# Demo-DRandClustering

May 17, 2022

### 0.0.1 (V)AE models for dimensionality reduction and clustering

In this section we'll look at the pros and const of different autoencoder models for the purpose of dimensionality reduction and clutering, specifically applied to a scRNAseq data set, the gtex version 9 (https://gtexportal.org/home/datasets). This data set was used in the research of Eraslan et. al., bioRxiv, 2021, and it comes with their umap and the latent space of the CVAE so it is a good reference point.

This dataset is fairly large (over 200k cells, over 17k genes)

[1]:
```
# load libraries and code
#import gdown
import matplotlib.pyplot as plt
import numpy as np
#import os
import pandas as pd
import pyro
import pyro.distributions as pyrodist
import scanpy as sc
import seaborn as sns
#import time
import torch
import torch.utils.data
import torchvision.utils as vutils
import umap
from abc import abstractmethod
from anndata.experimental.pytorch import AnnLoader
from importlib import reload
from math import pi, sin, cos, sqrt, log
from pyro.infer import SVI, JitTrace_ELBO, Trace_ELBO, TraceGraph_ELBO
from pyro.optim import Adam
import sklearn
from sklearn import datasets as skds
from sklearn.cluster import KMeans
from sklearn.metrics.cluster import normalized_mutual_info_score
from sklearn.preprocessing import OneHotEncoder, LabelEncoder
from sklearn import mixture
from toolz import partial, curry
from torch import nn, optim, distributions, Tensor
```

```
from torch.nn.functional import one_hot
from torchvision import datasets, transforms, models
from torchvision.utils import save_image, make_grid
from typing import Callable, Iterator, Union, Optional, TypeVar
from typing import List, Set, Dict, Tuple
from typing import Mapping, MutableMapping, Sequence, Iterable
from typing import Union, Any, cast, IO, TextIO
from torch.utils.data import WeightedRandomSampler
# my own sauce
from my_torch_utils import denorm, normalize, mixedGaussianCircular
from my_torch_utils import fclayer, init_weights, buildNetwork
from my_torch_utils import fnorm, replicate, logNorm, log_gaussian_prob
from my_torch_utils import plot_images, save_reconstructs,␣
 ↪save_random_reconstructs
from my_torch_utils import scsimDataset
import my_torch_utils as ut
from importlib import reload
from torch.nn import functional as F
import gmmvae03 as M3
import gmmvae04 as M4
import gmmvae05 as M5
import gmmvae06 as M6
import gmmvae07 as M7
print(torch.cuda.is_available())


sc.settings.verbosity = 3
sc.logging.print_header()
sc.settings.set_figure_params(dpi=120, facecolor='white', )

enc = OneHotEncoder(sparse=False, dtype=np.float32)
enc_ct = LabelEncoder()

adata = sc.read("./data/GTEx_8_tissues_snRNAseq_atlas_071421.public_obs.h5ad",)
# keep a second copy in the original state
adatac = sc.read("./data/GTEx_8_tissues_snRNAseq_atlas_071421.public_obs.h5ad",)
```

```
True
True
True
True
True
True
scanpy==1.8.2 anndata==0.7.8 umap==0.5.2 numpy==1.22.1 scipy==1.7.3
pandas==1.4.0 scikit-learn==1.0.2 statsmodels==0.13.1 python-igraph==0.9.9
louvain==0.7.1 pynndescent==0.5.6
```

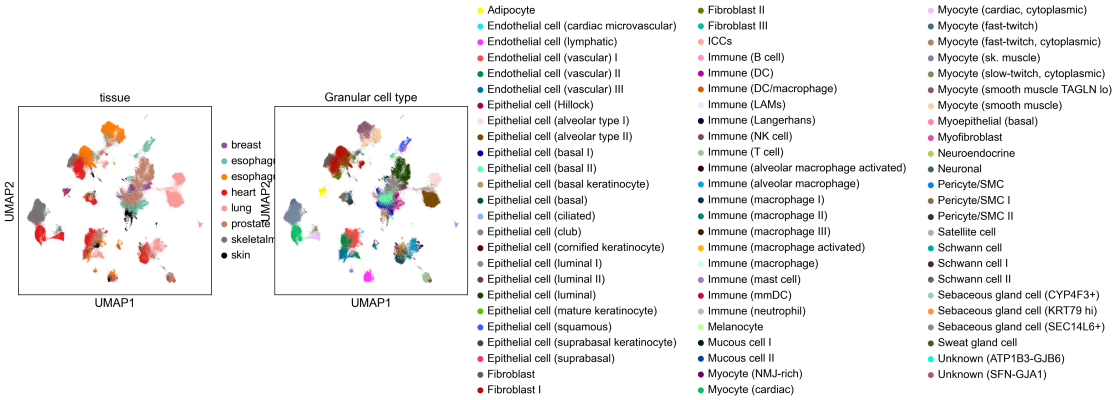
results from Eraslan et. al. They used a type of CVAE (conditional VAE) for dimensionality

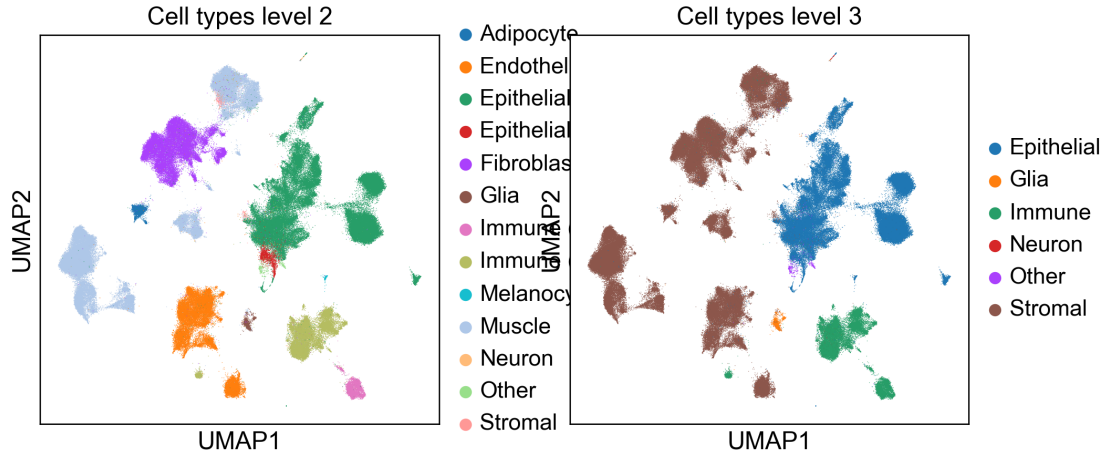
reduction and batch effect reduction.

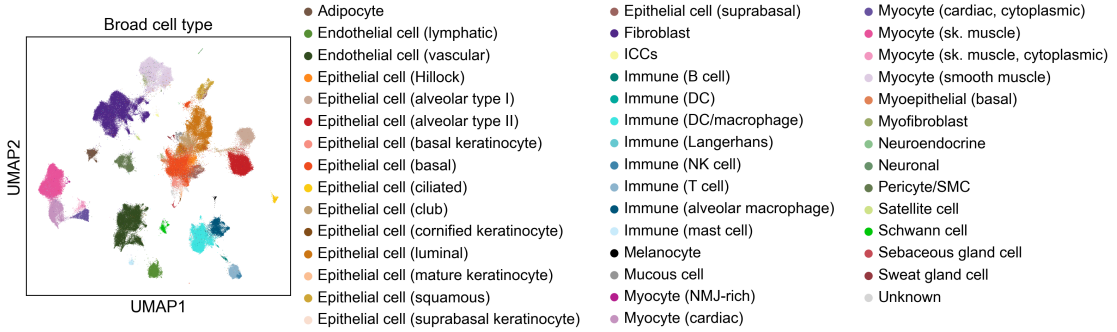
[2]:
```
adatac
```

[2]:
```
AnnData object with n_obs × n_vars = 209126 × 17695
    obs: 'n_genes', 'fpr', 'tissue', 'prep', 'individual', 'nGenes', 'nUMIs',
'PercentMito', 'PercentRibo', 'Age_bin', 'Sex', 'Sample ID', 'Participant ID',
'Sample ID short', 'RIN score from PAXgene tissue Aliquot', 'RIN score from
Frozen tissue Aliquot', 'Autolysis Score', 'Sample Ischemic Time (mins)',
'Tissue Site Detail', 'scrublet', 'scrublet_score', 'barcode', 'batch',
'n_counts', 'tissue-individual-prep', 'Broad cell type', 'Granular cell type',
'introns', 'junctions', 'exons', 'sense', 'antisense', 'intergenic', 'batch-
barcode', 'exon_ratio', 'intron_ratio', 'junction_ratio', 'log10_nUMIs',
'leiden', 'leiden_tissue', 'Tissue composition', 'Cell types level 2', 'Cell
types level 3', 'Broad cell type numbers', 'Broad cell type (numbers)',
'Tissue', 'channel'
    var: 'gene_ids', 'Chromosome', 'Source', 'Start', 'End', 'Strand',
'gene_name', 'gene_source', 'gene_biotype', 'gene_length', 'gene_coding_length',
'Approved symbol', 'Approved name', 'Status', 'Previous symbols', 'Alias
symbols', 'gene_include', 'n_cells'
    uns: 'Broad cell type (numbers)_colors', 'Broad cell type numbers_colors',
'Broad cell type_colors', 'Broad cell type_logregcv_vae_colors', 'Broad cell
type_sizes', 'Granular cell type_colors', 'Participant ID_colors', 'Sex_colors',
'Tissue composition_colors', 'Tissue_colors', "dendrogram_['Broad cell type']",
'leiden', 'leiden_colors', 'leiden_sub_colors', 'neighbors', 'paga',
'prep_colors', 'tissue_colors', 'umap'
    obsm: 'X_pca', 'X_umap', 'X_umap_tissue', 'X_vae_mean', 'X_vae_mean_tissue',
'X_vae_samples', 'X_vae_var'
    varm: 'spring_leiden_sub'
    layers: 'counts'
    obsp: 'connectivities', 'distances'
```

[20]:
```
sc.pl.umap(adatac, color=["tissue", "Granular cell type"],)

sc.pl.umap(adatac, color=["Cell types level 2", "Cell types level 3"],)

sc.pl.umap(adatac, color="Broad cell type",)
```



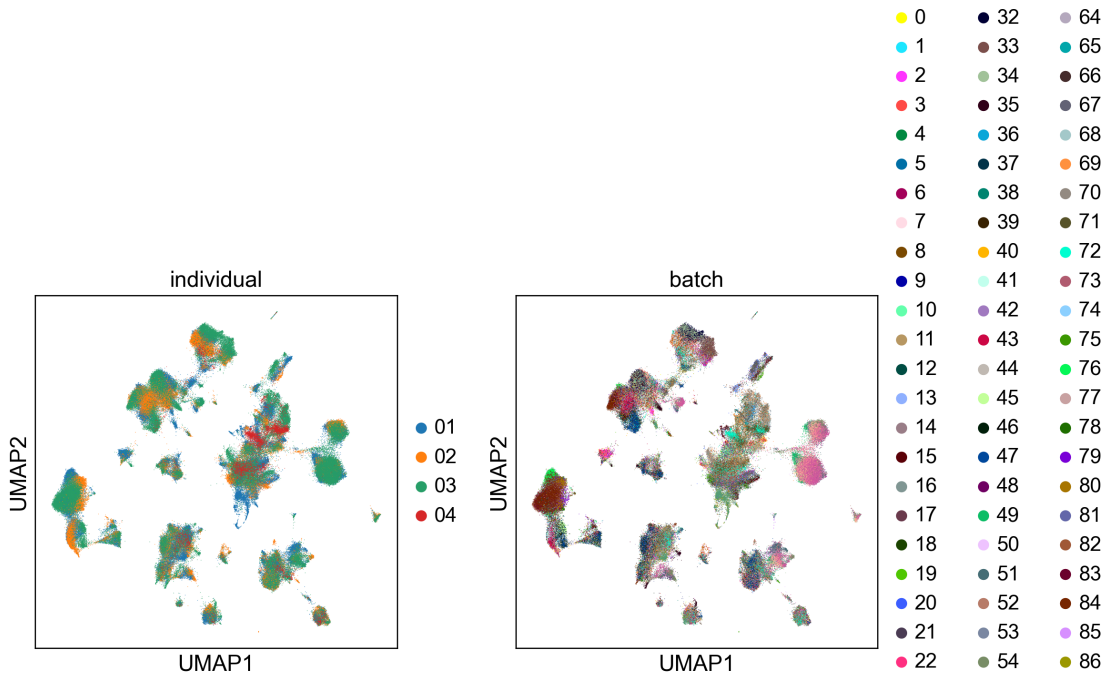



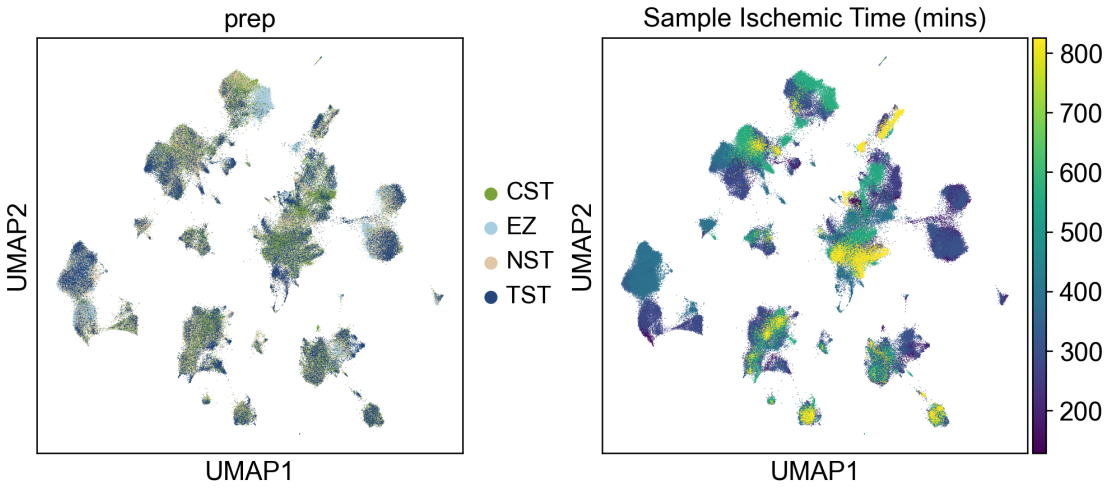




**You can double click on the plots to zoom in**

The plots are all umap projection of the latent space provided with the dataset. So it looks like their clusters mostly represent cell types. Now lets see some batch effects As the plots below show there are still some batch effects remaining.

