# Notes on QC

Initially I filtered cells based on QC variables.
Then I reassessed the filtered cells and ran scrublet for doublet simulation.

I will filter both with cutoffs and clusters. Sometimes it's difficult to design cutoffs that capture all the bad cells.
Clustering will often group them together well. If when assessing a sample I find a cluster of what I percieve as entirely bad cells, I will remove the cluster.

# ME8

Looking at ME8 I chose the following filtering parameters:
'min_counts':10000, 'max_counts':100000, 'min_genes':2900, 'max_genes':10000, 'min_mito':.1, 'max_mito':1.5, 'min_rb':15, 'max_rb':31

In our scatter and boxplots we can see 5 clusters that fall outside the filtering parameters.
Clusters 12 and 15 -> high Rb, low Mt and low genes Cluster 18 -> low genes Cluster 20 -> low Rb, low Mt and low genes
Cluster 22 -> low Mt and low genes

I want to check what they are before removing them.

Cluster 12 -> Poor quality brain cells. Very high Rb and low Mt. (Pou3f1)
Cluster 15 -> Poor quality ectoderm cells. Very high Rb and low Mt. (Epcam)
Cluster 18 -> Poor quality cells. No markers at all. Low counts
Cluster 20 -> Blood cells. Hence the low transcripts and low Rb and Mt
Cluster 22 -> Poor quality muscle cells. absent mitochondrial transcripts. Likely apoptotic or only nucleus. (Myl4)

I will mark the bad clusters as failed.:
['12','15','18','20','22']

Cells after QC:

Keeping the cells that passed the QC, I ran scrublet for doublet simulation.
Here cluster 14 stands out as being composed of likely doublets. In addition we can see small clumps of doublets around.

I filtered all cells by a doublet score of 0.15 and removed cluster 14.

# ME9

At ME9 I chose the following filtering parameters:
'min_counts':9000, 'max_counts':100000, 'min_genes':4000, 'max_genes':10000, 'min_mito':.2, 'max_mito':2, 'min_rb':10, 'max_rb':25

In the scatter and boxplots 9 clusters fall outside the filtering parameters.
0 -> low genes
2 -> low genes, high Mt
9 -> low genes
10 -> high Rb, low genes
12 -> high Rb, low genes, low Mt
14 -> low Rb, low genes, high Mt
19 -> high Rb, low genes
20 -> low genes, high Mt
22 -> low Rb, low genes, low Mt

I want to check what they are before removing them.

0 -> Markers include mostly blood genes. Hbb-y etc. I'm not sure if it's blood or just ambient RNA, either way it is not important.
2 -> Markers are mitochondrial genes. Probably bad cells or just captured mitochondria.
9 -> Blood.
10 -> Low quality cells. No markers.
12 -> Low quality cells. No markers.
14 -> Low quality brain cells. Markers such as Fgf14. Groups with rest of Sox2+ cells. Very high Mt and low genes.
19 -> Low quality mesoderm/mesenchyme. Markers such as Twist1.
20 -> This cluster seems mixed. Around half is bad cells, the other half seem legitimate. Markers such as Isl1 and Ccdc141. Luckily this agrees with the box/scatterplots.
The bad cells fall outside of the cutoffs.
22 -> Blood.

I will mark the bad clusters as failed.:
['0','2','9','10','12','14','19','22']

Cells after QC:

Keeping the cells that passed the QC, I ran scrublet for doublet simulation.
Here cluster 16 stands out as being composed of likely doublets. In addition we can see small clumps of doublets around.
I filtered all cells by a doublet score of 0.18 and removed cluster 16.

# ME10

At ME10 I chose the following filtering parameters:
'min_counts':5000, 'max_counts':100000, 'min_genes':2200, 'max_genes':10000, 'min_mito':.3, 'max_mito':2.2, 'min_rb':5, 'max_rb':28

In the scatter and boxplots 4 clusters fall outside the filtering parameters.
11 -> Low genes, high Mt
12 -> High Rb, low genes, low Mt
16 -> High Rb, low genes, low Mt
20 -> Low genes

Check them before removing.

11 -> Marker genes are all mitochondrial.
12 -> Low quality cells.
16 -> Low quality ectoderm. Epcam positive, but groups with low quality cells because of low genes, low Mt and low Rb.
20 -> Blood Hbb-bt.

I will mark the bad clusters as failed.:
['11','12','16','20']

Cells after QC:

Keeping the cells that passed the QC, I ran scrublet for doublet simulation.
Here cluster 12 stands out as being composed of likely doublets. In addition we can see small clumps of doublets around.
I filtered all cells by a doublet score of 0.18 and removed cluster 12.

# ME11

At ME11 I chose the following filtering parameters:
'min_counts':6000, 'max_counts':100000, 'min_genes':3100, 'max_genes':10000, 'min_mito':.5, 'max_mito':2.5, 'min_rb':9, 'max_rb':25

In the scatter and boxplots 6 clusters fall outside the filtering parameters.
0 -> Low genes, low Mt
3 -> Low genes, high Mt
12 -> Low Rb
15 -> Low Genes, low Mt
16 -> High Rb, low genes, low Mt
20 -> Low genes, low Mt

I want to check what they are before removing them.
0 -> High Hbb-y and other trash. Bad cells.
3 -> Mostly mito genes.
12 -> These cells seem good, they have markers. Their main difference is that they are predicted to be in G1 cell cycle phase. I will readjust the min_rb to 5 and keep them.
15 -> Blood.
16 -> Low quality ectoderm. Epcam positive.
20 -> Low quality cns. Tubb3

Ajusted filtering parameters to:
'min_counts':6000, 'max_counts':100000, 'min_genes':3100, 'max_genes':10000, 'min_mito':.5, 'max_mito':2.5, 'min_rb':5, 'max_rb':25

And I will mark the bad clusters as failed too.:
['0','3','15','16','20']

Cells after QC:

Keeping the cells that passed the QC, I ran scrublet for doublet simulation.
Here cluster 22 stands out as being composed of likely doublets. In addition we can see small clumps of doublets around.
I filtered all cells by a doublet score of 0.2 and removed cluster 22.

# ME12

At ME12 I chose the following filtering parameters:
'min_counts':2000, 'max_counts':100000, 'min_genes':2200, 'max_genes':10000,
'min_mito':.5, 'max_mito':2.5, 'min_rb':9, 'max_rb':30

A lot of the clusters have a portion of cells with really high Mt. I don't believe these cells are good. They generally have low counts and low genes as well.
I checked if they were overrepresented for any cell cycle state, but they were not. I'm not exactly sure what to do with these cells. Traditionally I have kept
the filtering parameters fairly strict unless there was a clear reason not to do so. For now I'll follow that and assume that these cells are bad in some way.

As for the low Rb in cluster 11, this is mainly G1 cell cycle. So I will relax this filtering parameter, similarly to the previous stage. Set min_rb to 5.

0 -> Low genes, low Mt
1 -> Low genes, high Mt
11 -> Low Rb
15 -> Low Rb, low genes, low Mt
23 -> Low genes
24 -> Low Rb, low genes, low Mt

0 -> No real markers, just crap.
1 -> Mostly Mito genes.
11 -> G1 phase neurons/brain.
15 -> Blood.
23 -> Poor quality neuronal cells (Tubb3) 24 -> Blood.

Ajusted filtering parameters to:
'min_counts':2000, 'max_counts':100000, 'min_genes':2200, 'max_genes':10000,
'min_mito':.5, 'max_mito':2.5, 'min_rb':5, 'max_rb':30

And I will mark the bad clusters as failed too.:
['0','1','15','23','24']

Cells after QC:

Just wanted to note the amount of failed cells here. These have low genes and a very high mitochondrial content.
This is generally believed to be indicative of bad/stressed cells. In addition this sample had

way more cells than the other samples.
I would not be surprised if that contributed.

Keeping the cells that passed the QC, I ran scrublet for doublet simulation.
Here no cluster stands out as being composed of likely doublets. But we can see small clumps of doublets around.
I filtered all cells by a doublet score of 0.18.

In [1]:
```
import logging
logging.getLogger('matplotlib.font_manager').setLevel(logging.ERROR)
import scanpy as sc
import scvelo as scv
import scvi
import seaborn as sns
import plotly.express as px
import numpy as np
from dash import Dash, dcc, html, Input, Output

import pandas as pd

import os
import sys
import time
import gc
os.environ['R_HOME'] = sys.exec_prefix+"/lib/R/"

# Plotting
import matplotlib
import matplotlib.pyplot as plt
import matplotlib as mpl
from matplotlib.backends.backend_pdf import PdfPages
from matplotlib.colors import LinearSegmentedColormap, ListedColormap
from matplotlib.lines import Line2D

from copy import copy
reds = copy(mpl.cm.Reds)
reds.set_under("lightgray")

project_directory = '/Cranio_Lab/Louk_Seton/4_species_project'
os.chdir(os.path.expanduser("~")+project_directory)
```

In [2]:
```
##TODO

#try soft integration with just hvgs > look into better hvg selection
#else hard integration with mira for example
#try to automate plotting good DEGs
#roughly annotate cells
#plot annotated umap and dotplots

#repeat on other organisms

######Mouse mm10######
```

```
start_time=time.strftime("%Y_%m_%d-%I_%M_%S_%p")
print('start time:',start_time)

sample_names = ['ME8','ME9','ME10','ME11','ME12'] #specify the sample names
species = 'mouse' #specify the species
genome = 'mm10' #specify the genome
output_prefix = 'cellranger_related/cellranger_output/sc_rnaseq/' #specify t

##plotting vars
output_dir = 'figures_publish/'+species+'/'+genome+'/QC_plots/'
!mkdir -p {output_dir}
genes_to_plot = ['Sox10','Plp1','Hbb-y','Epcam','Twist2','Prrx2','Tfap2b','A
dot_size = 1
linewidth = 0
scatter_color = 'pct_counts_hb'
scatter_cmap = 'coolwarm'
y1 = 'pct_counts_mt' #these are for the QC scatter plots
y2 = 'pct_counts_rb'
x1 = 'n_genes_by_counts'
x2 = 'total_counts'
##########################################################################
cellranger_locs = []
velocyto_locs = []

for sample in sample_names: #simple loop to create a list with the location
    cellranger_locs = cellranger_locs + [output_prefix+species+'/'+genome+'/
    velocyto_locs = velocyto_locs + [output_prefix+species+'/'+genome+'/'+sa

adata_files={}
print(time.strftime("%Y_%m_%d-%I_%M_%S_%p"),'Reading file(s)')
for i,j,k in zip(sample_names, cellranger_locs, velocyto_locs):
    ##Read and merge 10x and velocyto output, preserving gene names and ense
    print(time.strftime("%Y_%m_%d-%I_%M_%S_%p"),'Sample ',i)
    adata_cellr = sc.read_10x_mtx(j)
    print(time.strftime("%Y_%m_%d-%I_%M_%S_%p"),'Loaded 10x mtx')
    adata_cellr.obs.index = adata_cellr.obs.index.str[:-2]
    adata_cellr.var['gene_name'] = adata_cellr.var.index
    adata_cellr.var.index = adata_cellr.var['gene_ids']
    adata_veloc = sc.read(k)
    print(time.strftime("%Y_%m_%d-%I_%M_%S_%p"),'Loaded velocyto loom')
    adata_veloc.obs.index = adata_veloc.obs.index.str.split(':').str[1]
    adata_veloc.obs.index = adata_veloc.obs.index.str[:-1]
    adata_veloc.var['gene_name'] = adata_veloc.var.index
    adata_veloc.var.index = adata_veloc.var['Accession']
    print(time.strftime("%Y_%m_%d-%I_%M_%S_%p"),'Merging data')
    adata_merge = scv.utils.merge(adata_cellr, adata_veloc)
    adata_files['adata'+i]=adata_merge
    adata_files['adata'+i].obs['sample']=i
    adata_files['adata'+i].obs['ref_genome']=genome
    adata_files['adata'+i].obs['barcode']=adata_files['adata'+i].obs.index
    adata_files['adata'+i].obs.index=adata_files['adata'+i].obs['barcode']+'

    adata_files['adata'+i].var.index = list(adata_files['adata'+i].var['gene

    ##Calculate QC values
    print(time.strftime("%Y_%m_%d-%I_%M_%S_%p"),'Calculating QC values for c
```

```
    adata_files['adata'+i].var['mt'] = adata_files['adata'+i].var['gene_name
    adata_files['adata'+i].var['rb'] = adata_files['adata'+i].var['gene_name
    adata_files['adata'+i].var['hb'] = adata_files['adata'+i].var['gene_name
    adata_files['adata'+i].var_names_make_unique()
    adata_files['adata'+i].obs_names_make_unique()
    sc.pp.calculate_qc_metrics(adata_files['adata'+i], qc_vars=["mt", "rb",
    #clip to meaningful numbers, better for comparisons
    adata_files['adata'+i].obs['pct_counts_mt'] = adata_files['adata'+i].obs
    adata_files['adata'+i].obs['pct_counts_hb'] = adata_files['adata'+i].obs
    adata_files['adata'+i].obs['pct_counts_rb'] = adata_files['adata'+i].obs
    adata_files['adata'+i].obs['total_counts'] = adata_files['adata'+i].obs[

    ##calculate doublet score
    print(time.strftime("%Y_%m_%d-%I_%M_%S_%p"),'Running scrublet for double
    sc.pp.scrublet(adata_files['adata'+i])

    print(time.strftime("%Y_%m_%d-%I_%M_%S_%p"),'Initial preprocess and clus
    ##standard normalization
    adata_files['adata'+i].layers["original_counts"] = adata_files['adata'+i
    sc.pp.normalize_total(adata_files['adata'+i]) # Normalizing to median to
    sc.pp.log1p(adata_files['adata'+i]) # Logarithmize the data
    adata_files['adata'+i].layers["normalized_counts"] = adata_files['adata'

    ##highly variable genes
    sc.pp.highly_variable_genes(adata_files['adata'+i], n_top_genes=2000,)

    ##dimensionality reduction and clustering
    sc.tl.pca(adata_files['adata'+i])
    sc.pp.neighbors(adata_files['adata'+i])
    sc.tl.umap(adata_files['adata'+i])
    sc.tl.leiden(adata_files['adata'+i],)

    ##cell cycle scoring
    print(time.strftime("%Y_%m_%d-%I_%M_%S_%p"),'Prepare cell cycle scoring'
    cell_cycle_genes = [x.strip() for x in open('required_files/regev_lab_ce
    s_genes = cell_cycle_genes[:43]
    g2m_genes = cell_cycle_genes[43:]
    print(time.strftime("%Y_%m_%d-%I_%M_%S_%p"),'Cell cycle scoring')
    sc.tl.score_genes_cell_cycle(adata_files['adata'+i], s_genes=s_genes, g2


    #plot pdfs of qc
    print(time.strftime("%Y_%m_%d-%I_%M_%S_%p"),'Plotting QC and saving to P
    with PdfPages(output_dir+i+'_QC.pdf') as pdf:
        mpl.rcParams['axes.grid'] = False
        ax = sc.pl.violin(adata_files['adata'+i],["n_genes_by_counts", "tota
                          stripplot=False, multi_panel=True,show=False,
                          height=4, aspect=.8,col_wrap = 3)
        plt.suptitle(i, x = 0.1)
        pdf.savefig()  # saves the current figure into a pdf page
        plt.close() # close the plot and page

        ###Scatterplots
        fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2,2,layout='constrained
        s=dot_size
        linewidth=linewidth
```

```
        cmap=scatter_cmap
        c = scatter_color
        ax1.scatter(data=adata_files['adata'+i].obs, x=x1,y=y1,
                    c = c,s = s,linewidth=linewidth, cmap = cmap,rasterized=
        ax1.set_xlabel(x1)
        ax1.set_ylabel(y1)
        ax2.scatter(data=adata_files['adata'+i].obs, x=x2,y=y1,
                    c = c,s = s,linewidth=linewidth, cmap = cmap,rasterized=
        ax2.set_xlabel(x2)
        ax2.set_ylabel(y1)
        ax3.scatter(data=adata_files['adata'+i].obs, x=x1,y=y2,
                    c = c,s = s,linewidth=linewidth, cmap = cmap,rasterized=
        ax3.set_xlabel(x1)
        ax3.set_ylabel(y2)
        ax4.scatter(data=adata_files['adata'+i].obs, x=x2,y=y2,
                    c = c,s = s,linewidth=linewidth, cmap = cmap,rasterized=
        ax4.set_xlabel(x2)
        ax4.set_ylabel(y2)


        fig.colorbar(mpl.cm.ScalarMappable(norm=mpl.colors.Normalize(0, max(
                     ax=ax1, orientation='vertical', label=c)
        pdf.savefig(dpi=150)
        plt.close()

        ##nice jointplot shows good cells
        mpl.rcParams['axes.grid'] = False
        sns_plot = sns.jointplot(data=adata_files['adata'+i].obs,
                                 x="log1p_total_counts",
                                 y="log1p_n_genes_by_counts",
                                 kind="hex",
                                 marginal_kws=dict(bins=100,),
                                 joint_kws = dict(gridsize=50,),
                                )
        sns_plot.fig.suptitle(i,x=0.05)
        pdf.savefig(dpi=150)
        plt.close()

        ##custom hvg plot
        fig, ((ax1), (ax2)) = plt.subplots(1,2,layout='constrained',figsize
        s=dot_size
        linewidth=linewidth
        cmap = ListedColormap(['grey','black'])
        c = 'highly_variable'
        scatter = ax1.scatter(data=adata_files['adata'+i].var, x='means',y="
                              c=c,s = s,cmap=cmap,linewidth=linewidth,raster
        ax1.set_xlabel('mean expression')
        ax1.set_ylabel('dispersion')
        ax1.legend(scatter.legend_elements()[0][0:2],['other genes','highly
                   loc="upper right",)
        scatter = ax2.scatter(data=adata_files['adata'+i].var, x='means',y="
                              c=c,s = s,cmap=cmap,linewidth=linewidth,raster
        ax2.legend(scatter.legend_elements()[0][0:2],['other genes','highly
                   loc="upper right",)
        ax2.set_xlabel('mean expression')
        ax2.set_ylabel('dispersions_norm')
```

```
        pdf.savefig(dpi=150)
        plt.close()

        ##PCA variance plot
        sc.pl.pca_variance_ratio(adata_files['adata'+i], log=True, show=Fals
        pdf.savefig()
        plt.close()

        ##umaps
        plt.rcParams['figure.figsize'] = [5,4]
        ax = sc.pl.umap(adata_files['adata'+i],
                        color=['doublet_score','leiden','total_counts','pct_
                        show = False,ncols = 2)
        for p in ax:
            p.set_rasterized(True)
        pdf.savefig(dpi=150)
        plt.close()

        ##Most highly expressed genes
        plt.rcParams['figure.figsize'] = [8,4]
        adata_files['adata'+i].X = adata_files['adata'+i].layers["original_c
        ax = sc.pl.highest_expr_genes(adata_files['adata'+i],
                                      gene_symbols = 'gene_name',
                                      n_top=20,log=True,show=False)
        ax.set_rasterized(True)
        pdf.savefig()
        plt.close()
        ##return normalised counts to X
        adata_files['adata'+i].X = adata_files['adata'+i].layers["normalized

        ##doublet scatterplots
        ###Scatterplots
        fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2,2,layout='constrained
        s=dot_size
        linewidth=linewidth
        cmap=scatter_cmap
        c = scatter_color
        ax1.scatter(data=adata_files['adata'+i].obs, x=x1,y='doublet_score',
                    c = c,s = s,linewidth=linewidth, cmap = cmap,rasterized=
        ax1.set_xlabel(x1)
        ax4.set_ylabel('doublet_score')
        ax2.scatter(data=adata_files['adata'+i].obs, x=x2,y='doublet_score',
                    c = c,s = s,linewidth=linewidth, cmap = cmap,rasterized=
        ax2.set_xlabel(x2)
        ax4.set_ylabel('doublet_score')
        ax3.scatter(data=adata_files['adata'+i].obs, x=y1,y='doublet_score',
                    c = c,s = s,linewidth=linewidth, cmap = cmap,rasterized=
        ax3.set_xlabel(y1)
        ax4.set_ylabel('doublet_score')
        ax4.scatter(data=adata_files['adata'+i].obs, x=y2,y='doublet_score',
                    c = c,s = s,linewidth=linewidth, cmap = cmap,rasterized=
        ax4.set_xlabel(y2)
        ax4.set_ylabel('doublet_score')
        fig.colorbar(mpl.cm.ScalarMappable(norm=mpl.colors.Normalize(0, max(
                     ax=ax1, orientation='vertical', label=c)
        pdf.savefig(dpi=150)
```

```
        plt.close()

        #boxplot of doublet scores
        ax = sns.boxplot(data=adata_files['adata'+i].obs, x="leiden", y="dou
            palette = adata_files['adata'+i].uns['leiden_colors'],
                 fliersize=0)
        cnum = 0
        for line in ax.lines:
            if line.get_linestyle() == 'None':
                sns.swarmplot(x=[int(b) for b in line.get_xdata()], y=line.g
                              size = 1.5, color = adata_files['adata'+i].uns
                cnum = cnum+1
        pdf.savefig(dpi=150)
        plt.close()

    ##saving h5ad
    print(time.strftime("%Y_%m_%d-%I_%M_%S_%p"),'Saving data to h5ad file')
    adata_files['adata'+i].write('h5ad_files/'+species+'/'+genome+'/'+i+'_be


    ##remove used variables from memory
    del adata_cellr
    del adata_veloc
    del adata_merge
    gc.collect()
    print(time.strftime("%Y_%m_%d-%I_%M_%S_%p"),'Loaded files for ',i)
print(time.strftime("%Y_%m_%d-%I_%M_%S_%p"),'Finished Reading file(s)')
```

```
start time: 2025_03_14-05_46_03_PM
2025_03_14-05_46_03_PM Reading file(s)
2025_03_14-05_46_03_PM Sample  ME8
2025_03_14-05_46_11_PM Loaded 10x mtx
2025_03_14-05_46_50_PM Loaded velocyto loom
2025_03_14-05_46_50_PM Merging data
2025_03_14-05_47_09_PM Calculating QC values for cells
2025_03_14-05_47_12_PM Running scrublet for doublet detection
2025_03_14-05_48_47_PM Initial preprocess and clustering
2025_03_14-05_49_03_PM Prepare cell cycle scoring
2025_03_14-05_49_03_PM Cell cycle scoring
2025_03_14-05_49_07_PM Plotting QC and saving to PDF
2025_03_14-05_49_24_PM Saving data to h5ad file
2025_03_14-05_49_37_PM Loaded files for  ME8
2025_03_14-05_49_37_PM Sample  ME9
2025_03_14-05_49_53_PM Loaded 10x mtx
2025_03_14-05_50_57_PM Loaded velocyto loom
2025_03_14-05_50_57_PM Merging data
2025_03_14-05_51_20_PM Calculating QC values for cells
2025_03_14-05_51_25_PM Running scrublet for doublet detection
2025_03_14-05_53_46_PM Initial preprocess and clustering
2025_03_14-05_54_03_PM Prepare cell cycle scoring
2025_03_14-05_54_03_PM Cell cycle scoring
2025_03_14-05_54_09_PM Plotting QC and saving to PDF
2025_03_14-05_54_30_PM Saving data to h5ad file
2025_03_14-05_54_48_PM Loaded files for  ME9
2025_03_14-05_54_48_PM Sample  ME10
2025_03_14-05_55_08_PM Loaded 10x mtx
2025_03_14-05_56_16_PM Loaded velocyto loom
2025_03_14-05_56_16_PM Merging data
2025_03_14-05_56_38_PM Calculating QC values for cells
2025_03_14-05_56_42_PM Running scrublet for doublet detection
2025_03_14-05_58_50_PM Initial preprocess and clustering
2025_03_14-05_59_09_PM Prepare cell cycle scoring
2025_03_14-05_59_09_PM Cell cycle scoring
2025_03_14-05_59_16_PM Plotting QC and saving to PDF
2025_03_14-05_59_36_PM Saving data to h5ad file
2025_03_14-05_59_55_PM Loaded files for  ME10
2025_03_14-05_59_55_PM Sample  ME11
2025_03_14-06_00_15_PM Loaded 10x mtx
2025_03_14-06_01_32_PM Loaded velocyto loom
2025_03_14-06_01_32_PM Merging data
2025_03_14-06_02_00_PM Calculating QC values for cells
2025_03_14-06_02_05_PM Running scrublet for doublet detection
2025_03_14-06_04_44_PM Initial preprocess and clustering
2025_03_14-06_05_02_PM Prepare cell cycle scoring
2025_03_14-06_05_02_PM Cell cycle scoring
2025_03_14-06_05_08_PM Plotting QC and saving to PDF
2025_03_14-06_05_30_PM Saving data to h5ad file
2025_03_14-06_05_49_PM Loaded files for  ME11
2025_03_14-06_05_49_PM Sample  ME12
2025_03_14-06_06_15_PM Loaded 10x mtx
2025_03_14-06_07_54_PM Loaded velocyto loom
2025_03_14-06_07_54_PM Merging data
2025_03_14-06_08_20_PM Calculating QC values for cells
2025_03_14-06_08_26_PM Running scrublet for doublet detection
```

```
2025_03_14-06_12_05_PM Initial preprocess and clustering
2025_03_14-06_12_32_PM Prepare cell cycle scoring
2025_03_14-06_12_32_PM Cell cycle scoring
2025_03_14-06_12_40_PM Plotting QC and saving to PDF
2025_03_14-06_13_08_PM Saving data to h5ad file
2025_03_14-06_13_32_PM Loaded files for  ME12
2025_03_14-06_13_32_PM Finished Reading file(s)
```

In [3]:

```
##Filter using QC vars and make plots
filter_vars = {'ME8':{'min_counts':10000, 'max_counts':100000, 'min_genes':2
                      'clusters':['12','15','18','20','22']},
               'ME9':{'min_counts':9000, 'max_counts':100000, 'min_genes':40
                      'clusters':['0','2','9','10','12','14','19','22']},
               'ME10':{'min_counts':5000, 'max_counts':100000, 'min_genes':2
                      'clusters':['11','12','16','20']},
               'ME11':{'min_counts':6000, 'max_counts':100000, 'min_genes':3
                      'clusters':['0','3','15','16','20']}, #adjusted
               'ME12':{'min_counts':2000, 'max_counts':100000, 'min_genes':2
                      'clusters':['0','1','15','23','24']}, #adjusted
}
filter_vars = pd.DataFrame(filter_vars).T


y1 = 'pct_counts_mt' #these are for the QC scatter plots
y2 = 'pct_counts_rb'
x1 = 'n_genes_by_counts'
x2 = 'total_counts'
dot_size = 1
linewidth = 0
scatter_color = 'QC'
output_dir = 'figures_publish/'+species+'/'+genome+'/QC_plots/'
!mkdir -p {output_dir}

for sample in list(filter_vars.index):
    min_counts = filter_vars.loc[sample,'min_counts']
    max_counts = filter_vars.loc[sample,'max_counts']
    min_genes = filter_vars.loc[sample,'min_genes']
    max_genes = filter_vars.loc[sample,'max_genes']
    min_mito = filter_vars.loc[sample,'min_mito']
    max_mito = filter_vars.loc[sample,'max_mito']
    min_rb = filter_vars.loc[sample,'min_rb']
    max_rb = filter_vars.loc[sample,'max_rb']

    conditions = [
    (adata_files['adata'+sample].obs['predicted_doublet'] == True),
    (adata_files['adata'+sample].obs['n_genes_by_counts'] < min_genes),
    (adata_files['adata'+sample].obs['n_genes_by_counts'] > max_genes),
    (adata_files['adata'+sample].obs['total_counts'] < min_counts),
    (adata_files['adata'+sample].obs['total_counts'] > max_counts),
    (adata_files['adata'+sample].obs['pct_counts_mt'] > max_mito),
    (adata_files['adata'+sample].obs['pct_counts_mt'] < min_mito),
    (adata_files['adata'+sample].obs['pct_counts_rb'] < min_rb),
    (adata_files['adata'+sample].obs['pct_counts_rb'] > max_rb),
    (adata_files['adata'+sample].obs['pct_counts_mt'] <= max_mito) & (adata_
    (adata_files['adata'+sample].obs['total_counts'] >= min_counts) & (adata
    (adata_files['adata'+sample].obs['predicted_doublet'] != True)
```

```
    ]

    values = ['Doublet','Low_nFeature', 'High_nFeature', 'Low_Counts','High_
    adata_files['adata'+sample].obs['QC'] = np.select(conditions, values)
    adata_files['adata'+sample].obs.loc[adata_files['adata'+sample].obs['lei
    adata_files['adata'+sample].obs['QC'] = adata_files['adata'+sample].obs[

    colors = {'Doublet':'#bc80bd',
                  'Low_nFeature':'#d9d9d9',
                  'High_nFeature':'#8dd3c7',
                  'Low_Counts':'#ffffb3',
                  'High_Counts':'#bebada',
                  'High_MT':'#fb8072',
                  'Low_MT':'#80b1d3',
                  'Low_RB':'#fdb462',
                  'High_RB':'#fccde5',
                  'Bad_Cluster':'#000000',
                  'Pass':'#b3de69',
                 }
    with PdfPages(output_dir+sample+'_filter_params.pdf') as pdf:
        plt.rcParams['figure.figsize'] = [5,4]
        ax = sc.pl.umap(adata_files['adata'+sample], color = ['QC'],palette
        ax.set_rasterized(True)
        pdf.savefig(dpi=150,bbox_inches='tight')  # saves the current figure
        plt.close() # close the plot and page

        plt.rcParams['figure.figsize'] = [5,4]
        ax = sc.pl.umap(adata_files['adata'+sample], color = ['n_genes_by_co
        for p in ax:
            p.set_rasterized(True)
        pdf.savefig(dpi=150)  # saves the current figure into a pdf page
        plt.close() # close the plot and page

        ###Scatterplots
        fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2,2,layout='constrained
        s=dot_size
        linewidth=linewidth
        c = scatter_color
        ax1.scatter(data=adata_files['adata'+sample].obs, x=x1,y=y1,
                    c = adata_files['adata'+sample].obs[c].map(colors),s = s
        ax1.set_xlabel(x1)
        ax1.set_ylabel(y1)
        ax2.scatter(data=adata_files['adata'+sample].obs, x=x2,y=y1,
                    c = adata_files['adata'+sample].obs[c].map(colors),s = s
        ax2.set_xlabel(x2)
        ax2.set_ylabel(y1)
        ax3.scatter(data=adata_files['adata'+sample].obs, x=x1,y=y2,
                    c = adata_files['adata'+sample].obs[c].map(colors),s = s
        ax3.set_xlabel(x1)
        ax3.set_ylabel(y2)
        scatter = ax4.scatter(data=adata_files['adata'+sample].obs, x=x2,y=y
                    c = adata_files['adata'+sample].obs[c].map(colors),s = s
        ax4.set_xlabel(x2)
        ax4.set_ylabel(y2)
        handles = [Line2D([0], [0], marker='o', color='w', markerfacecolor=v
        ax4.legend(title='color', handles=handles, bbox_to_anchor=(1.05, 1),
```

```
        pdf.savefig(dpi=150)
        plt.close()
```

In [ ]:

```
##doublet stuff and some more plots
genes_to_plot = ['Sox10','Plp1','Hbb-y','Epcam','Twist2','Prrx2','Tfap2b','A
dot_size = 1
linewidth = 0
scatter_color = 'pct_counts_hb'
scatter_cmap = 'coolwarm'
y1 = 'pct_counts_mt' #these are for the QC scatter plots
y2 = 'pct_counts_rb'
x1 = 'n_genes_by_counts'
x2 = 'total_counts'

for i in sample_names:
    sn = 'adata'+i #I got tired of long ass code so shortening the samplenam
    adata_files[sn+'_pass'] = adata_files[sn][adata_files[sn].obs['QC']=='Pa
    sn = sn+'_pass'
    adata_files[sn].X = adata_files[sn].layers["original_counts"].copy() #re
    sc.pp.scrublet(adata_files[sn])

    print(time.strftime("%Y_%m_%d-%I_%M_%S_%p"),'Initial preprocess and clus
    ##standard normalization
    sc.pp.normalize_total(adata_files[sn]) # Normalizing to median total cou
    sc.pp.log1p(adata_files[sn]) # Logarithmize the data
    adata_files[sn].layers["normalized_counts"] = adata_files[sn].X.copy()

    ##highly variable genes
    sc.pp.highly_variable_genes(adata_files[sn], n_top_genes=2000,)

    ##dimensionality reduction and clustering
    sc.tl.pca(adata_files[sn])
    sc.pp.neighbors(adata_files[sn])
    sc.tl.umap(adata_files[sn])
    sc.tl.leiden(adata_files[sn],resolution = 1, key_added = 'leiden_filt')
    sc.tl.leiden(adata_files[sn],resolution = 2, key_added = 'leiden_filt_hi

    ##plotting new stuff
    output_dir = 'figures_publish/'+species+'/'+genome+'/QC_plots/'
    !mkdir -p {output_dir}

    with PdfPages(output_dir+i+'_doublet_check.pdf') as pdf:
        ##umaps
        plt.rcParams['figure.figsize'] = [5,4]
        ax = sc.pl.umap(adata_files[sn],
                        color=['doublet_score','leiden','total_counts','leic
                        show = False,ncols = 2)
        for p in ax:
            p.set_rasterized(True)
        pdf.savefig(dpi=150)
        plt.close()

        #boxplot of doublet scores
        plt.rcParams['figure.figsize'] = [8,4]
        ax = sns.boxplot(data=adata_files[sn].obs, x="leiden", y="doublet_sc
```

```
        palette = adata_files[sn].uns['leiden_colors'],
             fliersize=0)
    pdf.savefig(dpi=150)
    plt.close()

    ax = sns.boxplot(data=adata_files[sn].obs, x="leiden_filt", y="doubl
        palette = adata_files[sn].uns['leiden_filt_colors'],
             fliersize=0)
    cnum = 0
    for line in ax.lines:
        if line.get_linestyle() == 'None':
            sns.swarmplot(x=[int(b) for b in line.get_xdata()], y=line.g
                          size = 1.5, color = adata_files[sn].uns['leide
            cnum = cnum+1
    pdf.savefig(dpi=150)
    plt.close()

    ax = sns.boxplot(data=adata_files[sn].obs, x="leiden_filt_highres",
        palette = adata_files[sn].uns['leiden_filt_highres_colors'],
             fliersize=0)
    cnum = 0
    for line in ax.lines:
        if line.get_linestyle() == 'None':
            sns.swarmplot(x=[int(b) for b in line.get_xdata()], y=line.g
                          size = 1.5, color = adata_files[sn].uns['leide
            cnum = cnum+1
    pdf.savefig(dpi=150)
    plt.close()

    ##doublet scatterplots
    ###Scatterplots
    plt.rcParams['figure.figsize'] = [5,4]
    fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2,2,layout='constrained
    s=dot_size
    linewidth=linewidth
    cmap=scatter_cmap
    c = scatter_color
    ax1.scatter(data=adata_files[sn].obs, x=x1,y='doublet_score',
                c = c,s = s,linewidth=linewidth, cmap = cmap,rasterized=
    ax1.set_xlabel(x1)
    ax4.set_ylabel('doublet_score')
    ax2.scatter(data=adata_files[sn].obs, x=x2,y='doublet_score',
                c = c,s = s,linewidth=linewidth, cmap = cmap,rasterized=
    ax2.set_xlabel(x2)
    ax4.set_ylabel('doublet_score')
    ax3.scatter(data=adata_files[sn].obs, x=y1,y='doublet_score',
                c = c,s = s,linewidth=linewidth, cmap = cmap,rasterized=
    ax3.set_xlabel(y1)
    ax4.set_ylabel('doublet_score')
    ax4.scatter(data=adata_files[sn].obs, x=y2,y='doublet_score',
                c = c,s = s,linewidth=linewidth, cmap = cmap,rasterized=
    ax4.set_xlabel(y2)
    ax4.set_ylabel('doublet_score')
    fig.colorbar(mpl.cm.ScalarMappable(norm=mpl.colors.Normalize(0, max(
                 ax=ax1, orientation='vertical', label=c)
```

```
        pdf.savefig(dpi=150)
        plt.close()
```

```
WARNING: adata.X seems to be already log-transformed.
2025_03_14-06_14_53_PM Initial preprocess and clustering
```

In [ ]:

```
doublet_vars = {'ME8_pass':{'doublet_cutoff':.15, 'doublet_clusters':['14']}
                'ME9_pass':{'doublet_cutoff':.18, 'doublet_clusters':['16']}
                'ME10_pass':{'doublet_cutoff':.18, 'doublet_clusters':['12']
                'ME11_pass':{'doublet_cutoff':.2, 'doublet_clusters':['22']}
                'ME12_pass':{'doublet_cutoff':.18, 'doublet_clusters':None},
}
doublet_vars = pd.DataFrame(doublet_vars).T

genes_to_plot = ['Sox10','Plp1','Epcam','Twist2','Prrx2','Tfap2b','Alx3','Dl

for index in list(doublet_vars.index):
    adata_obj = 'adata'+index
    if 'pass' in adata_obj:
        doublet_cutoff = doublet_vars.loc[index,'doublet_cutoff']
        doublet_clusters = doublet_vars.loc[index,'doublet_clusters']
        adata_files[adata_obj].obs['doublet_cutoff'] = 'pass'
        adata_files[adata_obj].obs.loc[adata_files[adata_obj].obs['doublet_s
        if isinstance(doublet_clusters, list):
            adata_files[adata_obj].obs.loc[adata_files[adata_obj].obs['leide
        adata_files[adata_obj].obs['doublet_cutoff'] = adata_files[adata_obj

        sn = adata_obj+'_doublet'
        adata_files[sn] = adata_files[adata_obj][adata_files[adata_obj].obs[
        adata_files[sn].X = adata_files[sn].layers["original_counts"].copy()
        print(time.strftime("%Y_%m_%d-%I_%M_%S_%p"),'Initial preprocess and
        ##standard normalization
        sc.pp.normalize_total(adata_files[sn]) # Normalizing to median total
        sc.pp.log1p(adata_files[sn]) # Logarithmize the data
        adata_files[sn].layers["normalized_counts"] = adata_files[sn].X.copy

        ##highly variable genes
        sc.pp.highly_variable_genes(adata_files[sn], n_top_genes=1000,)

        ##dimensionality reduction and clustering
        sc.tl.pca(adata_files[sn])
        sc.pp.neighbors(adata_files[sn])
        sc.tl.umap(adata_files[sn])
        sc.tl.leiden(adata_files[sn],resolution = 1, key_added = 'leiden_pos
        sc.tl.leiden(adata_files[sn],resolution = 2, key_added = 'leiden_pos

        ##plotting new stuff
        output_dir = 'figures_publish/'+species+'/'+genome+'/QC_plots/'
        !mkdir -p {output_dir}

        i = adata_obj.split('adata')[1].split('_')[0]
        with PdfPages(output_dir+i+'_post_QC.pdf') as pdf:
            ##umaps
            plt.rcParams['figure.figsize'] = [5,4]
            ax = sc.pl.umap(adata_files[sn],
                            color=['leiden_post_QC','leiden_post_QC_highres'
```

```
                                show = False,ncols = 2)
            for p in ax:
                p.set_rasterized(True)
            pdf.savefig(dpi=150)
            plt.close()

        adata_files[sn].write('h5ad_files/'+species+'/'+genome+'/'+i+'_after
```

In [17]:

```
## code for any markdown figures ##
output_dir = 'markdown_images/'+species+'/'+genome+'/markdown_plots/'
!mkdir -p {output_dir}

## let's start with ME8
i = 'ME8'
filter_vars = {'ME8':{'min_counts':10000, 'max_counts':100000, 'min_genes':2
}

dot_size = 1.5
linewidth = 0
scatter_color = 'pct_counts_hb'
scatter_cmap = 'coolwarm'
y1 = 'pct_counts_mt' #these are for the QC scatter plots
y2 = 'pct_counts_rb'
x1 = 'n_genes_by_counts'
x2 = 'total_counts'
fig, ((ax1), (ax2)) = plt.subplots(1,2,layout='constrained',figsize=(8,4))
s=dot_size
linewidth=linewidth
colors = dict(zip(list(adata_files['adata'+i].obs['leiden'].cat.categories),
                  list(adata_files['adata'+i].uns['leiden_colors'])))

ax1.scatter(data=adata_files['adata'+i].obs, x=x1,y=y1,
            s = s,linewidth=linewidth,
            c = adata_files['adata'+i].obs['leiden'].map(colors),
            rasterized=True)
ax1.axhline(y=filter_vars[i]['min_mito'],c = 'red', linewidth = .5)
ax1.axhline(y=filter_vars[i]['max_mito'],c = 'red', linewidth = .5)

ax1.axvline(x=filter_vars[i]['min_genes'],c = 'red', linewidth = .5)
ax1.axvline(x=filter_vars[i]['max_genes'],c = 'red', linewidth = .5)
ax1.set_xlabel(x1)
ax1.set_ylabel(y1)
ax1.set_ylim(0,3)
handles = [Line2D([0], [0], marker='o', color='w', markerfacecolor=v, label=
ax1.legend(title='leiden', handles=handles, bbox_to_anchor=(.5, 1), loc='upp
           ncol=3, fontsize = 9,labelspacing=0.2,columnspacing= .2,handletex

ax2.scatter(data=adata_files['adata'+i].obs, x=x1,y=y2,
            s = s,linewidth=linewidth,
            c = adata_files['adata'+i].obs['leiden'].map(colors),
            rasterized=True)

ax2.axhline(y=filter_vars[i]['min_rb'],c = 'red', linewidth = .5)
ax2.axhline(y=filter_vars[i]['max_rb'],c = 'red', linewidth = .5)
```

```
ax2.axvline(x=filter_vars[i]['min_genes'],c = 'red', linewidth = .5)
ax2.axvline(x=filter_vars[i]['max_genes'],c = 'red', linewidth = .5)

ax2.set_xlabel(x1)
ax2.set_ylabel(y2)
plt.savefig(output_dir+i+'_scatter_qc.png', dpi = 80)
plt.close()

#boxplot of doublet scores
fig, (ax1, ax2,ax3) = plt.subplots(3,1,layout='constrained',figsize=(8,8))

ax = sns.boxplot(data=adata_files['adata'+i].obs, x="leiden", y="pct_counts_
                 palette = adata_files['adata'+i].uns['leiden_colors'],
                 fliersize=0, ax = ax1)
ax1.axhline(y=filter_vars[i]['min_rb'],c = 'red', linewidth = .5)
ax1.axhline(y=filter_vars[i]['max_rb'],c = 'red', linewidth = .5)

cnum = 0
for line in ax.lines:
    if line.get_linestyle() == 'None':
        sns.swarmplot(x=[int(b) for b in line.get_xdata()], y=line.get_ydata
                      size = 1.5, color = adata_files['adata'+i].uns['leider
        cnum = cnum+1

ax = sns.boxplot(data=adata_files['adata'+i].obs, x="leiden", y="n_genes_by_
                 palette = adata_files['adata'+i].uns['leiden_colors'],
                 fliersize=0, ax = ax2)
ax2.axhline(y=filter_vars[i]['min_genes'],c = 'red', linewidth = .5)
ax2.axhline(y=filter_vars[i]['max_genes'],c = 'red', linewidth = .5)
cnum = 0
for line in ax.lines:
    if line.get_linestyle() == 'None':
        sns.swarmplot(x=[int(b) for b in line.get_xdata()], y=line.get_ydata
                      size = 1.5, color = adata_files['adata'+i].uns['leider
        cnum = cnum+1

ax = sns.boxplot(data=adata_files['adata'+i].obs, x="leiden", y="pct_counts_
                 palette = adata_files['adata'+i].uns['leiden_colors'],
                 fliersize=0, ax = ax3)
ax3.axhline(y=filter_vars[i]['min_mito'],c = 'red', linewidth = .5)
ax3.axhline(y=filter_vars[i]['max_mito'],c = 'red', linewidth = .5)
cnum = 0
for line in ax.lines:
    if line.get_linestyle() == 'None':
        sns.swarmplot(x=[int(b) for b in line.get_xdata()], y=line.get_ydata
                      size = 1.5, color = adata_files['adata'+i].uns['leider
        cnum = cnum+1

plt.savefig(output_dir+i+'_boxplot.png', dpi = 80)
plt.close()

plt.rcParams['figure.figsize'] = [4,3]
ax = sc.pl.umap(adata_files['adata'+i], color = ['leiden','Epcam','Pou3f1','
plt.savefig(output_dir+i+'_umap.png', dpi = 80,bbox_inches='tight')
plt.close()
```

```
i = 'ME9'
filter_vars = {'ME9':{'min_counts':9000, 'max_counts':100000, 'min_genes':40
}

dot_size = 1.5
linewidth = 0
scatter_color = 'pct_counts_hb'
scatter_cmap = 'coolwarm'
y1 = 'pct_counts_mt' #these are for the QC scatter plots
y2 = 'pct_counts_rb'
x1 = 'n_genes_by_counts'
x2 = 'total_counts'
fig, ((ax1), (ax2)) = plt.subplots(1,2,layout='constrained',figsize=(8,4))
s=dot_size
linewidth=linewidth
colors = dict(zip(list(adata_files['adata'+i].obs['leiden'].cat.categories),
                  list(adata_files['adata'+i].uns['leiden_colors'])))

ax1.scatter(data=adata_files['adata'+i].obs, x=x1,y=y1,
            s = s,linewidth=linewidth,
            c = adata_files['adata'+i].obs['leiden'].map(colors),
            rasterized=True)
ax1.axhline(y=filter_vars[i]['min_mito'],c = 'red', linewidth = .5)
ax1.axhline(y=filter_vars[i]['max_mito'],c = 'red', linewidth = .5)

ax1.axvline(x=filter_vars[i]['min_genes'],c = 'red', linewidth = .5)
ax1.axvline(x=filter_vars[i]['max_genes'],c = 'red', linewidth = .5)
ax1.set_xlabel(x1)
ax1.set_ylabel(y1)
ax1.set_ylim(0,3)
handles = [Line2D([0], [0], marker='o', color='w', markerfacecolor=v, label=
ax1.legend(title='leiden', handles=handles, bbox_to_anchor=(.5, 1), loc='upp
           ncol=3, fontsize = 9,labelspacing=0.2,columnspacing= .2,handletex

ax2.scatter(data=adata_files['adata'+i].obs, x=x1,y=y2,
            s = s,linewidth=linewidth,
            c = adata_files['adata'+i].obs['leiden'].map(colors),
            rasterized=True)

ax2.axhline(y=filter_vars[i]['min_rb'],c = 'red', linewidth = .5)
ax2.axhline(y=filter_vars[i]['max_rb'],c = 'red', linewidth = .5)

ax2.axvline(x=filter_vars[i]['min_genes'],c = 'red', linewidth = .5)
ax2.axvline(x=filter_vars[i]['max_genes'],c = 'red', linewidth = .5)

ax2.set_xlabel(x1)
ax2.set_ylabel(y2)
plt.savefig(output_dir+i+'_scatter_qc.png', dpi = 80)
plt.close()

#boxplot of doublet scores
fig, (ax1, ax2,ax3) = plt.subplots(3,1,layout='constrained',figsize=(8,8))

ax = sns.boxplot(data=adata_files['adata'+i].obs, x="leiden", y="pct_counts_
                 palette = adata_files['adata'+i].uns['leiden_colors'],
                 fliersize=0, ax = ax1)
```

```
ax1.axhline(y=filter_vars[i]['min_rb'],c = 'red', linewidth = .5)
ax1.axhline(y=filter_vars[i]['max_rb'],c = 'red', linewidth = .5)

cnum = 0
for line in ax.lines:
    if line.get_linestyle() == 'None':
        sns.swarmplot(x=[int(b) for b in line.get_xdata()], y=line.get_ydata
                      size = 1.5, color = adata_files['adata'+i].uns['leider
        cnum = cnum+1

ax = sns.boxplot(data=adata_files['adata'+i].obs, x="leiden", y="n_genes_by_
                 palette = adata_files['adata'+i].uns['leiden_colors'],
                 fliersize=0, ax = ax2)
ax2.axhline(y=filter_vars[i]['min_genes'],c = 'red', linewidth = .5)
ax2.axhline(y=filter_vars[i]['max_genes'],c = 'red', linewidth = .5)
cnum = 0
for line in ax.lines:
    if line.get_linestyle() == 'None':
        sns.swarmplot(x=[int(b) for b in line.get_xdata()], y=line.get_ydata
                      size = 1.5, color = adata_files['adata'+i].uns['leider
        cnum = cnum+1

ax = sns.boxplot(data=adata_files['adata'+i].obs, x="leiden", y="pct_counts_
                 palette = adata_files['adata'+i].uns['leiden_colors'],
                 fliersize=0, ax = ax3)
ax3.axhline(y=filter_vars[i]['min_mito'],c = 'red', linewidth = .5)
ax3.axhline(y=filter_vars[i]['max_mito'],c = 'red', linewidth = .5)
cnum = 0
for line in ax.lines:
    if line.get_linestyle() == 'None':
        sns.swarmplot(x=[int(b) for b in line.get_xdata()], y=line.get_ydata
                      size = 1.5, color = adata_files['adata'+i].uns['leider
        cnum = cnum+1

plt.savefig(output_dir+i+'_boxplot.png', dpi = 80)
plt.close()

plt.rcParams['figure.figsize'] = [4,3]
ax = sc.pl.umap(adata_files['adata'+i], color = ['leiden','Hbb-y','Fgf14','T
plt.savefig(output_dir+i+'_umap.png', dpi = 80,bbox_inches='tight')
plt.close()

i = 'ME10'
filter_vars = {'ME10':{'min_counts':5000, 'max_counts':100000, 'min_genes':2
}

dot_size = 1.5
linewidth = 0
scatter_color = 'pct_counts_hb'
scatter_cmap = 'coolwarm'
y1 = 'pct_counts_mt' #these are for the QC scatter plots
y2 = 'pct_counts_rb'
x1 = 'n_genes_by_counts'
x2 = 'total_counts'
fig, ((ax1), (ax2)) = plt.subplots(1,2,layout='constrained',figsize=(8,4))
s=dot_size
```

```
linewidth=linewidth
colors = dict(zip(list(adata_files['adata'+i].obs['leiden'].cat.categories),
                  list(adata_files['adata'+i].uns['leiden_colors'])))

ax1.scatter(data=adata_files['adata'+i].obs, x=x1,y=y1,
            s = s,linewidth=linewidth,
            c = adata_files['adata'+i].obs['leiden'].map(colors),
            rasterized=True)
ax1.axhline(y=filter_vars[i]['min_mito'],c = 'red', linewidth = .5)
ax1.axhline(y=filter_vars[i]['max_mito'],c = 'red', linewidth = .5)

ax1.axvline(x=filter_vars[i]['min_genes'],c = 'red', linewidth = .5)
ax1.axvline(x=filter_vars[i]['max_genes'],c = 'red', linewidth = .5)
ax1.set_xlabel(x1)
ax1.set_ylabel(y1)
ax1.set_ylim(0,3)
handles = [Line2D([0], [0], marker='o', color='w', markerfacecolor=v, label=
ax1.legend(title='leiden', handles=handles, bbox_to_anchor=(.5, 1), loc='upp
           ncol=3, fontsize = 9,labelspacing=0.2,columnspacing= .2,handletex

ax2.scatter(data=adata_files['adata'+i].obs, x=x1,y=y2,
            s = s,linewidth=linewidth,
            c = adata_files['adata'+i].obs['leiden'].map(colors),
            rasterized=True)

ax2.axhline(y=filter_vars[i]['min_rb'],c = 'red', linewidth = .5)
ax2.axhline(y=filter_vars[i]['max_rb'],c = 'red', linewidth = .5)

ax2.axvline(x=filter_vars[i]['min_genes'],c = 'red', linewidth = .5)
ax2.axvline(x=filter_vars[i]['max_genes'],c = 'red', linewidth = .5)

ax2.set_xlabel(x1)
ax2.set_ylabel(y2)
plt.savefig(output_dir+i+'_scatter_qc.png', dpi = 80)
plt.close()

#boxplot of doublet scores
fig, (ax1, ax2,ax3) = plt.subplots(3,1,layout='constrained',figsize=(8,8))

ax = sns.boxplot(data=adata_files['adata'+i].obs, x="leiden", y="pct_counts_
                 palette = adata_files['adata'+i].uns['leiden_colors'],
                 fliersize=0, ax = ax1)
ax1.axhline(y=filter_vars[i]['min_rb'],c = 'red', linewidth = .5)
ax1.axhline(y=filter_vars[i]['max_rb'],c = 'red', linewidth = .5)

cnum = 0
for line in ax.lines:
    if line.get_linestyle() == 'None':
        sns.swarmplot(x=[int(b) for b in line.get_xdata()], y=line.get_ydata
                      size = 1.5, color = adata_files['adata'+i].uns['leider
        cnum = cnum+1

ax = sns.boxplot(data=adata_files['adata'+i].obs, x="leiden", y="n_genes_by_
                 palette = adata_files['adata'+i].uns['leiden_colors'],
                 fliersize=0, ax = ax2)
ax2.axhline(y=filter_vars[i]['min_genes'],c = 'red', linewidth = .5)
```

```
ax2.axhline(y=filter_vars[i]['max_genes'],c = 'red', linewidth = .5)
cnum = 0
for line in ax.lines:
    if line.get_linestyle() == 'None':
        sns.swarmplot(x=[int(b) for b in line.get_xdata()], y=line.get_ydata
                      size = 1.5, color = adata_files['adata'+i].uns['leider
        cnum = cnum+1

ax = sns.boxplot(data=adata_files['adata'+i].obs, x="leiden", y="pct_counts_
                 palette = adata_files['adata'+i].uns['leiden_colors'],
                 fliersize=0, ax = ax3)
ax3.axhline(y=filter_vars[i]['min_mito'],c = 'red', linewidth = .5)
ax3.axhline(y=filter_vars[i]['max_mito'],c = 'red', linewidth = .5)
cnum = 0
for line in ax.lines:
    if line.get_linestyle() == 'None':
        sns.swarmplot(x=[int(b) for b in line.get_xdata()], y=line.get_ydata
                      size = 1.5, color = adata_files['adata'+i].uns['leider
        cnum = cnum+1

plt.savefig(output_dir+i+'_boxplot.png', dpi = 80)
plt.close()

plt.rcParams['figure.figsize'] = [4,3]
ax = sc.pl.umap(adata_files['adata'+i], color = ['leiden','Epcam','Hbb-bt'],
plt.savefig(output_dir+i+'_umap.png', dpi = 80,bbox_inches='tight')
plt.close()

i = 'ME11'
filter_vars = {'ME11':{'min_counts':6000, 'max_counts':100000, 'min_genes':3
}

dot_size = 1.5
linewidth = 0
scatter_color = 'pct_counts_hb'
scatter_cmap = 'coolwarm'
y1 = 'pct_counts_mt' #these are for the QC scatter plots
y2 = 'pct_counts_rb'
x1 = 'n_genes_by_counts'
x2 = 'total_counts'
fig, ((ax1), (ax2)) = plt.subplots(1,2,layout='constrained',figsize=(8,4))
s=dot_size
linewidth=linewidth
colors = dict(zip(list(adata_files['adata'+i].obs['leiden'].cat.categories),
                  list(adata_files['adata'+i].uns['leiden_colors'])))

ax1.scatter(data=adata_files['adata'+i].obs, x=x1,y=y1,
            s = s,linewidth=linewidth,
            c = adata_files['adata'+i].obs['leiden'].map(colors),
            rasterized=True)
ax1.axhline(y=filter_vars[i]['min_mito'],c = 'red', linewidth = .5)
ax1.axhline(y=filter_vars[i]['max_mito'],c = 'red', linewidth = .5)

ax1.axvline(x=filter_vars[i]['min_genes'],c = 'red', linewidth = .5)
ax1.axvline(x=filter_vars[i]['max_genes'],c = 'red', linewidth = .5)
ax1.set_xlabel(x1)
```

```
ax1.set_ylabel(y1)
ax1.set_ylim(0,3)
handles = [Line2D([0], [0], marker='o', color='w', markerfacecolor=v, label=
ax1.legend(title='leiden', handles=handles, bbox_to_anchor=(.5, 1), loc='upp
           ncol=3, fontsize = 9,labelspacing=0.2,columnspacing= .2,handletex

ax2.scatter(data=adata_files['adata'+i].obs, x=x1,y=y2,
            s = s,linewidth=linewidth,
            c = adata_files['adata'+i].obs['leiden'].map(colors),
            rasterized=True)

ax2.axhline(y=filter_vars[i]['min_rb'],c = 'red', linewidth = .5)
ax2.axhline(y=filter_vars[i]['max_rb'],c = 'red', linewidth = .5)

ax2.axvline(x=filter_vars[i]['min_genes'],c = 'red', linewidth = .5)
ax2.axvline(x=filter_vars[i]['max_genes'],c = 'red', linewidth = .5)

ax2.set_xlabel(x1)
ax2.set_ylabel(y2)
plt.savefig(output_dir+i+'_scatter_qc.png', dpi = 80)
plt.close()

#boxplot of doublet scores
fig, (ax1, ax2,ax3) = plt.subplots(3,1,layout='constrained',figsize=(8,8))

ax = sns.boxplot(data=adata_files['adata'+i].obs, x="leiden", y="pct_counts_
                 palette = adata_files['adata'+i].uns['leiden_colors'],
                 fliersize=0, ax = ax1)
ax1.axhline(y=filter_vars[i]['min_rb'],c = 'red', linewidth = .5)
ax1.axhline(y=filter_vars[i]['max_rb'],c = 'red', linewidth = .5)

cnum = 0
for line in ax.lines:
    if line.get_linestyle() == 'None':
        sns.swarmplot(x=[int(b) for b in line.get_xdata()], y=line.get_ydata
                      size = 1.5, color = adata_files['adata'+i].uns['leider
        cnum = cnum+1

ax = sns.boxplot(data=adata_files['adata'+i].obs, x="leiden", y="n_genes_by_
                 palette = adata_files['adata'+i].uns['leiden_colors'],
                 fliersize=0, ax = ax2)
ax2.axhline(y=filter_vars[i]['min_genes'],c = 'red', linewidth = .5)
ax2.axhline(y=filter_vars[i]['max_genes'],c = 'red', linewidth = .5)
cnum = 0
for line in ax.lines:
    if line.get_linestyle() == 'None':
        sns.swarmplot(x=[int(b) for b in line.get_xdata()], y=line.get_ydata
                      size = 1.5, color = adata_files['adata'+i].uns['leider
        cnum = cnum+1

ax = sns.boxplot(data=adata_files['adata'+i].obs, x="leiden", y="pct_counts_
                 palette = adata_files['adata'+i].uns['leiden_colors'],
                 fliersize=0, ax = ax3)
ax3.axhline(y=filter_vars[i]['min_mito'],c = 'red', linewidth = .5)
ax3.axhline(y=filter_vars[i]['max_mito'],c = 'red', linewidth = .5)
cnum = 0
```

```
for line in ax.lines:
    if line.get_linestyle() == 'None':
        sns.swarmplot(x=[int(b) for b in line.get_xdata()], y=line.get_ydata
                      size = 1.5, color = adata_files['adata'+i].uns['leider
        cnum = cnum+1

plt.savefig(output_dir+i+'_boxplot.png', dpi = 80)
plt.close()

plt.rcParams['figure.figsize'] = [4,3]
ax = sc.pl.umap(adata_files['adata'+i], color = ['leiden','Hbb-y','Epcam','p
plt.savefig(output_dir+i+'_umap.png', dpi = 80,bbox_inches='tight')
plt.close()

plt.rcParams['figure.figsize'] = [4,3]
ax = sc.pl.umap(adata_files['adata'+i], color = ['phase',], ncols = 3,show =
plt.savefig(output_dir+i+'phase_umap.png', dpi = 80,bbox_inches='tight')
plt.close()

filter_vars = {'ME11':{'min_counts':6000, 'max_counts':100000, 'min_genes':3
}

fig, ((ax1), (ax2)) = plt.subplots(1,2,layout='constrained',figsize=(8,4))
s=dot_size
linewidth=linewidth
colors = dict(zip(list(adata_files['adata'+i].obs['leiden'].cat.categories),
                  list(adata_files['adata'+i].uns['leiden_colors'])))

ax1.scatter(data=adata_files['adata'+i].obs, x=x1,y=y1,
            s = s,linewidth=linewidth,
            c = adata_files['adata'+i].obs['leiden'].map(colors),
            rasterized=True)
ax1.axhline(y=filter_vars[i]['min_mito'],c = 'red', linewidth = .5)
ax1.axhline(y=filter_vars[i]['max_mito'],c = 'red', linewidth = .5)

ax1.axvline(x=filter_vars[i]['min_genes'],c = 'red', linewidth = .5)
ax1.axvline(x=filter_vars[i]['max_genes'],c = 'red', linewidth = .5)
ax1.set_xlabel(x1)
ax1.set_ylabel(y1)
ax1.set_ylim(0,3)
handles = [Line2D([0], [0], marker='o', color='w', markerfacecolor=v, label=
ax1.legend(title='leiden', handles=handles, bbox_to_anchor=(.5, 1), loc='upp
           ncol=3, fontsize = 9,labelspacing=0.2,columnspacing= .2,handletex

ax2.scatter(data=adata_files['adata'+i].obs, x=x1,y=y2,
            s = s,linewidth=linewidth,
            c = adata_files['adata'+i].obs['leiden'].map(colors),
            rasterized=True)

ax2.axhline(y=filter_vars[i]['min_rb'],c = 'red', linewidth = .5)
ax2.axhline(y=filter_vars[i]['max_rb'],c = 'red', linewidth = .5)

ax2.axvline(x=filter_vars[i]['min_genes'],c = 'red', linewidth = .5)
ax2.axvline(x=filter_vars[i]['max_genes'],c = 'red', linewidth = .5)

ax2.set_xlabel(x1)
```

```
ax2.set_ylabel(y2)
plt.savefig(output_dir+i+'_scatter_adjusted_qc.png', dpi = 80)
plt.close()

i = 'ME12'
filter_vars = {'ME12':{'min_counts':2000, 'max_counts':100000, 'min_genes':2
}

dot_size = 1.5
linewidth = 0
scatter_color = 'pct_counts_hb'
scatter_cmap = 'coolwarm'
y1 = 'pct_counts_mt' #these are for the QC scatter plots
y2 = 'pct_counts_rb'
x1 = 'n_genes_by_counts'
x2 = 'total_counts'
fig, ((ax1), (ax2)) = plt.subplots(1,2,layout='constrained',figsize=(8,4))
s=dot_size
linewidth=linewidth
colors = dict(zip(list(adata_files['adata'+i].obs['leiden'].cat.categories),
                  list(adata_files['adata'+i].uns['leiden_colors'])))

ax1.scatter(data=adata_files['adata'+i].obs, x=x1,y=y1,
            s = s,linewidth=linewidth,
            c = adata_files['adata'+i].obs['leiden'].map(colors),
            rasterized=True)
ax1.axhline(y=filter_vars[i]['min_mito'],c = 'red', linewidth = .5)
ax1.axhline(y=filter_vars[i]['max_mito'],c = 'red', linewidth = .5)

ax1.axvline(x=filter_vars[i]['min_genes'],c = 'red', linewidth = .5)
ax1.axvline(x=filter_vars[i]['max_genes'],c = 'red', linewidth = .5)
ax1.set_xlabel(x1)
ax1.set_ylabel(y1)
ax1.set_ylim(0,15)
handles = [Line2D([0], [0], marker='o', color='w', markerfacecolor=v, label=
ax1.legend(title='leiden', handles=handles, bbox_to_anchor=(.5, 1), loc='upp
           ncol=3, fontsize = 9,labelspacing=0.2,columnspacing= .2,handletex

ax2.scatter(data=adata_files['adata'+i].obs, x=x1,y=y2,
            s = s,linewidth=linewidth,
            c = adata_files['adata'+i].obs['leiden'].map(colors),
            rasterized=True)

ax2.axhline(y=filter_vars[i]['min_rb'],c = 'red', linewidth = .5)
ax2.axhline(y=filter_vars[i]['max_rb'],c = 'red', linewidth = .5)

ax2.axvline(x=filter_vars[i]['min_genes'],c = 'red', linewidth = .5)
ax2.axvline(x=filter_vars[i]['max_genes'],c = 'red', linewidth = .5)

ax2.set_xlabel(x1)
ax2.set_ylabel(y2)
plt.savefig(output_dir+i+'_scatter_qc.png', dpi = 80)
plt.close()

#boxplot of doublet scores
fig, (ax1, ax2,ax3) = plt.subplots(3,1,layout='constrained',figsize=(8,8))
```

```
ax = sns.boxplot(data=adata_files['adata'+i].obs, x="leiden", y="pct_counts_
                 palette = adata_files['adata'+i].uns['leiden_colors'],
                 fliersize=0, ax = ax1)
ax1.axhline(y=filter_vars[i]['min_rb'],c = 'red', linewidth = .5)
ax1.axhline(y=filter_vars[i]['max_rb'],c = 'red', linewidth = .5)

cnum = 0
for line in ax.lines:
    if line.get_linestyle() == 'None':
        sns.swarmplot(x=[int(b) for b in line.get_xdata()], y=line.get_ydata
                      size = 1.5, color = adata_files['adata'+i].uns['leider
        cnum = cnum+1

ax = sns.boxplot(data=adata_files['adata'+i].obs, x="leiden", y="n_genes_by_
                 palette = adata_files['adata'+i].uns['leiden_colors'],
                 fliersize=0, ax = ax2)
ax2.axhline(y=filter_vars[i]['min_genes'],c = 'red', linewidth = .5)
ax2.axhline(y=filter_vars[i]['max_genes'],c = 'red', linewidth = .5)
cnum = 0
for line in ax.lines:
    if line.get_linestyle() == 'None':
        sns.swarmplot(x=[int(b) for b in line.get_xdata()], y=line.get_ydata
                      size = 1.5, color = adata_files['adata'+i].uns['leider
        cnum = cnum+1

ax = sns.boxplot(data=adata_files['adata'+i].obs, x="leiden", y="pct_counts_
                 palette = adata_files['adata'+i].uns['leiden_colors'],
                 fliersize=0, ax = ax3)
ax3.axhline(y=filter_vars[i]['min_mito'],c = 'red', linewidth = .5)
ax3.axhline(y=filter_vars[i]['max_mito'],c = 'red', linewidth = .5)
cnum = 0
for line in ax.lines:
    if line.get_linestyle() == 'None':
        sns.swarmplot(x=[int(b) for b in line.get_xdata()], y=line.get_ydata
                      size = 1.5, color = adata_files['adata'+i].uns['leider
        cnum = cnum+1

plt.savefig(output_dir+i+'_boxplot.png', dpi = 80)
plt.close()

plt.rcParams['figure.figsize'] = [4,3]
ax = sc.pl.umap(adata_files['adata'+i], color = ['leiden','Tubb3','pct_count
plt.savefig(output_dir+i+'_umap.png', dpi = 80,bbox_inches='tight')
plt.close()

plt.rcParams['figure.figsize'] = [4,3]
ax = sc.pl.umap(adata_files['adata'+i], color = ['phase',], ncols = 3,show =
plt.savefig(output_dir+i+'phase_umap.png', dpi = 80,bbox_inches='tight')
plt.close()

filter_vars = {'ME12':{'min_counts':2000, 'max_counts':100000, 'min_genes':2
}


fig, ((ax1), (ax2)) = plt.subplots(1,2,layout='constrained',figsize=(8,4))
```

```
s=dot_size
linewidth=linewidth
colors = dict(zip(list(adata_files['adata'+i].obs['leiden'].cat.categories),
                  list(adata_files['adata'+i].uns['leiden_colors'])))

ax1.scatter(data=adata_files['adata'+i].obs, x=x1,y=y1,
            s = s,linewidth=linewidth,
            c = adata_files['adata'+i].obs['leiden'].map(colors),
            rasterized=True)
ax1.axhline(y=filter_vars[i]['min_mito'],c = 'red', linewidth = .5)
ax1.axhline(y=filter_vars[i]['max_mito'],c = 'red', linewidth = .5)

ax1.axvline(x=filter_vars[i]['min_genes'],c = 'red', linewidth = .5)
ax1.axvline(x=filter_vars[i]['max_genes'],c = 'red', linewidth = .5)
ax1.set_xlabel(x1)
ax1.set_ylabel(y1)
ax1.set_ylim(0,15)
handles = [Line2D([0], [0], marker='o', color='w', markerfacecolor=v, label=
ax1.legend(title='leiden', handles=handles, bbox_to_anchor=(.5, 1), loc='upp
           ncol=3, fontsize = 9,labelspacing=0.2,columnspacing= .2,handletex

ax2.scatter(data=adata_files['adata'+i].obs, x=x1,y=y2,
            s = s,linewidth=linewidth,
            c = adata_files['adata'+i].obs['leiden'].map(colors),
            rasterized=True)

ax2.axhline(y=filter_vars[i]['min_rb'],c = 'red', linewidth = .5)
ax2.axhline(y=filter_vars[i]['max_rb'],c = 'red', linewidth = .5)

ax2.axvline(x=filter_vars[i]['min_genes'],c = 'red', linewidth = .5)
ax2.axvline(x=filter_vars[i]['max_genes'],c = 'red', linewidth = .5)

ax2.set_xlabel(x1)
ax2.set_ylabel(y2)
plt.savefig(output_dir+i+'_scatter_adjusted_qc.png', dpi = 80)
plt.close()

##Post filter plots
filter_vars = {'ME8':{'min_counts':10000, 'max_counts':100000, 'min_genes':2
                      'clusters':['12','15','18','20','22']},
               'ME9':{'min_counts':9000, 'max_counts':100000, 'min_genes':40
                      'clusters':['0','2','9','10','12','14','19','22']},
               'ME10':{'min_counts':5000, 'max_counts':100000, 'min_genes':2
                      'clusters':['11','12','16','20']},
               'ME11':{'min_counts':6000, 'max_counts':100000, 'min_genes':3
                      'clusters':['0','3','15','16','20']}, #adjusted
               'ME12':{'min_counts':2000, 'max_counts':100000, 'min_genes':2
                      'clusters':['0','1','15','23','24']}, #adjusted
}

filter_vars = pd.DataFrame(filter_vars).T

for sample in list(filter_vars.index):
    min_counts = filter_vars.loc[sample,'min_counts']
    max_counts = filter_vars.loc[sample,'max_counts']
    min_genes = filter_vars.loc[sample,'min_genes']
```

```
    max_genes = filter_vars.loc[sample,'max_genes']
    min_mito = filter_vars.loc[sample,'min_mito']
    max_mito = filter_vars.loc[sample,'max_mito']
    min_rb = filter_vars.loc[sample,'min_rb']
    max_rb = filter_vars.loc[sample,'max_rb']

    conditions = [
    (adata_files['adata'+sample].obs['predicted_doublet'] == True),
    (adata_files['adata'+sample].obs['n_genes_by_counts'] < min_genes),
    (adata_files['adata'+sample].obs['n_genes_by_counts'] > max_genes),
    (adata_files['adata'+sample].obs['total_counts'] < min_counts),
    (adata_files['adata'+sample].obs['total_counts'] > max_counts),
    (adata_files['adata'+sample].obs['pct_counts_mt'] > max_mito),
    (adata_files['adata'+sample].obs['pct_counts_mt'] < min_mito),
    (adata_files['adata'+sample].obs['pct_counts_rb'] < min_rb),
    (adata_files['adata'+sample].obs['pct_counts_rb'] > max_rb),
    (adata_files['adata'+sample].obs['pct_counts_mt'] <= max_mito) & (adata_
    (adata_files['adata'+sample].obs['total_counts'] >= min_counts) & (adata
    (adata_files['adata'+sample].obs['predicted_doublet'] != True)
    ]

    values = ['Fail','Fail', 'Fail', 'Fail','Fail', 'Fail','Fail', 'Fail' ,'
    adata_files['adata'+sample].obs['QC'] = np.select(conditions, values)
    adata_files['adata'+sample].obs.loc[adata_files['adata'+sample].obs['lei
    adata_files['adata'+sample].obs['QC'] = adata_files['adata'+sample].obs[

    plt.rcParams['figure.figsize'] = [4,3]
    ax = sc.pl.umap(adata_files['adata'+sample], color = ['QC',], ncols = 3,
    plt.savefig(output_dir+sample+'_QC_umap.png', dpi = 80,bbox_inches='tigh
    plt.close()


####Doublet stuff####
#ME8
i = 'ME8_pass'
doublet_cutoff = .15
doublet_clusters = ['14']

plt.rcParams['figure.figsize'] = [4,3]
ax = sc.pl.umap(adata_files['adata'+i], color = ['leiden_filt','doublet_scor
plt.savefig(output_dir+i+'_umap.png', dpi = 80,bbox_inches='tight')
plt.close()

plt.rcParams['figure.figsize'] = [6,3]
ax = sns.boxplot(data=adata_files['adata'+i].obs, x="leiden_filt", y="double
                 palette = adata_files['adata'+i].uns['leiden_filt_colors'],
                 fliersize=0)
cnum = 0
for line in ax.lines:
    if line.get_linestyle() == 'None':
        sns.swarmplot(x=[int(b) for b in line.get_xdata()], y=line.get_ydata
                      size = 1.5, color = adata_files['adata'+i].uns['leider
        cnum = cnum+1
plt.savefig(output_dir+i+'_doublet_boxplot.png',dpi=150, bbox_inches = 'tigh
plt.close()
```

```
adata_files['adata'+i].obs['doublet_cutoff'] = 'pass'
adata_files['adata'+i].obs.loc[adata_files['adata'+i].obs['doublet_score'] >
adata_files['adata'+i].obs.loc[adata_files['adata'+i].obs['leiden_filt'].isi
adata_files['adata'+i].obs['doublet_cutoff'] = adata_files['adata'+i].obs['c
plt.rcParams['figure.figsize'] = [4,3]
ax = sc.pl.umap(adata_files['adata'+i], color = 'doublet_cutoff', show = Fal
plt.savefig(output_dir+i+'_doublet_umap.png', dpi = 80,bbox_inches='tight')
plt.close()


#ME9
i = 'ME9_pass'
doublet_cutoff = .18
doublet_clusters = ['16']

plt.rcParams['figure.figsize'] = [4,3]
ax = sc.pl.umap(adata_files['adata'+i], color = ['leiden_filt','doublet_scor
plt.savefig(output_dir+i+'_umap.png', dpi = 80,bbox_inches='tight')
plt.close()

plt.rcParams['figure.figsize'] = [6,3]
ax = sns.boxplot(data=adata_files['adata'+i].obs, x="leiden_filt", y="double
                 palette = adata_files['adata'+i].uns['leiden_filt_colors'],
                 fliersize=0)
cnum = 0
for line in ax.lines:
    if line.get_linestyle() == 'None':
        sns.swarmplot(x=[int(b) for b in line.get_xdata()], y=line.get_ydata
                      size = 1.5, color = adata_files['adata'+i].uns['leider
        cnum = cnum+1
plt.savefig(output_dir+i+'_doublet_boxplot.png',dpi=150, bbox_inches = 'tigh
plt.close()

adata_files['adata'+i].obs['doublet_cutoff'] = 'pass'
adata_files['adata'+i].obs.loc[adata_files['adata'+i].obs['doublet_score'] >
adata_files['adata'+i].obs.loc[adata_files['adata'+i].obs['leiden_filt'].isi
adata_files['adata'+i].obs['doublet_cutoff'] = adata_files['adata'+i].obs['c
plt.rcParams['figure.figsize'] = [4,3]
ax = sc.pl.umap(adata_files['adata'+i], color = 'doublet_cutoff', show = Fal
plt.savefig(output_dir+i+'_doublet_umap.png', dpi = 80,bbox_inches='tight')
plt.close()

#ME10
i = 'ME10_pass'
doublet_cutoff = .18
doublet_clusters = ['12']

plt.rcParams['figure.figsize'] = [4,3]
ax = sc.pl.umap(adata_files['adata'+i], color = ['leiden_filt','doublet_scor
plt.savefig(output_dir+i+'_umap.png', dpi = 80,bbox_inches='tight')
plt.close()

plt.rcParams['figure.figsize'] = [6,3]
ax = sns.boxplot(data=adata_files['adata'+i].obs, x="leiden_filt", y="double
                 palette = adata_files['adata'+i].uns['leiden_filt_colors'],
                 fliersize=0)
```

```
cnum = 0
for line in ax.lines:
    if line.get_linestyle() == 'None':
        sns.swarmplot(x=[int(b) for b in line.get_xdata()], y=line.get_ydata
                      size = 1.5, color = adata_files['adata'+i].uns['leider
        cnum = cnum+1
plt.savefig(output_dir+i+'_doublet_boxplot.png',dpi=150, bbox_inches = 'tigh
plt.close()

adata_files['adata'+i].obs['doublet_cutoff'] = 'pass'
adata_files['adata'+i].obs.loc[adata_files['adata'+i].obs['doublet_score'] >
adata_files['adata'+i].obs.loc[adata_files['adata'+i].obs['leiden_filt'].isi
adata_files['adata'+i].obs['doublet_cutoff'] = adata_files['adata'+i].obs['c
plt.rcParams['figure.figsize'] = [4,3]
ax = sc.pl.umap(adata_files['adata'+i], color = 'doublet_cutoff', show = Fal
plt.savefig(output_dir+i+'_doublet_umap.png', dpi = 80,bbox_inches='tight')
plt.close()

#ME11
i = 'ME11_pass'
doublet_cutoff = .2
doublet_clusters = ['22']

plt.rcParams['figure.figsize'] = [4,3]
ax = sc.pl.umap(adata_files['adata'+i], color = ['leiden_filt','doublet_scor
plt.savefig(output_dir+i+'_umap.png', dpi = 80,bbox_inches='tight')
plt.close()

plt.rcParams['figure.figsize'] = [6,3]
ax = sns.boxplot(data=adata_files['adata'+i].obs, x="leiden_filt", y="double
                 palette = adata_files['adata'+i].uns['leiden_filt_colors'],
                 fliersize=0)
cnum = 0
for line in ax.lines:
    if line.get_linestyle() == 'None':
        sns.swarmplot(x=[int(b) for b in line.get_xdata()], y=line.get_ydata
                      size = 1.5, color = adata_files['adata'+i].uns['leider
        cnum = cnum+1
plt.savefig(output_dir+i+'_doublet_boxplot.png',dpi=150, bbox_inches = 'tigh
plt.close()

adata_files['adata'+i].obs['doublet_cutoff'] = 'pass'
adata_files['adata'+i].obs.loc[adata_files['adata'+i].obs['doublet_score'] >
adata_files['adata'+i].obs.loc[adata_files['adata'+i].obs['leiden_filt'].isi
adata_files['adata'+i].obs['doublet_cutoff'] = adata_files['adata'+i].obs['c
plt.rcParams['figure.figsize'] = [4,3]
ax = sc.pl.umap(adata_files['adata'+i], color = 'doublet_cutoff', show = Fal
plt.savefig(output_dir+i+'_doublet_umap.png', dpi = 80,bbox_inches='tight')
plt.close()

#ME12
i = 'ME12_pass'
doublet_cutoff = .18
#doublet_clusters = ['22']

plt.rcParams['figure.figsize'] = [4,3]
```

```
ax = sc.pl.umap(adata_files['adata'+i], color = ['leiden_filt','doublet_scor
plt.savefig(output_dir+i+'_umap.png', dpi = 80,bbox_inches='tight')
plt.close()

plt.rcParams['figure.figsize'] = [6,3]
ax = sns.boxplot(data=adata_files['adata'+i].obs, x="leiden_filt", y="double
                 palette = adata_files['adata'+i].uns['leiden_filt_colors'],
                 fliersize=0)
cnum = 0
for line in ax.lines:
    if line.get_linestyle() == 'None':
        sns.swarmplot(x=[int(b) for b in line.get_xdata()], y=line.get_ydata
                      size = 1.5, color = adata_files['adata'+i].uns['leider
        cnum = cnum+1
plt.savefig(output_dir+i+'_doublet_boxplot.png',dpi=150, bbox_inches = 'tigh
plt.close()

adata_files['adata'+i].obs['doublet_cutoff'] = 'pass'
adata_files['adata'+i].obs.loc[adata_files['adata'+i].obs['doublet_score'] >
#adata_files['adata'+i].obs.loc[adata_files['adata'+i].obs['leiden_filt'].is
adata_files['adata'+i].obs['doublet_cutoff'] = adata_files['adata'+i].obs['c
plt.rcParams['figure.figsize'] = [4,3]
ax = sc.pl.umap(adata_files['adata'+i], color = 'doublet_cutoff', size = 4,
plt.savefig(output_dir+i+'_doublet_umap.png', dpi = 80,bbox_inches='tight')
plt.close()
```

In [9]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

```
######Mouse mm39######
start_time=time.strftime("%Y_%m_%d-%I_%M_%S_%p")
print('start time:',start_time)

sample_names = ['ME8','ME9','ME10','ME11','ME12'] #specify the sample names
species = 'mouse' #specify the species
genome = 'mm39' #specify the genome
output_prefix = 'cellranger_related/cellranger_output/sc_rnaseq/' #specify t

cellranger_locs = []
velocyto_locs = []

for sample in sample_names: #simple loop to create a list with the location
    cellranger_locs = cellranger_locs + [output_prefix+species+'/'+genome+'/
    velocyto_locs = velocyto_locs + [output_prefix+species+'/'+genome+'/'+sa

adata_files={}
adata_list=[]
print(time.strftime("%Y_%m_%d-%I_%M_%S_%p"),'Reading file(s)')
for i,j,k in zip(sample_names, cellranger_locs, velocyto_locs):
    adata_cellr = sc.read_10x_mtx(j)
    adata_cellr.obs.index = adata_cellr.obs.index.str[:-2]
    adata_cellr.var['gene_name'] = adata_cellr.var.index
    adata_cellr.var.index = adata_cellr.var['gene_ids']
    adata_veloc = sc.read(k)
    adata_veloc.obs.index = adata_veloc.obs.index.str.split(':').str[1]
    adata_veloc.obs.index = adata_veloc.obs.index.str[:-1]
    adata_veloc.var['gene_name'] = adata_veloc.var.index
    adata_veloc.var.index = adata_veloc.var['Accession']
    adata_merge = scv.utils.merge(adata_cellr, adata_veloc)
    adata_files['adata'+i]=adata_merge
    adata_files['adata'+i].obs['sample']=i
    adata_files['adata'+i].obs['barcode']=adata_files['adata'+i].obs.index
    adata_files['adata'+i].obs.index=adata_files['adata'+i].obs['barcode']+'
    adata_list.append('adata'+i)
    del adata_cellr
    del adata_veloc
    del adata_merge
    gc.collect()
print(time.strftime("%Y_%m_%d-%I_%M_%S_%p"),'Finished Reading file(s)')
```