Week 13, Lecture 25

István Albert
Bioinformatics Consulting Center
Penn State
R for high throughput data analysis

• Strengths
  – extensive statistical libraries
  – data frames → outstanding data structure
  – “loop-less” transformations and indexing
  – *it just works* on every platform with (minimal installation woes)

• Weaknesses
  – the environment seems to have some design ambiguities (*or perhaps I don’t understand it well enough - though I tried*)
  – often it is difficult to identify what an object actually is/does
  – silent errors and default behaviors
Personal observations on complex bioinformatics R packages

• Often built by novice programmers

• Usually overspecialized → work for only one type of problem and with tacit assumptions

• Riddled with extremely subtle errors and unexpected behaviors (bug or feature?)

• Many were designed to be a black box

• Exceedingly difficult to trace the chain of events and/or verify that a result is correct!
R and BioConductor

About Bioconductor

Bioconductor provides tools for the analysis and comprehension of high-throughput genomic data. Bioconductor uses the R statistical programming language, and is open source and open development. It has two releases each year, 517 software packages, and an active user community.

Use Bioconductor for...

- **Microarrays**
  Import Affymetrix, Illumina, Nimblegen, Agilent, and other platforms. Perform quality assessment, normalization, differential expression, clustering, classification, gene set enrichment, genetical genomics and other workflows for expression, exon, copy number, SNP, methylation and other assays. Access GEO, ArrayExpress, Biomart, UCSC, and other community resources.

- **Sequence Data**
  Import fasta, fastq, ELAND, MAQ, BWA, Bowtie, BAM, gff, bed, wig, and other sequence formats. Trim, transform, align, and manipulate sequences. Perform quality assessment, ChIP-seq, differential expression, RNA-seq, and other workflows. Access the Sequence Read Archive.

- **Annotation**
  Use microarray probe, gene, pathway, gene ontology, homology and other annotations. Access GO, KEGG, NCBI, Biomart, UCSC, vendor, and other sources.

**Microarrays**

**Sequence Data**

**Annotation**

**High Throughput Assays**

Import, transform, edit, analyze and visualize flow cytometric, mass spec, HTqPCR, cell-based, and other assays.

Mailing Lists

Re: Attribute in snp mart to get SNP
27 minutes ago

Events

**Advanced R Programming**
28 - 29 November 2011 — Heidelberg, Germany

Bioconductor 2.0 released
Following the usual 6-month cycle, the Bioconductor community released Bioconductor 2.0 on November 1st, 2011.
Particular strengths of R

- processing/filtering/visualizing datasets stored in a delimited (row x column) formats
- at some point most of our results will be in such format
- processing data that easily fits into memory
- R is great for generic data analysis
- Rstudio is an environment around R that makes facilitates R use.
Only if you know what an object is, can you find out what it does.

- \texttt{? something} \rightarrow \texttt{help} \\

Every data in R has:

- \texttt{mode()} \rightarrow \text{object storage mode} \\
- \texttt{class()} \rightarrow \text{object class} \\
- \texttt{typeof()} \rightarrow \text{internal storage mode} \\

Data collection classes: \texttt{vector, matrix, list, table, data.frame}
• all elements of a vectors will be of the same mode.
• R will not tell you what an object is, you have to ask `is.vector`, `is.list`
• indices may be named and the names may be changed later
Factors represent categorical data. Strings in a file are usually loaded directly as factors.

```r
a = c("up", "down", "up", "up")
b = factor(a)
a
[1] "up"   "down" "up"   "up"
b
[1] up  down up  up
Levels: down up
mode(a)
[1] "character"
mode(b)
[1] "numeric"
is.factor(a)
[1] FALSE
is.factor(b)
[1] TRUE
class(a)
[1] "character"
class(b)
[1] "factor"
```
Data Frames
the workhorse of data analysis

