GenomicVis

Jonathan J. Ellis and Lutz Krause

July 2, 2014

Contents

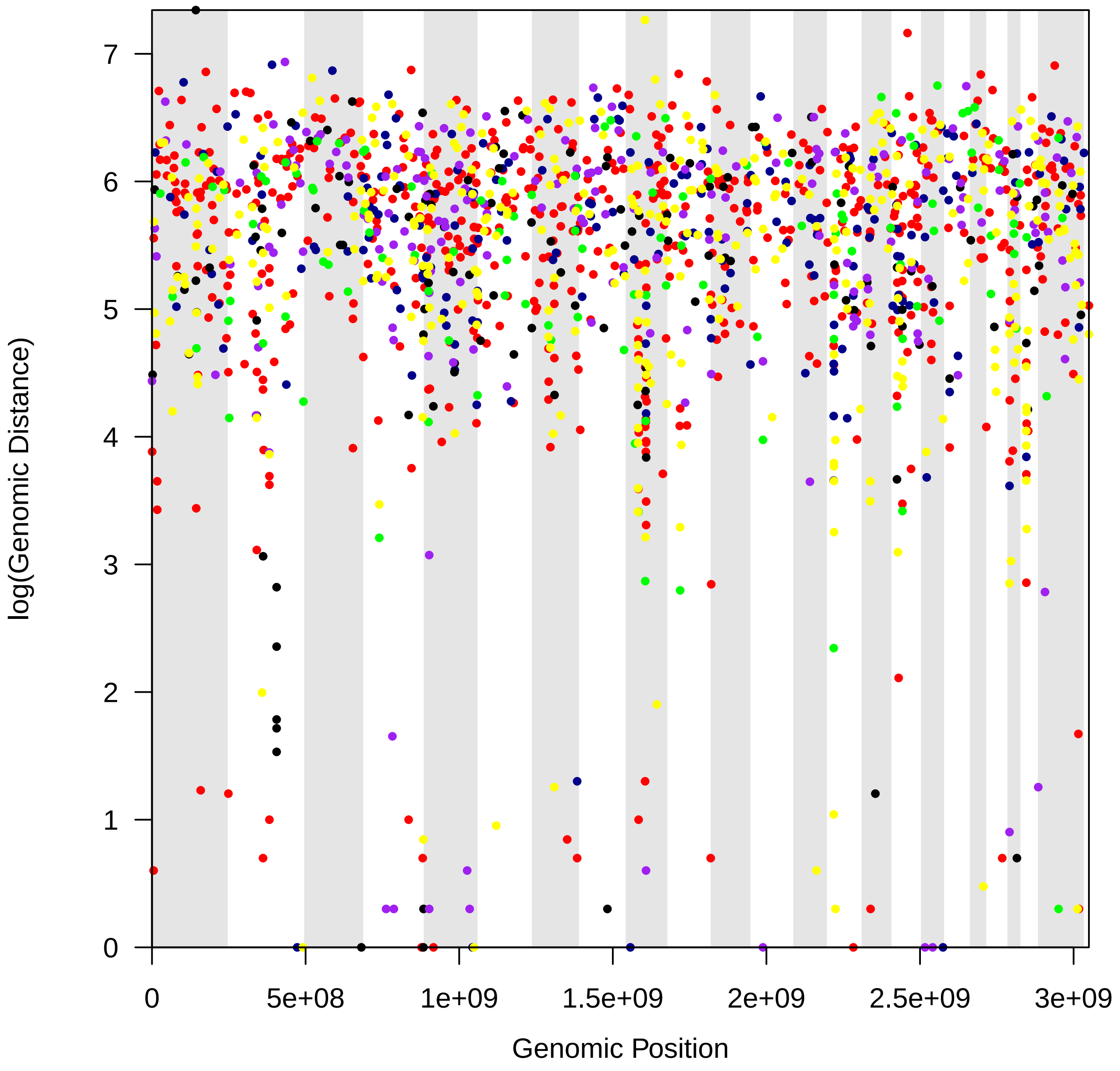
The *GenomicVis* package contains a collection of function for the visualisation of genomic data typically derived from high-throughput whole genome sequencing and SNP-Chip experiments. Most of the functions provided by this package are agnostic about where the data actually comes from, that is, they required input in simple *data.frame* instances that are not tied to a particular file format; however, we also provide many examples of creating these simply *data.frame*s from standard file formats such as VCFs and from common software tools such as *GAP* and *Breakdancer*.

```
library(GenomicVis)
```