[21]:
```
sc.pl.umap(adatac, color=["individual", "batch"],)

sc.pl.umap(adatac, color=["prep", "Sample Ischemic Time (mins)"],)
```



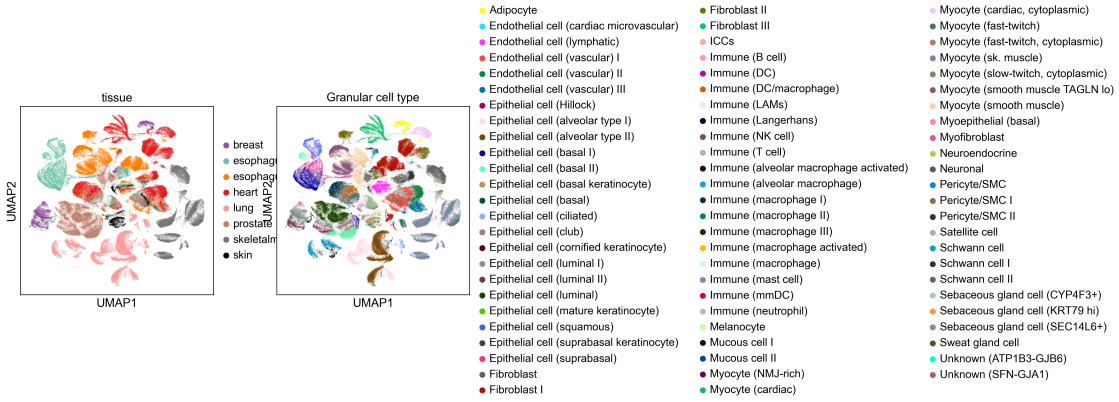
## 0.1 Comparing the umap from the paper's CVAE to a umap over PCA

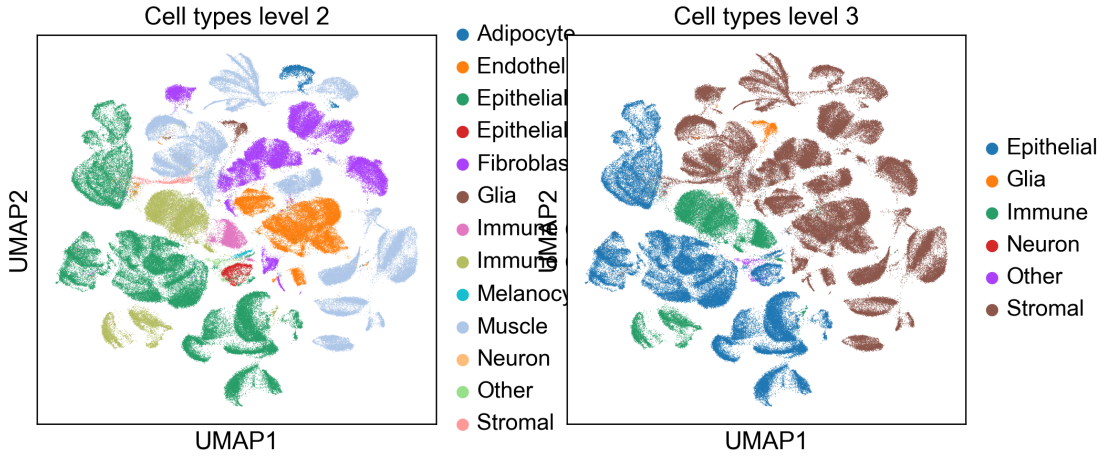
Just to see if simply doing umap over the PC space doesn't given the same or better results.

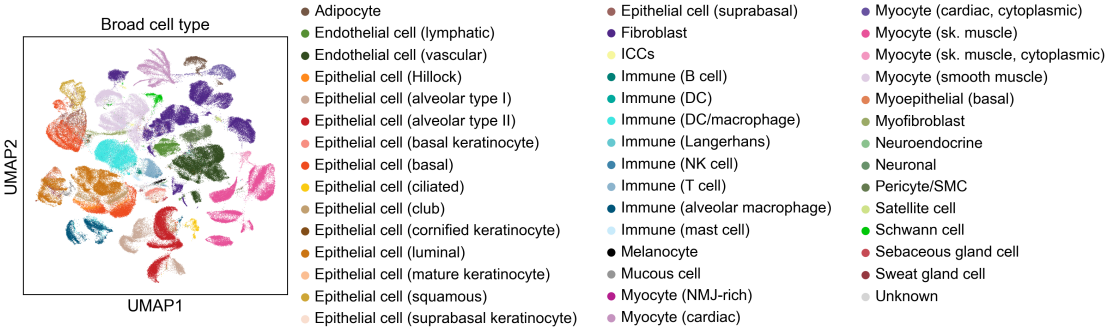
[45]:
```
sc.pp.neighbors(adata, use_rep="X_pca", n_neighbors=10,)
sc.tl.umap(adata,)
```

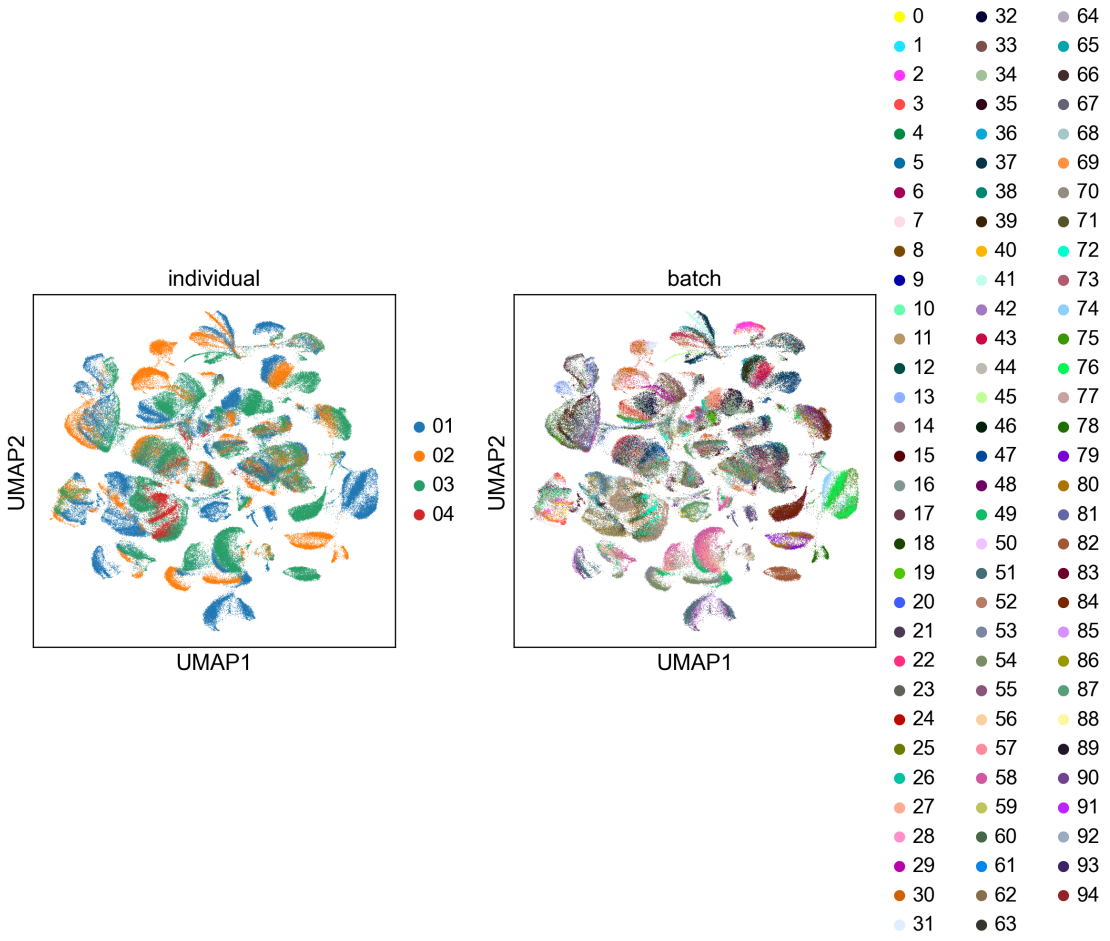
```
computing neighbors
    finished: added to `.uns['neighbors']`
    `.obsp['distances']`, distances for each pair of neighbors
    `.obsp['connectivities']`, weighted adjacency matrix (0:00:11)
computing UMAP
    finished: added
    'X_umap', UMAP coordinates (adata.obsm) (0:02:22)
```

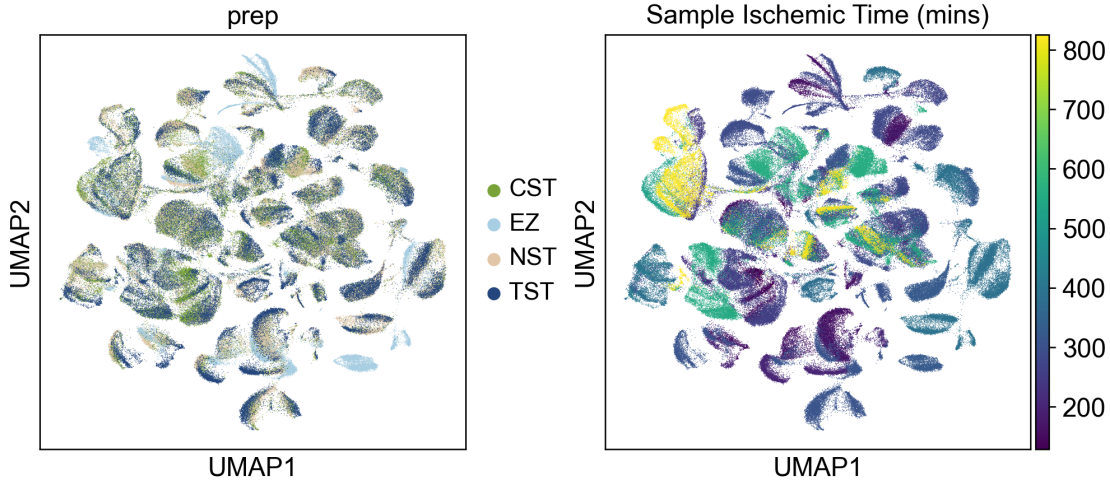
[47]:
```
sc.pl.umap(adata, color=["tissue", "Granular cell type"],)
sc.pl.umap(adata, color=["Cell types level 2", "Cell types level 3"],)
sc.pl.umap(adata, color="Broad cell type",)

sc.pl.umap(adata, color=["individual", "batch"],)
sc.pl.umap(adata, color=["prep", "Sample Ischemic Time (mins)"],)
```




Cell types level 2
UMAP2
UMAP1
Adipocyte
Endothel
Epithelial
Epithelial
Fibroblas
Glia
Immune
Immune
Melanocy
Muscle
Neuron
Other
Stromal
Cell types level 3
UMAP2
UMAP1
Epithelial
Glia
Immune
Neuron
Other
Stromal


Broad cell type
UMAP2
UMAP1
Adipocyte
Endothelial cell (lymphatic)
Endothelial cell (vascular)
Epithelial cell (Hillock)
Epithelial cell (alveolar type I)
Epithelial cell (alveolar type II)
Epithelial cell (basal keratinocyte)
Epithelial cell (basal)
Epithelial cell (ciliated)
Epithelial cell (club)
Epithelial cell (cornified keratinocyte)
Epithelial cell (luminal)
Epithelial cell (mature keratinocyte)
Epithelial cell (squamous)
Epithelial cell (suprabasal keratinocyte)
Epithelial cell (suprabasal)
Fibroblast
ICCs
Immune (B cell)
Immune (DC)
Immune (DC/macrophage)
Immune (Langerhans)
Immune (NK cell)
Immune (T cell)
Immune (alveolar macrophage)
Immune (mast cell)
Melanocyte
Mucous cell
Myocyte (NMJ-rich)
Myocyte (cardiac)
Myocyte (cardiac, cytoplasmic)
Myocyte (sk. muscle)
Myocyte (sk. muscle, cytoplasmic)
Myocyte (smooth muscle)
Myoepithelial (basal)
Myofibroblast
Neuroendocrine
Neuronal
Pericyte/SMC
Satellite cell
Schwann cell
Sebaceous gland cell
Sweat gland cell
Unknown


individual
UMAP2
UMAP1
01
02
03
04
batch
UMAP2
UMAP1
0 32 64
1 33 65
2 34 66
3 35 67
4 36 68
5 37 69
6 38 70
7 39 71
8 40 72
9 41 73
10 42 74
11 43 75
12 44 76
13 45 77
14 46 78
15 47 79
16 48 80
17 49 81
18 50 82
19 51 83
20 52 84
21 53 85
22 54 86
23 55 87
24 56 88
25 57 89
26 58 90
27 59 91
28 60 92
29 61 93
30 62 94
31 63


prep
UMAP2
UMAP1
CST
EZ
NST
TST
Sample Ischemic Time (mins)
UMAP2
UMAP1
800
700
600
500
400
300
200

## 0.2 Autoencoder

Now we will train an autoencoder (not variational) and check its usefulness as at dimensionality reduction.

But first we'll process the data itself, filtering out genes and keeping only the 1000 most influential genes. Otherwise we keep the same, already normalized dataset as the input.

We will use 64 dimensions for the lattent space since that it also what was used in the paper.

