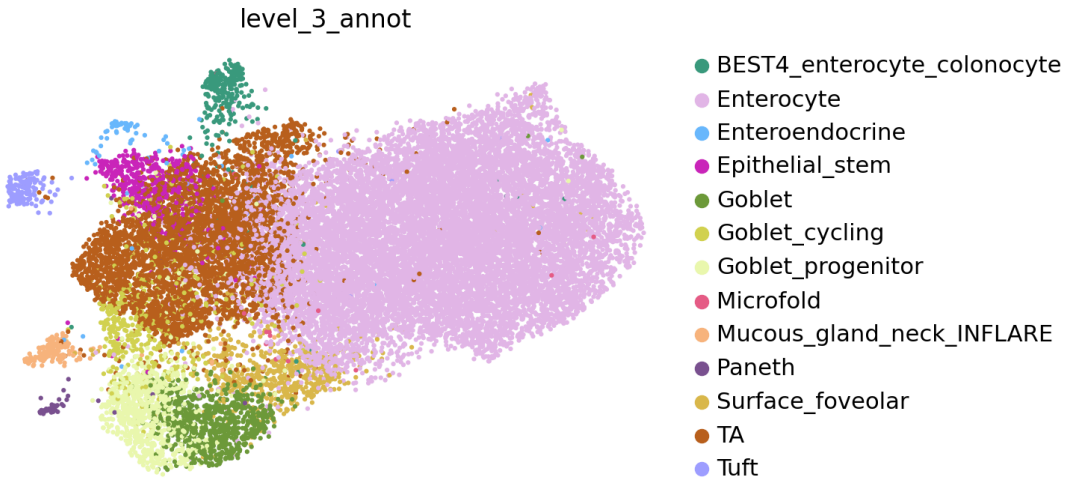
V2_CellRank-DUO

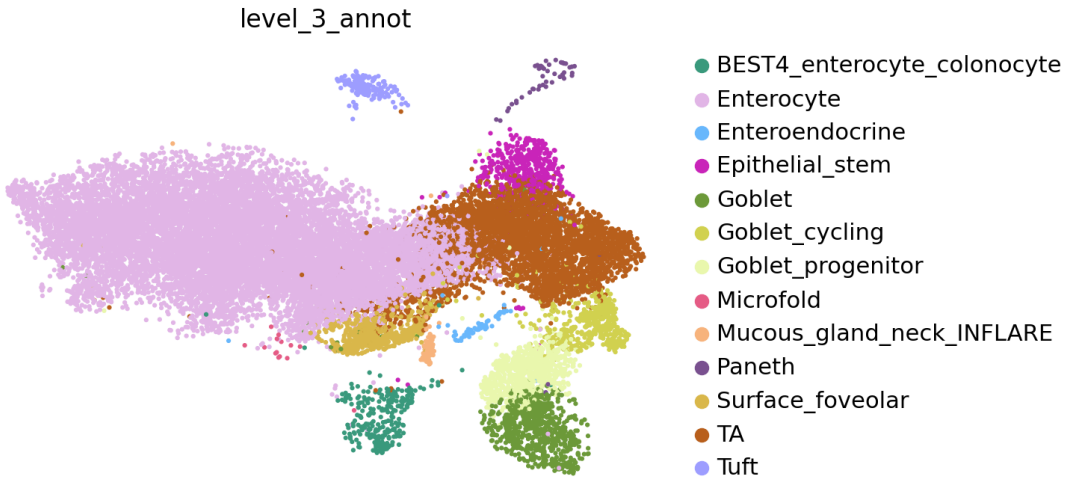
June 4, 2024

1 Cell Rank based pseudotime analysis with multiple kernels

[2]:
```
import cellrank as cr
from cellrank.kernels import CytoTRACEKernel
import scanpy as sc
import scvelo as scv
import numpy as np
import warnings
import anndata
import pandas as pd
import numpy as np
warnings.simplefilter("ignore", category=UserWarning)
warnings.simplefilter(action='ignore', category=FutureWarning)

sc.settings.set_figure_params(frameon=False, dpi=100)
cr.settings.verbosity = 2
scv.settings.verbosity = 3
scv.settings.set_figure_params("scvelo")

adata = anndata.read_h5ad('duo_healthy_ileum_disease.h5ad')
meta = pd.read_csv('meta_subset_duo_healthy_ileum_disease.csv')
adata.obs = meta.set_index('index')
sc.pl.embedding(adata , basis='umap_scvi', color='level_3_annot', s=20 )
sc.pl.embedding(adata , basis='umap', color='level_3_annot', s=20 )

adata.obs['clusters'] = adata.obs.level_3_annot
```

```
Global seed set to 0
```





[1]:
```
import cellrank as cr
cr.__version__
```

```
Global seed set to 0
```

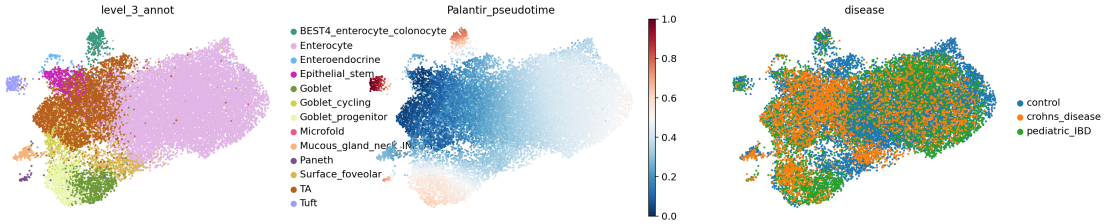
[1]:
```
'2.0.1'
```

[3]:
```
adata
```

[3]:
```
AnnData object with n_obs × n_vars = 20765 × 36601
    obs: 'latent_cell_probability', 'latent_RT_efficiency', 'cecilia22_predH',
'cecilia22_predH_prob', 'cecilia22_predH_uncertain', 'cecilia22_predL',
'cecilia22_predL_prob', 'cecilia22_predL_uncertain', 'elmentaite21_pred',
```

```
'elmentaite21_pred_prob', 'elmentaite21_pred_uncertain', 'suo22_pred',
'suo22_pred_prob', 'suo22_pred_uncertain', 'n_counts', 'log1p_n_counts',
'n_genes', 'log1p_n_genes', 'percent_mito', 'n_counts_mito', 'percent_ribo',
'n_counts_ribo', 'percent_hb', 'n_counts_hb', 'percent_top50', 'n_counts_raw',
'log1p_n_counts_raw', 'n_genes_raw', 'log1p_n_genes_raw', 'percent_mito_raw',
'n_counts_mito_raw', 'percent_ribo_raw', 'n_counts_ribo_raw', 'percent_hb_raw',
'n_counts_hb_raw', 'percent_top50_raw', 'n_counts_spliced',
'log1p_n_counts_spliced', 'n_genes_spliced', 'log1p_n_genes_spliced',
'percent_mito_spliced', 'n_counts_mito_spliced', 'percent_ribo_spliced',
'n_counts_ribo_spliced', 'percent_hb_spliced', 'n_counts_hb_spliced',
'percent_top50_spliced', 'n_counts_unspliced', 'log1p_n_counts_unspliced',
'n_genes_unspliced', 'log1p_n_genes_unspliced', 'percent_mito_unspliced',
'n_counts_mito_unspliced', 'percent_ribo_unspliced', 'n_counts_ribo_unspliced',
'percent_hb_unspliced', 'n_counts_hb_unspliced', 'percent_top50_unspliced',
'percent_soup', 'percent_spliced', 'qc_cluster', 'pass_auto_filter_mito20',
'good_qc_cluster_mito20', 'pass_auto_filter_mito50', 'good_qc_cluster_mito50',
'pass_auto_filter_mito80', 'good_qc_cluster_mito80', 'pass_auto_filter',
'good_qc_cluster', 'pass_default', 'sampleID', 'sourceID', 'donorID_original',
'study', 'donorID_corrected', 'donorID_unified', 'donor_category',
'donor_disease', 'organ_original', 'organ_unified', 'organ_broad',
'age_original', 'age_unified', 'age_continuousadult', 'age_continuousdev',
'sex', 'sample_type', 'sample_category', 'sample_retrieval', 'tissue_fraction',
'cell_fraction', 'cell_fraction_unified', 'cell_sorting', 'technology',
'include_150722', 'cluster_scrublet_score', 'bh_pval', 'scrublet_score',
'scrublet_score_z', 'doublet', 'stringent_doublet', 'integration_grouping',
'_scvi_batch', '_scvi_labels', 'broad_annot_20220914', 'martin19_pred',
'martin19_pred_prob', 'martin19_pred_uncertain', 'warner20_pred',
'warner20_pred_prob', 'warner20_pred_uncertain', 'broad_annot_20220917',
'donor_organ_lineage', 'fine_annot', 'fine_annot_original', 'level_1_annot',
'level_2_annot', 'level_3_annot', 'broad_predicted_labels',
'broad_predicted_labels_uncert', 'batch', 'organ_groups', 'control_vs_disease',
'disease', 'ID', 'clusters', 'Palantir_pseudotime'
    var: 'gene_ids', 'feature_type', 'mito', 'ribo', 'hb', 'cc', 'ig', 'tcr',
'n_counts-0', 'n_counts_raw-0', 'n_counts_spliced-0', 'n_counts_unspliced-0',
'n_cells-0', 'n_cells_raw-0', 'n_cells_spliced-0', 'n_cells_unspliced-0',
'n_counts-1', 'n_counts_raw-1', 'n_counts_spliced-1', 'n_counts_unspliced-1',
'n_cells-1', 'n_cells_raw-1', 'n_cells_spliced-1', 'n_cells_unspliced-1'
    uns: 'disease_colors', 'level_3_annot_colors', 'organ_unified_colors'
    obsm: 'X_mde', 'X_scANVI', 'X_scVI', 'X_umap', 'X_umap_scvi',
'_scvi_extra_continuous_covs'
    layers: 'counts', 'spliced', 'unspliced'
```

[4]:
```
sc.pl.embedding(adata , basis='umap_scvi',␣
↪color=['level_3_annot','Palantir_pseudotime','disease'], s=20 )
```



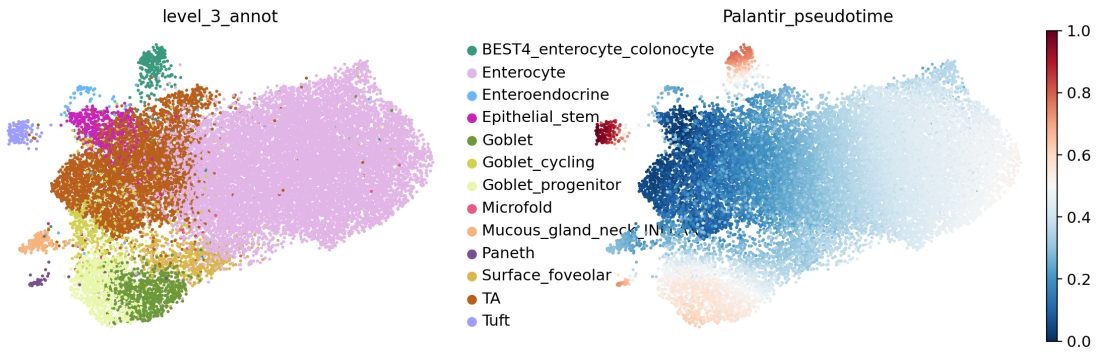
[6]:
```
import psutil
# Get RAM usage information
ram = psutil.virtual_memory()

# Convert bytes to gigabytes
total_gb = ram.total / (1024 ** 3)
used_gb = ram.used / (1024 ** 3)
free_gb = ram.free / (1024 ** 3)

print(f"Total RAM: {total_gb:.2f} GB")
print(f"Used RAM: {used_gb:.2f} GB")
print(f"Free RAM: {free_gb:.2f} GB")
```

```
Total RAM: 102.29 GB
Used RAM: 25.85 GB
Free RAM: 66.86 GB
```

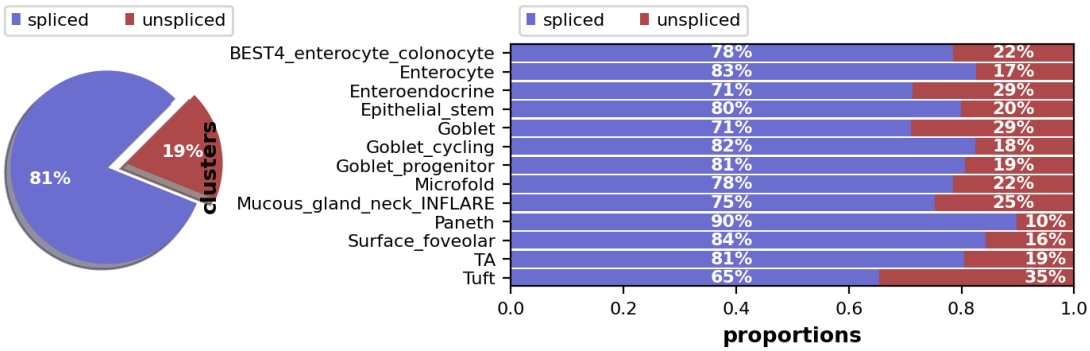
[7]:
```
adata.X.data
```

[7]:
```
array([ 2.,  1.,  2., …, 64.,  6., 63.], dtype=float32)
```

[8]:
```
sc.pl.embedding(adata , basis='umap_scvi',␣
 ↪color=['level_3_annot','Palantir_pseudotime'], s=20 )
```



[9]:
```
scv.pl.proportions(adata)
```



[10]:
```
adata_original = adata.copy()
adata_all = adata_original.copy()
```

[11]:
```
adata.X.data
```

[11]:
```
array([ 2.,  1.,  2., …, 64.,  6., 63.], dtype=float32)
```

[12]:
```
adata = adata_original.copy()
```

[13]:
```
adata.X.data
```

[13]:
```
array([ 2.,  1.,  2., …, 64.,  6., 63.], dtype=float32)
```

[14]:
```
adata
```

[14]:
```
AnnData object with n_obs × n_vars = 20765 × 36601
    obs: 'latent_cell_probability', 'latent_RT_efficiency', 'cecilia22_predH',
'cecilia22_predH_prob', 'cecilia22_predH_uncertain', 'cecilia22_predL',
'cecilia22_predL_prob', 'cecilia22_predL_uncertain', 'elmentaite21_pred',
'elmentaite21_pred_prob', 'elmentaite21_pred_uncertain', 'suo22_pred',
'suo22_pred_prob', 'suo22_pred_uncertain', 'n_counts', 'log1p_n_counts',
'n_genes', 'log1p_n_genes', 'percent_mito', 'n_counts_mito', 'percent_ribo',
'n_counts_ribo', 'percent_hb', 'n_counts_hb', 'percent_top50', 'n_counts_raw',
'log1p_n_counts_raw', 'n_genes_raw', 'log1p_n_genes_raw', 'percent_mito_raw',
'n_counts_mito_raw', 'percent_ribo_raw', 'n_counts_ribo_raw', 'percent_hb_raw',
'n_counts_hb_raw', 'percent_top50_raw', 'n_counts_spliced',
'log1p_n_counts_spliced', 'n_genes_spliced', 'log1p_n_genes_spliced',
'percent_mito_spliced', 'n_counts_mito_spliced', 'percent_ribo_spliced',
'n_counts_ribo_spliced', 'percent_hb_spliced', 'n_counts_hb_spliced',
'percent_top50_spliced', 'n_counts_unspliced', 'log1p_n_counts_unspliced',
'n_genes_unspliced', 'log1p_n_genes_unspliced', 'percent_mito_unspliced',
'n_counts_mito_unspliced', 'percent_ribo_unspliced', 'n_counts_ribo_unspliced',
'percent_hb_unspliced', 'n_counts_hb_unspliced', 'percent_top50_unspliced',
'percent_soup', 'percent_spliced', 'qc_cluster', 'pass_auto_filter_mito20',
```

```
'good_qc_cluster_mito20', 'pass_auto_filter_mito50', 'good_qc_cluster_mito50',
'pass_auto_filter_mito80', 'good_qc_cluster_mito80', 'pass_auto_filter',
'good_qc_cluster', 'pass_default', 'sampleID', 'sourceID', 'donorID_original',
'study', 'donorID_corrected', 'donorID_unified', 'donor_category',
'donor_disease', 'organ_original', 'organ_unified', 'organ_broad',
'age_original', 'age_unified', 'age_continuousadult', 'age_continuousdev',
'sex', 'sample_type', 'sample_category', 'sample_retrieval', 'tissue_fraction',
'cell_fraction', 'cell_fraction_unified', 'cell_sorting', 'technology',
'include_150722', 'cluster_scrublet_score', 'bh_pval', 'scrublet_score',
'scrublet_score_z', 'doublet', 'stringent_doublet', 'integration_grouping',
'_scvi_batch', '_scvi_labels', 'broad_annot_20220914', 'martin19_pred',
'martin19_pred_prob', 'martin19_pred_uncertain', 'warner20_pred',
'warner20_pred_prob', 'warner20_pred_uncertain', 'broad_annot_20220917',
'donor_organ_lineage', 'fine_annot', 'fine_annot_original', 'level_1_annot',
'level_2_annot', 'level_3_annot', 'broad_predicted_labels',
'broad_predicted_labels_uncert', 'batch', 'organ_groups', 'control_vs_disease',
'disease', 'ID', 'clusters', 'Palantir_pseudotime'
    var: 'gene_ids', 'feature_type', 'mito', 'ribo', 'hb', 'cc', 'ig', 'tcr',
'n_counts-0', 'n_counts_raw-0', 'n_counts_spliced-0', 'n_counts_unspliced-0',
'n_cells-0', 'n_cells_raw-0', 'n_cells_spliced-0', 'n_cells_unspliced-0',
'n_counts-1', 'n_counts_raw-1', 'n_counts_spliced-1', 'n_counts_unspliced-1',
'n_cells-1', 'n_cells_raw-1', 'n_cells_spliced-1', 'n_cells_unspliced-1'
    uns: 'disease_colors', 'level_3_annot_colors', 'organ_unified_colors'
    obsm: 'X_mde', 'X_scANVI', 'X_scVI', 'X_umap', 'X_umap_scvi',
'_scvi_extra_continuous_covs'
    layers: 'counts', 'spliced', 'unspliced'
```

[15]:
```python
import psutil

# Get RAM usage information
ram = psutil.virtual_memory()

# Convert bytes to gigabytes
total_gb = ram.total / (1024 ** 3)
used_gb = ram.used / (1024 ** 3)
free_gb = ram.free / (1024 ** 3)

print(f"Total RAM: {total_gb:.2f} GB")
print(f"Used RAM: {used_gb:.2f} GB")
print(f"Free RAM: {free_gb:.2f} GB")
```

```
Total RAM: 102.29 GB
Used RAM: 28.54 GB
Free RAM: 64.16 GB
```

[16]:
```python
def compute_RNA_velocity_kernel(adata, umap_basis):

    # https://scvelo.readthedocs.io/en/stable/VelocityBasics/
```

```
    # Velocity represent the direction and speed of movement of the individual␣
↪cells (this is obtained using modelling of transcriptional dynamics
    # of splicing kinetics either stochastic or deterministic).
    # + velocity = gene is upregulated (cells with higher abundance of␣
↪unspliced mRNA than expected in steady state)
    # - velocity - gene is downregulated
    # for each gene of a cell, velocity is computed

    # === COMPUTE THE RNA VELOCITY KERNEL
    #Filter out genes which don't have enough spliced/unspliced counts,␣
↪normalize and log transform the data
    #and restrict to the top highly variable genes. Further, compute principal␣
↪components and moments for velocity estimation.
    scv.pp.filter_and_normalize(adata, min_shared_counts=20, n_top_genes=2000,␣
↪subset_highly_variable=False)
    sc.tl.pca(adata)
    sc.pp.neighbors(adata, n_pcs=10, n_neighbors=30, random_state=0,␣
↪use_rep='X_scVI')
    scv.pp.moments(adata, n_pcs=10, n_neighbors=30, use_rep='X_scVI')
    scv.tl.recover_dynamics(adata, n_jobs=8) #  scVelo's dynamical model to␣
↪estimate model parameters
    scv.tl.velocity(adata, mode="dynamical")
    vk = cr.kernels.VelocityKernel(adata)
    vk.compute_transition_matrix() #and then transition probability matrix␣
↪based on that
    vk.plot_projection(basis=umap_basis, color="clusters", legend_loc="right",␣
↪recompute=True)
    return vk, adata
```

[17]:
```
velocity_kernel, adata = compute_RNA_velocity_kernel(adata, umap_basis =␣
↪"umap_scvi")
```

```
Filtered out 25756 genes that are detected 20 counts (shared).
Normalized count data: X, spliced, unspliced.
Extracted 2000 highly variable genes.
```

```
/opt/conda/envs/pygpcca_env/lib/python3.8/site-
packages/scvelo/preprocessing/utils.py:705: DeprecationWarning: `log1p` is
deprecated since scVelo v0.3.0 and will be removed in a future version. Please
use `log1p` from `scanpy.pp` instead.
  log1p(adata)
```

```
Logarithmized X.
computing moments based on connectivities
    finished (0:00:14) --> added
    'Ms' and 'Mu', moments of un/spliced abundances (adata.layers)
recovering dynamics (using 8/16 cores)
```

```
  0%|          | 0/642 [00:00<?, ?gene/s]
```

```
Global seed set to 0
Global seed set to 0
Global seed set to 0
Global seed set to 0
Global seed set to 0
Global seed set to 0
Global seed set to 0
Global seed set to 0
```

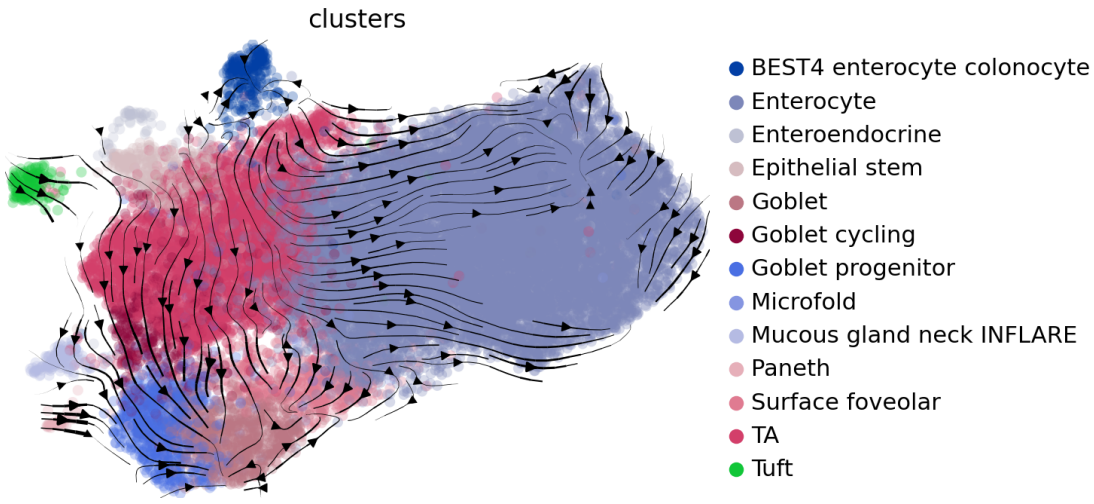
```
    finished (0:02:54) --> added
    'fit_pars', fitted parameters for splicing dynamics (adata.var)
computing velocities
    finished (0:00:14) --> added
    'velocity', velocity vectors for each individual cell (adata.layers)
Computing transition matrix using `'deterministic'` model
```

```
  0%|          | 0/20765 [00:00<?, ?cell/s]
```

```
Using `softmax_scale=7.4908`
```

```
  0%|          | 0/20765 [00:00<?, ?cell/s]
```

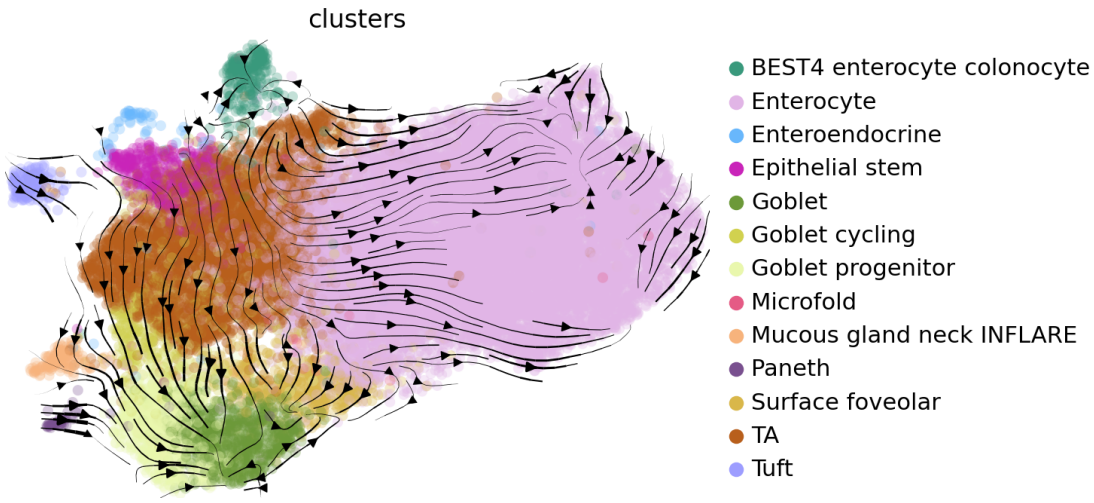
```
    Finish (0:00:43)
Projecting transition matrix onto `umap_scvi`
Adding `adata.obsm['T_fwd_umap_scvi']`
    Finish (0:00:07)
```



[18]:
```
our_cmap = {}
our_palette = adata.uns['level_3_annot_colors']
i = 0
for c in np.unique(adata.obs['level_3_annot']) :
    our_cmap[c] = our_palette[i]
```

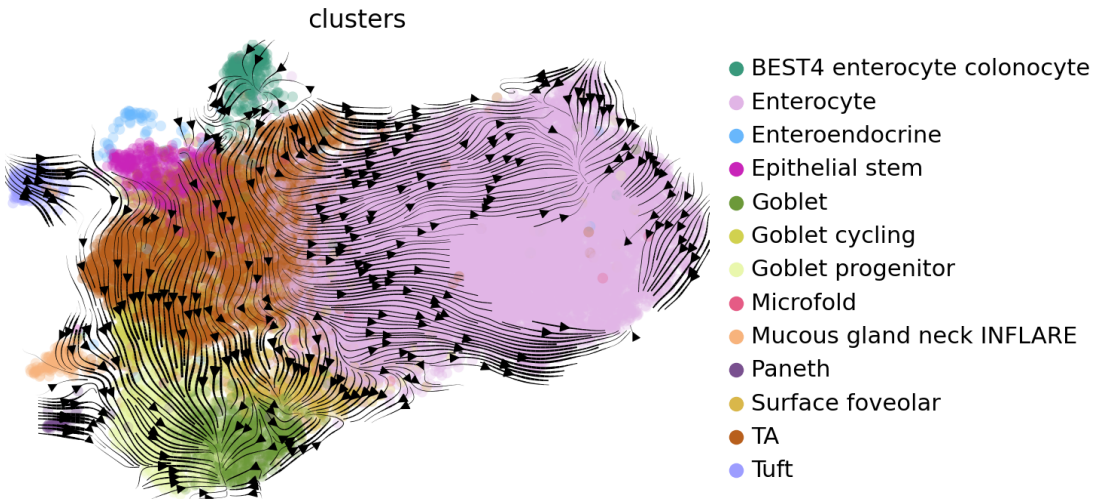
```
    i+=1
velocity_kernel.plot_projection(basis='umap_scvi', color="clusters",␣
 ↪legend_loc="right", recompute=True,
                                palette = our_cmap)
```

```
Projecting transition matrix onto `umap_scvi`
Adding `adata.obsm['T_fwd_umap_scvi']`
    Finish (0:00:07)
```



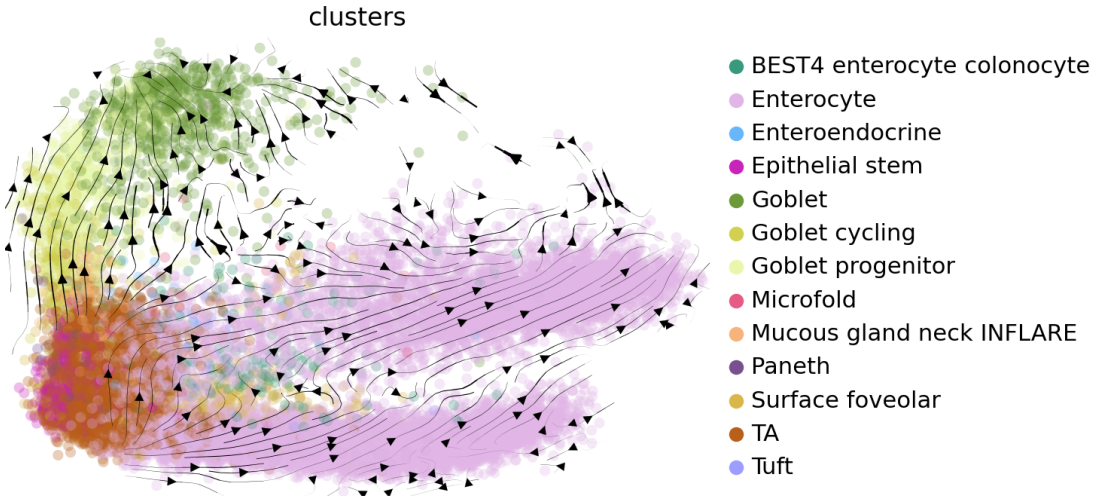
[19]:
```
velocity_kernel.plot_projection(basis='umap_scvi', color="clusters",␣
 ↪legend_loc="right", recompute=True,
                                palette = our_cmap, density=5)
```

```
Projecting transition matrix onto `umap_scvi`
Adding `adata.obsm['T_fwd_umap_scvi']`
    Finish (0:00:07)
```



[20]:
```
velocity_kernel.plot_projection(basis='pca', color="clusters",␣
↪legend_loc="right", recompute=True,
                                palette = our_cmap)
```

```
Projecting transition matrix onto `pca`
Adding `adata.obsm['T_fwd_pca']`
    Finish (0:00:10)
```



[21]:
```
adata.X.data
```

[21]:
```
array([1.9597793, 1.3984402, 1.9597793, …, 1.2246249, 1.2246249,
       1.2246249], dtype=float32)
```

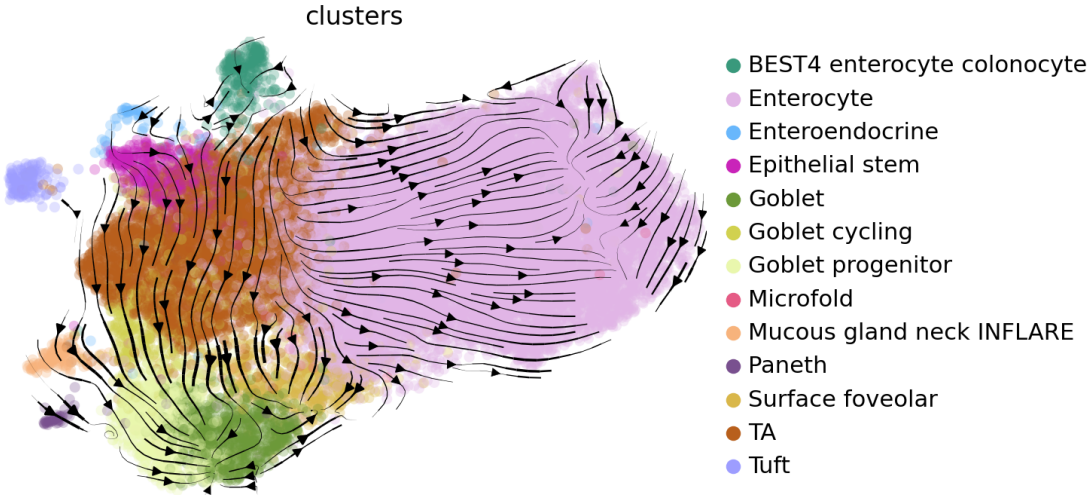
[22]:
```
#adata = adata_original.copy()
#scv.pp.filter_and_normalize(adata, min_shared_counts=20, ␣
 ↪subset_highly_variable=False) #n_top_genes=4000,
#sc.tl.pca(adata)
#sc.pp.neighbors(adata, n_pcs=30, n_neighbors=30, random_state=0)
#scv.pp.moments(adata, n_pcs=30, n_neighbors=30)
#scv.tl.velocity(adata) # velocity vector for each cell across the genes: in␣
 ↪adata.layers['velocity'] same dims as GEX
```

