# CSAMA Brixen 2014: Visualization lab

Vince Carey

Modified: 31 May, 2014. Compiled: June 23, 2014

## Contents

## 1 Exploratory data analysis: transcription factor binding in yeast

The abstract to Harbison et al. 2004, Transcriptional regulatory code of a eukaryotic genome, reads:

```
DNA-binding transcriptional regulators interpret the genome's regulatory code by binding to
specific sequences to induce or repress gene expression. Comparative genomics has
recently been used to identify potential cis-regulatory sequences within the yeast
genome on the basis of phylogenetic conservation, but this information alone
does not reveal if or when transcriptional regulators occupy these binding
sites. We have constructed an initial map of yeast's transcriptional regulatory
code by identifying the sequence elements that are bound by regulators
under various conditions and that are conserved among Saccharomyces species. The
organization of regulatory elements in promoters and the environment-dependent use of
these elements by regulators are discussed. We find that environment-specific use
```

```
   of regulatory elements predicts mechanistic models for the function of a
   large population of yeast's transcriptional regulators.
```

The data for genome-wide location analysis on which this is based is found in the *harbChIP* package.

```
> library(harbChIP)
> data(harbChIP)
```

```
> harbChIP
```

```
ExpressionSet (storageMode: lockedEnvironment)
assayData: 6230 features, 204 samples
  element names: exprs, se.exprs
protocolData: none
phenoData
  sampleNames: A1 (MATA1) ABF1 ... ZMS1 (204 total)
  varLabels: txFac
  varMetadata: labelDescription
featureData
  featureNames: YAL001C YAL002W ... MRH1 (6230 total)
  fvarLabels: ID PLATE ... REV_SEQ (12 total)
  fvarMetadata: labelDescription
experimentData: use 'experimentData(object)'
  pubMedIds: 15343339
Annotation:
```

While the ExpressionSet container is used, the columns correspond to DNA-binding proteins used in ChIP-chip experiments on arrays composed of probes for intergenic regions of the yeast genome.

```
> sampleNames(harbChIP)[1:10]
```

```
 [1] "A1 (MATA1)" "ABF1"       "ABT1"       "ACA1"       "ACE2"       "ADR1"      
 [7] "AFT2"       "ARG80"      "ARG81"      "ARO80"     
```

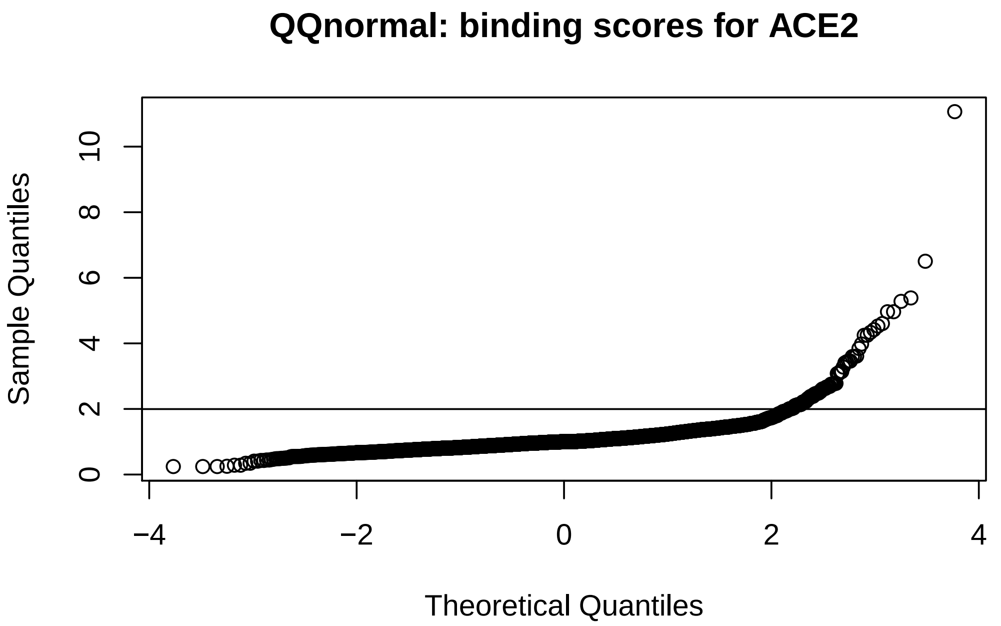
## 1.1 Outlier detection and declaration of TF binding

How can we decide which genes are regulated by the ACE2 transcription factor? The viz14 package has some tools to help visualize the associated genome-wide binding score profile. The `calout.detect` procedure of the *parody* package is used to test for outliers according to a formal procedure, and a line is drawn at the outlier boundary.

```
> library(viz14)
> ace2 = makebs("ACE2")
> ace2
```

```
bindingScores for  ACE2
top 5:
  YDR230W   YHR143W   YBR158W   YFR017C   YER124C   YER125W 
11.067550  6.505691  5.386609  5.281784  4.965746  4.965746 
```

```
> QQnorm(ace2)
```



The list of genes that would declared to be bound on the basis of 'outlying binding score' can be obtained as follows:

```
> ace2bnd = boundGenes(ace2)
```

## Exercises 1

1. Obtain a list of names of putatively bound genes for the transcription factors ACE2, SWI5, SWI6, SWI4, MBP1, FKH1, FKH2, NDD1, MCM1.
2. Using the VennDiagram package, assess overlap between five of these bound sets.
3. Check the documentation for `calout.detect`. How would you apply the standard boxplot outlier rules to define binding events?

```
> library(VennDiagram)
> v1 = venn.diagram( bg[1:5], filename=NULL )
> grid.draw(v1)
```

