pyGenomeTracks

Release 3.3

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

May 07, 2020

CONTENTS:

CHAPTER

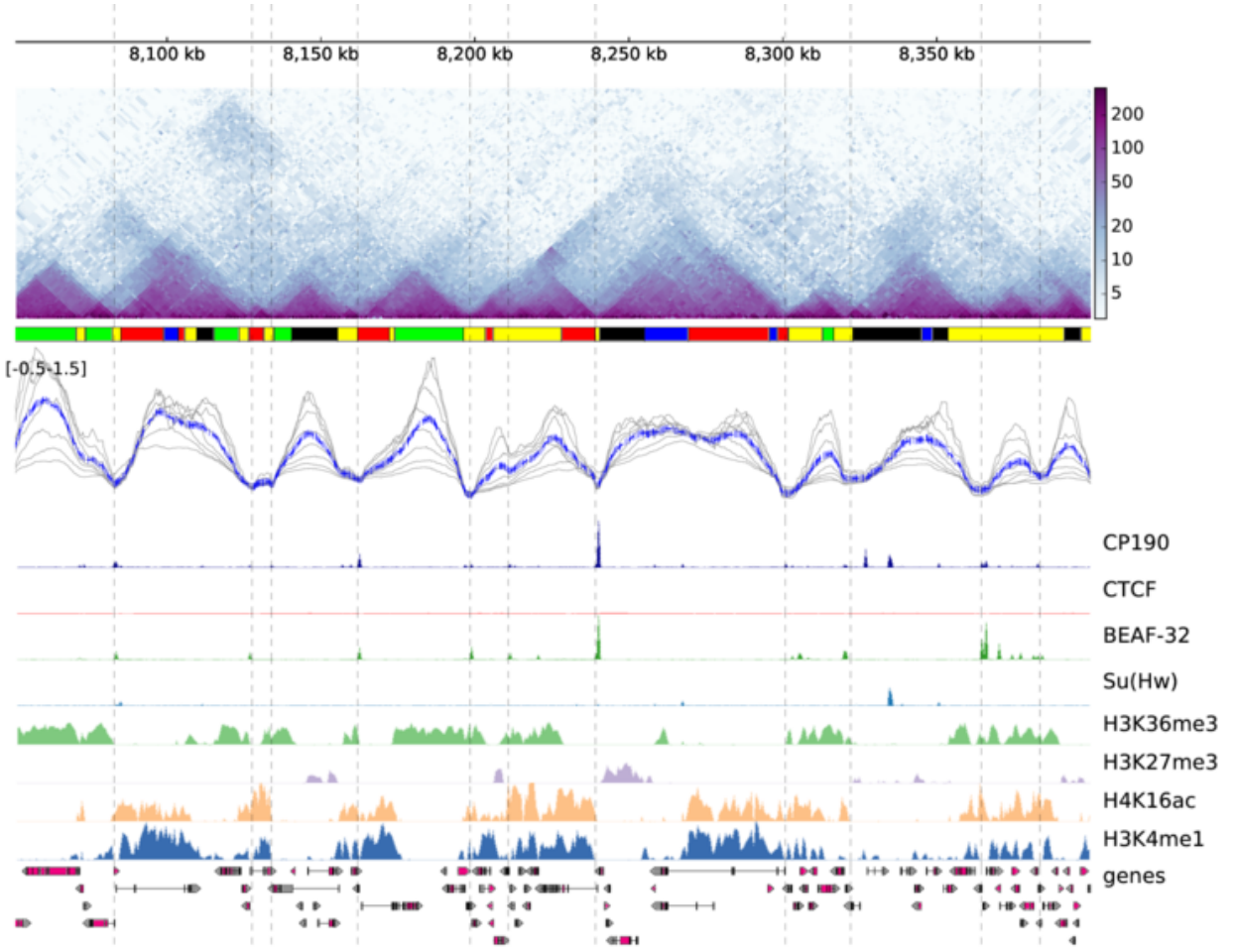
ONE

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 >=3.6
- numpy >= 1.16
- intervaltree >=2.1.0
- pyBigWig >= 0.3.4
- hicmatrix >= 0.14
- pysam >= 0.8
- matplotlib >= 3.1.1
- gffutils >=0.9

The fastest way to obtain **Python 3.6 together with numpy** is via the Anaconda Scientific Python Distribution. Just download the version that's suitable for your operating system and follow the directions for its installation. All of the requirements for pyGenomeTracks can be installed in Anaconda with:

```
$ conda install -c bioconda -c conda-forge pygenometracks
```

2.1.2 Command line installation using `pip`

Install pyGenomeTracks using the following command:

```
$ pip install pyGenomeTracks
```

All python requirements should be automatically installed.

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
```

2.1.3 Command line installation without `pip`

You are highly recommended to use *pip* rather than these more complicated steps.

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

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
```

2.1.4 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. 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 above:

2.2.2 make_tracks_file

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 Examples

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

2.4.1 Basic Examples

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

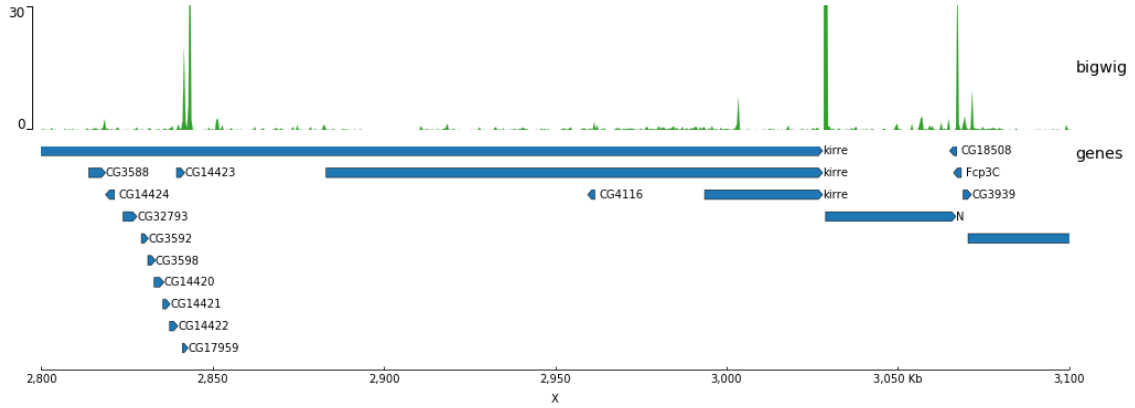
```
[bigwig file test]
file = bigwig.bw
# height of the track in cm (optional value)
height = 4
title = bigwig
min_value = 0
max_value = 30
```

```
$ 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
```