[23]:
```
scv.tl.velocity_graph(adata) # then we can project  velocities into low␣
 ↪dimension space by first estimating cell2cell transition.
# mutliplying transcription state change vector and RNA vector
# This velocity graph is then converted to a transition matrix by transforming␣
 ↪cosine correlations into actual probs (via a Gaussian kernel).
```

```
computing velocity graph (using 1/16 cores)
  0%|          | 0/20765 [00:00<?, ?cells/s]
    finished (0:00:35) --> added
    'velocity_graph', sparse matrix with cosine correlations (adata.uns)
```

[24]:
```
scv.pl.velocity_embedding_stream(adata, basis='umap_scvi',␣
 ↪legend_loc='right',recompute=True,palette = our_cmap)
# projecting the velocity vectors onto low dimensional umap embedding by␣
 ↪applying the mean transition with respect to the transition probabilities
```

```
computing velocity embedding
    finished (0:00:06) --> added
    'velocity_umap_scvi', embedded velocity vectors (adata.obsm)
```



[25]:
```
adata_original
```

[25]:
```
AnnData object with n_obs × n_vars = 20765 × 36601
    obs: 'latent_cell_probability', 'latent_RT_efficiency', 'cecilia22_predH',
'cecilia22_predH_prob', 'cecilia22_predH_uncertain', 'cecilia22_predL',
'cecilia22_predL_prob', 'cecilia22_predL_uncertain', 'elmentaite21_pred',
'elmentaite21_pred_prob', 'elmentaite21_pred_uncertain', 'suo22_pred',
'suo22_pred_prob', 'suo22_pred_uncertain', 'n_counts', 'log1p_n_counts',
'n_genes', 'log1p_n_genes', 'percent_mito', 'n_counts_mito', 'percent_ribo',
'n_counts_ribo', 'percent_hb', 'n_counts_hb', 'percent_top50', 'n_counts_raw',
'log1p_n_counts_raw', 'n_genes_raw', 'log1p_n_genes_raw', 'percent_mito_raw',
'n_counts_mito_raw', 'percent_ribo_raw', 'n_counts_ribo_raw', 'percent_hb_raw',
'n_counts_hb_raw', 'percent_top50_raw', 'n_counts_spliced',
'log1p_n_counts_spliced', 'n_genes_spliced', 'log1p_n_genes_spliced',
'percent_mito_spliced', 'n_counts_mito_spliced', 'percent_ribo_spliced',
'n_counts_ribo_spliced', 'percent_hb_spliced', 'n_counts_hb_spliced',
'percent_top50_spliced', 'n_counts_unspliced', 'log1p_n_counts_unspliced',
'n_genes_unspliced', 'log1p_n_genes_unspliced', 'percent_mito_unspliced',
'n_counts_mito_unspliced', 'percent_ribo_unspliced', 'n_counts_ribo_unspliced',
'percent_hb_unspliced', 'n_counts_hb_unspliced', 'percent_top50_unspliced',
'percent_soup', 'percent_spliced', 'qc_cluster', 'pass_auto_filter_mito20',
'good_qc_cluster_mito20', 'pass_auto_filter_mito50', 'good_qc_cluster_mito50',
'pass_auto_filter_mito80', 'good_qc_cluster_mito80', 'pass_auto_filter',
'good_qc_cluster', 'pass_default', 'sampleID', 'sourceID', 'donorID_original',
'study', 'donorID_corrected', 'donorID_unified', 'donor_category',
'donor_disease', 'organ_original', 'organ_unified', 'organ_broad',
'age_original', 'age_unified', 'age_continuousadult', 'age_continuousdev',
'sex', 'sample_type', 'sample_category', 'sample_retrieval', 'tissue_fraction',
'cell_fraction', 'cell_fraction_unified', 'cell_sorting', 'technology',
'include_150722', 'cluster_scrublet_score', 'bh_pval', 'scrublet_score',
'scrublet_score_z', 'doublet', 'stringent_doublet', 'integration_grouping',
'_scvi_batch', '_scvi_labels', 'broad_annot_20220914', 'martin19_pred',
'martin19_pred_prob', 'martin19_pred_uncertain', 'warner20_pred',
'warner20_pred_prob', 'warner20_pred_uncertain', 'broad_annot_20220917',
'donor_organ_lineage', 'fine_annot', 'fine_annot_original', 'level_1_annot',
'level_2_annot', 'level_3_annot', 'broad_predicted_labels',
'broad_predicted_labels_uncert', 'batch', 'organ_groups', 'control_vs_disease',
'disease', 'ID', 'clusters', 'Palantir_pseudotime'
    var: 'gene_ids', 'feature_type', 'mito', 'ribo', 'hb', 'cc', 'ig', 'tcr',
'n_counts-0', 'n_counts_raw-0', 'n_counts_spliced-0', 'n_counts_unspliced-0',
'n_cells-0', 'n_cells_raw-0', 'n_cells_spliced-0', 'n_cells_unspliced-0',
'n_counts-1', 'n_counts_raw-1', 'n_counts_spliced-1', 'n_counts_unspliced-1',
'n_cells-1', 'n_cells_raw-1', 'n_cells_spliced-1', 'n_cells_unspliced-1'
    uns: 'disease_colors', 'level_3_annot_colors', 'organ_unified_colors'
    obsm: 'X_mde', 'X_scANVI', 'X_scVI', 'X_umap', 'X_umap_scvi',
'_scvi_extra_continuous_covs'
    layers: 'counts', 'spliced', 'unspliced'
```

[26]:
```
adata.obsp['connectivities'].data
```

[26]:
```
array([0.11081272, 0.16505973, 0.04109649, …, 0.0593112 , 0.1033306 ,
       0.2130326 ], dtype=float32)
```

[27]:
```
# re-assigning neighbourhood graph
#sc.pp.neighbors(adata_all, use_rep="X_scVI",method='gauss')
#adata.uns['neighbors']['distances'] = adata.obsp['distances']
#adata.uns['neighbors']['connectivities'] = adata.obsp['connectivities']

#scv.tl.paga(adata, groups='clusters', use_time_prior="Palantir_pseudotime")
#df = scv.get_df(adata, 'paga/transitions_confidence', precision=2).T
#df.style.background_gradient(cmap='Blues').format('{:.2g}')
#scv.pl.paga(adata, basis='umap_scvi', size=50, alpha=.1,
#            min_edge_width=1, node_size_scale=1.5, palette = our_cmap,␣
 ↪threshold=0.1)
#df.style.background_gradient(cmap='Blues').format('{:.2g}')
```

[ ]:
```

```

[28]:
```
adata.obsp['connectivities'].data
```

[28]:
```
array([0.11081272, 0.16505973, 0.04109649, …, 0.0593112 , 0.1033306 ,
       0.2130326 ], dtype=float32)
```

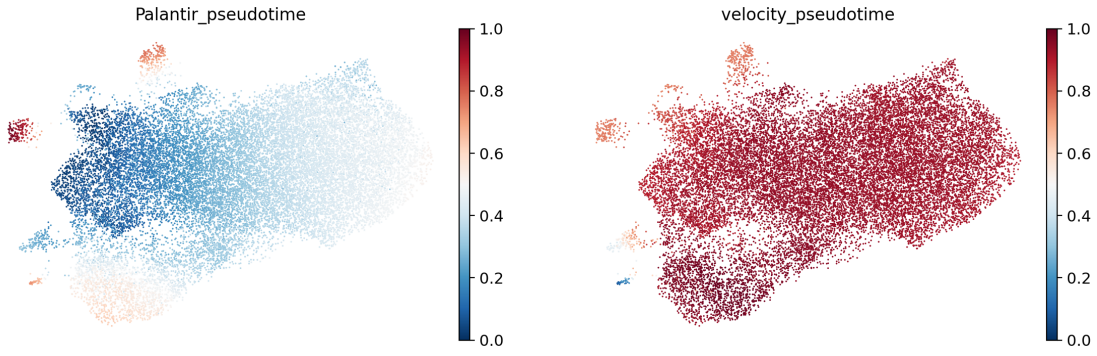
[29]:
```
#adata.uns['paga']
#scv.tl.paga(adata, groups='clusters')
#df = scv.get_df(adata, 'paga/transitions_confidence', precision=2).T
#df.style.background_gradient(cmap='Blues').format('{:.2g}')
#scv.pl.paga(adata, basis='umap_scvi', size=50, alpha=.1,
#            min_edge_width=0.4, node_size_scale=1.5, palette = our_cmap)
#df.style.background_gradient(cmap='Blues').format('{:.2g}')
#scv.pl.velocity_embedding(adata, basis='umap_scvi', arrow_length=10,␣
 ↪arrow_size=5, dpi=150, density=0.5,palette = our_cmap)

# checking and interpreting rna velo for a set of genes
#scv.pl.velocity(adata, ['Cpe',  'Gnao1', 'Ins2', 'Adk'], ncols=2)
#scv.tl.rank_velocity_genes(adata, groupby='clusters', min_corr=.3)
#df = pd.DataFrame(adata.uns['rank_velocity_genes']['names'])
#print(df.head())
#sc.pl.embedding(adata, basis='umap_scvi', color=['LGR5', 'SMOC2', 'RGMB',␣
 ↪'OLFM4'], cmap='YlOrBr', s=30)
#scv.tl.velocity_confidence(adata)
#keys = 'velocity_length', 'velocity_confidence'
#scv.pl.scatter(adata, c=keys, cmap='coolwarm', perc=[5, 95])
#adata.layers['velocity'].shape
```

```
#scv.pl.scatter(adata, color='velocity_pseudotime', cmap='gnuplot')
#scv.tl.paga(adata, groups='clusters', use_time_prior="velocity_pseudotime")
#df = scv.get_df(adata, 'paga/transitions_confidence', precision=2).T
#df.style.background_gradient(cmap='Blues').format('{:.2g}')
#scv.pl.paga(adata, basis='umap_scvi', size=50, alpha=.1,
#   min_edge_width=1, node_size_scale=1.5, palette = our_cmap)
#df.style.background_gradient(cmap='Blues').format('{:.2g}')
#scv.tl.paga(adata, groups='clusters')
#scv.pl.paga(adata, basis='umap_scvi', size=50, alpha=.1,
#   min_edge_width=2, node_size_scale=1.5, palette = our_cmap)
#scv.tl.paga(adata, groups='clusters', use_time_prior='Palantir_pseudotime')
#scv.pl.paga(adata, basis='umap_scvi', size=50, alpha=.1,
#            min_edge_width=2, node_size_scale=1.5, palette = our_cmap)
```

[30]:
```
scv.tl.velocity_pseudotime(adata)
sc.pl.embedding(adata, basis ='umap_scvi', color=['Palantir_pseudotime',␣
 ↪'velocity_pseudotime'])
```

```
computing terminal states
    identified 3 regions of root cells and 1 region of end points .
    finished (0:00:07) --> added
    'root_cells', root cells of Markov diffusion process (adata.obs)
    'end_points', end points of Markov diffusion process (adata.obs)
```

```
/opt/conda/envs/pygpcca_env/lib/python3.8/site-
packages/scanpy/plotting/_tools/scatterplots.py:163:
MatplotlibDeprecationWarning: The get_cmap function was deprecated in Matplotlib
3.7 and will be removed two minor releases later. Use
``matplotlib.colormaps[name]`` or ``matplotlib.colormaps.get_cmap(obj)``
instead.
  cmap = copy(get_cmap(cmap))
```



[31]:
```
sc.tl.score_genes(adata, gene_list=['LGR5', 'ASCL2', 'RGMB', 'OLFM4'] ,␣
 ↪score_name='start_score')
```

```
start_cell_id = 10848# same used for palantir run #np.argmax(adata.obs.
 ↪start_score)
start_cell =   adata.obs_names[start_cell_id]
start_cell_id
```

[31]:
```
10848
```

[32]:
```
print(start_cell)

#'CGTTAGATCCTTTACA-HT-188-Adult-Duo-0'
```

```
CGTTAGATCCTTTACA-HT-188-Adult-Duo-0
```

[28]:
```
#scv.pl.velocity_embedding(adata, arrow_length=3, arrow_size=2, dpi=120)
#sc.tl.score_genes(adata, gene_list=['LGR5', 'ASCL2', 'RGMB', 'OLFM4']  ,␣
 ↪score_name='start_score')
#start_cell_id = np.argmax(adata.obs.start_score)
#start_cell =  adata.obs_names[start_cell_id]
#start_cell_id
#scv.utils.get_cell_transitions(adata, basis='X_scVI',␣
 ↪starting_cell=start_cell_id )
#scv.tl.velocity_pseudotime(adata)
#scv.pl.scatter(adata,  basis=␣
 ↪'umap_scvi',color=['velocity_pseudotime',start_cell_id], cmap='gnuplot')
```

[33]:
```
adata.obsp['connectivities'].data
```

[33]:
```
array([0.11081272, 0.16505973, 0.04109649, …, 0.0593112 , 0.1033306 ,
       0.2130326 ], dtype=float32)
```

[34]:
```
#adata_original.obsp['connectivities'].data
```

[35]:
```
adata_original
```

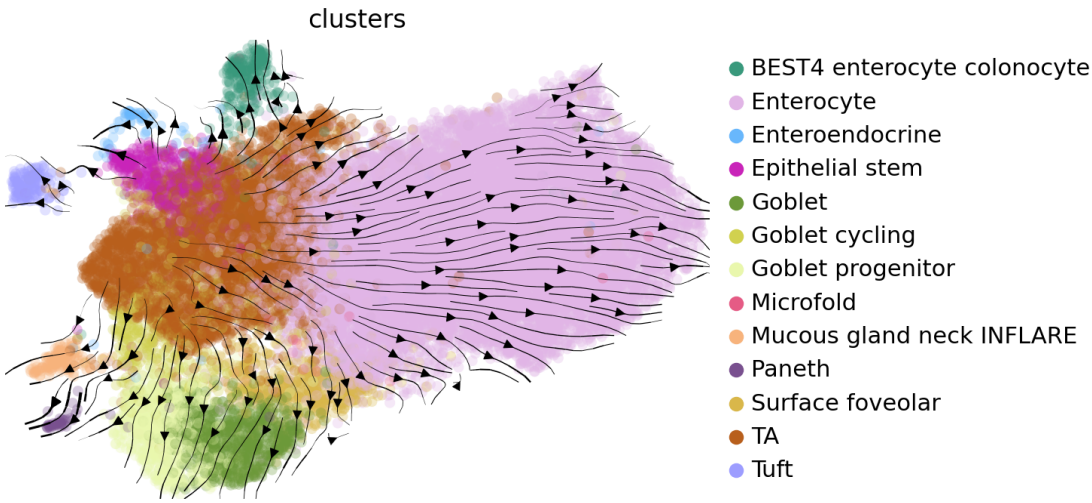
[35]:
```
AnnData object with n_obs × n_vars = 20765 × 36601
    obs: 'latent_cell_probability', 'latent_RT_efficiency', 'cecilia22_predH',
'cecilia22_predH_prob', 'cecilia22_predH_uncertain', 'cecilia22_predL',
'cecilia22_predL_prob', 'cecilia22_predL_uncertain', 'elmentaite21_pred',
'elmentaite21_pred_prob', 'elmentaite21_pred_uncertain', 'suo22_pred',
'suo22_pred_prob', 'suo22_pred_uncertain', 'n_counts', 'log1p_n_counts',
'n_genes', 'log1p_n_genes', 'percent_mito', 'n_counts_mito', 'percent_ribo',
'n_counts_ribo', 'percent_hb', 'n_counts_hb', 'percent_top50', 'n_counts_raw',
'log1p_n_counts_raw', 'n_genes_raw', 'log1p_n_genes_raw', 'percent_mito_raw',
'n_counts_mito_raw', 'percent_ribo_raw', 'n_counts_ribo_raw', 'percent_hb_raw',
'n_counts_hb_raw', 'percent_top50_raw', 'n_counts_spliced',
'log1p_n_counts_spliced', 'n_genes_spliced', 'log1p_n_genes_spliced',
'percent_mito_spliced', 'n_counts_mito_spliced', 'percent_ribo_spliced',
'n_counts_ribo_spliced', 'percent_hb_spliced', 'n_counts_hb_spliced',
```

```
'percent_top50_spliced', 'n_counts_unspliced', 'log1p_n_counts_unspliced',
'n_genes_unspliced', 'log1p_n_genes_unspliced', 'percent_mito_unspliced',
'n_counts_mito_unspliced', 'percent_ribo_unspliced', 'n_counts_ribo_unspliced',
'percent_hb_unspliced', 'n_counts_hb_unspliced', 'percent_top50_unspliced',
'percent_soup', 'percent_spliced', 'qc_cluster', 'pass_auto_filter_mito20',
'good_qc_cluster_mito20', 'pass_auto_filter_mito50', 'good_qc_cluster_mito50',
'pass_auto_filter_mito80', 'good_qc_cluster_mito80', 'pass_auto_filter',
'good_qc_cluster', 'pass_default', 'sampleID', 'sourceID', 'donorID_original',
'study', 'donorID_corrected', 'donorID_unified', 'donor_category',
'donor_disease', 'organ_original', 'organ_unified', 'organ_broad',
'age_original', 'age_unified', 'age_continuousadult', 'age_continuousdev',
'sex', 'sample_type', 'sample_category', 'sample_retrieval', 'tissue_fraction',
'cell_fraction', 'cell_fraction_unified', 'cell_sorting', 'technology',
'include_150722', 'cluster_scrublet_score', 'bh_pval', 'scrublet_score',
'scrublet_score_z', 'doublet', 'stringent_doublet', 'integration_grouping',
'_scvi_batch', '_scvi_labels', 'broad_annot_20220914', 'martin19_pred',
'martin19_pred_prob', 'martin19_pred_uncertain', 'warner20_pred',
'warner20_pred_prob', 'warner20_pred_uncertain', 'broad_annot_20220917',
'donor_organ_lineage', 'fine_annot', 'fine_annot_original', 'level_1_annot',
'level_2_annot', 'level_3_annot', 'broad_predicted_labels',
'broad_predicted_labels_uncert', 'batch', 'organ_groups', 'control_vs_disease',
'disease', 'ID', 'clusters', 'Palantir_pseudotime'
    var: 'gene_ids', 'feature_type', 'mito', 'ribo', 'hb', 'cc', 'ig', 'tcr',
'n_counts-0', 'n_counts_raw-0', 'n_counts_spliced-0', 'n_counts_unspliced-0',
'n_cells-0', 'n_cells_raw-0', 'n_cells_spliced-0', 'n_cells_unspliced-0',
'n_counts-1', 'n_counts_raw-1', 'n_counts_spliced-1', 'n_counts_unspliced-1',
'n_cells-1', 'n_cells_raw-1', 'n_cells_spliced-1', 'n_cells_unspliced-1'
    uns: 'disease_colors', 'level_3_annot_colors', 'organ_unified_colors'
    obsm: 'X_mde', 'X_scANVI', 'X_scVI', 'X_umap', 'X_umap_scvi',
'_scvi_extra_continuous_covs'
    layers: 'counts', 'spliced', 'unspliced'
```

2 Palantir based cellrank kernel

[36]:
```
pk_palantir = cr.kernels.PseudotimeKernel(adata, time_key="Palantir_pseudotime")
pk_palantir.compute_transition_matrix()
pk_palantir.plot_projection(basis="umap_scvi", color="clusters",␣
↪legend_loc="right", recompute=True)
```

```
Computing transition matrix based on pseudotime
  0%|          | 0/20765 [00:00<?, ?cell/s]
    Finish (0:00:11)
Projecting transition matrix onto `umap_scvi`
Adding `adata.obsm['T_fwd_umap_scvi']`
    Finish (0:00:07)
```



[37]:
```
adata.obsp['connectivities'].data
```

[37]:
```
array([0.11081272, 0.16505973, 0.04109649, …, 0.0593112 , 0.1033306 ,
       0.2130326 ], dtype=float32)
```

[38]:
```
#pk_palantir = cr.kernels.PseudotimeKernel(adata_original,␣
 ↪time_key="Palantir_pseudotime")
#pk_palantir.compute_transition_matrix()
#pk_palantir.plot_projection(basis="umap_scvi", color="clusters",␣
 ↪legend_loc="right", recompute=True)
```

[39]:
```
def compute_connectivity_kernel(nn_basis, umap_basis):
    # === COMPUTE THE GENE EXPRESSION SIMILARITY BASED KERNEL (as RNA velocity␣
 ↪could be noisy)
    sc.pp.normalize_per_cell(adata, 10000)
    sc.pp.log1p(adata)
    sc.tl.pca(adata)
    sc.pp.neighbors(adata, use_rep=nn_basis,method='gauss')
    ck = cr.kernels.ConnectivityKernel(adata)
    ck.compute_transition_matrix()
    ck.plot_projection(basis=umap_basis, color="clusters", legend_loc="right",␣
 ↪recompute=True)
    return ck



def compute_DPT_kernel(adata, umap_basis):
    # === COMPUTE THE Diffusion time pseudotime KERNEL
    sc.tl.diffmap(adata)
```

```
    np.random.seed(0)
    c = np.random.choice(adata[adata.obs.clusters == 'Epithelial_stem'].
↪obs_names)
    root_ixs = np.where(adata.obs_names == c) [0][0]
    scv.pl.scatter(
        adata,
        basis=umap_basis,
        c=["clusters", root_ixs],
        legend_loc="right", #    components=["2, 3"],
    )
    adata.uns["iroot"] = root_ixs
    sc.tl.dpt(adata)
    sc.pl.embedding(
        adata,
        basis=umap_basis,
        color=["dpt_pseudotime"],
        color_map="gnuplot2",
    )
    pk = cr.kernels.PseudotimeKernel(adata, time_key="dpt_pseudotime")
    pk.compute_transition_matrix()
    pk.plot_projection(basis="umap_scvi", color="clusters", legend_loc="right",␣
↪recompute=True)
    return pk , adata
```

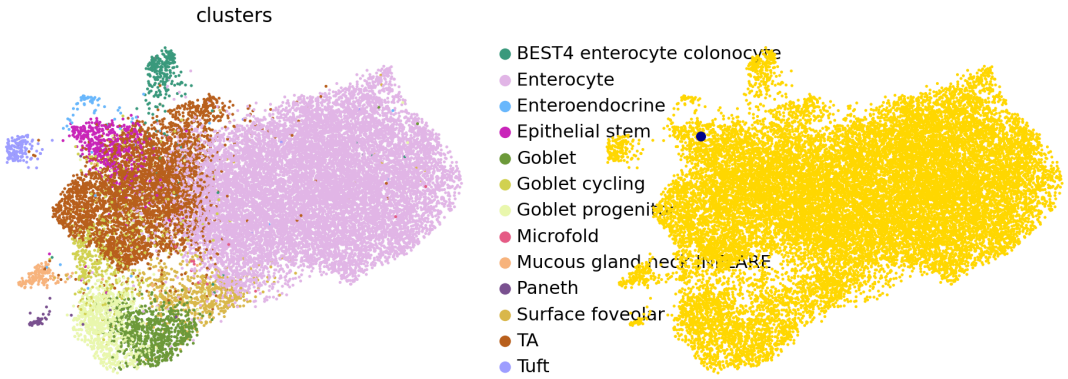
3 Diffusion time based pseudotime kernel

[40]:
```
#sc.tl.score_genes(adata, gene_list=['LGR5', 'ASCL2', 'SMOC2', 'RGMB', 'OLFM4']␣
 ↪ , score_name='start_score')
#start_cell_id = np.argmax(adata.obs.start_score)
#start_cell =  adata.obs_names[start_cell_id]
#start_cell_id
#start_cell_id
```

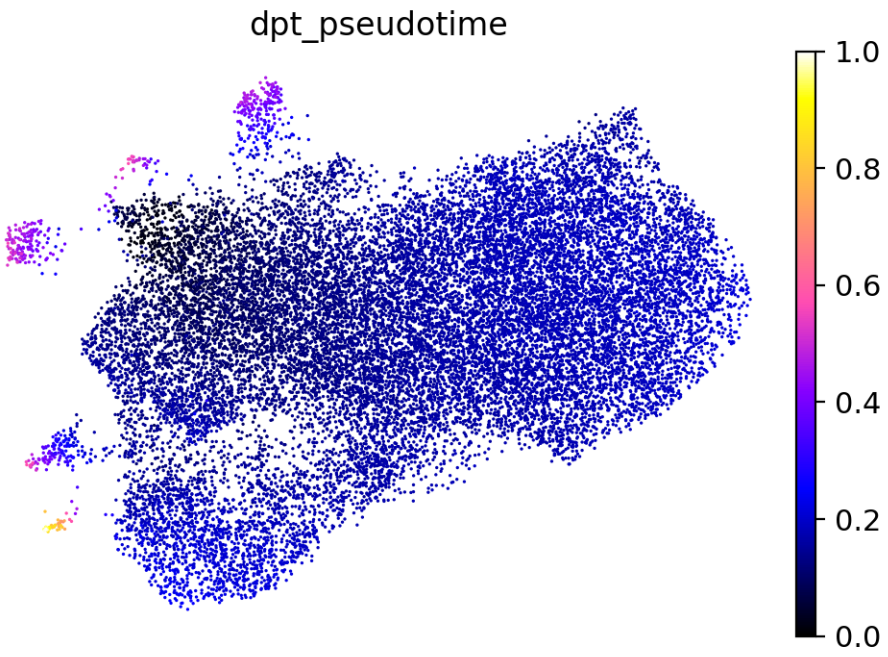
[ ]:
```

```

[41]:
```
    umap_basis = 'umap_scvi'
    sc.tl.diffmap(adata)
    np.random.seed(0)
    scv.pl.scatter(
        adata,
        basis=umap_basis,
        c=["clusters", start_cell_id],
        legend_loc="right", #    components=["2, 3"],
    )
    adata.uns["iroot"] = start_cell_id
    sc.tl.dpt(adata)
```

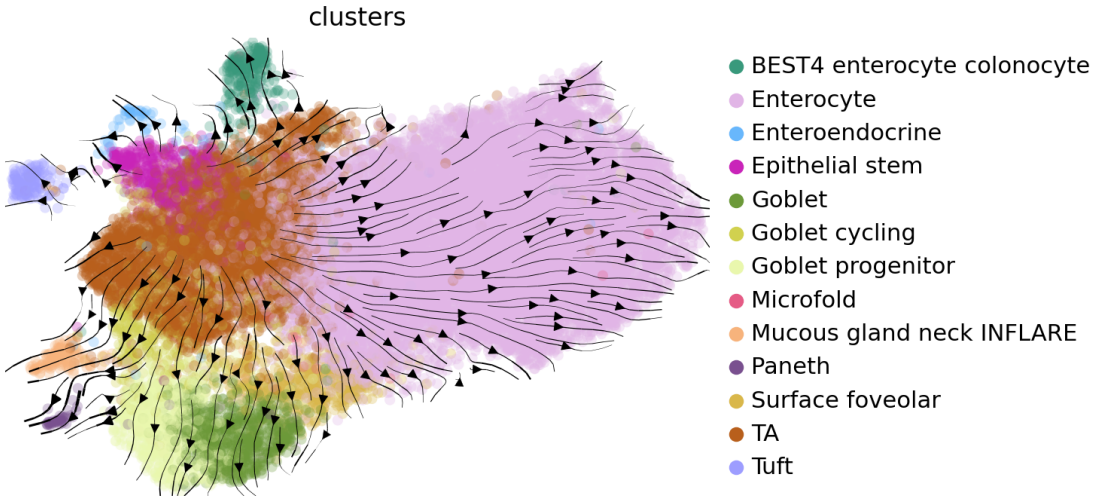
```
    sc.pl.embedding(
        adata,
        basis=umap_basis,
        color=["dpt_pseudotime"],
        color_map="gnuplot2",
    )
    pk = cr.kernels.PseudotimeKernel(adata, time_key="dpt_pseudotime")
    pk.compute_transition_matrix()
    pk.plot_projection(basis="umap_scvi", color="clusters", legend_loc="right",␣
↪recompute=True,palette = our_cmap)
```




```
/opt/conda/envs/pygpcca_env/lib/python3.8/site-
packages/scanpy/plotting/_tools/scatterplots.py:163:
MatplotlibDeprecationWarning: The get_cmap function was deprecated in Matplotlib
3.7 and will be removed two minor releases later. Use
``matplotlib.colormaps[name]`` or ``matplotlib.colormaps.get_cmap(obj)``
instead.
  cmap = copy(get_cmap(cmap))
```

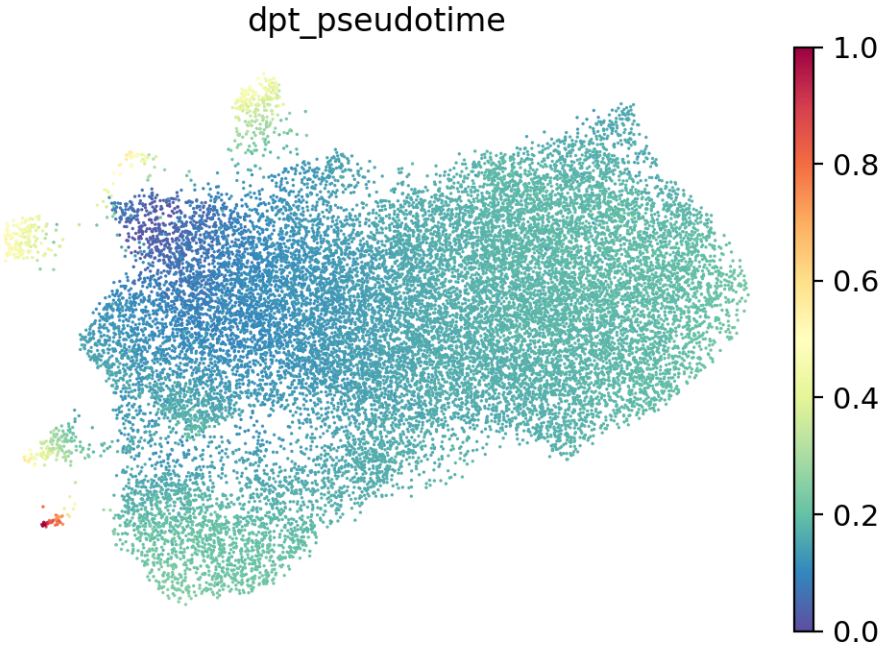



```
Computing transition matrix based on pseudotime
  0%|          | 0/20765 [00:00<?, ?cell/s]
    Finish (0:00:11)
Projecting transition matrix onto `umap_scvi`
Adding `adata.obsm['T_fwd_umap_scvi']`
    Finish (0:00:07)
```



[42]:
```
sc.pl.embedding(adata, basis ='umap_scvi', color=['dpt_pseudotime'],␣
↪cmap='Spectral_r')
```

```
/opt/conda/envs/pygpcca_env/lib/python3.8/site-
packages/scanpy/plotting/_tools/scatterplots.py:163:
MatplotlibDeprecationWarning: The get_cmap function was deprecated in Matplotlib
3.7 and will be removed two minor releases later. Use
``matplotlib.colormaps[name]`` or ``matplotlib.colormaps.get_cmap(obj)``
instead.
  cmap = copy(get_cmap(cmap))
```



[43]:
```
#scv.tl.paga(adata, groups='clusters', use_time_prior='dpt_pseudotime')
#scv.pl.paga(adata, basis='umap_scvi', size=50, alpha=.1,
#            min_edge_width=2, node_size_scale=1.5, palette = our_cmap)
```

