

# timeViz: Timeline Visualization User Guide

Seth Babin, Kianna Blount, Ivan De Dios, Travis Jensen, Mariam Saleem, Ashley Stinger

November 01, 2021

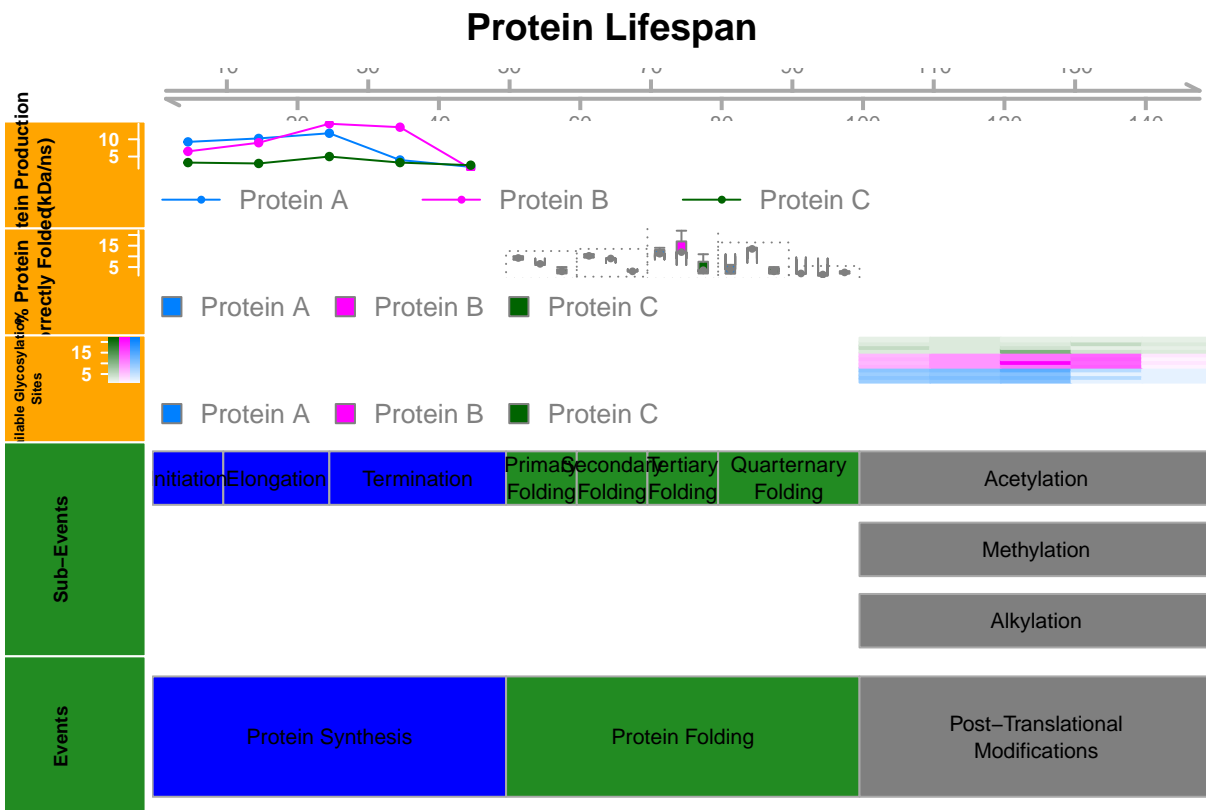
## timeViz

timeViz is an R-based software derived from a popular track-based genomic visualization package called Gviz (Hahne & Ivanek, 2016).

This powerful tool will allow users to supply global configuration options and data to plot annotation or data tracks above or below one or more timelines. The user can control the labels, coloring, grouping, shapes, and sizes for several items in each of these tracks.

The following user guide includes basic features of the timeViz tool, example visualizations, and corresponding code.

