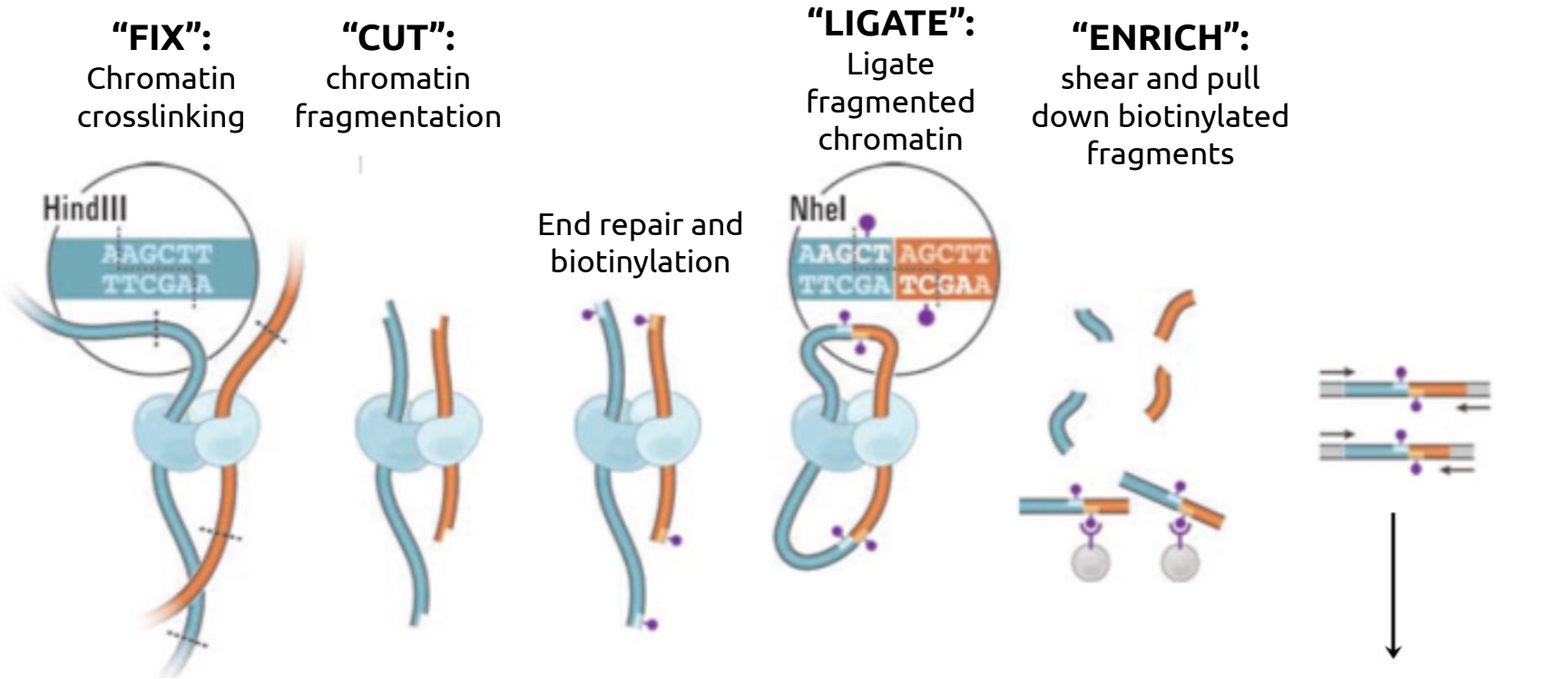
What we see in Hi-C: Introduction to chromatin biology and sequencing assays, Hi-C data browsing and feature extraction

Aleksandra Galitsyna

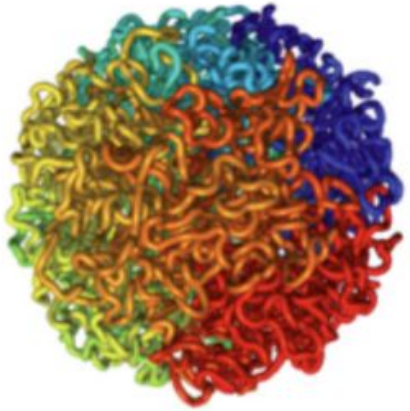
Tips and answers for the workshop
https://github.com/mirnylab/Hi-C_Workshop

Chromosomes Conformation Capture with sequencing – Hi-C



Lieberman-Aiden et al. 2009

Hi-C walkthrough: theoretical example

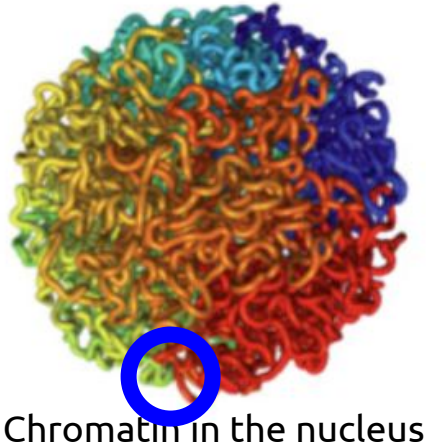


Chromatin in the nucleus

“Stretched” genome as a coordinate system

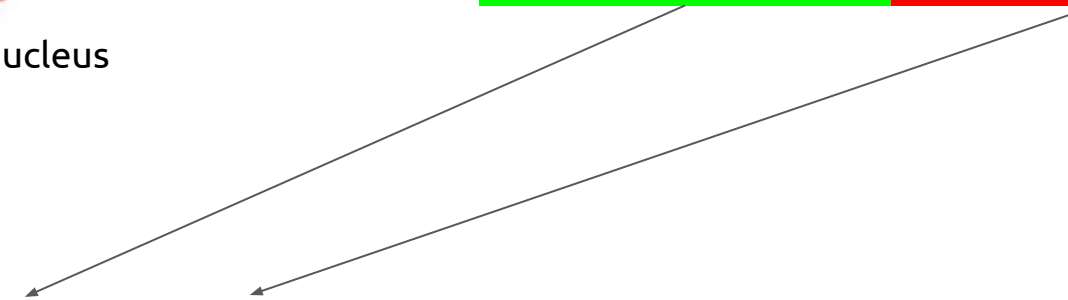
Slides: Sergey Venev

Hi-C walkthrough: single read



Chromatin in the nucleus

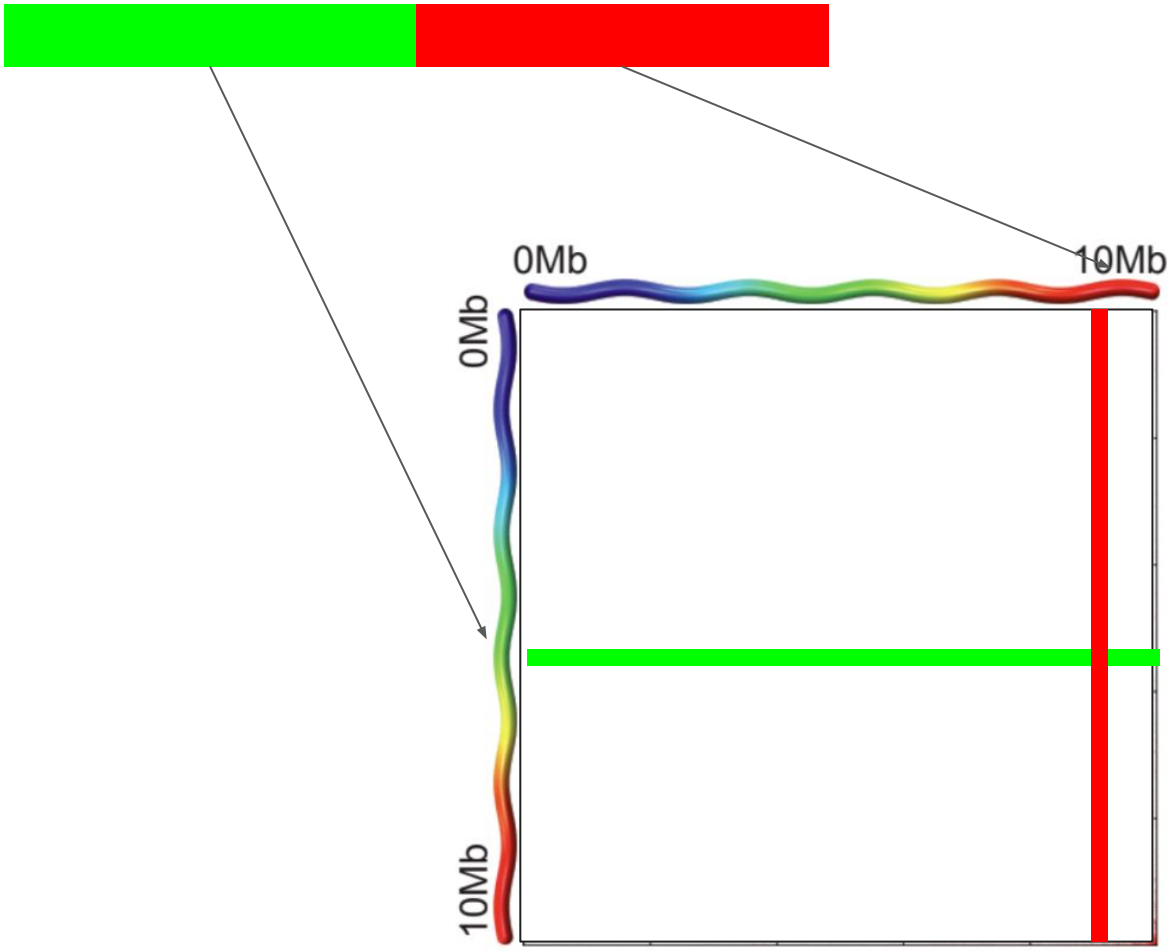
A typical “chimeric” Hi-C read
representing long-range interaction



