Bioconductor intro

Thomas Lumley
Ken Rice

UW Biostatistics

Seattle, June 2009

What is Bioconductor?

Bioconductor is an open source and open development software project for the analysis and comprehension of genomic data.

project news

- 2009-01-07
  R, the open source platform used by Bioconductor, featured in a series of articles in the New York Times.
  More...

BioC2009 Conference

Seattle, WA, 27-28 July 2009. Details and Registration

CSAMA09 BioConductor Course

Brixen, Italy, 14-19 June 2009. Details and Registration

Statistical Microarray Analysis Using R (and Bioconductor)

London, England, 18-19 June 2009. Details and Registration

Statistical analysis of gene expression data with R and Bioconductor

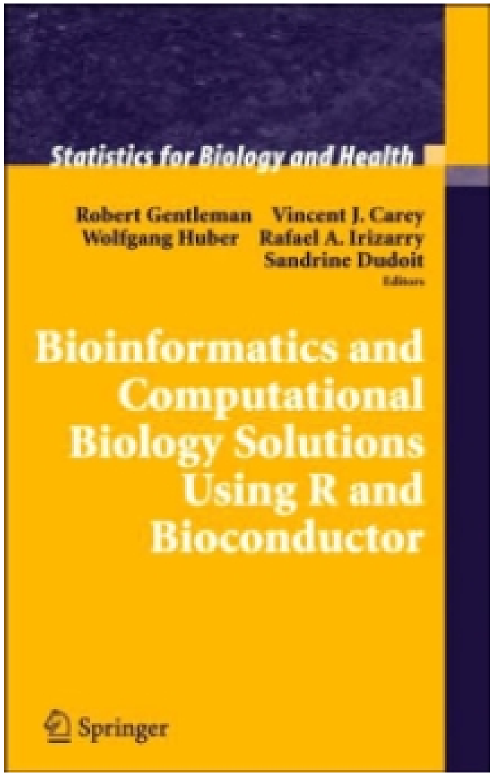
Copenhagen, Denmark, 17-21 August 2009. Details and Registration

What is Bioconductor?

- `www.bioconductor.org`
- Software project for analysis of genomic data – and related tools, resources/datasets
- **Open source** and **Open development**
- **Free**

You could use commercial software; but **experts** typically **write** `R` **code first**. The help manuals are **not a sales pitch** and encourage **appropriate** use

Bioconductor basics



Statistics for Biology and Health

Robert Gentleman Vincent J. Carey
Wolfgang Huber Rafael A. Irizarry
Sandrine Dudoit
Editors

Bioinformatics and Computational Biology Solutions Using R and Bioconductor

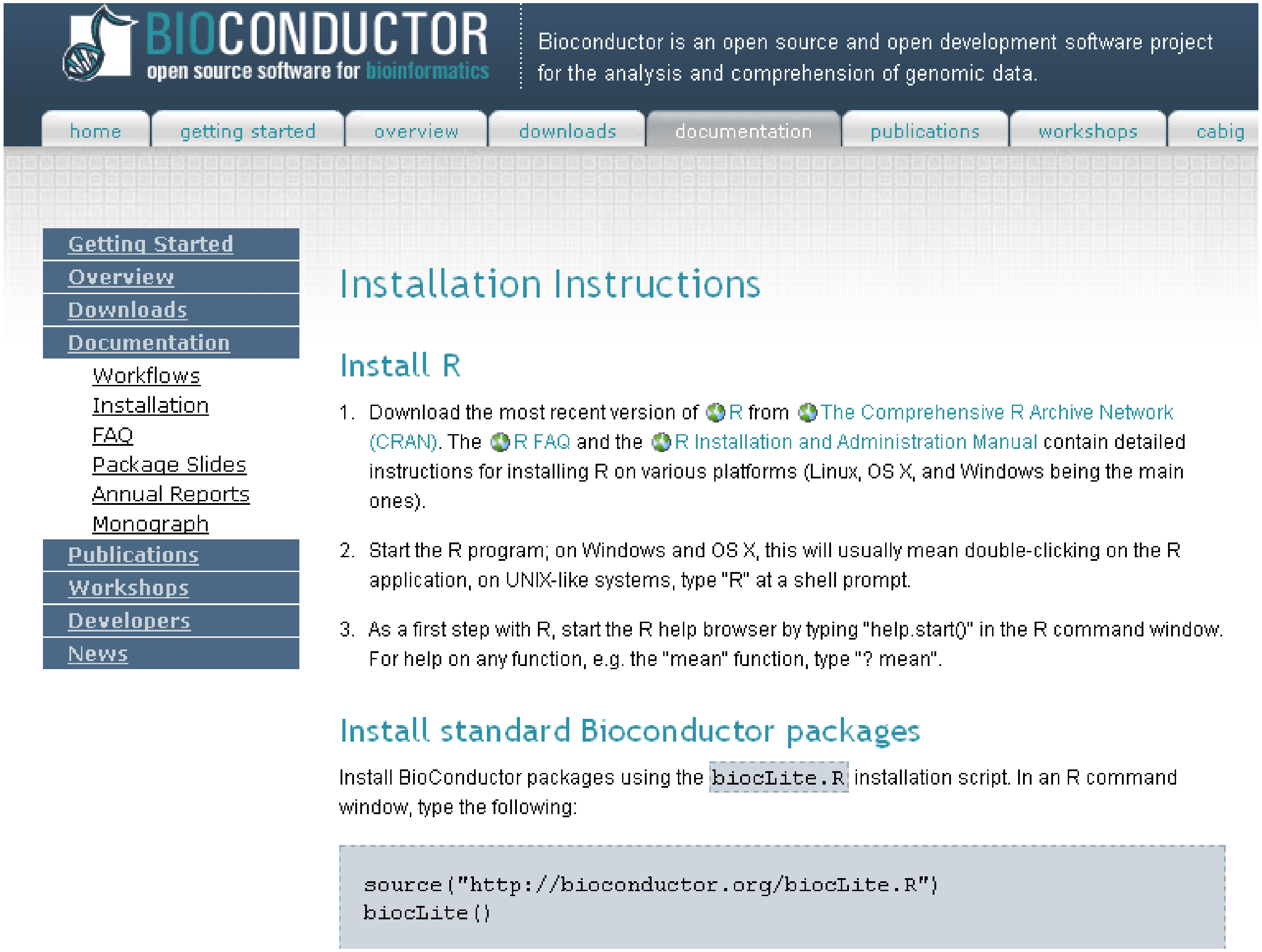
Springer

- Begun in 2001, based at Harvard and now FHCRC (Seattle)
- A large collection of `R` packages (they also convert good software to `R`)
- Far too much for our little course!

We'll give examples of what Bioconductor can do, and how to learn more. Gentleman et al (above) is a helpful reference text

Bioconductor basics

Getting started...



Bioconductor is an open source and open development software project for the analysis and comprehension of genomic data.

Installation Instructions

Install R

1. Download the most recent version of R from The Comprehensive R Archive Network (CRAN). The R FAQ and the R Installation and Administration Manual contain detailed instructions for installing R on various platforms (Linux, OS X, and Windows being the main ones).
2. Start the R program; on Windows and OS X, this will usually mean double-clicking on the R application, on UNIX-like systems, type "R" at a shell prompt.
3. As a first step with R, start the R help browser by typing "help.start()" in the R command window. For help on any function, e.g. the "mean" function, type "? mean".

Install standard Bioconductor packages

Install BioConductor packages using the `biocLite.R` installation script. In an R command window, type the following:

```
source("http://bioconductor.org/biocLite.R")
biocLite()
```

Bioconductor basics

```
> source("http://bioconductor.org/biocLite.R")
> biocLite()
```

installs the following libraries;

`affy`, `affydata`, `affyPLM`, `annaffy`, `annotate`, `Biobase`, `Biostrings`, `DynDoc`, `gcrma`, `genefilter`, `geneplotter`, `hgu95av2.db`, `limma`, `marray`, `matchprobes`, `multtest`, `ROC`, `vsn`, `xtable`, `affyQCReport`

... then you use e.g. `library(ROC)` as before.