## Visualization Plot: Example 1



## Visualization Code: Example 1

```
## import XLSX sheet 1 (configuration)
config = read.xlsx("TimevizConfig.xlsx", sheet = "Configuration")

## parse out.file =
## config$Value[config$Variable.Name=='Output PDF']
out.height = as.numeric(config$Value[config$Variable.Name ==
  "Output Height"])
out.width = as.numeric(config$Value[config$Variable.Name ==
  "Output Width"])
main = config$Value[config$Variable.Name == "Main Title"]
main.size = as.numeric(config$Value[config$Variable.Name ==
  "Main Font Size"])
track.width = as.numeric(config$Value[config$Variable.Name ==
  "Track Box Width"])
from.to = as.numeric(unlist(str_split(config$Value[config$Variable.Name ==
  "From To"], ";")))
track.sheet.names = unlist(str_split(config$Value[config$Variable.Name ==
  "Track List"], ";"))
track.types = unlist(str_split(config$Value[config$Variable.Name ==
  "Track Type"], ";"))
track.heights = as.numeric(unlist(str_split(config$Value[config$Variable.Name ==
  "Track Heights"], ";")))
track.names = unlist(str_split(config$Value[config$Variable.Name ==
  "Track Names"], ";"))
track.box.colors = unlist(str_split(config$Value[config$Variable.Name ==
  "Track Box Color"], ";"))
track.bg.color = unlist(str_split(config$Value[config$Variable.Name ==
  "Track Background Color"], ";"))
track.label.colors = unlist(str_split(config$Value[config$Variable.Name ==
  "Track Label Color"], ";"))
track.label.sizes = as.numeric(unlist(str_split(config$Value[config$Variable.Name ==
  "Track Label Size"], ";")))
data.types = unlist(str_split(config$Value[config$Variable.Name ==
  "Data Type"], ";"))
data.groups = unlist(str_split(config$Value[config$Variable.Name ==
  "Data Groups"], ";"))
data.aggs = unlist(str_split(config$Value[config$Variable.Name ==
  "Data Aggregate"], ";"))

## initilize plot list object and other variables
plot.list = list()
data.count = 1

## for each track do
for (i in 1:length(track.types)) {
  track.type = track.types[i]
  track.sheet.name = track.sheet.names[i]
  track.name = gsub("\\\\n", "\n", track.names[i])
  track.box.color = track.box.colors[i]
  track.label.color = track.label.colors[i]
  track.label.size = track.label.sizes[i]
```

```

## determine track type and append track to list
## object
if (track.type == "time") {

  ## initialize an axis track and add to plot list
  ## object
  plot.list[[i]] = GenomeAxisTrack(name = track.name,
    background.title = track.box.color, background.panel = track.bg.color,
    fontcolor.title = track.label.color, cex.title = track.label.size)

  ## Data track
} else if (track.type == "data") {
  data.type = data.types[data.count]
  data.group = unlist(str_split(data.groups[data.count],
    ","))
  data.agg = data.aggs[data.count]

  ## Import sheet to get label
  config.data = read.xlsx("TimevizConfig.xlsx",
    sheet = track.sheet.name)

  ## determine starts and stops
  starts = as.numeric(unlist(lapply(str_split(config.data$Time.Ranges,
    ";"), function(x) {
    x[1]
  })))
  ends = as.numeric(unlist(lapply(str_split(config.data$Time.Ranges,
    ";"), function(x) {
    x[2]
  })))

  ## pull in a dummy data-set from the examples in the
  ## user guide
  config.data.granges = GRanges(seqnames = "chrX",
    strand = rep("*", length(starts)), ranges = IRanges(start = starts,
    end = ends), mcols = config.data[,
    2:ncol(config.data)])

  ## Is there any grouping?

  if (any(nchar(data.group) == 0)) {
    ## plot a data track without grouping
    dTrack <- DataTrack(config.data.granges,
      name = track.name, type = data.type,
      background.title = track.box.color,
      background.panel = track.bg.color,
      fontcolor.title = track.label.color,
      cex.title = track.label.size)
  } else {

    ## Aggregate data on the mean?
    if (data.agg == "NULL") {

```

```

    ## plot a data track with grouping no aggregate
    plot.list[[i]] = DataTrack(config.data.granges,
                               name = track.name, groups = data.group,
                               type = data.type, background.title = track.box.color,
                               background.panel = track.bg.color,
                               fontcolor.title = track.label.color,
                               cex.title = track.label.size)
  } else {
    ## plot a data track with and aggregate
    plot.list[[i]] = DataTrack(config.data.granges,
                               name = track.name, groups = data.group,
                               type = data.type, aggregateGroups = TRUE,
                               background.title = track.box.color,
                               background.panel = track.bg.color,
                               fontcolor.title = track.label.color,
                               cex.title = track.label.size)
  }
}

## advance counter
data.count = data.count + 1

## annotation Track
} else if (track.type == "annotation") {

  ## Import sheet to get label
  config.annot = read.xlsx("TimevizConfig.xlsx",
                           sheet = track.sheet.name)

  ## determine starts and stops
  starts = as.numeric(unlist(lapply(str_split(config.annot$Time.Ranges,
                                             ";"), function(x) {
    x[1]
  })))
  ends = as.numeric(unlist(lapply(str_split(config.annot$Time.Ranges,
                                             ";"), function(x) {
    x[2]
  })))

  ## get grouping factor
  group.factor = as.numeric(as.factor(config.annot$Annotation.Name))

  ## plot an annotation track plot annotation label
  ## above/below/on box,
  plot.list[[i]] = AnnotationTrack(start = starts,
                                    end = ends, chromosome = "chrX", strand = rep("*",
                                    length(starts)), id = gsub("\\\\n",
                                    "\n", config.annot$Annotation.Name),
                                    name = track.name, shape = "box", featureAnnotation = "id",
                                    group = group.factor, stacking = "squish",
                                    fontcolor.feature = config.annot$Annotation.Label.Color,
                                    cex.feature = config.annot$Annotation.Label.Size,
                                    fill = config.annot$Annotation.Color, background.title = track.box.color,