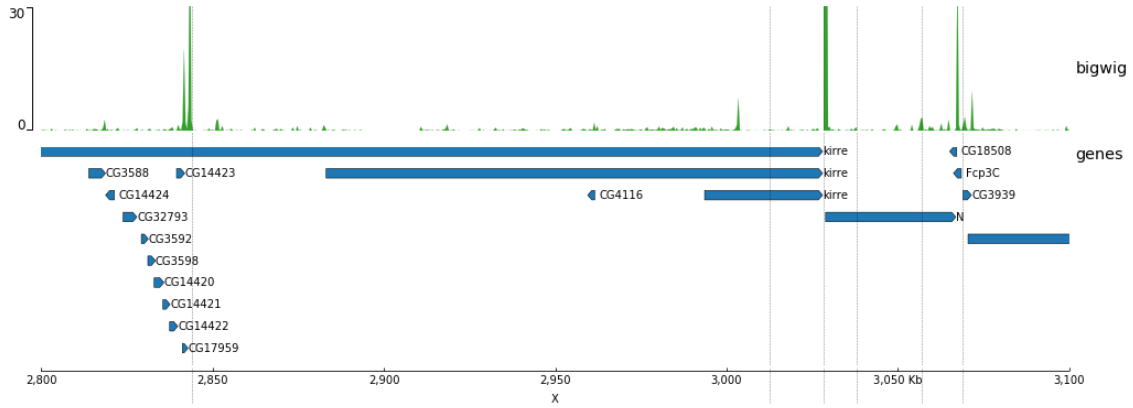
```
$ pyGenomeTracks --tracks bigwig_with_genes.ini --region X:2,800,000-3,100,000 -o␣
↪bigwig_with_genes.png
```



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.

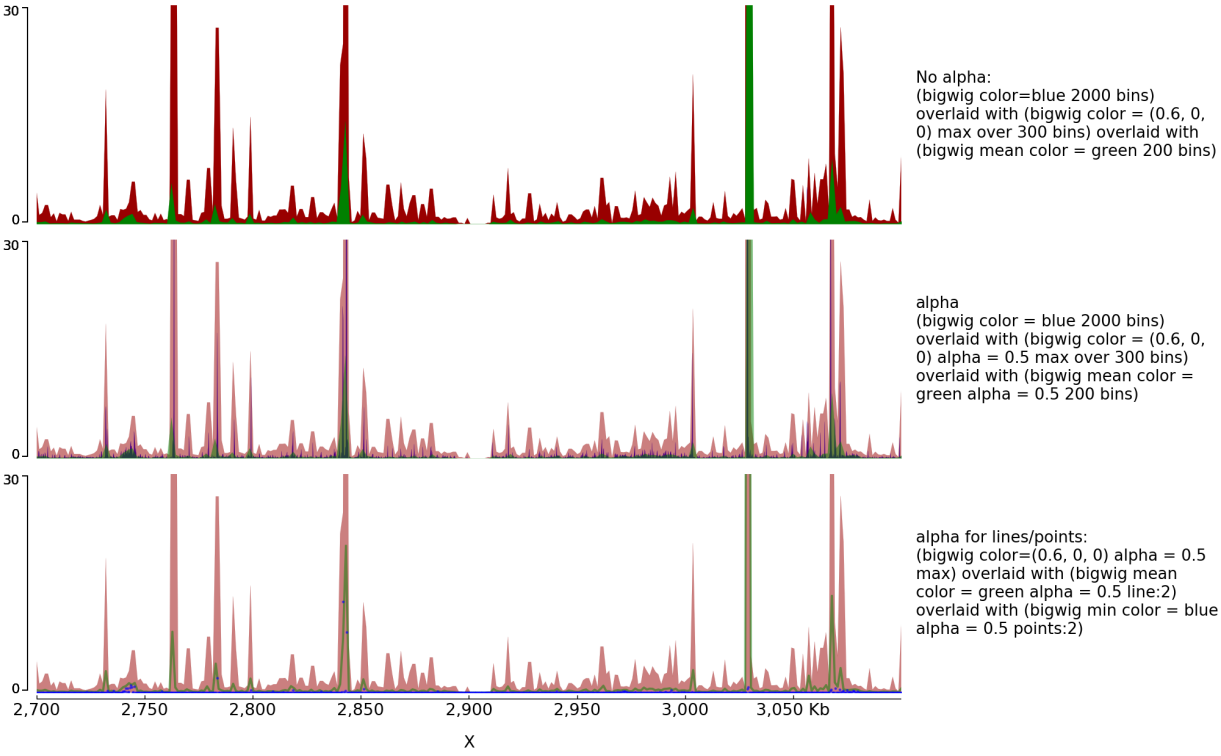
```
[test bigwig]
file = bigwig2_X_2.5e6_3.5e6.bw
color = blue
height = 7
title = No alpha:
        (bigwig color=blue 2000 bins) overlaid with (bigwig color = (0.6, 0, 0) max␣
↪over 300 bins) overlaid with (bigwig mean color = green 200 bins)
number_of_bins = 2000
min_value = 0
max_value = 30

[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
```



2.4.2 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)

