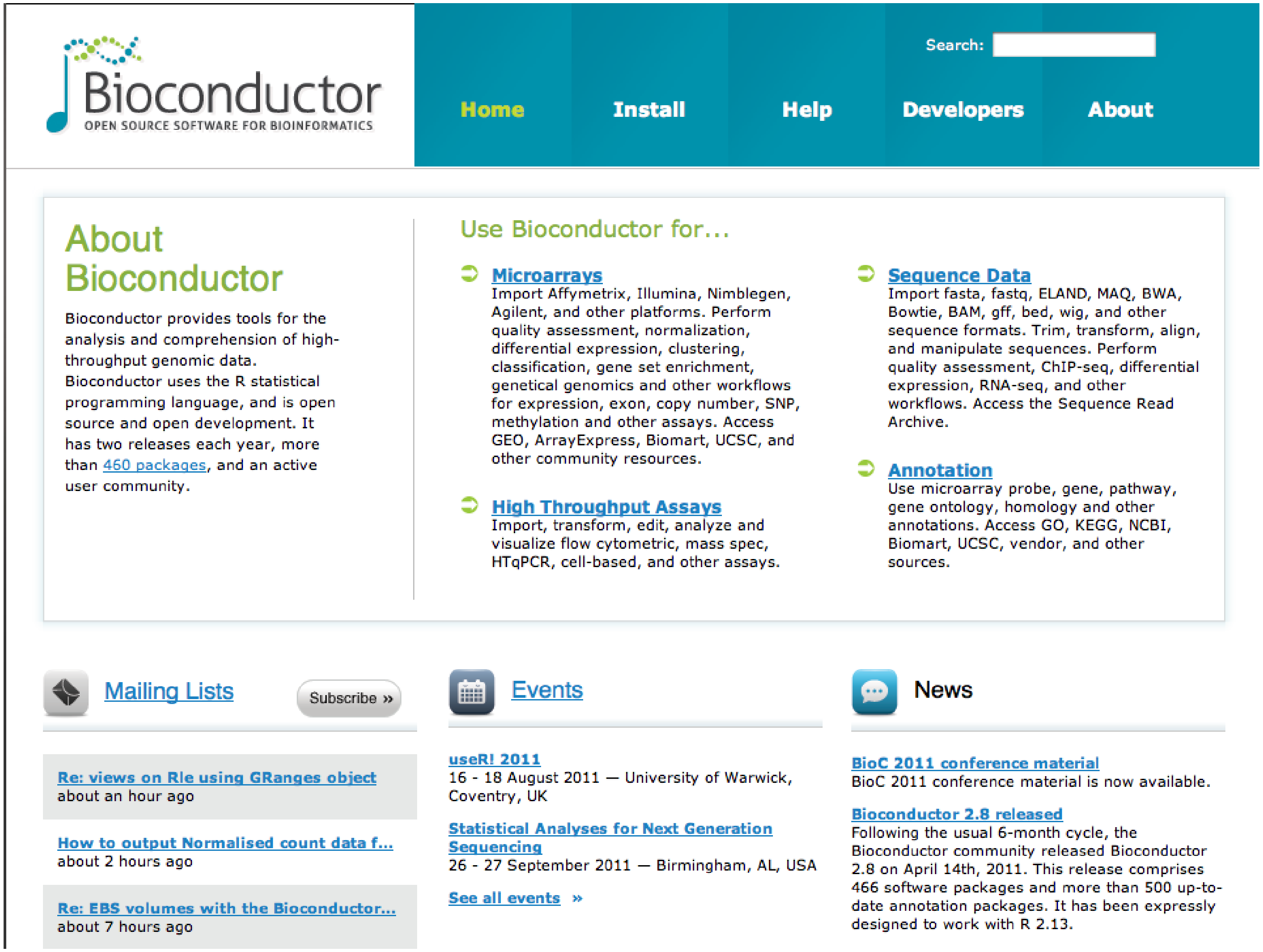
8. Bioconductor Intro and Annotation

Ken Rice
Thomas Lumley

Universities of Washington and Auckland

Seattle, July 2016

What is 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, more than 460 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.

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

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.

Mailing Lists

Re: views on Rle using GRanges object
about an hour ago

How to output Normalised count data f...
about 2 hours ago

Re: EBS volumes with the Bioconductor...
about 7 hours ago

Events

useR! 2011
16 - 18 August 2011 — University of Warwick, Coventry, UK

Statistical Analyses for Next Generation Sequencing
26 - 27 September 2011 — Birmingham, AL, USA

See all events »

News

BioC 2011 conference material
BioC 2011 conference material is now available.