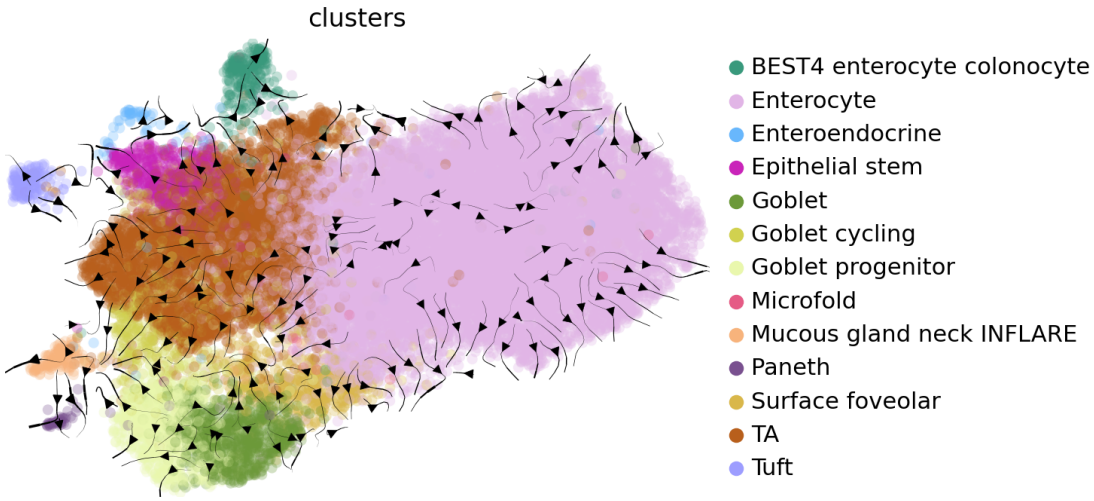
[44]:
```
adata.obsp['connectivities'].data
```

[44]:
```
array([0.110812716, 0.16505973 , 0.04109649 , …, 0.0593112  ,
       0.103330605, 0.2130326  ], dtype=float32)
```

[45]:
```
def compute_connectivity_kernel(adata, nn_basis, umap_basis):
    # === COMPUTE THE GENE EXPRESSION SIMILARITY BASED KERNEL (as RNA velocity␣
↪could be noisy)
  # sc.pp.normalize_per_cell(adata, 10000)
  # sc.pp.log1p(adata)
```

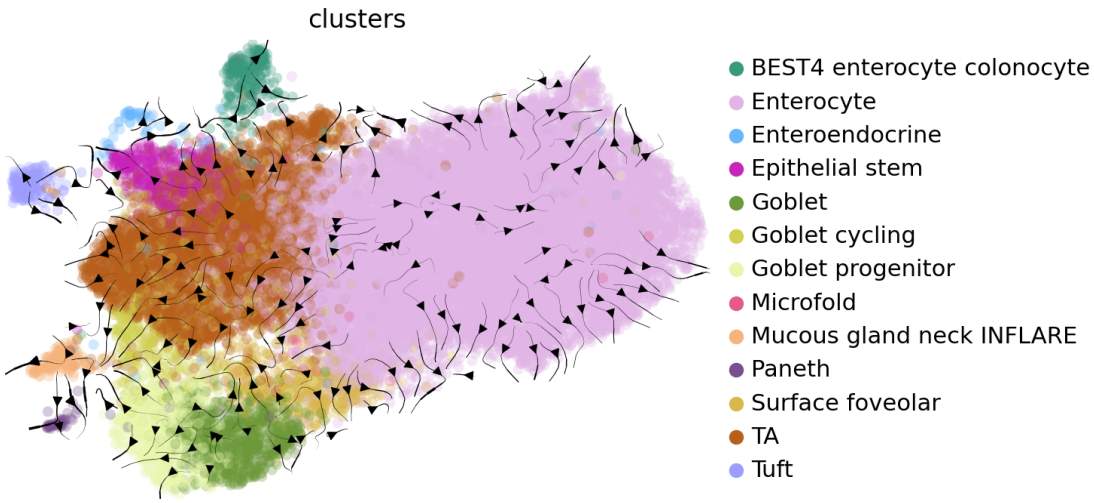
```
    # sc.tl.pca(adata)
    # sc.pp.neighbors(adata, use_rep=nn_basis,method='gauss')
     ck = cr.kernels.ConnectivityKernel(adata) # transcriptomoic similarities␣
 ↪computed using neighbours
     ck.compute_transition_matrix()
     ck.plot_projection(basis=umap_basis, color="clusters", legend_loc="right",␣
 ↪recompute=True)
     return ck
connectivity_kernel_scvi = compute_connectivity_kernel(adata, nn_basis =␣
 ↪"X_scVI", umap_basis = "umap_scvi")
```

```
Computing transition matrix based on `adata.obsp['connectivities']`
    Finish (0:00:00)
Projecting transition matrix onto `umap_scvi`
Adding `adata.obsm['T_fwd_umap_scvi']`
    Finish (0:00:07)
```



[46]:
```
connectivity_kernel_scvi.plot_projection(basis="umap_scvi", color="clusters",␣
 ↪legend_loc="right", recompute=True, palette = our_cmap)
```

```
Projecting transition matrix onto `umap_scvi`
Adding `adata.obsm['T_fwd_umap_scvi']`
    Finish (0:00:07)
```



[47]:
```
adata.obsp['connectivities'].data
```

[47]:
```
array([0.110812716, 0.16505973 , 0.04109649 , …, 0.0593112  ,
       0.103330605, 0.2130326  ], dtype=float32)
```

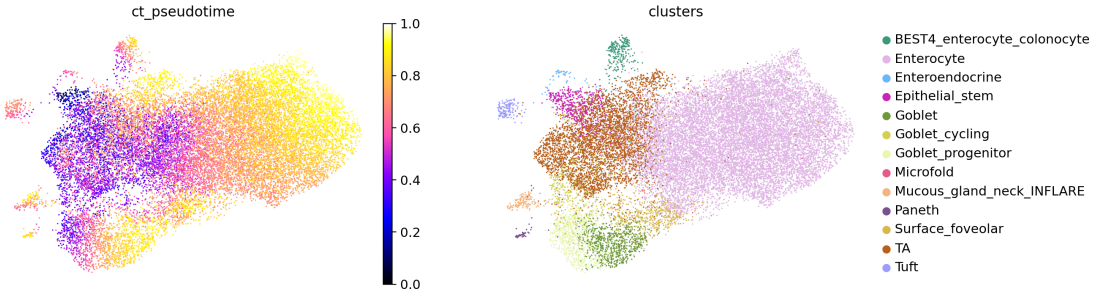
4 CYTOTRACE

[48]:
```
def compute_cytoTRACE_kernel(adata, umap_basis):
    # === COMPUTE THE CytoTRACE differentiation potential based pseudotime␣
 ↪KERNEL
    ctk = CytoTRACEKernel(adata).compute_cytotrace()
    sc.pl.embedding(
        adata,
        color=["ct_pseudotime", "clusters"],
        basis=umap_basis,
        color_map="gnuplot2",
    )
    ctk.compute_transition_matrix(threshold_scheme="soft", nu=0.5)
    #  - ```'hard'``` removes some edges that point against the direction of␣
 ↪increasing pseudotime.
    # To avoid disconnecting the graph, it does not remove all edges that point␣
 ↪against the direction of increasing pseudotime,
    # but keeps the ones that point to cells inside a close radius. This radius␣
 ↪is chosen according to the local cell density.
    #   - ```'soft'```  down-weights edges that points against the direction of␣
 ↪increasing pseudotime. Essentially, the further "behind"
    # a query cell is in pseudotime with respect to the current reference cell,␣
 ↪the more penalized will be its graph-connectivity.
```

```
    return ctk , adata

cytoTRACE_kernel, adata = compute_cytoTRACE_kernel(adata, umap_basis =␣
 ↪'umap_scvi')
```

```
Computing CytoTRACE score with `10845` genes
Adding `adata.obs['ct_score']`
       `adata.obs['ct_pseudotime']`
       `adata.obs['ct_num_exp_genes']`
       `adata.var['ct_gene_corr']`
       `adata.var['ct_correlates']`
       `adata.uns['ct_params']`
    Finish (0:00:01)
```

```
/opt/conda/envs/pygpcca_env/lib/python3.8/site-
packages/scanpy/plotting/_tools/scatterplots.py:163:
MatplotlibDeprecationWarning: The get_cmap function was deprecated in Matplotlib
3.7 and will be removed two minor releases later. Use
``matplotlib.colormaps[name]`` or ``matplotlib.colormaps.get_cmap(obj)``
instead.
  cmap = copy(get_cmap(cmap))
```



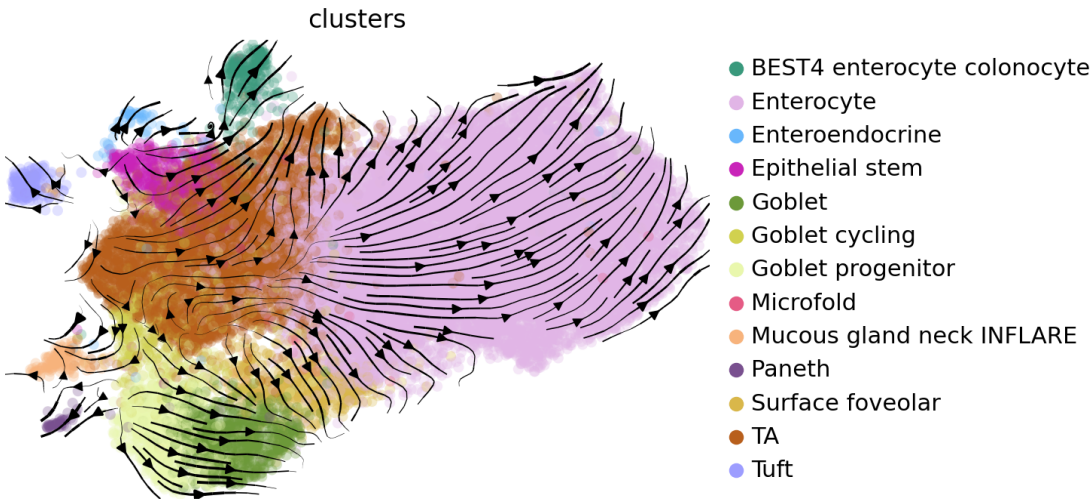

```
Computing transition matrix based on pseudotime
```

```
  0%|          | 0/20765 [00:00<?, ?cell/s]
```

```
    Finish (0:00:13)
```

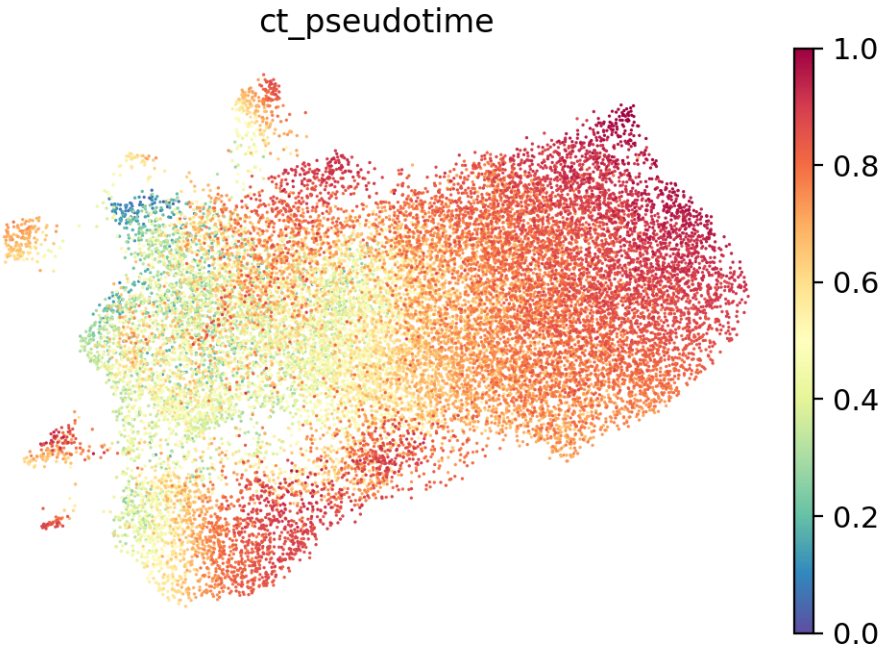
[49]:
```
cytoTRACE_kernel.plot_projection(basis="umap_scvi", color="clusters",␣
 ↪legend_loc="right", recompute=True, palette = our_cmap)
```

```
Projecting transition matrix onto `umap_scvi`
Adding `adata.obsm['T_fwd_umap_scvi']`
    Finish (0:00:07)
```



[50]:
```
sc.pl.embedding(adata, basis='umap_scvi', color=['ct_pseudotime'],␣
↪cmap='Spectral_r')
```

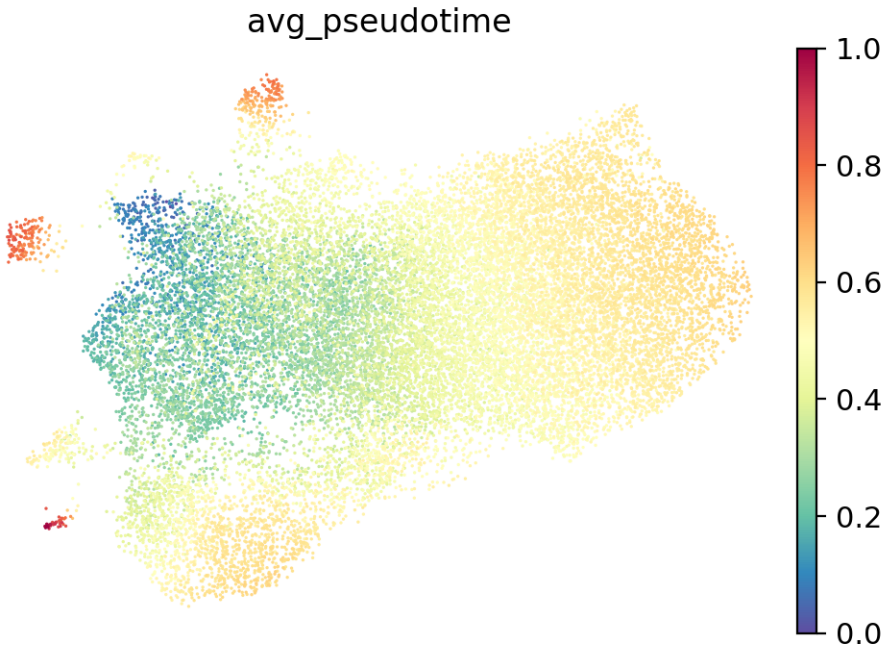
```
/opt/conda/envs/pygpcca_env/lib/python3.8/site-
packages/scanpy/plotting/_tools/scatterplots.py:163:
MatplotlibDeprecationWarning: The get_cmap function was deprecated in Matplotlib
3.7 and will be removed two minor releases later. Use
``matplotlib.colormaps[name]`` or ``matplotlib.colormaps.get_cmap(obj)``
instead.
  cmap = copy(get_cmap(cmap))
```



[51]:
```
#scv.tl.paga(adata, groups='clusters', use_time_prior='ct_pseudotime')
#scv.pl.paga(adata, basis='umap_scvi', size=50, alpha=.1,
#            min_edge_width=2, node_size_scale=1.5, palette = our_cmap)
```

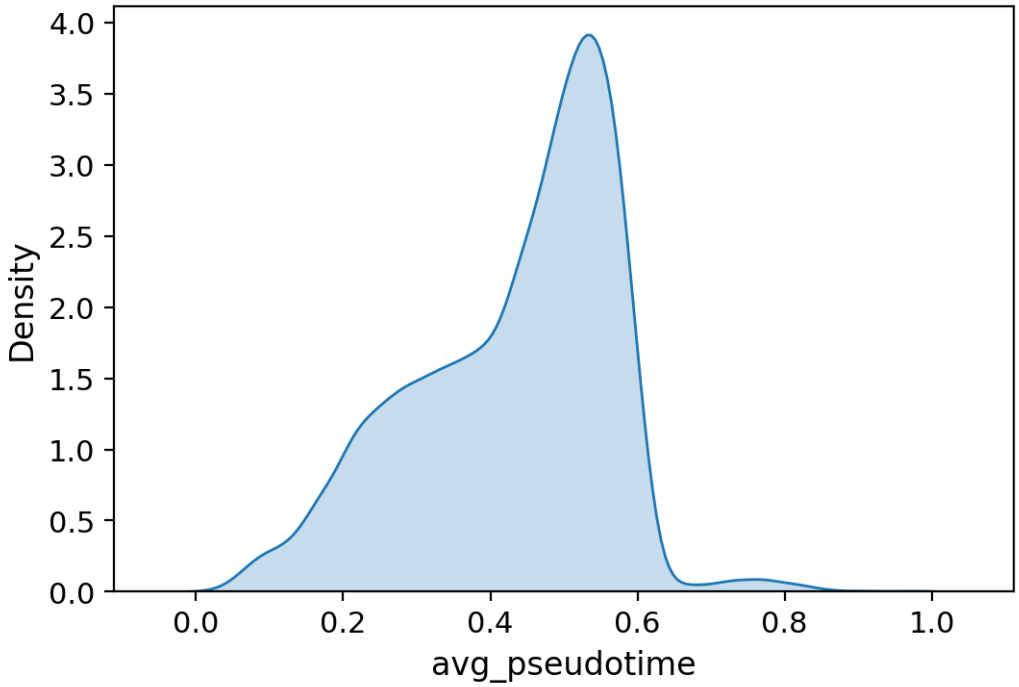
[52]:
```
adata.obs['avg_pseudotime'] = (adata.obs.ct_pseudotime + adata.obs.
 ↪dpt_pseudotime + adata.obs.Palantir_pseudotime)/3
_min = np.min(adata.obs['avg_pseudotime'])
_max = np.max(adata.obs['avg_pseudotime'])
adata.obs['avg_pseudotime'] = (adata.obs['avg_pseudotime'] - _min)/(_max-_min)
sc.pl.embedding(adata, basis='umap_scvi', color=['avg_pseudotime'],␣
 ↪cmap='Spectral_r')
#scv.tl.paga(adata, groups='clusters', use_time_prior='avg_pseudotime')
#scv.pl.paga(adata, basis='umap_scvi', size=50, alpha=.1,
 #            min_edge_width=2, node_size_scale=1.5, palette = our_cmap)
```

```
/opt/conda/envs/pygpcca_env/lib/python3.8/site-
packages/scanpy/plotting/_tools/scatterplots.py:163:
MatplotlibDeprecationWarning: The get_cmap function was deprecated in Matplotlib
3.7 and will be removed two minor releases later. Use
``matplotlib.colormaps[name]`` or ``matplotlib.colormaps.get_cmap(obj)``
instead.
  cmap = copy(get_cmap(cmap))
```



[53]:
```
import seaborn as sb
sb.kdeplot(adata.obs['avg_pseudotime'], fill=True)
```

[53]:
```
<Axes: xlabel='avg_pseudotime', ylabel='Density'>
```

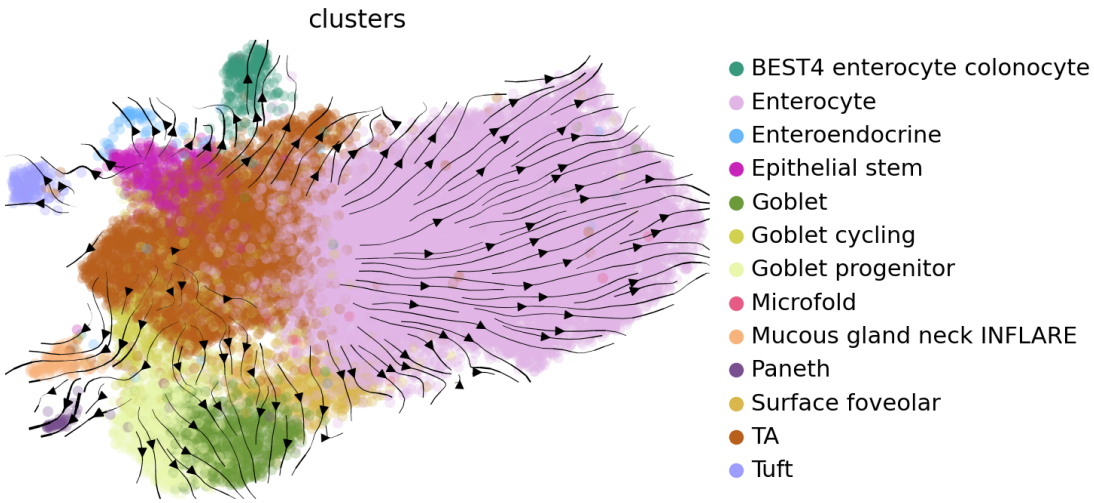


[54]:
```
adata.obsp['connectivities'].data
```

[54]:
```
array([0.110812716, 0.16505973 , 0.04109649 , …, 0.0593112  ,
       0.103330605, 0.2130326  ], dtype=float32)
```

[55]:
```
    pk_avg = cr.kernels.PseudotimeKernel(adata, time_key="avg_pseudotime")
    pk_avg.compute_transition_matrix()
    pk_avg.plot_projection(basis="umap_scvi", color="clusters",␣
 ↪legend_loc="right", recompute=True,palette = our_cmap)
```

```
Computing transition matrix based on pseudotime
  0%|          | 0/20765 [00:00<?, ?cell/s]
    Finish (0:00:10)
Projecting transition matrix onto `umap_scvi`
Adding `adata.obsm['T_fwd_umap_scvi']`
    Finish (0:00:07)
```



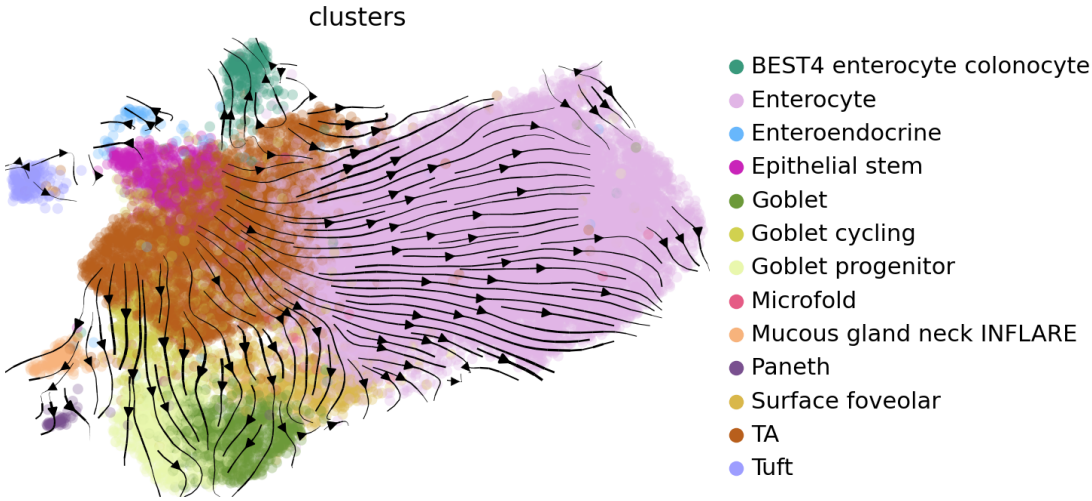
[56]:
```
#adata_all = adata_original.copy()
#connectivity_kernel_scvi = compute_connectivity_kernel(nn_basis = "X_scVI",␣
 ↪umap_basis = "umap")
#adata_all = adata_original.copy()
#connectivity_kernel_pca = compute_connectivity_kernel(nn_basis = "X_pca",␣
 ↪umap_basis = "umap")
#cytoTRACE_kernel = compute_cytoTRACE_kernel(umap_basis = 'umap')
#dpt_kernel = compute_DPT_kernel(umap_basis = 'umap') #umap_scvi
```

[57]:
```
#!pip install python-igraph --upgrade --quiet
```

5 Combining kernels

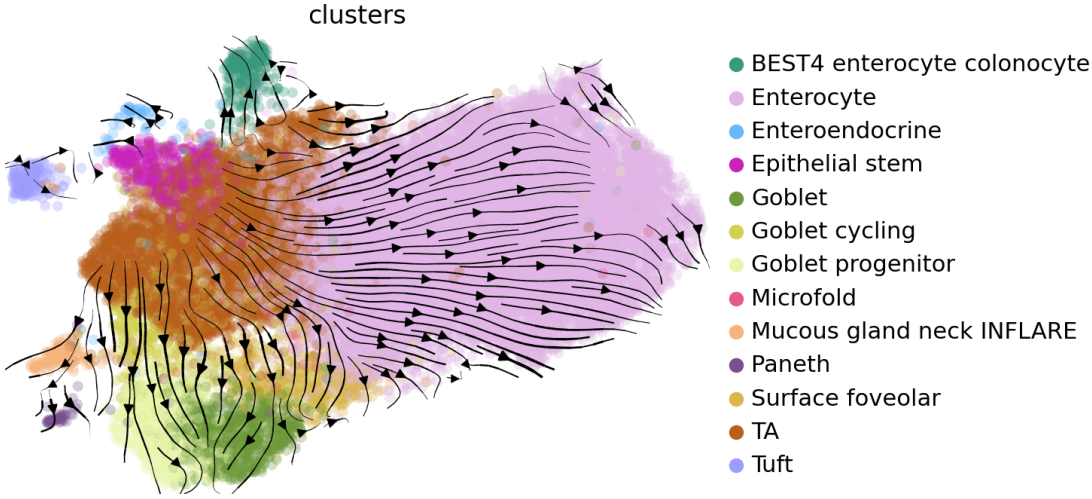
[58]:
```
# velo kernel
# cytoTRACE kernel
# palantir kernel
# dpt kernel
# scvi connectivity kernel
## ============================== COMBINING DIFFERENT KERNELS␣
 ↪==============================
combined_kernel =  (0.75*pk_palantir) + (0.25*velocity_kernel)
print(combined_kernel)
combined_kernel.plot_projection(basis="umap_scvi", color="clusters",␣
 ↪legend_loc="right", recompute=True, palette = our_cmap)
```

```
(0.75 * PseudotimeKernel[n=20765] + 0.25 * VelocityKernel[n=20765])
Projecting transition matrix onto `umap_scvi`
Adding `adata.obsm['T_fwd_umap_scvi']`
    Finish (0:00:07)
```



[59]:
```
## ============================ COMBINING DIFFERENT KERNELS␣
 ↪============================
combined_kernel =  (0.75*pk_palantir) + (0.25*velocity_kernel)
print(combined_kernel)
combined_kernel.plot_projection(basis="umap_scvi", color="clusters",␣
 ↪legend_loc="right", recompute=True, palette = our_cmap)
```

```
(0.75 * PseudotimeKernel[n=20765] + 0.25 * VelocityKernel[n=20765])
Projecting transition matrix onto `umap_scvi`
Adding `adata.obsm['T_fwd_umap_scvi']`
    Finish (0:00:07)
```

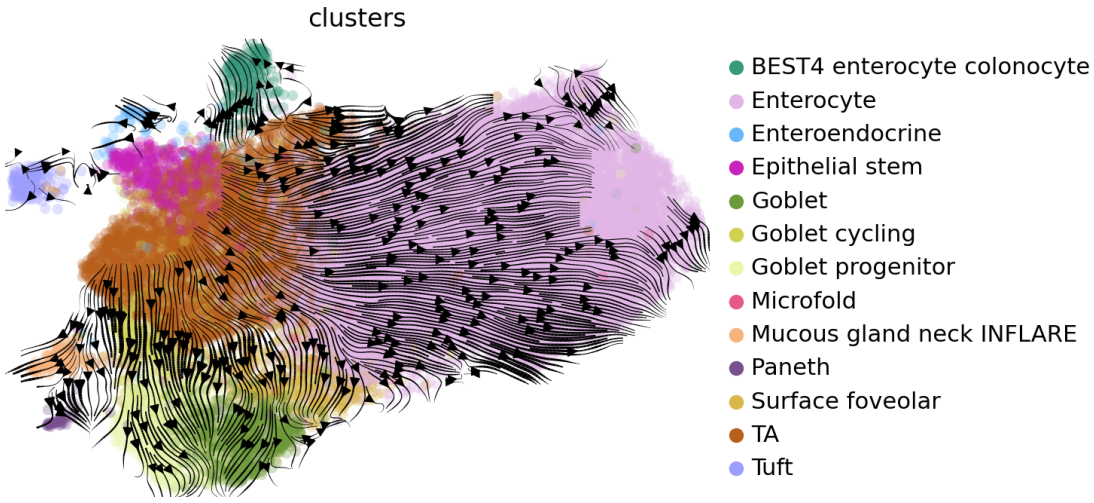


[60]:
```
#combined_kernel.plot_projection(basis="umap_scvi", color="clusters",␣
↪legend_loc="right", recompute=True, palette = our_cmap)
```

[ ]:
```
# ============================== COMBINING DIFFERENT KERNELS␣
↪==============================
#combined_kernel =  (0.25*pk_palantir) + (0.25*pk) + (0.25*velocity_kernel) +␣
↪(0.25* cytoTRACE_kernel)
#print(combined_kernel)
#combined_kernel.plot_projection(basis="umap_scvi", color="clusters",␣
↪legend_loc="right", recompute=True, palette = our_cmap)
```

[62]:
```
combined_kernel.plot_projection(basis="umap_scvi", color="clusters",␣
↪legend_loc="right", recompute=True, palette = our_cmap, density=6)
```

```
Adding `adata.obsm['T_fwd_umap_scvi']`
    Finish (0:00:07)
```



[63]:
```
# ============================== COMBINING DIFFERENT KERNELS␣
↪==============================
#combined_kernel = (0.5*ck_scvi) + (0.5*ck_pca)
#print(combined_kernel)
#combined_kernel.plot_projection(basis="umap_scvi", color="clusters",␣
↪legend_loc="right", recompute=True)
#(0.3 * VelocityKernel[n=8764] + 0.3 * ConnectivityKernel[n=8764] + 0.4 *␣
↪CytoTRACEKernel[n=8764]) GOOD for visual
```

[64]:
```
# Get RAM usage information
ram = psutil.virtual_memory()

# Convert bytes to gigabytes
```

```
total_gb = ram.total / (1024 ** 3)
used_gb = ram.used / (1024 ** 3)
free_gb = ram.free / (1024 ** 3)

print(f"Total RAM: {total_gb:.2f} GB")
print(f"Used RAM: {used_gb:.2f} GB")
print(f"Free RAM: {free_gb:.2f} GB")
```

```
Total RAM: 102.29 GB
Used RAM: 40.28 GB
Free RAM: 52.42 GB
```

[ ]:

[120]:
```
#combined_kernel.plot_projection(basis="umap_scvi", color="clusters",␣
↪legend_loc="right", recompute=True)
#(0.3 * VelocityKernel[n=8764] + 0.3 * ConnectivityKernel[n=8764] + 0.4 *␣
↪CytoTRACEKernel[n=8764]) GOOD for visual
```

[126]:
```
#combined_kernel.plot_random_walks(basis = 'umap_scvi', start_ixs={"clusters":␣
↪"Epithelial_stem"}, max_iter=200, seed=0)
```

[127]:
```
# THEN COMBINE THE TWO KERNELS
#combined_kernel = (0.3 * vk) + (0.4*ctk) + (0.3*pk)
#print(combined_kernel)
#combined_kernel.plot_projection(basis="umap_scvi", color="clusters",␣
↪legend_loc="right", recompute=True)
```

