reComBat_tutorial

August 31, 2022

1 reComBat Tutorial

In this tutorial we investigate the basic functionality of the `reComBat` package and discuss best use cases. A full documentation is available here.

You can install the `reComBat` via `pip install recombat`.

In this tutorial we are going to use a preprocessed version of the public pseudomonas aeruginosa dataset used in recent paper. The preprocessing is described in our paper. If you would like to redo the preprocessing yourself, please follow the steps in `harmonizedDataCreation.py` of this repository.

First, we import necessary libraries.

[1]:

```
import os
import numpy as np
import pandas as pd
import scanpy as sc
import anndata as an
from reComBat import reComBat
```

Define a utility plotting function to make visualisations more convenient

[2]:

```
def plot(data,metadata,type='tsne',plot_mode='all',name=None):
    adata = an.AnnData(X=data,obs=metadata)

    if plot_mode == 'all':
        to_colour_by = ['gse',
                        'strain',
                        'MediumCoarse',
                        'GrowthPhase',
                        'Oxygenation',
                        'Temperature_Coarse',
                        'Culture_Coarse',
                        'ZeroHop'
                        ]
    else:
        to_colour_by = [
                        'Zero-hop cluster'
                        ]
    if type == 'tsne':
        sc.tl.tsne(adata,use_rep='X')
        if name is not None:
            sc.pl.tsne(adata,color=to_colour_by,show=False,ncols=1,hspace=0.
↪25,legend_fontsize=8,save='_'+name)
        else:
            sc.pl.tsne(adata,color=to_colour_by,show=True,ncols=1,hspace=0.
↪25,legend_fontsize=8)
    elif type == 'umap':
        sc.pp.neighbors(adata,use_rep='X')
        sc.tl.umap(adata)
        if name is not None:
            sc.pl.umap(adata,color=to_colour_by,show=True,ncols=1,hspace=0.
↪25,legend_fontsize=8,save='_'+name)
        else:
            sc.pl.umap(adata,color=to_colour_by,show=True,ncols=1,hspace=0.
↪25,legend_fontsize=8)
    elif type == 'pca':
        sc.tl.pca(adata,use_highly_variable=False)
        if name is not None:
            sc.pl.pca(adata,color=to_colour_by,show=True,ncols=1,hspace=0.
↪25,legend_fontsize=8,save='_'+name)
        else:
            sc.pl.pca(adata,color=to_colour_by,show=True,ncols=1,hspace=0.
↪25,legend_fontsize=8)
```

Set the data path. In our case it is the current directory.

[3]:

```
data_path = '.'
```

Load the data. This is a snapshot of the preprocessed dataset used in our publication.

[4]:

```
data      = pd.read_csv(os.path.join(data_path,'data_standardized_20211027.
 ↪csv'),index_col=0)
metadata  = pd.read_csv(os.path.join(data_path,'metadata_allChip_20211027.
 ↪csv'),index_col=0)
```

The zero hops are treated as categorical data.

[5]:

```
metadata.ZeroHop = metadata.ZeroHop.astype(str)
```

Use only the coarsed information to obtain more sample overlap

[6]:

```
metadata_coarse  = metadata[['gse',
                             'strain',
                             'MediumCoarse',
                             'GrowthPhase',
                             'Oxygenation',
                             'Temperature_Coarse',
                             'Culture_Coarse',
                             'Antibiotic',
                             'ZeroHop']]
```

Also fill possible `nan` values.

[7]:
```
metadata_coarse = metadata_coarse.fillna('None')
```

Only use samples which are both in the metadata and expression data. Some expression data might have been filtered out by QC.

[8]:
```
valid_ids = list(set(data.index).intersection(set(metadata_coarse.index)))
valid_ids.sort()
data_fit = data.loc[valid_ids]
metadata_fit = metadata_coarse.loc[valid_ids]
```

