Quick Manual for iGenomicViewer

Lori A. Shepherd and Daniel P. Gaile

September 16, 2009

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 a tutorial 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.

Important Note: Please also see section 5 for a few different examples of use.

2 Band, Mapping, and Annotation Information

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. The following sections utilize files which are provided through the writeExFiles function.

```
> writeExFiles()
```

2.1 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. The band information column should not include chromosome (i.e. 'p36.11', 'q42.3').

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)
```

This default band.info object is also provided as a data object that may be loaded:

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

Note: The band.info file allows for correct plotting of chromosome and band information in graphs. In can be used multiple times once created until an updated band location file is released. This default band.info object therefore can be applied to most data.

2.2 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 package offers a few different options for creating a mapping object depending on other objects being used. Brief descriptions of options are as follows:

- mappingObj : creates mapping object from a file
- mappingObjADF : creates mapping object from an annotatedDataFrame
- mappingObjDF : creates mapping object from a data.frame
- mappingObjMarray : creates mapping object from object of the class marrayInfo, marrayRaw or marrayNorm

The following will describe the mappingObj function. Please see section of examples for use of other mapping functions. The mappingObj 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. The function provides a convert functions for assisting in generating correct genomic locations.

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. If no band.info object is specified (band.info=NA), the default band.info object provided with the package will be used. 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 )
```

This default mapping.info object is also provided as a data object that may be loaded:

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

Note: The mapping file is key to the versatility of the package, and generally is unique to experiment or lab. This default mapping object will only be applicable to the example data provided with the package.

2.3 Annotation Object

The annotation object allows the user to display certain regions of interest or certain data sets of important information. The provided default annotation object provides displays for known Cancer genes, Disease genes, and DNA repair genes (This information was found on the UCSC website). It can be used with any data set and may be loaded with the following call:

```
data(annObj)
```

The following will describe how to make this object from files. 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)
```

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 is an example function call:

```
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:

```
# 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")
```

3 Initializing 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.

3.1 initGGV

The initGGV function initializes a 'GGVobj', a generic genomic viewer object. 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.

3.1.1 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

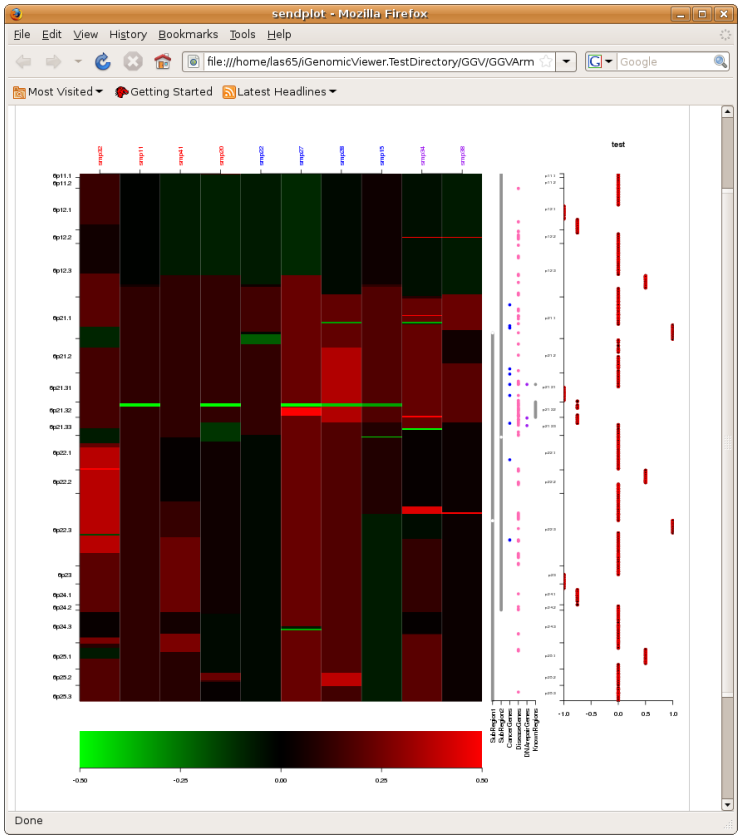

Figure 1: chromosome arm 6p

therefore directly corresponds to 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. The function provides default objects which will be used.