`vignette(package="ROC")` tells you to look at `vignette("ROCnotes")` for a **worked example** – a very helpful introduction. (Or use e.g. `openVignette("ROC")` from the `Biobase` package)

Bioconductor basics

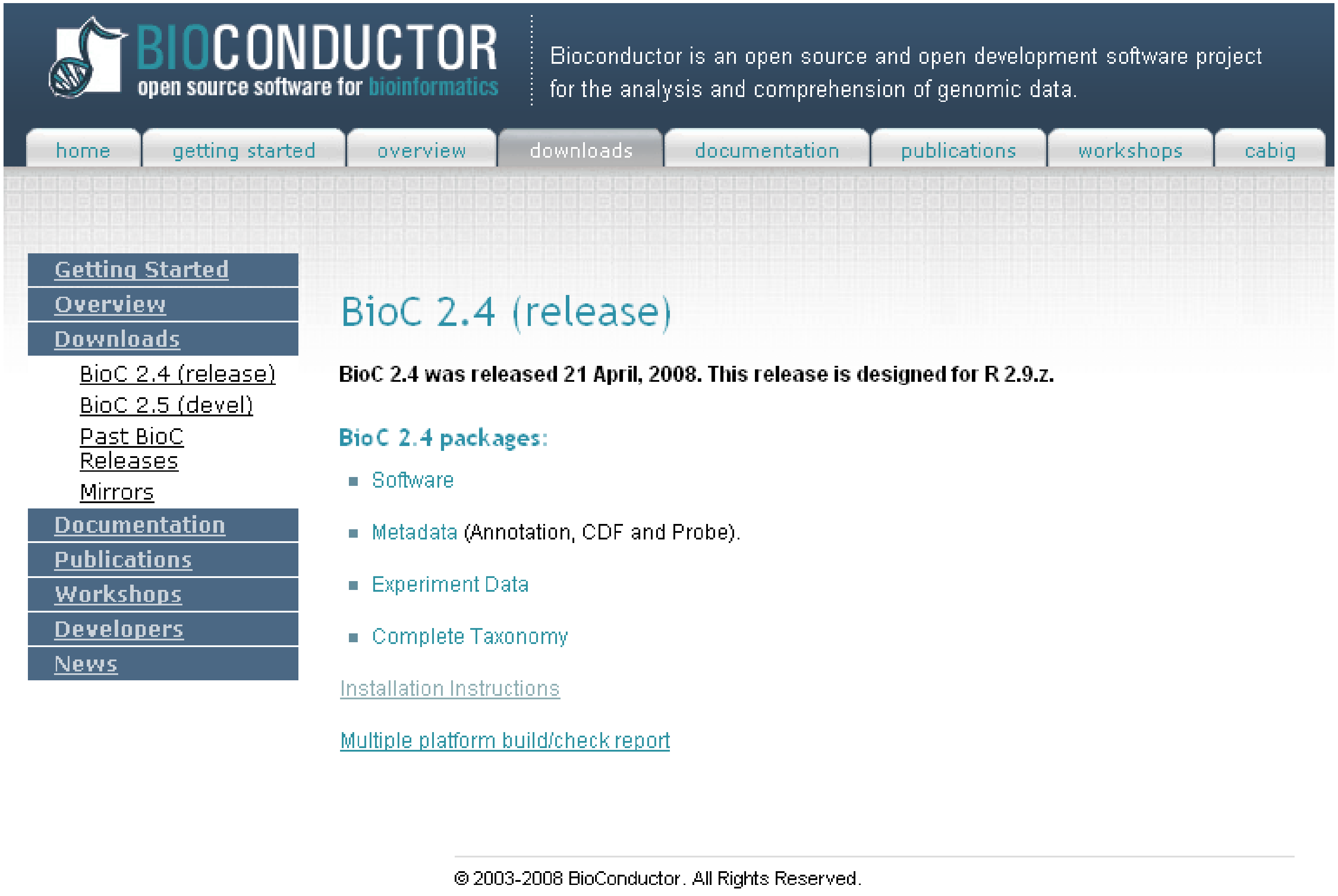
To get other packages, use e.g. `biocLite("SNPchip")`

Do not need to type `biocLite()` after you install (even in a new R session).

This would install everything again – which is harmless, but slow.

What to install?

Back to the front page – click ‘Downloads’



Bioconductor is an open source and open development software project for the analysis and comprehension of genomic data.

BioC 2.4 (release)

BioC 2.4 was released 21 April, 2008. This release is designed for R 2.9.z.

BioC 2.4 packages:

- Software
- Metadata (Annotation, CDF and Probe).
- Experiment Data
- Complete Taxonomy

Installation Instructions

Multiple platform build/check report

© 2003-2008 BioConductor. All Rights Reserved.

What to install?

- **Software** – probably what you want
- **Annotation data** – e.g. probe sequence data for microarrays of different types
- **Experiment data** – e.g. datasets from `hapmap.org`, some expression datasets
- **Complete Taxonomy** – just everything!

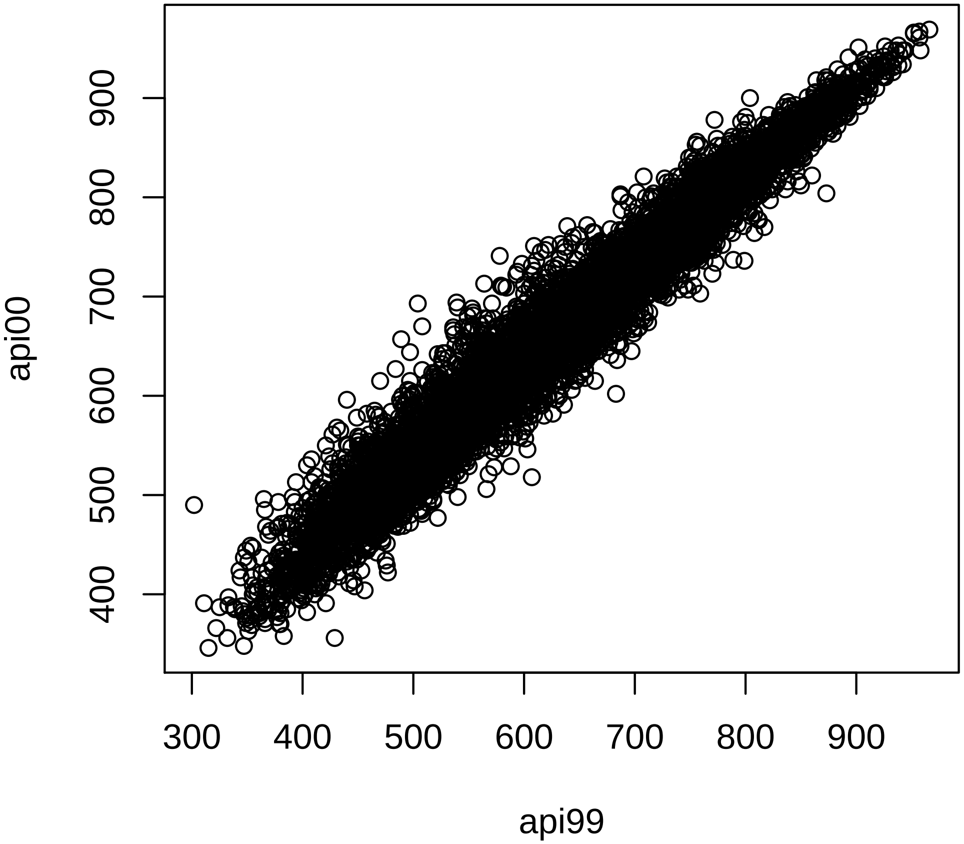
Software example – hexbin

Genetics/Genomics tends to produce **massive** datasets. On any (standard) plot of e.g. 10,000 points, **many** will overlap

Recall the California schools example – the California Academic Performance Index reported from 6194 schools; download the (standard) package

```
> install.packages("survey")
> library(survey)
> data(api)
> plot(api00~api99,data=apipop) # plain plot
```

Software example – hexbin



Hexbin – a better way

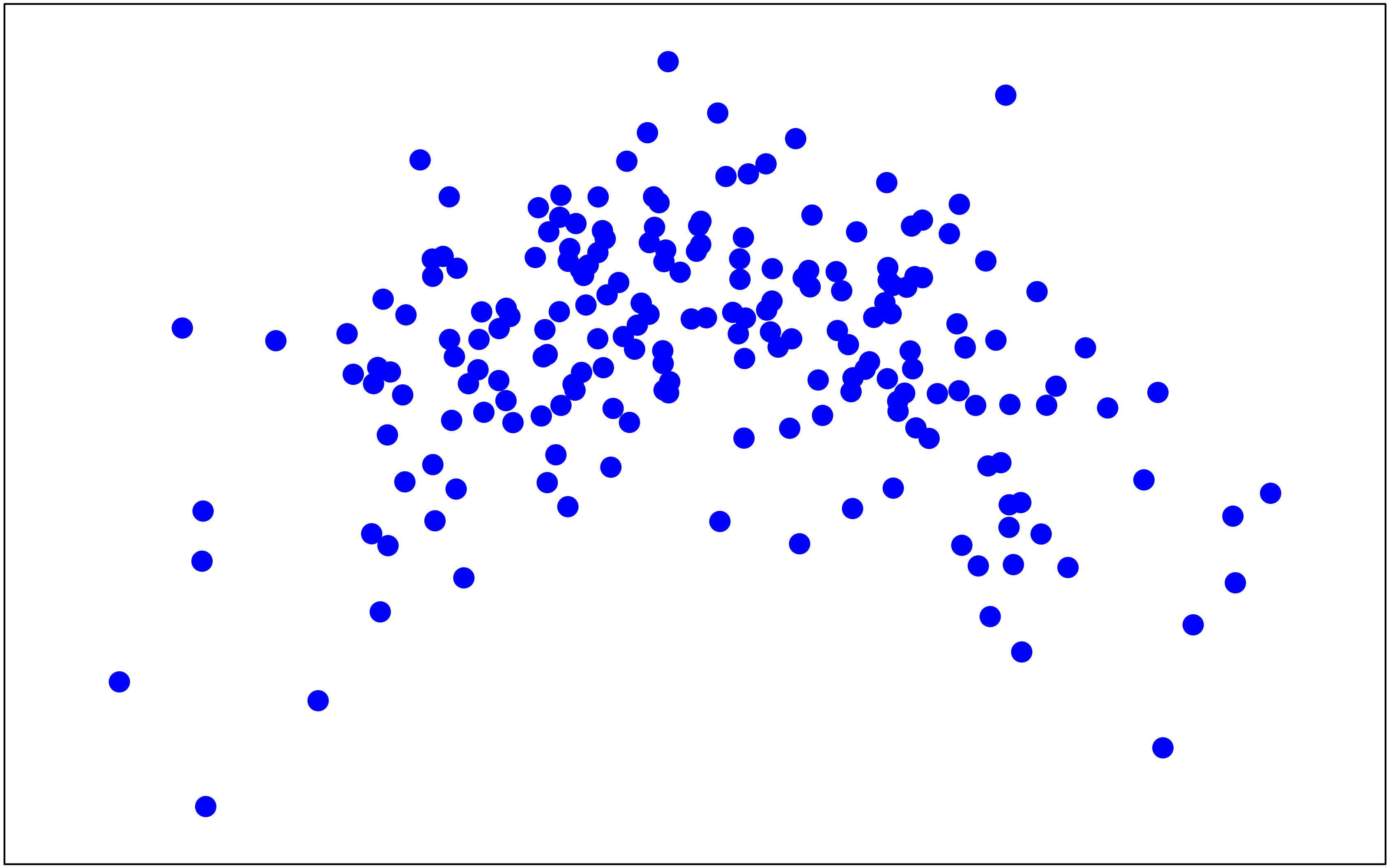
We don't *really* care about the exact location of every single point.

