_}} ) } keys %bac_to_yac );`

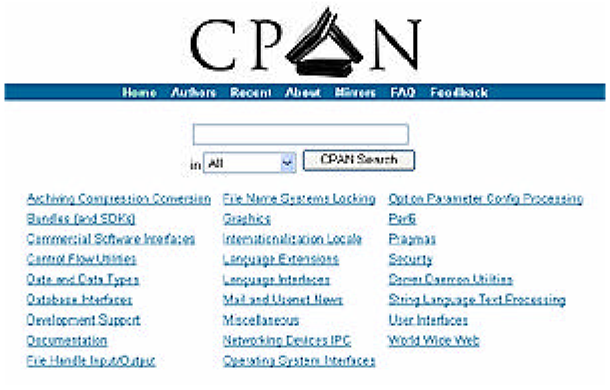
Two Problems – Part II

CPAN

- CPAN contains 5,000+ modules of all types – fun & serious
  - Perl Data Language (PDL) for matrix manipulation (PDL)
  - convert time to Swedish Chef speak (Acme::Time::Baby)

```
#!/usr/local/bin/perl
use Acme::Time::Baby language => "swedish chef";
print babytime "5:35";


Zee beeg hund is un zee sefen und zee little hund
is un zee six. Bork, bork, bork!
```

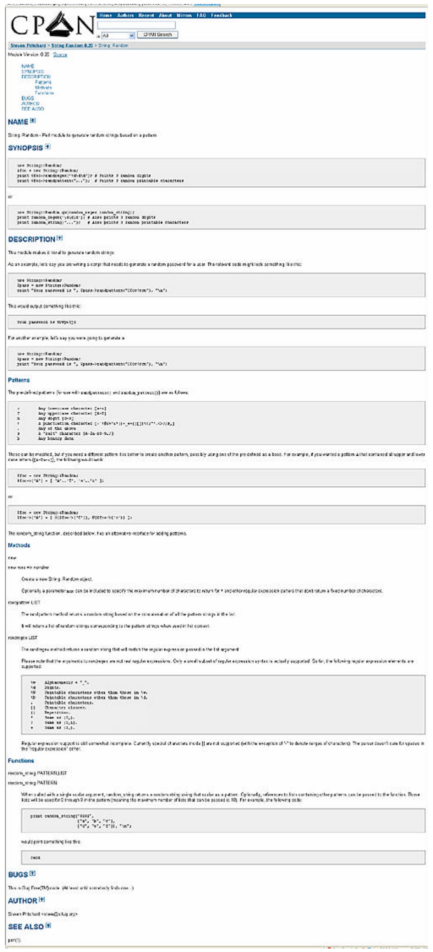


SEARCH.CPAN.ORG

- Graph::Base to create directed and undirected graphs

- GraphViz to generate GIF/TXT/EPS/PNG/…s from graph

Standardized Module Documentation


String::Random

Math::VecStat

NAME

```
Grinder - grinds coffee
```

SYNOPSIS

```
use Grinder;

$g = Grinder->new();
$g->grind("coarse");
$g->empty();
```