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

3.1.2 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.

3.1.3 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. 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, see package help files for more details on this function.

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

3.1.4 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.

3.1.5 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.

3.1.6 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.

3.1.7 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.

3.1.8 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")
```

3.2 initTile

The initTile functions initializes a 'TIplot', tiled image plotting object.

3.2.1 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. 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)
```

3.2.2 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.

3.2.3 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.
- 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.

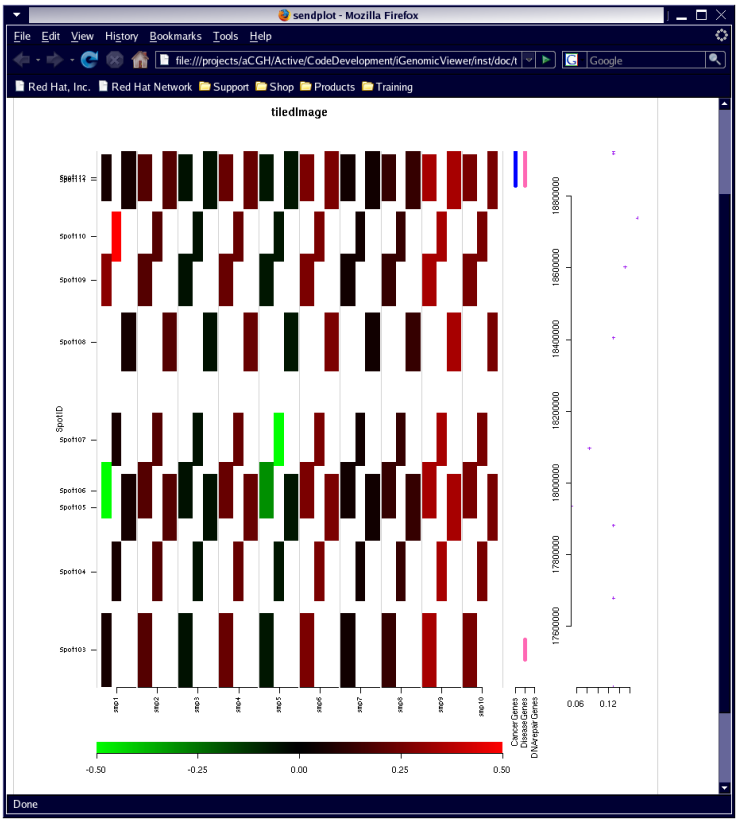

Figure 2: tiled image plot

3.2.4 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.

3.2.5 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")
```

3.3 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 4.3 for more details.

4 Making Plots

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

4.1 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.

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.

4.1.1 specifying objects, spot index and sample index

The GGV argument is a 'GGVobj' object.. The GGV object contains all information needed to make a directory structure of interactive, linked genomic

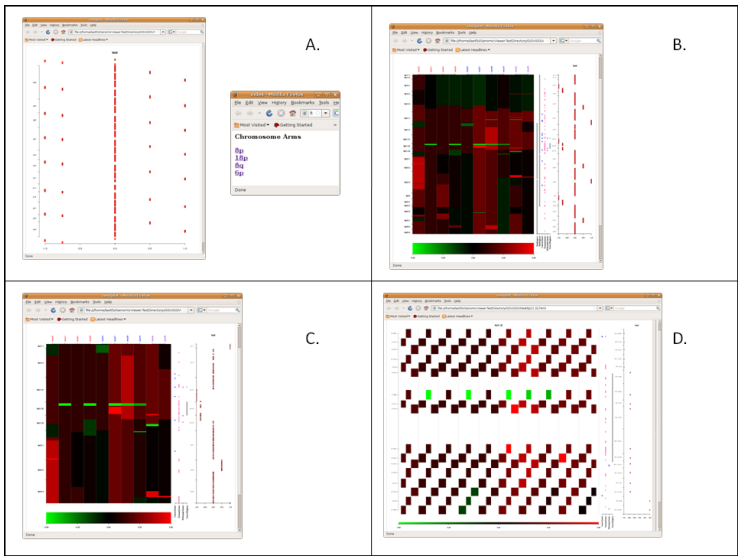

Figure 3: Different levels of plots when using makeGGV function

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 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.

4.1.2 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"
```

4.1.3 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.

4.1.4 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.

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.

4.1.5 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))

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.

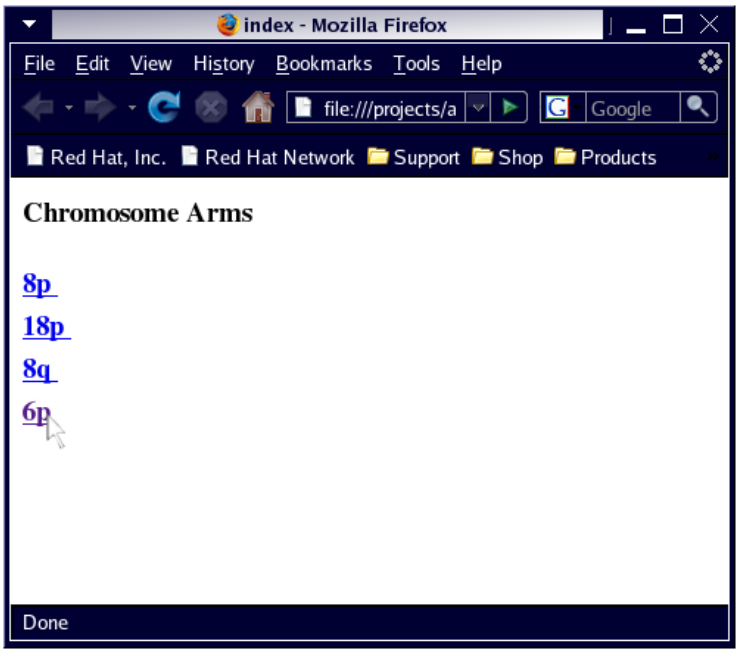

Figure 4: If we begin with the Index file, select a region of interest. The regions listed are determined by the user settings in chrArms, plot.dx, and known regions of interest when creating the GGVobj.

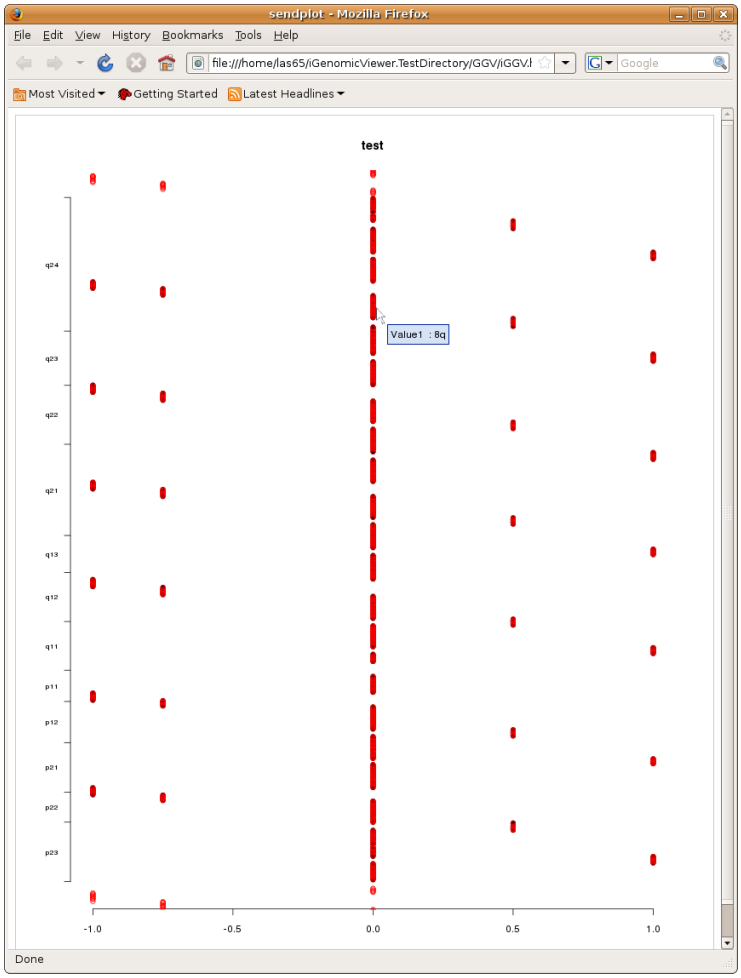

Figure 5: If we begin with the Genomic file, select a point of interest. The region depicted is dtermined by the plot.dx when creating a GGVobj. If no additional plot is given, only the index file is created.

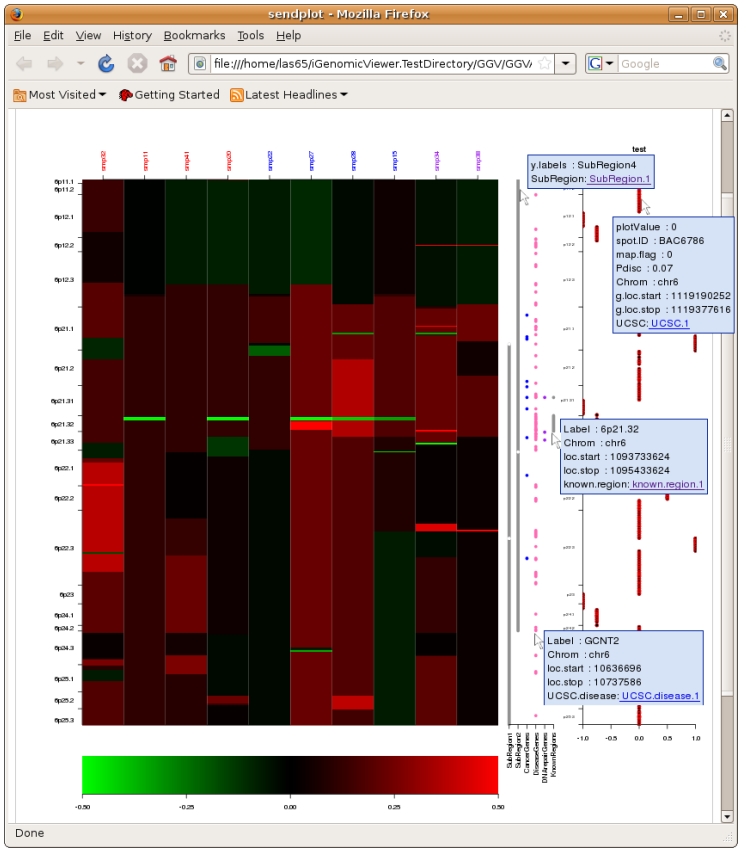

Figure 6: Shows next level, chromosome arm, using 6p. Notice the different areas that have interactivity. If the user clicks on the underlined hyperlinks in the tool-tip, a new plot or website will appear. The link SubRegion.1 will bring up another fully interactive heatmap of the region between the 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.

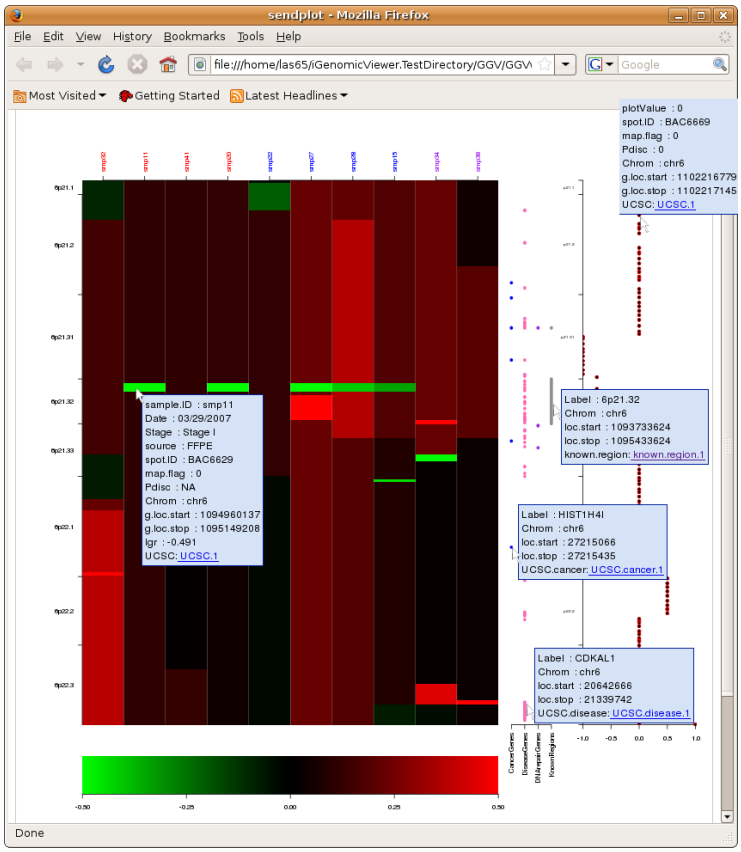

Figure 7: Shows next level, smaller viewer of chromosome arm 6p. Notice the different areas that have interactivity, and that now the heatmap is also interactive. If the user clicks on the underlined hyperlinks in the tool-tip, a new plot or website will appear. All tracks of the annotation plot are 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.

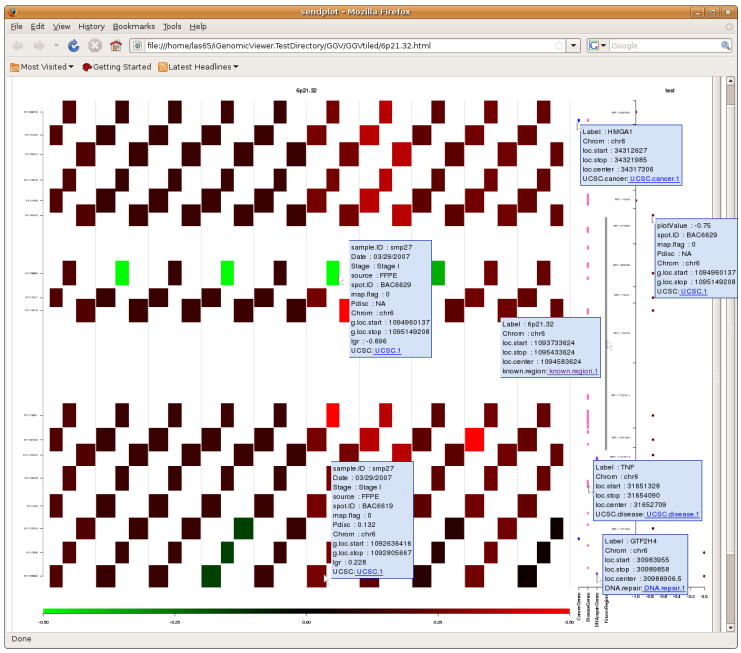

Figure 8: The lowest level depcits a tiled image plot. It is made for smaller regions to show spot overlaps and gaps. Notice the different areas that have interactivity. Each box in the track is interactive, therefore there are multiple tracks per samples as shown. All tracks of the annotation plot are 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.

4.2 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.

4.2.1 specifying objects

The TIplot argument is a 'TIplot' object. 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. The example will continue with the annotation object provided by the library.

```
data(annObj)
```

4.2.2 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.

4.2.3 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 3.1.5.
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.

4.2.4 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.

4.2.5 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 3.1.6 for more details.

4.2.6 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.

4.2.7 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.

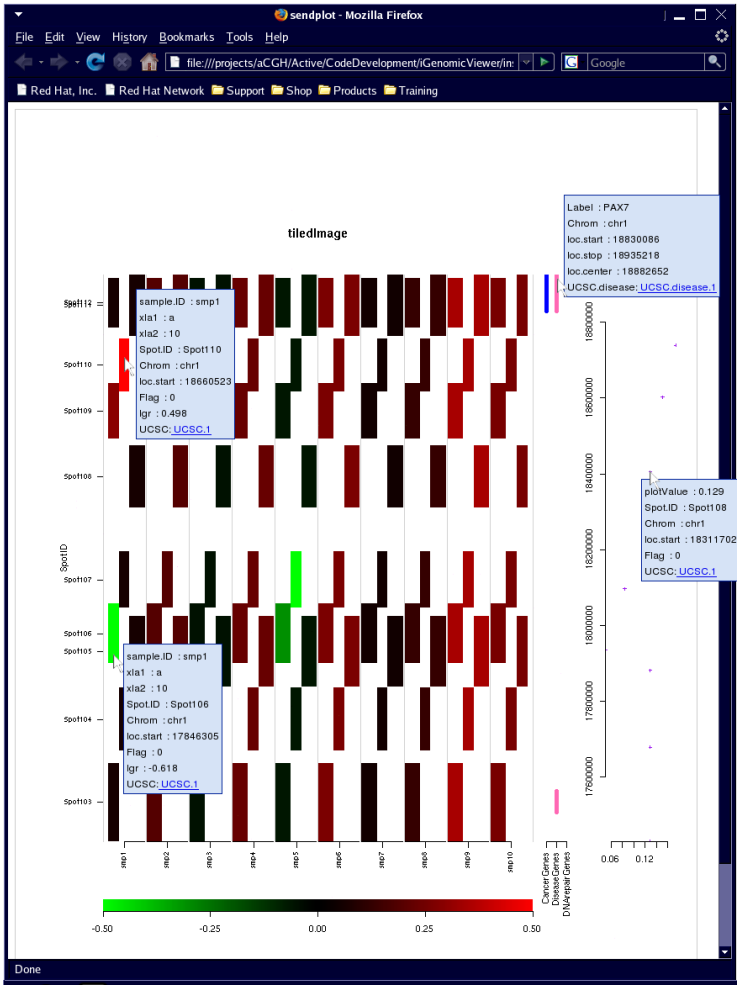

Figure 9: the tiled image view is made for smaller regions to show spot overlaps and gaps. Notice the different areas that have interactivity. Each box in the track is interactive, therefore there are multiple tracks per samples as shown. All tracks of the annotation plot are interactive. This example 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.

4.3 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.

4.3.1 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. 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. The function provides default objects which will be used.

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

4.3.2 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 3.1.2. Please see this section for more details. Revisiting the code from section 3.1.2:

```
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)
```

4.3.3 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.

4.3.4 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

4.3.5 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
```

4.3.6 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 3.1.6 for more details.

4.3.7 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 overwrite-SourcePlot.
- 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

4.3.8 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)


}
```

4.4 makeTiled: a static plot

The makeTiled function creates a single, static tiled image heatmap. The following is an example function call:

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

The argument TIplot is a TIplot object. The example will continue assuming the object in 3.2.5 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.

5 Examples

This section will show a few differetn examples, including some not using default objects to show versatility and compatiblity with other R objects.

5.1 1

The first 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, chrom.levels=1:23,
          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)
```

5.2 2

This example will recap the code in the vignette for a GGVobj.

```
# load library and objects
library(iGenomicViewer)
data(annObj)
data(mapping.info)

# initialize data for interactive window
y.lbls$Pdisc = round(y.lbls$Pdisc,3)
y.labels = y.lbls
```

```
# make object to indicate regions of interest
trackRegions = makeTrack(Fine.Band = c("8p11.22","6p21.32","18p11.21"),
                         genomicLoc = NA, geneName = "FANCE")

# initializes object to create side plot
# This is completely optional
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")


# initialize object
GGV = initGGV(vls = round(mat,3),
              mapObj = mapping.info,
              annObj = annObj,
              x.labels=x.lbls,
              y.labels=y.labels,
              xy.labels=list(lgr=vls),
              chrArms=c("8p", "18p"),
              trackRegions=trackRegions,
              side.plot.extras=side.plot.extras,
              plot.vec=plot.vec,
              plot.dx=plot.dx,
              mapObj.columns=c(2,8,10),
              smpLines=TRUE,
              divCol="lightgrey")

# color samples
smp.color= c(rep(c("red", "blue", "purple", "green","yellow"), each=4), rep("pink", 2))

# make plots
makeGGV(GGV=GGV, goodDX=NA, smplDX=1:10,
        smp.color=smp.color, tileNum=3,
        tiled.window.size = "1200x1100")
```

5.3 3

This example will recap the code in the vignette for a Tiled Plot.

```
# load library and objects
library(iGenomicViewer)
```

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

# initialize object
TIplot = initTile(Z=mat,
                  bacDX=103:112,
                  mapObj=mapping.info,
                  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")


# initializes object to create side plot
# This is completely optional
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)"

# make plot
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")
```