- How **many** points in one 'vicinity' compared to others?
- Any 'outliers' far from all other data points?

In one dimension, histograms answer these questions by **binning** the data

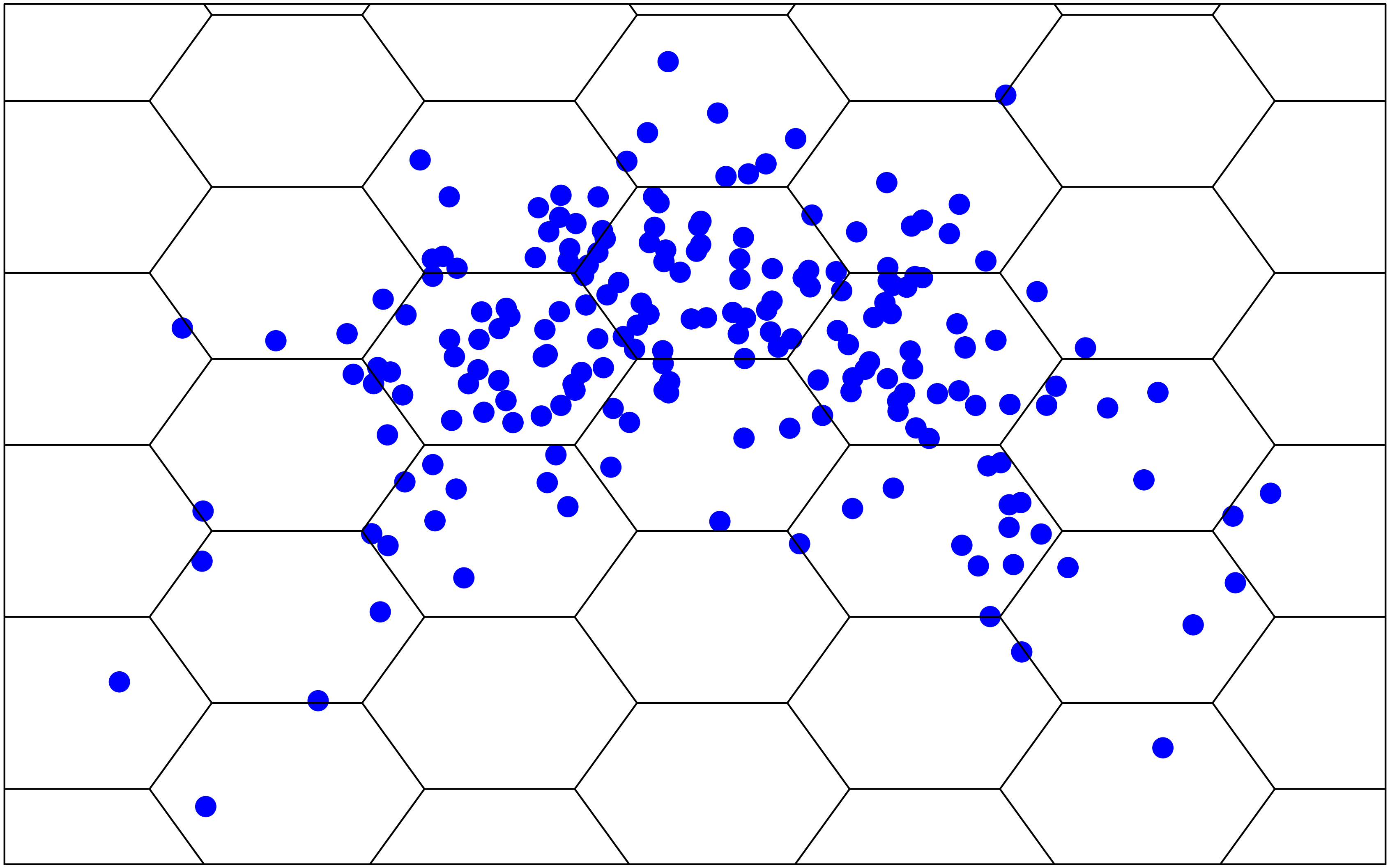
Hexbin – a better way

Binning in two dimensions;



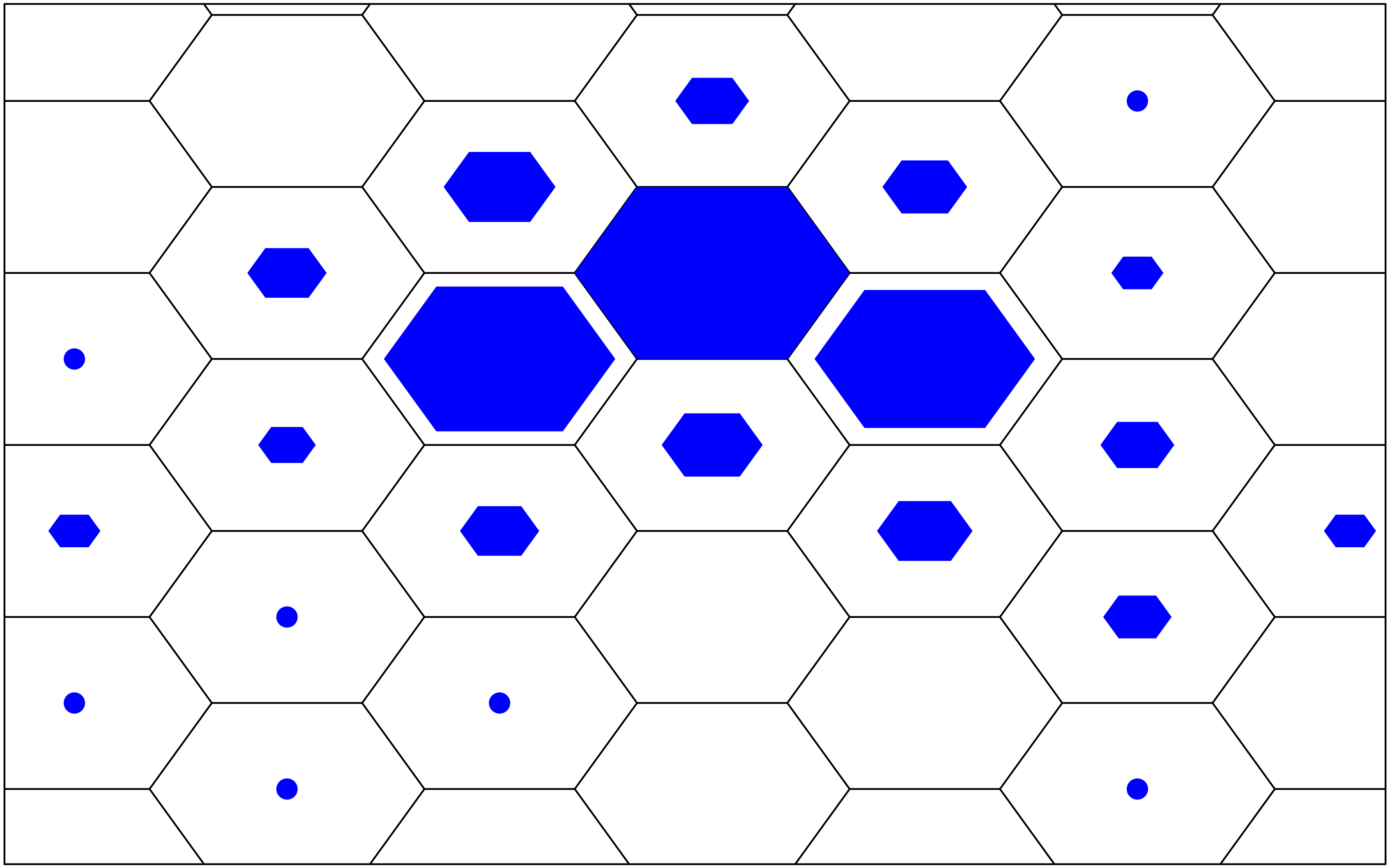
Hexbin – a better way

Binning in two dimensions;