```

```

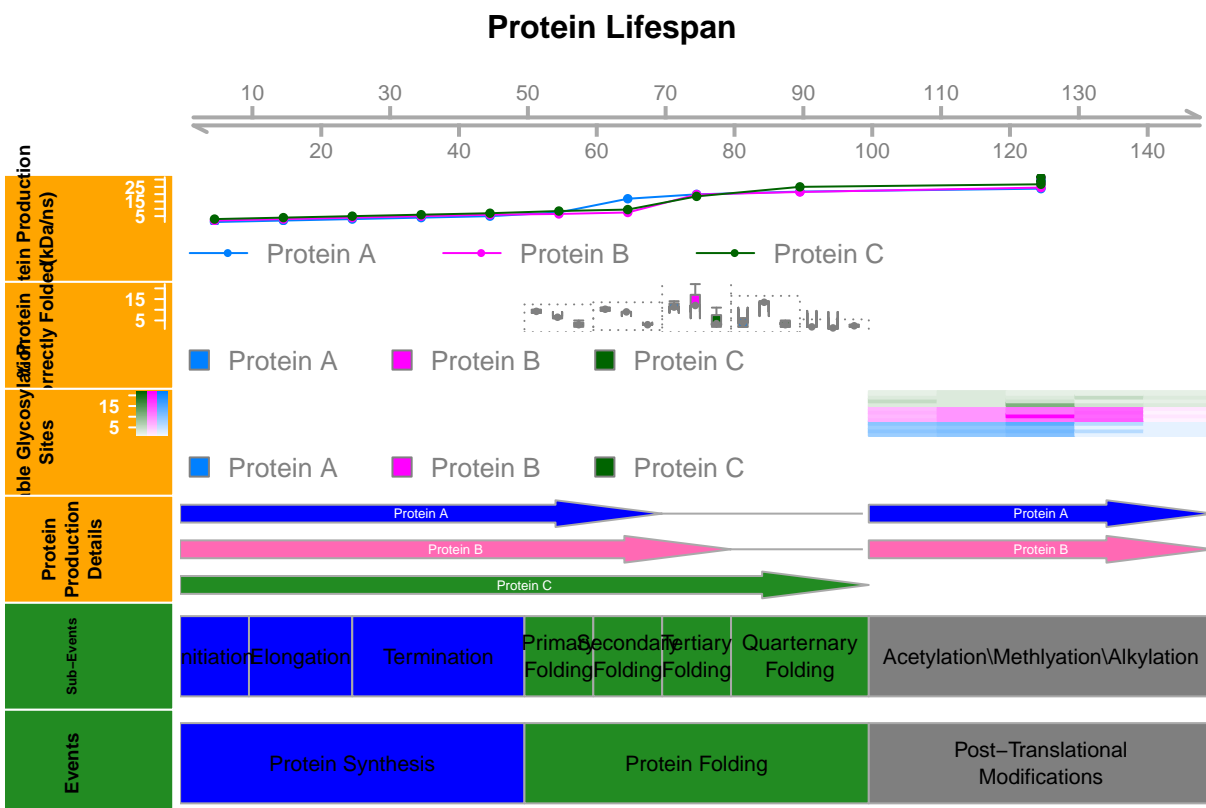
background.panel = track.bg.color, fontcolor.title = track.label.color,
cex.title = track.label.size)

} else {
  ## invalid input track type -- exit with non-zero
  ## status
  print(paste0("Please provide valid track type.
               The given track type is not valid: ",
               track.type))
  quit(status = 3) ## non 0 exit status
}
}

## Uncomment code below to plot tracks
## plotTracks(plot.list, from = from.to[1], to =
## from.to[2],
## sizes=track.heights,main=main,cex.main=main.size,
## title.width=track.width)

```

## Visualization Plot: Example 2



## Visualization Code: Example 2

```
## set working directory
infile.config = "FinalTimevizConfig.xlsx"

##### PARSE Configuration

## import XLSX sheet 1 (configuration)
config = read.xlsx(infile.config, sheet = "Configuration")

## parse out.file =
## config$Value[config$Variable.Name=='Output PDF']
out.height = as.numeric(config$Value[config$Variable.Name ==
  "Output Height"])
out.width = as.numeric(config$Value[config$Variable.Name ==
  "Output Width"])
main = config$Value[config$Variable.Name == "Main Title"]
main.size = as.numeric(config$Value[config$Variable.Name ==
  "Main Font Size"])
track.width = as.numeric(config$Value[config$Variable.Name ==
  "Track Box Width"])
from.to = as.numeric(unlist(str_split(config$Value[config$Variable.Name ==
  "From To"], ";")))
track.sheet.names = unlist(str_split(config$Value[config$Variable.Name ==
  "Track List"], ";"))
track.types = unlist(str_split(config$Value[config$Variable.Name ==
  "Track Type"], ";"))
track.heights = as.numeric(unlist(str_split(config$Value[config$Variable.Name ==
  "Track Heights"], ";")))
track.names = unlist(str_split(config$Value[config$Variable.Name ==
  "Track Names"], ";"))
track.box.colors = unlist(str_split(config$Value[config$Variable.Name ==
  "Track Box Color"], ";"))
track.bg.color = unlist(str_split(config$Value[config$Variable.Name ==
  "Track Background Color"], ";"))
track.label.colors = unlist(str_split(config$Value[config$Variable.Name ==
  "Track Label Color"], ";"))
track.label.sizes = as.numeric(unlist(str_split(config$Value[config$Variable.Name ==
  "Track Label Size"], ";")))
data.types = unlist(str_split(config$Value[config$Variable.Name ==
  "Data Type"], ";"))
data.groups = unlist(str_split(config$Value[config$Variable.Name ==
  "Data Groups"], ";"))
data.aggs = unlist(str_split(config$Value[config$Variable.Name ==
  "Data Aggregate"], ";"))
data.group.details = unlist(str_split(config$Value[config$Variable.Name ==
  "Data Group Details"], ";"))
data.details.name = unlist(str_split(config$Value[config$Variable.Name ==
  "Data Details Name"], ";"))
details.types = unlist(str_split(config$Value[config$Variable.Name ==
  "Detail Type"], ";"))

##### Prepare Desired Tracks
```

```

## initilize plot list object and other variables
plot.list = list()
data.count = 1

## for each track do
for (i in 1:length(track.types)) {
  track.type = track.types[i]
  track.sheet.name = track.sheet.names[i]
  track.name = gsub("\\\\n", "\n", track.names[i])
  track.box.color = track.box.colors[i]
  track.label.color = track.label.colors[i]
  track.label.size = track.label.sizes[i]

  ## determine track type and append track to list
  ## object
  if (track.type == "time") {

    ## initialize an axis track and add to plot list
    ## object wishlist: add ticksAt (minor/major ticks
    ## based on time units), figure out how to add the
    ## title - currently not working
    plot.list[[i]] = GenomeAxisTrack(name = track.name,
      background.title = track.box.color, background.panel = track.bg.color,
      fontcolor.title = track.label.color, cex.title = track.label.size)

    ## Data track
  } else if (track.type == "data") {
    data.type = data.types[data.count]
    data.group = unlist(str_split(data.groups[data.count],
      ","))
    data.agg = data.aggs[data.count]

    ## Import sheet to get label
    config.data = read.xlsx(infile.config, sheet = track.sheet.name)

    ## determine starts and stops
    starts = as.numeric(unlist(lapply(str_split(config.data$Time.Ranges,
      ";"), function(x) {
        x[1]
      })))
    ends = as.numeric(unlist(lapply(str_split(config.data$Time.Ranges,
      ";"), function(x) {
        x[2]
      })))

    ## pull in a dummy data-set from the examples in the
    ## user guide
    config.data.granges = GRanges(seqnames = "chrX",
      strand = rep("*", length(starts)), ranges = IRanges(start = starts,
        end = ends), mcols = config.data[,
        2:ncol(config.data)])

    ## Is there any grouping? wishlist: Add group

