PREPROCESSING EXPLANATION

Due to the nature of DCA, it (And subsequentially all autoencoders) were using a different preprocessing.

|  |  |
| --- | --- |
| **Normal Preprocessing** | **DCA / SCA: Countdata preprocessing** |
| Filter out cells with less than [min\_genes\_per\_cell] genes detected. | |
| Filter out genes present in less than [min\_cells\_pre\_gene] cells. | |
| Filter out cells with more than [max\_mt\_perc] % of mitochondrial genes | |
| Filter out cells with more than [max\_num\_features] genes. (doublets ?) | |
| Normalize (sc.pp.normalize\_total) |  |
| Logarithmize (sc.pp.log1p)  *is expected for next step* |  |
| Feature selection of [num\_top\_genes + 1] | |
| regress out effects of total\_counts\_per\_cell & percentage of mitochondrial genes. (I don’t know how exactly, it is an inbuilt function) | - |
| train test split | |
|  | re-filter sets, to make sure we don’t have a set of training or testdata which in isolation would contain a “zero gene”. |
|  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Baselines** | **BCA** | **SCA** | **DCA** |
| Scale with standardscaler  tSNE, UMAP, LSA, ICA, PCA | Scale with standardscaler |  |  |
|  |  | filter again:  min genes = 1, min cells = 1  *this shouldn’t have any effect anymore, but eraslan was REALLY keen on not having zero genes anymore, so I carried it over to SCA. At this point there already was an assert statement earlier, so whatev* | |
|  |  |  |  |
|  |  | create size\_factors for each cell: using counts (not unique) →  n\_counts / median(n\_counts) | |
|  |  | logarithmize (sc.pp.log1p) | |
|  |  | normalize (sc.pp.scale) | |