[65]:
```
adata
```

[65]:
```
AnnData object with n_obs × n_vars = 20765 × 10845
    obs: 'latent_cell_probability', 'latent_RT_efficiency', 'cecilia22_predH',
'cecilia22_predH_prob', 'cecilia22_predH_uncertain', 'cecilia22_predL',
'cecilia22_predL_prob', 'cecilia22_predL_uncertain', 'elmentaite21_pred',
'elmentaite21_pred_prob', 'elmentaite21_pred_uncertain', 'suo22_pred',
'suo22_pred_prob', 'suo22_pred_uncertain', 'n_counts', 'log1p_n_counts',
'n_genes', 'log1p_n_genes', 'percent_mito', 'n_counts_mito', 'percent_ribo',
'n_counts_ribo', 'percent_hb', 'n_counts_hb', 'percent_top50', 'n_counts_raw',
'log1p_n_counts_raw', 'n_genes_raw', 'log1p_n_genes_raw', 'percent_mito_raw',
'n_counts_mito_raw', 'percent_ribo_raw', 'n_counts_ribo_raw', 'percent_hb_raw',
'n_counts_hb_raw', 'percent_top50_raw', 'n_counts_spliced',
'log1p_n_counts_spliced', 'n_genes_spliced', 'log1p_n_genes_spliced',
'percent_mito_spliced', 'n_counts_mito_spliced', 'percent_ribo_spliced',
'n_counts_ribo_spliced', 'percent_hb_spliced', 'n_counts_hb_spliced',
'percent_top50_spliced', 'n_counts_unspliced', 'log1p_n_counts_unspliced',
'n_genes_unspliced', 'log1p_n_genes_unspliced', 'percent_mito_unspliced',
'n_counts_mito_unspliced', 'percent_ribo_unspliced', 'n_counts_ribo_unspliced',
'percent_hb_unspliced', 'n_counts_hb_unspliced', 'percent_top50_unspliced',
```

```
'percent_soup', 'percent_spliced', 'qc_cluster', 'pass_auto_filter_mito20',
'good_qc_cluster_mito20', 'pass_auto_filter_mito50', 'good_qc_cluster_mito50',
'pass_auto_filter_mito80', 'good_qc_cluster_mito80', 'pass_auto_filter',
'good_qc_cluster', 'pass_default', 'sampleID', 'sourceID', 'donorID_original',
'study', 'donorID_corrected', 'donorID_unified', 'donor_category',
'donor_disease', 'organ_original', 'organ_unified', 'organ_broad',
'age_original', 'age_unified', 'age_continuousadult', 'age_continuousdev',
'sex', 'sample_type', 'sample_category', 'sample_retrieval', 'tissue_fraction',
'cell_fraction', 'cell_fraction_unified', 'cell_sorting', 'technology',
'include_150722', 'cluster_scrublet_score', 'bh_pval', 'scrublet_score',
'scrublet_score_z', 'doublet', 'stringent_doublet', 'integration_grouping',
'_scvi_batch', '_scvi_labels', 'broad_annot_20220914', 'martin19_pred',
'martin19_pred_prob', 'martin19_pred_uncertain', 'warner20_pred',
'warner20_pred_prob', 'warner20_pred_uncertain', 'broad_annot_20220917',
'donor_organ_lineage', 'fine_annot', 'fine_annot_original', 'level_1_annot',
'level_2_annot', 'level_3_annot', 'broad_predicted_labels',
'broad_predicted_labels_uncert', 'batch', 'organ_groups', 'control_vs_disease',
'disease', 'ID', 'clusters', 'Palantir_pseudotime', 'initial_size_unspliced',
'initial_size_spliced', 'initial_size', 'velocity_self_transition',
'root_cells', 'end_points', 'velocity_pseudotime', 'start_score',
'dpt_pseudotime', 'ct_score', 'ct_pseudotime', 'ct_num_exp_genes',
'avg_pseudotime'
    var: 'gene_ids', 'feature_type', 'mito', 'ribo', 'hb', 'cc', 'ig', 'tcr',
'n_counts-0', 'n_counts_raw-0', 'n_counts_spliced-0', 'n_counts_unspliced-0',
'n_cells-0', 'n_cells_raw-0', 'n_cells_spliced-0', 'n_cells_unspliced-0',
'n_counts-1', 'n_counts_raw-1', 'n_counts_spliced-1', 'n_counts_unspliced-1',
'n_cells-1', 'n_cells_raw-1', 'n_cells_spliced-1', 'n_cells_unspliced-1',
'gene_count_corr', 'means', 'dispersions', 'dispersions_norm',
'highly_variable', 'fit_r2', 'fit_alpha', 'fit_beta', 'fit_gamma', 'fit_t_',
'fit_scaling', 'fit_std_u', 'fit_std_s', 'fit_likelihood', 'fit_u0', 'fit_s0',
'fit_pval_steady', 'fit_steady_u', 'fit_steady_s', 'fit_variance',
'fit_alignment_scaling', 'velocity_genes', 'ct_gene_corr', 'ct_correlates'
    uns: 'disease_colors', 'level_3_annot_colors', 'organ_unified_colors',
'log1p', 'pca', 'neighbors', 'recover_dynamics', 'velocity_params',
'T_fwd_params', 'clusters_colors', 'velocity_graph', 'velocity_graph_neg',
'diffmap_evals', 'iroot', 'ct_params'
    obsm: 'X_mde', 'X_scANVI', 'X_scVI', 'X_umap', 'X_umap_scvi',
'_scvi_extra_continuous_covs', 'X_pca', 'T_fwd_umap_scvi', 'T_fwd_pca',
'velocity_umap_scvi', 'X_diffmap'
    varm: 'PCs', 'loss'
    layers: 'counts', 'spliced', 'unspliced', 'Ms', 'Mu', 'fit_t', 'fit_tau',
'fit_tau_', 'velocity', 'velocity_u'
    obsp: 'distances', 'connectivities'
```

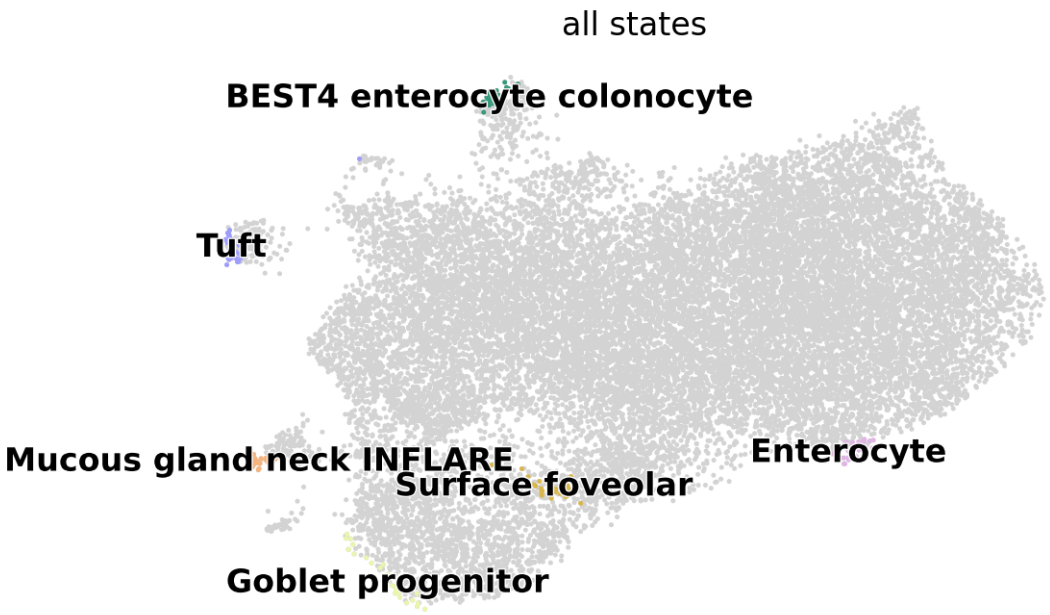
[66]:
```
combined_kernel
```

[66]:
```
(0.75 * PseudotimeKernel[n=20765, dnorm=False, scheme='hard', frac_to_keep=0.3]
+ 0.25 * VelocityKernel[n=20765, model='deterministic',
similarity='correlation', softmax_scale=7.491])
```

[67]:
```
del adata_all
```

[68]:
```
from cellrank.estimators import GPCCA

g = GPCCA(combined_kernel)
print(g)
g.fit(n_states=6, cluster_key="clusters")
g.plot_macrostates(which="all", basis='umap_scvi')
```

```
GPCCA[kernel=(0.75 * PseudotimeKernel[n=20765] + 0.25 *
VelocityKernel[n=20765]), initial_states=None, terminal_states=None]
WARNING: Unable to import `petsc4py` or `slepc4py`. Using `method='brandts'`
WARNING: For `method='brandts'`, dense matrix is required. Densifying
Computing Schur decomposition
Adding `adata.uns['eigendecomposition_fwd']`
       `.schur_vectors`
       `.schur_matrix`
       `.eigendecomposition`
    Finish (0:27:30)
Computing `6` macrostates
Adding `.macrostates`
       `.macrostates_memberships`
       `.coarse_T`
       `.coarse_initial_distribution
       `.coarse_stationary_distribution`
       `.schur_vectors`
       `.schur_matrix`
       `.eigendecomposition`
    Finish (0:00:43)
```



[69]:
```
adata
```

[69]:
```
AnnData object with n_obs × n_vars = 20765 × 10845
    obs: 'latent_cell_probability', 'latent_RT_efficiency', 'cecilia22_predH',
'cecilia22_predH_prob', 'cecilia22_predH_uncertain', 'cecilia22_predL',
'cecilia22_predL_prob', 'cecilia22_predL_uncertain', 'elmentaite21_pred',
'elmentaite21_pred_prob', 'elmentaite21_pred_uncertain', 'suo22_pred',
'suo22_pred_prob', 'suo22_pred_uncertain', 'n_counts', 'log1p_n_counts',
'n_genes', 'log1p_n_genes', 'percent_mito', 'n_counts_mito', 'percent_ribo',
'n_counts_ribo', 'percent_hb', 'n_counts_hb', 'percent_top50', 'n_counts_raw',
'log1p_n_counts_raw', 'n_genes_raw', 'log1p_n_genes_raw', 'percent_mito_raw',
'n_counts_mito_raw', 'percent_ribo_raw', 'n_counts_ribo_raw', 'percent_hb_raw',
'n_counts_hb_raw', 'percent_top50_raw', 'n_counts_spliced',
'log1p_n_counts_spliced', 'n_genes_spliced', 'log1p_n_genes_spliced',
'percent_mito_spliced', 'n_counts_mito_spliced', 'percent_ribo_spliced',
'n_counts_ribo_spliced', 'percent_hb_spliced', 'n_counts_hb_spliced',
'percent_top50_spliced', 'n_counts_unspliced', 'log1p_n_counts_unspliced',
'n_genes_unspliced', 'log1p_n_genes_unspliced', 'percent_mito_unspliced',
'n_counts_mito_unspliced', 'percent_ribo_unspliced', 'n_counts_ribo_unspliced',
'percent_hb_unspliced', 'n_counts_hb_unspliced', 'percent_top50_unspliced',
'percent_soup', 'percent_spliced', 'qc_cluster', 'pass_auto_filter_mito20',
'good_qc_cluster_mito20', 'pass_auto_filter_mito50', 'good_qc_cluster_mito50',
'pass_auto_filter_mito80', 'good_qc_cluster_mito80', 'pass_auto_filter',
'good_qc_cluster', 'pass_default', 'sampleID', 'sourceID', 'donorID_original',
'study', 'donorID_corrected', 'donorID_unified', 'donor_category',
```

```
'donor_disease', 'organ_original', 'organ_unified', 'organ_broad',
'age_original', 'age_unified', 'age_continuousadult', 'age_continuousdev',
'sex', 'sample_type', 'sample_category', 'sample_retrieval', 'tissue_fraction',
'cell_fraction', 'cell_fraction_unified', 'cell_sorting', 'technology',
'include_150722', 'cluster_scrublet_score', 'bh_pval', 'scrublet_score',
'scrublet_score_z', 'doublet', 'stringent_doublet', 'integration_grouping',
'_scvi_batch', '_scvi_labels', 'broad_annot_20220914', 'martin19_pred',
'martin19_pred_prob', 'martin19_pred_uncertain', 'warner20_pred',
'warner20_pred_prob', 'warner20_pred_uncertain', 'broad_annot_20220917',
'donor_organ_lineage', 'fine_annot', 'fine_annot_original', 'level_1_annot',
'level_2_annot', 'level_3_annot', 'broad_predicted_labels',
'broad_predicted_labels_uncert', 'batch', 'organ_groups', 'control_vs_disease',
'disease', 'ID', 'clusters', 'Palantir_pseudotime', 'initial_size_unspliced',
'initial_size_spliced', 'initial_size', 'velocity_self_transition',
'root_cells', 'end_points', 'velocity_pseudotime', 'start_score',
'dpt_pseudotime', 'ct_score', 'ct_pseudotime', 'ct_num_exp_genes',
'avg_pseudotime', 'macrostates_fwd'
    var: 'gene_ids', 'feature_type', 'mito', 'ribo', 'hb', 'cc', 'ig', 'tcr',
'n_counts-0', 'n_counts_raw-0', 'n_counts_spliced-0', 'n_counts_unspliced-0',
'n_cells-0', 'n_cells_raw-0', 'n_cells_spliced-0', 'n_cells_unspliced-0',
'n_counts-1', 'n_counts_raw-1', 'n_counts_spliced-1', 'n_counts_unspliced-1',
'n_cells-1', 'n_cells_raw-1', 'n_cells_spliced-1', 'n_cells_unspliced-1',
'gene_count_corr', 'means', 'dispersions', 'dispersions_norm',
'highly_variable', 'fit_r2', 'fit_alpha', 'fit_beta', 'fit_gamma', 'fit_t_',
'fit_scaling', 'fit_std_u', 'fit_std_s', 'fit_likelihood', 'fit_u0', 'fit_s0',
'fit_pval_steady', 'fit_steady_u', 'fit_steady_s', 'fit_variance',
'fit_alignment_scaling', 'velocity_genes', 'ct_gene_corr', 'ct_correlates'
    uns: 'disease_colors', 'level_3_annot_colors', 'organ_unified_colors',
'log1p', 'pca', 'neighbors', 'recover_dynamics', 'velocity_params',
'T_fwd_params', 'clusters_colors', 'velocity_graph', 'velocity_graph_neg',
'diffmap_evals', 'iroot', 'ct_params', 'schur_matrix_fwd',
'eigendecomposition_fwd', 'macrostates_fwd_colors', 'coarse_fwd'
    obsm: 'X_mde', 'X_scANVI', 'X_scVI', 'X_umap', 'X_umap_scvi',
'_scvi_extra_continuous_covs', 'X_pca', 'T_fwd_umap_scvi', 'T_fwd_pca',
'velocity_umap_scvi', 'X_diffmap', 'schur_vectors_fwd',
'macrostates_fwd_memberships'
    varm: 'PCs', 'loss'
    layers: 'counts', 'spliced', 'unspliced', 'Ms', 'Mu', 'fit_t', 'fit_tau',
'fit_tau_', 'velocity', 'velocity_u'
    obsp: 'distances', 'connectivities'
```

[70]:
```
g.predict_terminal_states(method="top_n", n_states=6, allow_overlap=True)
```

```
Adding `adata.obs['term_states_fwd']`
       `adata.obs['term_states_fwd_probs']`
       `.terminal_states`
       `.terminal_states_probabilities`
       `.terminal_states_memberships
```

```
    Finish`
```

[70]:
```
GPCCA[kernel=(0.75 * PseudotimeKernel[n=20765] + 0.25 *
VelocityKernel[n=20765]), initial_states=None,
terminal_states=['BEST4_enterocyte_colonocyte', 'Enterocyte',
'Goblet_progenitor', 'Mucous_gland_neck_INFLARE', 'Surface_foveolar', 'Tuft']]
```

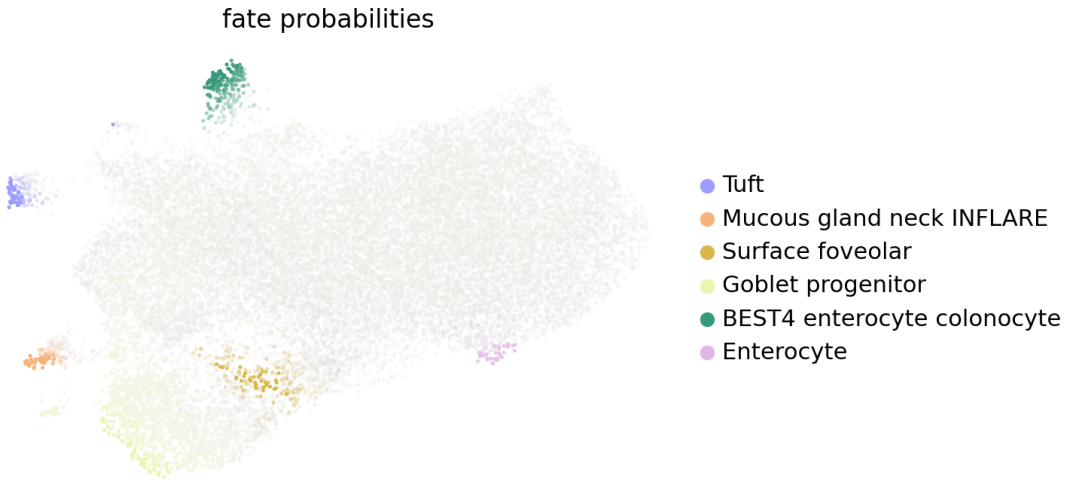
[71]:
```
g.predict_initial_states(n_states=1, allow_overlap=True) # not correct
```

```
Adding `adata.obs['init_states_fwd']`
       `adata.obs['init_states_fwd_probs']`
       `.initial_states`
       `.initial_states_probabilities`
       `.initial_states_memberships
    Finish`
```

[71]:
```
GPCCA[kernel=(0.75 * PseudotimeKernel[n=20765] + 0.25 *
VelocityKernel[n=20765]), initial_states=['Mucous_gland_neck_INFLARE'],
terminal_states=['BEST4_enterocyte_colonocyte', 'Enterocyte',
'Goblet_progenitor', 'Mucous_gland_neck_INFLARE', 'Surface_foveolar', 'Tuft']]
```

[72]:
```
g.compute_fate_probabilities()
g.plot_fate_probabilities(legend_loc="right", basis='umap_scvi')
```

```
Computing fate probabilities
WARNING: Unable to import petsc4py. For installation, please refer to:
https://petsc4py.readthedocs.io/en/stable/install.html.
Defaulting to `'gmres'` solver.

  0%|          | 0/6 [00:00<?, ?/s]

Adding `adata.obsm['lineages_fwd']`
       `.fate_probabilities`
    Finish (0:00:03)
```



[73]:
```
adata.obsm['term_states_fwd_memberships']
```

[73]:
```
Lineage([[0.032987137 , 0.0081068575, 0.3424707551, 0.359191934 ,
          0.0072092288, 0.2500340876],
         [0.0498111056, 0.0143850249, 0.3444279032, 0.3100691725,
          0.0085562264, 0.2727505673],
         [0.0337544252, 0.0055603371, 0.0796737439, 0.1128049037,
          0.0036994481, 0.7645071419],
         …,
         [0.0032452972, 0.0032945462, 0.3161376231, 0.6641767585,
          0.003740275 , 0.0094055   ],
         [0.0235487815, 0.0089109814, 0.2787119022, 0.4905899154,
          0.0071373567, 0.1911010627],
         [0.0636938415, 0.0118942743, 0.2402709234, 0.2953818845,
          0.008569464 , 0.3801896122]],
  names([Tuft, Mucous_gland_neck_INFLARE, Surface_foveolar, Goblet_progenitor,
BEST4_enterocyte_colonocyte, Enterocyte]))
```

[74]:
```
#adata.obsm['term_states_fwd_memberships'].lineages
```

[75]:
```
adata.obsm['macrostates_fwd_memberships']
```

[75]:
```
Lineage([[0.0081068575, 0.032987137 , 0.0072092288, 0.3424707551,
          0.359191934 , 0.2500340876],
         [0.0143850249, 0.0498111056, 0.0085562264, 0.3444279032,
          0.3100691725, 0.2727505673],
         [0.0055603371, 0.0337544252, 0.0036994481, 0.0796737439,
          0.1128049037, 0.7645071419],
         …,
         [0.0032945462, 0.0032452972, 0.003740275 , 0.3161376231,
          0.6641767585, 0.0094055   ],
         [0.0089109814, 0.0235487815, 0.0071373567, 0.2787119022,
          0.4905899154, 0.1911010627],
         [0.0118942743, 0.0636938415, 0.008569464 , 0.2402709234,
          0.2953818845, 0.3801896122]],
  names([Mucous_gland_neck_INFLARE, Tuft, BEST4_enterocyte_colonocyte,
Surface_foveolar, Goblet_progenitor, Enterocyte]))
```

[76]:
```
adata.obsm['macrostates_fwd_memberships'].names
```

[76]:
```
array(['Mucous_gland_neck_INFLARE', 'Tuft', 'BEST4_enterocyte_colonocyte',
       'Surface_foveolar', 'Goblet_progenitor', 'Enterocyte'],
      dtype='<U27')
```

[77]:
```
df = pd.DataFrame(adata.obsm['term_states_fwd_memberships'])
df.columns = adata.obsm['term_states_fwd_memberships'].names
df['Mucous_gland_neck_INFLARE']
```

[77]:
```
0        0.008107
1        0.014385
2        0.005560
3        0.005914
4        0.005817
           …
20760    0.009803
20761    0.002423
20762    0.003295
20763    0.008911
20764    0.011894
Name: Mucous_gland_neck_INFLARE, Length: 20765, dtype: float64
```

[78]:
```
adata.obs['Mucous_gland_neck_INFLARE_lineage_prob'] =␣
 ↪list(df['Mucous_gland_neck_INFLARE'])
```

[79]:
```
adata.obs['Enterocyte_prob'] = list(df['Enterocyte'])
```

[80]:
```
adata.obs['Enterocyte_prob']
```

[80]:
```
index
ACTGCTCAGTCCAGGA-HCA_A_GT12934998-0    0.250034
CGAGCCACAGTCTTCC-HCA_A_GT12934998-0    0.272751
AACACGTCAAGTCTAC-Pan_T7917827-0        0.764507
AACTCAGTCTGTGCAA-Pan_T7917827-0        0.731658
AACTCTTCAGGAATCG-Pan_T7917827-0        0.703849
                                          …
GCTGGGTTCAAAGTAG-GSM3972030-1          0.555725
GTATCTTAGCCAGGAT-GSM3972030-1          0.848658
GTCGTAAAGACTCGGA-GSM3972030-1          0.009406
GTGGGTCGTCGGATCC-GSM3972030-1          0.191101
TTCTACATCGTACGGC-GSM3972030-1          0.380190
Name: Enterocyte_prob, Length: 20765, dtype: float64
```

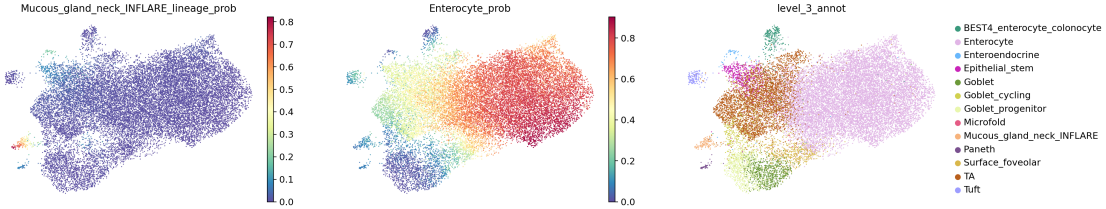
[ ]:
```

```

[ ]:
```

```

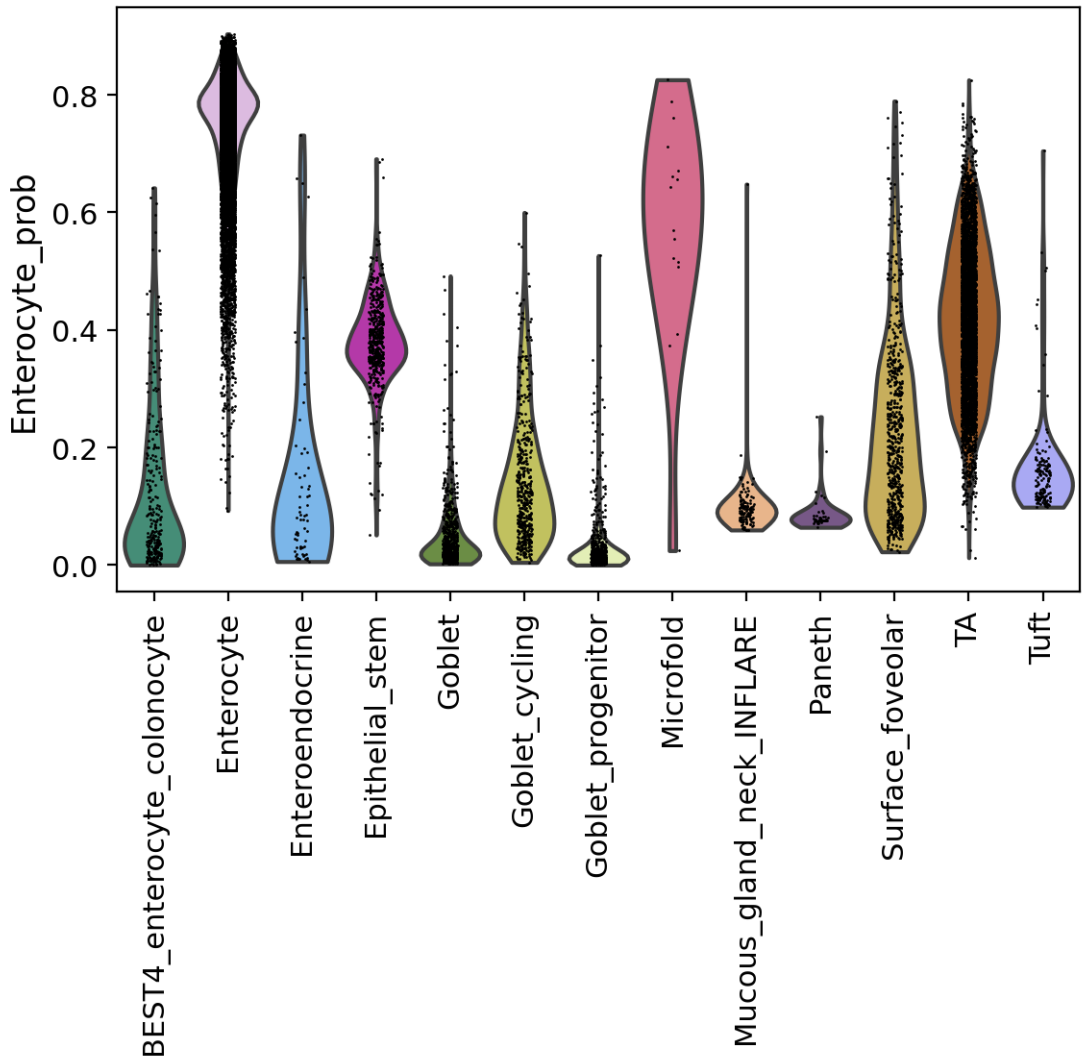
[81]:
```
#adata.obs['Mucous_gland_neck_INFLARE_lineage_prob']
sc.pl.embedding(adata, basis='umap_scvi',␣
 ↪color=['Mucous_gland_neck_INFLARE_lineage_prob','Enterocyte_prob','level_3_annot'],␣
 ↪cmap='Spectral_r')
```

```
/opt/conda/envs/pygpcca_env/lib/python3.8/site-
```

```
packages/scanpy/plotting/_tools/scatterplots.py:163:
MatplotlibDeprecationWarning: The get_cmap function was deprecated in Matplotlib
3.7 and will be removed two minor releases later. Use
``matplotlib.colormaps[name]`` or ``matplotlib.colormaps.get_cmap(obj)``
instead.
  cmap = copy(get_cmap(cmap))
```

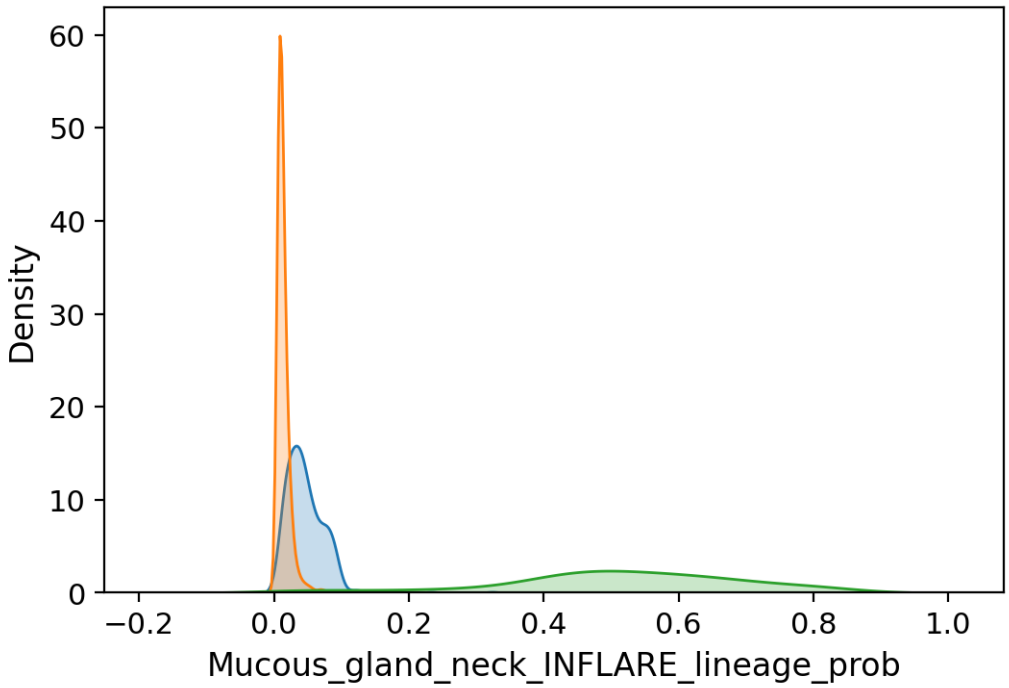


[82]:
```
sc.pl.violin(adata, keys=["Enterocyte_prob"], groupby="clusters", rotation=90)
```

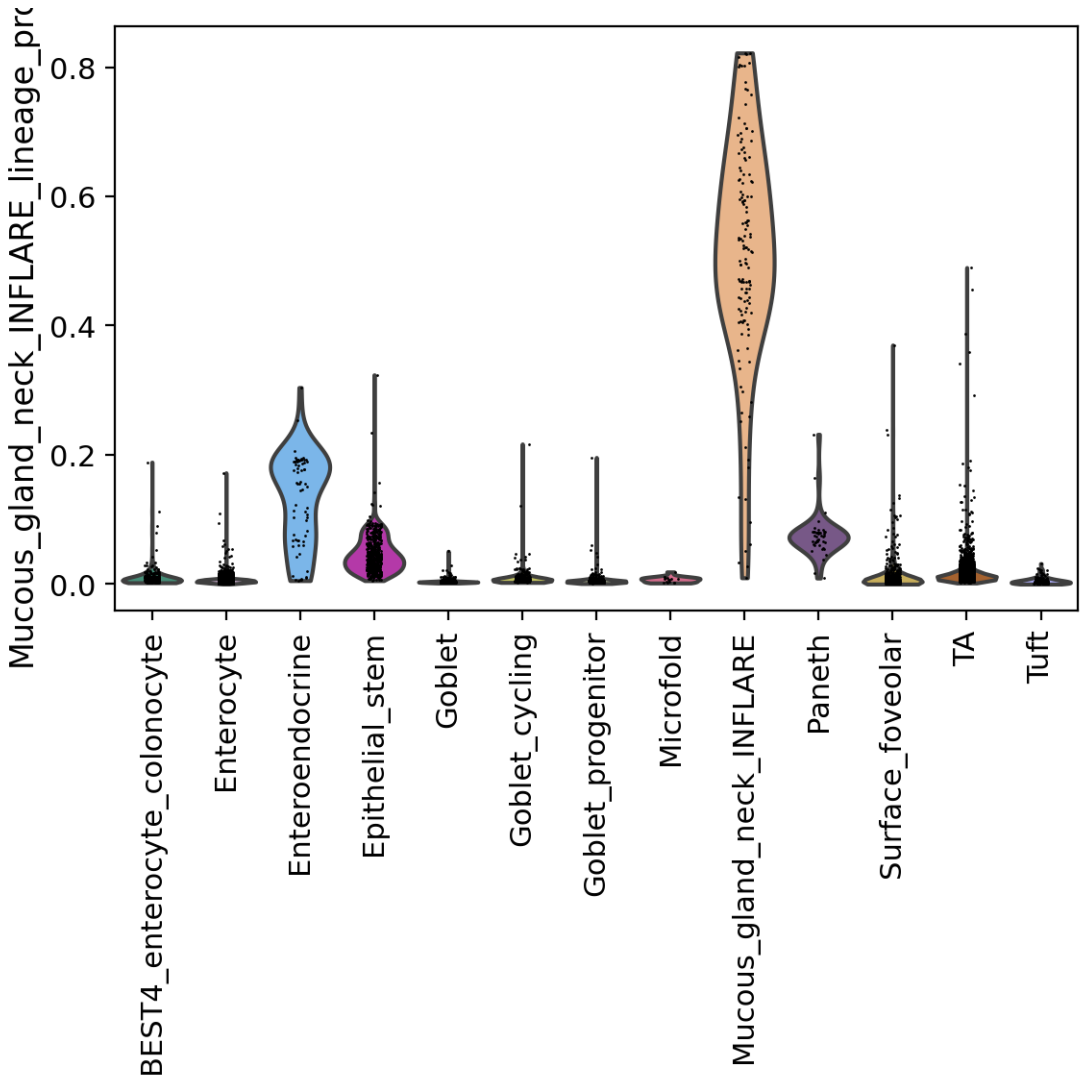


[83]:
```
import seaborn as sb
x = adata[adata.obs.level_3_annot == 'Epithelial_stem'].
  ↪obs['Mucous_gland_neck_INFLARE_lineage_prob']
y= adata[adata.obs.level_3_annot == 'TA'].
  ↪obs['Mucous_gland_neck_INFLARE_lineage_prob']
z = adata[adata.obs.level_3_annot == 'Mucous_gland_neck_INFLARE'].
  ↪obs['Mucous_gland_neck_INFLARE_lineage_prob']
sb.kdeplot(x, fill=True)
sb.kdeplot(y, fill=True)
sb.kdeplot(z, fill=True)
```

