pareidolia

Release v1.2.0

Cyril Matthey-Doret

Feb 25, 2022

CONTENTS:

Pareidolia is a command line tool and python package for the analysis of chromosome contact maps. It uses Chromosight's convolution algorithm to detect specific patterns such as chromatin loops and domain borders, and quantify changes in their intensity between two conditions. You can follow Pareidolia's development on Github .

CHAPTER

ONE

TUTORIAL

1.1 Tutorial

1.1.1 Detection

The simplest way to run pareidolia is to specify input files, their corresponding conditions (in the same order) and the output file:

```
pareidolia ctrl1.cool,ctrl2.cool,treat1.cool,treat2.cool control,control,treated,treated␣
↪treatment_loops.tsv
```

Which will detect differential loops between the two conditions and return their coordinates and differential score (treated - control). The condition of the first input file is assumed to be the control and scores will be relative to this condition.

1.1.2 Quantification

In some cases, the set of loops is already known and we just need a differential score for those coordinates. This can be done by specifying the `--bed2d-file` option. When this option is provided, pareidolia skips detection and returns scores for each input position.

```
# Compute differential loop scores at predefined positions.
# Loops appearing upon treatments have positive values and vice versa
pareidolia --bed2d-file my_loops.bed2d \
           ctrl1.cool,ctrl2.cool,treat1.cool,treat2.cool \
           control,control,treated,treated \
           known_loops_change.tsv
```

1.1.3 Additional options

Pareidolia applies 3 filters to reduce spurious detections and only return high confidence scores. Each of those filters is controlled by a command line option:

- `--pearson`: First, only coordinates where at least one sample has a high Chromosight scores are considered
- `--snr`: Then, regions with a signal-to-noise-ratio above a certain value are kept. This is defined as the pixel-wise Chromosight score difference between conditions divided by the standard deviation within conditions.
- `--density`: Finally, pixels of low contact density are filtered out. Contact density is defined as the proportion of nonzero pixel in the window used to compute the Chromosight score.

In some cases, it can be useful to skip those filters altogether when running in quantification mode (with `--bed2d`) to retrieve scores for all input coordinates. For those cases, the `--no-filter` option will skip all 3 filters.

The analysis can also be sped up by restricting computations to a chromosome or region of interest (e.g. 'chr1' or 'chr1:13000-20000'). It is also possible to exploit multiple cpu cores using `--n-cpus` (e.g. `--n-cpus 4` to run with 4 cpus in parallel). This will analyze samples in parallel on different CPUS, which means that memory requirements will increase linearly with the number of CPUs specified and that there can be no more parallel tasks than there are samples to process.

1.1.4 Custom kernels

Standard patterns provided by chromosight can be provided as strings given to the `--kernel`. The default value is `loops`, but we can detect domain borders instead by specifying `--kernel borders`. For those standard Chromosight kernels, default parameter values for `--max_dist` and `--pearson` will be adapted based on the pattern.

It is also possible to provide a custom kernel matrix in the form of a tab-separated text file containing a square numeric matrix with an odd number of rows. This also means that you should specify `--pearson` and `--max-dist` values for the custom kernel.

```
# Generate dummy kernel matrix
echo "0\t1\t0\n0\t1\t0\n0\t1\t0" > mat.tsv
pareidolia --kernel mat.tsv \
           --pearson 0.3 \
           --max-dist 300000 \
           ctrl1.cool,ctrl2.cool,treat1.cool,treat2.cool \
           control,control,treated,treated \
           treatment_custom.tsv
```

CHAPTER

TWO

REFERENCE API

2.1 pareidolia

2.1.1 pareidolia package

Subpackages

pareidolia.cli package

Module contents

This submodule implements a command line interface for pareidolia's pattern change detection pipeline.

Submodules

pareidolia.detection module

Functions used for change detection. cmdoret, 20200403

pareidolia.detection.**get_sse_mat**(*mats*)

Given multiple matrices, return the matrix of position-wise sum of squared errors to median. All input matrices must have the same shape, nonzero coordinates and format. Output matrix will have the same format as inputs.

