pyGenomeTracks

Release 3.5.1

Lucille Lopez-Delisle, Leily Rabbani, Joachim Wolff, Thomas Manl

Oct 12, 2020

CONTENTS:

CHAPTER

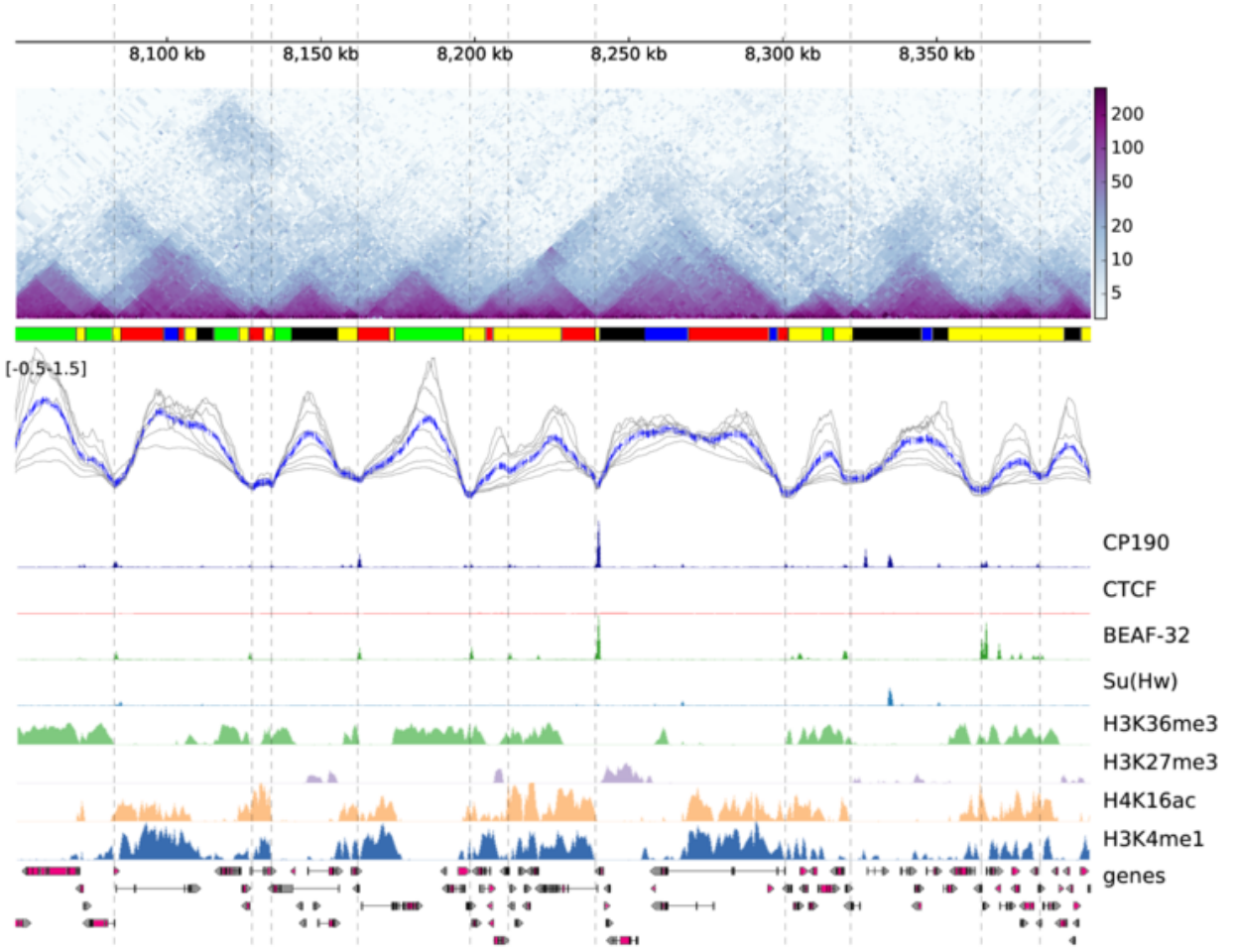
ONE

STANDALONE PROGRAM AND LIBRARY TO PLOT BEAUTIFUL GENOME BROWSER TRACKS

pyGenomeTracks aims to produce high-quality genome browser tracks that are highly customizable. Currently, it is possible to plot:

- bigwig
- bed/gtf (many options)
- bedgraph
- epilogos
- narrow peaks
- links
- Hi-C matrices

pyGenomeTracks can make plots with or without Hi-C data. The following is an example output of pyGenomeTracks from Ramírez et al. 2017.



There are 3 ways for using pyGenomeTracks:

- **Galaxy usage** – the public European Galaxy server let's you use pyGenomeTracks within the familiar Galaxy framework without the need to master the command line
- **command line usage** – simply download and install the tool (see *Installation* and *Usage*)

CHAPTER

TWO

TABLE OF CONTENT

2.1 Installation

Remember – pyGenomeTracks is available for **command line usage** as well as for **integration into Galaxy servers**!

2.1.1 Requirements

Python dependencies:

- Python >= 3.6
- numpy >= 1.16
- intervaltree >= 2.1.0
- pyBigWig >= 0.3.16
- hicmatrix >= 15
- pysam >= 0.14
- matplotlib == 3.1.1
- gffutils >= 0.9
- pybedtools >= 0.8.1
- tqdm >= 4.20

External dependencies:

- BEDTools

2.1.2 Command line installation using `conda`

We encourage users to use `conda` installation to install pygenometracks. All of the requirements for pyGenomeTracks can be installed via Anaconda with:

```
$ conda create -n pygenometracks -c bioconda -c conda-forge pygenometracks python=3.7
```

To get a specific version, one can specify it. For example:

```
$ conda create -n pygenometracks -c bioconda -c conda-forge pygenometracks=3.5␣
↪python=3.7
```

2.1.3 Command line installation using `pip`

Install pyGenomeTracks using the following command:

```
$ pip install pyGenomeTracks
```

All python requirements should be automatically installed.

Since version 3.5, pyGenomeTracks require BEDTools, do not forget to install it or load it into your environment.

If you need to specify a specific path for the installation of the tools, make use of *pip install*'s numerous options:

```
$ pip install --install-option="--prefix=/MyPath/Tools/pyGenomeTracks" git+https://
↪github.com/deeptools/pyGenomeTracks.git
```

2.1.4 Command line installation without `pip`

You are highly recommended to use *conda install* rather than the following complicated steps.

1. Install the requirements listed above in the "requirements" section. This is done automatically by *pip* (except BEDTools).

2. Download source code

```
$ git clone https://github.com/deeptools/pyGenomeTracks.git
```

or if you want a particular release, choose one from https://github.com/deeptools/pygenometracks/releases:

```
$ wget https://github.com/deeptools/pyGenomeTracks/archive/3.1.tar.gz
$ tar -xzvf
```

3. install the source code (if you don't have root permission, you can set a specific folder using the `--prefix` option)

```
$ python setup.py install --prefix /User/Tools/pyGenomeTracks3.1
```

2.1.5 Galaxy installation

pyGenomeTracks can be easily integrated into a local Galaxy. The wrapper and its dependencies are available in the Galaxy Tool Shed.

Installation via Galaxy API (recommended)

First generate an API Key for your admin user and run the the installation script:

```
$ python ./scripts/api/install_tool_shed_repositories.py \
        --api YOUR_API_KEY -l http://localhost/ \
        --url http://toolshed.g2.bx.psu.edu/ \
        -o iuc -r <revision> --name pygenometracks \
        --tool-deps --repository-deps --panel-section-name plots
```

The `-r` argument specifies the version of pygenometracks.

You can watch the installation status under: Top Panel –> Admin –> Manage installed tool shed repositories

Installation via web browser

- go to the admin page
- select *Search and browse tool sheds*
- Galaxy tool shed –> Visualization –> pygenometracks
- install pygenometracks

2.2 Usage

2.2.1 Starting usage

To run pyGenomeTracks a configuration file describing the tracks is required.

In a configuration file, each track is defined as a block of parameters starting with its name `[track name]` and continues with the parameters for that track such as the file location, its title, height, color etc.

The tracks are then plotted in the order of the configuration file from top to bottom.

The easiest way to create this file is using the program `make_tracks_file` which creates a configuration file with defaults that can be easily changed. `make_tracks_file` uses the file ending to guess the file type. Then, a region can be plotted using `pyGenomeTracks`. Both programs are described below:

2.2.2 make_tracks_file

Facilitates the creation of a configuration file for pyGenomeTracks. The program takes a list of files and does the boilerplate for the configuration file.

```
usage: make_tracks_file --trackFiles <bigwig file> <bed file> etc. -o tracks.ini
```

Named Arguments

--trackFiles, -f	Files to use in for the tracks. The ending of the file is used to define the type of track. E.g. *.bw* for bigwig, *.bed* for bed etc. For a arcs or links file, the file ending recognized is *.arcs* or *.links*
--out, -o	File to save the tracks
--version	show program's version number and exit

2.2.3 pyGenomeTracks

2.3 Citation

If you use pyGenomeTracks in your analysis, you can cite the following paper :

Fidel Ramírez, Vivek Bhardwaj, Laura Arrigoni, Kin Chung Lam, Björn A. Grüning, José Villaveces, Bianca Habermann, Asifa Akhtar & Thomas Manke. High-resolution TADs reveal DNA sequences underlying genome organization in flies. Nature Communications (2018) doi:10.1038/s41467-017-02525-w.

2.4 All available tracks

2.4.1 bed

Description

A track for all bed-like files (with first column chromosome, second start and third end). If the other columns fit the requirement as defined in UCSC additional fields can be used. For example, the 5th and 9th column can be used to change the color of intervals, the 6th column indicate the strand... By default, intervals without strand are displayed as rectangle and for intervals with strand an arrow is added at the extremity (not included in the interval). In case of BED12 format, the introns are displayed into another color. But other styles are available.

Parameters

Necessary:

- **file**

Optional:

- **title**: Put here a title which will apprear on the right.
- **height**: *0.5* (default) or float above 0.
- **overlay_previous**: *no* (default) or yes or share-y.
- **fontsize**: *12* (default) or any float above 0
- **orientation**: by default this option is not set but you can also put: inverted.
- **line_width**: *0.5* (default) or any float above 0
- **color**: *#1f78b4* (default)
- **max_value**: by default this option is not set but you can also put: any float
- **min_value**: by default this option is not set but you can also put: any float
- **border_color**: *black* (default)
- **prefered_name**: *transcript_name* (default)
- **merge_transcripts**: *false* (default) or true.
- **labels**: *true* (default) or false.
- **style**: *flybase* (default) or UCSC or tssarrow.
- **display**: *stacked* (default) or collapsed, triangles or interleaved.
- **max_labels**: *60* (default) or any integer above 0
- **global_max_row**: *false* (default) or true.
- **gene_rows**: by default this option is not set but you can also put: any integer above 0
- **arrow_interval**: *2* (default) or any integer above 1
- **arrowhead_included**: *false* (default) or true.
- **color_utr**: *grey* (default)
- **height_utr**: *1* (default) or any float above 0 below 1
- **arrow_length**: by default this option is not set but you can also put: any integer above 0
- **all_labels_inside**: *false* (default) or true.
- **labels_in_margin**: *false* (default) or true.

Output of `make_tracks_file`:

```
# title of track (plotted on the right side)
title =
# height of track in cm (ignored if the track is overlay on top the previous track)
height = 2
# if you want to plot the track upside-down:
# orientation = inverted
# if you want to plot the track on top of the previous track. Options are 'yes' or
↪'share-y'.
# For the 'share-y' option the y axis values is shared between this plot and the␣
↪overlay plot.
# Otherwise, each plot use its own scale
```

(continues on next page)

(continued from previous page)

```
#overlay_previous = yes

# If the bed file contains the exon
# structure (bed 12) then this is plotted. Otherwise
# a region **with direction** is plotted.
# If the bed file contains a column for color (column 9), then this color can be used␣
↪by
# setting:
#color = bed_rgb
# if color is a valid colormap name (like RbBlGn), then the score (column 5) is mapped
# to the colormap.
# In this case, the the min_value and max_value for the score can be provided,␣
↪otherwise
# the maximum score and minimum score found are used.
#color = RdYlBu
#min_value=0
#max_value=100
# If the color is simply a color name, then this color is used and the score is not␣
↪considered.
color = darkblue
# whether printing the labels
labels = false
# optional:
# by default the labels are not printed if you have more than 60 features.
# to change it, just increase the value:
#max_labels = 60
# optional: font size can be given to override the default size
fontsize = 10
# optional: line_width
#line_width = 0.5
# the display parameter defines how the bed file is plotted.
# Default is 'stacked' where regions are plotted on different lines so
# we can see all regions and all labels.
# The other options are ['collapsed', 'interleaved', 'triangles']
# These options assume that the regions do not overlap.
# `collapsed`: The bed regions are plotted one after the other in one line.
# `interleaved`: The bed regions are plotted in two lines, first up, then down, then␣
↪up etc.
# optional, default is black. To remove the border, simply set 'border_color' to none
# Not used in tssarrow style
#border_color = black
# style to plot the genes when the display is not triangles
#style = UCSC
#style = flybase
#style = tssarrow
# maximum number of gene rows to be plotted. This
# field is useful to limit large number of close genes
# to be printed over many rows. When several images want
# to be combined this must be set to get equal size
# otherwise, on each image the height of each gene changes
#gene_rows = 10
# by default the ymax is the number of
# rows occupied by the genes in the region plotted. However,
# by setting this option, the global maximum is used instead.
# This is useful to combine images that are all consistent and
# have the same number of rows.
#global_max_row = true
```

(continues on next page)

(continued from previous page)

```
# If you want to plot all labels inside the plotting region:
#all_labels_inside = true
# If you want to display the name of the gene which goes over the plotted
# region in the right margin put:
#labels_in_margin = true
# if you use UCSC style, you can set the relative distance between 2 arrows on introns
# default is 2
#arrow_interval = 2
# if you use tssarrow style, you can choose the length of the arrow in bp
# (default is 4% of the plotted region)
#arrow_length = 5000
# if you use flybase or tssarrow style, you can choose the color of non-coding␣
↪intervals:
#color_utr = grey
# as well as the proportion between their height and the one of coding
# (by default they are the same height):
#height_utr = 1
# By default, for oriented intervals in flybase style,
# or bed files with less than 12 columns, the arrowhead is added
# outside of the interval.
# If you want that the tip of the arrow correspond to
# the extremity of the interval use:
# arrowhead_included = true
# optional. If not given is guessed from the file ending.
file_type = bed
```

2.4.2 bedgraph

Description

A track for bedgraph files.

Parameters

Necessary:

- **file**

Optional:

- **title**: Put here a title which will apprear on the right.
- **height**: *0.5* (default) or float above 0.
- **overlay_previous**: *no* (default) or yes or share-y.
- **orientation**: by default this option is not set but you can also put: inverted.
- **color**: *#a6cee3* (default)
- **alpha**: *1* (default) or any float above 0 below 1
- **max_value**: by default this option is not set but you can also put: any float

- **min_value**: by default this option is not set but you can also put: any float
- **use_middle**: *false* (default) or true.
- **show_data_range**: *true* (default) or false.
- **type**: *fill* (default)
- **negative_color**: by default this option is not set
- **nans_to_zeros**: *false* (default) or true.
- **summary_method**: by default this option is not set but you can also put: mean, average, max, min, stdev, dev, coverage, cov or sum.
- **number_of_bins**: *700* (default) or any integer above 1
- **transform**: *no* (default) or log, log1p, -log, log2 or log10.
- **log_pseudocount**: *0* (default) or any float
- **y_axis_values**: *transformed* (default) or original.
- **second_file***: by default this option is not set
- **operation***: *file* (default)
- **grid**: *false* (default) or true.
- **rasterize**: *false* (default) or true.

* While pyGenomeTracks can convert coverage tracks on the fly, this might be a time-consuming step, especially on large files and if you want to replot many times. In this situation, we recommend using the deepTools suite to convert your files in advance. For example bamCoverage or bamCompare

Output of `make_tracks_file`:

```
# title of track (plotted on the right side)
title =
# height of track in cm (ignored if the track is overlay on top the previous track)
height = 2
# if you want to plot the track upside-down:
# orientation = inverted
# if you want to plot the track on top of the previous track. Options are 'yes' or
↪'share-y'.
# For the 'share-y' option the y axis values is shared between this plot and the␣
↪overlay plot.
# Otherwise, each plot use its own scale
#overlay_previous = yes

color = green
# To use a different color for negative values
#negative_color = red
# To use transparency, you can use alpha
# default is 1
# alpha = 0.5
# the default for min_value and max_value is 'auto' which means that the scale will go
# roughly from the minimum value found in the region plotted to the maximum value␣
↪found.
min_value = 0
#max_value = auto
# to convert missing data (NaNs) into zeros. Otherwise, missing data is not plotted.
```

(continues on next page)

(continued from previous page)

```
nans_to_zeros = true
# for type, the options are: line, points, fill. Default is fill
# to add the preferred line width or point size use:
# type = line:lw where lw (linewidth) is float
# similarly points:ms sets the point size (markersize (ms) to the given float
# type = line:0.5
# type = points:0.5
# If you want to plot a 4C track where you want to link
# the non-missing data (NaNs) together and only use the
# middle of the region instead of the region itself:
# Default is false.
# use_middle = true
# By default the bedgraph is plotted at the base pair
# resolution. This can lead to very large pdf/svg files
# If plotting large regions.
# If you want to decrase the size of your file.
# You can either rasterize the bedgraph profile by using:
# rasterize = true
# Or use a summary method on a given number of bin:
# The possible summary methods are given by pyBigWig:
# mean/average/stdev/dev/max/min/cov/coverage/sum
# summary_method = mean
# number_of_bins = 700
# set show_data_range to false to hide the text on the left showing the data range
show_data_range = true
# to compute operations on the fly on the file
# or between 2 bedgraph files
# operation will be evaluated, it should contains file or
# file and second_file,
# we advice to use nans_to_zeros = true to avoid unexpected nan values
#operation = 0.89 * file
#operation = - file
#operation = file - second_file
#operation = log2((1 + file) / (1 + second_file))
#operation = max(file, second_file)
#second_file = path for the second file
# To log transform your data you can also use transform and log_pseudocount:
# For the transform values:
# 'log1p': transformed_values = log(1 + initial_values)
# 'log': transformed_values = log(log_pseudocount + initial_values)
# 'log2': transformed_values = log2(log_pseudocount + initial_values)
# 'log10': transformed_values = log10(log_pseudocount + initial_values)
# '-log': transformed_values = log(log_pseudocount + initial_values)
# For example:
#tranform = log
#log_pseudocount = 2
# When a transformation is applied, by default the y axis
# gives the transformed values, if you prefer to see
# the original values:
#y_axis_values = original
# If you want to have a grid on the y-axis
#grid = true
file_type = bedgraph
```

2.4.3 bedgraph_matrix

Description

A track for file like bedgraph but with more than 4 columns, like the insulation score from hicPlotTADs

Parameters

Necessary:

- **file**

Optional:

- **title**: Put here a title which will apprear on the right.
- **height**: *0.5* (default) or float above 0.
- **overlay_previous**: *no* (default) or yes or share-y.
- **orientation**: by default this option is not set but you can also put: inverted.
- **max_value**: by default this option is not set but you can also put: any float
- **min_value**: by default this option is not set but you can also put: any float
- **show_data_range**: *true* (default) or false.
- **type**: *matrix* (default) or lines.
- **rasterize**: *true* (default) or false.
- **pos_score_in_bin**: *center* (default) or block.
- **plot_horizontal_lines**: *false* (default) or true.
- **colormap**: *viridis* (default)

Output of `make_tracks_file`:

```
# title of track (plotted on the right side)
title =
# height of track in cm (ignored if the track is overlay on top the previous track)
height = 2
# if you want to plot the track upside-down:
# orientation = inverted
# if you want to plot the track on top of the previous track. Options are 'yes' or
↪'share-y'.
# For the 'share-y' option the y axis values is shared between this plot and the␣
↪overlay plot.
# Otherwise, each plot use its own scale
#overlay_previous = yes

# a bedgraph matrix file is like a bedgraph, except that per bin there
# are more than one value separated by tab: E.g.
# This file type is produced by HiCExplorer tool hicFindTads and contains
# the TAD-separation score at different window sizes
```

(continues on next page)

(continued from previous page)

```
# chrX        18279        40131        0.399113        0.364118        0.
↪320857        0.274307
# chrX        40132        54262        0.479340        0.425471        0.
↪366541        0.324736
#min_value = 0.10
#max_value = 0.70
# if type is set as lines, then the TAD score lines are drawn instead
# of the matrix otherwise a heatmap is plotted
type = lines
# If the type is not lines, you can choose to keep the matrix as not rasterized
# (only used if you use pdf or svg output format) by using:
# rasterize = false
# The different options for color maps can be found here:
# https://matplotlib.org/users/colormaps.html
# the default color map is viridis
# If you want your own colormap you can put the values of the color you want
# For example, colormap = ['blue', 'yellow', 'red']
# or colormap = ['white', (1, 0.88, 2./3), (1, 0.74, 0.25), (1, 0.5, 0), (1, 0.19, 0),
↪ (0.74, 0, 0), (0.35, 0, 0)]
#colormap = Reds
# pos_score_in_bin means 'position of score with respect to bin start and end'
# if the lines option is used, the y values can be put at the
# center of the bin (default) or they can be plot as 'block',
# which mean to plot the values as a line between the start and end of bin
pos_score_in_bin = center
show_data_range = true

# only when type lines is used. Adds horizontal lines
plot_horizontal_lines = false
file_type = bedgraph_matrix
```

2.4.4 bigwig

Description

A track for bigwig files.