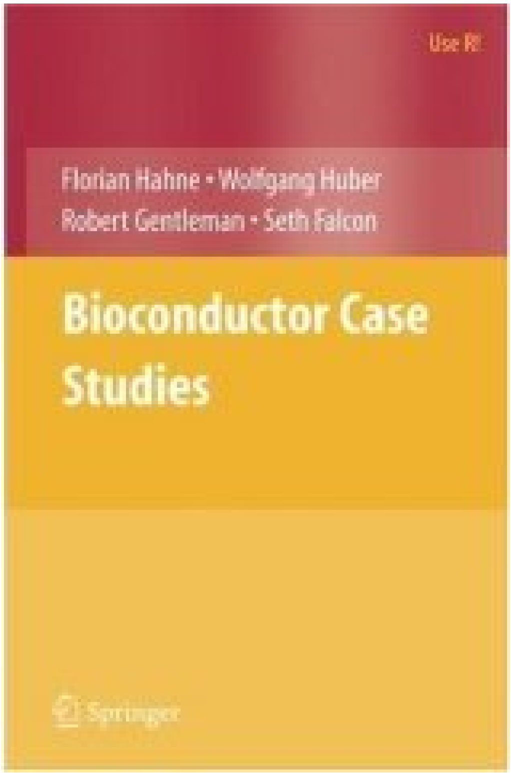
Bioconductor 2.8 released
Following the usual 6-month cycle, the Bioconductor community released Bioconductor 2.8 on April 14th, 2011. This release comprises 466 software packages and more than 500 up-to-date annotation packages. It has been expressly designed to work with R 2.13.

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. Also, the help manuals are free from ‘sales pitch’ and encourage appropriate use.

