Week 15, Lecture 30

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

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

• Weaknesses
  – the environment seems to have some design ambiguities
  – often it is difficult to identify what an object actually is/does
  – silent errors and default behaviors
  – over the decades it has accumulated lots of odd behaviors
Personal observations on bioinformatics R packages

• Often built by novice programmers

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

• Are 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, 749 software packages, and an active user community. Bioconductor is also available as an Amazon Machine Image (AMI).

Use Bioconductor for...

- **Variants**
  Read and write VCF files. Identify structural location of variants and compute amino acid coding changes for non-synonymous variants. Use SIFT and PolyPhen database packages to predict consequence of amino acid coding changes.

- **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.

Recent Courses

Explore material from recent courses, including BioC2013, userR! 2013, CSAMA 2013, Intermediate R / Bioconductor for High Throughput Sequence Analysis.

Counting Reads for Differential Expression

The parathyroidSE ExperimentData package and vignette illustrates how to count reads and perform other common operations required for differential expression analysis.
R hero: Hadley Wickham

had.co.nz

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 Heike Hofmann.

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

Teaching
- stat545: 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 08 and Fall 07.)

Amazing R packages that change the way we operate on data:
- stringr
- reshape
- ggplot2
What if I told you that you have to pick a side?

Computational tools have an effect on us.

When we pick a tool it will change how we think about problems.
Usually people pick one and stick with it

- Python, awk, command line tools, shell – is like being an ant building a solution from small pieces

- You can also use R for all of the above but it will be like re-engineering an existing building into a different one

*What you probably cannot do easily is switch between the approaches.*
Understand early on: R is idiosyncratic

• There are features that make little sense
• There are operations that are greatly counterintuitive
• There are fundamental flaws in the language
• No one has authority to fix the above

Many (very smart) people deeply dislike this state and therefore avoid R as plague.

Coping mechanism: “That’s the way the cookie crumbles”
Particular strengths of R

• processing/filtering/visualizing datasets stored in a TABULAR (row x column) format

• Processing data that easily fits into memory

• Plotting and visualizing data

• Rstudio is a environment around R
Discovery: if you know what an object is, can you find out what it does?

- `? something` → `help`

Every object in R has a:

- `mode()` → object storage mode
- `class()` → object class
- `typeof()` → internal storage mode

Data representation classes: vector, matrix, list, table, data.frame
Data Frames: workhorse of data analysis

Other people are so envious of the data frame that they implemented them in other languages **Python → Panda (Python Data Analysis Library)**

```
# the data frame is a data representation format
annot = read.delim("annot.gff", comment.char="#", header=FALSE)

# show the names associated with the columns
names(annot)

# you can look at these columns
head(annot$V1)

# you may also assign your own names to these columns
names(annot) <- c("chrom", "source", "type", "start", "end", "value", "strand", "phase", "attr")

# now you can access columns by name
annot$start
```
Data Frames are the workhorses of biological data analysis

# now you can access columns by name
annot$start

# set the maximal print to 10 rows
options(max.print=100)

# slice rows, columns
annot[1:3]
annot[1:3, 1:2]

# look at this object anno
annot$chrom

# investigate the objects
class(annot$chrom)
mode(annot$chrom)
typeof(annot$chrom)

# look at head or tail
head(annot$chrom)
• all elements of a vector 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

```r
> a = 1
> mode(a)
[1] "numeric"
> # combine values
> b = c(1, 2, 3)
> mode(b)
[1] "numeric"
> is.vector(a)
[1] TRUE
> is.vector(b)
[1] TRUE
> c = c(1, 2, "3")
> mode(c)
[1] "character"

> # more combinations
> a = c(one=1, two=2, three=3)
> a
one  two  three
1     2     3
> a['one']
one
    1
> names(a)
[1] "one"  "two"  "three"
> names(a)[1:2] = c("unos", "dos")
> a
unos   dos   three
    1     2     3
> a['unos']
unos
    1
```
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"
```
Summary: output depends on data type

```r
> summary(gff)

              chr  source      type       start
chr04:2015   landmark:    78   CDS       :7054  Min.  :   1
chr12:1490   SGD          :16587 gene     :6607  1st Qu.: 178214
chr07:1483   noncoding_exon:  480  Median : 391311
chr15:1416   long_terminal_repeat: 383  Mean   : 444533
chr13:1276   intron       : 376  3rd Qu.:  651072
chr16:1274   ARS         : 337  Max.   : 1531900
(chrOther):7711 (Other):1428

                  end       value      strand     phase
Min.    :44       :16665      -:7947      :9611
1st Qu. :180610   .: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
```
We want to match to keep verified strings but the default R string manipulation is lacking.

Install `stringr`

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

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

```r
> names(res)
[1] "ARS"       "CDS"       "centromere_DNA_Element_I" "centromere_DNA_Element_II"
[7] "centromere_DNA_Element_III" "external_transcribed_sp"
[9] "gene"      "insertion"  "intron"     "LTR_retrotransposon"

> class(res$ARS)
[1] "table"
> mode(res$ARS)
[1] "character"
> res$ARS[1:6]
[1] "chr04 : 40"  "chr07 : 28"  "chr16 : 27"
```
Visualizing high dimensionality data

by Hadley Wickham: http://had.co.nz/

ggplot2

ggplot2 is a plotting system for R, based on the grammar of graphics, which tries to take the good parts of base and lattice graphics and none of the bad parts. It takes care of many of the fiddly details that make plotting a hassle (like drawing legends) as well as providing a powerful model of graphics that makes it easy to produce complex multi-layered graphics.

A copy of this site for local use is available here, as a 6 meg zip file. To use it, unzip and open the index.html page.

Search the site: Google Custom Search Search

Mailing list

There is nothing like it in any programming environment!

Parts of this presentation follow the tutorial of ggplot2
This is a pie chart describing my favorite bars.

And this is a bar graph describing my favorite pies.
Chapter 2

Getting started with qplot

2.1 Introduction

In this chapter, you will learn to make a wide variety of plots with your first ggplot2 function, qplot(), short for quick plot. qplot makes it easy to produce complex plots, often requiring several lines of code using other plotting systems, in one line. qplot() can do this because it’s based on the grammar of graphics, which allows you to create a simple, yet expressive, description of the plot. In later chapters you’ll learn to use all of the expressive power of the grammar, but here we’ll start simple so you can work your way up. You will also start to learn some of the ggplot2 terminology that will be used throughout the book.

qplot has been designed to be very similar to plot, which should make it easy if you’re already familiar with plotting in R. Remember, during an R session you can get a summary of all the arguments to qplot with R help, ?qplot.

In this chapter you’ll learn:

- The basic use of qplot—If you’re already familiar with plot, this will be trivial.

We will start out with example plots from this manual

Then at the end we generate a peak distribution plot around gene starts sites.

Install ggplot2

```
1 # install the package first
2 # this needs to be done a single time only
3 install.packages('ggplot2')
4
5 # load the ggplot 2 library
6 # this needs to be done once per session only
7 library(ggplot2)
8
9 # some libraries come with test data
10 # ggplot has a dataset called diamonds
11 # you can see the content as a file by executing
12 write.table(diamonds, 'diamonds.txt', sep="\t", quote=FALSE)
13
14 # you can get some information on the data with
15 ?diamonds
16```
<table>
<thead>
<tr>
<th>carat</th>
<th>cut</th>
<th>color</th>
<th>clarity</th>
<th>depth</th>
<th>table</th>
<th>price</th>
<th>x</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.23</td>
<td>Ideal</td>
<td>SI2</td>
<td>61.5</td>
<td>55</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0.21</td>
<td>Premium</td>
<td>SI1</td>
<td>59.8</td>
<td>61</td>
<td>326</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0.23</td>
<td>Good</td>
<td>VS1</td>
<td>56.9</td>
<td>65</td>
<td>327</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0.29</td>
<td>Premium</td>
<td>VS2</td>
<td>62.4</td>
<td>58</td>
<td>334</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0.31</td>
<td>Good</td>
<td>SI2</td>
<td>63.3</td>
<td>58</td>
<td>335</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0.24</td>
<td>Very Good</td>
<td>J</td>
<td>VVS2</td>
<td>62.8</td>
<td>57</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0.24</td>
<td>Very Good</td>
<td>I</td>
<td>VVS1</td>
<td>62.3</td>
<td>57</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>0.26</td>
<td>Very Good</td>
<td>H</td>
<td>SI1</td>
<td>61.9</td>
<td>55</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>0.22</td>
<td>Fair</td>
<td>VS2</td>
<td>65.1</td>
<td>61</td>
<td>337</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>0.23</td>
<td>Very Good</td>
<td>H</td>
<td>VS1</td>
<td>59.4</td>
<td>61</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>0.3</td>
<td>Good</td>
<td>SI1</td>
<td>64</td>
<td>55</td>
<td>339</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>0.23</td>
<td>Ideal</td>
<td>VS1</td>
<td>62.8</td>
<td>56</td>
<td>340</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>0.22</td>
<td>Premium</td>
<td>SI1</td>
<td>60.4</td>
<td>61</td>
<td>342</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>0.31</td>
<td>Ideal</td>
<td>SI2</td>
<td>62.2</td>
<td>54</td>
<td>344</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>0.2</td>
<td>Premium</td>
<td>SI2</td>
<td>60.2</td>
<td>62</td>
<td>345</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>0.32</td>
<td>Premium</td>
<td>T1</td>
<td>60.9</td>
<td>58</td>
<td>345</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>
NOTE
For the next few slides
I will be changing only line 10
(sometimes we use all data or just the small data)
9  # plot by different symbols
10  res = qplot(carat, price, data=small, shape=cut)

9  # plot by different colors
10  res = qplot(carat, price, data=small, color=cut)
# change the geometry to bargraph
```r
res = qplot(color, data=diamonds, geom="bar")
```

# add transparency to deal with overplotting
```r
res = qplot(price, carat, data=diamonds, alpha=I(1/20))
```
# histograms of carats

res = qplot(carat, data = diamonds, geom = "histogram")

res = qplot(carat, data=diamonds, geom="density", fill=color)
# a histogram within certain limits and small binsize

```r
res = qplot(carat, data=diamonds, geom="histogram", binwidth=0.01, xlim=c(0, 3))
```

# add a smooth linear geometry

```r
res = qplot(carat, price, data=small, geom=c("point", "smooth"))
```
# add a boxplot geometry
res = qplot(color, carat/price, data=diamonds, geom="boxplot")

# add a jitter geometry with transparency (alpha)
res = qplot(color, carat/price, data=diamonds, geom="jitter", alpha=I(1/10))
ggplot2 concepts

• geometry $\rightarrow$ what plot looks like

• faceting $\rightarrow$ how many plots/panels

• statistics $\rightarrow$ transformation on the data

• positioning $\rightarrow$ fine tunes locations in the plot

• scales $\rightarrow$ maps data to an x,y coordinate
Faceting - multiplots

```r
# faceting by colors
res = qplot(carat, data=diamonds, facets=color~., binwidth=0.1, xlim=c(0,4))
```
Faceting and shapes and colors

```r
# two panel plot price vs carat colored by column color
# with transparency
res = qplot(carat, price, data=diamonds,
            facets=cut~clarity, color=color, alpha=I(1/10))
```
scripts are in supporting data located in the 26.tar.gz file on the website

```r
# Transform my annot.gff to a more user friendly format
#
# load the annotation data
gff = read.table("annot.gff", header=FALSE, comment.char="#")
#
# prepare the annotation data
names(gff) = c('chrom', 'source', 'type', 'start',
  'end', 'value', 'strand', 'phase', 'attrib')

# get rid of the attribute column
attrib = NULL

# add a new column called size
size = gff$end - gff$start

# keep only certain columns
keep = c("gene", "binding_site", "tRNA", "long_terminal_repeat")
gff = subset(gff, gff$type %in% keep)

# save this data as a new file
write.table(gff, 'annot.txt', sep="\t", quote=FALSE)

# reload the file to check its properties
gff = read.delim('annot.txt')
```
# generate the plot
```r
res = qplot(size, data=gff, xlim=c(0, 2000), facets=type ~ .)
```