All the way back to annot.gff that contains all annotations but not the FASTA sequence

```r
# reading the annotations
gff = read.delim('annot.gff', comment.char="#", header=FALSE)

# naming the columns
names(gff) = c('chrom', 'source', 'type', 'start', 'end', 'value',
               'strand', 'phase', 'attrib')

> head(gff$chrom)
[1] chr01 chr01 chr01 chr01 chr01 chr01
18 Levels: 2-micron chr01 chr02 chr03 chr04 chr05 chr06 chr07 chr08 ... chrmt

> head(gff$type)
[1] chromosome  telomeric_repeat
[3] telomere     X_element_combinatorial_repeat
[5] gene         CDS
36 Levels: ARS binding_site CDS centromere ... Y_prime_element

> head(gff$start)
[1] 1 1 1 63 335 335
```
Summary: output depends on data type

```r
> summary(gff)

    chrom     source             type         start
chr04  :2015  landmark:   78  CDS             :7054  Min.  :   1
chr12  :1490   SGD     :16587  gene          :6607  1st Qu.:178214
chr07  :1483     gene     :6607  Median : 391311
chr15  :1416  noncoding_exon:  480  Mean  : 444533
chr13  :1276     gene     :376  3rd Qu.: 651072
chr16  :1274  long_terminal_repeat: 383  Max. :1531900
(chr7) :7711     (Other) :1428

        end     value      strand  phase
Min.  :   34  :16665 -7947 -9611
1st Qu.:180310  .:355  0:6859
Median : 393180  +:8363 1: 82
Mean   : 446458                                2: 113
3rd Qu.: 652439
Max.   :1531933

attrib
Parent=Q0045_mRNA;Name=Q0045_CDS;orf_classification=Verified   :  8
Parent=Q0045_mRNA;Name=Q0045_intron;orf_classification=Verified:  7
Parent=Q0105_mRNA;Name=Q0105_CDS;orf_classification=Verified   :  6
Parent=Q0070_mRNA;Name=Q0070_CDS;orf_classification=Verified   :  5
Parent=Q0105_mRNA;Name=Q0105_intron;orf_classification=Verified:  5
```
Indexing vectors, sub-setting data frames

```r
> one.mill = gff$start[ gff $ start > 1000000 ]
> length(one.mill)
[1] 1033
> length(gff$start)
[1] 16665
>
> # slicing data frames
> res1 = gff[ gff $ start > 1000000, ]
> dim(res1)
[1] 1033   9
>
> # a better slicing that deals with missing values
> res2 = subset(gff, start > 1000000)
> dim(res2)
[1] 1033   9
> ```
Sub-select for genes

```r
> # select rows that contain gene type
genes = subset(gff, type == "gene")

> dim(genes)
[1] 6607 9

> # write the output to a file
write.table(genes, "genes.gff", col.names=FALSE)
```

We want to match to keep verified strings but the default R string manipulation is lacking.

Install `stringr`

```r
> install.packages('stringr')
```

Few if any other programming or data analysis environments have installation processes that are so simple!
Hadley Wickham to the rescue

This is the website of Hadley Wickham. I'm an Assistant Professor of Statistics at Rice University, interested in interactive and dynamic graphics, in developing practical tools for data analysis, and in gaining better understanding of complex statistical models through visualisation. In July 2008, I completed my PhD in statistics at Iowa State University with Di Cook and Helke Hofmann.

Professional resources

- My academic vita
- My academic portfolio
- Short courses
- My thesis: practical tools for exploring data and models

Teaching

- stat645: Data visualisation. Rice University, Spring 2011.
- stat310: Introduction to probability and mathematical statistics. Rice University, Spring 2011 (previously taught Spring 10 and Spring 09).
- stat405: Statistical computing. Rice University, Fall 2009. (Previously taught in Fall 2008.

Amazing R packages that change the way we operate on data

stringr
reshape
ggplot2
Sub-selecting verified genes from the annotation file

```r
# load up the string matching library
library(stringr)

# create a new column on gff to be verified
gff$verified = str_detect(gff$attrib, 'Verified')

# filter for genes
genes = subset(gff, type == "gene")

# filter for verified genes
genes = subset(gff, verified == TRUE)

# delete the verified column
genes$verified = NULL

# write the file
write.table(genes, "genes.gff", col.names=FALSE)
```

There are probably many-many other ways to get the same result – shorter, longer, more or less explicit, subtle and not so subtle.
Grouping by Factors

```r
# read the annotation
 gff = read.delim('annot.gff', comment.char="#", header=FALSE)

# add the column names
 names(gff) = c('chrom', 'source', 'type', 'start', 'end', 'value',
               'strand', 'phase','attrib')

# split the data by the factor stored in column 
# type and run summary on each subgroup
 res = by(gff, gff$type, summary)

> names(res)
[1] "ARS"        "CDS"        "centromere_DNA_Element_I"
[5] "centromere_DNA_Element_II" "centromere_DNA_Element_III"
[9] "external_transcribed_sp"
[11] "gene"       "insertion"   "intron"
[13] "LTR_retrotransposon"

> class(res$ARS)
[1] "table"
> mode(res$ARS)
[1] "character"
> res$ARS[1:6]
  [1] "chr04 : 40    "chr07 : 28    "chr16 : 27    
```
• Using R compute the following

  – median length of tRNA elements (create new column with lengths, then summarize via by then summarize)
  – save genes with a length of less than 10kb to a different file