```

```

## specific color coding, color coding for y-axis
## (default to white right now)
if (any(nchar(data.group) == 0)) {
  ## plot a data track without grouping
  dTrack <- DataTrack(config.data.granges,
    name = track.name, type = data.type,
    background.title = track.box.color,
    background.panel = track.bg.color,
    fontcolor.title = track.label.color,
    cex.title = track.label.size)

} else {

  ## Aggregate data on the mean?
  if (data.agg == "NULL") {
    ## plot a data track with grouping no aggregate
    plot.list[[i]] = DataTrack(config.data.granges,
      name = track.name, groups = data.group,
      type = data.type, background.title = track.box.color,
      background.panel = track.bg.color,
      fontcolor.title = track.label.color,
      cex.title = track.label.size)
  } else {
    ## plot a data track with and aggregate
    plot.list[[i]] = DataTrack(config.data.granges,
      name = track.name, groups = data.group,
      type = data.type, aggregateGroups = TRUE,
      background.title = track.box.color,
      background.panel = track.bg.color,
      fontcolor.title = track.label.color,
      cex.title = track.label.size)
  }
}

## advance counter
data.count = data.count + 1

## details annotation

} else if (track.type == "details") {

  ## Import sheet to get label
  config.details = read.xlsx(infile.config, sheet = track.sheet.name)

  ## determine starts and stops
  starts = as.numeric(unlist(lapply(str_split(config.details$Time.Ranges,
    ";"), function(x) {
    x[1]
  })))
  ends = as.numeric(unlist(lapply(str_split(config.details$Time.Ranges,
    ";"), function(x) {
    x[2]
  })))
}

```



```

## get grouping factor
group.factor = as.numeric(as.factor(config.details$Details.Name))

## plot an annotation track
plot.list[[i]] = AnnotationTrack(start = starts,
  end = ends, chromosome = "chrX", strand = rep("+",
    length(starts)), id = gsub("\\\\n",
    "\\n", config.details$Details.Name),
  name = track.name, shape = "arrow", featureAnnotation = "id",
  group = group.factor, stacking = "squish",
  collapse = TRUE, fontcolor.feature = config.details$Details.Label.Color,
  cex.feature = config.details$Details.Label.Size,
  fill = config.details$Details.Color, background.title = track.box.color,
  background.panel = track.bg.color, fontcolor.title = track.label.color,
  cex.title = track.label.size)

## annotation Track
} else if (track.type == "annotation") {

  ## Import sheet to get label
  config.annot = read.xlsx(infile.config, sheet = track.sheet.name)

  ## determine starts and stops
  starts = as.numeric(unlist(lapply(str_split(config.annot$Time.Ranges,
    ";"), function(x) {
    x[1]
  })))
  ends = as.numeric(unlist(lapply(str_split(config.annot$Time.Ranges,
    ";"), function(x) {
    x[2]
  })))

  ## get grouping factor
  group.factor = as.numeric(as.factor(config.annot$Annotation.Name))

  ## plot an annotation track
  plot.list[[i]] = AnnotationTrack(start = starts,
    end = ends, chromosome = "chrX", strand = rep("*",
      length(starts)), id = gsub("\\\\n",
      "\\n", config.annot$Annotation.Name),
    name = track.name, shape = "box", featureAnnotation = "id",
    group = group.factor, stacking = "squish",
    fontcolor.feature = config.annot$Annotation.Label.Color,
    cex.feature = config.annot$Annotation.Label.Size,
    fill = config.annot$Annotation.Color, background.title = track.box.color,
    background.panel = track.bg.color, fontcolor.title = track.label.color,
    cex.title = track.label.size)

} else {
  ## invalid input track type -- exit with non-zero
  ## status

```

```

        print(paste0("Please provide valid track type. The given track type is not valid: ",
            track.type))
        quit(status = 3) ## non 0 exit status
    }
}

##### Prepare Desired Tracks

# pdf(out.file,height=out.height,width=out.width)

# Uncomment code below to plot tracks
# plotTracks(plot.list, from = from.to[1], to =
# from.to[2],
# sizes=track.heights,main=main,cex.main=main.size,
# title.width=track.width)

# dev.off()

