browser_plot.R Tutorial

This is a brief tutorial on how to use the function “browser_plot.R” to create publication quality “browser shots” from mixed types of signal.

Loading function and packages

```r
# Provide the path to the script, loads the function "browser_plotter"
source("browser_plot.R")

# Dependencies - these are all automatically loaded by the function, but are
#   listed here as well
library(BRGenomics)
library(ggplot2)
library(facetscales)
library(gtable)
library(grid)
library(readr)
library(purrr)
library(dplyr)
```

Basic use case

This section illustrates how to create a browser plot object from different types of genomic data

Loading lists of different classes of signal

Loading PRO-seq data

```r
PROseq.lst <- list(
    "LACZ_PROseq" = import_bigWig(
        paste0(prefix, "bw/PROseq/merged_normed/LACZ_PROseq_fwd.bw"),
        paste0(prefix, "bw/PROseq/merged_normed/LACZ_PROseq_rev.bw"),
        genome = "dm6"
    ),
    "GAF_PROseq" = import_bigWig(
        paste0(prefix, "bw/PROseq/merged_normed/GAF_PROseq_fwd.bw"),
        paste0(prefix, "bw/PROseq/merged_normed/GAF_PROseq_rev.bw"),
        genome = "dm6"
    )
)
```

PRO-seq data was loaded using BRGenomics::import_bigWig, so data is already single-width intervals and stranded

Loading ATAC-seq data

```
ATACseq.lst <- list(
    "LACZ_ATACseq" = import.bw(
        paste0(prefix, "bw/ATACseq/DHS/merged_normed/LACZ_ATACDHS.bw")),
    "GAF_ATACseq" = import.bw(
        paste0(prefix, "bw/ATACseq/DHS/merged_normed/GAF_ATACDHS.bw"))
)
```

ATAC-seq data was loaded using rtracklayer::import.bw, so data is run-compressed and unstranded

Loading ChIP-seq data

```
ChIPseq.lst <- list(
    "GAF_ChIPseq" = import.bw(
        paste0(prefix, "bw/ChIPseq/GAF_ChIPseq.bw")
    )
)
```

ChIP-seq data was loaded using rtracklayer::import.bw, so data is run-compressed and unstranded

Loading gene list

```
genes.gr <- makeGRangesFromDataFrame(
    read_tsv(paste0(prefix, "bed/filtered_dm6_genes.bed"),
             col_names = c("chr", "start", "end", "gene_name", "tx_name", "strand")),
    keep.extra.columns = TRUE
)
```

Combining all data sets into a list of lists

Each sublist of dataset.grlist must be either all stranded or all unstranded. Each sublist will be group-autoscaled in the final output

```
dataset.grlist <- list(
    "PROseq" = PROseq.lst,
    "ATACseq" = ATACseq.lst,
    "ChIPseq" = ChIPseq.lst
)
```

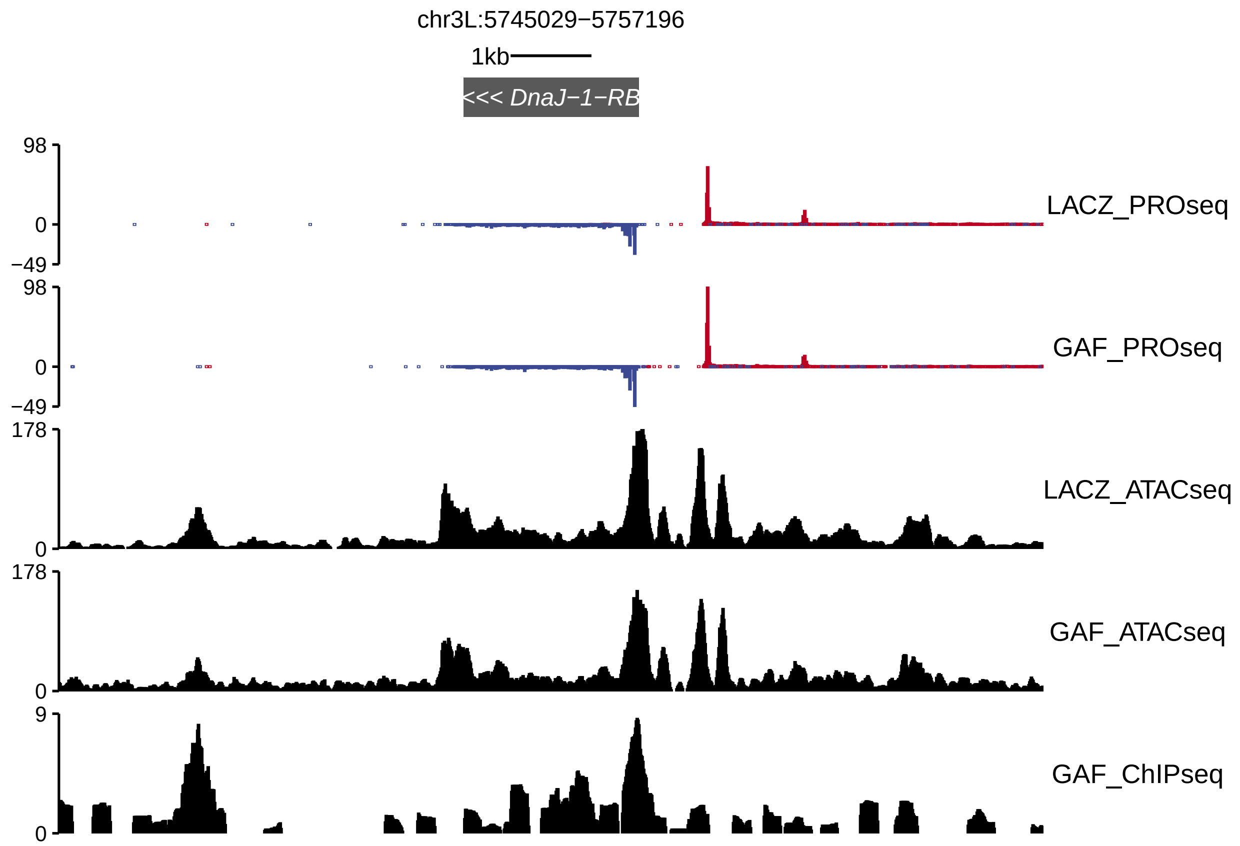
Filtering genelist to a single gene of interest

```
# Plotting DnaJ-1-RB
regions.gr <- genes.gr[which(genes.gr$tx_name == "DnaJ-1-RB")]
```