[83]: <Axes: xlabel='Mucous_gland_neck_INFLARE_lineage_prob', ylabel='Density'>



[84]:
```
sc.pl.violin(adata, keys=["Mucous_gland_neck_INFLARE_lineage_prob"],␣
↪groupby="clusters", rotation=90)
```



[85]:
```
#neg_probs = adata[~np.in1d(adata.obs.level_3_annot,␣
 ↪['TA','Mucous_gland_neck_INFLARE', 'Epithelial_stem' ])].
 ↪obs['Mucous_gland_neck_INFLARE_lineage_prob']
#mean_neg_prob= np.median(neg_probs)
```

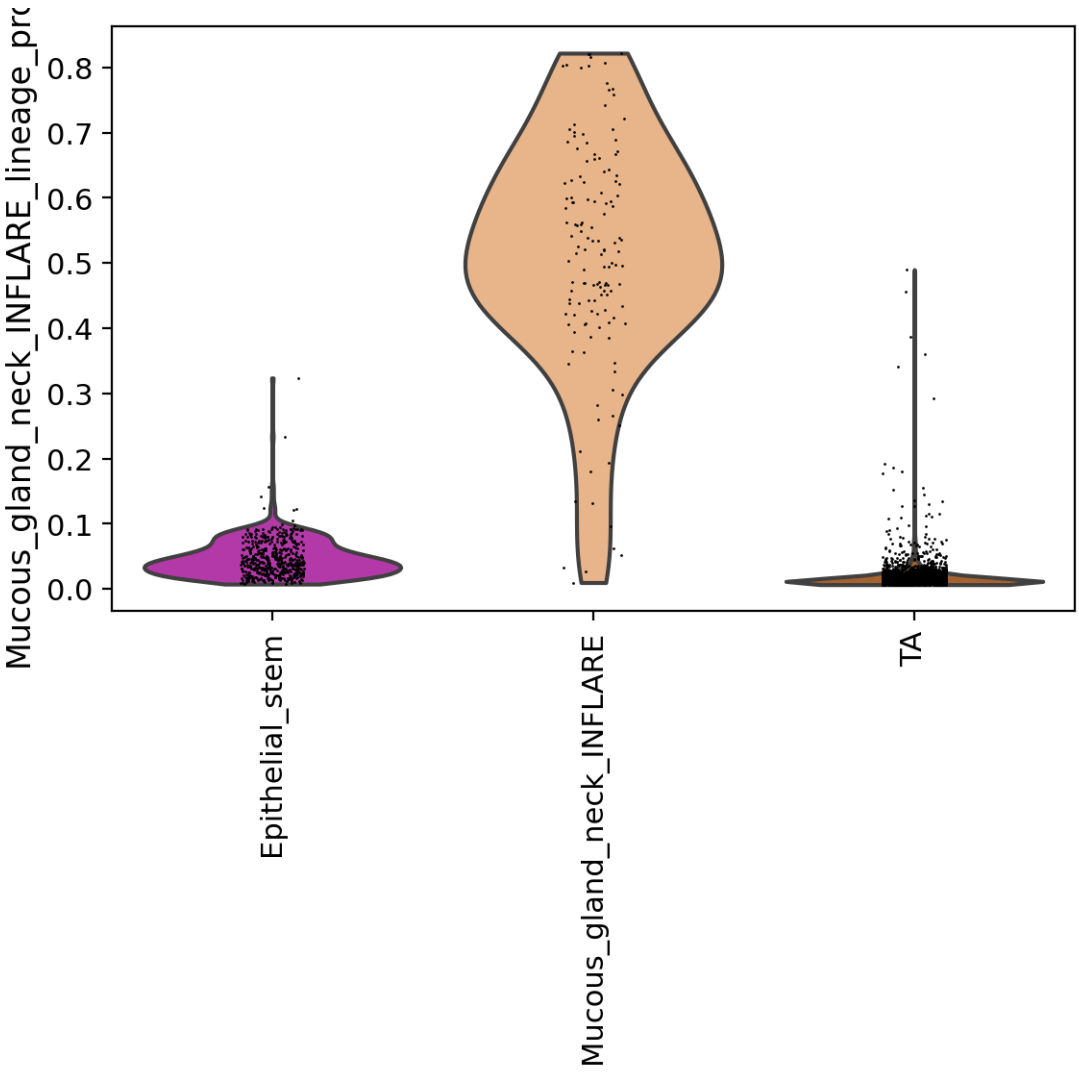
[92]:
```
neg_probs = adata[~np.in1d(adata.obs.level_3_annot,␣
 ↪['TA','Mucous_gland_neck_INFLARE', 'Epithelial_stem' ])].
 ↪obs['Mucous_gland_neck_INFLARE_lineage_prob']
neg_prob_75 = np.percentile(neg_probs, 75)
```

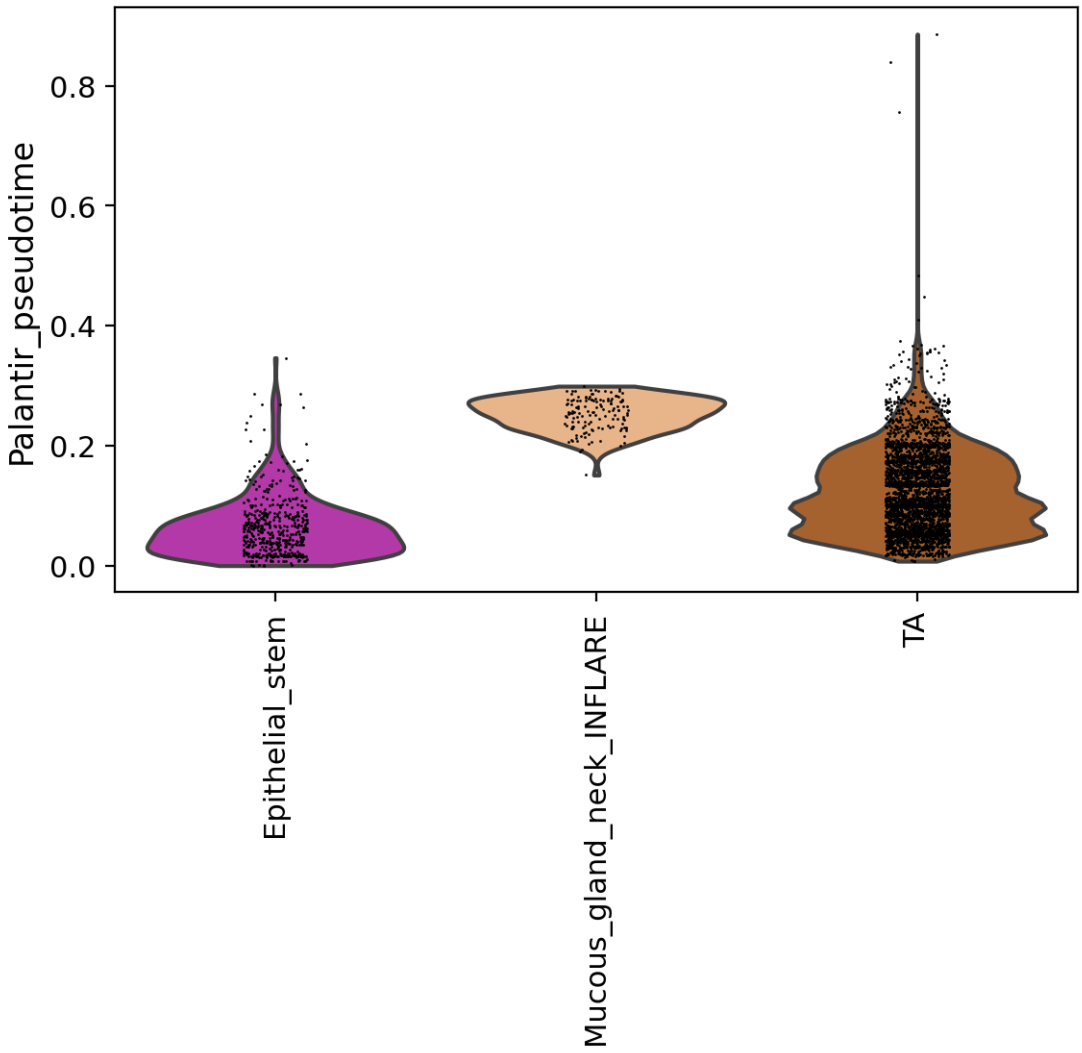
[93]:
```
x = adata[adata.obs.level_3_annot == 'Epithelial_stem'].
 ↪obs['Mucous_gland_neck_INFLARE_lineage_prob']
y= adata[adata.obs.level_3_annot == 'TA'].
 ↪obs['Mucous_gland_neck_INFLARE_lineage_prob']
```

```
z = adata[adata.obs.level_3_annot == 'Mucous_gland_neck_INFLARE'].
 ↪obs['Mucous_gland_neck_INFLARE_lineage_prob']
```

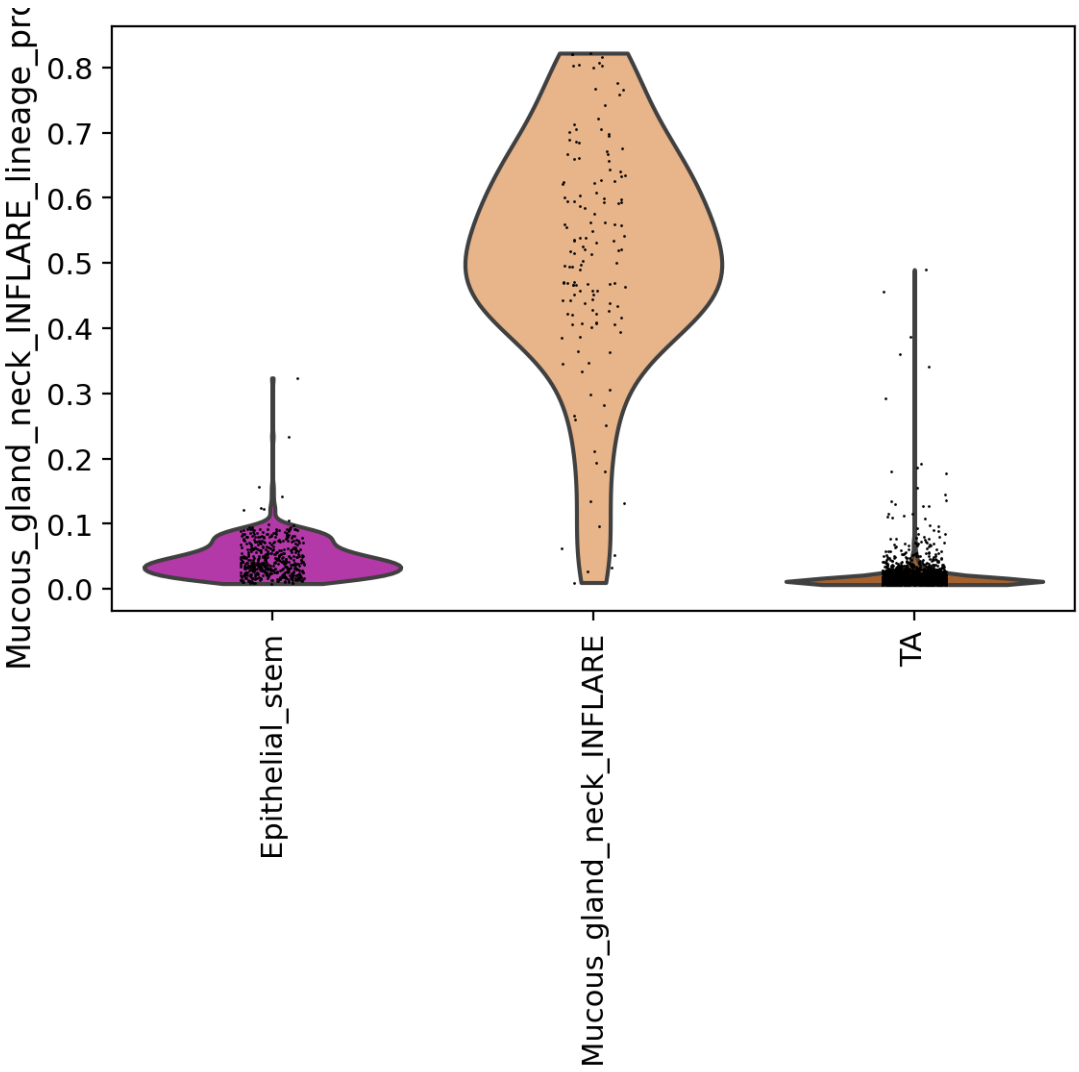
[111]:
```
inflare_cells = adata[adata.obs.level_3_annot == 'Mucous_gland_neck_INFLARE']
TA_cells = adata[adata.obs.level_3_annot == 'TA']
TA_cells = TA_cells[TA_cells.obs['Mucous_gland_neck_INFLARE_lineage_prob']  >␣
 ↪neg_prob_75]
#TA_cells = TA_cells[TA_cells.obs['Palantir_pseudotime']< np.mean(inflare_cells.
 ↪obs.Palantir_pseudotime)] # because there seems to be a considerable number␣
 ↪of TA cells having pseudotime more than INFLARE cells
epi_cells = adata[adata.obs.level_3_annot == 'Epithelial_stem']
epi_cells = epi_cells[epi_cells.obs['Mucous_gland_neck_INFLARE_lineage_prob'] ␣
 ↪> neg_prob_75]
#epi_cells = epi_cells[epi_cells.obs['Palantir_pseudotime']< np.
 ↪mean(inflare_cells.obs.Palantir_pseudotime)]

lof_adata = adata[list(TA_cells.obs_names) + list(epi_cells.obs_names) +␣
 ↪list(inflare_cells.obs_names)]
```

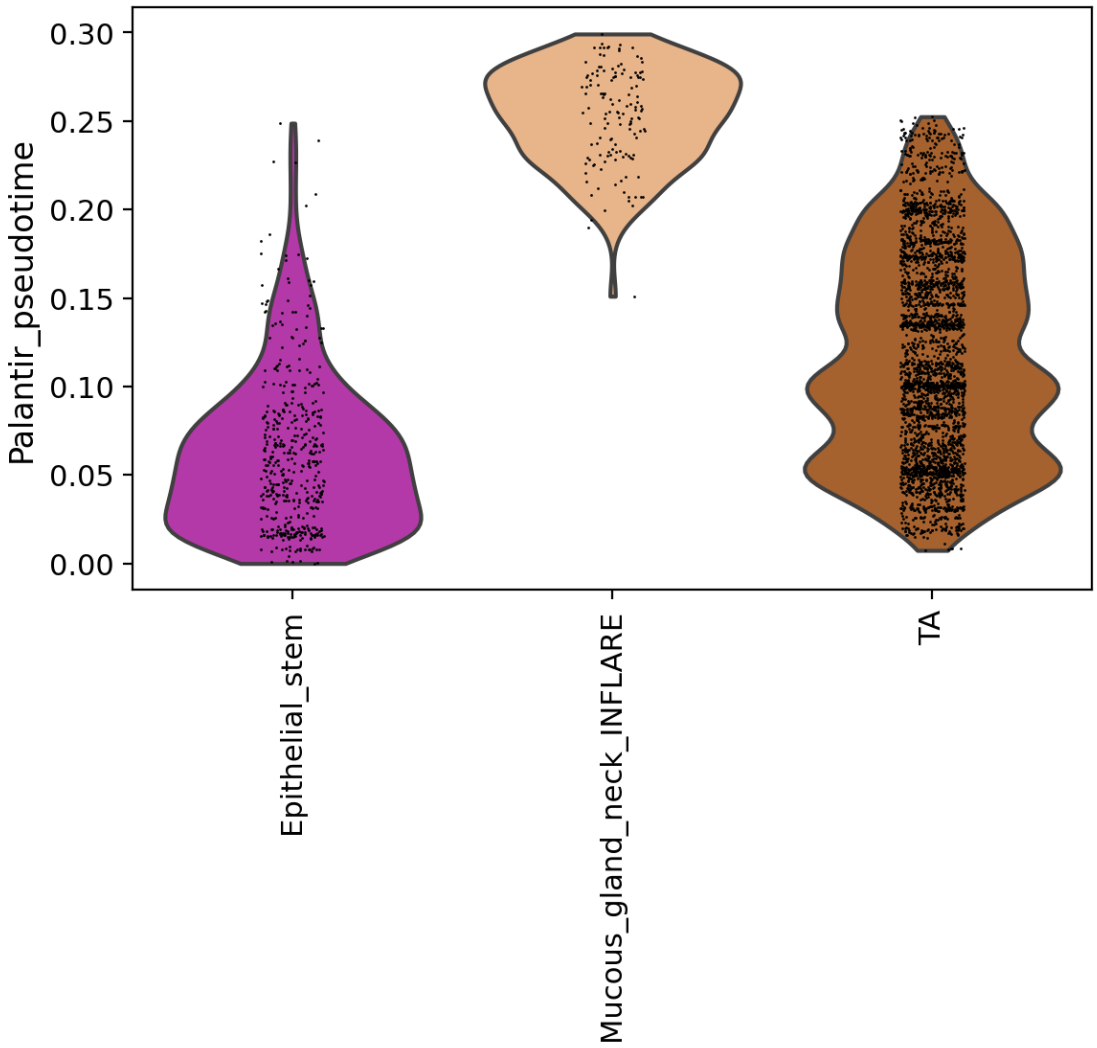
[112]:
```
sc.pl.violin(lof_adata, keys=["Mucous_gland_neck_INFLARE_lineage_prob"],␣
 ↪groupby="clusters", rotation=90)
sc.pl.violin(lof_adata, keys=["Palantir_pseudotime"], groupby="clusters",␣
 ↪rotation=90)
```


Mucous_gland_neck_INFLARE_lineage_pro
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
Epithelial_stem
Mucous_gland_neck_INFLARE
TA



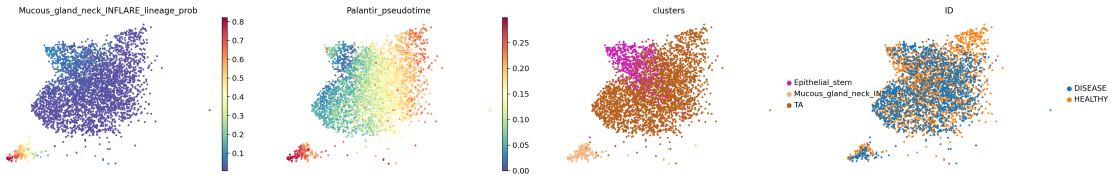
[98]:
```
sc.pl.violin(lof_adata, keys=["Mucous_gland_neck_INFLARE_lineage_prob"],␣
↪groupby="clusters", rotation=90)
sc.pl.violin(lof_adata, keys=["Palantir_pseudotime"], groupby="clusters",␣
↪rotation=90)
```


Mucous_gland_neck_INFLARE_lineage_pro
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
Epithelial_stem
Mucous_gland_neck_INFLARE
TA



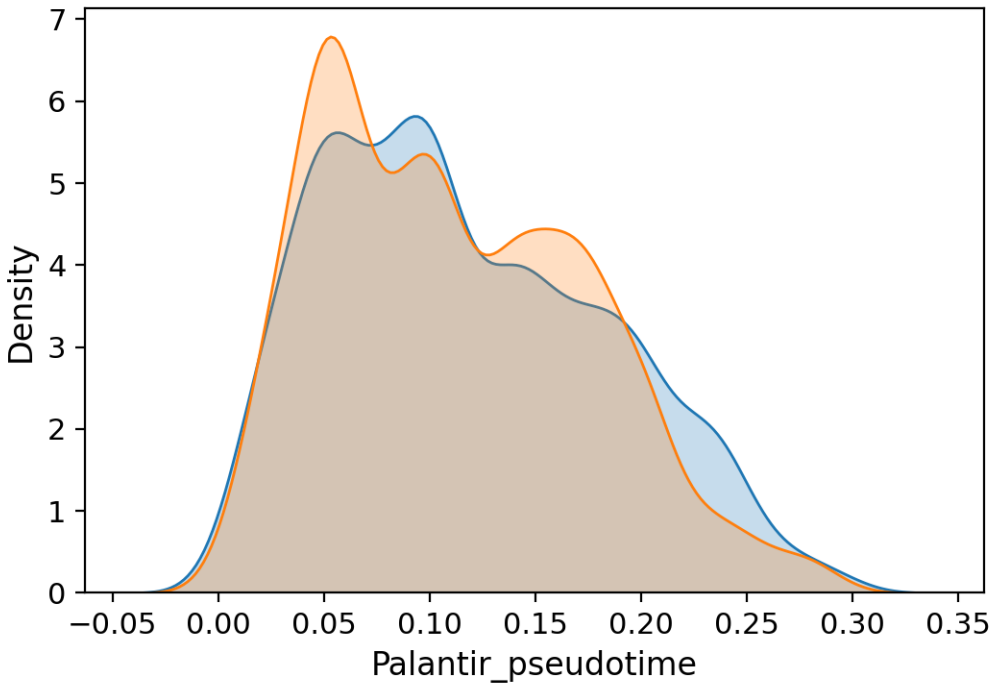
[99]:
```
sc.pl.embedding(lof_adata, basis='umap_scvi',␣
 ↪color=['Mucous_gland_neck_INFLARE_lineage_prob','Palantir_pseudotime','clusters','ID'],␣
 ↪cmap='Spectral_r')
sb.kdeplot(lof_adata[lof_adata.obs.ID=='HEALTHY'].obs['Palantir_pseudotime'],␣
 ↪fill=True)
sb.kdeplot(lof_adata[lof_adata.obs.ID=='DISEASE'].obs['Palantir_pseudotime'],␣
 ↪fill =True)
```

```
/opt/conda/envs/pygpcca_env/lib/python3.8/site-
packages/scanpy/plotting/_tools/scatterplots.py:163:
MatplotlibDeprecationWarning: The get_cmap function was deprecated in Matplotlib
3.7 and will be removed two minor releases later. Use
``matplotlib.colormaps[name]`` or ``matplotlib.colormaps.get_cmap(obj)``
instead.
  cmap = copy(get_cmap(cmap))
```



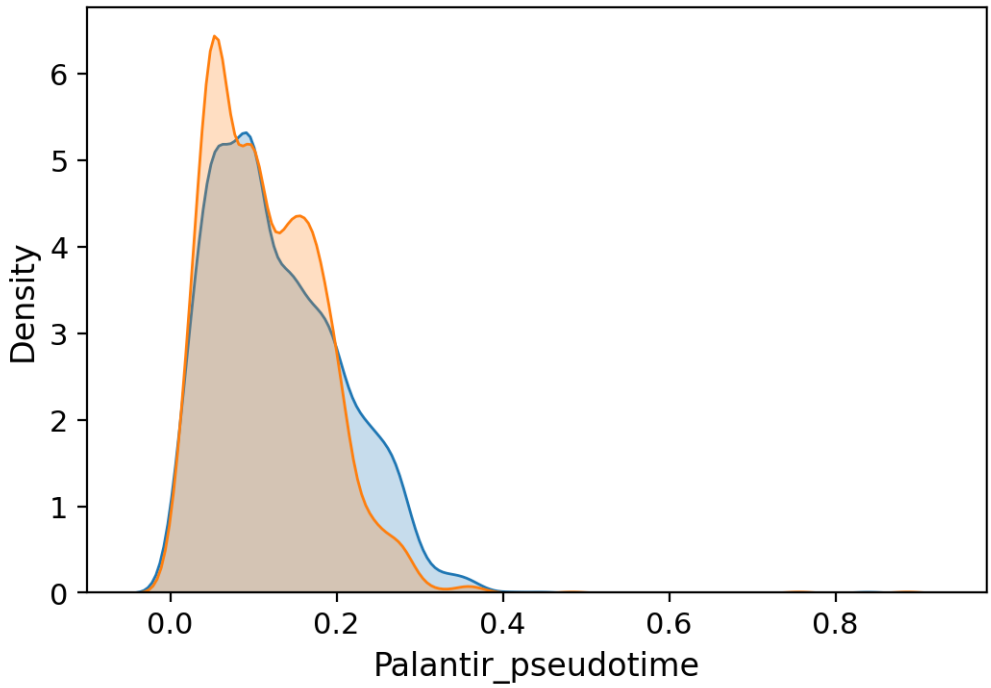
[100]:
```
sb.kdeplot(lof_adata[lof_adata.obs.ID=='HEALTHY'].obs['Palantir_pseudotime'],␣
↪fill=True)
sb.kdeplot(lof_adata[lof_adata.obs.ID=='DISEASE'].obs['Palantir_pseudotime'],␣
↪fill =True)
```

[100]: <Axes: xlabel='Palantir_pseudotime', ylabel='Density'>



[113]:
```
sb.kdeplot(lof_adata[lof_adata.obs.ID=='HEALTHY'].obs['Palantir_pseudotime'],␣
↪fill=True)
sb.kdeplot(lof_adata[lof_adata.obs.ID=='DISEASE'].obs['Palantir_pseudotime'],␣
↪fill =True)
```

[113]: <Axes: xlabel='Palantir_pseudotime', ylabel='Density'>



[127]:
```
inflare_cells = adata[adata.obs.level_3_annot == 'Mucous_gland_neck_INFLARE']
TA_cells = adata[adata.obs.level_3_annot == 'TA']
TA_cells = TA_cells[TA_cells.obs['Mucous_gland_neck_INFLARE_lineage_prob']  >␣
 ↪neg_prob_75]
TA_cells = TA_cells[TA_cells.obs['Palantir_pseudotime']< np.mean(inflare_cells.
 ↪obs.Palantir_pseudotime)] # because there seems to be a considerable number␣
 ↪of TA cells having pseudotime more than INFLARE cells
epi_cells = adata[adata.obs.level_3_annot == 'Epithelial_stem']
epi_cells = epi_cells[epi_cells.obs['Mucous_gland_neck_INFLARE_lineage_prob'] ␣
 ↪> neg_prob_75]
epi_cells = epi_cells[epi_cells.obs['Palantir_pseudotime']< np.
 ↪mean(inflare_cells.obs.Palantir_pseudotime)]

lof_adata = adata[list(TA_cells.obs_names) + list(epi_cells.obs_names) +␣
 ↪list(inflare_cells.obs_names)]
lof_adata
```

[127]:
```
View of AnnData object with n_obs × n_vars = 3832 × 10845
    obs: 'latent_cell_probability', 'latent_RT_efficiency', 'cecilia22_predH',
'cecilia22_predH_prob', 'cecilia22_predH_uncertain', 'cecilia22_predL',
'cecilia22_predL_prob', 'cecilia22_predL_uncertain', 'elmentaite21_pred',
'elmentaite21_pred_prob', 'elmentaite21_pred_uncertain', 'suo22_pred',
'suo22_pred_prob', 'suo22_pred_uncertain', 'n_counts', 'log1p_n_counts',
'n_genes', 'log1p_n_genes', 'percent_mito', 'n_counts_mito', 'percent_ribo',
```

```
'n_counts_ribo', 'percent_hb', 'n_counts_hb', 'percent_top50', 'n_counts_raw',
'log1p_n_counts_raw', 'n_genes_raw', 'log1p_n_genes_raw', 'percent_mito_raw',
'n_counts_mito_raw', 'percent_ribo_raw', 'n_counts_ribo_raw', 'percent_hb_raw',
'n_counts_hb_raw', 'percent_top50_raw', 'n_counts_spliced',
'log1p_n_counts_spliced', 'n_genes_spliced', 'log1p_n_genes_spliced',
'percent_mito_spliced', 'n_counts_mito_spliced', 'percent_ribo_spliced',
'n_counts_ribo_spliced', 'percent_hb_spliced', 'n_counts_hb_spliced',
'percent_top50_spliced', 'n_counts_unspliced', 'log1p_n_counts_unspliced',
'n_genes_unspliced', 'log1p_n_genes_unspliced', 'percent_mito_unspliced',
'n_counts_mito_unspliced', 'percent_ribo_unspliced', 'n_counts_ribo_unspliced',
'percent_hb_unspliced', 'n_counts_hb_unspliced', 'percent_top50_unspliced',
'percent_soup', 'percent_spliced', 'qc_cluster', 'pass_auto_filter_mito20',
'good_qc_cluster_mito20', 'pass_auto_filter_mito50', 'good_qc_cluster_mito50',
'pass_auto_filter_mito80', 'good_qc_cluster_mito80', 'pass_auto_filter',
'good_qc_cluster', 'pass_default', 'sampleID', 'sourceID', 'donorID_original',
'study', 'donorID_corrected', 'donorID_unified', 'donor_category',
'donor_disease', 'organ_original', 'organ_unified', 'organ_broad',
'age_original', 'age_unified', 'age_continuousadult', 'age_continuousdev',
'sex', 'sample_type', 'sample_category', 'sample_retrieval', 'tissue_fraction',
'cell_fraction', 'cell_fraction_unified', 'cell_sorting', 'technology',
'include_150722', 'cluster_scrublet_score', 'bh_pval', 'scrublet_score',
'scrublet_score_z', 'doublet', 'stringent_doublet', 'integration_grouping',
'_scvi_batch', '_scvi_labels', 'broad_annot_20220914', 'martin19_pred',
'martin19_pred_prob', 'martin19_pred_uncertain', 'warner20_pred',
'warner20_pred_prob', 'warner20_pred_uncertain', 'broad_annot_20220917',
'donor_organ_lineage', 'fine_annot', 'fine_annot_original', 'level_1_annot',
'level_2_annot', 'level_3_annot', 'broad_predicted_labels',
'broad_predicted_labels_uncert', 'batch', 'organ_groups', 'control_vs_disease',
'disease', 'ID', 'clusters', 'Palantir_pseudotime', 'initial_size_unspliced',
'initial_size_spliced', 'initial_size', 'velocity_self_transition',
'root_cells', 'end_points', 'velocity_pseudotime', 'start_score',
'dpt_pseudotime', 'ct_score', 'ct_pseudotime', 'ct_num_exp_genes',
'avg_pseudotime', 'macrostates_fwd', 'term_states_fwd', 'term_states_fwd_probs',
'init_states_fwd', 'init_states_fwd_probs', 'clusters_gradients',
'Mucous_gland_neck_INFLARE_lineage_prob', 'Enterocyte_prob'
   var: 'gene_ids', 'feature_type', 'mito', 'ribo', 'hb', 'cc', 'ig', 'tcr',
'n_counts-0', 'n_counts_raw-0', 'n_counts_spliced-0', 'n_counts_unspliced-0',
'n_cells-0', 'n_cells_raw-0', 'n_cells_spliced-0', 'n_cells_unspliced-0',
'n_counts-1', 'n_counts_raw-1', 'n_counts_spliced-1', 'n_counts_unspliced-1',
'n_cells-1', 'n_cells_raw-1', 'n_cells_spliced-1', 'n_cells_unspliced-1',
'gene_count_corr', 'means', 'dispersions', 'dispersions_norm',
'highly_variable', 'fit_r2', 'fit_alpha', 'fit_beta', 'fit_gamma', 'fit_t_',
'fit_scaling', 'fit_std_u', 'fit_std_s', 'fit_likelihood', 'fit_u0', 'fit_s0',
'fit_pval_steady', 'fit_steady_u', 'fit_steady_s', 'fit_variance',
'fit_alignment_scaling', 'velocity_genes', 'ct_gene_corr', 'ct_correlates'
   uns: 'disease_colors', 'level_3_annot_colors', 'organ_unified_colors',
'log1p', 'pca', 'neighbors', 'recover_dynamics', 'velocity_params',
```

```
'T_fwd_params', 'clusters_colors', 'velocity_graph', 'velocity_graph_neg',
'diffmap_evals', 'iroot', 'ct_params', 'schur_matrix_fwd',
'eigendecomposition_fwd', 'macrostates_fwd_colors', 'coarse_fwd',
'term_states_fwd_colors', 'init_states_fwd_colors', 'clusters_gradients_colors'
    obsm: 'X_mde', 'X_scANVI', 'X_scVI', 'X_umap', 'X_umap_scvi',
'_scvi_extra_continuous_covs', 'X_pca', 'T_fwd_umap_scvi', 'T_fwd_pca',
'velocity_umap_scvi', 'X_diffmap', 'schur_vectors_fwd',
'macrostates_fwd_memberships', 'term_states_fwd_memberships',
'init_states_fwd_memberships', 'lineages_fwd'
    varm: 'PCs', 'loss'
    layers: 'counts', 'spliced', 'unspliced', 'Ms', 'Mu', 'fit_t', 'fit_tau',
'fit_tau_', 'velocity', 'velocity_u'
    obsp: 'distances', 'connectivities'
```

[ ]:

[101]:
```
max(lof_adata.obs['Palantir_pseudotime'])
```

[101]:
```
0.2991462749150012
```

[106]:
```
lof_adata.obs.to_csv('v2_metadata_sub_inflare_lineage_for_g2g_RQ1.csv')
```

[109]:
```
lof_adata
```

[109]:
```
AnnData object with n_obs × n_vars = 3832 × 10845
    var: 'gene_ids', 'feature_type', 'mito', 'ribo', 'hb', 'cc', 'ig', 'tcr',
'n_counts-0', 'n_counts_raw-0', 'n_counts_spliced-0', 'n_counts_unspliced-0',
'n_cells-0', 'n_cells_raw-0', 'n_cells_spliced-0', 'n_cells_unspliced-0',
'n_counts-1', 'n_counts_raw-1', 'n_counts_spliced-1', 'n_counts_unspliced-1',
'n_cells-1', 'n_cells_raw-1', 'n_cells_spliced-1', 'n_cells_unspliced-1',
'gene_count_corr', 'means', 'dispersions', 'dispersions_norm',
'highly_variable', 'fit_r2', 'fit_alpha', 'fit_beta', 'fit_gamma', 'fit_t_',
'fit_scaling', 'fit_std_u', 'fit_std_s', 'fit_likelihood', 'fit_u0', 'fit_s0',
'fit_pval_steady', 'fit_steady_u', 'fit_steady_s', 'fit_variance',
'fit_alignment_scaling', 'velocity_genes', 'ct_gene_corr', 'ct_correlates'
    uns: 'disease_colors', 'level_3_annot_colors', 'organ_unified_colors',
'log1p', 'pca', 'neighbors', 'recover_dynamics', 'velocity_params',
'T_fwd_params', 'clusters_colors', 'velocity_graph', 'velocity_graph_neg',
'diffmap_evals', 'iroot', 'ct_params', 'schur_matrix_fwd',
'eigendecomposition_fwd', 'macrostates_fwd_colors', 'coarse_fwd',
'term_states_fwd_colors', 'init_states_fwd_colors', 'clusters_gradients_colors',
'ID_colors'
    obsm: 'X_mde', 'X_scANVI', 'X_scVI', 'X_umap', 'X_umap_scvi',
'_scvi_extra_continuous_covs', 'X_pca', 'T_fwd_umap_scvi', 'T_fwd_pca',
'velocity_umap_scvi', 'X_diffmap', 'schur_vectors_fwd',
'macrostates_fwd_memberships', 'term_states_fwd_memberships',
'init_states_fwd_memberships', 'lineages_fwd'
```

```
    varm: 'PCs', 'loss'
    layers: 'counts', 'spliced', 'unspliced', 'Ms', 'Mu', 'fit_t', 'fit_tau',
'fit_tau_', 'velocity', 'velocity_u'
    obsp: 'distances', 'connectivities'
```

[107]:
```
del lof_adata.obs
lof_adata.write_h5ad('v2_inflare_lineage_for_g2g_RQ1.h5ad')
```

[110]:
```
lof_adata
```

[110]:
```
AnnData object with n_obs × n_vars = 3832 × 10845
    var: 'gene_ids', 'feature_type', 'mito', 'ribo', 'hb', 'cc', 'ig', 'tcr',
'n_counts-0', 'n_counts_raw-0', 'n_counts_spliced-0', 'n_counts_unspliced-0',
'n_cells-0', 'n_cells_raw-0', 'n_cells_spliced-0', 'n_cells_unspliced-0',
'n_counts-1', 'n_counts_raw-1', 'n_counts_spliced-1', 'n_counts_unspliced-1',
'n_cells-1', 'n_cells_raw-1', 'n_cells_spliced-1', 'n_cells_unspliced-1',
'gene_count_corr', 'means', 'dispersions', 'dispersions_norm',
'highly_variable', 'fit_r2', 'fit_alpha', 'fit_beta', 'fit_gamma', 'fit_t_',
'fit_scaling', 'fit_std_u', 'fit_std_s', 'fit_likelihood', 'fit_u0', 'fit_s0',
'fit_pval_steady', 'fit_steady_u', 'fit_steady_s', 'fit_variance',
'fit_alignment_scaling', 'velocity_genes', 'ct_gene_corr', 'ct_correlates'
    uns: 'disease_colors', 'level_3_annot_colors', 'organ_unified_colors',
'log1p', 'pca', 'neighbors', 'recover_dynamics', 'velocity_params',
'T_fwd_params', 'clusters_colors', 'velocity_graph', 'velocity_graph_neg',
'diffmap_evals', 'iroot', 'ct_params', 'schur_matrix_fwd',
'eigendecomposition_fwd', 'macrostates_fwd_colors', 'coarse_fwd',
'term_states_fwd_colors', 'init_states_fwd_colors', 'clusters_gradients_colors',
'ID_colors'
    obsm: 'X_mde', 'X_scANVI', 'X_scVI', 'X_umap', 'X_umap_scvi',
'_scvi_extra_continuous_covs', 'X_pca', 'T_fwd_umap_scvi', 'T_fwd_pca',
'velocity_umap_scvi', 'X_diffmap', 'schur_vectors_fwd',
'macrostates_fwd_memberships', 'term_states_fwd_memberships',
'init_states_fwd_memberships', 'lineages_fwd'
    varm: 'PCs', 'loss'
    layers: 'counts', 'spliced', 'unspliced', 'Ms', 'Mu', 'fit_t', 'fit_tau',
'fit_tau_', 'velocity', 'velocity_u'
    obsp: 'distances', 'connectivities'
```

[122]:
```
inflare_cells = adata[adata.obs.level_3_annot == 'Mucous_gland_neck_INFLARE']
TA_cells = adata[adata.obs.level_3_annot == 'TA']
#TA_cells = TA_cells[TA_cells.obs['Mucous_gland_neck_INFLARE_lineage_prob']  >␣
 ↪neg_prob_75]
TA_cells = TA_cells[TA_cells.obs['Palantir_pseudotime']< np.mean(inflare_cells.
 ↪obs.Palantir_pseudotime)] # because there seems to be a considerable number␣
 ↪of TA cells having pseudotime more than INFLARE cells
epi_cells = adata[adata.obs.level_3_annot == 'Epithelial_stem']
```

```
#epi_cells = epi_cells[epi_cells.obs['Mucous_gland_neck_INFLARE_lineage_prob'] ␣
 ↪> neg_prob_75]
epi_cells = epi_cells[epi_cells.obs['Palantir_pseudotime']< np.
 ↪mean(inflare_cells.obs.Palantir_pseudotime)]

lof_adata = adata[list(TA_cells.obs_names) + list(epi_cells.obs_names) +␣
 ↪list(inflare_cells.obs_names)]
```

[125]:
```
lof_adata
```

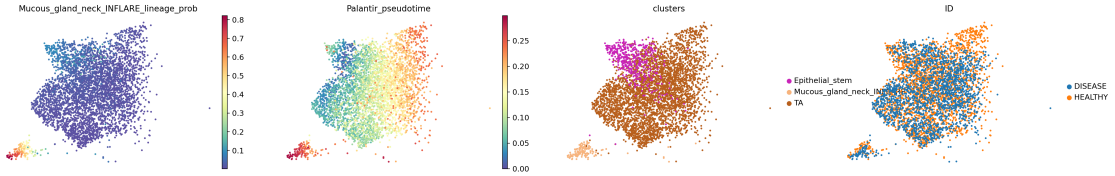
[125]:
```
AnnData object with n_obs × n_vars = 4418 × 10845
    var: 'gene_ids', 'feature_type', 'mito', 'ribo', 'hb', 'cc', 'ig', 'tcr',
'n_counts-0', 'n_counts_raw-0', 'n_counts_spliced-0', 'n_counts_unspliced-0',
'n_cells-0', 'n_cells_raw-0', 'n_cells_spliced-0', 'n_cells_unspliced-0',
'n_counts-1', 'n_counts_raw-1', 'n_counts_spliced-1', 'n_counts_unspliced-1',
'n_cells-1', 'n_cells_raw-1', 'n_cells_spliced-1', 'n_cells_unspliced-1',
'gene_count_corr', 'means', 'dispersions', 'dispersions_norm',
'highly_variable', 'fit_r2', 'fit_alpha', 'fit_beta', 'fit_gamma', 'fit_t_',
'fit_scaling', 'fit_std_u', 'fit_std_s', 'fit_likelihood', 'fit_u0', 'fit_s0',
'fit_pval_steady', 'fit_steady_u', 'fit_steady_s', 'fit_variance',
'fit_alignment_scaling', 'velocity_genes', 'ct_gene_corr', 'ct_correlates'
    uns: 'disease_colors', 'level_3_annot_colors', 'organ_unified_colors',
'log1p', 'pca', 'neighbors', 'recover_dynamics', 'velocity_params',
'T_fwd_params', 'clusters_colors', 'velocity_graph', 'velocity_graph_neg',
'diffmap_evals', 'iroot', 'ct_params', 'schur_matrix_fwd',
'eigendecomposition_fwd', 'macrostates_fwd_colors', 'coarse_fwd',
'term_states_fwd_colors', 'init_states_fwd_colors', 'clusters_gradients_colors'
    obsm: 'X_mde', 'X_scANVI', 'X_scVI', 'X_umap', 'X_umap_scvi',
'_scvi_extra_continuous_covs', 'X_pca', 'T_fwd_umap_scvi', 'T_fwd_pca',
'velocity_umap_scvi', 'X_diffmap', 'schur_vectors_fwd',
'macrostates_fwd_memberships', 'term_states_fwd_memberships',
'init_states_fwd_memberships', 'lineages_fwd'
    varm: 'PCs', 'loss'
    layers: 'counts', 'spliced', 'unspliced', 'Ms', 'Mu', 'fit_t', 'fit_tau',
'fit_tau_', 'velocity', 'velocity_u'
    obsp: 'distances', 'connectivities'
```

[124]:
```
lof_adata.obs.to_csv('v3_metadata_sub_inflare_lineage_for_g2g_RQ1.csv')

del lof_adata.obs
lof_adata.write_h5ad('v3_inflare_lineage_for_g2g_RQ1.h5ad')
```

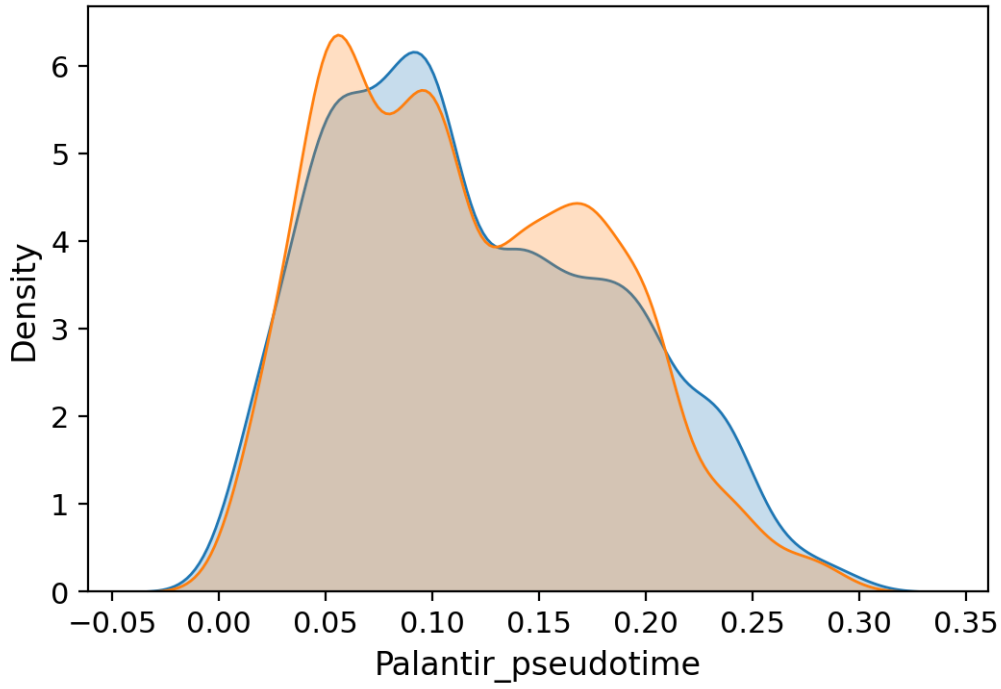
[120]:
```
sc.pl.embedding(lof_adata, basis='umap_scvi',␣
 ↪color=['Mucous_gland_neck_INFLARE_lineage_prob','Palantir_pseudotime','clusters','ID'],␣
 ↪cmap='Spectral_r')
sb.kdeplot(lof_adata[lof_adata.obs.ID=='HEALTHY'].obs['Palantir_pseudotime'],␣
 ↪fill=True)
```

```
sb.kdeplot(lof_adata[lof_adata.obs.ID=='DISEASE'].obs['Palantir_pseudotime'],␣
 ↪fill =True)
```

```
/opt/conda/envs/pygpcca_env/lib/python3.8/site-
packages/scanpy/plotting/_tools/scatterplots.py:163:
MatplotlibDeprecationWarning: The get_cmap function was deprecated in Matplotlib
3.7 and will be removed two minor releases later. Use
``matplotlib.colormaps[name]`` or ``matplotlib.colormaps.get_cmap(obj)``
instead.
  cmap = copy(get_cmap(cmap))
```

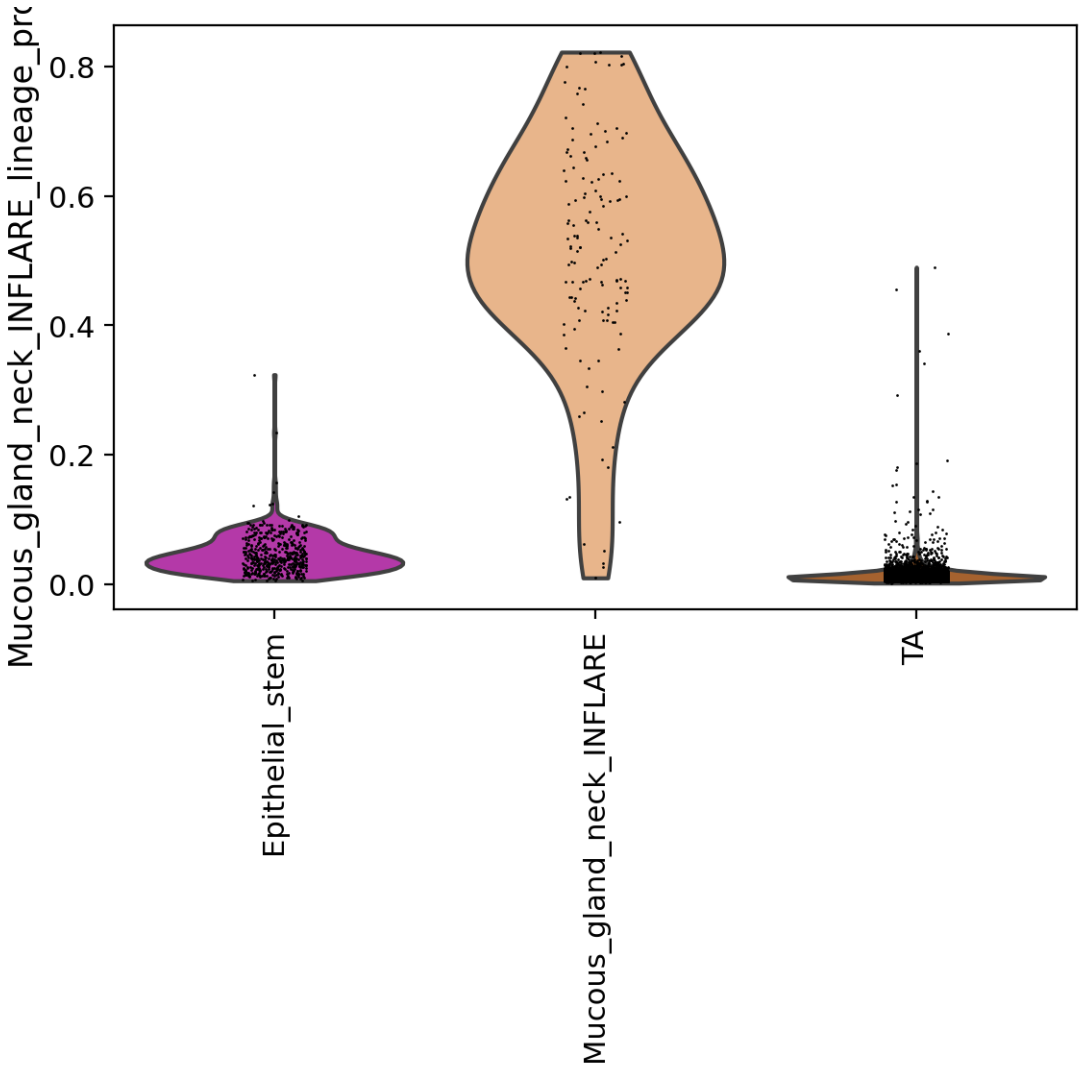


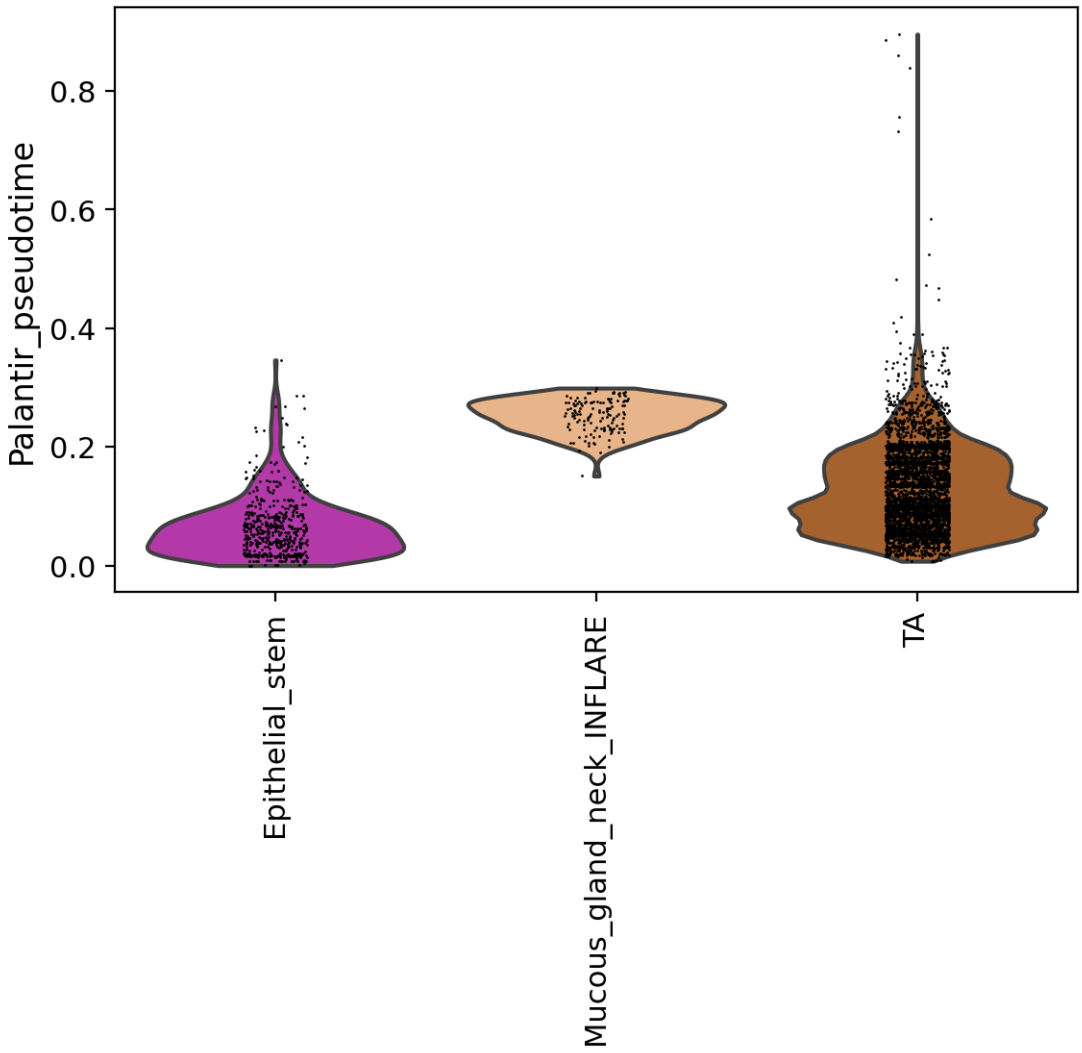
[120]: <Axes: xlabel='Palantir_pseudotime', ylabel='Density'>



[121]:
```
del lof_adata.obs
lof_adata.write_h5ad('v3_inflare_lineage_for_g2g_RQ1.h5ad')
```

[115]:
```
sc.pl.violin(lof_adata, keys=["Mucous_gland_neck_INFLARE_lineage_prob"],␣
↪groupby="clusters", rotation=90)
sc.pl.violin(lof_adata, keys=["Palantir_pseudotime"], groupby="clusters",␣
↪rotation=90)
```





[161]:
```
#scv.tl.recover_latent_time(adata, root_key="initial_states_fwd_probs",␣
↪end_key="terminal_states_fwd_probs")
```

[162]:
```
#combined_kernel.transition_matrix
```

[163]:
```
#adata.obsm['X_scVI'].shape
```

[164]:
```
#g.plot_fate_probabilities(same_plot=False,  basis='umap_scvi')
```

[158]:
```
#sc.pp.neighbors(adata, use_rep="X_scVI")
#adata.uns['neighbors']['distances'] = adata.obsp['distances']
#adata.uns['neighbors']['connectivities'] = adata.obsp['connectivities']
```

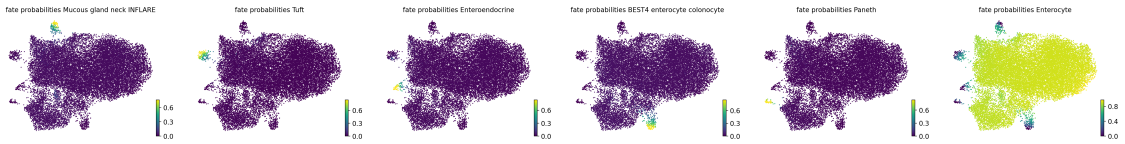
[159]:
```
#scv.tl.recover_latent_time(adata, root_key="initial_states_fwd_probs",␣
 ↪end_key="terminal_states_fwd_probs")
```

[157]:
```
#all_dfs = []
#scv.tl.paga(adata, groups='clusters', vkey='velocity',␣
 ↪use_time_prior="latent_time", root_key=start_cell_id,␣
 ↪end_key="terminal_states_fwd_probs")
#scv.pl.paga(adata, basis='umap_scvi')
#all_dfs.append(scv.get_df(adata, 'paga/transitions_confidence', precision=2).T
```

[160]:
```
#scv.get_df(adata, 'paga/connectivities', precision=2).T
```

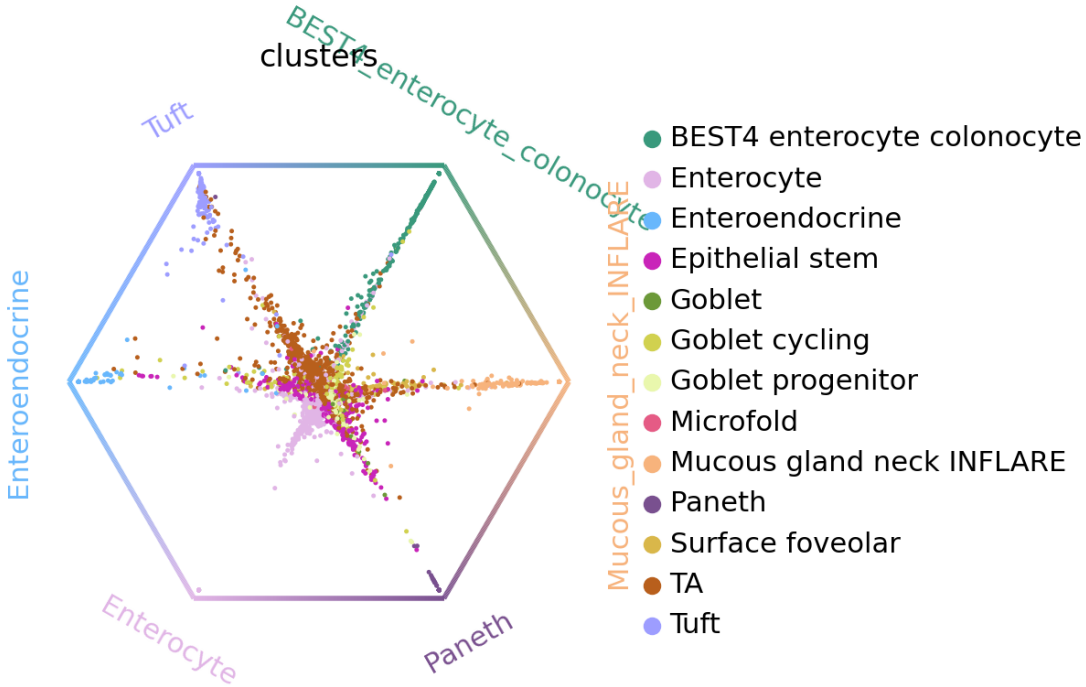
[161]:
```
#df_velo = scv.get_df(adata, 'paga/transitions_confidence', precision=2).T
#df.style.background_gradient(cmap='Blues').format('{:.2g}')
```

[44]:
```
g.plot_fate_probabilities(same_plot=False,  basis='umap_scvi')
```



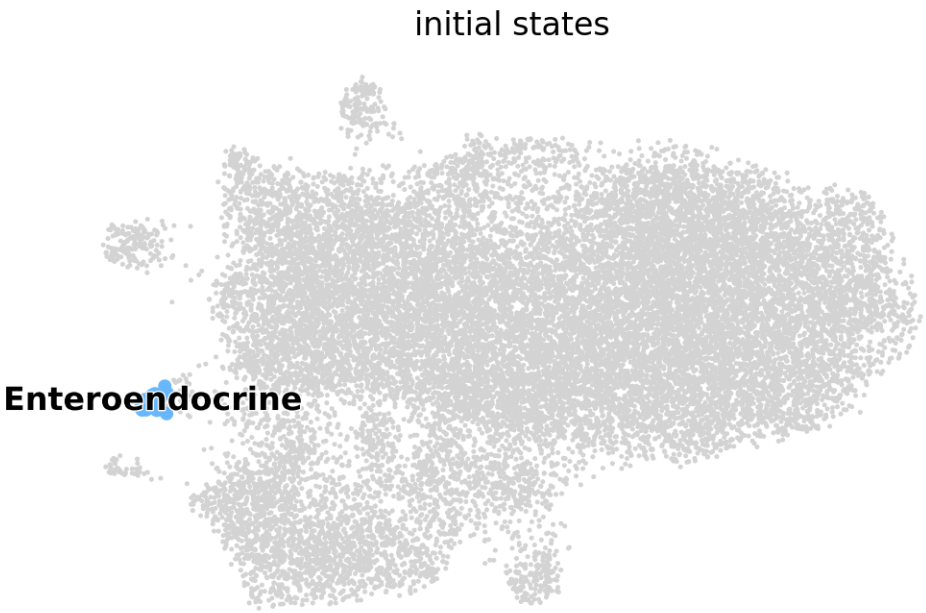
[45]:
```
cr.pl.circular_projection(adata, keys="clusters", legend_loc="right")
```

```
Solving TSP for `6` states
```

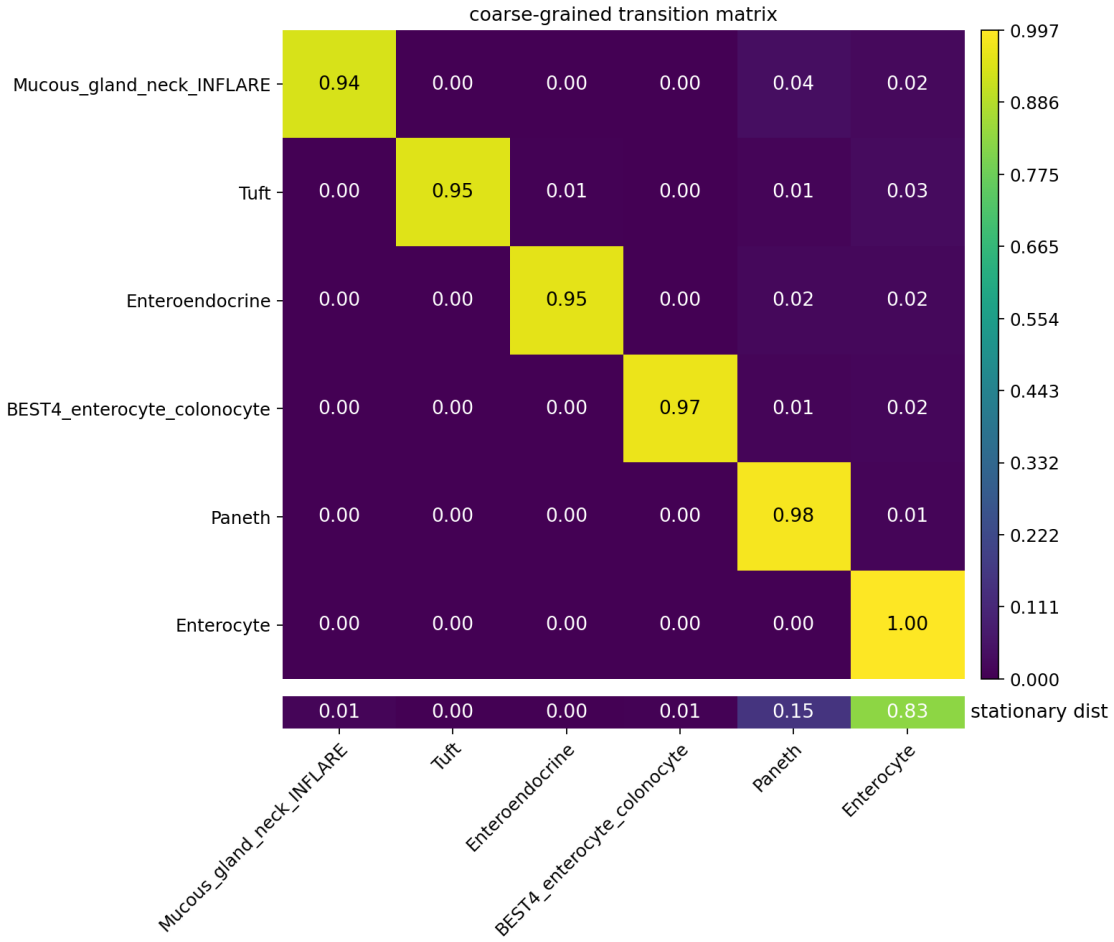


[46]:
```
g.predict_initial_states(allow_overlap=True)
g.plot_macrostates(which="initial", s=100,basis='umap_scvi')
```

```
Adding `adata.obs['init_states_fwd']`
       `adata.obs['init_states_fwd_probs']`
       `.initial_states`
       `.initial_states_probabilities`
       `.initial_states_memberships
    Finish`
```



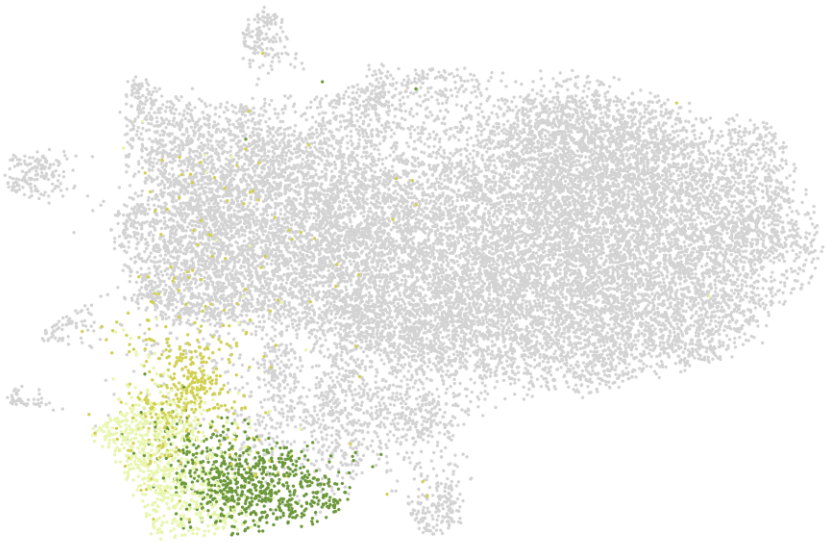
[47]:
```
g.plot_coarse_T()
```



[48]:
```
#g.compute_fate_probabilities()
#g.plot_fate_probabilities(basis = 'umap_scvi', same_plot=False)
```

[49]:
```
fev_states = ['Goblet_progenitor','Goblet_cycling','Goblet']
sc.pl.embedding(
    adata, basis="umap_scvi", color="clusters", groups=fev_states,␣
 ↪legend_loc="right"
)
```

```
/opt/conda/envs/pygpcca_env/lib/python3.8/site-
packages/scanpy/plotting/_tools/scatterplots.py:163:
MatplotlibDeprecationWarning: The get_cmap function was deprecated in Matplotlib
3.7 and will be removed two minor releases later. Use
``matplotlib.colormaps[name]`` or ``matplotlib.colormaps.get_cmap(obj)``
instead.
  cmap = copy(get_cmap(cmap))
```



[50]:
```
np.unique(adata.obs.clusters)
```

[50]:
```
array(['BEST4_enterocyte_colonocyte', 'Enterocyte', 'Enteroendocrine',
       'Epithelial_stem', 'Goblet', 'Goblet_cycling', 'Goblet_progenitor',
       'Microfold', 'Mucous_gland_neck_INFLARE', 'Paneth',
       'Surface_foveolar', 'TA', 'Tuft'], dtype=object)
```

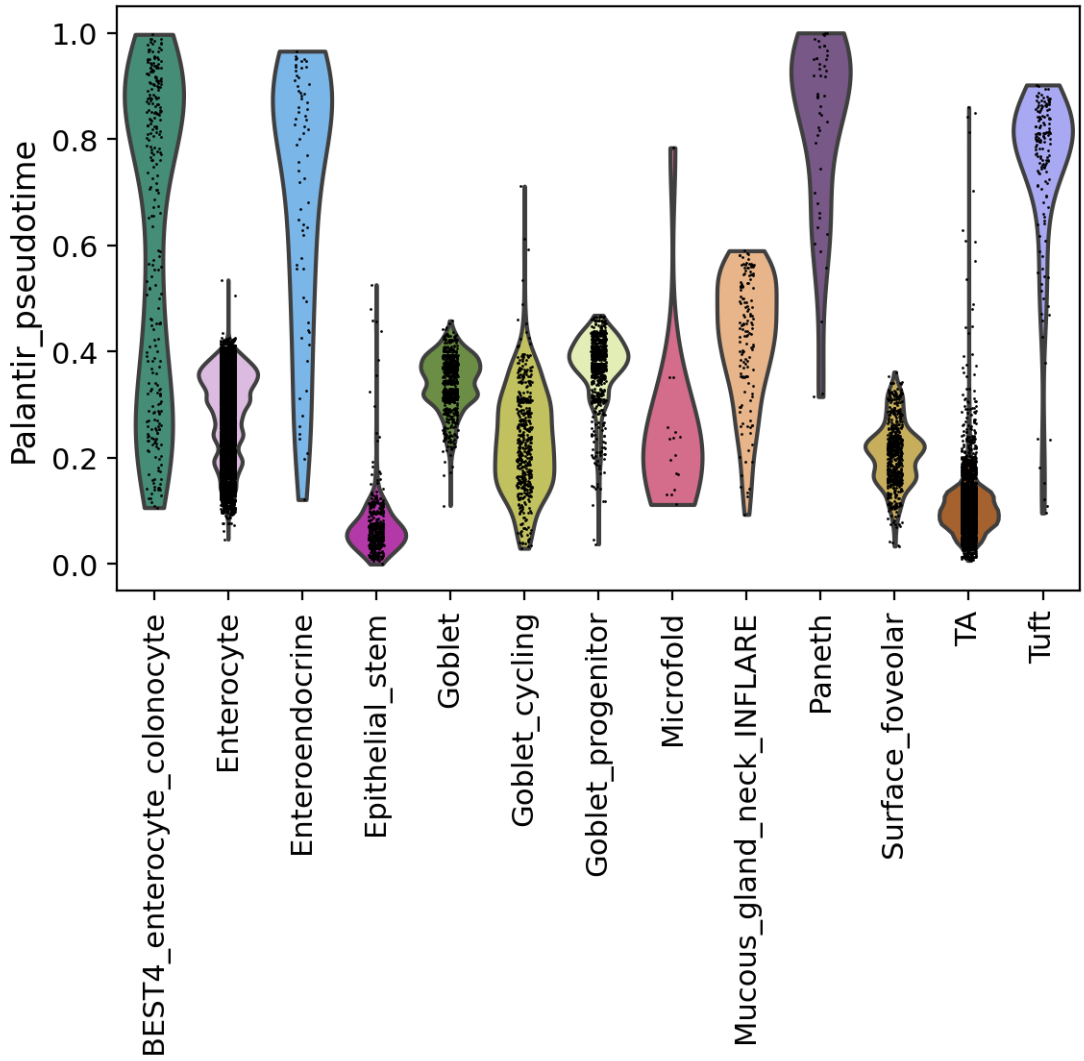
[51]:
```
adata
```

[51]:
```
AnnData object with n_obs × n_vars = 20765 × 10845
    obs: 'latent_cell_probability', 'latent_RT_efficiency', 'cecilia22_predH',
'cecilia22_predH_prob', 'cecilia22_predH_uncertain', 'cecilia22_predL',
'cecilia22_predL_prob', 'cecilia22_predL_uncertain', 'elmentaite21_pred',
'elmentaite21_pred_prob', 'elmentaite21_pred_uncertain', 'suo22_pred',
'suo22_pred_prob', 'suo22_pred_uncertain', 'n_counts', 'log1p_n_counts',
'n_genes', 'log1p_n_genes', 'percent_mito', 'n_counts_mito', 'percent_ribo',
'n_counts_ribo', 'percent_hb', 'n_counts_hb', 'percent_top50', 'n_counts_raw',
'log1p_n_counts_raw', 'n_genes_raw', 'log1p_n_genes_raw', 'percent_mito_raw',
'n_counts_mito_raw', 'percent_ribo_raw', 'n_counts_ribo_raw', 'percent_hb_raw',
'n_counts_hb_raw', 'percent_top50_raw', 'n_counts_spliced',
'log1p_n_counts_spliced', 'n_genes_spliced', 'log1p_n_genes_spliced',
'percent_mito_spliced', 'n_counts_mito_spliced', 'percent_ribo_spliced',
'n_counts_ribo_spliced', 'percent_hb_spliced', 'n_counts_hb_spliced',
'percent_top50_spliced', 'n_counts_unspliced', 'log1p_n_counts_unspliced',
'n_genes_unspliced', 'log1p_n_genes_unspliced', 'percent_mito_unspliced',
'n_counts_mito_unspliced', 'percent_ribo_unspliced', 'n_counts_ribo_unspliced',
```

'percent_hb_unspliced', 'n_counts_hb_unspliced', 'percent_top50_unspliced',
'percent_soup', 'percent_spliced', 'qc_cluster', 'pass_auto_filter_mito20',
'good_qc_cluster_mito20', 'pass_auto_filter_mito50', 'good_qc_cluster_mito50',
'pass_auto_filter_mito80', 'good_qc_cluster_mito80', 'pass_auto_filter',
'good_qc_cluster', 'pass_default', 'sampleID', 'sourceID', 'donorID_original',
'study', 'donorID_corrected', 'donorID_unified', 'donor_category',
'donor_disease', 'organ_original', 'organ_unified', 'organ_broad',
'age_original', 'age_unified', 'age_continuousadult', 'age_continuousdev',
'sex', 'sample_type', 'sample_category', 'sample_retrieval', 'tissue_fraction',
'cell_fraction', 'cell_fraction_unified', 'cell_sorting', 'technology',
'include_150722', 'cluster_scrublet_score', 'bh_pval', 'scrublet_score',
'scrublet_score_z', 'doublet', 'stringent_doublet', 'integration_grouping',
'_scvi_batch', '_scvi_labels', 'broad_annot_20220914', 'martin19_pred',
'martin19_pred_prob', 'martin19_pred_uncertain', 'warner20_pred',
'warner20_pred_prob', 'warner20_pred_uncertain', 'broad_annot_20220917',
'donor_organ_lineage', 'fine_annot', 'fine_annot_original', 'level_1_annot',
'level_2_annot', 'level_3_annot', 'broad_predicted_labels',
'broad_predicted_labels_uncert', 'batch', 'organ_groups', 'control_vs_disease',
'disease', 'ID', 'clusters', 'Palantir_pseudotime', 'initial_size_unspliced',
'initial_size_spliced', 'initial_size', 'velocity_self_transition',
'root_cells', 'end_points', 'velocity_pseudotime', 'start_score',
'macrostates_fwd', 'term_states_fwd', 'term_states_fwd_probs',
'init_states_fwd', 'init_states_fwd_probs', 'clusters_gradients'
    var: 'gene_ids', 'feature_type', 'mito', 'ribo', 'hb', 'cc', 'ig', 'tcr',
'n_counts-0', 'n_counts_raw-0', 'n_counts_spliced-0', 'n_counts_unspliced-0',
'n_cells-0', 'n_cells_raw-0', 'n_cells_spliced-0', 'n_cells_unspliced-0',
'n_counts-1', 'n_counts_raw-1', 'n_counts_spliced-1', 'n_counts_unspliced-1',
'n_cells-1', 'n_cells_raw-1', 'n_cells_spliced-1', 'n_cells_unspliced-1',
'gene_count_corr', 'means', 'dispersions', 'dispersions_norm',
'highly_variable', 'fit_r2', 'fit_alpha', 'fit_beta', 'fit_gamma', 'fit_t_',
'fit_scaling', 'fit_std_u', 'fit_std_s', 'fit_likelihood', 'fit_u0', 'fit_s0',
'fit_pval_steady', 'fit_steady_u', 'fit_steady_s', 'fit_variance',
'fit_alignment_scaling', 'velocity_genes'
    uns: 'disease_colors', 'level_3_annot_colors', 'organ_unified_colors',
'log1p', 'pca', 'neighbors', 'recover_dynamics', 'velocity_params',
'T_fwd_params', 'clusters_colors', 'velocity_graph', 'velocity_graph_neg',
'schur_matrix_fwd', 'eigendecomposition_fwd', 'macrostates_fwd_colors',
'coarse_fwd', 'term_states_fwd_colors', 'init_states_fwd_colors',
'clusters_gradients_colors'
    obsm: 'X_mde', 'X_scANVI', 'X_scVI', 'X_umap', 'X_umap_scvi',
'_scvi_extra_continuous_covs', 'X_pca', 'T_fwd_umap_scvi', 'T_fwd_pca',
'velocity_umap_scvi', 'schur_vectors_fwd', 'macrostates_fwd_memberships',
'term_states_fwd_memberships', 'init_states_fwd_memberships',
'X_fate_simplex_fwd'
    varm: 'PCs', 'loss'
    layers: 'counts', 'spliced', 'unspliced', 'Ms', 'Mu', 'fit_t', 'fit_tau',
'fit_tau_', 'velocity', 'velocity_u'

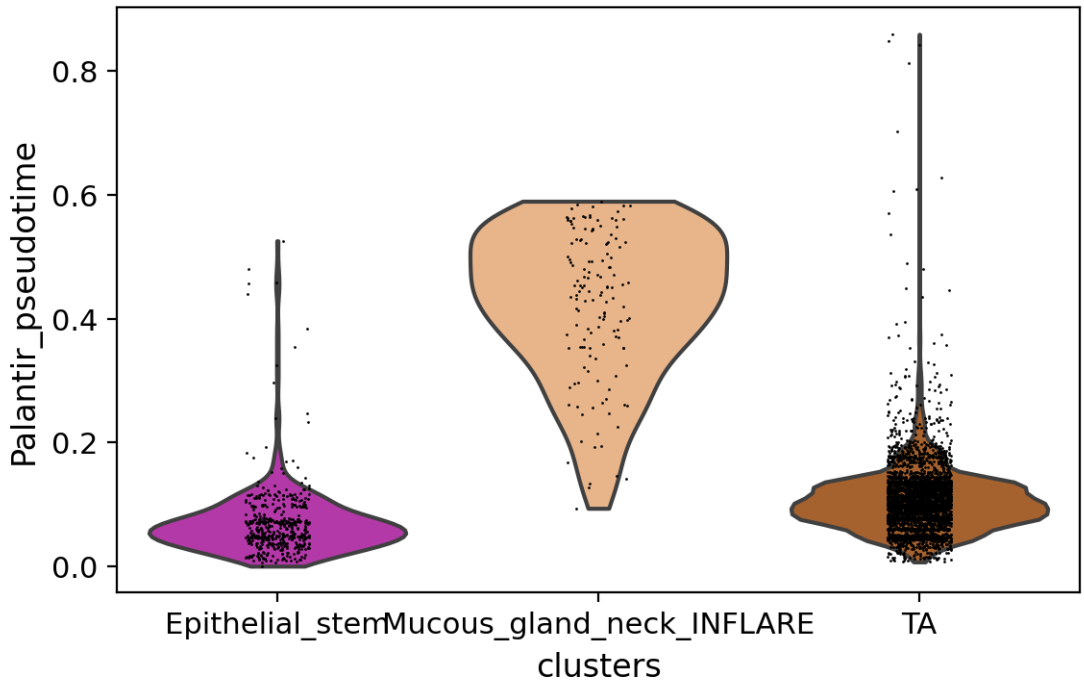
```
    obsp: 'distances', 'connectivities'
```

[52]:
```
sc.pl.violin(adata, keys=["Palantir_pseudotime"], groupby="clusters",␣
↪rotation=90)
```



[54]:
```
temp = adata[np.in1d(adata.obs.clusters, ['TA','Mucous_gland_neck_INFLARE',␣
↪'Epithelial_stem' ])]

#np.unique(adata.obs.clusters)
```

[55]:
```
#sc.pl.violin(temp, keys=["dpt_pseudotime"], groupby="clusters", rotation=90)
sc.pl.violin(temp, keys=["Palantir_pseudotime"], groupby="clusters")
#sc.pl.violin(temp, keys=["ct_pseudotime"], groupby="clusters", rotation=90)
```



[64]:
```
adata.obsm['']
```

[64]:
```
AxisArrays with keys: X_mde, X_scANVI, X_scVI, X_umap, X_umap_scvi,
_scvi_extra_continuous_covs, X_pca, T_fwd_umap_scvi, T_fwd_pca,
velocity_umap_scvi, schur_vectors_fwd, macrostates_fwd_memberships,
term_states_fwd_memberships, init_states_fwd_memberships, X_fate_simplex_fwd
```

[60]:
```
# re-assigning neighbourhood graph
#sc.pp.neighbors(adata_all, use_rep="X_scVI",method='gauss')
#adata.uns['neighbors']['distances'] = adata.obsp['distances']
#adata.uns['neighbors']['connectivities'] = adata.obsp['connectivities']

#scv.tl.paga(adata, groups='clusters', use_time_prior="palantir_pseudotime")
#df = scv.get_df(adata, 'paga/transitions_confidence', precision=2).T
#df.style.background_gradient(cmap='Blues').format('{:.2g}')
#scv.pl.paga(adata, basis='umap_scvi', size=50, alpha=.1,
 #           min_edge_width=1, node_size_scale=1.5, palette = our_cmap,␣
 ↪threshold=0.1)
#df.style.background_gradient(cmap='Blues').format('{:.2g}')
```

[61]:
```
adata.obs
```

[61]:
```
AnnData object with n_obs × n_vars = 20765 × 10845
    obs: 'latent_cell_probability', 'latent_RT_efficiency', 'cecilia22_predH',
```

'cecilia22_predH_prob', 'cecilia22_predH_uncertain', 'cecilia22_predL',
'cecilia22_predL_prob', 'cecilia22_predL_uncertain', 'elmentaite21_pred',
'elmentaite21_pred_prob', 'elmentaite21_pred_uncertain', 'suo22_pred',
'suo22_pred_prob', 'suo22_pred_uncertain', 'n_counts', 'log1p_n_counts',
'n_genes', 'log1p_n_genes', 'percent_mito', 'n_counts_mito', 'percent_ribo',
'n_counts_ribo', 'percent_hb', 'n_counts_hb', 'percent_top50', 'n_counts_raw',
'log1p_n_counts_raw', 'n_genes_raw', 'log1p_n_genes_raw', 'percent_mito_raw',
'n_counts_mito_raw', 'percent_ribo_raw', 'n_counts_ribo_raw', 'percent_hb_raw',
'n_counts_hb_raw', 'percent_top50_raw', 'n_counts_spliced',
'log1p_n_counts_spliced', 'n_genes_spliced', 'log1p_n_genes_spliced',
'percent_mito_spliced', 'n_counts_mito_spliced', 'percent_ribo_spliced',
'n_counts_ribo_spliced', 'percent_hb_spliced', 'n_counts_hb_spliced',
'percent_top50_spliced', 'n_counts_unspliced', 'log1p_n_counts_unspliced',
'n_genes_unspliced', 'log1p_n_genes_unspliced', 'percent_mito_unspliced',
'n_counts_mito_unspliced', 'percent_ribo_unspliced', 'n_counts_ribo_unspliced',
'percent_hb_unspliced', 'n_counts_hb_unspliced', 'percent_top50_unspliced',
'percent_soup', 'percent_spliced', 'qc_cluster', 'pass_auto_filter_mito20',
'good_qc_cluster_mito20', 'pass_auto_filter_mito50', 'good_qc_cluster_mito50',
'pass_auto_filter_mito80', 'good_qc_cluster_mito80', 'pass_auto_filter',
'good_qc_cluster', 'pass_default', 'sampleID', 'sourceID', 'donorID_original',
'study', 'donorID_corrected', 'donorID_unified', 'donor_category',
'donor_disease', 'organ_original', 'organ_unified', 'organ_broad',
'age_original', 'age_unified', 'age_continuousadult', 'age_continuousdev',
'sex', 'sample_type', 'sample_category', 'sample_retrieval', 'tissue_fraction',
'cell_fraction', 'cell_fraction_unified', 'cell_sorting', 'technology',
'include_150722', 'cluster_scrublet_score', 'bh_pval', 'scrublet_score',
'scrublet_score_z', 'doublet', 'stringent_doublet', 'integration_grouping',
'_scvi_batch', '_scvi_labels', 'broad_annot_20220914', 'martin19_pred',
'martin19_pred_prob', 'martin19_pred_uncertain', 'warner20_pred',
'warner20_pred_prob', 'warner20_pred_uncertain', 'broad_annot_20220917',
'donor_organ_lineage', 'fine_annot', 'fine_annot_original', 'level_1_annot',
'level_2_annot', 'level_3_annot', 'broad_predicted_labels',
'broad_predicted_labels_uncert', 'batch', 'organ_groups', 'control_vs_disease',
'disease', 'ID', 'clusters', 'Palantir_pseudotime', 'initial_size_unspliced',
'initial_size_spliced', 'initial_size', 'velocity_self_transition',
'root_cells', 'end_points', 'velocity_pseudotime', 'start_score',
'macrostates_fwd', 'term_states_fwd', 'term_states_fwd_probs',
'init_states_fwd', 'init_states_fwd_probs', 'clusters_gradients'
   var: 'gene_ids', 'feature_type', 'mito', 'ribo', 'hb', 'cc', 'ig', 'tcr',
'n_counts-0', 'n_counts_raw-0', 'n_counts_spliced-0', 'n_counts_unspliced-0',
'n_cells-0', 'n_cells_raw-0', 'n_cells_spliced-0', 'n_cells_unspliced-0',
'n_counts-1', 'n_counts_raw-1', 'n_counts_spliced-1', 'n_counts_unspliced-1',
'n_cells-1', 'n_cells_raw-1', 'n_cells_spliced-1', 'n_cells_unspliced-1',
'gene_count_corr', 'means', 'dispersions', 'dispersions_norm',
'highly_variable', 'fit_r2', 'fit_alpha', 'fit_beta', 'fit_gamma', 'fit_t_',
'fit_scaling', 'fit_std_u', 'fit_std_s', 'fit_likelihood', 'fit_u0', 'fit_s0',
'fit_pval_steady', 'fit_steady_u', 'fit_steady_s', 'fit_variance',

```
'fit_alignment_scaling', 'velocity_genes'
    uns: 'disease_colors', 'level_3_annot_colors', 'organ_unified_colors',
'log1p', 'pca', 'neighbors', 'recover_dynamics', 'velocity_params',
'T_fwd_params', 'clusters_colors', 'velocity_graph', 'velocity_graph_neg',
'schur_matrix_fwd', 'eigendecomposition_fwd', 'macrostates_fwd_colors',
'coarse_fwd', 'term_states_fwd_colors', 'init_states_fwd_colors',
'clusters_gradients_colors', 'paga', 'clusters_sizes'
    obsm: 'X_mde', 'X_scANVI', 'X_scVI', 'X_umap', 'X_umap_scvi',
'_scvi_extra_continuous_covs', 'X_pca', 'T_fwd_umap_scvi', 'T_fwd_pca',
'velocity_umap_scvi', 'schur_vectors_fwd', 'macrostates_fwd_memberships',
'term_states_fwd_memberships', 'init_states_fwd_memberships',
'X_fate_simplex_fwd'
    varm: 'PCs', 'loss'
    layers: 'counts', 'spliced', 'unspliced', 'Ms', 'Mu', 'fit_t', 'fit_tau',
'fit_tau_', 'velocity', 'velocity_u'
    obsp: 'distances', 'connectivities'
```

[ ]:

6 Gene trends

[177]:
```
stem_markers = ['LGR5','LEFTY1','OLFM4','SOX9']
inflare_markers = ['PGC','MUC6','AQP5']
```

[178]:
```
#model = cr.models.GAMR(adata, n_knots=6, smoothing_penalty=10.0)

model = cr.models.GAM(adata, distribution='gaussian', link='identity')
```

[179]:
```
g
```

[179]:
```
GPCCA[kernel=(0.75 * PseudotimeKernel[n=8764] + 0.25 * VelocityKernel[n=8764]),
initial_states=['Mucous_gland_neck_INFLARE'], terminal_states=['Enterocyte_1',
'Enterocyte_2', 'Goblet_progenitor', 'Mucous_gland_neck_INFLARE', 'TA', 'Tuft']]
```

[200]:
```
beta_drivers
```

[200]:

	Mucous_gland_neck_INFLARE_corr	Mucous_gland_neck_INFLARE_pval	\
BPIFB1	0.89	0.00e+00	
MUC6	0.87	0.00e+00	
PGC	0.86	0.00e+00	
AQP5	0.78	0.00e+00	
C6orf58	0.61	0.00e+00	
…	…	…	
FABP1	-0.19	6.12e-69	
ALDOB	-0.19	7.92e-72	
FABP2	-0.19	5.79e-73	
PRAP1	-0.19	6.30e-74	

```
PHGR1                                -0.28                          7.35e-162

         Mucous_gland_neck_INFLARE_qval  Mucous_gland_neck_INFLARE_ci_low  \
BPIFB1                         0.00e+00                              0.89
MUC6                           0.00e+00                              0.87
PGC                            0.00e+00                              0.85
AQP5                           0.00e+00                              0.77
C6orf58                        0.00e+00                              0.60
…                                   …                                 …
FABP1                          1.23e-66                             -0.21
ALDOB                          1.71e-69                             -0.21
FABP2                          1.28e-70                             -0.21
PRAP1                          1.43e-71                             -0.21
PHGR1                         2.78e-159                             -0.30

         Mucous_gland_neck_INFLARE_ci_high
BPIFB1                                0.89
MUC6                                  0.88
PGC                                   0.86
AQP5                                  0.79
C6orf58                               0.62
…                                        …
FABP1                                -0.17
ALDOB                                -0.17
FABP2                                -0.17
PRAP1                                -0.17
PHGR1                                -0.26

[9080 rows x 5 columns]
```

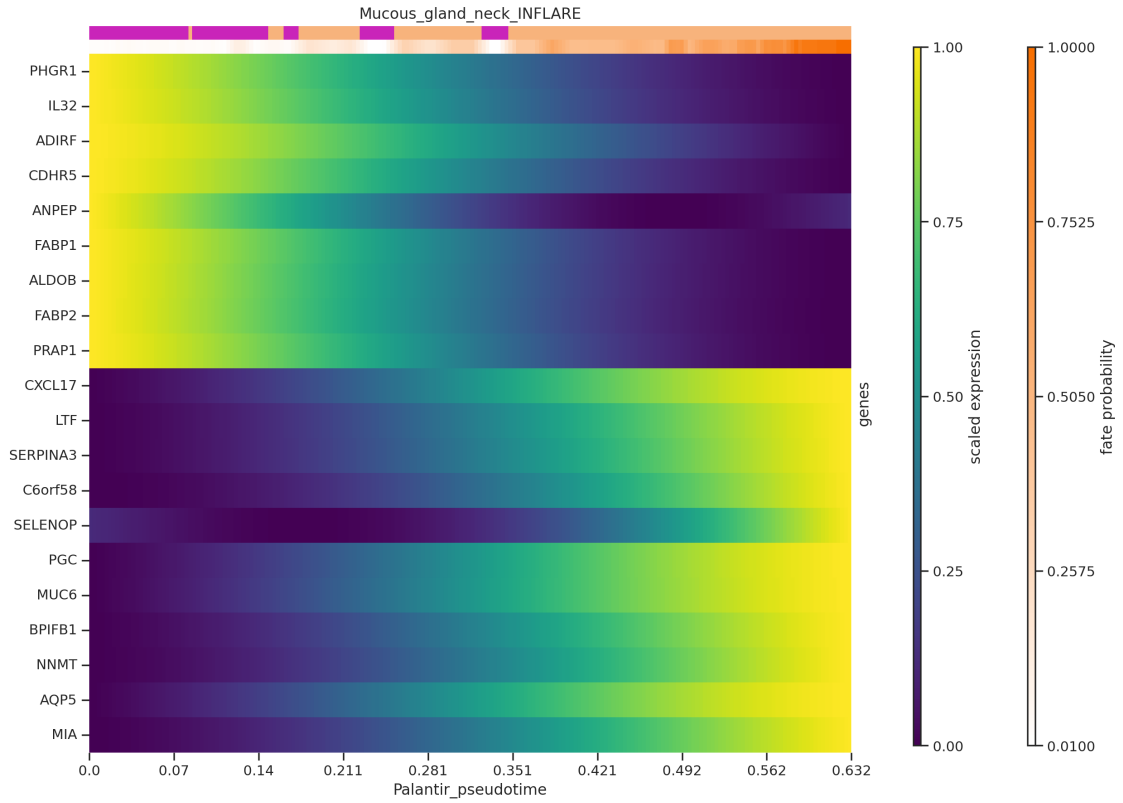
[214]:
```
# putative driver genes are found by correlating fate probabilities with gene␣
 ↪expression
#  -- if a gene is systematically higher or lower expressed in cells that are␣
 ↪more or less likely to differentiate towards a given terminal state,
#then it is called driver gene

# compute putative drivers for the Beta trajectory
g= gpcca
model = cr.models.GAM(INFLARE_lineage_cells, distribution='gaussian',␣
 ↪link='identity')
beta_drivers = g.compute_lineage_drivers(lineages="Mucous_gland_neck_INFLARE")

# plot heatmap
cr.pl.heatmap(
    INFLARE_lineage_cells,
    model=model,  # use the model from before
    lineages="Mucous_gland_neck_INFLARE",
```

```
    cluster_key="clusters",
    show_fate_probabilities=True,
    #data_key="magic_imputed_data",
    genes= list(beta_drivers.tail(10).index) + list(beta_drivers.head(10).
 ↪index),
    time_key="Palantir_pseudotime",
    figsize=(12, 10),
    show_all_genes=True,
    weight_threshold=(1e-3, 1e-3),
)
```

```
Adding `adata.varm['terminal_lineage_drivers']`
       `.lineage_drivers`
    Finish (0:00:00)
Computing trends using `1` core(s)
```

```
100%|      | 20/20 [00:00<00:00, 54.73gene/s]
```

```
    Finish (0:00:00)
```



[187]:
```
gpcca
```

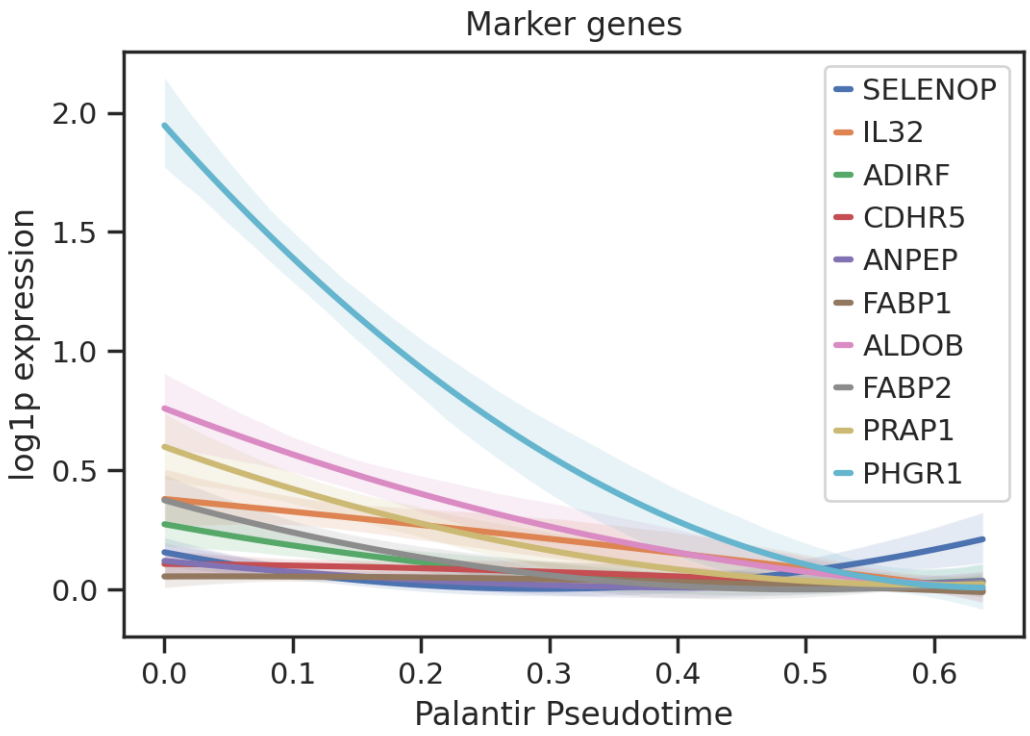
[187]:
```
GPCCA[kernel=(0.75 * PseudotimeKernel[n=8764] + 0.25 * VelocityKernel[n=8764]),
initial_states=['Mucous_gland_neck_INFLARE'], terminal_states=['Enterocyte_1',
'Enterocyte_2', 'Goblet_progenitor', 'Mucous_gland_neck_INFLARE', 'TA', 'Tuft']]
```

[148]:
```
INFLARE_lineage_cells = adata[np.in1d(adata.obs.clusters,␣
↪['Mucous_gland_neck_INFLARE','Epithelial_stem' ])]
```

[191]:
```
beta_drivers.head(5).index
```

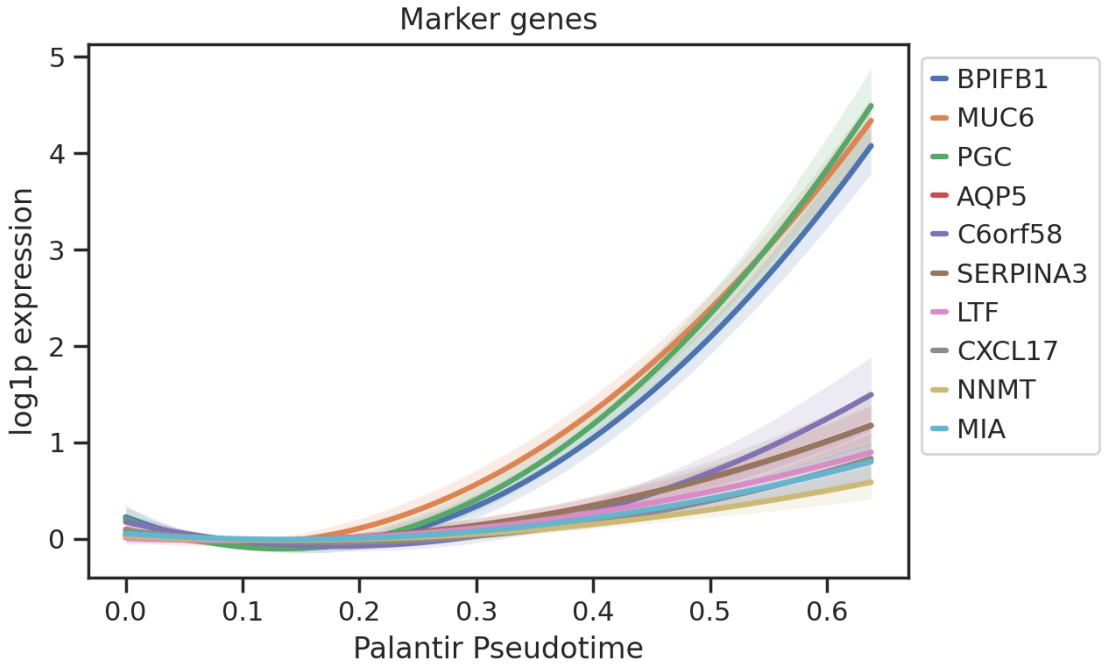
[191]:
```
Index(['BPIFB1', 'MUC6', 'PGC', 'AQP5', 'C6orf58'], dtype='object')
```

[228]:
```
import matplotlib.pyplot as plt


datasets = {}
gof= list(beta_drivers.tail(10).index) #stem_markers+inflare_markers
x = list(INFLARE_lineage_cells.obs.avg_pseudotime)
for gene in gof:
    y = np.asarray(INFLARE_lineage_cells[:,gene].X.todense() ).flatten()
    data = pd.DataFrame([x,y]).transpose()
    data.columns = ['x','y']
    datasets[gene] = data

# Create the first lmplot
sb.set(style="ticks")
fig, ax = plt.subplots()

for g in gof:
# Plot the first lmplot on the specified axes
    sb.regplot(x="x", y="y", data=datasets[g], ax=ax, scatter_kws={"s": 10},␣
↪label=g, order=2,scatter=False)

# Set labels and title
ax.set_ylabel("log1p expression")
ax.set_xlabel("Palantir Pseudotime")
ax.set_title("Marker genes")
#ax.set_ylim([-0.5,5])


# Show the legend
ax.legend(bbox_to_anchor=(1, 1))


# Show the plots
plt.show()
```



[229]:
```
datasets = {}
gof= list(beta_drivers.head(10).index) #stem_markers+inflare_markers
x = list(INFLARE_lineage_cells.obs.avg_pseudotime)
for gene in gof:
    y = np.asarray(INFLARE_lineage_cells[:,gene].X.todense() ).flatten()
    data = pd.DataFrame([x,y]).transpose()
    data.columns = ['x','y']
    datasets[gene] = data

# Create the first lmplot
sb.set(style="ticks")
fig, ax = plt.subplots()

for g in gof:
# Plot the first lmplot on the specified axes
    sb.regplot(x="x", y="y", data=datasets[g], ax=ax, scatter_kws={"s": 10},␣
 ↪label=g, order=2,scatter=False)

# Set labels and title
ax.set_ylabel("log1p expression")
ax.set_xlabel("Palantir Pseudotime")
ax.set_title("Marker genes")
#ax.set_ylim([-0.5,5])
```

```
# Show the legend
ax.legend(bbox_to_anchor=(1, 1))


# Show the plots
plt.show()
```



[ ]:

[ ]:

[208]:
```
import matplotlib.pyplot as plt


datasets = {}
gof=list(beta_drivers.index[0:5])
gof = stem_markers + inflare_markers +  [ 'MUC5B',  'CEACAM7',␣
 ↪'SLC26A3','MKI67']

x = list(INFLARE_cells.obs.Palantir_pseudotime)
for g in gof:
    y = np.asarray(INFLARE_cells[:,g].X.todense() ).flatten()
```

```
    data = pd.DataFrame([x,y]).transpose()
    data.columns = ['x','y']
    datasets[g] = data

# Create the first lmplot
sb.set(style="ticks")
fig, ax = plt.subplots()

for g in gof:
# Plot the first lmplot on the specified axes
    if(g=='LEFTY1'):
        continue
    sb.regplot(x="x", y="y", data=datasets[g], ax=ax, scatter_kws={"s": 10},␣
 ↪label=g, order=2, scatter=False)

# Set labels and title
ax.set_ylabel("log1p expression")
ax.set_xlabel("Palantir Pseudotime")
ax.set_title("Marker genes")
ax.set_ylim([-0.5,5])


# Show the legend
ax.legend(bbox_to_anchor=(1, 1))


# Show the plots
plt.show()
```

[208]:
```
1.5144212
```

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

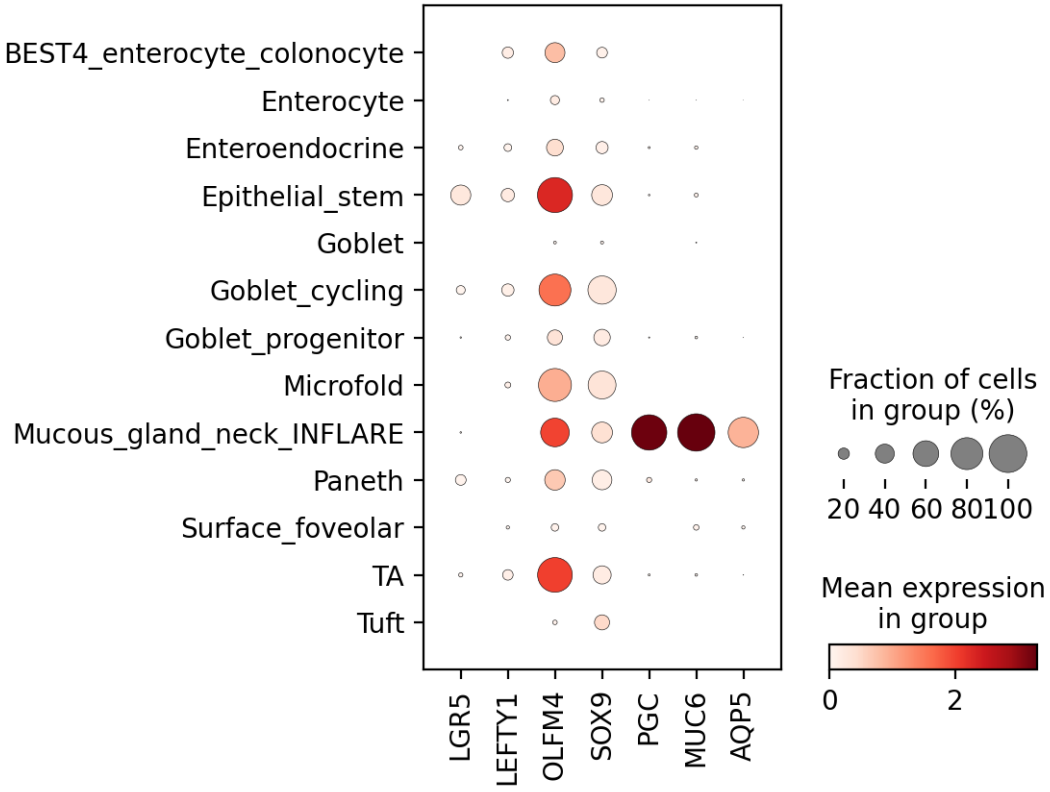
[ ]:

[ ]:

[129]:
```
np.asarray(adata[:,'LEFTY1'].X.todense()).flatten()
```

[129]:
```
array([0., 0., 0., …, 0., 0., 0.], dtype=float32)
```

[131]:
```
sc.pl.dotplot(adata[:,stem_markers+inflare_markers],␣
↪stem_markers+inflare_markers, groupby='clusters', swap_axes=False)
```



[132]:
```
adata[adata.obs.clusters == 'Mucous_gland_neck_INFLARE'][:,'LEFTY1'].X.data # 0␣
↪expressed in this subset
```

[132]:
```
array([], dtype=float32)
```

[133]:
```
adata.obs.organ_unified
```

[133]:
```
index
AAAGATGAGTCCAGGA-4918STDY7273964    ileum
AAAGATGTCTAACTTC-4918STDY7273964    ileum
AAAGCAATCTTGTCAT-4918STDY7273964    ileum
AAAGTAGAGAATGTTG-4918STDY7273964    ileum
```

```
AAAGTAGAGTCCAGGA-4918STDY7273964    ileum
                                    …
GCTGGGTTCAAAGTAG-GSM3972030         ileum
GTATCTTAGCCAGGAT-GSM3972030         ileum
GTCGTAAAGACTCGGA-GSM3972030         ileum
GTGGGTCGTCGGATCC-GSM3972030         ileum
TTCTACATCGTACGGC-GSM3972030         ileum
Name: organ_unified, Length: 8764, dtype: category
Categories (1, object): ['ileum']
```

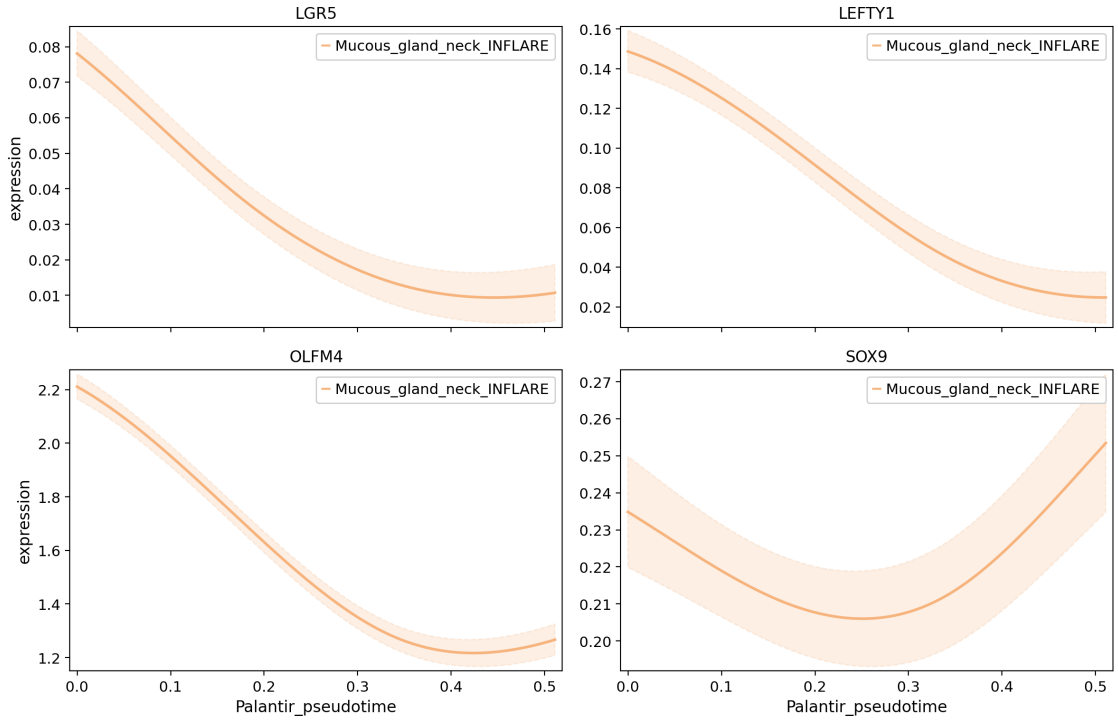
[153]:
```
beta_drivers
```

[153]:
```
         Mucous_gland_neck_INFLARE_corr  Mucous_gland_neck_INFLARE_pval  \
BPIFB1                             0.89                        0.00e+00
MUC6                               0.87                        0.00e+00
PGC                                0.86                        0.00e+00
AQP5                               0.78                        0.00e+00
C6orf58                            0.61                        0.00e+00
…                                     …                           …
FABP1                             -0.19                        6.12e-69
ALDOB                             -0.19                        7.92e-72
FABP2                             -0.19                        5.79e-73
PRAP1                             -0.19                        6.30e-74
PHGR1                             -0.28                       7.35e-162

         Mucous_gland_neck_INFLARE_qval  Mucous_gland_neck_INFLARE_ci_low  \
BPIFB1                         0.00e+00                              0.89
MUC6                           0.00e+00                              0.87
PGC                            0.00e+00                              0.85
AQP5                           0.00e+00                              0.77
C6orf58                        0.00e+00                              0.60
…                                 …                                 …
FABP1                          1.23e-66                             -0.21
ALDOB                          1.71e-69                             -0.21
FABP2                          1.28e-70                             -0.21
PRAP1                          1.43e-71                             -0.21
PHGR1                         2.78e-159                             -0.30

         Mucous_gland_neck_INFLARE_ci_high
BPIFB1                                0.89
MUC6                                  0.88
PGC                                   0.86
AQP5                                  0.79
C6orf58                               0.62
…                                        …
FABP1                                -0.17
ALDOB                                -0.17
```

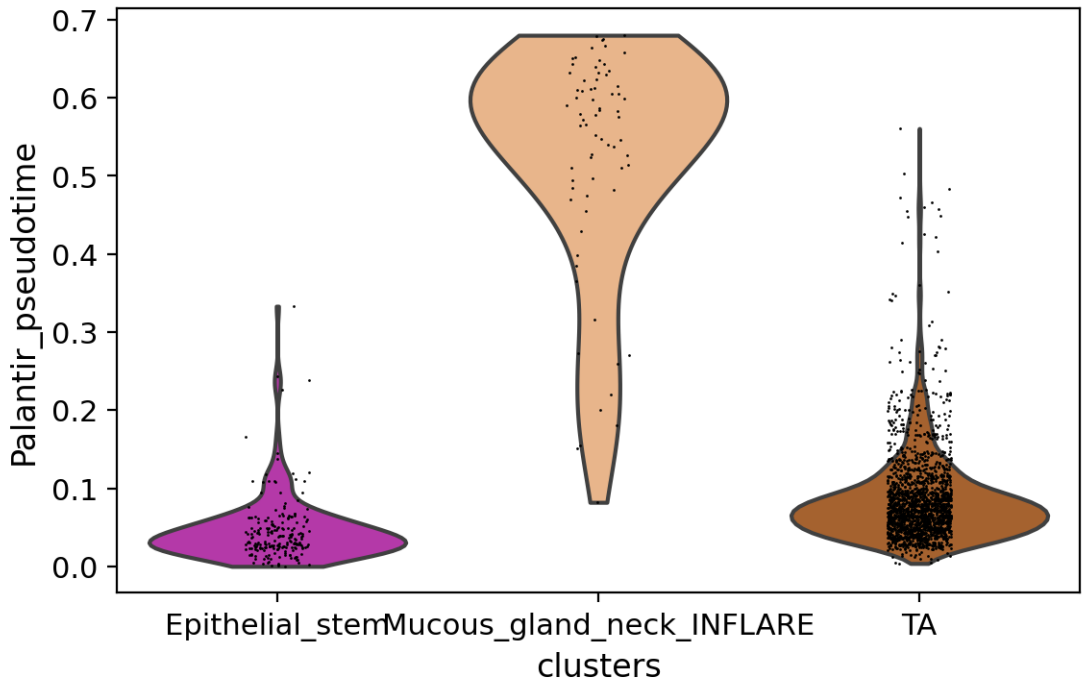
```
FABP2                                -0.17
PRAP1                                -0.17
PHGR1                                -0.26

[9080 rows x 5 columns]
```

[135]:
```
cr.pl.gene_trends(
    adata,
    model=model,
    lineages = ['Mucous_gland_neck_INFLARE'],#np.unique(temp.obs.clusters),
   # data_key="magic_imputed_data",
    genes=stem_markers,
    same_plot=True,
    sharex=True,
    ncols=2,
    time_key="Palantir_pseudotime",
    hide_cells=True,
    weight_threshold=(1e-3, 1e-3),
)
```

```
Computing trends using `1` core(s)

100%|      | 4/4 [00:00<00:00, 10.60gene/s]

    Finish (0:00:00)
Plotting trends
```



[136]:
```
sc.pl.violin(temp, keys=["Palantir_pseudotime"], groupby="clusters")
```



[137]:
```
adata.write_h5ad('final_adata_v1_05122023.h5ad')
```

[138]:
```
print('a')
```

```
a
```

[ ]:
```

```

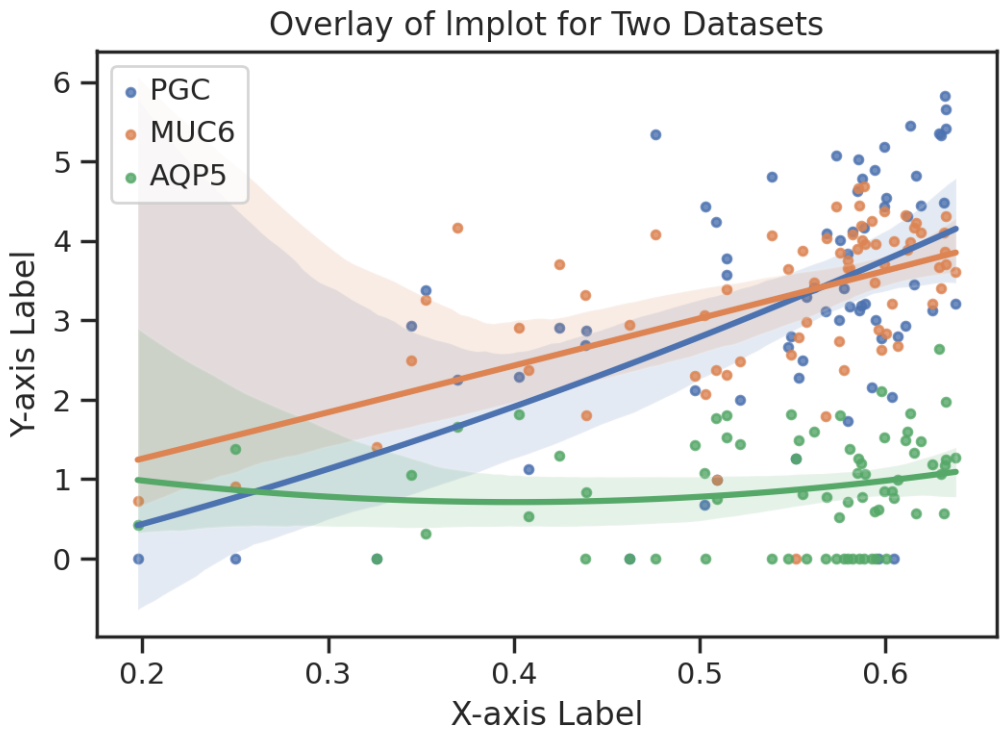
[139]:
```
TA_cells = adata[adata.obs.clusters == 'TA']
INFLARE_cells = adata[adata.obs.clusters == 'Mucous_gland_neck_INFLARE']
stem_cells = adata[adata.obs.clusters == 'Epithelial_stem']
print(TA_cells.shape)
print(INFLARE_cells.shape)
print(stem_cells.shape)
```

```
(1927, 9080)
(71, 9080)
(209, 9080)
```

[140]:
```
stem_markers + inflare_markers
```

[140]:
```
['LGR5', 'LEFTY1', 'OLFM4', 'SOX9', 'PGC', 'MUC6', 'AQP5']
```

[141]:
```
import matplotlib.pyplot as plt


datasets = {}
gof= inflare_markers
x = list(INFLARE_cells.obs.avg_pseudotime)
for g in gof:
    y = np.asarray(INFLARE_cells[:,g].X.todense() ).flatten()
    data = pd.DataFrame([x,y]).transpose()
    data.columns = ['x','y']
    datasets[g] = data

# Create the first lmplot
sb.set(style="ticks")
fig, ax = plt.subplots()

for g in gof:
# Plot the first lmplot on the specified axes
    sb.regplot(x="x", y="y", data=datasets[g], ax=ax, scatter_kws={"s": 10},␣
 ↪label=g, order=2)

# Set labels and title
ax.set_xlabel("X-axis Label")
ax.set_ylabel("Y-axis Label")
ax.set_title("Overlay of lmplot for Two Datasets")

# Show the legend
ax.legend()

# Show the plots
plt.show()
```

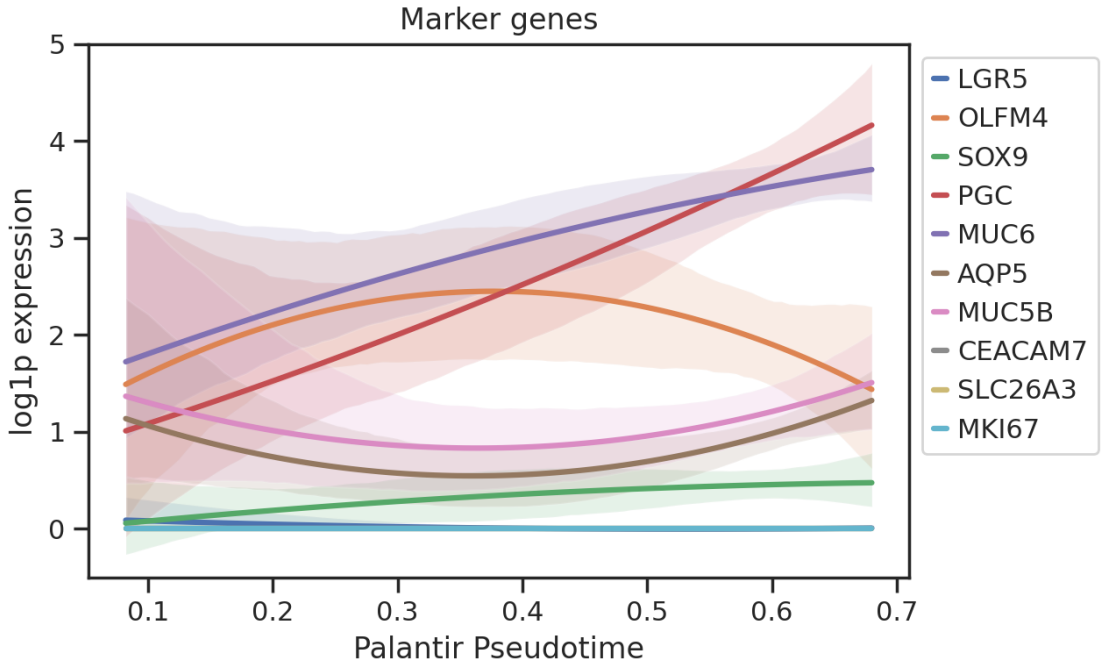


[142]:
```
list(beta_drivers.index[0:5])
```

[142]:
```
['BPIFB1', 'MUC6', 'PGC', 'AQP5', 'C6orf58']
```

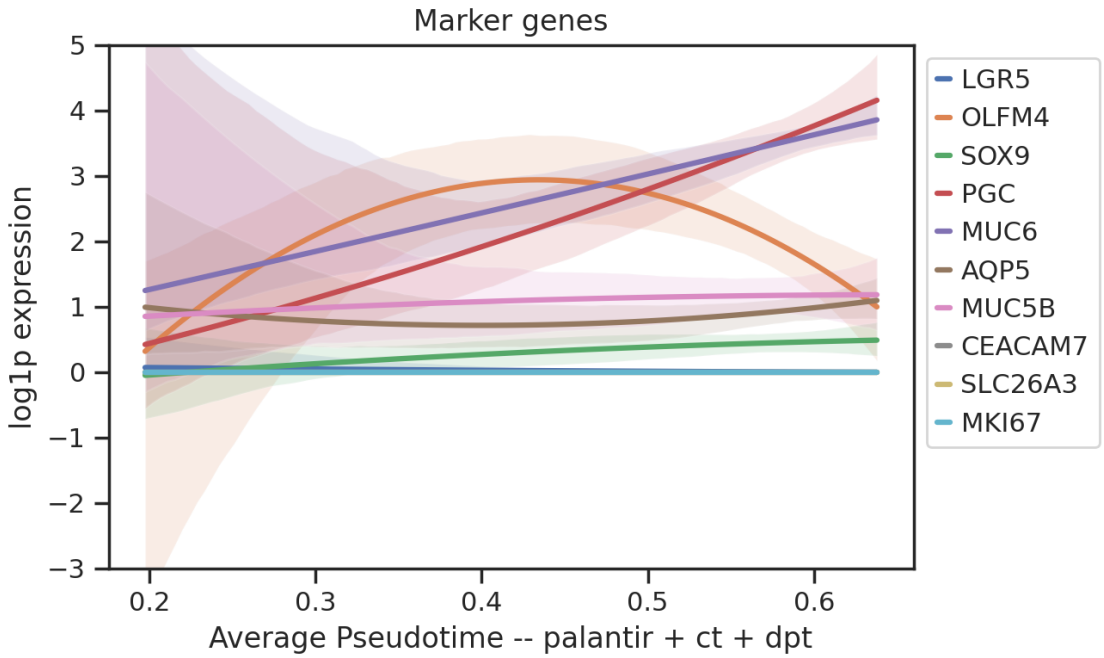
[143]:
```
import matplotlib.pyplot as plt


datasets = {}
gof=list(beta_drivers.index[0:5])
gof = stem_markers + inflare_markers +  [ 'MUC5B',  'CEACAM7',␣
 ↪'SLC26A3','MKI67']

x = list(INFLARE_cells.obs.Palantir_pseudotime)
for g in gof:
    y = np.asarray(INFLARE_cells[:,g].X.todense() ).flatten()
    data = pd.DataFrame([x,y]).transpose()
    data.columns = ['x','y']
    datasets[g] = data

# Create the first lmplot
sb.set(style="ticks")
fig, ax = plt.subplots()
```

```
for g in gof:
# Plot the first lmplot on the specified axes
    if(g=='LEFTY1'):
        continue
    sb.regplot(x="x", y="y", data=datasets[g], ax=ax, scatter_kws={"s": 10},␣
 ↪label=g, order=2, scatter=False)

# Set labels and title
ax.set_ylabel("log1p expression")
ax.set_xlabel("Palantir Pseudotime")
ax.set_title("Marker genes")
ax.set_ylim([-0.5,5])


# Show the legend
ax.legend(bbox_to_anchor=(1, 1))


# Show the plots
plt.show()
```



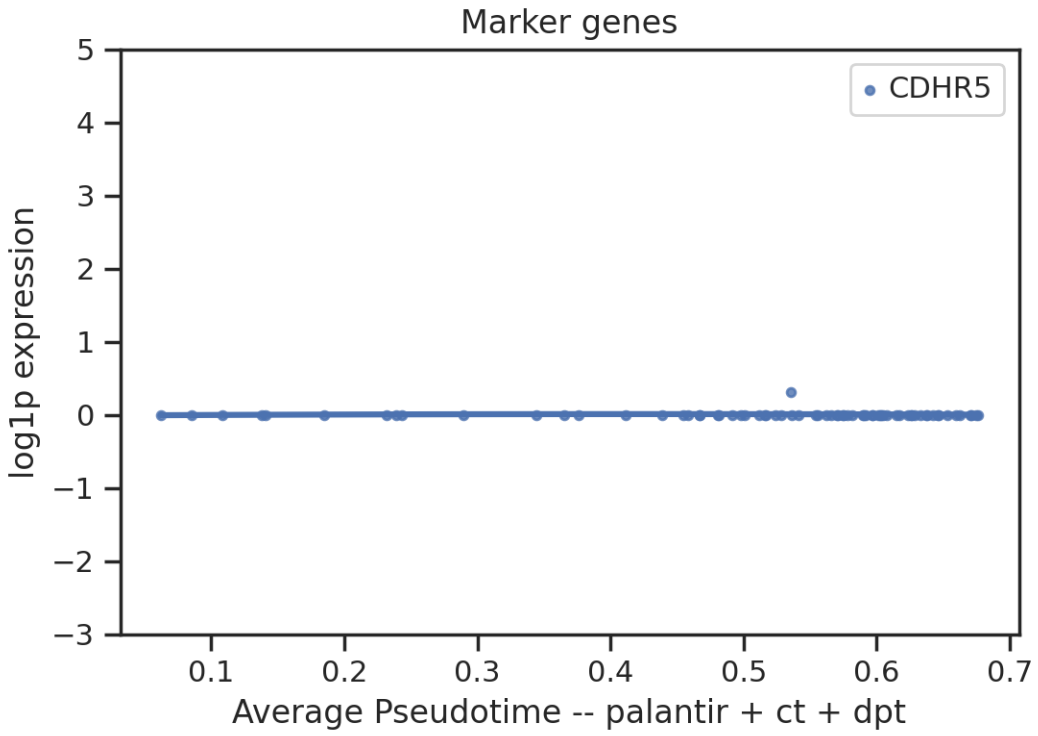
[144]:
```
import matplotlib.pyplot as plt
```

```
datasets = {}
gof=list(beta_drivers.index[0:5])
gof = stem_markers + inflare_markers +  [ 'MUC5B',  'CEACAM7',␣
 ↪'SLC26A3','MKI67']

x = list(INFLARE_cells.obs.avg_pseudotime)
for g in gof:
    y = np.asarray(INFLARE_cells[:,g].X.todense() ).flatten()
    data = pd.DataFrame([x,y]).transpose()
    data.columns = ['x','y']
    datasets[g] = data

# Create the first lmplot
sb.set(style="ticks")
fig, ax = plt.subplots()

for g in gof:
# Plot the first lmplot on the specified axes
    if(g=='LEFTY1'):
        continue
    sb.regplot(x="x", y="y", data=datasets[g], ax=ax, scatter_kws={"s": 10},␣
 ↪label=g, order=2, scatter=False)

# Set labels and title
ax.set_ylabel("log1p expression")
ax.set_xlabel("Average Pseudotime -- palantir + ct + dpt")
ax.set_title("Marker genes")
ax.set_ylim([-3,5])


# Show the legend
ax.legend(bbox_to_anchor=(1, 1))


# Show the plots
plt.show()
```



[351]:
```
import matplotlib.pyplot as plt


datasets = {}
gof=list(beta_drivers.index[0:5])
gof = ['CDHR5' ]

x = list(INFLARE_cells.obs.Palantir_pseudotime)
for g in gof:
    y = np.asarray(INFLARE_cells[:,g].X.todense() ).flatten()
    data = pd.DataFrame([x,y]).transpose()
    data.columns = ['x','y']
    datasets[g] = data

# Create the first lmplot
sb.set(style="ticks")
fig, ax = plt.subplots()

for g in gof:
# Plot the first lmplot on the specified axes
    if(g=='LEFTY1'):
        continue
    sb.regplot(x="x", y="y", data=datasets[g], ax=ax, scatter_kws={"s": 10},␣
 ↪label=g, order=2, scatter=True)
```

```
# Set labels and title
ax.set_ylabel("log1p expression")
ax.set_xlabel("Average Pseudotime -- palantir + ct + dpt")
ax.set_title("Marker genes")
ax.set_ylim([-3,5])


# Show the legend
ax.legend(bbox_to_anchor=(1, 1))


# Show the plots
plt.show()
```



[ ]: