A tutorial for the iGenomicViewer R package

Lori A. Shepherd and Daniel P. Gaile

April 6, 2012

Statistical Genetics and Genomics Research Group
Department of Biostatistics, University at Buffalo
New York State Center of Excellence in Bioinformatics and Life Sciences
Roswell Park Cancer Institute

las65@buffalo.edu

Contents

1 Introduction

The iGenomicViewer package is a wrapper to the sendplot library that contains functions for interactive, generic, genomic plots. The functions in the sendplot library allow R users to generate interactive plots with tool-tip content. A pair of files are created: a Portable Network Graphics (PNG) file which is a bitmap image [or Joint Photographic Experts Group (JPEG)] and an HTML file which contains embedded Javascript code for dynamically generating tool-tips. When opened with a supported browser, the HTML file displays the PNG [JPEG] image and the user is able to mouse over and view tool-tip windows for user-specified image locations. The information that appears in the tool-tip windows is user specified and highly customizable. The tool-tip functionality is provided by code from the wz_tooltip.js Javascript library (Zorn 2007) which is embedded in the HTML output. Please see the sendplot documentation available on CRAN (http://cran.r-project.org/) or the University at Buffalo Biostatistics Research Software Page (http://sphhp.buffalo.edu/biostat/research/software/index.php). The iGenomicViewer functions are platform independent with respect to data, which allows for a completely generic and customizable plot. As long as identifiers have genomic locations and chromosome information, the data can be used. The ability to utilize any mapping and to create any customized annotation, through identification of genomic locations enhances the utility and adaptablitiy of the application.

There are two main functions to initialize plotting objects in the 'iGenomicViewer' library: initGGV and initTiled. These functions create objects that contain the necessary information to make an interactive layout of genomic plots. The library contains four main functions for plotting: makeGGV, iGGVtiled, iGGV, and makeTiled. Brief descriptions of the six functions are as follows:

- initGGV : initializes a 'GGVobj', generic genomic viewer object, to use with makeGGV. See appendix A.3 for more details on object structure.
- initTiled : initializes a 'TIplot', tiled image plot, object to use with iGGtiled or makeTiled. See appendix A.5 for more details on object structure.
- makeGGV : creates a series of interactive plots across the genome.
- iGGVtiled : creates an interactive layout of plots with a tiled image as the main heatmap. A tiled image depicts the overlap and gaps in spot.ID coverage.
- iGGV : creates a single interactive layout of plots.
- makeTiled : creates a single static layout of plots with a tiled image as the main heatmap.

The functions in the iGenomicViewer library allow for an interactive layout of genomic plots. The layout of plots will have a main heatmap which can be

the standard view or a new tiled view, and a legend for the heatmap. The tiled view should be used for small genomic regions to investigate overlap and gaps in spot.ID coverage. The layout of plots can optionally contain a customizable annotation plot, showing any number of different annotations simultaneously, as well as an optional additional, customized genomic plot specifically designed in the interest of depicting values of statistical analysis.

The remainder of this document will provide detailed tutorials for the use of the functions: iGGV, iGGVtiled, and the other main iGenomicViewer functions. All sections assume library has been loaded and will use the example dataset, iGGVex, provided:

```
> library(iGenomicViewer)
> data(iGGVex)
```

Important Note: The iGenomicViewer/sendplot output has been tested on Firefox and Internet Explorer browsers. Internet Explorer users may need to modify their preferences to allow blocked content, as Internet Explorer may initially block the scripts from running. A warning message normally appears towards the top of the browser; if the user clicks on this warning, it will give an option to allow blocked content.

2 Initialize Objects

The applications take an object oriented approach. It is first necessary to initialize either a generic genomic viewer 'GGVobj' or a tiled image 'TIplot' object.

2.1 initGGV

The initGGV function initializes a 'GGVobj', a generic genomic viewer object. See appendix A.3 for more details on object structure. The following shows the function definition, note which arguments must be defined and which have default values:

```
initGGV(vls,
        mapObj,
        annObj,
        x.labels=NA,y.labels=NA,xy.labels=NA,
        x.links=NA,y.links=NA,xy.links=NA,asLinks=NA,
        x.images=NA, y.images=NA, xy.images=NA,
        chrArms=NA, trackRegions=NA,
        side.plot.extras=NA,plot.vec=NA,plot.dx=NA,
        maxLabels=25,mat = NA,mai.mat = NA,mai.prc=FALSE,
        plot.extras=NA,smpLines=TRUE,divCol="lightgrey",lims = c(-0.5,0.5),
        annotation = NA,clrs=c("blue", "hotpink", "purple", "orange"),
        mapObj.columns = NA,
        returnVl=TRUE,saveFlag=FALSE,saveName="GGVobj.RData")
```

Figure 1 is an example of one of the plots generated, arm 6p. Note the heatmap with legend, the annotation track, and the additional side plot.

2.1.1 specifiying the heatmap matrix, mapping object, and annotation object

The vls argument of initGGV is a matrix of values to be used for the heatmap. The y, or first dimension, should correspond to genomic locations. This length should be equivalent to the mapObj's number of spot.IDs. The vls matrix therefore directly corresponds to the mapping object. Please see section 4.2 for more details on the mapping object. The user will be given an opportunity to subset the spot.ID's when executing the plots; the user should NOT attempt to subset the y axis/genomic locations at this step. The x, or second dimension, corresponds to samples.

This function assumes that a mapping object and annotation object have already been created. Please see sections 4.2 and 4.3 for more details of generating these objects. The function provides default objects which will be used.

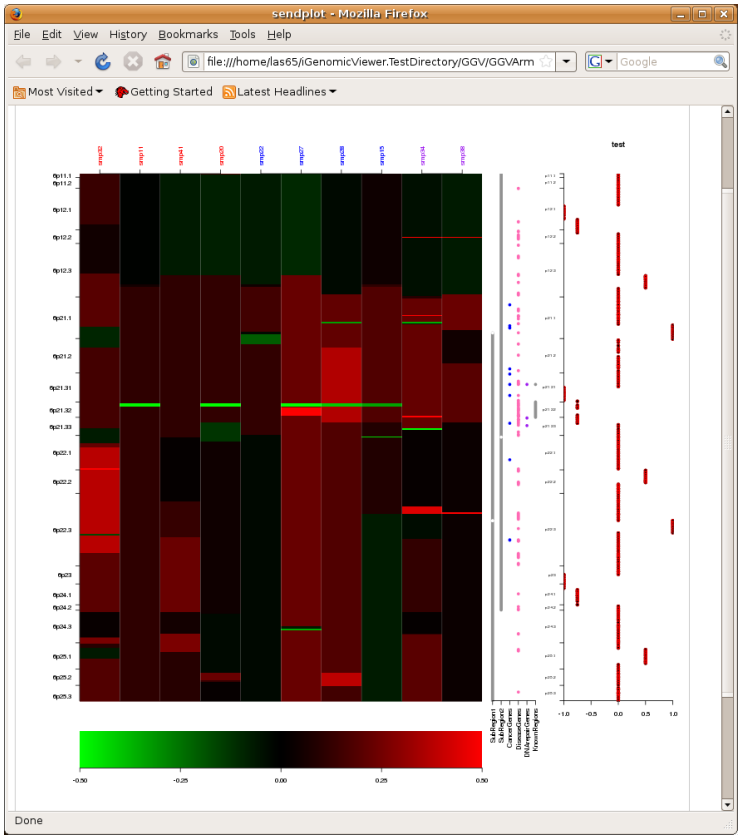

Figure 1: chromosome arm 6p

```
vls = round(mat, 3)
data(mapping.info)
mapObj = mapping.info
data(annObj)
```

2.1.2 specifying the tool-tip content and incorporating hyperlinks

The x.labels, y.labels, and xy.labels control what is displayed in the interactive window when the user hovers the mouse over heatmap subregions. The x.labels and y.labels arguments refer to data that is specific to x [sample data] and y [genomic data] respectively. x.labels and y.labels are data.frames of the dimension n by m. For x.labels, n is equal to the number of samples, or the vls matrix second dimension; for y.labels, n is equal to the number of spot.ID's, or the vls matrix first dimension. Each row is specific to a certain x or y value and each column is a unique variable or characteristic of x or y respectively. The first row of the data.frames should contain column headers; these names will be used as diplay names in the interactive window that appears. The xy.labels argument is slightly different; it governs data specific to both x and y locations. The function argument xy.labels is a list of matrices; each matrix is of the same dimension as vls.
Additional genomic information from the given mapping object may be displayed in the interactive window. The mapObj's mapping.info object is a data.frame with information for each spot location. The user may include any, all, or none of these columns using the mapObj.columns argument. The argument is a numeric vector or a character vector indicating which of the mapping.info data.frame columns to include. All columns may be included by specifing mapObj.column as NA. None of the columns are included if mapObj.columns is 0.
Consider the iGGVex. Looking at the possible y.labels and mapObj.columns options:

```
> names(y.lbls)

[1] "spot.ID"  "map.flag" "Pdisc"

> names(mapObj$mapping.info)

 [1] "Spot.ID"      "Chrom"        "loc.start"    "loc.stop"     "loc.center"
 [6] "Mapped.by"    "Flag"         "g.loc.start"  "g.loc.center" "g.loc.stop"
```

The y.lbls data.frame already contains spot.ID but nothing indicating location. Chromosome location and genomic start and stop locations are taken from the mapping object.

```
x.labels=x.lbls
xy.labels = list(lgr=vls)
```

```
y.lbls$Pdisc = round(y.lbls$Pdisc,3)
y.labels = y.lbls
mapObj.columns = c(2,8,10)
```

Hyperlinks may be included through the asLinks, x.links, y.links, and xy.links arguments. The x.links, y.links, and xy.links behave similarly to xy.labels, y.labels, and xy.labels respectively, however, they contain complete web addresses as character strings. The asLinks argument has several acceptable forms. It may be a matrix or data frame with the same dimensions as vls. asLinks may also be a vector of length equal to length of x times length of y, thus a vector version of the aforementioned matrix or data frame. These options may be useful when xy specific hyperlinks are desired (similar to an xy.lbls argument). asLinks may also be a vector of length equal to the length of x or y, indicating x or y specific hyperlinks. If asLinks is of length x, the vector will be repeated along the length of y so that every similar x value will be the same hyperlink, and vice-versa for y. If asLinks is of length one and is not NA, the value will be repeated for every grid location. NA represents a point that is not a hyperlink. Every asLink entry should be a character string for a complete web address or NA.
Images may also be included in the tool-tip through the x.images, y.images, and xy.images arguments. The x.images, y.images, and xy.images argument behave similarly to x.labels, y.labels, and xy.labels, however, they contain paths to images as character strings.

2.1.3 specifying chromosome arms and known regions of interest

When the GGVobj is used in makeGGV, a series of interactive plots are created. Specific chromosome arms and known regions of interest may be indicated for plotting. The chrArms argument is a list of chromosome arms that should be plotted. The format of how arms are indicated should match the mapObj's band.info information for arms. In makeGGV, an index html with these chromosome arms listed is created. Known regions of interests for example, a gene or band that is listed in literature as significant, may be identified through a trackRegion object. Please see secion 5.2 and appendix A.6 for more details on making a trackRegion object. A tiled image heatmap is created automatically for each of these known regions. The regions are also displayed as part of the annotation track on chromosome arm plots.
For the given example, chosen at random, arms 8p and 18p will be deemed chromosome arms of interest. Also chosen at random, regions 8p11.22, 6p21.32, 18p11.21 and gene FANCE will be known regions of interest. Note in the following, the makeTrack function is used. Please see section 5.2 for more details.

```
chrArms = c("8p", "18p")
trackRegion = makeTrack(Fine.Band = c("8p11.22","6p21.32","18p11.21"),
                        genomicLoc = NA, geneName = "FANCE")
```

2.1.4 adding an additional [statistical] genomic plot

When using this application for datasets, it was requested that an additional, optional plot be allowed to show statistical values. This may be done using the arguments side.plot.extras, plot.vec, and plot.dx. This plot is added to the right of the annoation track. The argument plot.vec contains the x-axis values for the plot. It is assumes the y-axis is genomic locations. The y-axis values will be automatically determined based on plot.dx. **Note:** The plot.vec argument should be in regards to the entire genome. No subset for chromosomes or regions should be used. Multiples of the dimension are allowed to account for say two values for each y-value as in the case for frequency gain and frequency loss, etc. These values that will be automatically subset based on a given index or viewing window. The side.plot.extras argument is a character value containing additional plotting features for this side plot. Multiple plotting may be specified by separating commands with a semicolon. See the plot.extras argument more details, as it behaves the same except that it is a single variable not a list. When evaluated, the plot will be interactive with the x-values and any genomic specific data is added to the main heatmap. When makeGGV is used, it not only creates the index of chromosome arms mentioned in the previous section, but also a genomic plot of statistical values, if a plot.vec is specified. It may be the case that a specific chromosome arm or region is desired instead of having this opening plot across the entire genome. The argument plot.dx, is the index to subset plot.vec when creating this initial genomic plot.
Consider the following:

```
pvls = rep(rep(rep(c(-1,rep(0,3),1,rep(0,3),.5,rep(0,3),-.75), each=10),
           150))[1:length(mapObj$mapping.info$g.loc.center)]
plot.vec = pvls[1:length(mapObj$mapping.info$g.loc.center)]
side.plot.extras="points(pvls, GGV$values$mapObj$mapping.info$g.loc.center,
                  col='red', pch=21); title(main='test')"
plot.dx=which(mapObj$mapping.info$Chrom=="chr8")
```

This is a 'toy' example plot and does not depict real data. The values are repeated 10 times each covering the length of the genome. The genomic plot initially created would focus on chromosome 8. Notice that the call is a character string that will be evaluated as multiple function calls separated by a semicolon. Arguments of type character within these calls are specified with a single quotation rather than the double quotations used originally, or vice versa (see col argument). Any variables used in arguments should be in local memory before running the function to evaluate the GGVobj. Besides subsetting reasons, this is also why we recommend using plot.vec and mapObj$mapping.info$g.loc.center whenever possible.

2.1.5 controlling plotting features

The following arugments will be mentioned briefly. They help control some of the plotting features. If the user does not specify these argument, default settings will be used.

- maxLabels : maximum number of labels to appear on the heatmap y axis. Based on this number, the function will automatically determine if arms, broad.band, fine.bands, or individual spot.ID's should appear for the y axis.

- mat : matrix indicating layout. This argument will be passed into the graphics package layout call as mat. Each value in the matrix must be '0' or a positive integer. If N is the largest positive integer in the matrix, then the integers 1,...,N-1 must also appear at least once in the matrix. '0' indicates region of no plotting. This may be left as NA, and a default will be used. This matrix will be used for Chromosome Arm and Sub.Arm Plots. This is left as an argument in case the user finds the default plots too large or small based on customization. N is 3 if plot.call is NA, and 4 if plot.calls is specified.

- mai.mat : n x 4 matrix of values to be passed in for each plots par mai. n will be 3 if plot.call is NA, and 4 if plot.calls is specified. This will be used for Chromosome Arm and Sub.Arm plots. The four columns represent the four different plot margins: bottom, left, top, right respectively.

- mai.prc : logical indicating if mai mat values are percentages of original size or hard coded values. If mai.prc is T, indicates percentage. This will be used for Chromosome Arm and Sub.Arm plots.

- plot.extras : List of length equal to the number of plots: 3 if plot.call is NA, 4 if plot.call is specified. This object is a list of lists. The sublists contain any additional plotting calls that should be executed for the plot. Each entry must be a character vector. If no additional plotting is equired, an NA should be used.

- smpLines : logical indicating if vertical lines should be added between each sample of the heatmap.

- divCol : If smpLines, the color of the dividing lines.

- lims : Lower and upper limit for vls. Any value above of below will be changed to max and min value respectively.

Note: The arguments mat, mai.mat, and mai.prc mention they are for Chromosome Arm and Sub.Arm plots. When using the makeGGV, the mat, mai.mat, and mai.prc for tiled images may be specified in the function call.

2.1.6 controlling annotation plotting

The annotation track is dependent on the annotation object used. Please see section 4.3 and appendix A.1 for more details on making the annotation object. Any or all of the annotation tracks may be displayed through the annotation argument. It is a numeric indication of which annotation information objects to include from the annObj. If NA all are used. The colors for the different tracks are controlled by the argument clrs. It will use the vector of clrs in order. When plotting, the function adds annotation tracks for subsetting the chromosome region and for displaying known regions of interests. These tracks are always shown in gray.

2.1.7 returning and saving object

The final arguments for initGGV are returnVl, saveFlag, and saveName. If the user wishes the newly created GGVobj to be returned, returnVl should be TRUE. If the user wishes the newly created GGVobj to be saved to a file, saveFlag should be TRUE. If saveFlag, saveName is the path and file name to save the object.

2.1.8 summary of code used to generate 'GGVobj'

Let's recap the code thus far and put it together with the initGGV function call:

```
vls = round(mat, 3)
data(mapping.info)
mapObj = mapping.info
data(annObj)

x.labels=x.lbls
xy.labels = list(lgr=vls)
y.lbls$Pdisc = round(y.lbls$Pdisc,3)
y.labels = y.lbls
mapObj.columns = c(2,8,10)

chrArms = c("8p", "18p")
trackRegions = makeTrack(Fine.Band = c("8p11.22","6p21.32","18p11.21"),
                         genomicLoc = NA, geneName = "FANCE")

pvls = rep(rep(rep(c(-1,rep(0,3),1,rep(0,3),.5,rep(0,3),-.75), each=10),
            150))[1:length(mapObj$mapping.info$g.loc.center)]
plot.vec = pvls[1:length(mapObj$mapping.info$g.loc.center)]
side.plot.extras="points(pvls, GGV$values$mapObj$mapping.info$g.loc.center,
                  col='red', pch=21); title(main='test')"
plot.dx=which(mapObj$mapping.info$Chrom=="chr8")
```

```
GGV = initGGV(vls = vls,
              mapObj = mapObj,
              annObj = annObj,
              x.labels=x.labels,
              y.labels=y.labels,
              xy.labels=xy.labels,
              chrArms=chrArms,
              trackRegions=trackRegions,
              side.plot.extras=side.plot.extras,
              plot.vec=plot.vec,
              plot.dx=plot.dx,
              mapObj.columns=mapObj.columns,
              smpLines=TRUE,
              divCol="lightgrey")
```

2.2 initTile

The initTile functions initializes a 'TIplot', tiled image plotting object. See appendxi A.5 for more details on object structure. The following shows the function definition, note which arguments must be defined and which have default values:

```
initTile(Z,
         bacDX,
         goodDX=NA,mapObj=NA,H=2,
         zlims=c(-0.5,0.5),smplDX=NA,
         ylabels=NA, xlabels=NA,
         x.axis.cex =0.5,y.axis.cex =0.5,
         xlab="samples",ylab="BAC location",ttl=NA,
         returnVl=TRUE,saveFlag=FALSE,saveName="TIplot.RData")
```

2.2.1 specifying heatmap matrix, mapping object, and tiling

The Z argument of initTile is a matrix of values for image. The number of rows and columns should be equal to the lenghts of bacDX and smplDX. If the matrix is larger the matrix will be subset based on bacDX and smplDX. Z, therefore, may either be a complete or already subset matrix of values. Zlims controls the maximum and minumum values in Z. Any value in Z outside the xlim range will be rounded to the min and max value respectively.
This function assumes that a mapping object and annotation object have already been created. Please see sections 4.2 and 4.3 for more details of generating these objects. The function provides default objects which will be used.
The number of tracks or tiles the spot.ID's will be broken into is controlled by H. **Helpful Hint:** If an error occurs regarding Ysegs, an incorrect number of

dimension, the number of spot.IDs requested in the bacDX is too small to split into the given number of tracks. Try making H smaller.

Consider the example which uses the example data to break the range into three different tracks:

```
data(mapping.info)
mapObj = mapping.info
Z = mat
H=3
zlim=c(-.5,.5)
```

2.2.2 controlling and subsetting data

The bacDX is the range of spot.IDs to graph. The bacDX should correspond to the index of spot.ID's in the mapping object, mapObj. This will be used to determine the genomic starting and stopping locations for the plot. If the dimension of Z is larger than the bacDX, the function assumes the full matrix of values has been given and will subset Z based on bacDX. There may be instances where users know certain spots to be of 'bad' or 'questionable' quality. These spots may be removed through the use of goodDX. goodDX is a list of acceptable y values and should also correspond to the numeric location in the mapObj$mapping.info data.frame. The intersect of bacDX and goodDX is used to find acceptable spots. If no goodDX is given, goodDX=NA, all spots are assumed to be used.
Similarly, a sample index may be specified using smplDX. smplDX is a subset for the x axis. If Z is larger than or equal to the length of smplDX, Z is subset based on smplDX.

```
bacDX = 103:112
smplDX = 1:10
goodDX = NA
```

The above uses the first ten samples. The bac range is from spot.IDs 103 to 112 and all spots are of good quality..
Figure 2 is an example of a tiled image. Notice how each sample track has multiple columns showing the spot overlap and gaps.

2.2.3 controlling axis labels and size

The following arugments will be mentioned briefly. They help control some of the plotting features. If the user does not specify these argument, default settings will be used.

- xlab : main x axis label for plot.
- ylab : main y axis label for plot.

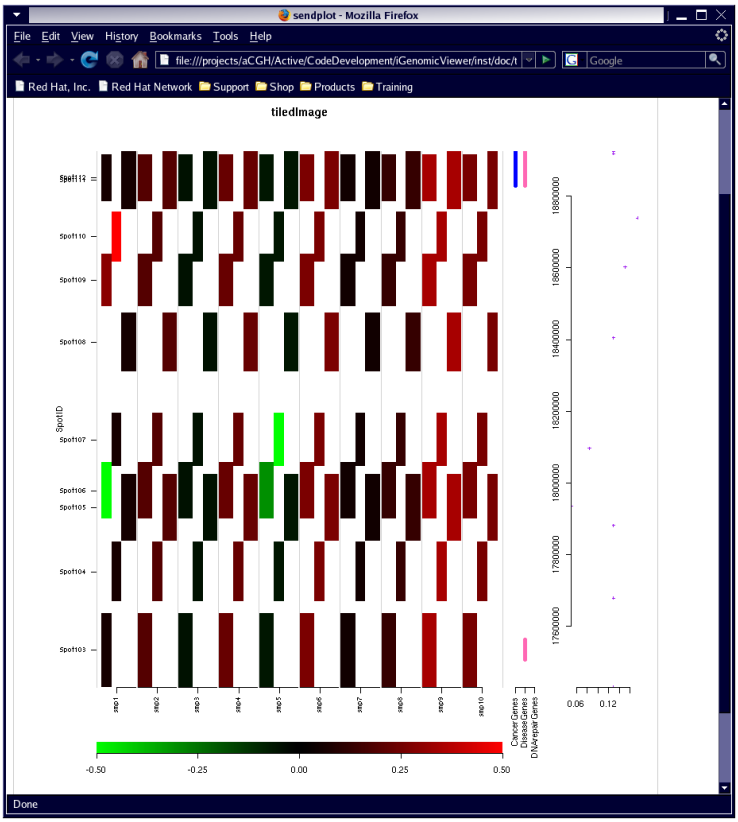

Figure 2: tiled image plot

- ttl : main title for plot.
- x.axis.cex: display size of xlabels.
- y.axis.cex: display size of ylabels.
- ylabels: vector indicating labels for Y axis. Should be equal in length to the number of rows in Z or Y.
- xlabels: vector indicating labels for X axis. Should be equal in length to the number of columns in Z.

2.2.4 returning and saving object

The final arguments for initTile are returnVl, saveFlag, and saveName. If the user wishes the newly created TIplot to be returned, returnVl should be TRUE. If the user wishes the newly created TIplot to be saved to a file, saveFlag should be TRUE. If saveFlag, saveName is the path and file name to save the object.

2.2.5 summary code used to generate 'TIplot'

Let's recap the code thus far and put it together with the initTile function call:

```
data(mapping.info)
mapObj = mapping.info
Z = mat
H=3
zlim=c(-.5,.5)
bacDX = 103:112
smplDX = 1:10
goodDX=NA

TIplot = initTile(Z=Z,
                  bacDX=bacDX,
                  mapObj=mapObj, smplDX=smplDX,
                  H=3,zlims=zlim,
                  ylabels=paste("Spot",bacDX, sep=""),
                  xlabels=paste("smp",smplDX, sep=""),
                  xlab="Samples",ylab="SpotID",ttl="tiledImage")
```

2.3 Skipping object initialization

It is possible to make a single interactive graph through the iGGV function. Utilizing this function will allow the user to skip initializing an object. Please see section 3.3 for more details.

3 Making Plots

Now that the objects are initialized, it is possible to make interactive layouts of plots.

3.1 MakeGGV: plot a 'GGVobj' object

The makeGGV function call creates and populates a directory structure of interactive, linked genomic plots. The linked html and image output allows users to examine genomic wide plots and then drill down to visualizations of regions of interest. At the topmost level, an index of identified chromosome arms of interest and, optionally, a highly customizable genomic wide plot of values are generated. These plots link to chromosome arm displays. On these displays users can interrogate a panel of plots which include: 1) a heat map of the data with tool-tip display of sample and assay specific data, all data displayed is user customized (e.g., assay values, sample IDs, and hyperlinks to UCSC browser and sample specific images); 2) a set of interactive customized annotation tracks (e.g. display location of cancer, disease and DNA repair genes); 3) an optional plot which displays statistical values such as -log10 p-values or aberration frequencies for the spot assays depicted in the heatmap. For the smallest regions of interest, the panel of plots contains a tiled heatmap which depict the overlap and gaps in spot coverage, which can be especially useful in context of gene locations represented in the adjacent annotation track.
To account for high dimension data, the heatmap is not interactive at the second most level, the chromosome arms. The function splits the chromosome arm into sub-arm plots. This is designed to maintain efficiency while still allowing interactivity of large datasets. Two tracks are added by the function to the annotation plot to the right of the heatmap: SubRegion1 and SubRegion2. When the user hovers over a section of the gray bar, a tool-tip will display containing a link to a subregion. If the user clicks on this link, a plot depicting the region contained within the length of that section of the gray bar will appear. This plot is fully interactive, including the main heatmap.
Layouts containing a tiled heatmap image will be generated for any regions specified in the GGVobj's trackRegion, or object containing known regions of interest. The function adds another track to the annotation plot for chromosome arms and sub-arm plots: KnownRegions. If the user hovers over this gray track, information about the region along with a link to the tiled plot is displayed.

Figure 3 show the different levels of plots generated by the makeGGV function. The Index file lists different regions of interest, while a genomic plot, if utilized, displays a graphical region of interest seen in 3A. The chromosome arm plots are the next level, 3B. This is followed by 3C a closer interactive heatmap. The smallest level, 3D, is a tiled image of a particular region.

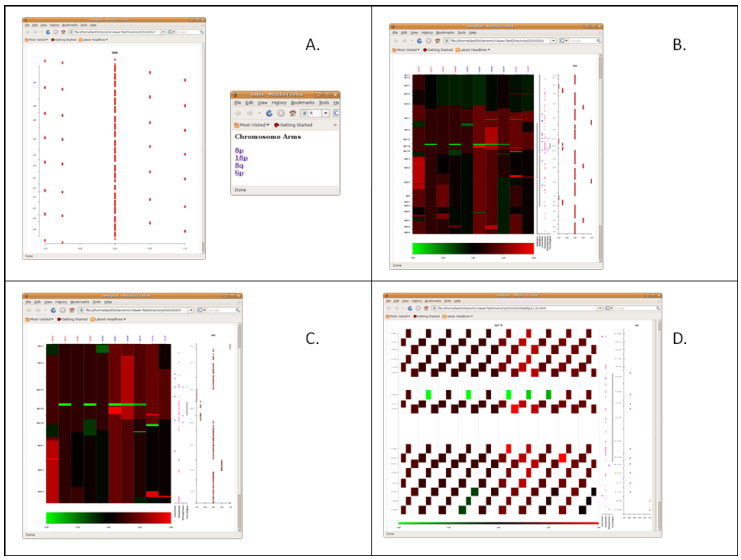

Figure 3: Different levels of plots when using makeGGV function

Individual plot sets can be sent via email, and the larger directory structure can be placed on password protected servers. This allows for ease of sharing select data with investigators and collaborators globally.

The following shows the function definition, note which arguments must be defined and which have default values:

```
makeGGV(GGV,
        goodDX=NA,smplDX=NA,
        smp.color=NA,break.num = 125,
        tileNum = 2,buffer = 5,
        makeWinArms=TRUE,
        tiledMat=NA,tiledMai.mat = NA,tiledMai.prc=FALSE,
        fname.root="iGGV",dir="GGV/",
        overwriteSourcePlot = NA, header="v3",
        window.size = "800x1100",image.size = "800x1100",
        tiled.window.size = "800x1100",tiled.image.size = "1200x1100",
        cleanDir = TRUE)
```

3.1.1 specifying objects, spot index and sample index

The GGV argument is a 'GGVobj' object. Please see section 2.1 and appendix A.3 for more information on creating 'GGVobj'. The example will assume that

the object in section 2.1 has been created. The GGV object contains all information needed to make a directory structure of interactive, linked genomic plots.
As data is preprocessed, it may become apparent that some spots may be 'faulty' or have resulted in 'bad quality' data. If data is not trusted for certain spots it is possible to remove them. This is accomplished through goodDX. The argument goodDX is a numeric list of acceptable y values with respect to the mapObj$mapping.info object. Any spot that is not listed in goodDX will be removed and not plotted on any of the plots. The default, when goodDX is NA, is to assume all spots should be utilized.
It is also possible to specify a select group of samples to plot. The argument smplDX is a list of samples that should be plotted. The default, when smplDX is NA, is to assume all samples should be utilized. The smplDX should be a numeric list that corresponds to the columns in the GGVobj's matrix of heatmap values, GGVobj$values$vls. The smplDX can also be used for ordering samples. The color of the samples may be controlled through the smp.color argument. This vector of colors should be equal to and in the original order of the values matrix. The colors will be re-ordered based on the sample index automatically.

Continuing with the 2.1 example:

```
goodDX=NA
smplDX=1:10
smp.color= c(rep(c("red", "blue", "purple", "green","yellow"), each=4), rep("pink", 2))
```

The above will use all spots and the first ten samples. The other argument break.num refers to the number of spots to include in subheatmaps before creating a new region for interactivity.

3.1.2 tiled heatmap options

If trackRegions, or known regions of interest, were indentified when making the GGVobj, layouts with tiled heatmaps will be generated for each region. There are a number of arguments that relate to these tiled image plots. The following will give brief descriptions of those arguments:

- tileNum : the number of tracks or tiles into which the spot IDs will be broken. If the dimension is too high to tile, the function will automatically reduce the number to an acceptable value.
- buffer : an additional number of spots to plot surrounding known regions. The known region is +/- this buffer.
- tiledMat : matrix indicating layout. This argument will be passed into the graphics package layout call as mat.Each value in the matrix must be '0' or a positive integer. If N is the largest positive integer in the matrix, then the integers 1,...,N-1 must also appear at least once in the matrix. '0' indicates region of no plotting. This may be left as NA, and a default

will be used. This is left as an argument in case the user finds the default plots too large or small based on customization. N is 3 if plot.call is NA, and 4 if plot.calls is specified. This matrix will be used only for the tiled heatmap plots.

- tiledMai.mat : n x 4 matrix of values to be passed in for each plots par mai. n will be 3 if plot.call is NA, and 4 if plot.calls is specified. The four columns represent the four different plot margins: bottom, left, top, right respectively. This matrix will be used only for the tiled heatmap plots.
- tiledMai.prc : logical indicating if mai mat values are percentages of original size or hard coded values. If mai.prc is T, it indicates percentage. This will be used only for the tiled heatmap plots.
- tiled.image.size : character indicating resize value of image,'width'x'height' tiled image plots. See initSplot of the sendplot library for more details.
- tiled.window.size : size of the html window for tiled image plots.see makeSplot of the sendplot library for more details.

