LIQUORICE

Release 0.5.5

Peter Peneder

May 04, 2022

CONTENTS:

1 Quickstart **3**

1.1 Introduction . . . 4
1.1.1 Motivation . . . 4
1.1.2 Potential application areas . . . 4
1.1.3 What LIQUORICE does . . . 5
1.1.4 Region-sets . . . 6
1.1.5 How to use LIQUORICE . . . 8
1.1.6 Contact . . . 9
1.1.7 Source code on Github . . . 9
1.2 Installation . . . 9
1.2.1 conda . . . 9
1.2.2 docker . . . 9
1.3 Citation . . . 9
1.4 The *liquorice* python package . . . 10
1.4.1 liquorice.LIQUORICE module . . . 10
1.4.2 liquorice.LIQUORICE_summary module . . . 10
1.4.3 Subpackage: liquorice.utils . . . 15
1.5 The LIQUORICE command-line-tool . . . 40
1.5.1 Arguments . . . 40
1.5.2 LIQUORICE's output . . . 43
1.6 LIQUORICE's summary tool . . . 43
1.6.1 Arguments . . . 43
1.6.2 Output of LIQUORICE's summary tool . . . 44
1.6.3 Assessment of significant differences . . . 45
1.7 Usage, Parameters, and Examples . . . 45
1.7.1 Advice on parameter settings . . . 45
1.7.2 Parallelization . . . 47
1.7.3 Sources for input files . . . 47
1.7.4 Test LIQUORICE with provided test data . . . 48
1.7.5 Example usage of LIQUORICE and the summary tool . . . 48

2 Indices and tables **51**

Python Module Index **53**

Index **55**



Welcome to LIQUORICE's documentation! You can navigate through the menu on the left to learn more about the tool. A very condensed overview follows below:

CHAPTER

ONE

QUICKSTART

LIQUORICE is a bioinformatic command-line-tool and python package for bias correction and quantification of changes in sequencing coverage around regions of interest in cfDNA WGS datasets. LIQUORICE can be used to detect and quantify tissue or cancer-specific epigenetic signatures in liquid biopsies. The tool takes four different files as input:

- a FASTA file of the reference genome
- an indexed BAM file containing the aligned paired-end whole-genome sequencing reads of a liquid biopsy sample
- a BIGWIG mappability file, available for download for hg19 and hg38 here
- one or more BED files representing a set of regions of interest (such as DNase I hypersensitivity sites or enhancer regions specific for a tissue or tumor). Read more *here*.

You can install LIQUORICE like so (*details*):

```
# to install on Linux
conda create -n LIQUORICE -c bioconda -c conda-forge liquorice ray-core

# to activate the environment
conda activate LIQUORICE # or: 'source activate LIQUORICE' for older conda versions
```

You can test LIQUORICE on a small test dataset we provide by following the example below. Please note that the .bam and .bw file used in this example should only be used for testing purposes. They have been processed to keep the file size small.

```
# Set desired nr. of cpus
N_CPUS=5

# download and unzip the reference genome and reference mappability file
wget https://hgdownload.soe.ucsc.edu/goldenPath/hg38/bigZips/p12/hg38.p12.fa.gz
gunzip hg38.p12.fa.gz
wget https://github.com/epigen/LIQUORICE/raw/master/liquorice/data/hg38.p12.fa.fai
wget https://github.com/epigen/LIQUORICE/raw/master/liquorice/data/hg38.fa.mappability_
↪100bp.subsetted_for_testdata.bw

# download .bam file of a healthy control liquid biopsy sample (pre-processed to keep␣
↪the size small)
wget https://github.com/epigen/LIQUORICE/raw/master/liquorice/data/Ctrl_17_testdata.bam
wget https://github.com/epigen/LIQUORICE/raw/master/liquorice/data/Ctrl_17_testdata.bam.
↪bai

# download .bed file for universally accessible DHSs
wget https://github.com/epigen/LIQUORICE/raw/master/liquorice/data/universal_DHSs.bed
```

(continues on next page)

(continued from previous page)

```
# run LIQUORICE
LIQUORICE --bamfile Ctrl_17_testdata.bam --refgenome_fasta "hg38.p12.fa" \
        --mappability_bigwig "hg38.fa.mappability_100bp.subsetted_for_testdata.bw" \
        --bedpathlist "universal_DHSs.bed" \
        --blacklist "hg38" --n_cpus "${N_CPUS}" --extend_to 15000
```



1.1 Introduction

LIQUORICE is a command-line-tool and python package for bias correction and quantification of changes in sequencing coverage around regions of interest whole-genome sequencing datasets of cell-free DNA. LIQUORICE can be used to detect and quantify tissue or cancer-specific epigenetic signatures in liquid biopsies. This allows accurate quantification of the fraction of tumor-derived cell-free DNA, as demonstrated in our publication in Nature Communications. As a tool, LIQUORICE has recently been published in Bioinformatics Advances.

1.1.1 Motivation

The fragmentation of cell-free DNA (cfDNA) obtained from liquid biopsies is non-random, and contains information about the epigenetic state of its cell-of-origin. DNA in nucleosomes is protected from cleavage, resulting in an increase in sequencing coverage in genomic regions that are occupied by nucleosomes. If a set of genomic regions is specifically accessible in a cell-, tissue- or cancer type of interest, an observed decrease in sequencing coverage around these regions can indicate that cells of this type are present in the organism, and are releasing cfDNA into the bloodstream.

1.1.2 Potential application areas

LIQUORICE could be useful for a variety of cfDNA-related analysis tasks:

- Cancer detection, classification, monitoring and survival prognosis (as demonstrated in our recent publication)
- Classification of cancers of unknown primary
- Location of metastasis
- Tissue damage monitoring for transplantations, heart attack, and other diseases

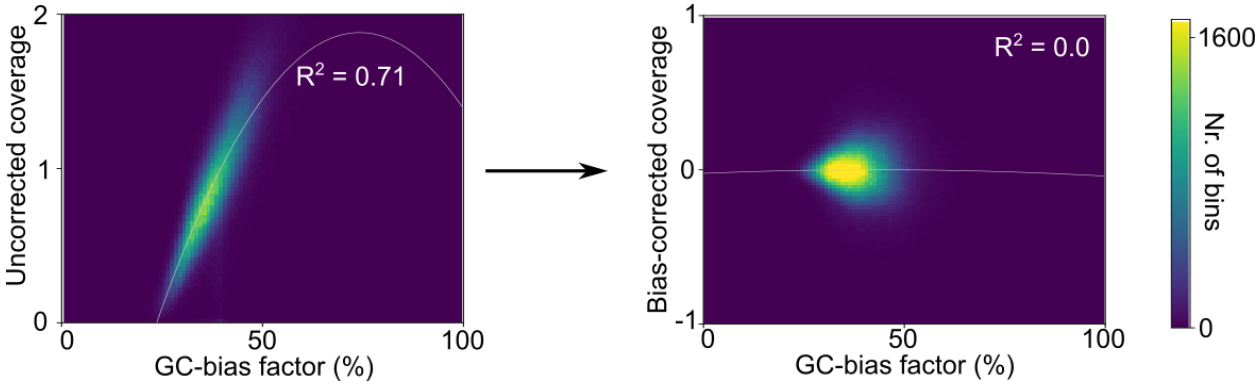
1.1.3 What LIQUORICE does

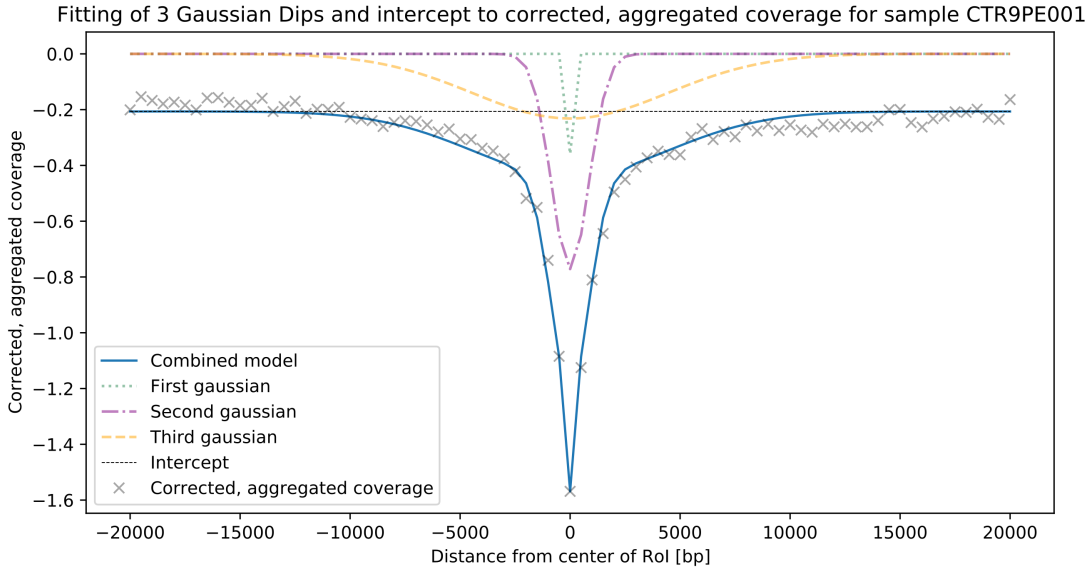
LIQUORICE addresses two problems regarding the quantification of epigenetic signatures from a sequencing coverage signal:

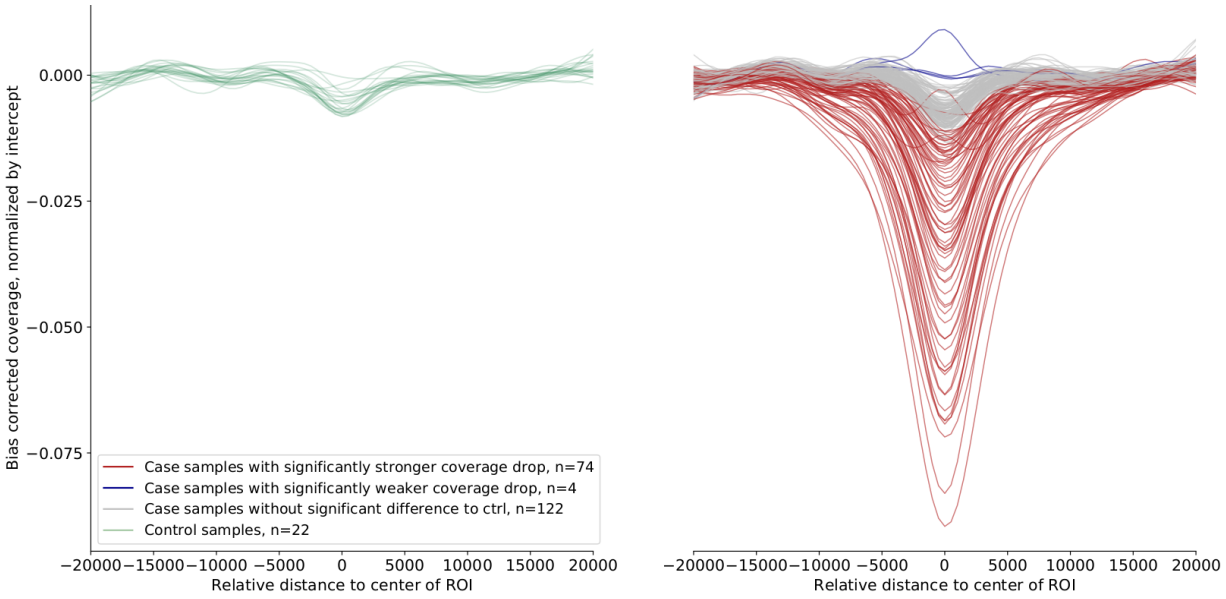
- Coverage is influenced by bias-factors, such as GC-content, mappability and other factors related to sequence composition
- The coverage signal needs to be summarized over all regions in a region-set, and needs to be quantified as a number that is robust and comparable between samples

In a first step, LIQUORICE trains a bias-model, which learns the association between bias-factors and coverage. Then, the trained model is used to correct the coverage signal in the regions of interest. Finally, the signal is aggregated between regions and quantified by fitting a model to the data. LIQUORICE produces tables as well as plots that allow for the visual inspection of biases and of the coverage signal. Furthermore, a summary tool is included, which allows for the convenient comparison of signals between samples and region-sets.

Here are a few examples of LIQUORICE's output:



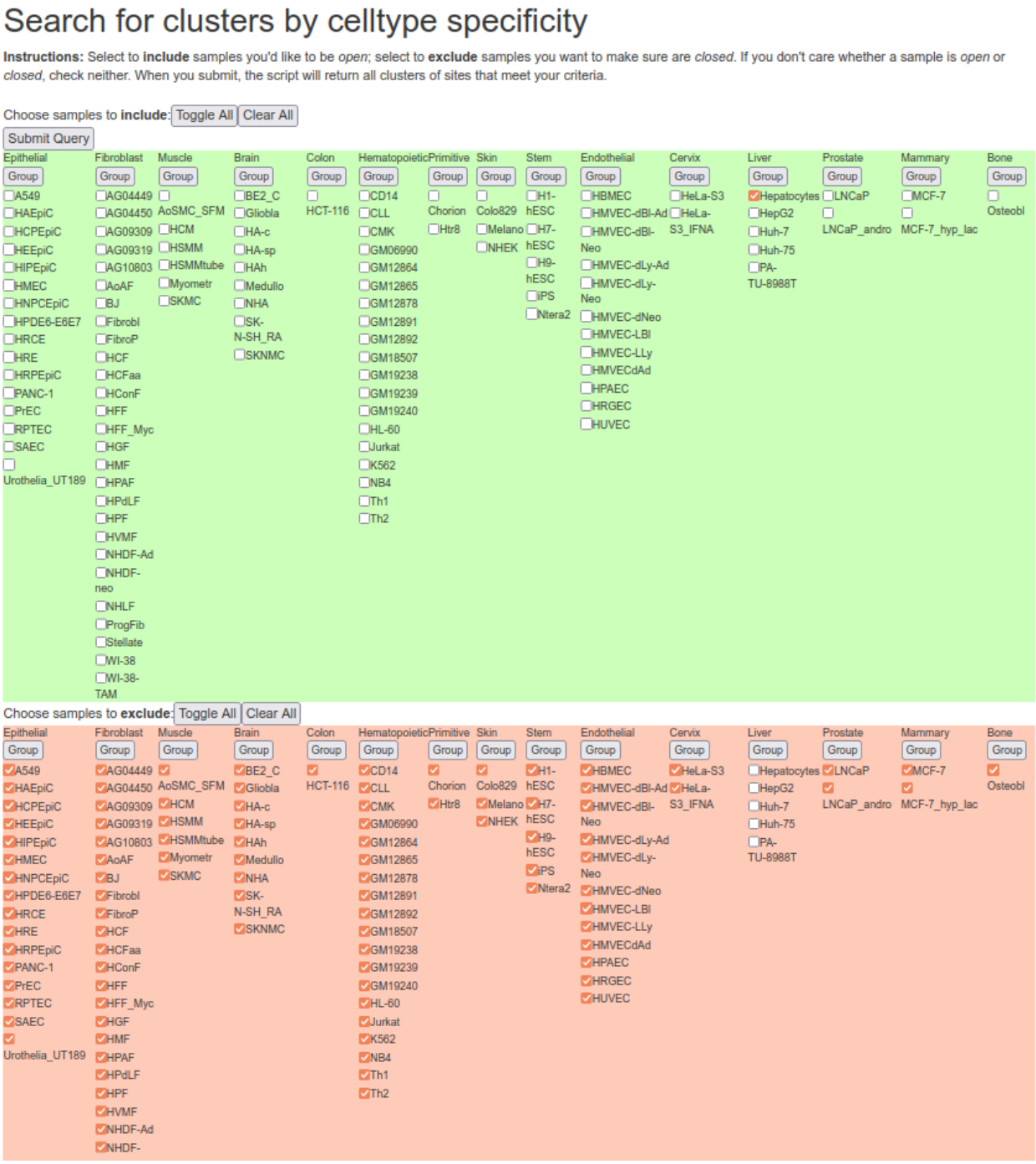




1.1.4 Region-sets

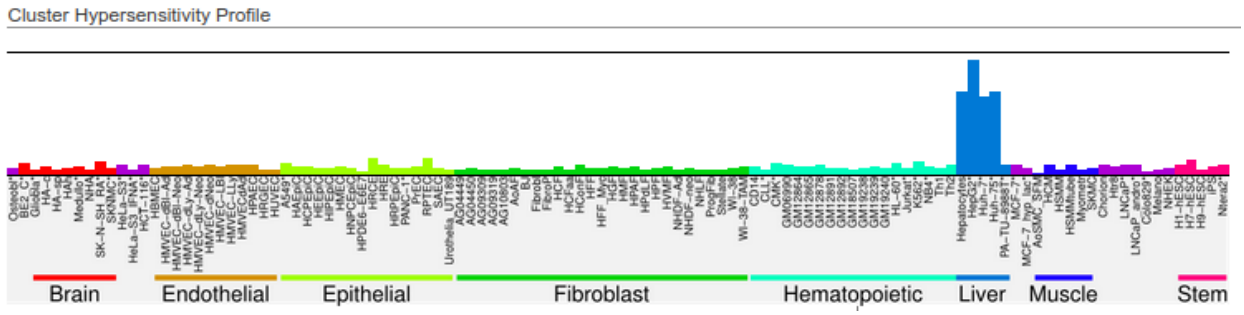
In the context of `LIQUORICE`, we refer to a region-set as a set of genomic regions with similar properties. This could be, e.g., enhancers or DNase I hypersensitivity sites that are specific for a cell-type, cancer-type, or tissue. Useful ressources to obtain such region-sets include http://dnase.genome.duke.edu/ and ENCODE.