Bioconductor basics

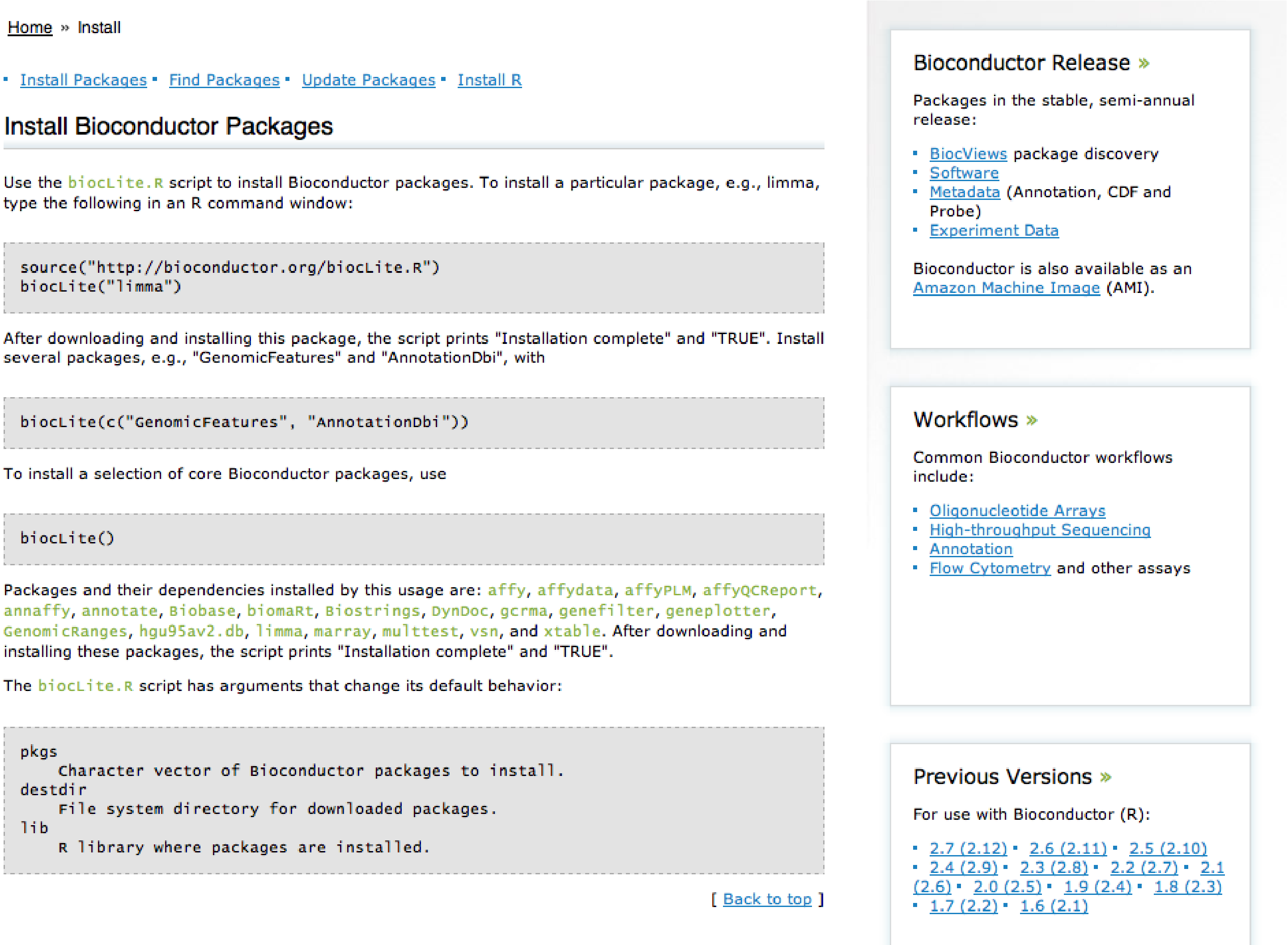

Use R!
Florian Hahne · Wolfgang Huber
Robert Gentleman · Seth Falcon
Bioconductor Case Studies
Springer

- 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. Hahne et al (above) is a helpful reference text

Bioconductor basics

Getting started...



Home » Install

- Install Packages ▪ Find Packages ▪ Update Packages ▪ Install R

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")
```

After downloading and installing this package, the script prints "Installation complete" and "TRUE". Install several packages, e.g., "GenomicFeatures" and "AnnotationDbi", with

```
biocLite(c("GenomicFeatures", "AnnotationDbi"))
```

To install a selection of core Bioconductor packages, use

```
biocLite()
```

Packages and their dependencies installed by this usage are: affy, affydata, affyPLM, affyQCReport, annaffy, annotate, Biobase, biomaRt, Biostrings, DynDoc, gcrma, genefilter, geneplotter, GenomicRanges, hgu95av2.db, limma, marray, multtest, vsn, and xtable. After downloading and installing these packages, the script prints "Installation complete" and "TRUE".

The biocLite.R script has arguments that change its default behavior:

```
pkgs
    Character vector of Bioconductor packages to install.
destdir
    File system directory for downloaded packages.
lib
    R library where packages are installed.
```

[Back to top]

Bioconductor Release »

Packages in the stable, semi-annual release:

- BiocViews package discovery
- Software
- Metadata (Annotation, CDF and Probe)
- Experiment Data

Bioconductor is also available as an Amazon Machine Image (AMI).

Workflows »

Common Bioconductor workflows include:

- Oligonucleotide Arrays
- High-throughput Sequencing
- Annotation
- Flow Cytometry and other assays

Previous Versions »

For use with Bioconductor (R):

- 2.7 (2.12) ▪ 2.6 (2.11) ▪ 2.5 (2.10)
- 2.4 (2.9) ▪ 2.3 (2.8) ▪ 2.2 (2.7) ▪ 2.1 (2.6) ▪ 2.0 (2.5) ▪ 1.9 (2.4) ▪ 1.8 (2.3)
- 1.7 (2.2) ▪ 1.6 (2.1)

Bioconductor basics

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

installs the following general-purpose libraries;

`Biobase, IRanges, AnnotationDbi`

... then you use e.g. `library("Biobase")` as before. (NB older versions used to download much more than this)

`vignette(package="Biobase")` tells you to look at `vignette("esApply")` for a worked example – a very helpful introduction. (Or use e.g. `openVignette()`, which is in the `Biobase` package itself)

Bioconductor basics

To get other packages, use the `source()` command as before, then use e.g.

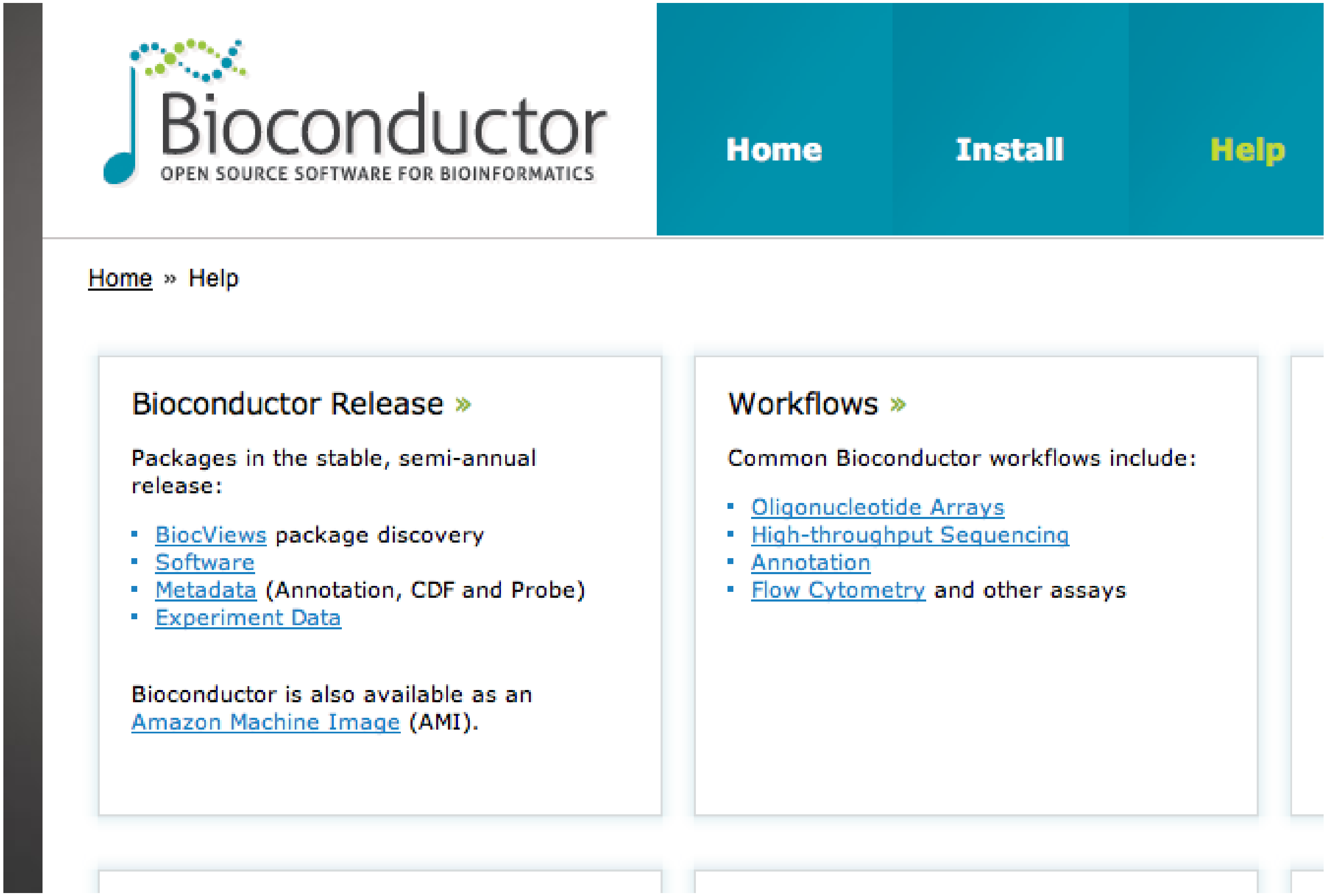
```
biocLite("SNPchip")
biocLite(c("limma", "siggenes"))
```

You do not need to type `biocLite()` again (even in a new R session). This would install the general-purpose packages again – which is harmless, but a waste of time.

Note; if, due to access privileges, you need to write to non-default directories, follow the onscreen commands and then start again. On Windows, 'Run as Administrator' may cut out this step.

What to install?

Back to the front page – click ‘Help’



Home » Help

Bioconductor Release »

Packages in the stable, semi-annual release:

- BiocViews package discovery
- Software
- Metadata (Annotation, CDF and Probe)
- Experiment Data

Bioconductor is also available as an Amazon Machine Image (AMI).

Workflows »

Common Bioconductor workflows include:

- Oligonucleotide Arrays
- High-throughput Sequencing
- Annotation
- Flow Cytometry and other assays

What to install?

- **Software** – probably what you want
- **Metadata** – e.g. annotation data, probe sequence data for microarrays of different types
- **Experiment data** – e.g. datasets from `hapmap.org`, some expression datasets

Simple QC graphics

The `splots` package plots values from 96 or 384-well plates, for QC purposes...

First, install it;

```
biocLite("splots")
```

Then load into R;