Note: The arguments tiledMat, tiledMai.mat, and tiledMai.prc mention they are for tiled heatmap plots. When creating the GGV object, the mat, mai.mat, and mai.prc for arm and sub-arm images may be specified.
In the example, most of the defaults will be used:

```
tileNum=3
tiled.window.size = "1200x1100"
```

3.1.3 plotting options

The user has some options with file names and what files are created. The following are options:

- makeWinArms : controls whether or not the chromosome arm subplots are generated. If the user opts out of this option, only the chromosome arm and the tiled plots of known regions are generated.
- dir : the subdirectories have unchangeable names; the main directory that these subdirectories are located, however may be controlled through this argument. dir should be the complete path and name of the directory for which the directory structure should be created. **Note:** The dir argument should end with a backslash: /.
- fname.root : root name for the index and optional genomic plot created at the topmost level of makeGGV. The fname.root will be used as the base (E.G. fname.root='GGV' then the index file GGV.Index.html and the genome plot GGV.html are generated).

- overwriteSourcePlot : By default, an html file and a png file are generated. The user may opt to have a jpeg, tiff, or postscript file generated. The four options for this argument are "ps","png","jpeg", or "tiff". This argument may also be a character vector or any combination of the four file types. Please see the sendplot library's makeSplot function for more details on overwriteSourcePlot.

- header : May either be "v1","v2", or "v3". Determines which tooltip header will be in the html file. Please see the sendplot library's sp.header or makeSplot for more details on header.

- window.size : size of the html window for chromosome arm and sub chromosome arm plots. Please see the sendplot library's makeSplot function for more details

- image.size : character indicating resize value of image,'width'x'height' for chromosome arms and sub chromosome arm plots. Please see the sendplot library's makeSplot function for more details

The argument cleanDir is unique. The function produces output not needed by the user-intermediate steps for determining mappings. The user may clean the directory structure of all the un-needed output through the cleanDir argument. If TRUE, all the unnecessary plots will be deleted, leaving only the necessary files for viewing and interrogating data. This is an attempt to save space on user workspace.

The example will use all default settings.

3.1.4 updating plots and directories

The function checks to see if plots have already been generated. If the plots already exist they will not be regenerated unless an update is necessary. An update is necessary if the chromosome plot needs to be updated with new regions of interest. In this case the chromosome and all sub-chromosome plots will be overwritten. To update the trackRegion of a GGV object please see section 5.3.

Note: Files should only be deleted manually to be regenerated if: 1) new matrix values are being used; 2) the number of known regions specified by genomic location remains the same but different regions are actually used.

3.1.5 summary of code for makeGGV

Let's recap the code thus far and put it together with the makeGGV function call. Remember the GGVobj came from section 2.1:

```
goodDX=NA
smplDX=1:10
smp.color= c(rep(c("red", "blue", "purple", "green","yellow"), each=4), rep("pink", 2))
```

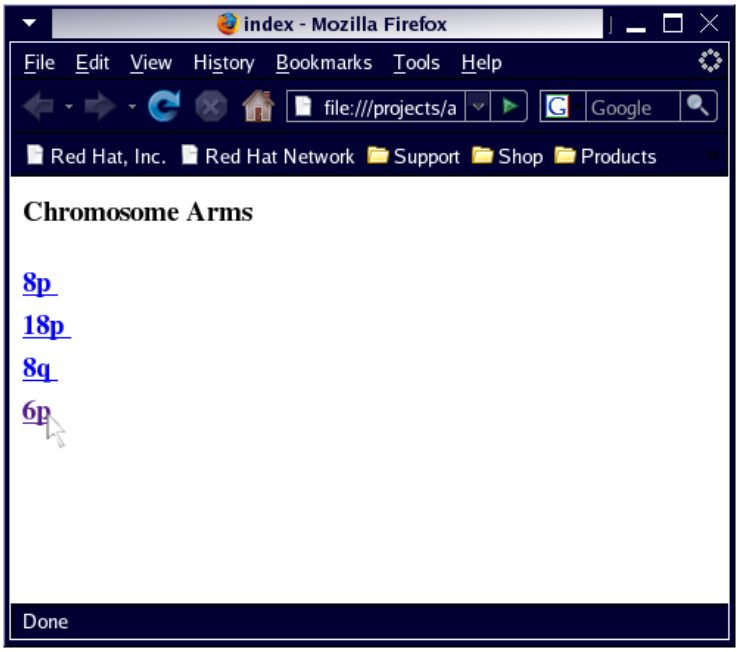

Figure 4: If we begin with the Index file, select a region of interest. The regions listed are determined by the user settings in chrArms, plot.dx, and known regions of interest when creating the GGVobj.

```
tileNum=3

makeGGV(GGV=GGV, goodDX=goodDX, smplDX=smplDX,
        smp.color=smp.color, tileNum=tileNum, tiled.window.size = tiled.window.size)
```

The following set of figures 4-8 take the user through the plots in Figure 3 in more detail, showing which objects in the figures are interactive in a web browser. Please note: only one tool-tip object will be displayed when interactive, the multiple interactive windows are only shown here as reference. We begin either with the Index file or a genomic plot.

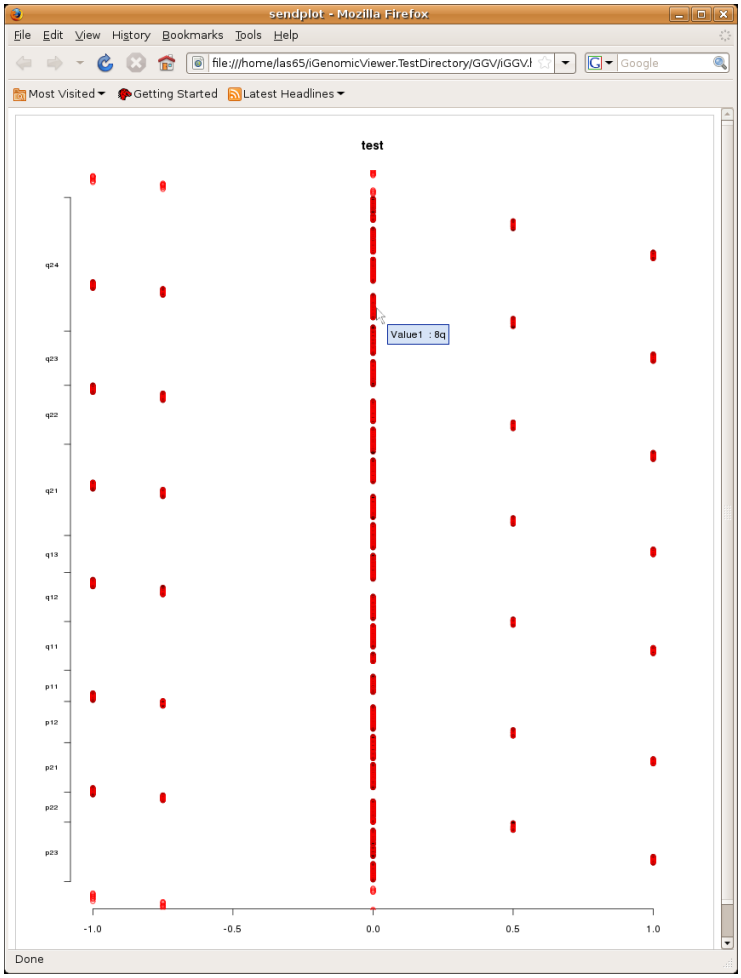

Figure 5: If we begin with the Genomic file, select a point of interest. The region depicted is dtermined by the plot.dx when creating a GGVobj. If no additional plot is given, only the index file is created.

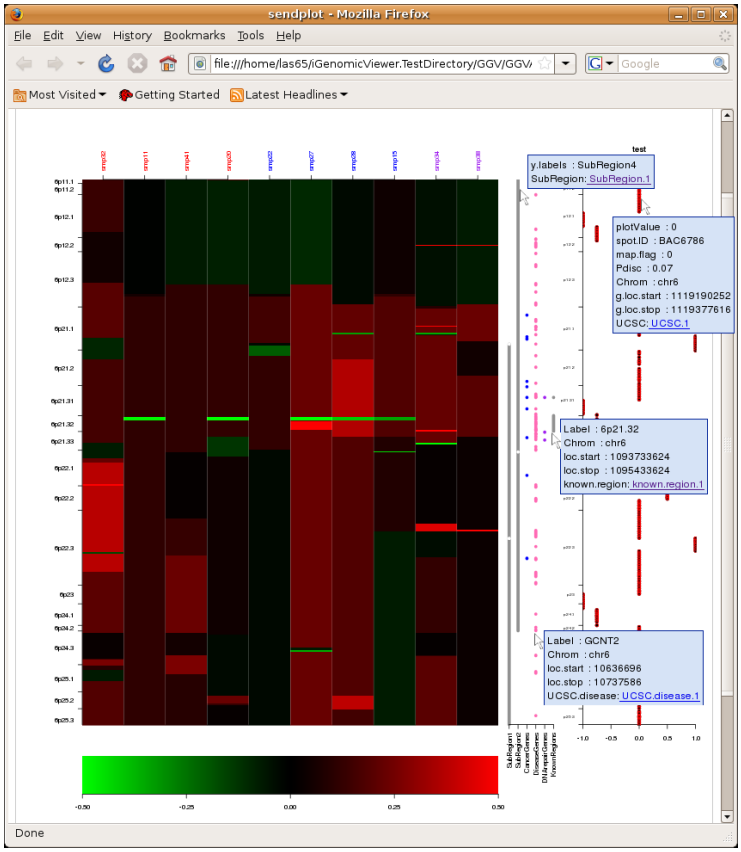

Figure 6: Shows next level, chromosome arm, using 6p. Notice the different areas that have interactivity. If the user clicks on the underlined hyperlinks in the tool-tip, a new plot or website will appear. The link SubRegion.1 will bring up another fully interactive heatmap of the region between the gray line selected. The link known.region.1 will bring up a tiled image map for the region between the gray line. The UCSC.disease will bring up the UCSC genome browser for the gene selected. The UCSC.1 link will bring up the UCSC genome browser equivalent to the spot location selected. Remember all data included in the tool-tip is customized by the user. Figure 7 shows the plot from the SubRegion link and Figure 8 shows the plot from the knownRegion link.

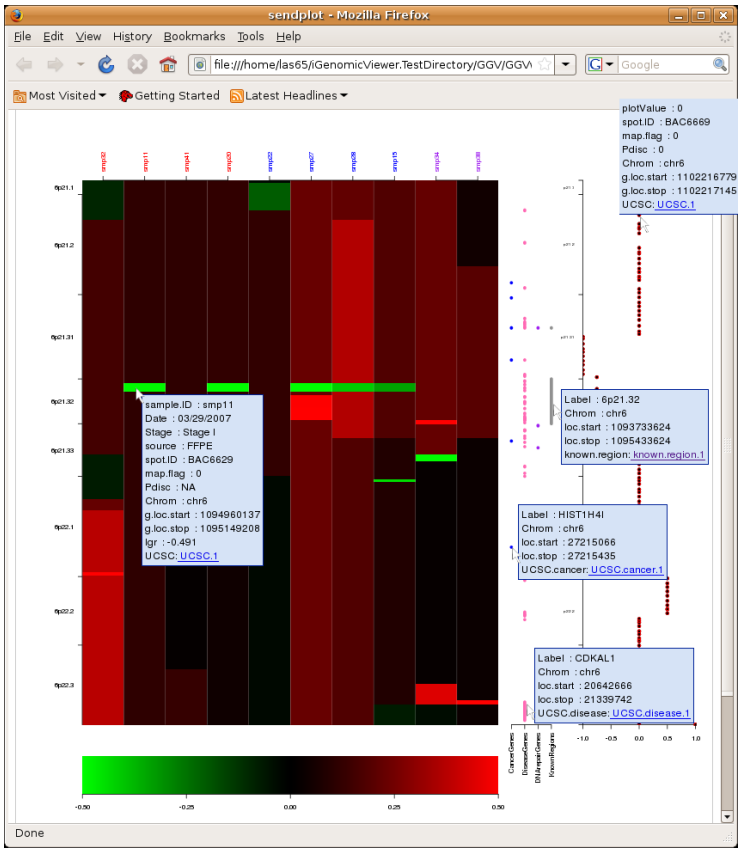

Figure 7: Shows next level, smaller viewer of chromosome arm 6p. Notice the different areas that have interactivity, and that now the heatmap is also interactive. If the user clicks on the underlined hyperlinks in the tool-tip, a new plot or website will appear. All tracks of the annotation plot are interactive. This example shows the interactivity of a Cancer gene, Disease Gene, and known region, the DNArepair track has the same functionalty. The link known.region.1 will bring up a tiled image map for the region between the gray line. The UCSC.disease and UCSC.cancer links will bring up the UCSC genome browser for the gene selected. The UCSC.1 link will bring up the UCSC genome browser equivalent to the spot location selected. Remember all data included in the tool-tip is customized by the user. Figure 8 shows the plot from the knownRegion link.

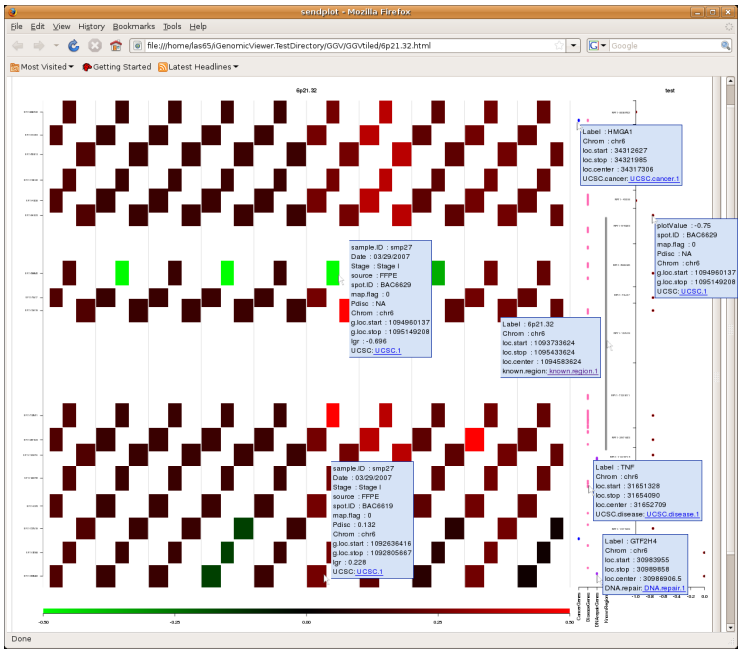

Figure 8: The lowest level depcits a tiled image plot. It is made for smaller regions to show spot overlaps and gaps. Notice the different areas that have interactivity. Each box in the track is interactive, therefore there are multiple tracks per samples as shown. All tracks of the annotation plot are interactive. This example shows interactivity for a gene of each annotation: cancer, disease, and dna repair. The known track region is also shown so the user can find information on the region displayed. The additional plot, if used, is also interactive. Remember all data included in the tool-tip is customized by the user.