## 1.2 Associating binding factors with peak expression times in the cell cycle

The `trigFits` matrix in the *viz14* package includes information from de Lichtenberg, "Comparison of computational methods for the identification of cell cycle-regulated genes", Bioinformatics 2005. The location of peak expression times is reported for 300 genes considered to have periodic expression patterns. The units are percentage of time elapsed from M/G1 boundary to peak expression, in variable `dtf`.

```
> data(trigFits)
> summary(trigFits[, "dtf"])
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
   0.00   16.00   27.00   38.09   64.00   99.00    5878
```

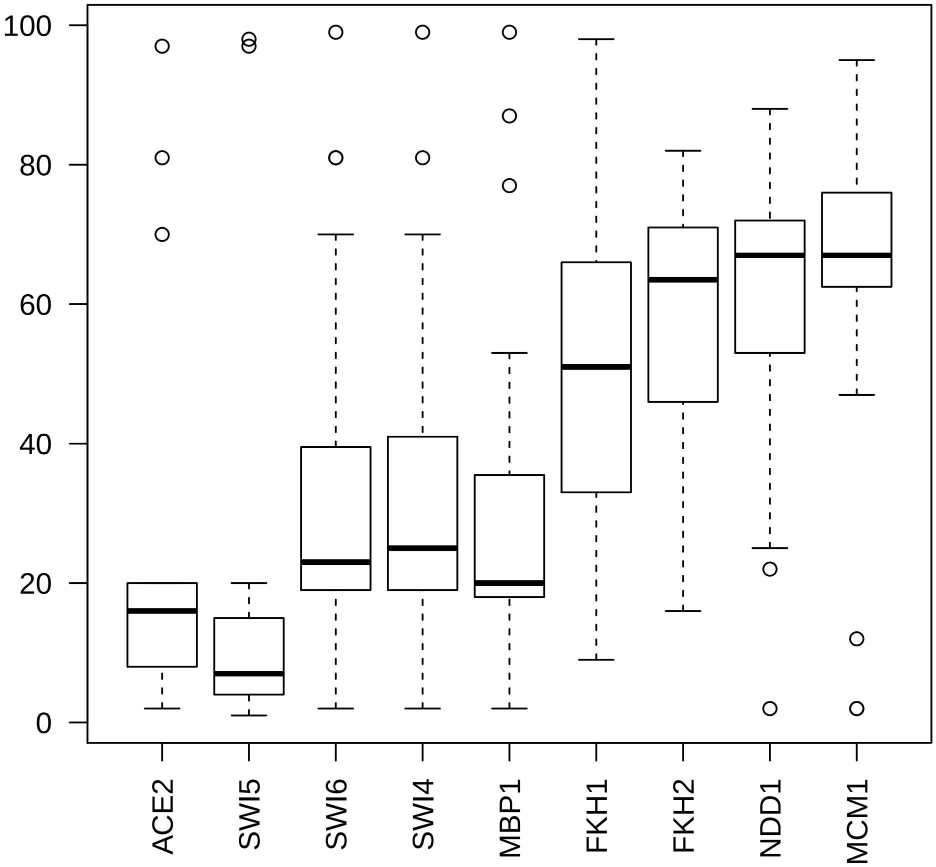
Here we obtain the cell cycle percentages when available, for gene promoters to which various TFs were asserted to bind using the outlier detection procedure described above.

```
> facs = c("ACE2", "SWI5", "SWI6", "SWI4", "MBP1",
+    "FKH1", "FKH2", "NDD1", "MCM1")
> bg = lapply(facs, function(x) boundGenes(makebs(x)))
> bgrps = lapply(bg, function(x) trigFits[
+    intersect(x, rownames(trigFits)), "dtf" ] )
> names(bgrps) = facs
> sapply(bgrps, function(x) length(na.omit(x)))
```

```
ACE2 SWI5 SWI6 SWI4 MBP1 FKH1 FKH2 NDD1 MCM1
  13   12   64   54   40   26   40   41   23
```

## Exercises 2

1. Interpret the following plot:
   `> boxplot(bgrps, las=2)`



2. The TF SKN7 was omitted from `facs`. Introduce the timings for gene group associated with SKN7 into the boxplot. Interpret the new display in terms of potential combinatorial relations among TFs? How do you reorder the plot for clearer ingestion?
3. Your paper on this finding has been rejected with the comment that the figure does not include any appraisal of statistical significance. How do you respond?

# 2 Modeling transcription regulation in the yeast cell cycle

## 2.1 Spellman's data on expression after colony synchronization

The *yeastCC* package includes an *ExpressionSet* instance collecting information on a number of cell cycle studies.

```
> library(yeastCC)
> data(spYCCES)
> spYCCES
```

```
ExpressionSet (storageMode: lockedEnvironment)
assayData: 6178 features, 77 samples
  element names: exprs
protocolData: none
phenoData
  sampleNames: cln3_40 cln3_30 ... elu_390 (77 total)
  varLabels: syncmeth time
  varMetadata: labelDescription
featureData: none
experimentData: use 'experimentData(object)'
  pubMedIds: 9843569
Annotation:
```

```
> experimentData(spYCCES)
```

```
Experiment data
  Experimenter name: Spellman PT
  Laboratory: Department of Genetics, Stanford University Medical Center, Stanford, California 94306-5120,
  Contact information:
  Title: Comprehensive identification of cell cycle-regulated genes of the yeast Saccharomyces cerevisiae b
  URL:
  PMIDs: 9843569

  Abstract: A 150 word abstract is available. Use 'abstract' method.
```

The package includes a character vector enumerating 800 genes deemed transcriptionally regulated in concert with cell cycling.

```
> data(orf800)
> orf800[1:4]
```

```
[1] "YAL022C" "YAL040C" "YAL053W" "YAL067C"
```