Constructing plot

```r
DNAJ1 <- browser_plotter(
    regions.gr = regions.gr, # Provide GRanges containing only gene of interest
    dataset.grlist = dataset.grlist, # Provide list of lists of signal
    binsize = 10, # Set size for binning
    bin_FUN = mean, # function for summarizing signal in bins
    pad_left = 5000, # distance in bp to add on either side of gene
    pad_right = 5000,
    scale_bar_size = 1000, # size of scale bar in bp
    color_plus = "#BB0021", # colors for different data types
    color_minus = "#3B4992",
    color_unstranded = "black",
    tx_name = "tx_name" # Field in your regions.gr that contains the gene or transcript name
)

# The resulting object is a gtable object, and can be rendered by grid.draw()
grid.newpage()
grid.draw(DNAJ1)
```



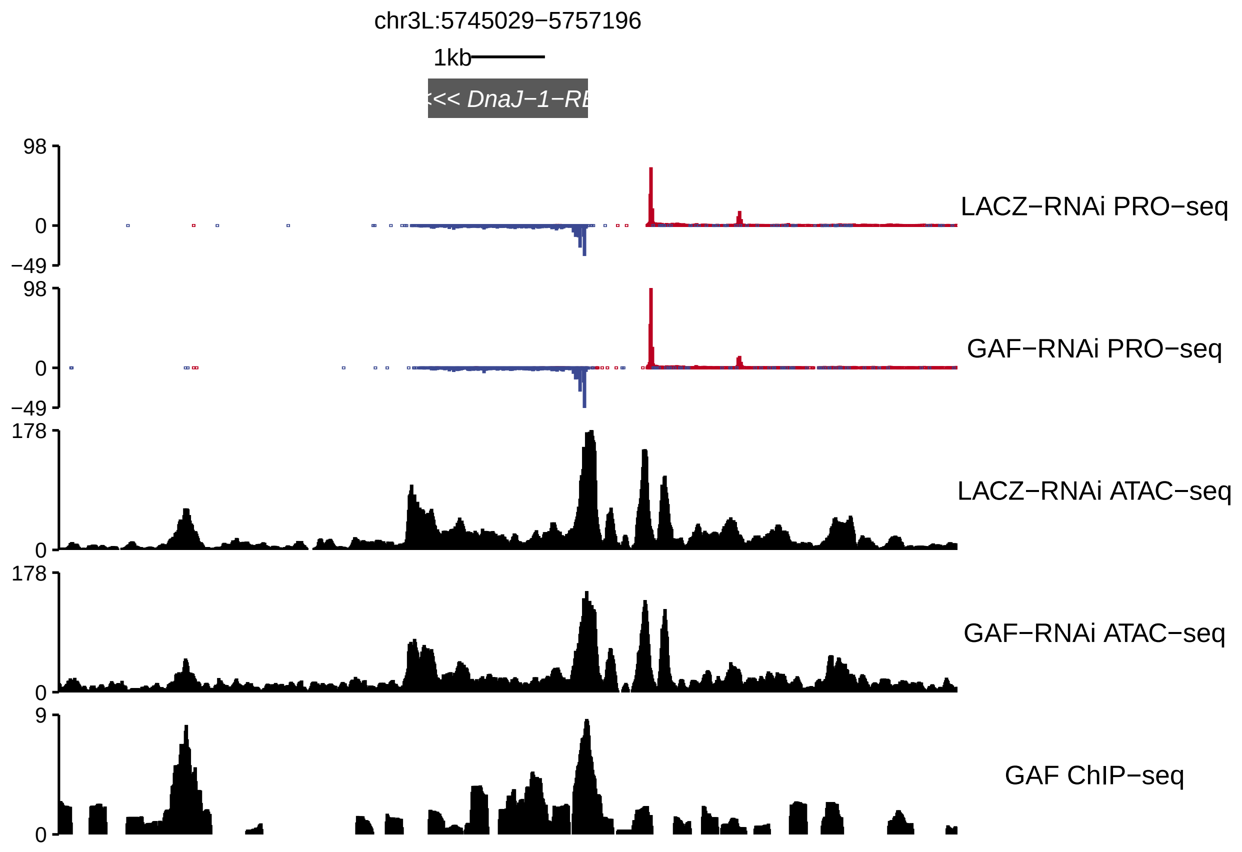
Adding custom labels

By default, the labels are the names used in your lists of signal. The .labs argument allows you to change this.

```r
# .labs is a character vector of labels in the order they appear in dataset.grlist

# Creating vector of labels
labs <- c("LACZ-RNAi PRO-seq", "GAF-RNAi PRO-seq",
          "LACZ-RNAi ATAC-seq", "GAF-RNAi ATAC-seq",
          "GAF ChIP-seq")

# Plotting
DNAJ1 <- browser_plotter(
    regions.gr = regions.gr, # Provide GRanges containing only gene of interest
    dataset.grlist = dataset.grlist, # Provide list of lists of signal
    binsize = 10, # Set size for binning
    bin_FUN = mean, # function for summarizing signal in bins
    pad_left = 5000, # distance in bp to add on either side of gene
    pad_right = 5000,
    scale_bar_size = 1000, # size of scale bar in bp
    color_plus = "#BB0021", # colors for different data types
    color_minus = "#3B4992",
    color_unstranded = "black",
    tx_name = "tx_name", # Field in your regions.gr that contains the gene or transcript name
    .labs = labs
)
grid.newpage()
grid.draw(DNAJ1)
```



Increasing speed by passing .expand_ranges argument

Because the data in these plots is often a mixture of run-length compressed signal (ATAC-seq and ChIP-seq in this example) and single-width signal (PRO-seq in this example), data must all be coerced to single-width before getting signal counts. BRGenomics::getCountsByPositions offers an efficient way of expanding signal using the "expand_ranges" argument. By default, this is