SpatialDM Documentation

Release 0.1.0

Zhuoxuan Li

Jun 30, 2023

MAIN

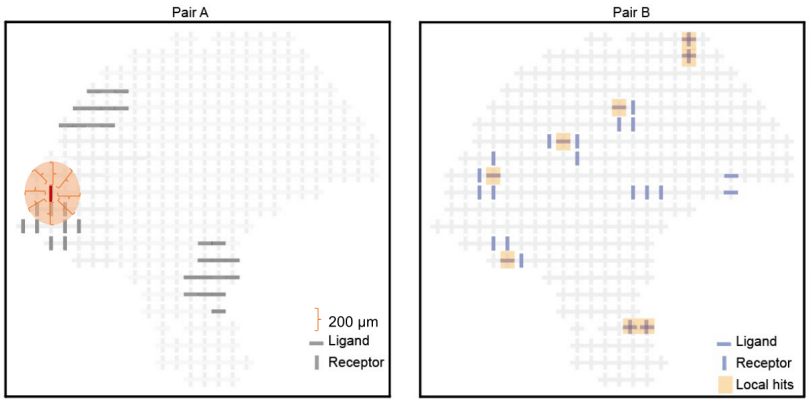
CHAPTER

ONE

ABOUT SPATIALDM

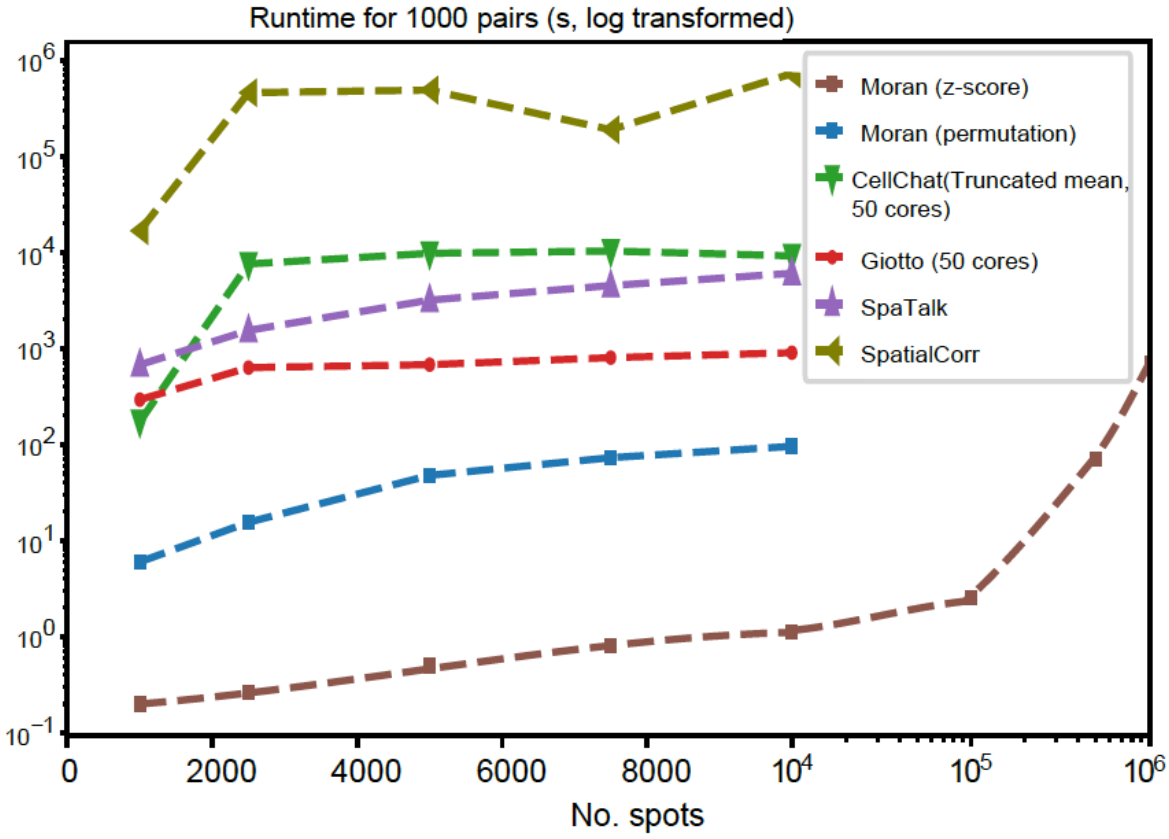
SpatialDM (Spatial Direct Messaging, or Spatial co-expressed ligand and receptor Detected by Moran's bivariant extension) is a statistical model and toolbox to identify the spatial co-expression (i.e., spatial association) between a pair of ligand and receptor.

Uniquely, SpatialDM can distinguish co-expressed ligand and receptor pairs from spatially separating pairs, and identify the spots of interaction.



Traditional approaches	☑	Omitted due to cluster-level non-specificity / low expression
SpatialDM	Non-significant due to spatial range	✅

With the analytical testing method, SpatialDM is scalable to 1 million spots within 12 min with only one core.



SpatialDM comprises two main steps:

1) global selection with `spatialdm_global` to identify significantly interacting LR pairs;
2) local selection with `spatialdm_local` to identify local spots for each interaction.

Please refer to our tutorials for details:

- *Permutation-based SpatialDM (Recommended for small datasets, <10k spots).*
- *Differential analyses of whole interactome among varying conditions.*

CHAPTER

TWO

REFERENCES

SpatialDM manuscript with more details is available on bioRxiv now and is currently under review.

2.1 Installation

SpatialDM is available through PyPI. To install, type the following command line and add `-U` for updates:

```
pip install -U SpatialDM
```

Alternatively, you can install from this GitHub repository for latest (often development) version by the following command line:

```
pip install -U git+https://github.com/leeyoyohku/SpatialDM
```

Installation time: < 1 min

2.2 API

Import SpatialDM as::
    import spatialdm as sdm

2.2.1 Datasets

The *spatialdm.datasets* module provides functions for loading and accessing spatial transcriptomics datasets. The following datasets are currently available:

- *dataset.melanoma()*: Sample 1 rep 2 human melanoma slide from Thrane's melanoma dataset.
- *dataset.SVZ()*: Mouse sub-ventricular zone (SVZ) from Eng's seqfish+ dataset.
- *dataset.A1()*: Adult colon with colorectal cancer or IBD, pcw: Adult.
- *dataset.A2()*: Adult colon with colorectal cancer or IBD, pcw: Adult.
- *dataset.A3()*: Fetal colon, pcw: 12PCW.
- *dataset.A4()*: Fetal colon, pcw: 19PCW.
- *dataset.A6()*: Fetal small intestine, pcw: 12PCW.
- *dataset.A7()*: Fetal small intestine, pcw: 12PCW.
- *dataset.A8()*: Fetal small intestine, pcw: 12PCW.

- *dataset.A9()*: Fetal small intestine, pcw: 12PCW.

2.2.2 Usage

To use the *spatialdm.datasets* module, simply import it as follows:

```
from spatialdm.datasets import dataset
```

Then, you can load a dataset using the corresponding function. For example, to load the melanoma dataset:

```
adata = dataset.melanoma()
```

This will return an *anndata* object containing the expression data for the melanoma dataset in *.X*, the cell type decomposition values in *.obs*, and the spatial coordinates in *.obsm['spatial']*.

2.3 Release History

2.3.1 TODO

- support cache for downloaded h5ad data
- rename local_permI to local_permI_L (also global)
- tentatively: make new adata with LR genes only (if raw exist, other make raw)

2.3.2 Version 0.1.0 (15/04/2023)

- Minor fix with supporting sparse matrix for *adata.X*
- Disabled the output of local permutaiton data in notebook
- More efficient KNN graph construction, with obsp elements into sparse matrix
- Added LR data into package
- Minor fix concat_obj() function
- Minor updates on notebooks: SpatialDE limited one CPU; diff-test only z-score
- Suggest adding *adata.obsm['celltypes']* dataframe to replace *adata.obs*

2.3.3 Version 0.0.8 (14/03/2023)

- SpatialDM wrapped into AnnData object, fixed typos

2.3.4 Version 0.0.1 (11/08/2022)

- Alpha version of SpatialDM released

2.4 Permutation-based selection in the melanoma data

[1]:
```
import os
import pandas as pd
import numpy as np
import anndata as ann

import spatialdm as sdm
from spatialdm.datasets import dataset
import spatialdm.plottings as pl

import matplotlib.pyplot as plt
print("SpatailDM version: %s" %sdm.__version__)
```

Data type cannot be displayed: application/javascript, application/vnd.holoviews_load.v0+json

Data type cannot be displayed: application/javascript, application/vnd.holoviews_load.v0+json

```
SpatailDM version: 0.0.7
```

The melanoma dataset from Thrane, et al. was publicly available, and we obtained raw counts, pre-processed log counts, and spatial coordinates from Matt Stone (Git repository). For easier reuse, we included them in an anndata object which can be loaded directly in SpatialDM Python package.

[3]:
```
adata = dataset.melanoma()
```

```
/home/yoyo/miniconda2/envs/CC/lib/python3.9/site-packages/anndata/_core/anndata.py:121:␣
↪ImplicitModificationWarning: Transforming to str index.
  warnings.warn("Transforming to str index.", ImplicitModificationWarning)
```

The anndata object contains the following:
log-transformed expression in `adata.X`,
raw expression in `adata.raw`,
cell types computed by RCTD in `adata.obs`,
and spatial coordinates in `adata.obsm['spatial']`

2.4.1 SpatialDM object and preprocessing

We generate a suitable weight matrix for the SpatialDM object at first. Here considering the scale of the spatial coordinates and spot-spot distance (200 micrometers here), we set radial basis kernel parameter l = 1.2, and trimmed all weights < 0.2 (cutoff) to match the normal range of CCC (200 micrometers, 1 spot away from the sender cell here)

Note Alternative to the cutoff parameter, we can set n_neighbors to around 8 to restrain the range of CCC.

[5]:
```
sdm.weight_matrix(adata, l=1.2, cutoff=0.2, single_cell=False) # weight_matrix by rbf␣
↪kernel
```

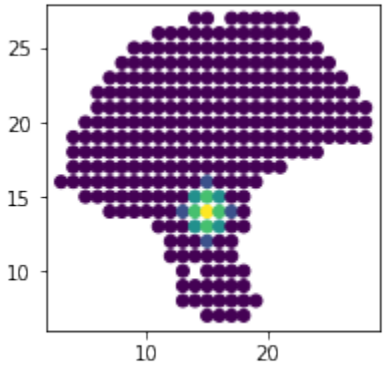
```
/home/yoyo/miniconda2/envs/CC/lib/python3.9/site-packages/sklearn/utils/validation.py:70:
↪ FutureWarning: Pass n_neighbors=6 as keyword args. From version 1.0 (renaming of 0.
↪25) passing these as positional arguments will result in an error
  warnings.warn(f"Pass {args_msg} as keyword args. From version "
```

Note

If the scale of the spatial coordinates is larger (e.g. several thousand in the intestinal example) or smaller, or when the spot-spot distance varies (which is common for different platforms), the `l` can be very different. It's a crucial step to determine a suitable `l` to match the biological context. We recommend the following plotting to check the resulting weight matrix from the previous step. If needed, users can run `weight_matrix` iteratively to decide the most optimal `l` and `cutoff`.