3.2 iGGVtiled: plot a 'TIplot' object

The iGGVtiled function creates a panel of interactive plots which includes: 1) a tiled heatmap of the data with tool-tip display of sample and assay specific data which is customizable; 2) a set of customized annotation tracks; 3) optional plot which displays statistical values for the spot assays depicted in the heatmap. The tiled heatmap is useful for viewing the overlap and gaps in spot coverage. The following shows the function definition, note which arguments must be defined and which have default values:

```
iGGVtiled(TIplot,
          annObj,
          x.labels=NA,y.labels=NA,xy.labels=NA,
          x.links=NA,y.links=NA,xy.links=NA,asLinks=NA,
          x.images=NA, y.images=NA, xy.images=NA,
          mat=NA,mai.mat = NA,mai.prc=FALSE,
          plot.extras=NA,smpLines=TRUE,divCol="lightgrey",
          plot.call=NA,plot.vec=NA,lims = c(-0.5,0.5),
          annotation = NA,clrs=c("blue", "hotpink", "purple", "orange"),
          mapObj.columns = NA,
          fname.root="iGGV", dir="./",overwriteSourcePlot = NA,
          makeInteractive=TRUE,overrideInteractive=NA, header="v3",
          window.size = "800x1100",image.size= "800x1100",
          vrb=TRUE, ...)
```

3.2.1 specifying objects

The TIplot argument is a 'TIplot' object. Please see section 2.2 and appendix A.5 for more information on creating 'TIplot'. The example will assume that the object in secion 2.2 has been created. The TIplot object contains all necessary information for making a layout of plots which includes a tiled heatmap.
This function also requires use on an annotation object. Please see section 4.3 for more details on generating this object. The example will continue with the annotation object provided by the library.

```
data(annObj)
```

3.2.2 specifying tool-tip content and incorporating hyperlinks

The arguments x.labels, y.labels, xy.labels, x.links, y.links,xy.links, asLinks, x.images, y.images, xy.images and mapObj.columns work the exact same way as when used with the initGGV function with minor differences. Please see section 2.1.2. The data.frames and data matrices may be complete or already subset based on the sample index and bac index used when creating the TIplot object. **NOTE:** If the length of the sample index in the TIplot object is equal to the dimensions corresponding to those in x.labels, xy.labels, x.links, xy.links, x.images,

and xy.images the function will try and reorder the samples. If the sample index was reordering samples, and these matrices were subset, they should be taken out of the original data matrix in order. The function will reorder.

3.2.3 controlling plotting features

The arguments mat, mai.mat, mai.prc, plot.extras, smpLines, divCol, and lims function the same as in the initGGV function. Please see section 2.1.5.
An additional argument, overrideInteractive, controls which of the plots in the layout will be interactive. If NA, the default settings are used. This argument should be NA or the length of the number of plots in the layout: 3 if no additional statistical plot, 4 if there is an additional statistical plot. This argument turns off the tool-tip function for a plot. Plot 1 is the tiled heatmap, plot 2 is the legend for the heatmap, plot 3 is the annotation track, and plot 4 is the additional plot. By default overrideInteractive is either c(TRUE, FALSE, TRUE) or c(TRUE, FALSE, TRUE, TRUE). If, for instance, the user no longer wishes the annotation track to display tool-tip interactivity, overrideInteractive would become either c(TRUE, FALSE, FALSE) or c(TRUE, FALSE, FALSE, TRUE). The ... argument represents additional arguments for the sendplot library function makeImap that are not already set in the function call. Some possible options are arguments that alter tool-tip display or functionality are spot.radius, font.type, font.color, font.size, and bg.color. Please see the sendplot library function makeImap for further details. **Note:** the additional arguments will be used to set interactive points for all plots.

3.2.4 adding an additional [statistical] genomic plot

The plot.call argument is a character vector containing a plot call that will be evaluated. This plot is added to the right of the annoation tracks. If NA, no plot will be added to the display. This plot will have the x-value and any genomic specific data added to the display for the tiled heatmap. The argument plot.vec is the vector of x-values plotted in plot.call; this is needed to add the values to the interactive display. The plot.call and plot.vec should be over the range of y values [genomic spot IDs] that will be displayed in the tiled heatmap. The data, therefore, must already be subset based on the spot index.

For example, let the example side plot be the average of the values in the matrix. Recall TIplot was made over the spot index of 103 to 112:

```
spot.indx = 103:112
plot.vec = round(rowMeans(TIplot$vls$Z),3)
plot.call = "image(x=0:1,y=0:1,z=matrix(rep(NA,4),ncol=2),
                xlim=c(range(plot.vec,na.rm=T)),
                ylim=range(mapObj$mapping.info$g.loc.center[spot.indx],na.rm=T),
                zlim=c(0,1),axes=F,xlab='',ylab='');
                points(x=plot.vec,y=mapObj$mapping.info$g.loc.center[spot.indx],
                pch=3, cex=0.5, col='purple');axis(2);axis(1)"
```

Notice that the call is a character string that will be evaluated as multiple function calls separated by a semicolon. Arguments of type character within these calls are specified with a single quotation rather than the double quotations used originally, or vice versa (see col argument). Any variables used in arguments, such as spot.indx, should be in local memory before running the function to evaluate the iGGVtiled.

3.2.5 controlling annotation plotting

The arguments annotation and clrs function the same as when being used in the initGGV function. Brief recap: The annotation argument is a numeric corresponding to the order of the annotation information objects in the annObj. NA will display all. 0 will display none. Please see section 2.1.6 for more details.

3.2.6 plotting and output options

The following arguments control some of the plotting and output of the function:

- fname.root : base name to use for files created.
- dir : directory path to where files should be created. **Note:** The dir argument should end with a backslash: /.
- overwriteSourcePlot : By default, an html file and a png file are generated. The user may opt to have a jpeg, postscript or tiff file generated. The four options for this argument are "ps","png","tiff", or "jpeg". This argument may also be a character vector of any combination of the four. Please see the sendplot library's makeSplot function for more details on overwriteSourcePlot.
- makeInteractive : logical determining if an interactive html file should be created. If FALSE, only the static images will be generated. See makeSplot for more details.
- header : May either be "v1","v2", or "v3". Determines which tooltip header will be in the html file. Please see the sendplot library's sp.header or makeSplot for more details on header.
- image.size : character indicating resize value of image,'width'x'height'. Please see the sendplot library's makeSplot function for more details
- window.size : size of the html window. Please see the sendplot library's makeSplot function for more details.
- vrb : logical indicating if status messages should be printed.

3.2.7 summary code for iGGVtiled

Let's recap the code thus far and put it together with the iGGVtiled function call. Remember the TIplot object came from section 2.2. In this code random data is included for x.labels, y.labels, and xy.labels to show tool-tip functionality:

```
data(annObj)

spot.indx = 103:112
plot.vec = round(rowMeans(TIplot$vls$Z),3)
plot.call = "image(x=0:1,y=0:1,z=matrix(rep(NA,4),ncol=2),
                 xlim=c(range(plot.vec,na.rm=T)),
                 ylim=range(mapObj$mapping.info$g.loc.center[spot.indx],na.rm=T),
                 zlim=c(0,1),axes=F,xlab='',ylab='');
                 points(x=plot.vec,y=mapObj$mapping.info$g.loc.center[spot.indx],
                 pch=3, cex=0.5, col='purple');axis(2);axis(1)"



iGGVtiled(TIplot=TIplot,
          annObj=annObj,
          x.labels=as.data.frame(list(
                  sample.ID=paste("smp",1:TIplot$vls$nsmp,sep=""),
                  xla1=c("a","b","c","d","e","f","g","h","i","j"),
                  xla2=10:1)),
          y.labels=as.data.frame(list(
                  Spot.ID=paste("Spot",bacDX,sep=""))),
          xy.labels=list(lgr=round(Z,3)),
          plot.call=plot.call, plot.vec=plot.vec,
          mapObj.columns = c(2,3,7),
          fname.root="iGGVtiled")
```

The following Figure 8 shows a tiled Image and which objects in the figure are interactive in a web browser. Please note: only one tool-tip object will be displayed when interactive, the multiple interactive windows are only shown here as reference.

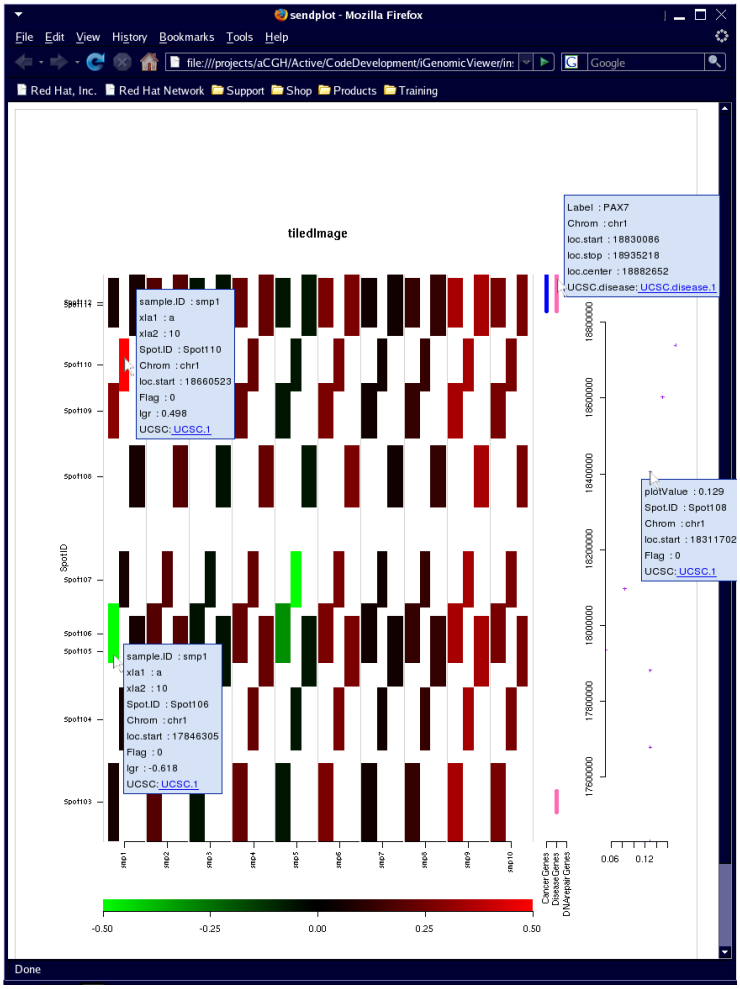

Figure 9: the tiled image view is made for smaller regions to show spot overlaps and gaps. Notice the different areas that have interactivity. Each box in the track is interactive, therefore there are multiple tracks per samples as shown. All tracks of the annotation plot are interactive. This example shows interactivity for a disease gene. An additional plot, if used, is also interactive. Remember all data included in the tool-tip is customized by the user.

3.3 iGGV: no object needed

The iGGV function creates a single interactive layout of plots. The user can interrogate a panel of plots which include: 1) a heat map of the data with tool-tip display of sample and assay specific data, all data displayed is user customized (e.g., assay values, sample IDs, hyperlinks to UCSC browser and sample specific images); 2) a set of interactive customized annotation tracks (e.g. display location of cancer, disease and DNA repair genes); 3) an optional plot which displays statistical values such as -log10 p-values or aberration frequencies for the spot assays depicted in the heatmap.
The following shows the function definition, note which arguments must be defined and which have default values:

```
iGGV(vls,
     mapObj,
     annObj,
     x.labels=NA,y.labels=NA,xy.labels=NA,
     x.links=NA,y.links=NA,xy.links=NA,asLinks=NA,
     x.images=NA, y.images=NA, xy.images=NA,
     mat=NA,maxLabels=25,mai.mat = NA,mai.prc=FALSE,
     plot.x.index=NA,smp.color = NA,
     plot.y.index=NA,goodDX=NA,
     genomic.start=NA,genomic.stop=NA,
     genomic.region=NA,region.type="chrom",
     plot.extras=NA,smpLines=TRUE,divCol="lightgrey",
     plot.call=NA,plot.vec=NA,lims = c(-0.5,0.5),
     annotation = NA,clrs=c("blue", "hotpink", "purple", "orange"),
     mapObj.columns = NA,fname.root="iGGV",dir="./",
     overwriteSourcePlot = NA,makeInteractive=TRUE,overrideInteractive=NA,
     header="v3",window.size = "800x1100",image.size= "800x1100",...)
```

3.3.1 specifiying the heatmap matrix, mapping object, and annotation object

The vls argument of initGGV is a matrix of values to be used for the heatmap. The y, or first dimension, should correspond to genomic locations. This length should be equivalent to the mapObj's number of spot.IDs. The vls matrix therefore directly corresponds to the mapping object. Please see section 4.2 for more details on the mapping object. The user will be given an opportunity to subset the spot.ID's when executing the plots; the user should NOT attempt to subset the y axis/genomic locations at this step. The x, or second dimension, corresponds to samples.
This function assumes that a mapping object and annotation object have already been created. Please see sections 4.2 and 4.3 for more details of generating these object. The function provides default objects which will be used.

```
vls = round(mat, 3)
data(mapping.info)
mapObj = mapping.info
data(annObj)
```

3.3.2 specifiying the tool-tip content and incorporating hyperlinks

The arguments x.labels, y.labels, xy.labels, x.links, y.links, xy.links, asLinks, x.images, y.images, xy.images, and mapObj.columns function the same as in section 2.1.2. Please see this section for more details. Revisiting the code from section 2.1.2:

```
x.labels=x.lbls
xy.labels = list(lgr=vls)

y.lbls$Pdisc = round(y.lbls$Pdisc,3)
y.labels = y.lbls
mapObj.columns = c(2,8,10)
```

3.3.3 subsetting data

There are three ways to indicate y values that should be plotted. They may be specified directly through the plot.y.index, a numeric vector which corresponds to the ordering in the mapping object. They may be determined by giving a genomic starting and ending location, genomic.start and genomic.stop respectively. Both starting and ending locations must be given if this option is utilized. The genomic locations should be the genomic location with respect to the entire genome, not within a chromosome. If locations within chromosome are known, please see additional function convertCloc in section 6.1. Finally, they may be specified by listing a single specific region to be plotted with genomic.region. If this option is used, the user must also indicate what type of region is listed in the region.type argument. The four options for this argument are chrom, arm, broad.band, fine.band. The region given should match up to a region in the mapping object.

For example, the following would plot arm 11q:

```
genomic.region="11q"
region.type="arm"
```

As data is preprocessed, it may becomes apparent that some spots may be 'faulty' or have resulted in 'bad quality' data. If data is not trusted for certain spots it is possible to remove them. This is accomplished through goodDX. The argument goodDX is a numeric list of acceptable y values with respect to the mapObj$mapping.info object. Any spot that is not listed in goodDX will be removed and not plotted on any of the plots. The default, when goodDX is NA, assumes all spots should be utilized.

It is also possible to specify a select group of samples to plot. The argument plot.x.index is a list of samples that should be plotted. The default, when plot.x.index is NA, assumes all samples should be utilized. The plot.x.index should be a numeric list that corresponds to the columns in the vls matrix. The plot.x.index can also be used for ordering samples.

3.3.4 plotting options

The following arugments will be mentioned briefly. They help control some of the plotting features. If the user does not specify these argument, default settings will be used.

- maxLabels : maximum number of labels to appear on the heatmap y axis. Based on this number, the function will automatically determine if arms, broad.band, fine.bands, or individual spot.ID's should appear for the y axis.

- mat : matrix indicating layout. This argument will be passed into the graphics package layout call as mat. Each value in the matrix must be '0' or a positive integer. If N is the largest positive integer in the matrix, then the integers 1,...,N-1 must also appear at least once in the matrix. '0' indicates region of no plotting. This may be left as NA, and a default will be used. This is left as an argument in case the user finds the default plots too large or small based on customization. N is 3 if plot.call is NA, and 4 if plot.calls is specified.

- mai.mat : n x 4 matrix of values to be passed in for each plots par mai. n will be 3 if plot.call is NA, and 4 if plot.calls is specified. The four columns represent the four different plot margins: bottom, left, top, right respectively.

- mai.prc : logical indicating if mai mat values are percentages of original size or hard coded values. If mai.prc is T, it indicates percentage.

- plot.extras : list of length equal to the number of plots: 3 if plot.call is NA, 4 if plot.call is specified. This object is a list of lists. The sublists contain any additional plotting calls that should be executed for the plot. Each entry must be a character vector. If no additional plotting is equired, NA should be used.

- smpLines : logical indicating if vertical lines should be added between each sample of the heatmap

- divCol : If smpLines, the color of the dividing lines

- lims : Lower and upper limit for vls. Any value above of below will be changed to max and min value respectively.

- smp.color : Colors for the x-axis samples. This vector of colors should be equal to and in the original order of the values matrix. The colors will be re-ordered based on the sample index automatically.

- overrideInteractive : controls which of the plots in the layout will be interactive. If NA, the default settings are used. This argument should be NA or the length of the number of plots in the layout: 3 if no additional statistical plot, 4 if there is an additional statistical plot. This argument turns off the tool-tip function for a plot. Plot 1 is the heatmap, plot 2 is the legend for the heatmap, plot 3 is the annotation track, and plot 4 is the additional plot. By default, overrideInteractive is either c(TRUE, FALSE, TRUE) or c(TRUE, FALSE, TRUE, TRUE). If, for instance, the user no longer wishes the annotation track to display tool-tip interactivity, overrideInteractive would become either c(TRUE, FALSE, FALSE) or c(TRUE, FALSE, FALSE, TRUE).

