§Bioinfo-Lab-3

§Mouse Retina Dataset

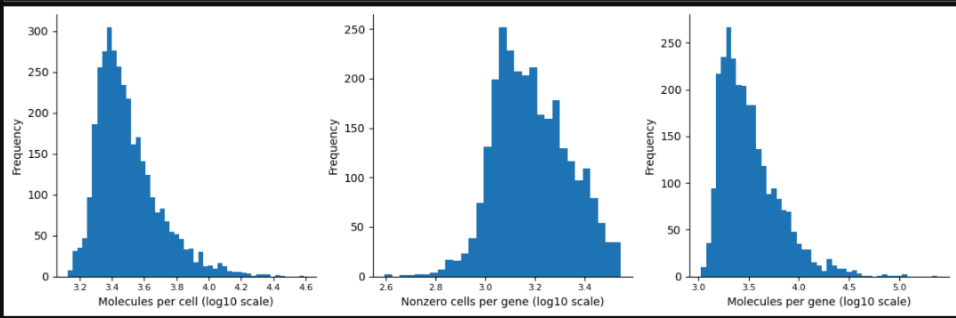
Task 1: Filter and Normalization

[4]:
```
# !pip install palantir
```

[5]:
```
import palantir

fig, ax = palantir.plot.plot_molecules_per_cell_and_gene(data.T)

filtered_data = palantir.preprocess.filter_counts_data(data.T, cell_min_molecules=1000, genes_min_cells=10)
```



[6]:
```
from sklearn.preprocessing import MinMaxScaler, StandardScaler

scaler_minmax = MinMaxScaler()

data_minmax= pd.DataFrame(scaler_minmax.fit_transform(filtered_data), columns=filtered_data.columns)


print("Min-Max Scaling:")
data_minmax
```

Min-Max Scaling:

[6]:

	CEP290	CCDC59	PPFIA2	LIN7A	PPP1R12A	SYT1	ZDHHC17	OSBPL8	NAP1L1	KRR1	...	TBL1X	GRIPAP1	PQBP1	PCSK1N	DMD	MAGED1	ARHGEF9	LAS1L	DYNLT3	PJA1
0	0.028571	0.157895	0.200000	0.214286	0.071429	0.723404	0.4	0.3	0.061728	0.4375	...	0.583333	0.3	0.285714	0.531646	0.02	0.402985	0.692308	0.352941	0.882353	0.571429
1	0.085714	0.052632	0.266667	0.392857	0.357143	0.542553	0.4	0.3	0.061728	0.1250	...	0.416667	0.4	0.714286	0.202532	0.08	0.298507	0.692308	0.176471	1.000000	0.642857
2	0.000000	0.052632	0.133333	0.214286	0.071429	0.531915	0.0	0.1	0.037037	0.0625	...	0.333333	0.1	0.142857	0.215190	0.04	0.313433	0.153846	0.058824	0.470588	0.785714
3	0.057143	0.157895	0.200000	0.250000	0.000000	0.563830	0.4	0.5	0.024691	0.1250	...	0.250000	0.3	0.142857	0.101266	0.00	0.149254	0.076923	0.058824	0.647059	0.142857
4	0.285714	0.157895	0.133333	0.000000	0.071429	0.265957	0.4	0.1	0.074074	0.0000	...	0.083333	1.0	0.285714	0.025316	0.18	0.014925	0.000000	0.117647	0.294118	0.214286
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
3512	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0	0.2	0.000000	0.0000	...	0.000000	0.0	0.000000	0.000000	0.00	0.044776	0.000000	0.000000	0.000000	0.000000
3513	0.000000	0.052632	0.000000	0.000000	0.000000	0.010638	0.0	0.0	0.037037	0.0000	...	0.000000	0.0	0.000000	0.000000	0.08	0.014925	0.000000	0.117647	0.000000	0.000000
3514	0.000000	0.263158	0.000000	0.000000	0.000000	0.021277	0.0	0.0	0.000000	0.0000	...	0.000000	0.0	0.000000	0.012658	0.00	0.000000	0.000000	0.000000	0.000000	0.000000
3515	0.000000	0.000000	0.000000	0.000000	0.071429	0.053191	0.0	0.0	0.000000	0.0000	...	0.083333	0.1	0.000000	0.000000	0.00	0.000000	0.153846	0.000000	0.000000	0.071429
3516	0.000000	0.000000	0.000000	0.000000	0.071429	0.000000	0.0	0.0	0.000000	0.0000	...	0.000000	0.0	0.000000	0.000000	0.00	0.029851	0.000000	0.000000	0.000000	0.000000

3517 rows × 2773 columns

[7]:
```
scaler_zscore = StandardScaler()
data_zscore = pd.DataFrame(scaler_zscore.fit_transform(filtered_data),columns=filtered_data.columns)
print("\nZ-Score Normalization:")
data_zscore
```

Z-Score Normalization:

[7]:

	CEP290	CCDC59	PPFIA2	LIN7A	PPP1R12A	SYT1	ZDHHC17	OSBPL8	NAP1L1	KRR1	...	TBL1X	GRIPAP1	PQBP1	PCSK1N	DMD	MAGED1	ARHGEF9	LAS1L	DYNLT3	PJA1
0	-0.382141	1.237542	1.271030	1.015943	0.387928	6.149090	2.761013	2.454297	0.900040	6.980350	...	4.820180	1.948172	1.371529	6.861591	-0.272221	5.927894	5.784232	4.359605	11.830845	4.402182
1	0.283530	-0.085186	1.910545	2.387668	4.172511	4.369979	2.761013	2.454297	0.900040	1.605089	...	3.255785	2.817239	4.395771	2.355885	0.721278	4.272339	5.784232	1.838051	13.479709	5.039452
2	-0.714976	-0.085186	0.631514	1.015943	0.387928	4.265325	-0.660581	0.457128	0.246023	0.530037	...	2.473587	0.210039	0.363448	2.529181	0.058945	4.508847	0.869412	0.157015	6.059821	6.313992
3	-0.049305	1.237542	1.271030	1.290288	-0.558218	4.579286	2.761013	4.451465	-0.080985	1.605089	...	1.691389	1.948172	0.363448	0.969514	-0.603388	1.907261	0.167294	0.157015	8.533117	0.578562
4	2.613376	1.237542	0.631514	-0.630128	0.387928	1.648986	2.761013	0.457128	1.227049	-0.545015	...	0.126993	8.031640	1.371529	-0.070265	2.377110	-0.221310	-0.534823	0.997533	3.586525	1.215832
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
3512	-0.714976	-0.746550	-0.647516	-0.630128	-0.558218	-0.967354	-0.660581	1.455713	-0.735002	-0.545015	...	-0.655205	-0.659028	-0.644633	-0.416857	-0.603388	0.251706	-0.534823	-0.683503	-0.535635	-0.695978
3513	-0.714976	-0.085186	-0.647516	-0.630128	-0.558218	-0.862700	-0.660581	-0.541456	0.246023	-0.545015	...	-0.655205	-0.659028	-0.644633	-0.416857	0.721278	-0.221310	-0.534823	0.997533	-0.535635	-0.695978
3514	-0.714976	2.560270	-0.647516	-0.630128	-0.558218	-0.758047	-0.660581	-0.541456	-0.735002	-0.545015	...	-0.655205	-0.659028	-0.644633	-0.243561	-0.603388	-0.457818	-0.534823	-0.683503	-0.535635	-0.695978
3515	-0.714976	-0.746550	-0.647516	-0.630128	0.387928	-0.444086	-0.660581	-0.541456	-0.735002	-0.545015	...	0.126993	0.210039	-0.644633	-0.416857	-0.603388	-0.457818	0.869412	-0.683503	-0.535635	-0.058708
3516	-0.714976	-0.746550	-0.647516	-0.630128	0.387928	-0.967354	-0.660581	-0.541456	-0.735002	-0.545015	...	-0.655205	-0.659028	-0.644633	-0.416857	-0.603388	0.015198	-0.534823	-0.683503	-0.535635	-0.695978