“Stretched” genome as a coordinate system

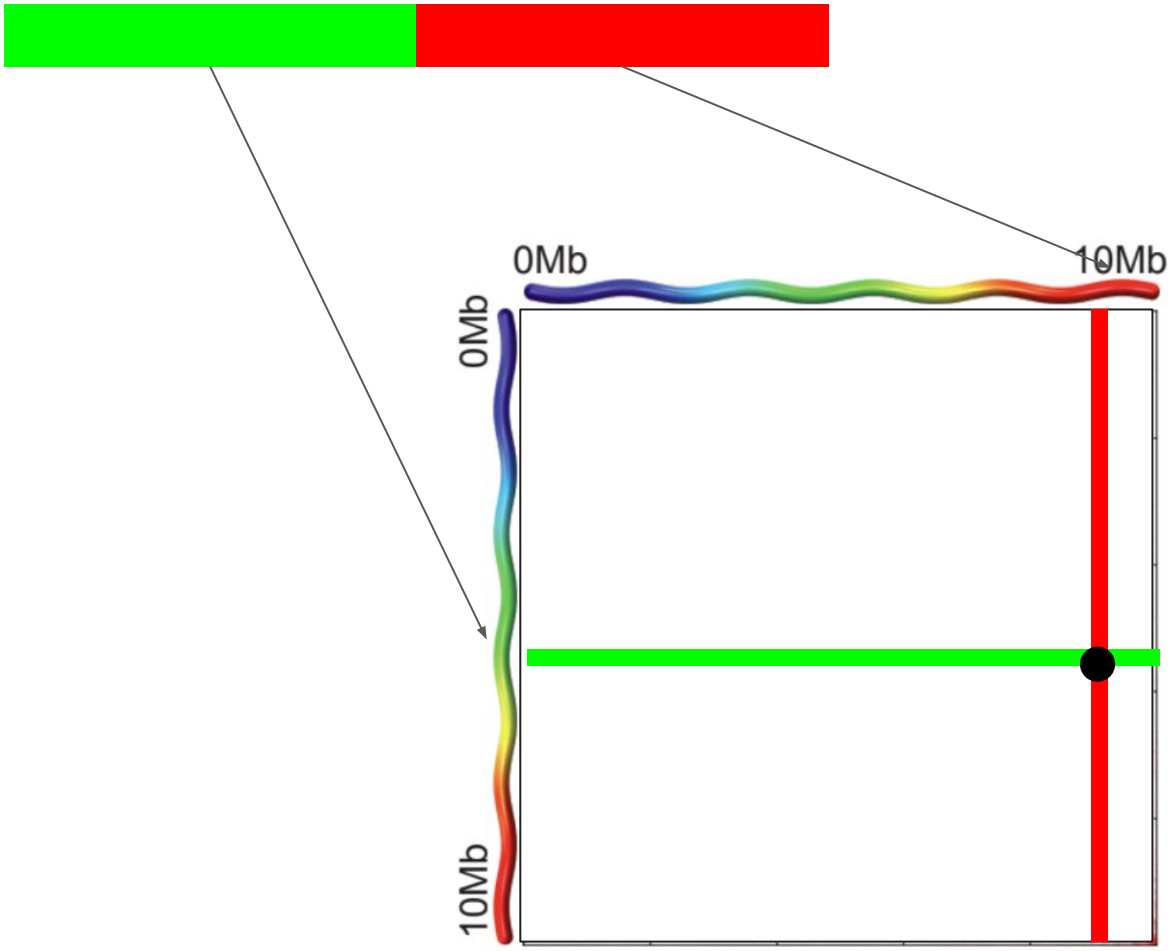
Slides: Sergey Venev

Hi-C walkthrough: single read mapping



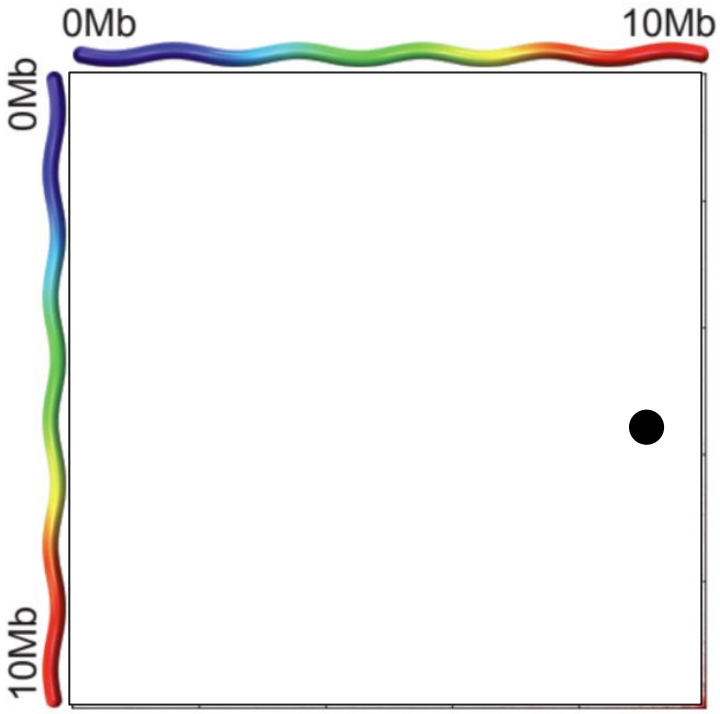
Slides: Sergey Venev

Hi-C walkthrough: single read mapping



Slides: Sergey Venev

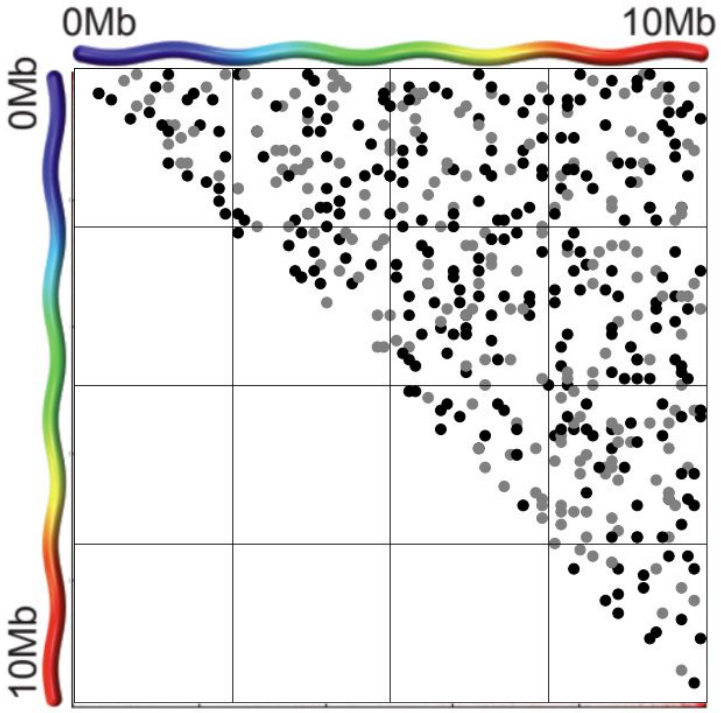
Hi-C walkthrough: single read contribution



keep adding interactions …

Slides: Sergey Venev

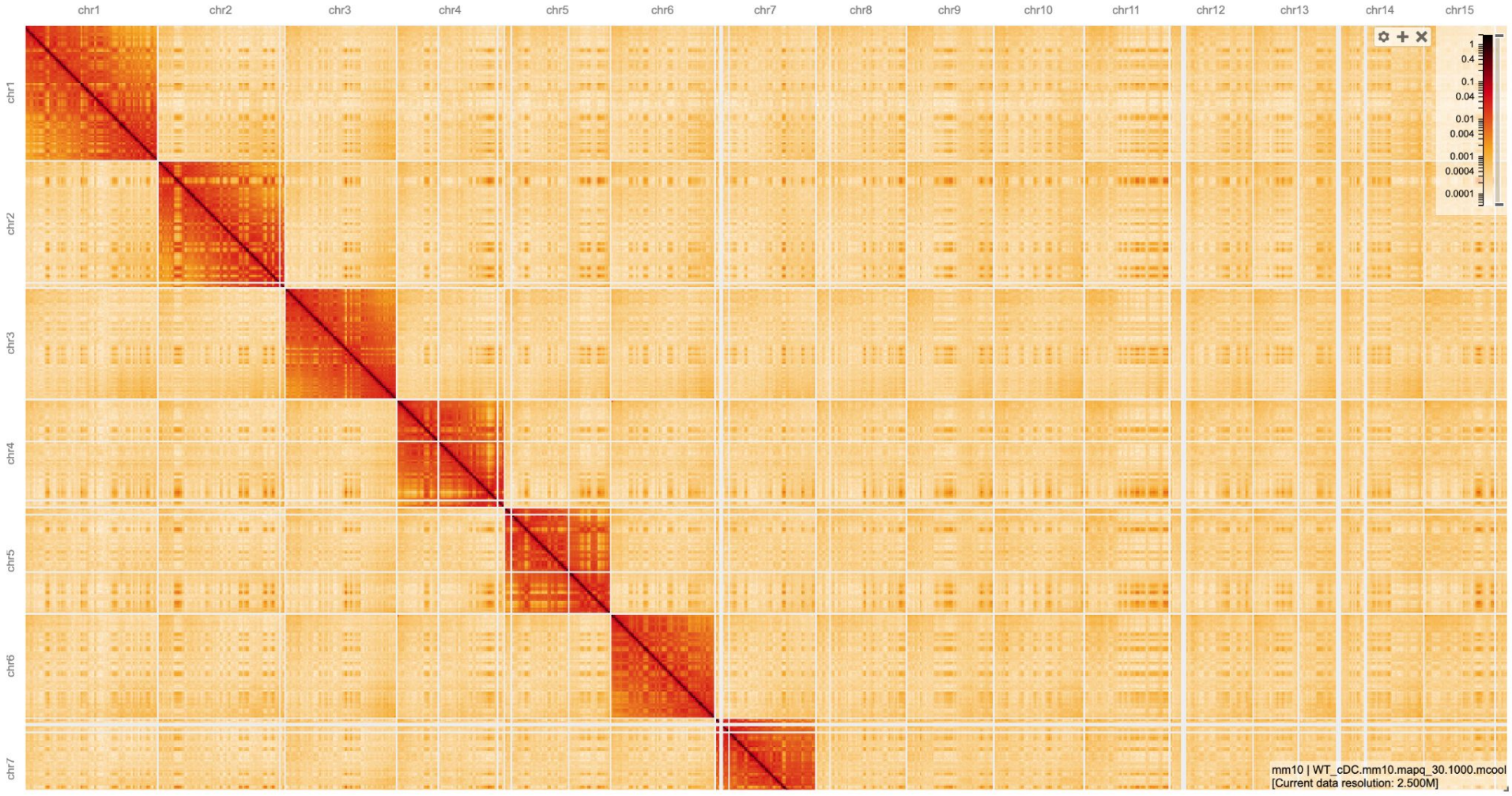
Hi-C walkthrough: multiple reads



~10’000’000 - 10’000’000’000 interactions later …

Slides: Sergey Venev

Example Hi-C map:



Workshop outline

Part I. Qualitative analysis of Hi-C with higlass and higlass-based browsers

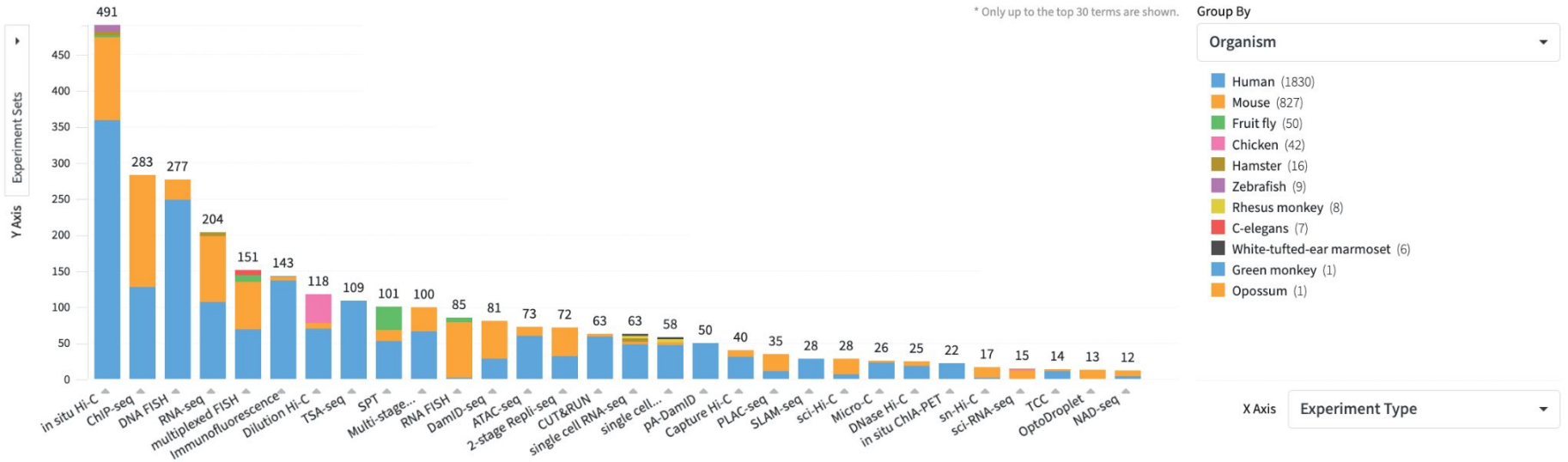
Part II. Quantitative analysis of Hi-C with Open2C tools:

1. Maps visualization
2. Insulation
3. Compartments
4. Scaling plots

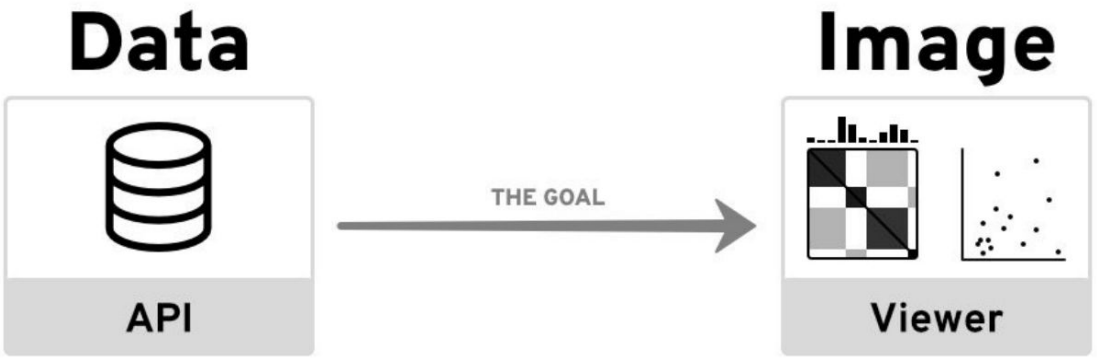
https://github.com/mirnylab/Hi-C_Workshop

Introduction: Hi-C and genomics data browsing

3D genome database with Hi-C and multiple variations of it:
https://data.4dnucleome.org/

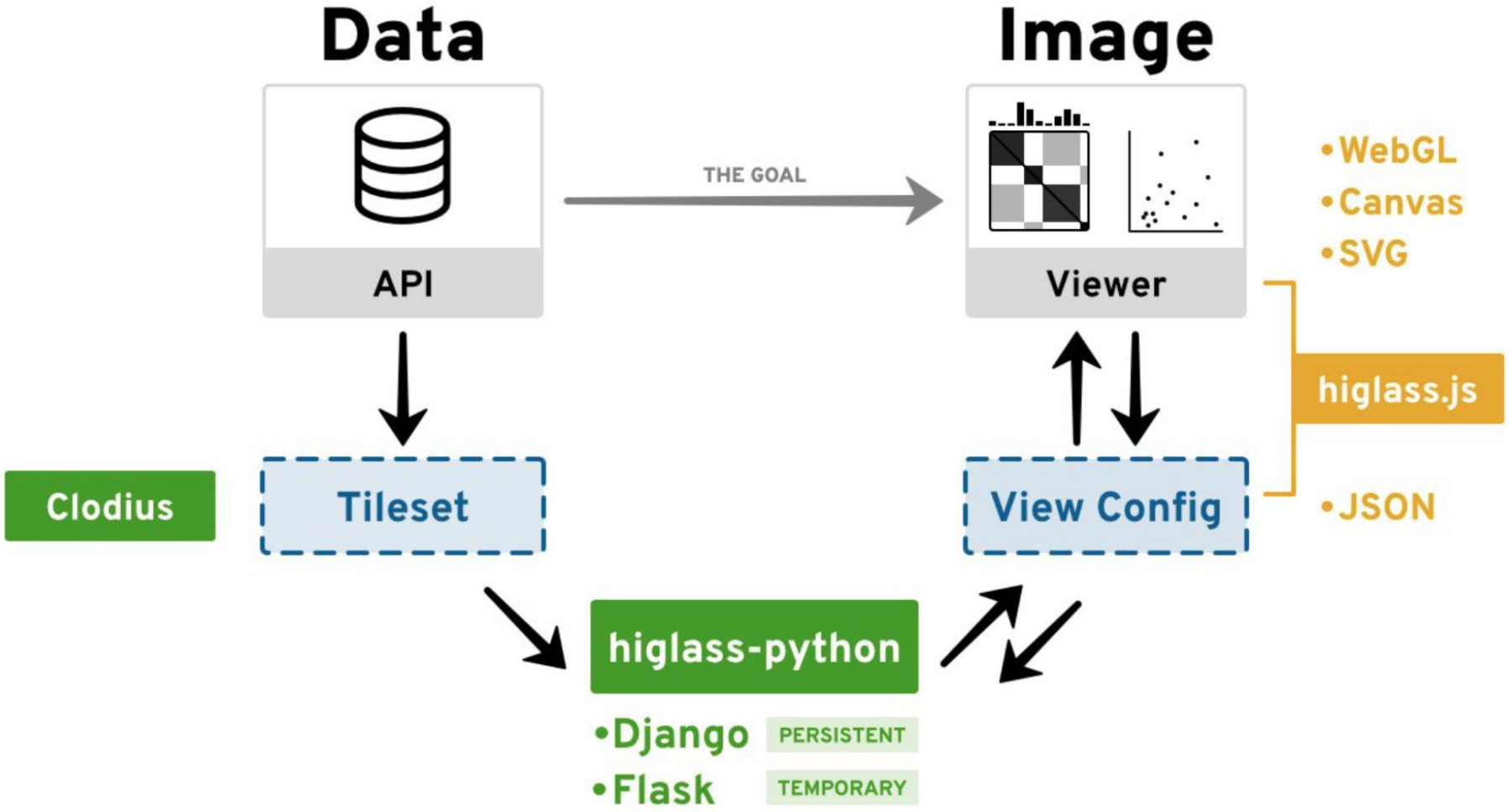


Why showing Hi-C data is not a simple task?