- ... : additional arguments for the sendplot library function makeImap that are not already set in the function call. Some possible options are arguments that alter tool-tip display or functionality such as spot.radius, font.type, font.color, font.size, and bg.color. Please see the sendplot library function makeImap for further details. **Note:** the additional arguments will be used to set interactive points for all plots

3.3.5 adding an additional [statistical] genomic plot

The plot.call argument is a character vector containing a plot call that will be evaluated. This plot is added to the right of the annoation tracks. If NA, no plot will be added to the display. This plot will have the x-value and any genomic specific data added to the display for the heatmap. The argument plot.vec is the vector of x-values plotted in plot.call; this is needed to add the values to the interactive display. The plot.call and plot.vec should be over the range of y values [genomic spot IDs] that will be displayed in the heatmap. The data, therefore, must already be subset based on the spot index.

For this example, no side plot will be added

```
plot.call=NA
plot.vec=NA
```

3.3.6 controlling annotation plotting

The arguments annotation and clrs function the same as when being used in the initGGV function. Brief recap: The annotation argument is a numeric corresponding to the order of the annotation information objects in the annObj. NA will display all. 0 will display none. Please see section 2.1.6 for more details.

3.3.7 plotting and output options

The following arguments control some of the plotting and output of the function:

- fname.root : Base name to use for files created
- dir : directory path to where files should be created. **Note:** The dir argument should end with a backslash: /.
- overwriteSourcePlot : By default, an html file and a png file are generated. The user may opt to have a jpeg, tiff, or postscript file generated. The four options for this argument are "ps","png","tiff", or "jpeg". Please see the sendplot library's makeSplot function for more details on overwriteSourcePlot.
- makeInteractive : logical, determining if an an interactive html file should be created. If FALSE, only the static images will be generated. See makeSplot for more details
- header : May either be "v1","v2", or "v3". Determines which tooltip header will be in the html file. Please see the sendplot library's sp.header or makeSplot for more details on header.
- image.size : character indicating resize value of image,'width'x'height'.Please see the sendplot library's makeSplot function for more details
- window.size : size of the html window. Please see the sendplot library's makeSplot function for more details

3.3.8 summary of code for iGGV

Let's recap the code thus far and put it together with the iGGV function call.

```
vls = round(mat, 3)
data(mapping.info)
mapObj = mapping.info
data(annObj)

x.labels=x.lbls
xy.labels = list(lgr=vls)
y.lbls$Pdisc = round(y.lbls$Pdisc,3)
y.labels = y.lbls
mapObj.columns = c(2,8,10)

genomic.region="11q"
region.type="arm"

iGGV(vls = vls,
     mapObj=mapObj,
```

```
    annObj=annObj,
    x.labels=x.labels,
    y.labels=y.labels,
    xy.labels=xy.labels,
    genomic.region=genomic.region,
    region.type=region.type,
    mapObj.columns =mapObj.columns)


}
```

3.4 makeTiled: a static plot

The makeTiled function creates a single, static tiled image heatmap. The following shows the function definition, note which arguments must be defined and which have default values:

```
makeTiled(TIplot,
          smpDiv=TRUE,
          divCol="lightgrey")
```

The argument TIplot is a TIplot object. Please see section 2.2 and appendix A.5 for more details. The example will continue assuming the object in 2.2.5 has been created.
The smpDiv argument is a logical indicating if vertical lines should be added between each sample of the heatmap. The color of the lines is controlled with divCol.
The above code will generate a single static tiled image heatmap.

4 Mapping and Annotation

The ability to create mapping and annotation objects allows for complete platform independent use of the functions in the iGenomicViewer library. The following sections will explain what minimal information is needed and how to build required objects. All the following sections utilize files which are provided through the writeExFiles function. See section 5.4 for more details.
A temporary directory is created to store output files:

```
> writeExFiles()
```

4.1 Band Information Object

The 'bandinfo', or band information, object contains genomic location information for chromosome, arms, broad bands, and fine bands. Based on a file

which contains columns for chromosome, start location, stop location, and band information, the function makeBandInfo will create useful data frames of starting and stopping locations for each level. The starting and stopping locations in this file should be within chromosome - not across the entire genome. See section 6.1 for more details on converting chromosome genomic location and genomic location. The band information column should not include chromosome (i.e. 'p36.11', 'q42.3'). The following shows the function definition, note which arguments must be defined and which have default values:

```
makeBandInfo(file,
             chrom.levels,
             file.sep="\t",
             autosomes=1:22,
             X.chrom = 23,
             Y.chrom = 24,
             chr.dx = 1,
             band.dx = 4,
             start.dx = 2,
             stop.dx = 3,
             returnVl=TRUE,
             saveFile=FALSE,
             saveName = "BandInfo.RData",
             ...)
```

The first task is to specify the file that should be used for determining information. The package provides the file cytoband.txt. Cytoband.txt is a tab delimited text file with columns for chromosome, start location, stop location, and band. The function reads this file through the R base package's read.table function. The separation character for the file should be given in the file.sep argument. Any additional arguments that should be passed into the read.table function may be included; this is where the ... arguments are utilized. The example file includes a header line, therefore header=TRUE should be included in the list of arguments.

Next information about the chromosome level should be provided. The argument chrom.level is a vector indicating how the chrom column in the file is represented (i.e chr1, chrom1, 1). The file provided uses chr1, chr2, ... chrX, chrY. This argument will be used to factor the chromosome column. It is also important to specify how many autosomes by using the autosome argument, and which are sex chromsomes by using the X.chrom and Y.chrom arguments. This allows for the use of different species; the default is for homo sapiens.

The arguments chr.dx, band.dx, start.dx, and stop.dx are numeric indications for which column in the file corresponds to chromosome information, band information, genomic starting location, and genomic stopping location; the minimal information needed to create a band information object. The defaults are set

up to read the file provided with the function.

Lastly, returnVl, saveFile, and saveName determine if the created object should be returned or saved. If returnVl is true, the object is returned. If saveFile is true, the object is saved as an R data object. The argument saveName is the complete path and name for the R data object.

Using the example data:

```
band.info = makeBandInfo(file="cytoBand.txt",
                         chrom.levels=c("chr1","chr2","chr3","chr4","chr5","chr6",
                                        "chr7","chr8","chr9","chr10","chr11","chr12",
                                        "chr13","chr14","chr15","chr16","chr17",
                                        "chr18","chr19","chr20","chr21","chr22",
                                        "chrX","chrY"),
                         file.sep="\t",
                         returnVl=TRUE,
                         header=TRUE)
```

For additional information on the class structure and provided objects see also appendix A.2 and B.2.

4.2 Mapping Object

The 'mapobj', or mapping object, contains all mapping information which includes but is not limited to: spotIDs, chromosome locations, and genomic locations. The mapping object is unique to the experimental platform; this object allows for use of any genomic experiment data within the package. The following shows the function definition, note which arguments must be defined and which have default values:

```
mappingObj(file,
           spot.ID,
           chrom,
           chrom.levels,
           loc=NA,
           loc.start=NA,
           loc.stop=NA,
           file.sep="\t",
           additional=NA,
           names.additional=NA,
           links=NA,
           names.links=NA,
           images=NA,
           names.images=NA,
```

```
band.info = NA,
returnVl = TRUE,
saveFile = FALSE,
saveName="MapObj.RData",
...)
```

This function operates off a file that should minimally contain spot.IDs, chromosome, and genomic location. The file name should be given by the file argument. The package includes example file HB19Kv2.HG18.txt which is a tab-delimited text file with columns for BAC name, chromosome, start location, stop location, central location, genomic location, band, mapped by, flag, and weblink to UCSC Genome Browser. The function reads this file through the R base package's read.table function. The separation character for the file should be given in the file.sep argument. Any additional arguments that should be passed into the read.table function may be included; this is where the ... arguments are utilized. The example file includes a header line, therefore header=TRUE should be included in the list of arguments.

The spot.ID and chromosome arguments are indications for which column in the file correspond to the spot.ID and chromosome information. They may be numeric, or if a header indicating column names is present in the file, a character. The argument chrom.levels is a vector indicating how the chrom column in the file is represented (i.e chr1, chrom1, 1). The file provided uses chr1, chr2, ... chrX, chrY.

There are two ways to indicate genomic location for each spot. The recommended way is to provide both start and stop locations through the loc.start and loc.stop respectively. The arguments should be a numeric, or if a header indicating column names is present in the file, a character. If loc.start and loc.stop are used loc should be NA. Alternatively, one may provide a central, midpoint location through loc. Again, it may be a numeric or character indicating the corresponding column in the file. If loc is used, loc.start and loc.stop should be NA. **Note:** All genomic locations should be within the chromosome not across the genome. See section 6.1 for more details on converting chromosome genomic location and genomic location.

There may be any number of additional columns in the file that the user wishes to include, perhaps a column on spot quality or how the spots were mapped. Additional columms may be included with the additional argument. This may be a numeric or character vector of corresponding columns in the file. The names.additional is an optional vector to specify names for the additional columns included; this is particularly useful when the file does not contain a header line. If additional=0 then no additional columns are included. If additional=NA, all additional columns in the file are included.

It is also possible to include hyperlinks for the data. Our example data, for

example, includes links to the UCSC browser. Links may be included in two ways through the links argument. If links are in the file given, links is a numeric or character vector of corresponding columns in the file. The argument links may also be a data.frame or matrix. If this option is utilized, the function assumes the table is in the correct order with respect to the orginal file. The argument names.links is an optional vector to specify names for the links included; this is particularly useful when the file does not contain a header line.

Images may also be included for the data. Images may be included in two ways through the images argument. If images are in the file given, images is a numeric or character vector of corresponding columns in the file. The argument images may also be a data.frame or matrix. If this option is utilized, the function assumes the table is in the correct order with respect to the orginal file. The argument names.images is an optional vector to specify names for the images included; this is particularly useful when the file does not contain a header line.

Lastly, a 'band.info' object must be included. See previous section, 4.1 on building this object. If no band.info object is specified (band.info=NA), the default band.info object provided with the package will be used. (See appendix B.2) The band.info object is used to organize and correctly plot and graph spot.IDs. It maps spot.IDs to chromosome, arm, broad bands and fine bands.

Lastly, returnVl, saveFile, and saveName determine if the created object should be returned or saved. If returnVl is true, the object is returned. If saveFile is true, the object is saved as an R data object. The argument saveName is the complete path and name for the R data object.

Using the example data:

```
data(Band.Info)

mapping.info = mappingObj(file="HB19Kv2.HG18.txt",
                          spot.ID="Clone", chrom="Chromosome",
                          chrom.levels=c("chr1","chr2","chr3","chr4","chr5","chr6",
                                         "chr7","chr8","chr9","chr10","chr11","chr12",
                                         "chr13","chr14","chr15","chr16","chr17",
                                         "chr18","chr19","chr20","chr21","chr22",
                                         "chrX","chrY"),
                          loc.start="start", loc.stop="Stop",
                          file.sep="\t", header=TRUE,
                          additional=c("Mapped.by", "Flag"),
                          links=10, names.links="UCSC",
                          band.info=band.info,
                          returnVl=TRUE )
```

See appendix A.4, B.7, and B.9 for more information on class structure and provided files or objects. Also see section 5 on wrappers to bioconductor objects for other mapping functions.

4.3 Annotation Object

The annotation object consists of individual 'anninfo' objects. The anninfo object contains all information for a given annoation set. This includes but is not limited to: names, chromosome locations,and genomic locations. This allows the user to include as many annotation sets as they choose. The package provides three possible annotation sets for known cancer genes, known disease genes, and known DNA repair genes. These will be used to illustrate the building of an annotation object. See appendix B.3, B.5, and B.6 for information on provided annotation files.

4.3.1 makeAnnotation: 'anninfo' object

The makeAnnotation function makes 'anninfo' objects. The following shows the function definition, note which arguments must be defined and which have default values:

```
makeAnnotation(file,
               label,
               chrom,
               chrom.levels,
               band.info=NA,
               loc=NA,
               loc.start=NA,
               loc.stop=NA,
               file.sep="\t",
               additional=NA,
               names.additional = NA,
               links=NA,
               names.links=NA,
               images=NA,
               names.images=NA,
               returnVl = TRUE,
               saveVl = FALSE,
               saveName="Annotation.RData",
               ...)
```

The annotation file must minimally contain columns for name, chromosome, and genomic location. The file name should be given by the file argument. The package includes example files CancerGenes.txt, DiseaseGenes.txt, and DNArepairgenes.txt. All are tab-delimited text files with columns for gene name, chromosome, start location, end location, and weblink to UCSC Genome Browser.

The function reads a file through the R base package's read.table function. The separation character for the file should be given in the file.sep argument.. Any additional arguments that should be passed into the read.table function may be included; this is where the ... arguments are utilized. The example files include a header line, therefore header=TRUE should be included in the list of arguments.

The label and chrom arguments are indications for which columns in the file correspond to the region label and chromosome information. They may be numeric, or, if a header indicating column names is present in the file, a character. The argument chrom.levels is a vector indicating how the chrom column in the file is represented (i.e chr1, chrom1, 1). The file provided uses chr1, chr2, ... chrX, chrY.

There are two ways to indicate genomic location for each spot. The recommended way is to provide both start and stop locations through loc.start and loc.stop resepectively. The arguments should be a numeric, or if a header indicating column names is present in the file, a character. If loc.start and loc.stop are used, loc should be NA. Alternatively, a central, midpoint location through loc may be used. Again, it may be numeric or character indicating the corresponding column in the file. If loc is used, loc.start and loc.stop should be NA. **Note:** All genomic locations should be within chromosome not across the genome. See section 6.1 for more details on converting chromosome genomic location and genomic location.

There may be any number of additional columns in the file that the user wishes to include. Additional columms may be included with the additional argument. This may be a numeric or character vector of corresponding columns in the file. The names.additional is an optional vector to specify names for the additional columns included; this is particularly useful when the file does not contain a header line. If additional=0 then no additional columns are included. If additional=NA, all additional columns in the file are included.

It is also possible to include hyperlinks for the data. Our example data, for example, includes links to the UCSC browser. Links may be included in two ways through the links argument. If links are in the file given, links is a numeric or character vector of corresponding columns in the file. The argument links may also be a data.frame or matrix. If this option is utilized, the function assumes the table is in the correct order with respect to the orginal file. The argument names.links is an optional vector to specify names for the links included; this is particularly useful when the file does not contain a header line.

Images may also be included for the data. Images may be included in two ways through the images argument. If images are in the file given, images is a numeric or character vector of corresponding columns in the file. The argument images may also be a data.frame or matrix. If this option is utilized, the function assumes the table is in the correct order with respect to the orginal file. The

argument names.images is an optional vector to specify names for the images included; this is particularly useful when the file does not contain a header line.

Lastly, a 'band.info' object must be included. See section, 4.1 on building this object. If no band.info object is specified (band.info=NA), the default band.info object provided with the package will be used. (See appendix B.2) The band.info object is used to organize and correctly plot and graph annotation. It maps annotation to chromosome, arm, broad bands and fine bands.

Lastly, returnVl, saveFile, and saveName determine if the created object should be returned or saved. If returnVl is true, the object is returned. If saveFile is true, the object is saved as an R data object. The argument saveName is the complete path and name for the R data object.