Here is an example on how we obtained liver-specific DNase I hypersensitivity sites from the http://dnase.genome.duke.edu/celltype.php interface. First, we required all non-liver cells to be "closed", and we required hepatocytes to be "open" at a given DHS:

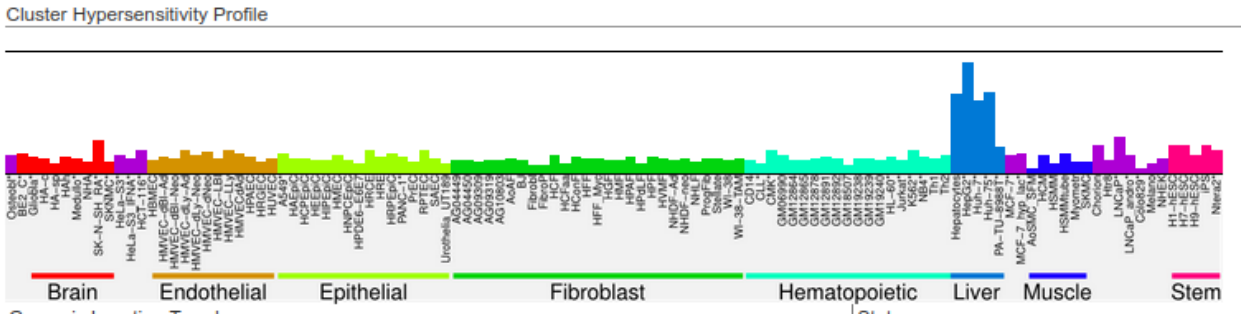


Then, we selected two clusters that have an adequate tissue-specific accessibility pattern:

http://dnase.genome.duke.edu/clusterDetail.php?clusterID=1066



and http://dnase.genome.duke.edu/clusterDetail.php?clusterID=1115



We downloaded the .bed files (with all regions) from these clusters, and finally, we concatenated the downloaded files, merging the DHSs of these two clusters in order to get our final liver-specific DHS set. Note that the regions on the database are stored in hg19 coordinates - you can use LiftOver to convert the data to hg38.

1.1.5 How to use `LIQUORICE`

The easiest way to use `LIQUORICE` is via its *command-line interface*. Also check out the convenient *summary tool* if you are analyzing multiple samples or region-sets.

If you require more control about the steps performed by LIQUORICE, check out the *LIQUORICE python package* - you can start by looking at the function-level documentation and source code of the *liquorice.LIQUORICE module* as well as the the *liquorice.utils.Workflows module*.

More details on parameters, test examples and usage examples can be found *here*

1.1.6 Contact

If you have any questions about LIQUORICE and how to apply it to your data, create an issue on github or contact peter.peneder@ccri.at - we are happy to hear from you.

1.1.7 Source code on Github

The github repository of LIQUORICE can be found at https://github.com/epigen/liquorice.

1.2 Installation

1.2.1 conda

LIQUORICE can easily be installed via conda (or mamba , a faster alternative to conda). We recommend installing LIQUORICE in its own conda environment to keep dependencies clean:

```
# to install on Linux
conda create -n LIQUORICE -c bioconda -c conda-forge liquorice ray-core

# to install on macOS
# conda create -n LIQUORICE -c bioconda -c conda-forge liquorice

# to activate the environment
conda activate LIQUORICE # or: 'source activate LIQUORICE' for older conda versions

# to run
LIQUORICE <flags and arguments>
```

'ray-core' is an optional dependency of LIQUORICE that helps it run faster on multiple cores. It is not possible to install it via conda on macOS, so installing it can either be skipped (find details about an alternative parallelization approach *here*) or it can be installed as described here (e.g. *pip install -U ray==1.1.0*). LIQUORICE has been tested for ray version 1.1.0.

1.2.2 docker

You can also use the LIQUORICE docker image which is available here. To pull, use

```
docker pull peneder/liquorice
```

1.3 Citation

If you use LIQUORICE in any published work, please cite:

Peneder, P., Bock, C. & Tomazou, E. M. (2022). LIQUORICE: detection of epigenetic signatures in liquid biopsies based on whole-genome sequencing data. Bioinformatics Advances, 2(1), vbac017.

Peneder, P., Stütz, A. M., Surdez, D., Krumbholz, M., Semper, S., Chicard, M., … & Tomazou, E. M. (2021). Multimodal analysis of cell-free DNA whole-genome sequencing for pediatric cancers with low mutational burden. Nature communications, 12(1), 1-16.

1.4 The *liquorice* python package

1.4.1 liquorice.LIQUORICE module

class liquorice.LIQUORICE.**FullPaths**(*option_strings*, *dest*, *nargs=None*, *const=None*, *default=None*, *type=None*, *choices=None*, *required=False*, *help=None*, *metavar=None*)

Bases: argparse.Action

Expand user- and relative-paths. From: https://gist.github.com/brantfaircloth/1252339. Does not convert the path to the blacklist if the value is 'hg38' or '10k_random, which is interpreted as an instruction to use the shipped list instead.

liquorice.LIQUORICE.**main**()

Main function for the LIQUORICE command line tool. Performs the following steps: Sanity checks for user input, setup of environment variables and general setup-steps, handles file paths, selects which steps need to be performed based on user input, logs the progress and performs the actual LIQUORICE analysis by calling *liquorice.utils.GlobalFragmentSize.get_list_of_fragment_lengths_and_avg_readlength()*, *liquorice.utils.MeanSequencingDepth.sample_bam_to_get_sequencing_depth()*, *liquorice.utils.Workflows.train_biasmodel_for_sample()*, *liquorice.utils.Workflows.run_liquorice_on_regionset_with_pretrained_biasmodel()*, and/or *liquorice.utils.Workflows.run_liquorice_train_biasmodel_on_same_regions()* after generating the corresponding objects.

liquorice.LIQUORICE.**parse_args**()

Parses the arguments from the command line. For a full list of arguments, see the documentation of the LIQUORICE command line tool.

Returns An *argparse.ArgumentParser* object storing the arguments.

1.4.2 liquorice.LIQUORICE_summary module

liquorice.LIQUORICE_summary.**boxplot_score_summary**(*summary_df*, *comparison_col*, *use_uncorrected_coverage*)

Summarize scores via boxplots, with one box per group (case/control) - region-set combination.

Parameters

- **summary_df** (DataFrame) – The output of the function *create_summary_table_LIQUORICE()*.
- **comparison_col** (str) – Use this column for the y axis of the boxplots.
- **use_uncorrected_coverage** (bool) – If True, indicate in the output filename that the uncorrected coverage scores are shown.

liquorice.LIQUORICE_summary.**check_difference_to_control**(*row*, *control_df*, *col*, *prediction_interval_control_col*, *negative_is_strong*)

For a row in a *pandas.DataFrame*, find control values for the same regionset and return if sample is significantly different compared to the controls, as measured by metric **col**. A sample is deemed significantly different if

its score in metric **col** lies outside the prediction interval of the control group.

Parameters

- **row** (Series) – A *pandas.Series* with the columns 'region-set', 'is control', and **col**, corresponding to a single sample.

- **control_df** (DataFrame) – A *pandas.DataFrame* with at least the columns 'region-set', and **col**. Should only contain data of control samples.
- **col** (str) – Use this column as a metric to determine differences.
- **prediction_interval_control_col** (str) – Name of the column that contains the prediction interval of the control group.
- **negative_is_strong** (bool) – Set to *True* if a very negative value of the metric is associated with a strong dip signal, such as for the dip area.

Return type str

Returns A string indicating the result of the comparison, or np.nan if row[col] is NaN.

liquorice.LIQUORICE_summary.**create_summary_table_LIQUORICE**(*dirname*, *control_name_list*, *these_regionsets_only*, *use_uncorrected_coverage=False*, *prediction_interval_alpha=0.05*)

For a LIQUORICE result directory, creates and writes to csv a pd.DataFrame summarizing the coverage drop metrics for all samples and region-sets. If **control_name_list** is given, compares the case samples to control samples and assesses significant differences, separately for each region-set.

Parameters

- **use_uncorrected_coverage** (bool) – If True, summarize the coverage profile that is not corrected for biases by LIQUORICE instead of the corrected coverage.
- **these_regionsets_only** (List[str]) – Summarize only data for these regionsets.
- **dirname** (str) – Output directory of LIQUORICE in which to search for fitted_gaussians_parameter_summary.csv (or uncorrected_fitted_gaussians_parameter_summary.csv) files
- **control_name_list** (List[str]) – Sample names of the controls, which will be used as reference for generating z-scores.
- **prediction_interval_alpha** (int) – Alpha level for the prediction interval. Default 0.05: 95% prediction interval

Return type DataFrame

Returns A *pandas.DataFrame* in which all parameters saved in the 'fitted_gaussians_parameter_summary.csv' (or uncorrected_fitted_gaussians_parameter_summary.csv) files are summarized over all samples and region-sets. If control_name_list is not empty, additional columns of the DataFame contain metric comparisons to the the control samples in the form of z-scores of dip area and depth.

liquorice.LIQUORICE_summary.**get_binsize_and_extendto_from_saved_settings**(*dirname*, *regionset*)

Calls *verify_consistant_binning_settings()* and returns the binsize and extend_to settings used by LIQUORICE for a given regionset, if no error is raised.

Parameters

- **dirname** (str) – Path to LIQUORICE's working directory.
- **regionset** (str) – Name of the regionset of interest.

Return type Tuple[int, int]

Returns A tuple with two integers: binsize and extend_to.

liquorice.LIQUORICE_summary.**get_list_of_coveragefiles**(*dirname*, *regionset*, *use_uncorrected_coverage=False*)

Return a list of 'corrected_coverage_mean_per_bin.csv' or 'uncorrected_coverage_mean_per_bin.csv' files, or return an error if no such files could be found.

Parameters

- **dirname** (str) – path to LIQUORICE output directory
- **regionset** (str) – Name of the regionset of interest
- **use_uncorrected_coverage** (bool) – If True, load 'uncorrected_coverage_mean_per_bin.csv' files, else, load 'corrected_coverage_mean_per_bin.csv files

Return type List[str]

Returns A list of 'corrected_coverage_mean_per_bin.csv' or 'uncorrected_coverage_mean_per_bin.csv' file paths

liquorice.LIQUORICE_summary.**get_prediction_interval**(*values*, *alpha=0.05*)

Returns the upper and lower prediction interval for a list of values.

Parameters

- **values** (List) – Scores for which the prediction interval shall be calculated
- **alpha** (int) – Alpha level for the prediction interval. Default 0.05: 95% prediction interval

Return type Tuple[float, float]

Returns A tuple (lower_prediction_interval, upper_prediction_interval)

liquorice.LIQUORICE_summary.**get_prediction_interval_per_row**(*row*, *control_df*, *col*, *alpha=0.05*)

For a row in a *pandas.DataFrame*, find control values for the same regionset and return the prediction interval.

Parameters

- **row** (Series) – A *pandas.Series* with the columns 'region-set', 'is control', and **col**, corresponding to a single sample.
- **control_df** (DataFrame) – A *pandas.DataFrame* with at least the columns 'region-set', and **col**. Should only contain data of control samples.
- **col** (str) – Use this column as a metric to determine differences.
- **alpha** (int) – Alpha level for the prediction interval. Default 0.05: 95% prediction interval

Return type Tuple[float, float]

Returns A tuple (lower prediction interval,higher prediction interval), or (np.nan,np.nan) if calculation is not possible.

liquorice.LIQUORICE_summary.**get_ymin_ymax_over_all_samples**(*coverage_filelist*, *normalize_by_intercept*, *regionset*, *summary_df*)

Get the the upper and lower limits for the y-axis for a given regionset, such that the same scale is used for all samples.

Parameters

- **coverage_filelist** (List[str]) – List of 'corrected_coverage_mean_per_bin.csv' (or 'uncorrected_coverage_mean_per_bin.csv') files

- **normalize_by_intercept** (bool) – f True, extract the intercept from the fitted model to normalize the dips between samples (intercept of each sample is positioned at y=0). Otherwise, the mean coverage of each sample is positioned at y=0.
- **regionset** (str) – Name of the regionset of intest as given in the summary_df
- **summary_df** (DataFrame) – pd.DataFrame with columns 'sample', 'region-set', and 'Intercept'.

Return type Tuple[float, float]

Returns A tuple of floats: y_min,y_max

liquorice.LIQUORICE_summary.**main**()

Main function for the LIQUOIRCE_summary tool. Calls the argument parser, checks user input, and calls the functions *verify_consistant_binning_settings()* *create_summary_table_LIQUORICE()* (saving output to .csv), *plot_overlay()*, *plot_as_subplots()*, *boxplot_score_summary()* and *plot_control_distributions_and_test_normality()*.

liquorice.LIQUORICE_summary.**parse_args**()

Parses the arguments from the command line. For a full list of arguments, see the documentation of the LIQUORICE_summary command line tool.

Returns An *argparse.ArgumentParser* object storing the arguments.

liquorice.LIQUORICE_summary.**plot_as_subplots**(*dirname*, *summary_df*, *control_name_list*, *extend_to=None*, *binsize=None*, *normalize_by_intercept=True*, *smooth_sigma=0*, *significance_col='Dip area: interpretation vs controls in same region set'*, *use_uncorrected_coverage=False*, *y_min_fixed=None*, *y_max_fixed=None*)

Summarize plots, create one set of plots per regionset. Case samples with significant differences to controls (based on **significance_col**) are marked by color.

smooth_sigma can be used to make the plots smoother and easier to compare, this is a visual effect only.

Parameters

- **dirname** (str) – Path to LIQUORICE's working directory.
- **summary_df** (DataFrame) – The output of the function *create_summary_table_LIQUORICE()*.
- **control_name_list** (List[str]) – List of names of samples that should be plotted as controls. Can be empty.
- **extend_to** (Optional[int]) – *extend_to* setting that was used when running LIQUORICE. If None, infer from the binning_settings.json files that are saved by LIQUORICE by default.
- **binsize** (Optional[int]) – *binsize* setting that was used when running LIQUORICE. If None, infer from the binning_settings.json files that are saved by LIQUORICE by default.
- **normalize_by_intercept** (bool) – If True, extract the intercept from the fitted model to normalize the dips between samples (intercept of each sample is positioned at y=0). Otherwise, the mean coverage of each sample is positioned at y=0.
- **smooth_sigma** (float) – Visually smooth the coverage signals, using this strength of smoothing. Higher values indicate stronger smoothing. Set to 0 for no smoothing.

