An Introduction to *ggbio*

Tengfei Yin

September 13, 2011

Contents

1 Introduction

ggbio is a R package aim to provide a toolkit for visualization of biological data in terms of different kinds of static R graphics. This package is mainly built on *ggplot2* package[1], which is an elegant plotting systeim for R, based on the grammar of graphics. So we can follow similar API and at the same time, the grammar of graphics. And for those parts which require a low level manipulation on graphics. *ggplot2* may not be flexible enough, so we also develop graphics sometime based on pure *grid* or *gridExtra*.

Our goal is to provide high quality graphics for both analysis and publication purpose. So we try to follow some rules here:

- Be general.
- Be object-oriented. We provide generic function for most used R objects in Bioconductor project.
- To use *ggplot2* to develop graphics as possible as we can, but hiding details as much as possible. Most function return a `ggplot` object, which leave users more power to manipualte directly on this object, for instance, adding labels, changing color scheme.etc.
- Be easy-to-use and user-friendly.
- For specifc question or most used graphics, we provide convenient function to give users the graphics they need as simple as we can.

2 Grammar of Graphics

How to describe a statistical plot in several consecutive steps(Leland Wilkinson):

data Performs the actual statistical computations, art of the graphics pipeline

transformation, scale, coordinates Operations

geoms What is actually plotted (points, lignes, but also shapes)

guides Axes, legends and other elements that help read the plot

display Produces the picture, but should also provide interactivity (brushing, drill down, zooming)

As Figrue 1 shows, *ggplot2* redefine the operation + to make the manipulation more descriptive.

To get more information about how to create high-quality graphics by *ggplot2*, please visit the on-line documentation, you could see the most used scale/geoms/ colors/coordinate systems/facet...

However, we need to realize the very **imporatant** difference between *ggbio* and *ggplot2*.

- In most graphcis, the x-scale in *ggbio* is always on a genomic coordinates or protein space. We don't provide flexible x value as you can do in *ggplot2*, you can only specify the x to be *start, end, midpoint* and this only make difference when it's interval data with width over 1.
- Based on the rule above, most time, when we facet the graphics, we only allow users facet by rows and the column could be only faceted by space.
- Automatically facet by **seqnames**, because thsoe data are not allowed to be mixed together since their x-scale is always genomic coordinates.

[1] `http://had.co.nz/ggplot2/`

```
> library(ggplot2)
> p <- ggplot(data=mtcars) +
+     geom_point(aes(x = mpg,
+                    y = wt,
+                    colour= factor(cyl)))+
+      scale_y_log10() +
+      facet_grid(gear ~ .)
> print(p)
```

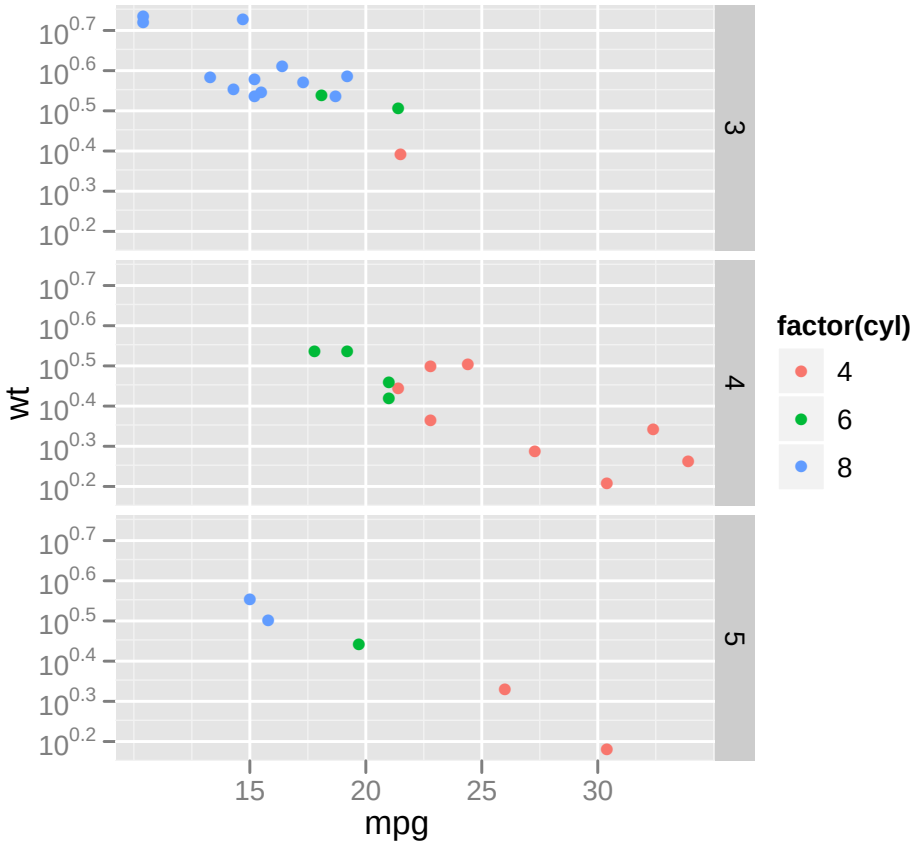

Figure 1: ggplot2 example

3 Generic Visualization Method

As mentioned above, we are trying to be general and object-oriented, at the same time following the API from *ggplot2*, so we use the *quick plot* function `qplot` in *ggplot2* package. And redefine this in to a **S4** generic function.