[6]:
```
# visualize range of interaction
import matplotlib.pyplot as plt
plt.figure(figsize=(3,3))

# plt.scatter(adata.obsm['spatial'][:,0], adata.obsm['spatial'][:,1],
#             c=adata.obsp['weight'][50])

plt.scatter(adata.obsm['spatial'][:,0], adata.obsm['spatial'][:,1],
            c=adata.obsp['weight'].A[50])
```

[6]:
```
<matplotlib.collections.PathCollection at 0x7fd4b64386d0>
```



Next step is to extract valid LR pairs from the database (by default, we use CellChatDB). Filtering out sparsely expressed ligand or receptor (e.g. 3) can help us get more interpretable results later.

[7]:
```
sdm.extract_lr(adata, 'human', min_cell=3)      # find overlapping LRs from CellChatDB
```

2.4.2 Global and local selection (permutation approach)

It is a crucial step to identify dataset-specific interacting LR pairs (global selection) for ensuring quality analysis and reliable interpretation of the putative CCC. With high computation efficiency, SpatialDM can complete global selection in seconds.

[13]:
```
import time
start = time.time()
sdm.spatialdm_global(adata, 1000, specified_ind=None, method='both', nproc=1)     #␣
↪global Moran selection
sdm.sig_pairs(adata, method='permutation', fdr=True, threshold=0.1)     # select␣
↪significant pairs
print("%.3f seconds" %(time.time()-start))
```

```
100%|| 1000/1000 [00:06<00:00, 155.27it/s]
```

```
8.346 seconds
```

We used fdr corrected global p-values and a threshold FDR < 0.1 (default) to determine which pairs to be included in the following local identification steps. There are 103 pairs being selected in this data.

[14]:
```
print(adata.uns['global_res'].selected.sum())
adata.uns['global_res'].sort_values(by='fdr').head()
```

```
103
```

[14]:

	Ligand0	Ligand1	Ligand2	Receptor0	Receptor1	Receptor2	z_pval	z	perm_pval	fdr	selected
CSPG4_ITGA2_ITGB1	CSPG4	None	None	ITGA3	ITGB1	None	1.372040e-06	4.689101	0.0	0.0	True
PECAM1_PECAM1	PECAM1	None	None	PECAM1	None	None	2.792191e-52	15.170039	0.0	0.0	True
SEMA3D_NRP2_PLXNA1	SEMA3D	None	None	NRP2	PLXNA1	None	1.267594e-04	3.658679	0.0	0.0	True
HLA-C_CD8A	HLA-C	None	None	CD8A	None	None	7.035725e-07	4.823990	0.0	0.0	True
COL6A1_ITGA1_ITGB1	COL6A1	None	None	ITGA1	ITGB1	None	8.586635e-06	4.298790	0.0	0.0	True

Local selection is then carried out for the selected 103 pairs to identify where the LRI take place, in a single-spot resolution.

[15]:
```
start = time.time()
sdm.spatialdm_local(adata, n_perm=1000, method='both', specified_ind=None, nproc=1)
↪# local spot selection
sdm.sig_spots(adata, method='permutation', fdr=False, threshold=0.1)     # significant␣
↪local spots
print("%.3f seconds" %(time.time()-start))
```

```
/home/yoyo/miniconda2/envs/CC/lib/python3.9/site-packages/anndata/_core/raw.py:139:␣
↪FutureWarning: X.dtype being converted to np.float32 from int64. In the next version␣
↪of anndata (0.9) conversion will not be automatic. Pass dtype explicitly to avoid this␣
↪warning. Pass `AnnData(X, dtype=X.dtype, ...)` to get the future behavour.
  return anndata.AnnData(
100%|| 1000/1000 [00:00<00:00, 3934.37it/s]
100%|| 1000/1000 [00:00<00:00, 4645.87it/s]
```

```
1.115 seconds
```

By default, local selection is performed for all selected pairs in the previous step.

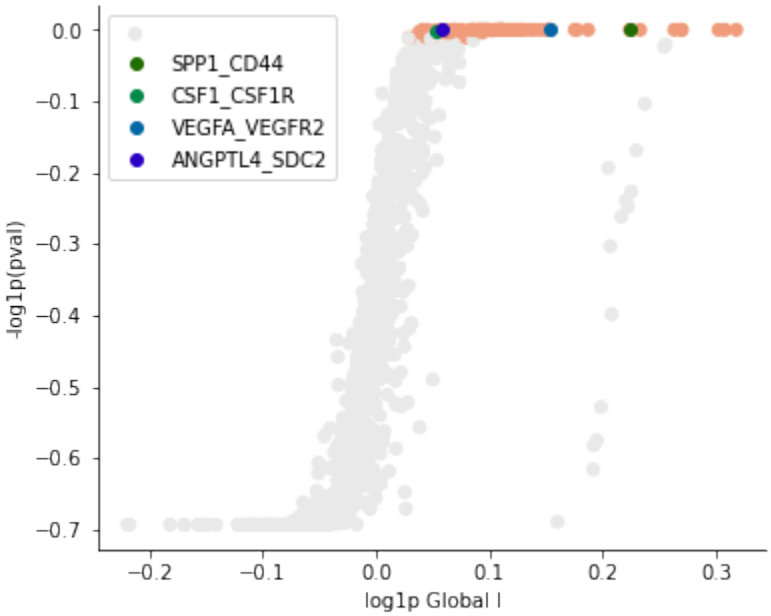
Note Apply an array of integer indices of selected pairs to `select_num` to run local selection in selected pairs

The global and local results are easily accessible through `global_res` and `local_perm_p` or `local_z_p`

2.4.3 Visualize pair(s)

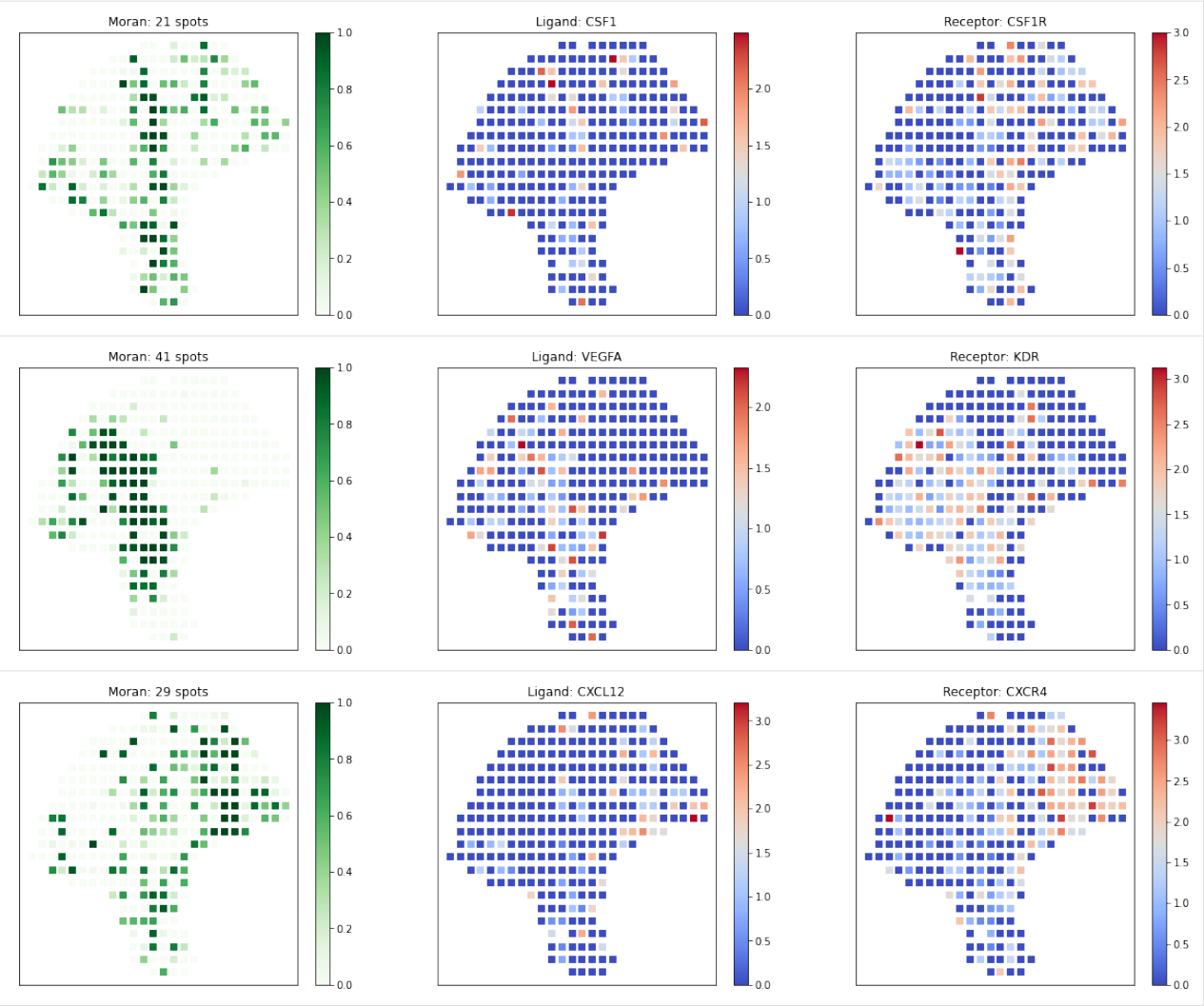
SpatialDM provides plotting utilities for general summary of global results:

[17]:
```
pl.global_plot(adata, pairs=['SPP1_CD44', 'CSF1_CSF1R', 'VEGFA_VEGFR2', 'ANGPTL4_SDC2'],
               figsize=(6,5), cmap='RdGy_r', vmin=-1.5, vmax=2)
```



Known melanoma-related pairs were all observed in the selected pairs. It would then be meaningful to investigate the spatial context of their interactions.

[18]:
```
# visualize known melanoma pairs
pl.plot_pairs(adata, ['CSF1_CSF1R', 'VEGFA_VEGFR2', 'CXCL12_CXCR4'], marker='s')
```



2.4.4 Spatial Clustering of Local Spots

SpatialDE allows clustering of spatially auto-correlated genes. Here, we repurposed SpatialDE to identify spatially auto-correlated interactions by using binary local selection status as input.

[19]:
```
import NaiveDE
import SpatialDE
```

Filter out sparse interactions with fewer than 3 identified interacting spots. Cluster into 6 patterns.