1 Kataegis

The package detects and visualises kataegis from single nucleotide variations (SNVs). The package currently implements two different algorithms to detect kataegis and visualises SNVs as rainfall plots. To create a rainfall plot, you do not necessarily need to identify regions of kataegis first. To create a rainfall plot, you need to have your SNVs in a *GRanges* instance containing only SNVs (i.e., no INDELs) with single ALT alleles. The *GRanges* instance must also contain a valid *Seqinfo* instance that provides the lengths of the chromosomes. The easiest way to obtain the appropriate instance from a VCF file is to use the `read.vcf` function in the *GenomicVis* package. This function returns a

```r
# We need to import VariantAnnotation to get access to rowData
suppressMessages(library(VariantAnnotation))
vcf.file <- system.file('extdata', 'example.vcf', package = 'GenomicVis')
vcf <- read.vcf(vcf.file, 'GRCh37')
x <- rowData(vcf)
plotKataegis(x)
```



To create a kataegis plot of each VCF in the current directory, you could use the following code:

```r
vcf.files <- list.vcffiles()
for (vcf.file in vcf.files) {
  vcf <- read.vcf(vcf.file, 'GRCh37')
  x <- rowData(vcf)
  kat <- kataegis(x)
  name <- tools::file_path_sans_ext(vcf.file)
  png(file = paste0(name, '.png'))
  plotKataegis(x, main = name)
  dev.off()
}
```

1.1 Kataegis Detection

To detect regions of kataegis, use the `kataegis` function. This take the same *GRanges* instance as `plotKataegis`, and returns another *GRanges* instance containing the kataegis regions. There are two different algorithms for the detection of kataegis: one that uses piecewise constant curve fitting and one that uses a brute force approach.

1.1.1 Piecewise Constant Curve Fitting Detection

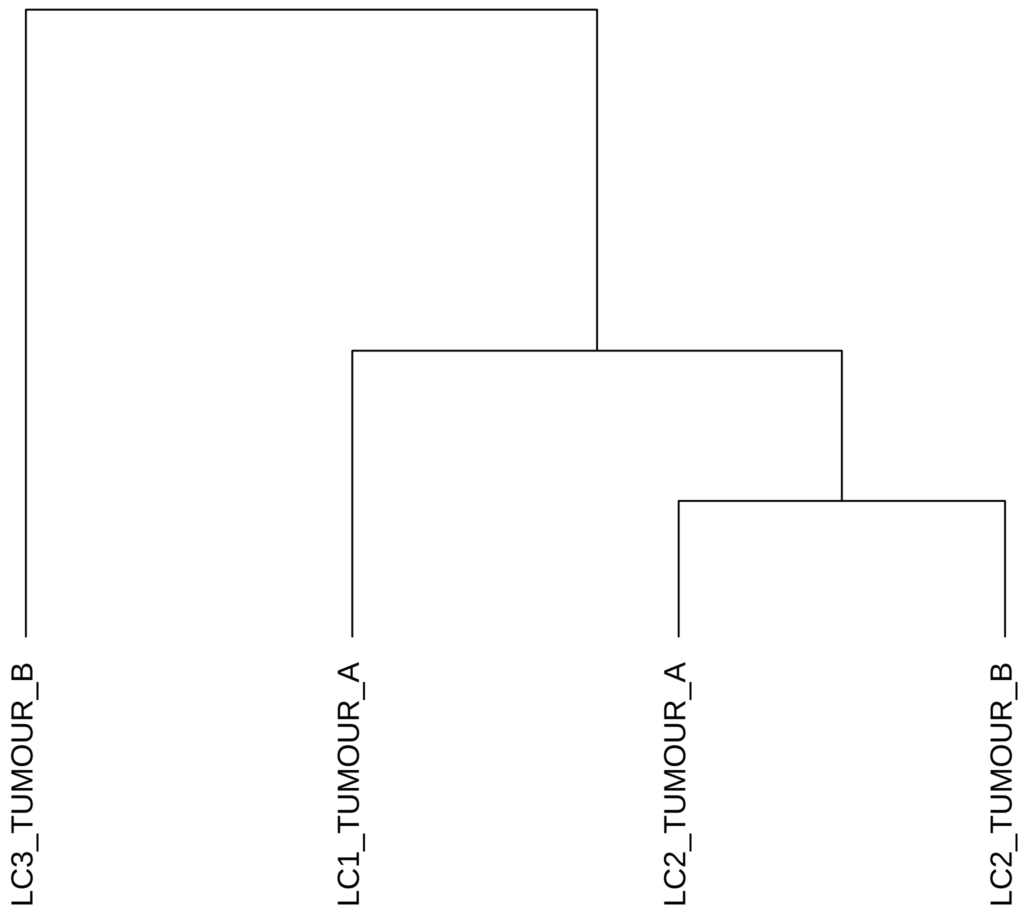
```
kat <- kataegis(x, pcf = TRUE, ncpus = 4)
```