Parameters

Necessary:

- **file**

Optional:

- **title**: Put here a title which will apprear on the right.
- **height**: *0.5* (default) or float above 0.
- **overlay_previous**: *no* (default) or yes or share-y.
- **orientation**: by default this option is not set but you can also put: inverted.
- **color**: *#33a02c* (default)
- **alpha**: *1* (default) or any float above 0 below 1
- **max_value**: by default this option is not set but you can also put: any float
- **min_value**: by default this option is not set but you can also put: any float
- **show_data_range**: *true* (default) or false.
- **type**: *fill* (default)
- **negative_color**: by default this option is not set
- **nans_to_zeros**: *false* (default) or true.
- **summary_method**: *mean* (default) or average, max, min, stdev, dev, coverage, cov or sum.
- **number_of_bins**: *700* (default) or any integer above 1
- **transform**: *no* (default) or log, log1p, -log, log2 or log10.
- **log_pseudocount**: *0* (default) or any float
- **y_axis_values**: *transformed* (default) or original.
- **second_file***: by default this option is not set
- **operation***: *file* (default)
- **grid**: *false* (default) or true.

* While pyGenomeTracks can convert coverage tracks on the fly, this might be a time-consuming step, especially on large files and if you want to replot many times. In this situation, we recommend using the deepTools suite to convert your files in advance. For example bamCoverage or bamCompare

Output of `make_tracks_file`:

```
# title of track (plotted on the right side)
title =
# height of track in cm (ignored if the track is overlay on top the previous track)
height = 2
# if you want to plot the track upside-down:
# orientation = inverted
# if you want to plot the track on top of the previous track. Options are 'yes' or
↪'share-y'.
# For the 'share-y' option the y axis values is shared between this plot and the␣
↪overlay plot.
# Otherwise, each plot use its own scale
#overlay_previous = yes

color = #666666
# To use a different color for negative values
#negative_color = red
# To use transparency, you can use alpha
# default is 1
# alpha = 0.5
# the default for min_value and max_value is 'auto' which means that the scale will go
# roughly from the minimum value found in the region plotted to the maximum value␣
↪found.
min_value = 0
#max_value = auto
# The number of bins takes the region to be plotted and divides it
# into the number of bins specified
# Then, at each bin the bigwig mean value is computed and plotted.
# A lower number of bins produces a coarser tracks
number_of_bins = 700
# to convert missing data (NaNs) into zeros. Otherwise, missing data is not plotted.
nans_to_zeros = true
# The possible summary methods are given by pyBigWig:
# mean/average/stdev/dev/max/min/cov/coverage/sum
# default is mean
summary_method = mean
# for type, the options are: line, points, fill. Default is fill
# to add the preferred line width or point size use:
# type = line:lw where lw (linewidth) is float
# similarly points:ms sets the point size (markersize (ms) to the given float
# type = line:0.5
# type = points:0.5
# set show_data_range to false to hide the text on the left showing the data range
show_data_range = true
# to compute operations on the fly on the file
# or between 2 bigwig files
# operation will be evaluated, it should contains file or
# file and second_file,
# we advice to use nans_to_zeros = true to avoid unexpected nan values
#operation = 0.89 * file
#operation = - file
#operation = file - second_file
#operation = log2((1 + file) / (1 + second_file))
#operation = max(file, second_file)
#second_file = path for the second file
# To log transform your data you can also use transform and log_pseudocount:
```

(continues on next page)

(continued from previous page)

```
# For the transform values:
# 'log1p': transformed_values = log(1 + initial_values)
# 'log': transformed_values = log(log_pseudocount + initial_values)
# 'log2': transformed_values = log2(log_pseudocount + initial_values)
# 'log10': transformed_values = log10(log_pseudocount + initial_values)
# '-log': transformed_values = - log(log_pseudocount + initial_values)
# For example:
#tranform = log
#log_pseudocount = 2
# When a transformation is applied, by default the y axis
# gives the transformed values, if you prefer to see
# the original values:
#y_axis_values = original
# If you want to have a grid on the y-axis
#grid = true
file_type = bigwig
```

2.4.5 domains

Description

A track for bed files that you want to see as triangles.

Parameters

Necessary:

- **file**

Optional:

- **title**: Put here a title which will apprear on the right.
- **height**: *0.5* (default) or float above 0.
- **overlay_previous**: *no* (default) or yes or share-y.
- **orientation**: by default this option is not set but you can also put: inverted.
- **line_width**: *0.5* (default) or any float above 0
- **color**: *#1f78b4* (default)
- **max_value**: by default this option is not set but you can also put: any float
- **min_value**: by default this option is not set but you can also put: any float
- **border_color**: *black* (default)
- **prefered_name**: *transcript_name* (default)
- **merge_transcripts**: *false* (default) or true.

Output of `make_tracks_file`:

```
# title of track (plotted on the right side)
title =
# height of track in cm (ignored if the track is overlay on top the previous track)
height = 2
# if you want to plot the track upside-down:
# orientation = inverted
# if you want to plot the track on top of the previous track. Options are 'yes' or
↪'share-y'.
# For the 'share-y' option the y axis values is shared between this plot and the␣
↪overlay plot.
# Otherwise, each plot use its own scale
#overlay_previous = yes

# If the bed file contains a column for color (column 9), then this color can be used␣
↪by
# setting:
#color = bed_rgb
# if color is a valid colormap name (like RbBlGn), then the score (column 5) is mapped
# to the colormap.
# In this case, the the min_value and max_value for the score can be provided,␣
↪otherwise
# the maximum score and minimum score found are used.
#color = RdYlBu
#min_value=0
#max_value=100
# If the color is simply a color name, then this color is used and the score is not␣
↪considered.
color = darkblue
# optional: line_width
#line_width = 0.5
# optional, default is black. To remove the border, simply set 'border_color' to none
#border_color = black
# optional. If not given it is guessed from the file ending.
file_type = domains
```

2.4.6 epilogos

Description

A track for epilogos files.

Parameters

Necessary:

- **file**

Optional:

- **title**: Put here a title which will apprear on the right.
- **height**: *0.5* (default) or float above 0.
- **overlay_previous**: *no* (default) or yes or share-y.
- **categories_file**: by default this option is not set
- **orientation**: by default this option is not set but you can also put: inverted.

Output of `make_tracks_file`:

```
# title of track (plotted on the right side)
title =
# height of track in cm (ignored if the track is overlay on top the previous track)
height = 2
# if you want to plot the track upside-down:
# orientation = inverted
# if you want to plot the track on top of the previous track. Options are 'yes' or
↪'share-y'.
# For the 'share-y' option the y axis values is shared between this plot and the␣
↪overlay plot.
# Otherwise, each plot use its own scale
#overlay_previous = yes

# The categories file should contain the color information for each category id
# A categories file should look like:
# {
# "categories":{
#            "1":["Active TSS","#ff0000"],
#            "2":["Flanking Active TSS","#ff4500"],
#            "3":["Transcr at gene 5" and 3"","#32cd32"],
#            "4":["Strong transcription","#008000"],
#            "5":["Weak transcription","#006400"]
#         }
#}
categories_file = <path to json categories file>
# optional. If not given, it is guessed from the file ending.
file_type = epilogos
```

2.4.7 gtf

Description

A track for gtf files.

Parameters

Necessary:

- **file**

Optional:

- **title**: Put here a title which will apprear on the right.
- **height**: *0.5* (default) or float above 0.
- **overlay_previous**: *no* (default) or yes or share-y.
- **fontsize**: *12* (default) or any float above 0
- **orientation**: by default this option is not set but you can also put: inverted.
- **line_width**: *0.5* (default) or any float above 0
- **color**: *#1f78b4* (default)
- **border_color**: *black* (default)
- **prefered_name**: *transcript_name* (default)
- **merge_transcripts**: *false* (default) or true.
- **labels**: *true* (default) or false.
- **style**: *flybase* (default) or UCSC or tssarrow.
- **display**: *stacked* (default) or collapsed, triangles or interleaved.
- **max_labels**: *60* (default) or any integer above 0
- **global_max_row**: *false* (default) or true.
- **gene_rows**: by default this option is not set but you can also put: any integer above 0
- **arrow_interval**: *2* (default) or any integer above 1
- **arrowhead_included**: *false* (default) or true.
- **color_utr**: *grey* (default)
- **height_utr**: *1* (default) or any float above 0 below 1
- **arrow_length**: by default this option is not set but you can also put: any integer above 0
- **all_labels_inside**: *false* (default) or true.
- **labels_in_margin**: *false* (default) or true.

Output of `make_tracks_file`:

```
# title of track (plotted on the right side)
title =
# height of track in cm (ignored if the track is overlay on top the previous track)
height = 2
# if you want to plot the track upside-down:
# orientation = inverted
# if you want to plot the track on top of the previous track. Options are 'yes' or
↪'share-y'.
# For the 'share-y' option the y axis values is shared between this plot and the␣
↪overlay plot.
# Otherwise, each plot use its own scale
#overlay_previous = yes

# By default the transcript_name is used.
# If you want to use the gene_name:
# prefered_name = gene_name
# By default, the gtf is transformed to transcripts
# If you want to use see only one structure per gene
# merge_transcripts = true
# You can change the color of coding sequences by:
color = darkblue
# height of track in cm
height = 5
# whether printing the labels
labels = false
# optional:
# by default the labels are not printed if you have more than 60 features.
# to change it, just increase the value:
#max_labels = 60
# optional: font size can be given to override the default size
fontsize = 10
# optional: line_width
#line_width = 0.5
# the display parameter defines how the gtf file is plotted.
# Default is 'stacked' where regions are plotted on different lines so
# we can see all regions and all labels.
# The other options are ['collapsed', 'interleaved', 'triangles']
# These options assume that the regions do not overlap.
# `collapsed`: The gtf regions are plotted one after the other in one line.
# `interleaved`: The gtf regions are plotted in two lines, first up, then down, then␣
↪up etc.
# optional, default is black. To remove the border, simply set 'border_color' to none
# Not used in tssarrow style
#border_color = black
# style to plot the genes when the display is not triangles
#style = UCSC
#style = flybase
#style = tssarrow
# maximum number of gene rows to be plotted. This
# field is useful to limit large number of close genes
# to be printed over many rows. When several images want
# to be combined this must be set to get equal size
# otherwise, on each image the height of each gene changes
#gene_rows = 10
# by default the ymax is the number of
```

(continues on next page)

(continued from previous page)

```
# rows occupied by the genes in the region plotted. However,
# by setting this option, the global maximum is used instead.
# This is useful to combine images that are all consistent and
# have the same number of rows.
#global_max_row = true
# If you want to plot all labels inside the plotting region:
#all_labels_inside = true
# If you want to display the name of the gene which goes over the plotted
# region in the right margin put:
#labels_in_margin = true
# if you use UCSC style, you can set the relative distance between 2 arrows on introns
# default is 2
#arrow_interval = 2
# if you use tssarrow style, you can choose the length of the arrow in bp
# (default is 4% of the plotted region)
#arrow_length = 5000
# if you use flybase or tssarrow style, you can choose the color of non-coding␣
↪intervals:
#color_utr = grey
# as well as the proportion between their height and the one of coding
# (by default they are the same height):
#height_utr = 1
# By default, for oriented intervals in flybase style,
# or bed files with less than 12 columns, the arrowhead is added
# outside of the interval.
# If you want that the tip of the arrow correspond to
# the extremity of the interval use:
# arrowhead_included = true
# optional. If not given is guessed from the file ending.
file_type = gtf
```

2.4.8 hic_matrix

Description

A track for matrix files. Only cool format and h5 format from HiCExplorer are supported. For other format, please first convert to a supported format, for example with *hicConvertFormat <https://hicexplorer.readthedocs.io/en/latest/content/tools/hicConvertFormat.html#hicconvertformat>*.

Parameters

Necessary:

- **file**

Optional:

- **title**: Put here a title which will apprear on the right.
- **height**: If you do not set it, the height will be adjusted so that each bin is a square, else you can choose any float above 0.
- **overlay_previous**: *no* (default) or yes or share-y.
- **orientation**: by default this option is not set but you can also put: inverted.
- **max_value**: by default this option is not set but you can also put: any float
- **min_value**: by default this option is not set but you can also put: any float
- **transform**: *no* (default) or log, log1p or -log.
- **rasterize**: *true* (default) or false.
- **colormap**: *RdYlBu_r* (default)
- **depth**: *100000* (default) or any integer above 1
- **show_masked_bins**: *false* (default) or true.
- **scale_factor**: *1* (default) or any float

Output of `make_tracks_file`:

```
# The different options for color maps can be found here:
# https://matplotlib.org/users/colormaps.html
# the default color map is RdYlBu_r (_r) stands for reverse
# If you want your own colormap you can put the values of the color you want
# For example, colormap = ['blue', 'yellow', 'red']
# or colormap = ['white', (1, 0.88, 2./3), (1, 0.74, 0.25), (1, 0.5, 0), (1, 0.19, 0),
↪ (0.74, 0, 0), (0.35, 0, 0)]
#colormap = RdYlBu_r
# depth is the maximum distance that should be plotted.
# If it is more than 125% of the plotted region, it will
# be adjsted to this maximum value.
depth = 100000
# height of track (in cm) can be given.
# Otherwise, the height is computed such that the proportions of the
# hic matrix are kept (e.g. the image does not appear shrink or extended)
# height = 10
# min_value and max_value refer to the contacts in the matrix.
#min_value =2.8
#max_value = 3.0
# the matrix can be transformed using the log1p (or log or -log, but zeros could be␣
↪problematic)
transform = log1p
# show masked bins plots as white lines
# those bins that were not used during the correction
# the default is to extend neighboring bins to
# obtain an aesthetically pleasant output
show_masked_bins = false
# optional if the values in the matrix need to be scaled the
# following parameter can be used. This is useful to plot multiple hic-matrices on␣
↪the same scale
# scale_factor = 1
```

(continues on next page)

(continued from previous page)

```
# You can choose to keep the matrix as not rasterized
# (only used if you use pdf or svg output format) by using:
# rasterize = false
file_type = hic_matrix
```

2.4.9 hlines

Description

A track to add horizontal lines to your tracks with `overlay_previous = share-y` or just to separate group of tracks.

Parameters

Necessary:

- **y_values**

Optional:

- **title**: Put here a title which will apprear on the right.
- **height**: *0.5* (default) or float above 0.
- **overlay_previous**: *no* (default) or yes or share-y.
- **orientation**: by default this option is not set but you can also put: inverted.
- **line_width**: *0.5* (default) or any float above 0
- **line_style**: *solid* (default) or dashed, dotted or dashdot.
- **color**: *black* (default)
- **alpha**: *1* (default) or any float above 0 below 1
- **max_value**: by default this option is not set but you can also put: any float
- **min_value**: by default this option is not set but you can also put: any float
- **show_data_range**: *true* (default) or false.

Output of `make_tracks_file`:

```
# title of track (plotted on the right side)
title =
# height of track in cm (ignored if the track is overlay on top the previous track)
height = 2
# if you want to plot the track upside-down:
# orientation = inverted
# if you want to plot the track on top of the previous track. Options are 'yes' or
↪'share-y'.
```

(continues on next page)

(continued from previous page)

```
# For the 'share-y' option the y axis values is shared between this plot and the␣
↪overlay plot.
# Otherwise, each plot use its own scale
#overlay_previous = yes

# color of the lines
color = black
# To use transparency, you can use alpha
# default is 1
# alpha = 0.5
# the default for min_value and max_value is 'auto' which means that the scale will go
# roughly from the minimum value found in the region plotted to the maximum value␣
↪found.
min_value = 0
#max_value = auto
# line width:
# line_width = 0.5
# options for line_style are 'solid', 'dashed', 'dotted', and 'dashdot'
#line_style = solid
# y values where horizontal lines should be plotted separated by comma:
y_values = 10, 200
# set show_data_range to false to hide the text on the upper-left showing the data␣
↪range
show_data_range = true
file_type = hlines
```

2.4.10 links

Description

A track for pairs of intervalls, the supported format is (tab separated): chr1 start1 end1 chr2 start2 end2 (score ...) The score field is optional and fields after the score are ignored.

Parameters

Necessary:

- **file**

Optional:

- **title**: Put here a title which will apprear on the right.
- **height**: *0.5* (default) or float above 0.
- **overlay_previous**: *no* (default) or yes or share-y.
- **orientation**: by default this option is not set but you can also put: inverted.
- **links_type**: *arcs* (default) or triangles or loops.
- **line_width**: by default this option is not set but you can also put: any float above 0

- **line_style**: *solid* (default) or dashed, dotted or dashdot.
- **color**: *blue* (default)
- **alpha**: *0.8* (default) or any float above 0 below 1
- **max_value**: by default this option is not set but you can also put: any float
- **min_value**: by default this option is not set but you can also put: any float
- **ylim**: by default this option is not set but you can also put: any float above 0
- **compact_arcs_level**: *0* (default) or 1 or 2.
- **use_middle**: *false* (default) or true.