So now the `qplot` function could dispatch on different R objects and we also provide new **geom** for each type of object.

In the following section, we will introduce how to use `qplot` to plot different types of data in different ways.

3.1 For `data.frame` and `matrix` object

This is a wrapper around the original `qplot` function. when reading in `data.frame` or `matrix` object, you can use `qplot` as usual without any change, it essentially just call ggplot2::qplot.

```
> library(ggbio)
> p <- qplot(data = mtcars, mpg, cyl)
> print(p)
```

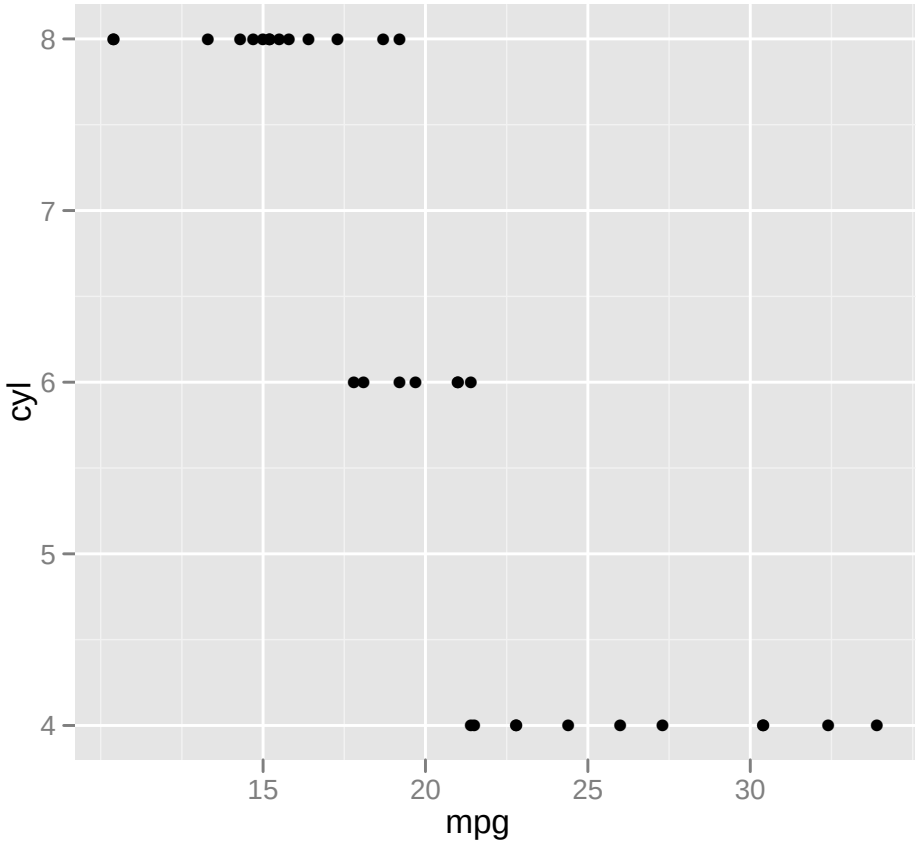

Figure 2: qplot for data.frame

3.2 For GRanges object

GRanges object is defined in *GenomicRanges* package, whic is one of the most important infrastructure to describe genomic data along meta data information. It describe genomic data as intervals, currently in R we don't have any convenient function to visualize interval object in different ways.

