gelviz

Release 0.9

Matthias Bieg

Feb 25, 2021

CONTENTS:

CHAPTER

ONE

INTRODUCTION

gelviz is a python package for plotting a wide range of genomic elements in a genome browser-like fashion. It comes with basic plotting functions that can be used for plotting different kind of genomic elements.

gelviz is part of the Python Package Index (PyPI) and can be installed via

```
pip install gelviz
```

Alternatively you can clone the sources via the project's github page

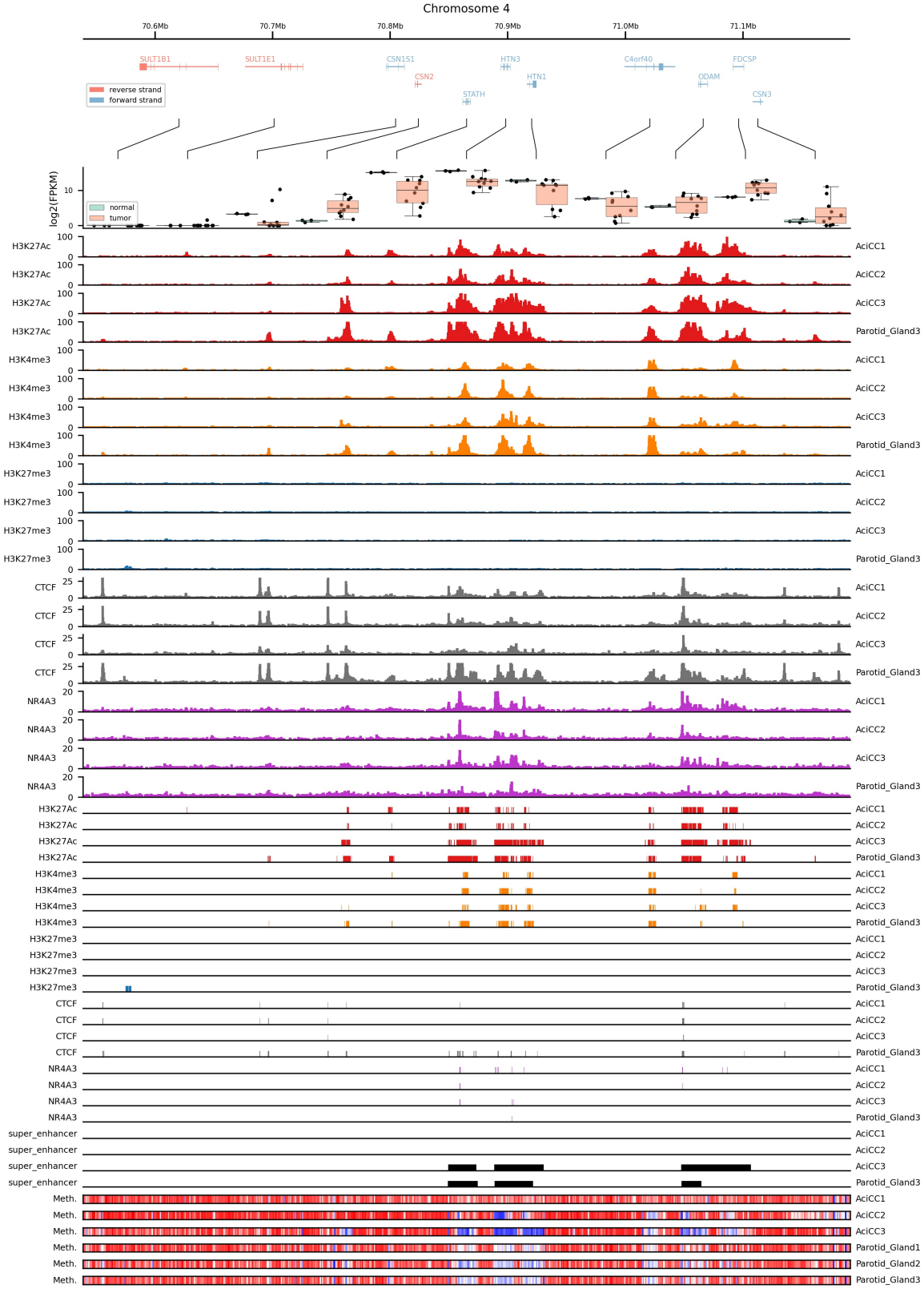
https://github.com/HiDiHlabs/gelviz

CHAPTER

TWO

EXAMPLE

The following example was created using published genomics and epigenomics data from https://zenodo.org/record/4557352 The data originates from the publication https://www.nature.com/articles/s41467-018-08069-x