Output of `make_tracks_file`:

```
# title of track (plotted on the right side)
title =
# height of track in cm (ignored if the track is overlay on top the previous track)
height = 2
# if you want to plot the track upside-down:
# orientation = inverted
# if you want to plot the track on top of the previous track. Options are 'yes' or
↪'share-y'.
# For the 'share-y' option the y axis values is shared between this plot and the␣
↪overlay plot.
# Otherwise, each plot use its own scale
#overlay_previous = yes

# the file format for links is (tab separated)
#   chr1 start1 end1 chr2 start2 end2 (score ...)
# The score field is optional
# The fields after the score field will be ignored
# for example:
#   chr1 100 200 chr1 250 300 0.5
# depending on the value of links_type either 'arcs' or 'triangles' or 'loops' can be␣
↪plotted.
# If arcs, a line will be drawn from the center of the first region (chr1: 150),
# to the center of the other region (chr1: 275).
# if triangles, the vertix of the triangle will be drawn at the center between the␣
↪two points
# (also the extremity of each position is used)
# if loops, a rectangle highlighting the intersection between the 2 regions will be␣
↪shown
# the triangles, and loops options are convenient to overlay over a
# Hi-C matrix to highlight the matrix pixel of the highlighted link
# For these tracks do not hesitate to put large line_width like 5 or 10.
links_type = arcs
# For triangles and arcs, by default the extremities coordinates are used
# To use the middle of start1 and end1 and the middle of start2 and end2
#use_middle = true
# color of the lines
color = red
# if color is a valid colormap name (like RdYlGn),
# then the score is mapped to the colormap.
#color = RdYlGn
# To set the minimum and maximum value of the colormap:
```

(continues on next page)

(continued from previous page)

```
#min_value = 0
#max_value = 1.2
# To use transparency, you can use alpha
# default is 0.8
# alpha = 0.5
# if line_width is not given, the score is used to set the line width
# using the following formula (0.5 * square root(score)
#line_width = 0.5
# options for line_style are 'solid', 'dashed', 'dotted', and 'dashdot'
line_style = solid
# If you want to compact the arcs (when you have both long and short arcs)
# You can choose a compact level of
# 1 (the height is proportional to the square root of the distance)
# 2 (the height is the same for all distances)
# (default is 0 proportional to distance)
#compact_arcs_level = 2
# To be able to see small arcs when big arcs exists, you can set
# the upper y limit.
# The unit is bp. This corresponds to the longest arc you will see.
# This option is incompatible with compact_arcs_level = 2
#ylim = 100000
file_type = links
```

2.4.11 narrow_peak

Description

A track for ENCODE narrowPeak format.

Parameters

Necessary:

- **file**

Optional:

- **title**: Put here a title which will apprear on the right.
- **height**: *0.5* (default) or float above 0.
- **overlay_previous**: *no* (default) or yes or share-y.
- **orientation**: by default this option is not set but you can also put: inverted.
- **line_width**: *1* (default) or any float above 0
- **color**: *#FF000080* (default)
- **max_value**: by default this option is not set but you can also put: any float
- **show_data_range**: *true* (default) or false.
- **show_labels**: *true* (default) or false.

- **use_summit**: *true* (default) or false.
- **width_adjust**: *1.5* (default) or any float above 0
- **type**: *peak* (default) or box.

Output of `make_tracks_file`:

```
# title of track (plotted on the right side)
title =
# height of track in cm (ignored if the track is overlay on top the previous track)
height = 2
# if you want to plot the track upside-down:
# orientation = inverted
# if you want to plot the track on top of the previous track. Options are 'yes' or
↪'share-y'.
# For the 'share-y' option the y axis values is shared between this plot and the␣
↪overlay plot.
# Otherwise, each plot use its own scale
#overlay_previous = yes

color = #FF000080
#max_value = 0.70
show_data_range = true
show_labels = true
# the narrowPeak format provides the information of the
# peak summit. By default this information is used
# although some peaks may look crooked.
use_summit = true
# type of plot: either box or peak
# box will plot a rectangle of the peak width
# peak will plot the shape of the peak, whose height is the
# narrowPeak file signal value (usually peak coverage)
type = peak
# if the peaks look too thin, the can be adjusted
width_adjust = 1.5
# optional: line_width
#line_width = 0.5
file_type = narrow_peak
```

2.4.12 scalebar

Description

A track to put scalebar like in the UCSC browser or put highlight a distance between two coordinates.

Parameters

Necessary:

Optional:

- **title**: Put here a title which will apprear on the right.
- **height**: *0.5* (default) or float above 0.
- **overlay_previous**: *no* (default) or yes or share-y.
- **where**: *left* (default) or right, top or bottom.
- **fontsize**: *12* (default) or any float above 0
- **line_width**: *0.5* (default) or any float above 0
- **color**: *black* (default)
- **alpha**: *1* (default) or any float above 0 below 1
- **x_center**: by default this option is not set but you can also put: any integer above 0
- **size**: by default this option is not set but you can also put: any integer above 0

Output of `make_tracks_file`:

```
# title of track (plotted on the right side)
title =
# height of track in cm (ignored if the track is overlay on top the previous track)
height = 2
# if you want to plot the track upside-down:
# orientation = inverted
# if you want to plot the track on top of the previous track. Options are 'yes' or
↪'share-y'.
# For the 'share-y' option the y axis values is shared between this plot and the␣
↪overlay plot.
# Otherwise, each plot use its own scale
#overlay_previous = yes

# color of the scalebar
color = black
# To use transparency, you can use alpha
# default is 1
# alpha = 0.5
# line width:
# line_width = 0.5
# x_center: coordinate where the scale bar should be plotted (center)
# if not set it will be in the middle of the plotted area
#x_center = 3100000
```

(continues on next page)

(continued from previous page)

```
# size: in bp the length of the scale bar
# if not set it will be like in UCSC:
# the higher number that begins with 1, 2 or 5 followed by 0s
# that is less than half the plotted area
#size = 100000
# where: where the size of the scale bar should
# appear among left, right, top, bottom
# default is left
#where = right
# fontsize: default is 12
#fontsize = 10
file_type = scalebar
```

2.4.13 spacer

Description

A track to include a space between tracks. To use it you can either put `file_type = spacer` or call your section `[spacer]`

Parameters

Necessary:

Optional:

- **title**: Put here a title which will apprear on the right.
- **height**: *0.5* (default) or float above 0.
- **overlay_previous**: *no* (default) or yes or share-y.

Output of `make_tracks_file`:

```
# title of track (plotted on the right side)
title =
# height of track in cm (ignored if the track is overlay on top the previous track)
height = 2
# if you want to plot the track upside-down:
# orientation = inverted
# if you want to plot the track on top of the previous track. Options are 'yes' or
↪'share-y'.
# For the 'share-y' option the y axis values is shared between this plot and the␣
↪overlay plot.
# Otherwise, each plot use its own scale
#overlay_previous = yes
```

2.4.14 x_axis

Description

A track to include a x axis. To use it you can either put `file_type = x_axis` or call your section `[x-axis]`

Parameters

Necessary:

Optional:

- **title**: Put here a title which will apprear on the right.
- **height**: If you do not set it, the height will be `fontsize / 8`, else you can choose any float above 0.
- **overlay_previous**: *no* (default) or yes or share-y.
- **where**: *bottom* (default) or top.
- **fontsize**: *15* (default) or any float above 0

Output of `make_tracks_file`:

```
# title of track (plotted on the right side)
title =
# height of track in cm (ignored if the track is overlay on top the previous track)
height = 2
# if you want to plot the track upside-down:
# orientation = inverted
# if you want to plot the track on top of the previous track. Options are 'yes' or
↪'share-y'.
# For the 'share-y' option the y axis values is shared between this plot and the␣
↪overlay plot.
# Otherwise, each plot use its own scale
#overlay_previous = yes
```

2.5 Examples

These examples and the input data for these examples can found in the examples/ or test_data/ folders of the github repository.

2.5.1 Basic Examples

A minimal example of a configuration file with a single bigwig track looks like this:

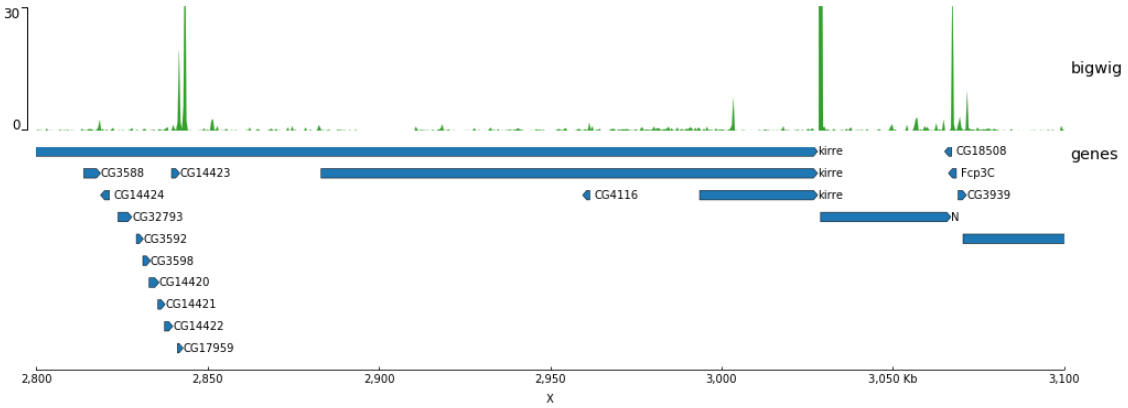
```
[bigwig file test]
file = bigwig.bw
# height of the track in cm (optional value)
height = 4
title = bigwig
min_value = 0
max_value = 30
```

```
$ pyGenomeTracks --tracks bigwig_track.ini --region X:2,500,000-3,000,000 -o bigwig.
↪png
```

Now, let's add the genomic location and some genes:

```
[bigwig file test]
file = bigwig.bw
# height of the track in cm (optional value)
height = 4
title = bigwig
min_value = 0
max_value = 30

[spacer]
# this simply adds an small space between the two tracks.

[genes]
file = genes.bed.gz
height = 7
title = genes
fontsize = 10
file_type = bed
gene_rows = 10

[x-axis]
fontsize=10
```

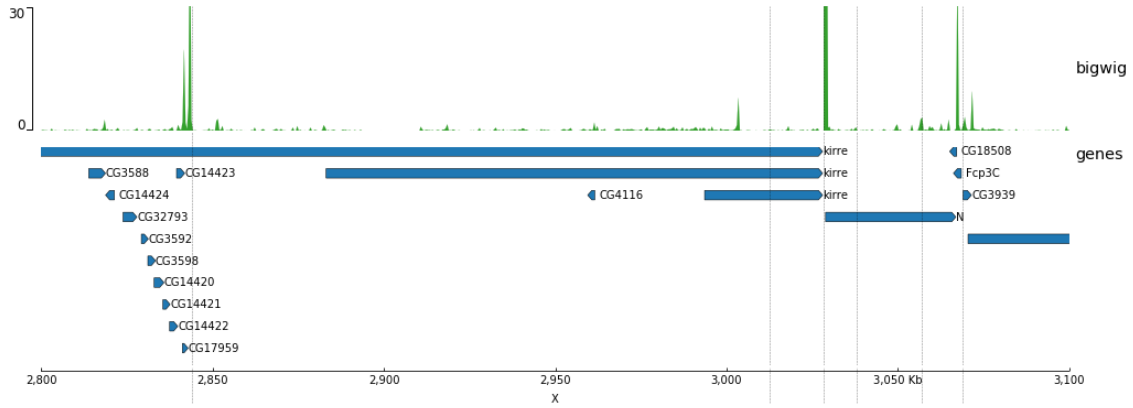
```
$ pyGenomeTracks --tracks bigwig_with_genes.ini --region X:2,800,000-3,100,000 -o␣
↪bigwig_with_genes.png
```



Now, we will add some vertical lines across all tracks. The vertical lines should be in a bed format.

```
[bigwig file test]
file = bigwig.bw
# height of the track in cm (optional value)
height = 4
title = bigwig
min_value = 0
max_value = 30

[spacer]
# this simply adds an small space between the two tracks.

[genes]
file = genes.bed.gz
height = 7
title = genes
fontsize = 10
file_type = bed
gene_rows = 10

[x-axis]
fontsize=10

[vlines]
file = domains.bed
type = vlines
```

```
$ pyGenomeTracks --tracks bigwig_with_genes_and_vlines.ini --region X:2,800,000-3,100,
↪000 -o bigwig_with_genes_and_vlines.png
```



You can also overlay bigwig with or without transparency.

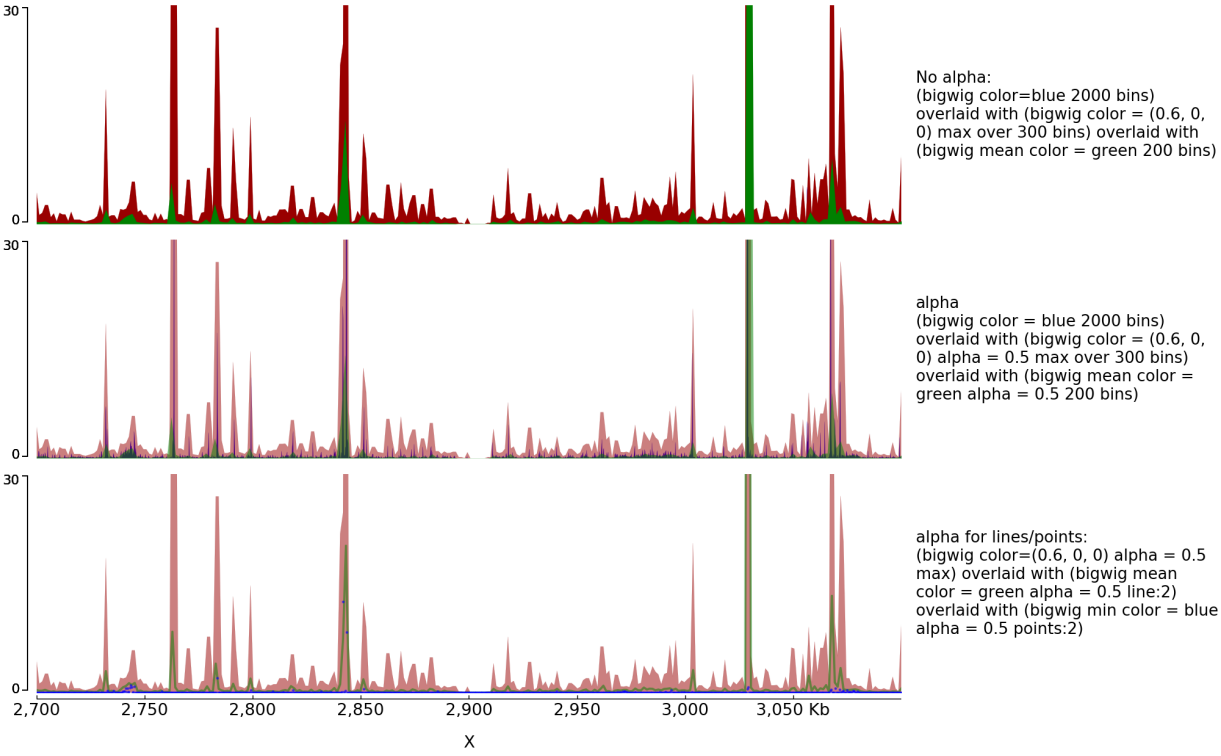
```
[test bigwig]
file = bigwig2_X_2.5e6_3.5e6.bw
color = blue
height = 7
title = No alpha:
        (bigwig color=blue 2000 bins) overlaid with (bigwig color = (0.6, 0, 0) max␣
↪over 300 bins) overlaid with (bigwig mean color = green 200 bins)
number_of_bins = 2000
min_value = 0
max_value = 30

[test bigwig max]
file = bigwig2_X_2.5e6_3.5e6.bw
color = (0.6, 0, 0)
summary_method = max
number_of_bins = 300
overlay_previous = share-y

[test bigwig mean]
file = bigwig2_X_2.5e6_3.5e6.bw
color = green
type = fill
number_of_bins = 200
overlay_previous = share-y

[spacer]

[test bigwig]
file = bigwig2_X_2.5e6_3.5e6.bw
color = blue
height = 7
title = alpha
        (bigwig color = blue 2000 bins) overlaid with (bigwig color = (0.6, 0, 0)␣
↪alpha = 0.5 max over 300 bins) overlaid with (bigwig mean color = green alpha = 0.5␣
↪200 bins)
number_of_bins = 2000
min_value = 0
max_value = 30

[test bigwig max]
file = bigwig2_X_2.5e6_3.5e6.bw
```

(continues on next page)

(continued from previous page)

```
color = (0.6, 0, 0)
alpha = 0.5
summary_method = max
number_of_bins = 300
overlay_previous = share-y

[test bigwig mean]
file = bigwig2_X_2.5e6_3.5e6.bw
color = green
alpha = 0.5
type = fill
number_of_bins = 200
overlay_previous = share-y

[spacer]

[test bigwig]
file = bigwig2_X_2.5e6_3.5e6.bw
height = 7
title = alpha for lines/points:
        (bigwig color=(0.6, 0, 0) alpha = 0.5 max) overlaid with (bigwig mean color =␣
↪green alpha = 0.5 line:2) overlaid with (bigwig min color = blue alpha = 0.5␣
↪points:2)
color = (0.6, 0, 0)
alpha = 0.5
summary_method = max
number_of_bins = 300
min_value = 0
max_value = 30

[test bigwig mean]
file = bigwig2_X_2.5e6_3.5e6.bw
color = green
type = line:2
alpha = 0.5
summary_method = mean
number_of_bins = 300
overlay_previous = share-y

[test bigwig min]
file = bigwig2_X_2.5e6_3.5e6.bw
color = blue
summary_method = min
number_of_bins = 1000
type = points:3
alpha = 0.5
overlay_previous = share-y

[x-axis]
```

```
$ pyGenomeTracks --tracks alpha.ini --region X:2700000-3100000 --trackLabelFraction 0.
↪2 --dpi 130 -o master_alpha.png
```



2.5.2 Examples with bed and gtf

Here is an example to explain the parameters for bed and gtf:

```
[x-axis]
where = top
title = where =top

[spacer]
height = 0.05

[genes 2]
file = dm3_genes.bed.gz
height = 7
title = genes (bed12) style = UCSC; fontsize = 10
style = UCSC
fontsize = 10

[genes 2bis]
file = dm3_genes.bed.gz
height = 7
title = genes (bed12) style = UCSC; arrow_interval=10; fontsize = 10
style = UCSC
arrow_interval = 10
fontsize = 10

[spacer]
height = 1
```

(continues on next page)

(continued from previous page)