DESCRIPTION

Models a Rancillio burr coffee grinder

HISTORY

9 October 2003 - docs

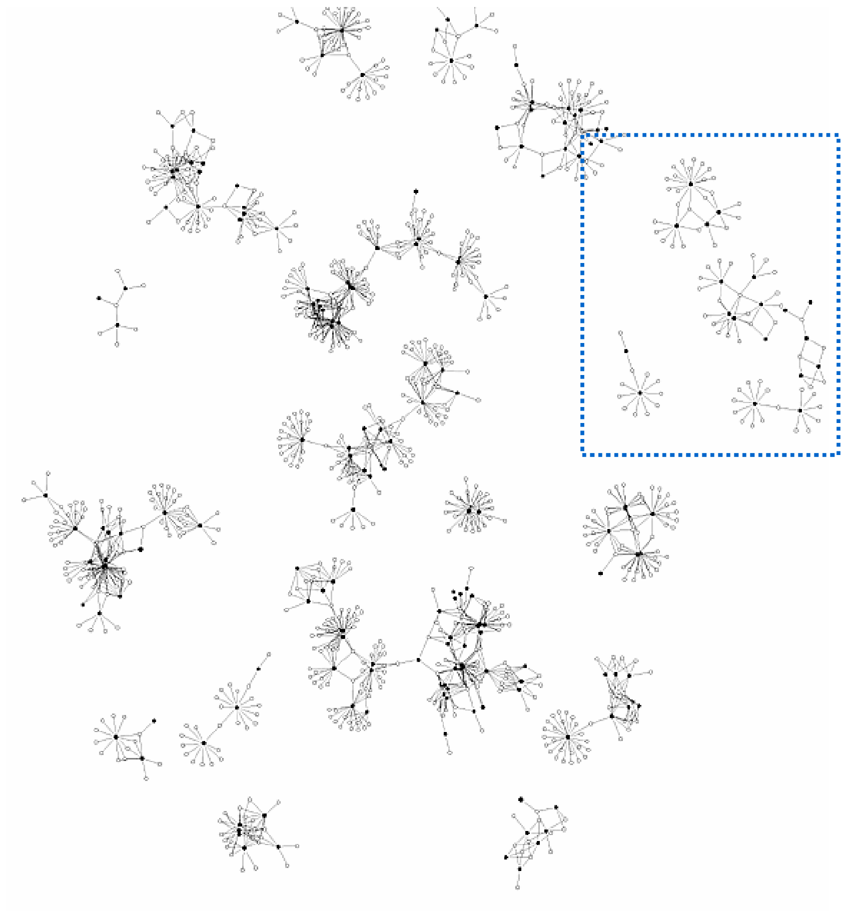
BUGS

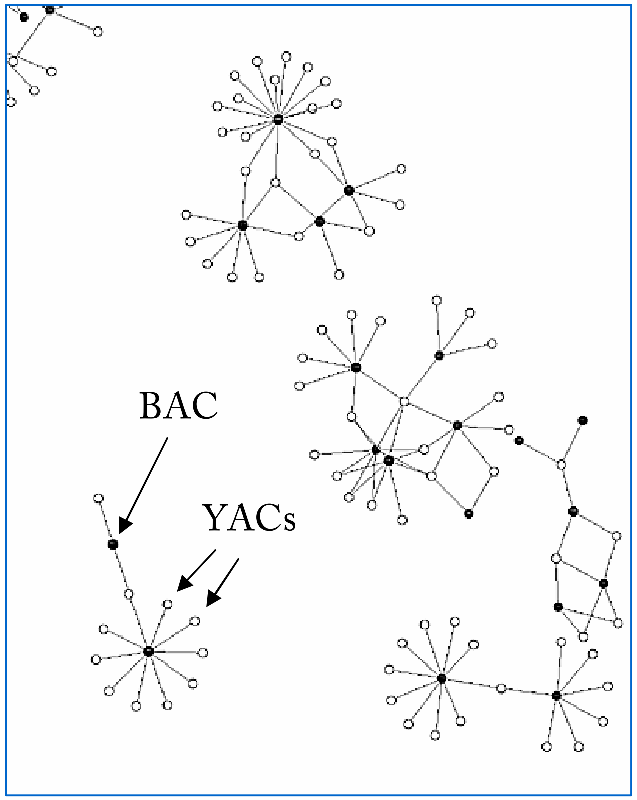
If found, remove from grinder

AUTHOR

M Krzywinski

GraphViz – Big Bang for Little Buck





Creating Graphs with Graph:: and GraphViz

map {} @x

IDIOM

BAD USAGE

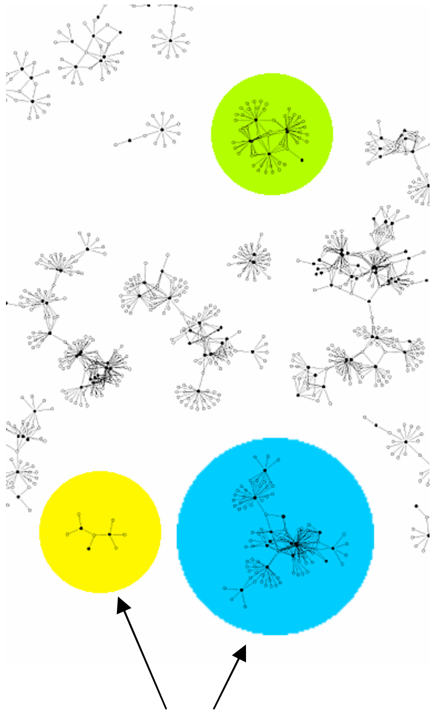
```
my $graph    = Graph::Undirected->new();
my $graphviz = GraphViz->new(directed=>0);

# for each BAC in the hash
foreach my $bac (keys %bac_to_yacs) {
    # get a list of all YACs for this BAC
    my @yacs = @{$bac_to_yacs{$bac}};
    # add edge between bac & yac in Graph::Undirected object
    map {$graph->add_edge($bac,$_) } @yacs;
    # for vizualization do the same for GraphViz object
    map { $graphviz->add_edge($bac,$_) } @yacs; # map {} IDIOM
    }
}

# create PNG image of graph
open(GRAPH,">/home/martink/www/htdocs/tmp/bacyac.png");
print GRAPH $graphviz->as_png;
close(GRAPH);
```

List Clones in Contigs

List connected components, or contigs, created by BAC-YAC links.



contig is a connected component

```
# make a list of lists which contain connected vertices
my @groups = $graph->strongly_connected_components;
# iterate through each vertex list
foreach my $group_idx (0..@groups-1) {
    # get the vertices for this list
    my @vertices = @{$groups[$group_idx]};
    # for each vertex, report the group (contig) index,
    # vertex index and name
    foreach my $vertex_idx (0..@vertices-1) {
          printf("%d %d %s\n",
                 $group_idx,
                 $vertex_idx,
                 $vertices[$vertex_idx]);
    }
}
```

Output - Create Output to STDOUT