Using the example data:

```
data(Band.Info)

# makes anninfo object for cancerGenes
annotation1 = makeAnnotation(file="CancerGenes.txt",
                             file.sep="\t", header=TRUE,
                             label=2, chrom=3,
                             chrom.levels=c("chr1","chr2","chr3","chr4","chr5","chr6",
                                            "chr7","chr8","chr9","chr10","chr11",
                                            "chr12","chr13","chr14","chr15","chr16",
                                            "chr17","chr18","chr19","chr20","chr21",
                                            "chr22","chrX","chrY"),
                             band.info=band.info,
                             loc=NA, loc.start=4, loc.stop=5,
                             additional=0,  links=6)


# makes anninfo object for DiseaseGenes
annotation2 = makeAnnotation(file="DiseaseGenes.txt",
                             file.sep="\t", header=TRUE,
                             label=2, chrom=3,
                             chrom.levels=c("chr1","chr2","chr3","chr4","chr5","chr6",
                                            "chr7","chr8","chr9","chr10","chr11",
                                            "chr12","chr13","chr14","chr15","chr16",
                                            "chr17","chr18","chr19","chr20","chr21",
                                            "chr22","chrX","chrY"),
                             band.info=band.info,
                             loc=NA, loc.start=4, loc.stop=5,
                             additional=0,  links=7)
```

```
# makes anninfo object for DNArepairGenes
annotation3 = makeAnnotation(file="DNArepairgenes.txt",
                             file.sep="\t", header=TRUE,
                             label=1, chrom=2,
                             chrom.levels=c("chr1","chr2","chr3","chr4","chr5","chr6",
                                            "chr7","chr8","chr9","chr10","chr11",
                                            "chr12","chr13","chr14","chr15","chr16",
                                            "chr17","chr18","chr19","chr20","chr21",
                                            "chr22","chrX","chrY"),
                             band.info=band.info,
                             loc=NA, loc.start=3, loc.stop=4,
                             additional=0,links=5)
```

4.3.2 annotationObj: initialize and add to annotation object

Now that the 'anninfo' objects have been created, an annotation object may be initialized and populated. The annotationObj is a larger object containing all individual annotation information desired for a genomic mapping. These annotations will be represented in a track alongside of the main heatmap of the iGGV function. Each individual annotation information object must be added separately to the main annotationObj. The following shows the function definition, note which arguments must be defined and which have default values:

```
annotationObj(annotation,
              annotationObj = NA,
              obj.name = NA,
              returnVl = TRUE,
              saveVl = FALSE,
              saveName="AnnotationObj.RData")
```

The annotation argument is the 'anninfo' object that should be added to the annotation object. The obj.name argument is the name that should be given to this annotation set provided by the 'anninfo' object. The argument annotationObj, is the annotation object that the 'anninfo' object should be added to. If annotationObj is NA, the function initializes a new annotation object. Each 'anninfo' object must be added separately to the annotation object.

The arguments returnVl, saveFile, and saveName determine if the created object should be returned or saved. If returnVl is true, the object is returned. If saveFile is true, the object is saved as an R data object. The argument saveName is the complete path and name for the R data object.

Using the example data and anninfo objects created in the previous section 4.3.1:

```
# initalizes annotation object
annObj = annotationObj(annotation1, obj.name="CancerGenes")

# adds additional anninfo objects to created annotation object, annObj
annObj = annotationObj(annotation2, annotationObj=annObj, obj.name="DiseaseGenes")
annObj = annotationObj(annotation3, annotationObj=annObj, obj.name="DNArepairGenes")
```

See appendix A.1 and B.1 for more information on annotation object structure and provided objects with package.

5 Wrappers to Bioconductor Objects

There are three other functions to creating a mapping object not listed in section 4: mappingObjDF, mappingObjADF and mappingObjMarray. The mappingObjDF and mappingObjADF take in an object of the class data.frame and annotatedDataFrame to create a mapping object. The mappingObjMarray will take an object of the class marrayRaw, marrayNorm, or marrayInfo. The inclusion of these functions allows for easy compatiblity with a variety of different formats. The rest of these functions' arguments behave as the generic mappingObj function in section 4.2. Please see this section for further details. The additional argument base.chrm and reg.exp are used instead of chrom.levels. The argument base.chrm should be a character vector listing all prefix or suffix that should be removed from the chromosome in order to have a numeric. For example, "chrm10b", the base.chrm would be c("chrm","b"). The reg.exp argument indicates if the entry in base.chrm should be directly matched to the chromosomes or if perl regular expression matching should be used; an entry of FALSE means the argument will be matched exactly.

6 Additional Functions

The following sections are additional functions provided by the package that the user may find useful.

6.1 convertLoc - converting genomic location

Sometimes it is necessary to have genomic location with respect to chromosome. At other times, it may be useful to have the same locations but with respect to the entire genome. The functions convertGloc and convertCloc are convenience functions for converting back and forth between within chromosome location and across the genome location.

6.1.1 convertGloc

The convertGloc function takes genomic locations across the entire genome and returns locations with respect to chromosome. The following shows the function

definition, note which arguments must be defined and which have default values:

```
convertGloc(x,
            chr,
            row=TRUE,
            bandobj=NA)
```

The argument x is the values to be converted. It is either a numeric vector or numeric matrix. The numeric values should correspond to genomic locations across the entire genome. **Note:** X is numeric only.

The argument chr indicates which chromosome the value is on. If x is a vector, chr should be a vector of equal length, or a single value indicating all values of x on the same chromosome. If x is a matrix, the values may be applied by row or by column. If by row, chr should be equal to dim(x)[1] and if by column, chr should be equal to dim(x)[2]. If x is a matrix, the row argument indicates if the chr values are applied by row. A single value may be used if all values are on the same chromosome. **Note:** chr is numeric only, therefore if sex chromosomes are included they should be a numeric also not 'X' or 'Y' (i.e. for homo sapiens any X should be 23 and Y 24).

The offset is calculated through the band.info object. See appendix A.2 and B.2 for more information. If a bandobj is not specified the package's default bandinfo object is used.

6.1.2 convertCloc

The convertCloc function takes genomic locations within chromosome and returns locations with respect to the entire genome.

```
convertCloc(x,
            chr,
            row=TRUE,
            bandobj=NA)
```

The arguments of convertCloc are the same as convertGloc with one exception. The values in x should correspond to all genomic locations with respect to genomic locations across the entire genome.

6.1.3 example convert code

The following is a mock example to show how offset is applied:

```
> x = matrix(0,24,3)
> x[,2] = 1
> x[,3] = 2
> chr = 1:24
>
```

The origional x matrix is:

```
> x
```

```
      [,1] [,2] [,3]
 [1,]    0    1    2
 [2,]    0    1    2
 [3,]    0    1    2
 [4,]    0    1    2
 [5,]    0    1    2
 [6,]    0    1    2
 [7,]    0    1    2
 [8,]    0    1    2
 [9,]    0    1    2
[10,]    0    1    2
[11,]    0    1    2
[12,]    0    1    2
[13,]    0    1    2
[14,]    0    1    2
[15,]    0    1    2
[16,]    0    1    2
[17,]    0    1    2
[18,]    0    1    2
[19,]    0    1    2
[20,]    0    1    2
[21,]    0    1    2
[22,]    0    1    2
[23,]    0    1    2
[24,]    0    1    2
```

Now convert genomic location to within chromosome location:

```
> newx = convertGloc(x=x, chr=chr, row=TRUE, bandobj=NA)
```

This is x after conversion:

```
> newx
```

```
             [,1]        [,2]        [,3]
 [1,]           0           1           2
 [2,]  -247249719  -247249718  -247249717
 [3,]  -490200868  -490200867  -490200866
 [4,]  -689702695  -689702694  -689702693
 [5,]  -880975758  -880975757  -880975756
 [6,] -1061833624 -1061833623 -1061833622
 [7,] -1232733616 -1232733615 -1232733614
 [8,] -1391555040 -1391555039 -1391555038
```

```
 [9,] -1537829866 -1537829865 -1537829864
[10,] -1678103118 -1678103117 -1678103116
[11,] -1813477855 -1813477854 -1813477853
[12,] -1947930239 -1947930238 -1947930237
[13,] -2080279773 -2080279772 -2080279771
[14,] -2194422753 -2194422752 -2194422751
[15,] -2300791338 -2300791337 -2300791336
[16,] -2401130253 -2401130252 -2401130251
[17,] -2489957507 -2489957506 -2489957505
[18,] -2568732249 -2568732248 -2568732247
[19,] -2644849402 -2644849401 -2644849400
[20,] -2708661053 -2708661052 -2708661051
[21,] -2771097017 -2771097016 -2771097015
[22,] -2818041340 -2818041339 -2818041338
[23,] -2867732772 -2867732771 -2867732770
[24,] -3022646526 -3022646525 -3022646524
```

Now convert the new matrix back to the original, across the genome:

```
> returnx = convertCloc(x=newx, chr=chr, row=TRUE, bandobj=NA)
```

This is newx converted back to original:

```
> returnx
```

```
      [,1] [,2] [,3]
 [1,]    0    1    2
 [2,]    0    1    2
 [3,]    0    1    2
 [4,]    0    1    2
 [5,]    0    1    2
 [6,]    0    1    2
 [7,]    0    1    2
 [8,]    0    1    2
 [9,]    0    1    2
[10,]    0    1    2
[11,]    0    1    2
[12,]    0    1    2
[13,]    0    1    2
[14,]    0    1    2
[15,]    0    1    2
[16,]    0    1    2
[17,]    0    1    2
[18,]    0    1    2
[19,]    0    1    2
[20,]    0    1    2
[21,]    0    1    2
```

```
[22,]    0    1    2
[23,]    0    1    2
[24,]    0    1    2
```

Now using real data. Consider the example dataset HB19Kv2.HG18. The column start is stating genomic locations with respect to the chromosomes listed in the column chromosome. To convert these locations to across genome the following would be used:

```
data(HB19Kv2.HG18)

chr = as.character(HB19Kv2.HG18$Chromosome)
# remove chr
chr = sub(pattern="chr", replacement="", chr)
# remove X and Y
chr[which(chr == "X")] = 23
chr[which(chr == "Y")] = 24
# change to numeric
chr = as.numeric(chr)

genomicLoc = convertCloc(HB19Kv2.HG18$start, chr)
```

The argument x can be a vector or a matrix. The function will return the same format as the input. The above used a vector. If we did the same conversion, but used both starting and stopping genomic location, the retuned object genomicLoc would be a matrix or data.frame instead of a vector.

```
data(HB19Kv2.HG18)

chr = as.character(HB19Kv2.HG18$Chromosome)
# remove chr
chr = sub(pattern="chr", replacement="", chr)
# remove X and Y
chr[which(chr == "X")] = 23
chr[which(chr == "Y")] = 24
# change to numeric
chr = as.numeric(chr)

glocs = HB19Kv2.HG18[,c(3,4)]

genomicLoc = convertCloc(glocs, chr)
```

6.2 makeTrack

This function creates a trackRegion object for use in the functions initGGV or updateGGV. Please see appendix A.6 for more on trackRegion structure. The following shows the function definition, note which arguments must be defined and which have default values:

```
makeTrack(Broad.Band=NA,
          Fine.Band=NA,
          genomicLoc=NA,
          geneName=NA)
```

The trackRegion object stores known regions of interest. The four ways to specify regions are the arguments Broad.Band, Fine.Band, genomicLoc and geneName.

Broad.Band and Fine.Band are character vectors containing names of broad band or fine band regions. They should match an entry in the broad.band or fine.band column in a mapObj$band.info data frames object.

The geneName character vector is associated with an annObj. This character vector should contain entries that match the labels in the columns of an annObj's annotation data.frame. The list may contain entries from any and all objects of the annObj.

The genomicLoc entry may either be a numeric vector or matrix. If it is a vector it assumes the entries are listed in order of start loc,stop loc, start loc, stop loc, etc. for given regions. If it is a matrix, it assumes the first column is start locations and the second is stop locations. All locations should be across the entire genome. See previous section 6.1 on converting genomic locations.

Any of the four entries may be left as NA.

The following specifies two fine.band regions, no broad.band, one genomic location, and one gene:

```
trackRegion = makeTrack(Broad.Band=NA,
                        Fine.Band = c("8p11.22","18p11.21"),
                        genomicLoc = c(0,6200000),
                        geneName = "FANCE")
```

6.3 updateGGV

The function updateGGV is a way to update a GGV object's (see section 2.1) list of known regions of interest, or trackRegion object. The user has the option to totally replace the existing trackRegion object or to append to the existing

trackRegion object of a GGV object. The following shows the function definition, note which arguments must be defined and which have default values:

```
updateGGV(GGV,
          trackRegions,
          appendTo=TRUE,
          returnVl=TRUE,
          saveFlag=FALSE,
          saveName="GGVobj.RData")
```

The GGV argument is a 'GGVobj'. The trackRegions argument is a trackRegion object (see previous section 5.2). The argument appendTo indicates if the existing GGV object's trackRegion should be replaced or if the new regions should be appended to the existing.

The arguments returnVl, saveFile, and saveName determine if the created object should be returned or saved. If returnVl is true, the object is returned. If saveFile is true, the object is saved as an R data object. The argument saveName is the complete path and name for the R data object.

Using the GGV object created in 2.1.8 and the trackRegion object in 5.2:

```
newGGV = updateGGV(GGV=GGV,
                   trackRegions=trackRegion,
                   appendTo=TRUE)
```

6.4 writeExFiles

The writeExFiles creates text files to be used with examples and vignettes. The files created are text files for cancer genes, disease genes, DNA repair genes, cytoband (band.info), and HB19Kv2.HG18 (mapping.info). See appendix B for more information on these files.

```
writeExFiles(direct="./")
```

The argument direct determines where the files are created. The default will create all files in the current working directory.

7 example code

7.1 summary of all code

The following takes the user through all steps in utilizing application. Keep in mind, for many labs, bandinfo objects and mapping objects may only need to be generated once. It can then be used for any other experiment for which the data applies. The same is also true for the annotation object. It is possible to exclude certain ann.info objects within the annotation object when plotting tracks. The user may therefore create a master annotation object to use for multiple projects.

```
Begin by making the band.info object since the mapping and annotation objects
require a band.info object.

band.info = makeBandInfo(file="cytoBand.txt",
                         chrom.levels=c("chr1","chr2","chr3","chr4","chr5","chr6",
                                        "chr7","chr8","chr9","chr10","chr11","chr12",
                                        "chr13","chr14","chr15","chr16","chr17",
                                        "chr18","chr19","chr20","chr21","chr22",
                                        "chrX","chrY"),
                         file.sep="\t",
                         returnVl=TRUE,
                         header=TRUE)


Remember the band.info object provided with the package may also be used.
data(Band.Info)




Next, build the mapping.info object.

mapping.info = mappingObj(file="HB19Kv2.HG18.txt",
                          spot.ID="Clone", chrom="Chromosome",
                          chrom.levels=c("chr1","chr2","chr3","chr4","chr5","chr6",
                                         "chr7","chr8","chr9","chr10","chr11","chr12",
                                         "chr13","chr14","chr15","chr16","chr17",
                                         "chr18","chr19","chr20","chr21","chr22",
                                         "chrX","chrY"),
                          loc.start="start", loc.stop="Stop",
                          file.sep="\t", header=TRUE,
                          additional=c("Mapped.by", "Flag"),
                          links=10, names.links="UCSC",
```

```
                              band.info=band.info,
                              returnVl=TRUE )
```

```
Next, assemble an annotation object.
First create individual ann.info objects.

# makes anninfo object for cancerGenes
annotation1 = makeAnnotation(file="CancerGenes.txt", file.sep="\t", header=TRUE,
                              label=2, chrom=3,
                              chrom.levels=c("chr1","chr2","chr3","chr4","chr5","chr6",
                                             "chr7","chr8","chr9","chr10","chr11",
                                             "chr12","chr13","chr14","chr15","chr16",
                                             "chr17","chr18","chr19","chr20","chr21",
                                             "chr22","chrX","chrY"),
                              band.info=band.info,
                              loc=NA, loc.start=4, loc.stop=5,
                              additional=0,  links=6)


# makes anninfo object for DiseaseGenes
annotation2 = makeAnnotation(file="DiseaseGenes.txt", file.sep="\t", header=TRUE,
                              label=2, chrom=3,
                              chrom.levels=c("chr1","chr2","chr3","chr4","chr5","chr6",
                                             "chr7","chr8","chr9","chr10","chr11",
                                             "chr12","chr13","chr14","chr15","chr16",
                                             "chr17","chr18","chr19","chr20","chr21",
                                             "chr22","chrX","chrY"),
                              band.info=band.info,
                              loc=NA, loc.start=4, loc.stop=5,
                              additional=0,  links=7)


# makes anninfo object for DNArepairGenes
annotation3 = makeAnnotation(file="DNArepairgenes.txt", file.sep="\t", header=TRUE,
                              label=1, chrom=2,
                              chrom.levels=c("chr1","chr2","chr3","chr4","chr5","chr6",
                                             "chr7","chr8","chr9","chr10","chr11",
                                             "chr12","chr13","chr14","chr15","chr16",
                                             "chr17","chr18","chr19","chr20","chr21",
                                             "chr22","chrX","chrY"),
                              band.info=band.info,
                              loc=NA, loc.start=3, loc.stop=4,
```

```
                                additional=0,links=5)

Now, assemble ann.info objects into an annotation object.

# initalizes annotation object
annObj = annotationObj(annotation1, obj.name="CancerGenes")

# adds additional anninfo objects to created annotation object, annObj
annObj = annotationObj(annotation2, annotationObj=annObj, obj.name="DiseaseGenes")
annObj = annotationObj(annotation3, annotationObj=annObj, obj.name="DNArepairGenes")



Remember the annotation object provided with the package may also be used.
data(annObj)
```

At this point, the user has three options:

1. plot a single interactive heatmap.
2. initialize a GGVobj for generating multiple interactive heatmaps across the genome.
3. initialize a TIplot object for a static or interactive tiled heatmap image

Each option will be examined further.

7.1.1 single interactive

It is possible to create an interactive heatmap without initializing an object. The following will plot chromosome arm 11q with no additional statistical plot. The annotation tracks will only include Cancer and Disease genes, not DNArepair.

```
# initialize variables used in function
xy.labels = list(lgr=vls)
y.lbls$Pdisc = round(y.lbls$Pdisc,3)
y.labels = y.lbls

# plot in current directory
iGGV(vls = round(mat, 3),
     mapObj=mapping.info,
     annObj=annObj,
     x.labels=x.lbls,
     y.labels=y.labels,
```

```
    xy.labels=xy.labels,
    genomic.region="11q",
    region.type="arm",
    mapObj.columns = c(2,8,10),
    annotation=c(1,2))
```

7.1.2 initialize GGVobj and plot using makeGGV

If necessary, make a trackRegion object.

```
trackRegions = makeTrack(Fine.Band = c("8p11.22","18p11.21"),
                         genomicLoc = NA, geneName = "FANCE")
```

Now, initialize the plotting object. This section will initialize a GGVobj for generating interactive plots across the genome. The initial genome plot will be of chromosome 8. A side plot of mock p.values is included. Only the known Cancer and Disease genes will be plotted from the annotation object. Note: this sets up an object to make a series of interactive plots, automatically generating directories. Remember: data should not be subset. Complete data should be used. The user has an opportunity to subset data when executing plots.

```
# sets up variables that will be used in function
xy.labels = list(lgr=vls)
y.lbls$Pdisc = round(y.lbls$Pdisc,3)
y.labels = y.lbls
pvls = rep(rep(rep(c(-1,rep(0,3),1,rep(0,3),.5,rep(0,3),-.75), each=10),
           150))[1:length(mapObj$mapping.info$g.loc.center)]


# initialize GGVobj
GGV = initGGV(vls = round(mat, 3),
              mapObj = mapping.info,
              annObj = annObj,
              x.labels=x.lbls,
              y.labels=y.labels,
              xy.labels=xy.labels,
              chrArms=c("8p", "18p"),
              trackRegions=trackRegions,
              side.plot.extras= "points(pvls,
                  GGV$values$mapObj$mapping.info$g.loc.center,
                  col='red', pch=21); title(main='test')",
              plot.vec=pvls[1:length(mapObj$mapping.info$g.loc.center)],
              plot.dx=which(mapObj$mapping.info$Chrom=="chr8"),
              annotation = c(1,2),
              mapObj.columns= c(2,8,10),
```

```
                smpLines=TRUE,
                divCol="lightgrey")
```

After initialization of the object, the user may use the makeGGV to execute plots. The user has options to subset samples and spots for graphs in the makeGGV function. The following will use all spots, but only the first ten samples. When initializing the GGVobj, regions of interest were specified through the trackRegion argument. These regions will automatically have tiled heatmaps generated; the following splits the tiled heatmap into two tracks.

```
makeGGV(GGV=GGV,
        smplDX=1:10,
        smp.color=c(rep(c("red", "blue", "purple", "green","yellow"),
                    each=4), rep("pink", 2)),
        tileNum=2)
```

7.1.3 initialize TIplot and plot using makeTiled or iGGVtiled

This section will initialize a 'TIplot' object to be used with either makeTiled or iGGVtiled. A tiled image is designed to show spot overlap and gaps in coverage. The following sets up a plotting object that will plot the 103-112 spots. The 106 spot is not included in the goodDX and therefore will not be plotted. Each sample will be broken into three tracks. Only the first ten samples are utilized.

```
TIplot = initTile(Z=mat,
                  bacDX=103:112,
                  mapObj=mapping.info,
                  goodDX=c(1:105, 107:dim(mapping.info$mapping.info)[1]),
                  smplDX=1:10,
                  H=3,
                  zlims=c(-.5,.5),
                  ylabels=paste("Spot",bacDX, sep=""),
                  xlabels=paste("smp",smplDX, sep=""),
                  xlab="Samples",ylab="SpotID",ttl="tiledImage")
```

After the initialization of the object, the user may use one of two functions for plotting: makeTiled or iGGVtiled. The makeTiled function produces a single, static tiled heatmap. The iGGVtiled function produces an interactive layout of plots with the main heatmap as a tiled image.

A makeTiled call:

```
makeTiled(TIplot,
          smpDiv=TRUE,
          divCol="lightgrey")
```

A iGGVtiled call:

```
# initialize side plot
spot.indx = TIplot$map$bacDX
plot.vec = round(rowMeans(TIplot$vls$Z),3)
plot.call = "image(x=0:1,y=0:1,z=matrix(rep(NA,4),ncol=2),
                xlim=c(range(plot.vec,na.rm=T)),
                ylim=range(mapObj$mapping.info$g.loc.center[spot.indx],na.rm=T),
                zlim=c(0,1),axes=F,xlab='',ylab='');
                points(x=plot.vec,y=mapObj$mapping.info$g.loc.center[spot.indx],
                pch=3, cex=0.5, col='purple');axis(2);axis(1)"


iGGVtiled(TIplot=TIplot,
          annObj=annObj,
          x.labels=as.data.frame(list(
                  sample.ID=paste("smp",1:TIplot$vls$nsmp,sep=""),
                  xla1=c("a","b","c","d","e","f","g","h","i","j"),
                  xla2=10:1)),
          y.labels=as.data.frame(list(
                  Spot.ID=paste("Spot",TIplot$map$bacDX,sep=""))),
          xy.labels=list(lgr=round(Z,3)),
          plot.call=plot.call, plot.vec=plot.vec,
          mapObj.columns = c(2,3,7),
          fname.root="iGGVtiled")
```

7.2 Example using data from aCGH package

This example uses the data set provided in the R library aCGH and the default annotation object from the iGenomicViewer library. This example also shows how to create a mapping object from a data.frame.

```
# load libraries
library(iGenomicViewer)
library(aCGH)

# load data
data(colorectal)
data(annObj)

# create the mapping object
mapObj = mappingObjDF(df=colorectal$clones.info,
          spot.ID=1, chrom=3,
          locBy="within", loc=4)

# initialize the GGVobj
GGV = initGGV(vls=colorectal$log2.ratios,
       mapObj=mapObj, annObj=annObj,
       x.labels = colorectal$phenotype[,1:3],
       xy.labels = list(vls=colorectal$log2.ratios))

# make plots
makeGGV(GGV=GGV)
```

Figure 10 shows the generated iGenomicViewer plot for the data in the aCGH package.

7.3 Example using data from SNPchip package

This example uses the data set provided in the R library SNPchip and the default annotation object from the iGenomicViewer library. This example also shows how to create a mapping object from a data.frame and plays with the plot layout margins.

```
# load libraries
library(iGenomicViewer)
library(SNPchip)

# load data
data(hapmap)
```

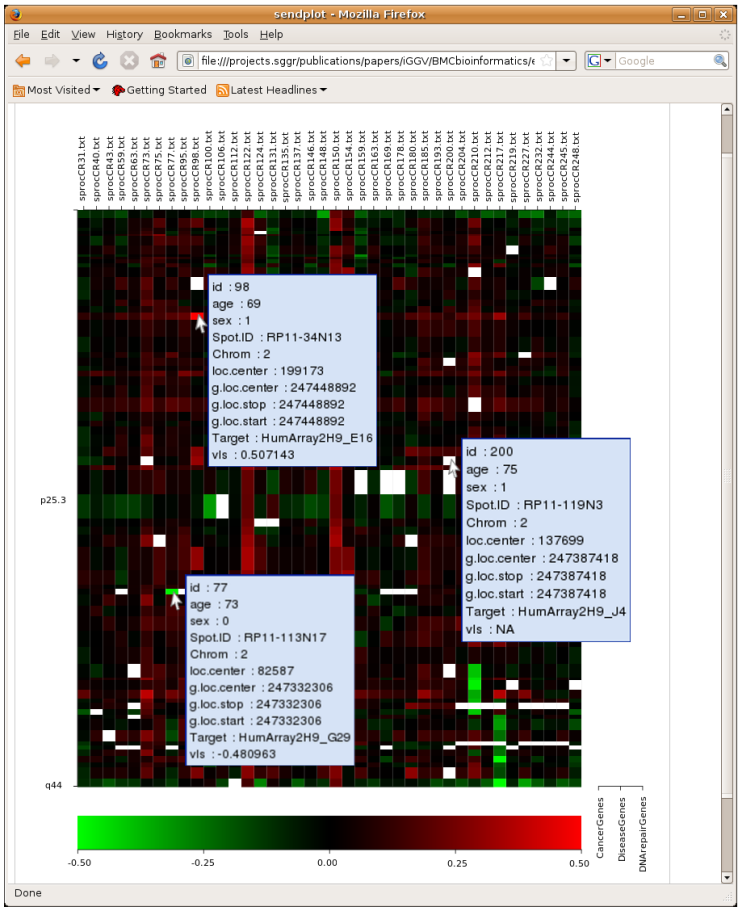

Figure 10: example interactive plot for aCGH data

```
data(sample.snpset)
data(annObj)

# create the mapping object
mapd = cbind(sample.snpset@featureData@data,
             row.names(sample.snpset@featureData@data))
names(mapd)[dim(mapd)[2]] = "ID"
mapObj = mappingObjDF(df =mapd, spot.ID=9, chrom=2,
            locBy="within", loc=3, additional=c(4:6),
            names.additional=c("strand", "alleleA", "alleleB"))


# sets values matrix to color code heatmap
mm = match(row.names(hapmap$calls),row.names(sample.snpset@featureData@data))
bddx = which(is.na(mm))
vls = hapmap$copyNumber[-(bddx),]

# plays with layout margins
mat = matrix(c(rep(c(rep(1,7), rep(3,1)), 8),
               rep(c(rep(2,7), rep(0,1)), 1)),
               byrow=TRUE, ncol=8)
mai.mat = matrix(c(
  0.2,  1.0, 1.0, 0.05,
  0.5,  1.0, 0.3, 0.05,
  0.2, 0.05, 1.0, 0.2)
  , byrow=TRUE, ncol=4)


# initialize the GGVobj
GGV = initGGV(vls=vls, mapObj=mapObj, annObj=annObj,
         xy.labels=list(copynumber=round(hapmap$copyNumber[-(bddx),],3),
                        calls=round(hapmap$calls[-(bddx),],3),
                        confidenceCalls=round(hapmap$callsConfidence[-(bddx),],3)),
                        lims=c(0,5), chrArms="6q", mat=mat, mai.mat=mai.mat)

# make plots
makeGGV(GGV=GGV)
```

Figure 11 shows the generated iGenomicViewer plot for the data in the SNPchip package.

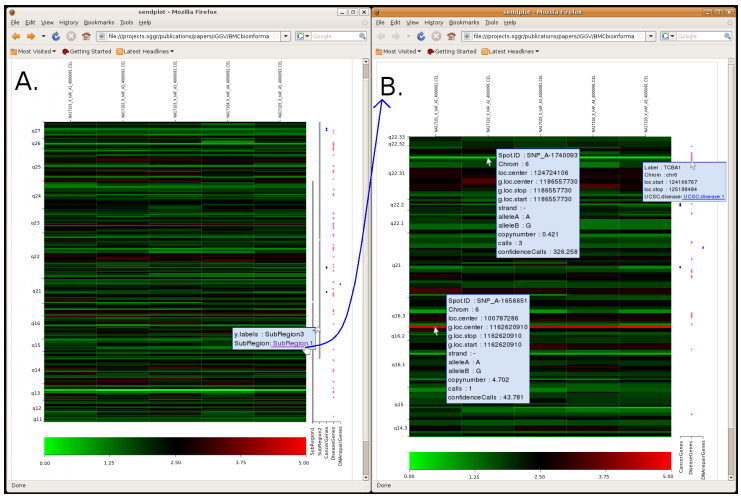

Figure 11: example interactive plot for SNP data

8 closing remarks

The iGenomicViewer application is highly adaptable and user customizable. Its platform independent design allows for use with all genomic data.
Please also see sendplot, the base package of this application, for a more general version of interactive layouts of plots.

Appendix

A Objects and Classes

This section contains brief desciptions and the structure of data objects and classes within the iGenomicViewer pacakge.

A.1 anninfo

The anninfo object contains all information for a given annoation set. This includes but is not limited to names, chromosome locations and genomic locations. An annotation Object is a list of 'anninfo' objects. Structure:

```
anninfo - class 'anninfo' - list of length 2

  1. annotation - data.frame
       minimally contains columns for label and chromosome as well as genomic location.
       genomic location columns are locations within chromosome. either a single column
       representing central midpoint or two columns representing starting and stopping.
       Additional columns depend on user setting when creating object with
       makeAnnotation

  2. links - data.frame
       contains complete web address information for annotation set.
       (i.e http://....com, etc.)
       it should have the same row dimension as the annotation data.frame.
```

A.2 bandinfo

The bandinfo object contains useful data.frames of starting and ending genomic locations for chromosomes, arms, broad.bands, and fine.bands. Each data.frame has five columns: the region (chrom, arm), label, lower, center, and upper. lower, center, and upper refer to genomic locations across the entire genome. label is the label that will be used for the region when plotting. The bandinfo object will also contain an offset object. This is a numeric vector of length equal to the number of chromsomes. It contains the numeric buffer that should be added to genomic locations given within chromosome to get the location with respect to the entire genome. Structure:

band.info - class 'bandinfo' - list of length 5

1. offset - numeric vector
   buffers to convert genomic locations with respect to chromosome to genomic locations with respect to the entire genome. equal in length to the number of chromosomes

2. Chrom - data.frame - n x 5
   The five columns are: Chrom, label, lower, center, upper
   label is the label that will be used for the region when plotting.
   lower, center, and upper genomic locations with repect to the entire genome

3. Arm - data.frame - n x 5
   The five columns are: Arm, label, lower, center, upper
   label is the label that will be used for the region when plotting.
   lower, center, and upper genomic locations with repect to the entire genome

4. Broad.Band -data.frame - n x 5
   The five columns are: Broad.Band, label, lower, center, upper
   label is the label that will be used for the region when plotting.
   lower, center, and upper genomic locations with repect to the entire genome

5. Fine.Band - data.frame - n x 5
   The five columns are: Fine.Band, label, lower, center, upper
   label is the label that will be used for the region when plotting.
   lower, center, and upper genomic locations with repect to the entire genome

A.3 GGVobj

The GGVobj is a genomic genomic viewer object that contains all necessary information to plot a series of interactive plots. Structure:

GGVobj - class GGVobj' - list of length 3

1. values - list of length 9 contains values, objects, and matricies with main data source

   (a) vls - matrix of values for the heatmap. The rows should correspond to the spot.IDs of an associated mapObj

(b) mapObj - an object of the class mapobj see section 4.2 or appendix A.4

(c) annObj - an annotation object see section 4.3 or appendix A.1

(d) chrArms - character vector of chromosome arms to be plotted. These names should match arm names in the mapObj's band.info. (mapObjArmArm). When GGVobj is plotted using iGGV, these chromosome arms appear in the index file

(e) trackRegions - a list containing known regions of interest see appendix A.6 and section 5.2. Max length 4. May be NA.

(f) mat - matrix indicating layout. This argument will be passed into the graphics package layout call as mat.Each value in the matrix must be '0' or a positive integer. If N is the largest positive integer in the matrix, then the integers 1,...,N-1 must also appear at least once in the matrix. '0' indicates region of no plotting. This may be left as NA, and a default will be used. This matrix will be used for Chromosome Arm and Sub.Arm Plots, when the GGVobj is plotted using iGGV.

(g) plot.call - character containing plot call that will be evaluated. This plot is added to the right of the annoation tracks. It is designed to add additional statistical analysis such as p.values, linear order statistics, etc. If NA, no plot will be added to the display.

(h) plot.vec - vector of values that will be plotted in plot.call. While this may be specified in the plot.call, it is also necessary in order to add interactive tool-tip to values.

(i) plot.dx - Index to subset plot.call and plot.vec on when creating an initial genomic plot in the iGGV function.

2. interactive - list of length 7 contains matricies and vectors of interactive data for plots

(a) x.labels - data frame of n x m which contains values relating to the x-axis values. n is equal to the second dimension of the values object vls. This information is displayed in the interactive plot window

(b) y.labels - data frame of n x m which contains values relating to the y-axis values. n is equal to the first dimension of the values object vls. This information is displayed in the interactive plot window

(c) xy.labels - list of matricies. All matricies should be of n x m. n is equal to the first dimension of the values object vls and m is equal to the second dimension of the values object vls . These matricies therefore are of the same dimension as the values object vls. This information is displayed in the interactive plot window

(d) x.links - data frame of n x m which contains web addresses for links relating to the x-axis values. n is equal to the second dimension of

the values object vls. This information is displayed in the interactive plot window

(e) y.links - data frame of n x m which contains web addresses for links relating to the y-axis values. n is equal to the first dimension of the values object vls. This information is displayed in the interactive plot window

(f) xy.links - list of matricies. All matricies should be of n x m. n is equal to the first dimension of the values object vls and m is equal to the second dimension of the values object vls . These matricies therefore are of the same dimension as the values object vls. This information is displayed in the interactive plot window

(g) asLinks - contains complete web address for points that should be treated as hyperlinks. Either NA or equal to the length of n x m of the vls matrix.

3. info - list of length 10 contains values of other arguments used in functions calls

(a) maxLables - maximum number of labels to appear on the heatmap y axis. When plotting using iGGV, the function will automatically determine, based on this number, if chromosome, arm, broad.band, fine.band, or spot.IDs should be used on the y-axis.

(b) mai.mat - n x 4 matrix of values to be passed in for each plots par mai. n will be 3 if plot.call is NA, and 4 if plot.calls is specified. This will be used for Chromosome Arm and Sub.Arm plots when iGGV is utilized. The 4 columns represent bottom, left, top, right respectively.

(c) mai.prc - logical indicating if mai mat values are percentages or hard coded values. If mai.prc is T, indicates percentage. This will be used for Chromosome Arm and Sub.Arm plots when iGGV is utilized for plotting.

(d) plot.extras - List of length equal to the number of plots: 3 if plot.call is NA, 4 if plot.call is specified. This object is a list of lists. The sublists contain any additional plotting calls that should be executed for the plot. Each entry must be a character vector. If no additional plotting is equired, an NA should be used

(e) smpLines - logical indicating if vertical lines should be added between each sample of the heatmap when plotted

(f) divCol - If smpLines, the color of the dividing lines

(g) lims - Lower and Upper limit for vls. numeric vector of length 2. Any value outside this limit will be rounded to the lower or upper bound.

(h) annotation - Numeric indication of which annotation information objects to include from the annObj. If NA all are used. Allows user to

control which of the annotation sets (anninfo objects) of the annotation object to use.

(i) clrs - Character vector of colors to use for annotation track

(j) mapObj.columns - Which columns from mapObj's mapping.info data.frame to include in tool-tip. May be numeric or header names.

A.4 mapobj

The mapping object contains all mapping information which includes but is not limited to spotIDs, chromosome locations, and genomic locations within chromosome. The mapping object will also have a band.info object associated with it in order to map spot.IDs to chromosome, arm, broad bands and fine bands. The mapping object allows for customized use of iGenomicViewer independent of experiment platform. The IDs only need to have chromosome and genomic location information to be used with the application. Structure:

```
mapObj - class 'mapobj' -  list of length 3

  1. band.info - a bandinfo object
       see section 4.1, and appendix A.2 and B.2

  2. mapping.info - data.frame
       Minimally the data.frame must contain columns for IDs, Chromosome and genomic
       location. The genomic location[s] should be within chromosome.
       see section 6.1 on converting location.
       It is recommended to have two columns of genomic location: a starting location
       and ending location. A single centeral location may also be given.
       Additional columns of information included depends on the mapping file used and
       the user settings when creating the object with mappingObj function. Any number
       of additional columns may be included in the data.frame

  3. links - data.frame
       contains complete web addresses for mapping (i.e http://....org). It should
       correspond to the rows in the mapping.info data.frame. NA represent no link.
       Each column of data.frame may represent a different set of links
```

Please also see section 4 and appendix B.9 for more information on creating a mapping object and the object provided with the package.

A.5 TIplot

The TIplot, or tiled image, object contains all necessary information to plot a tiled image heatmap. Please see section 2.2 for more details on creating a TIplot

object. Structure:

```
TIplot - class 'TIplot' - list of length 7

  1. tractBound - list - length variable based on number of tracks desired
       Each item in the list is also a list of length 4.
        trackObject structure:

        a. tbound - a numeric vector of genomic starting and stopping locations for
             spot.IDs on this track. The values will be in genomic order and also
             include the minimum and maximum Y-axis range.

        b. Ymap - index of starting genomic locations for spot.IDs on this track
             with respect to tbound. In other words, which values in tbound are
             genomic starting locations

        c. xdx - x index for tiled graph. The tiled graph x-axis will be
             0:(number of samples * the number of tracks). This index is which x
             values correspond to this number of track.
             i.e. if there are two tracks being utilized and 5 samples: the x axis is
             1:10 and the xdx for the first trackobject is 1,3,5,7,9. if there are
             three tracks, the x-axis is 1:15 and the xdx of the first object is
             1,4,7,10,13

        d. ydx - The index of which spot.IDs are graphed on this track with respect
             to the order in the bacDX. If an index of length 10 is used for the bacDX,
             which of the 10 spots used should be on this track.
             i.e. If there are three tracks for ten spots the ydx would typically
             be 1,4,7,10.

  2. vls - list of length 7

       a. Y - numeric matrix - n x 2
            matrix of starting and stopping genomic locations. The first column is
            starting genomic locations and the second stopping locations. n is
            dependent on the number of spot.IDs, or the length of bacDX in the
            function initTiled

       b. Z - numeric matrix - n x m
            a matrix of values to be plotted. These are used for the values in the
            heatmap. n is equal to the number of samples, or smplDX in initTiled. m
            is equal to the number of spot.IDs, or bacDX in initTiled.

       c. H - numeric - the number of tracks
```

```
     d. Xcoords - numeric vector - length equal to
                  (number of samples * number of tracks)
          The actual x-axis values to use for plotting.

     e. nsmp - numeric
          total number of samples used (length of smplDX when using initTiled)

     f. nBAC - numeric
          total number of spots used (length of bacDX when using initTiled)

     g. smplDX - numeric vector
          the index of samples that is used. (smplDX of initTiled)

3. lims - list of length 3

     a. xlim - numeric vector of length 2
          minimum and maximum x-axis values

     b. ylim -  numeric vector of length 2
          minimum and maximum y-axis values

     c. zlim - numeric vector of length 2
           minimum and maximum z-axis values

4. labels - list of length 5

     a. xlab - character
          main x-axis label. This may be NA for no label

     b. ylab - character
          main y-axis label.  This may be NA for no label

     c. ttl -  character
          title of tiled image plot.  This may be NA for no title

     d. xlabels - character vector
          optional names for x-axis. length is equal to the number of
          samples (length of smplDX). This may be NA

     e. ylabels - character vector
          optional names for y-axis. length is equal to the number of
          spots (length of bacDX). This may be NA

5. Zcol - character vector
     vector of colors for the Z matrix scale. This range of colors will
```

```
     be used to determine heatmap colors

  6. cex - list of length 2

       a. xcex - numeric
            point size, labels size for x-axis

       b. ycex - numeric
            point size, labels size for y-axis

  7. map - list of length 2

       a. bacDX - integer/numeric vector
            list of spot ID index used. (bacDX of initTiled)

       b. mapObj - mapobj
            please see section 4 for creating a mapobj or
            appendix A.4 and B.9 for mapobj structure
```

A.6 trackRegion

The trackRegion object is a list. It can be NA or a list up to length 4. This object is responsible for designating known regions of interest. There are four possible ways to identify a region of interest: broad.band, fine.band, genomic region, or a gene (really any annotation label) of interest. The trackRegion object therefour has those four possible options. Structure:

```
trackRegion - NA or list up to length 4
  any of these 4 may be NA or excluded from the object

  1. Broad.Band - NA, character, or character vector
       Indicates broad band regions that are of interest.
       These name[s] should correspond/match one of the broad.band names
       in the associated band.info object's
       band.info$Broad.Band$Broad.Band column
       see section 4.1 or appendix A.2 or B.2 for more information on band.info

  2. Fine.Band - NA, character, or character vector
       Indicates fine band regions that are of interest.
       These name[s] should correspond/match one of the fine.band names
       in the associated band.info object's
       band.info$Fine.Band$Fine.Band column
```

see section 4.1 or appendix A.2 or B.2 for more information on band.info

3. geneName - NA, character, or character vector
   The name may be slightly deceiving. This indicates an annotation label of interest. These name[s] should correspond/match one of the Labels in the annotation data.frame column Label of one of the anninfo objects in the associated annotation object
   see section 4.3 or appendix A.1 or B.1 for more information on annotation object and anninfo objects

4. genomicLoc -NA or numeric matrix, n x 2
   Indicates a genomic region of interest based on genomic location across the entire genome. The first column is starting genomic location and the second column is stopping genomic location.
   see section 6.1.2 for converting within chromosome location to across genome location

Again any of these four may be excluded from the trackRegion object.

B Datasets

This section gives brief descriptions and formats of objects or files provided with the iGenomicViewer library.

B.1 annobj

The annotation object provided has three annotation sets: Cancer Genes, Disease Genes, and DNA repair genes. Please see appendix B.3, B.5 and B.6 for more on the files used to create this object. Please also see appenidx A.1 for more on object structure and section 4.3 for creating annotation objects.

```
data(annObj)
```

B.2 Band.Info

The bandinfo object provided is created from cytoBand.txt. Please see appendix B.4 and A.2 for more information on this file and object structure. Please also see section 4.1 for creating bandinfo objects.

```
data(Band.Info)
```

B.3 CancerGenes

A reference for known cancer genes:

```
Futreal PA, Coin L, Marshall M, Down T, Hubbard T, Wooster R, Rahman
N, Stratton MR. A census of human cancer genes. Nat Rev Cancer. 2004
Mar;4(3):177-83
```

The cancer gene file has six columns: c1, gene, chrom, startLoc, endLoc, and UCSC.cancer. The columns represent entrez gene ID , gene label, chromosome, starting genomic location within chromosome, ending genomic location within chromosome, and a link to the UCSC genome browser. The links were added by using a perl script coded by our bioinformatists.

The following is a reference to the UCSC genome browser:

```
Kent WJ, Sugnet CW, Furey TS, Roskin KM, Pringle TH, Zahler AM,
Haussler D. The human genome browser at UCSC. Genome Res. 2002
Jun;12(6):996-1006
```

```
The following will load as a data.frame:
data(CancerGenes)

To have the file written to a directory for use (also see section 5.4):
writeExFiles()
```

B.4 cytoBand

A reference for the cytoband file:

```
Karolchik D, Kuhn, RM, Baertsch R, Barber GP, Clawson H, Diekhans M,
Giardine B, Harte RA, Hinrichs AS, Hsu F, Miller W, Pedersen JS, Pohl
A, Raney BJ, Rhead B, Rosenbloom KR, Smith KE, Stanke M,
Thakkapallayil A, Trumbower H, Wang T, Zweig AS, Haussler D, Kent
WJ. The UCSC Genome Browser Database: 2008 update. Nucleic Acids
Res. 2008 Jan;36:D773-9
```

The cytoband file has five columns: chrom, startLoc, endLoc, band, and c5. The columns represent chromosome location, starting genomic location within chromosome, ending genomic location within chromosome, band label, and strain.

```
The following will load as a data.frame:
data(cytoBand)

To have the file written to a directory for use (also see section 5.4):
writeExFiles()
```

B.5 DiseaseGenes

A reference for known disease genes:

```
Online Mendelian Inheritance in Man (OMIM), a knowledgebase of human
genes and genetic disorders. Hamosh A, Scott AF, Amberger J, Bocchini
C, Valle D, McKusick VA. Nucleic Acids Res. 2002 Jan 1;30(1):52-5
```

The disease gene file has seven columns: c1, gene, chrom, startLoc, endLoc, c5, UCSC.disease. The columns represent entrez gene ID, gene label, chromosome, starting genomic location within chromosome, ending genomic location within chromosome, OMIM disease ID , and a link to the UCSC genome browser. The links were added by using a perl script coded by our bioinformatists.

The following is a reference to the UCSC genome browser:

```
Kent WJ, Sugnet CW, Furey TS, Roskin KM, Pringle TH, Zahler AM,
Haussler D. The human genome browser at UCSC. Genome Res. 2002
Jun;12(6):996-1006
```

```
The following will load as a data.frame:
data(DiseaseGenes)
```

```
To have the file written to a directory for use (also see section 5.4):
writeExFiles()
```

B.6 DNArepairGenes

```
The following will load as a data.frame:
data(DNArepairgenes)

To have the file written to a directory for use (also see section 5.4):
writeExFiles()
```

B.7 HB19Kv2.HG18

A reference for the HB19Kv2.HG18 file:

```
The March 2006 human reference sequence (NCBI Build 36.1,
Hg18). http://www.ncbi.nlm.nih.gov/genome/guide/human/release_notes.html#b36
```

The HB19Kv2.HG18 file has ten columns: Clone, Chromosome, start, Stop, Center, g_loc, Band, Mapped.by, Flag, and UCSC. The columns represent clone or spot.ID name, chromosome location, starting genomic location within chromosome, ending genomic location within chromosome, central genomic location within chromosome (average of starting/ending locations), genomic location across the entire genome, band association location, indicating of how the clone was mapped, a quality flag, and a link to the UCSC human genome browser. The links were added by using a perl script coded by our bioinformatists.

The following is a reference to the UCSC genome browser:

```
Kent WJ, Sugnet CW, Furey TS, Roskin KM, Pringle TH, Zahler AM,
Haussler D. The human genome browser at UCSC. Genome Res. 2002
Jun;12(6):996-1006


The following will load as a data.frame:
data(HB19Kv2.HG18)

To have the file written to a directory for use (also see section 5.4):
writeExFiles()
```

B.8 iGGVex

The dataset iGGVex is an example dataset to run help file examples and vignette examples. This data is taken from an array competitive hybridization experiment using 19116 BAC clones and 22 samples. It has three objects: mat, x.lbls, and y.lbls. The object mat is a numeric matrix of log2(tumor/control)

ratios. It is of the dimension 19116 x 22. The x.lbls object is a data.frame that provides information concerning samples. It is of the dimension 22 x 4. The sample information provided are sample.ID, Date, Stage, and source of samples. The y.lbls object is a data.frame that provides information concerning the spot.ID's, in this case BACs. It is of the dimension 19116 x 3. The data information provided for the BACs are spot.ID, map.flag, and Pdisc.

```
data(iGGVex)
```

B.9 mapping.info

The mapping object provided is created from the default bandinfo object and the HB19Kv2.HG18 file. Please see appendix B.7 and B.2 for more information on this file and object. Please see appendix A.4 for structure information. Also see section 4.2 for creating a mapobj.

```
data(mapping.info)
```