3517 rows × 2773 columns

[8]:
```
import numpy as np
data_log = np.log(filtered_data+1)
print("Log-transformed Data:")
data_log
```

Log-transformed Data:

[8]:

	CEP290	CCDC59	PPFIA2	LIN7A	PPP1R12A	SYT1	ZDHHC17	OSBPL8	NAP1L1	KRR1	...	TBL1X	GRIPAP1	PQBP1	PCSK1N	DMD	MAGED1	ARHGEF9	LAS1L	DYNLT3	PJA1
r1_GGCCGCAGTCCG	0.693147	1.386294	1.386294	1.945910	0.693147	4.234107	1.609438	1.386294	1.791759	2.079442	...	2.079442	1.386294	1.098612	3.761200	0.693147	3.332205	2.302585	1.945910	2.772589	2.197225
r1_CTTGTGCGGGAA	1.386294	0.693147	1.609438	2.484907	1.791759	3.951244	1.609438	1.386294	1.791759	1.098612	...	1.791759	1.609438	1.791759	2.833213	1.609438	3.044522	2.302585	1.386294	2.890372	2.302585
r1_GCGCAACTGCTC	0.000000	0.693147	1.098612	1.945910	0.693147	3.931826	0.000000	0.693147	1.386294	0.693147	...	1.609438	0.693147	0.693147	2.890372	1.098612	3.091042	1.098612	0.693147	2.197225	2.484907
r1_GATTGGGAGGCA	1.098612	1.386294	1.386294	2.079442	0.000000	3.988984	1.609438	1.791759	1.098612	1.098612	...	1.386294	1.386294	0.693147	2.197225	0.000000	2.397895	0.693147	0.693147	2.484907	1.098612
r1_GTGCCGCCTCTC	2.397895	1.386294	1.098612	0.000000	0.693147	3.258097	1.609438	0.693147	1.945910	0.000000	...	0.693147	2.397895	1.098612	1.098612	2.302585	0.693147	0.000000	1.098612	1.791759	1.386294
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
p1_GTCTGATGCAAC	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.098612	0.000000	0.000000	...	0.000000	0.000000	0.000000	0.000000	0.000000	1.386294	0.000000	0.000000	0.000000	0.000000
p1_AGTGCCAGCTCG	0.000000	0.693147	0.000000	0.000000	0.000000	0.693147	0.000000	0.000000	1.386294	0.000000	...	0.000000	0.000000	0.000000	0.000000	1.609438	0.693147	0.000000	1.098612	0.000000	0.000000
p1_CCAATAGCCGAT	0.000000	1.791759	0.000000	0.000000	0.000000	1.098612	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000	0.000000	0.693147	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
p1_TTAGTCCTACCG	0.000000	0.000000	0.000000	0.000000	0.693147	1.791759	0.000000	0.000000	0.000000	0.000000	...	0.693147	0.693147	0.000000	0.000000	0.000000	0.000000	1.098612	0.000000	0.000000	0.693147
p1_GCATAAAGTATA	0.000000	0.000000	0.000000	0.000000	0.693147	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000	0.000000	0.000000	0.000000	1.098612	0.000000	0.000000	0.000000	0.000000

3517 rows × 2773 columns

Task 2: MAGIC

[22]:
```
# !pip install magic-impute
```

[10]:
```
import magic
magic_operator=magic.MAGIC()
data_magic =magic_operator.fit_transform(data_log, genes='all_genes')
```

```
Calculating MAGIC...
  Running MAGIC on 3517 cells and 2773 genes.
  Calculating graph and diffusion operator...
    Calculating PCA...
    Calculated PCA in 0.54 seconds.
    Calculating KNN search...
    Calculated KNN search in 1.23 seconds.
    Calculating affinities...
    Calculated affinities in 1.14 seconds.
  Calculated graph and diffusion operator in 2.94 seconds.
  Calculating imputation...
  Calculated imputation in 0.48 seconds.
Calculated MAGIC in 3.45 seconds.
```

[11]:
```
data_magic
```

[11]:

	CEP290	CCDC59	PPFIA2	LIN7A	PPP1R12A	SYT1	ZDHHC17	OSBPL8	NAP1L1	KRR1	...	TBL1X	GRIPAP1	PQBP1	PCSK1N	DMD	MAGED1	ARHGEF9	LAS1L	DYNLT3	PJA1
r1_GGCCGCAGTCCG	0.862227	1.083810	1.120356	2.028935	1.076615	3.900440	1.170198	0.980097	2.044446	1.054837	...	1.724078	1.266112	1.446556	2.884915	0.683855	2.606584	1.556724	1.466776	1.977347	2.026418
r1_CTTGTGCGGGAA	0.861889	1.080732	1.109982	2.023916	1.064033	3.894473	1.163940	0.975179	2.043652	1.043431	...	1.710471	1.261051	1.442919	2.876661	0.669476	2.596321	1.540784	1.459034	1.964914	2.018262
r1_GCGCAACTGCTC	0.844133	1.069059	1.068900	1.992562	1.020416	3.874745	1.138689	0.942444	2.040263	1.013713	...	1.666048	1.236924	1.412667	2.857988	0.613030	2.572097	1.485537	1.424778	1.919569	2.001678
r1_GATTGGGAGGCA	0.834013	1.000381	0.977214	1.901137	0.796713	3.686407	0.983522	0.867776	1.874945	0.794658	...	1.376634	1.133459	1.165508	2.654247	0.422974	2.278723	1.361222	1.266365	1.611574	1.786427
r1_GTGCCGCCTCTC	2.481328	1.651764	1.248324	0.475312	1.047405	2.948844	1.190809	1.141672	2.177910	0.964871	...	1.008528	1.471604	1.023622	1.261460	2.576503	1.053146	0.272308	1.170301	0.933983	0.751174
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
p1_GTCTGATGCAAC	0.074809	0.250276	0.069107	0.263597	0.180710	0.127756	0.093311	0.183249	0.156478	0.099712	...	0.395145	0.079952	0.150062	0.211371	0.221174	0.807290	0.112538	0.101429	0.396065	0.302892
p1_AGTGCCAGCTCG	0.782621	0.420609	0.184430	0.033470	0.169211	1.264931	0.200687	0.238599	0.722147	0.182929	...	0.183542	0.235960	0.152606	0.356440	0.834892	0.265389	0.048377	0.217577	0.196417	0.115361
p1_CCAATAGCCGAT	0.743084	0.501508	0.146600	0.047412	0.153437	1.170968	0.148179	0.256126	0.727583	0.179621	...	0.225725	0.234132	0.169391	0.394956	0.829194	0.235355	0.039011	0.202259	0.139369	0.119419
p1_TTAGTCCTACCG	0.503462	0.339239	0.388419	0.629527	0.242635	2.016967	0.232638	0.230235	0.653998	0.207670	...	0.382004	0.333536	0.339399	0.503003	0.406402	0.480049	0.343619	0.391367	0.175018	0.465066
p1_GCATAAAGTATA	0.096076	0.235807	0.057068	0.243076	0.173100	0.147513	0.099769	0.162345	0.162976	0.084243	...	0.418160	0.087797	0.168241	0.197920	0.242122	0.764070	0.131225	0.121671	0.419430	0.326460