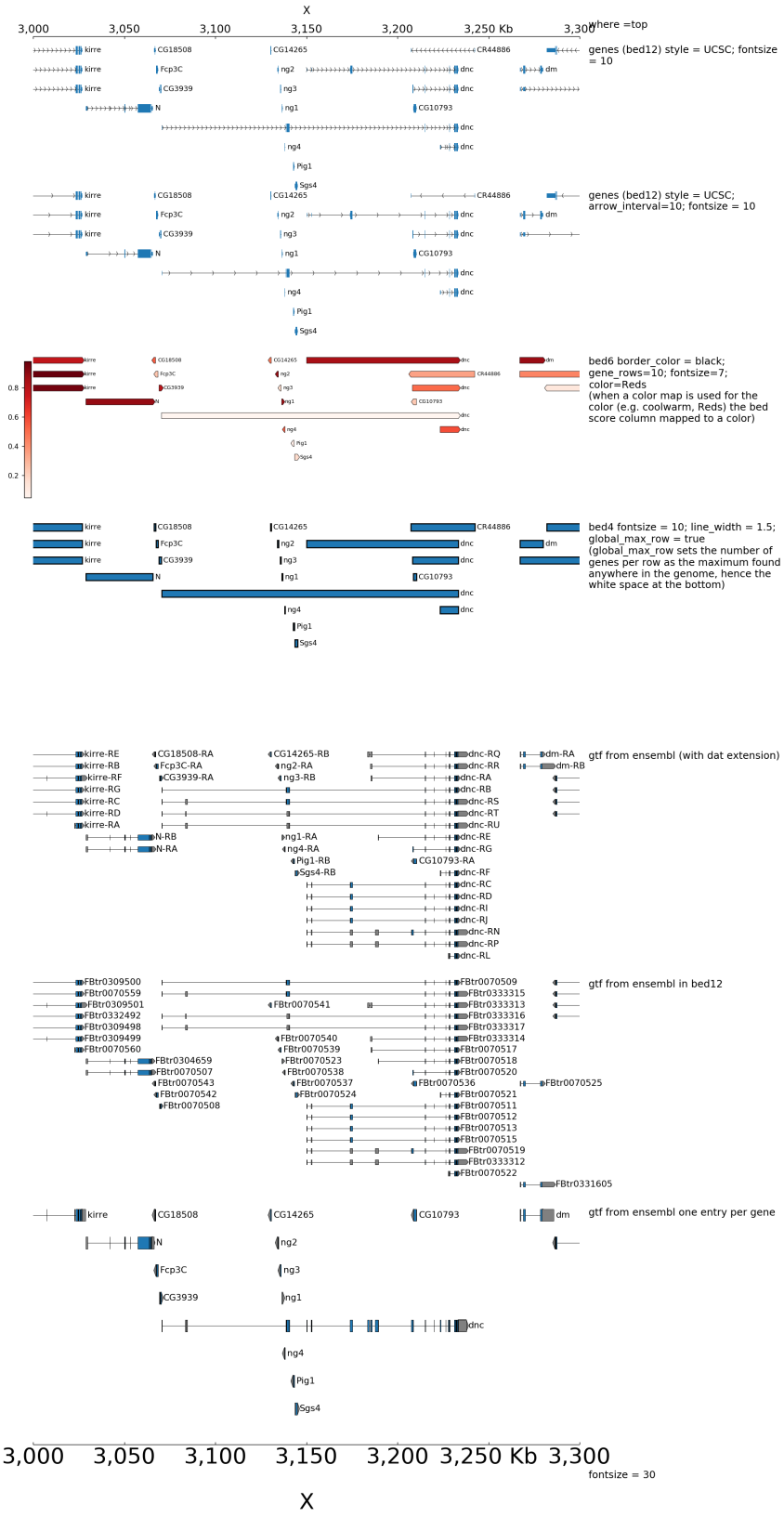
```
[test bed6]
file = dm3_genes.bed6.gz
height = 7
title = bed6 border_color = black; gene_rows=10; fontsize=7; color=Reds
        (when a color map is used for the color (e.g. coolwarm, Reds) the bed
        score column mapped to a color)
fontsize = 7
file_type = bed
color = Reds
border_color = black
gene_rows = 10

[spacer]
height = 1

[test bed4]
file = dm3_genes.bed4.gz
height = 10
title = bed4 fontsize = 10; line_width = 1.5; global_max_row = true
        (global_max_row sets the number of genes per row as the maximum found
        anywhere in the genome, hence the white space at the bottom)
fontsize = 10
file_type = bed
global_max_row = true
line_width = 1.5

[spacer]
height = 1

[test gtf]
file = dm3_subset_BDGP5.78_gtf.dat
height = 10
title = gtf from ensembl (with dat extension)
fontsize = 12
file_type = gtf

[spacer]
height = 1

[test bed]
file = dm3_subset_BDGP5.78_asbed_sorted.bed.gz
height = 10
title = gtf from ensembl in bed12
fontsize = 12
file_type = bed

[spacer]
height = 1

[test gtf collapsed]
file = dm3_subset_BDGP5.78.gtf.gz
height = 10
title = gtf from ensembl one entry per gene
merge_transcripts = true
prefered_name = gene_name
fontsize = 12
file_type = gtf
```

(continues on next page)

(continued from previous page)

```
[spacer]
height = 1

[x-axis]
fontsize = 30
title = fontsize = 30
```

```
$ pyGenomeTracks --tracks bed_and_gtf_tracks.ini --region X:3000000-3300000 --
↪trackLabelFraction 0.2 --width 38 --dpi 130 -o master_bed_and_gtf.png
```


X
3,000
3,050
3,100
3,150
3,200
3,250 Kb
3,300
where =top
genes (bed12) style = UCSC; fontsize = 10
genes (bed12) style = UCSC; arrow_interval=10; fontsize = 10
bed6 border_color = black; gene_rows=10; fontsize=7; color=Reds
(when a color map is used for the color (e.g. coolwarm, Reds) the bed score column mapped to a color)
bed4 fontsize = 10; line_width = 1.5; global_max_row = true
(global_max_row sets the number of genes per row as the maximum found anywhere in the genome, hence the white space at the bottom)
gtf from ensembl (with dat extension)
gtf from ensembl in bed12
gtf from ensembl one entry per gene
fontsize = 30

By default, when bed are displayed and interval are stranded, the arrowhead which indicates the direction is plotted outside of the interval. Here is an example to show how to put it inside:

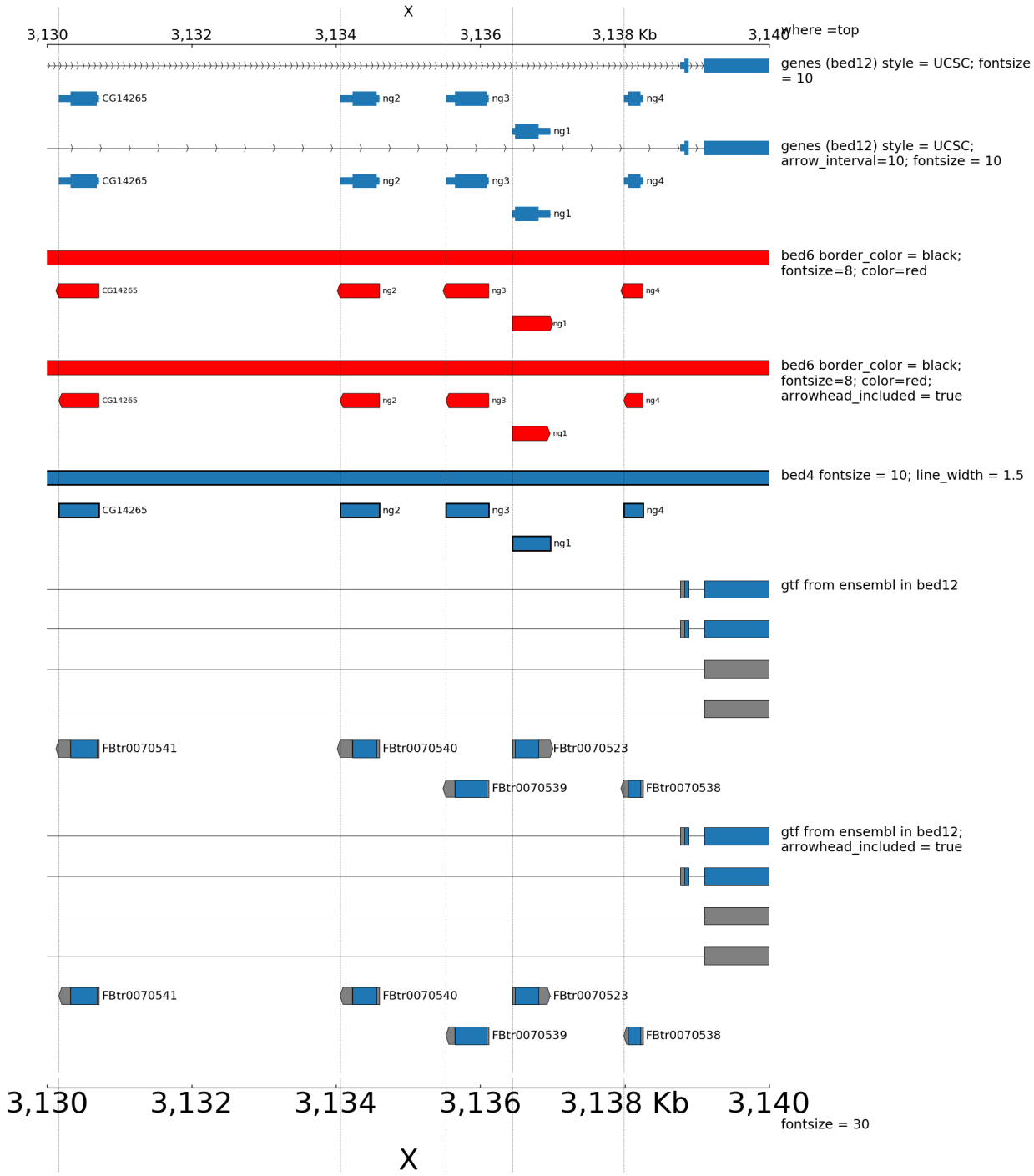
```
[x-axis]
where = top
title = where =top

[spacer]
height = 0.05

[genes 2]
file = dm3_genes.bed.gz
height = 3
title = genes (bed12) style = UCSC; fontsize = 10
style = UCSC
fontsize = 10

[genes 2bis]
file = dm3_genes.bed.gz
height = 3
title = genes (bed12) style = UCSC; arrow_interval=10; fontsize = 10
style = UCSC
arrow_interval = 10
fontsize = 10

[spacer]
height = 1

[test bed6]
file = dm3_genes.bed6.gz
height = 3
title = bed6 border_color = black; fontsize=8; color=red
fontsize = 8
file_type = bed
color = red
border_color = black

[spacer]
height = 1

[test bed6 arrowhead_included]
file = dm3_genes.bed6.gz
height = 3
title = bed6 border_color = black; fontsize=8; color=red; arrowhead_included = true
fontsize = 8
file_type = bed
color = red
border_color = black
arrowhead_included = true

[spacer]
height = 1

[test bed4]
file = dm3_genes.bed4.gz
height = 3
title = bed4 fontsize = 10; line_width = 1.5
fontsize = 10
```

(continues on next page)

(continued from previous page)

```
file_type = bed
line_width = 1.5

[spacer]
height = 1

[test bed]
file = dm3_subset_BDGP5.78_asbed_sorted.bed.gz
height = 8
title = gtf from ensembl in bed12
fontsize = 12
file_type = bed

[spacer]
height = 1

[test bed]
file = dm3_subset_BDGP5.78_asbed_sorted.bed.gz
height = 8
title = gtf from ensembl in bed12; arrowhead_included = true
fontsize = 12
file_type = bed
arrowhead_included = true

[spacer]
height = 1

[x-axis]
fontsize = 30
title = fontsize = 30

[vlines]
type = vlines
file = dm3_genes.bed4.gz
```

```
$ pyGenomeTracks --tracks bed_arrow_tracks.ini --region X:3130000-3140000 --
↪trackLabelFraction 0.2 --width 38 --dpi 130 -o master_bed_arrow_zoom.png
```



When genes are displayed with the default style (flybase), the color and the height of UTR can be set:

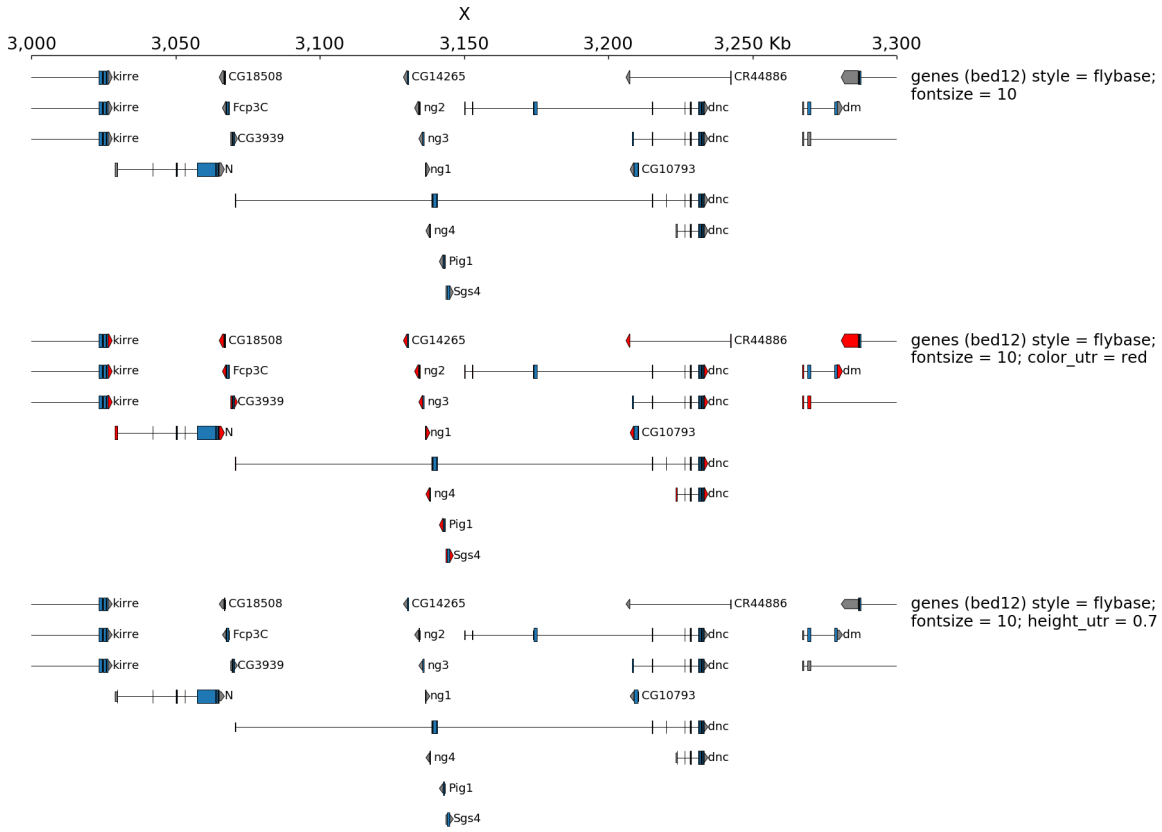
```
[x-axis]
where = top

[spacer]
```

(continues on next page)

(continued from previous page)

```
height = 0.05

[genes 0]
file = dm3_genes.bed.gz
height = 7
title = genes (bed12) style = flybase; fontsize = 10
style = flybase
fontsize = 10

[spacer]
height = 1

[genes 1]
file = dm3_genes.bed.gz
height = 7
title = genes (bed12) style = flybase; fontsize = 10; color_utr = red
style = flybase
fontsize = 10
color_utr = red

[spacer]
height = 1

[genes 2]
file = dm3_genes.bed.gz
height = 7
title = genes (bed12) style = flybase; fontsize = 10; height_utr = 0.7
style = flybase
fontsize = 10
height_utr = 0.7
```

```
$ pyGenomeTracks --tracks bed_flybase_tracks.ini --region X:3000000-3300000 --
↪trackLabelFraction 0.2 --width 38 --dpi 130 -o master_bed_flybase.png
```



2.5.3 Examples with 4C-seq

The output file of some 4C-seq pipeline are bedgraph where the coordinates are the coordinates of the fragment. In these cases, it can be interesting to remove the regions absent from the file and just link the middle of the fragments together instead of plotting a rectangle for each fragment. Here is an example of the option `use_middle`

```
[x-axis]
where = top

[spacer]
height = 0.05

[test bedgraph]
file = GSM3182416_E12DHL_WT_Hoxd11vp.bedgraph.gz
color = blue
height = 5
title = bedgraph rasterize = true
rasterize = true
max_value = 10

[test bedgraph]
file = GSM3182416_E12DHL_WT_Hoxd11vp.bedgraph.gz
color = blue
height = 5
title = bedgraph
max_value = 10
```

(continues on next page)

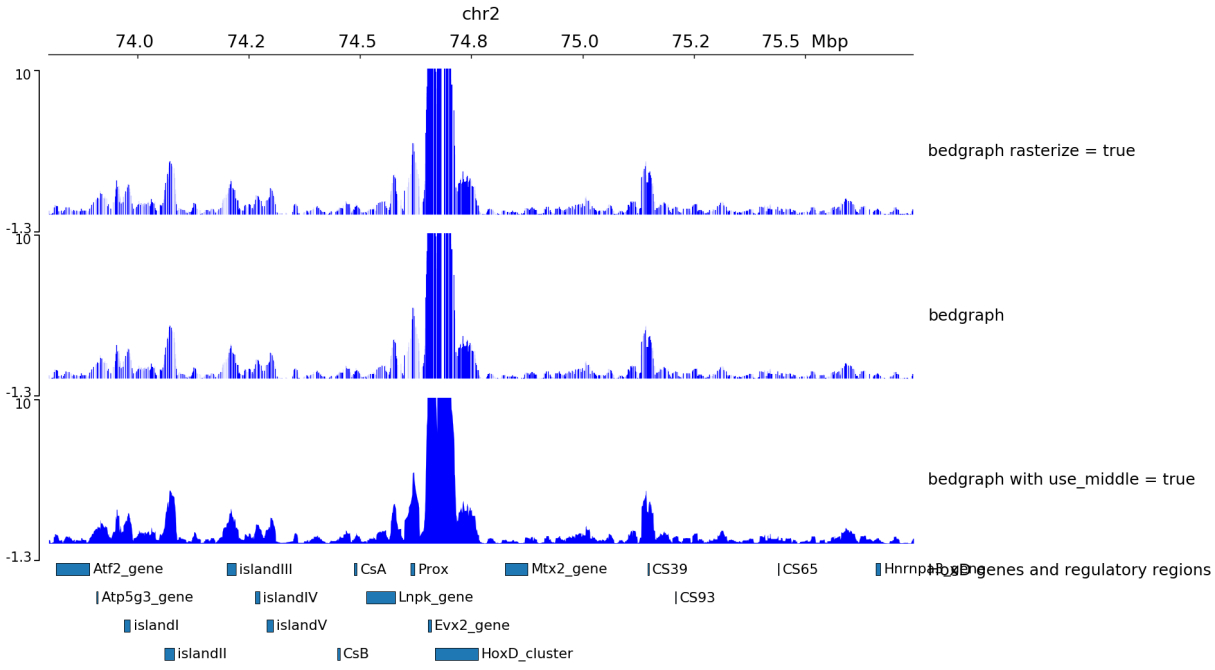
(continued from previous page)

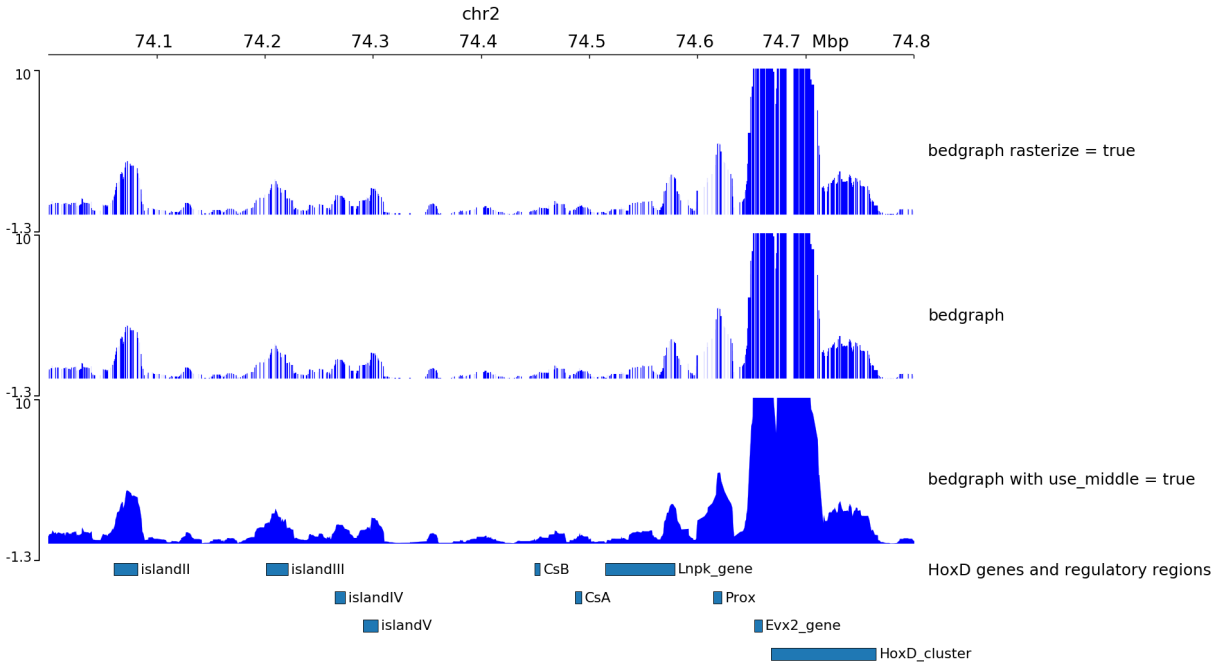
```
[test bedgraph use middle]
file = GSM3182416_E12DHL_WT_Hoxd11vp.bedgraph.gz
color = blue
height = 5
title = bedgraph with use_middle = true
max_value = 10
use_middle = true

[genes]
file = HoxD_cluster_regulatory_regions_mm10.bed
height = 3
title = HoxD genes and regulatory regions
```

We can generate two zooms using a bed instead of regions:

```
track type=bed name=regions_to_plot
chr2        73800000        75744000
chr2        74000000        74800000
```

```
$ pyGenomeTracks --tracks bedgraph_useMid.ini --BED regions_imbricated_chr2.bed --
↪trackLabelFraction 0.2 --width 38 --dpi 130  -o master_bedgraph_useMid.png
```





2.5.4 Examples with peaks

pyGenomeTracks has an option to plot peaks using MACS2 narrowPeak format.