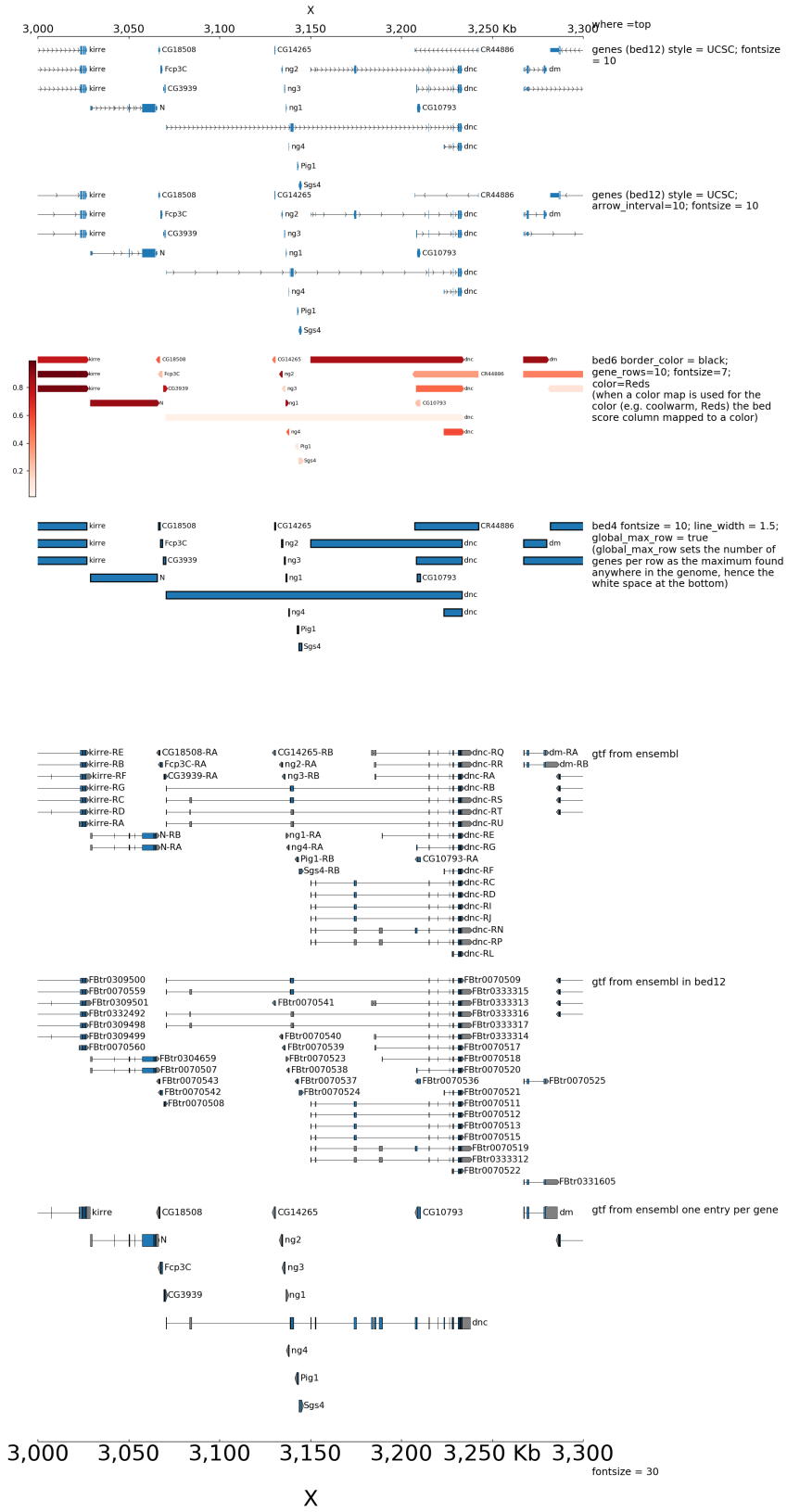
```
[test bed6]
file = dm3_genes.bed6.gz
height = 7
title = bed6 border_color = black; gene_rows=10; fontsize=7; color=Reds
        (when a color map is used for the color (e.g. coolwarm, Reds) the bed
        score column mapped to a color)
fontsize = 7
file_type = bed
color = Reds
border_color = black
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.gz
height = 10
title = gtf from ensembl
fontsize = 12
file_type = bed

[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 = bed
```

(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
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:

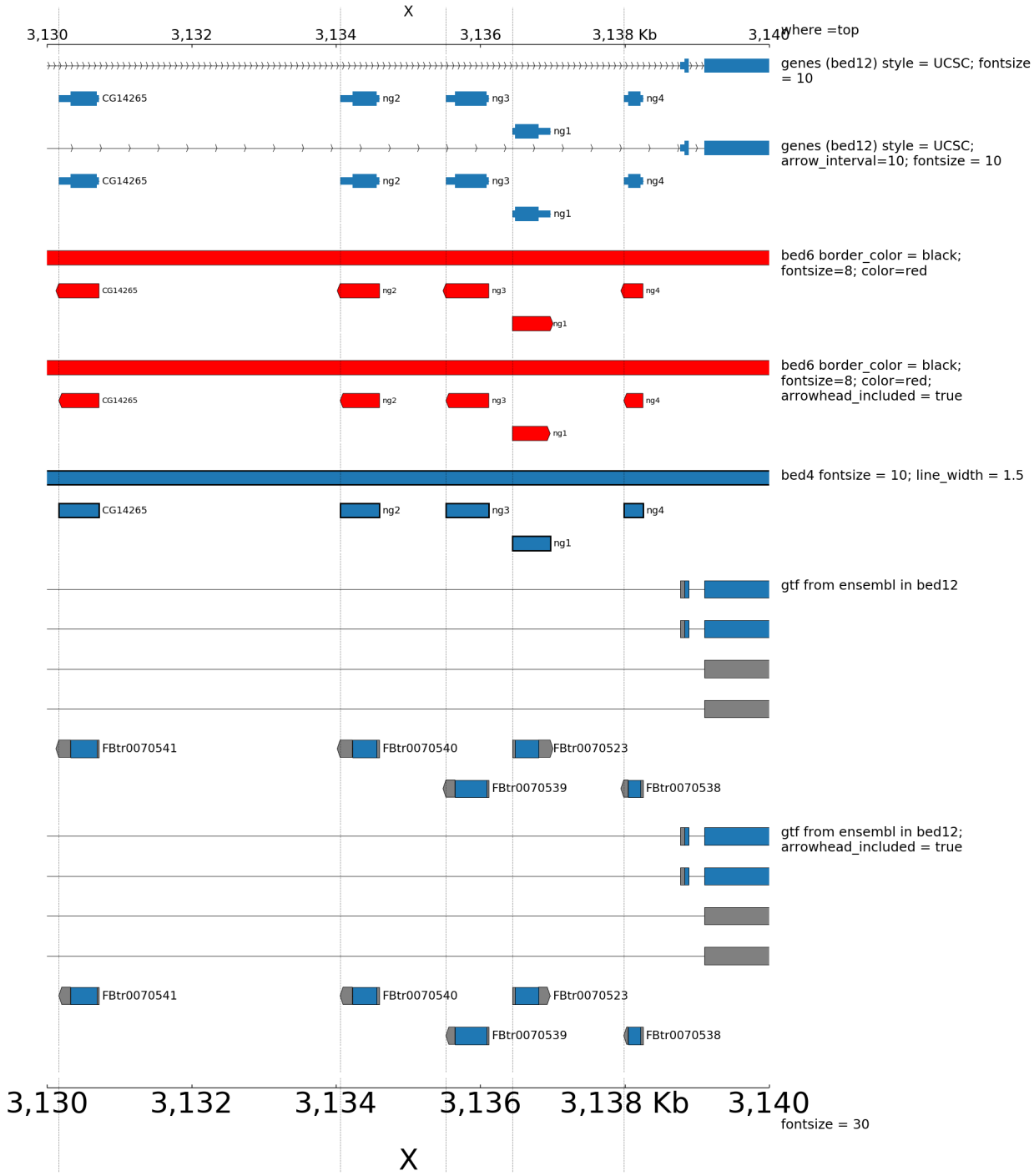
```
[x-axis]
where = top
title = where =top

[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:

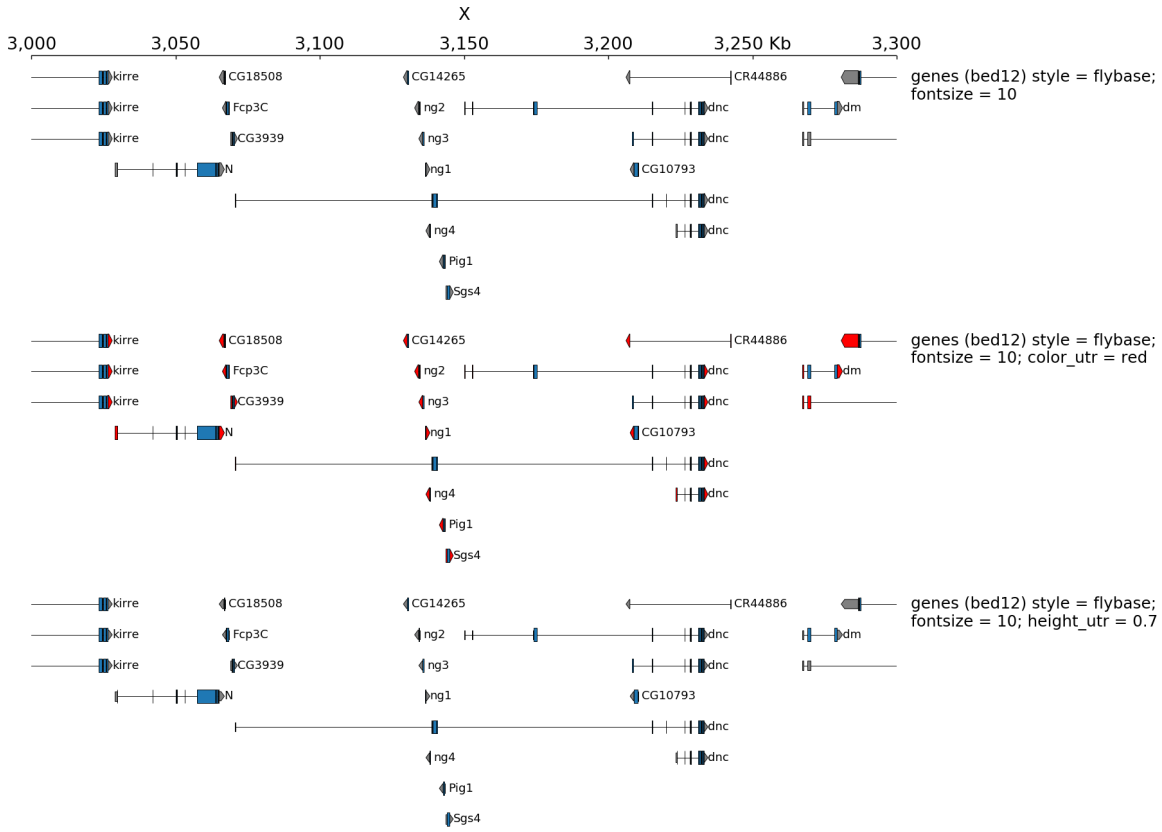
```
[x-axis]
where = top

[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
```



2.4.3 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
```

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

The output is available here.

2.4.4 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.