It's nice to create output to STDOUT, rather than a file, because you can pipe your script into other processes.

```
foreach my $vertex_idx (0..@vertices-1) {
   printf("%d %d %s\n",
          $group_idx,
          $vertex_idx,
          $vertices[$vertex_idx]);
}
```

```
0 0 RPCI31.80h3
0 1 MPMGy916.380h8
0 2 WIBRy933.259d6
...
0 45 WIBRy933.284b4
0 46 WIBRy933.219c12
0 47 MPMGy916.110d1
1 0 MPMGy916.369g12
1 1 MPMGy916.282g2
1 2 RPCI31.33m10
...
21 7 RPCI31.17n14
21 8 WIBRy933.106h8
21 9 RPCI31.17i17
```

CONTIG

CONTIG CLONE INDEX

CLONE NAME

- Perl is friendly – you can copy file handles
  - STDOUT to file
  - file to STDOUT

Munge at Prompt

Don't forget that the command prompt offers powerful tools to manipulate and extract data – generate maximally detailed reports and parse later

```
0 0 RPCI31.80h3
0 1 MPMGy916.380h8
0 2 WIBRy933.259d6
...
0 45 WIBRy933.284b4
0 46 WIBRy933.219c12
0 47 MPMGy916.110d1
1 0 MPMGy916.369g12
1 1 MPMGy916.282g2
1 2 RPCI31.33m10
...
21 7 RPCI31.17n14
21 8 WIBRy933.106h8
21 9 RPCI31.17i17
```

- how many contigs?
  - cut –d " " –f 1 data.txt | sort –u | wc
- how many clones?
  - cut –d " " –f 3 data.txt | sort –u | wc
- how many clones in contig 10?
  - grep –d "^10 " data.txt | wc
- which contigs have < 20 clones?
  - cut –d " " –f 1 data.txt | uniq –c | egrep " 1?[0-9] "

```
clones     contig          clones     contig
    16     13                  11     18
    18     14                   8     19
    18     15                   9     20
    13     16                  10     21
```

Two Problems – Part II

Perl

PRODUCTIVE

CREATIVE

LINGUAL

COMPACT

OPEN SOURCE

10 GOOD THING DOES NOT SPIT