The abstract of the paper is bound with the data.

```
   We sought to create a comprehensive catalog of yeast genes whose
   transcript levels vary periodically within the cell cycle. To this end,
   we used DNA microarrays and samples from yeast cultures synchronized by
   three independent methods: alpha factor arrest, elutriation, and arrest of a
   cdc15 temperature-sensitive mutant. Using periodicity and correlation algorithms, we identified 800
   genes that meet an objective minimum criterion for cell cycle regulation.
   In separate experiments, designed to examine the effects of inducing either
   the G1 cyclin Cln3p or the B-type cyclin Clb2p, we found
   that the mRNA levels of more than half of these 800
   genes respond to one or both of these cyclins. Furthermore, we
   analyzed our set of cell cycle-regulated genes for known and new
   promoter elements and show that several known elements (or variations thereof)
   contain information predictive of cell cycle regulation. A full description and
   complete data sets are available at http://cellcycle-www.stanford.edu
```

We'll focus on samples analyzed after synchronization with alpha pheromone.

```
> alp = spYCCES[ , spYCCES$syncmeth=="alpha"]
> alp
```

```
ExpressionSet (storageMode: lockedEnvironment)
assayData: 6178 features, 18 samples
  element names: exprs
protocolData: none
phenoData
  sampleNames: alpha_0 alpha_7 ... alpha_119 (18 total)
  varLabels: syncmeth time
  varMetadata: labelDescription
featureData: none
experimentData: use 'experimentData(object)'
  pubMedIds: 9843569
Annotation:
```

```
> table(alp$time)
```

```
  0   7  14  21  28  35  42  49  56  63  70  77  84  91  98 105 112 119
  1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
```

One observation every seven minutes!

## 2.2 A nonlinear regression model

We'll pick a gene with a normalized transcription trajectory that appears periodic. A simple visualization is given in Figure 1.

A simple sinusoidal model is easy to compute with `nls`.

```
> m1 = nls(yal040c~b*sin(d+a*time),data=df,start=list(d=.1,b=1,a=.1))
> m1
```

```
Nonlinear regression model
  model: yal040c ~ b * sin(d + a * time)
   data: df
       d        b        a 
-4.47043  0.67120  0.09875 
 residual sum-of-squares: 1.127

Number of iterations to convergence: 15 
Achieved convergence tolerance: 4.921e-06
```

Note that the parameter estimates from this procedure are not unique, and depend on the value supplied to start the algorithm. Predictions from convergent fits are unique, however.

## Exercises 3

1. Plot the predictions from the nonlinear regression on a fine grid of points from 0 to 120 minutes. Use type = 'l'.
2. Superimpose the data on these predictions.
3. Plot the residuals from the model over time.
4. If you did not use *ggplot2* for these visualizations, please do so. If you did use *ggplot2*, use the standard graphics.
5. Enhance the *ggplot2*-based version with a nonparametric model including pointwise standard errors. Interpret.

```
> yal040c = exprs(alp)["YAL040C",]
> df = data.frame(yal040c, time=alp$time)
> plot(yal040c~time, data=df)
```

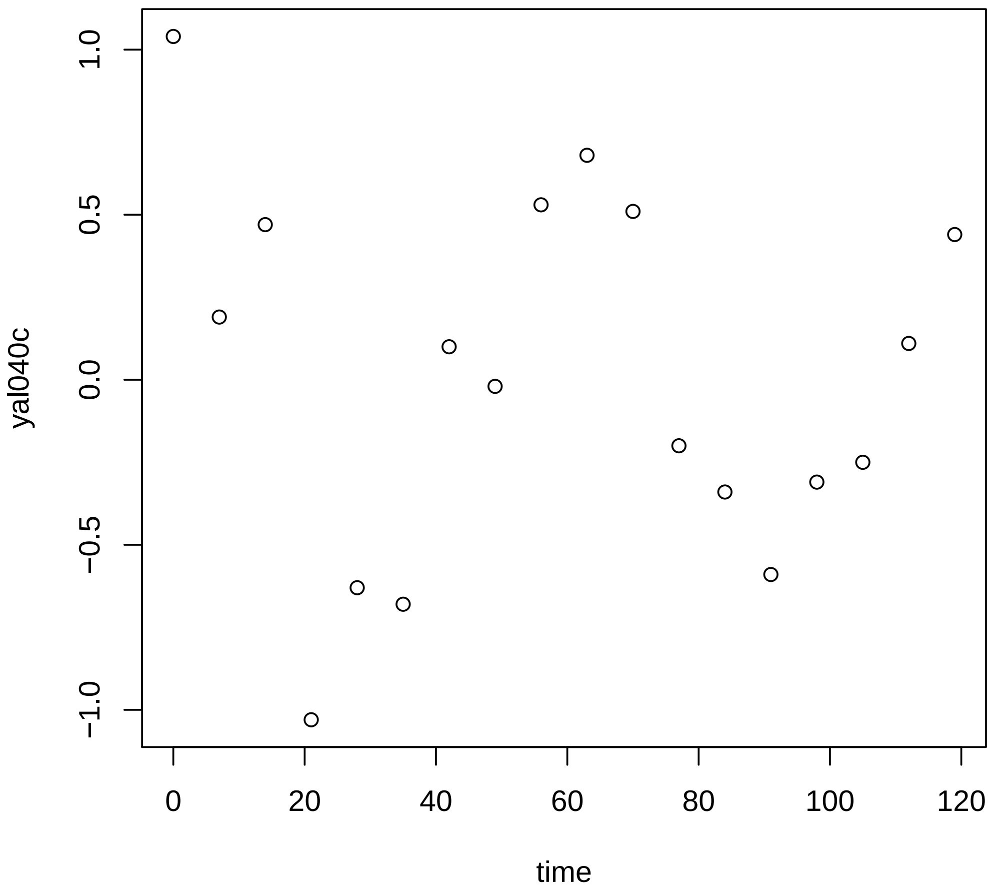


Figure 1: Expression of YAL040C over time.