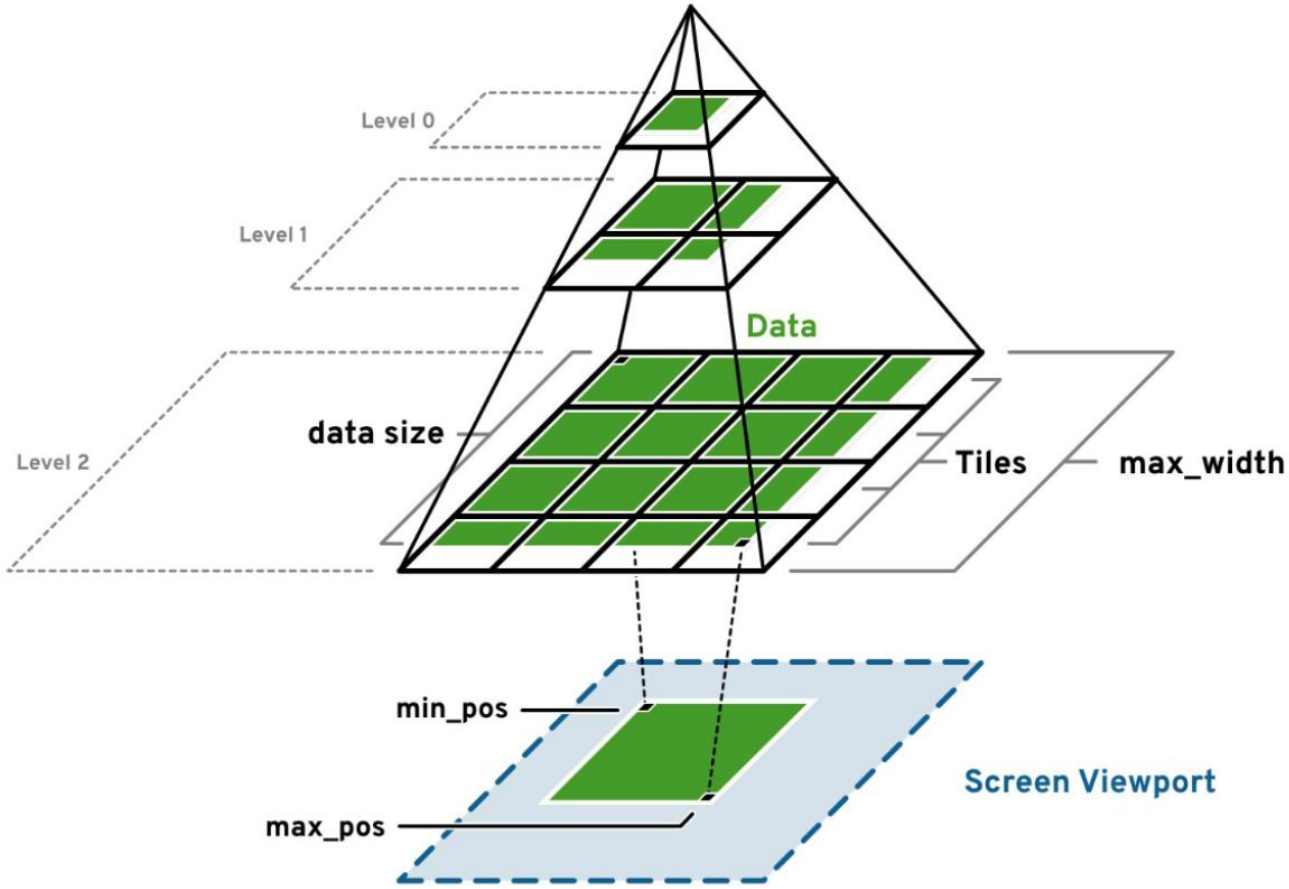
https://github.com/higlass/scipy19

Why showing Hi-C data is not a simple task?



https://github.com/higlass/scipy19

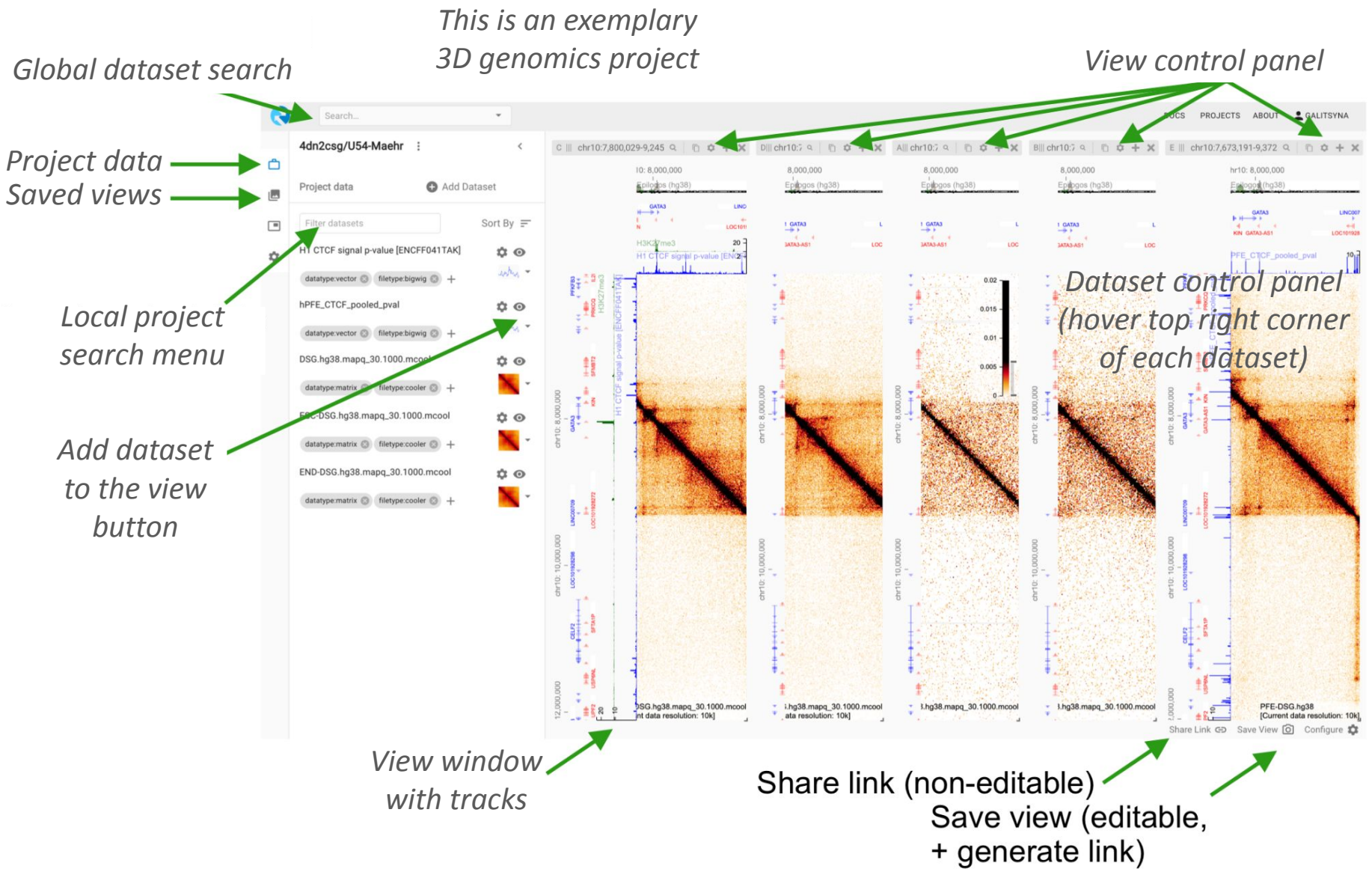
Fetching tilesets: wise way to treat immense amounts of data



Options where to browse Hi-C data:

- https://higlass.io/examples - HiGlass browser for numerous datasets
- https://data.4dnucleome.org/ - browser for 4DN datasets with HiGlass view for each
- https://mirnylab.mit.edu/projects/microc/ - HiGlass incorporated into a lab website example
- https://resgen.io/ - interactive HiGlass-based browser with opportunity to upload user data and store the views

Tips: Resgen.io navigation

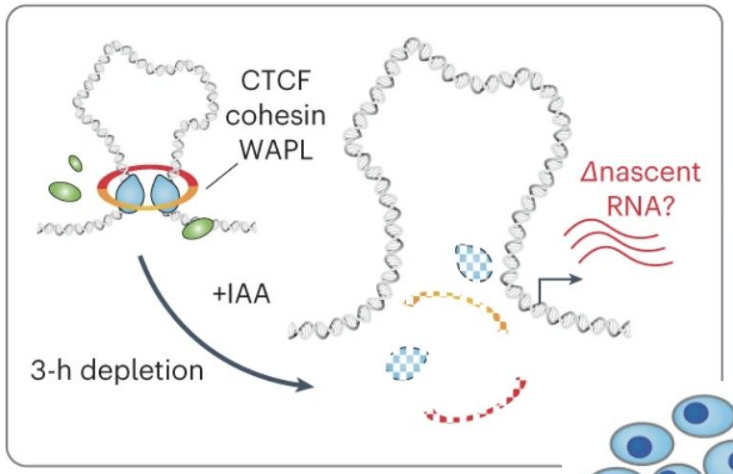


What is the data for the workshop?
Micro-C with acute degradation of loop extrusion factors

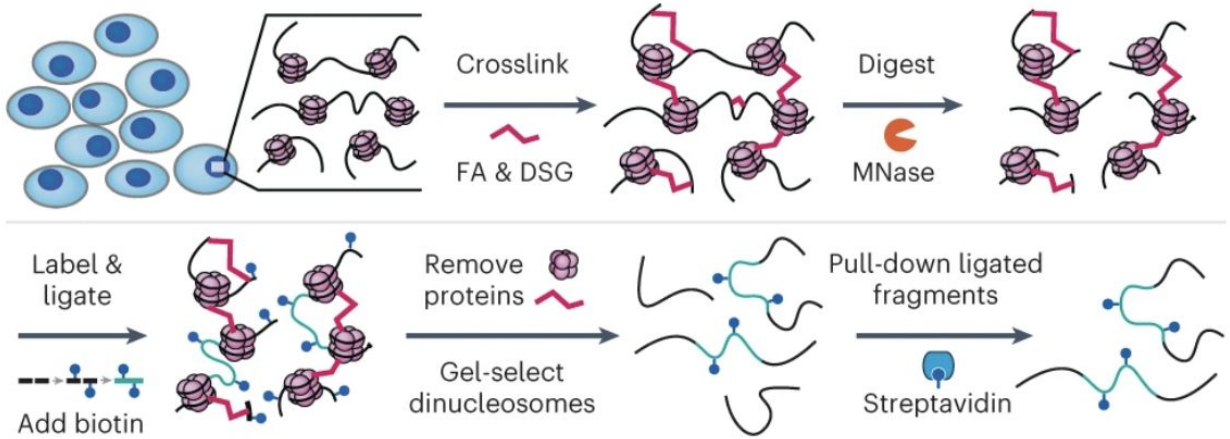
Hsieh 2021 (bioRxiv) - 2022 Nature https://www.nature.com/articles/s41588-022-01223-8

IAA = Indole-3-acetic acid, auxin family hormone

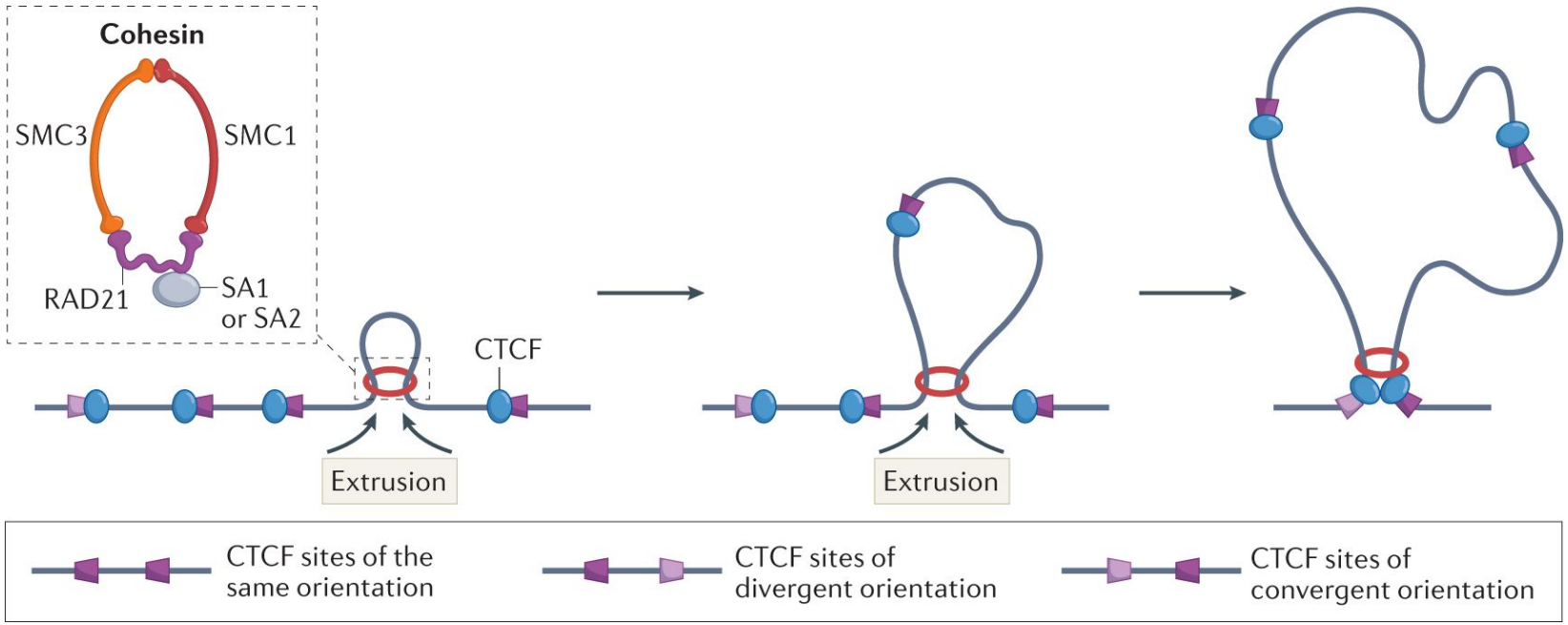
Induced degradation of the protein with Auxin Inducible Degron (AID)