- **significance_col** (str) – Use this columns as an indicator for significant differences between case and control samples. Can contain the following values: "Significantly stronger coverage drop", "Significantly weaker coverage drop", "n.s.", and "N/A".
- **use_uncorrected_coverage** (bool) – If True, plot the coverage profile that is not corrected for biases by LIQUORICE.
- **y_min_fixed** (Optional[float]) – If specified, use this value as the minimum value for the y axis.
- **y_min_fixed** – If specified, use this value as the maximum value for the y axis.

liquorice.LIQUORICE_summary.**plot_control_distributions_and_test_normality**(*summary_df*, *col*, *alpha=0.05*, *use_uncorrected_coverage=False*)

For each region-set in **summary_df**, plot the distribution of the control sample's score measured by metric **col** as histograms and probability plots (https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.probplot.html). Also run tests the Shapiro-Wilk test for normal distributions and prints a warning if this test detects significant deviations from the normal distribution. Plotting and testing is skipped for every region-set where n<4.

Parameters

- **summary_df** (DataFrame) – Input *pandas.DataFrame* with the columns "is_control" and **col**
- **col** (str) – Name of the column that should be used for plotting
- **use_uncorrected_coverage** (bool) – If True, indicate in the file name that the uncorrected coverage has been used to generate the underlying data.

Returns

liquorice.LIQUORICE_summary.**plot_overlay**(*dirname*, *summary_df*, *control_name_list*, *extend_to=None*, *binsize=None*, *normalize_by_intercept=True*, *smooth_sigma=3*, *significance_col='Dip area: interpretation vs controls in same region set'*, *use_uncorrected_coverage=False*, *alpha=0.5*, *linewidth=3*)

Summarize plots, create one plot per regionset. Case samples with significant differences to controls (based on **significance_col**) are marked by color. **smooth_sigma** can be used to make the plots smoother and easier to compare - this is a visual effect only.

Parameters

- **dirname** (str) – Path to LIQUORICE's working directory.
- **summary_df** (DataFrame) – The output of the function *create_summary_table_LIQUORICE()*.
- **control_name_list** (List[str]) – List of names of samples that should be plotted as controls. Can be empty.
- **extend_to** (Optional[int]) – *extend_to* setting that was used when running LIQUORICE. If None, infer from the binning_settings.json files that are saved by LIQUORICE by default.
- **binsize** (Optional[int]) – *binsize* setting that was used when running LIQUORICE. If None, infer from the binning_settings.json files that are saved by LIQUORICE by default.
- **normalize_by_intercept** (bool) – If True, extract the intercept from the fitted model to normalize the dips between samples (intercept of each sample is positioned at y=0). Otherwise, the mean coverage of each sample is positioned at y=0.

- **smooth_sigma** (float) – Visually smooth the coverage signals, using this strength of smoothing. Higher values indicate stronger smoothing. Set to 0 for no smoothing.
- **significance_col** (str) – Use this columns as an indicator for significant differences between case and control samples. Can contain the following values: "Significantly stronger coverage drop", "Significantly weaker coverage drop", "n.s.", and "N/A".
- **use_uncorrected_coverage** (bool) – If True, plot the coverage profile that is not corrected for biases by LIQUORICE.
- **alpha** (float) – Alpha (transparency) parameter for plotting.
- **linewidth** (float) – Linewidth parameter for plotting

liquorice.LIQUORICE_summary.**verify_consistant_binning_settings**(*dirname*, *regionset*)

Asserts that, for the given regionset, all samples have the same binning settings. Calls *sys.exit()* with an error message if settings are inconsistent or cannot be found.

Parameters **regionset_list** – A list of region-sets to be analyzed.

liquorice.LIQUORICE_summary.**zscore_to_control**(*row*, *control_df*, *col*)

For a row in a dataframe, find control values for the same regionset and return the zscore of the sample compared to these controls, as measured by metric "col".

Parameters

- **row** (Series) – A pd.Series with the columns 'region-set', 'is control', and **col**, corresponding to a single sample.
- **control_df** (DataFrame) – A *pandas.DataFrame* with at least the columns 'region-set', and **col**. Should only contain data of control samples.
- **col** (str) – Calculate z-score by comparing this metric between the sample and controls.

Return type float

Returns z-score, rounded to 2 decimals

1.4.3 Subpackage: liquorice.utils

liquorice.utils.AggregateAcrossRegions module

liquorice.utils.AggregateAcrossRegions.**aggregate_across_regions**(*df*, *column_of_interest*)

Parameters

- **df** (DataFrame) – A pandas.DataFrame, containing the columns "bin nr." and **column_of_interest**
- **column_of_interest** (str) – Column name for which the mean per bin nr. should be returned

Return type Series

Returns A pandas.Series with mean values of **column_of_interest** per bin, aggregated across all regions in the pandas.DataFrame.

Example:

```
mean_corrected_coverage_leftmost_bin = aggregate_across_regions(df,"corrected␣
↪coverage").loc[0]
```

liquorice.utils.BiasFactorWeights module

liquorice.utils.BiasFactorWeights.**get_GC_weights_binsize1**(*fragments*)
For a given fragment size distribution, returns a list corresponding to the GC weight vector for a bin of size 1

Parameters **fragments** (List[int]) – A list of fragment lengths that is representative of the sample's global fragment length distribution

Return type List[float]

Returns A list corresponding to the GC weight vector for a bin of size 1. Its length is *max(fragments)*2*, the weight corresponding to the bin itself is at index *max(fragments)*. Values represent influence of a nucleotide at a given position on the bin's coverage.

liquorice.utils.BiasFactorWeights.**get_GC_weights_binwide**(*binsize*, *fragments*, *dont_slide=False*)
For a given binsize, calculate the influence of a nucleotide at a given position on the bin's coverage

Parameters

- **binsize** (int) – Size of the bin
- **fragments** (List[int]) – A list of fragment lengths that is representative of the samples global fragment length distribution
- **dont_slide** (bool) – Set to False to return weights of 1 in the bin area and 0 outside of it

Return type List[float]

Returns A list of length *(2*max(fragments)+binsize)*, the weight corresponding to the first position within the bin itself is at index *max(fragments)*. Values represent influence of a nucleotide at a given position on the bin's coverage.

liquorice.utils.BiasFactorWeights.**get_dinuc_weights_binwide**(*GC_weights*)
For a given binsize, calculate the influence of a given dinucleotide starting at a given position on the bin's coverage. This simply averages the GC weight of the position of interest and the following position. For the last position, a weight of 0 is returned.

Parameters **GC_weights** (List[float]) – result of *get_GC_weights_binwide()* for the appropriate binsize and fragments of interest.

Return type List[float]

Returns A list of length *(2*max(fragments)+binsize)*, the weight corresponding to the first position within the bin itself is at index *max(fragments)*. Values represent influence of a dinucleotide starting at a given position on the bin's coverage.

liquorice.utils.BiasFactorWeights.**get_mapp_weights_binwide**(*binsize*, *fragments*, *dont_slide=False*)
For a given binsize, calculate the influence a fragment starting/ending at a given position on the bin's coverage

Parameters

- **binsize** (int) – Size of the bin
- **fragments** (List[int]) – A list of fragment lengths that is representative of the samples global fragment length distribution
- **dont_slide** (bool) – Set to False to return weights of 1 in the bin area and 0 outside of it

Return type Tuple[List[float]]

Returns Two lists of length *(2*max(fragments)+binsize)*. Values represent influence a fragment starting/ending at a given position on the bin's coverage. First list is for forward mappability (fragments starting at the position), second list is for reverse mappability (fragments ending at the position)

liquorice.utils.BiasFactorWeights.**get_nucleotide_dicts**()

Prepare dictionaries with di-and trinucleotides, as well as a dictionary for forward/reverse complement translation.

Return type Tuple[Dict[str, int]]

Returns Three dictionaries: All dinucleotides excluding reverse complements, all trinucleotides excluding reverse complements, and a dictonary that allows translation of reverse complements to their forward complement counterparts (in this order). Keys are all 0.

liquorice.utils.BiasFactorWeights.**get_trinuc_weights_binwide**(*GC_weights*)

For a given binsize, calculate the influence of a given trinucleotide starting at a given position on the bin's coverage. This simply averages the GC weight of the position of interest and the following 2 positions. For the last two positions, a weight of 0 is returned.

Parameters **GC_weights** (List[float]) – result of get_GC_weights_binwide for the binsize and fragments of interest.

Return type List[float]

Returns A list of length *(2*max(fragments)+binsize)*, the weight corresponding to the first position within the bin itself is at index *max(fragments)*. Values represent influence of a trinucleotide starting at a given position on the bin's coverage.

liquorice.utils.BiasModel module

class liquorice.utils.BiasModel.**BiasModel**(*training_df*, *df_to_correct*, *biasmodel_path='trained_biasmodel.joblib'*, *features='all'*, *nr_of_bins_for_training_and_testing=10000*, *sklearn_model=None*, *n_jobs=1*, *filename_performance_metrics='biasmodel_performance_metrics.csv'*, *filename_feature_importances=None*, *use_binsize_as_feature=False*)

Bases: object

An object that can be used to train a machine learning model that predicts coverage based on bias factors. The performance of the trained model can be tested, and it can be used to correct coverage values by regressing out the influence of the bias factors.

Parameters

- **training_df** (Optional[DataFrame]) – *pandas.DataFrame* used for training (and optionally for performance assessment) of the model. Must contain the column *coverage* and all columns specified under **features**. Can be None if *train_biasmodel()* won't be called. Ignored in :func`train_biasmodel_2fold_CV_and_predict`.
- **df_to_correct** (Optional[DataFrame]) – *pandas.DataFrame* for which coverage should be corrected by the trained model. Must contain the column *coverage* and all columns specified under **features**. Can be None if *get_table_with_corrected_coverage_using_trained_biasmodel()* won't be called.
- **biasmodel_path** (str) – Path to which the trained biasmodel should be saved to and/or loaded from. Must be a .joblib file. Can be None if *get_table_with_corrected_coverage_using_trained_biasmodel()* won't be called, in that case, no biasmodel will be saved.
- **features** (Union[str, List[str]]) – A list of bias factors that should be used as features for the machine learning model. Default "all" sets all bias-factors as features: forward,reverse,

and max mappability, di/trinucleotide factors and GC-content. Bin size is not included by default, it can be added as a feature via *use_binsize_as_feature*.

- **nr_of_bins_for_training_and_testing** (Optional[int]) – Subset the training_df to this many bins. Can speed up the computation time, but using too few bins will make the model less precise. Can be None, then all bins will be used.
- **sklearn_model** (Optional[*SklearnStyleRegressor*]) – A regressor that implements to functions .fit() and .predict() (e.g. from *sklearn*). Default of None means using sklearn.ensemble.HistGradientBoostingRegressor with default settings.
- **n_jobs** (int) – How many jobs to run in parallel when training the model
- **filename_performance_metrics** (Optional[str]) – If **test_fraction** or **cross_validate_k** is set, save a .csv containing the performance metrics (r2, MSE) to this path.
- **filename_feature_importances** (Optional[str]) – If set, save a .csv file containing the feature importances inferred from the trained model to this path.
- **use_binsize_as_feature** (bool) – If True, include "bin size" as a feature for the model.

get_table_with_corrected_coverage_using_trained_biasmodel()

Predicts coverage based on `features` in the DataFrame `df_to_correct` and the biasmodel under `biasmodel_path`. Subtracts this prediction from the observed coverage to regress out the effect of the bias-factors (i.e. `features`) on the coverage.

Return type DataFrame

Returns Returns `df_to_correct` with an additional column "corrected coverage".

train_biasmodel()

Train a machine learning model that predicts the values of the 'coverage' column based on the given features. If `test_fraction` is set, this fraction of the `training_df` is set aside to evaluate R^2 and MSE of the model. Prior to training and a potential train/test split, the `training_df` is subsetted to `nr_of_bins_for_training_and_testing` if it is not *None*. Writes writes a *.joblib* file containing the biasmodel to `biasmodel_path` (unless it is *None*)

Return type None

train_biasmodel_2fold_CV_and_predict(*exclude_these_bin_nrs=[]*)

Train a machine learning model that predicts the values of the 'coverage' column based on the given features. Will use each half of the `df_to_correct` to train the model for predictions of the other half. Ignores `nr_of_bins_for_training_and_testing`, `cross_validate_k` and writes returns the performance

metrics and returns the dataframe with predictions. Important: does not use :attr`training_df`.

Return type None

class liquorice.utils.BiasModel.**SklearnStyleRegressor**(**args*, ***kwds*)

Bases: typing_extensions.Protocol

Introduced here for typing purposes only.

fit(*X*, *y*, *sample_weight=None*)

predict(*X*)

score(*X*, *y*)

set_params(*X*, *y*)

liquorice.utils.BinTableBiasFactors module

class liquorice.utils.BinTableBiasFactors.**BiasFactorHandler**(*binsize*, *fragments*, *readlength*, *df*, *n_cores=1*, *skip_these_biasfactors=[]*)

Bases: object

Object used for calculation of per-bin bias factors. Typically, after creation of the object a user would call its method `get_table_with_bias_factors()` on it, and save the returned *DataFrame* for subsequent analysis and correction of associations of bias factors with coverage.

Parameters

- **binsize** (int) – Size of the bins. Higher values to reduce noise, lower values increase spatial resolution.
- **fragments** (List[int]) – A list containing fragment lengths that are representative of the sample's global fragment size distribution. Typically a few hundred fragments will suffice here.
- **readlength** (int) – Average length of reads in the .bam file.
- **df** (DataFrame) – *pandas.DataFrame* with one row per bin, containing columns "chromosome", "start", "end", "bin nr.", "coverage", "sequence", and "mappability". Suitable input is the output of the `get_complete_table()` method of the *liquorice.utils.CoverageAndSequenceTablePreparation* class object.
- **n_cores** (int) – Max number of cores to use for calculations.
- **skip_these_biasfactors** (List[str]) – Do not calculate these bias factors. Only these entries are allowed: ["di and trinucleotides and GC content","mappability", "di and trinucleotides"]

get_GC_and_di_and_trinuc_weights(*sequence*)

Calculate bias factors for GC-content as well as bias factors for all di- and trinucleotides (reverse and forward complements merged into single factors) for a given sequence. Factors are scaled between 0 and 1, 1 is highest (e.g. GC content: 0.. no G or C, 0.461… average GC content, 1… only G and C).

Parameters **sequence** (str) – Genomic sequence of the bin extended by *max(* `fragments` *)* in both directions, such that its length matches `GC_weights`.

Return type Dict[str, float]

Returns A dictionary with entries corresponding to the bias factors for GC-content as well as bias factors for all di- and trinucleotides (reverse and forward complements merged into single factors).

get_GC_bias_factor(*sequence*)

Returns a number in the range 0-1 that represents the bin's overall GC bias. Factors are scaled between 0 and 1, 1 is highest (e.g. GC content: 0.. no G or C, 0.461… average GC content, 1… only G and C).

Parameters **sequence** (str) – Genomic sequence of the bin extended by *max(* `fragments` *)* in both directions, such that its length matches `GC_weights`.

Return type float

Returns A number in the range 0-1 that represents the bin's overall GC bias.

get_fwd_mappability_bias_factor(*mappability*)

Returns a number representing the bin's mappability for fragments on the forward strand.

Parameters **mappability** (List[float]) – A vector of values between 0 and 1, representing the mappbility of positions in and around the bin of interest. Must correspond to the coordinates

<Bin start coord> - *max(* `fragments` *)* - `readlength` to <Bin end coord> + *max(* `fragments` *)* + `readlength`

Return type `float`

Returns A number in range 0-1 that represents the bin's overall forward mappability. 1 … highest

`get_rev_mappability_bias_factor`(*mappability*)

Returns a number representing the bin's mappability for fragments on the reverse strand.

Parameters **`mappability`** (`List[float]`) – A vector of values between 0 and 1, representing the mappbility of positions in and around the bin of interest. Must correspond to the coordinates <Bin start coord> - *max(* `fragments` *)* - `readlength` to <Bin end coord> + *max(* `fragments` *)* + `readlength`

Return type `float`

Returns A number in range 0-1 that represents the bin's overall reverse mappability. 1 … highest

`get_table_with_bias_factors`()

Main method to retrieve bias factor information.