3.2.1 Sample Granges object

Let's first create a sample GRanges object used for following examples. This sample data contains 1000 rows, three chromosomes, with some meta data, including grouping information and pairing information which mimic paired RNA-seq data.

```
> set.seed(1)
> N <- 1000
> library(GenomicRanges)
> gr <- GRanges(seqnames =
+               sample(c("chr1", "chr2", "chr3"),
+                      size = N, replace = TRUE),
+               IRanges(
+                       start = sample(1:300, size = N, replace = TRUE),
+                       width = sample(70:75, size = N,replace = TRUE)),
+               strand = sample(c("+", "-", "*"), size = N,
+                 replace = TRUE),
+               value = rnorm(N, 10, 3), score = rnorm(N, 100, 30),
+               group = sample(c("Normal", "Tumor"),
+                 size = N, replace = TRUE),
+               pair = sample(letters, size = N,
+                 replace = TRUE))
> head(gr)

GRanges with 6 ranges and 4 elementMetadata values:
      seqnames     ranges strand |     value     score       group
         <Rle>  <IRanges>  <Rle> | <numeric> <numeric> <character>
  [1]     chr1 [160, 234]      * |  7.341551 122.17345      Normal
  [2]     chr2 [206, 280]      - |  4.233235 111.59826      Normal
  [3]     chr2 [115, 189]      + | 14.859102 138.89192       Tumor
  [4]     chr3 [287, 358]      - | 11.557810  75.89325      Normal
  [5]     chr1 [ 36, 106]      * |  9.832450  51.92123       Tumor
  [6]     chr3 [ 12,  81]      * | 12.089253 127.99753      Normal
             pair
      <character>
  [1]           y
  [2]           z
  [3]           g
  [4]           d
  [5]           b
  [6]           t
  ---
  seqlengths:
   chr1 chr2 chr3
     NA   NA   NA
```

3.2.2 Supported Geoms

For `GRanges` we support following *geoms*:

full Show full stacked interval, a set of rectangles. Default.

segment Show full stacked interval, a set of segments.

line Show line. User need to provide

coverage.line Show coverage by using line.

coverage.polygon Show coveraing by using polygon.

reduce Show reduced `GRanges` object.

disjoin Show disjoin `GRanges` object.

histogram Show hisgogram.

Now you can simply visualize a `GRanges` object.

As we could see from Figure 3, the default is *automatically* facet by existing **seqnames** in the `GRanges` object. We can use `nrow` and `ncol` to control the wrapping.

We we get the `ggplot` object, we could use all features from *ggplot2* package to manupuate this plot. Here we show a simple theme change.

or adding a global line for coverage as shown in Figure 6.

You can also subset by the `which` argument.

```
> gr.sub <- gr[seqnames(gr) == "chr1"] #or
> ## p <- qplot(gr, seqnames = "chr1", ...) # or
> ## p <- qplot(gr, which = GRanges("chr1", IRanges(1e5, 2e5)), ...)
```

And let's plot all other geoms together by `grid.arrange` from package *gridExtra*

```
> p1 <- qplot(gr.sub, geom = "full")
> p2 <- qplot(gr.sub, geom = "point", y = value)
> p3 <- qplot(gr.sub, geom = "line", y = value)
> p4 <- qplot(gr.sub, geom = "coverage.line")
> p5 <- qplot(gr.sub, geom = "coverage.polygon")
> p6 <- qplot(gr.sub, geom = "reduce")
> p7 <- qplot(gr.sub, geom = "disjoin")
> p8 <- qplot(gr.sub, geom = "histogram")
> library(gridExtra)
```

```
> p <- qplot(gr)
> print(p)
```

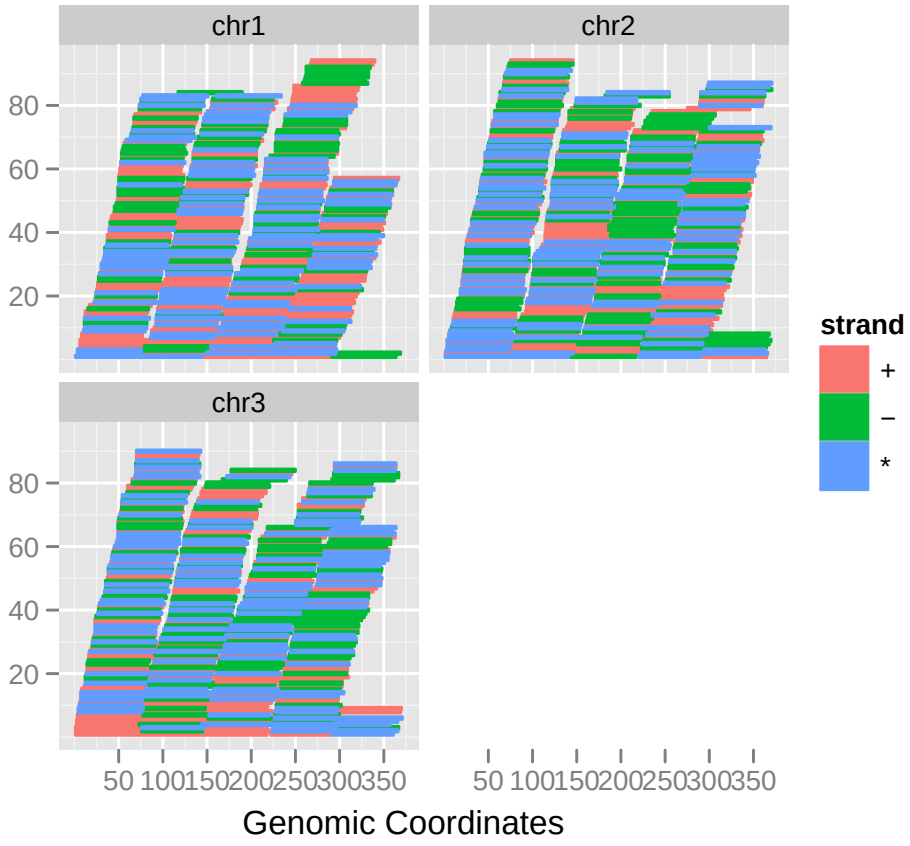

Figure 3: qplot for GRanges as geom full

```
> p <- qplot(gr, nrow = 1)
> print(p)
```