[20]:
```
# SpatialDE code
bin_spots = adata.uns['selected_spots'].astype(int)[adata.uns['local_stat']['n_spots']>2]
print(bin_spots.shape[0], " pairs used for spatial clustering")
```

```
103  pairs used for spatial clustering
```

[21]:
```
from threadpoolctl import threadpool_limits
```

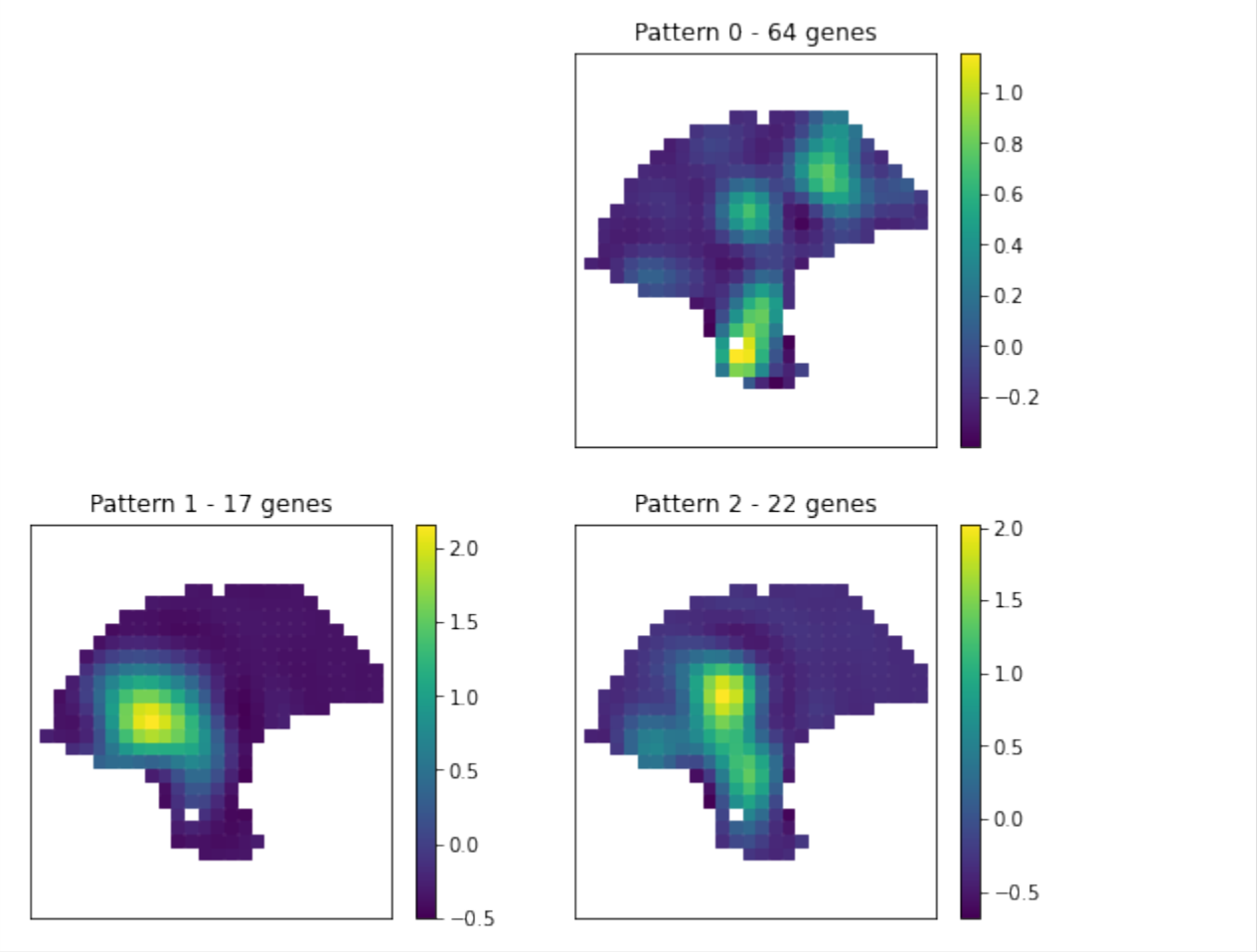
(continues on next page)

(continued from previous page)

```
with threadpool_limits(limits=1, user_api='blas'):
    results = SpatialDE.run(adata.obsm['spatial'], bin_spots.transpose())

    histology_results, patterns = SpatialDE.aeh.spatial_patterns(
        adata.obsm['spatial'], bin_spots.transpose(), results, C=3, l=3, verbosity=1)
```

```
Models:   0%|          | 0/10 [00:00<?, ?it/s]
  0%|          | 0/103 [00:00<?, ?it/s]
  0%|          | 0/103 [00:00<?, ?it/s]
  0%|          | 0/103 [00:00<?, ?it/s]
  0%|          | 0/103 [00:00<?, ?it/s]
  0%|          | 0/103 [00:00<?, ?it/s]
  0%|          | 0/103 [00:00<?, ?it/s]
  0%|          | 0/103 [00:00<?, ?it/s]
  0%|          | 0/103 [00:00<?, ?it/s]
  0%|          | 0/103 [00:00<?, ?it/s]
  0%|          | 0/103 [00:00<?, ?it/s]
iter 0, ELBO: -2.99e+10
iter 1, ELBO: -1.57e+10, delta_ELBO: 1.42e+10
iter 2, ELBO: -1.57e+10, delta_ELBO: 4.65e+03
iter 3, ELBO: -1.57e+10, delta_ELBO: 1.29e+03
iter 4, ELBO: -1.57e+10, delta_ELBO: 3.45e+00
iter 5, ELBO: -1.57e+10, delta_ELBO: 4.57e+00
iter 6, ELBO: -1.57e+10, delta_ELBO: 1.31e+00
iter 7, ELBO: -1.57e+10, delta_ELBO: 1.17e+00
iter 8, ELBO: -1.57e+10, delta_ELBO: 3.26e+01
iter 9, ELBO: -1.57e+10, delta_ELBO: 2.56e+01
iter 10, ELBO: -1.57e+10, delta_ELBO: 2.93e-01
iter 11, ELBO: -1.57e+10, delta_ELBO: 4.65e-04
Converged on iter 11
```

[22]:
```
plt.figure(figsize=(9,8))
for i in range(3):
    plt.subplot(2, 2, i + 2)
    plt.scatter(adata.obsm['spatial'][:,0], adata.obsm['spatial'][:,1], marker = 's',
↪c=patterns[i], s=35);
    plt.axis('equal')
    pl.plt_util('Pattern {} - {} genes'.format(i, histology_results.query('pattern == @i
↪').shape[0] ))
plt.savefig('mel_DE_clusters.pdf')
```



Note For detail parameter settings like `C` and `l`, please refer to SpatialDE tutorial

SpatialDM provides `compute_pathway` function to group a list of interactions based on the pathways they belong. The input can be a dictionary of several lists.

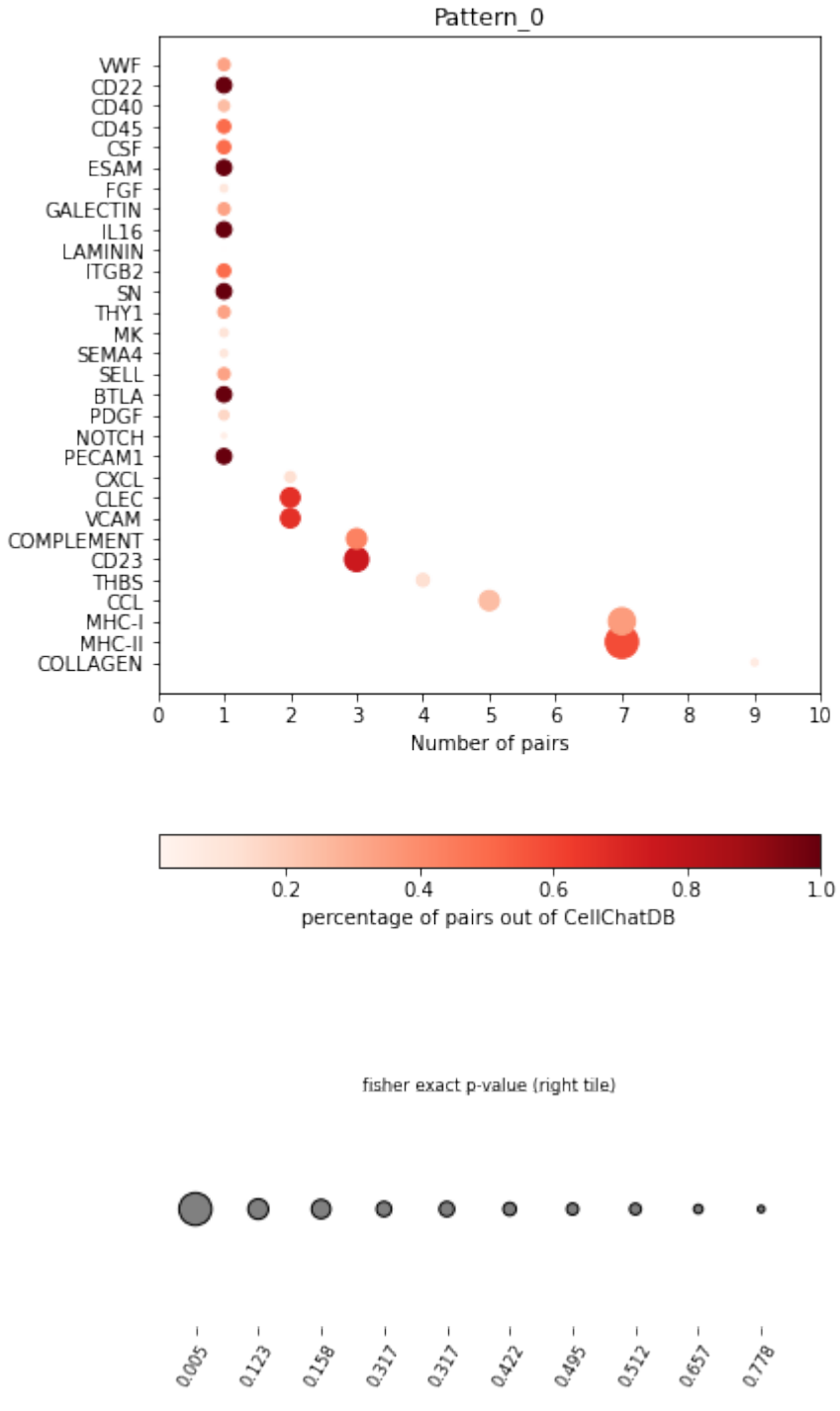
[23]:
```
dic=dict()
for i in histology_results.sort_values('pattern').pattern.unique():
    dic['Pattern_{}'.format(i)]=histology_results.query('pattern == @i').sort_values(
↪'membership')['g'].values
```