Return type `DataFrame`

Returns A *pandas.DataFrame* with one row per bin, containing columns of the input DataFrame `df` ("chromosome", "start", "end", "bin nr.", "coverage", "sequence") as well as newly added columns for bias factors for mappability, GC-content, and all di- and trinucleotides (with reverse and forward complements merged into single factors). Bias factors are scaled between 0 and 1. Higher values correspond to higher nucleotide content, mappability etc. Example: An average bin is expected to have a bias factor of 0.461 for humans (i.e. genomic GC content). Note that the input columns "mappability" and "sequence" will not be part of the returned dataframe in order to reduce the memory usage of this function.

liquorice.utils.CorrectForCNAs module

`liquorice.utils.CorrectForCNAs.`**`correct_coverage_per_bin_for_cnas`**(*df*, *cna_seg_filepath*, *n_cores=1*)

Extracts the log2(observed/expected) read depth ratio for the corresponding segment for each bin, and corrects each bin's coverage accordingly.

Parameters

- **`df`** (`DataFrame`) – A *pandas.DataFrame* with at least the following columns: "chromosome", "start","end","coverage"
- **`cna_seg_filepath`** (`str`) – A .seg file that should be used to correct the coverage by the values specified in this file. File must be tab-separated, with column names as first line. The second, third and fourth column must be chromosome, start, and end of the segment, and the last column must be the log2-ratio of observed/expected read depth.
- **`n_cores`** (`int`) – How many cores to use to parallelize the operation.

Return type `DataFrame`

Returns A *pandas.DataFrame* similar to the input *df*, with the values in the column "coverage" changed according to the CNA correction factor of each bin, and an additional column "CNA-uncorrected coverage" that contains the original coverage values.

liquorice.utils.CorrectForCNAs.**get_CNV_corrected_coverage_for_bin**(*bin_chrom*, *bin_start*, *bin_end*, *bin_coverage*, *seg_df*)

For a given bin, returns the coverage corrected for copy number aberrations.

Parameters

- **bin_chrom** (str) – Chromosome of the bin.
- **bin_start** (int) – Genomic start position of the bin.
- **bin_end** (int) – Genomic end position of the bin.
- **bin_coverage** (float) – Bin's coverage before CNA correction
- **seg_df** (DataFrame) – A *pandas.DataFrame* of a .seg file, with the following columns: "chromosome", "start","end", "1/log2_correction_factor".

Return type float

Returns A float indicating the CNV-corrected coverage for the bin.

liquorice.utils.CoverageAndSequenceTablePreparation module

class liquorice.utils.CoverageAndSequenceTablePreparation.**CoverageAndSequenceTablePreparation**(*bam_filepath*, *bins_bed_file*, *refgenome_filepath*, *refgenome_chrom_sizes_file*, *refgenome_mappability_bw_file*, *readlength*, *longest_fragment_length*, *mean_seq_depth*, *n_cores=1*, *min_mapq=None*, *skip_these_seq_ids=None*, ***additional_crpb_*

Bases: object

Object used to set up a *pandas.DataFrame* containing basic information about each bin: Coverage (normalized for overall sequencing depth), genomic sequence & mappability (both incl. surrounding regions; used for bias correction). Usually, a user would want to call `get_complete_table()` on this object, and save the resulting *pandas.DataFrame* for subsequent bias-correction.

Parameters

- **bam_filepath** (str) – Path to the .bam file containing the mapped reads for the sample. Does not need to be duplicate-filtered, this is done by the function that calculates the coverage.
- **bins_bed_filepath** (str) – Path to the .bed file containing chromosome, start, end, and bin nr. of every bin to be analyzed (in that order). Output of `liquorice.utils.RegionFilteringAndBinning.Binning.write_bin_bedfile()` is a suitable input here.
- **refgenome_filepath** (str) – Path to the reference genome .fa(.gz) file. Must have a .fai index in the same dirname.

- **refgenome_chromsizes_filepath** (str) – Path to a tab-delimited file containing the chromosome sizes for the reference genome. The first column must be the chromosome name, the second column its size. Can be the .fa.fai file associated to the reference genome.
- **refgenome_mappability_bigwig_path** (str) – Path to a .bigWig file containing (forward) mappability values scaled between 0 and 1.
- **readlength** (int) – (Average) length of the reads in the .bam file.
- **longest_fraglen** (int) – Length by which sequencing and mappability information is extended beyond a bin's borders. Typically, this should be the longest sampled fragment length.
- **min_mapq** (int) – Minimum mapping quality for a fragment to still be counted when calculating the coverage.
- **n_cores** (int) – Number of cores to be used by deeptool's *CountReadsPerBin*, which is used for coverage calculations. Set to higher values to speed up the process.
- **mean_seq_depth** (float) – A float that quantifies the global coverage/sequencing depth. Coverages per bin will be normalized (i.e. divided) by this value.
- **skip_these_steps** (List[str]) – A list containing the steps which should be skipped. May contain: "coverage","sequence", "mappability". Default []: Don't skip any steps.
- ****additional_crpb_kwargs** – Additional keywords to be used for deeptool's *CountReadsPerBin*. Can be used to override the default settings: *"ignoreDuplicates":True, "samFlag_exclude":256, "extendReads":True*

get_complete_table()

Main method to retrieve a *pandas.DateFrame* that can be used to calculate bias factors on. Calls read_bed', :func:`.get_coverage_per_bin_mean_over_per_basepair_values_chunked`, get_sequence_per_bin(), and get_mappability_per_bin', constructs :attr:`.df`, a *pandas.DataFrame*, from the results, and returns it.

Return type DataFrame

Returns A *pandas.DataFrame* where rows correspond to the genomic bins in bins_bed_filepath. This *DataFrame* has the following columns: "chromosome","start","end","bin nr.","coverage","sequence", "mappability". Entries in "sequence" and "mappability" columns are extended by longest_fraglen (and downstream additionally by readlength for "mappability").

get_coverage_per_bin()

Calculates the normalized coverage per bin, based on bins_bed_filepath and bam_filepath. Uses deeptool's *countReadsPerBin* for calculating the coverage in every bin in df. The obtained coverage value is then divided by mean_seq_depth. This function is faster than *get_coverage_per_bin_mean_over_per_basepair_values_chunked()*, but provides less accurate results: Rather than the mean over the coverage of each base-pair in the bin, the total number of reads mapping to the bin (even partially mapping reads) are reported (after normalization for *:attr:.mean_seq_depth*).

Return type Series

Returns A *pandas.Series* with the coverage per bin, normalized for total sequencing coverage.

get_coverage_per_bin_mean_over_per_basepair_values_chunked()

Calculates the normalized coverage per bin by averaging the per-base coverage over all positions in the bin and then deviding the result by mean_seq_depth. Requires the following columns in df: "chromosome","start","end","bin nr.","region nr.","bin size". Uses deeptool's *countReadsPerBin* for calculating the coverage in every bin in df. The obtained coverage value is then divided by mean_seq_depth.

Return type Series

Returns A *pandas.Series* with the coverage per bin, normalized for total sequencing coverage.

get_mappability_for_bin(*row*, *mapp_bw*)

For a single bin, get a list of (forward) mappability values in and around the bin based on its genomic coordinates and a *pyBigWig.bigWig* file that contains the genome-wide mappability.

Parameters

- **row** (Series) – *pandas.Series* containing columns "chromosome","start", and "end".
- **mapp_bw** (bigWigFile) – *pyBigWig.pyBigWig* object with mappability info (i.e. result of *pyBigWig.open()*)

Return type List[float]

Returns Mappability for the bin at base resolultion. Extended downstream by (longest_fraglen + readlength) and upstream by longest_fraglen.

get_mappability_per_bin()

Opens a *.bigWig* file using *pyBigWig* and calls get_mappability_for_bin() for every bin.

Return type Series

Returns A *pandas.Series* with mappability information per bin. Extended downstream by (longest_fraglen + readlength) and upstream by longest_fraglen.

get_sequence_per_bin()

Get a list of genomic sequences, one per entry in the .bed file attr:*bins_bed_filepath*. Sequences are extended by longest_fraglen

Return type List[str]

Returns A list of the genomic sequences of every bin in the attr:*bins_bed_filepath*.

read_bed()

Reads in a .bed file and return corresponding *pandas.DataFrame*. 4th column in the .bed file is interpreted as bin nr. Layout of the .bed file MUST be ["chromosome","start","end","bin nr."].

Return type DataFrame

Returns A *pandas.DataFrame* with the columns ["chromosome","start","end","bin nr."].

liquorice.utils.CoverageAndSequenceTablePreparation.**parallelize_get_coverage_of_region**(*df*, *n_cores*, *bam_filepath*, *min_mapq*, *additional_crpb_kwargs*)

Splits the provided *pandas.DataFrame* into chunks of the size :param`n_cores`, and runs :func`run_get_coverage_of_region_per_chunk` in parallel on every chunk. From: https://towardsdatascience.com/make-your-own-super-pandas-using-multiproc-1c04f41944a1

Parameters

- **df** – Full *pandas.DataFrame* with the columns "chromosome","region start","region end".
- **n_cores** – Run this many processes in parallel.
- **bam_filepath** – Path to the .bam file containing the mapped reads for the sample. Does not need to be duplicate-filtered, this is done by the function that calculates the coverage.
- **min_mapq** – Minimum mapping quality for a fragment to still be counted when calculating the coverage.

- **additional_crpb_kwargs** – Additional keywords to be used for deeptool's *CountReadsPerBin*. Can be used to override the default settings: *"ignoreDuplicates":True, "samFlag_exclude":256, "extendReads":True*

Returns Full *pandas.DataFrame* with the columns "chromosome","region start","region end" and "region coverage array".

liquorice.utils.CoverageAndSequenceTablePreparation.**run_get_coverage_of_region_per_chunk**(*dataframe_chunk*, *bam_filepath*, *min_mapq*, *n_cores*, *additional_crpb_kwarg*

Uses :func`deeptools.countReadsPerBin.get_coverage_of_region()` to calculate the coverage in every bin in **:param:`dataframe_chunk`**. Adds the result as a column "region coverage array" and returns param:*dataframe_chunk*. This function can receive only picklable arguments, and can therefore be called multiple times in parallel.

Parameters

- **dataframe_chunk** – A *pandas.DataFrame* with the columns "chromosome","region start","region end". Usually, this would be only a chunk of the full DataFrame.
- **bam_filepath** – Path to the .bam file containing the mapped reads for the sample. Does not need to be duplicate-filtered, this is done by the function that calculates the coverage.
- **min_mapq** – Minimum mapping quality for a fragment to still be counted when calculating the coverage.
- **n_cores** – Number of cores used by *deeptools.countReadsPerBin*. Should probably be 1 if the function is called in parallel
- **additional_crpb_kwargs** – Additional keywords to be used for deeptool's *CountReadsPerBin*. Can be used to override the default settings: *"ignoreDuplicates":True, "samFlag_exclude":256, "extendReads":True*

Returns A *pandas.DataFrame* with the columns "chromosome","region start","region end" and "region coverage array".

liquorice.utils.FitGaussians module

class liquorice.utils.FitGaussians.**FitGaussians**(*unfitted_y_values*, *extend_to*, *binsize*, *avg_centersize*, *samplename=None*, *regionset_name=None*)

Bases: `object`

An object that can be used for quantification of coverage drops by fitting gaussian functions and an intercept to the coverage values.

Parameters

- **unfitted_y_values** (`List[float]`) – A pandas *Series* of coverage values, one value per bin (and sorted by increasing bin nr.). A suitable input is the result of `liquorice.utils.AggregateAcrossrRegions.aggregate_across_regions()`.
- **extend_to** (`int`) – *extend_to* setting that was used when calculating the y values. Needed to calculate relative bin positions.
- **binsize** (`int`) – *binsize* setting that was used when calculating the y values. Needed to calculate relative bin positions.

- **samplename** (Optional[str]) – If not *None*, add this in a column "sample" to the result *DataFrame* when calling `fit_gaussian_models()`
- **regionset_name** (Optional[str]) – If not *None*, add this in a column "region-set" to the result *DataFrame* when calling `fit_gaussian_models()`
- **avg_centersize** (float) – Average size (i.e. width in bp) of the regions of interest (before extension by extend_to).

`fit_gaussian_models`(*g1_min_sigma=20*, *g1_max_sigma=200*, *g2_min_sigma=200*, *g2_max_sigma=3000*, *g3_min_sigma=3000*, *g3_max_sigma=40000*, *method='leastsq'*)

Fits three gaussian functions to `unfitted_y_values` (usually aggregated coverage values), based on given limits. Sets `x_values`, `g1_y_values`, `g2_y_values`, `g3_y_values`, `intercept_y_values` and `combined_model_y_values` for the object.

Parameters

- **g1_min_sigma** (Union[int, float]) – Min value for the smallest gaussian
- **g1_max_sigma** (Union[int, float]) – Max value for the smallest gaussian
- **g2_min_sigma** (Union[int, float]) – Min value for the middle gaussian
- **g2_max_sigma** (Union[int, float]) – Max value for the middle gaussian
- **g3_min_sigma** (Union[int, float]) – Min value for the widest gaussian
- **g3_max_sigma** (Union[int, float]) – Max value for the widest gaussian
- **method** (str) – A method for fitting the combined model, used as parameter in lmfit's .fit() function.

Return type DataFrame

Returns A *pandas.DataFrame* containing several summary metrics based on the fitted model: The amplitude and values for each of the three gaussians, the total depth of the fitted model at the central bin, the Bayesian Information Criterion (measures model fit), and the area under the curve of the fitted model, calculated as the area between the fitted intercept and the fitted model. If `samplename` or `regionset_name` are not *None*, columns "sample" or "region-set" are added, respectively.

liquorice.utils.GlobalFragmentSize module

`liquorice.utils.GlobalFragmentSize.getFragmentLength_worker`(*chrom*, *start*, *end*, *bamFile*, *distanceBetweenBins*)

This is a function from deeptools, modified from https://github.com/deeptools/deepTools/blob/ac42d29c298c026aa0c53c9db2553087ebc86b97/deeptools/getFragmentAndReadSize.py#L14 Queries the reads at the given region for the distance between reads and the read length. As opposed to the original version, does not disregard pairs that are flagged as "not properly paired", as this behaviour excludes fragments from the two-nucleosome peak of the cfDNA-distribution.

Parameters

- **chrom** (str) – chromosome name
- **start** (int) – region start
- **end** (int) – region end
- **bamFile** (str) – BAM file name
- **distanceBetweenBins** (int) – The number of bases at the end of each bin to ignore

:return an np.array, where first column is fragment length, the second is for read length

Return type array

liquorice.utils.GlobalFragmentSize.**getFragmentLength_wrapper**(*args*)

liquorice.utils.GlobalFragmentSize.**get_list_of_fragment_lengths_and_avg_readlength**(*bam_filepath*, *n_cores=1*, *n=1000*, *upper_limit=800*)

Sample fragments from the given .bam file to obtain a representative distribution of fragment lenghts.

Parameters

- **bam_filepath** (str) – Path to the .bam file for which fragments should be sampled.
- **n_cores** (int) – Number of cores to use by deeptools.getFragmentAndReadSize.get_read_and_fragment_length().
- **n** (int) – Number of randomly sampled fragment lengths to generate
- **upper_limit** (int) – Fragment lengths exceeding this limit will be excluded. Rarely, fragment size are wrongly inferred and therefore huge. Sampling one of those incorrect lengths would unnecessarily blow up the sequence and mapping information stored for each bin. As a default, 500 is used as a reasonable upper limit of relevant fragment lengths for cfDNA.

Return type Tuple[List[int], int]

Returns A tuple, consisting of a list of the randomly samples fragment lengths, and an integer value of the average read length.

liquorice.utils.GlobalFragmentSize.**get_read_and_fragment_length**(*bamFile*, *return_lengths=False*, *blackListFileName=None*, *binSize=50000*, *distanceBetweenBins=1000000*, *numberOfProcessors=None*, *verbose=False*)

This is a function from deeptools. It was included in LIQUORICE's source code to allow it to (indirectly) call the modified version of :func`getFragmentLength_worker`. Estimates the fragment length and read length through sampling

Parameters

- **bamFile** (str) – BAM file name
- **return_lengths** (bool) –
- **numberOfProcessors** (Optional[int]) –
- **verbose** (bool) –
- **binSize** (int) –
- **distanceBetweenBins** (int) –

.:return A tuple of two dictionaries, one for the fragment length and the other for the read length. The dictionaries summarise the mean, median etc. values

Return type Tuple[dict]

liquorice.utils.MeanSequencingDepth module

liquorice.utils.MeanSequencingDepth.**sample_bam_to_get_sequencing_depth**(*bam_filepath*, *n_sites_to_sample=10000*, *n_cores=1*, *min_mapq=20*, *chromosomes_list=None*, ***additional_crpb_kwargs*)

Estimates the overall sequencing coverage of the .bam file by sampling sites that are regularily distributed across the genome (using `deeptools.countReadsPerBin.CountReadsPerBin()` in a slightly modified version that allows fragments without the SAM flag is_proper_pair.).