RCMC version of it
(Goel 2023)

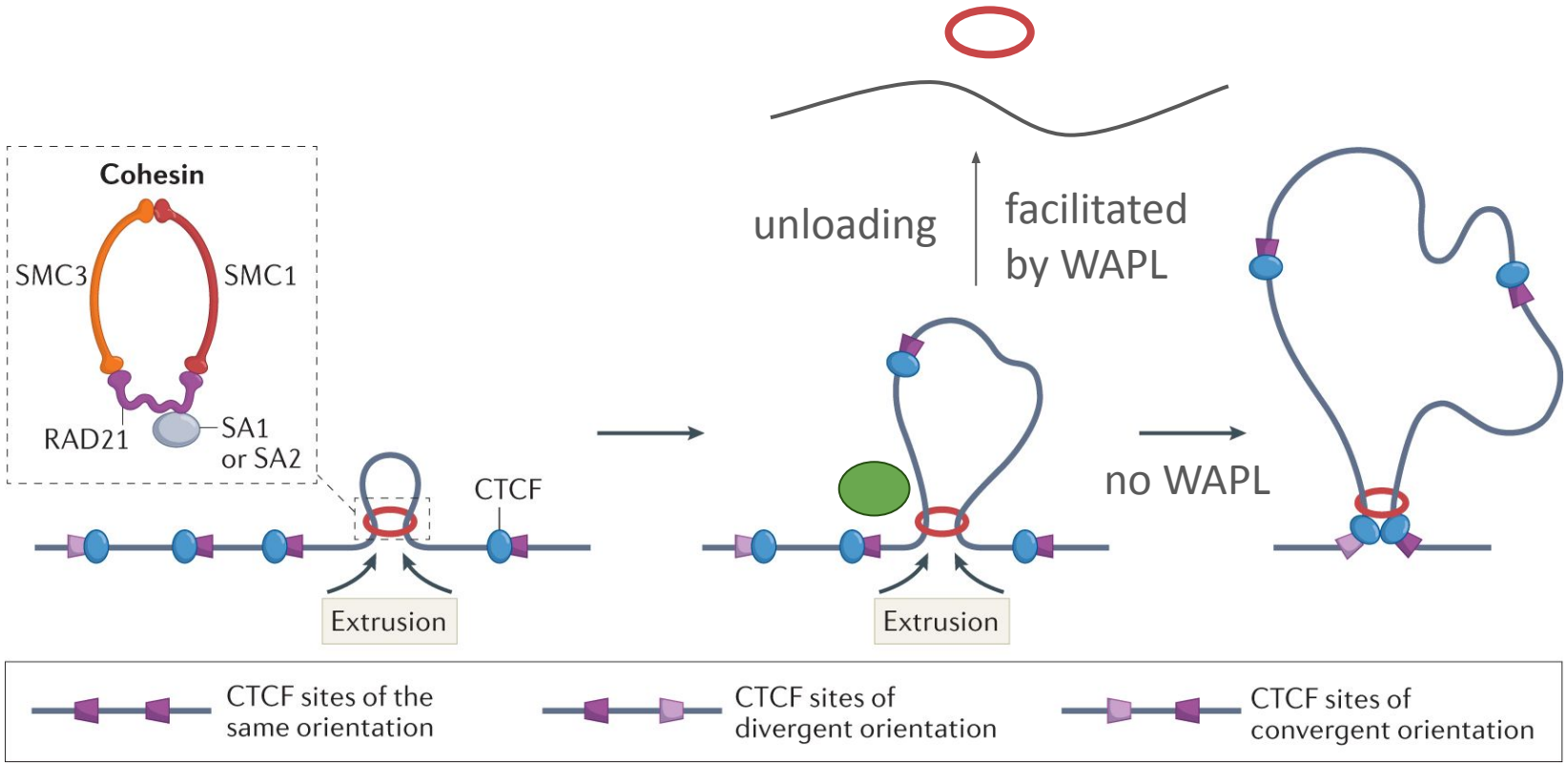


What are the degraded factors?



based on Zhang et al. 2022

What are the degraded factors?



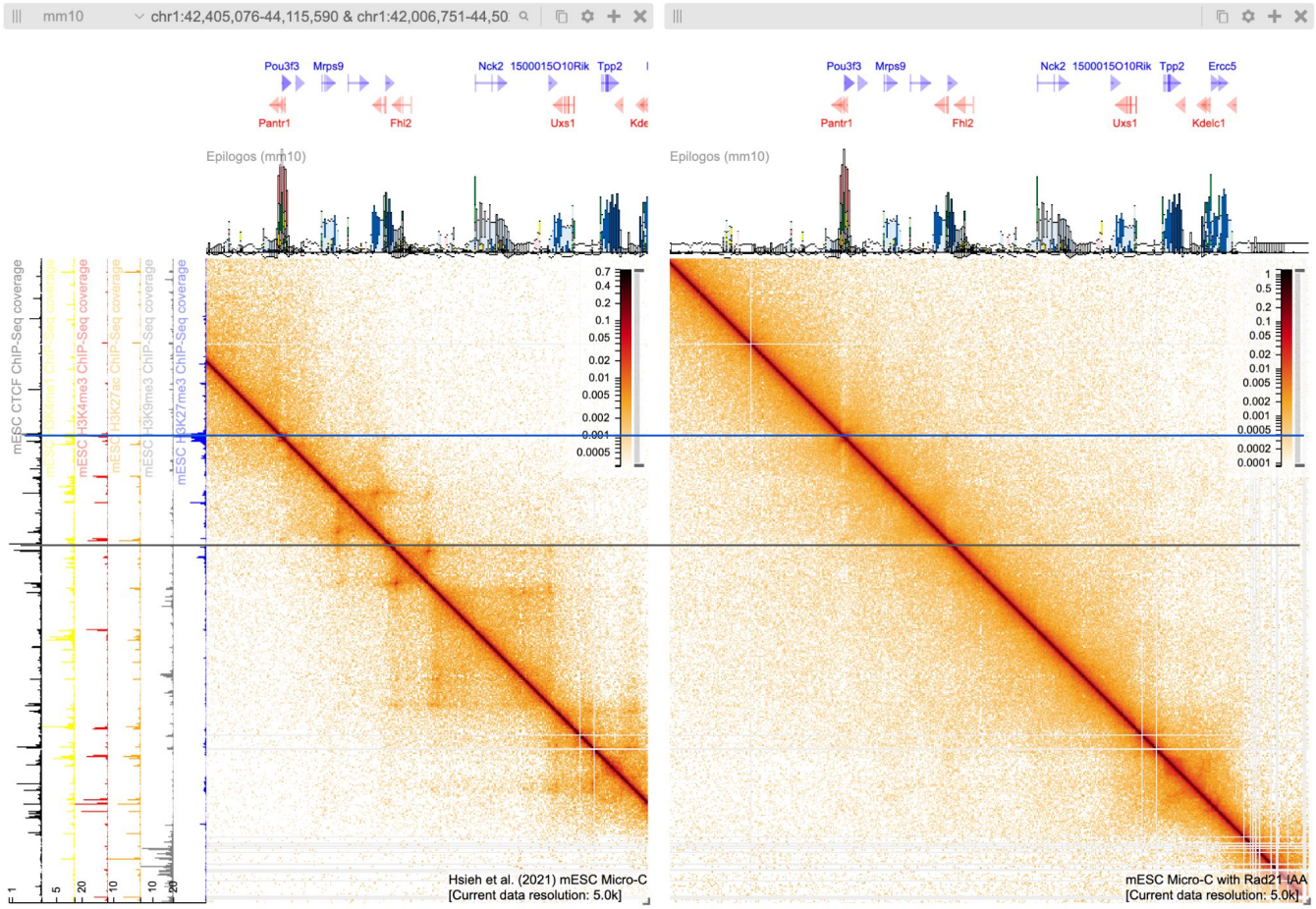
based on Zhang et al. 2022

Practice task 2.4: factors and structures

CTCF

H3K27ac-enriched regions

H3K27me3-enriched regions



Quantitative analysis of Hi-C

Open2C - Open Chromosome Collective

Mirny lab 2018, MIT

Max Imakaev

Aleks. Galitsyna

Geoff Fudenberg
(USC QCB)

Anton Goloborodko
(IMBA Vienna)

Ilya Flyamer
(Giorgetti lab, FMI Basel)

Nezar Abdennur
(UMassMed)

Sergey Venev
(Dekker lab, UMass)

Quantitative analysis of Hi-C

Open2C - Open Chromosome Collective on GitHub: https://github.com/open2c

Open Chromosome Collective

102 followers · https://open2c.github.io/ · https://bit.ly/open2c-slack · @Open2C_team · open.chromosome.collective@gmai...

cooler Public

A cool place to store your Hi-C

Python · 196 · 52

cooltools Public

The tools for your .cool's

Jupyter Notebook · 127 · 50

pairtools Public

Extract 3D contacts (.pairs) from sequencing alignments

Python · 93 · 33

bioframe Public

Genomic interval operations on Pandas DataFrames

Python · 169 · 27

distiller-nf Public

A modular Hi-C mapping pipeline

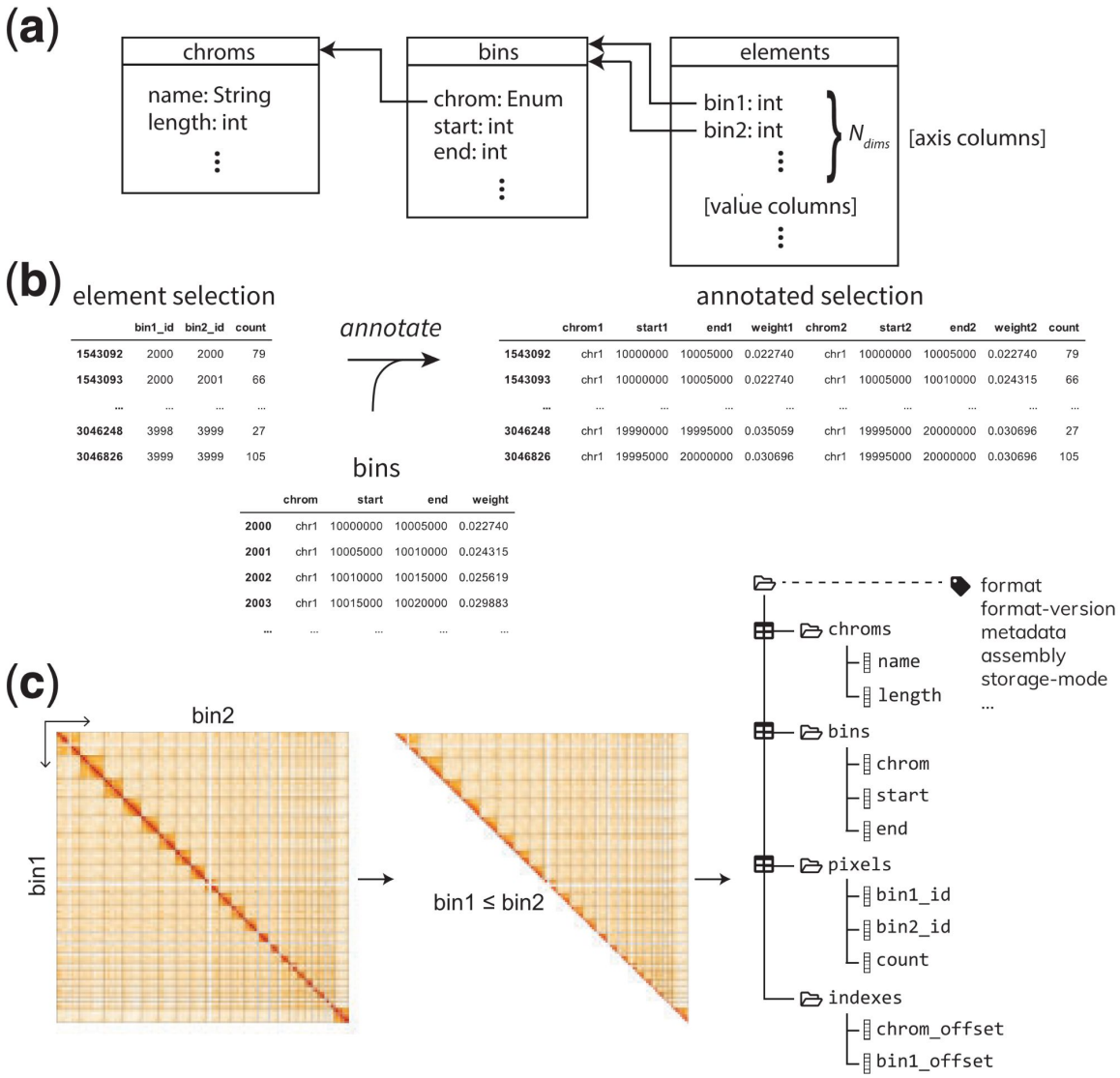
Groovy · 82 · 25

polychrom Public

Chromosome polymer simulations and in silico capture experiments

Python · 44 · 16

Cooler: sparse storage of Hi-C data



	bin1_id	bin2_id	count
1543092	2000	2000	79
1543093	2000	2001	66
...	...	...	...
3046248	3998	3999	27
3046826	3999	3999	105

	chrom1	start1	end1	weight1	chrom2	start2	end2	weight2	count
1543092	chr1	10000000	10005000	0.022740	chr1	10000000	10005000	0.022740	79
1543093	chr1	10000000	10005000	0.022740	chr1	10005000	10010000	0.024315	66
...	...	...	...	...	...	...	...	...	...
3046248	chr1	19990000	19995000	0.035059	chr1	19995000	20000000	0.030696	27
3046826	chr1	19995000	20000000	0.030696	chr1	19995000	20000000	0.030696	105

	chrom	start	end	weight
2000	chr1	10000000	10005000	0.022740
2001	chr1	10005000	10010000	0.024315
2002	chr1	10010000	10015000	0.025619
2003	chr1	10015000	10020000	0.029883
...	...	...	...	...

Fetching cooler data

matrix(*field=None, balance=True, sparse=False, as_pixels=False, join=False, ignore_index=True, divisive_weights=None, chunksize=10000000*) [source]

Contact matrix selector

Parameters:

- **field** (*str, optional*) – Which column of the pixel table to fill the matrix with. By default, the 'count' column is used.
- **balance** (*bool, optional*) – Whether to apply pre-calculated matrix balancing weights to the selection. Default is True and uses a column named 'weight'. Alternatively, pass the name of the bin table column containing the desired balancing weights. Set to False to return untransformed counts.
- **sparse** (*bool, optional*) – Return a scipy.sparse.coo_matrix instead of a dense 2D numpy array.
- **as_pixels** (*bool, optional*) – Return a DataFrame of the corresponding rows from the pixel table instead of a rectangular sparse matrix. False by default.
- **join** (*bool, optional*) – If requesting pixels, specifies whether to expand the bin ID columns into (chrom, start, end). Has no effect when requesting a rectangular matrix. Default is True.
- **ignore_index** (*bool, optional*) – If requesting pixels, don't populate the index column with the pixel IDs to improve performance. Default is True.
- **divisive_weights** (*bool, optional*) – Force balancing weights to be interpreted as divisive (True) or multiplicative (False). Weights are always assumed to be multiplicative by default unless named KR, VC or SQRT_VC, in which case they are assumed to be divisive by default.
- **chunksize** (*int*)

Returns: *Matrix selector*

Return type: *RangeSelector2D*

Notes

If `as_pixels=True`, only data explicitly stored in the pixel table will be returned: if the cooler's storage mode is symmetric-upper, lower triangular elements will not be generated. If `as_pixels=False`, those missing non-zero elements will automatically be filled in.

See cooler docs for more

Matrix selector

The `cooler.Cooler.matrix()` selector supports two types of queries:

- 2D bin range queries using slice indexing syntax
- 2D genomic range range queries using the `fetch` method

The matrix selector's fetch method is intended to represent a **2D range query** (rectangular window), similar to the slice semantics of a 2D array. Given a matrix selector `sel`, when calling `sel.fetch(region1, region2)` the `region1` and `region2` are single contiguous genomic ranges along the first and second axes of the contact matrix. This mirrors the global slice indexing interface of the matrix selector `sel[a:b, c:d]`, where the only difference is that the genomic range syntax cannot cross chromosome boundaries. If `region2` is not provided, it is taken to be the same as `region1`. That means that `sel.fetch('chr2:10M-20M')` returns the same result as `sel.fetch('chr2:10M-20M', 'chr2:10M-20M')`. As a single rectangular window, queries like `sel.fetch('chr2', 'chr3')` will return *inter*-chromosomal values and not intra-chromosomal ones.

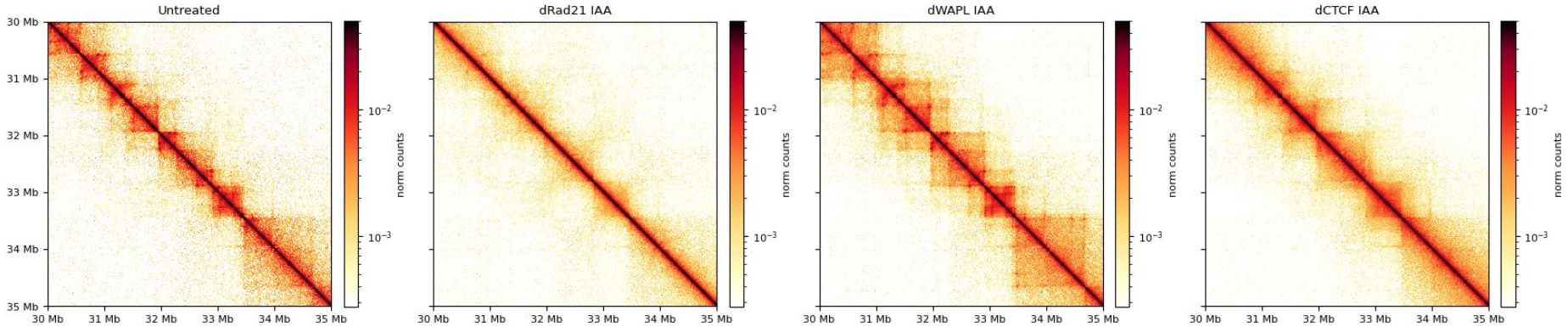
```
>>> c.matrix(balance=False)[1000:1005, 1000:1005]
array([[120022,  34107,  17335,  14053,   4137],
       [ 34107,  73396,  47427,  16125,   3642],
       [ 17335,  47427,  80458,  25105,   5394],
       [ 14053,  16125,  25105, 104536,  27214],
       [  4137,   3642,   5394,  27214, 114135]])

>>> matrix = c.matrix(sparse=True, balance=False)
>>> matrix
<cooler.core.RangeSelector2D at 0x7fdb2e245908>

>>> matrix[:]
<3114x3114 sparse matrix of type '<class 'numpy.int64'>'
    with 8220942 stored elements in COOrdinate format>

>>> c.matrix(balance=False, as_pixels=True, join=True)[1000:1005, 1000:1005]
   chrom1     start1       end1 chrom2     start2       end2   count
0    chr5  115000000  116000000   chr5  115000000  116000000  120022
1    chr5  115000000  116000000   chr5  116000000  117000000   34107
2    chr5  115000000  116000000   chr5  117000000  118000000   17335
3    chr5  115000000  116000000   chr5  118000000  119000000   14053
4    chr5  115000000  116000000   chr5  119000000  120000000    4137
5    chr5  116000000  117000000   chr5  116000000  117000000   73396
6    chr5  116000000  117000000   chr5  117000000  118000000   47427
7    chr5  116000000  117000000   chr5  118000000  119000000   16125
8    chr5  116000000  117000000   chr5  119000000  120000000    3642
9    chr5  117000000  118000000   chr5  117000000  118000000   80458
10   chr5  117000000  118000000   chr5  118000000  119000000   25105
11   chr5  117000000  118000000   chr5  119000000  120000000    5394
12   chr5  118000000  119000000   chr5  118000000  119000000  104536
13   chr5  118000000  119000000   chr5  119000000  120000000   27214
14   chr5  119000000  120000000   chr5  119000000  120000000  114135


>>> A1 = c.matrix().fetch('chr1')
>>> A2 = c.matrix().fetch('chr3:10,000,000-20,000,000')
>>> A3 = c.matrix().fetch( ('chr3', 10000000, 20000000) )
>>> A4 = c.matrix().fetch('chr2', 'chr3')
>>> A5 = c.matrix().fetch('chr3:10M-20M', 'chr3:35M-40M')
```

Tips: control color scale with matplotlib.colors

```
# plot heatmaps at megabase resolution with 3 levels of zoom in log-scale with a consistent colormap
from matplotlib.colors import LogNorm

norm = LogNorm(vmax=1e-1)

im = ax.matshow(
    data_chr17,
    norm=norm,
    extent=(0, clr.chromsizes['chr1'], clr.chromsizes['chr1'], 0)
);
```

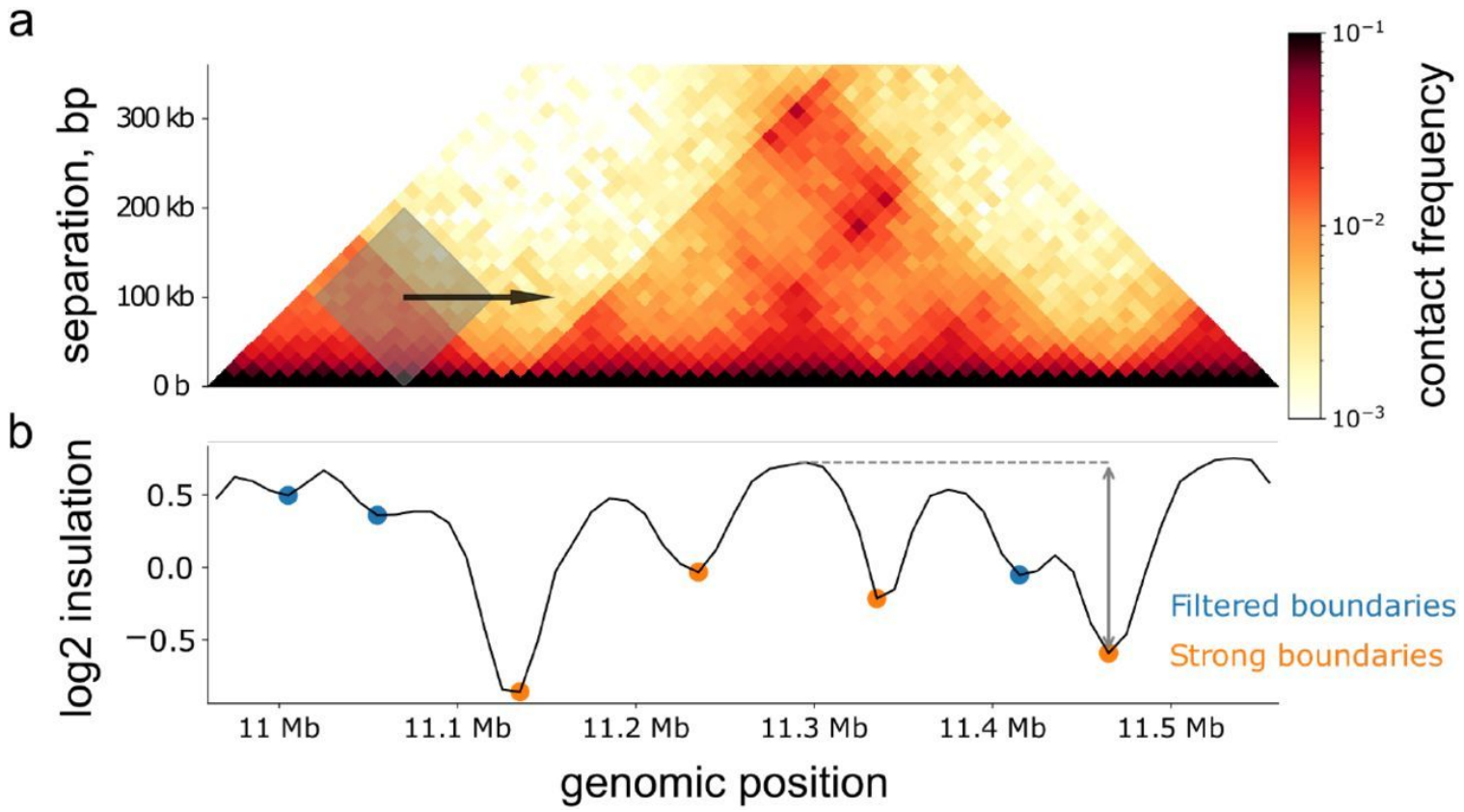
See matplotlib normalization docs for more

Practice task 1.5: Output for dRad21, dWAPL and dCTCF

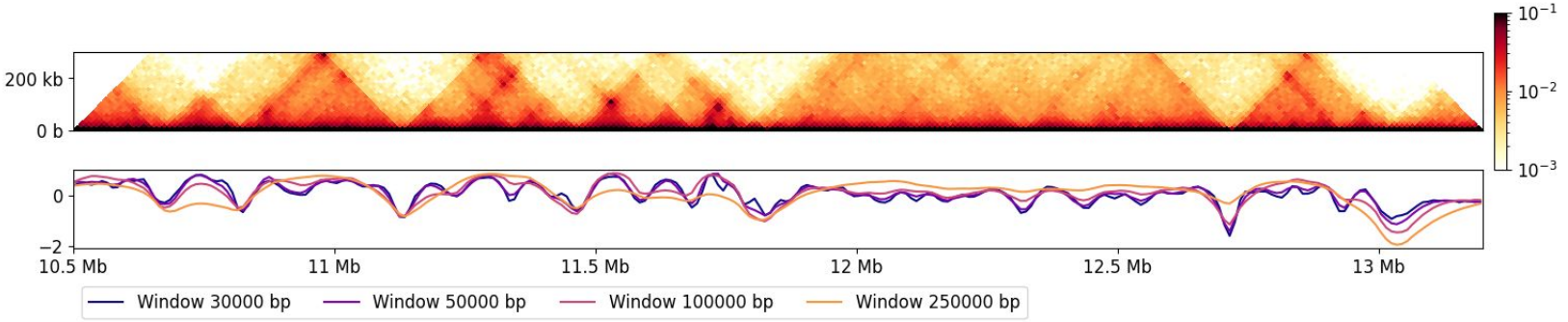


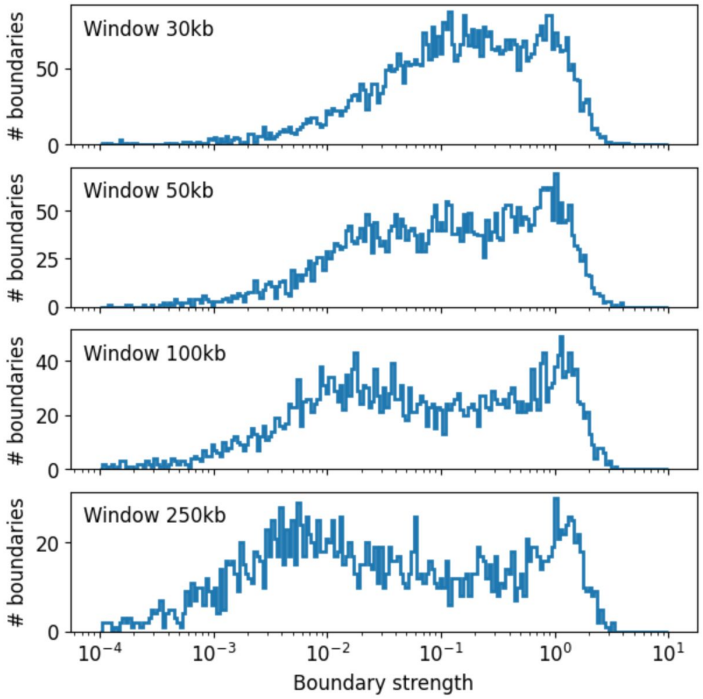

```
region = 'chr12:30,000,000-35,000,000'
chrom, start, end = bioframe.parse_region_string(region)
```