Chromosome 4
70.6Mb
70.7Mb
70.8Mb
70.9Mb
71.0Mb
71.1Mb
SULT1B1
SULT1E1
CSN1S1
CSN2
HTN3
HTN1
STATH
C4orf40
ODAM
FDCSP
CSN3
reverse strand
forward strand
log2(FPKM)
normal
tumor
H3K27Ac
H3K4me3
H3K27me3
CTCF
NR4A3
super_enhancer
Meth.
AciCC1
AciCC2
AciCC3
Parotid_Gland3
Parotid_Gland1
Parotid_Gland2

The above figure was created using the following code:

```
import pyBigWig
import os
import hmap
import pybedtools
import glob
import sys
import matplotlib.pyplot as plt
import pandas as pnd
import gelviz
import warnings
warnings.filterwarnings("ignore")

data_folder = "" # path to folder where data from https://zenodo.org/record/4557352␣
↪(DOI: 10.5281/zenodo.4557352) is stored
fpkm_matrix_filename = data_folder+os.sep+"/fpkm_matrix_human.csv"
samples_dict = {"normal": ["Parotid Gland3", "Parotid Gland4", "Parotid_Gland2"],
                "tumor": ["AciCC1", "AciCC3", "AciCC8", "AciCC4", "AciCC10", "AciCC9",
↪ "AciCC6", "AciCC5", "AciCC2", "AciCC7"]}

# File´s used for gene plotting
gelviz_repo_path = "" # Path to clone of https://github.com/HiDiHlabs/gelviz
genes_bed_filename = gelviz_repo_path+os.sep+"data/gencode.v19.annotation.genes_genes.
↪bed"
exons_bed_filename = gelviz_repo_path+os.sep+"data/gencode.v19.annotation.genes_exons.
↪bed"
introns_bed_filename = gelviz_repo_path+os.sep+"data/gencode.v19.annotation.genes_
↪introns.bed"
gene_map_filename = gelviz_repo_path+os.sep+"data/gencode_v19_ENSGID_GENENAME.tsv"

samples = ["AciCC1", "AciCC2", "AciCC3", "Parotid_Gland1", "Parotid_Gland2", "Parotid_
↪Gland3", "Parotid_Gland4"]
chip_types = ["H3K27Ac", "H3K4me3", "H3K27me3", "CTCF", "NR4A3", "super_enhancer"]

chip_color_dict = {"H3K27Ac": "#e31a1c",
                   "H3K4me3": "#ff7f00",
                   "H3K27me3": "#1f76b4",
                   "CTCF": "#737373",
                   "NR4A3": "#bc36cc",
                   "super_enhancer": "k"}

ylim_dict = {"H3K27Ac": [0, 100],
                   "H3K4me3": [0, 100],
                   "H3K27me3": [0, 100],
                   "CTCF": [0, 30],
                   "NR4A3": [0, 20]}

# Create a gene_id_map
gene_map_file = open(gene_map_filename, "r")
gene_id_map = {}
for line in gene_map_file:
    split_line = line.rstrip().split("\t")

    ensembl_gene_id = split_line[0].split(".")[0]
    hugo_gene_symbol = split_line[1]

    gene_id_map[ensembl_gene_id] = hugo_gene_symbol
```

(continues on next page)

(continued from previous page)

```
gene_map_file.close()

fpkm_matrix_df = pnd.read_csv(fpkm_matrix_filename, sep="\t", index_col = 14)


# # Create a browser like plot containing
#
# 1. Genomic coordinates
# 2. Genes
# 3. ChIP seq signal tracks
# 4. ChIP seq peak tracks
# 5. Methylation profiles

n_signal_tracks = len(glob.glob(data_folder+os.sep+"*_chip_signal_*"))
n_peak_tracks = len(glob.glob(data_folder+os.sep+"*_peaks_*"))
n_methylation_tracks = len(glob.glob(data_folder+os.sep+"*_methylationCalls_*"))

region = ["4", 70538611, 71191210]
region_bed = pybedtools.BedTool("\t".join([str(e) for e in region]), from_string =␣
↪True)

genes_bed = pybedtools.BedTool(genes_bed_filename)
introns_bed = pybedtools.BedTool(introns_bed_filename)
exons_bed = pybedtools.BedTool(exons_bed_filename)

get_ipython().run_line_magic('autoreload', '2')

plot_filename = "pics/salivary_gland_locus_acicc.jpeg"

# Create a figure grid
fig, gs = hmap.layout.layout.layoutGrid(4+n_signal_tracks+n_peak_tracks+n_methylation_
↪tracks,
                                        1,
                                        [189.],
                                        [5., 15., 10., 15.,]+n_signal_tracks*[5.]+n_
↪peak_tracks*[2.]+n_methylation_tracks*[2.],
                                        2.,
                                        1.,
                                        10.,
                                        20.,
                                        25.,
                                        25.)

# Plot genomic coordinates
ax = plt.subplot(gs[0,0])
plt.title("Chromosome "+str(region[0]), fontsize=8)
gelviz.basic.plotCoordinates(region[0],
                             region[1],
                             region[2],
                             ax = ax,
                             upper=True)

# Plot genes
ax = plt.subplot(gs[1, 0])
gelviz.basic.plotGenes(genes_bed.intersect(region_bed, wa=True),
                       exons_bed.intersect(region_bed, wa=True),
                       introns_bed.intersect(region_bed, wa=True),
```

(continues on next page)

(continued from previous page)

```
                        region_bed,
                        gene_map = gene_id_map,
                        ax = ax,
                        plot_legend = True,
                        legend_loc = "lower left")
ax.axis("off")

# Plot Gene Distance Equalizer
ax = plt.subplot(gs[2, 0])
gene_mid_points = gelviz.basic.distanceEqualizer(genes_bed.intersect(region_bed,␣
↪wa=True),
                                region[1],
                                region[2],
                                ax = ax)

# Plot Gene Expression
ax = plt.subplot(gs[3, 0])
gelviz.basic.plotGeneExpressionEqualDist(genes_bed.intersect(region_bed, wa=True),
                                         gene_mid_points,
                                         region,
                                         fpkm_matrix_df,
                                         groups=[samples_dict["normal"], samples_dict[
↪"tumor"]],
                                         ids=["normal", "tumor"],
                                         gene_names_map = gene_id_map,
                                         ax = ax,
                                         plot_legend = True,
                                         plot_gene_names = False,
                                         plot_points = True)
ax.spines["top"].set_visible(False)
ax.spines["right"].set_visible(False)
plt.xticks([], [])
plt.ylabel("log2(FPKM)", fontsize=7)

# Plot ChIP-Signals
chip_plot_idx = 0
for chip_type in chip_types:
    for sample in samples:
        signal_filename = glob.glob(data_folder+os.sep+sample+"_"+chip_type+"_chip_
↪signal_*")
        if(not(len(signal_filename) == 1)):
            continue

        signal_filename = signal_filename[0]
        bw = pyBigWig.open(signal_filename)
        signal_list = [ [region[0], i[0], i[1], i[2]] for i in bw.intervals(region[0],
↪ region[1], region[2]) ]

        color =chip_color_dict[chip_type]

        ax = plt.subplot(gs[4+chip_plot_idx])
        plt.yticks(fontsize=6)
        ax2 = ax.twinx()
        plt.yticks([], [])
        gelviz.basic.plotChIPSignals(signal_list,
                                     region[0],
                                     region[1],
```

(continues on next page)

(continued from previous page)

```
                                    region[2],
                                    ax = ax,
                                    offset = 100,
                                    merge = 10,
                                    color = color)

        ax.spines["top"].set_visible(False)
        ax.spines["right"].set_visible(False)
        ax.set_xticks([], [])
        ax.set_ylim(ylim_dict[chip_type])
        ax2.spines["top"].set_visible(False)
        ax2.spines["right"].set_visible(False)
        ax2.set_xticks([], [])
        ax2.set_ylim(ylim_dict[chip_type])
        ax.set_ylabel(chip_type,
                   fontsize=6,
                   rotation = 0,
                   horizontalalignment = "right",
                   verticalalignment = "center")

        ax2.yaxis.set_label_position("right")
        ax2.set_ylabel(sample,
                  fontsize=6,
                   rotation = 0,
                   horizontalalignment = "left",
                   verticalalignment = "center")

        plt.yticks(fontsize=6)

        chip_plot_idx += 1

# Plot ChIP-peaks
chip_peak_idx = 0
for chip_type in chip_types:
    for sample in samples:
        peak_filename = glob.glob(data_folder+os.sep+sample+"_"+chip_type+"_peaks_*")
        if(not(len(peak_filename) == 1)):
            continue
        peak_filename = peak_filename[0]

        peak_bed = pybedtools.BedTool(peak_filename)

        region_chr_bed = pybedtools.BedTool("\t".join(["chr"+region[0],
                                                       str(region[1]),
                                                       str(region[2])]),
                                            from_string=True)

        ax = plt.subplot(gs[4+chip_plot_idx+chip_peak_idx])
        plt.yticks([], [])
        ax2 = ax.twinx()
        plt.yticks([], [])
        gelviz.basic.plotRegions(peak_bed.intersect(region_chr_bed, wa=True),
                                 region[1],
                                 region[2],
                                 color = chip_color_dict[chip_type],
                                 edgecolor=False,
                                 ax = ax)
```

(continues on next page)

(continued from previous page)

```
        ax.set_ylabel(chip_type,
                      fontsize=6,
                      rotation = 0,
                      horizontalalignment = "right",
                      verticalalignment = "center")

        ax2.yaxis.set_label_position("right")
        ax2.set_ylabel(sample,
                       fontsize=6,
                       rotation = 0,
                       horizontalalignment = "left",
                       verticalalignment = "center")

        #ax. axis("off")
        ax.spines["top"].set_visible(False)
        ax.spines["left"].set_visible(False)
        ax.spines["right"].set_visible(False)
        ax2.spines["top"].set_visible(False)
        ax2.spines["left"].set_visible(False)
        ax2.spines["right"].set_visible(False)
        chip_peak_idx += 1

# Plot Methylation Profiles
methylation_idx = 0
for sample in samples:
    meth_filename = glob.glob(data_folder+os.sep+sample+"_"+"methylationCalls"+"*")
    if(not(len(meth_filename) == 1)):
        continue
    meth_filename = meth_filename[0]
    meth_bed = pybedtools.BedTool(meth_filename)

    meth_list = [ [str(e[0]),
                   int(e[1]),
                   int(e[2]),
                   int(e[4]),
                   int(e[5])] for e in meth_bed.intersect(region_bed, wa=True) ]

    ax = plt.subplot(gs[4+chip_plot_idx+chip_peak_idx+methylation_idx])
    plt.yticks([], [])
    ax2 = ax.twinx()
    plt.yticks([], [])
    gelviz.basic.plotMethylationProfileHeat(meth_list,
                                            region[0],
                                            region[1],
                                            region[2],
                                            ax = ax)
    ax.set_ylabel("Meth.",
                  fontsize=6,
                  rotation = 0,
                  horizontalalignment = "right",
                  verticalalignment = "center")

    ax2.yaxis.set_label_position("right")
    ax2.set_ylabel(sample,
                   fontsize=6,
                   rotation = 0,
                   horizontalalignment = "left",
```

(continues on next page)

(continued from previous page)

```
                    verticalalignment = "center")

    methylation_idx += 1

plt.savefig(plot_filename)
```

CHAPTER

THREE

DOCUMENTATION

3.1 The `gelviz.basic` module

gelviz.basic.**createGeneNameMap**(*gene_name_mapping_filename*)
Function that creates a mapping between gene ids

Parameters **gene_name_mapping_file** (*str*) – Path to a tab separated file, for which the first column is a ensemble gene id, and the second column is the HUGO gene name

Returns Dictionary containing the gene id mapping.

Return type dictionary

gelviz.basic.**determineYPosGene**(*genes_bed*, *region_size*, *distance_ratio*)
Function that determines the max y position for gene plotting via function plotGenes.

Parameters

- **genes_bed** (pybedtools.BedTool) – pybedtools.BedTool object containing genes to be plotted.
- **region_size** (*int*) – Size of region to be plotted in base pairs.
- **distance_ratio** (*float*) – Minimal distance between two genes, as ratio of ax width, such that two genes are plotted side by side. If this ratio is underwent, the genes will be stacked.

Returns

Tuple of

1. max_y_pos: Defines the number of stacked genes.
2. y_pos_dict: Dictionary with keys = gene ids and values = y position of gene.

Return type tuple

gelviz.basic.**distanceEqualizer**(*genomic_segments*, *start*, *end*, *direction='top_down'*, *color='k'*, *ax=None*)
Function that plots arcs from unequal distances of genomic segments to equal distances.

Parameters

- **genomic_segments** (*list*) – List of segments for which distances shall be equalized (each segment is of the form [<chrom>, <start>, <end>])
- **start** (*int*) – Start position of the genomic region.
- **end** (*int*) – End position of the genomic region.
- **color** (*str, optional*) – Color of lines equalizing distances, defaults to "k".

- **direction** (*str, optional.*) – Direction of distance equalization (top_down | bottom_up), defaults to "top_down".
- **ax** (matplotlib.axes._subplots.AxesSubplot, optional) – Axis on which to plot, defaults to None.

Returns List of equalized region midpoints.

Return type list

gelviz.basic.**plotCNVs**(*cnvs_bed*, *chromosome*, *start*, *end*, *ploidy=2*, *cnv_threshold=0.7*, *color_gain='g'*, *color_loss='r'*, *color_neutral='k'*, *ax=None*)

Function for plotting CNV segments

Parameters

- **cnvs_bed** (pybedtools.BedTool) – pybedtools.BedTool object containing CNVs with following entries:
  1. Chromosome,
  2. Start Position,
  3. End Position,
  4. Deviation from ploidy,
  5. True Copy Number)
- **chromosome** (*str*) – Chromosome for which to plot CNVs.
- **start** (*int*) – Start position on chromosome.
- **end** (*int*) – End position on chromosome.
- **ploidy** (*int, optional*) – Assumed ploidy of tumor, defaults to 2.
- **cnv_threshold** (*float, optional*) – Minimal deviation from ploidy to be considered as a CNV, defaults to 0.7.
- **color_gain** (*str, optional*) – Plot color of copy number gains, defaults to "g".
- **color_loss** (*str, optional*) – Plot color of copy number losses, defaults to "r".
- **color_neutral** (*str, optional*) – Plot color of copy number neutral regions, defaults to "k".
- **ax** (matplotlib.axes._subplots.AxesSubplot, optional) – Axis used for plotting.

Returns Nothing to be returned

Return type None

gelviz.basic.**plotCNVsHeat**(*cnvs_bed*, *chromosome*, *start*, *end*, *ploidy=2*, *cnv_threshold=0.7*, *cmap='bwr'*, *max_dev=None*, *ax=None*)

Function for plotting CNV segments as heatmap

Parameters

- **cnvs_bed** (pybedtools.BedTool) – pybedtools.BedTool object containing CNVs with following entries:
  1. Chromosome,
  2. Start Position,
  3. End Position,

4. Deviation from ploidy,
5. True Copy Number)

- **chromosome** (*str*) – Chromosome for which to plot CNVs.
- **start** (*int*) – Start position on chromosome.
- **end** (*int*) – End position on chromosome.
- **ploidy** (*int, optional*) – Assumed ploidy of tumor, defaults to 2.
- **cnv_threshold** (*float, optional*) – Minimal deviation from ploidy to be considered as a CNV, defaults to 0.7.
- **cmap** (*str, optional*) – Colormap used for plotting CNVs, defaults to "bwr".
- **max_dev** (*float, optional*) – Maximal deviation from ploidy to plot, defaults to None.
- **ax** (matplotlib.axes._subplots.AxesSubplot, optional) – Axis used for plotting, defaults to None.

Returns Nothing to be returned.

Return type None

gelviz.basic.**plotChIPSignals**(*chip_signals*, *r_chrom*, *r_start*, *r_end*, *ax=None*, *color='b'*, *offset=None*, *merge=None*)

Function that plots bedGraph like iterators.

Parameters

- **chip_signals** (*iterator*) – Iterator for which each element is a list-ike object containing:
  1. Chromosome
  2. Start postion
  3. End position
  4. Value to be plotted as bar
- **r_chrom** (*str*) – Chromosome of region to be plotted.
- **r_start** (*int*) – Start position of region to be plotted.
- **r_end** (*int*) – End position of region to be plotted.
- **ax** (matplotlib.axes._subplots.AxesSubplot, optional) – Axis of plot
- **color** (*str, optional*) – color of bars, defaults to "b".
- **offset** (*int, optional*) – Length of intervals, defaults to None.
- **merge** (*int, optional*) – Number of elements to be merged. If this value is not equal to 0, than merge elements will be averaged an plotted, defaults to 0.

Returns Nothing to be returned.

Return type None

gelviz.basic.**plotCoordinates**(*chrom*, *start*, *end*, *color='k'*, *ax=None*, *upper=True*, *loc_coordinates='up'*, *revert_coordinates=False*, *rotation=0*)

Function that plots genomic coordinates in a linea fashion.

Parameters

- **chrom** (*str*) – Chromosome of the region to be plotted.
- **start** (*int*) – Start position of the region to be plotted.
- **end** (*int*) – End position of the region to be plotted.
- **color** (*str, optional*) – Color of the genomic scales elements, defaults to "k".
- **ax** (matplotlib.axes._subplots.AxesSubplot, optional) – Axis of plot, defaults to None.
- **upper** (*bool, optional*) – If True, make less ticks, else if False make more ticks.
- **loc_coordinates** (*str, optional*) – Either of "up" | "down". If "up", plot ticks to upper direction, else if "down", plot ticks to lower direction, defaults to "up".
- **revert_coordinates** (*bool, optional*) – If True, coordinates are reverted to decreasing order. Else, coordinates stay in increasing order, defaults to False.
- **rotation** (*int, optional*) – Rotational angle of coordinate strings, defaults to 0.

Returns Nothing to be returned.

Return type None

gelviz.basic.**plotGeneExpression**(*genes_bed*, *region_bed*, *expression_df_g1*, *expression_df_g2*, *gene_names_map*, *blacklist=None*, *ax=None*, *plot_legend=False*, *color_g1='#fb8072'*, *color_g2='#80b1d3'*, *g1_id='tumor'*, *g2_id='normal'*, *plot_gene_names=True*)

Function for plotting paired gene expression (e.g. tumor and normal) on a gene region scale retaining the position of genes.

Parameters

- **genes_bed** (pybedtools.BedTool) – pybedtools.BedTool object containing TXstart, and TXend of genes.
- **region_bed** (pybedtools.BedTool) – pybedtools.BedTool object containing the region to be plotted
- **expression_df_g1** (pandas.DataFrame) – pandas.Dataframe containing the expression values of g1 samples (columns: sample ids; index: gene ids)
- **expression_df_g2** (pandas.DataFrame) – pandas.Dataframe containing the expression values of g2 samples (columns: sample ids; index: gene ids)
- **gene_names_map** (*dict.*) – Dictionary with keys: ENSEMBL GENE IDs, and values: HUGO GENE SYMBOLs.
- **blacklist** (*set, optional*) – Set containing gene ids not to be plotted, default to None.
- **ax** (matplotlib.axes._subplots.AxesSubplot, optional) – Axis used for plotting, defaults to None.
- **plot_legend** (*bool*) – If True legend is plotted, False otherwise, defaults to False.
- **color_g1** (*str, optional*) – Color used for plotting g1 samples expression, defaults to "#fb8072".
- **color_g2** (*str, optional*) – Color used for plotting g2 samples expression, defaults to "#80b1d3".
- **g1_id** (*str, optional*) – ID of g1 used for legend plotting, defaults to "tumor".
- **g2_id** (*str, optional*) – ID of g2 used for legend plotting, defaults to "normal".

- **plot_gene_names** (*bool.*) – If True, the HUGO GENE SYMBOLs will be shown, else the GENE SYMBOLs are hidden.

Returns Axis on which plot was placed.

Return type matplotlib.axes._subplots.AxesSubplot

gelviz.basic.**plotGeneExpressionEqualDist**(*genes_bed*, *gene_mid_points*, *region*, *expression_df*, *groups*, *gene_names_map=None*, *blacklist=None*, *ax=None*, *plot_legend=False*, *colors=None*, *ids=None*, *plot_gene_names=True*, *position_gene_names='bottom'*, *log_transformed=True*, *plot_points=False*, *alpha=0.5*)

Function for plotting grouped gene expression (e.g. tumor and normal) on a gene region scale equalizing the position of genes.

Parameters

- **genes_bed** (pybedtools.BedTool) – pybedtools.BedTool object containing gene regions.
- **gene_mid_points** (*list*) – list of integer values containing center positions of genes.
- **region** (*list*) – List containing the region to be plotted ([<chrom>, <start>, <end>]).
- **groups** (*list*) – List of lists containing the IDs of the different groups.
- **gene_names_map** (*dict.*) – Dictionary with keys: ENSEMBL GENE IDs, and values: HUGO GENE SYMBOLs.
- **expression_df** (class:*pandas.DataFrame*) – class:*pandas.DataFrame* object containing the expression values of all samples (columns: sample ids; index: gene ids).
- **blacklist** (*set, optional*) – Set containing gene ids not to be plotted, defaults to None,
- **ax** (matplotlib.axes._subplots.AxesSubplot, optional) – (default: None) Axis used for plotting, defaults to None.
- **plot_legend** (*bool, optional*) – If True plot legend, False otherwise, defaults to False.
- **colors** (*str, optional*) – List of colors used for plotting samples expression. The number of colors must be the same as the number of groups, defaults to None.
- **ids** (*list, optional.*) – IDs used for legend plotting, defaults to None. Number of ids must be the same as the number of groups.
- **plot_gene_names** (*bool, optional*) – True if gene names shall be plotted, False otherwise, defaults to True.
- **position_gene_names** (*str, optional*) – Either of "top", or "bottom", defaults to "bottom".
- **log_transformed** (*bool, optional*) – If True use log transformed values for plotting, non-transformed values otherwise.
- **plot_points** (*bool, optional*) – If True, a point per expression value is plotted in addition to the boxplot, no points are plotted otherwise, defaults to False.
- **alpha** (*float, optional*) – Alpha value for the background color of the boxplots boxes, defaults to 0.5.

Returns Plots axis.

Return type matplotlib.axes._subplots.AxesSubplot

gelviz.basic.**plotGenes**(*genes_bed*, *exons_bed*, *introns_bed*, *region_bed*, *blacklist=None*, *gene_map=None*, *plot_gene_ids=True*, *y_max=None*, *distance_ratio=0.1*, *ax=None*, *plot_legend=False*, *legend_loc='lower right'*, *color_plus='#80b1d3'*, *color_minus='#fb8072'*)

Function for plotting gene structures, i.e. introns exons of genes.

Parameters

- **genes_bed** (pybedtools.BedTool) – pybedtools.BedTool object containing TX start, and TX end of genes.
- **exons_bed** (pybedtools.BedTool) – pybedtools.BedTool object containing exons of genes.
- **introns_bed** (pybedtools.BedTool) – pybedtools.BedTool object containing introns
- **region_bed** (pybedtools.BedTool) – pybedtools.BedTool object containing the one region, for which the gene plot is created.
- **blacklist** (*list, optional*) – List of gene names, for genes that should not be shown on the plot, default is None
- **plot_gene_ids** (*bool, optional*) – If True, all gene ids will be included in the plot, False otherwise, default is True
- **y_max** (*bool, optional*) – Max y value in the gene plot. If not set, then y_max is the max number of stacked genes, default is None.
- **distance_ratio** (*float, optional*) – Minimal distance between two genes, as ratio of ax width, such that two genes are plotted side by side. If this ratio is underwent, the genes will be stacked, default is 0.1.
- **ax** (matplotlib.axes._subplots.AxesSubplot, optional) – Axes instance on which the genes are plotted, default is None.
- **plot_legend** (*bool, optional*) – If True, a legend describing plus or minus stranded genes is plotted, False otherwise. Default is False.
- **legend_loc** (*str, optional*) – Location of the legend. Either of "lower left", "lower right", "upper left", "upper right", default is "lower right".
- **color_plus** (*str, optional.*) – Color code for plus stranded genes, default is "#80b1d3".
- **color_minus** (*str, optional.*) – Color code for minus stranded genes, default is "#fb8072".

Returns

Tuple of max_y_pos+1.5, patch_list, patch_description_list, where

1. max_y_pos+1.5 is the max_y_position + 1.5. max_y_pos defines the number of stacked genes.
2. patch_list is the list of patches drawn on the ax.
3. patch_description_list is the list of descriptions for the patches drawn on the ax.

Return type list

gelviz.basic.**plotGenomicSegments**(*segments_list*, *chrom*, *start*, *end*, *ax=None*)

Function for plotting genomix segments in different colors

Parameters

- **segments_tabix_filename** – Path to tabixed bed file containing (chrom, start, end, name, score, strand, start, end, color). The color field is used to determine the color for plotting (R,G,B).
- **chrom** (*str*) – Chromosome of the region to be plotted.
- **start** (*str*) – Start position of the region to be plotted.
- **end** (*str*) – End position of the region to be plotted.
- **ax** (matplotlib.axes._subplots.AxesSubplot, optional) – Axis used for plotting, defaults to None.

Returns Dictionary with keys = names of segments, and values patch

Return type dict

gelviz.basic.**plotHiCContactMap**(*contact_map*, *start*, *end*, *segment_size*, *cmap='Greys'*, *vmin=None*, *vmax=None*, *location='top'*, *ax=None*)

Function that plots HiC contact maps as pyramid plots

Parameters

- **contact_map** (pandas.DataFrame) – Matrix that contains the intensity values of HiC contacts.
- **start** (*int*) – Chromosomal start position of region to be plotted.
- **end** (*int*) – Chromosomal end position of region to be plotted.
- **segment_size** (*int*) – Size of the segments for which contacts were called.
- **cmap** (*str, optional*) – Name of the colormap to be used for plotting HiC intensities, defaults to "Greys".
- **vmin** (*float, optional*) – Minimal value of intensity range to be plotted, defaults to None
- **vmax** (*float, optional*) – Maximal value of intensity range to be plotted, defaults to None.
- **location** (*str, optional*) – Either of "top" | "bottom". If location == "top", the pyramid points upwards, else if location == "bottom" the pyramid points downwards, defaults to top,
- **ax** (matplotlib.axes._subplots.AxesSubplot, optional) – Axis on which to plot contact map, defaults to None.

Returns Nothing to be returned.

Return type None

gelviz.basic.**plotMethylationProfile**(*meth_calls*, *chrom*, *start*, *end*, *color='k'*, *ax=None*)

Function that plots methylation values as dot plots.

Parameters

- **meth_calls** – Iterator containing list-like elements with the following entries:
  1. Chromsome
  2. Start position
  3. end position
  4. Number methylated cytosines

5. Number unmethylated cytosines

- **chrom** (*str*) – Chromosome of region to be plotted.
- **start** (*int*) – Start position of region to be plotted.
- **end** (*int*) – End position of region to be plotted.
- **color** (*str, optional*) – Color of points representing methylation values, defaults to "k".
- **ax** (matplotlib.axes._subplots.AxesSubplot, optional) – Axis of plot, defaults to None.

Returns Nothing to be returned

Return type None

gelviz.basic.**plotMethylationProfileHeat**(*methylation_bed*, *chrom*, *start*, *end*, *bin_size=1000*, *ax=None*)

Function for plotting methylation values as heatmap

Parameters

- **methylation_bed** (pybedtools.BedTool) – Methylation calls. Following fields must be included: Chrom, Start, End, Methylated Cs, Unmethylated Cs.
- **chrom** (*str*) – Chromosome of region to be plotted.
- **start** (*int*) – Start position of region to be plotted.
- **end** (*int*) – End position of region to be plotted.
- **bin_size** (*int, optional*) – size of bin to average methylation values, defaults to 1000.
- **ax** (matplotlib.axes._subplots.AxesSubplot, optional) – Axis to be used for plotting, defaults to None.

Returns Nothing to be returned

Return type None

gelviz.basic.**plotMotifDirections**(*motifs_bed*, *start*, *end*, *head_width=0.2*, *head_length=1000*, *overhang=0*, *color_plus='#80b1d3'*, *color_minus='#fb8072'*, *ax=None*)

Function that plots TF motifs as arrows, indicating their directionality.

Parameters

- **motifs_bed** (pybedtools.BedTool) – pybedtools.BedTool object containing regions of the TF sited to be plotted.
- **start** (*int*) – Start position of the region to be plotted.
- **end** (*int*) – End position of the region to be plotted.
- **head_width** (*float, optional*) – Width of the arrow head as proportion of the arrow, defaults to 0.2
- **head_length** (*int, optional*) – Length of the arrow in bp (depends on the region that is plotted), defaults to 1000.
- **overhang** (*float, optional*) – Fraction that the arrow is swept back (0 overhang means triangular shape). Can be negative or greater than one. Defaults to 0.

- **color_plus** (*str, optional*) – Color of plus stranded TF regions, defaults to "#80b1d3".
- **color_minus** (*str, optional*) – Color of plus stranded TF regions, defaults to "#fb8072".
- **ax** (matplotlib.axes._subplots.AxesSubplot, optional) – Axis on which to plot contact map, defaults to None.

Returns Nothing to be returned.

Return type None

gelviz.basic.**plotRegions**(*regions*, *start*, *end*, *color='#cbebc4'*, *edgecolor=False*, *alpha=1*, *ax=None*)

Functions that plots genomic regions as simple rectangles.

Parameters

- **regions** (*iterator*) – Iterator containig list-like elements with the following entries:
  1. Chromosome
  2. Start position
  3. End position
- **start** (*int*) – Start position of the region to be plotted.
- **end** (*int*) – End position of the region to be plotted.
- **color** (*str, optional*) – Color of the rectangles representing the regions to be plotted, defaults to "#cbebc4".
- **edge_color** (*str, optional*) – Color of region edge. If False, no edge is plotted, defaults to False.
- **alpha** (*float, optional.*) – Alpha value of the rectangle, representing the region to be plotted, defaults to 1.
- **ax** (matplotlib.axes._subplots.AxesSubplot, optional) – Axis of plot, defaults to None.

Returns Nothing to be returned

Return type None

gelviz.basic.**plotTX**(*chrom_r*, *start_r*, *end_r*, *TX_pos*, *direction='right'*, *color='k'*, *ax=None*)

Function that plots a translocation event as a bar, showing the part of the genome that is translocated.

Parameters

- **chrom_r** (*str*) – Chromosome of the region to be plotted.
- **start_r** (*int*) – Start position of the region to be plotted.
- **end_r** (*int*) – End position of the region to be plotted.
- **TX_pos** (*int*) – Position of the translocation.
- **direction** (*str, optional*) – Direction of the genomic part that is translocated. Either of "left" (upstream), or "right" (downstream), defaults to "left".
- **color** (*str, optional*) – Color of the bar representing the translocation, defaults to "k".
- **ax** (matplotlib.axes._subplots.AxesSubplot, optional) – Axis of plot, defaults to None.

Returns Nothing to be returned.

Return type None

gelviz.basic.**readACESeqAsBed**(*input_filename*)

Function that reads CNVs from ACESeq ("*most_important*") files and converts them to pybedtools.BedTool object

Parameters **input_filename** (*str*) – Full path to ACESeq "most_important" file

Returns pybedtools.BedTool object containing CNVs from ACESeq

Return type pybedtools.BedTool

CHAPTER

FOUR

ACKNOWLEDGEMENTS

This package was implemented during my time at the *German Cancer Research Center* in the group of *Theoretical Bioinformatics* headed by Prof. Dr. Roland Eils, where i was part of the core bioinformatics team of the *Heidelberg Institute of Personalized Oncology (HIPO)*. Further refinment and final upload to PyPI was done during my time at *Charite, Universitaetsmedizin Berlin, Berlin Institute of Health (BIH) in the Department of Digital Health* headed by Prof. Roland Eils.

CHAPTER

FIVE

LICENSE

CHAPTER

SIX

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

g

INDEX