Introduction to gff3Plotter

Oleg Sklyar

October 3, 2006

1 Introduction

The package is designed to provide plotting routines for tiling array experiments. Data of tiling array experiments is plotted alongside with a genomic layout and the results of further selected experiments. The name of the package originates from the data file format used to represent tiling array data, GFF version 3. However, the plotting routines are virtually independent from routines for reading and parsing GFF3 files.

2 Installation and Start-up

The package was designed for Unix/Linux or MacOS platform and was tested on such. However, it is written purely in R using only standard R packages. Therefore, the package is likely to work on Windows as well. It is not computationally extensive and should run on any modern hardware. Plotting routines that ouput PNG and JPG graphic files may be used together with the gd graphic library. In this case, the gd library itself and `GDD` R package must be installed before, which determine the platform dependence of `gff3Plotter`.

Follow the standard installation procedure for R packages to install `gff3Plotter`.

After installing the `gff3Plotter` and starting R, the package can be loaded by issuing

```
> library("gff3Plotter")
```

3 Loading and Parsing GFF3 Files

Reading and parsing GFF3 files is performed by function *read.gff* of `gff3Plotter`. It is assumed that files contain infromation about tiling array experiments

(coded as genomic_DNA in the column TYPE), however the function will work even if this information is missing, The following example demonstrates the usage:

```
> dataDir = system.file("extData", package = "gff3Plotter")
> gff3File = paste(dataDir, "test.gff3", sep = "/")
> data = read.gff(gff3File)

gffLoad# LOADING < /tmp/Rinst2687783752/gff3Plotter/extData/test.gff3 >
gffLoad# Removing Extra Data
gffLoad# Splitting Types
done
gffLoad# Splitting <gene> Attributes: Name-done; ID-done
gffLoad# Removing ATTRIBUTES Field From <gene>: done
gffLoad# Splitting <genomic_DNA> Attributes: expNames-done; scoreType-done; coeff-don
gffLoad# Removing ATTRIBUTES Field From <genomic_DNA>: done
gffLoad# Splitting <expro> Attributes: expNames-done; scoreType-done; coeff-done; coe
gffLoad# Removing ATTRIBUTES Field From <expro>: done
gffLoad# Splitting <insitu> Attributes: anatomy-done; devStage-done
gffLoad# Removing ATTRIBUTES Field From <insitu>: done
gffLoad# Removing ATTRIBUTES Field From Other Types: exon-done; complete
gffLoad# Sorting START of genomic_DNA:  complete
gffLoad# Removing <expro> Records matching no Genes - done
gffLoad# Updating <expro> START,END,STRAND Fields from <gene> - done
gffLoad# Splitting <expro> by <expName> - done
gffLoad# Removing <insitu> Records matching no Genes - done
gffLoad# Updating <insitu> START,END,STRAND Fields from <gene> - done
gffLoad# Splitting <genomic_DNA> by <expName> - done
gffLoad# LOAD COMPLETE
```

Further help on this function is available by typing ?read.gff in R session after loding the package.

read.gff requires that a GFF3 file contains a header of the following tab-separated structure: SEQ_ID SOURCE TYPE START END SCORE STRAND PHASE ATTRIBUTES. All fields must be present. Comments are allowed on top of the file. It is not necessary to have all possible types of information, but *read.gff* will automatically search for the following definitions in the TYPE field: gene, exon, insitu (in situ data for genes), regulatory_region, genomic_DNA - (tiling array experiment data), and expro (data of further experiments). Other types are ommitted and not parsed.

read.gff returns a list, elements of which are named by the above mentioned types. Apart from `genomic_DNA` and `expro`, which are further represented by lists, all other fields are `data.frame`-s. `genomic_DNA` and `expro` are in turn lists of `data.frame`-s with elements named by experiment names.

The package contains two more functions to help in parsing and organising data from GFF3 files: *getAttr* and *gffGetSubset* defined as:

`getAttr <- function (x, field, splitter = ";[[:space:]]?")` `gffGetSubset <- function(data, range)`

The latter one is used to subset data obtained by a call to `read.gff` down to a given region of interest:

```
> xrange = c(50000, 1e+05)
> subdata = gffGetSubset(data, xrange)
```

The function *getAttr* was adapted from the `tilingArray` package by W. Huber and is used to extract values of attributes from character vectors as in the following example for attribute `width`:

```
> attributes = c("height= 25; width= 30; depth=40;", "depth=18;height= 21;width= 16",
+     "width= 0.45;height=-10;depth= 34")
> getAttr(attributes, "width")

[1] " 30"   " 16"   " 0.45"
```

4 Plotting Tiling Array Data on the Genomic Layout

The package provides two routines to plot tiling array data, *plot.gff* and *plot.gff.toFile*. Selfexplanatory, *plot.gff.toFile* outputs the plot into a file, which can be either of a PNG or a JPG type, whereas *plot.gff* one outputs the plot onto a standard device - screen. As any other plot, the standard device can be changed prior to calls to this function.

plot.gff is defined as follows:

`plot.gff <- function(x, xrange, coco.tresholds, expro.tresholds, par.devstage, par.anatomy, h, nlevels = 8, coco.ramp = TRUE)`

Detailed help on arguments can be found by issuing *?plot.gff* in R session after loading the package. Briefly, it requires the data structure, x to be plotted (created for example by a call to *read.gff*), plotting range as in the above example for *gffGetSubset*, named character vectors of tresholds

for tiling array (coco) and further experiments (names determine which experiments and in which order are to be plotted and the tresholds define coloring), character vectors listing development stages and anatomy to define gene coloring. Argument *h* is a numeric vector that defines proportions of plot elements in vertical direction as well as distances between those. It has a predefined value obtained by a call to *def.h*.

plot.gff.toFile has the following defition and requires all the same parameters as *plot.gff* (hidden in ... in the function definition) in addition to those that determine the parameters of the graphic file:

```
plot.gff.toFile <- function(x, file, type = "jpg", width = 1024,
height = 768, use.GDD = FALSE, ...)
```

Available types are "jpg" and "png". Argument *use.GDD* determines if gd library and `GDD` package are to be used for file output. Its default value is false meaning that standard X11 or (standard Windows device) will be used to generate file outputs. Setting this parameter to `TRUE` might be necessary if this routine is called from the shell in older version or R: X11 was not loaded if R was activated in the command line mode.

5 Further help

Start R session, load `gff3Plotter` and type `?gff3Plotter` as well as `example(gff3Plotter)`.

6 Acknowledgements

Special thanks to Sophie Grosz for further development of the package during her time at the EMBL, Heidelberg.

7 Under Development

Parts of the package modified by Sophie enabled creating image maps for plots and coding for the integration of the package into a JavaServlet/JSP web server in order to enable interactive data browsing. Because much of the code was modified it requires extensive structuring and updating and will be added to the released package shortly. Interfaces of existing functions will not be changed (but extended by some ommittable parameters).