Insulation


a
separation, bp
300 kb
200 kb
100 kb
0 b
10^{-1}
10^{-2}
10^{-3}
contact frequency
b
log2 insulation
0.5
0.0
−0.5
Filtered boundaries
Strong boundaries
11 Mb
11.1 Mb
11.2 Mb
11.3 Mb
11.4 Mb
11.5 Mb
genomic position

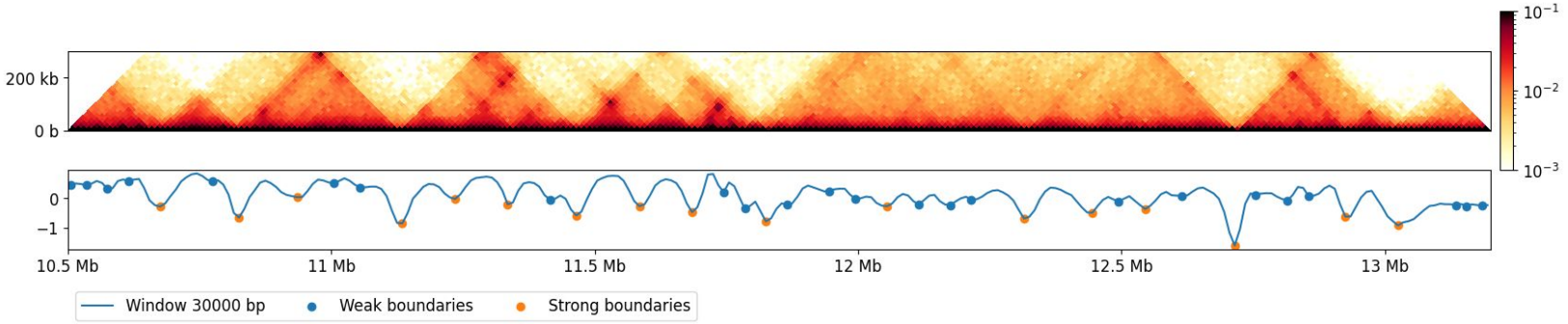
Boundary strength: strong vs weak insulation

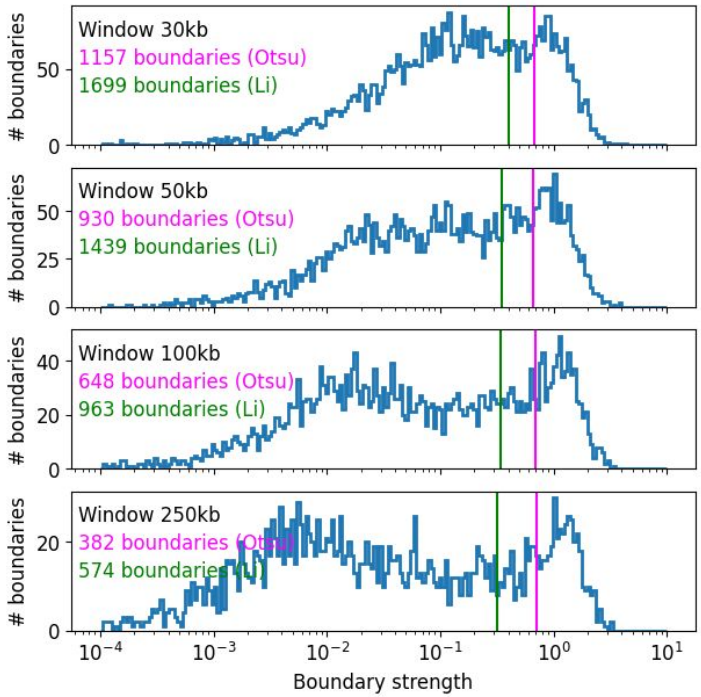




Based on Open2C example: https://github.com/open2c/open2c_examples/blob/master/insulation_and_boundaries.ipynb

Boundary strength: strong vs weak insulation





Based on Open2C example: https://github.com/open2c/open2c_examples/blob/master/insulation_and_boundaries.ipynb

Cooltools.insulation:
run insulation score and boundary calling at the same time

cooltools.api.insulation.insulation(*clr, window_bp, view_df=None, ignore_diags=None, clr_weight_name='weight', min_frac_valid_pixels=0.66, min_dist_bad_bin=0, threshold='Li', append_raw_scores=False, chunksize=20000000, verbose=False, nproc=1*) [source]

Find insulating boundaries in a contact map via the diamond insulation score.

For a given cooler, this function (a) calculates the diamond insulation score track, (b) detects all insulating boundaries, and (c) removes weak boundaries via an automated thresholding algorithm.

Parameters

- **clr** (*cooler.Cooler*) – A cooler with balanced Hi-C data.
- **window_bp** (*int or list of integers*) – The size of the sliding diamond window used to calculate the insulation score. If a list is provided, then a insulation score if done for each value of window_bp.
- **view_df** (*bioframe.viewframe or None*) – Viewframe for independent calculation of insulation scores for regions
- **ignore_diags** (*int | None*) – The number of diagonals to ignore. If None, equals the number of diagonals ignored during IC balancing.
- **clr_weight_name** (*str*) – Name of the column in the bin table with weight
- **min_frac_valid_pixels** (*float*) – The minimal fraction of valid pixels in a diamond to be used in boundary picking and prominence calculation.
- **min_dist_bad_bin** (*int*) – The minimal allowed distance to a bad bin to report insulation score. Fills bins that have a bad bin closer than this distance by nans.
- **threshold** (*"Li", "Otsu" or float*) – Rule used to threshold the histogram of boundary strengths to exclude weak boundaries. “Li” or “Otsu” use corresponding methods from skimage.thresholding. Providing a float value will filter by a fixed threshold
- **append_raw_scores** (*bool*) – If True, append columns with raw scores (sum_counts, sum_balanced, n_pixels) to the output table.
- **verbose** (*bool*) – If True, report real-time progress.
- **nproc** (*int, optional*) – How many processes to use for calculation

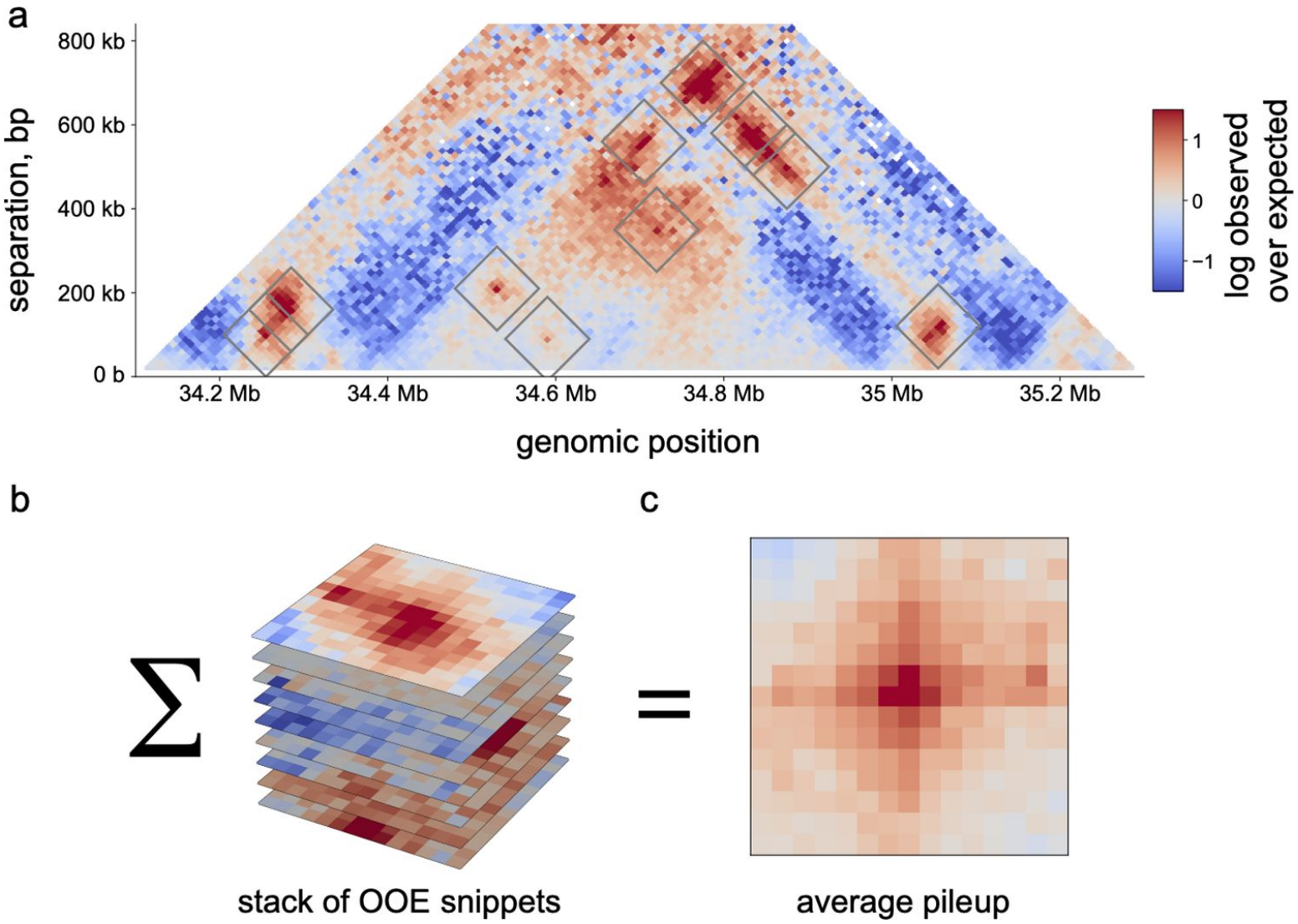
Returns

ins_table (*pandas.DataFrame*) – A table containing the insulation scores of the genomic bins

Control insulation score calculation

Control boundary calling

Pileup plot: averaging Hi-C snippets



Cooltools.pileup: create on-diagonal and off-diagonal pileups

cooltools.api.snipping.pileup(*clr, features_df, view_df=None, expected_df=None, expected_value_col='balanced.avg', flank=100000, min_diag='auto', clr_weight_name='weight', nproc=1, map_functor=<class 'map'>*) [source]

Pileup features over the cooler.

Parameters

- **clr** (*cooler.Cooler*) – Cooler with Hi-C data
- **features_df** (*pd.DataFrame*) – Dataframe in bed or bedpe format: has to have 'chrom', 'start', 'end' or 'chrom1', 'start1', 'end1', 'chrom2', 'start2', 'end2' columns.
- **view_df** (*pd.DataFrame*) – Dataframe with the genomic view for this operation (has to match the expected_df, if provided)
- **expected_df** (*pd.DataFrame*) – Dataframe with the expected level of interactions at different genomic separations
- **expected_value_col** (*str*) – Name of the column in expected used for normalizing.
- **flank** (*int*) – How much to flank the center of the features by, in bp
- **min_diag** (*str or int*) – All diagonals of the matrix below this value are ignored. 'auto' tries to extract the value used during the matrix balancing, if it fails defaults to 2
- **clr_weight_name** (*str*) – Value of the column that contains the balancing weights
- **force** (*bool*) – Allows start>end in the features (not implemented)
- **nproc** (*int, optional*) – How many processes to use for calculation. Ignored if map_functor is passed.
- **map_functor** (*callable, optional*) – Map function to dispatch the matrix chunks to workers. If left unspecified, pool_decorator applies the following defaults: if nproc>1 this defaults to multiprocess.Pool; If nproc=1 this defaults the builtin map.

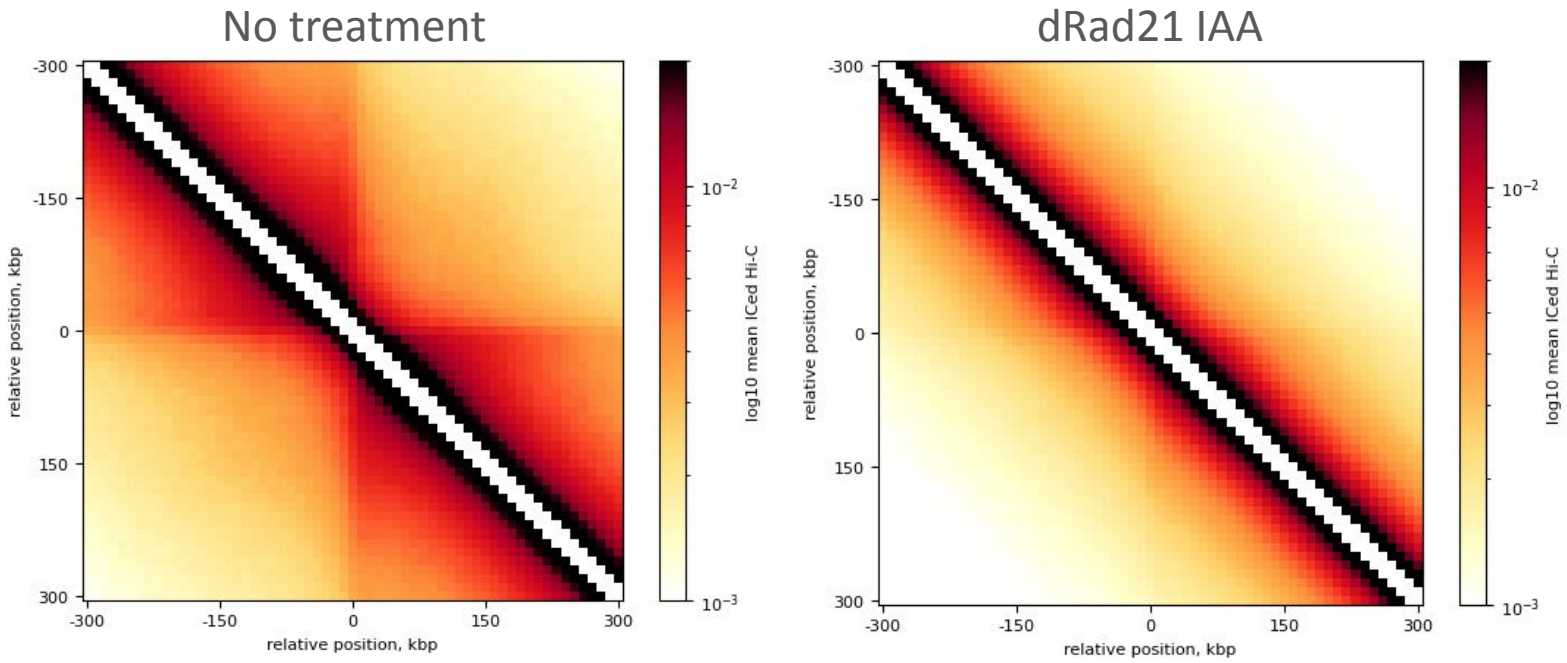
Returns

- **np.ndarray** (*a stackup of all snippets corresponding to the features, with shape*)
- *(n, D, D), where n is the number of snippets and (D, D) is the shape of each*
- *snippet*