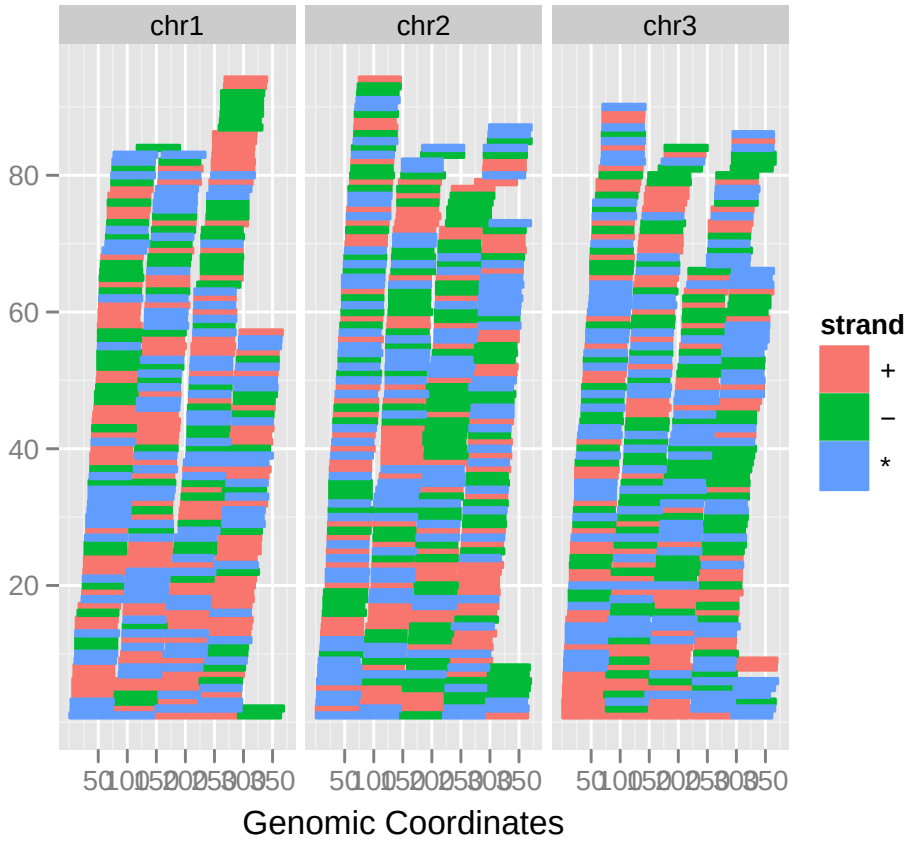

Figure 4: qplot for GRanges as geom full

```
> class(p)

[1] "ggplot"

> p <- p + theme_bw()
> print(p)
```