Parameters

- **bam_filepath** (`str`) – Path to the .bam file for which mean coverage should be calculated.
- **n_sites_to_sample** (`int`) – Number of sites (length 1) to sample.
- **n_cores** (`int`) – Number of cores to be used by `deeptools.countReadsPerBin.CountReadsPerBin()`.
- **min_mapq** (`int`) – *minMappingQuality* setting for *deeptools.countReadsPerBin.CountReadsPerBin*.
- **chromosomes_list** (`Optional[List[str]]`) – A list of chromosomes that should be analyzed. Regions on chromosomes that are not in this list will be excluded from analysis. Default None means ["chr1", ..., "chr22"].
- **additional_crpb_kwargs** (`dict`) – Use to override the default parameters for `deeptools.countReadsPerBin.CountReadsPerBin()`. If not specified otherwise in *additional_crpb_kwargs*, the following attributes are passed to *CountReadsPerBin*: *ignoreDuplicates=True*, *samFlag_exclude=256*, *extendReads=True*.

Return type `float`

Returns The average coverage determined from the sampled regions.

liquorice.utils.Plotting module

class liquorice.utils.Plotting.**Plotting**(*samplename*, *out_dir*)

Bases: `object`

An object used to generate plots of the aggragated coverage, differences between corrected and uncorrected coverage, as well as correlations between bias-factors and coverage.

Parameters

- **samplename** (`str`) – Name of the sample as it should appear in the plots
- **out_dir** (`str`) – Path to the directory to which plots should be saved.

none_or_list_of_float
alias of `Union[None, List[float]]`

none_or_list_of_int
alias of `Union[None, List[float]]`

none_or_list_of_str
alias of `Union[None, List[str]]`

plot_corrected_vs_uncorrected_coverage(*x_values*, *uncorrected_y_values*, *corrected_y_values*, *uncorrected_y_intercept*, *corrected_y_intercept*, *filename='auto'*, *return_figure=False*)

Plots the aggregated coverages, both corrected and uncorrected.

Parameters

- **x_values** (List[int]) – List of x coordinates - the coordinates of the bins' centers relative to the center of the regions of interest.
- **uncorrected_y_values** (List[float]) – List of aggregated, uncorrected coverage values.
- **corrected_y_values** (List[float]) – List of aggregated, uncorrected coverage values.
- **uncorrected_y_intercept** (float) – Intercept value that should be subtracted from every value in **uncorrected_y_values**
- **corrected_y_intercept** (float) – Intercept value that should be subtracted from every value in **corrected_y_values**
- **filename** (Optional[str]) – Filename to which the figure should be saved. "auto" sets this to " out_dir/corrected_vs_uncorrected_coverage.pdf ". Set to *None* to avoid saving as file. File extension determines in which format the plot is saved.
- **return_figure** (bool) – If True, return a *matplotlib figure* that can be altered, plotted, or saved.

Return type Optional[Figure]

Returns A *matplotlib figure* if **return_figure** is True, nothing otherwise.

Example:

```
plotting.plot_corrected_vs_uncorrected_coverage(
    x_values=fit_gaussians_obj.x_values,
    uncorrected_y_values=fit_gaussians_obj_uncorrected.unfitted_y_values,
    corrected_y_values=fit_gaussians_obj.unfitted_y_values,
    uncorrected_y_intercept=fit_gaussians_obj_uncorrected.intercept_y_values[0],
    corrected_y_intercept=fit_gaussians_obj.intercept_y_values[0])
```

plot_correlations_biasfactors_coverage_bin_nr(*df*, *x_varnames_list=None*, *y_varnames_list=None*, *bins=(50, 50)*, *return_figure=False*, *filename='bias_plots.pdf'*, *percentile_cutoffs=(0.1, 0.99, 0.1, 0.99)*, *fit_regression=True*)

Plot the correlation between coverage and a bias factor as a heatmap, and fit a trendline.

Parameters

- **df** (DataFrame) – *pandas.DataFrame* that contains the variables in **x_varnames_list** and **y_varnames_list**.
- **x_varnames_list** (Optional[List[str]]) – List of the columns that should be plotted as independent variables. Default None uses all columns in the **df** except for ["chromosome","start","end","sequence","mappability"].
- **y_varnames_list** (Optional[List[str]]) – List of the columns that should be plotted as dependent variables. Default None uses "coverage" and "bin nr.", plus "corrected coverage" if this column is present in the **df**.

- **bins** (Tuple[int, int]) – Tuple specifying the number of bins to be used for the x and y axis
- **return_figure** (bool) – If True, return a *matplotlib figure* that can be altered, plotted, or saved.
- **fit_regression** (bool) – If True, fit a qadratic polynomial and plot the regression function and R^2.
- **percentile_cutoffs** (Tuple[int, int, int, int]) – Tuple containing the min and max percentile of values that should be displayed on the x and y axes: (x_min,x_max,y_min,y_max). This setting is ignored for "bin nr.", for this column, all values are shown. Percentiles must be in the interval [0,1].
- **filename** (str) – Filename to which the figure should be saved. File extension determines in which format the plot is saved.

Return type Optional[Figure]

Returns A *matplotlib figure* if **return_figure** is True, nothing otherwise.

plot_coverage_bias_correlation(*df*, *biasfactor_column*, *coverage_column*, *percent=True*, *xmin=0*, *xmax=100*, *ymin=0*, *ymax=2*, *filename='auto'*, *return_figure=False*)

Plot the correlation between coverage and a bias factor as a heatmap, and fit a trendline.

Parameters

- **df** (DataFrame) – *pandas.DataFrame* that contains **biasfactor_column** and **coverage_column**.
- **biasfactor_column** (str) – Name of the column that contains data for the bias-factor of interest
- **coverage_column** (str) – Name of the column that contains the coverage data
- **percent** (bool) – Whether the output plot should multiply the x values by the factor 100
- **xmin** (Union[int, float]) – min x coordinate to show in plot
- **xmax** (Union[int, float]) – max y coordinate to show in plot
- **ymin** (Union[int, float]) – min x coordinate to show in plot
- **ymax** (Union[int, float]) – max y coordinate to show in plot
- **filename** (Optional[str]) – Filename to which the figure should be saved. "auto" sets this to " out_dir/**biasfactor_column**__vs__**coverage_column**.pdf " (with spaces in column names replaced by '_'). Set to *None* to avoid saving as file. File extension determines in which format the plot is saved.
- **return_figure** (bool) – If True, return a *matplotlib figure* that can be altered, plotted, or saved.

Return type Optional[Figure]

Returns A *matplotlib figure* if **return_figure** is True, nothing otherwise.

plot_fitted_model(*fit_gaussians_obj=None*, *x_values=None*, *unfitted_y_values=None*, *g1_y_values=None*, *g2_y_values=None*, *g3_y_values=None*, *intercept_y_values=None*, *combined_model_y_values=None*, *unfitted_y_values_label='Corrected, aggregated coverage'*, *filename='auto'*, *return_figure=False*)

Plots the fitted model, i.e. the raw data and lines for G1, G2, G3, the intercept, and the combined model.

Parameters

- **fit_gaussians_obj** (Optional[*FitGaussians*]) – object of class FitGaussians, on which fit_gaussian_models() has been called. This contains all nessesary information for plotting, so all other parameters except **filename** are ignored if this is set.
- **x_values** (Optional[List[float]]) – List of x coordinates - the coordinates of the bins' centers relative to the center of the region of interest. Required if fit_gaussians_obj is not set, ignored otherwise.
- **unfitted_y_values** (Optional[List[float]]) – List of y coordinates of the unfitted coverage data. Required if fit_gaussians_obj is not set, ignored otherwise.
- **g1_y_values** (Optional[List[float]]) – List of y coordinates for the first, smallest gaussian. Required if fit_gaussians_obj is not set, ignored otherwise.
- **g2_y_values** (Optional[List[float]]) – List of y coordinates for the second, middle gaussian. Required if fit_gaussians_obj is not set, ignored otherwise.
- **g3_y_values** (Optional[List[float]]) – List of y coordinates for the first, largest gaussian. Required if fit_gaussians_obj is not set, ignored otherwise.
- **intercept_y_values** (Optional[List[float]]) – List of y coordinates for the intercept. Required if fit_gaussians_obj is not set, ignored otherwise.
- **combined_model_y_values** (Optional[List[float]]) – List of y coordinates for the fitted, combined model. Required if fit_gaussians_obj is not set, ignored otherwise.
- **unfitted_y_values_label** (Optional[List[float]]) – Label for the unfitted_y_values. By default it is assumed that the corrected, aggregated, unfitted coverage is given.
- **filename** (Optional[str]) – Filename to which the figure should be saved. "auto" sets this to " out_dir/fitted_gaussians.pdf ". Set to *None* to avoid saving as file. File extension determines in which format the plot is saved.
- **return_figure** (bool) – If True, return a *matplotlib figure* that can be altered, plotted, or saved.

Return type Optional[Figure]

Returns A *matplotlib figure* if **return_figure** is True, nothing otherwise.

liquorice.utils.Plotting.**polyfit**(*x*, *y*, *degree*)

fit a trendline to data, see https://stackoverflow.com/questions/893657/how-do-i-calculate-r-squared-using-python-and-numpy

Parameters

- **x** (List[float]) – x values
- **y** (List[float]) – y values
- **degree** (int) – 1 for linear, 2 for quadratic, etc.

Return type dict

Returns A Dictionary, containing lists with the keys 'polynomial' (polynomial coefficients) and 'determination' (The coefficient of determination).

liquorice.utils.RegionFilteringAndBinning module

class liquorice.utils.RegionFilteringAndBinning.**RegionFilteringAndBinning**(*bed_filepath*, *binsize*, *extend_to*, *refgenome_chromsizes_filepath*, *chromosomes_list=['chr1', 'chr2', 'chr3', 'chr4', 'chr5', 'chr6', 'chr7', 'chr8', 'chr9', 'chr10', 'chr11', 'chr12', 'chr13', 'chr14', 'chr15', 'chr16', 'chr17', 'chr18', 'chr19', 'chr20', 'chr21', 'chr22']*, *blacklist_bed_filepath=None*)

Bases: `object`

A Binning object, which can be used to create a .bed file with the coordinates and bin numbers of every bin. The method `write_bin_bedfile()` of this object can be used to create output.

Parameters

- **bed_filepath** (`str`) – path to a .bed file containing regions that should be extended and split into bins. This could be e.g. a list of DNase I hypersensitivity sites or enhancer peaks.
- **binsize** (`int`) – Size of the bins. Use higher values to reduce noise, lower values to increase spatial resolution.
- **extend_to** (`int`) – The regions will be extended by this value in both directions. Outmost bins will have their center at <*center of the region*>*+-*<*extend_to*>.
- **refgenome_chromsizes_filepath** (`str`) – Path to a tab-delimited file containing the chromosome sizes for the reference genome. The first column must be the chromosome name, the second column its size. Can be the .fa.fai file associated to the reference genome. Must include all chromosomes given in chromosomes_list.
- **chromosomes_list** (`List[str]`) – A list of chromosomes that should be analyzed. Regions on chromosomes that are not in this list will be excluded from analysis. Default is ["chr1", ..., "chr22"].
- **blacklist_bed_filepath** (`Optional[str]`) – Optional: .bed file of a black list, such as the one from `here <https://github.com/Boyle-Lab/Blacklist/blob/master/lists/hg38-blacklist.v2.bed.gz> `_ for hg38 (unzip first). Regions that overlap any of the regions in this blacklist after extension by **extend_to** will be excluded from further analysis.

`filter_regions`(*df*)

Remove regions that do not pass filters.

Parameters **df** (`DataFrame`) – A *pandas.DataFrame* of a .bed file (with columns "chromosome", "start", "end")

Return type `DataFrame`

Returns Filtered *DataFrame*, with regions excluded if (after extension by `extend_by`) they : i) are not on standard chromosomes (`chromosomes_list`), ii) fall within

blacklist_bed_filepath (if defined), iii) extend beyond chromosome ends, or iv) if their start coordinate is larger than their end coordinate.

get_binstarts(*center*)

Gets the bin start coordinates for a given center

Parameters **center** (int) – Coordinate of the center of the region of interest

Return type List[int]

Returns A list with start coordinates, one per bin. Length / number of created bins depends on extend_to and binsize.

get_binstarts_percentile_split_central_roi(*start*, *end*)

Gets the bin start coordinates for a given core region, splitting the core region into 5 bins with a length of 10,15,50,15,and 10 % of the core region length, respectively. The other bins, outside the core region, have a length of binsize'. The most upstream bin starts :attr:.extend_to` bp upstream of the core region start, and the most downstream bin ends extend_to bp downstream of the core region end.

Parameters

- **start** – Coordinate of the start of the region of interest
- **end** – Coordinate of the end of the region of interest

Returns A list with start coordinates, one per bin. Length / number of created bins depends on extend_to and binsize.

is_within_chromsome_borders(*chrom*, *start*, *end*, *chromsize_dict*)

Check if a region is within its chromosome's borders.

Parameters

- **chrom** (str) – Chromosome name
- **start** (int) – Start coordinate
- **end** (int) – End coordinate
- **chromsize_dict** (Dict[str, int]) – Dictionary with chromosome names as keys and lengths as values. MUST contain **chrom** as a key, otherwise *sys.exit()* is called.

Return type bool

Returns True if the region is within the borders of the chromosome, False otherwise

write_bin_bedfile(*out_bedfile_path_bins*, *out_bedfile_path_regions_that_passed_filter*)

Splits every region in the bins_bed_filepath file into bins, such that the central bin's center is at the center of the region, and such that the outermost bins each extend binsize/2 over the edges of (<*the region's center*> - extend_to) or (<*the region's center*> + extend_to), respectively. Also assigns a bin nr. to every bin, starting with the most downstream bin.

Parameters

- **out_bedfile_path_bins** (str) – Path to which the output .bed file with bin coordinates should be written to.
- **out_bedfile_path_regions_that_passed_filter** (Optional[str]) – If not *None*, write the regions that passed all filters and that are the basis of the bins to this path (should end with .bed).

Return type None

write_bin_bedfile_percentile_split_central_roi(*out_bedfile_path_bins*, *out_bedfile_path_regions_that_passed_filter*)

Splits every region in the `bins_bed_filepath` file into bins. The core region (given in `bed_filepath`) is splitted into 5 bins with a length of 10,15,50,15,and 10 % of the core region length, respectively. The other bins, outside the core region, have a length of `binsize'. The most upstream bin starts :attr:`.extend_to` bp upstream of the core region start, and the most downstream bin ends `extend_to` bp downstream of the core region end. Also assigns a bin nr. to every bin, starting with the most downstream bin. :type out_bedfile_path_bins: `str` :param out_bedfile_path_bins: Path to which the output .bed file with bin coordinates should be written to. :type out_bedfile_path_regions_that_passed_filter: `Optional[str]` :param out_bedfile_path_regions_that_passed_filter: If not *None*, write the regions that passed all filters

and that are the basis of the bins to this path (should end with .bed).

Return type `None`

liquorice.utils.Workflows module

liquorice.utils.Workflows.**add_biasfactors_percentile_split**(*avg_readlength*, *liq_table*, *n_cores*, *sampled_fragment_lengths*, *skip_these_biasfactors=None*)

Returns a table with added bias-factors, taking into account different bin sizes. Also adds a "bin size" column.