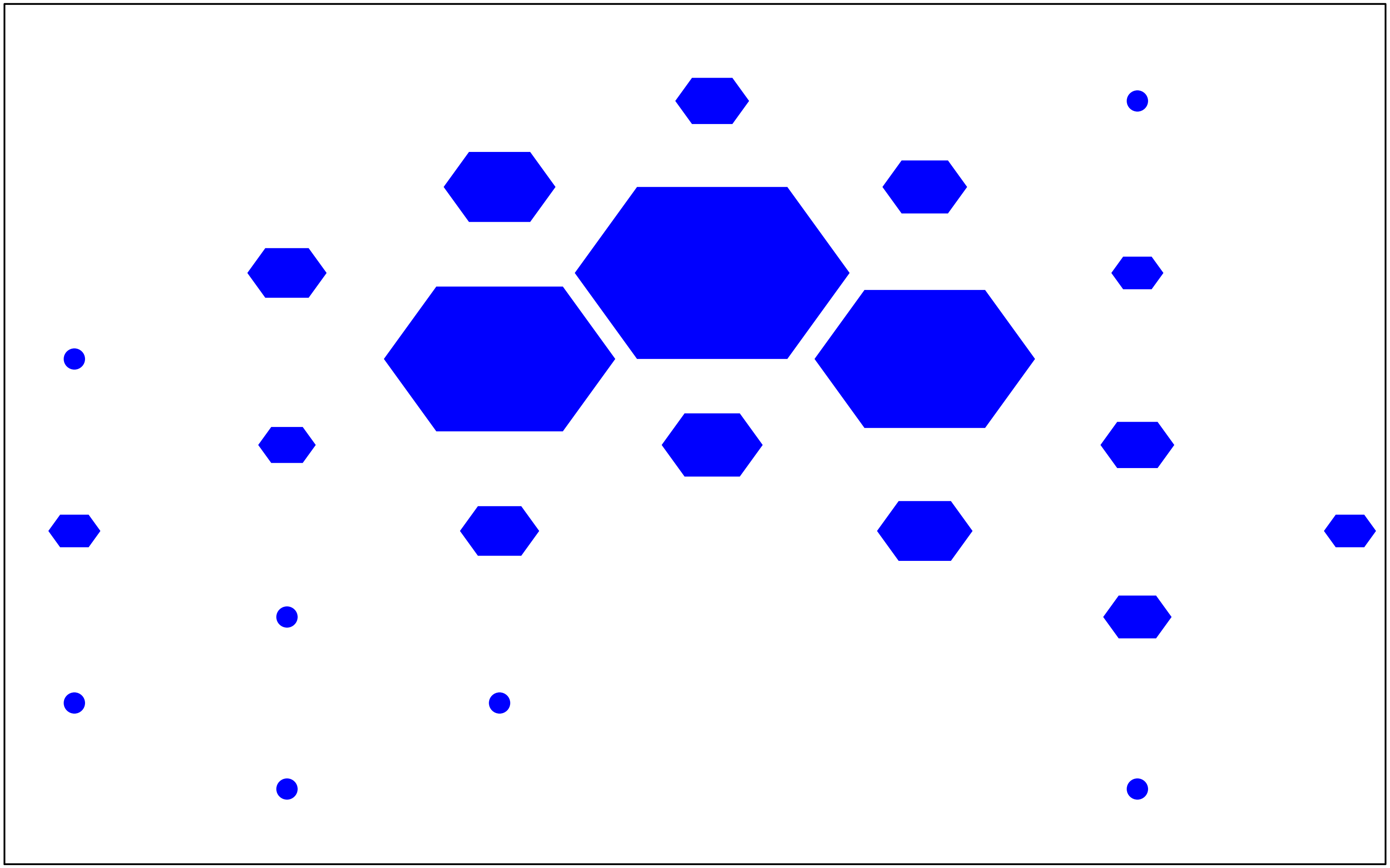
Hexbin – a better way

Binning in two dimensions;



Hexbin – a better way

Binning in two dimensions;



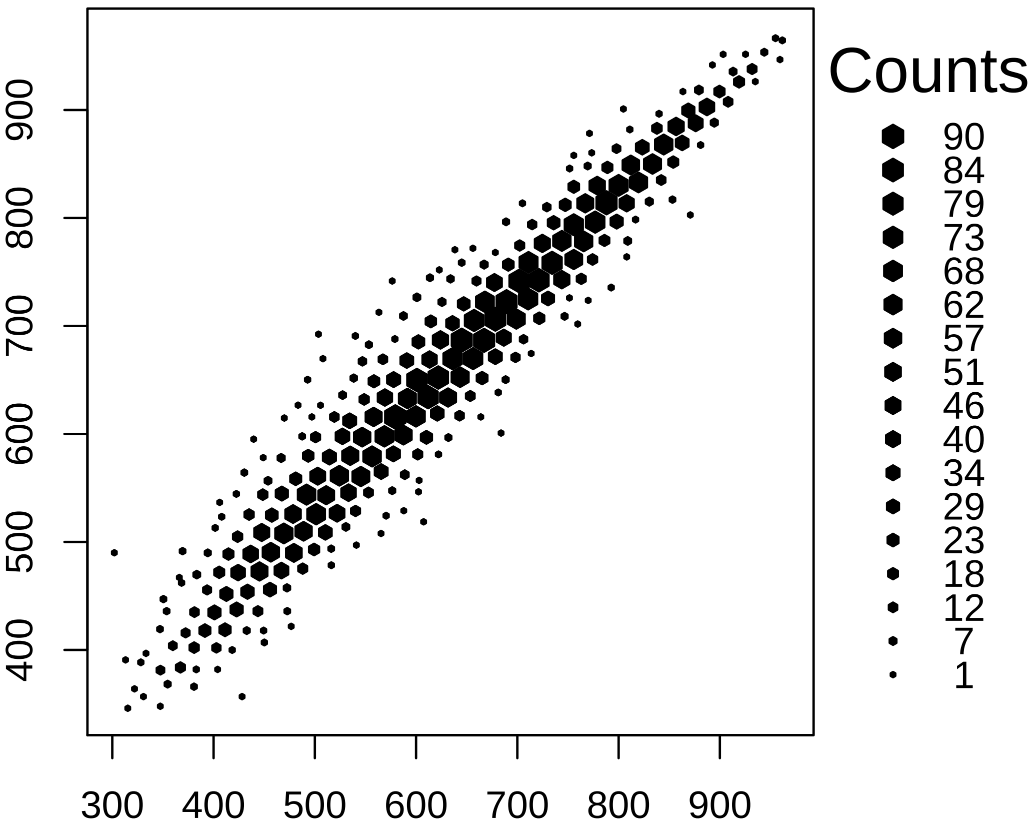
Hexbin — a better way

Back to `schools`

Now with `hexbin`; recall we download from Bioconductor, not CRAN

```
> biocLite("hexbin")
> library(hexbin)
> with(apipop, plot(hexbin(api99,api00), style="centroids"))
```

Hexbin – a better way



GWAS analysis

Genome-Wide Association Studies (GWAS) are currently popular – typically, these genotype e.g. 1M SNPs on several thousand subjects in (large) established studies

- Usually on 1000's of subjects
- 'Simple' t-tests, regressions, for each SNP (like microarrays)
- 1M *anything* takes a long time! (up to 72 hours)
- Just **loading** big datasets is non-trivial – but some tools are available

GWAS analysis

`snpMatrix` is a Bioconductor package for GWAS analysis – maintained by David Clayton (analysis lead on Wellcome Trust)

```
biocLite("snpMatrix")
library(snpMatrix)
data(for.exercise)
```

A ‘little’ case-control dataset (Chr 10) based on HapMap – three objects; `snp.support`, `subject.support` and `snps.10`

GWAS analysis

```
> summary(snp.support)
   chromosome     position              A1          A2
 Min.    :10   Min.    :   101955   A:14019    C: 2349
 1st Qu.:10   1st Qu.: 28981867   C:12166    G:12254
 Median :10   Median : 67409719   G: 2316    T:13898
 Mean    :10   Mean    : 66874497
 3rd Qu.:10   3rd Qu.:101966491
 Max.    :10   Max.    :135323432
> summary(subject.support)
       cc            stratum
 Min.    :0.0    CEU     :494
 1st Qu.:0.0    JPT+CHB:506
 Median :0.5
 Mean    :0.5
 3rd Qu.:1.0
 Max.    :1.0
```

GWAS analysis

```
> show(snps.10) # show() is generic
A snp.matrix with  1000 rows and  28501 columns
Row names:  jpt.869 ... ceu.464
Col names:  rs7909677 ... rs12218790
> summary(snps.10)
$rows
   Call.rate       Heterozygosity
 Min.   :0.9879   Min.   :0.0000
 Median :0.9900   Median :0.3078
 Mean   :0.9900   Mean   :0.3074
 Max.   :0.9919   Max.   :0.3386
$cols
     Calls         Call.rate           MAF                P.AA
 Min.   : 975   Min.   :0.975   Min.   :0.0000   Min.   :0.00000
 Median : 990   Median :0.990   Median :0.2315   Median :0.26876
 Mean   : 990   Mean   :0.990   Mean   :0.2424   Mean   :0.34617
 Max.   :1000   Max.   :1.000   Max.   :0.5000   Max.   :1.00000
      P.AB              P.BB              z.HWE
 Min.   :0.0000   Min.   :0.00000   Min.   :-21.9725
 Median :0.3198   Median :0.27492   Median : -1.1910
 Mean   :0.3074   Mean   :0.34647   Mean   : -1.8610
 Max.   :0.5504   Max.   :1.00000   Max.   :  3.7085
                                    NA's   :  4.0000
```

GWAS analysis

- 28501 SNPs, all with Allele 1, Allele 2

- 1000 subjects, 500 controls (`cc=0`) and 500 cases (`cc=1`)