The following is an example of the output in which the peak shape is drawn based on the start, end, summit and height of the peak.

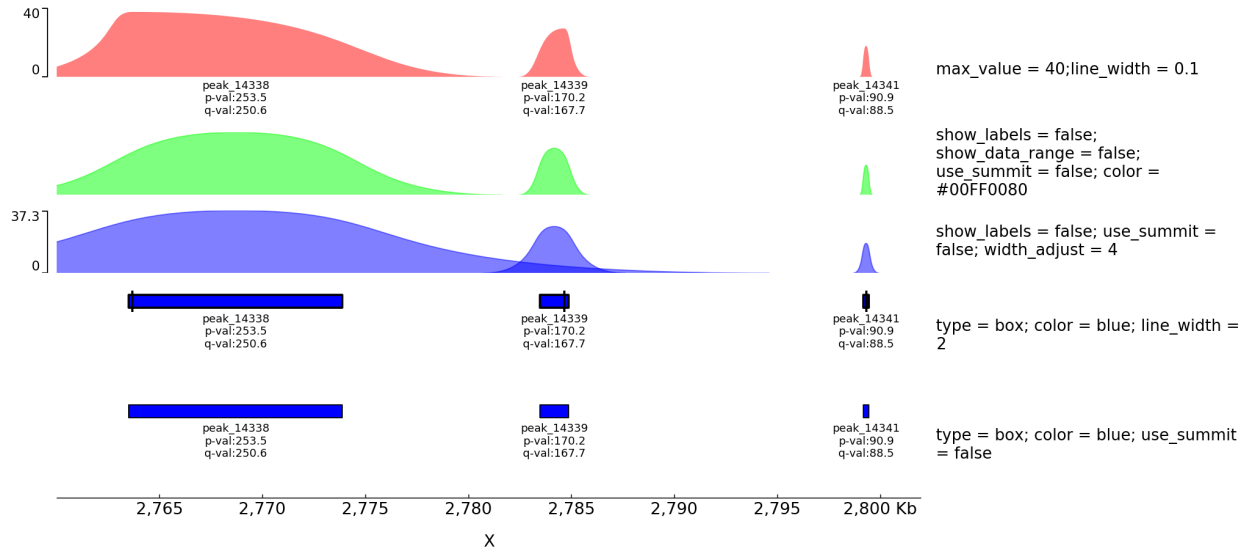
```
[narrow]
file = test2.narrowPeak
height = 4
max_value = 40
line_width = 0.1
title = max_value = 40;line_width = 0.1

[narrow 2]
file = test2.narrowPeak
height = 2
show_labels = false
show_data_range =  false
color = #00FF0080
use_summit = false
title = show_labels = false; show_data_range = false; use_summit = false; color =
↪#00FF0080

[spacer]

[narrow 3]
file = test2.narrowPeak
height = 2
show_labels = false
color = #0000FF80
use_summit = false
```

(continues on next page)

(continued from previous page)

```
width_adjust = 4
title = show_labels = false; use_summit = false; width_adjust = 4

[spacer]

[narrow 4]
file = test2.narrowPeak
height = 3
type = box
color = blue
line_width = 2
title = type = box; color = blue; line_width = 2

[spacer]

[narrow 5]
file = test2.narrowPeak
height = 3
type = box
color = blue
use_summit = false
title = type = box; color = blue; use_summit = false

[x-axis]
```

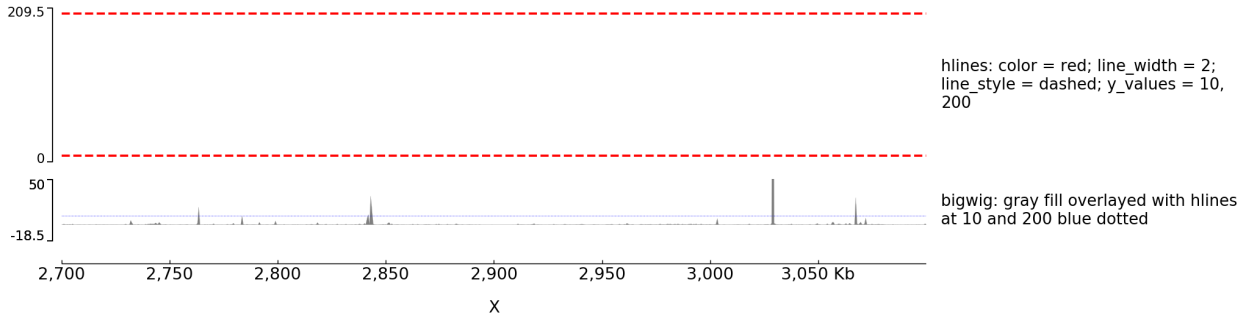
```
$ pyGenomeTracks --tracks narrow_peak2.ini --region X:2760000-2802000 --
↪trackLabelFraction 0.2 --dpi 130 -o master_narrowPeak2.png
```



2.5.5 Example with horizontal lines

```
[test hlines]
color = red
line_width = 2
line_style = dashed
y_values = 10, 200
min_value = 0
show_data_range = true
height = 5
title = hlines: color = red; line_width = 2; line_style = dashed; y_values = 10, 200
file_type = hlines

[spacer]

[test bigwig fill]
file = bigwig2_X_2.5e6_3.5e6.bw
color = gray
height = 2
type = fill
title = bigwig: gray fill overlayed with hlines at 10 and 200 blue dotted
max_value = 50

[test hlines ovelayed]
color = blue
line_style = dotted
y_values = 10, 200
overlay_previous = share-y
file_type = hlines

[spacer]

[x-axis]
```

```
$ pyGenomeTracks --tracks hlines.ini --region X:2700000-3100000 --trackLabelFraction␣
↪0.2 --dpi 130 -o master_hlines.png
```



2.5.6 Examples with Epilogos

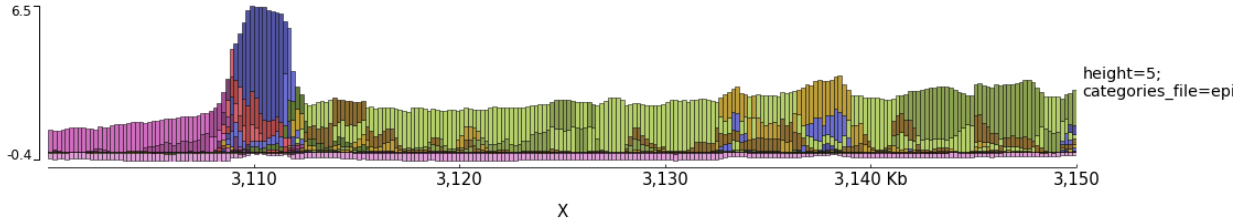
pyGenomeTracks can be used to visualize epigenetic states (for example from chromHMM) as epilogos. For more information see: https://epilogos.altiusinstitute.org/

To plot epilogos a `qcat` file is needed. This file can be crated using the epilogos software (https://github.com/Altius/epilogos).

An example track file for epilogos looks like:

```
[epilogos]
file = epilog.qcat.bgz
height = 5
title = height=5; categories_file=epilog_cats.json

[x-axis]
```

```
$ pyGenomeTracks  --tracks epilogos_track.ini --region X:3100000-3150000 -o epilogos_
↪track.png
```



The color of the bars can be set by using a `json` file. The structure of the file is like this

```
{
"categories":{
          "1":["Active TSS","#ff0000"],
          "2":["Flanking Active TSS","#ff4500"],
          "3":["Transcr at gene 5\" and 3\"","#32cd32"],
          "4":["Strong transcription","#008000"],
          "5":["Weak transcription","#006400"],
          "6":["Genic enhancers","#c2e105"],
          "7":["Enhancers","#ffff00"],
          "8":["ZNF genes & repeats","#66cdaa"],
          "9":["Heterochromatin","#8a91d0"],
          "10":["Bivalent/Poised TSS","#cd5c5c"],
          "11":["Flanking Bivalent TSS/Enh","#e9967a"],
          "12":["Bivalent Enhancer","#bdb76b"],
          "13":["Repressed PolyComb","#808080"],
          "14":["Weak Repressed PolyComb","#c0c0c0"],
          "15":["Quiescent/Low","#ffffff"]
        }
}
```

In the following examples the top epilogo has the custom colors and the one below is shown inverted.

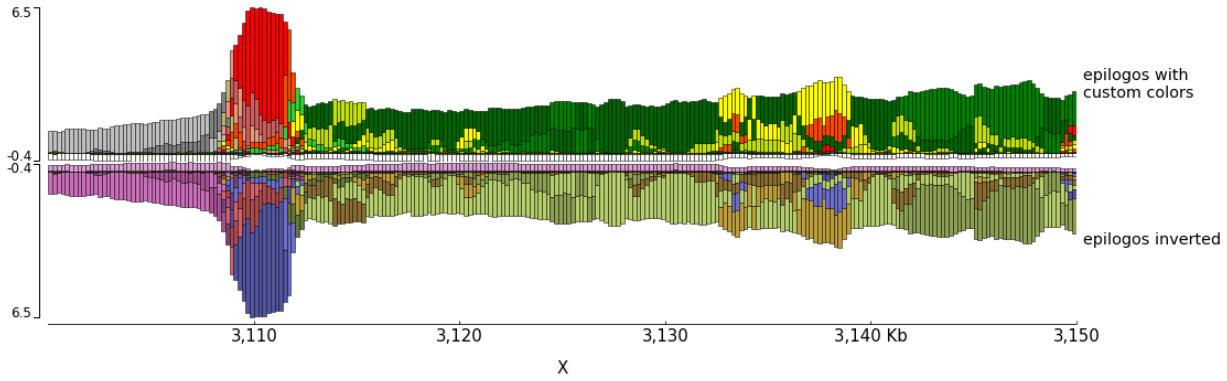
```
[epilogos]
file = epilog.qcat.bgz
height = 5
title = epilogos with custom colors
categories_file = epilog_cats.json
```

(continues on next page)

(continued from previous page)

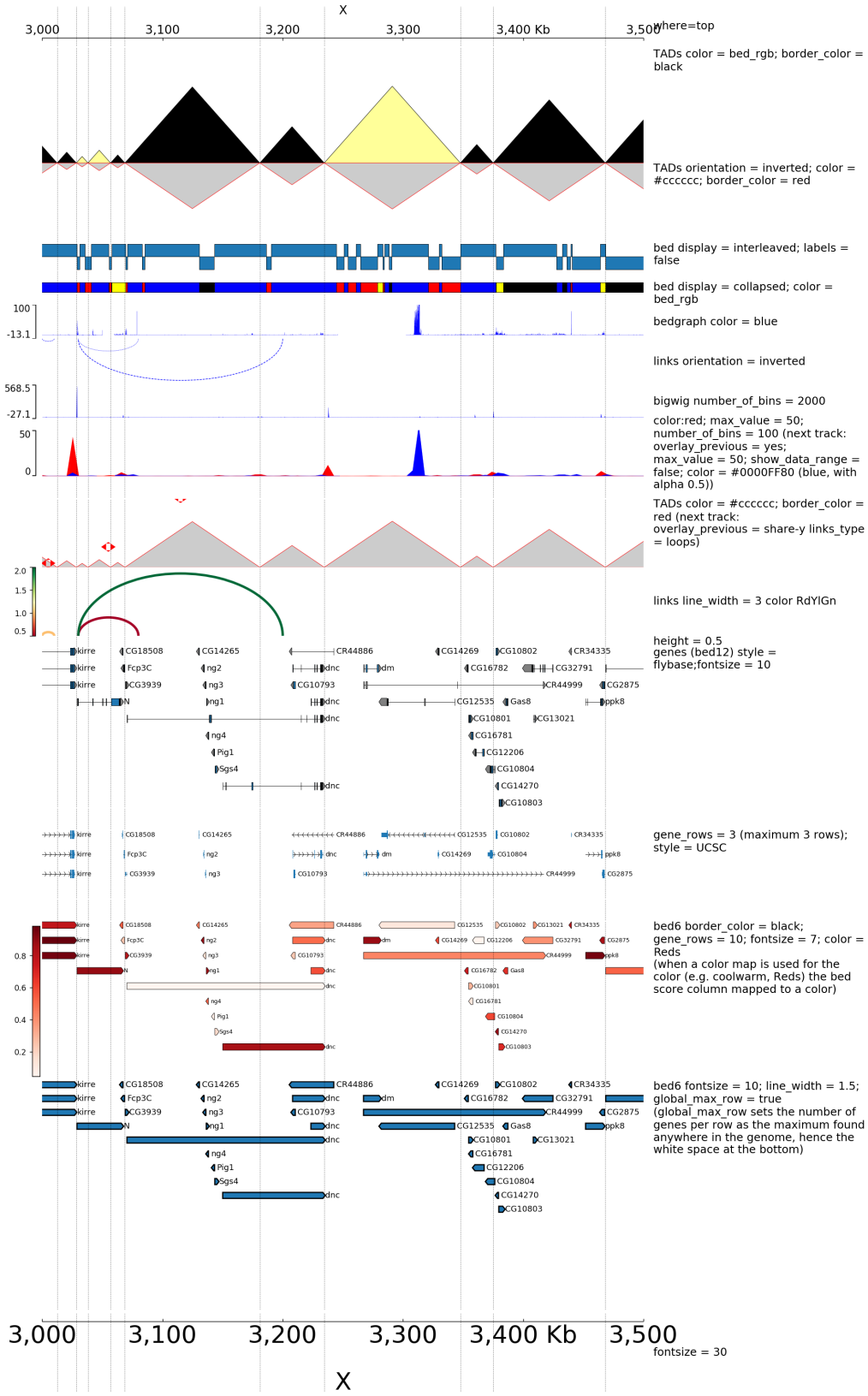
```
[epilogos inverted]
file = epilog.qcat.bgz
height = 5
title = epilogos inverted
orientation = inverted

[x-axis]
```

```
$ pyGenomeTracks  --tracks epilogos_track2.ini --region X:3100000-3150000 -o epilogos_
↪track2.png
```



2.5.7 Examples with multiple options

A comprehensive example of pyGenomeTracks can be found as part of our automatic testing. Note, that pyGenomeTracks also allows the combination of multiple tracks into one using the parameter: `overlay_previous = yes` or `overlay_previous = share-y`. In the second option the y-axis of the tracks that overlays is the same as the track being overlay. Multiple tracks can be overlay together.