Parameters

- **avg_readlength** – Average length of reads in the .bam file.
- **liq_table** – pandas.DataFrame` with one row per bin, containing columns "chromosome", "start", "end", "bin nr.", "coverage", "sequence", and "mappability". Suitable input is the output of the `get_complete_table()` method of the *liquorice.utils.CoverageAndSequenceTablePreparation* class object.
- **n_cores** – Max number of cores to use for calculations.
- **sampled_fragment_lengths** – A list containing fragment lengths that are representative of the sample's global fragment size distribution. Typically a few hundred fragments will suffice here.
- **skip_these_biasfactors** – Do not calculate these bias factors. Only these entries are allowed: ["di and trinucleotides and GC content","mappability", "di and trinucleotides"]

Returns A *pandas.DataFrame* with added bias-factors, taking into account different bin sizes, and a "bin size" column.

liquorice.utils.Workflows.**run_liquorice_on_regionset_with_pretrained_biasmodel**(*samplename*, *regionset_name*, *bam_filepath*, *bed_filepath*, *biasmodel_path*, *refgenome_filepath*, *refgenome_chromsizes_filepath*, *refgenome_mappability_bigwig_p* *blacklist_bed_filepath*, *sampled_fragment_lengths*, *avg_readlength*, *cna_seg_filepath*, *mean_seq_depth*, *n_cores*, *binsize=500*, *extend_to=20000*, *use_default_fixed_sigma_values* *save_biasfactor_table=False*, *save_corrected_coverage_table* *no_chr_prefix=False*, *percentile_split_core_rois=False*, *use_this_roi_biasfactortable=N*

Run the complete LIQUORICE workflow on a given region-set, using a pre-trained bias model. Main steps of this workflow include: Filtering regions in the input .bed and splitting remaining regions into bins; calculating sequence, coverage, and mappability for every bin; calculating bias factors for every bin, using the pre-trained model and the inferred bias-factors to correct the coverage; aggregating information across regions and fitting gaussian functions and an intercept to the corrected, aggregated coverage data. Also creates plots and result tables.

Parameters

- **samplename** (str) – Name of the sample (to be used in plots and output tables).
- **regionset_name** (str) – Name of the region-set (to be used in plots and output tables).
- **bam_filepath** (str) – Path to the .bam file containing the mapped reads for the sample. Does not need to be duplicate-filtered, this is done by the function that calculates the coverage.
- **bed_filepath** (str) – path to a .bed file containing regions-of-interest that should be extended and split into bins. This could be e.g. a list of DNase I hypersensitivity sites or enhancer peaks.
- **biasmodel_path** (str) – Path to a trained biasmodel.
- **refgenome_filepath** (str) – Path to the reference genome .fa(.gz) file. Must have a .fai index in the same dirname.
- **refgenome_chromsizes_filepath** (str) – Path to a tab-delimited file containing the

chromosome sizes for the reference genome. The first column must be the chromosome name, the second column its size. Can be the .fa.fai file associated to the reference genome.

- **refgenome_mappability_bigwig_path** (str) – Path to a .bigWig file containing (forward) mappability values scaled between 0 and 1.
- **blacklist_bed_filepath** (Optional[str]) – .bed file of a black list, such as the one from `here <https://github.com/Boyle-Lab/Blacklist/blob/master/lists/hg38-blacklist.v2.bed.gz> `_ for hg38 (unzip first). Regions that overlap any of the regions in this blacklist after extension by **extend_to** will be excluded from further analysis. Set to None to skip this step.
- **sampled_fragment_lengths** (List[int]) – A list containing fragment lengths that are representative of the sample's global fragment size distribution. Typically a few hundred fragments will suffice here.
- **avg_readlength** (int) – (Average) length of the reads in the .bam file.
- **cna_seg_filepath** (Optional[str]) – If specified, use this .seg file to correct the coverage by the values specified in this file prior to correcting coverage with the bias model. Use this if you want to normalize out the effects of copy number aberrations (CNAs) on the coverage. File must be tab-separated, with column names as first line. The second, third and fourth column must be chromosome, start, and end of the segment, and the last column must be the log2-ratio of observed/expected read depth.
- **mean_seq_depth** (float) – A float that quantifies the global coverage/sequencing depth. Coverages per bin will be normalized (i.e. divided) by this value.
- **binsize** (int) – Size of the bins. Use higher values to reduce noise, lower values to increase spatial resolution. Using the same binsize for the biasmodel generation as for the region-set-of-interest is probably preferable.
- **extend_to** (int) – The regions will be extended by this value in both directions. Outmost bins will have their center at * <center of the region> * +- * <extend_to> *.
- **use_default_fixed_sigma_values** (bool) – If *True*, use the following contraints for the sigma values of the small, medium, and large Gaussian, repectively: 149-150 bp, 757-758 bp, and 6078-6079 bp.
- **n_cores** (int) – Maximum number of cores to use during steps that allow multiprocessing/multithreading.
- **save_biasfactor_table** (bool) – If *True*, save a table of bin coordinates, bin number, coverage, corrected coverage and biasfactor values under "coverage_and_biasfactors_per_bin.csv".
- **save_corrected_coverage_table** (bool) – If *True*, save a table of bin coordinates, bin number, coverage, and corrected coverage under "coverage_per_bin.csv".
- **no_chr_prefix** (bool) – If True, set the list of allowed chromosomes to [str(i) for i in range(1,23)] instead of ["chr"+str(i) for i in range(1,23)]
- **percentile_split_core_rois** (bool) – If set, split the central region into 5 bins of variable size instead of always using a fixed binsize. *extend_to* should not be set to 0 if this is used.

Return type None

liquorice.utils.Workflows.**run_liquorice_train_biasmodel_on_same_regions**(*samplename*, *regionset_name*, *bam_filepath*, *bed_filepath*, *refgenome_filepath*, *refgenome_chromsizes_filepath*, *refgenome_mappability_bigwig_path*, *blacklist_bed_filepath*, *sampled_fragment_lengths*, *avg_readlength*, *cna_seg_filepath*, *mean_seq_depth*, *n_cores*, *binsize=500*, *extend_to=20000*, *biasmodel_output_path='trained_biasmodel.*, *nr_of_bins_for_training_and_testing=10*, *skip_central_n_bins_for_training=0*, *save_training_table=False*, *use_default_fixed_sigma_values=True*, *save_biasfactor_table=False*, *save_corrected_coverage_table=False*, *no_chr_prefix=False*, *percentile_split_core_rois=False*, *use_cross_validated_predictions=False*, *use_this_roi_biasfactortable=None*, *speed_mode=False*)

Run the complete LIQUORICE workflow on a given region-set, and train the bias-model on data from the same region- set. Main steps of this workflow include: Filtering regions in the input .bed and splitting remaining regions into bins; calculating sequence, coverage, and mappability for every bin; calculating bias factors for every bin, training a bias model, using the trained model and the inferred bias-factors to correct the coverage; aggregating information across regions and fitting gaussian functions and an intercept to the corrected, aggregated coverage data. Also creates plots and result tables.

Parameters

- **samplename** (str) – Name of the sample (to be used in plots and output tables).
- **regionset_name** (str) – Name of the region-set (to be used in plots and output tables).
- **bam_filepath** (str) – Path to the .bam file containing the mapped reads for the sample. Does not need to be duplicate-filtered, this is done by the function that calculates the coverage.
- **bed_filepath** (str) – path to a .bed file containing regions-of-interest that should be extended and split into bins. This could be e.g. a list of DNase I hypersensitivity sites or enhancer peaks.
- **refgenome_filepath** (str) – Path to the reference genome .fa(.gz) file. Must have a .fai index in the same dirname.
- **refgenome_chromsizes_filepath** (str) – Path to a tab-delimited file containing the chromosome sizes for the reference genome. The first column must be the chromosome

name, the second column its size. Can be the .fa.fai file associated to the reference genome.

- **refgenome_mappability_bigwig_path** (str) – Path to a .bigWig file containing (forward) mappability values scaled between 0 and 1.
- **blacklist_bed_filepath** (Optional[str]) – .bed file of a black list, such as the one from `here <https://github.com/Boyle-Lab/Blacklist/blob/master/lists/hg38-blacklist.v2.bed.gz> `_ for hg38 (unzip first). Regions that overlap any of the regions in this blacklist after extension by **extend_to** will be excluded from further analysis. Set to None to skip this step.
- **sampled_fragment_lengths** (List[int]) – A list containing fragment lengths that are representative of the sample's global fragment size distribution. Typically a few hundred fragments will suffice here.
- **avg_readlength** (int) – (Average) length of the reads in the .bam file.
- **cna_seg_filepath** (Optional[str]) – If specified, use this .seg file to correct the coverage by the values specified in this file prior to correcting coverage with the bias model. Use this if you want to normalize out the effects of copy number aberrations (CNAs) on the coverage. File must be tab-separated, with column names as first line. The second, third and fourth column must be chromosome, start, and end of the segment, and the last column must be the log2-ratio of observed/expected read depth.
- **mean_seq_depth** (float) – A float that quantifies the global coverage/sequencing depth. Coverages per bin will be normalized (i.e. divided) by this value.
- **binsize** (int) – Size of the bins. Use higher values to reduce noise, lower values to increase spatial resolution. Using the same binsize for the biasmodel generation as for the region-set-of-interest is probably preferable.
- **extend_to** (int) – The regions will be extended by this value in both directions. Outmost bins will have their center at * <center of the region> * +- * <extend_to> *.
- **use_default_fixed_sigma_values** (bool) – If *True*, use the following contraints for the sigma values of the small, medium, and large Gaussian, repectively: 149-150 bp, 757-758 bp, and 6078-6079 bp.
- **n_cores** (int) – Maximum number of cores to use during steps that allow multiprocessing/multithreading.
- **save_biasfactor_table** (bool) – If *True*, save a table of bin coordinates, bin number, coverage, corrected coverage and biasfactor values under "coverage_and_biasfactors_per_bin.csv".
- **save_corrected_coverage_table** (bool) – If *True*, save a table of bin coordinates, bin number, coverage, and corrected coverage under "coverage_per_bin.csv".
- **no_chr_prefix** (bool) – If True, set the list of allowed chromosomes to [str(i) for i in range(1,23)] instead of ["chr"+str(i) for i in range(1,23)]
- **biasmodel_output_path** (str) – Path to which the trained biasmodel should be saved to. Must have a .joblib extension.
- **nr_of_bins_for_training_and_testing** (Optional[int]) – Subset the training_df to this many bins. Can speed up the computation time of the model training, but using too few bins will make the model less precise. To speed up computations, we would recommend decreasing the number of regions in the .bed file rather than altering this parameter, as this is more efficient.
- **save_training_table** (bool) – If *True*, save the table that was used to train the biasmodel (coverage and biasfactors per bin) as "training_table.csv".

- **skip_central_n_bins_for_training** (int) – The n most central bins will not be used for training the bias model.
- **no_chr_prefix** – If True, set the list of allowed chromosomes to [str(i) for i in range(1,23)] instead of ["chr"+str(i) for i in range(1,23)]
- **percentile_split_core_rois** (bool) – If set, split the central region into 5 bins of variable size instead of always using a fixed binsize. *extend_to* should not be set to 0 if this is used.
- **use_cross_validated_predictions** (bool) – Instead of training on the full dataset, train twice on half of the dataset and predict the other half. Ignores nr_of_bins_for_training_and_testing
- **use_this_roi_biasfactortable** (Optional[str]) – If set use the specified biasfactor table and only train/apply the biasmodel.
- **speed_mode** (bool) – Only perform GC correction, don't correct using mappability or di/trinucleotides. Setting this flag makes LIQUORICE considerably faster, but may lead to less accurate results.

Return type None

liquorice.utils.Workflows.**train_biasmodel_for_sample**(*samplename*, *bam_filepath*, *bed_filepath*, *refgenome_filepath*, *refgenome_chromsizes_filepath*, *refgenome_mappability_bigwig_path*, *blacklist_bed_filepath*, *sampled_fragment_lengths*, *avg_readlength*, *mean_seq_depth*, *cna_seg_filepath*, *n_cores*, *extend_to=0*, *binsize=500*, *biasmodel_output_path='trained_biasmodel.joblib'*, *nr_of_bins_for_training_and_testing=None*, *save_training_table=False*, *no_chr_prefix=False*, *percentile_split_core_rois=False*)

Go through all steps of LIQUORICE up to the biasmodel training. The resulting biasmodel under **biasmodel_output_path** can then be used when LIQUORICE is run for region-sets of interest.

Parameters

- **samplename** (str) – Name of the sample (to be used in plots and output tables).
- **bam_filepath** (str) – Path to the .bam file containing the mapped reads for the sample. Does not need to be duplicate-filtered, this is done by the function that calculates the coverage.
- **bed_filepath** (str) – path to a .bed file containing regions that should be used to build the biasmodel. A .bed file that contains many random regions across the genome is recommended.
- **refgenome_filepath** (str) – Path to the reference genome .fa(.gz) file. Must have a .fai index in the same dirname.
- **refgenome_chromsizes_filepath** (str) – Path to a tab-delimited file containing the chromosome sizes for the reference genome. The first column must be the chromosome name, the second column its size. Can be the .fa.fai file associated to the reference genome.
- **refgenome_mappability_bigwig_path** (str) – Path to a .bigWig file containing (forward) mappability values scaled between 0 and 1.

- **blacklist_bed_filepath** (Optional[str]) – .bed file of a black list, such as the one from `here <https://github.com/Boyle-Lab/Blacklist/blob/master/lists/hg38-blacklist.v2.bed.gz> `_ for hg38 (unzip first). Regions that overlap any of the regions in this blacklist after extension by **extend_to** will be excluded from further analysis. Set to None to skip this step.
- **sampled_fragment_lengths** (List[int]) – A list containing fragment lengths that are representative of the sample's global fragment size distribution. Typically a few hundred fragments will suffice here.
- **avg_readlength** (int) – (Average) length of the reads in the .bam file.
- **mean_seq_depth** (float) – A float that quantifies the global coverage/sequencing depth. Coverages per bin will be normalized (i.e. divided) by this value.
- **cna_seg_filepath** (Optional[str]) – If specified, use this .seg file to correct the coverage by the values specified in this file prior to model training. Use this if you want to normalize out the effects of copy number aberrations (CNAs) on the coverage. File must be tab-separated, with column names as first line. The second, third and fourth column must be chromosome, start, and end of the segment, and the last column must be the log2-ratio of observed/expected read depth.
- **binsize** (int) – Size of the bins. Use higher values to reduce noise, lower values to increase spatial resolution. Using the same binsize for the biasmodel generation as for the region-set-of-interest is probably preferable.
- **extend_to** (int) – The regions will be extended by this value in both directions. Outmost bins will have their center at <*center of the region*>*+-*<*extend_to*>. Here, the default is 0, meaning only a single bin will be generated per region in the .bed file. This speeds up the computation - for more precise biasmodels, we recommend running for a larger set of regions rather than increasing the **extend_to** parameter.
- **n_cores** (int) – Maximum number of cores to use during steps that allow multiprocessing/multithreading.
- **biasmodel_output_path** (str) – Path to which the trained biasmodel should be saved to. Must have a .joblib extension.
- **nr_of_bins_for_training_and_testing** (Optional[int]) – Subset the training_df to this many bins. Can speed up the computation time of the model training, but using too few bins will make the model less precise. To speed up computations, we would recommend decreasing the number of regions in the .bed file rather than altering this parameter, as this is more efficient.
- **save_training_table** (bool) – If *True*, save the table that was used to train the biasmodel (coverage and biasfactors per bin) as "training_table.csv".
- **no_chr_prefix** (bool) – If True, set the list of allowed chromosomes to [str(i) for i in range(1,23)] instead of ["chr"+str(i) for i in range(1,23)]
- **percentile_split_core_rois** (bool) – If set, split the central region into 5 bins of variable size instead of always using a fixed binsize. *extend_to* should not be set to 0 if this is used.

Return type None

1.5 The LIQUORICE command-line-tool

1.5.1 Arguments

LIQUORICE: A tool for bias correction and quantification of changes in coverage around regions of interest in cfDNA WGS datasets. Documentation: https://liquorice.readthedocs.io; Publication: https://doi.org/10.1093/bioadv/vbac017.

```
usage: LIQUORICE [-h] --bamfile BAMFILE --refgenome_fasta REFGENOME_FASTA
                 --mappability_bigwig MAPPABILITY_BIGWIG
                 [--bedpathlist BEDPATHLIST [BEDPATHLIST ...]]
                 [--bedpath_biasmodel BEDPATH_BIASMODEL] [--binsize BINSIZE]
                 [--extend_to EXTEND_TO] [--blacklist BLACKLIST]
                 [--cna_seg_file CNA_SEG_FILE] [--detect_existing_biasmodel]
                 [--use_this_biasmodel USE_THIS_BIASMODEL]
                 [--extend_to_biasmodel EXTEND_TO_BIASMODEL] [--no_chr_prefix]
                 [--use_this_roi_biasfactortable USE_THIS_ROI_BIASFACTORTABLE]
                 [--speed_mode] [--all_bins_same_size]
                 [--dont_crossvalidate_if_train_on_rois] [--n_cpus N_CPUS]
                 [--tmpdir TMPDIR] [--samplename SAMPLENAME] [--quiet]
                 [--save_training_table] [--save_biasfactor_table]
                 [--save_corrected_coverage_table]
```

Required named arguments

--bamfile .bam file containing the mapped reads of the sample. Used to infer coverage, fragment size, and read length.

--refgenome_fasta Path to a .fa file of the reference genome. Must have a .fa.fai index in the same directory.

--mappability_bigwig Path to a bigWig file that contains (forward) mappability values for every base in the reference genome. Can be calculated with gem-mappability for the appropriate read length.

Optional named arguments - General settings

--bedpathlist List of paths to BED files, one for each region-set of interest. If unspecified, only the biasmodel will be trained (if indicated by the –bedpath_biasmodel, –detect_exisiting_biasmodel, and –use_this_biasmodel settings).

Default: []

--bedpath_biasmodel .bed file containing regions that are used for generating the bias model for the sample. E.g. random regions should work well. Incompatible with use_provided_biasmodel. If '10k_random' is specified, a set of 10k random regions for hg38 shipped with the package is used for training unless an existing biasmodel can be used. If not specified / None (default), and if –use_this_biasmodel is also not specified, train a seperate biasmodel for each region-set that is specified in –bedpathlist, using the flanking regions (+- extend_to) for each region in the set.

--binsize Bin size is important for the resolution of the output plots & data, and for the bias model itself. Smaller bin sizes give higher resolution, but take longer to calculate and may result in more noise

Default: 500

--extend_to Size of the flanking region, in bp. Must be devidable by –binsize and must be a multiple of 2. The regions will be extended by this value in both directions. The most upstream bin starts <extend_to> bp upstream of the core region start, and the most downstream bin ends <extend_to> bp downstream of the core region end. If –all_bins_same_size is set, insteadthe outmost bins will have their center at <center of the region>+-<extend_to>.

Default: 20000

--blacklist Exclude regions if they overlap with the regions in this .bed file (after extension by –extend_to). Default: None. Set to 'hg38' to use the Boyle lab's hg38-blacklist.v2 that is shipped with LIQUORICE.

--cna_seg_file If specified, use this .seg file to correct the coverage by the values specified in this file prior to model training or bias correction. Use this if you want to normalize out the effects of copy number aberrations (CNAs) on the coverage. File must be tab-separated, with column names as first line. The second,third,and fourth column must be chromosome, start, and end of the segment, and the last column must be the log2-ratio of observed / expected read depth. This file can be generated e.g. by running ichorCNA on the sample first.

--detect_existing_biasmodel Check if a bias-model has already been built and saved under ./<samplename>/biasmodel/trained_biasmodel.joblib. If so, use it, otherwise build one using –on a new biasmodel, overwriting files with the same name.

Default: False

--use_this_biasmodel Use this bias model instead of training a model. IMPORTANT: This model has to come from the same sample/patient as the current one, otherwise the bias correction makes no sense.

--extend_to_biasmodel Ignored unless –bedpath_biasmodel is set. Size of the flanking region, in bp, to be used for the bias-model. Must be devidable by –binsize and must be a multiple of 2. The regions will be extended by this value in both directions. Outmost bins will have their center at <center of the region>+-<extend_to>. Default 0: Only place a single bin at the center of each provided region to speed up the training process.

Default: 0

--no_chr_prefix Specify this if the reference genome your .bam files are aligned to uses a chromosome naming scheme such as "1,2,3,..,X,Y" instead of "chr1,chr2,chr3,..,chrX,chrY", which is the default. Note that if your chromosomes are not named like the default, you must not use the "10k_random" setting for –bedpath_biasmodel or the "hg38" setting for –blacklist. Also, all other input files (refgenome_fasta, mappability_bigwig, bedpathlist, and cna_seg_file must follow the same notation.

Default: False

--use_this_roi_biasfactortable If set, use the specified biasfactor table and only train/apply the bias-model, skipping the calculation of coverage and bias factors.

--speed_mode Only perform GC correction, don't correct using mappability or di/trinucleotides. Setting this flag makes LIQUORICE considerably faster, but may lead to less

accurate results. Currently respected only if –bedpath_biasmodel is not specified.

Default: False

--all_bins_same_size If set, use always the same bin size (for both the core region provided with –bedpath_list and the flanking regions defined by –extend_to), instead of splitting the core region into bins with sizes corresponding to 10,15,25,15,and 10% of the core region's length.

Default: False

--dont_crossvalidate_if_train_on_rois Unless set, if –train_on_rois is specified, train two seperate bias models, each on half of the dataset, and use the trainedmodel to predict the other half.

Default: False

Optional named arguments - Technical settings

--n_cpus Number of processors to be used whereever multiprocessing/multithreading is used.

Default: 1

--tmpdir Use this directory as a temporary directory. Default None: search environment variables $TMPDIR,$TEMP,$TMP, and paths /tmp,/var/tmp and /usr/tmp, as well as the current working directory (in this order) until a suitable directory is found.

Optional named arguments - Output settings

--samplename Name of the sample that is being processed. This will be used for output plots and the names of directories. Default None: Infer from –bamfile by removing .bam extension

--quiet If set, the log level is set to "warning", making LIQUORICE less chatty.

Default: False

--save_training_table If set, save the training DataFrame of the bias model under ./<samplename>/biasmodel/training_table.csv (or ./<samplename>/<region-set name>/training_table.csv if –bedpath_biasmodel is not specified)

Default: False

--save_biasfactor_table If set, for each region-set, save a table of bin coordinates, bin number, coverage, corrected coverage and biasfactor values per bin under ./<samplename>/<region-set name>/coverage_and_biasfactors_per_bin.csv. (Filesize can get quite large)

Default: False

--save_corrected_coverage_table If set, for each region-set, save a table of bin coordinates, bin number, coverage, and corrected coverage per bin under ./<samplename>/<region-set name>/coverage_per_bin.csv

Default: False

1.5.2 LIQUORICE's output

LIQUORICE creates a folder named after the samplename in the current working directory, and places all its output there. For every region-set, a subfolder is created. Within these folders, the following files are generated by default:

- *bins.bed*: Genomic coordinates of all bins that passed the filtering and were used to generate the result. The fourth column contains the bin number (0 = most upstream bin of the region).
- *regions.bed*: Genomic coordinates of all regions that passed the filtering and were used to generate the bins.
- *corrected_coverage_mean_per_bin.csv*: The bin-wise, aggregated, bias-corrected coverage values.
- *uncorrected_coverage_mean_per_bin.csv*: The bin-wise, aggregated, un-corrected coverage values.
- *corrected_vs_uncorrected_coverage.pdf*: A plot showing the aggregated coverage profiles, comparing corrected and uncorrected values.
- *fitted_gaussians.pdf*: A plot showing the fitted model, its seperate elements, and the bin-wise, aggregated, bias-corrected coverage values.
- *fitted_gaussians_parameter_summary.csv*: A table summarizing the parameters of the fitted model. This includes the dip area and depth, the two main metrics that quantify the epigenetic signal.
- *GC_content__vs__corrected_coverage.pdf*: A heatmap-style correlation plot, showing the correlation between the GC content and the corrected coverage value in each bin.
- *GC_content__vs__coverage.pdf*: A heatmap-style correlation plot, showing the correlation between the GC content and the uncorrected coverage value in each bin.

1.6 LIQUORICE's summary tool

Useful for generating summaries across samples for multiple region-sets:

1.6.1 Arguments

Summarize output of LIQUORICE for multiple samples and regions of interests in a table and plots. Please make sure that for for any given region-set LIQUORICE has been run with consistent 'binsize' and 'extend_to' settings across samples.)

```
usage: LIQUORICE_summary [-h] [--dirname DIRNAME]
                         [--control_name_list CONTROL_NAME_LIST [CONTROL_NAME_LIST ...]]
                         [--binsize BINSIZE] [--extend_to EXTEND_TO]
                         [--smooth_sigma SMOOTH_SIGMA]
                         [--use_uncorrected_coverage]
                         [--these_regionsets_only THESE_REGIONSETS_ONLY [THESE_
↪REGIONSETS_ONLY ...]]
                         [--prediction_interval_alpha PREDICTION_INTERVAL_ALPHA]
```

Optional named arguments

--dirname Directory in which LIQUORICE's output is contained. Can contain output of multiple samples.

Default: "."

--control_name_list List of samples that serve as reference control samples. Used toinfer z-scores. Please do not surround the list with quotation marks: Example: –control_name_list sample1 sample2 would be correct, –control_name_list "sample1 sample2" would be incorrect.

Default: []

--binsize –binsize setting that was used for LIQUORICE. Default: infer automatically

--extend_to –extend_to that was used for LIQUORICE. Default: infer automatically

--smooth_sigma Determines how strong the coverage drops should be smoothed for the overlay plot. Set to 0 for no smoothing.

Default: 2.0

--use_uncorrected_coverage Per default the corrected coverages will be plotted. Set if you want to plot/summarize the uncorrected coverage instead.

Default: False

--these_regionsets_only List of region sets for which a summary should be calculated. Default: summarize all detected region-sets

Default: False

--prediction_interval_alpha Alpha level for the prediction interval. Samples are deemed significantly different from the controls if their score lies outside the prediction interval of the control group. Note: Significance testing assumes a normal distribution of the scores of the control group. If tests for normality fail, assessment of significant differences is unavailable. Default 0.05: 95 percent prediction interval.

Default: 0.05

1.6.2 Output of LIQUORICE's summary tool

In the current working directory, *LIQUORICE_summary* generates a file *"summary_across_samples_and_ROIs.csv"*, in which the individual *fitted_gaussians_parameter_summary.csv* files are summarized. If *--control_name_list* is specified, this also includes information on the significance of difference between a given case sample and the control samples. Two violinplots, *"boxplot_summary_by_dip_<comparison_metric=[area,depth]>.pdf"* are generated, in which the scores across groups of samples (if defined) and region-sets are summarized. Additionally, for each region-set, the following files are generated:

- *<regionset>_summary_plot_overlay_of_samples_by_Dip_<comparison_metric=[area,depth]>.pdf*: Two side-by-side overlay plots (separate for control and case samples), showing the aggregated, bias-corrected coverage profile. Lines are colored by significance of differences between a given case sample and the control samples.
- *<regionset>_summary_plot_subplots_of_samples_by_Dip_<comparison_metric=[area,depth]>.pdf*: A pdf document (multi-page if necessary) in which the aggregated, bias-corrected coverage profile is shown as a seperate plot for every sample. Lines are colored by significance of differences between a given case sample and the control samples.

- *<regionset>_control_distribution_dip_<comparison_metric=[area,depth]>.pdf*: Shows the distribution of scores of the control samples as histograms and probability plots. This can help assess whether the control samples follow a normal distribution.

If the option *--use_uncorrected_coverage* is chosen, all output files will have the suffix "_using_uncorrected_coverage.<[pdf,csv]>".

1.6.3 Assessment of significant differences

"Case" samples are classified as "significantly different" from the control group if they have a score (dip area or dip depth) that is outside of the prediction interval of the control group.

Here, the prediction interval is an estimate of an interval in which a future observation of a control sample will fall, with a certain probability (default: 95%), given the control samples that have already been observed. The prediction interval depends on the mean, standard deviation and number of samples of the control group - the smaller the standard deviation and the higher the sample number, the narrower the prediction interval will be (and therefore, the better significantly different "case" samples can be detected).

The calculation assumes that the control samples's scores follow a normal distribution. *LIQUORICE_summary* performs the Shapiro-Wilk test for normal distribution: If this test detects significant deviations from a normal distribution, LIQUORICE_summary will display a warning. Note that test for normality may fail to detect relevant deviations from normal distributions when the sample size (number of control samples) is low. So, generally you should treat the results of the significance testing with care if you have only very few samples in the control group. You can also assess the distribution of control scores by looking at the plots named *<regionset>_control_distribution_dip_<comparison_metric=[area,depth]>.pdf*. If the histogram looks very much unlike a normal distribution, and if the data points in the probability plot strongly/systematically deviate from the reference line, the results of *LIQUORICE_summary*'s significance testing may be invalid.

1.7 Usage, Parameters, and Examples

1.7.1 Advice on parameter settings

Apart from the input data, the parameters with the strongest influence on *LIQUORICE*'s results and runtime are:

- *binsize* (speed, memory & outcome)
- *extend_to* (speed, memory & outcome)
- *n_cpus* (speed, memory)
- *bedpath_biasmodel* (speed, memory & outcome)
- *all_bins_same_size* (outcome)
- *speed_mode* (speed, memory & outcome)

A few words of advice on how to set these parameters properly:

binsize and extend_to

We haven chosen a default *binsize* of 500bp, and a default *extend_to* setting of 20kb, because these settings have worked best for us - for the epigenetic signatures we have studied so far. The optimal settings may be different for your application. If you want to analyze signals that are very wide, and you observe that the coverage profile is not fully flat at the edges of the plot (say, if the profile is not flat in the outermost 5kb on both sides), you can try to increase the *extend_to* parameter. Likewise, if you are observing very narrow signals, you can try decreasing *extend_to* accordingly, as well as try to decrease the *binsize* parameter.

Note that the larger the *–extend_to* parameter, and the smaller the *binsize* parameter, the longer the runtime (and memory usage) of *LIQUORICE* will be.

bedpath_biasmodel, *use_this_biasmodel* and *detect_existing_biasmodel*

If *--bedpath_biasmodel* is not specified (default), *LIQUORICE* does the following:

- Train a seperate bias-model for each region-set (i.e. for each entry in *--bedpathlist*).
- For a given region-set, each region is extended by *--extend_to* in both directions and split into bins.
- As training data for the bias-model, all bins are used except (i) the 5 central bins that cover the core region, if *--all_bins_same_size* is not specified; or (ii) the one central bin, if *--all_bins_same_size* is specified.
- All bins are used to determine the composite, bias-corrected coverage signature.

If *--bedpath_biasmodel* is specified, *LIQUORICE* does the following:

- Train a single, common bias-model for every region-set (i.e. for each entry in *--bedpathlist*).
- Each region in the .bed file specified under *--bedpath_biasmodel* (or each region in *LIQUORICE*'s own set of 10000 random regions if *--bedpath_biasmodel 10k_random* is specified) is extended by *--extend_to_biasmodel* (default: 0) in both directions and split into bins.
- The resulting bins are used as training data for the bias-model.

Note that if *--use_this_biasmodel* is specified, neither of the above workflows is executed. Instead, the provided pre-trained model is used for the correction of all bins of every region-set (i.e. for each entry in *--bedpathlist*). The same applies if *--detect_existing_biasmodel* is specified and a valid model is present under *<samplename>/biasmodel/trained_biasmodel.joblib*.

We have decided to set the former option as default because it yielded somewhat better results for our own samples. However, we do encourage you to try both options for your own cohort. The latter option will also likely be a bit faster in case you are running *LIQUORICE* on a large number of region-sets.

all_bins_same_size

By default (i.e. if *--all_bins_same_size* is not specified), *LIQUORICE* does the following:

- Split each region-of-interest into five bins with sizes corresponding to bins of 10%, 15%, 50%, 15%, and 10% of the total length of the region, respectively. This is done in order to facilitate comparisons between regions of different lengths within the same region set. After splitting, every site consists of five bins, regardless of the initial length of the region.
- Next, the adjacent genomic region (*-- extend_to* basepairs to both sides) is split into bins of *--binsize*bp size. The most upstream bin starts *extend_to* bp upstream of the core region start, and the most downstream bin ends *extend_to* bp downstream of the core region end.

If *--all_bins_same_size* is specified, *LIQUORICE* does the following:

- Use a size of *--binsize* bp for all bins, also the ones at the center.

- The central bin is centered around the center of the region-of-interest. The other bins are tiled such that no gaps arise. Outmost bins will have their center at <center of the region>+-*extend_to*.

Also here, we have decided to set the former option as default because it yielded somewhat better results for our own samples. We do note, however, that differences in bin-size might introduce some slight biases in the coverage profile. While we have nevertheless found that this option works well for us, we do encourage you to try both options for your own cohort.

1.7.2 Parallelization

Increasing the *n_cpus* parameter will cause LIQUORICE to use more threads during the steps that are parallelized, and speed up the analysis. A (potentially faster) alternative to using this setting is to parallelize at the sample level, using GNU parallel (http://dx.doi.org/10.5281/zenodo.16303), which is automatically installed together with *LIQUORICE* :

```
SAMPLES="Sample1 Sample2 Sample3 Sample4 Sample5 Sample6 Sample7 Sample8 Sample9 Sample10
↪"
NR_OF_CORES_TO_BE_USED=5

# Write a simple bash file that contains all required parameters for liquorice, and␣
↪takes the sample name as an argument
# Replace the paths and file name according to your file locations.
echo 'LIQUORICE --bamfile "PATH_TO_BAMFILES/${1}.bam" --refgenome_fasta PATH_TO_
↪REFERENCE_GENOME/hg38.fa --mappability_bigwig PATH_TO_MAPPABILITY_BW/hg38_mappability_
↪75bp.bigwig --bedpathlist "PATH_TO_REGIONSETS/YOUR_REGIONSET_OF_INTEREST.bed" --
↪blacklist hg38 --n_cpus 1 --cna_seg_file "PATH_TO_SEGFILES/${1}.seg"' >LIQUORICE_
↪command.sh

parallel --results logs -j ${NR_OF_CORES_TO_BE_USED} bash LIQUORICE_command.sh  ::: $
↪{SAMPLES}
```

Note that the memory usage will increase with the number of parallel jobs (set by the *-j* parameter of parallel). We usually allow for 3GB of RAM for each job executed in parallel and set the *-j* parameter accordingly (*j* = <Total available Memory on the Computer/Server>/3 GB) when running LIQUORICE with default settings on a region-set of 6000 regions. Note that memory usage also depends on *extend_to*, *binsize*, *speed_mode*, and scales linearly with the number of regions in your region-sets. Finally: LIQUORICE's results will slightly differ based on whether you use *--n_cores 1* or *--n_cores <anything larger than 1>*. This is due to differences in the sampling of fragment lengths and nothing to worry about - both results are equally valid.

1.7.3 Sources for input files

- **bamfiles**: Use your own (or publically available) paired-end whole genome sequencing data from liquid biopsies here. Data should be quality-controlled and trimmed (e.g. using fastp with default settings) as well as mapped (we have used *bwa mem*). We have found that higher sequencing depth improves results - from our own experience, we would recommend using a depth of at least 1x (or higher, if possible). For details, see Figure 6 of our recent publication.
- **mappability .bigwig files**: This reference file should match i) your reference genome (e.g. hg38, hg19, ...) and ii) the read-length of your samples. We provide pre-calculated files for hg38/hg19 and readlengths 35,50,75,100,150 and 250 here. If you require a different read length or reference genome, you can run create_mappability_bigwigs.sh like so:

```
wget https://github.com/epigen/LIQUORICE/raw/master/liquorice/create_
↪mappability_bigwigs.sh
bash create_mappability_bigwigs.sh PATH_TO_GENOME_FASTA READLENGTH NR_OF_
↪CORES_TO_BE_USED
```

This will create a mappability bigwig file in the current directory.

- **regionsets-of-interest** (*bedpathlist*): See *Region-sets*.

1.7.4 Test LIQUORICE with provided test data

To test whether your installation of *LIQUORICE* works as expected, you can test it on a small test dataset from a healthy control sample that we provide. Just follow the example below:

```
# Set desired nr. of cpus
N_CPUS=5

# download and unzip the reference genome and reference mappability file
wget https://hgdownload.soe.ucsc.edu/goldenPath/hg38/bigZips/p12/hg38.p12.fa.gz
gunzip hg38.p12.fa.gz
wget https://github.com/epigen/LIQUORICE/raw/master/liquorice/data/hg38.p12.fa.fai
wget https://github.com/epigen/LIQUORICE/raw/master/liquorice/data/hg38.fa.mappability_
↪100bp.subsetted_for_testdata.bw

# download .bam file of a healthy control liquid biopsy sample (pre-processed to keep␣
↪the size small)
wget https://github.com/epigen/LIQUORICE/raw/master/liquorice/data/Ctrl_17_testdata.bam
wget https://github.com/epigen/LIQUORICE/raw/master/liquorice/data/Ctrl_17_testdata.bam.
↪bai


# download .bed file for universally accessible DHSs
wget https://github.com/epigen/LIQUORICE/raw/master/liquorice/data/universal_DHSs.bed

# run LIQUORICE
LIQUORICE --bamfile Ctrl_17_testdata.bam --refgenome_fasta "hg38.p12.fa" \
        --mappability_bigwig "hg38.fa.mappability_100bp.subsetted_for_testdata.bw" \
        --bedpathlist "universal_DHSs.bed" \
        --blacklist "hg38" --n_cpus "${N_CPUS}" --extend_to 15000
```

1.7.5 Example usage of LIQUORICE and the summary tool

```
# Run LIQUORICE for 4 samples and 3 region-sets, and summarize the results:

SAMPLES="sample1 sample2 sample3 sample4"
CONTROLS="sample1 sample2"
BAMS="./bams"
BEDS="./regionsets"
HG38="./hg38"

for SAMPLE in "$SAMPLES"
```

(continues on next page)

(continued from previous page)

```
do
LIQUORICE --bamfile "${BAMS}/${SAMPLE}.bam" --refgenome_fasta "${HG38}/hg38.fa" \
    --mappability_bigwig "${HG38}/hg38_mappability_75bp.bw" \
    --bedpathlist "${BEDS}/regionset1.bed" "${BEDS}/regionset2.bed" "${BEDS}/regionset3.
↪bed" \
    --blacklist "hg38" --n_cpus 8
done

LIQUORICE_summary --control_name_list ${CONTROLS}
# Please make sure to not put quotation marks around the list specified for --control-
↪name-list.
# Example: --control_name_list sample1 sample2 would be correct, --control_name_list
↪"sample1 sample2" would be incorrect.
```

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

l

liquorice.LIQUORICE, 10
liquorice.LIQUORICE_summary, 10
liquorice.utils.AggregateAcrossRegions, 15
liquorice.utils.BiasFactorWeights, 16
liquorice.utils.BiasModel, 17
liquorice.utils.BinTableBiasFactors, 19
liquorice.utils.CorrectForCNAs, 20
liquorice.utils.CoverageAndSequenceTablePreparation, 21
liquorice.utils.FitGaussians, 24
liquorice.utils.GlobalFragmentSize, 25
liquorice.utils.MeanSequencingDepth, 27
liquorice.utils.Plotting, 27
liquorice.utils.RegionFilteringAndBinning, 31
liquorice.utils.Workflows, 33

INDEX

A

add_biasfactors_percentile_split() (*in module liquorice.utils.Workflows*), 33
aggregate_across_regions() (*in module liquorice.utils.AggregateAcrossRegions*), 15

B

BiasFactorHandler (*class in liquorice.utils.BinTableBiasFactors*), 19
BiasModel (*class in liquorice.utils.BiasModel*), 17
boxplot_score_summary() (*in module liquorice.LIQUORICE_summary*), 10

C

check_difference_to_control() (*in module liquorice.LIQUORICE_summary*), 10
correct_coverage_per_bin_for_cnas() (*in module liquorice.utils.CorrectForCNAs*), 20
CoverageAndSequenceTablePreparation (*class in liquorice.utils.CoverageAndSequenceTablePreparation*), 21
create_summary_table_LIQUORICE() (*in module liquorice.LIQUORICE_summary*), 11

F

filter_regions() (*liquorice.utils.RegionFilteringAndBinning.RegionFilteringAndBinning method*), 31
fit() (*liquorice.utils.BiasModel.SklearnStyleRegressor method*), 18
fit_gaussian_models() (*liquorice.utils.FitGaussians.FitGaussians method*), 25
FitGaussians (*class in liquorice.utils.FitGaussians*), 24
FullPaths (*class in liquorice.LIQUORICE*), 10

G

get_binsize_and_extendto_from_saved_settings() (*in module liquorice.LIQUORICE_summary*), 11
get_binstarts() (*liquorice.utils.RegionFilteringAndBinning.RegionFilteringAndBinning method*), 32
get_binstarts_percentile_split_central_roi() (*liquorice.utils.RegionFilteringAndBinning.RegionFilteringAndBi method*), 32
get_CNV_corrected_coverage_for_bin() (*in module liquorice.utils.CorrectForCNAs*), 20
get_complete_table() (*liquorice.utils.CoverageAndSequenceTablePreparation.Coverage method*), 22
get_coverage_per_bin() (*liquorice.utils.CoverageAndSequenceTablePreparation.Coverage method*), 22
get_coverage_per_bin_mean_over_per_basepair_values_chunked (*liquorice.utils.CoverageAndSequenceTablePreparation.Coverage method*), 22
get_dinuc_weights_binwide() (*in module liquorice.utils.BiasFactorWeights*), 16
get_fwd_mappability_bias_factor() (*liquorice.utils.BinTableBiasFactors.BiasFactorHandler method*), 19
get_GC_and_di_and_trinuc_weights() (*liquorice.utils.BinTableBiasFactors.BiasFactorHandler method*), 19
get_GC_bias_factor() (*liquorice.utils.BinTableBiasFactors.BiasFactorHandler method*), 19
get_GC_weights_binsize1() (*in module liquorice.utils.BiasFactorWeights*), 16
get_GC_weights_binwide() (*in module liquorice.utils.BiasFactorWeights*), 16
get_list_of_coveragefiles() (*in module liquorice.LIQUORICE_summary*), 11
get_list_of_fragment_lengths_and_avg_readlength() (*in module liquorice.utils.GlobalFragmentSize*), 26
get_mapp_weights_binwide() (*in module liquorice.utils.BiasFactorWeights*), 16
get_mappability_for_bin() (*liquorice.utils.CoverageAndSequenceTablePreparation.Coverage method*), 23
get_mappability_per_bin() (*liquorice.utils.CoverageAndSequenceTablePreparation.Coverage method*), 23

get_nucleotide_dicts() (*in module liquorice.utils.BiasFactorWeights*), 16
get_prediction_interval() (*in module liquorice.LIQUORICE_summary*), 12
get_prediction_interval_per_row() (*in module liquorice.LIQUORICE_summary*), 12
get_read_and_fragment_length() (*in module liquorice.utils.GlobalFragmentSize*), 26
get_rev_mappability_bias_factor() (*liquorice.utils.BinTableBiasFactors.BiasFactorHandler method*), 20
get_sequence_per_bin() (*liquorice.utils.CoverageAndSequenceTablePreparation.CoverageAndSequenceTablePreparation method*), 23
get_table_with_bias_factors() (*liquorice.utils.BinTableBiasFactors.BiasFactorHandler method*), 20
get_table_with_corrected_coverage_using_trained_biasmodel() (*liquorice.utils.BiasModel.BiasModel method*), 18
get_trinuc_weights_binwide() (*in module liquorice.utils.BiasFactorWeights*), 17
get_ymin_ymax_over_all_samples() (*in module liquorice.LIQUORICE_summary*), 12
getFragmentLength_worker() (*in module liquorice.utils.GlobalFragmentSize*), 25
getFragmentLength_wrapper() (*in module liquorice.utils.GlobalFragmentSize*), 26

I

is_within_chromsome_borders() (*liquorice.utils.RegionFilteringAndBinning.RegionFilteringAndBinning method*), 32

L

liquorice.LIQUORICE
    module, 10
liquorice.LIQUORICE_summary
    module, 10
liquorice.utils.AggregateAcrossRegions
    module, 15
liquorice.utils.BiasFactorWeights
    module, 16
liquorice.utils.BiasModel
    module, 17
liquorice.utils.BinTableBiasFactors
    module, 19
liquorice.utils.CorrectForCNAs
    module, 20
liquorice.utils.CoverageAndSequenceTablePreparation
    module, 21
liquorice.utils.FitGaussians
    module, 24
liquorice.utils.GlobalFragmentSize
    module, 25
liquorice.utils.MeanSequencingDepth
    module, 27
liquorice.utils.Plotting
    module, 27
liquorice.utils.RegionFilteringAndBinning
    module, 31
liquorice.utils.Workflows
    module, 33

M

main() (*in module liquorice.LIQUORICE*), 10
main() (*in module liquorice.LIQUORICE_summary*), 13
module
    liquorice.LIQUORICE, 10
    liquorice.LIQUORICE_summary, 10
    liquorice.utils.AggregateAcrossRegions, 15
    liquorice.utils.BiasFactorWeights, 16
    liquorice.utils.BiasModel, 17
    liquorice.utils.BinTableBiasFactors, 19
    liquorice.utils.CorrectForCNAs, 20
    liquorice.utils.CoverageAndSequenceTablePreparation, 21
    liquorice.utils.FitGaussians, 24
    liquorice.utils.GlobalFragmentSize, 25
    liquorice.utils.MeanSequencingDepth, 27
    liquorice.utils.Plotting, 27
    liquorice.utils.RegionFilteringAndBinning, 31
    liquorice.utils.Workflows, 33

N

none_or_list_of_float (*liquorice.utils.Plotting.Plotting attribute*), 27
none_or_list_of_int (*liquorice.utils.Plotting.Plotting attribute*), 27
none_or_list_of_str (*liquorice.utils.Plotting.Plotting attribute*), 27

P

parallelize_get_coverage_of_region() (*in module liquorice.utils.CoverageAndSequenceTablePreparation*), 23
parse_args() (*in module liquorice.LIQUORICE*), 10
parse_args() (*in module liquorice.LIQUORICE_summary*), 13
plot_as_subplots() (*in module liquorice.LIQUORICE_summary*), 13
plot_control_distributions_and_test_normality() (*in module liquorice.LIQUORICE_summary*), 14

plot_corrected_vs_uncorrected_coverage() (*liquorice.utils.Plotting.Plotting method*), 27
plot_correlations_biasfactors_coverage_bin_nr() (*liquorice.utils.Plotting.Plotting method*), 28
plot_coverage_bias_correlation() (*liquorice.utils.Plotting.Plotting method*), 29
plot_fitted_model() (*liquorice.utils.Plotting.Plotting method*), 29
plot_overlay() (*in module liquorice.LIQUORICE_summary*), 14
Plotting (*class in liquorice.utils.Plotting*), 27
polyfit() (*in module liquorice.utils.Plotting*), 30
predict() (*liquorice.utils.BiasModel.SklearnStyleRegressor method*), 18

R

read_bed() (*liquorice.utils.CoverageAndSequenceTablePreparation.CoverageAndSequenceTablePreparation method*), 23
RegionFilteringAndBinning (*class in liquorice.utils.RegionFilteringAndBinning*), 31
run_get_coverage_of_region_per_chunk() (*in module liquorice.utils.CoverageAndSequenceTablePreparation*), 24
run_liquorice_on_regionset_with_pretrained_biasmodel() (*in module liquorice.utils.Workflows*), 33
run_liquorice_train_biasmodel_on_same_regions() (*in module liquorice.utils.Workflows*), 35

S

sample_bam_to_get_sequencing_depth() (*in module liquorice.utils.MeanSequencingDepth*), 27
score() (*liquorice.utils.BiasModel.SklearnStyleRegressor method*), 18
set_params() (*liquorice.utils.BiasModel.SklearnStyleRegressor method*), 18
SklearnStyleRegressor (*class in liquorice.utils.BiasModel*), 18

T

train_biasmodel() (*liquorice.utils.BiasModel.BiasModel method*), 18
train_biasmodel_2fold_CV_and_predict() (*liquorice.utils.BiasModel.BiasModel method*), 18
train_biasmodel_for_sample() (*in module liquorice.utils.Workflows*), 38

V

verify_consistant_binning_settings() (*in module liquorice.LIQUORICE_summary*), 15

W

write_bin_bedfile() (*liquorice.utils.RegionFilteringAndBinning.RegionFilteringAndBi method*), 32
write_bin_bedfile_percentile_split_central_roi() (*liquorice.utils.RegionFilteringAndBinning.RegionFilteringAndBi method*), 32

Z

zscore_to_control() (*in module liquorice.LIQUORICE_summary*), 15