- **Far too much** data for a regular `summary()` of `snps.10` – even in this small example

GWAS analysis

We'll use just the column summaries, and a (mildly) 'clean' subset;

```
> snpsum <- col.summary(snps.10)
> use <- with(snpsum, MAF > 0.01 & z.HWE^2 < 200)

> table(use)
use
FALSE  TRUE
  317 28184
```

GWAS analysis

Now do single-SNP tests for each SNP, and extract the p-value for each SNP, along with its location;

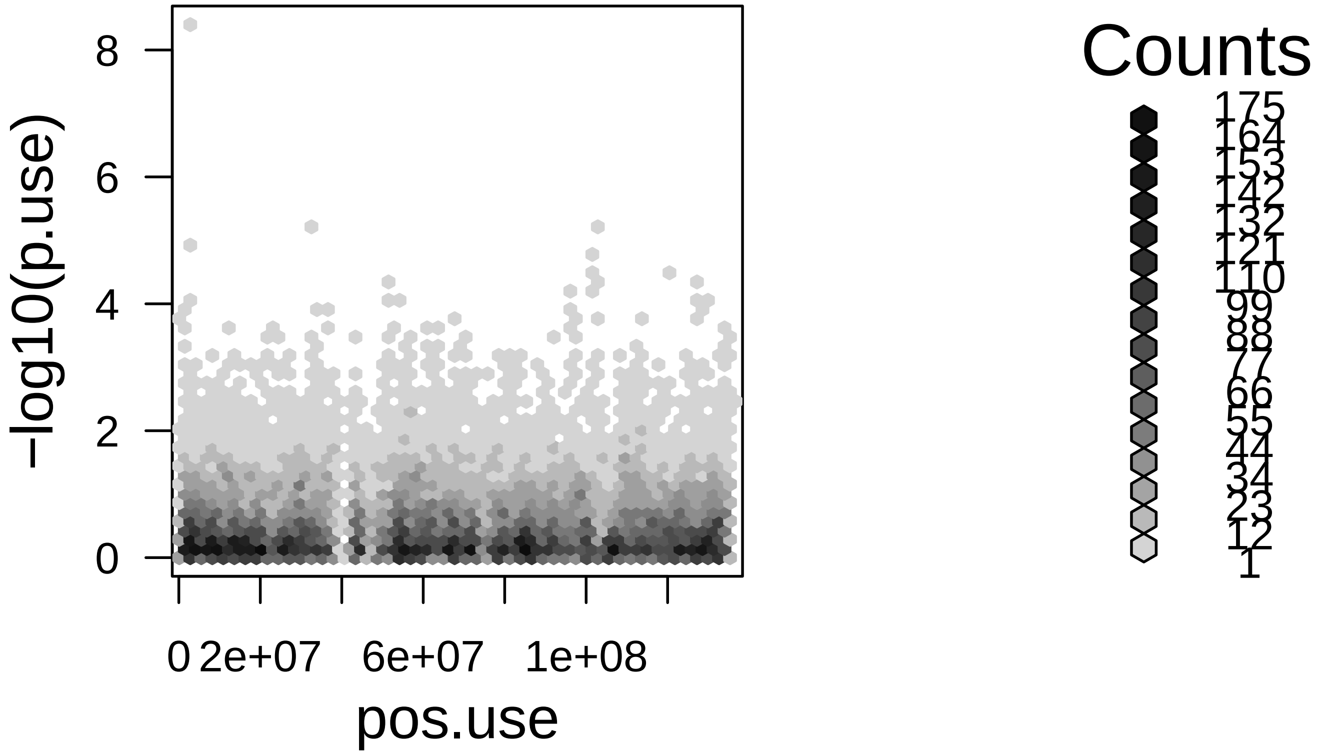
```
tests <- single.snp.tests(cc, data = subject.support,
+ snp.data = snps.10)

pos.use <- snp.support$position[use]
p.use   <- p.value(tests, df=1)[use]
```

We'd usually give a table of 'top hits,' but...

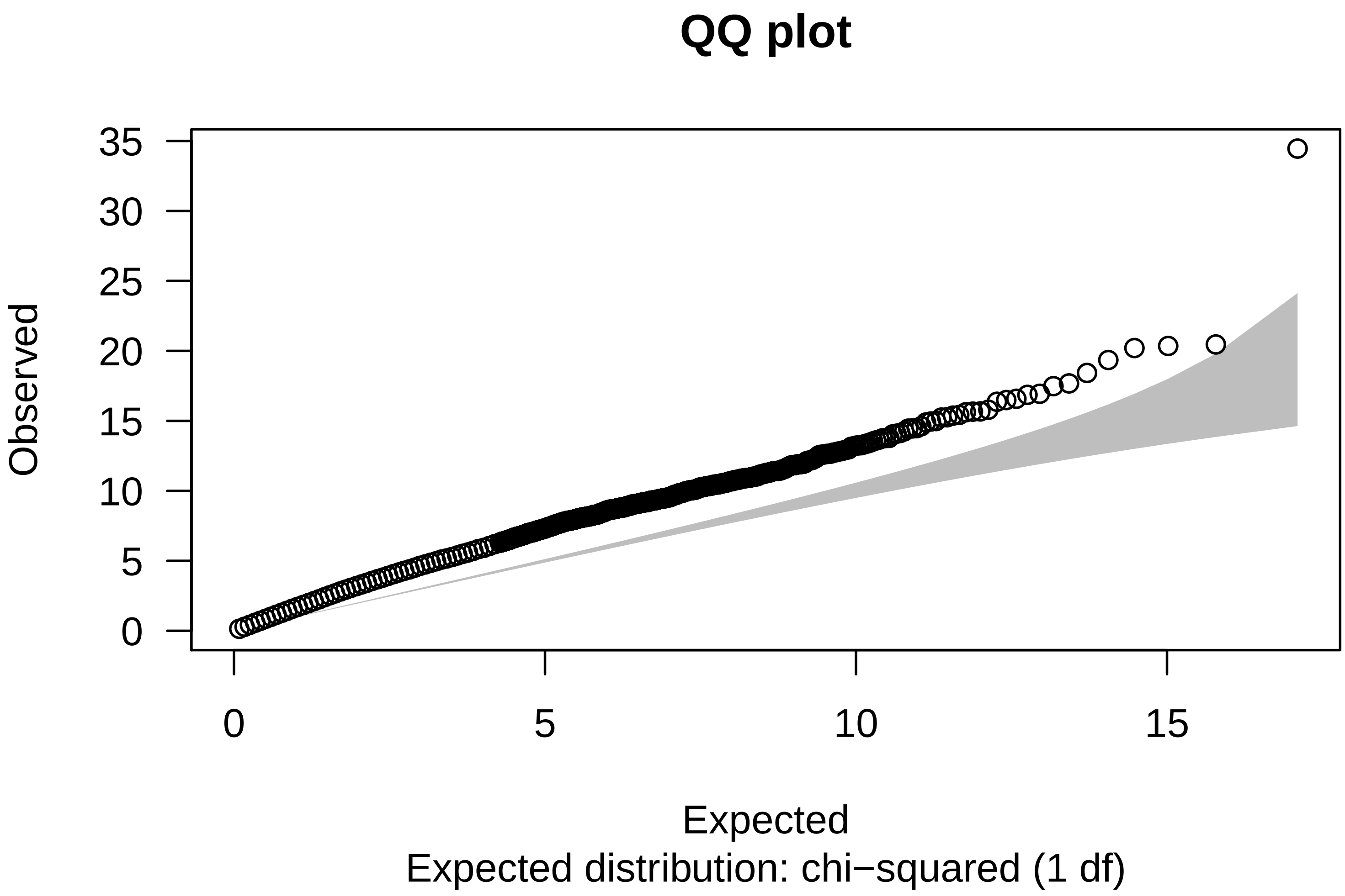
GWAS analysis

```
plot(hexbin(pos.use, -log10(p.use), xbin = 50))
```