### 2.3 A linear least squares procedure

Suppose that the length of the cell cycle is known to be 64 minutes. We then employ a transformation of time to fraction of cycle observed and fit a trigonometric regression as follows.

```
> df$ptime = 2*pi*(df$time %% 64)/64
> m2 = lm(yal040c ~ sin(ptime) + cos(ptime) -1, data=df)
> summary(m2)
```

```
Call:
lm(formula = yal040c ~ sin(ptime) + cos(ptime) - 1, data = df)

Residuals:
     Min       1Q   Median       3Q      Max 
-0.56459 -0.11207  0.01913  0.07457  0.52131 
```

```
Coefficients:
           Estimate Std. Error t value Pr(>|t|)
sin(ptime) -0.18125    0.08808  -2.058   0.0563 .
cos(ptime)  0.64820    0.08904   7.280 1.84e-06 ***
---
Signif. codes:  0 âĂŸ***âĂŹ 0.001 âĂŸ**âĂŹ 0.01 âĂŸ*âĂŹ 0.05 âĂŸ.âĂŹ 0.1 âĂŸ âĂŹ 1

Residual standard error: 0.2657 on 16 degrees of freedom
Multiple R-squared:  0.7804,	Adjusted R-squared:  0.7529
F-statistic: 28.42 on 2 and 16 DF,  p-value: 5.418e-06
```

```
> sum(resid(m1)^2)
```

```
[1] 1.127309
```

```
> sum(resid(m2)^2)
```

```
[1] 1.129129
```

Formally we have fit the model $y(t) = a \sin 2\pi\omega t + b \cos 2\pi\omega t + e(t)$ where $e(t)$ is an error term with mean zero and constant variance and $\omega t$ is the fraction of the period at which observation $y(t)$ is obtained. The amplitude of the sinusoidal model for the mean is then $\sqrt{a^2 + b^2}$ and the phase is $\tan^{-1}(a/b)$.

The following code will fit such a model, given a gene name, an *ExpressionSet* instance, and a stipulated period.

```
> gettrm = function(genename, es, period=64) {
+  stopifnot("time" %in% names(pData(es)))
+  ex = exprs(es)[genename,]
+  et = es$time
+  ptime = 2*pi*(df$time %% period)/period
+  ndf = data.frame(time=et, ptime=ptime)
+  ndf[[tolower(substitute(genename))]] = exprs(es)[genename,]
+  fm = as.formula(paste(tolower(substitute(genename)), "~ sin(ptime) + cos(ptime) - 1",
+     sep=" "))
+  lm(fm, data=ndf)
+ }
```

Check:

```
> gettrm("YAL040C", alp)
```

```
Call:
lm(formula = fm, data = ndf)

Coefficients:
sin(ptime)  cos(ptime)
   -0.1813      0.6482
```

## 2.4 Trigonometric regression over the transcriptome

Collect the mean squared error of prediction (msep), estimated amplitude (amp), and estimated phase in the matrix outs. Note the use of try to allow simple iteration over the whole yeast transcriptome – some of the fits will fail owing to missing data, and the corresponding elements of outs are NA.

```
> outs = matrix(NA, nc=3, nr=nrow(alp))
> yg = featureNames(alp)
> rownames(outs) = yg
> colnames(outs) = c("msep", "amp", "phase")
> suppressMessages({
+ for (i in 1:nrow(alp)) {
+   curg = force(yg[i])
+   m = try( gettrm( curg, alp ) )
+   if (inherits(m, "try-error")) next
+   cm = coef(m)
+   outs[i,"msep"] = mean(resid(m)^2)
+   outs[i,"amp"] = sqrt(sum(cm^2))
+   outs[i,"phase"] = atan(-cm[1]/cm[2])
+   }
+ })
```

### Exercises 4

1. Interpret pairs(outs). Choose boundaries on msep and amp that identify genes with robust cyclic transcription pattern, and subset outs to this set of genes. Is the overlap with Spellman's orf800 as you would expect?
2. Find a pair of genes with estimated phase values near -1.0 and 1.0 respectively. Plot the expression trajectories, superimposed, in the (expression,time) plane. Justify and suitably display the estimated values of amp for these genes.
3. Following the *ggplot2* code patterns of the lecture, plot these trajectories in polar coordinates.
4. Residual analysis: Obtain the residuals for trigonmetric fits to YOL012C and YPL256C and display in polar coordinates. Interpret.

## 3 RNA-seq application: hnRNP C and Alu exon suppression

Zarnack et al. (2013 Cell) present iCLIP-seq and RNA-seq data related to the competition between hnRNP C (heterogeneous nuclear ribonucleoproteins C1/C2) and the splicing factor U2AF65. One experiment reported here compares RNA-seq read coverage for cells for which an siRNA knockdown of hnRNP C was executed, to control HeLa cells. Alu exonization can be observed clearly over an antisense Alu element in gene CD55. Hervé Pagès has assembled BAM files from these experiments in the *RNAseqData.HNRNPC.bam.chr14* package, and we will use this and related metadata to investigate this.