```

## Time Track Parameters

| Parameter          | Description  |
|--------------------|--|
| out.file           | File location and name for output PDF  |
| out.height         | Height of the PDF in inches. Default = 8   |
| out.width          | Height of the PDF in inches. Default = 8   |
| main               | Main title for the plot (top). Default = No Title  |
| main.size          | main title font size (expansion factor). Default = 1   |
| track.width        | An expansion factor for the width of the track boxes that hold the track labels. Default = 1   |
| from.to            | Start position of tracks to end position of tracks in time unit. Can be used to plot only a subset of the total data. Semicolon separated.. Default=min(data);max(data)                                |
| track.sheet.names  | list of tracks (and the order of the tracks to plot top to bottom). Use the sheet names in this spreadsheet to list tracks. Semicolon separated.   |
| track.types        | list of track types - one for each track. Semicolon separated. Options include: data, annotation and time.   |
| track.heights      | heights for each track relative to the sum across tracks (ie: 1;3;3 = 1/7th of total height;3/7ths of total height;3/7ths of total height). Semicolon separated. Default = all track heights are equal |
| track.names        | Names to be printed for each track. Semicolon separated. Defaults = sheet names in this spreadsheet  |
| track.box.colors   | The color of the box the track title is printed inside of. Semicolon separated. Default = grey   |
| track.bg.color     | The color of the track background.   |
| track.label.colors | The color of the track label. Semicolon separated. Default = black   |
| track.label.sizes  | The font size (expansion factor) for each title label. Semicolon separated. Default = 1  |
| data.types         | For data tracks only, report the type of desired plot. Choose from the following: . Semicolon separated.   |
| data.groups        | For data tracks only, report the way samples should be grouped. Semicolon and comma separated. If empty, no grouping performed   |
| data.aggs          | Should the data be aggregated on the mean? (only if groups are supplied). Semicolon separated. Default = NULL  |

## Data Track Parameters

| Parameter        | Description  |
|------------------|--|
| aggregateGroups  | Logical scalar. Aggregate the values within a sample group using the aggregation function specified in the aggregationparameter.   |
| aggregation      | Function or character scalar. Used to aggregate values in windows or for collapsing overlapping items. The function has to accept a numeric vector as a single input parameter and has to return a numeric scalar with the aggregated value. Alternatively, one of the predefined options mean, median sum, min, max or extreme can be supplied as a character scalar. Defaults to mean. |
| alpha.confint    | Numeric scalar. The transparency for the confidence intervals in confint-type plots.   |
| amount           | Numeric scalar. Amount of jittering in xy-type plots.<br>See panel.xyplot for details.   |
| baseline         | Numeric scalar. Y-axis position of an optional baseline. This parameter has a special meaning for mountain-type and polygon-type plots, see the ‘Details’ section in DataTrack for more information.   |
| box.legend       | Logical scalar. Draw a box around a legend.  |
| box.ratio        | Numeric scalar. Parameter controlling the boxplot appearance.<br>See panel.bwplot for details.   |
| box.width        | Numeric scalar. Parameter controlling the boxplot appearance.<br>See panel.bwplot for details.   |
| cex              | Numeric scalar. The default pixel size for plotting symbols.   |
| cex.legend       | Numeric scalar. The size factor for the legend text.   |
| cex.sampleNames  | Numeric scalar. The size factor for the sample names text in heatmap or horizon plots. Defaults to an automatic setting.   |
| coef             | Numeric scalar. Parameter controlling the boxplot appearance.<br>See panel.bwplot for details.   |
| col              | Character or integer vector. The color used for all line and symbol elements, unless there is a more specific control defined elsewhere. Unless groups are specified, only the first color in the vector is usually regarded.  |
| col.baseline     | Character scalar. Color for the optional baseline, defaults to the setting of col.   |
| col.boxplotFrame | Character scalar. Line color of the frame around grouped boxplots.   |
| col.confint      | Character vector. Border colors for the confidence intervals for confint-type plots.   |
| col.histogram    | Character scalar. Line color in histogram-type plots.  |
| col.horizon      | The line color for the segments in the horizon-type plot.<br>See horizonplot for details.  |
| col.mountain     | Character scalar. Line color in mountain-type and polygon-type plots, defaults to the setting of col.  |
| col.sampleNames  | Character or integer scalar. The color used for the sample names in heatmap plots.   |
| collapse         | Logical scalar. Collapse overlapping ranges and aggregate the underlying data.   |
| degree           | Numeric scalar. Parameter controlling the loess calculation for smooth and mountain-type plots. See panel.loess for details.   |

(continued)

| Parameter         | Description   |
|-------------------|---|
| do.out            | Logical scalar. Parameter controlling the boxplot appearance. See <code>panel.bwplot</code> for details.  |
| evaluation        | Numeric scalar. Parameter controlling the loess calculation for smooth and mountain-type plots. See <code>panel.loess</code> for details.   |
| factor            | Numeric scalar. Factor to control amount of jittering in xy-type plots. See <code>panel.xyplot</code> for details.  |
| family            | Character scalar. Parameter controlling the loess calculation for smooth and mountain-type plots. See <code>panel.loess</code> for details.   |
| fill.confint      | Character vector. Fill colors for the confidence intervals for confint-type plots.  |
| fill.histogram    | Character scalar. Fill color in histogram-type plots, defaults to the setting of <code>fill</code> .  |
| fill.horizon      | The fill colors for the segments in the <code>horizon</code> -type plot. This should be a vector of length six, where the first three entries are the colors for positive changes, and the latter three entries are the colors for negative changes. Defaults to a red-blue color scheme. See <code>horizonplot</code> for details.   |
| fill.mountain     | Character vector of length 2. Fill color in mountain-type and polygon-type plots.   |
| fontcolor.legend  | Integer or character scalar. The font color for the legend text.  |
| fontface.legend   | Integer or character scalar. The font face for the legend text.   |
| fontfamily.legend | Integer or character scalar. The font family for the legend text.   |
| fontsize.legend   | Numeric scalar. The pixel size for the legend text.   |
| gradient          | Character vector. The base colors for the <code>gradient</code> plotting type or the <code>heatmap</code> type with a single group. When plotting heatmaps with more than one group, the <code>col</code> parameter can be used to control the group color scheme, however the gradient will always be from white to ‘col’ and thus does not offer as much flexibility as this <code>gradient</code> parameter. |
| grid              | Logical vector. Draw a line grid under the track content.   |
| groups            | Vector coercable to a factor. Optional sample grouping. See ‘Details’ section in <code>DataTrack</code> for further information.  |
| horizon.origin    | The baseline relative to which changes are indicated on the <code>horizon</code> -type plot. See <code>horizonplot</code> for details.  |
| horizon.scale     | The scale for each of the segments in the <code>horizon</code> -type plot. Defaults to 1/3 of the absolute data range. See <code>horizonplot</code> for details.  |
| jitter.x          | Logical scalar. Toggle on jittering on the x axis in xy-type plots. See <code>panel.xyplot</code> for details.  |
| jitter.y          | Logical scalar. Toggle off jittering on the y axis in xy-type plots. See <code>panel.xyplot</code> for details.   |
| legend            | Boolean triggering the addition of a legend to the track to indicate groups. This only has an effect if at least two groups are present.  |
| levels.fos        | Numeric scalar. Parameter controlling the boxplot appearance. See <code>panel.bwplot</code> for details.  |
| lineheight.legend | Numeric scalar. The line height for the legend text.  |
| lty.baseline      | Character or numeric scalar. Line type of the optional baseline, defaults to the setting of <code>lty</code> .  |
| lty.mountain      | Character or numeric scalar. Line type in mountain-type and polygon-type plots, defaults to the setting of <code>lty</code> .   |

(continued)

| Parameter       | Description   |
|-----------------|---|
| lwd.baseline    | Numeric scalar. Line width of the optional baseline, defaults to the setting of lwd.  |
| lwd.mountain    | Numeric scalar. Line width in mountain-type and polygon-type plots, defaults to the setting of lwd.   |
| min.distance    | Numeric scalar. The minimum distance in pixel below which to collapse ranges.   |
| missingAsZero   | Logical scalar. Defines how the missing values are treated in the aggregation procedure with running window. Setting it to TRUE fills empty positions with zeros, which is default. FALSE fills empty positions with NA.  |
| na.rm           | Boolean controlling whether to discard all NA values when plotting or to keep empty spaces for NAs  |
| ncolor          | Integer scalar. The number of colors for the ‘gradient’ plotting type   |
| notch           | Logical scalar. Parameter controlling the boxplot appearance. See panel.bwplot for details.   |
| notch.frac      | Numeric scalar. Parameter controlling the boxplot appearance. See panel.bwplot for details.   |
| pch             | Integer scalar. The type of glyph used for plotting symbols.  |
| separator       | Numeric scalar. Number of pixels used to separate individual samples in heatmap- and horizon-type plots.  |
| showColorBar    | Boolean. Indicate the data range color mapping in the axis for ‘heatmap’ or ‘gradient’ types.   |
| showSampleNames | Boolean. Display the names of the individual samples in a heatmap or a horizon plot.  |
| span            | Numeric scalar. Parameter controlling the loess calculation for smooth and mountain-type plots. See panel.loess for details.  |
| stackedBars     | Logical scalar. When there are several data groups, draw the histogram-type plots as stacked barplots or grouped side by side.  |
| stats           | Function. Parameter controlling the boxplot appearance. See panel.bwplot for details.   |
| transformation  | Function. Applied to the data matrix prior to plotting or when calling the score method. The function should accept exactly one input argument and its return value needs to be a numeric vector which can be coerced back into a data matrix of identical dimensionality as the input data.  |
| type            | Character vector. The plot type, one or several in p,l, b, a, a_confint, s, g, r, S, confint, smooth, histogram, mountain, polygon, h, bo<br>See ‘Details’ section in DataTrack for more information on the individual plotting types.  |
| varwidth        | Logical scalar. Parameter controlling the boxplot appearance. See panel.bwplot for details.   |
| window          | Numeric or character scalar. Aggregate the rows values of the data matrix to window equally sized slices on the data range using the method defined in aggregation. If negative, apply a running window of size windowSize using the same aggregation method. Alternatively, the special value auto causes the function to determine the optimal window size to avoid overplotting, and fixed uses fixed-size windows of size windowSize. |
| windowSize      | Numeric scalar. The size of the running window when the value of window is negative.  |

*(continued)*

| Parameter | Description  |
|-----------|--|
| yTicksAt  | Numeric vector. The points at which y-axis tick-marks are to be drawn. By default, when NULL, tickmark locations are computed. |
| ylim      | Numeric vector of length 2. The range of the y-axis scale.   |

## Annotation Track Parameters

| Parameter         | Description  |
|-------------------|--|
| arrowHeadMaxWidth | Numeric scalar. The maximum width of the arrow head in pixels if shape is arrow.   |
| arrowHeadWidth    | Numeric scalar. The width of the arrow head in pixels if shape is fixedArrow.  |
| cex               | Numeric scalar. The font expansion factor for item identifiers.  |
| cex.group         | Numeric scalar. The font expansion factor for the group-level annotation.  |
| col               | Character or integer scalar. The border color for all track items.   |
| col.line          | Character scalar. The color used for connecting lines between grouped items. Defaults to a light gray, but if set to NULL the same color as for the first item in the group is used.   |
| featureAnnotation | Character scalar. Add annotation information to the individual track elements. This can be a value in id, group or feature. Defaults to id. Only works if showFeatureId is not FALSE.  |
| fill              | Character or integer scalar. The fill color for untyped items. This is also used to connect grouped items. See grouping for details.   |
| fontcolor.group   | Character or integer scalar. The font color for the group-level annotation.  |
| fontcolor.item    | Character or integer scalar. The font color for item identifiers.  |
| fontface.group    | Numeric scalar. The font face for the group-level annotation.  |
| fontfamily.group  | Character scalar. The font family for the group-level annotation.  |
| fontsize.group    | Numeric scalar. The font size for the group-level annotation.  |
| groupAnnotation   | Character scalar. Add annotation information as group labels. This can be a value in id, group or feature. Defaults to group. Only works if showId is not FALSE.   |
| just.group        | Character scalar. the justification of group labels. Either left, right, above or below.   |
| lex               | Numeric scalar. The line expansion factor for all track items. This is also used to connect grouped items. See grouping for details.   |
| lineheight        | Numeric scalar. The font line height for item identifiers.   |
| lty               | Character or integer scalar. The line type for all track items. This is also used to connect grouped items. See grouping for details.  |
| lwd               | Integer scalar. The line width for all track items. This is also used to connect grouped items. See grouping for details.  |
| mergeGroups       | Logical scalar. Merge fully overlapping groups if collapse==TRUE.  |
| min.height        | Numeric scalar. The minimum range height in pixels to display. All ranges are expanded to this size in order to avoid rendering issues. See collapsing for details. For feathered bars indicating the strandedness of grouped items this also controls the height of the arrow feathers. |
| min.width         | Numeric scalar. The minimum range width in pixels to display. All ranges are expanded to this size in order to avoid rendering issues. See collapsing for details.   |
| rotation          | Numeric scalar. The degree of text rotation for item identifiers.  |
| rotation.group    | Numeric scalar. The degree of text rotation for group labels.  |
| rotation.item     | Numeric scalar. The degree of text rotation for item identifiers.  |



*(continued)*

| Parameter        | Description   |
|------------------|---|
| shape            | Character scalar. The shape in which to display the track items. Currently only <code>box</code> , <code>arrow</code> , <code>fixedArrow</code> , <code>ellipse</code> , and <code>smallArrow</code> are implemented. |
| showFeatureId    | Logical scalar. Control whether to plot the individual track item identifiers.  |
| showId           | Logical scalar. Control whether to annotate individual groups.  |
| showOverplotting | Logical scalar. Use a color gradient to show the amount of overplotting for collapsed items. This implies that <code>collapse==TRUE</code>  |