X
3,000
3,100
3,200
3,300
3,400 Kb
3,500
where=top
TADs color = bed_rgb; border_color = black
TADs orientation = inverted; color = #cccccc; border_color = red
bed display = interleaved; labels = false
bed display = collapsed; color = bed_rgb
100
-13.1
bedgraph color = blue
links orientation = inverted
568.5
-27.1
bigwig number_of_bins = 2000
50
0
color:red; max_value = 50; number_of_bins = 100 (next track: overlay_previous = yes; max_value = 50; show_data_range = false; color = #0000FF80 (blue, with alpha 0.5))
TADs color = #cccccc; border_color = red (next track: overlay_previous = share-y links_type = loops)
2.0
1.5
1.0
0.5
links line_width = 3 color RdYlGn
height = 0.5
genes (bed12) style = flybase;fontsize = 10
kirre
CG18508
CG14265
CR44886
CG14269
CG10802
CR34335
Fcp3C
ng2
dnc
dm
CG16782
CG32791
CG3939
ng3
CG10793
CR44999
CG2875
N
ng1
CG12535
Gas8
ppk8
CG10801
CG13021
CG16781
ng4
Pig1
CG12206
Sgs4
CG10804
CG14270
CG10803
gene_rows = 3 (maximum 3 rows); style = UCSC
bed6 border_color = black; gene_rows = 10; fontsize = 7; color = Reds
(when a color map is used for the color (e.g. coolwarm, Reds) the bed score column mapped to a color)
0.8
0.6
0.4
0.2
bed6 fontsize = 10; line_width = 1.5; global_max_row = true
(global_max_row sets the number of genes per row as the maximum found anywhere in the genome, hence the white space at the bottom)
3,000
3,100
3,200
3,300
3,400 Kb
3,500
fontsize = 30
X

The configuration file for this image is:

```
[x-axis]
where = top
title = where=top

[spacer]
height = 0.05

[tads]
file = tad_classification.bed
title = TADs color = bed_rgb; border_color = black
file_type = domains
border_color = black
color = bed_rgb
height = 5

[tads 2]
file = tad_classification.bed
title = TADs orientation = inverted; color = #cccccc; border_color = red
file_type = domains
border_color = red
color = #cccccc
orientation = inverted
height = 3

[spacer]
height = 0.5

[tad state]
file = chromatinStates_kc.bed.gz
height = 1.2
title = bed display = interleaved; labels = false
display = interleaved
labels = false

[spacer]
height = 0.5

[tad state]
file = chromatinStates_kc.bed.gz
height = 0.5
title = bed display = collapsed; color = bed_rgb
labels = false
color = bed_rgb
display = collapsed

[spacer]
height = 0.5

[test bedgraph]
file = bedgraph_chrx_2e6_5e6.bg
color = blue
height = 1.5
title = bedgraph color = blue
max_value = 100

[test arcs]
```

(continues on next page)

(continued from previous page)

```
file = test.arcs
title = links orientation = inverted
orientation = inverted
line_style = dashed
height = 2

[test bigwig]
file = bigwig2_X_2.5e6_3.5e6.bw
color = blue
height = 1.5
title = bigwig number_of_bins = 2000
number_of_bins = 2000

[spacer]

[test bigwig overlay]
file = bigwig2_X_2.5e6_3.5e6.bw
color = red
title = color:red; max_value = 50; number_of_bins = 100 (next track: overlay_previous␣
↪= yes;
        max_value = 50; show_data_range = false; color = #0000FF80 (blue, with alpha␣
↪0.5))
min_value = 0
max_value = 50
height = 2
number_of_bins = 100

[test bigwig overlay]
file = bigwig_chrx_2e6_5e6.bw
color = #0000FF80
title =
min_value = 0
max_value = 50
show_data_range = false
overlay_previous = yes
number_of_bins = 100

[spacer]
height = 1

[tads 3]
file = tad_classification.bed
title = TADs color = #cccccc; border_color = red (next track:
        overlay_previous = share-y links_type = loops)
file_type = domains
border_color = red
color = #cccccc
height = 3

[test arcs overlay]
file = test.arcs
color = red
line_width = 10
links_type = loops
overlay_previous = share-y

[test arcs]
```

(continues on next page)

(continued from previous page)

```
file = test.arcs
line_width = 3
color = RdYlGn
title = links line_width = 3 color RdYlGn
height = 3

[spacer]
height = 0.5
title = height = 0.5

[genes 2]
file = dm3_genes.bed.gz
height = 7
title = genes (bed12) style = flybase;fontsize = 10
style = flybase
fontsize = 10

[spacer]
height = 1

[test gene rows]
file = dm3_genes.bed.gz
height = 3
title = gene_rows = 3 (maximum 3 rows); style = UCSC
fontsize = 8
style = UCSC
gene_rows = 3

[spacer]
height = 1

[test bed6]
file = dm3_genes.bed6.gz
height = 7
title = bed6 border_color = black; gene_rows = 10; fontsize = 7; color = Reds
        (when a color map is used for the color (e.g. coolwarm, Reds) the bed
        score column mapped to a color)
fontsize = 7
file_type = bed
color = Reds
border_color = black
gene_rows = 10

[test bed6]
file = dm3_genes.bed6.gz
height = 10
title = bed6 fontsize = 10; line_width = 1.5; global_max_row = true
        (global_max_row sets the number of genes per row as the maximum found
        anywhere in the genome, hence the white space at the bottom)
fontsize = 10
file_type = bed
global_max_row = true
line_width = 1.5

[x-axis]
fontsize = 30
title = fontsize = 30
```

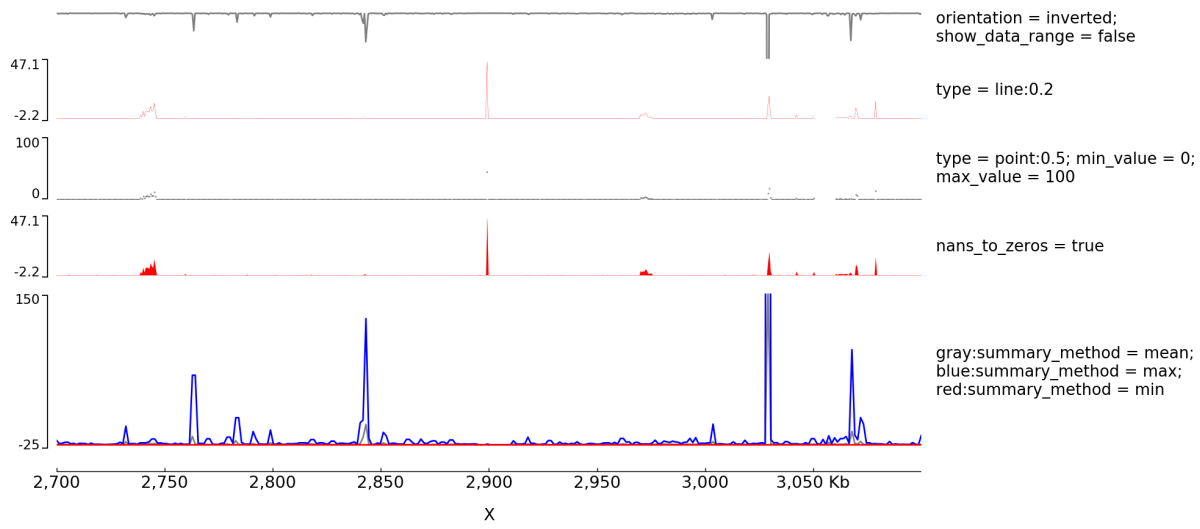
(continues on next page)

(continued from previous page)

```
[vlines]
file = tad_classification.bed
type = vlines
```

```
$ pyGenomeTracks  --tracks browser_tracks.ini --region X:3000000-3500000 --
↪trackLabelFraction 0.2 --width 38 --dpi 130  -o master_plot.png
```

2.5.8 Examples with multiple options for bigwig tracks



The configuration file for this image is:

```
[test bigwig lines]
file = bigwig2_X_2.5e6_3.5e6.bw
color = gray
height = 2
type = line
title = orientation = inverted; show_data_range = false
orientation = inverted
show_data_range = false
max_value = 50

[test bigwig lines:0.2]
file = bigwig_chrx_2e6_5e6.bw
color = red
height = 2
type = line:0.2
title = type = line:0.2
min_value = auto
max_value = auto

[spacer]

[test bigwig points]
```

(continues on next page)

(continued from previous page)

```
file = bigwig_chrx_2e6_5e6.bw
color = black
height = 2
min_value = 0
max_value = 100
type = points:0.5
title = type = point:0.5; min_value = 0; max_value = 100

[spacer]

[test bigwig nans to zeros]
file = bigwig_chrx_2e6_5e6.bw
color = red
height = 2
nans_to_zeros = true
title = nans_to_zeros = true

[spacer]

[test bigwig mean]
file = bigwig2_X_2.5e6_3.5e6.bw
color = gray
height = 5
title = gray:summary_method = mean; blue:summary_method = max;
        red:summary_method = min
type = line
summary_method = mean
max_value = 150
min_value = -5
show_data_range = false
number_of_bins = 300

[test bigwig max]
file = bigwig2_X_2.5e6_3.5e6.bw
#title = test
color = blue
type = line
summary_method = max
max_value = 150
min_value = -15
show_data_range = false
overlay_previous = share-y
number_of_bins = 300

[test bigwig min]
file = bigwig2_X_2.5e6_3.5e6.bw
color = red
type = line
summary_method = min
max_value = 150
min_value = -25
overlay_previous = share-y
number_of_bins = 300

[spacer]

[x-axis]
```

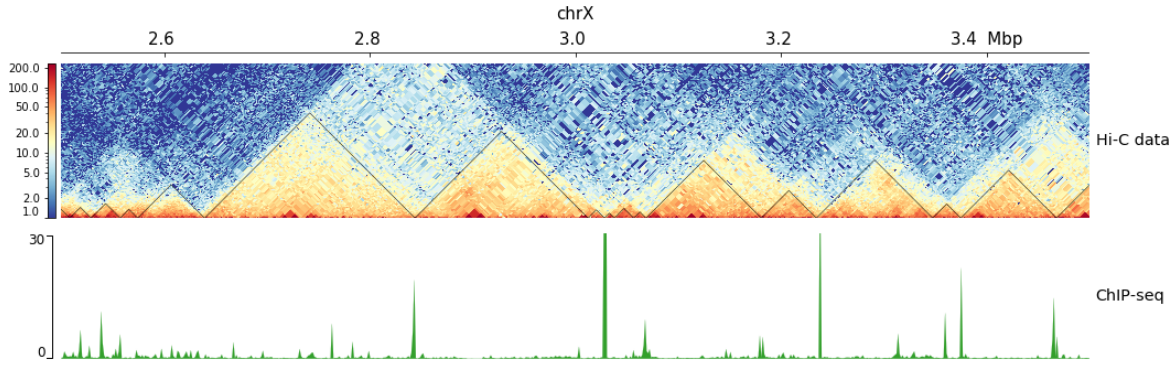
```
$ pyGenomeTracks  --tracks bigwig.ini --region X:2700000-3100000 --trackLabelFraction␣
↪0.2 --dpi 130 -o master_bigwig.png
```

2.5.9 Examples with Hi-C data

The following is an example with Hi-C data overlay with topologically associating domains (TADs) and a bigwig file.

```
[x-axis]
where = top

[hic matrix]
file = hic_data.h5
title = Hi-C data
# depth is the maximum distance plotted in bp. In Hi-C tracks
# the height of the track is calculated based on the depth such
# that the matrix does not look deformed
depth = 300000
transform = log1p
file_type = hic_matrix

[tads]
file = domains.bed
display = triangles
border_color = black
color = none
# the tads are overlay over the hic-matrix
# the share-y options sets the y-axis to be shared
# between the Hi-C matrix and the TADs.
overlay_previous = share-y

[spacer]

[bigwig file test]
file = bigwig.bw
# height of the track in cm (optional value)
height = 4
title = ChIP-seq
min_value = 0
max_value = 30
```

```
$ pyGenomeTracks --tracks hic_track.ini -o hic_track.png --region chrX:2500000-3500000
```



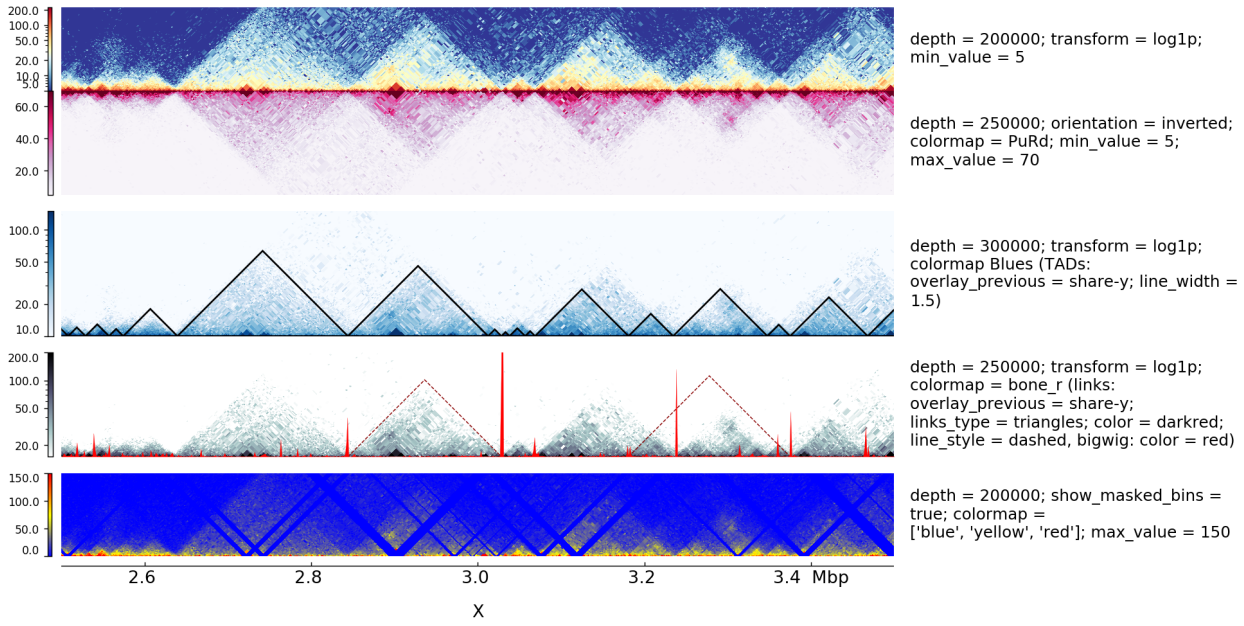
Here is an example where the height was set or not set and the heatmap was rasterized (default) or not rasterized (the dpi was set very low just to show the impact of the parameter).

```
[hic matrix]
file = Li_et_al_2015.cool
title = depth = 200000; transform = log1p; min_value = 5; height = 5
depth = 200000
min_value = 5
transform = log1p
file_type = hic_matrix
show_masked_bins = false
height = 5

[hic matrix 2]
file = Li_et_al_2015.h5
title = same but orientation=inverted; no height
depth = 200000
min_value = 5
transform = log1p
file_type = hic_matrix
show_masked_bins = false
orientation = inverted

[spacer]
height = 0.5

[hic matrix 3]
file = Li_et_al_2015.h5
title = same rasterize = false
depth = 200000
min_value = 5
transform = log1p
file_type = hic_matrix
rasterize = false
show_masked_bins = false

[x-axis]
```

```
$ pyGenomeTracks --tracks browser_tracks_hic_rasterize_height.ini --region X:2500000-
↪2600000 --trackLabelFraction 0.23 --width 38 --dpi 10 -o master_plot_hic_rasterize_
↪height.pdf
```

The output is available here: master_plot_hic_rasterize_height.pdf.

This examples is where the overlay tracks are more useful. Notice that any track can be overlay over a Hi-C matrix. Most useful is to overlay TADs or to overlay links using the `triangles` option that will point in the Hi-C matrix the pixel with the link contact. When overlaying links and TADs is useful to set `overlay_previous=share-y` such that the two tracks match the positions. This is not required when overlying other type of data like a bigwig file that has a different y-scale.



The configuration file for this image is:

```
[hic matrix]
file = Li_et_al_2015.h5
title = depth = 200000; transform = log1p; min_value = 5
depth = 200000
min_value = 5
transform = log1p
file_type = hic_matrix
show_masked_bins = false

[hic matrix]
file = Li_et_al_2015.h5
title = depth = 250000; orientation = inverted; colormap = PuRd; min_value = 5;
        max_value = 70
min_value = 5
max_value = 70
depth = 250000
colormap = PuRd
file_type = hic_matrix
show_masked_bins = false
orientation = inverted

[spacer]
height = 0.5

[hic matrix]
file = Li_et_al_2015.h5
title = depth = 300000; transform = log1p; colormap Blues (TADs:
        overlay_previous = share-y; line_width = 1.5)
colormap = Blues
min_value = 10
max_value = 150
depth = 300000
transform = log1p
```

(continues on next page)

(continued from previous page)

```
file_type = hic_matrix

[tads]
file = tad_classification.bed
#title = TADs color = none; border_color = black
file_type = domains
border_color = black
color = none
height = 5
line_width = 1.5
overlay_previous = share-y
show_data_range = false

[spacer]
height = 0.5

[hic matrix]
file = Li_et_al_2015.h5
title = depth = 250000; transform = log1p; colormap = bone_r (links: overlay_previous␣
↪= share-y;
        links_type = triangles; color = darkred; line_style = dashed, bigwig: color =␣
↪red)
colormap = bone_r
min_value = 15
max_value = 200
depth = 250000
transform = log1p
file_type = hic_matrix
show_masked_bins = false

[test arcs]
file = links2.links
title =
links_type = triangles
line_style = dashed
overlay_previous = share-y
line_width = 0.8
color = darkred
show_data_range = false


[test bigwig]
file = bigwig2_X_2.5e6_3.5e6.bw
color = red
height = 4
title =
overlay_previous = yes
min_value = 0
max_value = 50
show_data_range = false

[spacer]
height = 0.5

[hic matrix]
file = Li_et_al_2015.h5
title = depth = 200000; show_masked_bins = true; colormap =
```

(continues on next page)

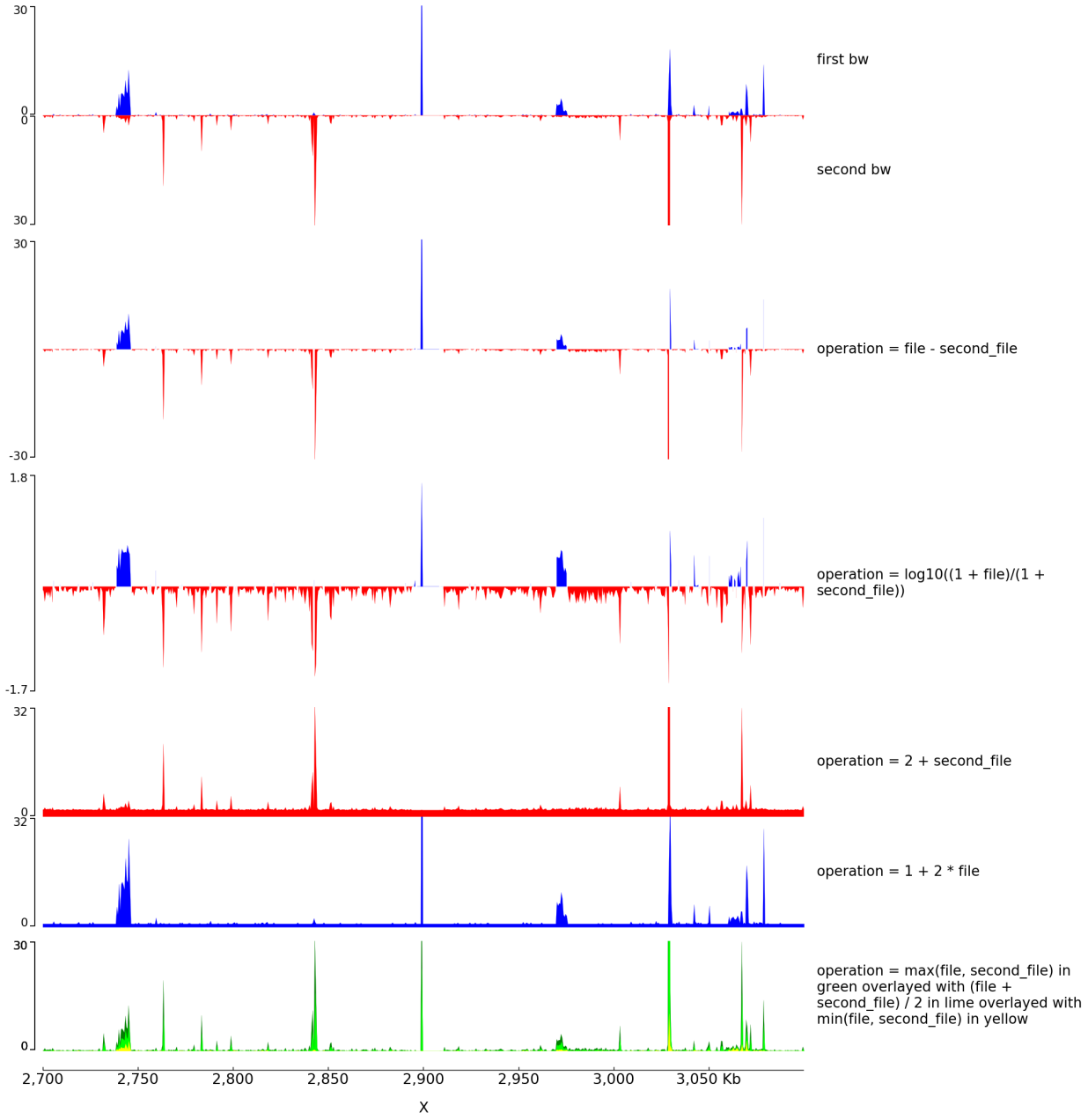
(continued from previous page)

```
        ['blue', 'yellow', 'red']; max_value = 150
depth = 200000
colormap = ['blue', 'yellow', 'red']
max_value = 150
file_type = hic_matrix
show_masked_bins = true

[spacer]
height = 0.1

[x-axis]
```

```
$ pyGenomeTracks  --tracks browser_tracks_hic.ini --region X:2500000-3500000 --
↪trackLabelFraction 0.23 --width 38 --dpi 130 -o master_plot_hic.png
```