Parameters **mats** (*Iterable of scipy.sparse.spmatrix*) – The list of data matrices from different samples.

Returns A sparse matrix where each nonzero pixel is the sum of squared difference of all samples to their median background, at the corresponding nonzero pixel.

Return type scipy.sparse.spmatrix

pareidolia.detection.**get_win_density**(*mat*, *win_size=3*, *sym_upper=False*)

Compute local pixel density in sparse matrices using convolution. The convolution is performed in 'full' mode: Computations are performed all the way to the edges by trimming the kernel.

Parameters

- **mat** (*scipy.sparse.csr_matrix*) – The input sparse matrix to convolve.
- **win_size** (*int*) – The size of the area in which to compute the proportion of nonzero pixels. This will be the kernel size used in convolution.
- **sym_upper** (*bool*) – Whether the input matrix is symmetric upper, in which case only the upper triangle is returned.

Returns The result of the convolution of the uniform kernel (win_size x win_size) on the binarized input matrix. Each value represents the proportion of nonzero pixels in the neighbourhood.

Return type scipy.sparse.csr_matrix

pareidolia.detection.**median_bg**(*mats*)

Given multiple sparse matrices with the same shape, format and nonzero coordinates, generate a background matrix made up of the median signal

Parameters **mats** (*Iterable of scipy.sparse.spmatrix:*) – A list of data matrices from different samples.

Returns The median background matrix, where each pixel is the median of the corresponding pixel from all samples.

Return type scipy.sparse.spmatrix

pareidolia.detection.**reps_bg_diff**(*mats*)

Given multiple sample matrices, return a 1D array of all differences between each sample matrix's pixels and their corresponding median background value. All input matrices must have the same shape, nonzero coordinates and format.

Parameters **mats** (*Iterable of scipy.sparse.spmatrix*) – The list of data matrices from different samples.

Returns The distribution of pixel differences to the median background. If there are S matrices of P nonzero pixels, this 1D array will contain P*S elements.

Return type numpy.ndarray of floats

pareidolia.hic_utils module

Functions used to apply Hi-C specific transformations and interact with cooler objects. cmdoret, 20200404

pareidolia.hic_utils.**change_detection_pipeline**(*cool_files*, *conditions*, *kernel='loops'*, *bed2d_file=None*, *region=None*, *max_dist=None*, *min_dist=None*, *subsample=True*, *pearson_thresh=None*, *density_thresh=0.1*, *cnr_thresh=1.0*, *n_cpus=4*)

Run end to end pattern change detection pipeline on input cool files. A list of conditions of the same lengths as the sample list must be provided. The first condition in the list is used as the reference (control) state.

Changes for a specific pattern are computed. A valid chromosight pattern name can be supplied (e.g. loops, borders, hairpins, ...) or a kernel matrix can be supplied directly instead. maximum scanning distance can be specified directly (in basepairs) to override the kernel default value.

Positive diff_scores mean the pattern intensity was increased relative to control (first condition).

Positions with significant changes will be reported in a pandas dataframe. In addition to the score, a contrast-to-noise ratio between 0 and 10 is given to give an estimation of the signal quality. Optionally, a 2D bed file with positions of interest can be specified, in which case change value at these positions will be reported instead. When using a bed2d file.

Parameters

- **cool_files** (*Iterable[str]*) – The list of paths to cool files for the input samples.
- **conditions** (*Iterable[str]*) – The list of conditions matching the samples.
- **kernel** (*Union[numpy.ndarray, str]*) – Either the kernel to use as pattern as a numpy array, or the name of a valid chromosight pattern.

- **bed2d_file** (*Optional[str]*) – Path to a bed2D file containing a list of 2D positions. If this is provided, pattern changes at these coordinates will be quantified. Otherwise, they will be detected based on a threshold.
- **region** (*Optional[Union[Iterable[str], str]]*) – Either a single UCSC format region string, or a list of multiple regions. The analysis will be restricted to those regions.
- **max_dist** (*Optional[int]*) – Maximum interaction distance (in basepairs) to consider in the analysis. If this is not specified and a chromosight kernel was specified, the default max_dist for that kernel is used. If the case of a custom kernel, the whole matrix will be scanned if no max_dist is specified.
- **subsample** (*bool*) – Whether all input matrices should be subsampled to the same number of contacts as the least covered sample.
- **pearson_thresh** (*Optional[float]*) – The pearson correlation threshold to use when detecting patterns. If None, the default value for the kernel is used.
- **density_thresh** (*Optional[float]*) – The pixel density threshold to require. Low coverage windows with a proportion of nonzero pixels below this value are discarded.
- **n_cpus** (*int*) – Number of CPU cores to allocate for parallel operations.
- **min_dist** (*Optional[int]*) –
- **cnr_thresh** (*Optional[float]*) –

Returns The list of reported 2D coordinates and their change intensities.

Return type pd.DataFrame

pareidolia.hic_utils.**coords_to_bins**(*clr*, *coords*)

Converts genomic coordinates to a list of bin ids based on the whole genome contact map.

Parameters

- **coords** (*pandas.DataFrame*) – Table of genomic coordinates, with columns chrom, pos.
- **clr** (*cooler.api.Cooler*) –

Returns Indices in the whole genome matrix contact map.

Return type numpy.array of ints

pareidolia.hic_utils.**detection_matrix**(*samples*, *kernel*, *region=None*, *subsample=None*, *max_dist=None*, *pearson_thresh=None*, *density_thresh=None*, *cnr_thresh=0.3*, *n_cpus=4*)

Run the detection process for a single chromosome or region. This is abstracted from all notions of chromosomes and genomic coordinates.

Parameters

- **samples** (*pandas.core.frame.DataFrame*) –
- **kernel** (*numpy.ndarray*) –
- **region** (*Optional[str]*) –
- **subsample** (*Optional[int]*) –
- **max_dist** (*Optional[int]*) –
- **pearson_thresh** (*Optional[float]*) –
- **density_thresh** (*Optional[float]*) –
- **cnr_thresh** (*Optional[float]*) –

- **n_cpus** (*int*) –

Return type *Tuple*[*Optional*[scipy.sparse.csr.csr_matrix], *Optional*[scipy.sparse.csr.csr_matrix]]

pareidolia.hic_utils.**get_min_contacts**(*coolers*, *region=None*)

Given several cooler objects, returns the number of contacts in the least covered matrix. Optionally, a region can be given in UCSC format, in which case coverage will be computed only in the matrix from that region.

Parameters

- **coolers** (*Iterable[cooler.api.Cooler]*) –
- **region** (*Optional[str]*) –

Return type int

pareidolia.hic_utils.**make_density_filter**(*mats*, *density_thresh=0.1*, *win_size=3*, *sym_upper=False*)

Given a list of sparse matrices, generate a "density filter". This new sparse matrix is a boolean mask where values indicate whether the proportion of nonzero pixels in the neighbourhood of diameter win_size is above the input threshold in all input matrices.

Parameters

- **mats** (*Iterable of scipy.sparse.csr_matrix*) – The matrices to be combined into a filter.
- **density_thresh** (*float*) – The required proportion of nonzero pixels in the neighbourhood pass the filter.
- **win_size** (*int*) – The diameter of the neighbourhood in which to compute the proportion of nonzero pixels.
- **sym_upper** (*bool*) – Whether the matrix is symmetric upper. In this case, computations are performed in the upper triangle.

Returns The sparse boolean mask representing the density filter. Values are True where all input matrices passed the threshold.

Return type scipy.sparse.csr_matrix of bools

pareidolia.hic_utils.**preprocess_hic**(*clr*, *min_contacts=None*, *region=None*)

Given an input cooler object, returns the preprocessed Hi-C matrix. Preprocessing involves (in that order): subsetting region, subsampling contacts, normalisation, detrending (obs / exp). Balancing weights must be precomputer in the referenced cool file. Region must be in UCSC format.