3517 rows × 2773 columns

[12]:
```
# Task 3: t-SNE
```

[13]:
```
from sklearn.manifold import TSNE
import matplotlib.pyplot as plt
```

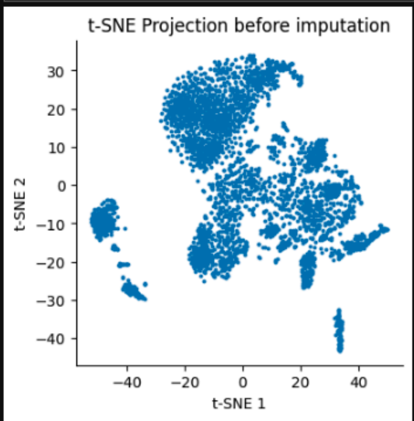
[14]:
```
tsne = TSNE(n_components=2)
data_tsne=tsne.fit_transform(data_log)
data_magic_tsne = tsne.fit_transform(data_magic)
```

Task 3: t-SNE

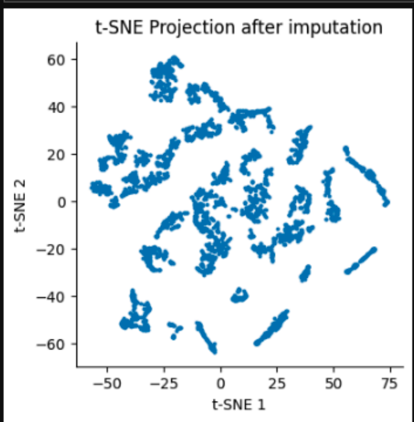
[15]:
```
from sklearn.manifold import TSNE
import matplotlib.pyplot as plt

tsne = TSNE(n_components=2)
data_tsne =tsne.fit_transform(data_log)
data_magic_tsne = tsne.fit_transform(data_magic)
```

[16]:
```
plt.scatter(data_tsne[:,0],data_tsne[:,1],s=3)
plt.title('t-SNE Projection before imputation')
plt.xlabel('t-SNE 1')
plt.ylabel('t-SNE 2')
plt.show()
```



[17]:
```
plt.scatter(data_magic_tsne[:,0],data_magic_tsne[:,1],s=3)
plt.title('t-SNE Projection after imputation')
plt.xlabel('t-SNE 1')
plt.ylabel('t-SNE 2')
plt.show()
```



Task 4: Dimension

[18]:
```
from sklearn.decomposition import PCA

pca = PCA(n_components=2)
data_pca =pca.fit_transform(data_log)

import umap
reducer =  umap.UMAP(n_neighbors=5)

data_umap = reducer.fit_transform(data_log)
```

Task 5: Clustering

[19]:
```
from sklearn.cluster import KMeans

kmeans = KMeans(n_clusters=10)
kmeans_labels = kmeans.fit_predict(data_umap)

from sklearn.cluster import AgglomerativeClustering

model =AgglomerativeClustering(n_clusters =10, metric = 'euclidean', linkage='ward')

hierarchical_labels = model.fit_predict(data_umap)
```

Task 6: ARI

[20]:
```
from sklearn.metrics import adjusted_rand_score
import pandas as pd

ground_truth = pd.read_csv("data/data1_mouse_retina/sample_cluster_ref_filtered.txt", sep='\t', header=None, index_col=0)

ground_truth.index = ground_truth.index.astype(str)
filtered_data.index = filtered_data.index.astype(str)

filter_truth = ground_truth[ground_truth.index.isin(filtered_data.index)]

truth_labels = filter_truth.iloc[:, 0].values.flatten()

ari_kmeans = adjusted_rand_score(truth_labels, kmeans_labels)
ari_hierarchical = adjusted_rand_score(truth_labels, hierarchical_labels)

ari_kmeans, ari_hierarchical
```

[20]:
```
(0.3574350970964889, 0.3957469246365283)
```

§ Mesc Dataset

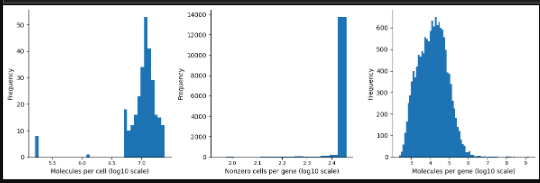
Task 1: Filter and Normalization

[4]:
```
# !pip install palantir
```

[5]:
```
import palantir

fig, ax = palantir.plot.plot_molecules_per_cell_and_gene(data.T)

filtered_data = palantir.preprocess.filter_counts_data(data.T, cell_min_molecules=1000, genes_min_cells=10)
```



[6]:
```
from sklearn.preprocessing import MinMaxScaler, StandardScaler

scaler_minmax = MinMaxScaler()

data_minmax= pd.DataFrame(scaler_minmax.fit_transform(filtered_data), columns=filtered_data.columns)

print("Min-Max Scaling:")
data_minmax
```

Min-Max Scaling:

[6]:

	ENSMUSG00000000001	ENSMUSG00000000028	ENSMUSG00000000031	ENSMUSG00000000037	ENSMUSG00000000056	ENSMUSG00000000078	ENSMUSG00000000085	ENSMUSG00000000088	ENSMUSG00000000094	ENSMUSG00000000120	...	ERCC.00164	ERCC.00165	ERCC.00168	ERCC.00170	ERCC.00171	Ambiguous	No_feature	Not_aligned	Too_low_aQual	Aligned
0	0.297666	0.437425	0.078971	0.199149	0.267462	0.311397	0.066586	0.368351	0.599017	0.042353	...	0.349851	0.317296	0.043817	0.389507	0.367298	0.514417	0.396289	0.578927	0.536092	0.458114
1	0.311678	0.448265	0.106848	0.205554	0.259273	0.313079	0.068457	0.360144	0.596087	0.044830	...	0.344282	0.319697	0.051821	0.391159	0.368039	0.522376	0.402710	0.587539	0.543162	0.462967
2	0.521736	0.542458	0.025673	0.341049	0.463726	0.224997	0.127140	0.273641	0.188037	0.229664	...	0.112335	0.213707	0.086098	0.302540	0.240185	0.512128	0.800223	0.555150	0.528920	0.392137
3	0.232426	0.282286	0.025724	0.332288	0.198940	0.172003	0.146274	0.132314	0.085921	0.082807	...	0.037660	0.101438	0.134252	0.110882	0.118232	0.273164	0.417318	0.299690	0.249203	0.199701
4	0.716787	0.395347	0.121912	0.103380	0.374167	0.392639	0.609028	0.304727	0.080298	0.004721	...	0.724508	0.552348	0.679754	0.597610	0.599019	0.581545	0.448116	0.682398	0.639069	0.621657
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
283	0.707367	0.617786	0.006732	0.425728	0.541494	0.652935	0.390615	0.324901	0.352629	0.881879	...	0.617162	0.438806	0.256923	0.488610	0.460305	0.701392	0.528644	0.632513	0.619711	0.598007
284	0.758283	0.673692	0.127770	0.372992	0.393126	0.480685	0.388750	0.273557	0.175108	0.414850	...	0.317901	0.325861	0.400683	0.426164	0.357468	0.640777	0.481201	0.568829	0.538333	0.509243
285	0.374616	0.530549	0.000359	0.450220	0.461087	0.633224	0.295914	0.269149	0.512427	0.979292	...	0.976154	0.702845	0.597365	0.686783	0.754129	0.633552	0.406821	0.568772	0.699958	0.738377
286	0.303875	0.719023	0.000026	0.919671	0.476084	0.645938	0.272513	0.310979	0.908790	0.705800	...	0.615431	0.972485	0.943900	0.962548	0.982112	0.844115	0.586892	0.777909	0.962706	0.972933
287	0.311352	0.732712	0.000017	0.942808	0.469941	0.659005	0.278455	0.318551	1.000000	0.700027	...	0.590438	0.990347	0.928460	0.985497	0.991852	0.850547	0.592422	0.784290	0.974668	0.983087

288 rows × 14328 columns

[7]:
```
scaler_zscore = StandardScaler()
data_zscore = pd.DataFrame(scaler_zscore.fit_transform(filtered_data),columns=filtered_data.columns)
print("\nZ-Score Normalization:")
data_zscore
```

Z-Score Normalization:

[7]:

	ENSMUSG00000000001	ENSMUSG00000000028	ENSMUSG00000000031	ENSMUSG00000000037	ENSMUSG00000000056	ENSMUSG00000000078	ENSMUSG00000000085	ENSMUSG00000000088	ENSMUSG00000000094	ENSMUSG00000000120	...	ERCC.00164	ERCC.00165	ERCC.00168	ERCC.00170	ERCC.00171	Ambiguous	No_feature	Not_aligned	Too_low_aQual	Aligned
0	-0.872115	-0.435657	-0.058925	-0.904358	-0.426047	-0.578923	-0.981970	0.151376	1.570250	-1.140929	...	0.180518	0.039349	-0.922172	0.244941	0.138034	-0.186552	-0.847955	0.159600	0.203582	0.014198
1	-0.795691	-0.376500	0.092790	-0.865235	-0.469178	-0.570866	-0.972575	0.109185	1.557656	-1.132539	...	0.158247	0.051097	-0.887288	0.252562	0.141554	-0.144178	-0.816568	0.207352	0.236293	0.038526
2	0.350032	0.137578	-0.348984	-0.037585	0.607765	-0.992812	-0.677809	-0.335501	-0.195747	-0.506491	...	-0.769385	-0.467559	-0.737913	-0.156264	-0.466322	-0.198738	1.126259	0.027751	0.170401	-0.316536
3	-1.227954	-1.282361	-0.348706	-0.091101	-0.786982	-1.246670	-0.581694	-1.062029	-0.634544	-1.003906	...	-1.068032	-1.016934	-0.528061	-1.040440	-1.046147	-1.470961	-0.745174	-1.388803	-1.123688	-1.281195
4	1.413906	-0.665308	0.174772	-1.489348	0.136021	-0.189747	1.742744	-0.175699	-0.658708	-1.268388	...	1.678884	1.189555	1.849212	1.204989	1.239743	0.170833	-0.594651	0.733358	0.679998	0.834019
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
283	1.362527	0.548693	-0.452067	0.479657	1.017407	1.057167	0.645644	-0.071990	0.511509	1.702605	...	1.249576	0.633945	0.006536	0.702137	0.580234	0.808887	-0.201071	0.456739	0.590438	0.715464
284	1.640237	0.853813	0.206650	0.157532	0.235883	0.232024	0.636274	-0.335934	-0.251308	0.120748	...	0.052737	0.081259	0.633033	0.414053	0.091297	0.486180	-0.432946	0.103605	0.213951	0.270504
285	-0.452405	0.072582	-0.486750	0.629267	0.593868	0.962744	0.169955	-0.358596	1.198168	2.032550	...	2.685295	1.925996	1.490167	1.616368	1.977214	0.447715	-0.796477	0.103289	0.961697	1.419124
286	-0.838252	1.101214	-0.488560	3.496825	0.672860	1.023646	0.052410	-0.143559	2.901355	1.106215	...	1.242653	3.245456	3.000349	2.888558	3.061154	1.568732	0.083614	1.262971	2.177277	2.594924
287	-0.797470	1.175926	-0.488609	3.638154	0.640503	1.086241	0.082255	-0.104630	3.293287	1.086662	...	1.142697	3.332865	2.933063	2.994429	3.107461	1.602977	0.110642	1.298355	2.232620	2.645829

288 rows × 14328 columns

[8]:
```
import numpy as np
data_log = np.log(filtered_data+1)
print("Log-transformed Data:")
data_log
```

Log-transformed Data:

[8]:

	ENSMUSG00000000001	ENSMUSG00000000028	ENSMUSG00000000031	ENSMUSG00000000037	ENSMUSG00000000056	ENSMUSG00000000078	ENSMUSG00000000085	ENSMUSG00000000088	ENSMUSG00000000094	ENSMUSG00000000120	...	ERCC.00164	ERCC.00165	ERCC.00168	ERCC.00170	ERCC.00171	Ambiguous	No_feature	Not_aligned	Too_low_aQual	Aligned
G1_cell1_count	5.580040	5.957189	2.063839	3.842878	4.032402	4.831875	2.789743	5.637894	4.295554	1.648280	...	2.574640	7.785348	1.148718	6.238373	11.065630	11.657558	13.398489	13.413494	11.792031	15.446574
G1_cell2_count	5.625257	5.981420	2.332484	3.873058	4.002060	4.837166	2.815636	5.616083	4.290722	1.688439	...	2.559914	7.792792	1.261887	6.242546	11.067612	11.672710	13.414297	13.428078	11.804835	15.456969
G1_cell3_count	6.133748	6.170417	1.174151	4.360975	4.572606	4.513696	3.404481	5.351596	3.168675	3.058899	...	1.597830	7.396170	1.639076	5.989722	10.649063	11.653158	14.092752	13.372087	11.778873	15.293355
G1_cell4_count	5.337499	5.524792	1.175523	4.335693	3.744584	4.252625	3.539983	4.669086	2.438222	2.160621	...	0.862538	6.671277	2.004710	5.016544	9.964339	11.036220	13.449351	12.766570	11.052052	14.633733
G1_cell5_count	6.448649	5.857132	2.452311	3.230120	4.361326	5.060010	4.942316	5.454875	2.377137	0.674121	...	3.259496	8.334344	3.501924	6.661542	11.548714	11.778683	13.519460	13.576046	11.964036	15.748258
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
S_cell92_count	6.435517	6.299443	0.461302	4.577128	4.725654	5.562940	4.502467	5.516611	3.775947	4.348792	...	3.106268	8.106093	2.581408	6.462213	11.288198	11.964056	13.682463	13.500965	11.933879	15.709870
S_cell93_count	6.504528	6.385473	2.495285	4.448090	4.409927	5.259735	4.497738	5.351301	3.100794	3.617044	...	2.487041	7.811653	2.995892	6.327111	11.038929	11.874592	13.589674	13.396118	11.796108	15.551015
S_cell94_count	5.806398	6.148403	0.030740	4.631830	4.566980	5.532553	4.228629	5.335747	4.141915	4.451564	...	3.546958	8.573750	3.377224	6.799430	11.777010	11.863374	13.424287	13.396019	12.053363	15.918733
S_cell95_count	5.600328	6.450174	0.002256	5.335079	4.598547	5.552258	4.147593	5.474418	4.707331	4.131074	...	3.103594	8.896888	3.821017	7.134703	12.039378	12.147629	13.785737	13.705744	12.367258	16.192545
S_cell96_count	5.624227	6.468917	0.001482	5.359664	4.585737	5.572114	4.168799	5.497589	4.802079	4.123068	...	3.064169	8.915015	3.804919	7.158133	12.049188	12.155159	13.795010	13.713838	12.379449	16.202862

288 rows × 14328 columns

Task 2: MAGIC

[9]:
```
# !pip install magic-impute
```

[10]:
```
import magic
magic_operator=magic.MAGIC()
data_magic =magic_operator.fit_transform(data_log, genes='all_genes')
```

```
Calculating MAGIC...
  Running MAGIC on 288 cells and 14328 genes.
  Calculating graph and diffusion operator...
    Calculating PCA...
    Calculated PCA in 0.32 seconds.
    Calculating KNN search...
    Calculated KNN search in 0.02 seconds.
    Calculating affinities...
    Calculated affinities in 0.01 seconds.
  Calculated graph and diffusion operator in 0.37 seconds.
  Running MAGIC with `solver='exact'` on 14328-dimensional data may take a long time. Consider denoising specific genes with `genes=<list-like>` or using `solver='approximate'`.
  Calculating imputation...
Calculated MAGIC in 0.39 seconds.
```

[11]:
```
data_magic
```

[11]:

	ENSMUSG00000000001	ENSMUSG00000000028	ENSMUSG00000000031	ENSMUSG00000000037	ENSMUSG00000000056	ENSMUSG00000000078	ENSMUSG00000000085	ENSMUSG00000000088	ENSMUSG00000000094	ENSMUSG00000000120	...	ERCC.00164	ERCC.00165	ERCC.00168	ERCC.00170	ERCC.00171	Ambiguous	No_feature	Not_aligned	Too_low_aQual	Alig
G1_cell1_count	5.702818	6.041416	2.491418	3.922101	3.985893	4.812679	2.964486	5.571891	4.199013	1.842740	...	2.521490	7.778821	1.332884	6.232240	11.046750	11.688494	13.495595	13.439617	11.814072	15.45
G1_cell2_count	5.711477	6.046713	2.517635	3.928738	3.982948	4.810340	2.977286	5.566199	4.189344	1.855492	...	2.518148	7.778156	1.339450	6.231899	11.045267	11.690358	13.503761	13.441138	11.815542	15.46
G1_cell3_count	6.104974	6.144834	1.440200	4.350960	4.428756	4.484971	3.400436	5.305131	3.049103	2.955975	...	1.651964	7.390888	1.648582	5.965917	10.649188	11.640026	14.082174	13.363263	11.769436	15.28
G1_cell4_count	5.239944	5.391878	1.149494	4.247360	3.653786	4.109561	3.422860	4.603623	2.389548	2.109147	...	0.864591	6.553364	1.984456	4.914008	9.857663	10.930637	13.340891	12.653245	10.963608	14.52
G1_cell5_count	6.377036	5.918715	2.528448	3.358129	4.309755	5.027801	4.740179	5.391986	2.393653	0.782965	...	3.263321	8.290853	3.425190	6.641720	11.505075	11.768587	13.523464	13.568288	11.966043	15.72
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
S_cell92_count	6.358779	6.310809	0.652419	4.563147	4.682203	5.551652	4.482343	5.465070	3.773524	4.266561	...	3.091925	8.135578	2.720175	6.496709	11.323516	11.943711	13.642370	13.478208	11.934464	15.71
S_cell93_count	6.454540	6.374689	2.336624	4.431324	4.339532	5.269696	4.531925	5.338520	3.085856	3.677251	...	2.592089	7.835647	2.903567	6.320761	11.047386	11.872996	13.595609	13.398356	11.802045	15.55
S_cell94_count	5.965255	6.303344	0.040051	4.684974	4.703823	5.666728	4.373411	5.436205	4.223554	4.424175	...	3.369189	8.489525	3.206038	6.748328	11.661220	11.958199	13.605384	13.499426	12.074724	15.89
S_cell95_count	5.663452	6.450815	0.004101	5.237546	4.628483	5.613858	4.227748	5.505535	4.678036	4.209838	...	3.117243	8.835547	3.772450	7.073863	11.976482	12.133344	13.794842	13.691547	12.329287	16.150
S_cell96_count	5.659481	6.453388	0.003826	5.247293	4.626520	5.611943	4.224937	5.505373	4.684226	4.205695	...	3.111372	8.840531	3.775927	7.079169	11.981011	12.136479	13.797880	13.694876	12.333538	16.15

288 rows × 14328 columns

[12]:
```
# Task 3: t-SNE
```

[13]:
```
from sklearn.manifold import TSNE
import matplotlib.pyplot as plt
```

[14]:
```
tsne = TSNE(n_components=2)
data_tsne=tsne.fit_transform(data_log)
data_magic_tsne = tsne.fit_transform(data_magic)
```

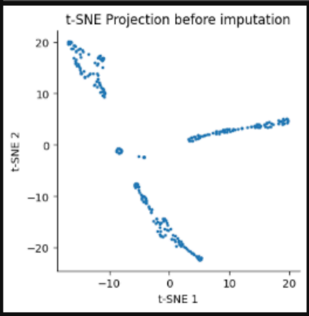
[ ]:

Task 3: t-SNE

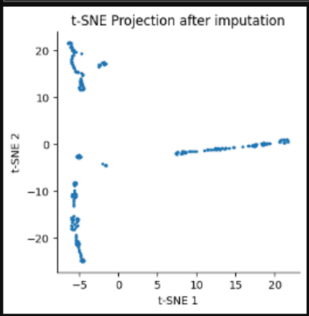
[15]:
```
from sklearn.manifold import TSNE
import matplotlib.pyplot as plt

tsne = TSNE(n_components=2)
data_tsne =tsne.fit_transform(data_log)
data_magic_tsne = tsne.fit_transform(data_magic)
```

[16]:
```
plt.scatter(data_tsne[:,0],data_tsne[:,1],s=3)
plt.title('t-SNE Projection before imputation')
plt.xlabel('t-SNE 1')
plt.ylabel('t-SNE 2')
plt.show()
```



[17]:
```
plt.scatter(data_magic_tsne[:,0],data_magic_tsne[:,1],s=3)
plt.title('t-SNE Projection after imputation')
plt.xlabel('t-SNE 1')
plt.ylabel('t-SNE 2')
plt.show()
```



Task 4: Dimension

[18]:
```
from sklearn.decomposition import PCA

pca = PCA(n_components=2)
data_pca =pca.fit_transform(data_log)

import umap
reducer =  umap.UMAP(n_neighbors=5)

data_umap = reducer.fit_transform(data_log)
```

```
/opt/conda/lib/python3.11/site-packages/sklearn/manifold/_spectral_embedding.py:329: UserWarning: Graph is not fully connected, spectral embedding may not work as expected.
  warnings.warn(
```


Task 4: Dimension

[18]:
```
from sklearn.decomposition import PCA

pca = PCA(n_components=2)
data_pca =pca.fit_transform(data_log)

import umap
reducer =  umap.UMAP(n_neighbors=5)

data_umap = reducer.fit_transform(data_log)
```

```
/opt/conda/lib/python3.11/site-packages/sklearn/manifold/_spectral_embedding.py:329: UserWarning: Graph is not fully connected, spectral embedding may not work as expected.
  warnings.warn(
```


Task 5: Clustering

[19]:
```
from sklearn.cluster import KMeans

kmeans = KMeans(n_clusters=10)
kmeans_labels = kmeans.fit_predict(data_umap)

from sklearn.cluster import AgglomerativeClustering

model =AgglomerativeClustering(n_clusters =10, metric = 'euclidean', linkage='ward')

hierarchical_labels = model.fit_predict(data_umap)
```

Task 6: ARI

[20]:
```
from sklearn.metrics import adjusted_rand_score
import pandas as pd

ground_truth = pd.read_csv("data/data2_mesc/sample_cluster_ref_filtered.txt", sep='\t', header=None, index_col=0)


ground_truth.index = ground_truth.index.astype(str)
filtered_data.index = filtered_data.index.astype(str)

filter_truth = ground_truth[ground_truth.index.isin(filtered_data.index)]

truth_labels = filter_truth.iloc[:, 0].values.flatten()

ari_kmeans = adjusted_rand_score(truth_labels, kmeans_labels)
ari_hierarchical = adjusted_rand_score(truth_labels, hierarchical_labels)

ari_kmeans, ari_hierarchical
```

[20]:
```
(0.24332549678853352, 0.3809128475299934)
```

§ Embryo cortex Dataset

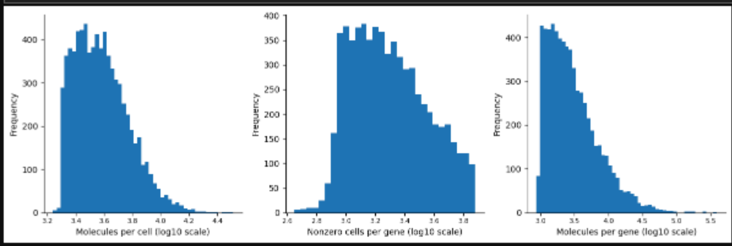
Task 1: Filter and Normalization

[4]:
```
# !pip install palantir
```

[5]:
```
import palantir

fig, ax = palantir.plot.plot_molecules_per_cell_and_gene(data.T)

filtered_data = palantir.preprocess.filter_counts_data(data.T, cell_min_molecules=1000, genes_min_cells=10)
```



[6]:
```
from sklearn.preprocessing import MinMaxScaler, StandardScaler

scaler_minmax = MinMaxScaler()

data_minmax= pd.DataFrame(scaler_minmax.fit_transform(filtered_data), columns=filtered_data.columns)

print("Min-Max Scaling:")
data_minmax
```

```
Min-Max Scaling:
```

[6]:

	0610007N19Rik	0610007P14Rik	0610009B22Rik	0610009D07Rik	0610009O20Rik	0610010F05Rik	0610011F06Rik	0610030E20Rik	0610037L13Rik	1110001A16Rik	...	l7Rn6	mt-Co1	mt-Cytb	mt-Nd1	mt-Nd2	mt-Nd4	mt-Nd5	mt-Nd6	mt-Rnr1	mt-Rnr2
0	0.0	0.166667	0.0	0.000000	0.0	0.000000	0.000	0.0	0.000000	0.000000	...	0.125	0.024390	0.072464	0.075949	0.000000	0.000000	0.014286	0.000000	0.000000	0.030973
1	0.0	0.000000	0.0	0.000000	0.0	0.000000	0.000	0.0	0.000000	0.000000	...	0.000	0.000000	0.094203	0.088608	0.036364	0.035714	0.000000	0.000000	0.000000	0.026549
2	0.0	0.000000	0.0	0.000000	0.0	0.066667	0.000	0.0	0.000000	0.000000	...	0.125	0.000000	0.086957	0.113924	0.072727	0.125000	0.100000	0.142857	0.083333	0.053097
3	0.0	0.000000	0.0	0.000000	0.0	0.066667	0.000	0.0	0.000000	0.000000	...	0.000	0.024390	0.130435	0.101266	0.090909	0.089286	0.014286	0.000000	0.000000	0.048673
4	0.0	0.000000	0.0	0.153846	0.0	0.000000	0.125	0.0	0.000000	0.000000	...	0.000	0.024390	0.043478	0.063291	0.054545	0.053571	0.000000	0.142857	0.000000	0.044248
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
7596	0.0	0.000000	0.0	0.461538	0.0	0.066667	0.000	0.0	0.000000	0.285714	...	0.000	0.048780	0.188406	0.253165	0.127273	0.107143	0.057143	0.142857	0.250000	0.132743
7597	0.0	0.000000	0.0	0.000000	0.5	0.000000	0.000	0.0	0.000000	0.142857	...	0.000	0.000000	0.007246	0.025316	0.018182	0.000000	0.014286	0.000000	0.083333	0.181416
7598	0.0	0.000000	0.0	0.076923	0.0	0.066667	0.000	0.0	0.000000	0.142857	...	0.125	0.024390	0.028986	0.063291	0.000000	0.000000	0.014286	0.142857	0.041667	0.088496
7599	0.0	0.000000	0.0	0.000000	0.0	0.000000	0.000	0.0	0.000000	0.000000	...	0.000	0.024390	0.072464	0.101266	0.018182	0.035714	0.028571	0.000000	0.041667	0.168142
7600	0.0	0.000000	0.0	0.153846	0.0	0.000000	0.000	0.0	0.142857	0.000000	...	0.125	0.073171	0.043478	0.075949	0.036364	0.035714	0.042857	0.000000	0.000000	0.119469

7601 rows × 6470 columns

[7]:
```
scaler_zscore = StandardScaler()
data_zscore = pd.DataFrame(scaler_zscore.fit_transform(filtered_data),columns=filtered_data.columns)
print("\nZ-Score Normalization:")
data_zscore
```

```
Z-Score Normalization:
```

[7]:

	0610007N19Rik	0610007P14Rik	0610009B22Rik	0610009D07Rik	0610009O20Rik	0610010F05Rik	0610011F06Rik	0610030E20Rik	0610037L13Rik	1110001A16Rik	...	l7Rn6	mt-Co1	mt-Cytb	mt-Nd1	mt-Nd2	mt-Nd4	mt-Nd5	mt-Nd6	mt-Rnr1	mt-Rnr2
0	-0.282364	0.958239	-0.535158	-1.032482	-0.355989	-0.663598	-0.503545	-0.371764	-0.583722	-0.537763	...	0.405438	-0.626387	-0.756392	-0.696496	-1.182838	-1.317843	-0.741729	-0.449034	-0.713510	-0.919172
1	-0.282364	-0.527295	-0.535158	-1.032482	-0.355989	-0.663598	-0.503545	-0.371764	-0.583722	-0.537763	...	-0.703499	-1.036408	-0.507933	-0.570999	-0.649768	-0.883994	-1.022178	-0.449034	-0.713510	-0.986133
2	-0.282364	-0.527295	-0.535158	-1.032482	-0.355989	0.176650	-0.503545	-0.371764	-0.583722	-0.537763	...	0.405438	-1.036408	-0.590753	-0.320007	-0.116699	0.200629	0.940969	1.127460	0.516562	-0.584368
3	-0.282364	-0.527295	-0.535158	-1.032482	-0.355989	0.176650	-0.503545	-0.371764	-0.583722	-0.537763	...	-0.703499	-0.626387	-0.093835	-0.445503	0.149836	-0.233220	-0.741729	-0.449034	-0.713510	-0.651329
4	-0.282364	-0.527295	-0.535158	0.237918	-0.355989	-0.663598	1.020727	-0.371764	-0.583722	-0.537763	...	-0.703499	-0.626387	-1.087671	-0.821992	-0.383234	-0.667069	-1.022178	1.127460	-0.713510	-0.718290
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
7596	-0.282364	-0.527295	-0.535158	2.778719	-0.355989	0.176650	-0.503545	-0.371764	-0.583722	2.161170	...	-0.703499	-0.216366	0.568722	1.060454	0.682905	-0.016296	0.099620	1.127460	2.976707	0.620927
7597	-0.282364	-0.527295	-0.535158	-1.032482	4.528259	-0.663598	-0.503545	-0.371764	-0.583722	0.811704	...	-0.703499	-1.036408	-1.501769	-1.198481	-0.916303	-1.317843	-0.741729	-0.449034	0.516562	1.357497
7598	-0.282364	-0.527295	-0.535158	-0.397282	-0.355989	0.176650	-0.503545	-0.371764	-0.583722	0.811704	...	0.405438	-0.626387	-1.253310	-0.821992	-1.182838	-1.317843	-0.741729	1.127460	-0.098474	-0.048681
7599	-0.282364	-0.527295	-0.535158	-1.032482	-0.355989	-0.663598	-0.503545	-0.371764	-0.583722	-0.537763	...	-0.703499	-0.626387	-0.756392	-0.445503	-0.916303	-0.883994	-0.461279	-0.449034	-0.098474	1.156614
7600	-0.282364	-0.527295	-0.535158	0.237918	-0.355989	-0.663598	-0.503545	-0.371764	0.698241	-0.537763	...	0.405438	0.193656	-1.087671	-0.696496	-0.649768	-0.883994	-0.180829	-0.449034	-0.713510	0.420045

7601 rows × 6470 columns

[8]:
```
import numpy as np
data_log = np.log(filtered_data+1)
print("Log-transformed Data:")
data_log
```

```
Log-transformed Data:
```

[8]:

	0610007N19Rik	0610007P14Rik	0610009B22Rik	0610009D07Rik	0610009O20Rik	0610010F05Rik	0610011F06Rik	0610030E20Rik	0610037L13Rik	1110001A16Rik	...	l7Rn6	mt-Co1	mt-Cytb	mt-Nd1	mt-Nd2	mt-Nd4	mt-Nd5	mt-Nd6	mt-Rnr1	mt-Rnr2
e14.WT10_AAAAATCTCTCC	0.0	0.693147	0.0	0.000000	0.000000	0.000000	0.000000	0.0	0.000000	0.000000	...	0.693147	0.693147	2.397895	1.945910	0.000000	0.000000	0.693147	0.000000	0.000000	2.079442
e14.WT10_AAAACACATTCC	0.0	0.000000	0.0	0.000000	0.000000	0.000000	0.000000	0.0	0.000000	0.000000	...	0.000000	0.000000	2.639057	2.079442	1.098612	1.098612	0.000000	0.000000	0.000000	1.945910
e14.WT10_AAAACCGTGGAT	0.0	0.000000	0.0	0.000000	0.000000	0.693147	0.000000	0.0	0.000000	0.000000	...	0.693147	0.000000	2.564949	2.302585	1.609438	2.079442	2.079442	0.693147	1.098612	2.564949
e14.WT10_AAAACGCGCCGA	0.0	0.000000	0.0	0.000000	0.000000	0.693147	0.000000	0.0	0.000000	0.000000	...	0.000000	0.693147	2.944439	2.197225	1.791759	1.791759	0.693147	0.000000	0.000000	2.484907
e14.WT10_AAAATGGACTCA	0.0	0.000000	0.0	1.098612	0.000000	0.000000	0.693147	0.0	0.000000	0.000000	...	0.000000	0.693147	1.945910	1.791759	1.386294	1.386294	0.000000	0.693147	0.000000	2.397895
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
e14.WT9.2_TTTTGGTCCCGN	0.0	0.000000	0.0	1.945910	0.000000	0.693147	0.000000	0.0	0.000000	1.098612	...	0.000000	1.098612	3.295837	3.044522	2.079442	1.945910	1.609438	0.693147	1.945910	3.433987
e14.WT9.2_TTTTGTCAGTCT	0.0	0.000000	0.0	0.000000	1.098612	0.000000	0.000000	0.0	0.000000	0.693147	...	0.000000	0.000000	0.693147	1.098612	0.693147	0.000000	0.693147	0.000000	1.098612	3.737670
e14.WT9.2_TTTTTAGATGTN	0.0	0.000000	0.0	0.693147	0.000000	0.693147	0.000000	0.0	0.000000	0.693147	...	0.693147	0.693147	1.609438	1.791759	0.000000	0.000000	0.693147	0.693147	0.693147	3.044522
e14.WT9.2_TTTTTCATCGGG	0.0	0.000000	0.0	0.000000	0.000000	0.000000	0.000000	0.0	0.000000	0.000000	...	0.000000	0.693147	2.397895	2.197225	0.693147	1.098612	1.098612	0.000000	0.693147	3.663562
e14.WT9.2_TTTTTTTACTTG	0.0	0.000000	0.0	1.098612	0.000000	0.000000	0.000000	0.0	0.693147	0.000000	...	0.693147	1.386294	1.945910	1.945910	1.098612	1.098612	1.386294	0.000000	0.000000	3.332205

7601 rows × 6470 columns

Task 2: MAGIC

[9]:
```
# !pip install magic-impute
```

[10]:
```
import magic
magic_operator=magic.MAGIC()
data_magic =magic_operator.fit_transform(data_log, genes='all_genes')
```

```
Calculating MAGIC...
  Running MAGIC on 7601 cells and 6470 genes.
  Calculating graph and diffusion operator...
    Calculating PCA...
    Calculated PCA in 1.25 seconds.
    Calculating KNN search...
    Calculated KNN search in 5.99 seconds.
    Calculating affinities...
    Calculated affinities in 5.30 seconds.
  Calculated graph and diffusion operator in 12.66 seconds.
  Running MAGIC with `solver='exact'` on 6470-dimensional data may take a long time. Consider denoising specific genes with `genes=<list-like>` or using `solver='approximate'`.
  Calculating imputation...
  Calculated imputation in 2.97 seconds.
Calculated MAGIC in 15.73 seconds.
```