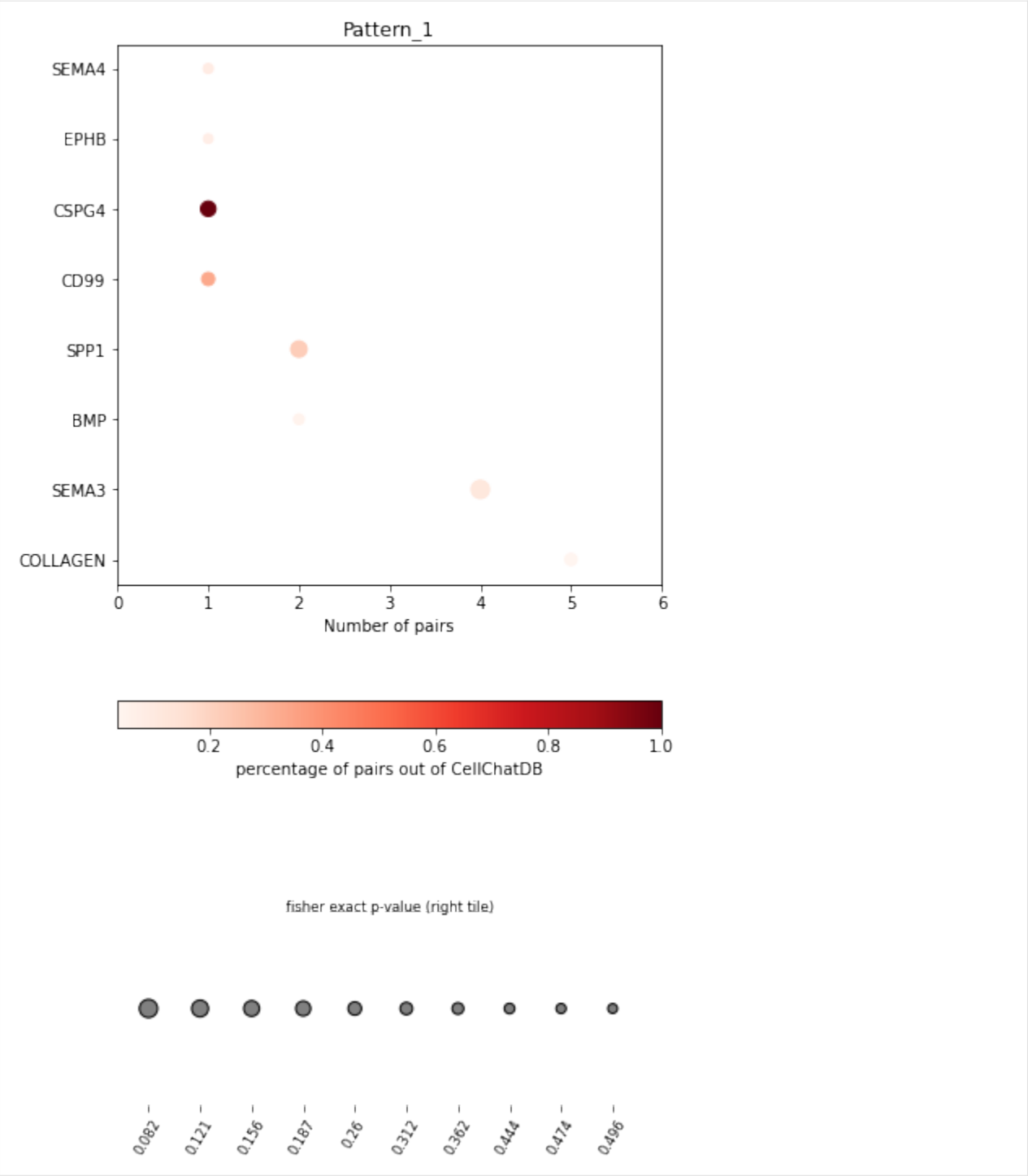
Save the anndata file to retrieve later

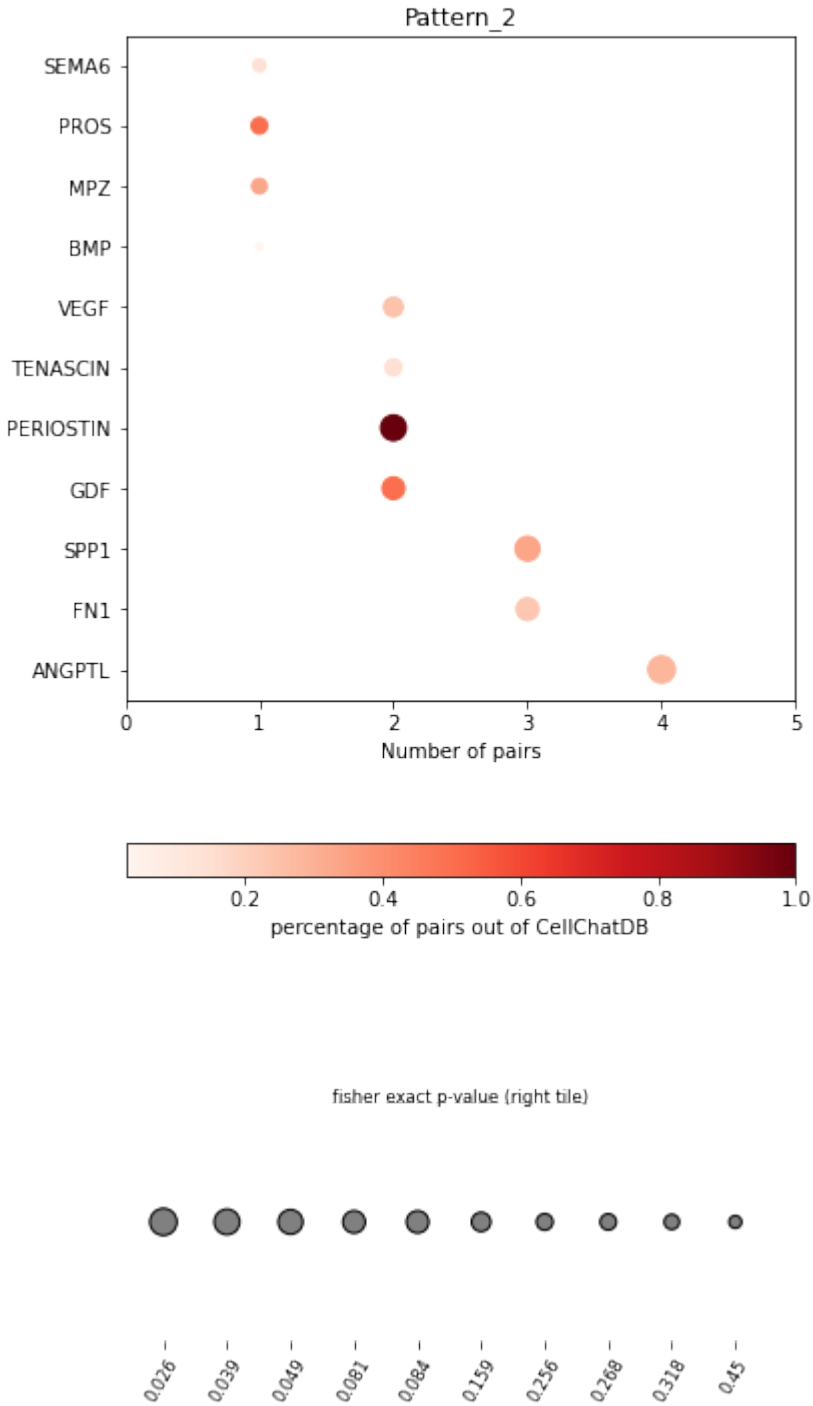
[25]:
```
adata_out = adata.copy()
adata_out.uns['local_stat']['local_permI'] = 'None'
adata_out.uns['local_stat']['local_permI_R'] = 'None'

sdm.write_spatialdm_h5ad(adata_out, filename='mel_sdm_out.h5ad')
# sdm.read_spatialdm_h5ad(filename='mel_sdm_out.h5ad')
```

In the dot plot of the lymphoid-associated pattern, we identified CD23 and other immune-related pathways.

[28]:
```
pl.dot_path(adata, dic=dic, figsize=(6,12))
```




Pattern_1
SEMA4
EPHB
CSPG4
CD99
SPP1
BMP
SEMA3
COLLAGEN
0
1
2
3
4
5
6
Number of pairs
0.2
0.4
0.6
0.8
1.0
percentage of pairs out of CellChatDB
fisher exact p-value (right tile)
0.082
0.121
0.156
0.187
0.26
0.312
0.362
0.444
0.474
0.496



SpatialDM provides `chord_celltype` and other chord diagrams to visualize the aggregated cell types (or interactions)

[30]:
```
adata.obsm['celltypes'] = adata.obs[adata.obs.columns]
pl.chord_celltype(adata, pairs=['FCER2A_CR2'], ncol=1, min_quantile=0.01)
# save='mel_FCER2A_CR2_ct.svg'
```

Data type cannot be displayed: application/javascript, application/vnd.bokehjs_exec.v0+json

[30]:
```
Column(id='1110', ...)
```

alternative DE input with Moran R values

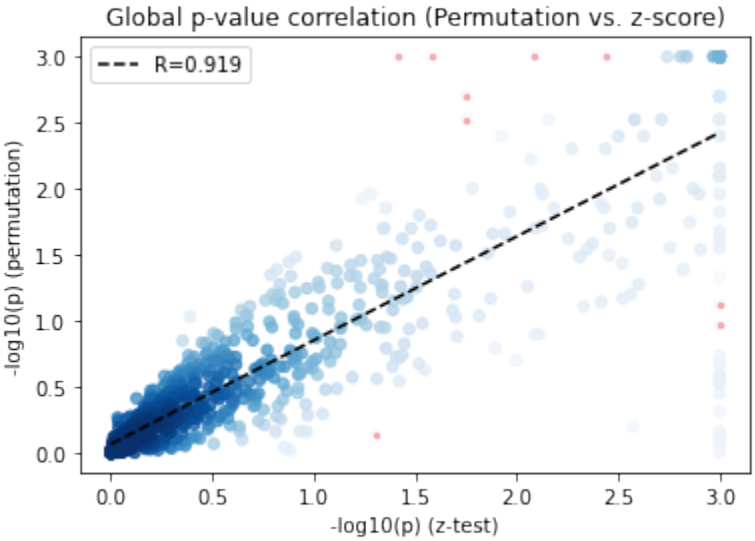
[16]:
```
# local_sum = adata.uns['selected_spots'].copy()
# local_sum=local_sum.transpose()
# local_sum[local_sum.columns] =(adata.uns['local_stat']['local_I'] + adata.uns['local_stat
↪']['local_I'])

# results = SpatialDE.run(adata.obsm['spatial'], local_sum)

# histologyesults, patterns = SpatialDE.aeh.spatial_patterns(adata.obsm['spatial'], local_
↪sum,
#                                                            results, C=3, l=5,
#                                                            verbosity=1)
# plt.figure(figsize=(9,8))
# for i in range(3):
#     plt.subplot(2, 2, i + 1)
#     plt.scatter(adata.obsm['spatial'][:,0], adata.obsm['spatial'][:,1], marker = 's',
↪c=patterns[i], s=35);
#     plt.axis('equal')
#     pl.plt_util('Pattern {} - {} genes'.format(i, histologyesults.query('pattern == @i').
↪shape[0] ))
```

2.4.5 Consistency between permutation and z-score approaches

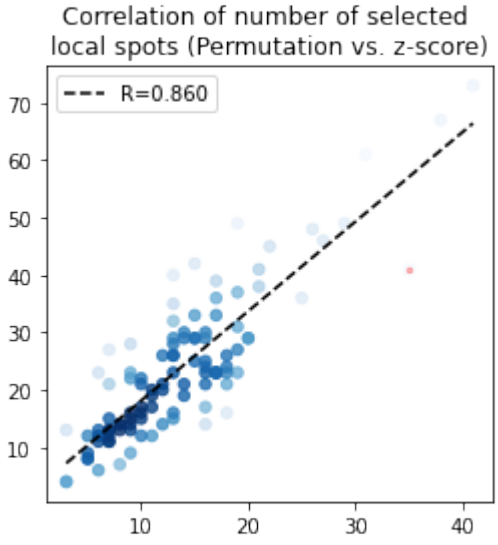
As an alternative to permutation method, we derived the first and second moments of the null distribution to analytically obtain a z-score and its according p-values, which are highly consistent with the permutation method.

[31]:
```
# global consistency
x = -np.log10(adata.uns['global_res'].z_pval.values)
y = -np.log10(adata.uns['global_res'].perm_pval)

x = np.where(x>3, 3, x)
y = np.where(np.isinf(y), 3, y)
pl.corr_plot(x, y, method='spearman')
plt.xlabel('-log10(p) (z-test)')
plt.ylabel('-log10(p) (permutation)')
plt.title('Global p-value correlation (Permutation vs. z-score)')
plt.savefig('mel_perm_z_corr.pdf')
```

```
/home/yoyo/miniconda2/envs/CC/lib/python3.9/site-packages/pandas/core/arraylike.py:364:␣
↪RuntimeWarning: divide by zero encountered in log10
  result = getattr(ufunc, method)(*inputs, **kwargs)
```



[32]:
```
# local consistency
plt.figure(figsize=(4,4))
x=(adata.uns['local_z_p']<0.1).sum(1).values
y=(adata.uns['local_perm_p']<0.1).sum(1).values
pl.corr_plot(x, y, method='pearson')
plt.title('Correlation of number of selected \nlocal spots (Permutation vs. z-score)')
plt.savefig('mel_local_cor.pdf')
```



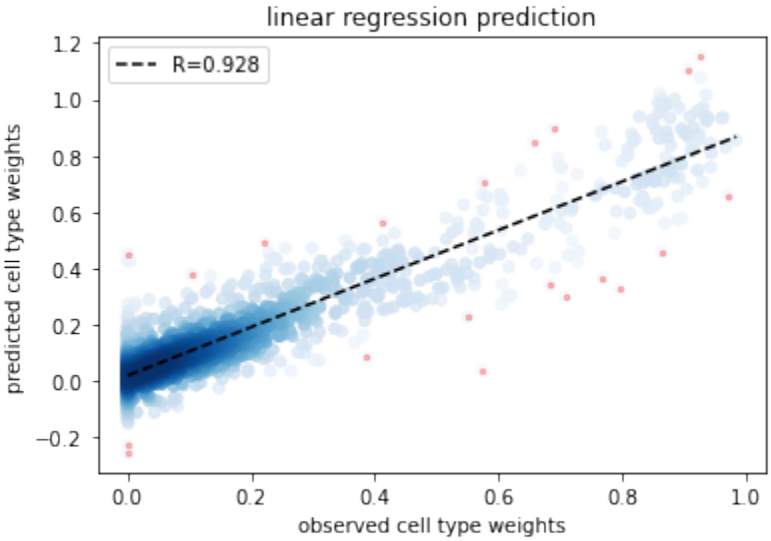
2.4.6 Cell type prediction with local Moran

[34]:
```
from sklearn.linear_model import LinearRegression
```

Note The prediction results may differ by different annotation accuracy and potentially other factor. Here we generated the cell type decomposition results using RCTD.

[35]:
```
X = adata.uns['local_perm_p'].values
y=adata.obs.values

reg = LinearRegression().fit(X.T, y)
y_pred = reg.predict(X.T)
pl.corr_plot(np.hstack(y),np.hstack(y_pred), method='pearson')
plt.xlabel('observed cell type weights')
plt.ylabel('predicted cell type weights')
plt.title('linear regression prediction')
# plt.savefig('linear_regression_celltype.pdf')
```

[35]:
```
Text(0.5, 1.0, 'linear regression prediction')
```



2.5 Differential analyses (z-score approach)

[1]:
```
import os
import pandas as pd
import numpy as np
import anndata as ann

import spatialdm as sdm
from spatialdm.datasets import dataset
import spatialdm.plottings as pl
print("SpatailDM version: %s" %sdm.__version__)
```

Data type cannot be displayed: application/javascript, application/vnd.holoviews_load.v0+json

Data type cannot be displayed: application/javascript, application/vnd.holoviews_load.v0+json

```
SpatailDM version: 0.0.7
```

2.5.1 z-score selection in batch

[2]:
```
data=["A1","A2","A3","A4","A6","A7","A8","A9"]
```

The intestine dataset from Corbett, et al. was publicly available on STAR-FINDer, from which we obtained raw counts and spatial coordinates, and log-transformed to normalize on the spot-level. Rawcounts (`.raw`), logcounts (`.X`), cell types (`.obs`), and spatial coordinates (`.obsm['spatial']`) have been included in the corresponding anndata object which can be loaded directly in SpatialDM Python package.

[4]:
```
A1 = dataset.A1()
A2 = dataset.A2()
A3 = dataset.A3()
A4 = dataset.A4()
A6 = dataset.A6()
A7 = dataset.A7()
A8 = dataset.A8()
A9 = dataset.A9()
```

Note

The dataset includes:
1) 2 adult colon samples from the same donor A, with IBD or colon cancer (A1, A2);
2) 2 replicates of 12-PCW TI from fetus donor D (A6, A7);
3) 3 12-PCW colon samples, 2 replicates from donor D (A8, A9), another from donor B (A3);
4) 1 19-PCW colon sample from donor C (A4).

---

[5]:
```
samples = [A1,A2,A3,A4,A6,A7,A8,A9]
```

Considering the scale of the spatial coordinates and spot-spot distance of 100 micrometers, `l` will be set to 75 here. The parameters here should be determined to match the context of CCC.

[9]:
```
for adata in samples:
    adata.obsm['spatial'] = adata.obsm['spatial'].values
    sdm.weight_matrix(adata, l=75, cutoff=0.2, single_cell=False) # weight_matrix by rbf␣
↪kernel
    sdm.extract_lr(adata, 'human', min_cell=3)      # find overlapping LRs from␣
```

(continues on next page)

(continued from previous page)

```
↪CellChatDB
    sdm.spatialdm_global(adata, 1000, specified_ind=None, method='z-score', nproc=1)
↪# global Moran selection
    sdm.sig_pairs(adata, method='z-score', fdr=True, threshold=0.1)     # select␣
↪significant pairs
    sdm.spatialdm_local(adata, n_perm=1000, method='z-score', specified_ind=None,␣
↪nproc=1)     # local spot selection
    sdm.sig_spots(adata, method='z-score', fdr=False, threshold=0.1)
    print(' done!')
```

```
/home/yoyo/miniconda2/envs/CC/lib/python3.9/site-packages/sklearn/utils/validation.py:70:
↪ FutureWarning: Pass n_neighbors=6 as keyword args. From version 1.0 (renaming of 0.
↪25) passing these as positional arguments will result in an error
  warnings.warn(f"Pass {args_msg} as keyword args. From version "
100%|| 1000/1000 [09:45<00:00,  1.71it/s]
/home/yoyo/miniconda2/envs/CC/lib/python3.9/site-packages/spatialdm/utils.py:130:␣
↪RuntimeWarning: invalid value encountered in true_divide
  X = X/X.max()
100%|| 1000/1000 [00:14<00:00, 69.16it/s]
100%|| 1000/1000 [00:06<00:00, 162.48it/s]
```

```
done!
```

filtered non-expressed LR pairs.

Save selection results in batch

2.5.2 differential test on 6 colon samples (adult vs. fetus)

[10]:
```
from spatialdm.diff_utils import *
```

First step is to concatenate all selection results, resulting in a `p_df` storing p-values across each sample before fdr correction, a `tf_df` indicating whether each pair is selected in each sample, and a `zscore_df` which stores z-scores. These will be essential for the differential analyses later.

[11]:
```
concat=concat_obj(samples, data, 'human', 'z-score', fdr=False)
```

```
/home/yoyo/miniconda2/envs/CC/lib/python3.9/site-packages/anndata/_core/anndata.py:1828:␣
↪UserWarning: Observation names are not unique. To make them unique, call `.obs_names_
↪make_unique`.
  utils.warn_names_duplicates("obs")
```

[12]:
```
concat.uns['p_df'].head()
```

[12]:

	A1	A2	A3	A4	A6	A7	A8	A9
VSIR_IGSF11	0.425453	0.395138	0.598898	0.315757	0.599201	0.327638	0.666588	0.760361
EFNA5_EPHA7	0.510958	0.774411	0.096287	0.039663	0.003395	0.006287		
EFNA5_EPHB2	0.306913	0.104032	0.111116	0.000113	0.079636	0.026080		
EFNB1_EPHA4	0.270509	0.002146	0.019308	0.008792	0.002633	0.008091		
EFNB1_EPHB2	0.428187	0.218473	0.994771	0.010403	0.005470	0.001725		

(continues on next page)

(continued from previous page)

```
EFNA5_EPHA7  0.058307  0.411695
EFNA5_EPHB2  0.766981  0.515436
EFNB1_EPHA4  0.357686  0.182043
EFNB1_EPHB2  0.217220  0.498809
```

[13]:
```
concat.uns['tf_df'].head()
```

[13]:
```
                A1     A2     A3     A4     A6     A7     A8     A9
VSIR_IGSF11  False  False  False  False  False  False  False  False
EFNA5_EPHA7  False  False  False   True   True   True  False  False
EFNA5_EPHB2  False  False  False   True  False   True  False  False
EFNB1_EPHA4  False   True   True   True   True   True  False  False
EFNB1_EPHB2  False  False  False   True   True   True  False  False
```

[14]:
```
concat.uns['zscore_df'].head()
```

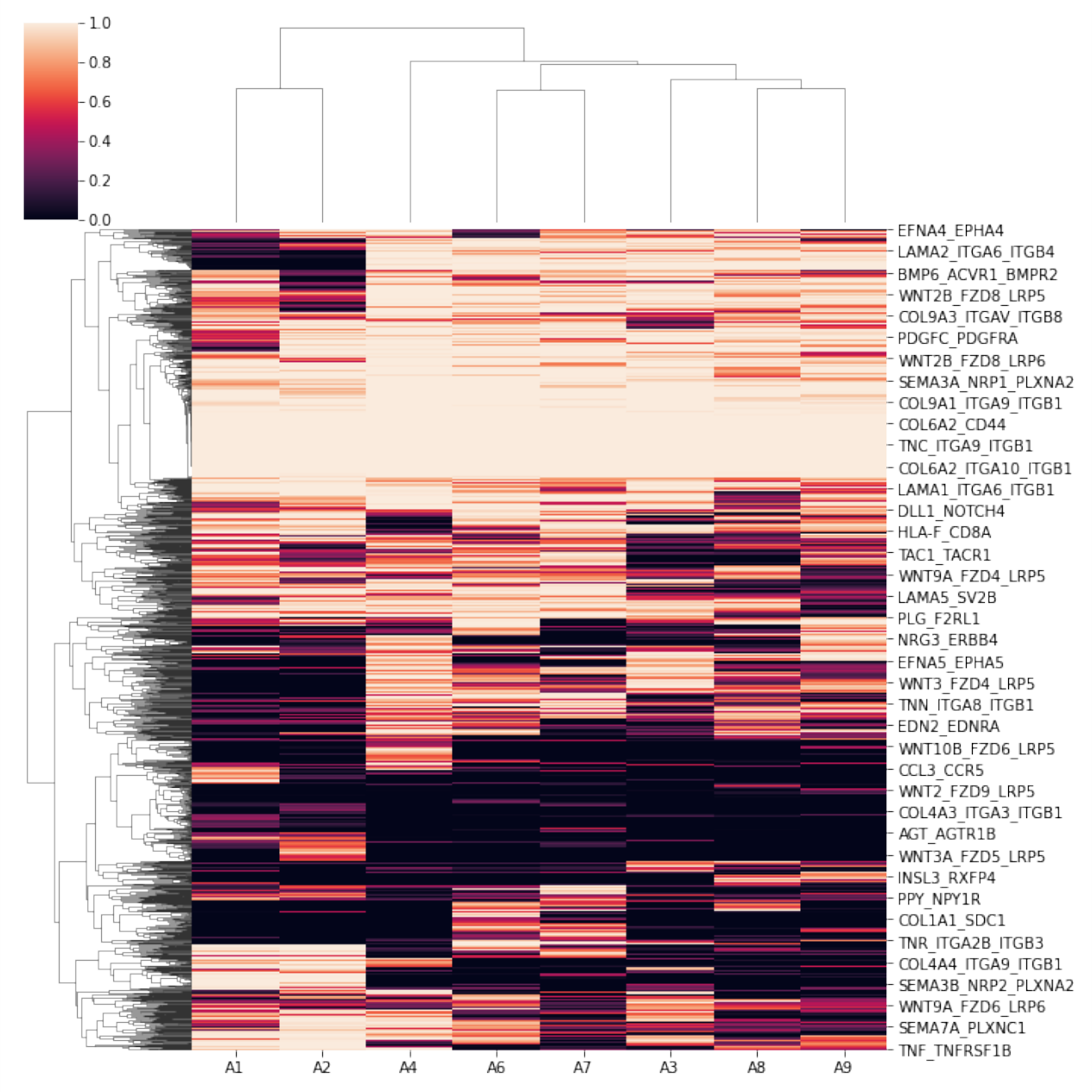
[14]:
```
                  A1        A2        A3        A4        A6        A7  \
VSIR_IGSF11  0.187962  0.265951 -0.250496  0.479598 -0.251280  0.446444
EFNA5_EPHA7 -0.027471 -0.753451  1.303003  1.754614  2.707015  2.495596
EFNA5_EPHB2  0.504619  1.258905  1.220614  3.688469  1.407522  1.941812
EFNB1_EPHA4  0.611274  2.855859  2.068248  2.374266  2.790333  2.404807
EFNB1_EPHB2  0.180993  0.777361 -2.560322  2.311493  2.544579  2.924481

                  A8        A9
VSIR_IGSF11 -0.430511 -0.707466
EFNA5_EPHA7  1.569141  0.223187
EFNA5_EPHB2 -0.728942 -0.038703
EFNB1_EPHA4  0.364652  0.907606
EFNB1_EPHB2  0.781615  0.002986
```

Whole-interactome clustering revealed the dendrogram relationships that are close to the sample kinship

[15]:
```
import seaborn as sns
sns.clustermap(1-concat.uns['p_df'])
```

```
/home/yoyo/miniconda2/envs/CC/lib/python3.9/site-packages/seaborn/matrix.py:649:␣
↪UserWarning: Clustering large matrix with scipy. Installing `fastcluster` may give␣
↪better performance.
  warnings.warn(msg)
```

[15]:
```
<seaborn.matrix.ClusterGrid at 0x7f9c2b7df8e0>
```



Next, we subset the 6 colon samples (2 adult vs. 4 fetus for differential analyses.

Use 1 and 0 to label different conditions.

Note z-score differential testing will also support differential test among 3 or more conditions, provided with sufficient samples.

---

[16]:
```
conditions = np.hstack((np.repeat([1],2), np.repeat([0],4)))
subset = ['A1', 'A2', 'A3', 'A4', 'A8', 'A9']
```

By `differential_test`, likelihood ratio test will be performed, and the differential p-values are stored in `uns['p_val']`, corresponding fdr values in `uns['diff_fdr']`, and difference in average zscores in `uns['diff']`.

[17]:
```
differential_test(concat, subset, conditions)
```

```
/home/yoyo/miniconda2/envs/CC/lib/python3.9/site-packages/statsmodels/regression/linear_
↪model.py:903: RuntimeWarning: divide by zero encountered in log
  llf = -nobs2*np.log(2*np.pi) - nobs2*np.log(ssr / nobs) - nobs2
/home/yoyo/miniconda2/envs/CC/lib/python3.9/site-packages/spatialdm/diff_utils.py:109:␣
↪RuntimeWarning: invalid value encountered in double_scalars
  LR_statistic[i] = -2 * (reduced_ll - full_ll)
```

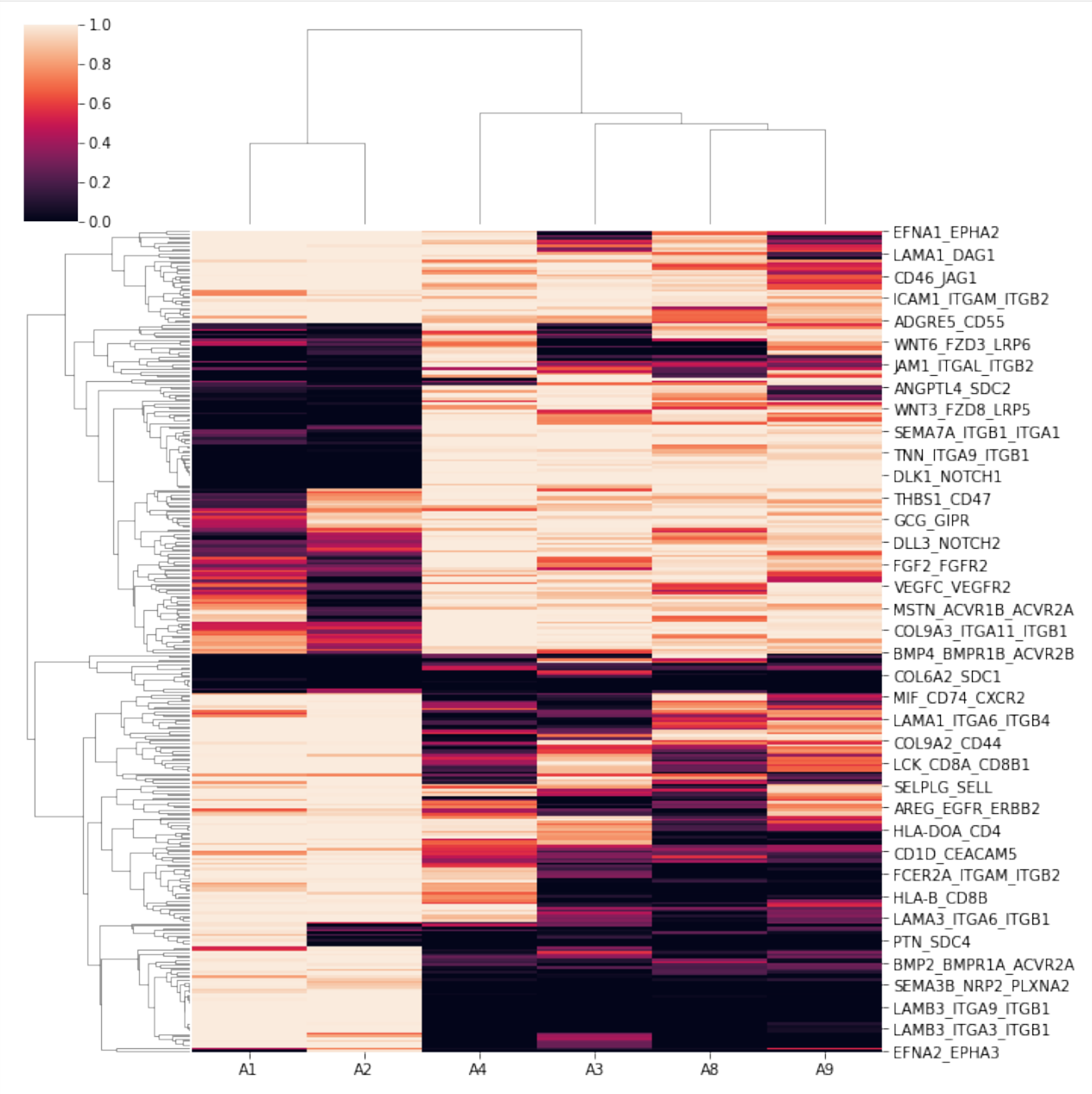
Later, we can focus on differential pairs within 2 contrasting conditions. By default, pairs with z-score difference greater than 30% quantile and a corrected differential significance (`.diff_fdr`) smaller than 0.1 are selected, while different parameters can be applied in `diff_quantile1`, `diff_quantile2`, and `fdr_co`, respectively.

`differential_volcano` allows easy check for target pair(s) in whether they are differential among conditions and in which condition it's specific.

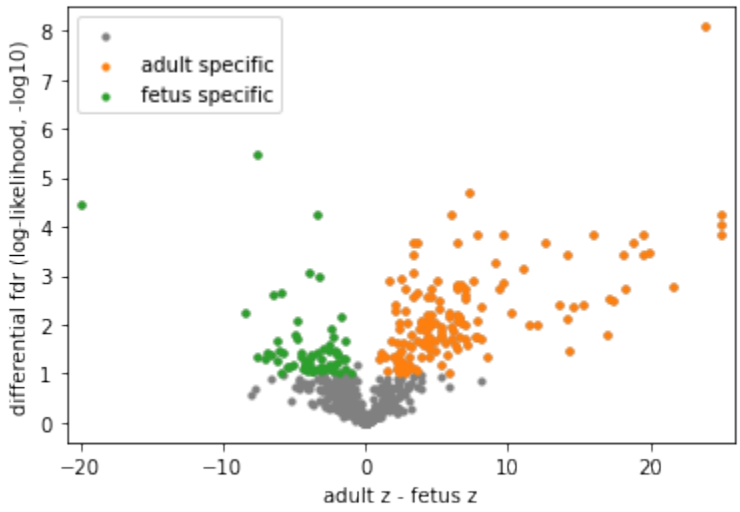
[18]:
```
group_differential_pairs(concat, 'adult', 'fetus')
```

[19]:
```
pl.differential_dendrogram(concat)
```

[19]:
```
<seaborn.matrix.ClusterGrid at 0x7f9871e74f10>
```



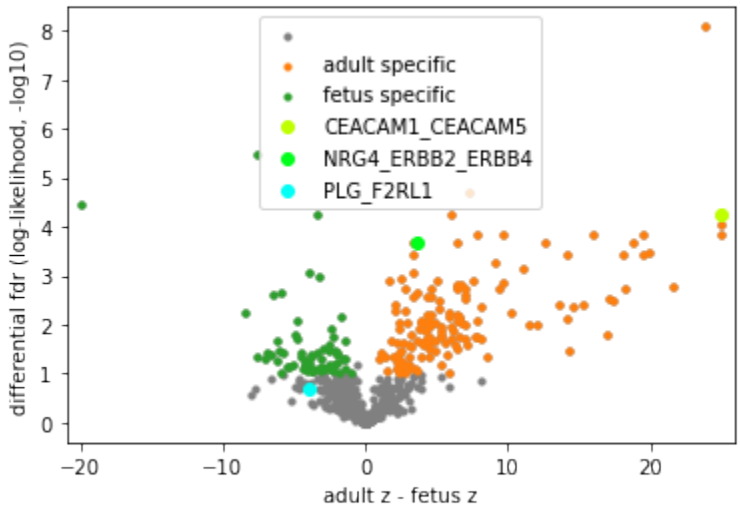
[20]:
```
pl.differential_volcano(concat, legend=['adult specific', 'fetus specific'])
```



Note

CEACAM1_CEACAM5 is sparsely identified in A3 in addition to two adult slices, but still considered adult-specific by differential analyses. NRG4 was found in human breast milk, and its oral supplementation can protect against inflammation in the intestine. PLG_F2RL1 is sparsely and exclusively identified in fetal samples with consistent cell type enrichment. Please refer to our manuscript for more discussions.

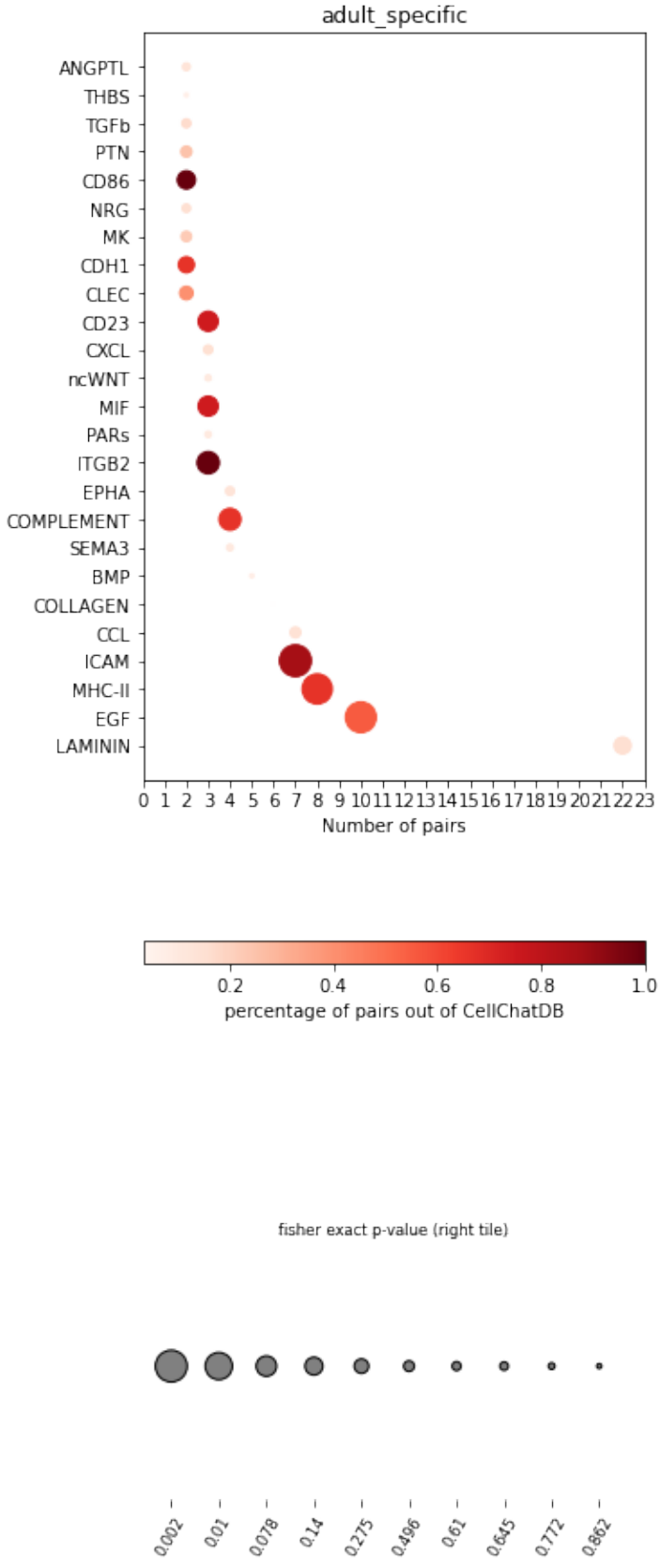
[27]:
```
pl.differential_volcano(concat, pairs=['CEACAM1_CEACAM5', 'NRG4_ERBB2_ERBB4', 'PLG_F2RL1
↪'],
        legend=['adult specific', 'fetus specific'], )
```



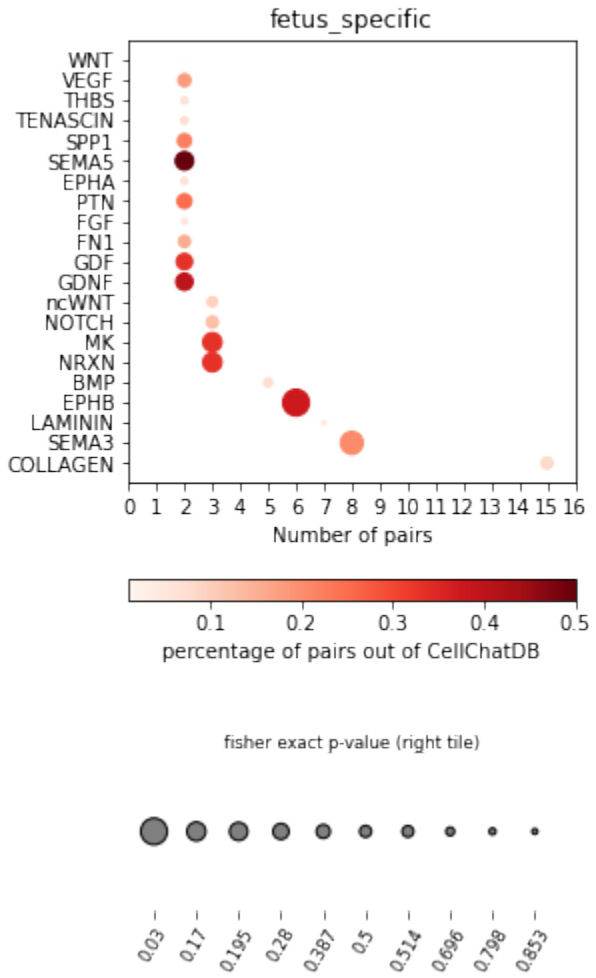
It will also be interpretable to analyze condition-specific pairs on a pathway level.

Here we identified an inflammatory microenvironment in adult colon samples and more development-related signatures in the fetal colon samples.

[25]:
```
pl.dot_path(concat, 'adult_specific', cut_off=2, figsize=(5,15))
```


adult_specific
ANGPTL
THBS
TGFb
PTN
CD86
NRG
MK
CDH1
CLEC
CD23
CXCL
ncWNT
MIF
PARs
ITGB2
EPHA
COMPLEMENT
SEMA3
BMP
COLLAGEN
CCL
ICAM
MHC-II
EGF
LAMININ
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Number of pairs
0.2 0.4 0.6 0.8 1.0
percentage of pairs out of CellChatDB
fisher exact p-value (right tile)
0.002 0.01 0.078 0.14 0.275 0.496 0.61 0.645 0.772 0.862

[26]:
```
pl.dot_path(concat, 'fetus_specific', cut_off=2, figsize=(4,8))
```



EGF pathway is highly enriched in adult samples.
We can use plotting functions to visualize some pairs like TGFA_EGFR and EGF_EGFR

[37]:
```
A1.uns['geneInter'].loc[(A1.uns['geneInter'].pathway_name=='EGF') \
                        &(A1.uns['geneInter'].index.isin(A1.uns['selected_spots'].
↪index))]
```

[37]:

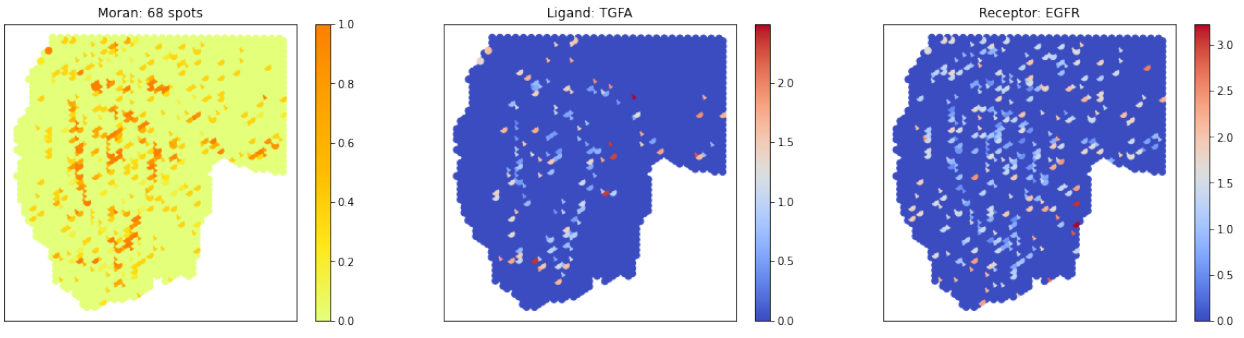
	interaction_name	pathway_name	agonist	antagonist	\
EREG_ERBB2_ERBB4	EREG_ERBB2_ERBB4	EGF	NaN	NaN	
EREG_EGFR_ERBB2	EREG_EGFR_ERBB2	EGF	NaN	NaN	
EREG_EGFR	EREG_EGFR	EGF	NaN	NaN	
HBEGF_ERBB2_ERBB4	HBEGF_ERBB2_ERBB4	EGF	NaN	NaN	
HBEGF_EGFR_ERBB2	HBEGF_EGFR_ERBB2	EGF	NaN	NaN	
HBEGF_EGFR	HBEGF_EGFR	EGF	NaN	NaN	
AREG_EGFR_ERBB2	AREG_EGFR_ERBB2	EGF	NaN	NaN	
AREG_EGFR	AREG_EGFR	EGF	NaN	NaN	

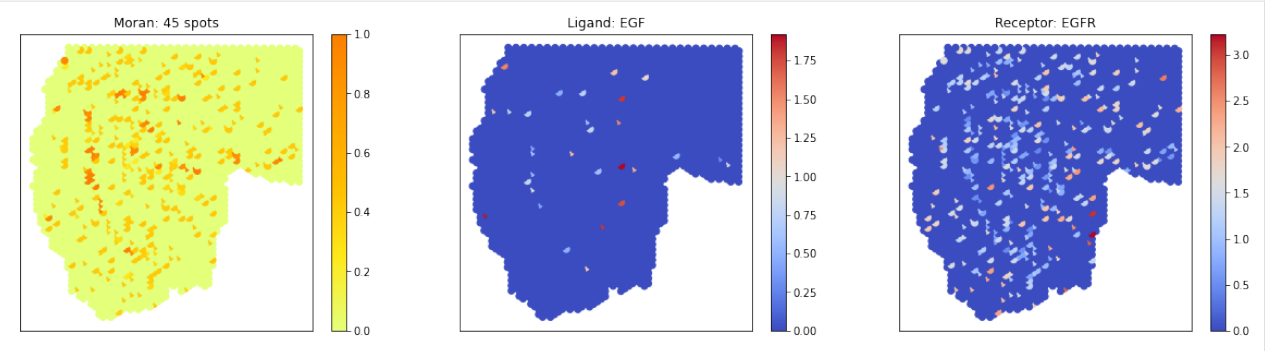
(continues on next page)

(continued from previous page)

```
TGFA_EGFR_ERBB2      TGFA_EGFR_ERBB2          EGF     NaN        NaN
TGFA_EGFR                  TGFA_EGFR          EGF     NaN        NaN
EGF_EGFR_ERBB2        EGF_EGFR_ERBB2          EGF     NaN        NaN
EGF_EGFR                    EGF_EGFR          EGF     NaN        NaN

                  co_A_receptor co_I_receptor        evidence  \
EREG_ERBB2_ERBB4            NaN           NaN  KEGG: hsa04012
EREG_EGFR_ERBB2             NaN           NaN  KEGG: hsa04012
EREG_EGFR                   NaN           NaN  KEGG: hsa04012
HBEGF_ERBB2_ERBB4           NaN           NaN  KEGG: hsa04012
HBEGF_EGFR_ERBB2            NaN           NaN  KEGG: hsa04012
HBEGF_EGFR                  NaN           NaN  KEGG: hsa04012
AREG_EGFR_ERBB2             NaN           NaN  KEGG: hsa04012
AREG_EGFR                   NaN           NaN  KEGG: hsa04012
TGFA_EGFR_ERBB2             NaN           NaN  KEGG: hsa04012
TGFA_EGFR                   NaN           NaN  KEGG: hsa04012
EGF_EGFR_ERBB2              NaN           NaN  KEGG: hsa04012
EGF_EGFR                    NaN           NaN  KEGG: hsa04012

                        annotation     interaction_name_2
EREG_ERBB2_ERBB4   Secreted Signaling   EREG - (ERBB2+ERBB4)
EREG_EGFR_ERBB2    Secreted Signaling    EREG - (EGFR+ERBB2)
EREG_EGFR          Secreted Signaling            EREG - EGFR
HBEGF_ERBB2_ERBB4  Secreted Signaling  HBEGF - (ERBB2+ERBB4)
HBEGF_EGFR_ERBB2   Secreted Signaling   HBEGF - (EGFR+ERBB2)
HBEGF_EGFR         Secreted Signaling           HBEGF - EGFR
AREG_EGFR_ERBB2    Secreted Signaling    AREG - (EGFR+ERBB2)
AREG_EGFR          Secreted Signaling            AREG - EGFR
TGFA_EGFR_ERBB2    Secreted Signaling    TGFA - (EGFR+ERBB2)
TGFA_EGFR          Secreted Signaling            TGFA - EGFR
EGF_EGFR_ERBB2     Secreted Signaling     EGF - (EGFR+ERBB2)
EGF_EGFR           Secreted Signaling             EGF - EGFR
```

[38]:
```
# visualize 3 pairs in a pattern
pl.plot_pairs(A1, ['TGFA_EGFR','EGF_EGFR'], cmap='Wistia')
```





[41]:
```
# double check if the last column is cell type or not
A1.obsm['celltypes'] = A1.obs[A1.obs.columns[:-1]]

pl.chord_celltype(A1, pairs=['TGFA_EGFR','EGF_EGFR'], ncol=2, min_quantile=0.01)
# save='A1_EGF.svg'
```

Data type cannot be displayed: application/javascript, application/vnd.bokehjs_exec.v0+json

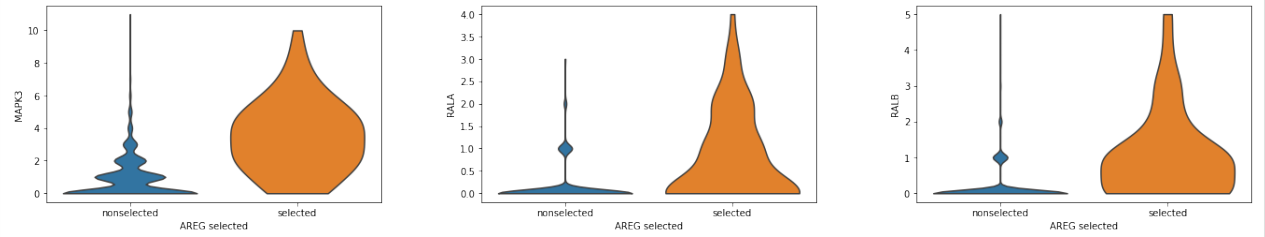
[41]:
```
Column(id='1429', ...)
```

Note

MAPK and RAL are reported upstream and downstream effectors of EGF, respectively. We examine if they express at the identified spots Please refer to our manuscript for more discussions.

[42]:
```
A1.obs['AREG_selected'] = np.where(A1.uns['selected_spots'].loc['AREG_EGFR'].values,
↪'selected','nonselected')
```

[43]:
```
import scanpy as sc
sc.pl.violin(A1, ['MAPK3','RALA', 'RALB'], 'AREG_selected', stripplot=False)
```



2.5.3 Cross-replicate consistency

Thanks to the multi-replicate setting, we could validate the consistency between biological / technical replicates

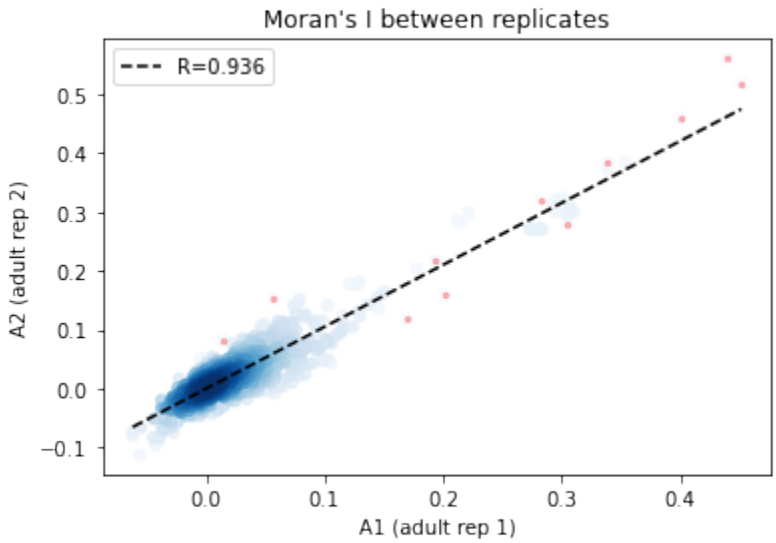
Global consistency

[29]:
```
I_df = pd.DataFrame(pd.Series(sample.uns['global_I'], index=sample.uns['global_res'].
↪index) for sample in samples)
I_df = I_df.transpose()
I_df.columns = data
```

Extract the Global R from each replicate.

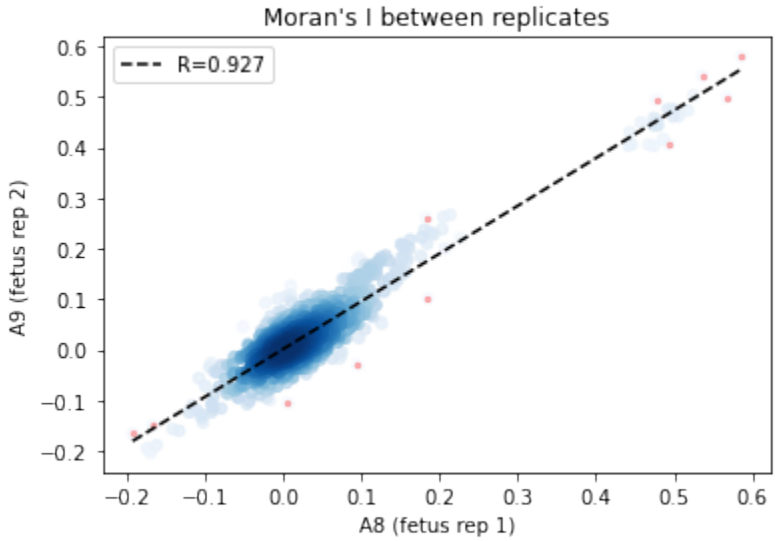
[30]:
```
from hilearn import *
_df = I_df.loc[:, ['A1', 'A2']].dropna()
corr_plot(_df.A1.values, _df.A2.values)
plt.xlabel('A1 (adult rep 1)')
plt.ylabel('A2 (adult rep 2)')
plt.title('Moran\'s I between replicates')
```

[30]:
```
Text(0.5, 1.0, "Moran's I between replicates")
```



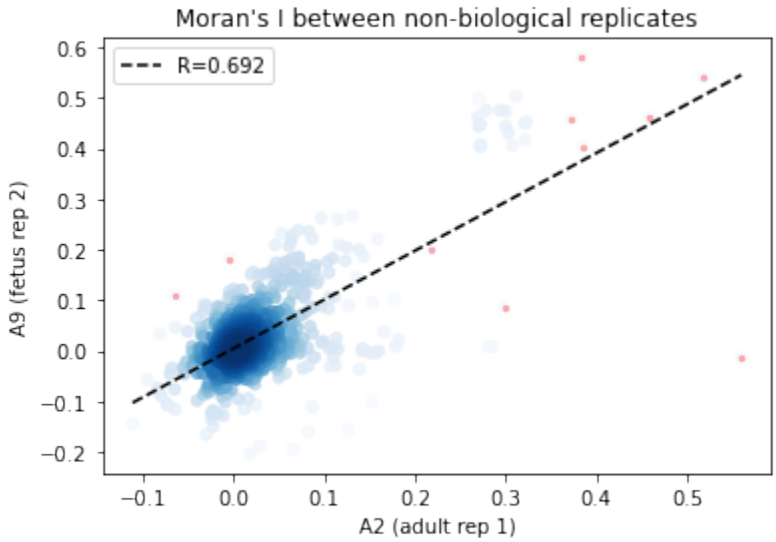
[31]:
```
_df = I_df.loc[:, ['A8', 'A9']].dropna()
corr_plot(_df.A8.values, _df.A9.values)
plt.xlabel('A8 (fetus rep 1)')
plt.ylabel('A9 (fetus rep 2)')
plt.title('Moran\'s I between replicates')
```

[31]:
```
Text(0.5, 1.0, "Moran's I between replicates")
```



[32]:
```
_df = I_df.loc[:, ['A2', 'A9']].dropna()
corr_plot(_df.A2.values, _df.A9.values)
plt.xlabel('A2 (adult rep 1)')
plt.ylabel('A9 (fetus rep 2)')
plt.title('Moran\'s I between non-biological replicates')
```

[32]:
```
Text(0.5, 1.0, "Moran's I between non-biological replicates")
```



From the above results, Global R is consistent between replicates.

Local consistency & selected cell type consistency

We can also assess the local selection consistency by comparing the cell type compositions of selected spots

[33]:
```
fetus_celltypes = A3.obs.columns
PLG_df = pd.DataFrame(fetus_celltypes, index=fetus_celltypes)
```

PLG_F2RL1 is sparsely detected in fetus samples, it will be interesting to see whether such selections are consistent.

[34]:
```
pair='PLG_F2RL1'
for d,sample in zip(data,samples):
    sample.celltype = locals()['{}'.format(d)].obs
    if pair in sample.uns['local_z_p'].index:
        ct=sample.celltype[sample.uns['local_z_p'].loc[pair].values<0.1].sum(0).sort_
↪values(ascending=False)
        ct=ct[ct>0]
        PLG_df['{}'.format(d)]=ct
    else:
        PLG_df['{}'.format(d)] = 0
```

Group spot weights by selecte cell type vs other cell types

[35]:
```
PLG_df.pop(0)
other=PLG_df.loc[~PLG_df.index.isin(['Distal Enterocytes','Distal Mature Enterocytes'])].
↪sum(0)
df =pd.concat((PLG_df.loc[PLG_df.index.isin(['Distal Enterocytes','Distal Mature␣
↪Enterocytes'])],
    pd.DataFrame(other, columns=['other cell types']).transpose()))

df = df.transpose()
df['sample']=df.index
```

PLG_F2RL1 is enriched in Enterocytes in 12-PCW colons, but not found in 19-PCW and adult samples. The two 12-PCW TI samples also have consistent PLG_F2RL1 enrichment, but the signaling celltypes are different from colon samples.

[36]:
```
plt.figure()
ax=df.sort_index().plot(x='sample',
        kind='bar',
        stacked=False,
        title='cell type weights in PLG_F2RL1 selected spot',
        color={'Distal Mature Enterocytes': [0.5       , 1.        , 0.65808859, 1.     ␣
↪  ],
               'Distal Enterocytes': [0.06331813, 0.62857143, 0.        , 1.        ],
#               'neuron':[0.99848342, 0.875     , 0.99522066, 1.        ],
               'other cell types': 'grey'})
ax.set_xticklabels(df.sort_index().index, rotation=0)
```

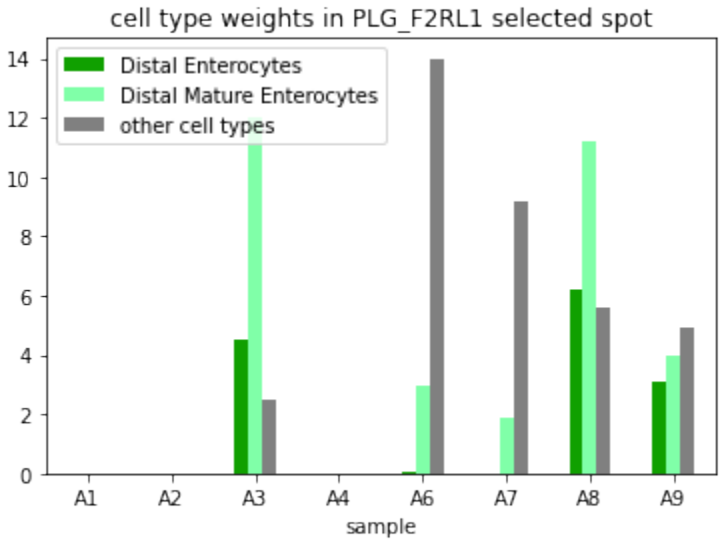
[36]:
```
[Text(0, 0, 'A1'),
 Text(1, 0, 'A2'),
 Text(2, 0, 'A3'),
 Text(3, 0, 'A4'),
 Text(4, 0, 'A6'),
 Text(5, 0, 'A7'),
```

(continues on next page)

(continued from previous page)

```
 Text(6, 0, 'A8'),
 Text(7, 0, 'A9')]
```

```
<Figure size 432x288 with 0 Axes>
```



PYTHON MODULE INDEX

S

INDEX

M

S