Parameters

- **clr** (*cooler.api.Cooler*) –
- **min_contacts** (*Optional[int]*) –
- **region** (*Optional[str]*) –

Return type scipy.sparse.csr.csr_matrix

pareidolia.io module

Functions used to load and save files. cmdoret, 20200406

pareidolia.io.**get_coolers**(*path_list*)

Load multiple cool files, ensuring they have the same resolution and shape.

Parameters **path_list** (*Iterable[str]*) –

Return type *List*[cooler.api.Cooler]

pareidolia.preprocess module

Functions used to clean and prepare sparse matrices for detection. cmdoret, 20200403

pareidolia.preprocess.**fill_nnz**(*mat*, *all_nnz*, *fill_value=1e-09*)

Given an input sparse matrix and a superset of nonzero coordinates, fill the matrix to ensure all values in the set are stored explicitely with the value of fill_value.

Parameters

- **mat** (*sp.csr_matrix*) – The sparse matrix of a single sample.
- **all_nnz** (*np.ndarray of ints*) – A 2D array of shape Nx2, where N is the number of nonzero elements to fill. The columns represent the row and column coordinates of those elements.
- **fill_value** (*float*) – The value to use when filling the nonzero elements in the matrix. Has to be the same datatype as the input matrix.

Returns The filled sparse matrix, where all coordinates in all_nnz have been filled with fill_value.

Return type sp.csr_matrix

pareidolia.preprocess.**get_common_valid_bins**(*mats*, *n_mads=5*)

Generates an array of valid bins indices, using the intersection of valid bins from all input sparse matrices. All input matrices must be square and have the same shape. Valid bins are defined based on their proportion of nonzero pixels.

Parameters

- **mats** (*Iterable of sp.csr_matrix*) – A list sparse matrices representing Hi-C contacts, each matrix represents a sample.
- **n_mads** (*float*) – A bin is considered missing if its proportion of nonzero pixels is lower than n_mads median absolute deviations below the median of the bin distribution for the whole matrix.

Returns A 1D array containing the indices of valid (non-missing) bins.

Return type np.ndarray of ints

pareidolia.preprocess.**get_nnz_union**(*mats*)

Given a list of sparse matrices, return the union of their nonzero coordinates, in the form of a 2D numpy array with 1 coordinate per row, with 2 columns representing coordinates rows and columns.

Parameters **mats** (*Iterable of sp.csr_matrix*) – List containing the sparse matrices from each sample.

Returns A 2D numpy array containing the union of nonzero coordinates in the input sparse matrices. The array has shape Nx2 where N is the number of coordinates. The columns represent row and column coordinates.

Return type np.ndarray of int

pareidolia.stats module

Functions used to evaluate statistical significance of changes. cmdoret, 20200403

pareidolia.stats.**pval_to_tval**(*pval*, *n_samples*)
Conversion of a single p-value to corresponding t Given a desired p-value and sample size, return the corresponding t-value. The absolute value for the 2 sided hypothesis is returned.

pareidolia.stats.**tvals_to_pvals**(*tvals*, *n_samples*)
Compute p-values from t-values Given an array of test statistics from multiple t-tests and the (fixed) number of samples used for each test, compute p-values for all tests

pareidolia.stats.**vals_to_percentiles**(*vals*, *dist*)
Return the percentiles corresponding to values in a distribution.

Parameters

- **vals** (*np.ndarray of floats*) – The values for which percentiles should be returned.
- **dist** (*np.ndarray of floats*) – The distribution to use when computing the percentiles.

Returns The array of percentiles corresponding to input values

Return type np.ndarray of floats

pareidolia.stats.**vec_lm**()

pareidolia.stats.**vec_ttest**(*arr1*, *arr2*)
Given 2 numpy arrays containing data for 2 different conditions, where each array has shape shape rxd where r is the number of replicates and N is the number of independent tests to perform. N must be equal in both array, however the number of replicates can differ.

Module contents

pareidolia - Multi-sample change detection in Hi-C patterns

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

p

INDEX