```
library("splots")
```

There is a single function: `plotScreen()` for displaying the results

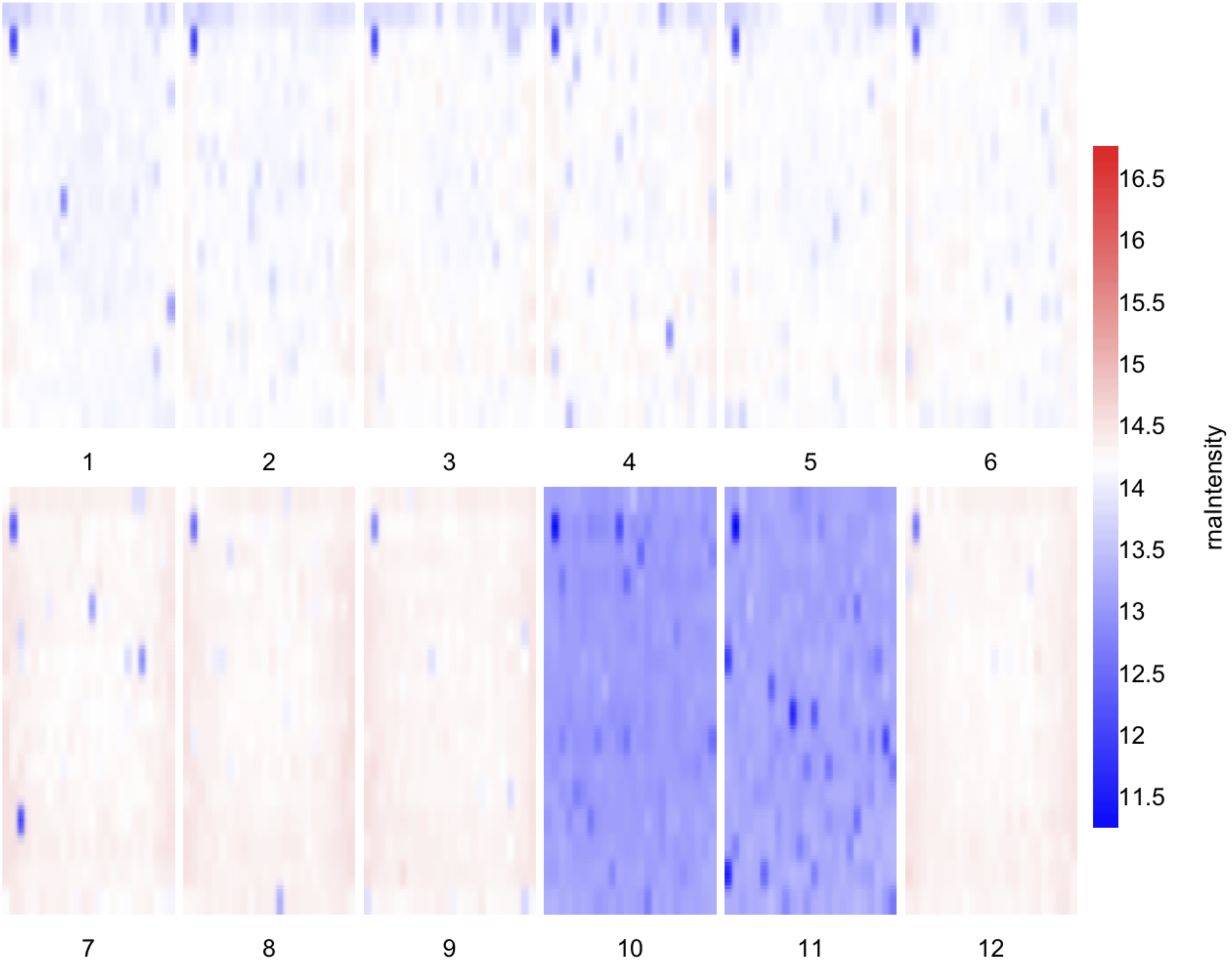
Example

The file `drosophila.rda` contains 12 of 114 plates from a RNAi gene-knockout study in fruit flies. Each spot represents a gene, and the intensity is low if knockout of that gene is lethal (data from the `RNAither` package)

```
load("drosophila.rda")
plotScreen(rnai)
```

The positive controls in the same position each plate are clear, and there are obvious plate effects that you might need to correct by normalization.

Example


1
2
3
4
5
6
7
8
9
10
11
12
16.5
16
15.5
15
14.5
14
13.5
13
12.5
12
11.5
rmaIntensity

Outline

One goal of Bioconductor is to provide efficient access inside R to the genome databases that are vital to interpreting associations.

We will look at a few of these

- annotate
- biomaRt
- genomeGraphs

The reason to have an R interface to these databases is to be able to analyze annotation data for *many* SNPs or RNA transcripts, as in e.g. bioinformatics work.

Online or stored data

Annotation data can be downloaded in a single file or retrieved for each query from an online database.

Local storage is faster, but may require too much space (e.g. Ensembl) or become obsolete too quickly.

Local storage is ideal for fixed annotation data such as gene names for a microarray or SNP chip.

Types of database

Translations of names: Affy probe `32972_at` is the gene NADPH oxidase 1 with symbol NOX1 and Ensembl gene id `ENSG00000007952`

Location: NOX1 is on Xq22.1, from 99984969 to 100015990, coded on the negative strand. There are 120 known polymorphisms (SNPs or indels) in this range.

Homology: The mouse version of NOX1 is also on the X chromosome, starting at 130621066 (and called Nox1)

Structure and function: NOX1 is a membrane protein (location), involved in voltage-gated ion channel activity (molecular function), and involved in signal transduction (biological process).

Annotate

Bioconductor distributes annotation packages for a wide range of gene expression microarrays. The annotate package is one way to use this annotation information.

```
> library("annotate")
> library("hgu95av2.db")
> library("GO.db")
```

loads the annotate package and the databases for the Gene Ontology and one of the Affymetrix human microarray chips.

Lookups

The databases are queried with `get()` or `mget()` for multiple queries

```
> mget(c("738_at", "40840_at", "32972_at"),  envir=hgu95av2GENENAME)
$‘738_at‘
[1] "5’-nucleotidase, cytosolic II"

$‘40840_at‘
[1] "peptidylprolyl isomerase F (cyclophilin F)"

$‘32972_at‘
[1] "NADPH oxidase 1"

> go<-get("738_at", envir=hgu95av2GO)
> names(go)
[1] "GO:0009117" "GO:0005829" "GO:0005737" "GO:0000166" "GO:0000287"
[6] "GO:0008253" "GO:0008253" "GO:0016787"
```

Lookups

```
> get("GO:0009117",  envir=GOTERM)
GOID: GO:0009117
Term: nucleotide metabolic process
Ontology: BP
Definition: The chemical reactions and pathways involving a
   nucleotide, a nucleoside that is esterified with (ortho)phosphate
   or an oligophosphate at any hydroxyl group on the glycose
   moiety; may be mono-, di- or triphosphate; this definition
   includes cyclic nucleotides (nucleoside cyclic phosphates).
Synonym: nucleotide metabolism
```

What lookups are there?

```
> library(help="hgu95av2.db")

hgu95av2ALIAS2PROBE  Map between Common Gene Symbol Identifiers and
                     Manufacturer Identifiers

> get("NOX1", envir=hgu95av2ALIAS2PROBE)
[1] "32972_at"   "32973_s_at"
```

You can also reverse a lookup table with `revmap()`

```
> get("NOX1", envir=revmap(hgu95av2SYMBOL))
[1] "32972_at"   "32973_s_at"
> get("X",revmap(hgu95av2CHR))
 [1] "1016_s_at"  "107_at"     "1100_at"    "112_g_at"    "1155_at"
 .... and lots more
```

BioMart

BioMart (`www.biomart.org`)is a query-oriented data management system developed jointly by the European Bioinformatics Institute (EBI) and Cold Spring Harbor Laboratory (CSHL).

`biomaRt` is an R interface to BioMart systems, in particular to Ensembl (`www.ensembl.org`). Ensembl is a joint project between EMBL - European Bioinformatics Institute (EBI) and the Wellcome Trust Sanger Institute (WTSI) to develop a software system which produces and maintains automatic annotation on selected eukaryotic genomes.

BioMart

We begin by making a connection to a BioMart of choice;

```
> library(biomaRt)
Loading required package: RCurl
> listMarts()
                              biomart
1                        ensembl                        ENSEMBL GENES 63 (SANGER U
2                            snp                   ENSEMBL  VARIATION 63 (SANGER U
3            functional_genomics                   ENSEMBL REGULATION 63 (SANGER U
4                           vega                                VEGA 43  (SANGER U
5               bacteria_mart_10                        ENSEMBL BACTERIA 10 (EBI U
6                  fungi_mart_10                           ENSEMBL FUNGI 10 (EBI U
7            fungi_variations_10                 ENSEMBL FUNGI VARIATION 10 (EBI U
8                metazoa_mart_10                         ENSEMBL METAZOA 10 (EBI U
9          metazoa_variations_10               ENSEMBL METAZOA VARIATION 10 (EBI U
...
60            ENSEMBL_MART_PLANT          GRAMENE 30 ENSEMBL GENES (CSHL/CORNELL U
61        ENSEMBL_MART_PLANT_SNP              GRAMENE 30 VARIATION (CSHL/CORNELL U
62             GRAMENE_MARKER_30                GRAMENE 30 MARKERS (CSHL/CORNELL U
63                GRAMENE_MAP_30               GRAMENE 30 MAPPINGS (CSHL/CORNELL U
64                      QTL_MART                 GRAMENE 32 QTL DB (CSHL/CORNELL U
65              salmosalar2_mart            UNIGENE SALMO SALAR DATABASE (CMM CHIL
66                   trucha_mart    UNIGENE ONCORHYNCHUS MYKISS DATABASE (CMM CHIL
> ens <- useMart("ensembl") # connector object
```

BioMart

We then make a connection to a chosen database;

```
> listDatasets(ens)
                          dataset                                   description
                          dataset                                 description
1          oanatinus_gene_ensembl      Ornithorhynchus anatinus genes (OANA5)
2           tguttata_gene_ensembl     Taeniopygia guttata genes (taeGut3.2.4)
3         cporcellus_gene_ensembl             Cavia porcellus genes (cavPor3)
4         gaculeatus_gene_ensembl      Gasterosteus aculeatus genes (BROADS1)
5          lafricana_gene_ensembl          Loxodonta africana genes (loxAfr3)
...
30         pvampyrus_gene_ensembl           Pteropus vampyrus genes (pteVam1)
...
58           btaurus_gene_ensembl                   Bos taurus genes (UMD3.1)
59          meugenii_gene_ensembl           Macropus eugenii genes (Meug_1.0)
60         sharrisii_gene_ensembl       Sarcophilus harrisii genes (DEVIL7.0)
61       cfamiliaris_gene_ensembl          Canis familiaris genes (CanFam3.1)
> hsap <- useDataset("hsapiens_gene_ensembl",mart=ens)
```

BioMart

The getGene() function queries the database for gene information. It accepts many forms of gene identifier, e.g. Entrez, HUGO, Affy transcript.

```
> getGene(id=1440, type="entrezgene", mart=hsap)
  entrezgene hgnc_symbol
1       1440        CSF3
                                                              description
1 colony stimulating factor 3 (granulocyte) [Source:HGNC Symbol;Acc:2438]
  chromosome_name  band strand start_position end_position ensembl_gene_id
1              17 q21.1      1       38171614     38174066 ENSG00000108342

>  getGene(id=c("AGT","AGTR1"), type="hgnc_symbol", mart=hsap)
  hgnc_symbol hgnc_symbol
1         AGT         AGT
2       AGTR1       AGTR1

1 angiotensinogen (serpin peptidase inhibitor, clade A, member 8) [Source:HGNC Sym
2                                 angiotensin II receptor, type 1 [Source:HGNC Sym
  chromosome_name  band strand start_position end_position ensembl_gene_id
1               1 q42.2     -1      230838269    230850043 ENSG00000135744
2               3   q24      1      148415571    148460795 ENSG00000144891
```

BioMart

getBM() is more general than getGene(). It specifies a list of filters for selecting genes or SNPs and attributes to return from the database.

```
> affyids <- c("202763_at", "209310_s_at", "207500_at")
>  getBM(attributes = c("affy_hg_u133_plus_2", "hgnc_symbol", "chromosome_name",
    "start_position", "end_position", "band"), filters = "affy_hg_u133_plus_2",
     values = affyids, mart = hsap)
     affy_hg_u133        hgnc   chromosome_name start_position end_position  band
1           202763_at      CASP3         4      185785844    185807623 q35.1
2           207500_at      CASP5        11      104370180    104384957 q22.3
3         209310_s_at      CASP4        11      104318804    104344535 q22.3
```

listAttributes(hsap) and listFilters(hsap) list the available attributes and filters... there are hundreds of these.

BioMart

```
> getBM(mart=hsap, attributes=c("band","hgnc_symbol"),
         filters=c("band_start","band_end","chromosome_name"),
         values=list("p21.33","p21.33",6))
      band hgnc_symbol
1   p21.33
2   p21.33    SNORD117
3   p21.33     SNORA38
4   p21.33     SNORD48
5   p21.33     SNORD52
6   p21.33      MIR877
7   p21.33     MIR1236
8   p21.33      GTF2H4
9   p21.33       VARS2
10  p21.33       SFTA2
11  p21.33       DPCR1
12  p21.33       MUC21
...
121 p21.33      HSPA1A
122 p21.33        TNXB
123 p21.33       STK19
124 p21.33         C4A
125 p21.33         C4B
```

Homology

getLDS() combines two data marts, for example to homologous genes in other species. We can look up the mouse equivalents of a particular Affy transcript, or of the NOX1 gene;

```
> human = useMart("ensembl", dataset = "hsapiens_gene_ensembl")
> mouse = useMart("ensembl", dataset = "mmusculus_gene_ensembl")
> getLDS(attributes = c("hgnc_symbol","chromosome_name", "start_position"),
+ filters = "hgnc_symbol", values = "NOX1", mart = human,
+ attributesL = c("chromosome_name","start_position","external_gene_id"),
+ martL = mouse)
    V1 V2        V3 V4        V5   V6
1 NOX1  X 100098313  X 134086421 Nox1
```

The mouse gene name is the same as the human one apart from capitalisation.

Homology

The `getSequence()` function looks up DNA or protein sequences by chromosome position or gene identifiers;

```
> agt<-getSequence(id="AGT",type="hgnc_symbol", seqType="peptide",mart=hsap)
> agt

1 MRKRAPQSEMAPAGVSLRATILCLLAWAGLAAGDRVYIHPFHLVIHNESTCEQLAKANAGKPKDPTFIPAPIQAKTS
PVDEKALQDQLVLVAAKLDTEDKLRAAMVGMLANFLGFRIYGMHSELWGVVHGATVLSPTAVFGTLASLYLGALDHTAD
RLQAILGVPWKDKNCTSRLDAHKVLSALQAVQGLLVAQGRADSQAQLLLSTVVGVFTAPGLHLKQPFVQGLALYTPVVL
PRSLDFTELDVAAEKIDRFMQAVTGWKTGCSLMGASVDSTLAFNTYVHFQGKMKGFSLLAEPQEFWVDNSTSVSVPMLS
GMGTFQHWSDIQDNFSVTQVPFTESACLLLIQPHYASDLDKVEGLTFQQNSLNWMKKLSPRTIHLTMPQLVLQGSYDLQ
DLLAQAELPAILHTELNLQKLSNDRIRVGEVLNSIFFELEADEREPTESTQQLNKPEVLEVTLNRPFLFAVYDQSATAL
HFLGRVANPLSTA*
```

Example: finding chromosomes

We had a set 1524 SNPs, of which 409 did not have their chromosome listed.

I needed to know which SNPs were on the X chromosome, to estimate sex from DNA intensity and heterozygous X-chromosome loci, for QC.

```
> head(unknown)
[1] "UGT1A3-001449-0_B_R_1538822" "LIPC-002761-0_B_R_1538453"
[3] "CETP-001265-0_B_R_1538254"   "F8-165293-0_T_F_1538626"
[5] "CPB2-051208-0_B_F_1539402"   "VDRDIL-1355-0_T_F_1539404"
```

A hand-search would be easy but tedious, so we want an automated approach

Example: finding chromosomes

First extract the gene names

```
genes <- sapply(unknown, function(snp) strsplit(snp,"-")[[1]][1])
ugenes <- unique(genes)
```

Now call out to Ensembl

```
getBM(attributes="chromosome_name", filters="hgnc_symbol",values=ugenes,
  mart=hsap)
```

works for all except VRDIL, which isn't recognized.

Finding SNPs

Human SNPs (and short indels) are in a separate database from gene information. We can look up known SNPs and other polymorphisms for the NOX1 gene

```
> snpmart = useMart("snp", dataset = "hsapiens_snp")
Checking attributes ... ok
Checking filters ... ok
> getBM(c("refsnp_id", "allele", "chrom_start", "chrom_strand"),
       filters = c("chr_name", "chrom_start","chrom_end"),
       values = list("X",99984969,100015990), mart = snpmart)
         refsnp_id             allele chrom_start chrom_strand
1     rs7054049              T/A    99985184            1
2    rs60975472              G/T    99985304            1
3    rs58902780              A/G    99985571            1
4   rs182188185              G/A    99985618            1
5   rs186748080              A/G    99985798            1
```

More metadata

The `citation()` function prints out information about how to cite a package

```
> citation("biomaRt")

To cite the biomaRt package in publications use:

  Mapping identifiers for the integration of genomic datasets with the
  R/Bioconductor package biomaRt. Steffen Durinck, Paul T. Spellman,
  Ewan Birney and Wolfgang Huber, Nature Protocols 4, 1184-1191 (2009).

  BioMart and Bioconductor: a powerful link between biological
  databases and microarray data analysis. Steffen Durinck, Yves Moreau,
  Arek Kasprzyk, Sean Davis, Bart De Moor, Alvis Brazma and Wolfgang
  Huber, Bioinformatics 21, 3439-3440 (2005).
```

Citations are one way academic software authors can prove to funders and promotion committees that software is worthwhile.

GenomeGraphs

This package makes pretty pictures from the annotation data.

For example, a pictures showing the standard and alternative splices for the NOX1 gene and the location of the gene on the X chromosome

```
> library(GenomeGraphs)
> gene <- makeGene(id = "NOX1", type = "hgnc_symbol",
      biomart = hsap)
> transcript<-makeTranscript(id="NOX1",type="hgnc_symbol",
      biomart=hsap)
> ideogram <- makeIdeogram(chromosome ="X")
> gdPlot(list(ideogram,gene,transcript))
```

GenomeGraphs