2.5.10 Log transform and Operation Examples

With the parameter `operation` you can make operations between one or two files (here two bigwig files but this is also working with two bedgraph files). For example, difference, log ratio, scaling…



The configuration file for this image is:

```
[test bigwig1]
file = bigwig_chrx_2e6_5e6.bw
second_file = bigwig2_X_2.5e6_3.5e6.bw
color = blue
height = 4
title = first bw
min_value = 0
max_value = 30
```

(continues on next page)

(continued from previous page)

```
[test bigwig2]
file = bigwig2_X_2.5e6_3.5e6.bw
color = red
height = 4
title = second bw
min_value = 0
max_value = 30
orientation = inverted

[spacer]
height = 0.5

[test bigwig dif]
file = bigwig_chrx_2e6_5e6.bw
second_file = bigwig2_X_2.5e6_3.5e6.bw
color = blue
negative_color = red
height = 8
title = operation = file - second_file
operation = file - second_file
min_value = -30
max_value = 30
nans_to_zeros = true

[spacer]
height = 0.5

[test bigwig op]
file = bigwig_chrx_2e6_5e6.bw
second_file = bigwig2_X_2.5e6_3.5e6.bw
color = blue
negative_color = red
height = 8
title = operation = log10((1 + file)/(1 + second_file))
operation = log10((1 + file)/(1 + second_file))
nans_to_zeros = true

[spacer]
height = 0.5

[test bigwig op2]
file = bigwig_chrx_2e6_5e6.bw
second_file = bigwig2_X_2.5e6_3.5e6.bw
color = red
height = 4
title = operation = 2 + second_file
operation = 2 + second_file
nans_to_zeros = true
max_value = 32
min_value = 0

[test bigwig op2bis]
file = bigwig_chrx_2e6_5e6.bw
second_file = bigwig2_X_2.5e6_3.5e6.bw
color = blue
height = 4
title = operation = 1 + 2 * file
```

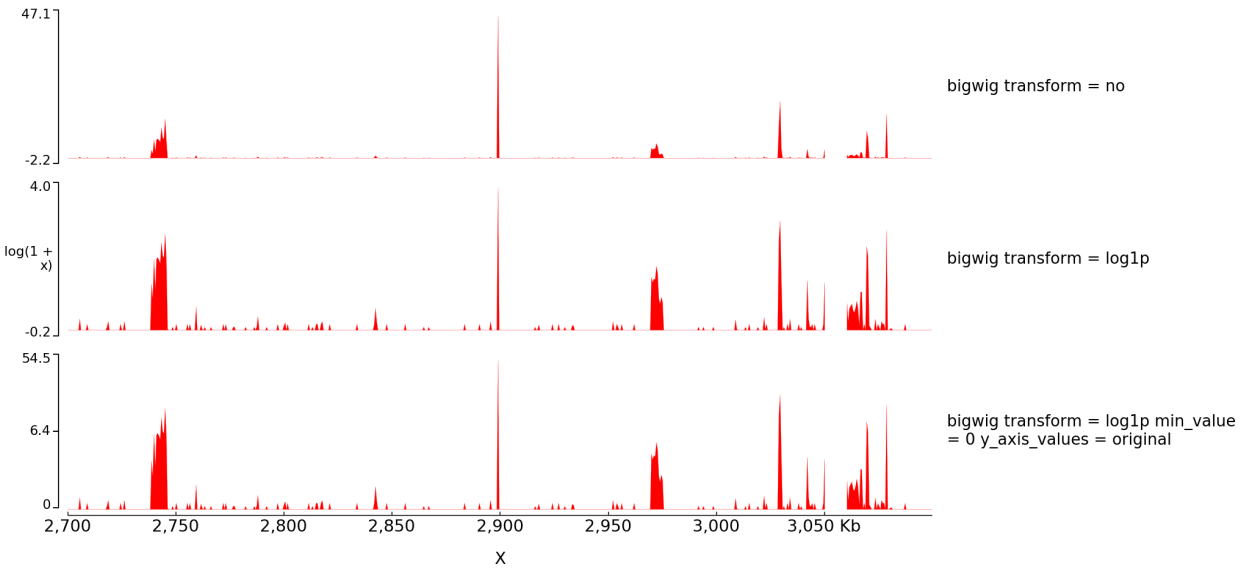
(continues on next page)

(continued from previous page)

```
operation = 1 + 2 * file
nans_to_zeros = true
max_value = 32
min_value = 0

[spacer]
height = 0.5

[test bigwig op3]
file = bigwig_chrx_2e6_5e6.bw
second_file = bigwig2_X_2.5e6_3.5e6.bw
color = green
height = 4
title = operation = max(file, second_file) in green overlayed with (file + second_
↪file) / 2 in lime overlayed with min(file, second_file) in yellow
operation = max(file, second_file)
nans_to_zeros = true
max_value = 30
min_value = 0

[test bigwig op4]
file = bigwig_chrx_2e6_5e6.bw
second_file = bigwig2_X_2.5e6_3.5e6.bw
color = lime
operation = (file + second_file) / 2
nans_to_zeros = true
overlay_previous = share-y

[test bigwig op5]
file = bigwig_chrx_2e6_5e6.bw
second_file = bigwig2_X_2.5e6_3.5e6.bw
color = yellow
operation = min(file, second_file)
nans_to_zeros = true
overlay_previous = share-y

[spacer]
height = 0.5


[x-axis]
```

```
$ pyGenomeTracks --tracks operation.ini --region X:2700000-3100000 --
↪trackLabelFraction 0.2 --dpi 130 -o master_operation.png
```

With the parameter `transformation` you can log transform your data and decide to put on the y axis either the transformed values or the original values:

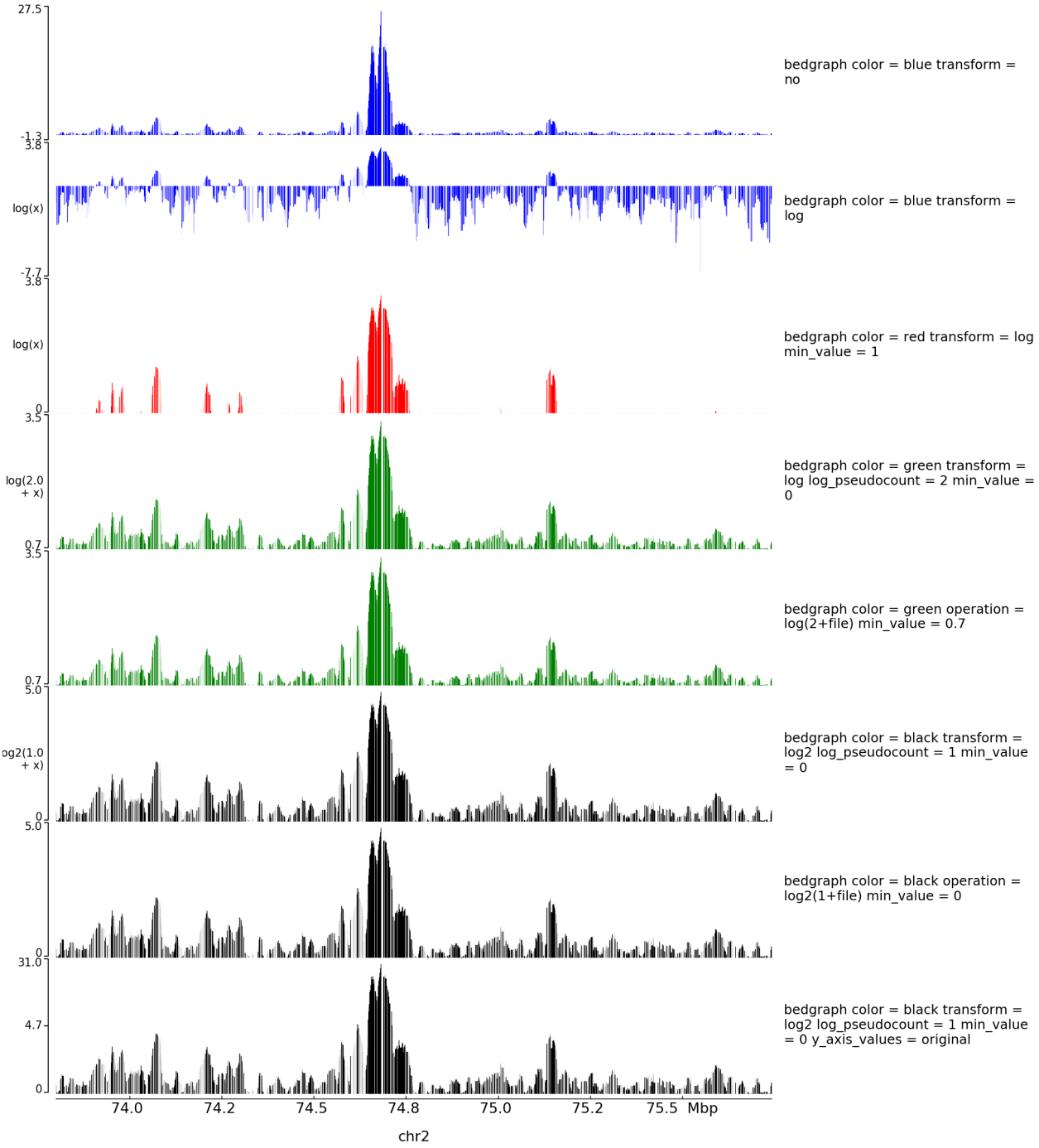


The configuration file for this image is:

```
[test bigwig]
file = bigwig_chrx_2e6_5e6.bw
color = red
height = 5
transform = no
title = bigwig transform = no

[spacer]

[test bigwig log]
file = bigwig_chrx_2e6_5e6.bw
color = red
height = 5
transform = log1p
title = bigwig transform = log1p

[spacer]

[test bigwig log]
file = bigwig_chrx_2e6_5e6.bw
color = red
min_value = 0
height = 5
transform = log1p
title = bigwig transform = log1p min_value = 0 y_axis_values = original
y_axis_values = original

[x-axis]
```

```
$ pyGenomeTracks --tracks log1p.ini --region X:2700000-3100000 --trackLabelFraction 0.
↪2 --dpi 130 -o master_log1p.png
```

With `operation` you can also do log transformation however nothing will be written on the left of the y axis:



The configuration file for this image is:

```
[test bedgraph]
file = GSM3182416_E12DHL_WT_Hoxd11vp.bedgraph.gz
color = blue
height = 5
title = bedgraph color = blue transform = no
```

(continues on next page)

(continued from previous page)

```
transform = no

[test bedgraph]
file = GSM3182416_E12DHL_WT_Hoxd11vp.bedgraph.gz
color = blue
height = 5
title = bedgraph color = blue transform = log
transform = log

[test bedgraph]
file = GSM3182416_E12DHL_WT_Hoxd11vp.bedgraph.gz
color = red
height = 5
title = bedgraph color = red transform = log min_value = 1
min_value = 1
transform = log

[test bedgraph]
file = GSM3182416_E12DHL_WT_Hoxd11vp.bedgraph.gz
color = green
height = 5
title = bedgraph color = green transform = log log_pseudocount = 2 min_value = 0
transform = log
log_pseudocount = 2
min_value = 0

[test bedgraph with operation]
file = GSM3182416_E12DHL_WT_Hoxd11vp.bedgraph.gz
color = green
height = 5
title = bedgraph color = green operation = log(2+file) min_value = 0.7
operation = log(2+file)
min_value = 0.7

[test bedgraph]
file = GSM3182416_E12DHL_WT_Hoxd11vp.bedgraph.gz
color = black
height = 5
title = bedgraph color = black transform = log2 log_pseudocount = 1 min_value = 0
transform = log2
log_pseudocount = 1
min_value = 0

[test bedgraph]
file = GSM3182416_E12DHL_WT_Hoxd11vp.bedgraph.gz
color = black
height = 5
title = bedgraph color = black operation = log2(1+file) min_value = 0
operation = log2(1+file)
min_value = 0

[test bedgraph]
file = GSM3182416_E12DHL_WT_Hoxd11vp.bedgraph.gz
color = black
height = 5
title = bedgraph color = black transform = log2 log_pseudocount = 1 min_value = 0 y_
↪axis_values = original
```

(continues on next page)

(continued from previous page)

```
transform = log2
log_pseudocount = 1
min_value = 0
y_axis_values = original

[x-axis]
```

```
$ pyGenomeTracks --tracks log.ini  --region chr2:73,800,000-75,744,000 --
↪trackLabelFraction 0.2 --width 38 --dpi 130  -o master_log.png
```

2.6 Possible parameters

Here is a table to summarize which are the parameters that can be use for each of the *file_type* and which is the default value: Empty means this parameter is not used. not set means that by default the parameter is commented.

parameter	*x_axis*	*epilogos*	*links*	*domains*	*bed*	*gtf*	*narrow_peak*	*b*
overlay_previous	no	no	no	no	no	no	no	no
where	bottom							
fontsize	15				12	12		
categories_file		not set						
orientation		not set	not set	not set	not set	not set	not set	no
links_type			arcs					
line_width			not set	0.5	0.5	0.5	1	
line_style			solid					
color			blue	#1f78b4	#1f78b4	#1f78b4	#FF000080	#3
alpha			0.8					1
max_value			not set	not set	not set		not set	no
min_value			not set	not set	not set			no
ylim			not set					
compact_arcs_level			0					
use_middle			false					
border_color				black	black	black		
prefered_name				transcript_name	transcript_name	transcript_name		
merge_transcripts				false	false	false		
labels					true	true		
style					flybase	flybase		
display					stacked	stacked		
max_labels					60	60		
global_max_row					false	false		
gene_rows					not set	not set		
arrow_interval					2	2		
arrowhead_included					false	false		
color_utr					grey	grey		
height_utr					1	1		
arrow_length					not set	not set		
all_labels_inside					false	false		
labels_in_margin					false	false		
show_data_range							true	tr
show_labels							true	

Table 1 – continued from previous page

parameter	*x_axis*	*epilogos*	*links*	*domains*	*bed*	*gtf*	*narrow_peak*	*b*
use_summit							true	
width_adjust							1.5	
type							peak	fi
negative_color								n
nans_to_zeros								fa
summary_method								m
number_of_bins								7
transform								n
log_pseudocount								0
y_axis_values								tr
second_file*								n
operation*								fi
grid								fa
rasterize								
pos_score_in_bin								
plot_horizontal_lines								
colormap								
depth								
show_masked_bins								
scale_factor								
x_center								
size								

* While pyGenomeTracks can convert coverage tracks on the fly, this might be a time-consuming step, especially on large files and if you want to replot many times. In this situation, we recommend using the deepTools suite to convert your files in advance. For example bamCoverage or bamCompare

Some parameters can take only discrete values.

They are summarized here:

- **overlay_previous**:
  - for *x_axis, epilogos, links, domains, bed, gtf, narrow_peak, bigwig, bedgraph, bedgraph_matrix, hlines, hic_matrix, scalebar, spacer*: no, yes, share-y
- **where**:
  - for *x_axis*: top, bottom
  - for *scalebar*: left, right, top, bottom
- **orientation**:
  - for *epilogos, links, domains, bed, gtf, narrow_peak, bigwig, bedgraph, bedgraph_matrix, hlines, hic_matrix*: inverted, not set
- **links_type**:
  - for *links*: arcs, triangles, loops
- **line_style**:
  - for *links, hlines*: solid, dashed, dotted, dashdot
- **compact_arcs_level**:
  - for *links*: 0, 1, 2

- **use_middle**:
  - for *links, bedgraph*: true, false
- **merge_transcripts**:
  - for *domains, bed, gtf*: true, false
- **style**:
  - for *bed, gtf*: flybase, UCSC, tssarrow
- **display**:
  - for *bed, gtf*: collapsed, triangles, interleaved, stacked
- **labels**:
  - for *bed, gtf*: true, false
- **global_max_row**:
  - for *bed, gtf*: true, false
- **arrowhead_included**:
  - for *bed, gtf*: true, false
- **all_labels_inside**:
  - for *bed, gtf*: true, false
- **labels_in_margin**:
  - for *bed, gtf*: true, false
- **type**:
  - for *narrow_peak*: peak, box
  - for *bedgraph_matrix*: matrix, lines
- **show_data_range**:
  - for *narrow_peak, bigwig, bedgraph, bedgraph_matrix, hlines*: true, false
- **show_labels**:
  - for *narrow_peak*: true, false
- **use_summit**:
  - for *narrow_peak*: true, false
- **summary_method**:
  - for *bigwig*: mean, average, max, min, stdev, dev, coverage, cov, sum
  - for *bedgraph*: mean, average, max, min, stdev, dev, coverage, cov, sum, not set
- **transform**:
  - for *bigwig, bedgraph*: no, log, log1p, -log, log2, log10
  - for *hic_matrix*: no, log, log1p, -log
- **y_axis_values**:
  - for *bigwig, bedgraph*: original, transformed
- **nans_to_zeros**:

  - for *bigwig, bedgraph*: true, false

- **grid**:
  - for *bigwig, bedgraph*: true, false
- **rasterize**:
  - for *bedgraph, bedgraph_matrix, hic_matrix*: true, false
- **pos_score_in_bin**:
  - for *bedgraph_matrix*: center, block
- **plot_horizontal_lines**:
  - for *bedgraph_matrix*: true, false
- **show_masked_bins**:
  - for *hic_matrix*: true, false

2.7 Adding new tracks

Adding new tracks to pyGenomeTracks only requires adding a new class to the `pygenometracks/tracks` folder. The name of the file must end with `Track.py`. The class must inherit the `GenomeTrack` (or other track class available) and must have a `plot` method. In order to work well with the config checker it should also have some global variable: - `DEFAULTS_PROPERTIES` is a dictionary where each key is a parameter and each value is the default value when it is not set or when something goes wrong. - `NECESSARY_PROPERTIES` is an array with all the parameters which are necessary for this track (usually 'file') - `SYNONYMOUS_PROPERTIES` is a dictionary where each key is a parameter, each value is a dictionnary where each key is a string that should be replaced by the value (for example, `SYNONYMOUS_PROPERTIES = {'max_value': {'auto': None}}`) - `POSSIBLE_PROPERTIES` is a dictionary where each key is a parameter, each value is an array with the only possible values for this parameter, if the value specified by the user is not part of them, it will be substituted by the default value. - `BOOLEAN_PROPERTIES` is an array with all parameters that should have a boolean value (a boolean value can be 0, 1, true, false, on, off) - `STRING_PROPERTIES` is an array with all parameters that have string values. It should always contains `title` and `file_type`. - `FLOAT_PROPERTIES` is a dictionnary where each key is a parameter, each value is an array with the min value (included) and the max value (included) that should have the parameter (You can use `[- np.inf, np.inf]` if there is no restriction). This dictionary should always contains `'height': [0, np.inf]` - `INTEGER_PROPERTIES` same as `FLOAT_PROPERTIES` for integer values.

Additionally, some basic description should be added.