[11]:
```
sc.pp.highly_variable_genes(adata, n_top_genes=1000, inplace=True, subset=True,)


data = torch.FloatTensor(adata.X.toarray())
enc_ct.fit(adata.obs["Broad cell type"])
labels = torch.IntTensor(
        enc_ct.transform(adata.obs["Broad cell type"]))
labels = F.one_hot(labels.long(), num_classes=44).float()
dataset = ut.SynteticDataSet(data, labels)
data_loader = torch.utils.data.DataLoader(
        dataset=dataset,
        batch_size=256,
        shuffle=True,
        )
```

```
If you pass `n_top_genes`, all cutoffs are ignored.
extracting highly variable genes
    finished (0:00:01)
--> added
    'highly_variable', boolean vector (adata.var)
    'means', float vector (adata.var)
    'dispersions', float vector (adata.var)
    'dispersions_norm', float vector (adata.var)
```

[13]:
```
model_ae = M7.AE_Primer_Type701(nx=1000, nh=1024, nz=64, )
model_ae.apply(init_weights)
M6.basicTrain(model_ae, data_loader, num_epochs=14, wt=0.0, )
```

```
training phase
rec: 0.7703759074211121
total_loss: 0.7703759074211121

training phase
rec: 0.13002024590969086
total_loss: 0.13002024590969086

training phase
rec: 0.1179540678858757
total_loss: 0.1179540678858757
```

```
training phase
rec: 0.11943306773900986
total_loss: 0.11943306773900986

training phase
rec: 0.10966562479734421
total_loss: 0.10966562479734421

training phase
rec: 0.10757967084646225
total_loss: 0.10757967084646225

training phase
rec: 0.10260479897260666
total_loss: 0.10260479897260666

training phase
rec: 0.10604943335056305
total_loss: 0.10604943335056305

training phase
rec: 0.10324528068304062
total_loss: 0.10324528068304062

training phase
rec: 0.10340182483196259
total_loss: 0.10340182483196259

training phase
rec: 0.10268589109182358
total_loss: 0.10268589109182358

training phase
rec: 0.09674365073442459
total_loss: 0.09674365073442459

training phase
rec: 0.09509523212909698
total_loss: 0.09509523212909698

training phase
rec: 0.09676181524991989
total_loss: 0.09676181524991989
```

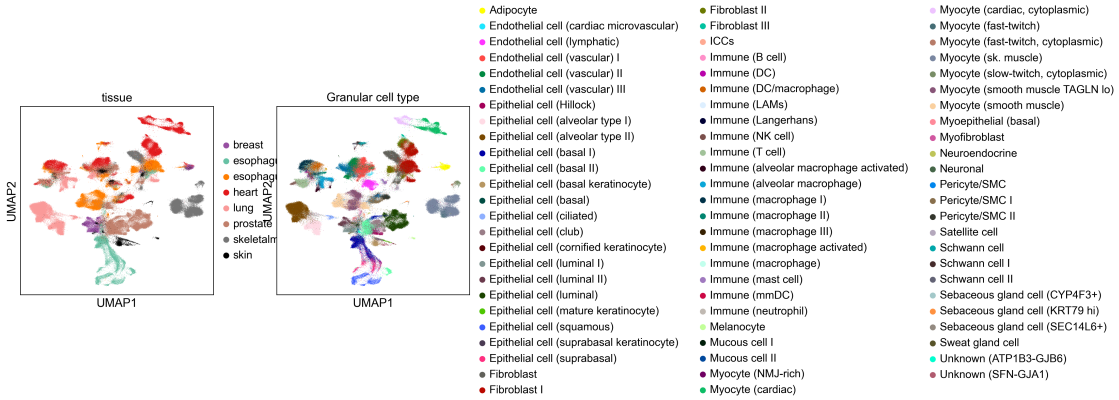
## 0.3 Umapping the AE

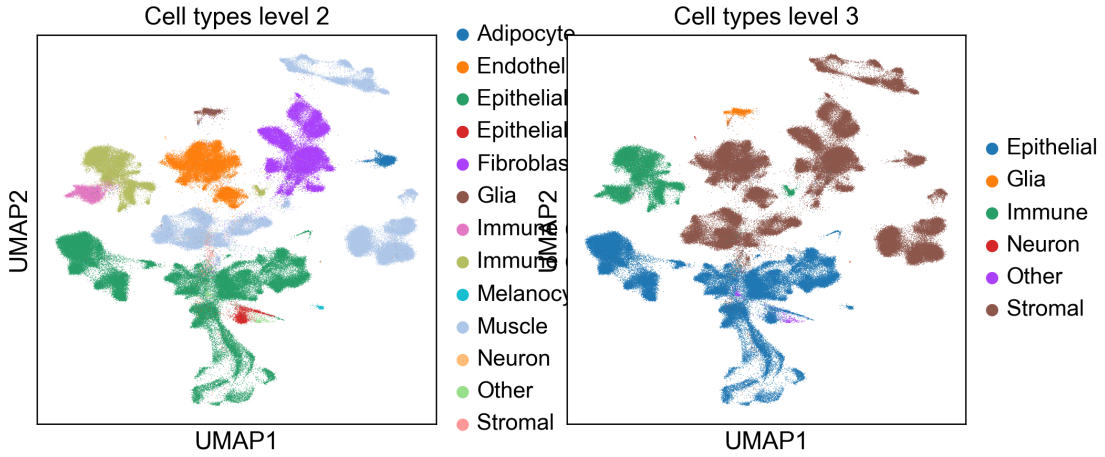
we trained the autoencoder. Now we'll use the encoder to project the data into the latent space and umap over that space.

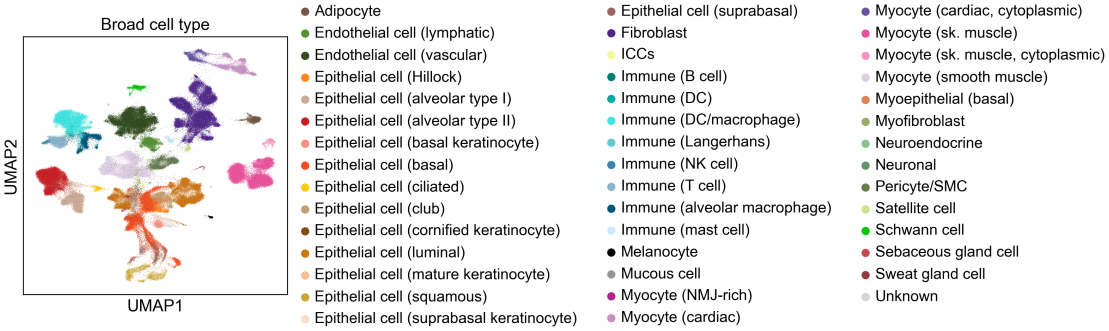
[14]:
```
model_ae.eval()
output = model_ae(data)
adata.obsm["z_ae"] = output["z"].detach().numpy()
sc.pp.neighbors(adata, use_rep="z_ae", n_neighbors=10,)
sc.tl.umap(adata,)
```

```
computing neighbors
    finished: added to `.uns['neighbors']`
    `.obsp['distances']`, distances for each pair of neighbors
    `.obsp['connectivities']`, weighted adjacency matrix (0:00:21)
computing UMAP
    finished: added
    'X_umap', UMAP coordinates (adata.obsm) (0:02:11)
```

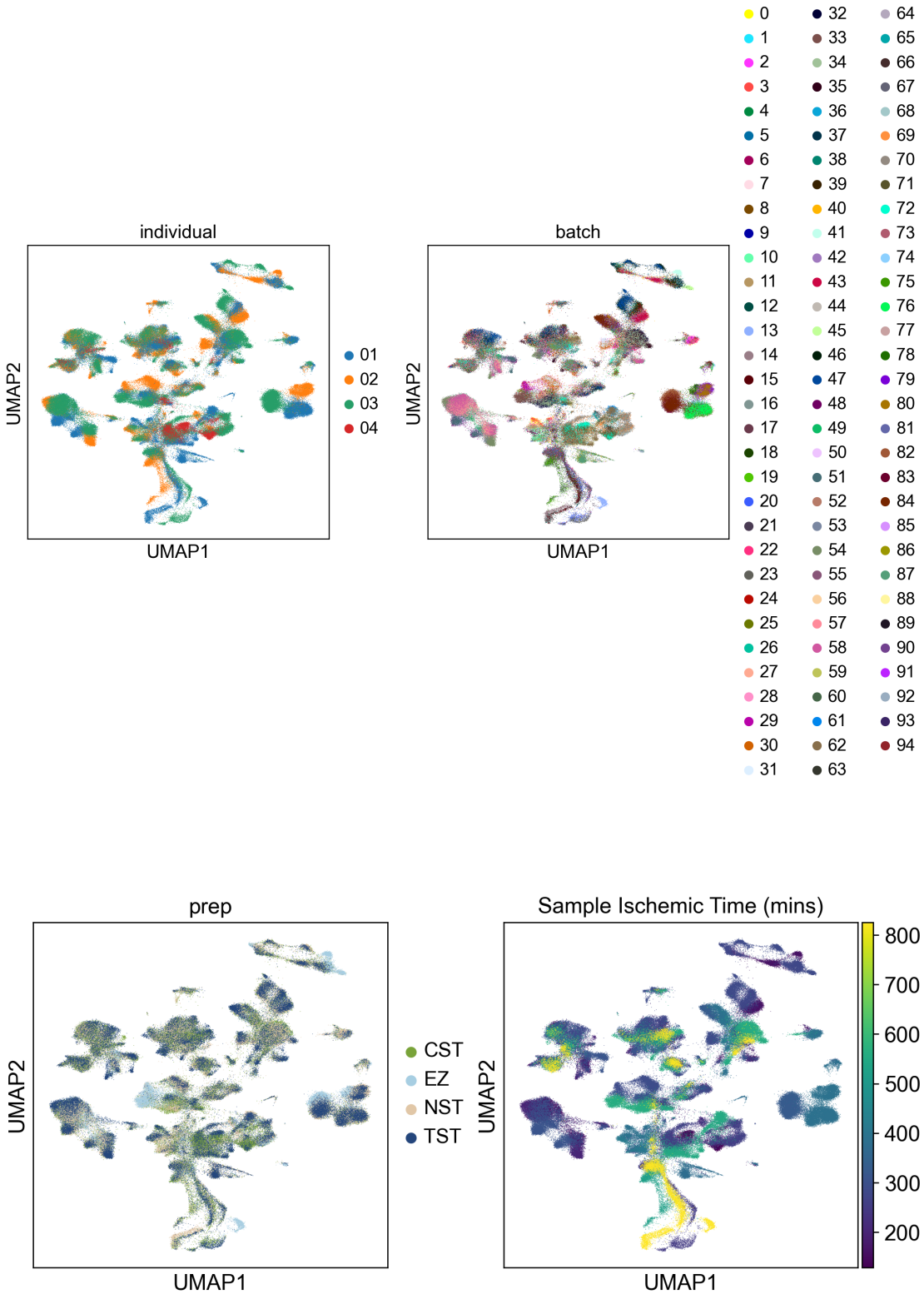
[24]:
```
sc.pl.umap(adata, color=["tissue", "Granular cell type"],)

sc.pl.umap(adata, color=["Cell types level 2", "Cell types level 3"],)

sc.pl.umap(adata, color="Broad cell type",)
```






[23]:
```
sc.pl.umap(adata, color=["individual", "batch"],)

sc.pl.umap(adata, color=["prep", "Sample Ischemic Time (mins)"],)
```


individual
UMAP2
UMAP1
01
02
03
04
batch
UMAP2
UMAP1
0 32 64
1 33 65
2 34 66
3 35 67
4 36 68
5 37 69
6 38 70
7 39 71
8 40 72
9 41 73
10 42 74
11 43 75
12 44 76
13 45 77
14 46 78
15 47 79
16 48 80
17 49 81
18 50 82
19 51 83
20 52 84
21 53 85
22 54 86
23 55 87
24 56 88
25 57 89
26 58 90
27 59 91
28 60 92
29 61 93
30 62 94
31 63
prep
UMAP2
UMAP1
CST
EZ
NST
TST
Sample Ischemic Time (mins)
UMAP2
UMAP1
800
700
600
500
400
300
200

### 0.4 remarks about AE results

A simple autoencoder seems to be able to embed the data in a similar way that the CVAE which was used in the original paper. cell types are clearly the main factor on the clusters of the umap.

### 0.5 moving to vanilla VAE

next we will test a "vanilla" variational autoencoder, meaning we don't try to represent the data as some complex mixture distribution. We only have one latent variable $z$ and it use a normal gaussian prior on it.

[26]:
```
model_vanillaAE = M7.VAE_Primer_Type700(nx=1000, nz=64, nh=1024,)
model_vanillaAE.apply(init_weights)
M6.basicTrain(model_vanillaAE, data_loader, num_epochs=14, wt=0.0, )
```

```
training phase
z: 56.25927734375
rec: 1310.408203125
total_loss: 1366.66748046875

training phase
z: 4.073150157928467
rec: 559.896728515625
total_loss: 563.9698486328125

training phase
z: 7.035953521728516
rec: 303.221923828125
total_loss: 310.25787353515625

training phase
z: 8.627479553222656
rec: 152.32887268066406
total_loss: 160.95635986328125

training phase
z: 9.208904266357422
rec: 112.22357177734375
total_loss: 121.43247985839844

training phase
z: 9.585808753967285
rec: 204.83682250976562
total_loss: 214.42263793945312

training phase
z: 9.45479679107666
rec: 62.55974578857422
total_loss: 72.01454162597656
```

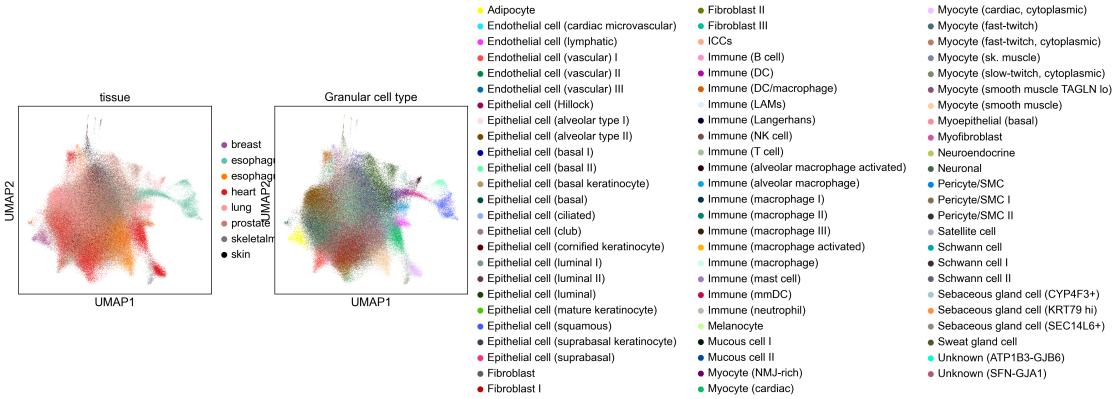
```
training phase
z: 10.448797225952148
rec: 59.41071319580078
total_loss: 69.85951232910156

training phase
z: 10.600569725036621
rec: 72.95214080810547
total_loss: 83.5527114868164

training phase
z: 9.90155029296875
rec: 105.63248443603516
total_loss: 115.5340347290039

training phase
z: 10.718955993652344
rec: 73.68013000488281
total_loss: 84.39908599853516

training phase
z: 11.728782653808594
rec: 74.11935424804688
total_loss: 85.84813690185547

training phase
z: 11.238344192504883
rec: 102.8114013671875
total_loss: 114.04974365234375

training phase
z: 11.08041763305664
rec: 5.163670539855957
total_loss: 16.24408721923828
```

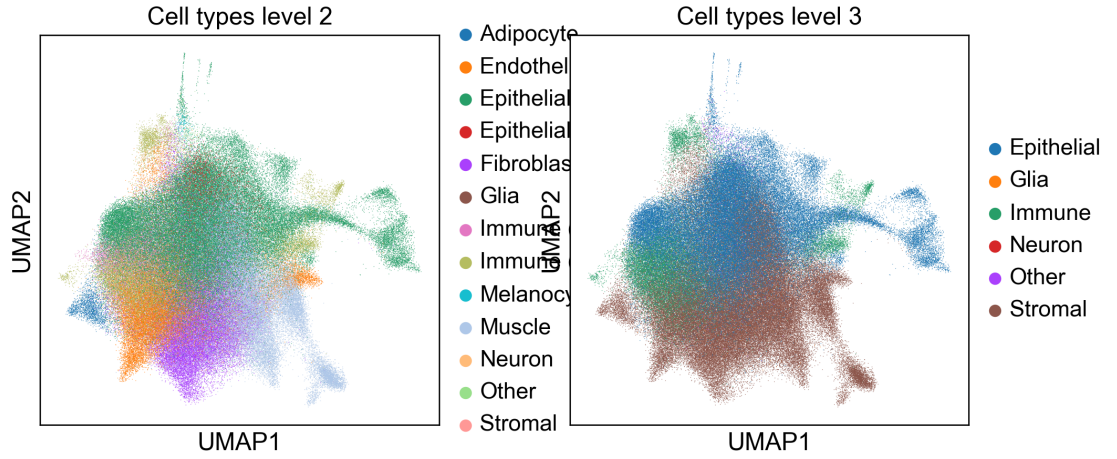
[28]:
```
model_vanillaAE.eval()
output = model_vanillaAE(data)
adata.obsm["z_vae"] = output["z"].detach().numpy()

sc.pp.neighbors(adata, use_rep="z_vae", n_neighbors=10,)
sc.tl.umap(adata,)
```

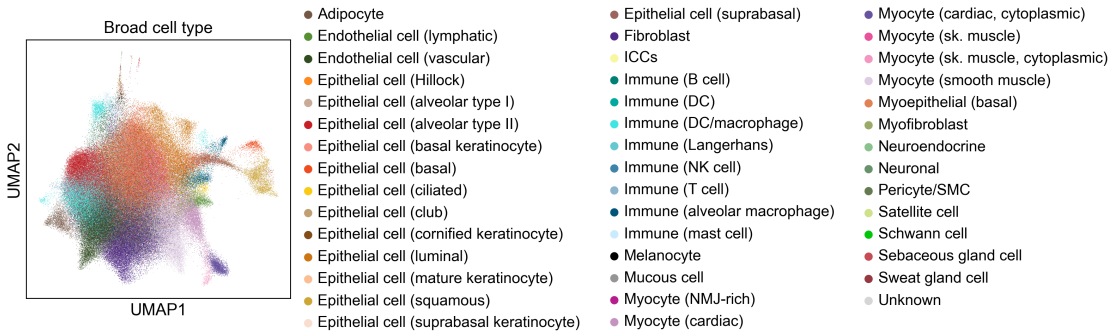
```
computing neighbors
    finished: added to `.uns['neighbors']`
    `.obsp['distances']`, distances for each pair of neighbors
    `.obsp['connectivities']`, weighted adjacency matrix (0:00:17)
```

```
computing UMAP
    finished: added
    'X_umap', UMAP coordinates (adata.obsm) (0:02:15)
```

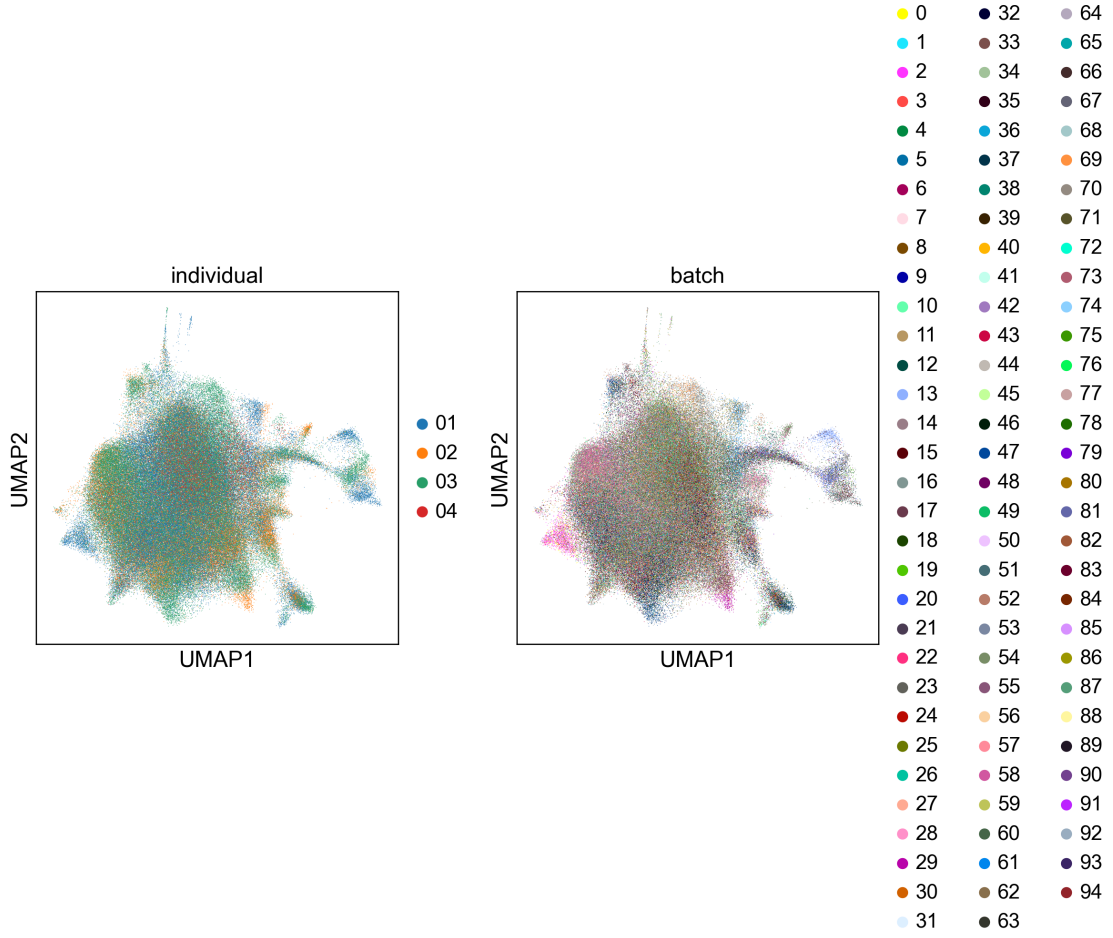
[29]:
```
sc.pl.umap(adata, color=["tissue", "Granular cell type"],)
sc.pl.umap(adata, color=["Cell types level 2", "Cell types level 3"],)
sc.pl.umap(adata, color="Broad cell type",)
```

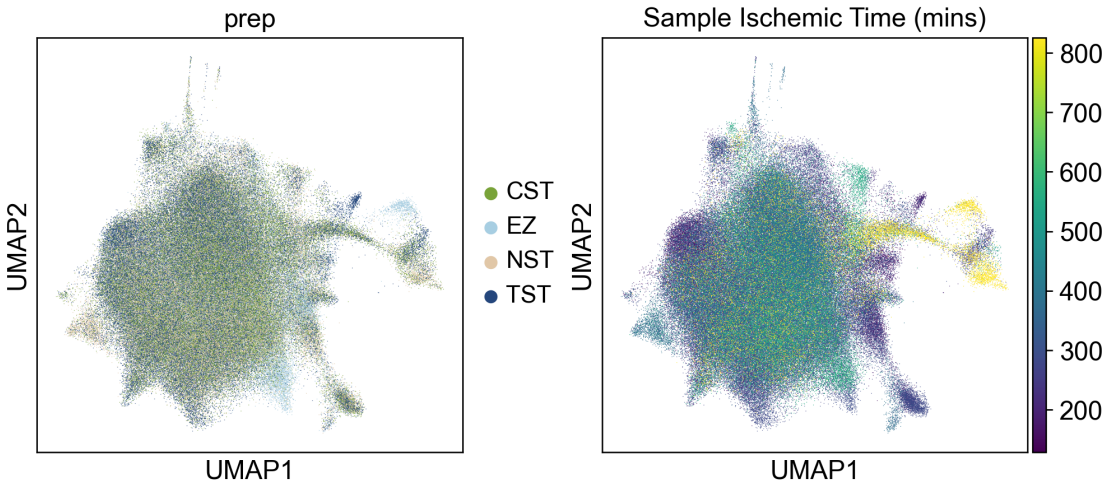





[30]:
```
sc.pl.umap(adata, color=["individual", "batch"],)
sc.pl.umap(adata, color=["prep", "Sample Ischemic Time (mins)"],)
```






## 0.6 Remarks about the VAE results

Basically the added stochasitty to the latent space made it into one big blob. The embedding still shows the spatial organisation of the data but if we ran a clustering algorithm on it. We use same Encoder/Decoder for the AE and VAE. So in conclusion, if dimensionality reduction is our goal, AE is usable and vanilla VAE isn't. VAE is maybe more usable as a generative model.

## 0.7 Now for the Gaussian Mixture VAE model

[31]:
```
model_gmmDVAE = M7.VAE_Dirichlet_Type705(nx=1000, nh=1024, nclasses=13, nz=64,␣
↪nw=15,)
model_gmmDVAE.apply(init_weights)
M6.basicTrain(model_gmmDVAE, data_loader, num_epochs=24, wt=0.0, )
```

```
training phase
rec: 1311.4144287109375
loss_p: 1.00538969039917
loss_z: 54.48384475708008
loss_w: 7.187105655670166
loss_l: 0.1405077576637268
loss_l_alt: 0.4193398952484131
loss_y: -0.27943068742752075
loss_d: 366.1248474121094
loss_y_alt: 1.6799585819244385
total_loss: 1740.8902587890625

training phase
rec: 483.2533874511719
loss_p: 13.945816040039062
loss_z: 5.652104377746582
```

```
loss_w: 2.7330257892608643
loss_l: 0.21472811698913574
loss_l_alt: 2.4989376068115234
loss_y: -0.973413348197937
loss_d: 0.0
loss_y_alt: 13.954390525817871
total_loss: 505.5928955078125

training phase
rec: 207.2654266357422
loss_p: 14.200817108154297
loss_z: 8.135747909545898
loss_w: 3.858743190765381
loss_l: 0.2015954852104187
loss_l_alt: 2.537320852279663
loss_y: -0.9885655641555786
loss_d: 0.0
loss_y_alt: 13.789939880371094
total_loss: 233.04986572265625

training phase
rec: 112.80696868896484
loss_p: 14.607566833496094
loss_z: 8.902649879455566
loss_w: 3.436638355255127
loss_l: 0.17409798502922058
loss_l_alt: 2.541393518447876
loss_y: -0.9911403059959412
loss_d: 0.0
loss_y_alt: 14.094375610351562
total_loss: 139.24063110351562

training phase
rec: 61.86579895019531
loss_p: 14.775175094604492
loss_z: 9.52446174621582
loss_w: 3.491978168487549
loss_l: 0.3042450249195099
loss_l_alt: 2.555447578430176
loss_y: -0.9968175888061523
loss_d: 0.0
loss_y_alt: 13.809160232543945
total_loss: 88.69139862060547

training phase
rec: 99.80829620361328
loss_p: 14.381632804870605
loss_z: 11.78453540802002
```

```
loss_w: 3.6046833992004395
loss_l: 0.2158646285533905
loss_l_alt: 2.543142318725586
loss_y: -0.9910159111022949
loss_d: 0.0
loss_y_alt: 13.970308303833008
total_loss: 129.16783142089844

training phase
rec: 33.573272705078125
loss_p: 15.026713371276855
loss_z: 11.59617805480957
loss_w: 3.502880573272705
loss_l: 0.2056000977754593
loss_l_alt: 2.5415985584259033
loss_y: -0.9921243190765381
loss_d: 0.0
loss_y_alt: 14.32591438293457
total_loss: 62.99824523925781

training phase
rec: 45.05812072753906
loss_p: 14.677817344665527
loss_z: 13.029695510864258
loss_w: 3.422605276107788
loss_l: 0.1721319705247879
loss_l_alt: 2.553053379058838
loss_y: -0.9960837364196777
loss_d: 0.0
loss_y_alt: 13.819767951965332
total_loss: 75.33018493652344

training phase
rec: -9.162028312683105
loss_p: 14.714296340942383
loss_z: 13.016968727111816
loss_w: 3.449716806411743
loss_l: 0.24594298005104065
loss_l_alt: 2.5577569007873535
loss_y: -0.9973887801170349
loss_d: 0.0
loss_y_alt: 13.986791610717773
total_loss: 21.29144859313965

training phase
rec: 29.975406646728516
loss_p: 14.78966999053955
loss_z: 13.681358337402344
```

```
loss_w: 3.276146411895752
loss_l: 0.2701225280761719
loss_l_alt: 2.5550103187561035
loss_y: -0.9967247247695923
loss_d: 0.0
loss_y_alt: 13.904905319213867
total_loss: 60.83781433105469

training phase
rec: 10.700956344604492
loss_p: 14.819862365722656
loss_z: 14.380273818969727
loss_w: 3.784928560256958
loss_l: 0.15831485390663147
loss_l_alt: 2.5461883544921875
loss_y: -0.991989016532898
loss_d: 0.0
loss_y_alt: 14.0347261428833
total_loss: 42.90088653564453

training phase
rec: 15.104032516479492
loss_p: 14.579402923583984
loss_z: 15.086021423339844
loss_w: 3.4652106761932373
loss_l: 0.14218980073928833
loss_l_alt: 2.5465102195739746
loss_y: -0.9930610656738281
loss_d: 0.0
loss_y_alt: 13.924365043640137
total_loss: 47.57963180541992

training phase
rec: -4.69301176071167
loss_p: 14.743656158447266
loss_z: 14.702139854431152
loss_w: 3.6073246002197266
loss_l: 0.17579230666160583
loss_l_alt: 2.549154758453369
loss_y: -0.9935017824172974
loss_d: 0.0
loss_y_alt: 13.773015975952148
total_loss: 27.389469146728516

training phase
rec: 42.17802429199219
loss_p: 14.658562660217285
loss_z: 15.909002304077148
```

```
loss_w: 3.625305652618408
loss_l: 0.21663911640644073
loss_l_alt: 2.5425286293029785
loss_y: -0.9896283745765686
loss_d: 0.0
loss_y_alt: 14.096395492553711
total_loss: 75.80873107910156

training phase
rec: -27.650562286376953
loss_p: 14.69140625
loss_z: 15.37512493133545
loss_w: 3.349076747894287
loss_l: 0.19382859766483307
loss_l_alt: 2.534548759460449
loss_y: -0.9866180419921875
loss_d: 0.0
loss_y_alt: 14.033623695373535
total_loss: 5.10726261138916

training phase
rec: -28.46678924560547
loss_p: 14.335302352905273
loss_z: 15.326631546020508
loss_w: 3.593787670135498
loss_l: 0.1727144569158554
loss_l_alt: 2.5438594818115234
loss_y: -0.9904652237892151
loss_d: 0.0
loss_y_alt: 13.558402061462402
total_loss: 4.012032508850098

training phase
rec: 97.18933868408203
loss_p: 14.999214172363281
loss_z: 15.906989097595215
loss_w: 3.7607150077819824
loss_l: 0.1411764919757843
loss_l_alt: 2.5541558265686035
loss_y: -0.9960918426513672
loss_d: 0.0
loss_y_alt: 14.26645278930664
total_loss: 131.1234893798828

training phase
rec: -24.30217742919922
loss_p: 14.85399055480957
loss_z: 15.155091285705566
```

```
loss_w: 3.522205352783203
loss_l: 0.08934930711984634
loss_l_alt: 2.535708427429199
loss_y: -0.9885605573654175
loss_d: 0.0
loss_y_alt: 14.109649658203125
total_loss: 8.484768867492676

training phase
rec: 2.1663732528686523
loss_p: 14.337600708007812
loss_z: 15.760515213012695
loss_w: 3.3920235633850098
loss_l: 0.13672903180122375
loss_l_alt: 2.5393567085266113
loss_y: -0.9898489713668823
loss_d: 0.0
loss_y_alt: 13.811381340026855
total_loss: 35.13029098510742

training phase
rec: -25.719017028808594
loss_p: 14.644383430480957
loss_z: 15.63105583190918
loss_w: 3.4669947624206543
loss_l: 0.09280578792095184
loss_l_alt: 2.535597085952759
loss_y: -0.9866849184036255
loss_d: 0.0
loss_y_alt: 13.959331512451172
total_loss: 7.338365077972412

training phase
rec: -4.817584991455078
loss_p: 14.615598678588867
loss_z: 17.165224075317383
loss_w: 3.626394271850586
loss_l: 0.12815013527870178
loss_l_alt: 2.5364081859588623
loss_y: -0.9858012199401855
loss_d: 0.0
loss_y_alt: 13.906145095825195
total_loss: 29.880178451538086

training phase
rec: -72.42041015625
loss_p: 15.139942169189453
loss_z: 16.862667083740234
```

```
loss_w: 3.5884923934936523
loss_l: 0.09422846138477325
loss_l_alt: 2.542619228363037
loss_y: -0.9912471771240234
loss_d: 0.0
loss_y_alt: 14.50794792175293
total_loss: -37.4613037109375

training phase
rec: -47.511844635009766
loss_p: 14.873885154724121
loss_z: 16.811777114868164
loss_w: 3.6354660987854004
loss_l: 0.14786183834075928
loss_l_alt: 2.544149160385132
loss_y: -0.9929433465003967
loss_d: 0.0
loss_y_alt: 14.497673988342285
total_loss: -12.566927909851074

training phase
rec: -93.68936920166016
loss_p: 14.393282890319824
loss_z: 16.406763076782227
loss_w: 3.560265064239502
loss_l: 0.12042630463838577
loss_l_alt: 2.542518138885498
loss_y: -0.9901018142700195
loss_d: 0.0
loss_y_alt: 13.614341735839844
total_loss: -60.108001708984375
```

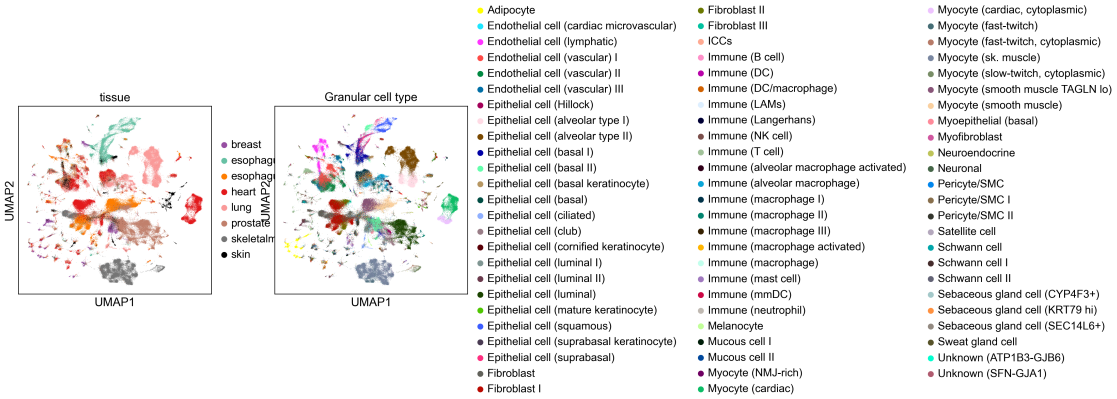
[32]:
```
model_gmmDVAE.eval()
output = model_gmmDVAE(data)
adata.obsm["z_gmmdvae"] = output["z"].detach().numpy()
adata.obs["predict_gmmdvae"] = output["q_y"].detach().argmax(-1).numpy().
 ↪astype(str)
```

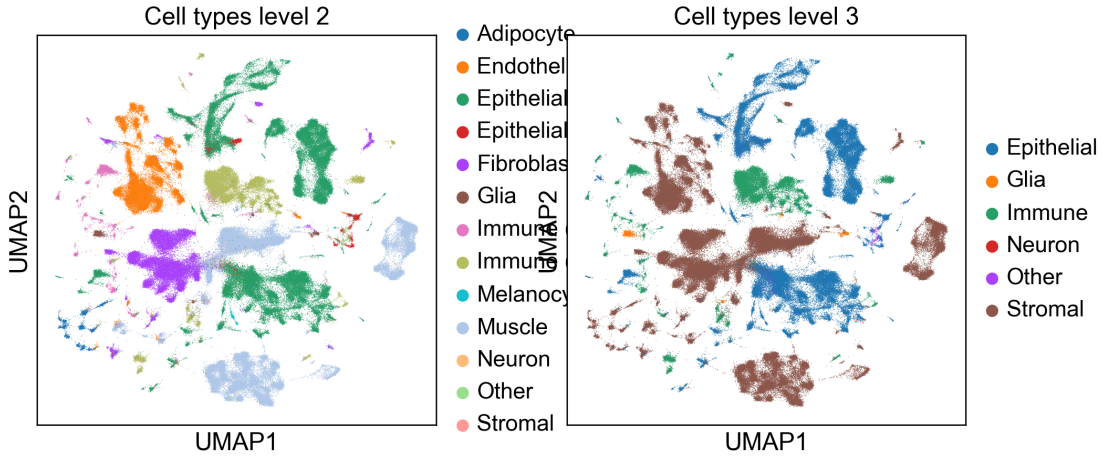
[33]:
```
sc.pp.neighbors(adata, use_rep="z_gmmdvae", n_neighbors=10,)
sc.tl.umap(adata,)
```

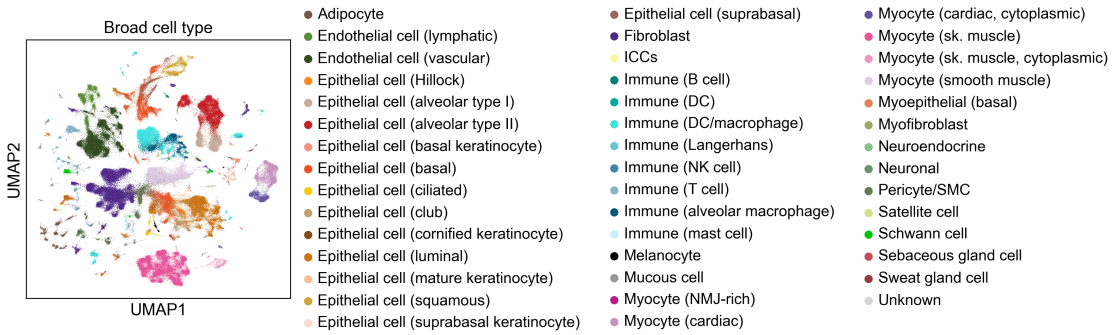
```
computing neighbors
    finished: added to `.uns['neighbors']`
    `.obsp['distances']`, distances for each pair of neighbors
    `.obsp['connectivities']`, weighted adjacency matrix (0:00:13)
computing UMAP
    finished: added
```

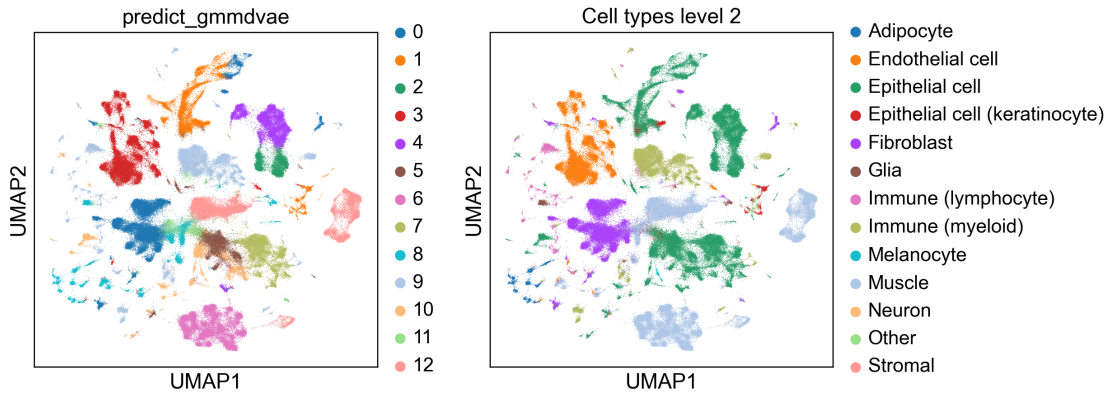
```
'X_umap', UMAP coordinates (adata.obsm) (0:02:09)
```

[39]:
```
sc.pl.umap(adata, color=["tissue", "Granular cell type"],)
sc.pl.umap(adata, color=["Cell types level 2", "Cell types level 3"],)
sc.pl.umap(adata, color="Broad cell type",)

sc.pl.umap(adata, color=["predict_gmmdvae","Cell types level 2"],)
```

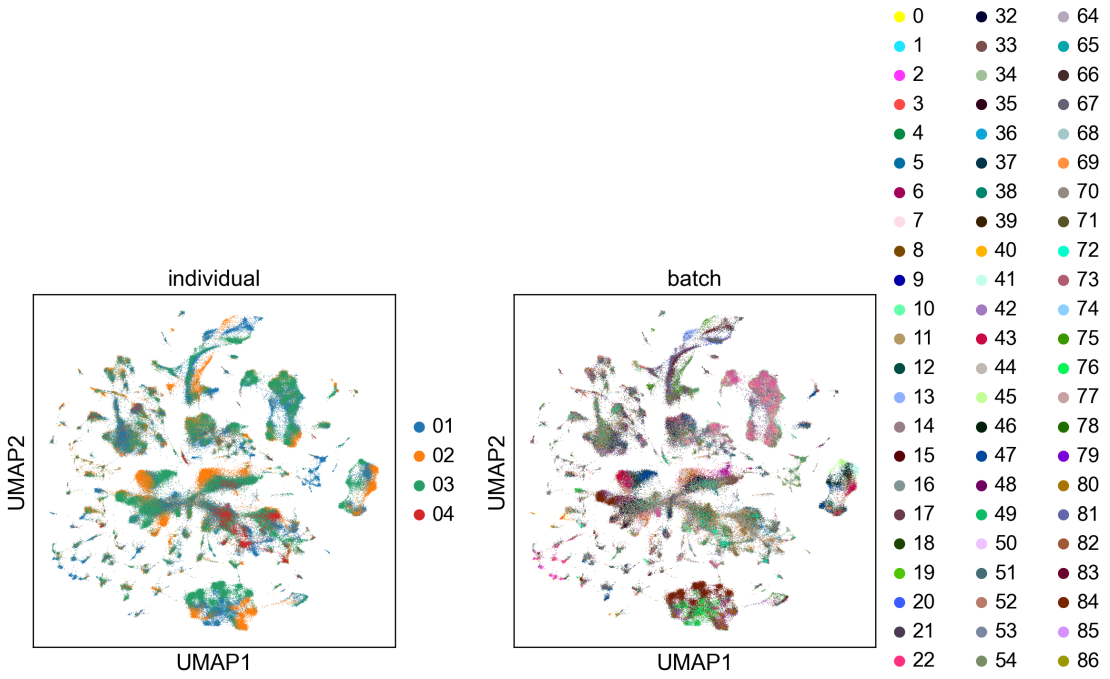


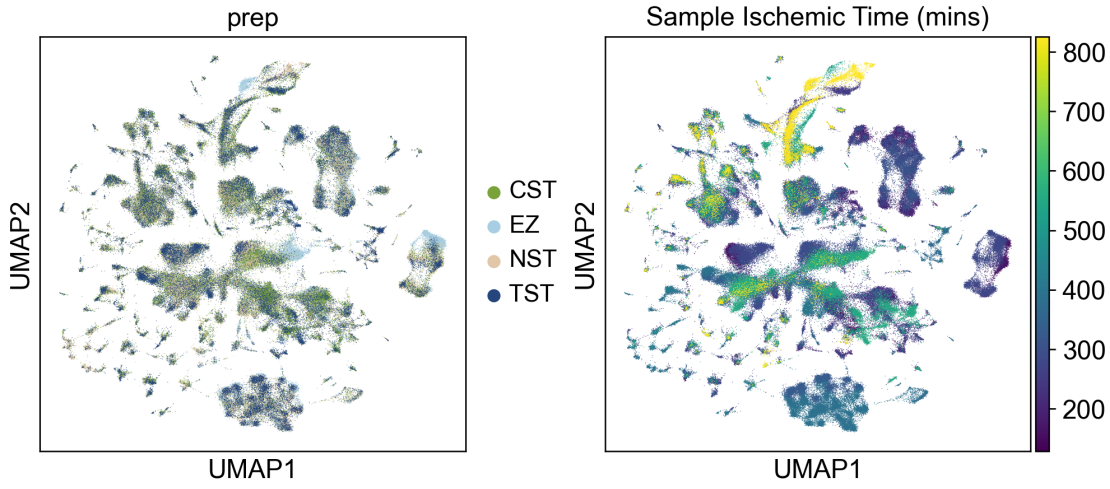






[38]:
```
sc.pl.umap(adata, color=["individual", "batch"],)
sc.pl.umap(adata, color=["prep", "Sample Ischemic Time (mins)"],)
```


individual
UMAP2
UMAP1
01
02
03
04
batch
UMAP2
UMAP1
0 32 64
1 33 65
2 34 66
3 35 67
4 36 68
5 37 69
6 38 70
7 39 71
8 40 72
9 41 73
10 42 74
11 43 75
12 44 76
13 45 77
14 46 78
15 47 79
16 48 80
17 49 81
18 50 82
19 51 83
20 52 84
21 53 85
22 54 86
23 55 87
24 56 88
25 57 89
26 58 90
27 59 91
28 60 92
29 61 93
30 62 94
31 63


prep
UMAP2
UMAP1
CST
EZ
NST
TST
Sample Ischemic Time (mins)
UMAP2
UMAP1
800
700
600
500
400
300
200

## 0.8 Remarks about the GMMVAE results

The encoder/decoder to/from $z$ space are the same in all the models.

So this VAE model represents the data in the lattent space in a much more clestered fashion when compared to the vanilla VAE. It was trained to find 13 components in the mixture, which is the same number as the categories in "Cell type level 2". Clearly the predicted classes aren't a perfect much with the ground truth cell type. Nevertheless the level 2 cell type categories are distributed in the (umap of the) z-space in a pretty defined manner.

So a more complex VAE model was able to improve the data representation, however if just clustering by cell type is of interest than perhaps the autoencoder was the best option.

## 0.9 training a GMMVAE with more classes

First we set the number of classes at 44 (number of categories in the "borad cell type")

[40]:
```
model_gmmDVAE2 = M7.VAE_Dirichlet_Type705(nx=1000, nh=1024, nclasses=44, nz=64,␣
↪nw=15,)
model_gmmDVAE2.apply(init_weights)
M6.basicTrain(model_gmmDVAE2, data_loader, num_epochs=24, wt=0.0, )
```

```
training phase
rec: 1313.241455078125
loss_p: 1.103355050086975
loss_z: 43.1353759765625
loss_w: 8.271791458129883
loss_l: 0.12249569594860077
loss_l_alt: 0.4265495836734772
loss_y: -0.12388856709003448
loss_d: 1328.00244140625
loss_y_alt: 1.7509922981262207
total_loss: 2694.402099609375

training phase
rec: 490.1571044921875
loss_p: 11.502349853515625
loss_z: 6.590069770812988
loss_w: 2.8546500205993652
loss_l: 0.5981159806251526
loss_l_alt: 2.431821346282959
loss_y: -0.6188969016075134
loss_d: 0.0
loss_y_alt: 14.663951873779297
total_loss: 514.2657470703125

training phase
rec: 230.88922119140625
loss_p: 12.960795402526855
loss_z: 8.343892097473145
```

```
loss_w: 3.4184889793395996
loss_l: 0.6304906010627747
loss_l_alt: 2.8306961059570312
loss_y: -0.7036824822425842
loss_d: 0.0
loss_y_alt: 14.494808197021484
total_loss: 257.1463928222656

training phase
rec: 112.41709899902344
loss_p: 14.056002616882324
loss_z: 10.005167961120605
loss_w: 2.9614062309265137
loss_l: 0.6014848947525024
loss_l_alt: 3.10961651802063
loss_y: -0.7687429785728455
loss_d: 0.0
loss_y_alt: 14.942798614501953
total_loss: 140.3264617919922

training phase
rec: 118.23377990722656
loss_p: 15.216028213500977
loss_z: 10.75645637512207
loss_w: 2.6117892265319824
loss_l: 0.6092725992202759
loss_l_alt: 3.3599934577941895
loss_y: -0.8491702079772949
loss_d: 0.0
loss_y_alt: 15.693169593811035
total_loss: 147.29519653320312

training phase
rec: 63.44042205810547
loss_p: 15.410517692565918
loss_z: 11.14309310913086
loss_w: 2.6755709648132324
loss_l: 0.6034603714942932
loss_l_alt: 3.506335735321045
loss_y: -0.898601770401001
loss_d: 0.0
loss_y_alt: 15.380701065063477
total_loss: 92.63977813720703

training phase
rec: 46.82801055908203
loss_p: 15.673166275024414
loss_z: 12.244026184082031
```

```
loss_w: 3.0355172157287598
loss_l: 0.49561774730682373
loss_l_alt: 3.531029224395752
loss_y: -0.9041576385498047
loss_d: 0.0
loss_y_alt: 15.512569427490234
total_loss: 77.62012481689453

training phase
rec: 32.96036148071289
loss_p: 16.134035110473633
loss_z: 11.934442520141602
loss_w: 2.6357598304748535
loss_l: 0.627595067024231
loss_l_alt: 3.646052837371826
loss_y: -0.9480481147766113
loss_d: 0.0
loss_y_alt: 15.727190017700195
total_loss: 63.25775909423828

training phase
rec: 1.6896238327026367
loss_p: 16.174400329589844
loss_z: 12.418055534362793
loss_w: 2.879746675491333
loss_l: 0.6042898893356323
loss_l_alt: 3.6550726890563965
loss_y: -0.9529292583465576
loss_d: 0.0
loss_y_alt: 16.162250518798828
total_loss: 33.14967727661133

training phase
rec: 31.326976776123047
loss_p: 15.944860458374023
loss_z: 13.588435173034668
loss_w: 2.8168275356292725
loss_l: 0.5562787055969238
loss_l_alt: 3.657280683517456
loss_y: -0.9469404220581055
loss_d: 0.0
loss_y_alt: 15.607471466064453
total_loss: 63.3397102355957

training phase
rec: 58.61461639404297
loss_p: 16.153209686279297
loss_z: 13.625999450683594
```

```
loss_w: 2.862710475921631
loss_l: 0.5821031332015991
loss_l_alt: 3.6745662689208984
loss_y: -0.9565501809120178
loss_d: 0.0
loss_y_alt: 16.167278289794922
total_loss: 91.27059936523438

training phase
rec: -8.928128242492676
loss_p: 15.838529586791992
loss_z: 13.565824508666992
loss_w: 2.82491135597229
loss_l: 0.6070975661277771
loss_l_alt: 3.672232151031494
loss_y: -0.9581151008605957
loss_d: 0.0
loss_y_alt: 15.646427154541016
total_loss: 23.10903549194336

training phase
rec: 63.57792663574219
loss_p: 16.387998580932617
loss_z: 13.992887496948242
loss_w: 2.848454475402832
loss_l: 0.5824233889579773
loss_l_alt: 3.6771507263183594
loss_y: -0.9602473974227905
loss_d: 0.0
loss_y_alt: 16.12023162841797
total_loss: 96.53950500488281

training phase
rec: 7.754005432128906
loss_p: 16.192852020263672
loss_z: 14.561111450195312
loss_w: 3.3583528995513916
loss_l: 0.597905158996582
loss_l_alt: 3.684846878051758
loss_y: -0.9611811637878418
loss_d: 0.0
loss_y_alt: 15.824174880981445
total_loss: 41.497642517089844

training phase
rec: -23.81523895263672
loss_p: 16.220197677612305
loss_z: 14.398149490356445
```

```
loss_w: 3.0792036056518555
loss_l: 0.5140367150306702
loss_l_alt: 3.6892752647399902
loss_y: -0.9605261087417603
loss_d: 0.0
loss_y_alt: 15.676870346069336
total_loss: 9.338984489440918

training phase
rec: -60.98521423339844
loss_p: 16.104345321655273
loss_z: 14.004192352294922
loss_w: 3.184235095977783
loss_l: 0.5309242606163025
loss_l_alt: 3.709595203399658
loss_y: -0.9687663316726685
loss_d: 0.0
loss_y_alt: 15.795394897460938
total_loss: -28.001392364501953

training phase
rec: 89.21495056152344
loss_p: 16.330841064453125
loss_z: 14.882196426391602
loss_w: 2.8553757667541504
loss_l: 0.5696861147880554
loss_l_alt: 3.707540988922119
loss_y: -0.9667956829071045
loss_d: 0.0
loss_y_alt: 15.760824203491211
total_loss: 122.71334838867188

training phase
rec: -53.80982971191406
loss_p: 16.643962860107422
loss_z: 14.771577835083008
loss_w: 3.0906996726989746
loss_l: 0.6137527823448181
loss_l_alt: 3.7000865936279297
loss_y: -0.965947151184082
loss_d: 0.0
loss_y_alt: 16.257884979248047
total_loss: -19.68967056274414

training phase
rec: -20.98184585571289
loss_p: 16.253910064697266
loss_z: 15.123857498168945
```

```
loss_w: 2.9639766216278076
loss_l: 0.5316281914710999
loss_l_alt: 3.6913704872131348
loss_y: -0.9627134799957275
loss_d: 0.0
loss_y_alt: 15.986374855041504
total_loss: 13.092363357543945

training phase
rec: -8.544828414916992
loss_p: 16.612577438354492
loss_z: 14.658380508422852
loss_w: 3.0762839317321777
loss_l: 0.5308897495269775
loss_l_alt: 3.6734619140625
loss_y: -0.9518274068832397
loss_d: 0.0
loss_y_alt: 15.814104080200195
total_loss: 25.00394058227539

training phase
rec: -31.930635452270508
loss_p: 16.320714950561523
loss_z: 15.460409164428711
loss_w: 2.8579649925231934
loss_l: 0.5122990012168884
loss_l_alt: 3.702547311782837
loss_y: -0.9678428769111633
loss_d: 0.0
loss_y_alt: 15.626590728759766
total_loss: 2.0143299102783203

training phase
rec: -74.20094299316406
loss_p: 16.25592041015625
loss_z: 15.480232238769531
loss_w: 3.2154064178466797
loss_l: 0.5370665788650513
loss_l_alt: 3.696959972381592
loss_y: -0.9662207365036011
loss_d: 0.0
loss_y_alt: 15.886983871459961
total_loss: -39.618316650390625

training phase
rec: -13.805767059326172
loss_p: 16.351839065551758
loss_z: 15.77919864654541
```

```
loss_w: 3.0518155097961426
loss_l: 0.5699034929275513
loss_l_alt: 3.6902055740356445
loss_y: -0.9628318548202515
loss_d: 0.0
loss_y_alt: 15.987045288085938
total_loss: 21.012292861938477

training phase
rec: -57.568809509277344
loss_p: 16.10009765625
loss_z: 16.453205108642578
loss_w: 3.4940147399902344
loss_l: 0.6062005758285522
loss_l_alt: 3.714160442352295
loss_y: -0.9731438159942627
loss_d: 0.0
loss_y_alt: 15.580289840698242
total_loss: -22.04129981994629
```

[41]:
```
model_gmmDVAE2.eval()
output = model_gmmDVAE2(data)
adata.obsm["z_gmmdvae2"] = output["z"].detach().numpy()
adata.obs["predict_gmmdvae2"] = output["q_y"].detach().argmax(-1).numpy().
  ↪astype(str)
```

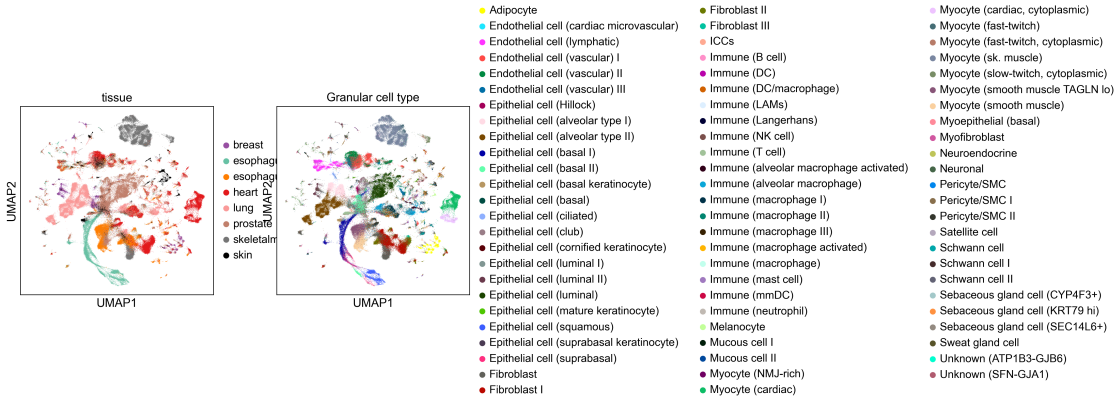
[42]:
```
sc.pp.neighbors(adata, use_rep="z_gmmdvae2", n_neighbors=10,)
sc.tl.umap(adata,)
```

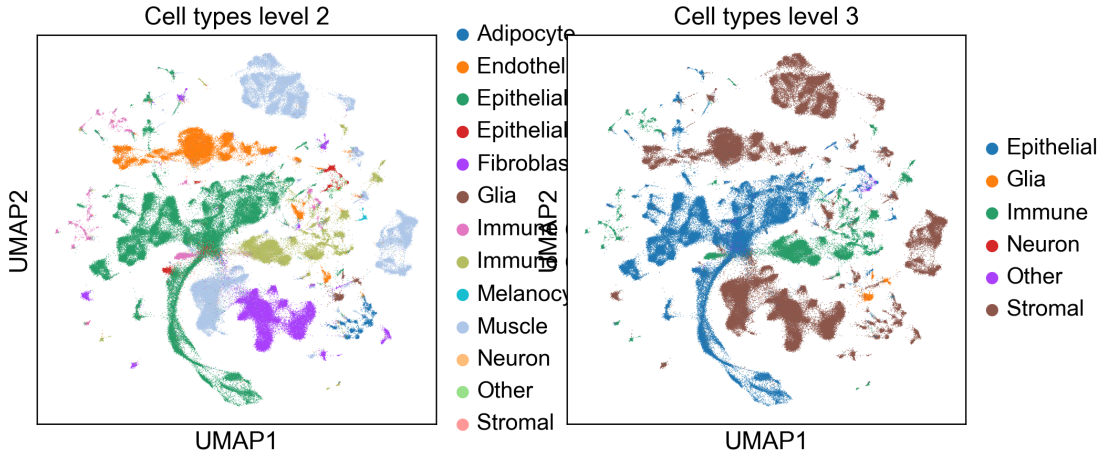
```
computing neighbors
    finished: added to `.uns['neighbors']`
    `.obsp['distances']`, distances for each pair of neighbors
    `.obsp['connectivities']`, weighted adjacency matrix (0:00:13)
computing UMAP
    finished: added
    'X_umap', UMAP coordinates (adata.obsm) (0:02:29)
```

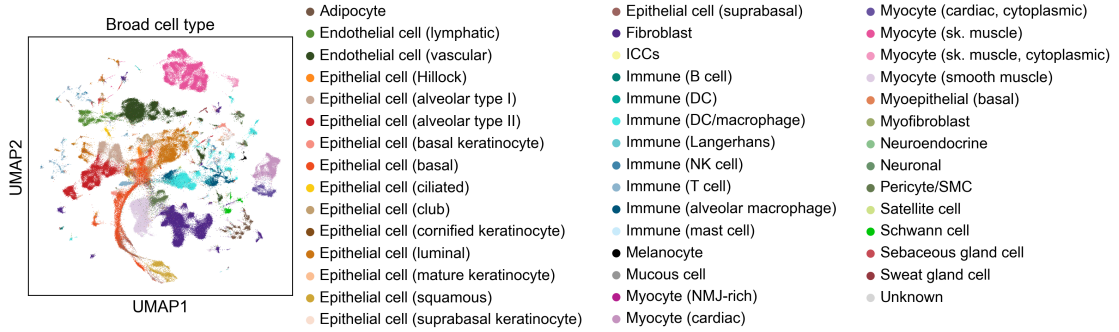
[43]:
```
sc.pl.umap(adata, color=["tissue", "Granular cell type"],)
sc.pl.umap(adata, color=["Cell types level 2", "Cell types level 3"],)
sc.pl.umap(adata, color="Broad cell type",)
sc.pl.umap(adata, color=["predict_gmmdvae2","Cell types level 2"],)
```

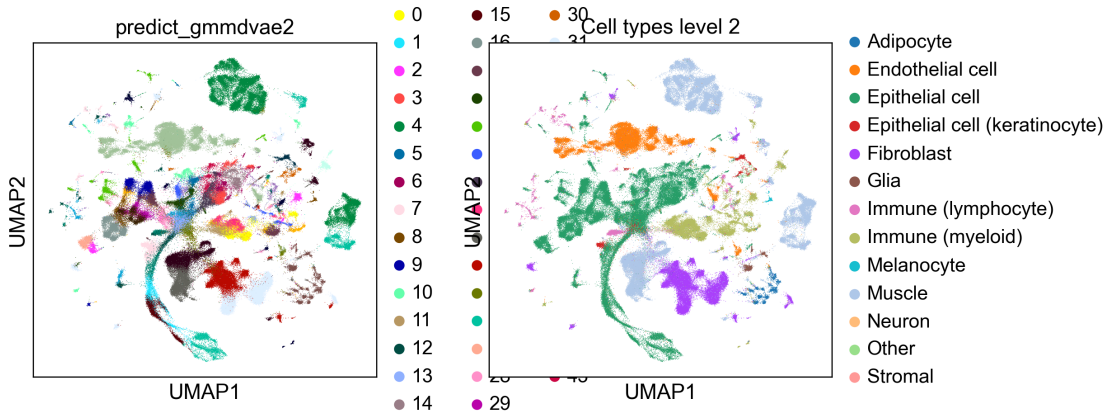
```
/home/ykolb/miniconda3/envs/torch/lib/python3.9/site-
packages/anndata/_core/anndata.py:1228: FutureWarning: The `inplace` parameter
in pandas.Categorical.reorder_categories is deprecated and will be removed in a
future version. Reordering categories will always return a new Categorical
object.
```

```
  c.reorder_categories(natsorted(c.categories), inplace=True)
… storing 'predict_gmmdvae2' as categorical
```

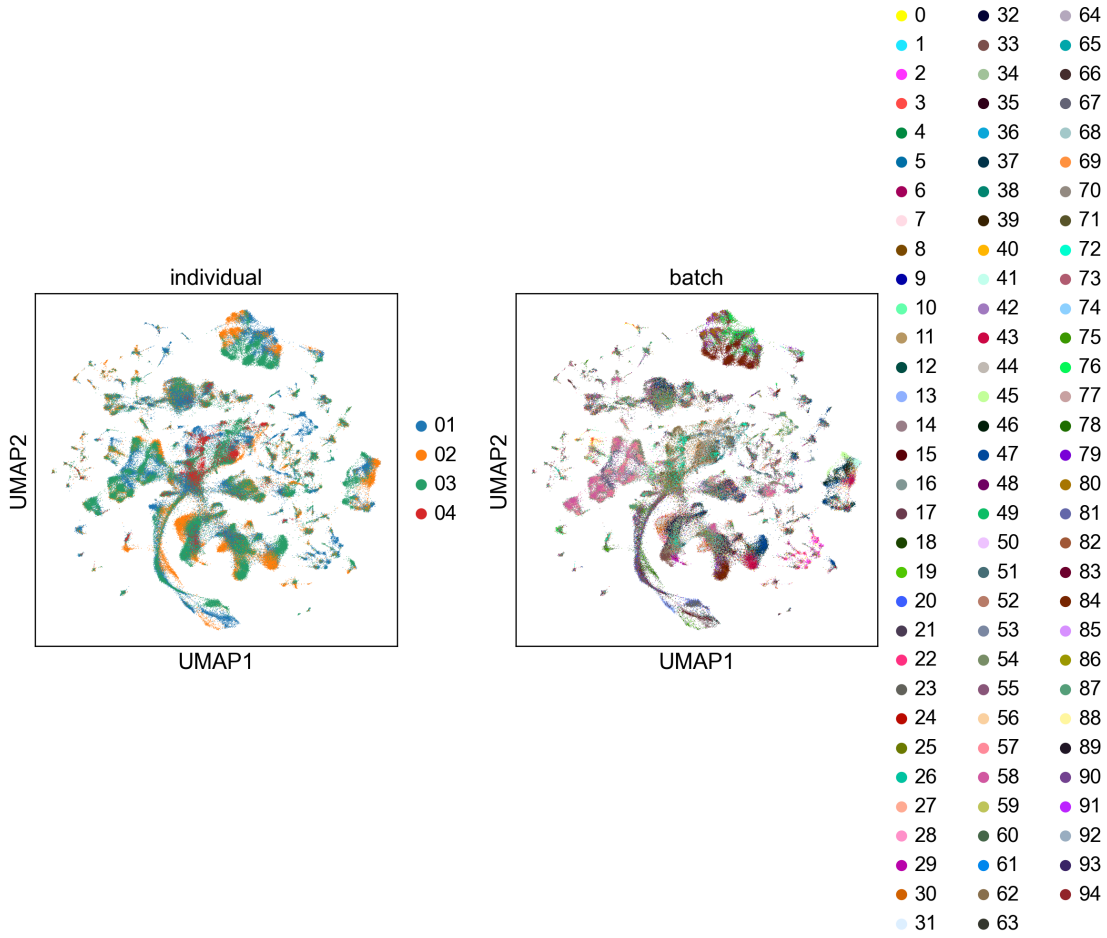


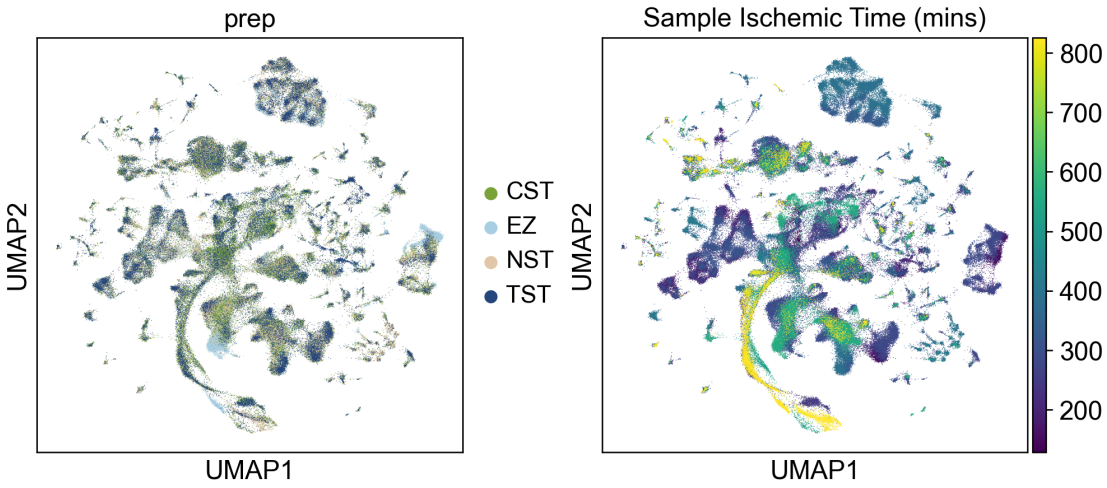






[44]:
```
sc.pl.umap(adata, color=["individual", "batch"],)
sc.pl.umap(adata, color=["prep", "Sample Ischemic Time (mins)"],)
```






Now try with 74 classes (the number of categories in the "Granular cell type")

[48]:
```
model_gmmDVAE3 = M7.VAE_Dirichlet_Type705(nx=1000, nh=1024, nclasses=74, nz=64,␣
↪nw=15,)
model_gmmDVAE3.apply(init_weights)
M6.basicTrain(model_gmmDVAE3, data_loader, num_epochs=24, wt=0.0, )
```

```
training phase
rec: 1310.1611328125
loss_p: 1.1232872009277344
loss_z: 40.98245620727539
loss_w: 7.209092140197754
loss_l: 0.07449870556592941
loss_l_alt: 0.3923454284667969
loss_y: -0.08165346831083298
loss_d: 2207.97705078125
loss_y_alt: 1.8463807106018066
total_loss: 3568.176025390625

training phase
rec: 477.2381591796875
loss_p: 11.461332321166992
loss_z: 6.172580718994141
loss_w: 2.827502489089966
loss_l: 0.5255260467529297
loss_l_alt: 2.7044620513916016
loss_y: -0.6008586883544922
loss_d: 0.0
loss_y_alt: 14.85645866394043
total_loss: 501.0946960449219
```

```
training phase
rec: 233.9641571044922
loss_p: 13.33985424041748
loss_z: 9.01465129852295
loss_w: 2.902534008026123
loss_l: 0.5356312394142151
loss_l_alt: 3.186002254486084
loss_y: -0.6886997222900391
loss_d: 0.0
loss_y_alt: 15.254629135131836
total_loss: 261.1359558105469

training phase
rec: 96.02761840820312
loss_p: 14.330652236938477
loss_z: 9.426209449768066
loss_w: 2.8801746368408203
loss_l: 0.5722915530204773
loss_l_alt: 3.4609246253967285
loss_y: -0.7545449137687683
loss_d: 0.0
loss_y_alt: 15.054637908935547
total_loss: 123.38864135742188

training phase
rec: 54.15977478027344
loss_p: 15.416055679321289
loss_z: 10.730406761169434
loss_w: 2.81062912940979
loss_l: 0.5435608625411987
loss_l_alt: 3.600112199783325
loss_y: -0.7891607880592346
loss_d: 0.0
loss_y_alt: 15.726799011230469
total_loss: 83.4276123046875

training phase
rec: 50.978511810302734
loss_p: 15.611857414245605
loss_z: 11.334901809692383
loss_w: 2.506096363067627
loss_l: 0.548898458480835
loss_l_alt: 3.6770033836364746
loss_y: -0.8052863478660583
loss_d: 0.0
loss_y_alt: 15.717177391052246
total_loss: 80.53668975830078
```

```
training phase
rec: 36.83389663696289
loss_p: 15.8553466796875
loss_z: 11.872323989868164
loss_w: 2.3544859886169434
loss_l: 0.5987735390663147
loss_l_alt: 3.8022594451904297
loss_y: -0.832818329334259
loss_d: 0.0
loss_y_alt: 15.68515396118164
total_loss: 66.74586486816406

training phase
rec: 19.113759994506836
loss_p: 15.808748245239258
loss_z: 12.376537322998047
loss_w: 2.4713540077209473
loss_l: 0.45308277010917664
loss_l_alt: 3.7877049446105957
loss_y: -0.8356813192367554
loss_d: 0.0
loss_y_alt: 15.84232234954834
total_loss: 49.80397415161133

training phase
rec: 14.711189270019531
loss_p: 15.86558723449707
loss_z: 12.662294387817383
loss_w: 2.378737211227417
loss_l: 0.5350846648216248
loss_l_alt: 3.8221516609191895
loss_y: -0.8401606678962708
loss_d: 0.0
loss_y_alt: 15.778671264648438
total_loss: 45.53089141845703

training phase
rec: 63.0758056640625
loss_p: 16.126567840576172
loss_z: 13.71830940246582
loss_w: 2.2868361473083496
loss_l: 0.5168424248695374
loss_l_alt: 3.991077184677124
loss_y: -0.896159291267395
loss_d: 0.0
loss_y_alt: 16.128904342651367
total_loss: 95.20985412597656
```

```
training phase
rec: -7.750042915344238
loss_p: 16.479236602783203
loss_z: 13.586618423461914
loss_w: 2.368070363998413
loss_l: 0.6004770994186401
loss_l_alt: 3.9790000915527344
loss_y: -0.8825371265411377
loss_d: 0.0
loss_y_alt: 16.14567756652832
total_loss: 24.350322723388672

training phase
rec: 5.02932071685791
loss_p: 15.761676788330078
loss_z: 14.581817626953125
loss_w: 2.491879940032959
loss_l: 0.4774598181247711
loss_l_alt: 3.9778053760528564
loss_y: -0.8858271241188049
loss_d: 0.0
loss_y_alt: 15.578544616699219
total_loss: 37.68156433105469

training phase
rec: 33.99256896972656
loss_p: 16.60576629638672
loss_z: 14.524539947509766
loss_w: 2.1646652221679688
loss_l: 0.48723992705345154
loss_l_alt: 4.060068130493164
loss_y: -0.9086523056030273
loss_d: 0.0
loss_y_alt: 16.248502731323242
total_loss: 66.9302749633789

training phase
rec: -20.27412986755371
loss_p: 16.62417221069336
loss_z: 14.747347831726074
loss_w: 2.3718156814575195
loss_l: 0.5181995630264282
loss_l_alt: 4.056509017944336
loss_y: -0.9102881550788879
loss_d: 0.0
loss_y_alt: 16.347074508666992
total_loss: 13.192108154296875
```

```
training phase
rec: -6.152655601501465
loss_p: 16.35297393798828
loss_z: 17.14803123474121
loss_w: 2.9431183338165283
loss_l: 0.46357911825180054
loss_l_alt: 4.110390663146973
loss_y: -0.9315273761749268
loss_d: 0.0
loss_y_alt: 16.308080673217773
total_loss: 30.24657440185547

training phase
rec: -77.23823547363281
loss_p: 16.545122146606445
loss_z: 14.64818286895752
loss_w: 2.301053524017334
loss_l: 0.4871978759765625
loss_l_alt: 4.084650039672852
loss_y: -0.9134984016418457
loss_d: 0.0
loss_y_alt: 16.51010513305664
total_loss: -43.77889633178711

training phase
rec: 35.082740783691406
loss_p: 16.437084197998047
loss_z: 16.23394012451172
loss_w: 2.8088128566741943
loss_l: 0.4716233015060425
loss_l_alt: 4.072210311889648
loss_y: -0.9125550389289856
loss_d: 0.0
loss_y_alt: 16.10635757446289
total_loss: 70.23184967041016

training phase
rec: -72.67969512939453
loss_p: 16.686893463134766
loss_z: 14.992250442504883
loss_w: 2.4085371494293213
loss_l: 0.5327689051628113
loss_l_alt: 4.150249004364014
loss_y: -0.9465469121932983
loss_d: 0.0
loss_y_alt: 16.182750701904297
total_loss: -39.09615707397461
```

```
training phase
rec: 39.6516227722168
loss_p: 16.843242645263672
loss_z: 15.025737762451172
loss_w: 2.5391974449157715
loss_l: 0.5306994915008545
loss_l_alt: 4.114178657531738
loss_y: -0.9244425296783447
loss_d: 0.0
loss_y_alt: 16.55931854248047
total_loss: 73.77587890625

training phase
rec: -57.27556228637695
loss_p: 16.56615447998047
loss_z: 15.793607711791992
loss_w: 2.6410446166992188
loss_l: 0.5150702595710754
loss_l_alt: 4.139255523681641
loss_y: -0.9359618425369263
loss_d: 0.0
loss_y_alt: 16.42471694946289
total_loss: -22.416194915771484

training phase
rec: 27.16637420654297
loss_p: 16.803550720214844
loss_z: 16.417098999023438
loss_w: 2.4223384857177734
loss_l: 0.49342796206474304
loss_l_alt: 4.15923547744751
loss_y: -0.9448443651199341
loss_d: 0.0
loss_y_alt: 16.565906524658203
total_loss: 62.571720123291016

training phase
rec: -36.193031311035156
loss_p: 16.770572662353516
loss_z: 16.611007690429688
loss_w: 2.4922935962677
loss_l: 0.48670077323913574
loss_l_alt: 4.172143936157227
loss_y: -0.9484829306602478
loss_d: 0.0
loss_y_alt: 16.47394561767578
total_loss: -0.61578369140625
```

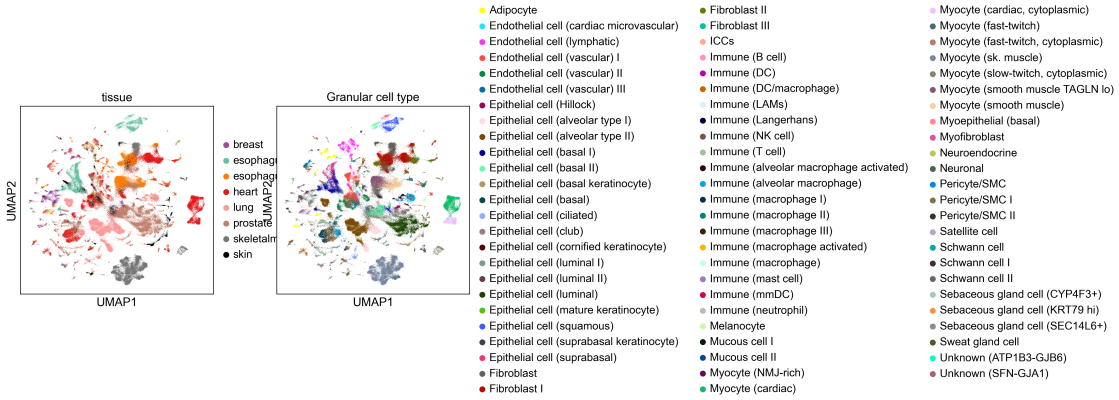
```
training phase
rec: 19.991575241088867
loss_p: 16.39247703552246
loss_z: 17.06316375732422
loss_w: 2.618678569793701
loss_l: 0.5062333345413208
loss_l_alt: 4.162381172180176
loss_y: -0.9447076320648193
loss_d: 0.0
loss_y_alt: 16.134212493896484
total_loss: 55.80763244628906

training phase
rec: -2.7116546630859375
loss_p: 16.713375091552734
loss_z: 16.167091369628906
loss_w: 2.6803884506225586
loss_l: 0.46791917085647583
loss_l_alt: 4.155095100402832
loss_y: -0.9414879083633423
loss_d: 0.0
loss_y_alt: 16.300216674804688
total_loss: 32.43604278564453
```

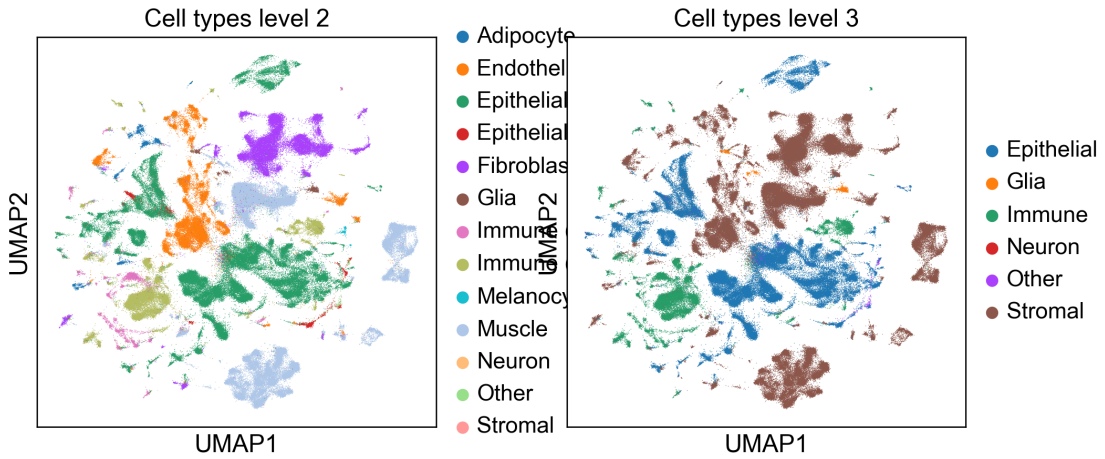
[49]:
```
model_gmmDVAE3.eval()
output = model_gmmDVAE3(data)
adata.obsm["z_gmmdvae3"] = output["z"].detach().numpy()
adata.obs["predict_gmmdvae3"] = output["q_y"].detach().argmax(-1).numpy().
 ↪astype(str)

sc.pp.neighbors(adata, use_rep="z_gmmdvae3", n_neighbors=10,)
sc.tl.umap(adata,)
```

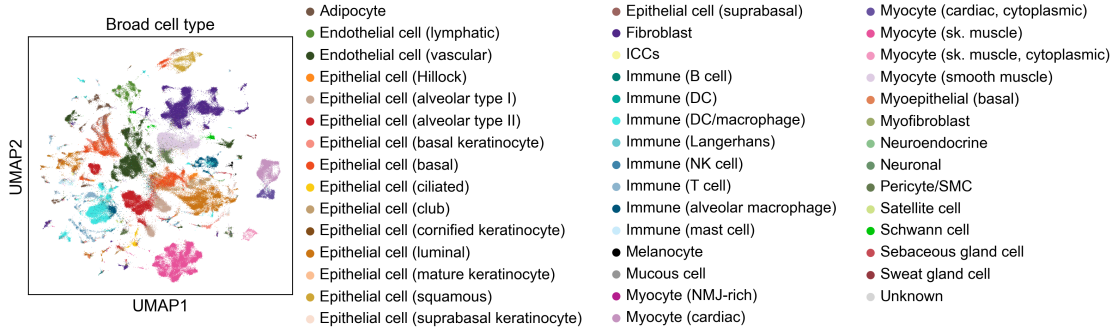
```
computing neighbors
    finished: added to `.uns['neighbors']`
    `.obsp['distances']`, distances for each pair of neighbors
    `.obsp['connectivities']`, weighted adjacency matrix (0:00:12)
computing UMAP
    finished: added
    'X_umap', UMAP coordinates (adata.obsm) (0:02:09)
```

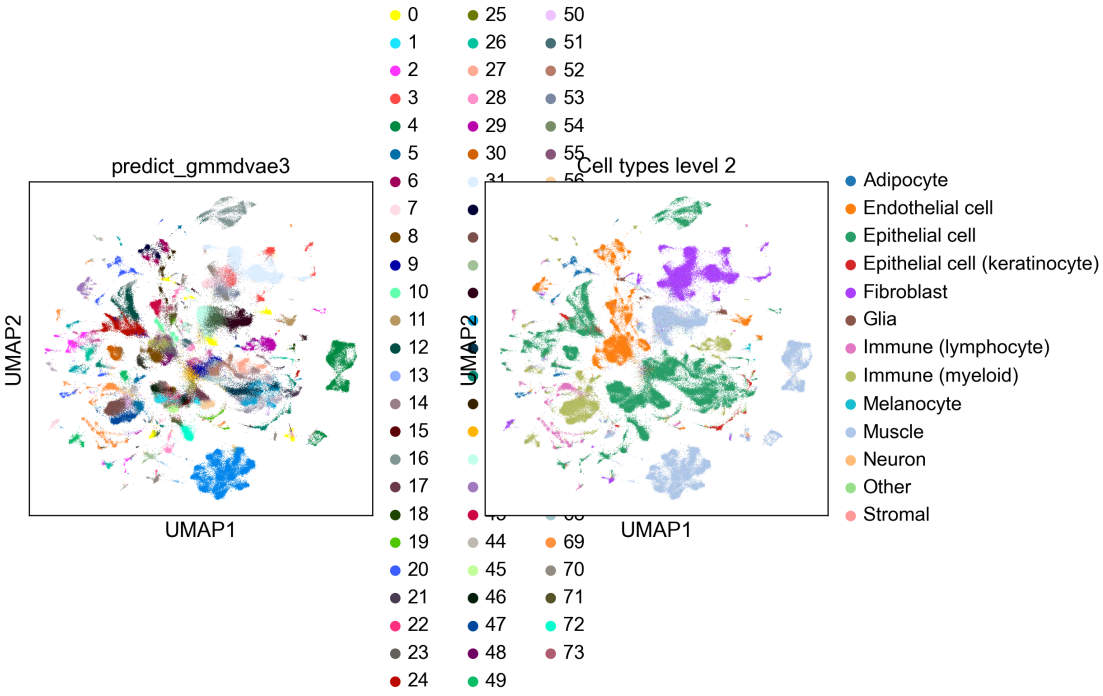
[52]:
```
sc.pl.umap(adata, color=["tissue", "Granular cell type"],)
sc.pl.umap(adata, color=["Cell types level 2", "Cell types level 3"],)
sc.pl.umap(adata, color="Broad cell type",)
sc.pl.umap(adata, color=["predict_gmmdvae3","Cell types level 2"],)
```

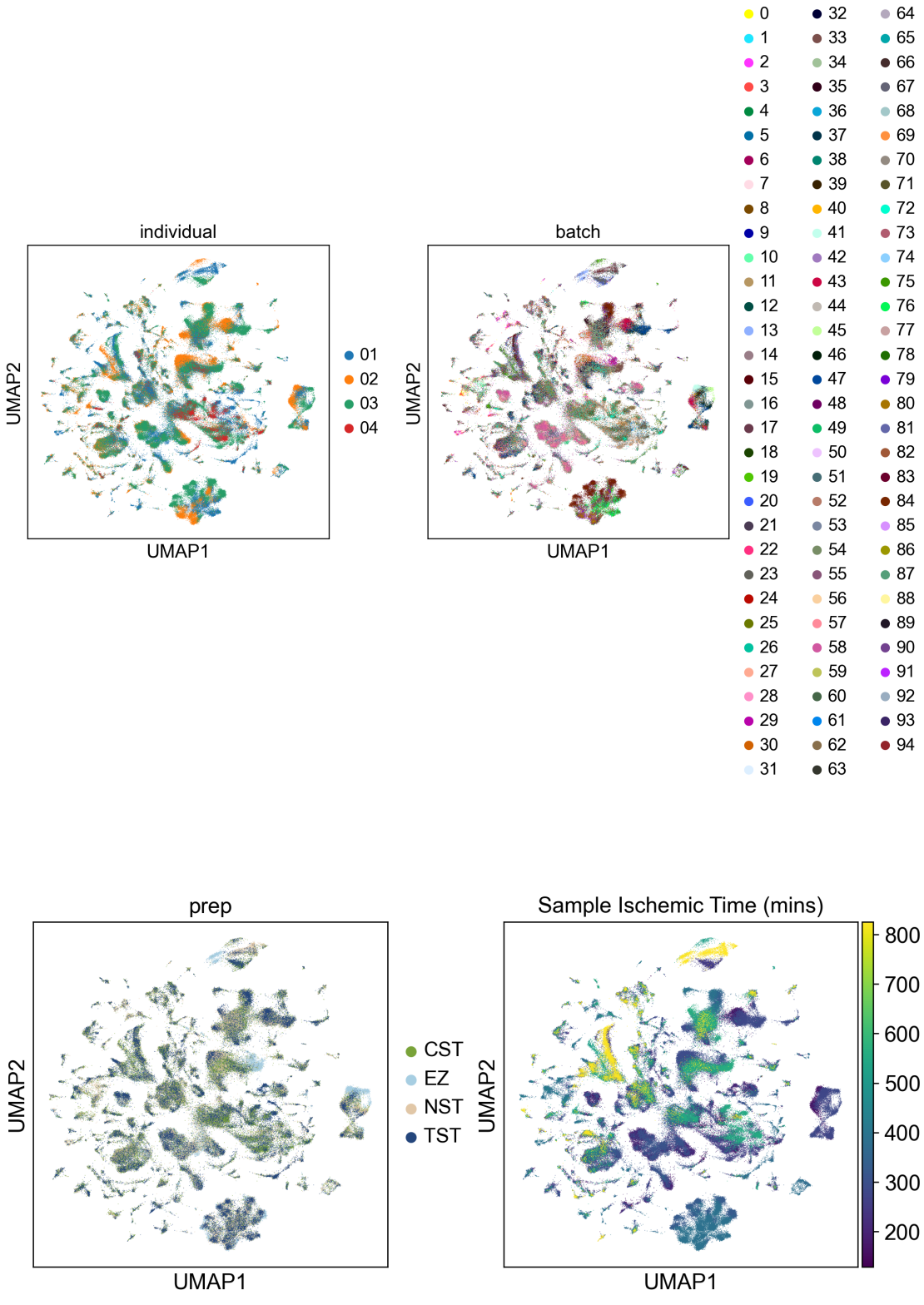
```
sc.pl.umap(adata, color=["individual", "batch"],)
sc.pl.umap(adata, color=["prep", "Sample Ischemic Time (mins)"],)
```



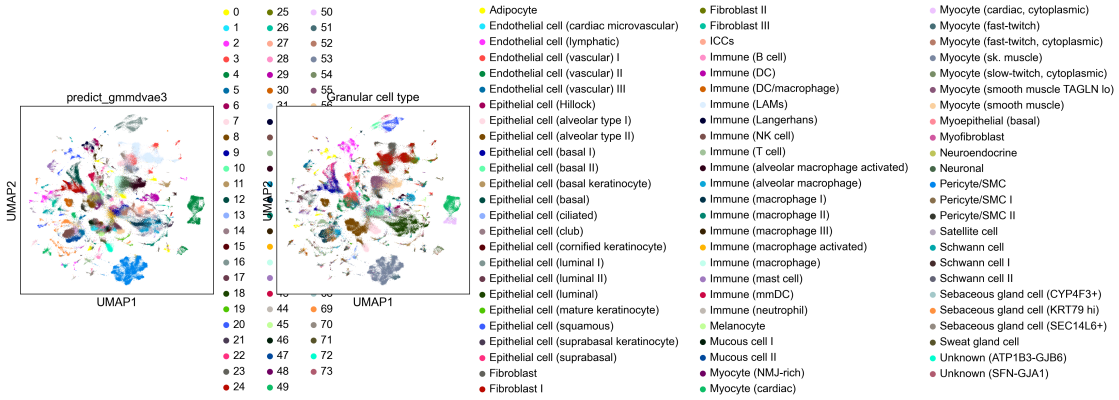




predict_gmmdvae3
UMAP2
UMAP1
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
44
45
46
47
48
49
50
51
52
53
54
55
69
70
71
72
73
Cell types level 2
UMAP2
UMAP1
Adipocyte
Endothelial cell
Epithelial cell
Epithelial cell (keratinocyte)
Fibroblast
Glia
Immune (lymphocyte)
Immune (myeloid)
Melanocyte
Muscle
Neuron
Other
Stromal


individual
UMAP2
UMAP1
01
02
03
04
batch
UMAP2
UMAP1
0 32 64
1 33 65
2 34 66
3 35 67
4 36 68
5 37 69
6 38 70
7 39 71
8 40 72
9 41 73
10 42 74
11 43 75
12 44 76
13 45 77
14 46 78
15 47 79
16 48 80
17 49 81
18 50 82
19 51 83
20 52 84
21 53 85
22 54 86
23 55 87
24 56 88
25 57 89
26 58 90
27 59 91
28 60 92
29 61 93
30 62 94
31 63
prep
UMAP2
UMAP1
CST
EZ
NST
TST
Sample Ischemic Time (mins)
UMAP2
UMAP1
800
700
600
500
400
300
200

[51]:
```
sc.pl.umap(adata, color=["predict_gmmdvae3","Granular cell type"],)
```




I think one thing that the GMM VAEs have in their favor vs. the CVAE from the paper is a better separation of the broad/granular cell types.

Next I think we'll try semi supervised training but later and maybe in a different notebook…

[ ]:
```

```