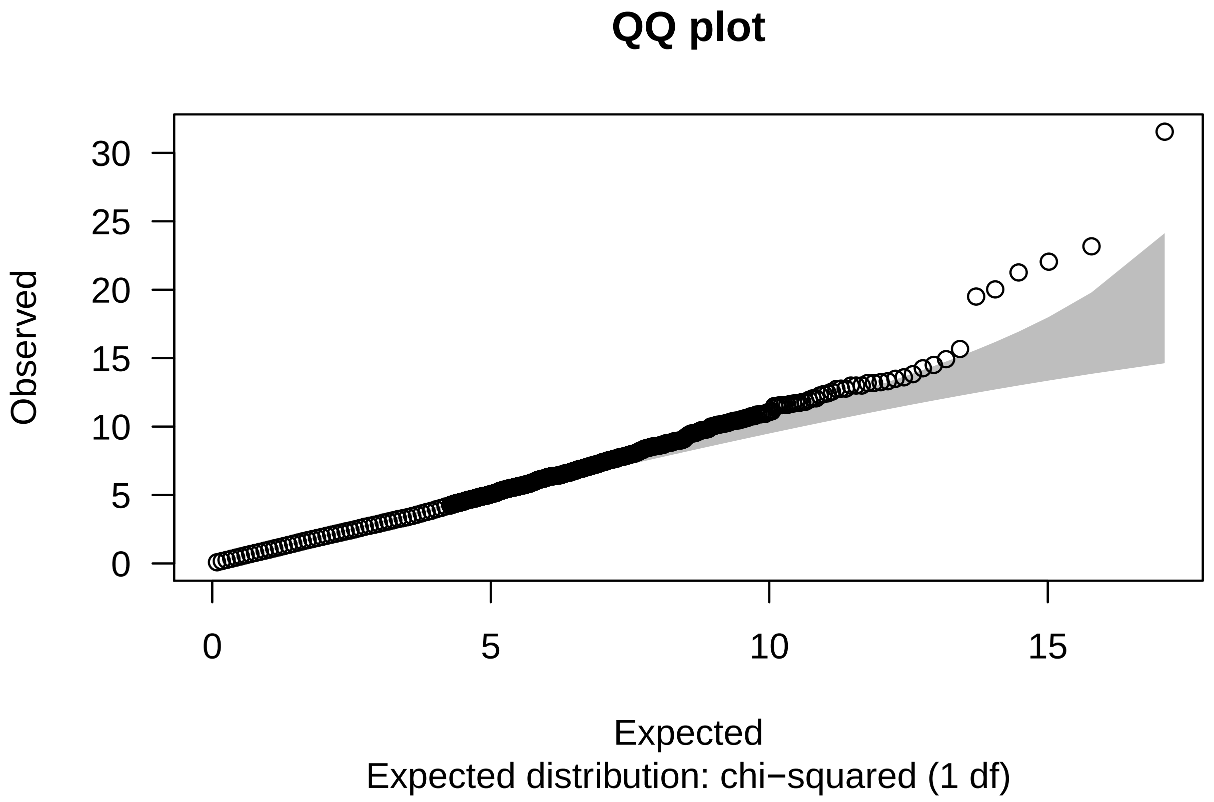
GWAS analysis

```
qq.chisq(chi.squared(tests, df=1)[use], df=1)
```



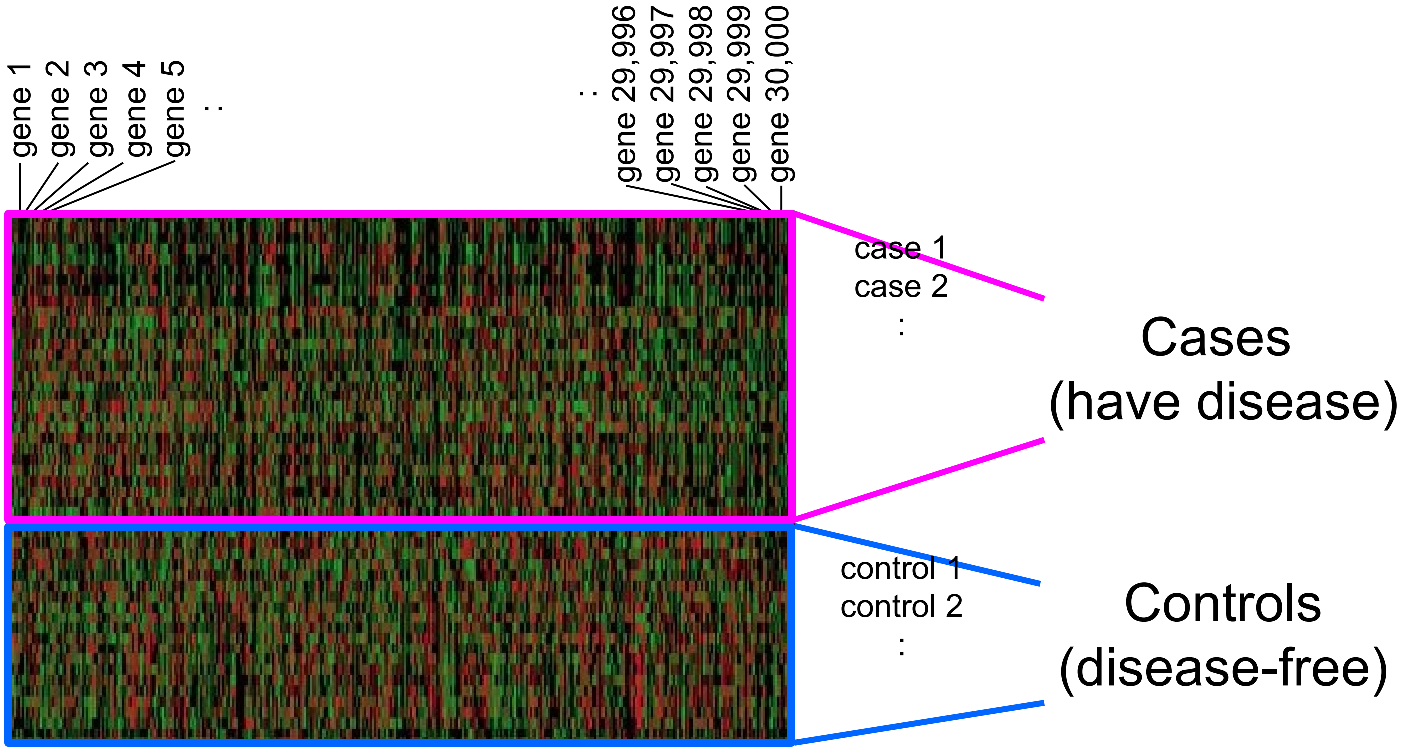
GWAS analysis

```
tests2 <- single.snp.tests(cc, stratum, data = subject.support,
+ snp.data = snps.10)
qq.chisq(chi.squared(tests2, 1)[use], 1)
```



Signficance Analysis of Microarrays (SAM)

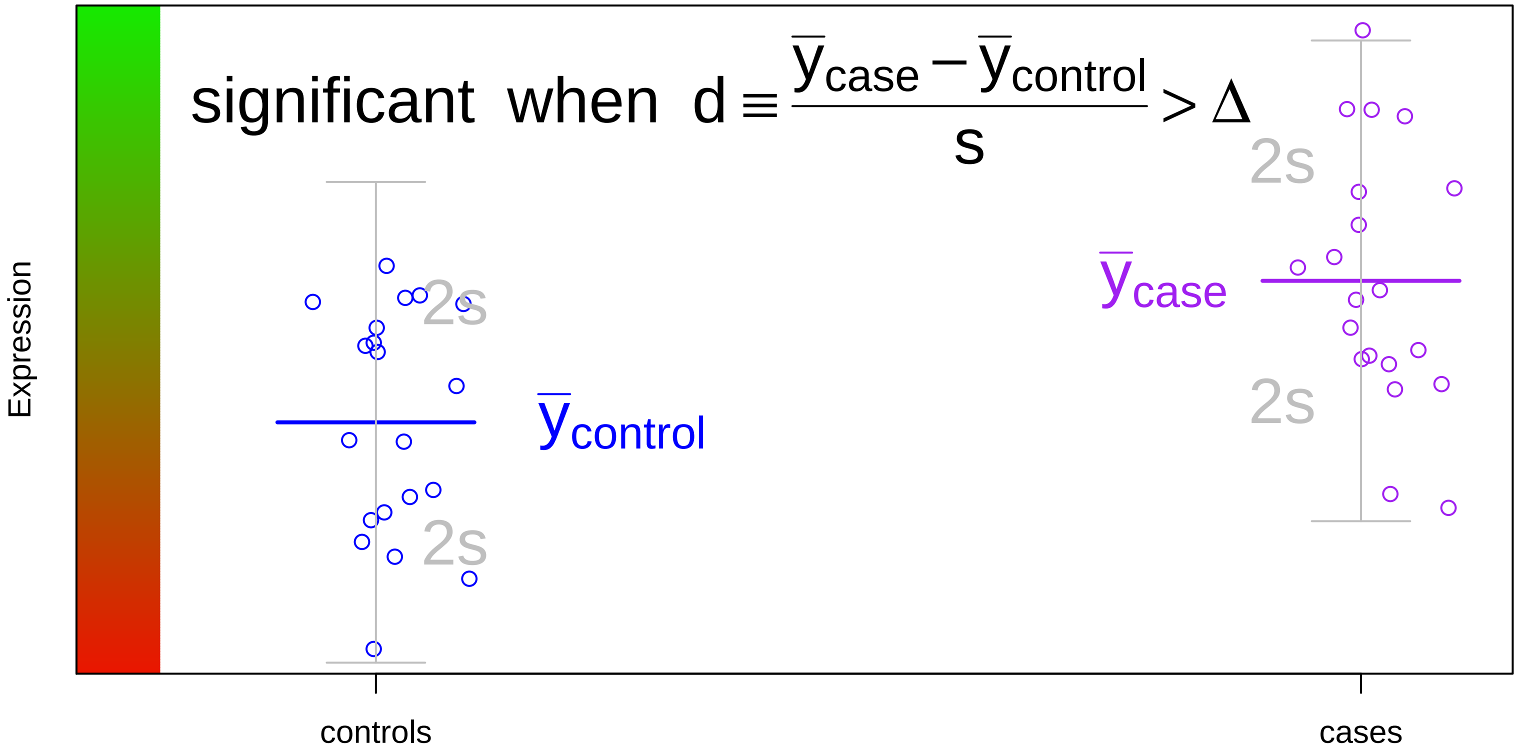
SAM is a popular new method (Tusher et al 2001) which identifies **differentially expressed genes**



i.e. large red/green difference between cases and controls

Signficance Analysis of Microarrays (SAM)

Why so popular? Here's the traditional method;



Do this ×30,000 genes; d in each is **quite unstable**. Small values of s give large d, which may give **false positive** results

Signficance Analysis of Microarrays (SAM)

SAM has a quick fix for this problem;

Traditional

$$d_i = \frac{\bar{y}_{i,\text{case}} - \bar{y}_{i,\text{control}}}{s_i}$$

SAM

$$d_i = \frac{\bar{y}_{i,\text{case}} - \bar{y}_{i,\text{control}}}{s_i + s_0}$$

For each gene (each i), SAM's s_0 **borrows strength** from the other genes.

SAM (and `siggenes`) then does some clever permutation testing to produce False Discovery Rates

Signficance Analysis of Microarrays (SAM)

Golub et al (1999) give differential expression for 3,051 genes, in 27 ‘controls’ (ALL) and 11 ‘cases’ (AML)

```
> library(multtest)
> data(golub)
> table(golub.cl)
 0  1
27 11
```

Now let’s do the SAM analysis; we give a **random seed** for the permutations – and tell `R` how many to do;

```
> sam.out <- sam(golub, golub.cl, B=100, rand = 123)
```

... takes only a few seconds. Use `B=1000` or more if you can

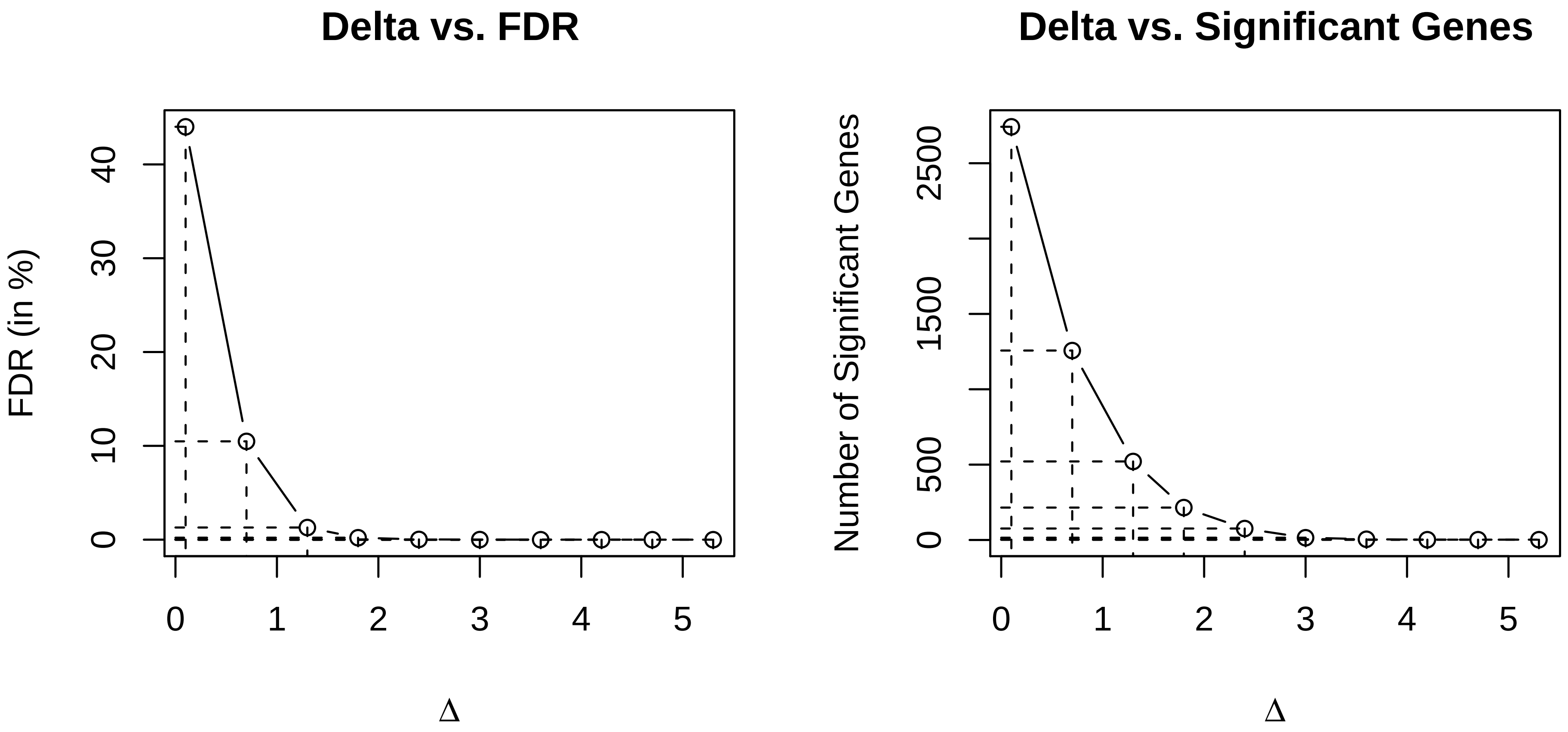
Signficance Analysis of Microarrays (SAM)

```
> summary(sam.out)
 s0 = 0.0584  (The 0 % quantile of the s values.)
 Number of permutations: 1000
   Delta    p0    False Called      FDR cutlow cutup   j2   j1
1    0.1 0.499 2420.329   2742 0.440123 -0.160 0.244 1446 1756
2    0.7 0.499  264.208   1257 0.104804 -1.247 1.438  746 2541
3    1.3 0.499   13.526    521 0.012945 -2.270 2.488  325 2856
4    1.8 0.499    0.903    215 0.002094 -3.119 3.311  139 2976
5    2.4 0.499    0.043     76 0.000282 -4.157 4.259   44 3020
6    3.0 0.499    0.003     15 9.97e-05 -5.577 5.139    4 3041
7    3.6 0.499        0      5        0   -Inf 5.971    0 3047
8    4.2 0.499        0      2        0   -Inf 7.965    0 3050
9    4.7 0.499        0      2        0   -Inf 7.965    0 3050
10   5.3 0.499        0      2        0   -Inf 7.965    0 3050
```

p0 is the **prior** probability of differential expression. Also note that the FDR values are **rounded**

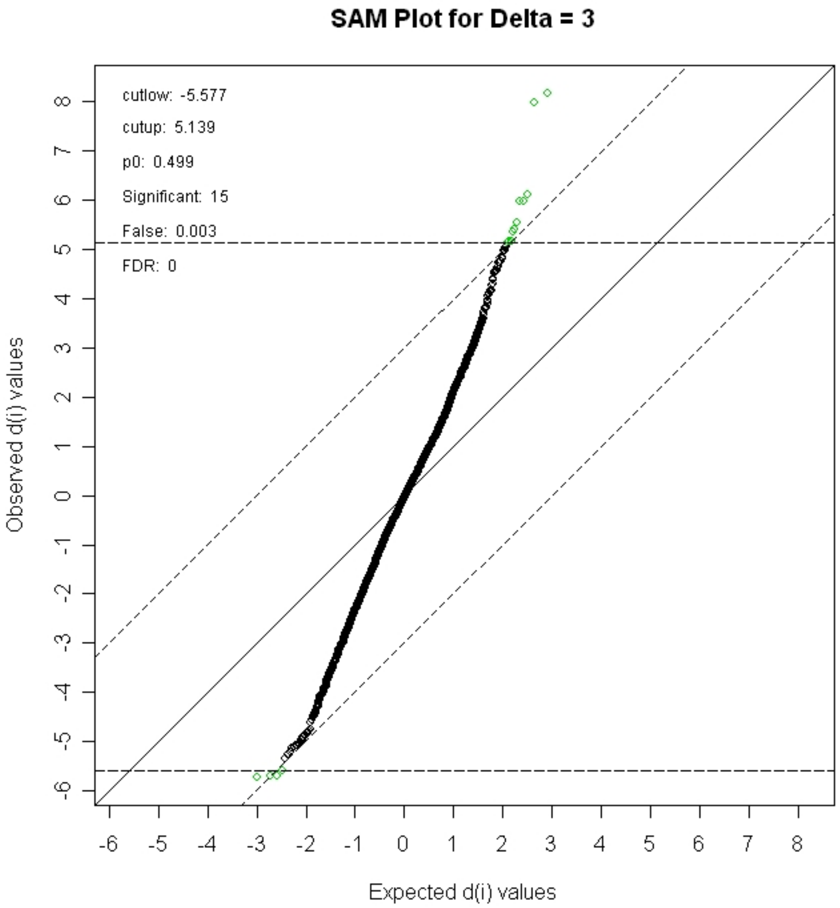
Signficance Analysis of Microarrays (SAM)

```
> plot(sam.out)
```



Signficance Analysis of Microarrays (SAM)

```
> plot(sam.out, 3) #specifies Delta
```



Microarray analysis with `limma`

The `limma` package can do **several** analyses for microarrays. It reads in **raw data**, in standard formats

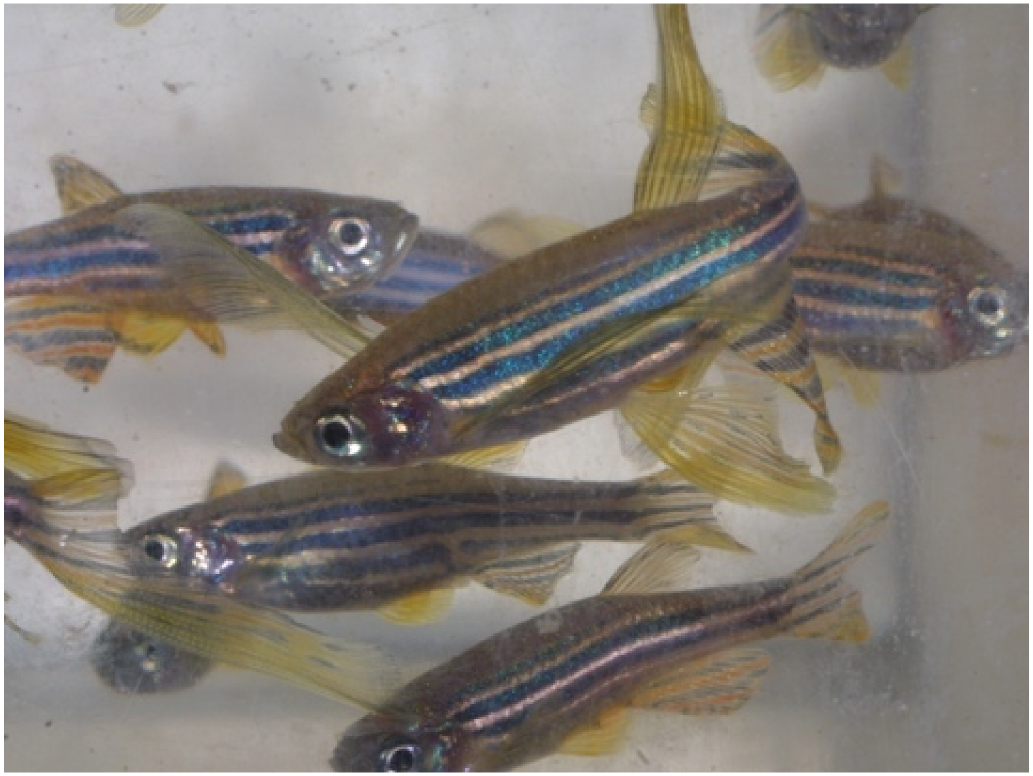
```
> library(limma)
> my.files <- dir(pattern=".spot")
> my.files
[1] "swirl.1.spot" "swirl.2.spot" "swirl.3.spot" "swirl.4.spot"
> RG <- read.maimages(my.files, source="spot")
Read swirl.1.spot
Read swirl.2.spot
Read swirl.3.spot
Read swirl.4.spot
```

Microarray analysis with `limma`

What is `swirl`? A mutation affecting **zebrafish**

Microarray analysis with `limma`

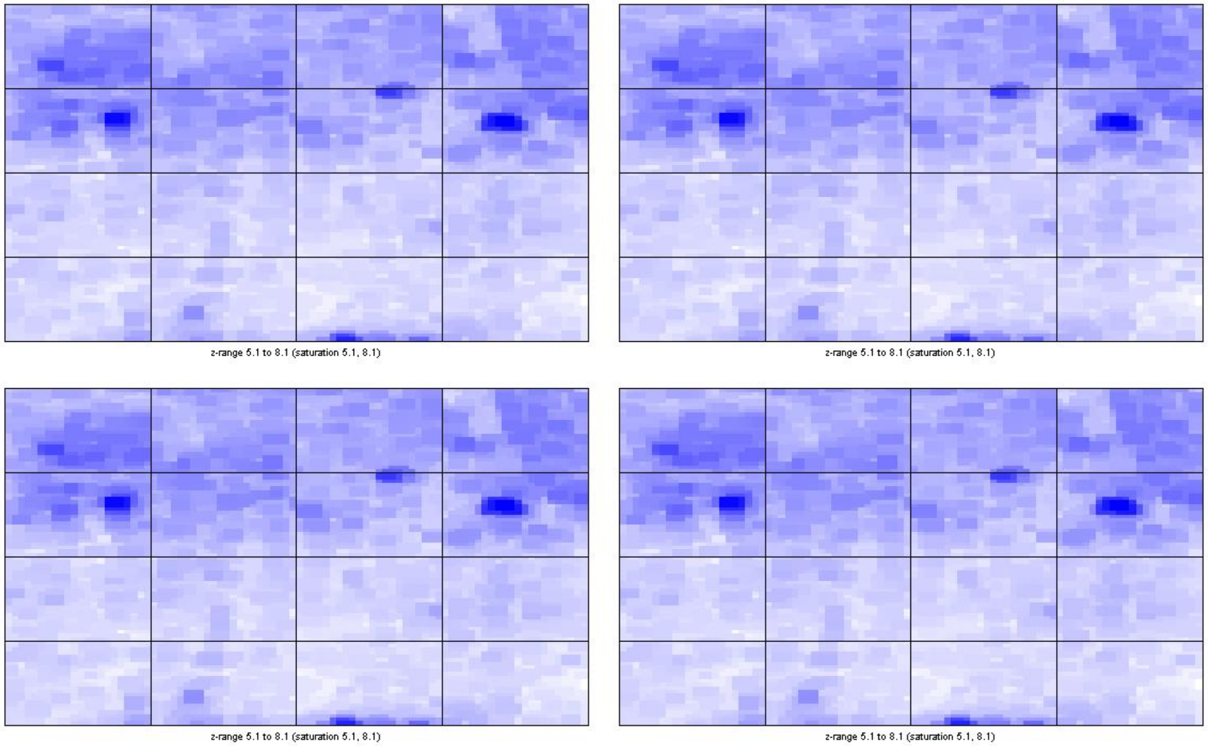
What is `swirl`? A mutation affecting **zebrafish**



We have 2 mutants, and 2 wild-type fish

Microarray analysis with `limma`

Here are the red intensities from each microarray;

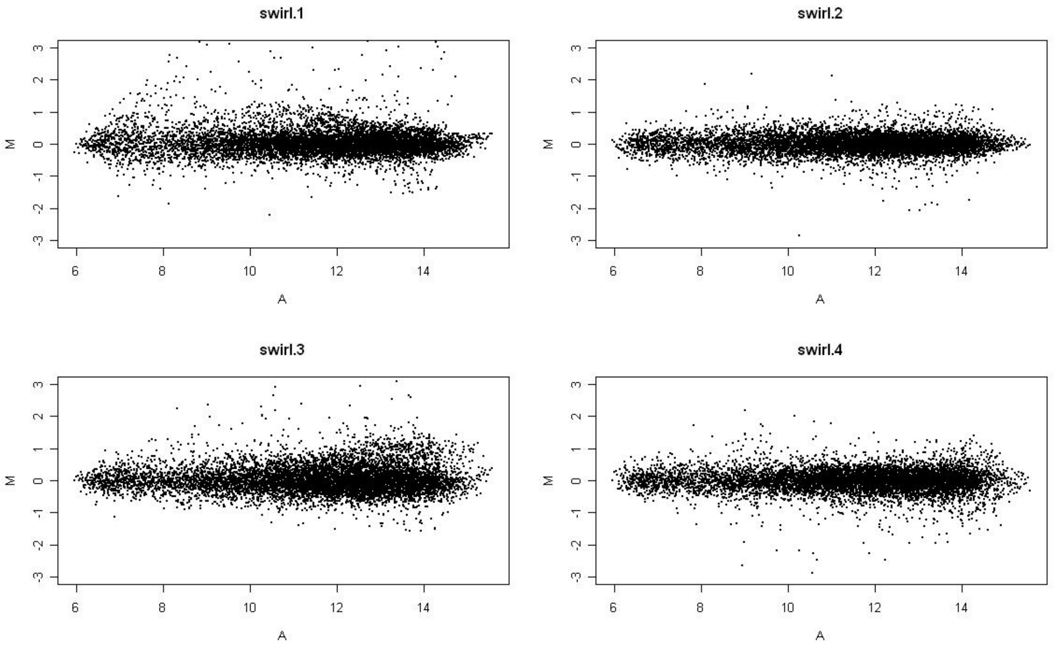


– need to **normalize** each array (or get a bigger sample!)

Microarray analysis with `limma`

`limma` has ‘default’ normalization techniques

```
> MA1 <- normalizeWithinArrays(RG)
> MA2 <- normalizeBetweenArrays(MA1)
```



Can you guess where the ‘signals’ are?

Microarray analysis with `limma`

`limma` fits ‘plain’ models to each gene, and also ‘robustifies’ them with an Empirical Bayes approach (much the same as SAM)

```
> fit1 <- lmFit(MA2, design=c(-1,1,-1,1))
> options(digits=3); toptable(fit, n=30, adjust="fdr")
         M     t  P.Value adj.P.Val    B
2961 -2.66 -20.8 1.44e-07   0.00121 7.55
3723 -2.19 -17.6 4.59e-07   0.00194 6.75
1611 -2.19 -16.1 8.44e-07   0.00238 6.29
7649 -1.60 -14.2 2.02e-06   0.00326 5.58
515   1.26  13.7 2.55e-06   0.00326 5.39

> fit2 <- eBayes(fit1)
> options(digits=3); topTable(fit2, n=30, adjust="fdr")
     Block Row Column      ID   Name     M     A     t  P.Value adj.P.Val    B
2961     6  14      9 fb85d05 18-F10 -2.66 10.33 -20.8 1.44e-07   0.00121 7.55
3723     8   2      3 control   Dlx3 -2.19 13.24 -17.6 4.59e-07   0.00194 6.75
1611     4   2      3 control   Dlx3 -2.19 13.45 -16.1 8.44e-07   0.00238 6.29
7649    15  11     17 fb58g10 11-L19 -1.60 13.49 -14.2 2.02e-06   0.00326 5.58
515      1  22     11 fc22a09 27-E17  1.26 13.19  13.7 2.55e-06   0.00326 5.39
```