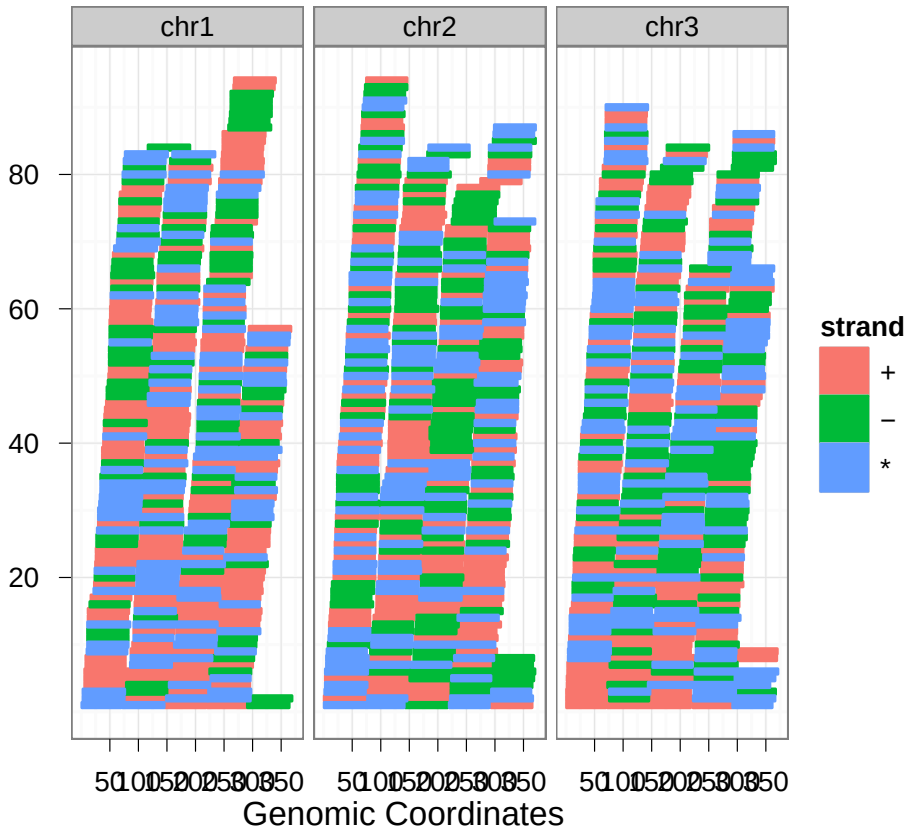

Figure 5: qplot for GRanges as geom full

```
> p <- qplot(gr, nrow = 1, geom = "coverage.p")
> p + geom_hline(yintercept = 40, color = "red",
+     size = 1)
```

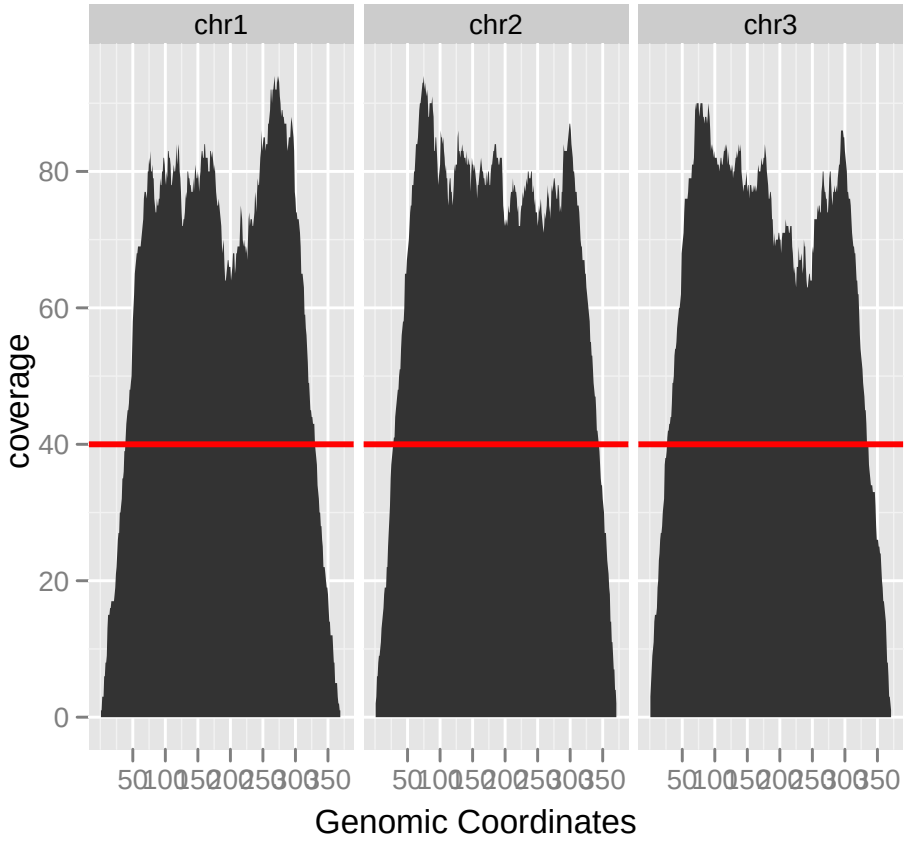

Figure 6: qplot for GRanges as geom full

3.2.3 Faceting

3.3 For `GRangesList` object

3.4 For `IRanges` object

3.5 For `GappedAlignments` object

3.6 For `BamFile` object

3.7 For `TranscriptDb` object

3.8 For `BSgenome` object

4 Overview

Bird Eye Overview is useful to see the overal distribution of certain events. For static graphic, we currently only support stacked overview as ideogram, or for single chromosome.

4.1 Stacked Overview

Stacked overview is useful to visualzie the annotation across the genome, you can use `plotOverview` function to directly plot the result from `getIdeogram` for certain species. And you could control wether to plot the cytoband or not.

Figure **??** shows how to plot stacked overview with cytoband. We change the name to make the label more clear.

Clearly, it's not good for visualizing the annotation at the same time, so we could plot it without cytoband. This accept a full ideogram which will be reduced automatically. You could also just use **hg19Ideogram** dataset.

Then we could simply use `geom_hotregion` function to read in a `GRanges` object as other geoms(except they read in `data.frame`). And use + to simply add a annotation track on top with overview, they will automatically plot on the same chromosome and on the same x scale.