### 3.1 Alu annotations

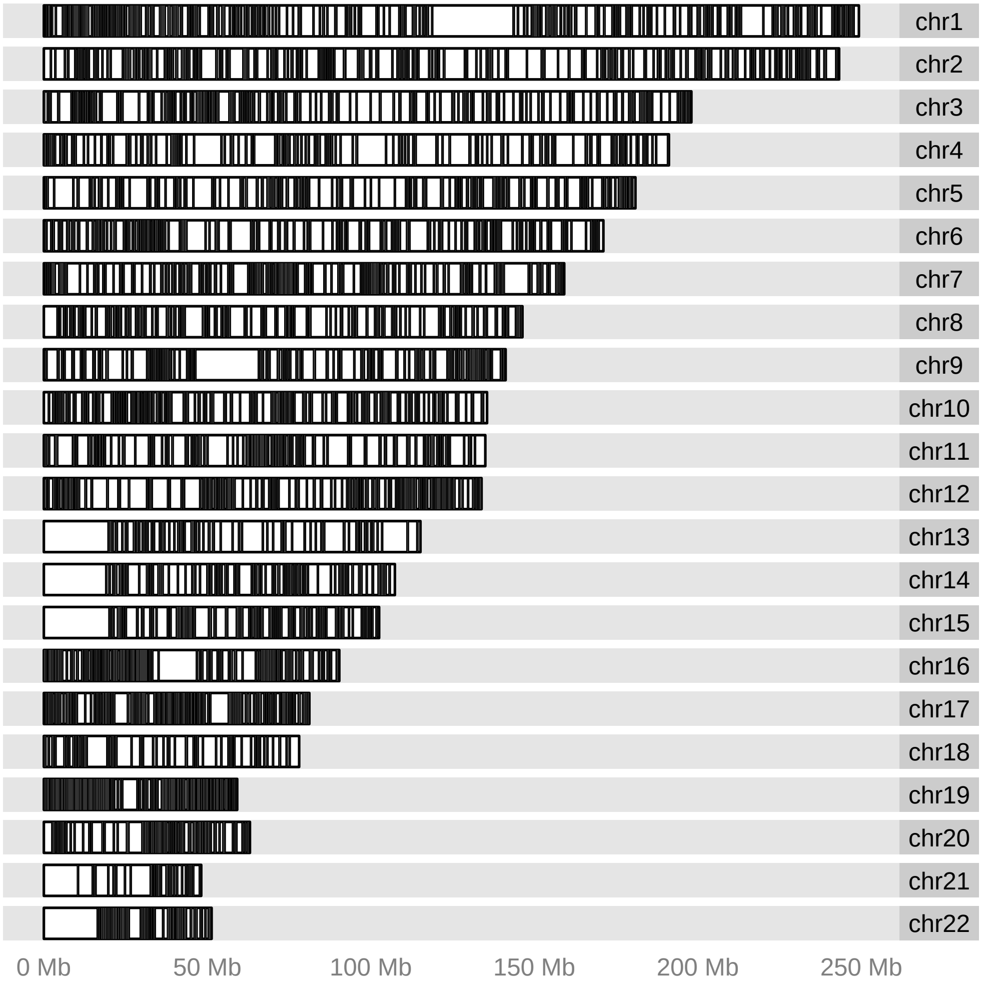
From AnnotationHub, we have a general resource on repetitive elements:

```
> library(viz14)
> data(nestrep)
> metadata(nestrep)[[2]]$RDataName

[1] "goldenpath.hg19.database.nestedRepeats_0.0.1.RData"
```

We limit the annotation to Alu elements on somatic chromosomes. *ggbio* is used give an overview of the genome-wide frequencies of Alu elements, but because the full collection takes time to process, we take a sample.

```
> soma = paste0("chr", 1:22)
> alu = nestrep[ grep("^Alu", nestrep$name) ]
> library(GenomicRanges)
> alu = alu[ which(as.character(seqnames(alu)) %in% soma) ]
> alu14 = alu[ which(seqnames(alu)=="chr14") ]
> seqlevels(alu) = soma
> alus = alu[ sample(1:length(alu), size=5000) ]
> library(ggbio)
> autoplot(alus, layout="karyogram")
```



We will obtain identifiers and genomic extents for genes on chr14 that overlap Alu elements.

```
> library(TxDb.Hsapiens.UCSC.hg19.knownGene)
> txdb <- TxDb.Hsapiens.UCSC.hg19.knownGene
> ucg = genes(txdb)
> ucg14 = ucg[which(seqnames(ucg)=="chr14")]
> egovalu = as.character(ucg14[which(ucg14 %over% alu14)]$gene_id)
> library(org.Hs.eg.db)
> symovalu = unique(unlist(mget(egovalu, org.Hs.egSYMBOL, ifnotfound=NA)))
> allg14 = get("14", revmap(org.Hs.egCHR))
> alls14 = unlist(mget(allg14, org.Hs.egSYMBOL))
> symnoalu = setdiff(alls14, symovalu)
```

## 3.2 Coincidence of Alu elements and genes implicated in GWAS

The *gwascat* package provides data from the NHGRI GWAS catalog.

```
> library(gwascat)
> gwrngs  # limit to chr14 :
```

```
gwasloc instance with 14719 records and 35 attributes per record.
Extracted:  2013-12-03
Excerpt:
GRanges with 5 ranges and 3 metadata columns:
      seqnames                 ranges strand |                      Disease.Trait
         <Rle>              <IRanges>  <Rle> |                        <character>
  [1]    chr10 [ 96405502,  96405502]      * |          Warfarin maintenance dose
  [2]    chr11 [  2024544,   2024544]      * | DNA methylation (parent-of-origin)
  [3]     chr4 [137526584, 137526584]      * | DNA methylation (parent-of-origin)
  [4]    chr22 [ 19691502,  19691502]      * | DNA methylation (parent-of-origin)
  [5]    chr11 [ 92784203,  92784203]      * | DNA methylation (parent-of-origin)
             SNPs   p.Value
      <character> <numeric>
  [1]  rs12777823     5e-12
  [2]   rs4930103     5e-16
  [3]  rs10012307     2e-08
  [4]   rs4819833     6e-08
  [5]   rs7931462     2e-09
  ---
  seqlengths:
        chr1      chr2      chr3      chr4 ...     chr20     chr21     chr22      chrX
   249250621 243199373 198022430 191154276 ...  63025520  48129895  51304566 155270560
```

```
> gwr14 = gwrngs[which(seqnames(gwrngs)=="chr14")]
```

We will use a very crude approach to estimating the proportion of genes harboring Alu elements that have been implicated for association with traits studied in GWAS. The Mapped_gene metadata variable is only informally coded.

```
> mapped14 = unique(gwr14$Mapped_gene)
> table(mapped14)[1:10]
```

```
mapped14
                  -               ACTN1               AKAP6       AKAP6 - NPAS3
                  1                   1                   1                   1
      ALDH6A1;LIN52               AP1G2                ATL1   ATP5C1P1 - CDKN3
                  1                   1                   1                   1
ATP5G2P2 - RPL32P29               ATXN3
                  1                   1
```

We estimate the proportion based on exact matches between gene symbols in the Alu and GWAS annotations. Application of grep would be more effective.

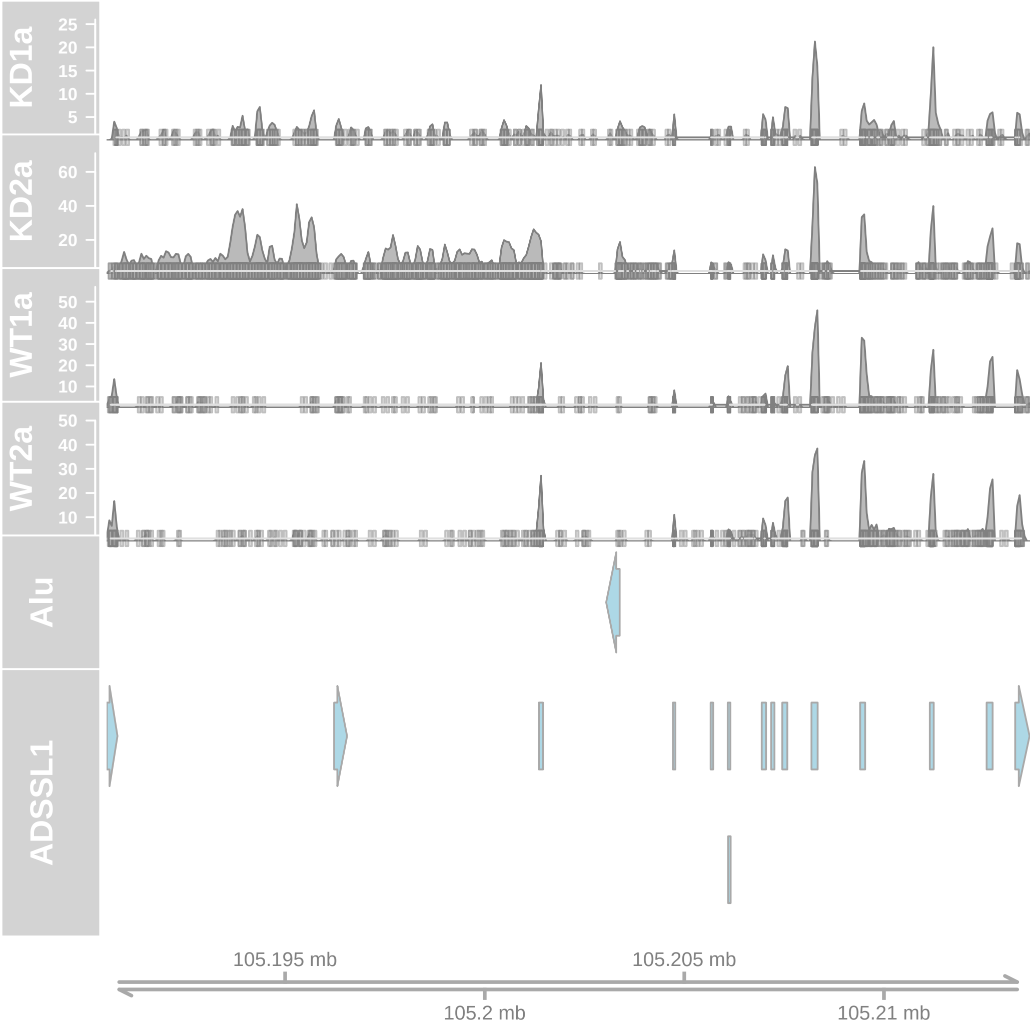
```
> mean(symovalu %in% mapped14)  # very crude
```

```
[1] 0.2397661
```

## 3.3 Exonization of Alu element in HNRNPC knockdowns: the case of ADSSL1

We work directly from four BAM files to compare first replicates of two samples.

```
> library(GenomicFiles)
> library(Gviz)
> GENENAME = "ADSSL1"    # gene to analyze
> gm = genemodel(GENENAME)
> rgm = range(gm)
> at = AnnotationTrack(gm, genome="hg19", name=GENENAME)
> #  now set up the references to RNA-seq data
> fn = dir(system.file(  # use Herve's package
+   "extdata", package="RNAseqData.HNRNPC.bam.chr14"), full=TRUE,
+   patt="bam$")
> bfv = BamFileViews(fn)  # lightweight reference
> # now ingest the BAM data into AlignmentsTrack instances
> STACKTYPE = "hide"      # for Gviz
> kd1a = AlignmentsTrack( path(fileList(bfv)[[1]]), isPaired=TRUE,
+     name="KD1a", chromosome="chr14", stacking=STACKTYPE )
> wt1a = AlignmentsTrack( path(fileList(bfv)[[5]]), isPaired=TRUE,
+     name="WT1a", chromosome="chr14", stacking=STACKTYPE )
> kd2a = AlignmentsTrack( path(fileList(bfv)[[3]]), isPaired=TRUE,
+     name="KD2a", chromosome="chr14", stacking=STACKTYPE )
> wt2a = AlignmentsTrack( path(fileList(bfv)[[7]]), isPaired=TRUE,
+     name="WT2a", chromosome="chr14", stacking=STACKTYPE )
> gt = GenomeAxisTrack(genome="hg19")
> #data(nestrep)
> #rep14 = nestrep[which(seqnames(nestrep)=="chr14")]
> #alu14 = rep14[ grep("^Alu", rep14$name) ]
> alut = AnnotationTrack(alu14, genome="hg19", name="Alu")
> ps.options(font="sans")
> plotTracks(list(kd1a, kd2a, wt1a, wt2a,
+     alut, at, gt), from=start(rgm), to=end(rgm))
```



### 3.4 Exercises 5

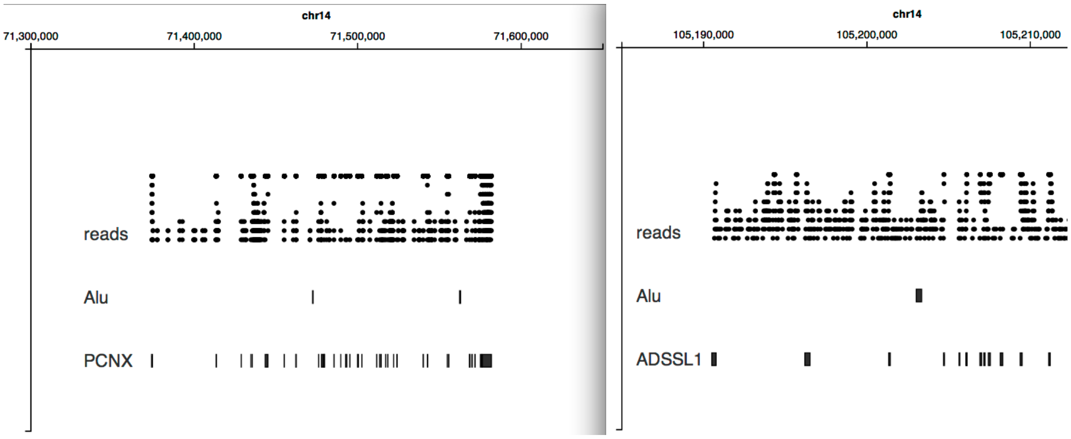
1. Show that, on chr14, genes harboring Alu elements are at least 5 times more likely to be identified as trait-associated in GWAS as those that do not harbor Alu elements.
2. The use of integer indices in the path-selecting code above is fragile. Use the information in the emtab1147sdrf data frame to improve reliability of code for these selections.
3. Transform the code for ADSSL1 into a function that will accept any gene symbol, and use it to investigate CGRRF1, PCNX (both mentioned in supplement to Zarnack paper) and other genes, for example, those harboring Alu and mapped to trait variation in GWAS.

## 4 ggvis: linked, interactive, browser-based visualization

### 4.1 Background and illustration of a ggvis application

In this section you will have a look at some components that push the *grammar of statistical graphics* concepts towards a developing *grammar of visualizations*, see `https://github.com/trifacta/vega/wiki/Documentation`. Of particular novelty are the interaction scenarios, some of which are now supported through *shiny* reactive programming methods; see https://github.com/trifacta/vega/wiki/Interaction-Scenarios for a general discussion of interaction.

This vignette does not address interaction but is concerned mainly with some basic examples of confronting genomic data modeling for web-based visualization. We have a function that produces, in your browser, data views like



based on the RNA-seq data explored in the preceding subsection.

In fact, with the following code, you can generate the ADSSL1 plot.

```
> library(ggvis)
> library(GenomicFiles)
> library(viz14)
> data(alu14)
> alu14 = alu14[,1:3]
> fn = dir(system.file(
+   "extdata", package="RNAseqData.HNRNPC.bam.chr14"), full=TRUE,
+   patt="bam$")
> bfv = BamFileViews(fn)
> browseWreads()
```

## 4.2 Dissecting the intermediate components

The code for browseWreads is relatively concise, but somewhat clumsy as well.

```
> browseWreads
```

```
function (sym = "ADSSL1", doAlu = TRUE, alugr = alu14, aluy = 2,
    bamf = path(fileList(bfv)[[1]]), ready = 3, cachedir = "~/.ggmcache",
    y = 1, y2 = 1.2, ylim = c(0, 6), padpct = 20)
{
    base = renderGene(sym, cachedir = cachedir, y = y, y2 = y2,
        ylim = ylim, padpct = padpct)
    rng = rangeOfGene(sym, ggmDefC())
    wid = end(rng) - start(rng)
    pad = padpct * wid/100
    left = start(rng) - pad
    if (doAlu)
        return(base %>% rectsAtY(sym = sym, gr = alugr, y = aluy) %>%
            addText(sym = "Alu", x = left, y = aluy) %>% readsAtY(sym = sym,
            bamf = bamf, y = ready) %>% addText(sym = "reads",
            x = left, y = ready))
    base %>% readsAtY(sym = sym, bamf = bamf, y = ready) %>%
        addText(sym = "reads", x = left, y = ready)
}
<environment: namespace:viz14>
```

We'll examine the statements in small groups.

```
>     base = renderGene(sym, cachedir = cachedir, y = y, y2 = y2,
+         ylim = ylim, padpct = padpct)
```

The renderGene function creates the fundamental ggvis instance on the basis of a gene symbol. The extraction of genomic coordinates for a gene model is costly, and a persistent cache is created to save models. Basic geometric properties of the display are determined by the gene model, which will be represented as a horizontal collection of rectangles. ylim is used to define the extent of the horizontal scale. y2 is the location of the upper edge of exon rectangles. padpct is the percentage of gene model width that is taken to pad the presentation on both right and left sides.

The renderGene function can be called on its own. As it returns an instance of ggvis, the call, when successful results in a rendering on the browser identified by getOption("browser").