1.1.2 Brute Force Detection

```
kat <- kataegis(x, ncpus = 4)
```

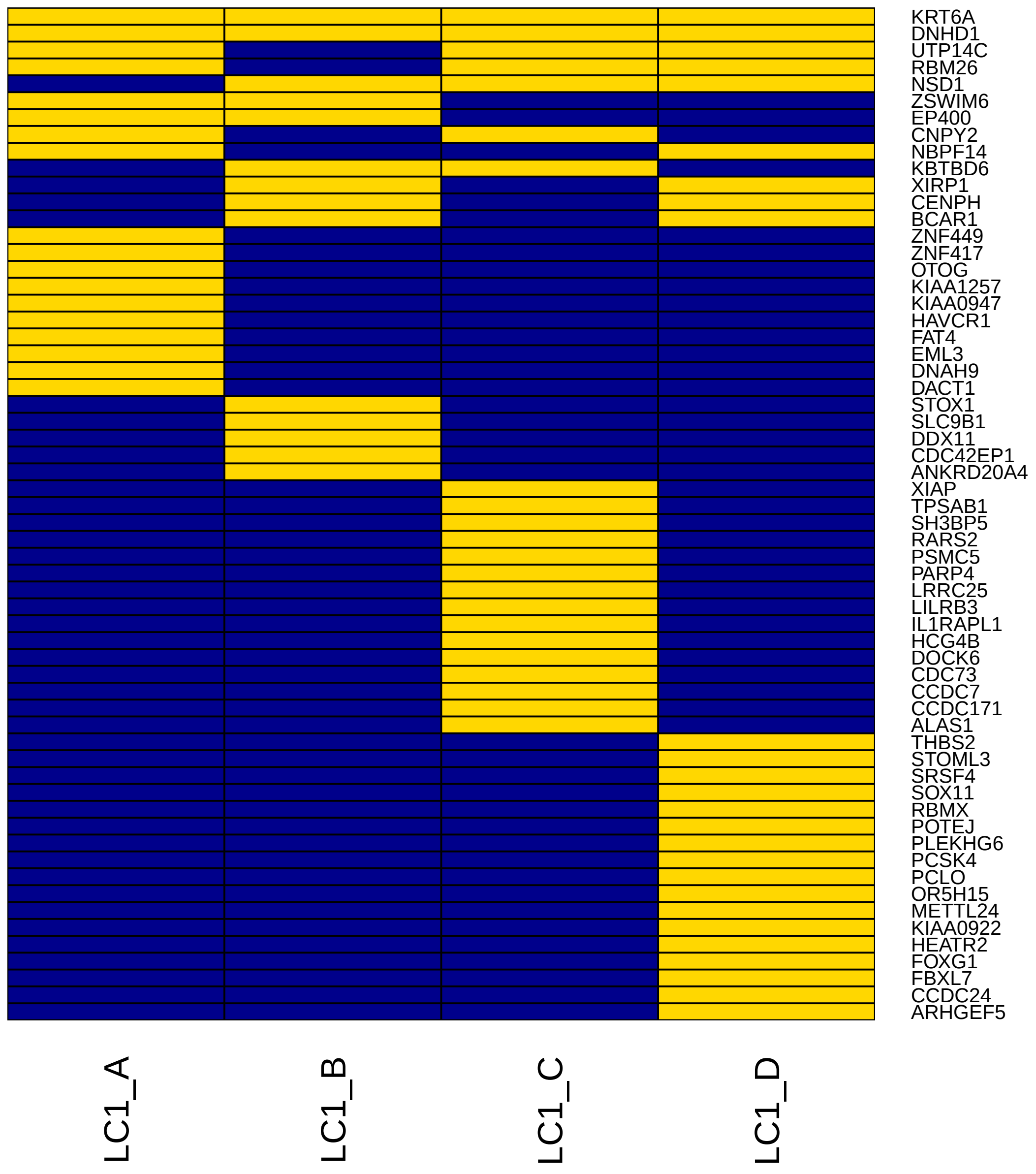
2 SNV Clustering

```
file.names <- sprintf('LC%s_TUMOUR_%s.vcf', rep(1:3, each = 2),
  rep(c('A', 'B'), each = 3))
vcf.files <- system.file('extdata', file.names, package = 'GenomicVis')
sample.names <- tools::file_path_sans_ext(basename(vcf.files))
snv.clustering(vcf.files, sample.names, genome = 'hg19')
```



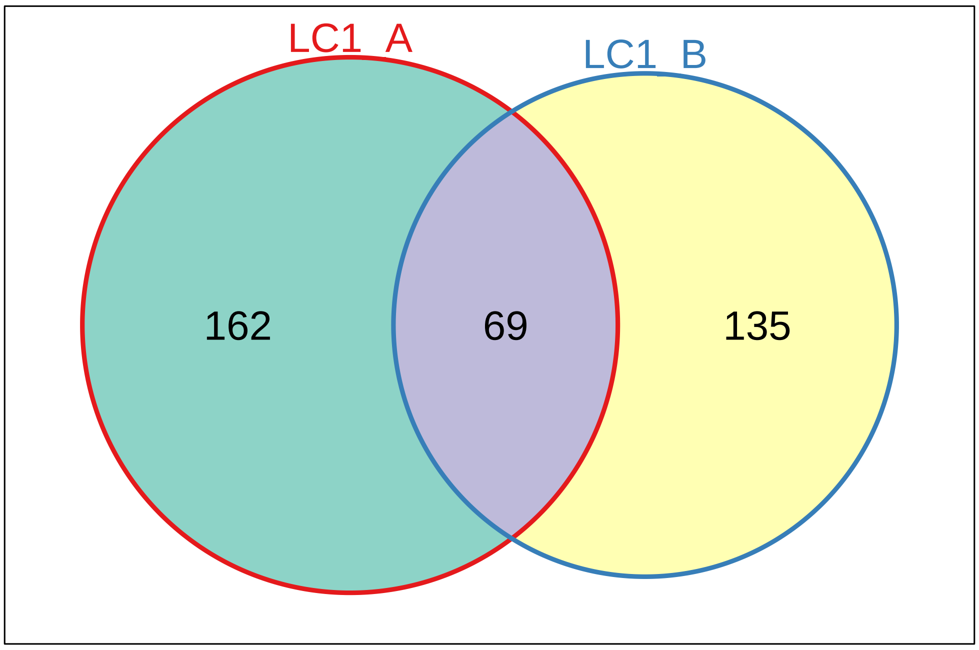
3 SNV Heatmaps

```
file.names <- c('LC1_A.snpeff.vcf', 'LC1_B.snpeff.vcf',
  'LC1_C.snpeff.vcf', 'LC1_D.snpeff.vcf')
vcf.files <- system.file('extdata', file.names, package = 'GenomicVis')
sample.names <- c('LC1_A', 'LC1_B', 'LC1_C', 'LC1_D')
dat <- read.snpeff.vcfs(vcf.files, 'GRCh37', sample.names)
snv.heatmap(dat, margins = c(5, 9), y.cex.axis = 0.7)
```



4 SNV Venn Diagrams

```
library(Vennerable)
file.names <- c('LC1_TUMOUR_A.vcf', 'LC1_TUMOUR_B.vcf')
vcf.files <- system.file('extdata', file.names, package = 'GenomicVis')
sample.names <- c('LC1_A', 'LC1_B')
v <- vcf.venn(vcf.files, 'GRCh37', sample.names)
plot(v$venn)
```



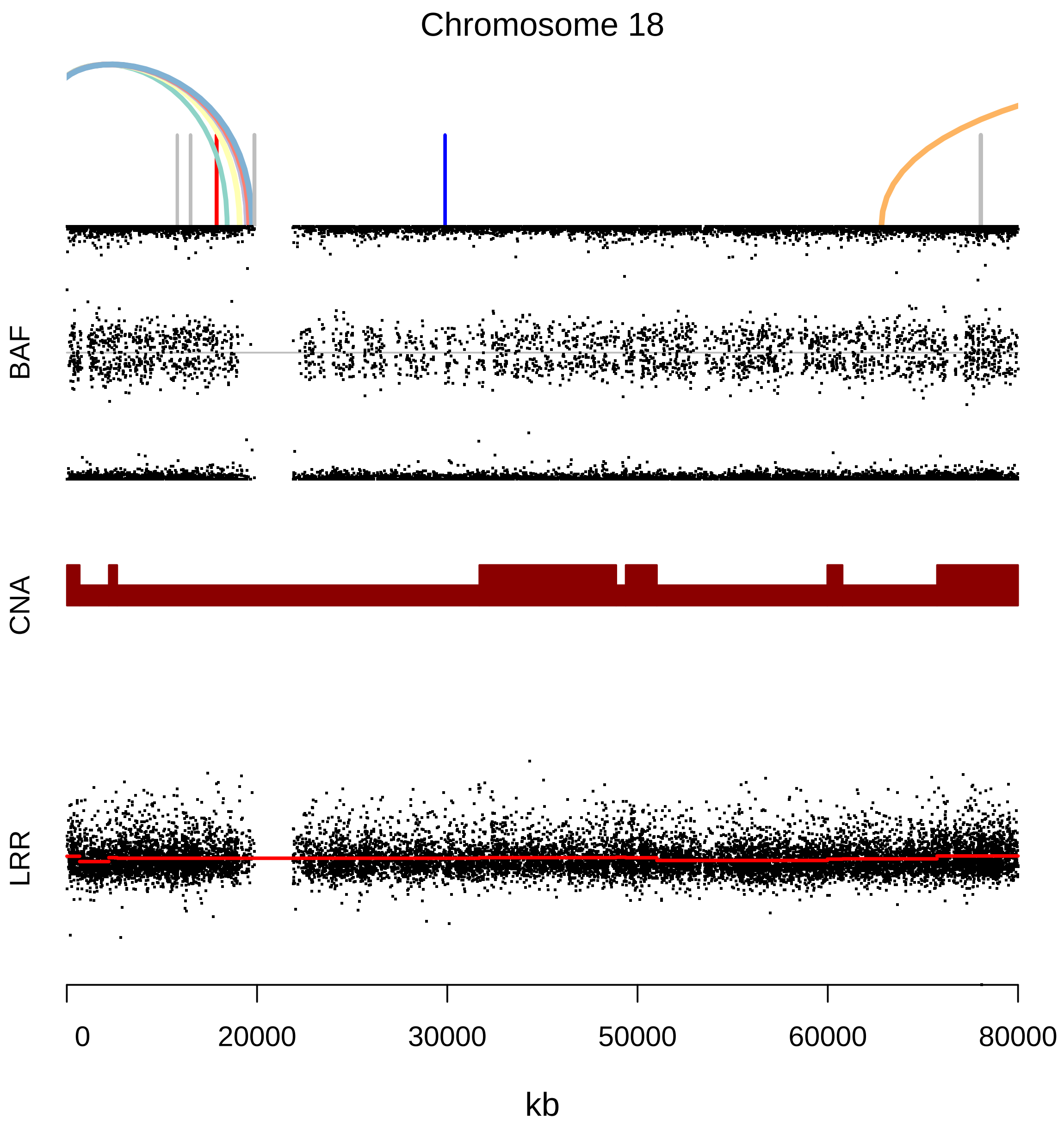
5 CNV Plots

`LC1_TUMOUR_A_FinalReport.txt` is the final report file produced by *GenomeStudio* from the SNP-Chips.

Breakdancer was run on the germline and tumour sample at the same time, i.e., the file contains calls from both samples. The file names that were input to *Breakdancer* contain the strings `BLOOD` and `TUMOUR` to indicate the normal and tumour samples respectively.

```
data(SNPExample)
data(CNVExample)
data(SVExample)
cnv.plot('18', SNPExample, CNVExample, SVExample)

## Found 13 SVs to plot
```



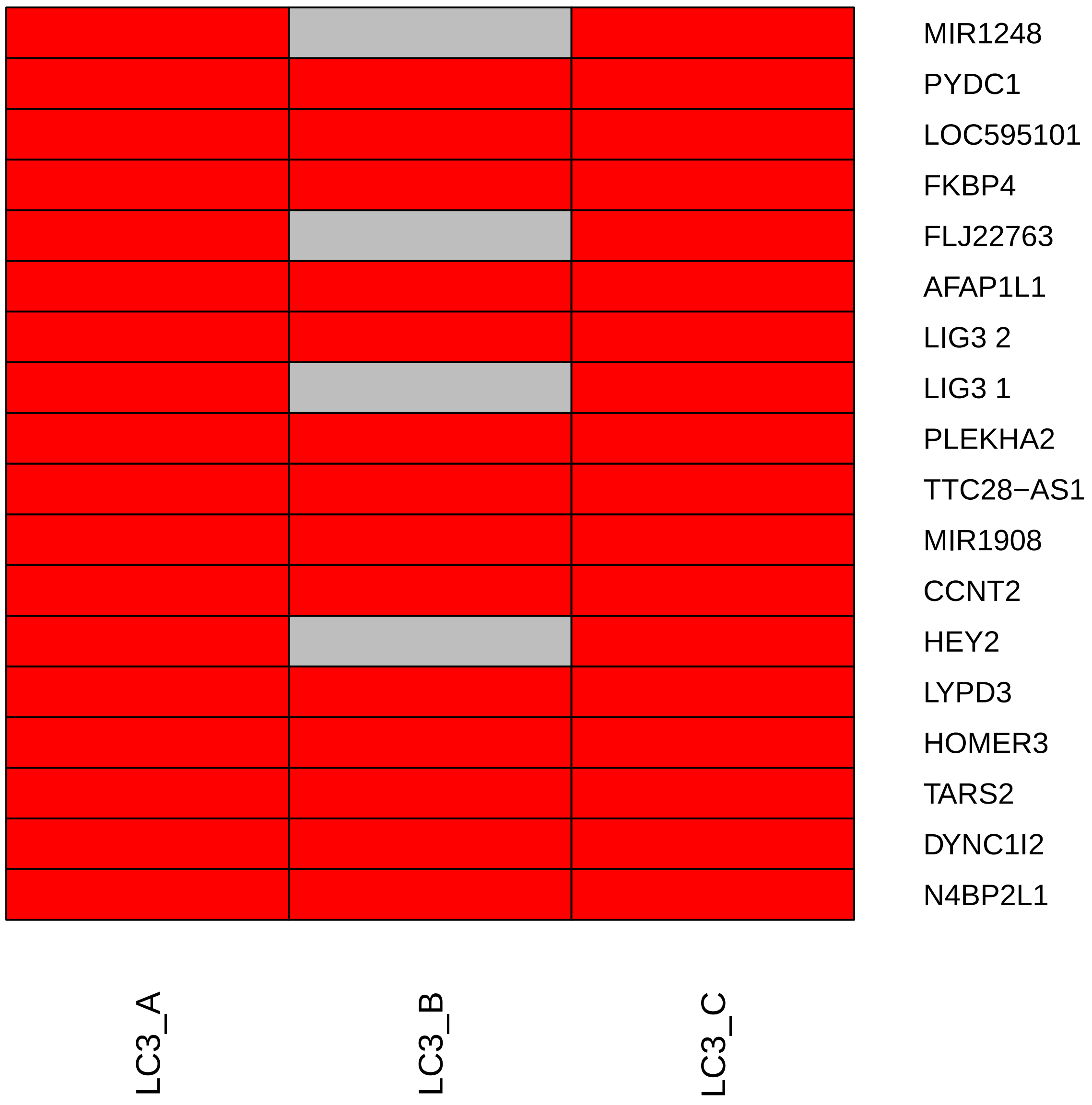
6 CNV Heatmaps

The genes.gr argument can easily be built from available Bioconductor packages. The following code shows how you could build the appropriate *GRanges* instance for human hg19 data.

```r
library(TxDb.Hsapiens.UCSC.hg19.knownGene)
library(org.Hs.eg.db)
genes.gr <- genes(TxDb.Hsapiens.UCSC.hg19.knownGene)
gene_ids <- unlist(genes.gr$gene_id)
symbol.map <- select(org.Hs.eg.db, gene_ids, 'SYMBOL')
genes.gr$symbol <- symbol.map$SYMBOL
```

The data set hg19.Genes.GRanges contains human genes with the appropriate gene symbols.

```
data(hg19Genes)
data(CNVData)
set.seed(100)
g <- sample(hg19Genes$symbol, 20)
cnv.heatmap(CNVData, symbols = g, genes.gr = hg19Genes)
```



To just use a subset of samples.

```
cnv.heatmap(CNVData, samples = c('LC3_A', 'LC3_B'), genes.gr = hg19Genes)
```

7 Example Datasets

The previous sections of this vignette have relied on prepared datasets distributed with the package. While these provide examples of the inputs required by the various plotting functions of this package, they do not provide guidance on how to create them from the various files produced by HTS software.

This section explains how each of the example data sets were created. Typically, they were created from standard file formats such as VCF or the output from particular software such as *GAP* or *Breakdancer*.

`hg19Genes` contains the ranges (start, end) positions of human RefSeq genes and corresponding gene symbols. The BED file used was downloaded from the UCSC Genome Browser web site.

```
library(org.Hs.eg.db)
library(GenomicRanges)
library(plyr)

x <- read.delim(
  'hg19_refGene.bed.gz',
  header = FALSE,
  stringsAsFactors = FALSE
)
x <- x[x$V1 %in% paste0("chr", c(1:22, 'X', 'Y')), ]
keys <- x$V4
dict <- select(org.Hs.eg.db, keys, 'SYMBOL', keytype = 'REFSEQ')
symbol.df <- ddply(dict, .(REFSEQ), summarise,
  symbol = paste(unique(SYMBOL), collapse = ';'))
rownames(symbol.df) <- symbol.df$REFSEQ
x$symbol <- symbol.df[x$V4, ]$symbol
hg19Genes <- GRanges(
  seqnames = Rle(x$V1),
  ranges = IRanges(start = x$V2, end = x$V3),
  strand = Rle(x$V6),
  refseq = x$V4,
  symbol = x$symbol
)
```

`CNVExample` was built from the output of GAP, specifically two files named `CN_BA_Illumina_MySeries.txt` and `Illum660K_annot_cut.csv`; although the exact names of these files will depend on the exact settings you use when running GAP.

```
CNVExample <- read.gap(
  system.file('extdata', 'CN_BA_Illumina_MySeries.chr18.txt',
    package = 'GenomicVis'),
  system.file('extdata', 'Illum660K_annot_cut.chr18.csv',
    package = 'GenomicVis'),
  'LC3_TUMOUR_C_FinalReport'
)
```

`LC1_TUMOUR_A_FinalReport.txt` is the final report file produced by GenomeStudio from the SNP-Chips.

```
filename <- system.file('extdata', 'LC3_TUMOUR_C_FinalReport.chr18.txt',
  package = 'GenomicVis')
SNPExample <- read.illumina(filename)
```

`SVData` contains structural variant data obtained from Breakdancer. Breakdancer was run on the germline and tumour sample at the same time, that is, the file contains calls from both samples. The file names that were input to Breakdancer contain the strings `BLOOD` and `TUMOUR` to indicate the normal and tumour samples respectively.

```
SVExample <- read.breakdancer(
  system.file('extdata', 'LC3_BLOOD_TUMOUR_C.chr18.txt',
    package = 'GenomicVis'),
  normal.regex = 'BLOOD'
)
SVExample <- filter.breakdancer(SVExample)
```

CNVData was also prepared from the output of GAP.

```
gap <- read.gap()
CNVData <- gap2cnv(gap)
CNVData <- CNVData[, c('Chr', 'Begin', 'End', 'LC3_TUMOUR_A_FinalReport',
  'LC3_TUMOUR_B_FinalReport', 'LC3_TUMOUR_C_FinalReport')]
colnames(CNVData) <- sub('_TUMOUR', '', colnames(CNVData))
colnames(CNVData) <- sub('_FinalReport', '', colnames(CNVData))
```

8 Session Information

```
## R version 3.1.0 (2014-04-10)
## Platform: x86_64-unknown-linux-gnu (64-bit)
##
## locale:
##  [1] LC_CTYPE=en_AU.UTF-8       LC_NUMERIC=C               LC_TIME=en_AU.UTF-8
##  [4] LC_COLLATE=C               LC_MONETARY=en_AU.UTF-8    LC_MESSAGES=en_AU.UTF-8
##  [7] LC_PAPER=en_AU.UTF-8       LC_NAME=C                  LC_ADDRESS=C
## [10] LC_TELEPHONE=C             LC_MEASUREMENT=en_AU.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] parallel  stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] VariantAnnotation_1.10.2 Rsamtools_1.16.1         Biostrings_2.32.0
## [4] XVector_0.4.0            GenomicRanges_1.16.3     GenomeInfoDb_1.0.2
## [7] IRanges_1.22.9           BiocGenerics_0.10.0      GenomicVis_1.0
##
## loaded via a namespace (and not attached):
##  [1] AnnotationDbi_1.26.0    BBmisc_1.6              BSgenome_1.32.0
##  [4] BatchJobs_1.2           Biobase_2.24.0          BiocParallel_0.6.1
##  [7] BiocStyle_1.2.0         DBI_0.2-7               GenomicAlignments_1.0.1
## [10] GenomicFeatures_1.16.2  RColorBrewer_1.0-5      RCurl_1.95-4.1
## [13] RSQLite_0.11.4          Rcpp_0.11.2             XML_3.98-1.1
## [16] biomaRt_2.20.0          bitops_1.0-6            brew_1.0-6
## [19] codetools_0.2-8         data.table_1.9.2        digest_0.6.4
## [22] evaluate_0.5.5          fail_1.2                foreach_1.4.2
## [25] formatR_0.10            grid_3.1.0              highr_0.3
## [28] iterators_1.0.7         knitr_1.6               plyr_1.8.1
## [31] reshape2_1.4            rtracklayer_1.24.2      sendmailR_1.1-2
## [34] stats4_3.1.0            stringr_0.6.2           tools_3.1.0
## [37] zlibbioc_1.10.0
```