Figure **??** shows an example of subset of RNA editing set,

We can also use `color` argument to use color to indicate a column in the elementMetadata.

4.2 Circular Overview

Circular view is inspired by the Circos project [2] which is essentially writen Perl[3].Circos visualize data in a circular layout, originally starting from visualize the genomic data, then extends to many other fields, turn out to be an elegant and useful way to visualize some other infomation.

The static version of circular view is not implemented in this pacakge yet, but it's defintely in the TODO. For users who are really interested in using a circular view in R, we have a highly experimental circular view in another package *visnab*, which is interactive visualization toolkit for genomic data.

[2] `http://circos.ca/`

[3] `http://www.perl.org/`

```
> data(hg19IdeogramCyto)
> ## make shorter and clean labels
> ideo.rmChr <- removePrefix(hg19IdeogramCyto, "chr")
> p <- plotOverview(ideo.rmChr, cytoband = TRUE)
> print(p)
```

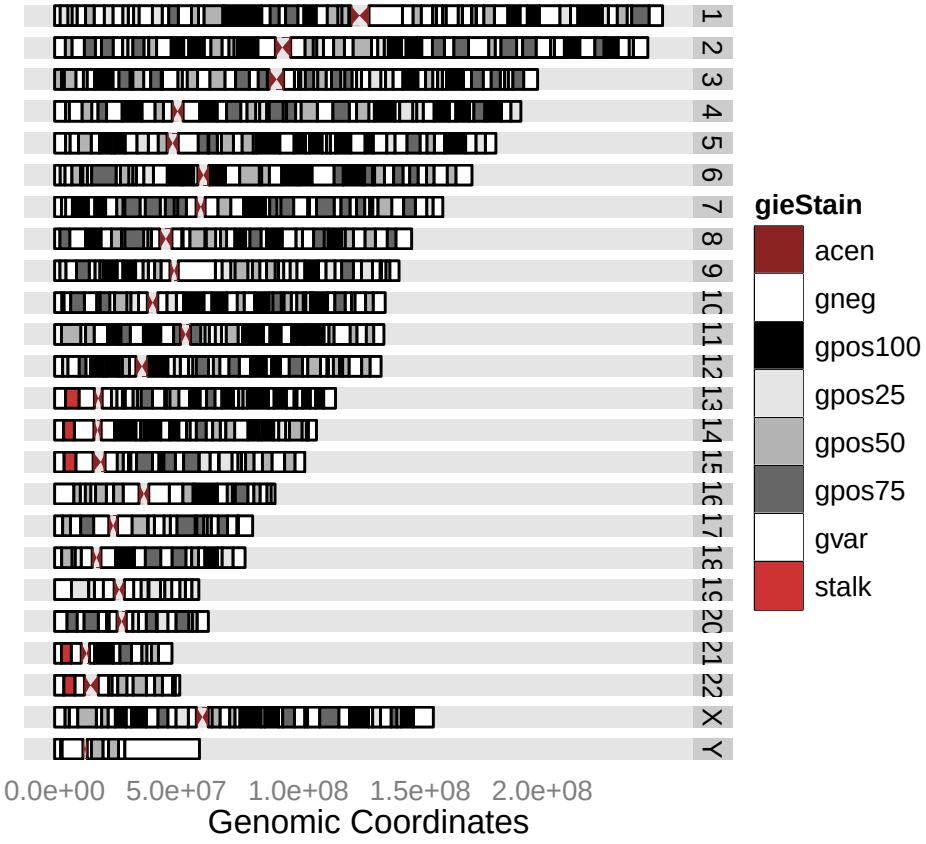

Figure 7: Stacked overview with cytoband

```
> p <- plotOverview(ideo.rmChr, cytoband = FALSE)
> print(p)
```