Here the cache is used to obtain information on extents of the gene model:

```
>     rng = rangeOfGene(sym, ggmDefC())
>     wid = end(rng) - start(rng)
>     pad = padpct * wid/100
>     left = start(rng) - pad
```

We'll ignore the branch that is present to deal with genes that don't subtend Alu elements.

```
>         return(base %>% rectsAtY(sym = sym, gr = alugr, y = aluy) %>%
+             addText(sym = "Alu", x = left, y = aluy) %>% readsAtY(sym = sym,
+             bamf = bamf, y = ready) %>% addText(sym = "reads",
+             x = left, y = ready))
```

The %>% operator allows us to refrain from heavy nesting. The base ggvis instance is transformed by calls to rectsAtY, readsAtY, and addText. Each of these components is defined in *viz14*.

- rectsAtY relates the information in a GRanges instance gr to the gene model for sym. Rectangles will be drawn at vertical position given by y. Full control is not offered to the browseWreads to simplify the interface; rectsAtY can tailor the sizes of rectangles through additional arguments.
- readsAtY works from a BAM file and plots points at basewise coverage values, truncating vertically at 7 in this implementation. More control is available but not propagated to the demonstration function browseWreads.
- addText deals with direct placement of textual tokens at selected positions.

## 4.3 At a slightly lower level

Here is the renderGene function. Axis handling and padding should be more autonomous and as ggvis matures there will undoubtedly be simplifications.

```
> renderGene
```

```
function (sym = "ADSSL1", cachedir = "~/.ggmcache", y = 1, y2 = 1.2,
    ylim = c(0, 6), padpct = 20)
{
    ggmcache = setupCache(cachedir)
    gvm = ggvForGeneModel(sym, ggmcache, padpct = padpct)
    rng = rangeOfGene(sym, ggmcache)
    wid = end(rng) - start(rng)
    pad = padpct * wid/100
    gvm %>% exonRects(y = y, y2 = y2) %>% addText(sym = sym,
        x = start(rng) - pad, y = y) %>% addText(sym = " ", x = end(rng) +
        pad, y = y) %>% set_dscale("y", type = "numeric", domain = ylim) %>%
        add_guide_axis(type = "y", scale = "y", grid = FALSE,
            orient = "left", ticks = 0, title = "") %>% add_guide_axis(type = "x",
        scale = "x", orient = "top", title = "chr14", grid = FALSE,
        ticks = 4)
}
<environment: namespace:viz14>
```

### 4.4 Opportunities for improved functionality

1. Flexible specification of (arbitrarily many) additional tracks.
2. Brushing to generate local statistics of interest.
3. Appropriate handling of strand information.
4. Protocol for managing metadata on BAM filesets and propagating that information usefully to the display.
5. *shiny* interface allowing selection of tracks including *AnnotationHub* elements.
6. Interactive filtering of reads to be visualized, based on quality criteria or other quantities derivable.
7. Zoom with translation to read content when scale is sufficiently reduced.

## 5 Session information

```
> sessionInfo()

R version 3.1.0 (2014-04-10)
Platform: x86_64-unknown-linux-gnu (64-bit)

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C                 LC_TIME=en_US.UTF-8
 [4] LC_COLLATE=C               LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=en_US.UTF-8       LC_NAME=C                  LC_ADDRESS=C
[10] LC_TELEPHONE=C             LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C

attached base packages:
 [1] grid      parallel  tools     stats     graphics  grDevices utils     datasets
 [9] methods   base

other attached packages:
 [1] Gviz_1.8.3                              GenomicFiles_1.0.1
 [3] rtracklayer_1.24.2                      Rsamtools_1.16.1
 [5] BiocParallel_0.6.1                      gwascat_1.8.0
 [7] org.Hs.eg.db_2.14.0                     RSQLite_0.11.4
 [9] DBI_0.2-7                               TxDb.Hsapiens.UCSC.hg19.knownGene_2.14.0
[11] GenomicFeatures_1.16.2                  AnnotationDbi_1.26.0
[13] ggbio_1.12.4                            ggplot2_1.0.0
[15] GenomicRanges_1.16.3                    GenomeInfoDb_1.0.2
[17] yeastCC_1.4.0                           viz14_0.0.10
[19] harbChIP_1.2.0                          Biostrings_2.32.0
[21] XVector_0.4.0                           Biobase_2.24.0
[23] IRanges_1.22.9                          BiocGenerics_0.10.0

loaded via a namespace (and not attached):
 [1] BBmisc_1.6               BSgenome_1.32.0          BatchJobs_1.2
 [4] BiocStyle_1.2.0          Formula_1.1-1            GenomicAlignments_1.0.1
 [7] Hmisc_3.14-4             MASS_7.3-33              Matrix_1.1-4
[10] R.methodsS3_1.6.1        RColorBrewer_1.0-5       RCurl_1.95-4.1
[13] RJSONIO_1.2-0.2          Rcpp_0.11.2              VariantAnnotation_1.10.4
[16] XML_3.98-1.1             assertthat_0.1           biomaRt_2.20.0
[19] biovizBase_1.12.1        bitops_1.0-6             brew_1.0-6
[22] caTools_1.17             cluster_1.15.2           codetools_0.2-8
[25] colorspace_1.2-4         dichromat_2.0-0          digest_0.6.4
[28] dplyr_0.2                fail_1.2                 foreach_1.4.2
[31] ggvis_0.3.0.99           gridExtra_0.9.1          gtable_0.1.2
[34] htmltools_0.2.4          httpuv_1.3.0             iterators_1.0.7
[37] labeling_0.2             lattice_0.20-29          latticeExtra_0.6-26
[40] magrittr_1.0.1           matrixStats_0.10.0       munsell_0.4.2
[43] parody_1.22.0            plyr_1.8.1               proto_0.3-10
[46] reshape2_1.4             scales_0.2.4             sendmailR_1.1-2
[49] shiny_0.10.0.9001        snpStats_1.14.0          splines_3.1.0
[52] stats4_3.1.0             stringr_0.6.2            survival_2.37-7
[55] xtable_1.7-3             zlibbioc_1.10.0
```