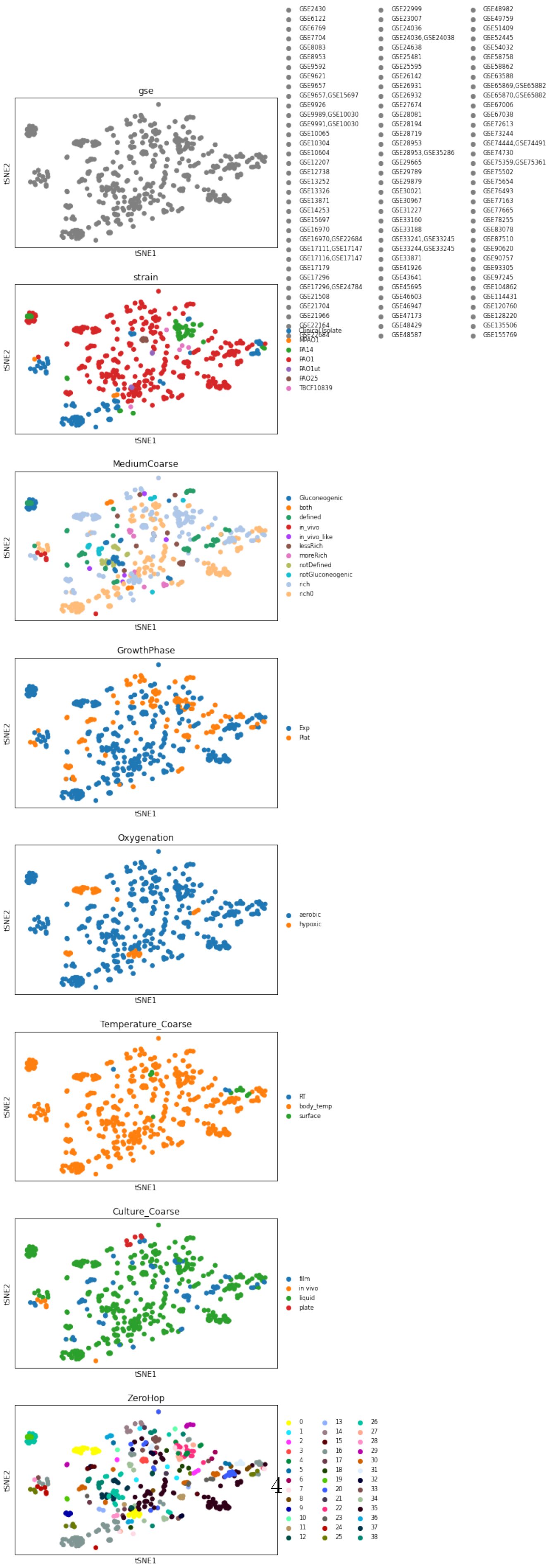
Also make sure that the samples are in correct order
E.g. line 10 of the expression data and metadata correspond to the same sample

[9]:
```
assert np.array(data_fit.index == metadata_fit.index).all()
```

Plot the initial data

[10]:
```
plot(data_fit,metadata_fit,type='tsne')
```

```
… storing 'gse' as categorical
… storing 'strain' as categorical
… storing 'MediumCoarse' as categorical
… storing 'GrowthPhase' as categorical
… storing 'Oxygenation' as categorical
… storing 'Temperature_Coarse' as categorical
… storing 'Culture_Coarse' as categorical
… storing 'Antibiotic' as categorical
… storing 'ZeroHop' as categorical
```


gse
tSNE2
tSNE1
GSE2430
GSE6122
GSE6769
GSE7704
GSE8083
GSE8953
GSE9592
GSE9621
GSE9657
GSE9657,GSE15697
GSE9926
GSE9989,GSE10030
GSE9991,GSE10030
GSE10065
GSE10304
GSE10604
GSE12207
GSE12738
GSE13252
GSE13326
GSE13871
GSE14253
GSE15697
GSE16970
GSE16970,GSE22684
GSE17111,GSE17147
GSE17116,GSE17147
GSE17179
GSE17296
GSE17296,GSE24784
GSE21508
GSE21704
GSE21966
GSE22164
GSE22684
GSE22999
GSE23007
GSE24036
GSE24036,GSE24038
GSE24638
GSE25481
GSE25595
GSE26142
GSE26931
GSE26932
GSE27674
GSE28081
GSE28194
GSE28719
GSE28953
GSE28953,GSE35286
GSE29665
GSE29789
GSE29879
GSE30021
GSE30967
GSE31227
GSE33160
GSE33188
GSE33241,GSE33245
GSE33244,GSE33245
GSE33871
GSE41926
GSE43641
GSE45695
GSE46603
GSE46947
GSE47173
GSE48429
GSE48587
GSE48982
GSE49759
GSE51409
GSE52445
GSE54032
GSE58758
GSE58862
GSE63588
GSE65869,GSE65882
GSE65870,GSE65882
GSE67006
GSE67038
GSE72613
GSE73244
GSE74444,GSE74491
GSE74730
GSE75359,GSE75361
GSE75502
GSE75654
GSE76493
GSE77163
GSE77665
GSE78255
GSE83078
GSE87510
GSE90620
GSE90757
GSE93305
GSE97245
GSE104862
GSE114431
GSE120760
GSE128220
GSE135506
GSE155769
strain
tSNE2
tSNE1
Clinical Isolate
MPAO1
PA14
PAO1
PAO1ut
PAO25
TBCF10839
MediumCoarse
tSNE2
tSNE1
Gluconeogenic
both
defined
in_vivo
in_vivo_like
lessRich
moreRich
notDefined
notGluconeogenic
rich
rich0
GrowthPhase
tSNE2
tSNE1
Exp
Plat
Oxygenation
tSNE2
tSNE1
aerobic
hypoxic
Temperature_Coarse
tSNE2
tSNE1
RT
body_temp
surface
Culture_Coarse
tSNE2
tSNE1
film
in vivo
liquid
plate
ZeroHop
tSNE2
tSNE1
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
31
32
33
34
35
36
37
38

Initialize the reComBat class

```
[11]: model = reComBat(parametric=True,            # use parametric or non-parametric␣
       ↪empirical Bayes method.
                                                   # The parametric method is␣
       ↪significantly faster, whereas the
                                                   # non-parametric method is more␣
       ↪flexible.
                         model='ridge',            # The regression model to be used.
                                                   # In our experience pure ridge␣
       ↪regression performs best for singular design matrices
                                                   # and pure linear regression is␣
       ↪best for non=singular matrices.
                         config={'alpha':1e-9},    # Optional arguments for the␣
       ↪regression model.
                                                   # We tend to use a tiny␣
       ↪regularisation parameter.
                                                   # This has also been cnfirmed by␣
       ↪CV.
                         conv_criterion=1e-4,      # The convergence criterion for␣
       ↪the empirical Bayes optimisation.
                                                   # This value works well in␣
       ↪practise.
                         max_iter=1000,            # The maximum number of iterations␣
       ↪to stop if convergence is not reached.
                                                   # This may also be useful for␣
       ↪smaller convergence criteria.
                         n_jobs=1,                 # This parameter is only useful in␣
       ↪non-parametric optimisation.
                                                   # The non-parametric optimisation␣
       ↪is very slow, but can be parallelised easily.
                                                   # Set this to the number of CPUs␣
       ↪on your machine for significant speed ups.
                         mean_only=False,          # Adjust the mean of your data␣
       ↪only (not the variance).
                                                   # This can be useful for single␣
       ↪sample batches (where the variance is infinite)
                         optimize_params=True,     # If False no empirical Bayes␣
       ↪optimisation is performed.
                         reference_batch=None,     # If a reference batch is present␣
       ↪(e.g. a batch which is considered "batch effect free")
                                                   # it can be set such that all data␣
       ↪is adjuste dwith respect to this reference batch.
                         verbose=True              # Turn log messages on or off
```

```
                 )
```

Fit reComBat. The input is the raw data, the batch identifiers and the design matrix

```
[12]: model.fit(data_fit,metadata_fit.gse,X=metadata_fit.
       ↪drop(['gse','ZeroHop'],axis=1))
```

```
[reComBat] 2022-08-31 17:05:01,000 Starting to fot reComBat.
[reComBat] 2022-08-31 17:05:01,021 Fit the linear model.
[reComBat] 2022-08-31 17:05:01,241 Starting the empirical parametric
optimisation.
[reComBat] 2022-08-31 17:05:01,639 Optimisation finished.
[reComBat] 2022-08-31 17:05:01,640 reComBat is fitted.
```

Not transform the data. The input stays the same

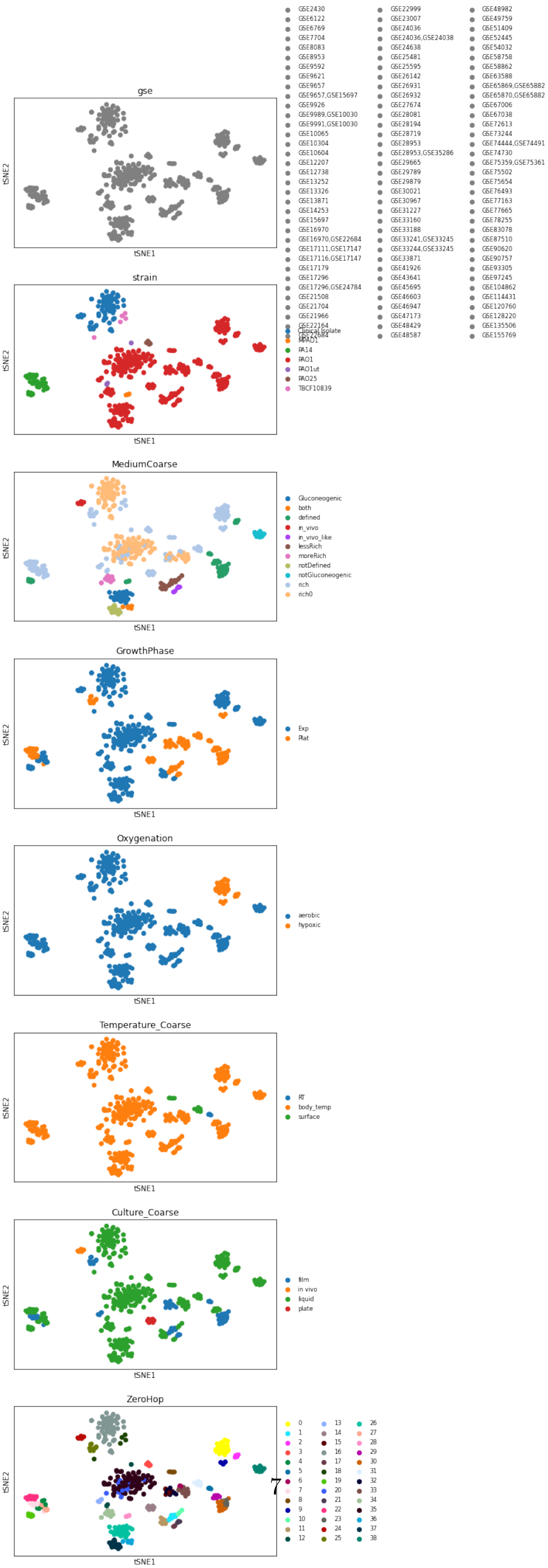
```
[13]: data_combat = model.transform(data_fit,metadata_fit.gse,X=metadata_fit.
       ↪drop(['gse','ZeroHop'],axis=1))
```

```
[reComBat] 2022-08-31 17:05:01,650 Starting to transform.
[reComBat] 2022-08-31 17:05:01,827 Transform finished.
```

Plot the output data for comparison

```
[14]: plot(data_combat,metadata_fit,type='tsne')
```

```
… storing 'gse' as categorical
… storing 'strain' as categorical
… storing 'MediumCoarse' as categorical
… storing 'GrowthPhase' as categorical
… storing 'Oxygenation' as categorical
… storing 'Temperature_Coarse' as categorical
… storing 'Culture_Coarse' as categorical
… storing 'Antibiotic' as categorical
… storing 'ZeroHop' as categorical
```


gse
tSNE2
tSNE1
GSE2430
GSE6122
GSE6769
GSE7704
GSE8083
GSE8953
GSE9592
GSE9621
GSE9657
GSE9657,GSE15697
GSE9926
GSE9989,GSE10030
GSE9991,GSE10030
GSE10065
GSE10304
GSE10604
GSE12207
GSE12738
GSE13252
GSE13326
GSE13871
GSE14253
GSE15697
GSE16970
GSE16970,GSE22684
GSE17111,GSE17147
GSE17116,GSE17147
GSE17179
GSE17296
GSE17296,GSE24784
GSE21508
GSE21704
GSE21966
GSE22164
GSE22684
GSE22999
GSE23007
GSE24036
GSE24036,GSE24038
GSE24638
GSE25481
GSE25595
GSE26142
GSE26931
GSE26932
GSE27674
GSE28081
GSE28194
GSE28719
GSE28953
GSE28953,GSE35286
GSE29665
GSE29789
GSE29879
GSE30021
GSE30967
GSE31227
GSE33160
GSE33188
GSE33241,GSE33245
GSE33244,GSE33245
GSE33871
GSE41926
GSE43641
GSE45695
GSE46603
GSE46947
GSE47173
GSE48429
GSE48587
GSE48982
GSE49759
GSE51409
GSE52445
GSE54032
GSE58758
GSE58862
GSE63588
GSE65869,GSE65882
GSE65870,GSE65882
GSE67006
GSE67038
GSE72613
GSE73244
GSE74444,GSE74491
GSE74730
GSE75359,GSE75361
GSE75502
GSE75654
GSE76493
GSE77163
GSE77665
GSE78255
GSE83078
GSE87510
GSE90620
GSE90757
GSE93305
GSE97245
GSE104862
GSE114431
GSE120760
GSE128220
GSE135506
GSE155769
strain
tSNE2
tSNE1
Clinical Isolate
MPAO1
PA14
PAO1
PAO1ut
PAO25
TBCF10839
MediumCoarse
tSNE2
tSNE1
Gluconeogenic
both
defined
in_vivo
in_vivo_like
lessRich
moreRich
notDefined
notGluconeogenic
rich
rich0
GrowthPhase
tSNE2
tSNE1
Exp
Plat
Oxygenation
tSNE2
tSNE1
aerobic
hypoxic
Temperature_Coarse
tSNE2
tSNE1
RT
body_temp
surface
Culture_Coarse
tSNE2
tSNE1
film
in vivo
liquid
plate
ZeroHop
tSNE2
tSNE1
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
31
32
33
34
35
36
37
38