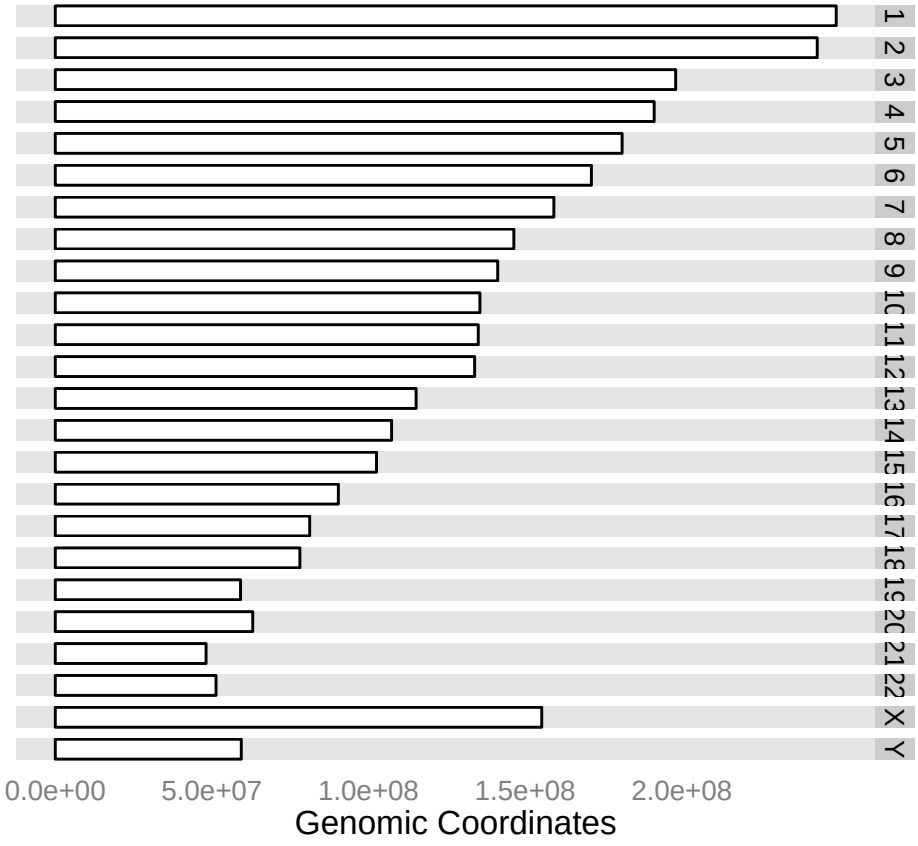

Figure 8: Stacked overview without cytoband

```
> p <- plotOverview(ideo.rmChr, cytoband = FALSE)
> data(darned_hg19_subset500)
> new.darned <- removePrefix(darned_hg19_subset500, prefix = "chr")
> p <- p + geom_hotregion(new.darned)
> print(p)
```

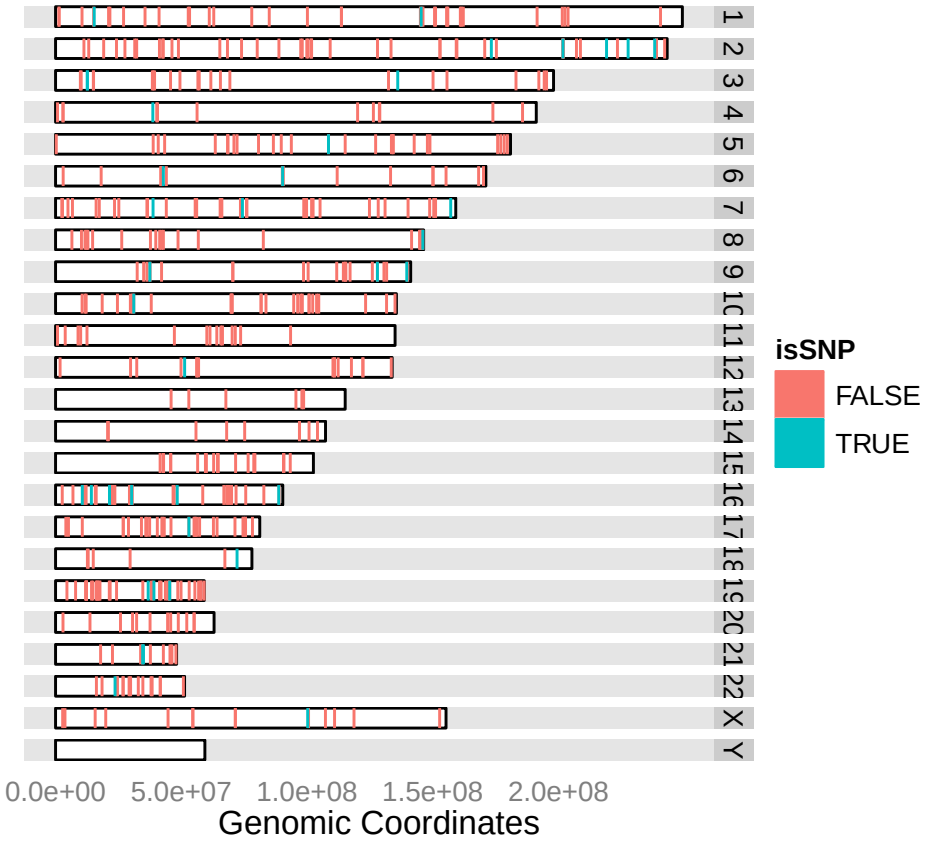

Figure 9: Stacked overview without cytoband and with subseted DARNED data on it

```
> p <- plotOverview(ideo.rmChr, cytoband = FALSE)
> data(darned_hg19_subset500)
> new.darned <- removePrefix(darned_hg19_subset500, prefix = "chr")
> values(new.darned)["isSNP"] <- sapply(as.character(values(new.darned)$snp), nchar)>0
> p <- p + geom_hotregion(new.darned, aes(color = isSNP))
> print(p)
```

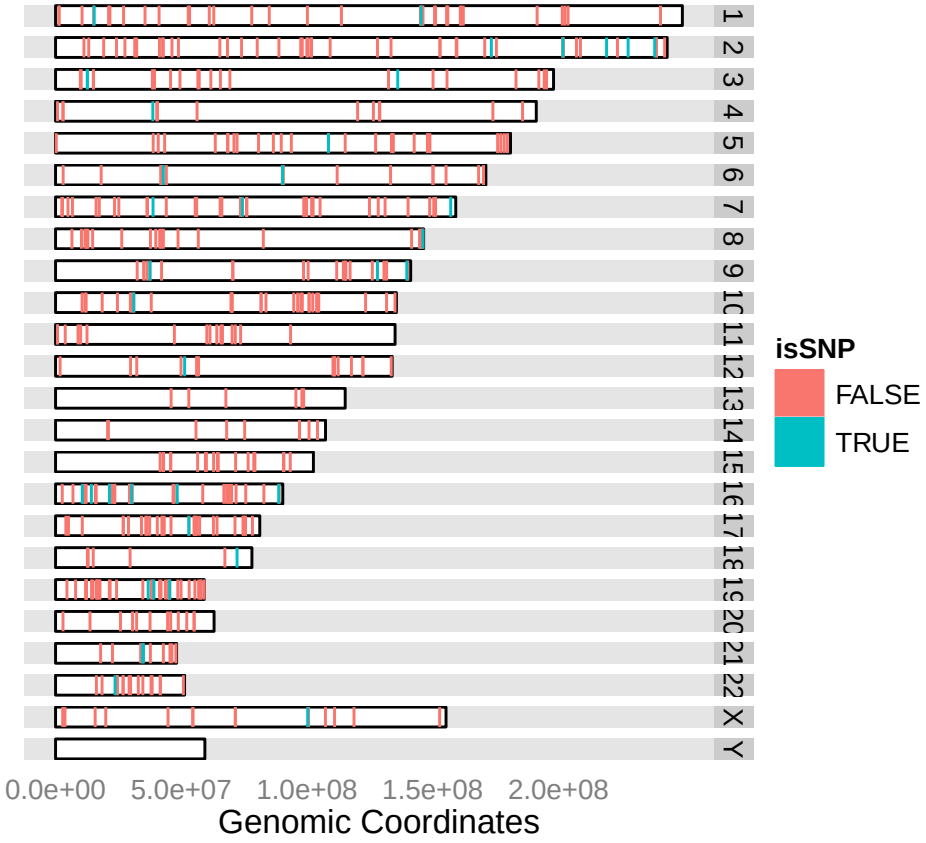

Figure 10: Stacked overview without cytoband and with subseted DARNED data on it

```
> vp1 <- viewport(width = 1, height = 0.14)
> p <- plotSingleChrom(hg19IdeogramCyto, subchr = "chr1")
> print(p, vp = vp1)
```

Figure 11: Single Chromosome as Ideogram

```
> vp2 <- viewport(width = 1, height = 0.14)
> p <- plotSingleChrom(hg19IdeogramCyto, subchr = "chr1",
+                  zoom.region = c(1e8, 1.5e8))
> print(p, vp = vp2)
```

Figure 12: Single Chromosome as Ideogram with zoomed rectangle

5 Building Tracks for Linear View

6 Question-oriented Specific Graphics

6.1 Fragment Length and Splicing plots

6.2 Sequencing Logo

To be implemented.

6.3 Manhattan Plots for SNP data

To be implemented.

7 Session Info

```
> sessionInfo()

R Under development (unstable) (2011-08-08 r56671)
Platform: x86_64-unknown-linux-gnu (64-bit)

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

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

other attached packages:
[1] GenomicRanges_1.5.36 IRanges_1.11.26      ggbio_0.0.7
[4] ggplot2_0.8.9        proto_0.3-9.2        reshape_0.8.4
[7] plyr_1.6             biovizBase_0.8.9

loaded via a namespace (and not attached):
 [1] AnnotationDbi_1.15.11  Biobase_2.13.9
 [3] biomaRt_2.9.2          Biostrings_2.21.9
 [5] BSgenome_1.21.3        colorspace_1.1-0
 [7] DBI_0.2-5              dichromat_1.2-3
 [9] digest_0.5.0           GenomicFeatures_1.5.17
[11] munsell_0.3            RColorBrewer_1.0-5
[13] RCurl_1.6-10           Rsamtools_1.5.59
[15] RSQLite_0.9-4          rtracklayer_1.13.13
[17] scales_0.1.0           stringr_0.5
[19] XML_3.4-2              zlibbioc_0.1.7
```