Introduction to R

Richard Wang, PhD
CBI fellow | Nelson & Miceli labs

Workshop 3: Introduction to R

- **Day 2**
  - Statistical methods
    - Probability distributions
    - Random number generation
    - Common statistical tests
      - Fisher's exact test, chi square, qvalue, etc…
  - Visualization
    - Plotting functions
    - Customizing plots
    - Saving plots for publication
    - IGV
    - CummeRbund

Lists

- Ordered collection of components
- Each component can be of a different type
- Each component can
- A dataframe is actually a list with more structure

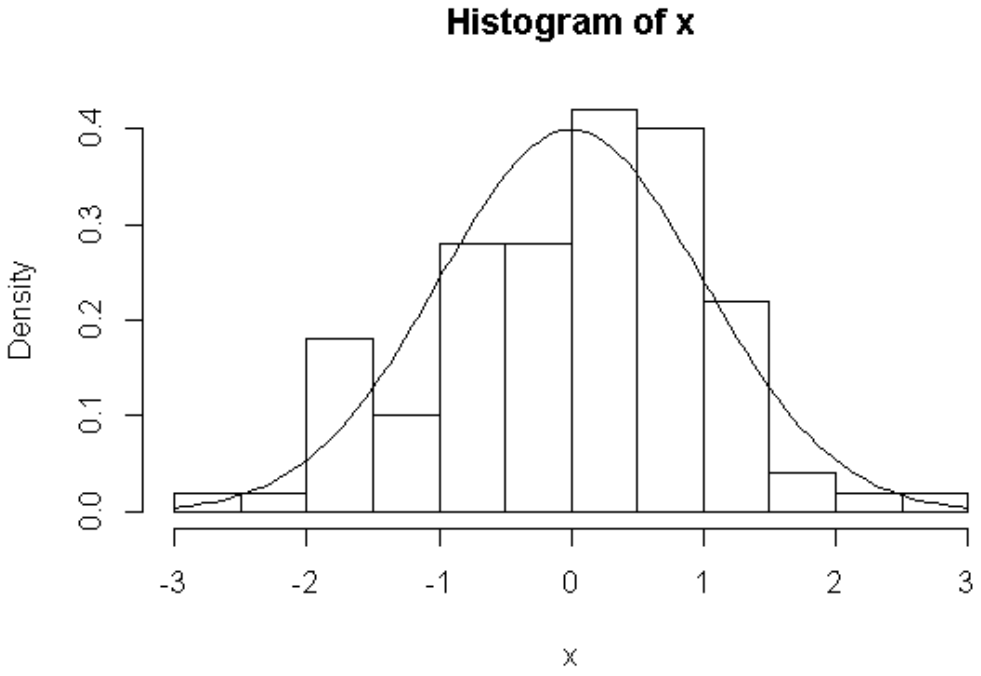
```
> lst = list(a=1:3, b="cairo", c=sqrt)
> lst[1] #get a sublist
$a
[1]  1  2  3
> lst[[1]] #returns the object of type stored in 1st position (ie numeric)
[1] 1 2 3
> lst$a
> lst["a"]]
> class(lst[1]); class(lst[[1]]); # compare the two
```

Probability distributions

- For every statistical distribution in R, these functions are available
  - cumulative distribution function
  - density
  - quantile
  - random numbers
- Example distributions
  - Gaussian or normal
  - binomial
  - poisson
  - gamma
  - beta
  - t
  - chi-square

Probability distributions

- For example in normal distribution (gaussian) there is a family for 4 functions
  - pnorm() = P(X < x)
  - dnorm() = P(x)
  - qnorm() = quantile
  - rnorm()



Statistics

- Many distributions and tests built in
- Others added as packages
- Interpreting output, names, pvalues

Basic stat functions

- summary()
- var()
- sd()
- mean()
- median()

Common statistical tests

- cor.test()
- t.test()
- fisher.test()
- chisq.test()
- wilcox.test()
- ks.test()
- p.adjust()
- regression: lm(y ~ x)
- anova: aov()

Try it out: ALL data

- x = read.table("ALL_subset.txt", header=T)
- Compare two samples by t.test
- t.test(x[,1], x[,2])
- Compute the correlation between two samples
- cor.test(x[,1], x[,2])
- Compute all pairwise correlations, returns matrix
- cor(x)

A chisquare example

```
> mat = matrix( c(1463, 48486, 7892, 334758), nrow=2, ncol=2)
> mat
      [,1]   [,2]
[1,]  1463   7892
[2,] 48486 334758
> chisq.test(mat)

        Pearson's Chi-squared test with Yates' continuity
correction

data:  mat
X-squared = 73.1195, df = 1, p-value < 2.2e-16

> mat.chisq = chisq.test(mat)
```

A chisquare example

```
> mat.chisq = chisq.test(mat)
> names(res.mat)
[1] "statistic" "parameter" "p.value"   "method"    "data.name"
"observed"
[7] "expected"  "residuals" "stdres"
> res.mat$p.value
[1] 1.220305e-17
```

Simple Regression

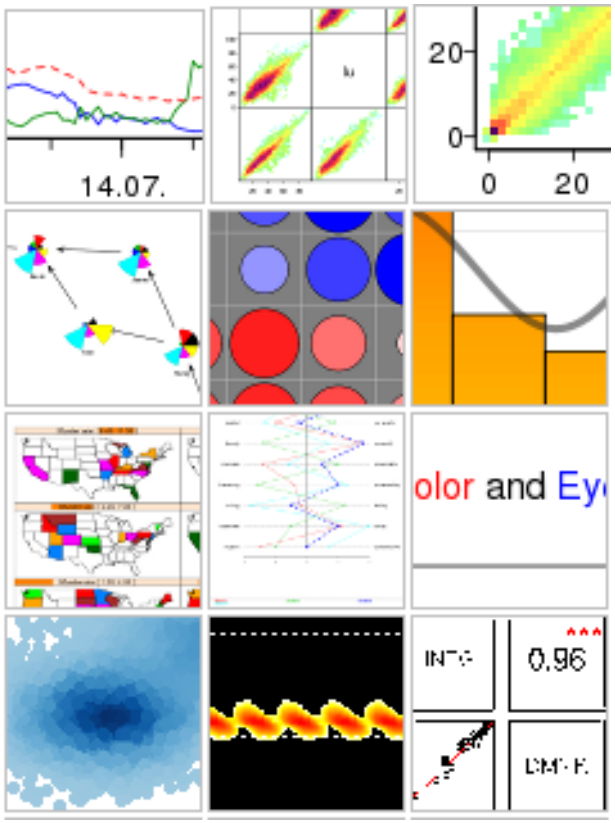
```
> data(iris)
> head(iris)
> # attach(iris) - did not run
> names(iris)
[1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"
"Species"
> lm(Sepal.Length ~ Petal.Length)


Call:
lm(formula = Sepal.Length ~ Petal.Length)


Coefficients:
 (Intercept)  Petal.Length
      4.3066        0.4089
```

Simple Regression

```
> a = lm(Sepal.Length~ Petal.Length)
> names(a)
 [1] "coefficients"  "residuals"     "effects"       "rank"          "fitted.values" "assign"        "qr"
"df.residual"   "xlevels"
[10] "call"          "terms"         "model"
> summary(a)

Call:
lm(formula = Sepal.Length ~ Petal.Length)

Residuals:
     Min       1Q   Median       3Q      Max
-1.24675 -0.29657 -0.01515  0.27676  1.00269

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)   4.30660    0.07839   54.94   <2e-16 ***
Petal.Length  0.40892    0.01889   21.65   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.4071 on 148 degrees of freedom
Multiple R-squared:  0.76,      Adjusted R-squared: 0.7583
F-statistic: 468.6 on 1 and 148 DF,  p-value: < 2.2e-16
```

Plotting

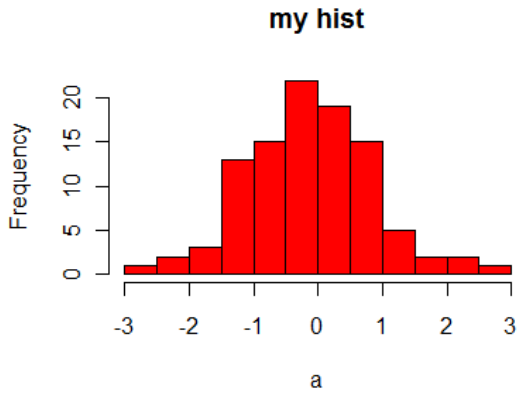
- A great strength of R is visualization
- Many type of plots:
  - line
  - histogram
  - scatter plot
  - topographical
- A great display of R graphs at http://gallery.r-enthusiasts.com/



Histogram

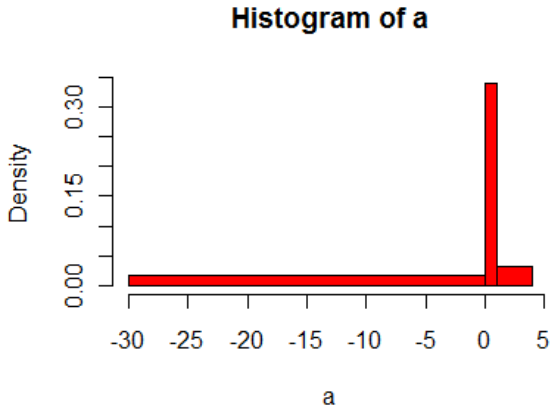
- Very useful!

```
> a = rnorm(100)
> hist(a)
> hist(a, breaks=10, col="red",
  main="my hist")
```



- breaks can be customized

```
> br = c(-30, 0, 1,4)
> br
[1] -30   0   1   4
> res =  hist(a, breaks=br, col="red")
```



Histogram

- returns an object of class "histogram"

```
> result.a = hist(a)
> class(result.a)
[1] "histogram"
> names(result.a)
[1] "breaks"      "counts"      "intensities" "density"
"mids"
[6] "xname"       "equidist"
```

Histogram

```
> result.a
$breaks
[1] -3 -2 -1  0  1  2  3
$counts
[1]  3 16 37 34  7  3
$intensities
[1] 0.03 0.16 0.37 0.34 0.07 0.03
$density
[1] 0.03 0.16 0.37 0.34 0.07 0.03
$mids
[1] -2.5 -1.5 -0.5  0.5  1.5  2.5
$xname
[1] "a"
$equidist
[1] TRUE
attr(,"class")
[1] "histogram"
>
```

Histogram

- hist() can plot counts as well as frequency
- but frequency definition may not be what you expect

```
> z = hist(c(rep(1,20), 2,3), freq=F)
> sum(z$density)
[1] 2
```

- The AUC sums to 1. Distance between breaks is 0.5

```
sum(z$density) * 0.5
> sum(z$density)*0.5
[1] 1
```

- If you want percentage, better to do this

```
br = c(0,1,2,3)
z = hist(c(rep(1,20), 2,3), breaks=br)
barplot(z$counts/sum(z$counts), names.arg=c(1,2,3))
```

Try it out

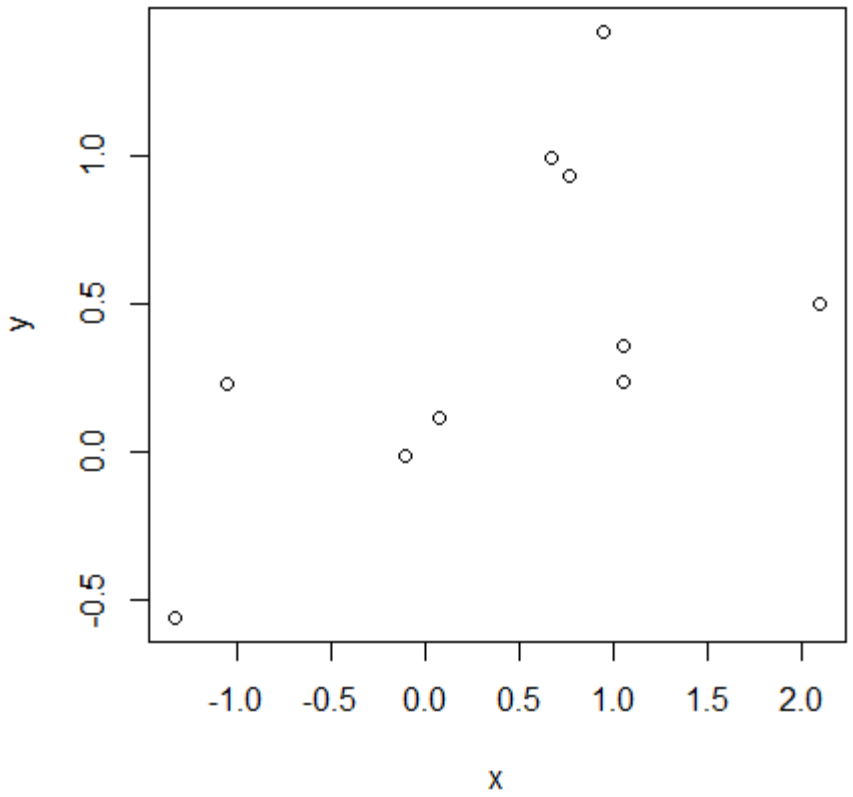
```
#Read in the ALL dataset
> x = read.table("ALL_subset.txt")
# how many expts are there? how many probes?
> dim(all)

# plot histogram
> hist(x[,1], xlim=c(0,25))
> hist(x[,2], xlim=c(0,25), col="blue", add=T)
> hist(x[,3], xlim=c(0,25), col="red", add=T)

# box plot of all expts; three expts
> boxplot(x);
> boxplot(x[, c(1,12,3)])
```

Plotting

- plots
  - histogram
  - boxplot
  - barplot
  - scatter plot
  - heatmap
  - cluster
  - points, lines, segments
  - abline
  - qqplot
  - & more
- customizations
  - axes
  - header text/title
  - margin text
  - identify
  - display symbols
  - layout
  - colors

A plain scatter plot

```
x = rnorm(10)
y = rnorm(10)
plot(x,y)
```



Graphical parameters

`adj`	controls text justification with respect to the left border of the text so that `0` is left-justified, `0.5` is centred, `1` is right-justified, values > 1 move the text further to the left, and negative values further to the right; if two values are given (e.g., `c(0, 0)`) the second one controls vertical justification with respect to the text baseline
`bg`	specifies the colour of the background (e.g., `bg="red"`, `bg="blue"`; the list of the 657 available colours is displayed with `colors()`)
`bty`	controls the type of box drawn around the plot, allowed values are: `"o"`, `"l"`, `"7"`, `"c"`, `"u"` ou `"]"` (the box looks like the corresponding character); if `bty="n"` the box is not drawn
`cex`	a value controlling the size of texts and symbols with respect to the default; the following parameters have the same control for numbers on the axes, `cex.axis`, the axis labels, `cex.lab`, the title, `cex.main`, and the sub-title, `cex.sub`
`col`	controls the colour of symbols; as for `cex` there are: `col.axis`, `col.lab`, `col.main`, `col.sub`
`font`	an integer which controls the style of text (`1`: normal, `2`: italics, `3`: bold, `4`: bold italics); as for `cex` there are: `font.axis`, `font.lab`, `font.main`, `font.sub`
`las`	an integer which controls the orientation of the axis labels (`0`: parallel to the axes, `1`: horizontal, `2`: perpendicular to the axes, `3`: vertical)
`lty`	controls the type of lines, can be an integer (`1`: solid, `2`: dashed, `3`: dotted, `4`: dotdash, `5`: longdash, `6`: twodash), or a string of up to eight characters (between `"0"` and `"9"`) which specifies alternatively the length, in points or pixels, of the drawn elements and the blanks, for example `lty="44"` will have the same effet than `lty=2`
`lwd`	a numeric which controls the width of lines
`mar`	a vector of 4 numeric values which control the space between the axes and the border of the graph of the form `c(bottom, left, top, right)`, the default values are `c(5.1, 4.1, 4.1, 2.1)`
`mfcol`	a vector of the form `c(nr,nc)` which partitions the graphic window as a matrix of `nr` lines and `nc` columns, the plots are then drawn in columns (see section 4.1.2)
`mfrow`	id. but the plots are then drawn in line (see section 4.1.2)
`pch`	controls the type of symbol, either an integer between 1 and 25, or any single character within `""` (Fig. 2)
`ps`	an integer which controls the size in points of texts and symbols

`pty`	a character which specifies the type of the plotting region, `"s"`: square, `"m"`: maximal
`tck`	a value which specifies the length of tick-marks on the axes as a fraction of the smallest of the width or height of the plot; if `tck=1` a grid is drawn
`tcl`	id. but as a fraction of the height of a line of text (by default `tcl=-0.5`)
`xaxt`	if `xaxt="n"` the x-axis is set but not drawn (useful in conjunction with `axis(side=1, ...)`)
`yaxt`	if `yaxt="n"` the y-axis is set but not drawn (useful in conjunction with `axis(side=2, ...)`)

R for Beginners

E. Paradis

Printing character

- you can specify the pch parameter
  - a number (pch=1 gives a circle)
  - a text character (pch="v" uses the letter "v")

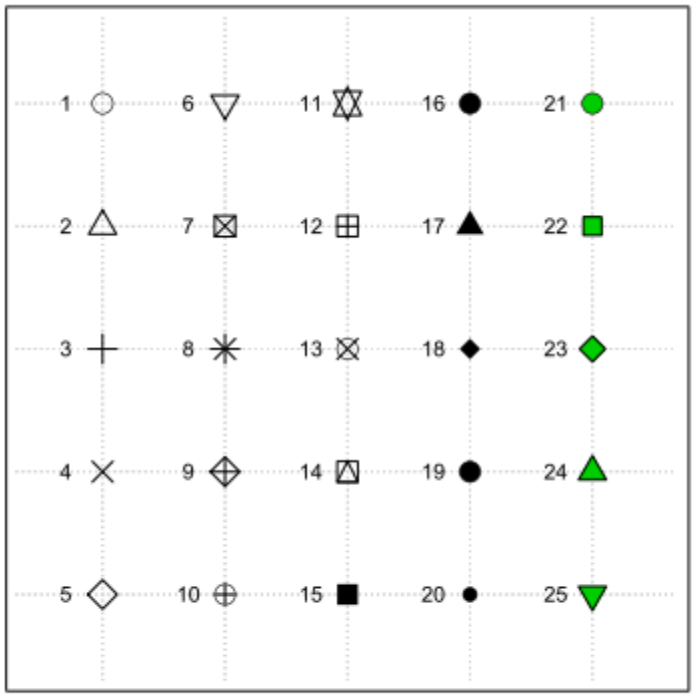


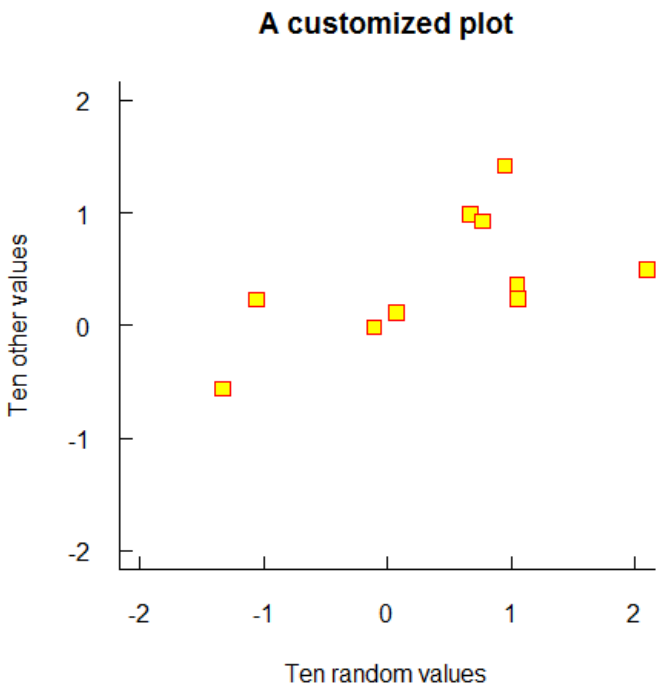
Figure 5: Plotting symbols

Try it out

```
#Read in the ALL dataset
> x = read.table("ALL_subset.txt")
# how many expts are there? how many probes?
> dim(all)
# plot the first three expts against the first on the same page
> par(mfrow=c(1,3));
> plot(x[,1], x[,2]);
> plot(x[,1], x[,3]);
> plot(x[,1], x[,4])
#close the window
> dev.off()
# plot one graph again, what's different?
plot(x[,1], x[,2])
# change some plotting options
> plot(x[,1], x[,2], pch=".", col="blue", xlab="expt1", ylab="expt2")
# add a title with the correlation in it
> x.cor = cor.test(x[,1], x[,2])
> title(paste("Expt 1 versus Expt 2\n cor=", round(x.cor$estimate, 2)))
# log transform
> plot(log2(x[,1]), log2(x[,2]), pch=".", col="blue", xlab="expt1", ylab="expt2",
main="Expt1 versus Expt2")
```

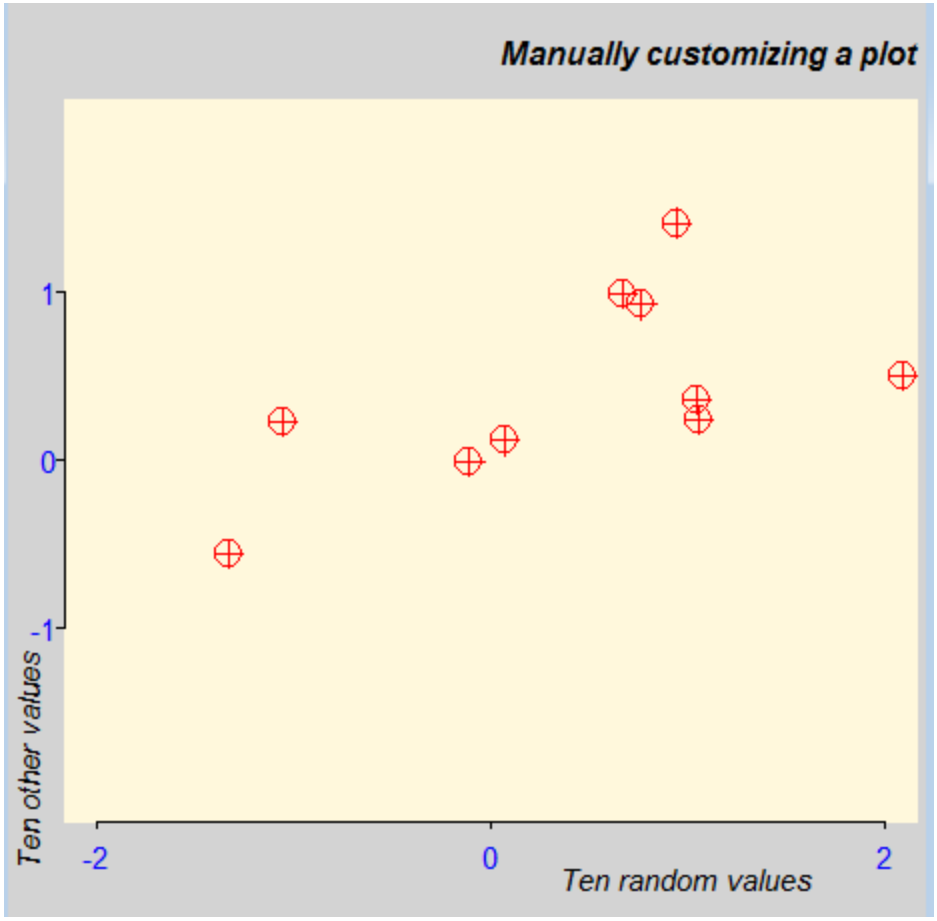
Let's add options

```
plot(x,y, xlab="Ten random values", ylab="Ten other values", xlim=c(-2,2), ylim=c(-2,2), pch=22, col="red", bg="yellow", bty="l", tcl=0.4, main="A customized plot", las=1, cex=1.5)
```



Plotting using base graphics

```
> opar = par()
> par(bg="lightgray", mar=c(2.5, 1.5, 2.5, 0.25))
> plot(x, y, type="n", xlab="", ylab="", xlim=c(-2,2), ylim=c(-2,2),
xaxt="n", yaxt="n")
> rect(-3, -3, 3, 3, col="cornsilk")
> points(x, y, pch=10, col="red", cex=2)
> axis(side=1, c(-2, 0, 2), tcl=-0.2, labels=FALSE)
> axis(side=2, -1:1, tcl=-0.2, labels=FALSE)
> title("Manually customizing a plot", font.main=4, adj=1, cex.main=1)
> mtext("Ten random values", side=1, line=1, at=1, cex=0.9, font=3)
> mtext("Ten other values", side=2, line=0.5, at=-1.8, cex=0.9, font=3)
> mtext(c(-2, 0, 2), side=1, las=1, at=c(-2,0,2), line=0.4, col="blue",
cex=0.9)
> mtext(-1:1, side=2, las=1, at=-1:1, line=0.2, col="blue", cex=0.9)
> par(opar)
```


Manually customizing a plot
1
0
-1
Ten other values
-2
0
2
Ten random values

Scatter plots, line plots

- the type parameter controls the scatter plot type
  - options include points, lines, points+lines and histogram

```
> somedata = rnorm(25)
> plot(somedata, type="p")
> plot(somedata, type="l")
> plot(somedata, type="b")
> plot(somedata, type="h")
```

A few tips

- If you have many points (100-1000's), the plot may become obscured
- Potential remedies:
  - downsample your data: suppose you have 1000 rows of data in a data frame, you can pick some indices to display

```
# this will pick 25 numbers between 1 and 100
ids = sample(1:100, 25)
```

  - try a smaller plotting symbol: pch="." will use a dot

```
plot(x[ids, 1],x[ids,2], pch=".")
```

smoothScatter

```
x1 = rnorm(100000)
y1 = rnorm(100000)
x2 = rnorm(100000, mean=2, sd=1.5)
y2 = rnorm(100000, mean=3, sd=1.5)
xy = rbind(cbind(x1, y1), cbind(x2, y2))
smoothScatter(xy)
```

Alpha values

```
# colors are given as amount of
# Red, Green, Blue and Alpha with range [0,1].
# you can specify maxColorValue=255 if you want to use [0,255]
plot(x,y, col=rgb(0,0.5,0,0.1))
```

- lines()
- abline(lm (y ~x))
- points()
- legend()
- qqnorm
- identify()
- barplot(), stacked barplot
- pie()
- boxplot()

Adding additional points to a plot

- points() lets you add points to an existing plot

```
x = rnorm(10); y=rnorm(10)
xx = rnorm(10); yy = rnorm(10)
plot(x, y)
points(xx, yy, col="green", pch="v")
```

- if nothing shows up, your axes might not contain the data area

Adding additional lines to a plot

```
data(orange)
tree1 = which(Orange$Tree == 1)
tree2 = which(Orange$Tree == 2)
lines(Orange$age[tree1], Orange$circumference[tree1], type="b", lwd=1.5, lty=1, col=2)
lines(Orange$age[tree2], Orange$circumference[tree2], type="b", lwd=1.5, lty=1, col=3)
legend("bottomright", legend=c("old", "young"), fill = c(2,3), col=c(2,3))
```

Adding a regression line

- Earlier we plotted the iris dataset

```
plot(iris$Petal.Length, iris$Sepal.Length)
```

- Compute the OLS regression and add to plot

```
model = lm(iris$Sepal.Length ~ iris$Petal.Length)
summary(model)
abline(model, lwd=3, lty=4)
```

More plots

- Barplot

```
barplot(rbind(1:4, 3:6))
barplot(rbind(1:4, 3:6), horiz=T)
```

- Boxplots

```
data(iris)
boxplot(iris$Sepal.Length ~ iris$Species)
```

- QQnorm

```
qqnorm(rnorm(100))
```

- QQplot

```
y <- rt(200, df = 5)
qqnorm(y); qqline(y, col = 2)
qqplot(y, rt(300, df = 5))  #lengths need not be identical
```

More plots

- Hierarchical clustering

```
all = read.table("ALL_subset.txt")
hier1 = hclust( dist(1-cor(all)))
plot(hier1)
```

- Heatmap

```
heatmap(cor(all), symm=T)
```

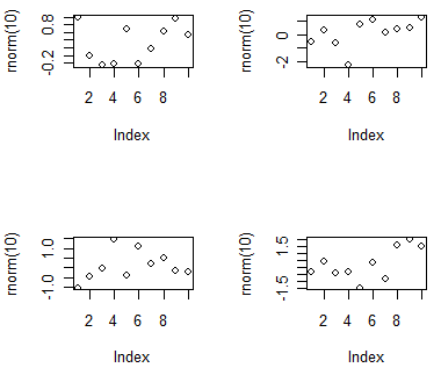
Multiple plots per page

- Use par

```
par(mfrow=c(2,1))
```

- Another method is to designate matrix of slots

```
layout( matrix(c(1,2,3,4), ncol=2, nrow=2))
for(i in 1:4){
plot(rnorm(10))
}
# compare with
layout(matrix(c(1,1,2,3),2,2)
```



- Once you close a window, settings disappear

Graphics devices

- R will "write" plots to a graphics device
- If you don't have one, for example you ssh into a remote machine, it will write to a file called Rplots.pdf in your current directory.

```
# show your current graphic device
dev.cur()
# show all devices
dev.list()
# select a device (from already open)
dev.set()
# create a new window
x11()
window()
quartz()
```

Output formats

- pdf(), png(), bmp(), tiff(), jpeg(), formats all supported
- png(filename="file.png", width=800, height=800)
- dev.cur(), dev.off(), dev.set()

- To create a graphic, think of it as writing an image to a file

```
pdf(file="foo.pdf")
plot(rnorm(100))
dev.off()
```

- Don't forget to dev.off() your device, otherwise your figure will get corrupted!

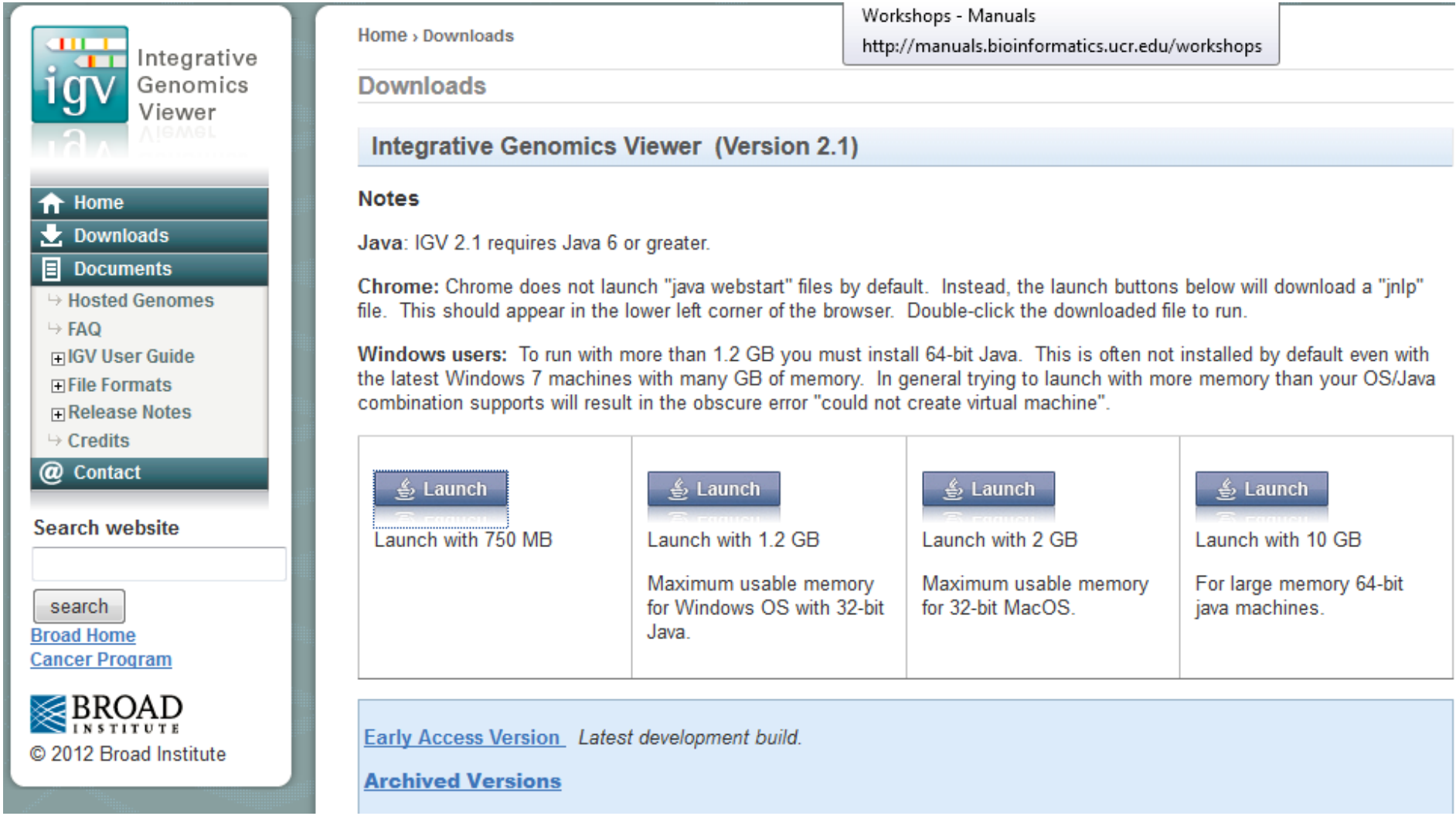
Fancier graphics

- There are more sophisticated plotting paradigms for multipanel, multidimensional data
  - lattice
  - trellis
  - ggplot

Genome visualization

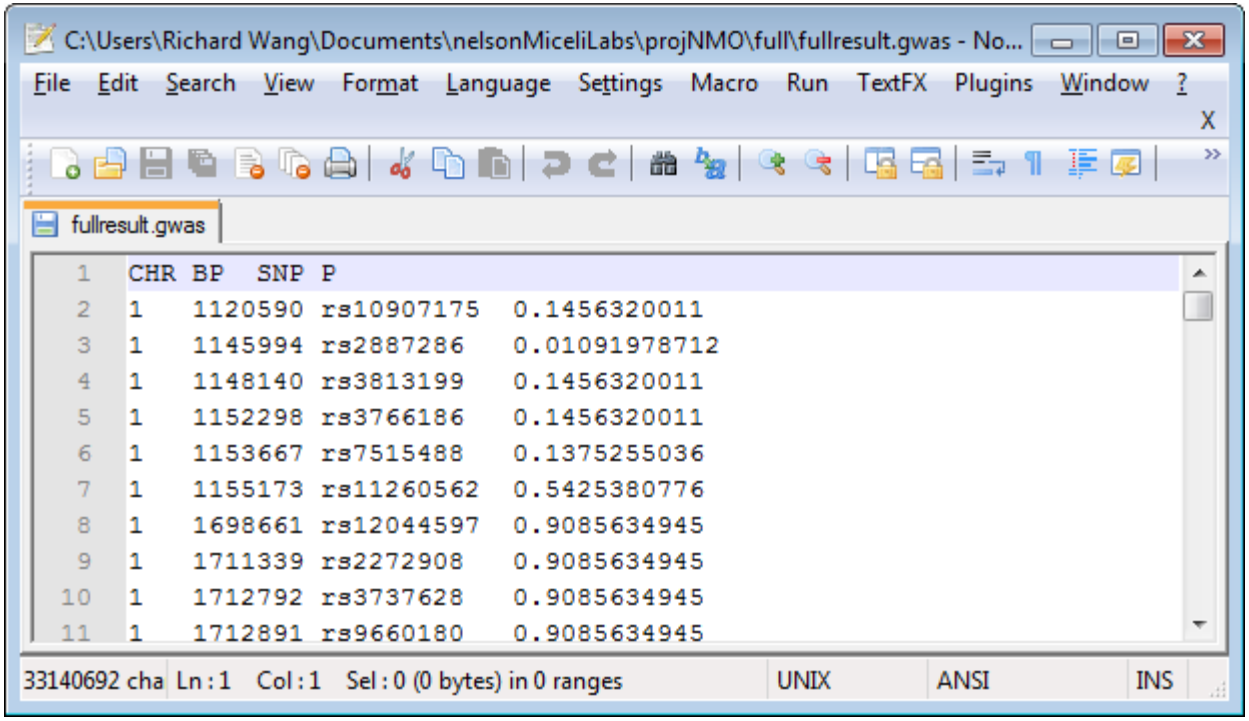
- Suppose you identify a genomic region of interest and you want to know what features are near it
  - high homozygosity, associated SNPs, Chip-Seq peak
- UCSC genome browser
  - use custom tracks to upload your data in BED format
- Broad IGV
  - http://www.broadinstitute.org/igv/
  - Great for seeing a ROI in context
  - Does not generate publication quality images

IGV

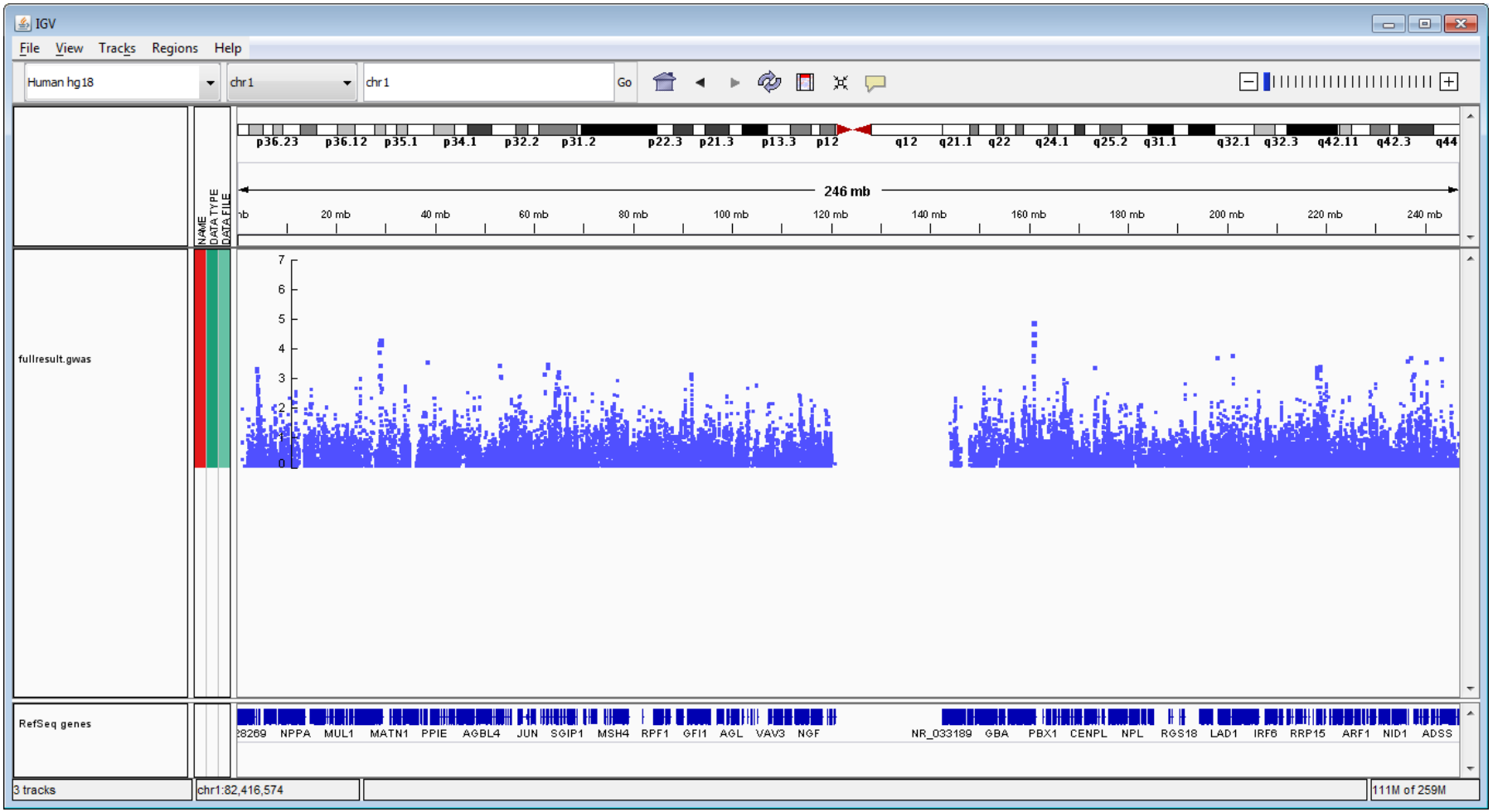

Integrative Genomics Viewer
Home
Downloads
Documents
Hosted Genomes
FAQ
IGV User Guide
File Formats
Release Notes
Credits
Contact
Search website
search
Broad Home
Cancer Program
BROAD INSTITUTE
© 2012 Broad Institute
Home › Downloads
Workshops - Manuals
http://manuals.bioinformatics.ucr.edu/workshops
Downloads
Integrative Genomics Viewer (Version 2.1)
Notes
Java: IGV 2.1 requires Java 6 or greater.
Chrome: Chrome does not launch "java webstart" files by default. Instead, the launch buttons below will download a "jnlp" file. This should appear in the lower left corner of the browser. Double-click the downloaded file to run.
Windows users: To run with more than 1.2 GB you must install 64-bit Java. This is often not installed by default even with the latest Windows 7 machines with many GB of memory. In general trying to launch with more memory than your OS/Java combination supports will result in the obscure error "could not create virtual machine".
Launch
Launch with 750 MB
Launch
Launch with 1.2 GB
Maximum usable memory for Windows OS with 32-bit Java.
Launch
Launch with 2 GB
Maximum usable memory for 32-bit MacOS.
Launch
Launch with 10 GB
For large memory 64-bit java machines.
Early Access Version Latest development build.
Archived Versions

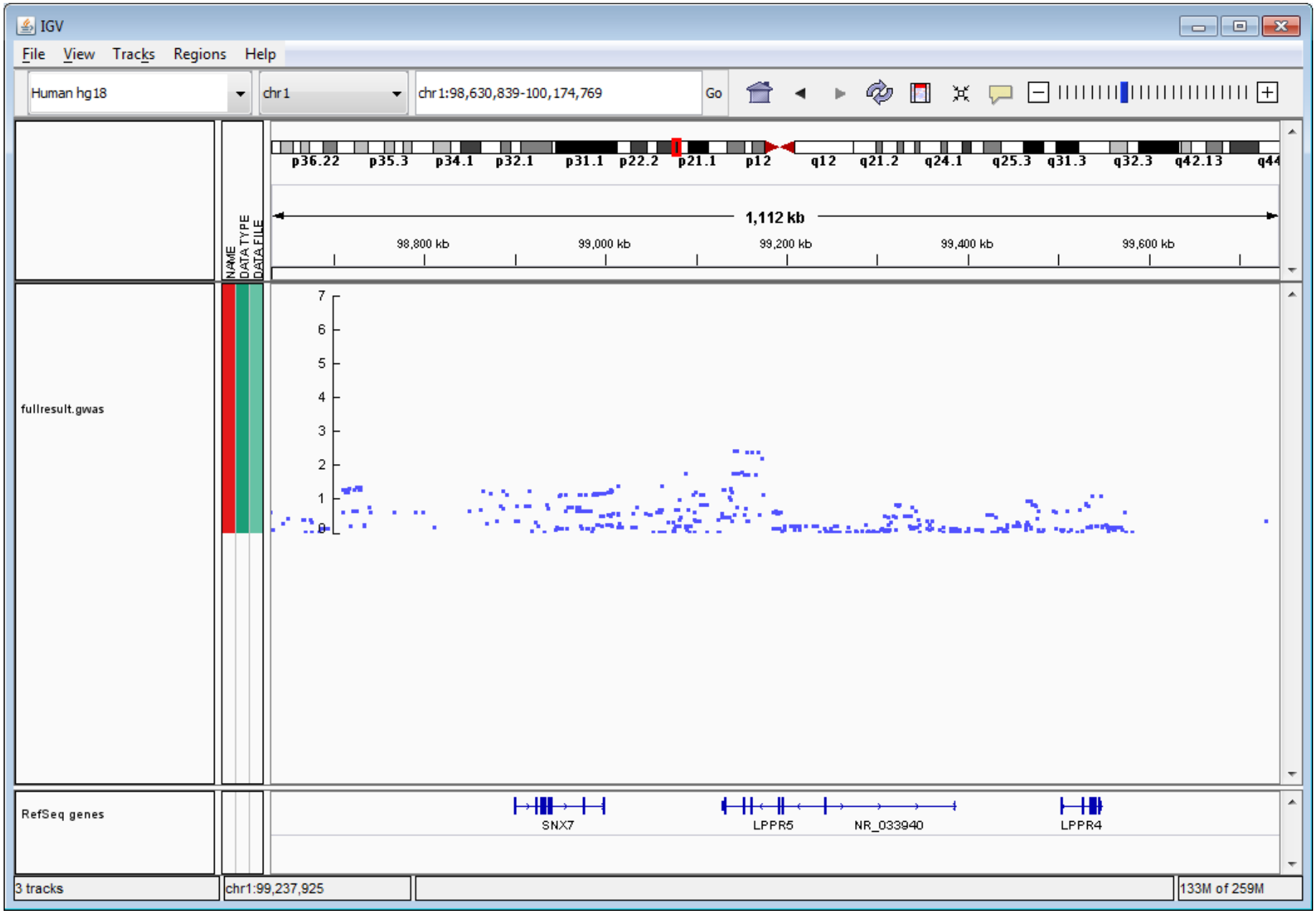
GWAS format

- IGV accepts many types of formats
- GWAS format is simply chrom, sart, snpid, pval



IGV


IGV
File View Tracks Regions Help
Human hg18
chr1
chr1
Go
p36.23 p36.12 p35.1 p34.1 p32.2 p31.2 p22.3 p21.3 p13.3 p12 q12 q21.1 q22 q24.1 q25.2 q31.1 q32.1 q32.3 q42.11 q42.3 q44
246 mb
20 mb 40 mb 60 mb 80 mb 100 mb 120 mb 140 mb 160 mb 180 mb 200 mb 220 mb 240 mb
NAME
DATA TYPE
DATA FILE
fullresult.gwas
7 6 5 4 3 2 1 0
RefSeq genes
NPPA MUL1 MATN1 PPIE AGBL4 JUN SGIP1 MSH4 RPF1 GFI1 AGL VAV3 NGF
NR_033189 GBA PBX1 CENPL NPL RGS18 LAD1 IRF6 RRP15 ARF1 NID1 ADSS
3 tracks
chr1:82,416,574
111M of 259M


IGV
File View Tracks Regions Help
Human hg18
chr1
chr1:98,630,839-100,174,769
Go
p36.22 p35.3 p34.1 p32.1 p31.1 p22.2 p21.1 p12 q12 q21.2 q24.1 q25.3 q31.3 q32.3 q42.13 q44
1,112 kb
98,800 kb
99,000 kb
99,200 kb
99,400 kb
99,600 kb
NAME
DATA TYPE
DATA FILE
fullresult.gwas
7
6
5
4
3
2
1
0
RefSeq genes
SNX7
LPPR5
NR_033940
LPPR4
3 tracks
chr1:99,237,925
133M of 259M

CummeRbund

- Designed for Cufflinks output
- Visualization of data
- quick run through
  - see protocol
  - manual
  - part of bioconductor

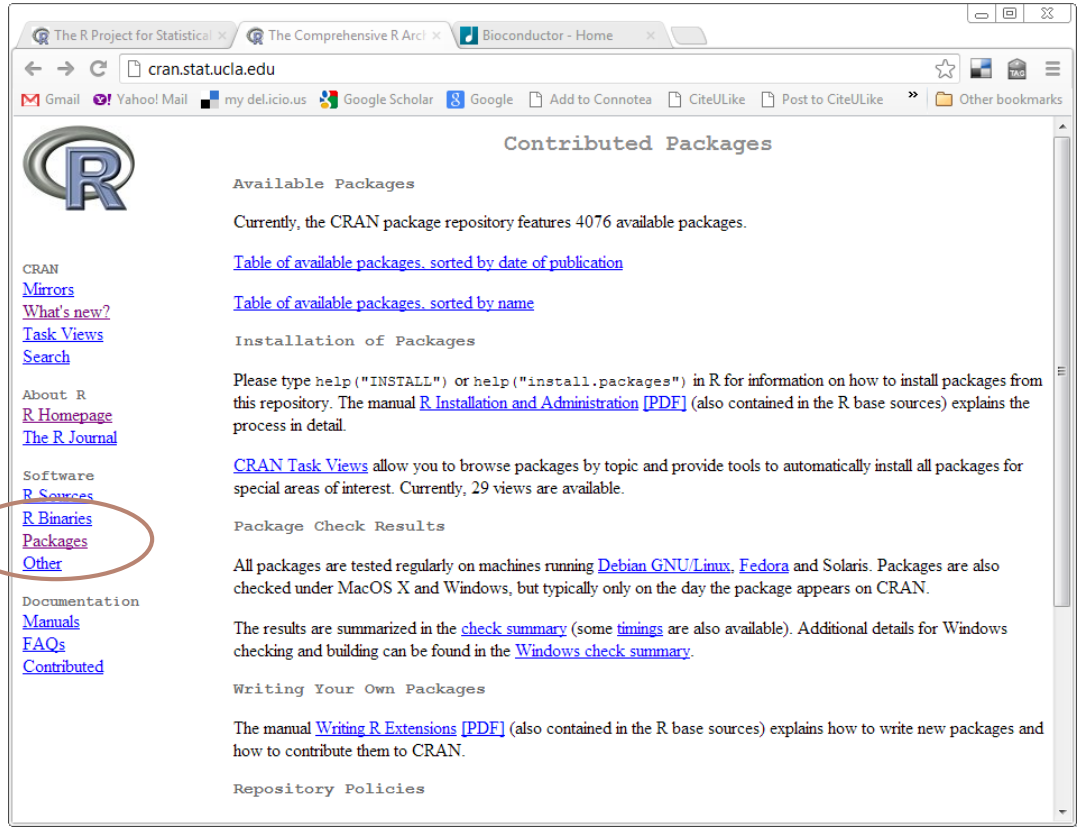
Extending R

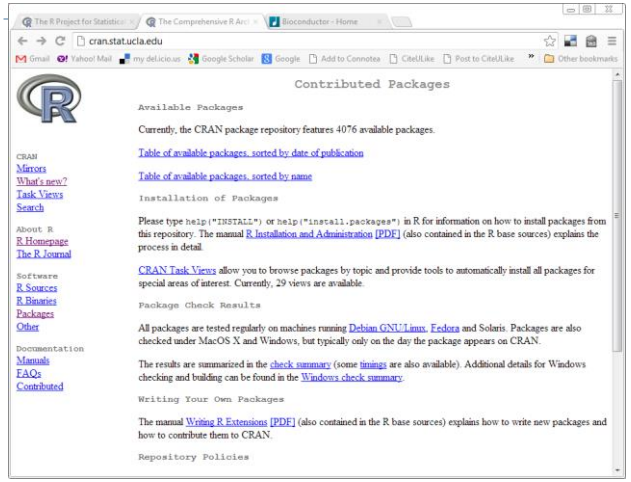
- Many distributions and tests built in
- Others added as packages
  - available from CRAN
  - bioconductor is specialized for life sciences
- Packages essentially add new functionality to R

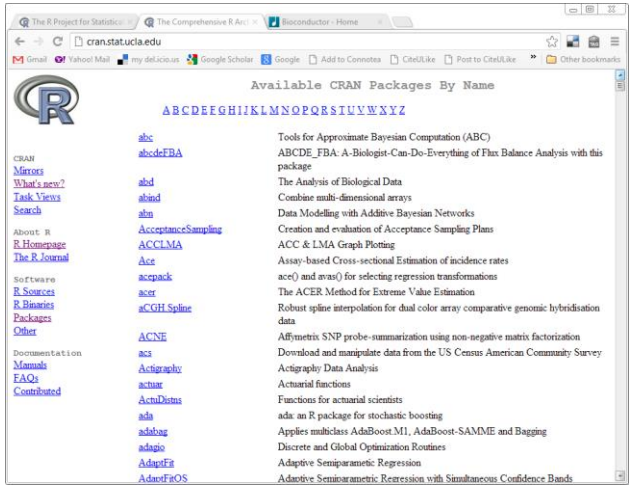
Installing package from CRAN

- installing packages is usually easy
- biggest problems are due to incompatibility of versions of R with versions of a package
  - eg MASS ver 2 requires R version >= 2.13
- source code is always available
- install.packages()
- where do files get stored?

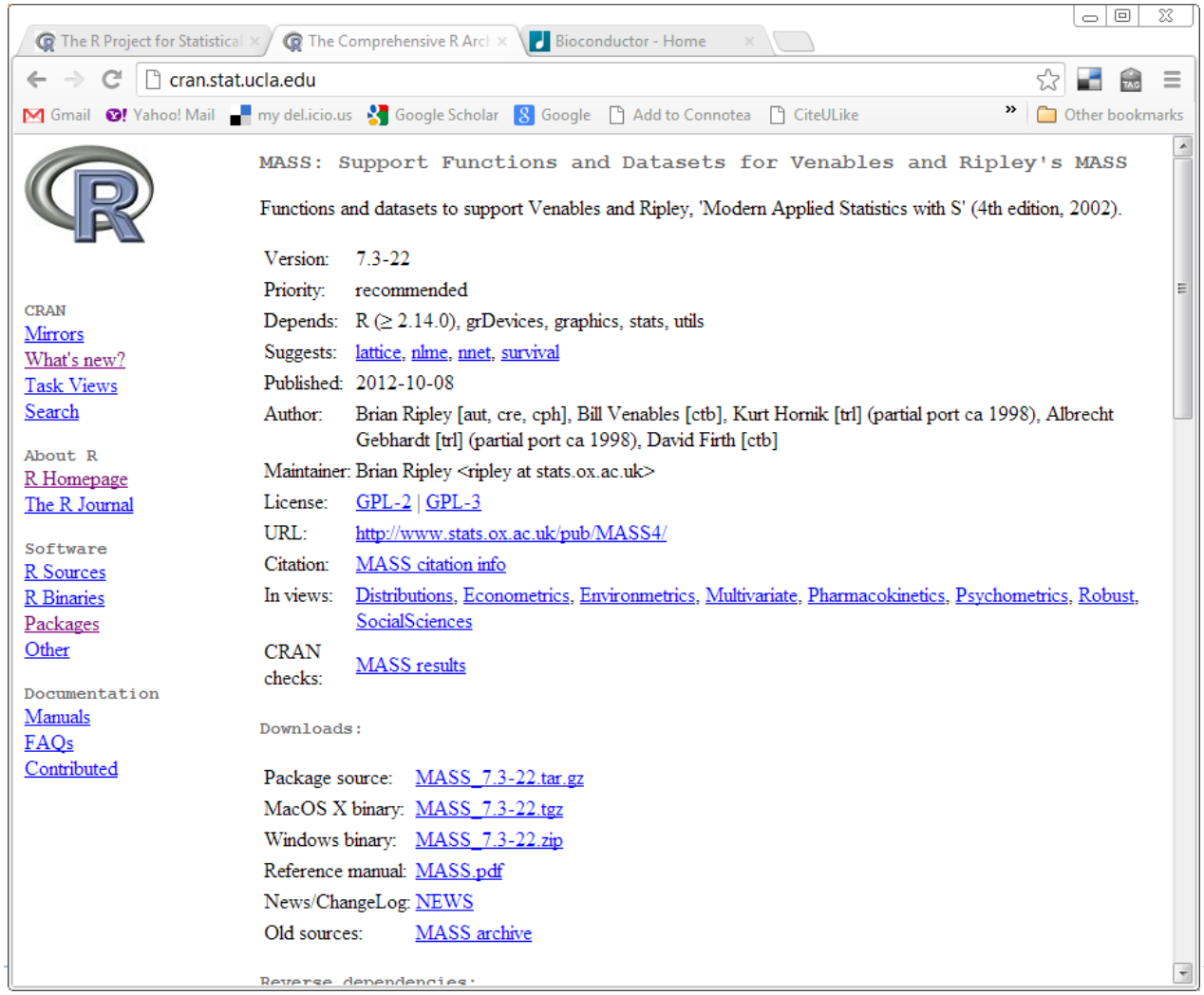
Finding packages







Finding packages



Using the GUI

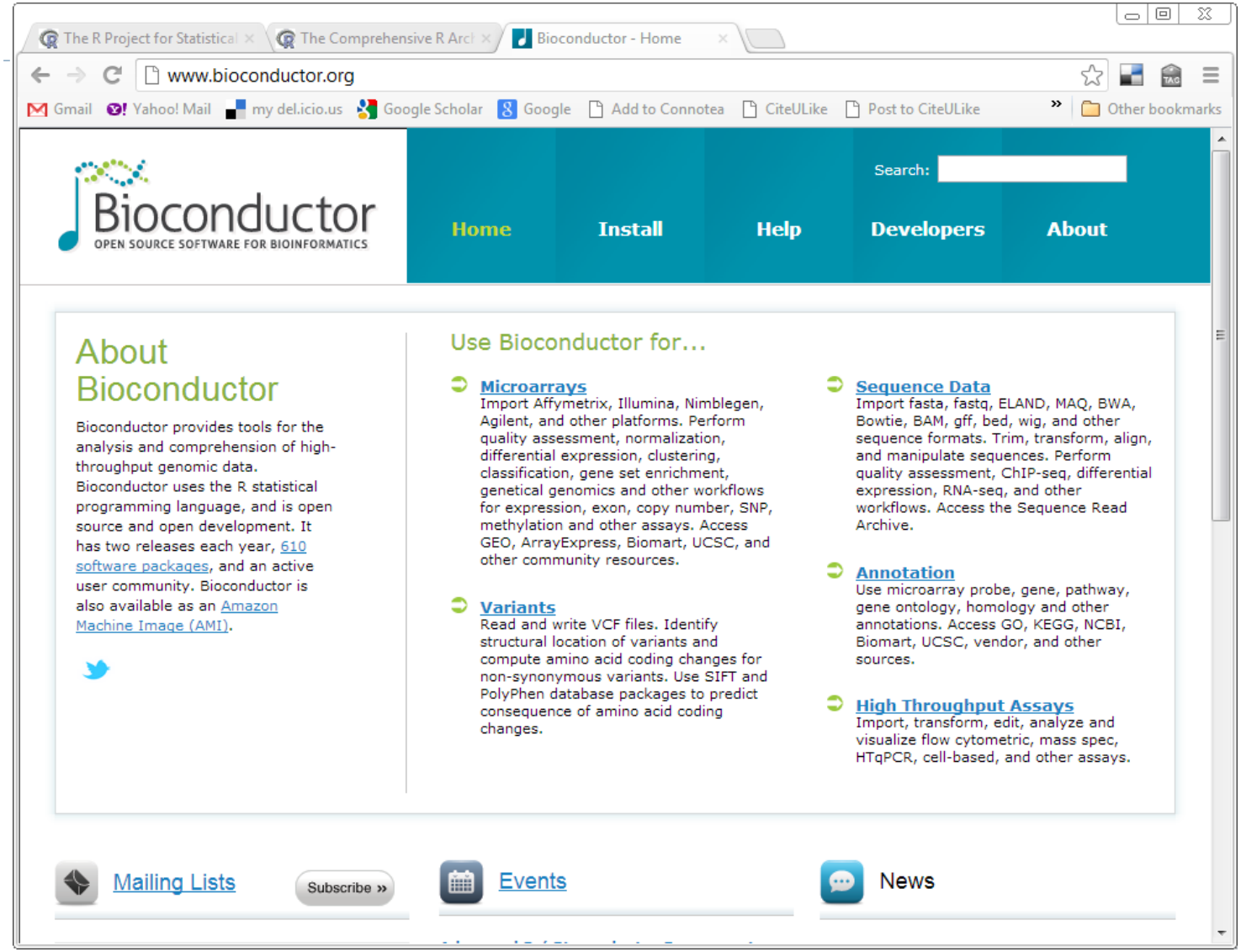
- Use "Package" menu
- select a mirror (aka geographically close repository)
- Repo should include CRAN
- Install packages allows you to select
- Update packages allows you to select which packages to update

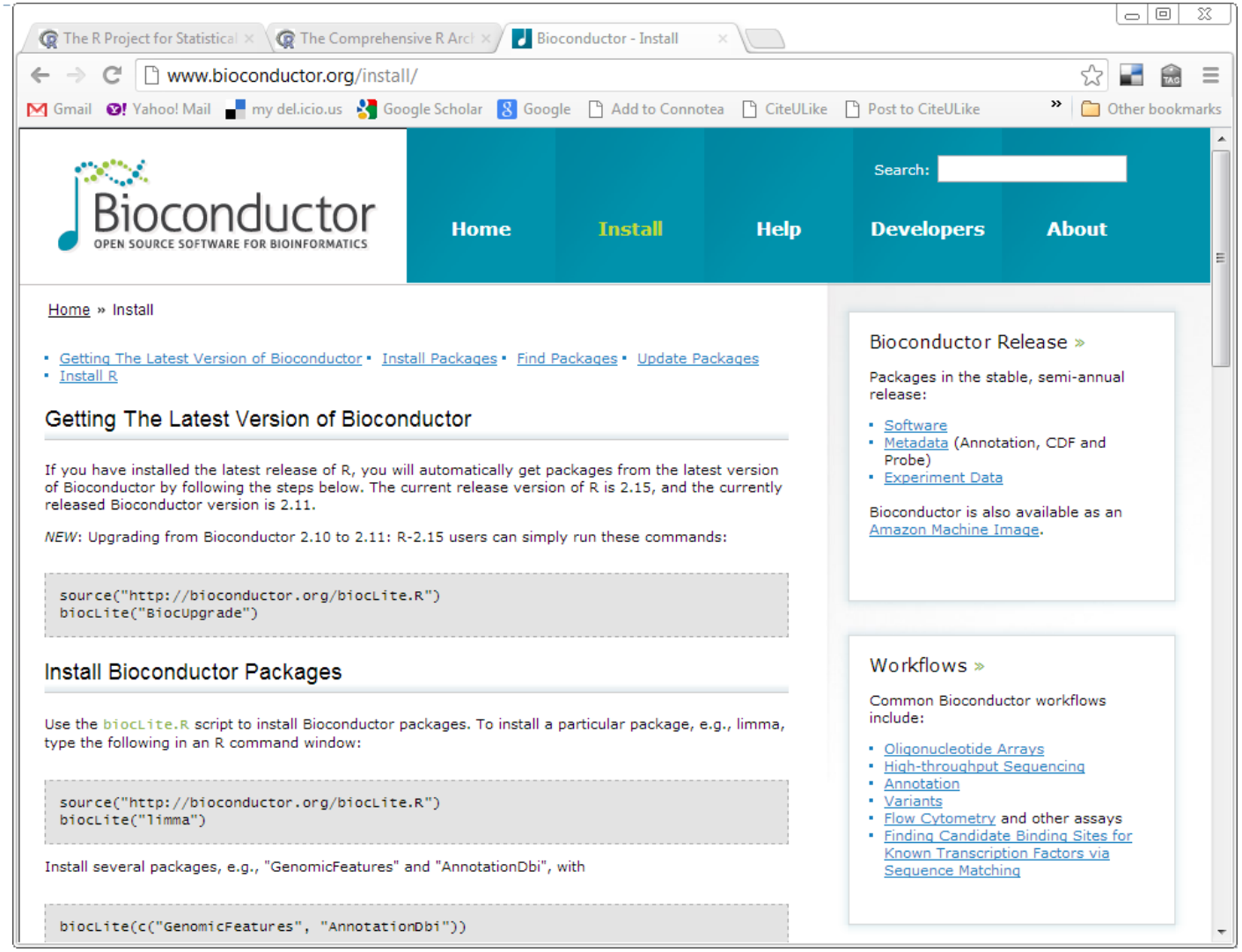
Troubleshooting install problems

- Usually a dependency is missing
  - missing dev package
  - out of date package
  - sometimes R needs access to a C or Fortran compilier
  - need administrator access

Installing from bioconductor

- set of packages tailored to life sciences
- requires its own installation method but relies on many tools from CRAN
- navigating the bioconductor website
- source("http://bioconductor.org/biocLite.R")
- example datasets


The R Project for Statistical
The Comprehensive R Arcl
Bioconductor - Home
www.bioconductor.org
Gmail
Yahoo! Mail
my del.icio.us
Google Scholar
Google
Add to Connotea
CiteULike
Post to CiteULike
Other bookmarks
Bioconductor
OPEN SOURCE SOFTWARE FOR BIOINFORMATICS
Search:
Home
Install
Help
Developers
About
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, 610 software packages, and an active user community. Bioconductor is also available as an Amazon Machine Image (AMI).
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.
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.
Annotation
Use microarray probe, gene, pathway, gene ontology, homology and other annotations. Access GO, KEGG, NCBI, Biomart, UCSC, vendor, and other sources.
High Throughput Assays
Import, transform, edit, analyze and visualize flow cytometric, mass spec, HTqPCR, cell-based, and other assays.
Mailing Lists
Subscribe »
Events
News


The R Project for Statistical
The Comprehensive R Arcl
Bioconductor - Install
www.bioconductor.org/install/
Gmail
Yahoo! Mail
my del.icio.us
Google Scholar
Google
Add to Connotea
CiteULike
Post to CiteULike
Other bookmarks
Bioconductor
OPEN SOURCE SOFTWARE FOR BIOINFORMATICS
Search:
Home
Install
Help
Developers
About
Home » Install
Getting The Latest Version of Bioconductor
Install Packages
Find Packages
Update Packages
Install R
Getting The Latest Version of Bioconductor
If you have installed the latest release of R, you will automatically get packages from the latest version of Bioconductor by following the steps below. The current release version of R is 2.15, and the currently released Bioconductor version is 2.11.
NEW: Upgrading from Bioconductor 2.10 to 2.11: R-2.15 users can simply run these commands:
source("http://bioconductor.org/biocLite.R")
biocLite("BiocUpgrade")
Install Bioconductor Packages
Use the biocLite.R script to install Bioconductor packages. To install a particular package, e.g., limma, type the following in an R command window:
source("http://bioconductor.org/biocLite.R")
biocLite("limma")
Install several packages, e.g., "GenomicFeatures" and "AnnotationDbi", with
biocLite(c("GenomicFeatures", "AnnotationDbi"))
Bioconductor Release »
Packages in the stable, semi-annual release:
Software
Metadata (Annotation, CDF and Probe)
Experiment Data
Bioconductor is also available as an Amazon Machine Image.
Workflows »
Common Bioconductor workflows include:
Oligonucleotide Arrays
High-throughput Sequencing
Annotation
Variants
Flow Cytometry and other assays
Finding Candidate Binding Sites for Known Transcription Factors via Sequence Matching

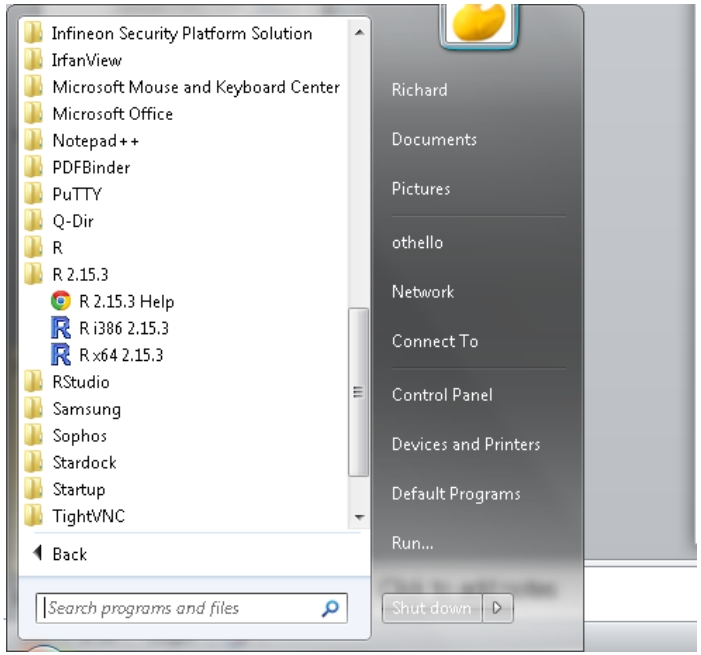
Exercise

- Try and install bioconductor packages
  - GenomicFeatures
  - cummeRbund
  - DEseq
  - Both will require several (many?) additional packages, so let's them install

```
source("http://bioconductor.org/biocLite.R")
# this installs core bioconductor packages
biocLite()
# install Rsamtools
biocLite("Rsamtools")
# GenomicFeatures and gene positions
biocLite("GenomicFeatures")
biocLite("TxDb.Hsapiens.UCSC.hg19.knownGene")
# this installs DEseq2
biocLite("DESeq2")
# other useful data sets
biocLite(c("pasilla", "parathyroidSE")
# this would install cummeRbund
biocLite("cummeRbund")
```

Exercises

- If you run into trouble installing packages
  - On a PC running Windows
    - Try running R in "administrator mode"
    - From the start menu, right click on the R program and select "Run as administrator"



Exercises

- If you run into trouble installing packages
  - On a Mac
    - You may need administrator privileges
  - On Unix/Linux
    - You may need to become root to install to the correct directory
    - Try running sudo either

      ```
      > sudo R
      ```

    - OR

      ```
      > sudo su
      > R
      ```

    - And then installing packages

Exercises

- did it work?
- if install works, issuing these commands will not give you an error
- `library("DESeq2")`
- `library("TxDb.Hsapiens.UCSC.hg19.knownGene")`

Additional resources

- Many resources but sometimes difficult to access
  - R-bloggers.com
  - R-project.org – FAQ and manuals
  - bioconductor.org
  - UCLA ATS (http://www.ats.ucla.edu/stat/r/)
    - learning modules
  - Quick R http://www.statmethods.net/index.html
  - http://onertipaday.blogspot.com/
  - http://wiki.stdout.org/rcookbook/
  - http://gettinggeneticsdone.blogspot.com/
  - manuals.bioinformatics.ucr.edu/workshops

- Good free books
  - An Introduction to R (on r-project.org) – Venables, Smith, R Core
  - Using R
  - R for Beginners by Emmanuel Paradis

Some R books

- R in Action, Robert I. Kabacoff
- R Cookbook, Paul Teetor
- The R book, Michael Crawley
- Introduction to the R Project for Statistical Computing for Use at the ITC, Rossiter

More Resources

- software-carpentry.org
- rosalind project