For example, to make a track that prints 'hello world' at a given location looks like this:

```
# -*- coding: utf-8 -*-
from . GenomeTrack import GenomeTrack
import numpy as np


class TextTrack(GenomeTrack):
    SUPPORTED_ENDINGS = ['.txt']  # this is used by make_tracks_file to guess the␣
↪type of track based on file name
    TRACK_TYPE = 'text'
    OPTIONS_TXT = """
height = 3
title =
text =
# x position of text in the plot (in bp)
```

(continues on next page)

(continued from previous page)

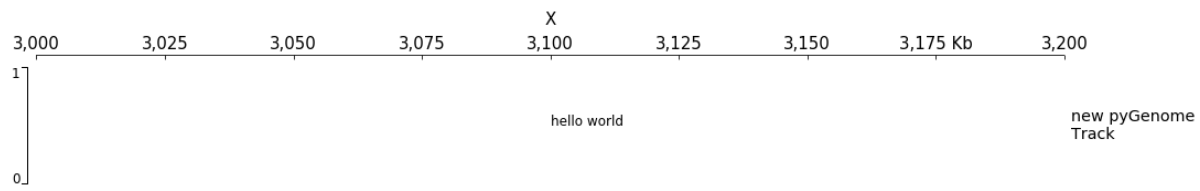
```
x position =
"""
    DEFAULTS_PROPERTIES = {'text': 'hello world'}
    NECESSARY_PROPERTIES = ['x_position']
    SYNONYMOUS_PROPERTIES = {}
    POSSIBLE_PROPERTIES = {}
    BOOLEAN_PROPERTIES = []
    STRING_PROPERTIES = ['text', 'title', 'file_type']
    FLOAT_PROPERTIES = {'height': [0, np.inf],
                        'x_position': [0, np.inf]}
    INTEGER_PROPERTIES = {}

    def plot(self, ax, chrom, region_start, region_end):
        """
        This example simply plots the given title at a fixed
        location in the axis. The chrom, region_start and region_end
        variables are not used.
        Args:
            ax: matplotlib axis to plot
            chrom_region: chromosome name
            start_region: start coordinate of genomic position
            end_region: end coordinate
        """
        # print text at position x = self.properties['x position'] and y = 0.5␣
↪(center of the plot)
        ax.text(self.properties['x_position'], 0.5, self.properties['text'])
```

The `OPTIONS_TXT` should contain the text to build a default configuration file. This information, together with the information about SUPPORTED_ENDINGS is used by the program `make_tracks_file` to create a default configuration file based on the endings of the files given.

The configuration file is:

```
[x-axis]
where = top

[new track]
file =
height = 4
title = new pyGenomeTrack
file_type = text
text = hello world
x position = 3100000
```

```
$ pyGenomeTracks  --tracks new_track.ini --region X:3000000-3200000 -o new_track.png
```



Notice that the resulting track already includes a y-axis (to the left) and a label to the right. Those are the defaults that can be changed by adding a `plot_y_axis` and `plot_label` methods.

Another more complex example is the plotting of multiple bedgraph data as matrices. The output of `HiCExplorer`

`hicFindTADs` produces a file whose data format is similar to a bedgraph but with more value columns. We call this a bedgraph matrix. The following track plot this bedgraph matrix:

```
# -*- coding: utf-8 -*-
import numpy as np
from . BedGraphTrack import BedGraphTrack
from . GenomeTrack import GenomeTrack


class BedGraphMatrixTrack(BedGraphTrack):
    # this track class extends a BedGraphTrack that is already part of
    # pyGenomeTracks. The advantage of extending this class is that
    # we can re-use the code for reading a bedgraph file
    SUPPORTED_ENDINGS = ['.bm', '.bm.gz']
    TRACK_TYPE = 'bedgraph_matrix'
    OPTIONS_TXT = GenomeTrack.OPTIONS_TXT + f"""
# a bedgraph matrix file is like a bedgraph, except that per bin there
# are more than one value (separated by tab). This file type is
# produced by the HiCExplorer tool hicFindTads and contains
# the TAD-separation score at different window sizes.
# E.g.
# chrX        18279        40131        0.399113        0.364118        0.
↪320857        0.274307
# chrX        40132        54262        0.479340        0.425471        0.
↪366541        0.324736
#min_value = 0.10
#max_value = 0.70
file_type = {TRACK_TYPE}
"""
    DEFAULTS_PROPERTIES = {'max_value': None,
                           'min_value': None,
                           'show_data_range': True,
                           'orientation': None}
    NECESSARY_PROPERTIES = ['file']
    SYNONYMOUS_PROPERTIES = {'max_value': {'auto': None},
                             'min_value': {'auto': None}}
    POSSIBLE_PROPERTIES = {'orientation': [None, 'inverted']}
    BOOLEAN_PROPERTIES = ['show_data_range']
    STRING_PROPERTIES = ['file', 'file_type', 'overlay_previous',
                         'orientation', 'title']
    FLOAT_PROPERTIES = {'max_value': [- np.inf, np.inf],
                        'min_value': [- np.inf, np.inf],
                        'height': [0, np.inf]}
    INTEGER_PROPERTIES = {}

    # In BedGraphTrack the method set_properties_defaults
    # has been adapted to a coverage track. Here we want to
    # go back to the initial method:
    def set_properties_defaults(self):
        GenomeTrack.set_properties_defaults(self)

    def plot(self, ax, chrom_region, start_region, end_region):
        """
        Args:
            ax: matplotlib axis to plot
            chrom_region: chromosome name
            start_region: start coordinate of genomic position
            end_region: end coordinate
```

(continues on next page)

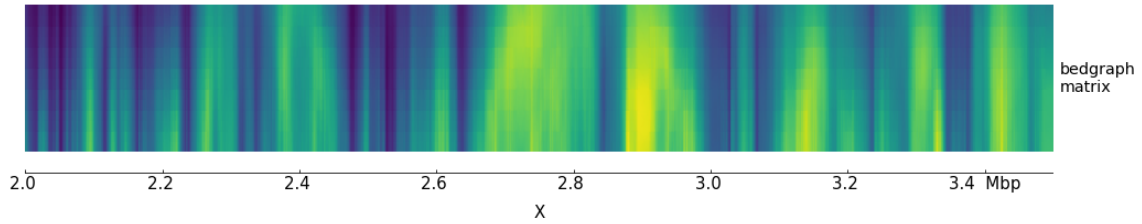
(continued from previous page)

```
        """
        start_pos = []
        matrix_rows = []

        # the BedGraphTrack already has methods to read files
        # in which the first three columns are chrom, start,end
        # here we used the interval_tree method inherited from the
        # BedGraphTrack class
        for region in sorted(self.interval_tree[chrom_region][start_region -␣
↪10000:end_region + 10000]):
            start_pos.append(region.begin)
            # the region.data contains all the values for a given region
            # In the following code, such list is converted to floats and
            # appended to a new list.
            values = list(map(float, region.data))
            matrix_rows.append(values)

        # using numpy, the list of values per line in the bedgraph file
        # is converted into a matrix whose columns contain
        # the bedgraph values for the same line (notice that
        # the matrix is transposed to achieve this)
        matrix = np.vstack(matrix_rows).T

        # using meshgrid we get x and y positions to plot the matrix at
        # corresponding positions given in the bedgraph file.
        x, y = np.meshgrid(start_pos, np.arange(matrix.shape[0]))

        # shading adds some smoothing to the pllot
        shading = 'gouraud'
        vmax = self.properties['max_value']
        vmin = self.properties['min_value']

        img = ax.pcolormesh(x, y, matrix, vmin=vmin, vmax=vmax, shading=shading)
        img.set_rasterized(True)

    def plot_y_axis(self, ax, plot_axis):
        """turn off y_axis plot"""
        pass

    def __del__(self):
        if self.tbx is not None:
            self.tbx.close()
```

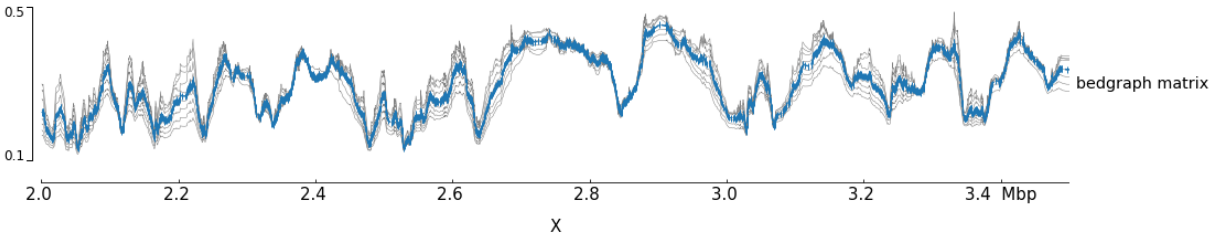
Let's create a track for this:

```
[bedgraph matrix]
file = tad_separation_score.bm.gz
file_type = bedgraph_matrix
title = bedgraph matrix
height = 5

[spacer]

[x-axis]
```

```
$ pyGenomeTracks  --tracks bedgraph_matrix.ini --region X:2000000-3500000 -o bedgraph_
↪matrix.png
```



Although this image looks interesting another way to plot the data is a overlapping lines with the mean value highlighted. Using the bedgraph version of `pyGenomeTracks` the following image can be obtained:



2.8 Releases

2.8.1 3.5.1

Bugfixes:

- Get a message when bedtools is installed instead of crashing without any message.
- Always test if a bedgraph is tabix indexed without checking the extension
- Fix a bug which was happening when `operation` or `summary_method` was used on bedgraph whereas the bedgraph had some missing intervals.
- Enforcing version 15 of HiCMatrix. Version 14 had a bug concerning the application of the correction factors of cool files.

2.8.2 3.5

Enhancements:

- pyGenomeTracks goes much faster:
  - We now use bedtools to load only a portion of bed, gtf, bedgraph, narrowPeak, bedgraphMatrix, Epilogos, links. This speeds up the process dramatically especially for gtf files.
  - When you use a cool matrix, only the plotted region is loaded, this speeds up the process. If multiple regions on different chromosomes are provided in a BED this option is not used.
  - Unfortunately, it is not possible to speed the process with h5 matrices so you may want to convert them to cool with hicConvertFormat.
- pyGenomeTracks does not require sorted bed anymore.
- For coverage tracks (bedgraph and bigwig), there is a new parameter: `grid` which allows to display horizontal lines.

- For links, you can choose to use the middle of start1 and end1 and the middle of start2 and end2 thanks to `use_middle` instead of the extremities coordinates.
- For Hi-C matrices, mcool files are now supported.
- For vlines, the line_width can now be set.

Minor enhancements:

- Bedgraph with NA values in the 4-th column do not raise error anymore.
- There is a progress bar for links.
- For Hi-C matrices,
  - there is no error when the plot goes over the chromosome size provided by the matrix.
  - if the depth is smaller than binsize it makes an empty plot instead of raising errors.
- When the `overlay_previous` is set in the first track it is ignored and a warning is given instead of giving an error with a not meaningful error message.
- The color for negative values in coverage tracks (bigwig and bedgraph) is now part of the output of `make_tracks_file`.
- Others output of `make_tracks_file` have been modified for better clarity.
- Others code related enhancements which are transparent to the users.
- The format of bed12 is more stringent (the blocks must span the chromStart and chromEnd).
- Improve the documentation regarding the installation. We highly recommand to use conda.

Bugfixes:

- For bed tracks when `gene_rows` or `global_max_row` was set the last row was very close to the bottom (sometimes even overlaying the next track). Now it is vertically centered and do not go over the track height.
- Solve some bugs how non conventional bed files are handle.
- Display more meaningful error messages.
- Only plot scale bar when in the region of plot.
- Do not show a warning when file_type is set for `x_axis` or `spacer`.
- Solve some incompatibilities with python 3.8. However, python 3.8 is not officially supported by now as bioconda do not support it yet https://bioconda.github.io/user/versions.html#python.

2.8.3 3.4

Improved documentation:

- fix url of test_data thanks to @sonnynguyen
- Better documentation for pyGenomeTracks / makeTracksFile
- Documentation of each track

Enhancements:

- Different bed files format (BED3, BED4, BED5, BED6, BED8, BED9, BED12) is guessed based on the first line and rely not only on number of columns but on the type of each column (string/float etc.). This way, broadPeaks/gappedPeaks and other bed-like formats can be accepted as `file_type = bed` without preprocessing.
- gtf has now its own `file_type`. **Warning**:`file_type = bed` for gtf will be no longer supported in a future version.
- You can use a decreasing x-axis (from larger coordinates to smaller coordinates) with the option `--decreasingXAxis` in `pyGenomeTracks`.
- For bedgraph and bigwig:
  - logScale: To apply a log transformation to your data you can set `transform` to `log1p`, `log`, `log2`, `log10`, `-log`. For others than `log1p`, you can set a `log_pseudocount`. By default the y axis values will represent the transformed values and the transformation will be written on the left of the y axis but you can decide to keep the original values by setting `y_axis_values = original`.
  - operation on files: You can plot the result of an operation on one or two files without preprocessing. To use it, put the operation in `operation` with the variable `file` or both variables `file` and `second_file`, for example `operation = 0.89 * file` or `operation = max(file, second_file)` or `operation = file - second_file`. In the two later cases, you need to put the path of the second file in `second_file`. However, this operation can be time consuming if you do it multiple times and you may prefer to convert your files in advance.

  For more examples on these two new features, visit the examples.
- Both `type = line` and `type = lines` are working in bedgraph_matrix

Bugfix:

- In version 3.3, if you were using a colormap in bed the `border_color` and `color_utr` were ignored. Now each one can be set to different color/colormap/bed_rgb.

2.8.4 3.3

Release 3.3 adds:

- a documentation on readthedocs: https://pygenometracks.readthedocs.io
- progress bar for bed, gtf, bedgraph, narrowPeak, epilogos
- all colors can be set as `(r, g, b)` with r, g, b values between 0 and 1.
- all colormap can be set as an array of colors, for example `[red, white, (0, 0, 0.5)]`
- the titles on the right can be align left, right or center with the argument `--trackLabelHAlign`
- **for the links track**
  - `compact_arcs_level` to be able to see both very long arcs and very small arcs.
  - `ylim` to be able to zoom and see small arcs.
- a new track type: `file_type = scalebar` which by default is close to the UCSC scale bar.
- **for the bed tracks**

  - a new gene type: `tssarrow`
  - `border_color` and `color_utr` can be set to `bed_rgb`.
  - `all_labels_inside` allow to put the name of the region on the left if it ends after the plotted region.
  - `labels_in_margin` allow to put the name of the region in the margin if it ends after the plotted region.
- `line_width` can now be set in the narrowPeak tracks.
- `colormap` can now be set in bedgraphMatrix tracks.

bug fixes:

- if a bed with no interval is provided, pgt no longer exit.
- When an exon had both UTR start and UTR end within it, only the UTR start was considered.
- alpha in links tracks was not used
- qcat files with missing values are now working.

2.8.5 3.2.1

- Small bug fix in parser when the parameter had spaces in the name and was a boolean (for example show data range instead of show_data_range).

Warning: In version 4.0 no parameter with space will be accepted.

2.8.6 3.2

This is a major release with a lot of new features but also some deprecations in preparation for a new 4.0 release.

During the years we have introduced several ways to enable/disable settings. We have used on/off, yes/no, 0/1, true/false. From now on we recommend to only use true/false so that we can unify our config files and make it more intuitive for all pgt-users.

We also removed all 2/3/4 words items and concatenate them now by _ #132. For example `line width` is now `line_width`. Thanks @LeilyR!

A few new features in this release:

- Every config is now checked for syntax errors before anything is executed (@lldelisle)
- Added the possibility to merge transcripts into one single gene representation when using gtf (@lldelisle)
- Added the possibility to rasterize bedgraph plots (@lldelisle)
- Added the possibility to use summary functions on bedgraphs (@lldelisle)
- Generate an empty track if a requested region is not in the given track-files. Fixed #120 (@LeilyR)
- Generate an empty track if a chromosome is missing in bedgraph files (@lldelisle)
- Improved UCSC style for intron arrows (@lldelisle)
- Flybase style now supports color_utr and height_uts (@lldelisle)
- A new tracktype `hlines` was added (@lldelisle)

- Allow to plot the arcs of the links with a color scale based on scores as proposed in #30 (@lldelisle)
- Allow to plot rectangle on a heatmap for loops. Fixed #47 (@Phlya, @lldelisle)
- A lot of HiC Matrix improvements (@lldelisle)
- The `alpha` property can now be used nearly everywhere (@lldelisle)

Also checkout our updated readme).

Special thanks to @lldelisle & @LeilyR for such a feature-rich release.

Merry Xmas and a happy new year! Your PGT team!

2.8.7 3.1.2

change infos to warnings

2.8.8 3.1.1

- small bug fix for summit usage (@lldelisle)
- I added a vertical bar in the narrowPeaks when plotted as box to be able to see the summits (@lldelisle)

2.8.9 3.1

- add the ability to control alpha of bedgraph tracks (thanks to @mikewolfe)
- add GTF support (thanks to @lldelisle)
- add support for various BED flavors (thanks to @lldelisle)
- adopt codebase to lastest matplotlib changes (thanks to @lldelisle)
- various new tests, code cleanups and smaller fixes all over

2.8.10 3.0

- support for Python 2 has been removed.
- regions smaller than 200000 can be plotted with a warning message for plotting TADs.
- non existing chromosomes in some or all of the input files can be asked as regions to plot the tracks.

2.8.11 2.1

- Support for epilogos #38
- HiC improvements and migration to use HiCMatrix. Thanks to *@joachimwolff* for these changes.
- Different coloring of negative and positive bigwig values is now possible. Thanks to *@Phlya* for this improvement.

2.8.12 2.0

This release makes it **very easy to add new tracks**! See the Readme for examples on how to do that.

Furthermore, this release:

- Adds Hi-C tracks
- Allows a track be plotted over the previous. There is not limit on the tracks that can be plotted on top of each other
- Removed unnecessary empty space around the resulting image
- Enlarged the README.md file with new examples and a mini-tutorial on how to add new tracks.
- Added chromosome name and title to x-axis plot
- Added option to plot links as 'triangles'. This is useful to overlay links with Hi-C data to identify the contact pixel.
- Added line width to bed track
- Added 'bed graph' track
- Added a track that can plot .narrowPeaks

2.8.13 1.0

This version added python3 compatibility to the code

Main changes in configuration files: Since v3.2:

- there is no more space in the name of the parameters (`line width` become `line_width`)
- all on/off, yes/no, 0/1 become true/false

Since v3.4:

- gtf has its own file_type (`file_type = gtf`).

Warning in version 4.0 the old configuration files will not work anymore

2.9 FAQ

2.9.1 Why the scale of my Hi-C plot suddenly changed?

pyGenomeTracks is using HiCMatrix to read the matrix from `h5` and `cool` format. From version 12 to version 13, a normalization step when reading `cool` file was removed. This normalization was mostly used when you were providing `cool` file from cooler balance.

If you want to keep the old scale you need to downgrade to HiCMatrix version 12 but version 13 also correct some bugs so we advice to change your `max_value` in your parameter file to adjust to the new scale.

2.9.2 No output generated with version 3.5 installed with pip

If you used pyGenomeTracks version 3.5 and the last line you get is:

```
INFO:pygenometracks.tracksClass:initialize x. [xxxxx]
```

It is highly probable that BEDTools is not installed or not loaded in your environment.

2.9.3 My Hi-C plot looks like no correction was applied when using cool matrix

pyGenomeTracks is using HiCMatrix to read the matrix from `cool` format. Unfortunately, a bug was introduced in version 14 ignoring the correction factors. This bug was fixed in version 15 so update HiCMatrix to last version should fix it.