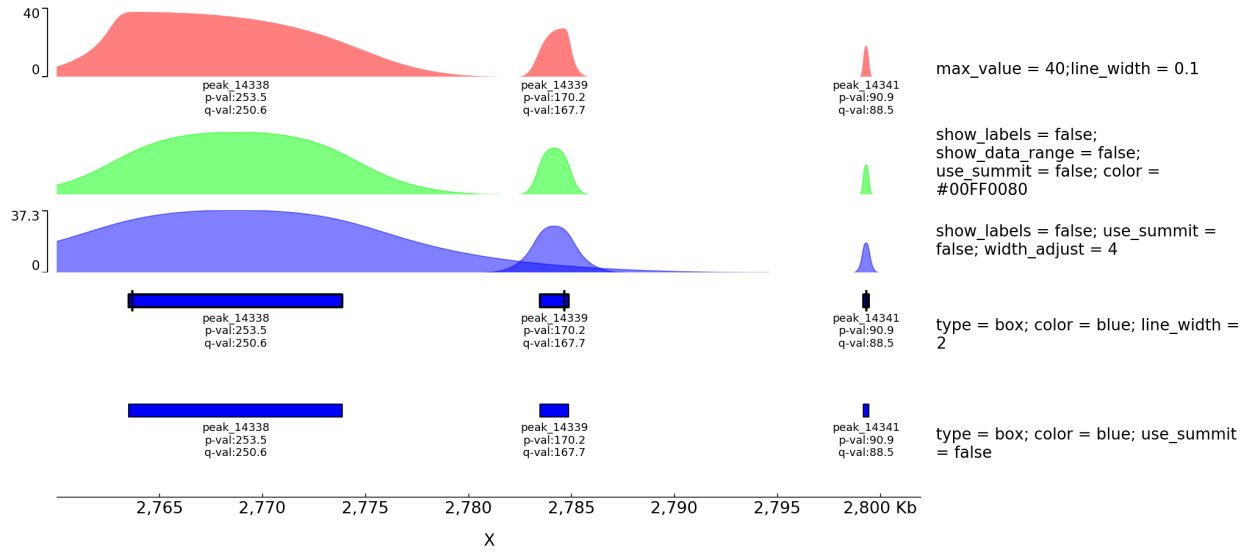
```
[narrow]
file = test2.narrowPeak
height = 4
max_value = 40
line_width = 0.1
title = max_value = 40;line_width = 0.1

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

[spacer]
```

(continues on next page)

(continued from previous page)

```
[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
```



2.4.5 Example with horizontal lines

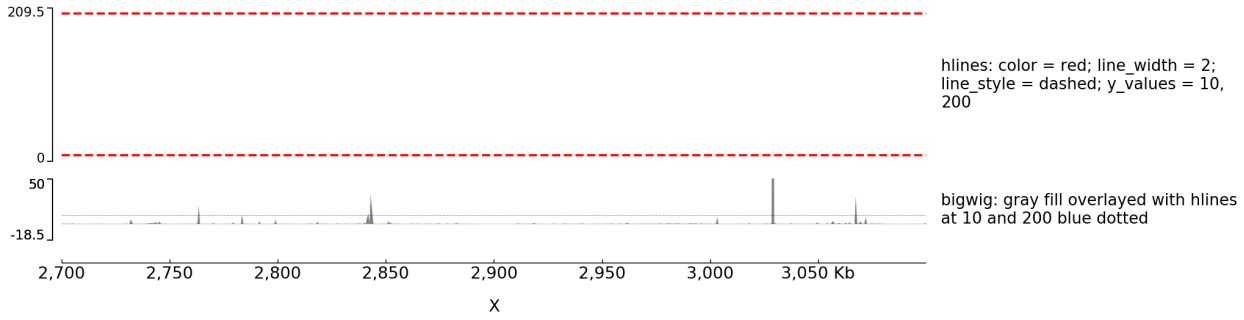
```
[test hlines]
color = red
line_width = 2
line_style = dashed
y_values = 10, 200
min_value = 0
show_data_range = true
height = 5
```

(continues on next page)

(continued from previous page)

```
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
```



2.4.6 Examples with Epilogos

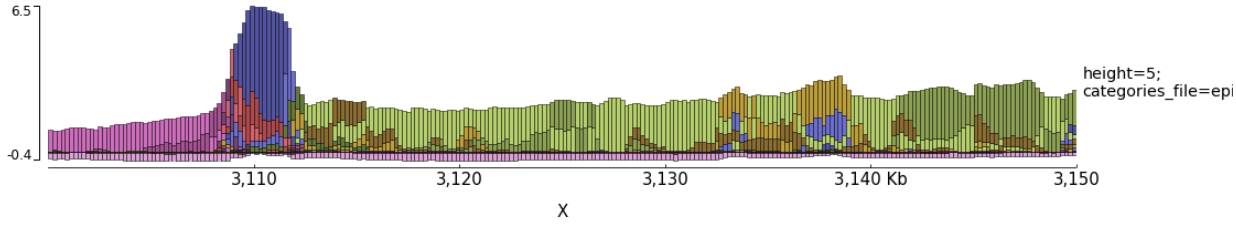
pyGenomeTracks can be used to visualize epigenetic states (for example from chromHMM) as epilogos. For more information see: https://epilogos.altiusinstitute.org/

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
```



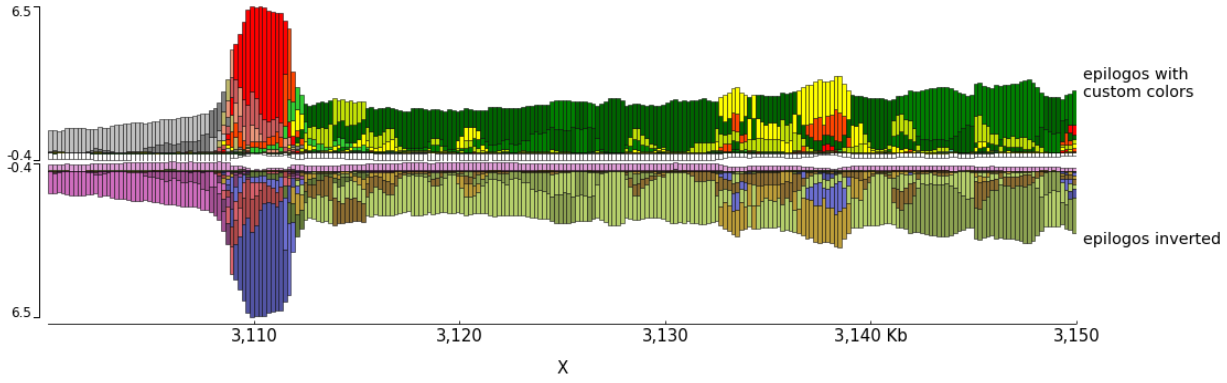
The color of the bars can be set by using a `json` file. The structure of the file is like this

```
{
"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.

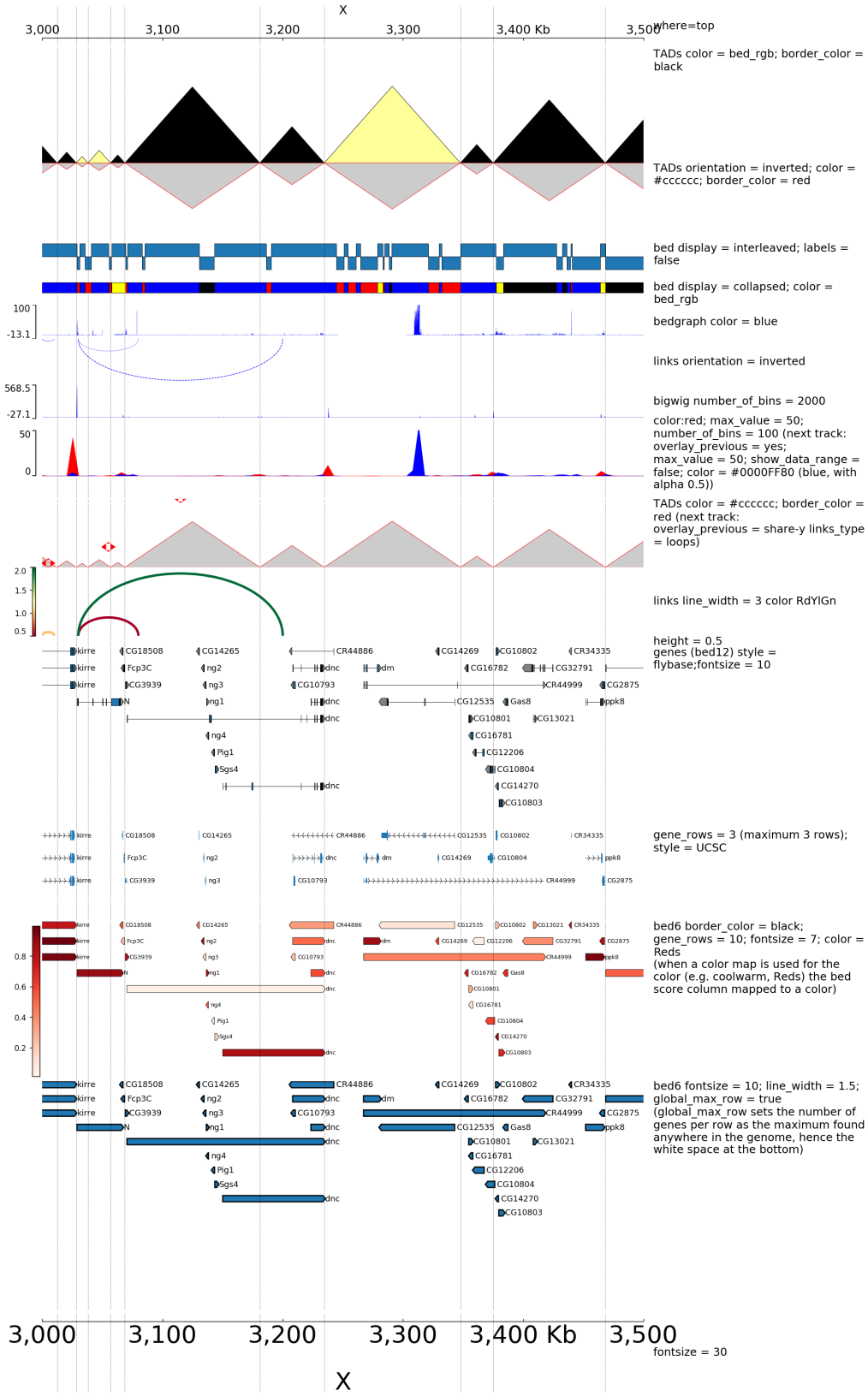
```
[epilogos]
file = epilog.qcat.bgz
height = 5
title = epilogos with custom colors
categories_file = epilog_cats.json

[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
```



2.4.7 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
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)
fontsize = 30

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
```

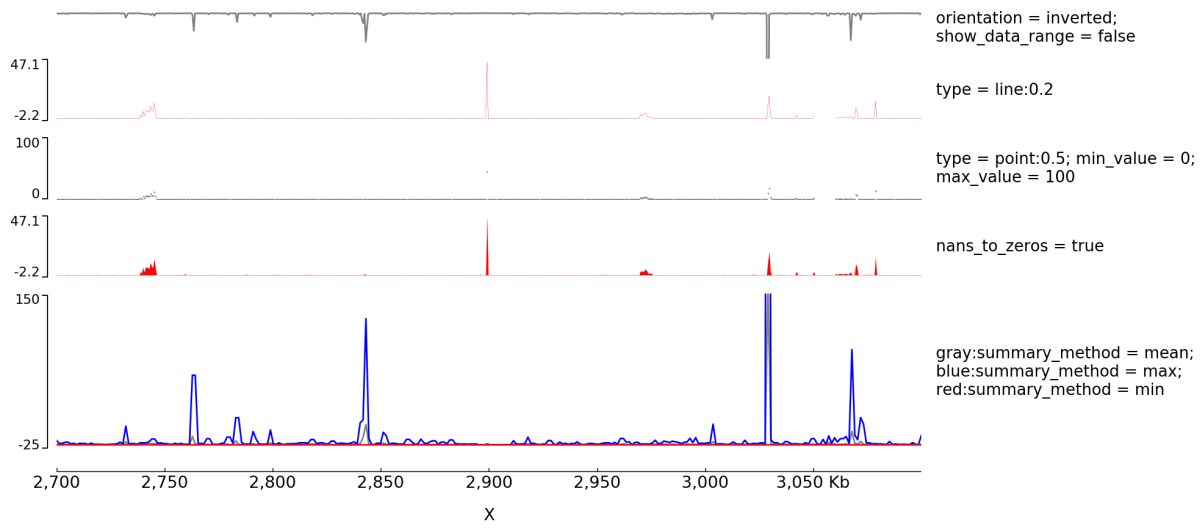
(continues on next page)

(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
```

2.4.8 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

[spacer]

[test bigwig points]
file = bigwig_chrx_2e6_5e6.bw
color = black
```

(continues on next page)

(continued from previous page)

```
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]
```

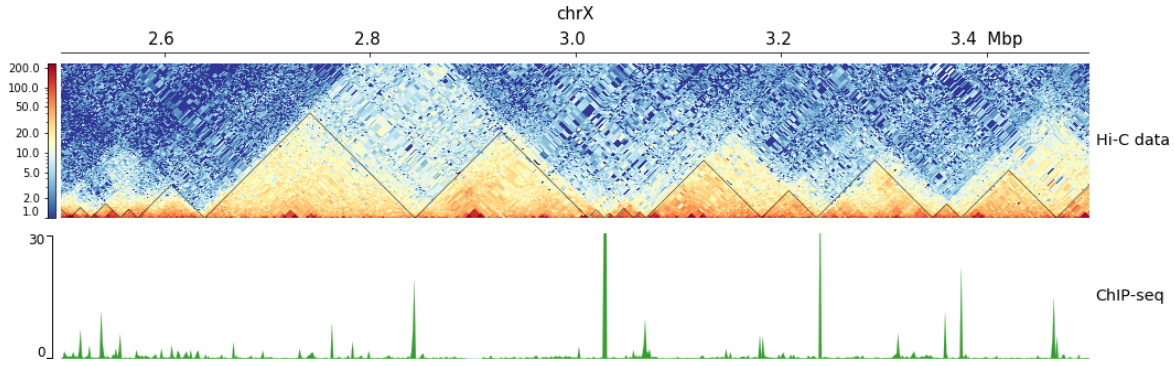
```
$ pyGenomeTracks  --tracks bigwig.ini --region X:2700000-3100000 --trackLabelFraction␣
↪0.2 --dpi 130 -o master_bigwig.png
```

2.4.9 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
```



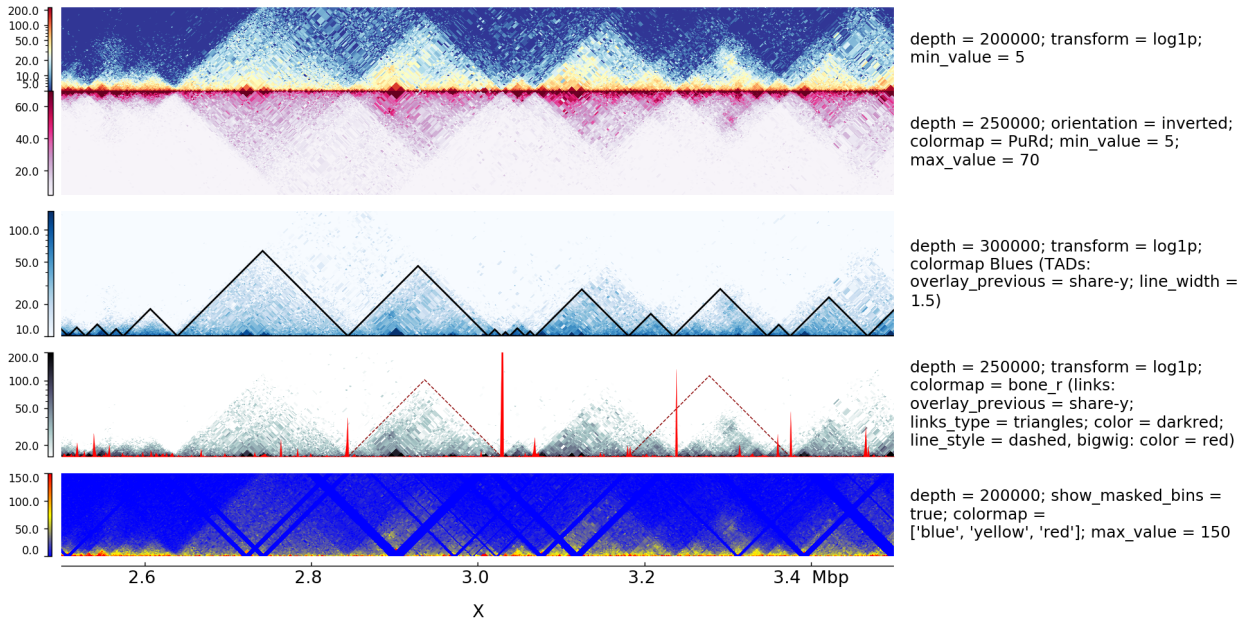
Here is an example where the height was set or not set and the heatmap was rasterized (default) or not rasterized (the dpi was set very low just to show the impact of the parameter).

```
[hic matrix]
file = Li_et_al_2015.h5
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 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	narrow_peak	bigwig	bedgraph
where	bottom							
fontsize	15			12	12			
categories_file		not set						
orientation		not set	not set	not set	not set	not set	not set	not set
links_type			arcs					
line_width			not set	0.5	0.5	1		
line_style			solid					
color			blue	#1f78b4	#1f78b4	#FF000080	#33a02c	#a6cee3
alpha			0.8				1	1
max_value			not set	not set	not set	not set	not set	not set
min_value			not set	not set	not set		not set	not set
ylim			not set					
compact_arcs_level			0					
border_color				black	black			
prefered_name				transcript_name	transcript_name			
merge_transcripts				false	false			
labels					true			
style					flybase			
display					stacked			
max_labels					60			
global_max_row					false			
gene_rows					not set			
arrow_interval					2			
arrowhead_included					false			
color_utr					grey			
height_utr					1			
arrow_length					not set			
all_labels_inside					false			

Table 1 – continued from previous page

parameter	x-axis	epilogos	links	domains	bed	narrow_peak	bigwig	bedgraph
labels_in_margin					false			
show_data_range						true	true	true
show_labels						true		
use_summit						true		
width_adjust						1.5		
type						peak	fill	fill
negative_color							not set	not set
nans_to_zeros							false	false
summary_method							mean	not set
number_of_bins							700	700
use_middle								false
rasterize								false
pos_score_in_bin								
plot_horizontal_lines								
colormap								
region								
depth								
show_masked_bins								
scale_factor								
transform								
x_center								
size								
height								

Some parameters can take only discrete values.

They are summarized here:

- **where**:
  - for *x-axis*: top, bottom
  - for *scalebar*: left, right, top, bottom
- **orientation**:
  - for *epilogos, links, domains, bed, 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
- **merge_transcripts**:
  - for *domains, bed*: true, false
- **style**:
  - for *bed*: flybase, UCSC, tssarrow

- **display**:
  - for *bed*: collapsed, triangles, interleaved, stacked
- **labels**:
  - for *bed*: true, false
- **global_max_row**:
  - for *bed*: true, false
- **arrowhead_included**:
  - for *bed*: true, false
- **all_labels_inside**:
  - for *bed*: true, false
- **labels_in_margin**:
  - for *bed*: 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
- **nans_to_zeros**:
  - for *bigwig, bedgraph*: true, false
- **use_middle**:
  - for *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
- **transform**:
  - for *hic_matrix*: no, log, log1p, -log

- **show_masked_bins**:
  - for *hic_matrix*: true, false

2.6 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)
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
```

(continues on next page)

(continued from previous page)

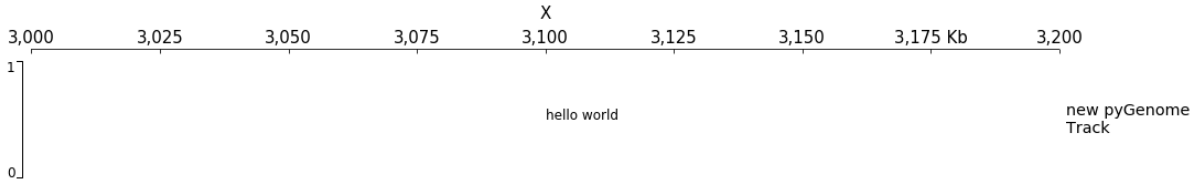
```
        location in the axis. The chrom, region_start and region_end
        variables are not used.
        Args:
            ax: matplotlib axis to plot
            chrom_region: chromosome name
            start_region: start coordinate of genomic position
            end_region: end coordinate
        """
        # 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'
```

(continues on next page)

(continued from previous page)

```
    OPTIONS_TXT = GenomeTrack.OPTIONS_TXT + """
        # 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 = {}
        """.format(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
        """
        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.
```

(continues on next page)

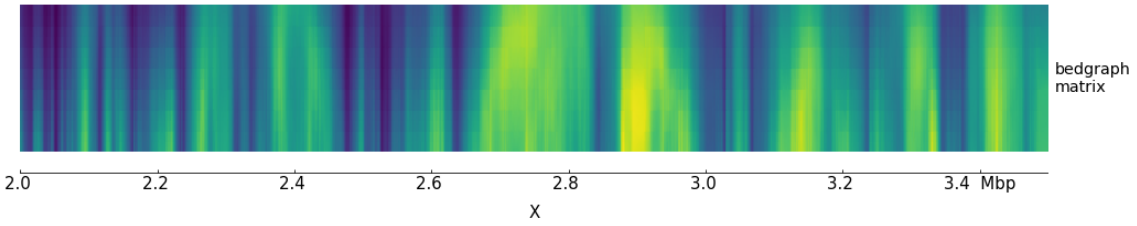
(continued from previous page)

```
            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
```

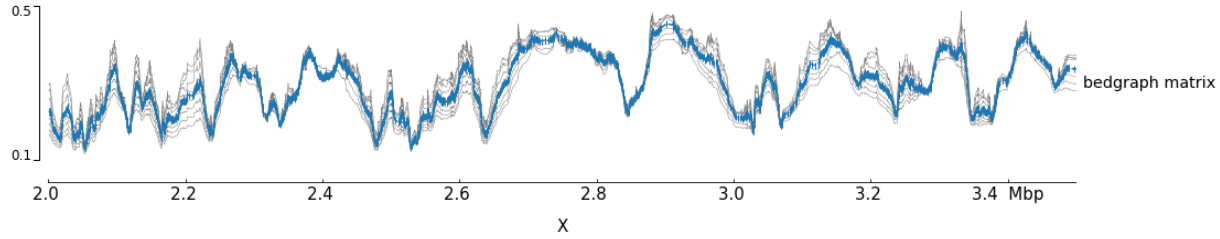
Let's create a track for this:

```
[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:


0.5
0.1
2.0
2.2
2.4
2.6
2.8
3.0
3.2
3.4 Mbp
X
bedgraph matrix