In [6]:
```
import scanpy as sc
import harmonypy as hmss
import pandas as pd
import anndata as ad
import numpy as np

%config InlineBackend.figure_format = 'retina'
%matplotlib inline

import matplotlib as mpl
import matplotlib.pyplot as plt
#import seaborn as sns

DPI=300
FONTSIZE=20 #42

sc.settings.verbosity = 3  # verbosity: errors (0), warnings (1), info (2), hints (
sc.logging.print_header()
sc.settings.set_figure_params(scanpy = True, dpi=100, transparent=True, vector_frie
from matplotlib import rcParams
rcParams['pdf.fonttype'] = 42
```

In [7]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [8]:
```
file_list = [
    'GSM2877132_SIGAH1.h5ad', 'GSM2877129_SIGAD1.h5ad', 'GSM2877130_SIGAF1.h5ad',
    'GSM2877127_SIGAB1.h5ad', 'GSM2877128_SIGAC1.h5ad', 'GSM2877131_SIGAG1.h5ad',
    'GSM2877134_SIGAH8.h5ad', 'GSM2877133_SIGAG8.h5ad'
]

adatas = []
for f in file_list:
    adatas.append(sc.read(f))

adata = ad.concat(adatas, label='batch', keys=file_list)
```

In [9]:
```
# Split the batch column into 'ID' and 'sample'
adata.obs[['ID', 'sample']] = adata.obs['batch'].str.replace('.h5ad', '').str.split
adata.obs
```

Out[9]:

Cell	n_genes_by_counts	total_counts	batch	
AAACCTGAGTTAACGA-1	3064	9185.0	GSM2877132_SIGAH1.h5ad	GSM287
AAACCTGCAACGCACC-1	1711	3375.0	GSM2877132_SIGAH1.h5ad	GSM287
AAACCTGCAATTCCTT-1	2635	6827.0	GSM2877132_SIGAH1.h5ad	GSM287
AAACCTGCACCTTGTC-1	5003	24419.0	GSM2877132_SIGAH1.h5ad	GSM287
AAACCTGCATCCAACA-1	1841	4668.0	GSM2877132_SIGAH1.h5ad	GSM287
...	...	...	...	
TTTGTCAGTAATTGGA-1	2742	10665.0	GSM2877133_SIGAG8.h5ad	GSM287
TTTGTCAGTAGAGCTG-1	2743	10959.0	GSM2877133_SIGAG8.h5ad	GSM287
TTTGTCAGTTTGGCGC-1	2731	10838.0	GSM2877133_SIGAG8.h5ad	GSM287
TTTGTCATCCTAAGTG-1	3050	11915.0	GSM2877133_SIGAG8.h5ad	GSM287
TTTGTCATCTGTGCAA-1	3716	19686.0	GSM2877133_SIGAG8.h5ad	GSM287

61122 rows × 5 columns

Q1) Preprocess the data (the quality control has already been performed and the data has been cleaned), perform dimensionality reduction and clustering.

Dimensiality Reduction + Neighbourhood graph + UMAP

In [10]:
```
sc.tl.pca(adata)
sc.pp.neighbors(adata, n_pcs=10)
```

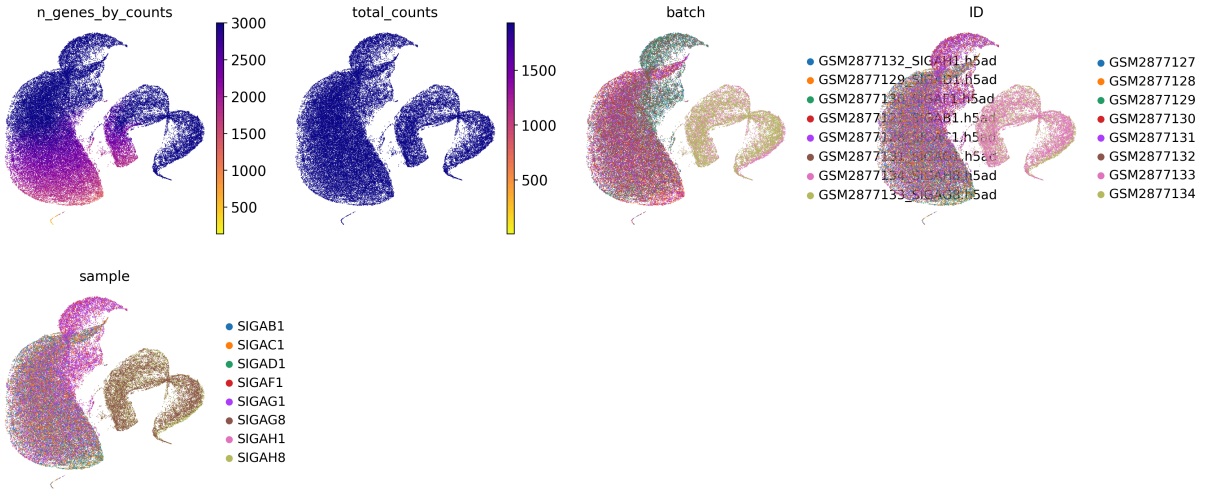
```
computing PCA
    with n_comps=50
    finished (0:01:09)
computing neighbors
    using 'X_pca' with n_pcs = 10
    finished: added to `.uns['neighbors']`
    `.obsp['distances']`, distances for each pair of neighbors
    `.obsp['connectivities']`, weighted adjacency matrix (0:00:37)
```

In [11]:
```
sc.tl.umap(adata)
```

```
computing UMAP
    finished: added
    'X_umap', UMAP coordinates (adata.obsm)
    'umap', UMAP parameters (adata.uns) (0:00:41)
```

Calculating Quality Metrics for

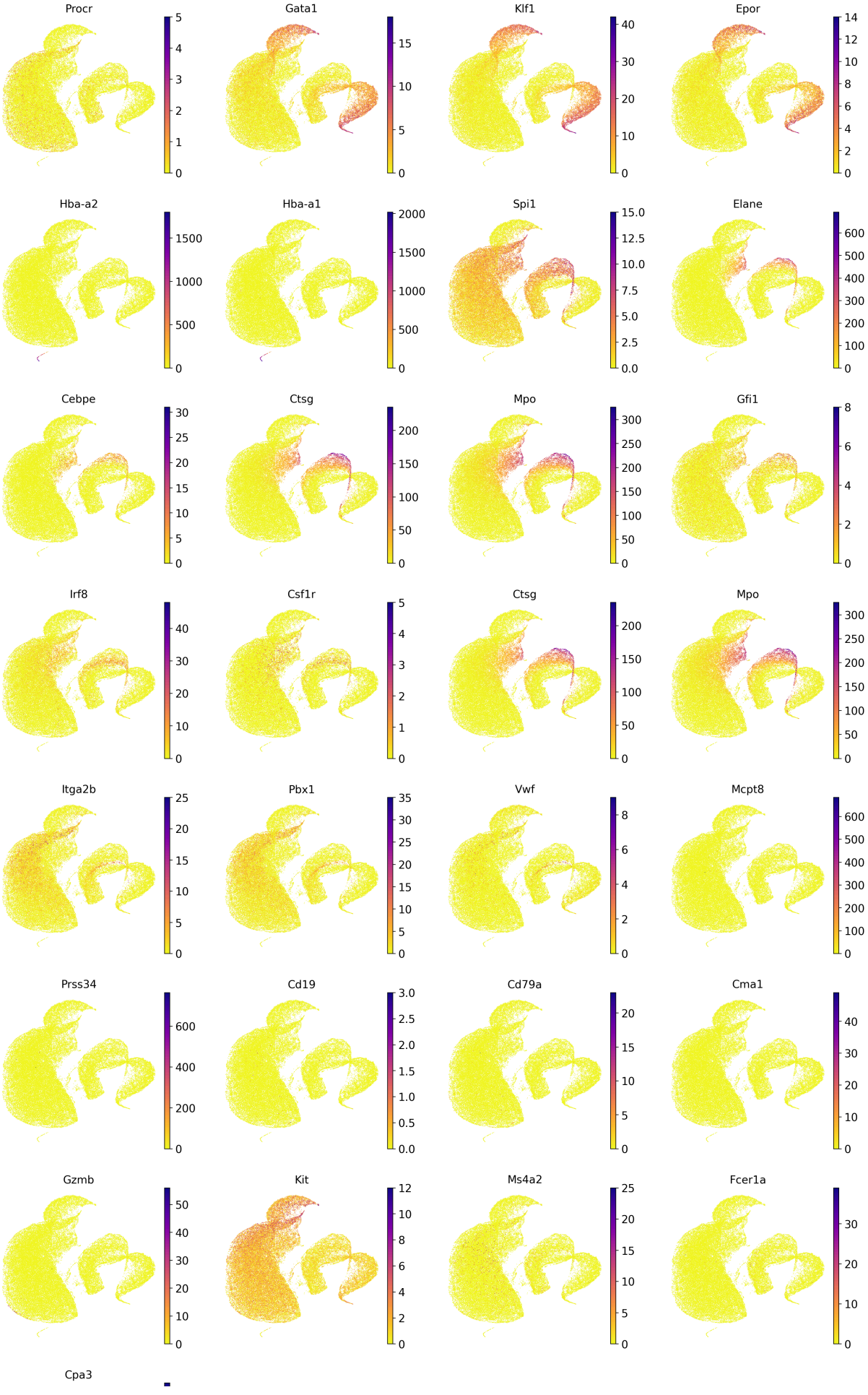
In [12]:
```
sc.pl.umap(adata,  color=["n_genes_by_counts", "total_counts", "batch", "ID", "samp
```

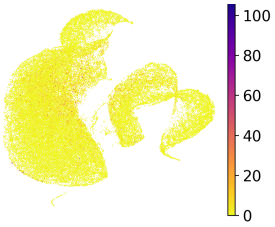


Selecting marker genes appropirate with Haematopoeisis:

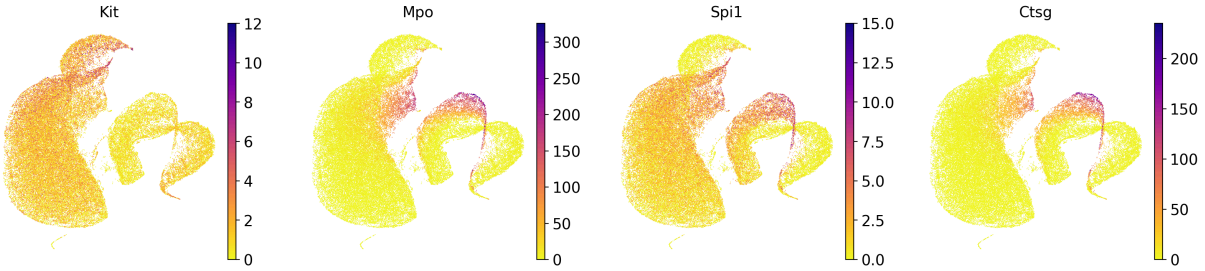
In [13]:
```
import itertools
markers = {
    "HSCs": ["Procr"],
    "Erythroids": ["Gata1", "Klf1", "Epor", "Hba-a2", "Hba-a1", "Spi1"],
    "Neutrophils": ["Elane", "Cebpe", "Ctsg", "Mpo", "Gfi1"],
    "Monocytes": ["Irf8", "Csf1r", "Ctsg", "Mpo"],
    "Megakaryocytes": ["Itga2b", "Pbx1", "Vwf"],  # CD41 = Itga2b
    "Basophils": ["Mcpt8", "Prss34"],
    "B cells": ["Cd19", "Cd79a"],
    "Mast cells": ["Cma1", "Gzmb", "Kit"],  # CD117 = Kit
    "Mast/Basophils": ["Ms4a2", "Fcer1a", "Cpa3"]  # CD203c is human-specific
}

markers = list(itertools.chain(*list(markers.values())))  # get all markers in a si
sc.pl.umap(adata, color = markers, color_map = 'plasma_r')
```


Procr
Gata1
Klf1
Epor
Hba-a2
Hba-a1
Spi1
Elane
Cebpe
Ctsg
Mpo
Gfi1
Irf8
Csf1r
Ctsg
Mpo
Itga2b
Pbx1
Vwf
Mcpt8
Prss34
Cd19
Cd79a
Cma1
Gzmb
Kit
Ms4a2
Fcer1a
Cpa3



In [14]:
```
genes = ["Kit", "Mpo", "Spi1", "Ctsg"]
sc.pl.umap(adata, color = genes, cmap = 'plasma_r')
```

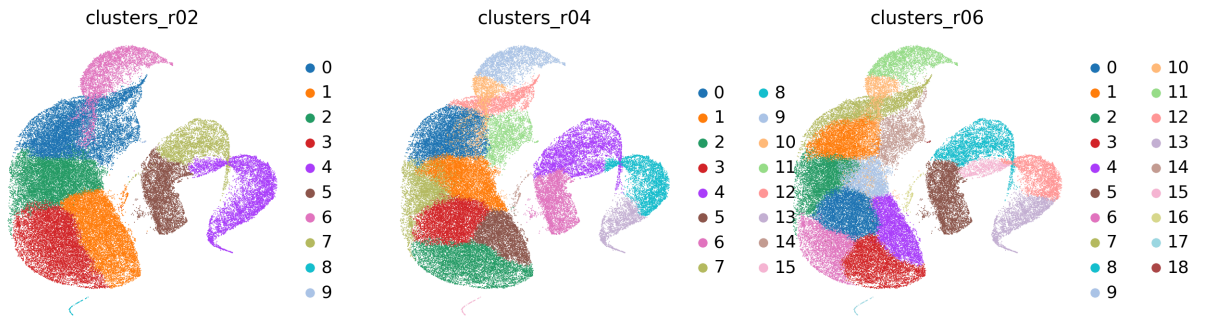


Leiden Clustering with Different Resolutions

In [15]:
```
sc.tl.leiden(adata, resolution=0.6, key_added = 'clusters_r06')
sc.tl.leiden(adata, resolution=0.4, key_added = 'clusters_r04')
sc.tl.leiden(adata, resolution=0.2, key_added = 'clusters_r02')
```

```
running Leiden clustering
    finished: found 19 clusters and added
    'clusters_r06', the cluster labels (adata.obs, categorical) (0:00:47)
running Leiden clustering
    finished: found 16 clusters and added
    'clusters_r04', the cluster labels (adata.obs, categorical) (0:00:57)
running Leiden clustering
    finished: found 10 clusters and added
    'clusters_r02', the cluster labels (adata.obs, categorical) (0:00:40)
```

In [16]:
```
sc.pl.umap(adata, color = ['clusters_r02','clusters_r04','clusters_r06'])
```



Q2) Perform a differential expression analysis using the previously gained clustering results, what are the top 5 differentially expressed genes per cluster?

Differential Gene Expression for Top 5 Genes Among Clusters

In [18]:
```
sc.pp.normalize_total(adata)
sc.pp.log1p(adata)
```

```
sc.tl.rank_genes_groups(adata, "clusters_r06", method='t-test', max_iter=1000, key_
top5_genes_r06 = pd.DataFrame(adata.uns["cluster_DEGs"]["names"]).head(5)
top5_genes_r06
```

```
normalizing counts per cell
    finished (0:00:00)
ranking genes
    finished: added to `.uns['cluster_DEGs']`
    'names', sorted np.recarray to be indexed by group ids
    'scores', sorted np.recarray to be indexed by group ids
    'logfoldchanges', sorted np.recarray to be indexed by group ids
    'pvals', sorted np.recarray to be indexed by group ids
    'pvals_adj', sorted np.recarray to be indexed by group ids (0:00:14)
```

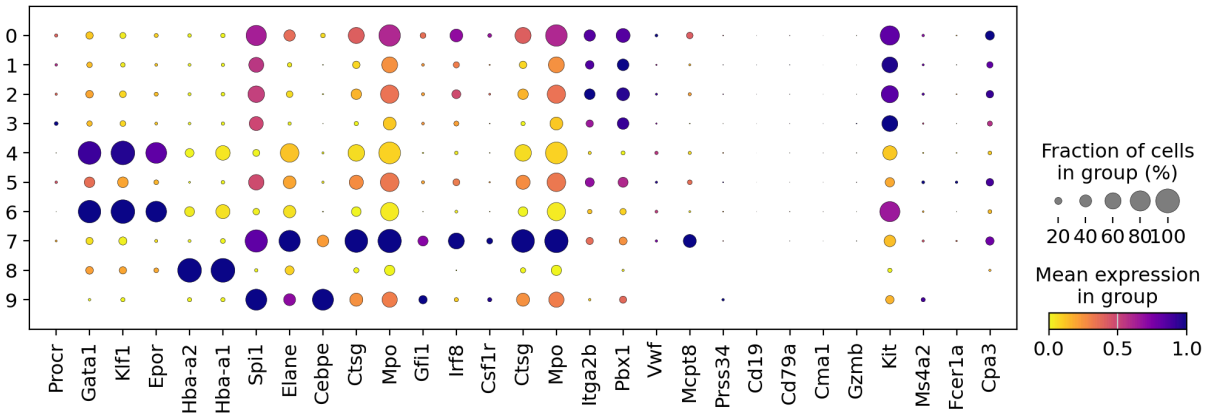
In [20]:
```
top5_genes_r06
```

Out[20]:

	0	1	2	3	4	5	6	7	8	9
0	Malat1	Stmn1	Malat1	Malat1	mt-Co1	Rpl23	Malat1	Malat1	Rps12	Tmsb4x
1	Rps27	Hmgn2	Xist	Rps27	B2m	Tpt1	Rps27	Arpc1b	Rpl30	Macroh2a1
2	Xist	Tuba1b	Stmn1	mt-Co1	H2bc4	Rps12	Sox4	Pkm	Rpl28	Plac8
3	Pabpc1	Top2a	Tuba1b	Sox4	Tmsb4x	Rps20	Xist	Macroh2a1	Rplp1	Coro1a
4	mt-Co1	Hmgb1	Pabpc1	Rsrp1	H3f3b	Rpl30	Rsrp1	Rps28	Rps20	Mpo

In [ ]:
```
Ans 2)
Cluster 0 is characterized by high expression of Malat1, Stmn1, Malat1 (again, indi
Cluster 1 shows strong expression of Rps27, Hmgn2, Xist, Rps27, and B2m.
Cluster 2 is defined by upregulation of Xist, Tuba1b, Stmn1, mt-Co1, and H2bc4.
Cluster 3 has prominent expression of Pabpc1, Top2a, Tuba1b, Sox4, and Tmsb4x.
Cluster 4 is marked by high expression of mt-Co1, Hmgb1, Pabpc1, Rsrp1, and H3f3b.
```

In [21]:
```
sc.pl.dotplot(adata, var_names = markers, color_map = 'plasma_r', groupby = 'cluste
```



The top five differentially expressed genes in each cluster, identified through the rank_genes_groups cluster 0 - Angpt1, Stmn1, Malat1, Hlf, and H2bc4 cluster 1 - Adgrl4, Hmgn2, Angpt1, Txnip, and Tcp11l2 cluster 2 - Meis1, Pkm, Xist, Ltb, and Ptms cluster 3 - Sox4, Tpm3, Rgs18, Gm19590, and Ifi27l2a cluster 4 - Gcnt2, Arpc1b, Meis1, Gcnt2, and H1f0

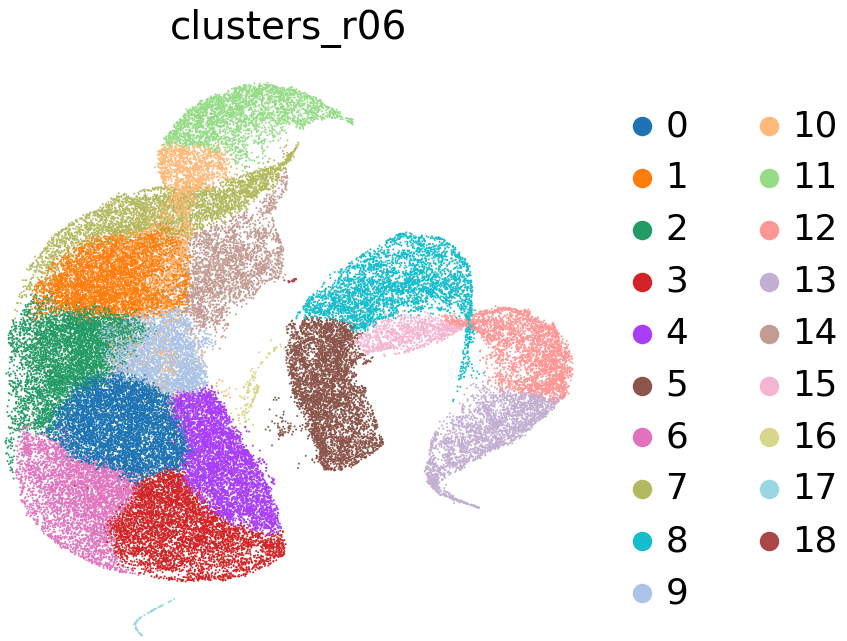
In [ ]:
```
3)      Produce a UMAP with cells coloured by their clusters and another coloured b
```

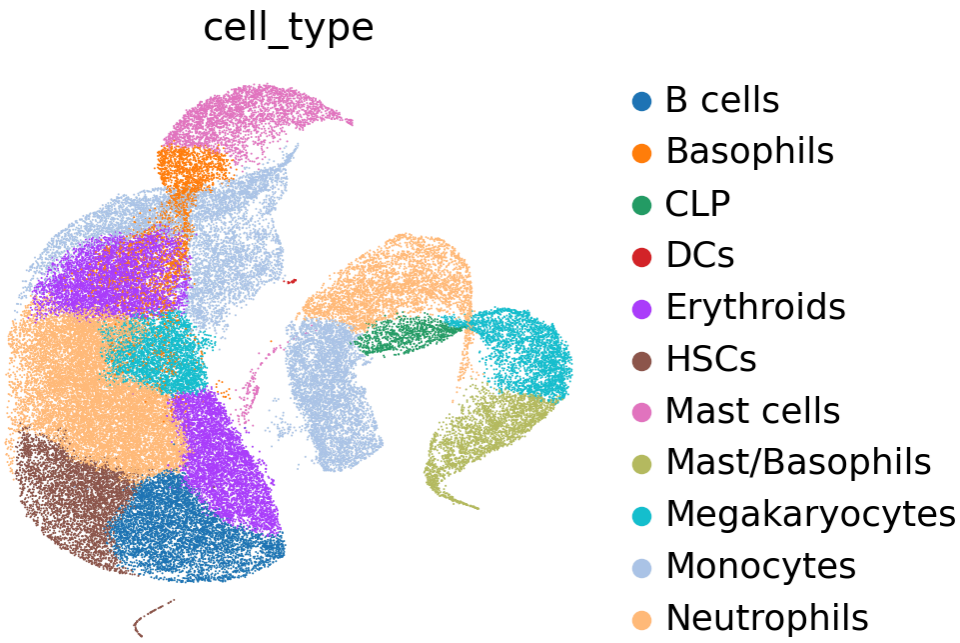
Labeling Clusters and Annotation HSC - haematopoeitic stem cell Ery - erythrocyte Mono - monocyte CLP - common lymphoid progenitor Mega - megakaryote DC - dendritic cell

In [22]:
```
cluster_to_celltype = {
    '0':  'Neutrophils',
    '1':  'Erythroids',
    '2':  'Neutrophils',
    '3':  'B cells',
    '4':  'Erythroids',
    '5':  'Monocytes',
    '6':  'HSCs',
    '7':  'Monocytes',
    '8':  'Neutrophils',
    '9':  'Megakaryocytes',
    '10': 'Basophils',
    '11': 'Mast cells',
    '12': 'Megakaryocytes',
    '13': 'Mast/Basophils',
    '14': 'Monocytes',
    '15': 'CLP',
    '16': 'Mast cells',
    '17': 'HSCs',
    '18': 'DCs'
}

adata.obs['cell_type'] = (
    adata.obs['clusters_r06']
    .map(cluster_to_celltype)
    .astype("category")
)
```

In [23]:
```
sc.pl.umap(adata, color = 'clusters_r06')
sc.pl.umap(adata, color = 'cell_type')
```





Q4) Perform DTP trajectory analysis. Produce a UMAP coloured by the DTP trajectory pseudotime. How many lineages are there?

Pseudotime Analysis

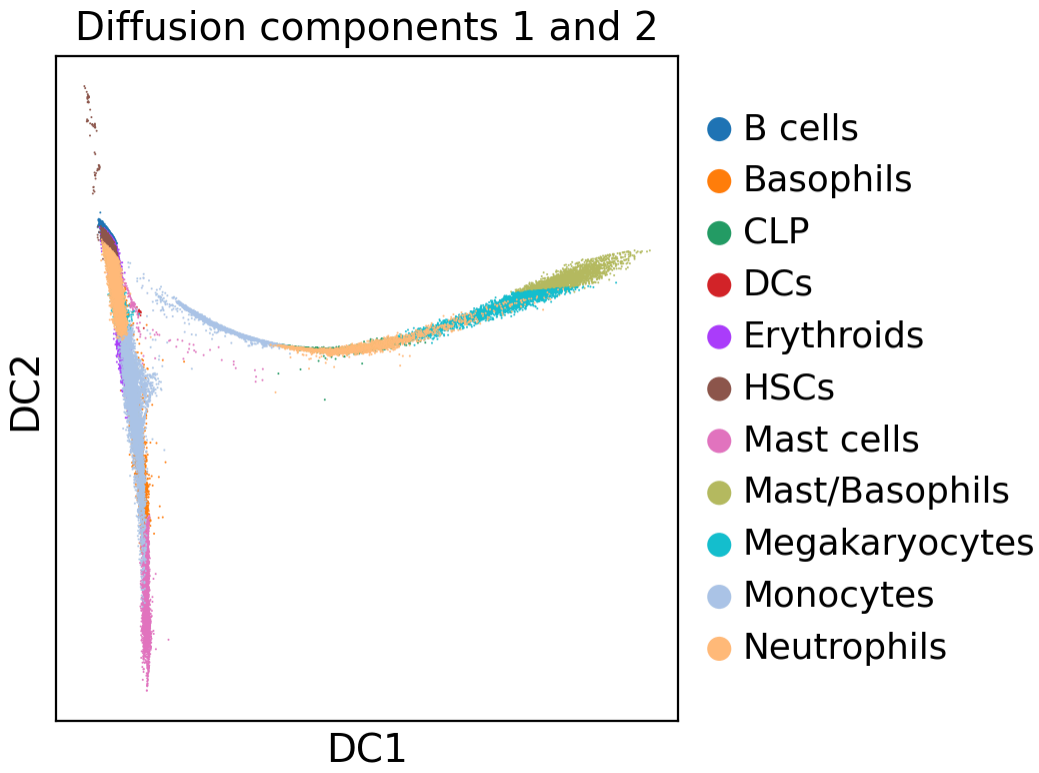
In [24]:
```
sc.tl.diffmap(adata) # calculating diffusion maps required for DPT
```

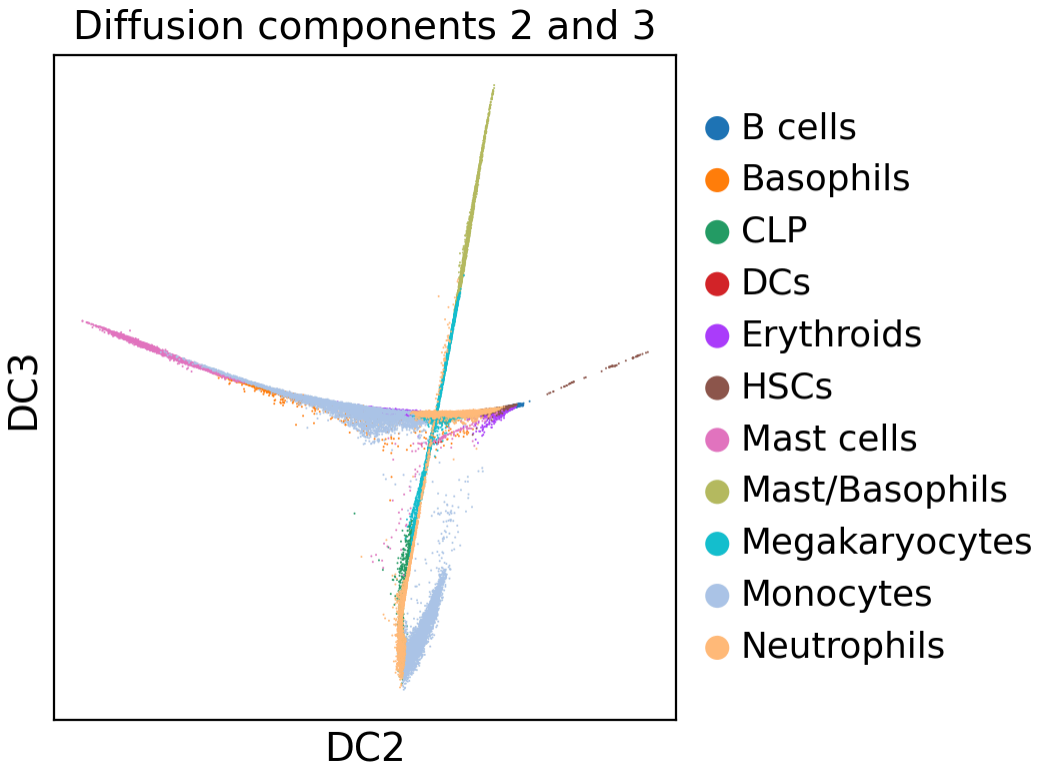
```
computing Diffusion Maps using n_comps=15(=n_dcs)
computing transitions
    finished (0:00:00)
    eigenvalues of transition matrix
    [1.         0.99974114 0.999059   0.99890053 0.9982664  0.9977546
     0.99747336 0.99666184 0.9951794  0.99421465 0.9936312  0.9934936
     0.991808   0.9914014  0.99090105]
    finished: added
    'X_diffmap', diffmap coordinates (adata.obsm)
    'diffmap_evals', eigenvalues of transition matrix (adata.uns) (0:00:02)
```

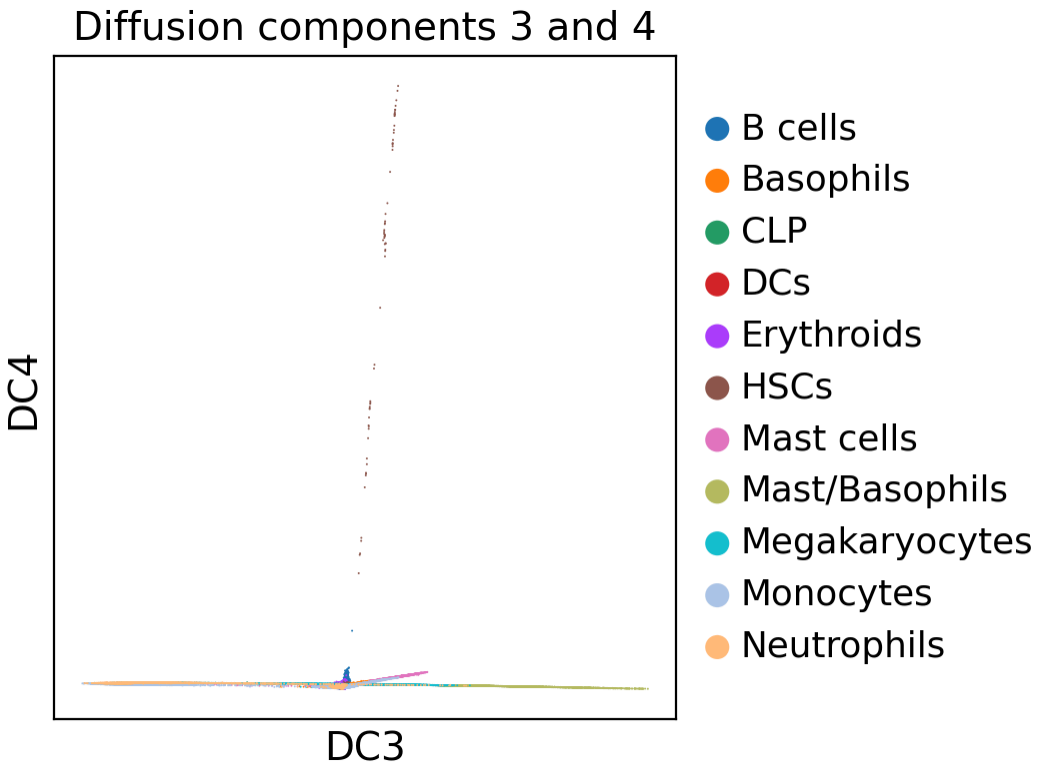
In [25]:
```
sc.pl.scatter(
    adata, basis="diffmap", color=["cell_type"], components=[1,2],
    frameon = True, title = 'Diffusion components 1 and 2')

sc.pl.scatter(
    adata, basis="diffmap", color=["cell_type"], components=[2,3],
    frameon = True, title = 'Diffusion components 2 and 3')

sc.pl.scatter(
    adata, basis="diffmap", color=["cell_type"], components=[3,4],
    frameon = True, title = 'Diffusion components 3 and 4')
```





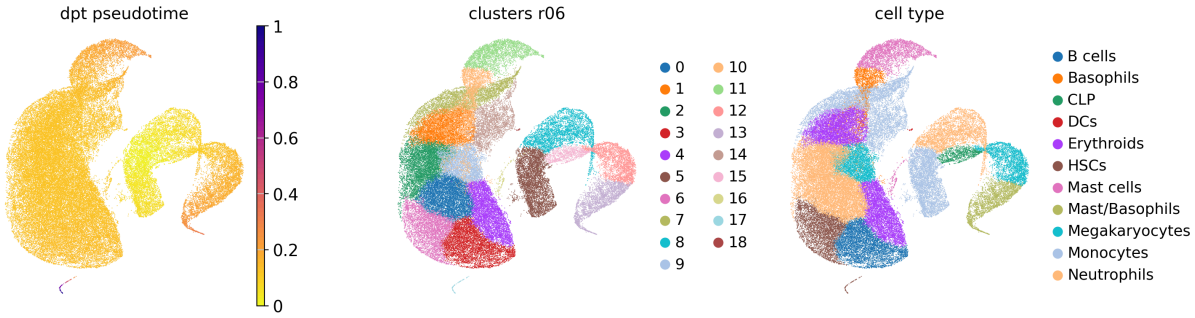


In [26]:
```
## Choosing root cell (HSCs) for DC3
root_ixs = adata.obsm["X_diffmap"][:, 3].argmin() #choosing the minimum component a
adata.uns["iroot"] = root_ixs
```

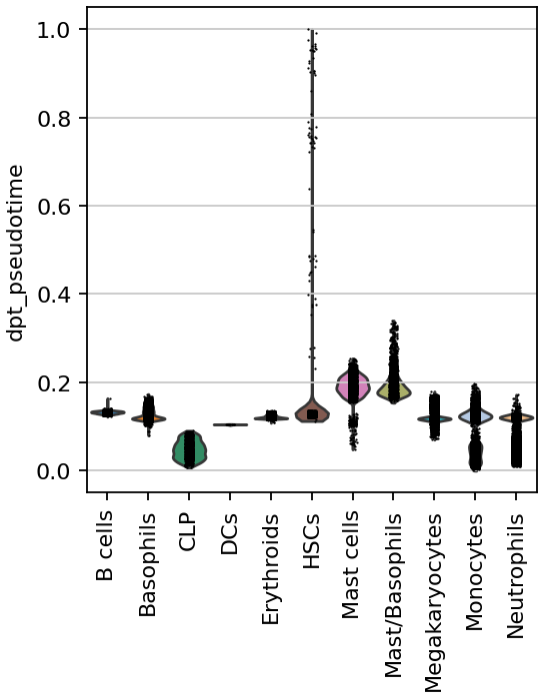
In [27]:
```
#Running pseudotime analysis
sc.tl.dpt(adata)
```

```
computing Diffusion Pseudotime using n_dcs=10
    finished: added
    'dpt_pseudotime', the pseudotime (adata.obs) (0:00:00)
```

In [28]:
```
#Plotting DPT pseudotime values alongside of annotated clusters
sc.pl.scatter(adata, basis="umap",
    color=["dpt_pseudotime", "clusters_r06", "cell_type"], color_map="plasma_r")
```



In [29]:
```
adata.obs_names_make_unique()
sc.settings.set_figure_params(scanpy = True, dpi=80, transparent=True, vector_frien
sc.pl.violin(
    adata, keys=["dpt_pseudotime"],
    groupby="cell_type", rotation=90)
```



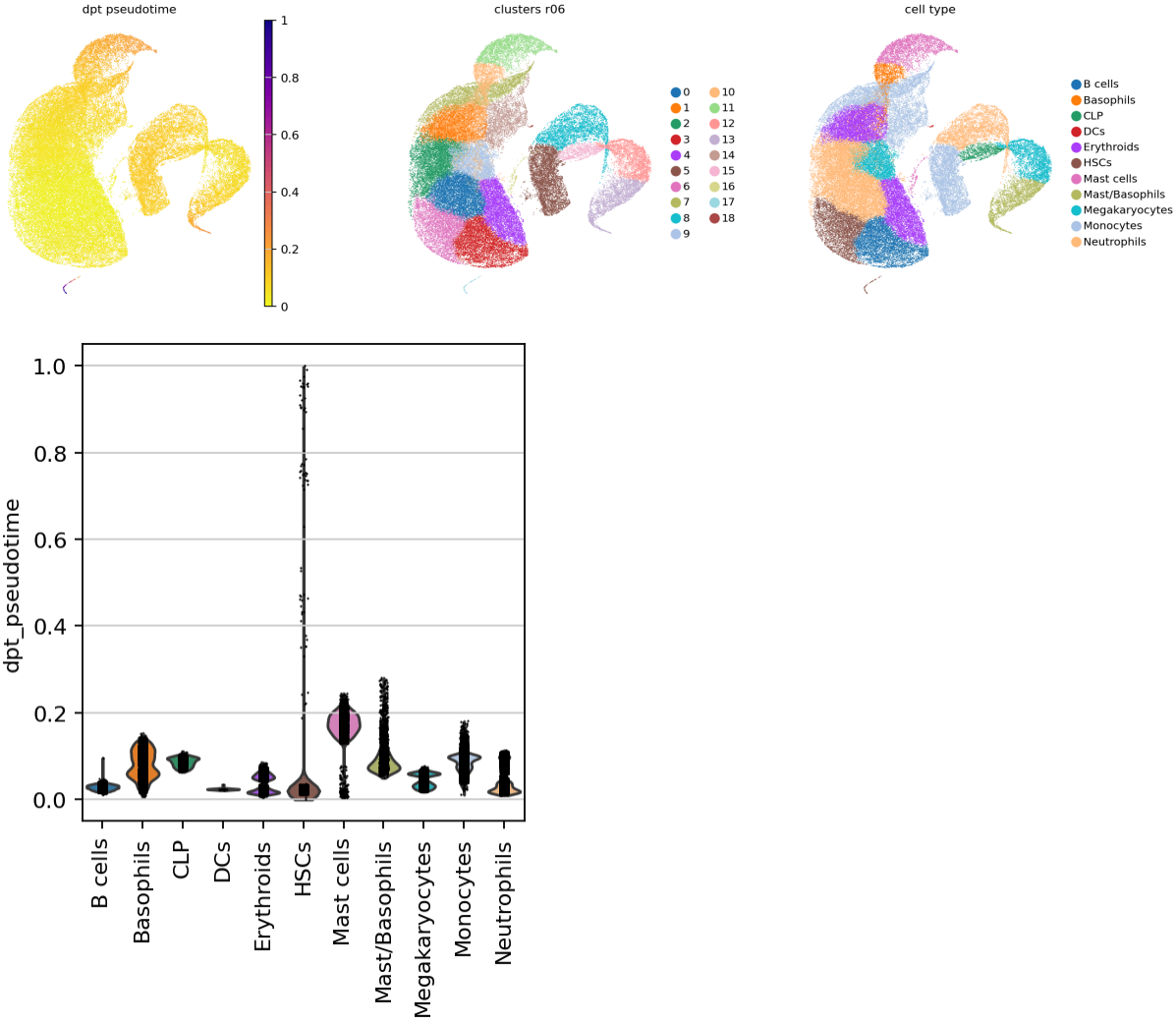
In [30]:
```
import scanpy as sc
adata.obs['DC3'] = adata.obsm['X_diffmap'][:, 3]
hsc_root = adata.obs[adata.obs['cell_type'] == 'HSCs'].sort_values('DC3').index[0]
adata.uns["iroot"] = adata.obs_names.get_loc(hsc_root)

sc.tl.dpt(adata)
sc.pl.scatter(adata, basis="umap",
    color=["dpt_pseudotime", "clusters_r06", "cell_type"], color_map="plasma_r")

adata.obs_names_make_unique()
```

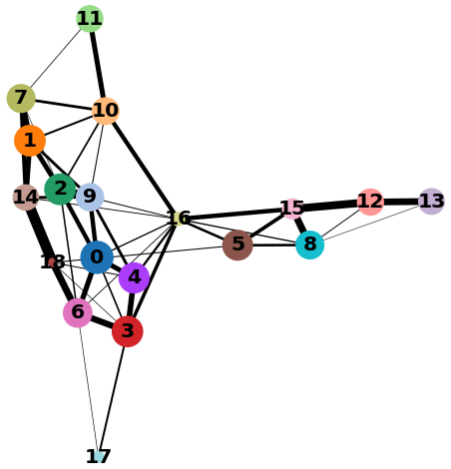
```
sc.settings.set_figure_params(scanpy = True, dpi=80, transparent=True, vector_frien
sc.pl.violin(
    adata, keys=["dpt_pseudotime"],
    groupby="cell_type", rotation=90)
```

```
computing Diffusion Pseudotime using n_dcs=10
    finished: added
    'dpt_pseudotime', the pseudotime (adata.obs) (0:00:00)
```



In [31]:
```
#Add PAGA to infer lineages:
import scanpy as sc
sc.tl.paga(adata, groups='clusters_r06')
sc.pl.paga(adata, threshold=0.03, show=True)  # Shows cluster connectivity
```

```
running PAGA
    finished: added
    'paga/connectivities', connectivities adjacency (adata.uns)
    'paga/connectivities_tree', connectivities subtree (adata.uns) (0:00:01)
--> added 'pos', the PAGA positions (adata.uns['paga'])
```



Ans 4) we can infer five distinct developmental lineages by tracing the main branches (thick black edges) that extend from the central/root region toward terminal ends.

Lineage 1: Cluster 0 → 2 → 1 → 7 → 10 → 11

Lineage 2: Cluster 0 → 2 → 1 → 14 → 18

Lineage 3: Cluster 0 → 4 → 6 → 17

Lineage 4: Cluster 0 → 4 → 3

Q5) Are there any limitations of this experiment? Would you make any changes to this analysis? Can you think of next steps for this analysis?

Ans5) a) Limitations:

Annotation is marker-based and may miss rare or novel populations.

DPT does not account for branching unless explicitly modeled.

b) I would ensure that all steps involving gene expression comparisons use appropriately normalized and log-transformed data. Incorporating additional quality control metrics, such as doublet detection and ambient RNA removal, could also refine the dataset further.

c) Next steps:- As for next steps, experimental validation of lineage paths using techniques like lineage tracing or time-course single-cell RNA-seq would strengthen conclusions. Integration with additional omics layers, such as chromatin accessibility (e.g., scATAC-seq), could also provide deeper insight into gene regulatory dynamics across lineages.