Control on- or off-diagonal pileups by providing anchor positions in BED (on-diag) or BEDPE (off-diag) format

Control normalization: raw or observed over expected data

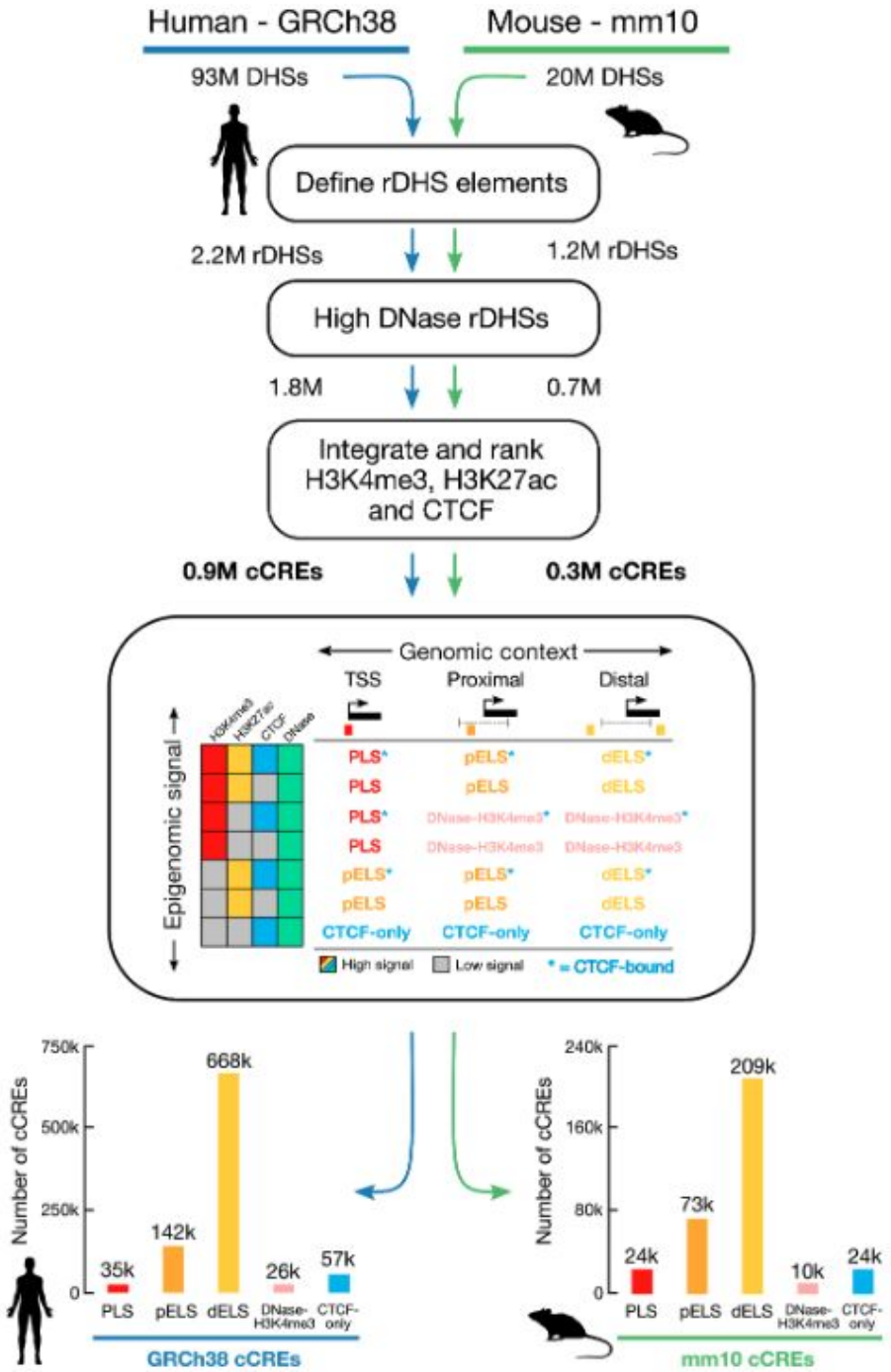
Practice task 3.1. Insulation at strong boundaries before and after dRad21 degradation



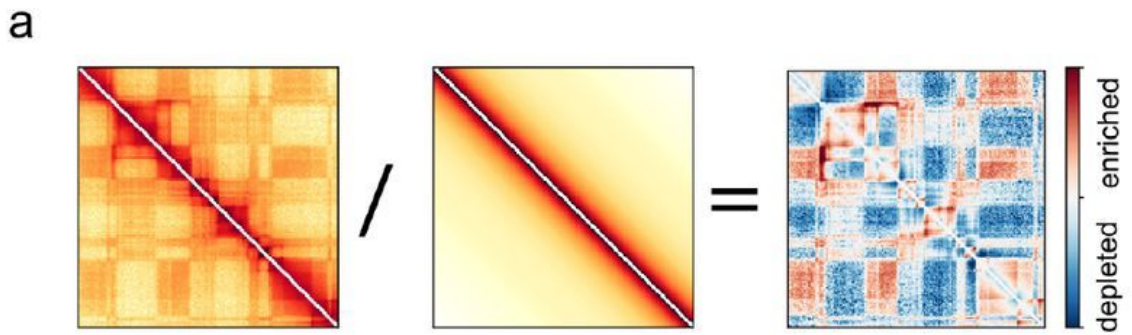
More examples of pileups (including raw/OOE, on- and off-diagonal):
https://github.com/open2c/open2c_examples/blob/master/pileup_CTCF.ipynb

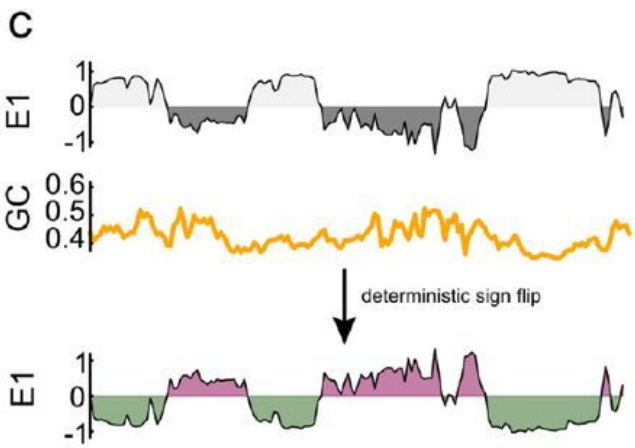
Cis-regulatory elements as defined by SCREEN:

https://screen.encodeproject.org/



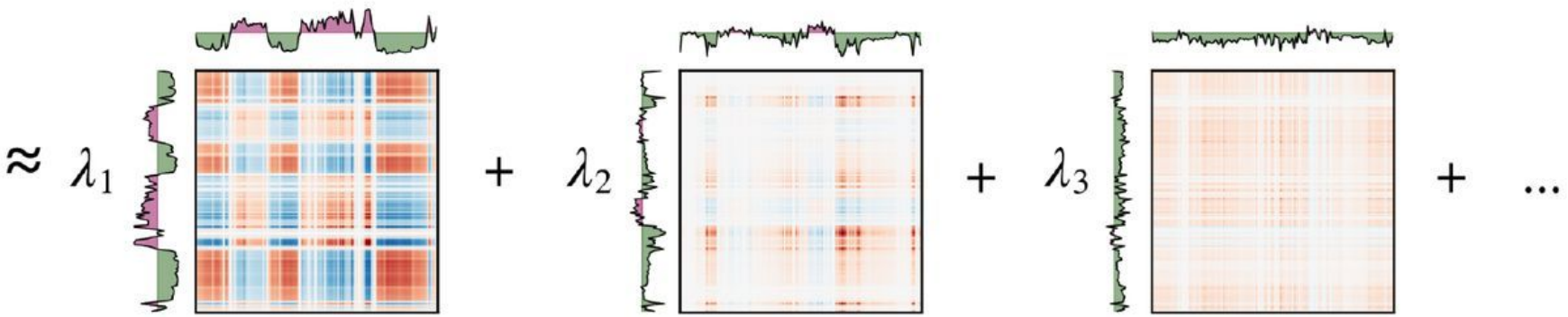
Compartment calling





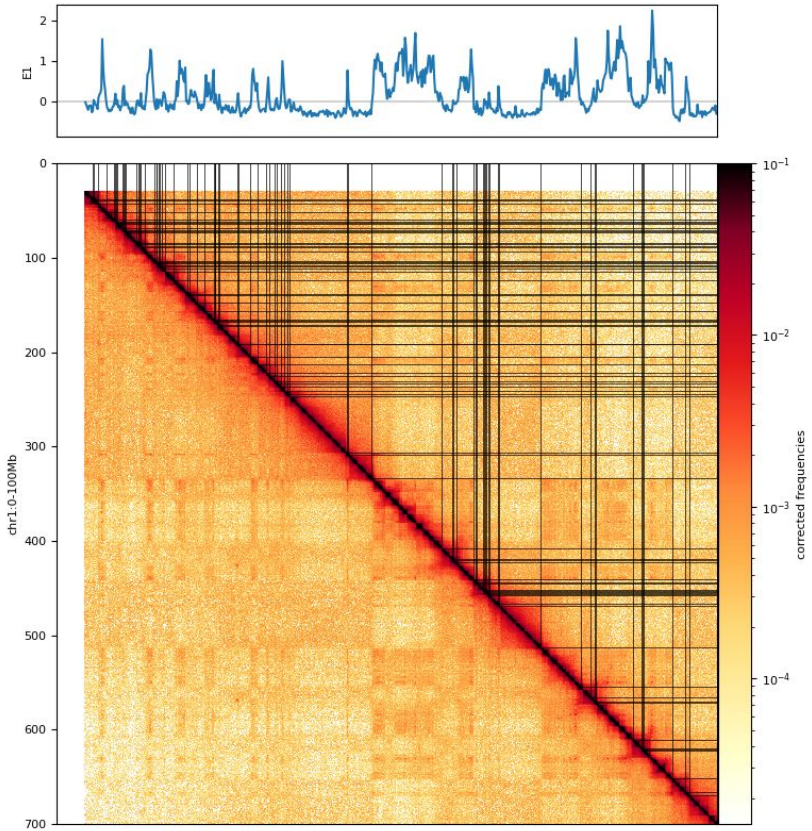
b

$$\approx \lambda_1 e_1 e_1^T + \lambda_2 e_2 e_2^T + \lambda_3 e_3 e_3^T + \ldots$$

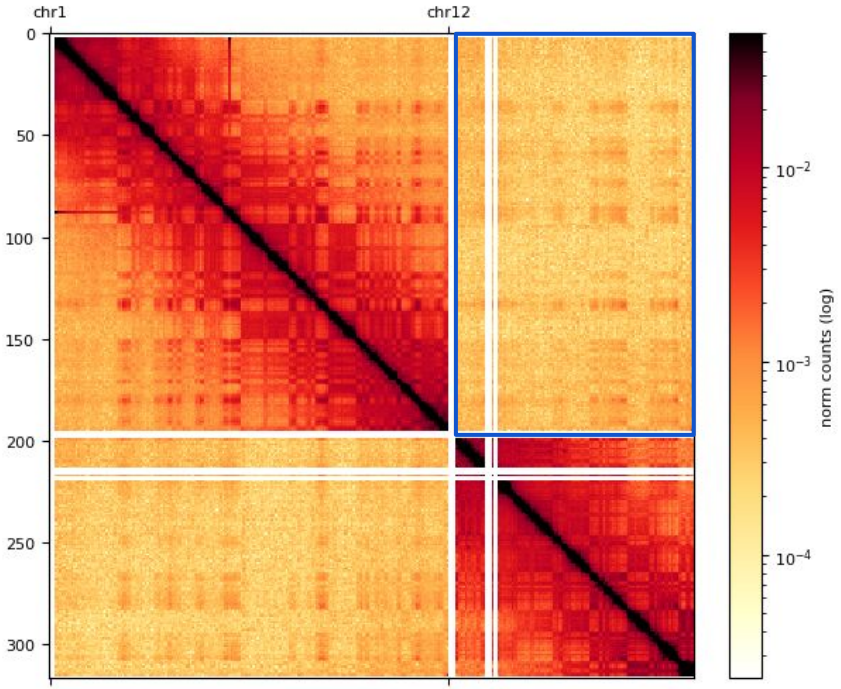


Cis vs Trans-compartments

Cis-only



Trans-only (blue box)



Practice task 3.2. Compartments in Rad21 degradation

```
resolution = 100_000
clr_100k_rad21 = cooler.Cooler(f"{cool_file_rad21}::resolutions/{resolution}")
```