[11]:
```
data_magic
```

[11]:

	0610007N19Rik	0610007P14Rik	0610009B22Rik	0610009D07Rik	0610009O20Rik	0610010F05Rik	0610011F06Rik	0610030E20Rik	0610037L13Rik	1110001A16Rik	...	l7Rn6	mt-Co1	mt-Cytb	mt-Nd1	mt-Nd2	mt-Nd4	mt-Nd5	mt-Nd6	mt-Rnr1	mt-Rnr2
e14.WT10_AAAAATCTCTCC	0.109840	0.206996	0.115742	0.643891	0.099529	0.067080	0.205552	0.135005	0.286174	0.097304	...	0.374501	0.725611	2.347440	2.015200	1.148511	1.173602	1.039130	0.121857	0.372056	2.468206
e14.WT10_AAAACACATTCC	0.031371	0.092785	0.143412	0.550419	0.045622	0.326628	0.161662	0.058085	0.154515	0.129869	...	0.209061	0.803348	2.523546	1.910459	1.070574	1.421690	0.997677	0.130361	0.368407	2.628901
e14.WT10_AAAACCGTGGAT	0.004517	0.131215	0.161768	0.705158	0.055906	0.497444	0.192594	0.058942	0.207240	0.144558	...	0.275756	0.939778	2.769773	2.206953	1.385610	1.695058	1.169969	0.152305	0.445458	2.762901
e14.WT10_AAAACGCGCCGA	0.003251	0.181579	0.177131	0.748381	0.063097	0.480947	0.127139	0.064207	0.154336	0.179146	...	0.259073	0.949996	2.805552	2.192233	1.420963	1.619639	1.058278	0.100979	0.442977	2.602655
e14.WT10_AAAATGGACTCA	0.009167	0.157340	0.075790	0.488262	0.036078	0.324670	0.109399	0.044952	0.116643	0.104344	...	0.168031	0.741609	2.358183	1.911669	1.040598	1.411182	0.861593	0.186776	0.440366	2.729482
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
e14.WT9.2_TTTTGGTCCCGN	0.021247	0.214575	0.261514	0.842657	0.120718	0.750286	0.239059	0.079195	0.231910	0.273649	...	0.414470	1.200818	3.133367	2.570247	1.668067	2.085694	1.491706	0.266827	0.657871	3.064519
e14.WT9.2_TTTTGTCAGTCT	0.025193	0.130666	0.110510	0.360292	0.104993	0.065555	0.049745	0.148393	0.159844	0.463985	...	0.214569	0.984493	2.193938	1.823790	1.106785	1.336605	0.950563	0.063014	0.294444	2.711100
e14.WT9.2_TTTTTAGATGTN	0.162792	0.154091	0.152496	0.672807	0.073537	0.087037	0.202186	0.062100	0.253371	0.148239	...	0.324199	0.642162	2.289271	1.928464	1.030899	1.246740	0.985313	0.086245	0.397821	2.531839
e14.WT9.2_TTTTTCATCGGG	0.078222	0.115470	0.145186	0.642985	0.061533	0.283289	0.130275	0.102586	0.192536	0.122243	...	0.229061	0.765094	2.489140	2.004241	1.098735	1.394066	1.018053	0.089507	0.422404	2.660914
e14.WT9.2_TTTTTTTACTTG	0.008413	0.068521	0.111652	0.435362	0.042058	0.345261	0.139451	0.057752	0.161401	0.078887	...	0.194914	0.825207	2.405163	1.925775	1.183573	1.411600	1.150732	0.119312	0.530925	2.843672

7601 rows × 6470 columns

[12]:
```
# Task 3: t-SNE
```

[13]:
```
from sklearn.manifold import TSNE
import matplotlib.pyplot as plt
```

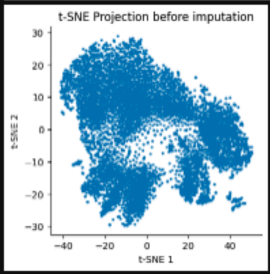
[14]:
```
tsne = TSNE(n_components=2)
data_tsne=tsne.fit_transform(data_log)
data_magic_tsne = tsne.fit_transform(data_magic)
```

Task 3: t-SNE

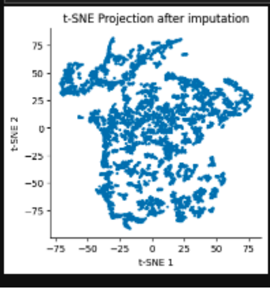
[15]:
```
from sklearn.manifold import TSNE
import matplotlib.pyplot as plt

tsne = TSNE(n_components=2)
data_tsne =tsne.fit_transform(data_log)
data_magic_tsne = tsne.fit_transform(data_magic)
```

[16]:
```
plt.scatter(data_tsne[:,0],data_tsne[:,1],s=3)
plt.title('t-SNE Projection before imputation')
plt.xlabel('t-SNE 1')
plt.ylabel('t-SNE 2')
plt.show()
```



[17]:
```
plt.scatter(data_magic_tsne[:,0],data_magic_tsne[:,1],s=3)
plt.title('t-SNE Projection after imputation')
plt.xlabel('t-SNE 1')
plt.ylabel('t-SNE 2')
plt.show()
```



Task 4: Dimension

[18]:
```
from sklearn.decomposition import PCA

pca = PCA(n_components=2)
data_pca =pca.fit_transform(data_log)

import umap
reducer =  umap.UMAP(n_neighbors=5)

data_umap = reducer.fit_transform(data_log)
```

Task 5: Clustering

[19]:
```
from sklearn.cluster import KMeans

kmeans = KMeans(n_clusters=10)
kmeans_labels = kmeans.fit_predict(data_umap)

from sklearn.cluster import AgglomerativeClustering

model =AgglomerativeClustering(n_clusters =10, metric = 'euclidean', linkage='ward')

hierarchical_labels = model.fit_predict(data_umap)
```

Task 6: ARI

•[25]:
```
from sklearn.metrics import adjusted_rand_score
import pandas as pd

ground_truth = pd.read_csv("data/data3_embryo_cortex/sample_cluster_ref_filtered.txt", sep=' ', header=None, index_col=0)


ground_truth.index = ground_truth.index.astype(str)
filtered_data.index = filtered_data.index.astype(str)
print(ground_truth.head())
filter_truth = ground_truth[ground_truth.index.isin(filtered_data.index)]

truth_labels = filter_truth.iloc[:, 0].values.flatten()

ari_kmeans = adjusted_rand_score(truth_labels, kmeans_labels)
ari_hierarchical = adjusted_rand_score(truth_labels, hierarchical_labels)

ari_kmeans, ari_hierarchical
```

```
ground_truth 的形状: (7601, 1)
ground_truth 的前几行:
                                1
0
e14.WT10_AAAAATCTCTCC          RG1
e14.WT10_AAAACACATTCC    LayerV-VI
e14.WT10_AAAACCGTGGAT    LayerV-VI
e14.WT10_AAAACGCGCCGA    LayerV-VI
e14.WT10_AAAATGGACTCA  Striatal_inh2
```

[25]:
```
(0.3384487409938317, 0.3289003480647635)
```