enabled for all samples in browser_plotter. However, this slows the function because single-width data does not actually need to be expanded, but is by default. The ".expand_ranges" argument of browser_plotter accepts a named list of logical values indicating whether getCountsByPositions should expand run-lenth compressed data for each list element of dataset.grlist. It is always safe to use the default setting, but specifying this argument can speed up the function.

```r
# Named list of logicals, with one entry for each sublist of dataset.grlist
.expand_ranges <- list(
    "PROseq" = FALSE, # PRO-seq is already single width
    "ATACseq" = TRUE, # ATAC-seq and RNA-seq are run-compressed
    "ChIPseq" = TRUE
)

# Timing function with or without .expand_ranges
library(tictoc)

tic()
DNAJ1 <- browser_plotter(
    regions.gr = regions.gr, # Provide GRanges containing only gene of interest
    dataset.grlist = dataset.grlist, # Provide list of lists of signal
    binsize = 10, # Set size for binning
    bin_FUN = mean, # function for summarizing signal in bins
    pad_left = 5000, # distance in bp to add on either side of gene
    pad_right = 5000,
    scale_bar_size = 1000, # size of scale bar in bp
    color_plus = "#BB0021", # colors for different data types
    color_minus = "#3B4992",
    color_unstranded = "black",
    tx_name = "tx_name", # Field in your regions.gr that contains the gene or transcript name
    .labs = labs
)
toc()
## 9.859 sec elapsed

tic()
DNAJ1 <- browser_plotter(
    regions.gr = regions.gr, # Provide GRanges containing only gene of interest
    dataset.grlist = dataset.grlist, # Provide list of lists of signal
    binsize = 10, # Set size for binning
    bin_FUN = mean, # function for summarizing signal in bins
    pad_left = 5000, # distance in bp to add on either side of gene
    pad_right = 5000,
    scale_bar_size = 1000, # size of scale bar in bp
    color_plus = "#BB0021", # colors for different data types
    color_minus = "#3B4992",
    color_unstranded = "black",
    tx_name = "tx_name", # Field in your regions.gr that contains the gene or transcript name
    .labs = labs,
    .expand_ranges = .expand_ranges # adding .expand_ranges argument
)
toc()
## 8.129 sec elapsed
```

The output in either case is idential, but providing .expand_ranges here reduces the time required