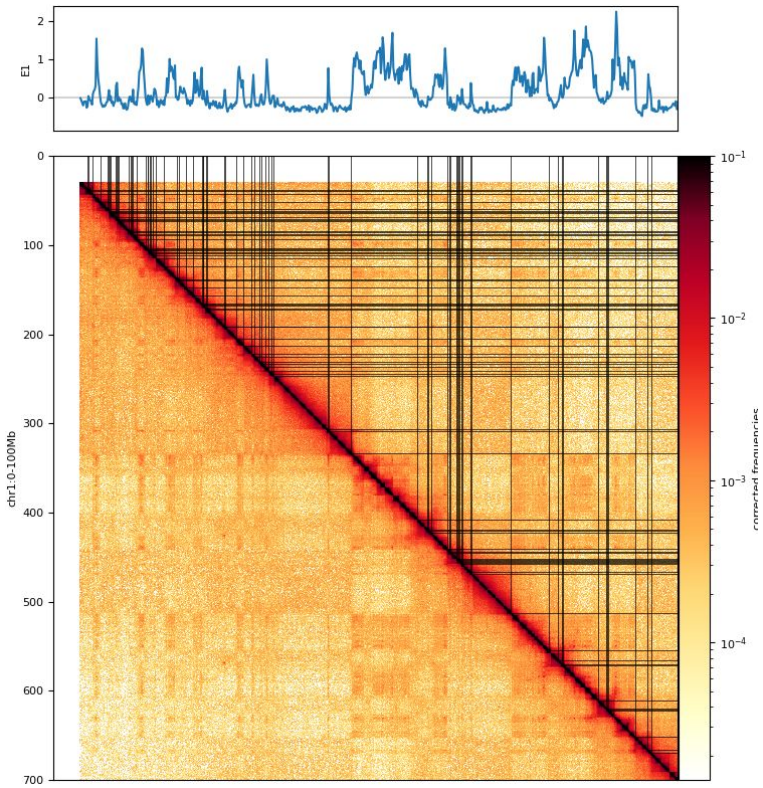
```
# obtain first 3 eigenvectors
cis_eigs = cooltools.eigs_cis(clr_100k_rad21,
                        gc_cov,
                        view_df=view_df,
                        n_eigs=3,
                        )

# cis_eigs[0] returns eigenvalues, here we focus on eigenvectors
eigenvector_track = cis_eigs[1][['chrom', 'start', 'end', 'E1']]
```

```
from matplotlib.colors import LogNorm
from mpl_toolkits.axes_grid1 import make_axes_locatable

f, ax = plt.subplots(
    figsize=(15, 10),
)

norm = LogNorm(vmax=0.1)

im = ax.matshow(
    clr_100k.matrix()[:],
    norm=norm,
    cmap='fall'
);
plt.axis([0, 700, 700, 0])

divider = make_axes_locatable(ax)
cax = divider.append_axes("right", size="5%", pad=0.01)
plt.colorbar(im, cax=cax, label='corrected frequencies');
ax.set_ylabel('chr1:0-100Mb')
ax.xaxis.set_visible(False)

ax1 = divider.append_axes("top", size="20%", pad=0.25, sharex=ax)
weights = clr_100k.bins()[:]['weight'].values
ax1.plot([0, 700], [0, 0], 'k',lw=0.25)
ax1.plot( eigenvector_track['E1'].values, label='E1')

ax1.set_ylabel('E1')
ax1.set_xticks([]);


for i in np.where(np.diff( (cis_eigs[1]['E1']>0).astype(int)))[0]:
    ax.plot([i, 700], [i, i], 'k', lw=0.5)
    ax.plot([i, i], [0, i], 'k', lw=0.5)
```

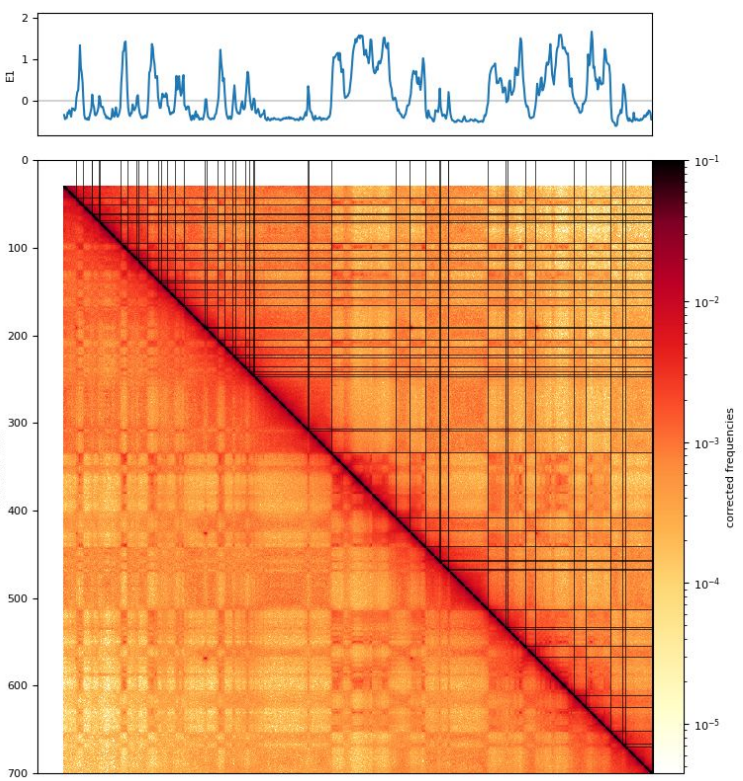
Practice task 3.2. Compare untreated and dRad21 IAA

- Compartments become more clear and less noisy
- Compartments have approximately the same positions

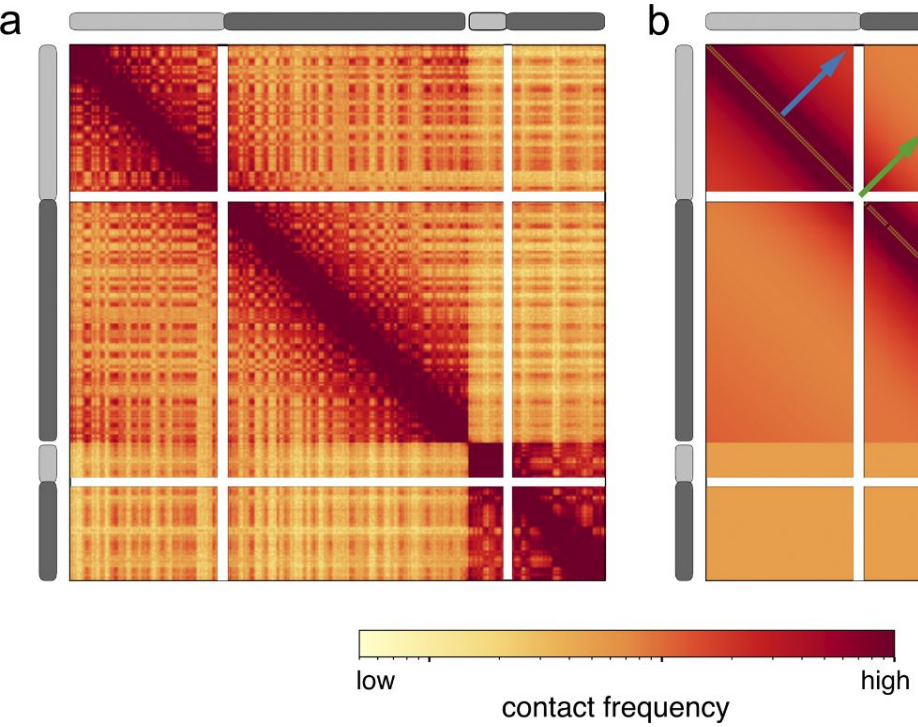
Untreated mESC

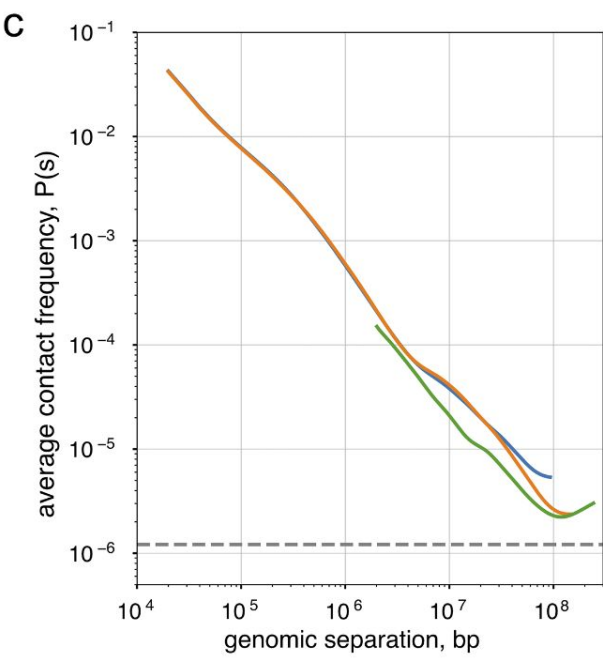


dRad21 IAA-treated mESC



Scaling plot (contact probability vs distance)





cis-scaling: left chromosome arm

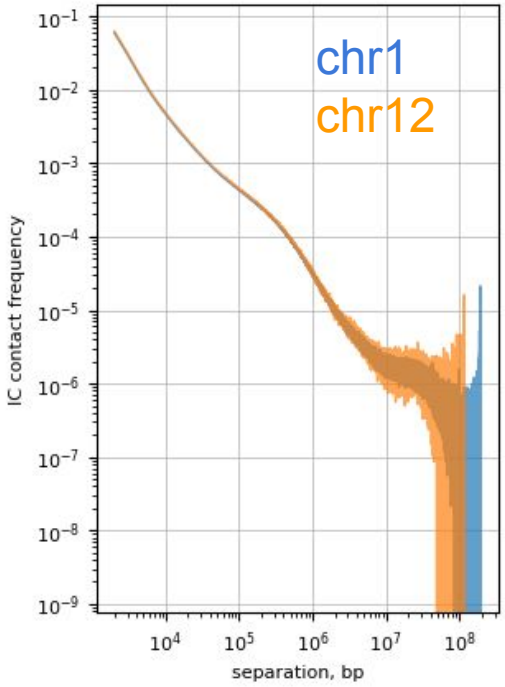
cis-scaling: right chromosome arm

inter-arm scaling

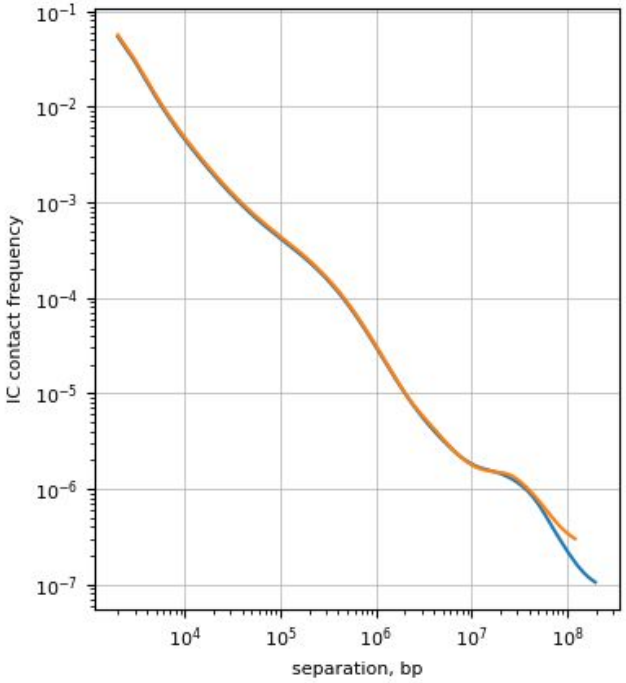
trans-baseline

Scaling plot (contact probability vs distance)

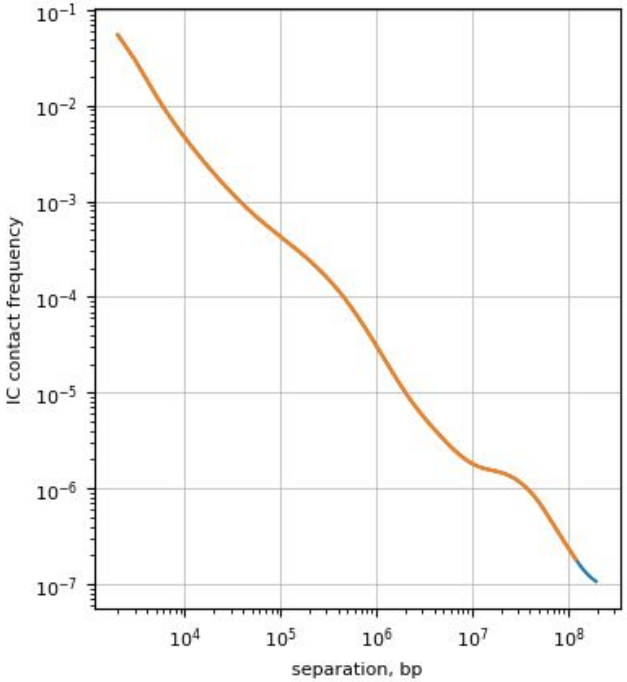
Non-smoothed and non-aggregated:



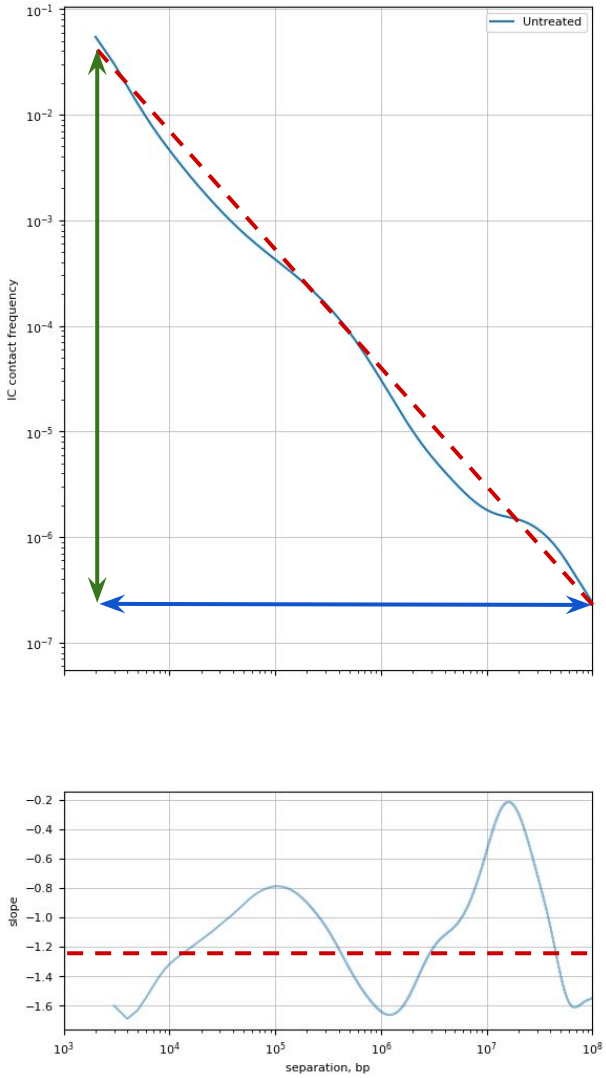
Smoothed and non-aggregated:



Smoothed and aggregated:



Derivative of P(s) curve:



Average slope of Hi-C

contact frequency drop

genomic distance range

P(s) and its derivative for Rad21 and Wapl degrons: