BarCluster Tutorial

Lee Richman

01/11/2022

Walkthrough

Welcome to BarCluster. This RMarkDown file is a tutorial using sample data to generate hybrid clusters from lineage barcoded single cell sequencing data. In the naive clustering setting, the absence of known “ground truth” cell types makes the significance of clusters uncertain. Using BarCluster will better align clusters with the ground truth represented by lineage barcodes. We will:

1. Read in our data (count matrix and barcode assignments).
2. Run PCA on our count matrix.
3. Create our hybrid clusters across a range of values and plot concordance.
4. Determine alpha values for analysis based on cluster number at fixed resolution.
5. Generate Sankey plots to show cluster rearrangement.
6. Identify cluster markers with ROC analysis.
7. Plot a Sankey for marker positivity.
8. Plot marker strength.
9. Perform UMAP and warp it!

The sample data provided is derived from one of our published samples (link), reduced in size and complexity to host on GitHub. It consists of the top 15 barcodes plus 100 singlet cells and the 501 most variable genes in replicate 1 of high dose BRAF treated WM989 cells from Goyal *et al. biorXiv* 2021 (https://www.biorxiv.org/content/10.1101/2021.12.08.471833v1).

Part 1: Setup

Load required packages, get our tutorial data files:

```
library(data.table)
library(magrittr)
library(Seurat)
library(BarCluster)
library(ggplot2)
library(pheatmap)
library(cowplot)

# get the location of the sample data installed with barcluster
dir <- system.file(package = "BarCluster") %>% file.path(., "extdata")

# count matrix file
cm <- file.path(dir, "YG1_sample_genes.txt")

# barcode assignment file
bt <- file.path(dir, "YG1_sample_barcodes.txt")
```

Now lets read in and see what our files look like.

```
# read in barcodes
bt <- data.table::fread(bt)

# what does it look like
head(bt)
```

```
##                 rn Barcode
## 1: AAACGAAAGAGGGTCT      C9
## 2: AAAGGTAAGGTAGTAT      C9
## 3: AAAGGTACACCATTCC      C9
## 4: AAAGGTACAGTAGTGG      C9
## 5: AAAGTGAAGGCCACCT      C9
## 6: AACCATGGTTTGACAC      C9
```

```
# read in count matrix
cm <- data.table::fread(cm)

# what does it look like
cm[1:5,1:5]
```

```
##       rn AAACGAAAGAGGGTCT AAAGGTAAGGTAGTAT AAAGGTACACCATTCC AAAGGTACAGTAGTGG
## 1: ACTA2       -0.9795829       -1.5483850        0.6847103       -1.2496894
## 2:  MYLK       -1.8769908       -1.8002466        0.8519252       -1.3883718
## 3: S100B       -1.5113756       -0.8227752       -0.9428514       -1.1884587
## 4: CRYAB       -1.8161940       -1.7863511       -1.8369420       -1.4849199
## 5:  CCL5       -1.1543626       -0.2587900       -1.1720407       -0.8835463
```

Everything read in correctly but the count matrix is formatted with cells as columns (a common output format). BarCluster expects cells as rows. No problem, I've included the functions `tdt` and `dt2m` in BarCluster to reformat the data.table and turn it into a matrix, respectively.

```
# transpose the data table we read in to get cells as rows using the tdt function
cm <- tdt(cm)

# convert data table to matrix using the dt2m function
cm <- dt2m(cm)

# what does it look like?
cm[1:5, 1:5]
```

```
##                       ACTA2       MYLK      S100B     CRYAB       CCL5
## AAACGAAAGAGGGTCT -0.9795829 -1.8769908 -1.5113756 -1.816194 -1.1543626
## AAAGGTAAGGTAGTAT -1.5483850 -1.8002466 -0.8227752 -1.786351 -0.2587900
## AAAGGTACACCATTCC  0.6847103  0.8519252 -0.9428514 -1.836942 -1.1720407
## AAAGGTACAGTAGTGG -1.2496894 -1.3883718 -1.1884587 -1.484920 -0.8835463
## AAAGTGAAGGCCACCT -0.7632957 -1.6382829 -1.4243413 -1.012571 -1.0797454
```

Part 2: Perform PCA.

The next step is to generate our PCA matrix. The function `irlba_wrap` is a wrapper that will do this for you. You can set the random seed and the number of output PCs if you want, see `?irlba_wrap`. For this analysis, let's summarize our 501 genes to 25 PCs:

```
# using 25 PCs
pca <- irlba_wrap(cm, npc = 25)

# peak at the first 5 columns and rows
pca[1:5, 1:5]
```

```
##                       PC_1       PC_2       PC_3       PC_4      PC_5
## AAACGAAAGAGGGTCT 22.84446  19.888593 -32.175042  32.374631 -4.999031
## AAAGGTAAGGTAGTAT 22.32305  -5.231187 -21.544455 -15.831288 -1.927493
## AAAGGTACACCATTCC 30.69033   4.068853 -27.413052  32.316806 -6.976072
## AAAGGTACAGTAGTGG 13.27684 -15.774159  -3.780693  -9.031577 -6.779002
## AAAGTGAAGGCCACCT 20.12647  -3.578279 -17.679333 -12.897062  2.571924
```

Part 3: Generate hybrid clusters across a range of alpha values.

The steps to perform hybrid clustering are:

1. Generate transcriptome shared nearest neighbor graph
2. Generate size normalized barcode graph
3. Integrate the two with the BarCluster model
4. Repeat for many values of alpha to determine max alpha with tolerable cluster numbers.

Luckily, this is all executed with a single function whose inputs are the barcode table and the PCA we just made. The `barcluster` function does all this for one or more alpha values. The `beta` parameter will affect how each step in alpha affects the output, we are using 0.1 here, but feel free to experiment. The `res` argument is passed to the `Seurat` implementation of the Louvain algorithm. It will affect the number of clusters at low alpha, but at high alpha the determining factor is the number of barcodes in the sample. I suggest you choose a value for resolution that gives you a number of manageable clusters at alpha == 0, or simply the same value as whatever initial `Seurat` analysis you have performed on the data.

```
# lets get our range of alpha values

als <- seq(0, 1, by = 0.1)

# return the cluster assignments for range of alphas
clust <- barcluster(pca, bt, alpha = als, beta = 0.1, res = 1.5)
```

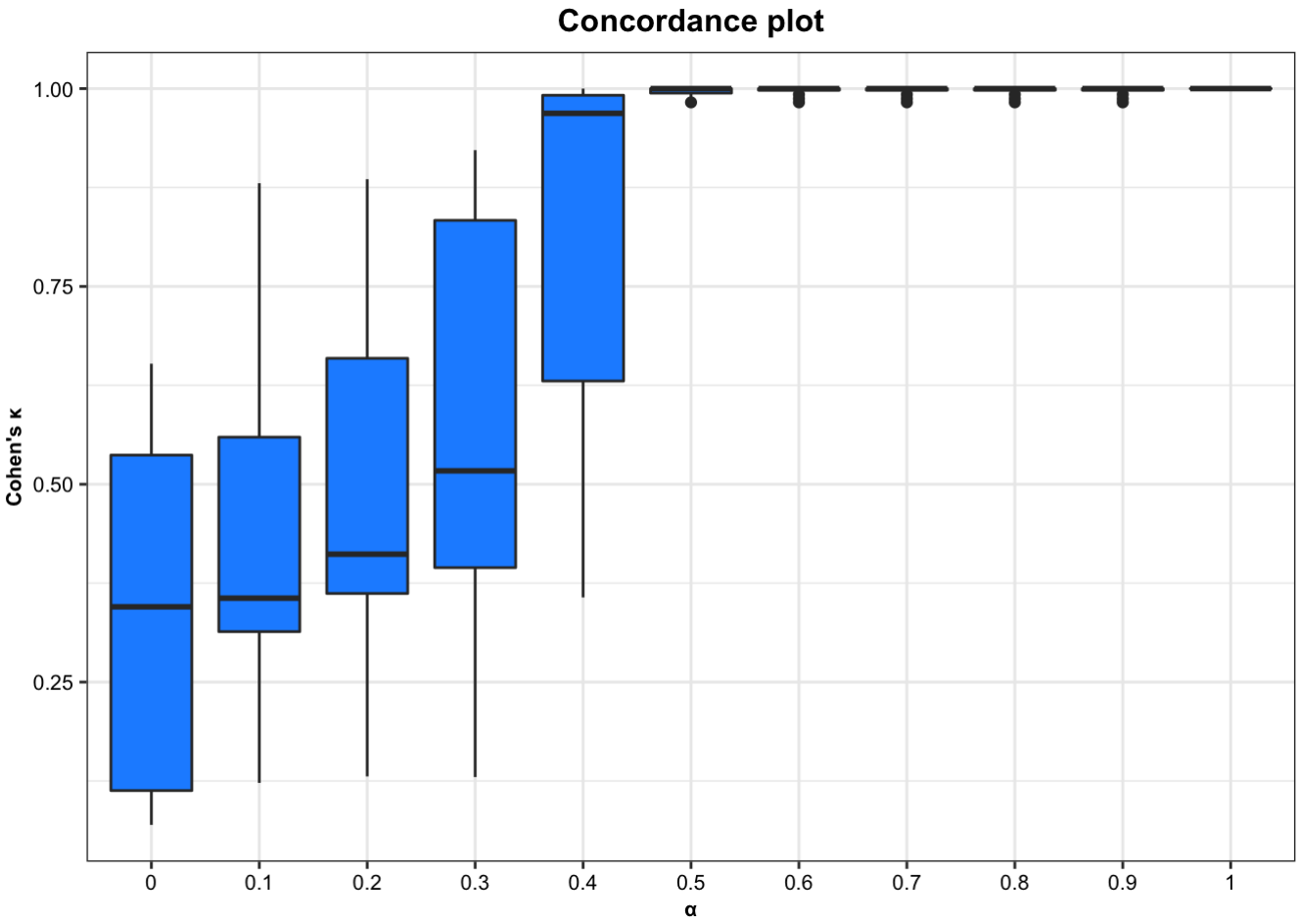
```
# lets take a peak at the output cluster assignments
head(clust)
```

```
##                  rn Group alpha resolution beta
## 1: AAACGAAAGAGGGTCT     8     0        1.5  0.1
## 2: AAAGGTAAGGTAGTAT     2     0        1.5  0.1
## 3: AAAGGTACACCATTCC     8     0        1.5  0.1
## 4: AAAGGTACAGTAGTGG     1     0        1.5  0.1
## 5: AAAGTGAAGGCCACCT     2     0        1.5  0.1
## 6: AACCATGGTTTGACAC     1     0        1.5  0.1
```

These are your hybrid clusters. Let's compute some confusion matrix statistics and plot them just so we can see the trends. Removing singlets makes the visualization much better so we will do that as well.

```
# loop over alphas
confusion <- lapply(clust[, alpha %>% unique], function(a){

  # function to compute confusion statistics
  ct <- cast_confusion(clust[alpha == a, .(rn, Group)], # subsetted, 2 column table is t
he input
                       bt) # barcode assignments

  # append alpha value
  ct[, alpha := a]

  return(ct)

}) %>% data.table::rbindlist() # merge to one big table

# get barcode ids of singlets
singlets <- bt[, .N, by = Barcode] %>% .[N == 1, Barcode]
```

```
# plot the cohen's kappa
ggplot(confusion[!barcode %chin% singlets], # ignore singlets
  aes(x = as.factor(alpha), y = cohens_k)) +
  geom_boxplot(fill = "dodgerblue") +
  theme_bw() +
  ttheme +
  theme(plot.title = element_text(size = 12)) +
  ggtitle("Concordance plot") +
  xlab("\u03B1") +
  ylab("Cohen's \u03BA")
```

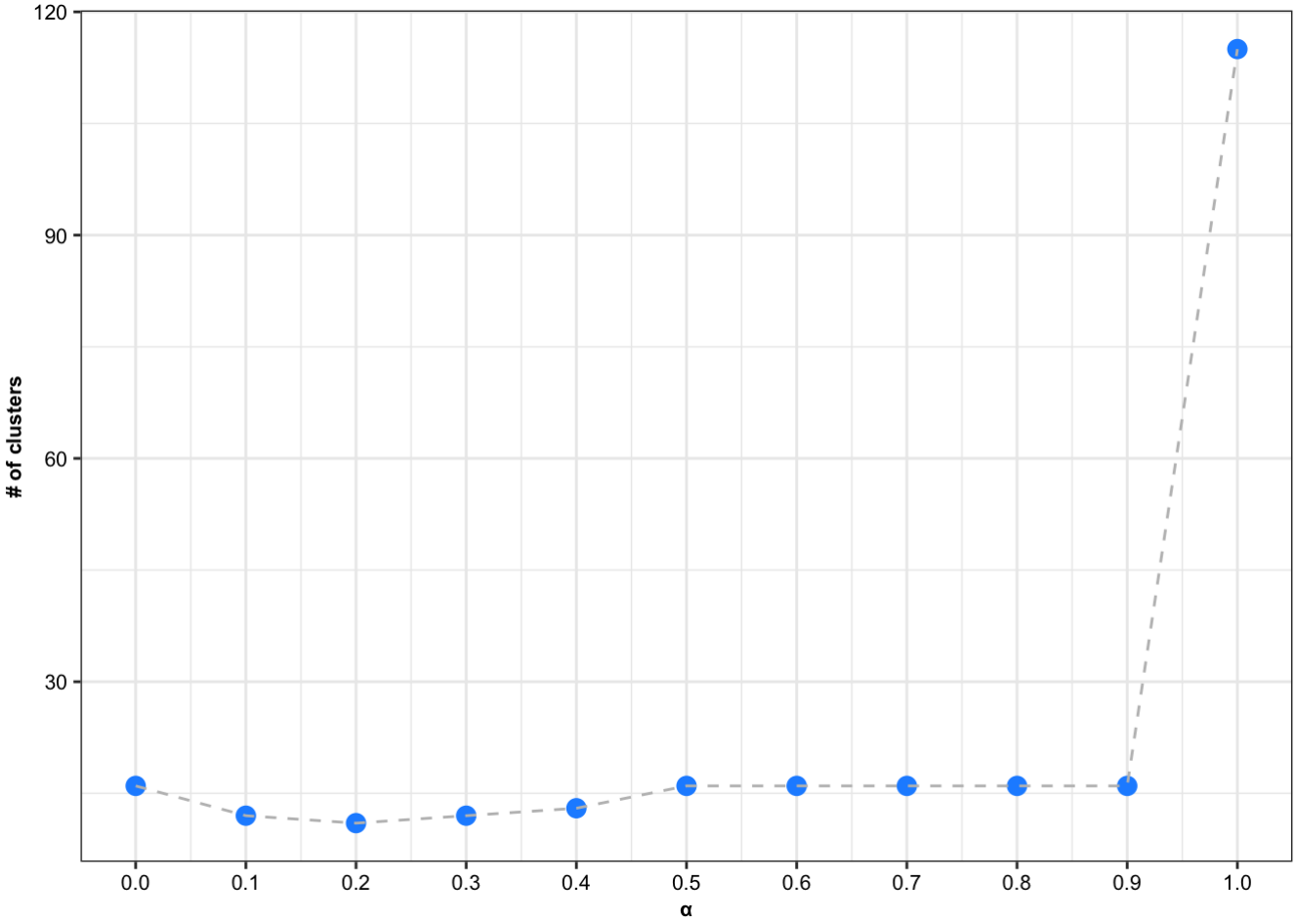


By alpha = 0.5, we have pretty much maxed out our concordance. Something to keep in mind when we choose target alpha values based on the number of clusters in the next step.

Part 4: Identify target alpha values

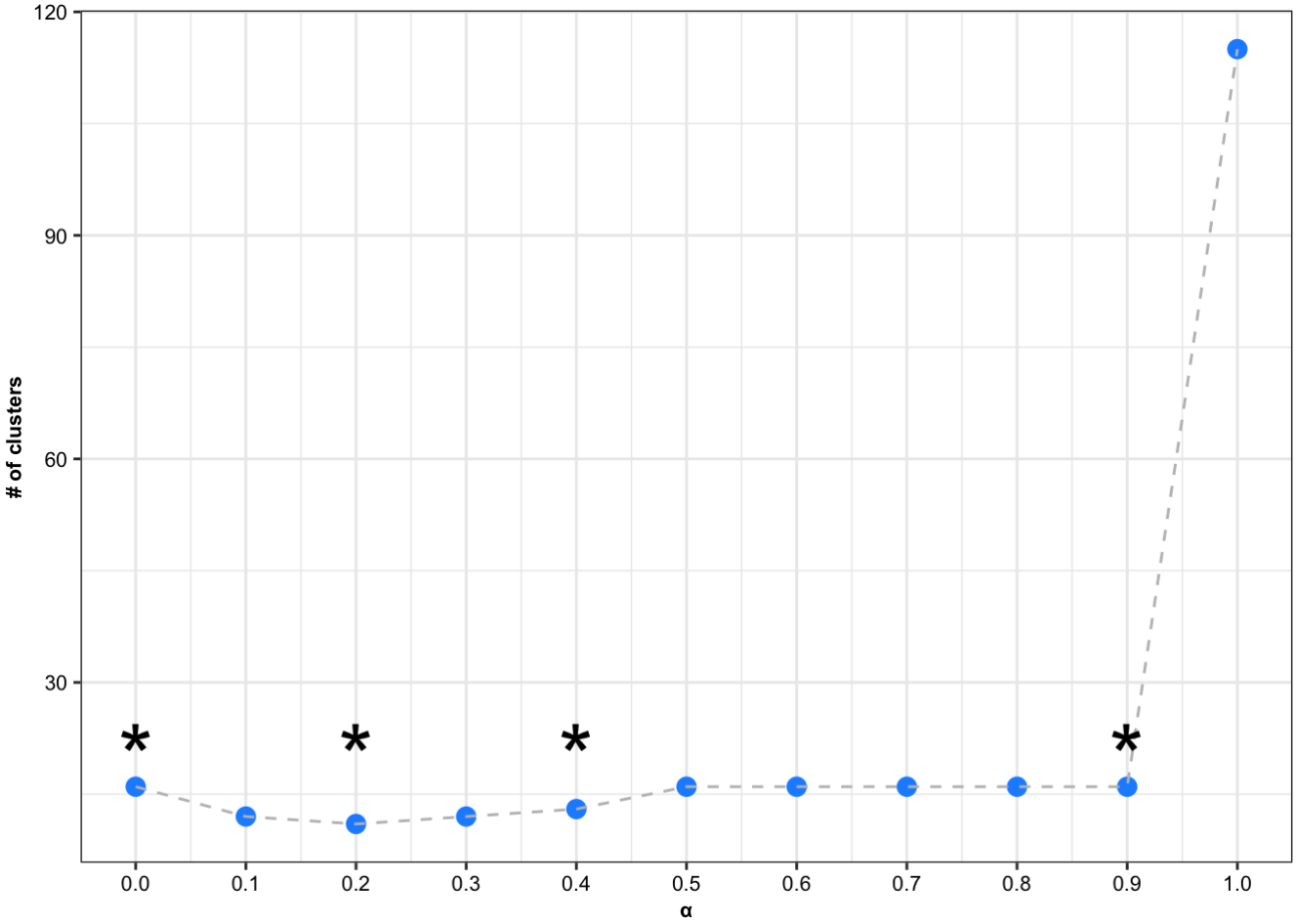
`clust` is a data table with the cluster assignment of each cell at all alpha values. First, lets identify points to conduct our analysis by making a plot of cluster number vs alpha.

```
# returns a table with column V1L number of groups at that value of alpha
nt <- clust[, Group %>% unique %>% length, by = "alpha"]

# plot it
ggplot(nt, aes(x = alpha, y = V1)) +
  geom_point(color = "dodgerblue", size = 3) +
  geom_line(color = "grey", linetype = "dashed") +
  theme_bw() +
  ttheme +
  xlab("\u03B1") +
  ylab("# of clusters") +
  scale_x_continuous(breaks = seq(0, 1, by = 0.1))
```



There isn't a ton of change in cluster numbers until alpha == 1, which is an artifact of having reduced the number of non-singlet barcodes in the sample to only 15. In real data with hundreds of barcodes of various sizes, expect a larger range of cluster numbers that trends upward with alpha.

It looks like there are four interesting points on the plot for this sample data (downsampled for top barcodes and 100 singlets from the WM989 high dose BRAF inhibitor replicate 1). I'll annotate them with asterisks here:

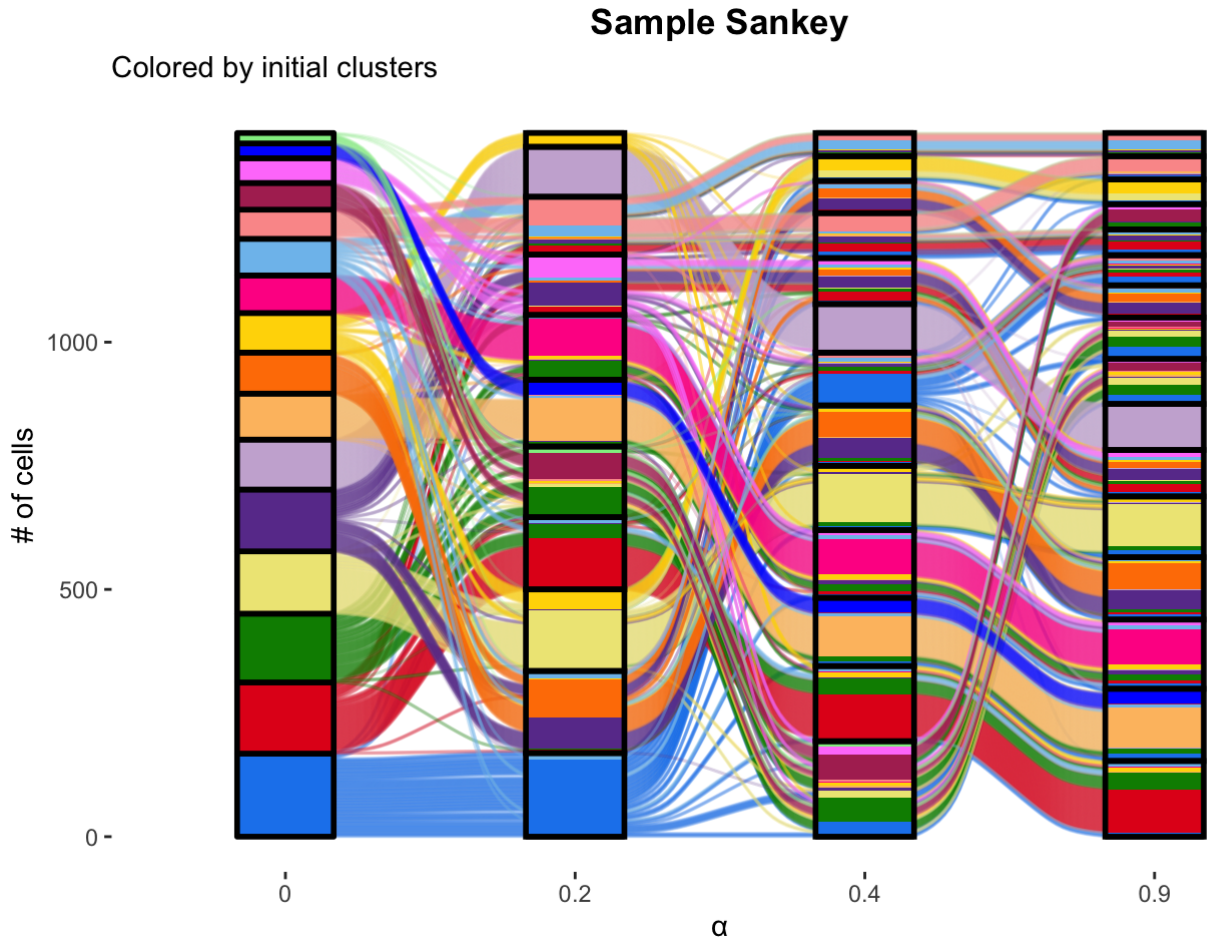


From left to right, these points are the transcriptome clusters (alpha == 0), a nadir where addition of lineage information causes clusters to join (alpha == 0.2), the inflection point (alpha == 0.4), and the maximum value before clusters break apart into individual barcodes, including singlets (alpha == 0.9).

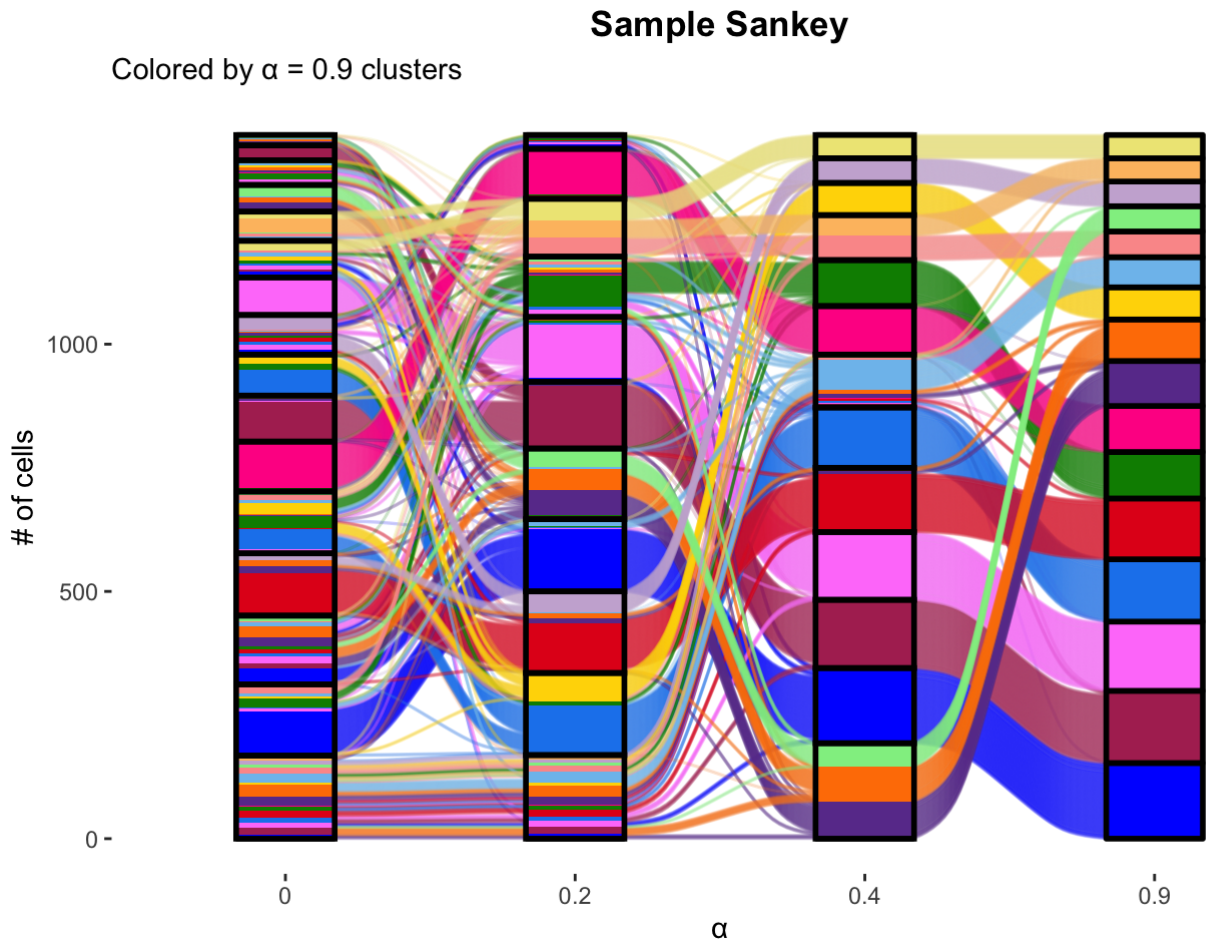
Part 5: Sankey visualization

Let's make a sankey plot to see the reorganization, subsetted on our alphas of interest. You could do this on the full range of alphas we ran by omitting the brackets and their contents after `clust` in the next call, but 100+ nodes on a sankey requiring individual discrete colors is not easy to grok. The function `Plot_alluvia` generates two sankeys with default settings, colored by the node assignments at each end of the plot.

```
notable_alphas <- c(0, 0.2, 0.4, 0.9)

# function to plot two sankeys colored by both ending nodes
p <- Plot_alluvia(clust[alpha %in% notable_alphas], # subset table on these alpha values
                  bt, # barcode table
                  title = "Sample Sankey",
                  xlab = "\u03B1", # unicode symbol for alpha
                  ylab = "# of cells",
                  border_size = 1, # border around the nodes
                  label_nodes = FALSE, # labels nodes but hard to viz here
                  cols = BarCluster::c25 # colors of nodes and ribbons
                  )

# change the subtitle of the second plot because we aren't coloring by barcodes
p[[2]] <- p[[2]] + ggtitle("Sample Sankey", subtitle = "Colored by \u03B1 = 0.9 cluster
s")

# plot colored by initial transcriptome
p[[1]]
```




```
# plot colored by highest alpha value in table
p[[2]]
```



In the first plot, you can see that the purple transcriptome cluster almost immediately splits up, and this appears to track with the reorganization of the yellow high alpha cluster in the second plot. This likely represents a barcode splitting off from the rest of that cluster. You can also see from the second plot how the bottom blue transcriptome cluster of the first plot has contributions from all different high alpha clusters, suggesting multiple lineages are contributing to this cluster. This actually represents the singlets combined with a few cells from each of the large barcode. They have a distinct SOX10 positive non-proliferative phenotype (hence why many of them are singlets) that each of the large barcodes appear to contribute some cells to as well.

Part 6: Identify cluster markers.

Let's find the top cluster marker per cluster at each level. We will do this for ROC analysis for in vs out of cluster cells across all genes in the count matrix. I've written a function to make this easier. This is not fast however. If you already have a Seurat object, this may be quicker by inputting clusters as the active identity and using the `Seurat::FindAllMarkers` function and I've created a wrapper for that too. Scroll to the next code chunk.

If you want to skip running this step and just go right to the discovered markers for the sample data, see the hashed annotations in this code.

```
# to skip this step:
# auc_table <- data.table::fread(file.path(dir, "YG1_markers.txt"))
auc_all <- Find_Markers_ROC(clust[alpha %in% notable_alphas], cm)

# you can save this table with:
# auc_table %>% data.table::fwrite("mytable.txt")
```

If you want to use the Seurat FindAllMarkers implementation (which may be faster) use the function `FindAllMarkers_Seurat`. I have not directly compared these two methods so perform your own validation if needed. This method does not return the thresholds so you will be unable to complete part 7 of this tutorial. If you identify target genes this way, you could go back to the `Find_Markers_ROC` function and subset the count matrix later to get your thresholds by using `cm[, my_vector_of_gene_names]`.

```
auc_all <- FindAllMarkers_Seurat(so, # seurat object
                                 clust, # barcluster function output
                                test.use = "roc") # if you let this default you will get
wilcox.tests
```

Now lets go back to the workflow after using the `Find_Markers_ROC` function.

```
# what does it look like?
head(auc_all)
```

```
##       rn      auc      thresh direction Group alpha
## 1: ACTA2 54.74387 -0.80533402   greater    10     0
## 2:  MYLK 72.84893  0.07535532   greater    10     0
## 3: S100B 60.21109 -1.23971079      less    10     0
## 4: CRYAB 73.24235 -1.60858012      less    10     0
## 5:  CCL5 64.60441 -0.96643621      less    10     0
## 6: ACTG2 63.16149  0.06621759   greater    10     0
```

```
# If you want to take the best auc for any cluster at each alpha
auc_table <- auc_all[order(-auc)] %>% unique(by = c("rn", "alpha"))

head(auc_table)
```

```
##       rn      auc    thresh direction Group alpha
## 1: IFIT3 98.78745 13.306560   greater    15   0.0
## 2:  CKS2 98.65591  2.622634   greater    10   0.2
## 3: UBE2S 98.60983  4.128507   greater    10   0.2
## 4: IFIT1 98.32552  5.817067   greater    15   0.0
## 5: CCNB1 98.20917  3.886104   greater    10   0.2
## 6:  MYLK 98.18751 14.881343   greater     3   0.0
```

Part 7: Tracking marker positivity by Sankey

It may be useful to look at how marker strength changes across hybrid clusters. To assess this, we will first take the best possible AUC for each marker at each alpha and make a wide table to look at strength.

```
# take our long format data and make it wide
aucw <- auc_table[, auc, by = c("alpha", "rn")] %>%
  dcast(rn ~ alpha, value.var = "auc")

# table of marker strength
head(aucw)
```

```
##            rn        0      0.2      0.4      0.9
## 1:        A2M 80.64516 67.80267 69.17082 69.17082
## 2:     ABI3BP 70.22387 64.79895 68.64055 68.51136
## 3: AC104654.2 73.36854 70.31086 71.18387 71.18387
## 4: AC131025.8 80.57216 78.81208 79.13879 78.06223
## 5:      ACTA2 97.15644 91.17165 94.24483 93.57567
## 6:      ACTG2 84.63426 86.68160 85.64068 86.40088
```

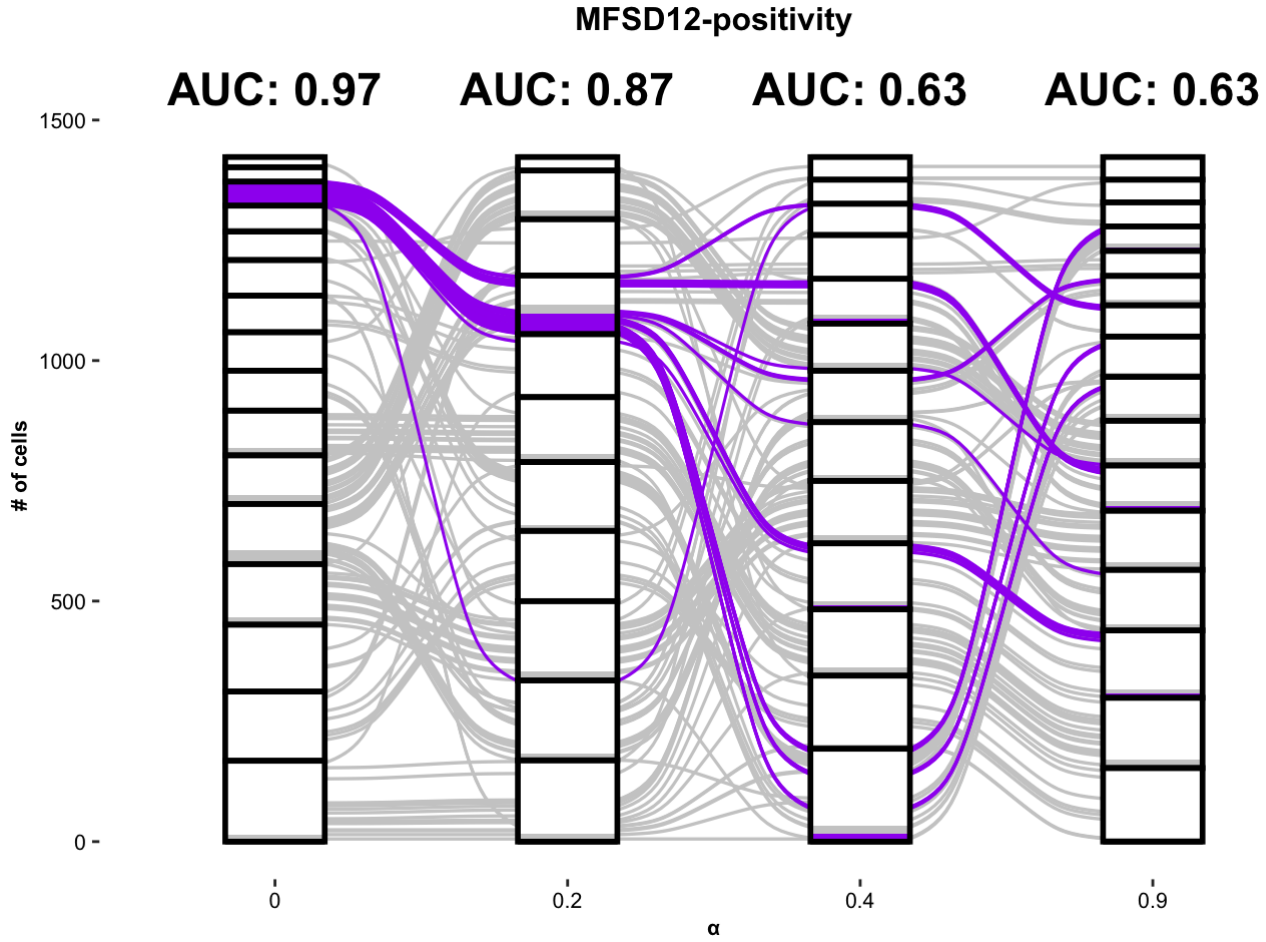
Let's identify some markers to plot it on Sankey.

```
# get the maximum diff between our highest alpha and our lowest
aucw[, delta := `0` - `0.9`]

# marker that falls off the most
# taking second most actually because first marks a small cluster that's hard to visuali
ze
transcriptome_only <- aucw[order(-delta), rn[2]]

# marker that improves the most
alpha_enhanced <- aucw[order(delta), rn[1]]

# which markers are they?
print(c(transcriptome_only, alpha_enhanced))
```

```
## [1] "MFSD12" "KCNMA1"
```

Let's plot the transcriptome only marker. We will make **in cluster, marked cells in purple** and **out of cluster marked cells in grey**.

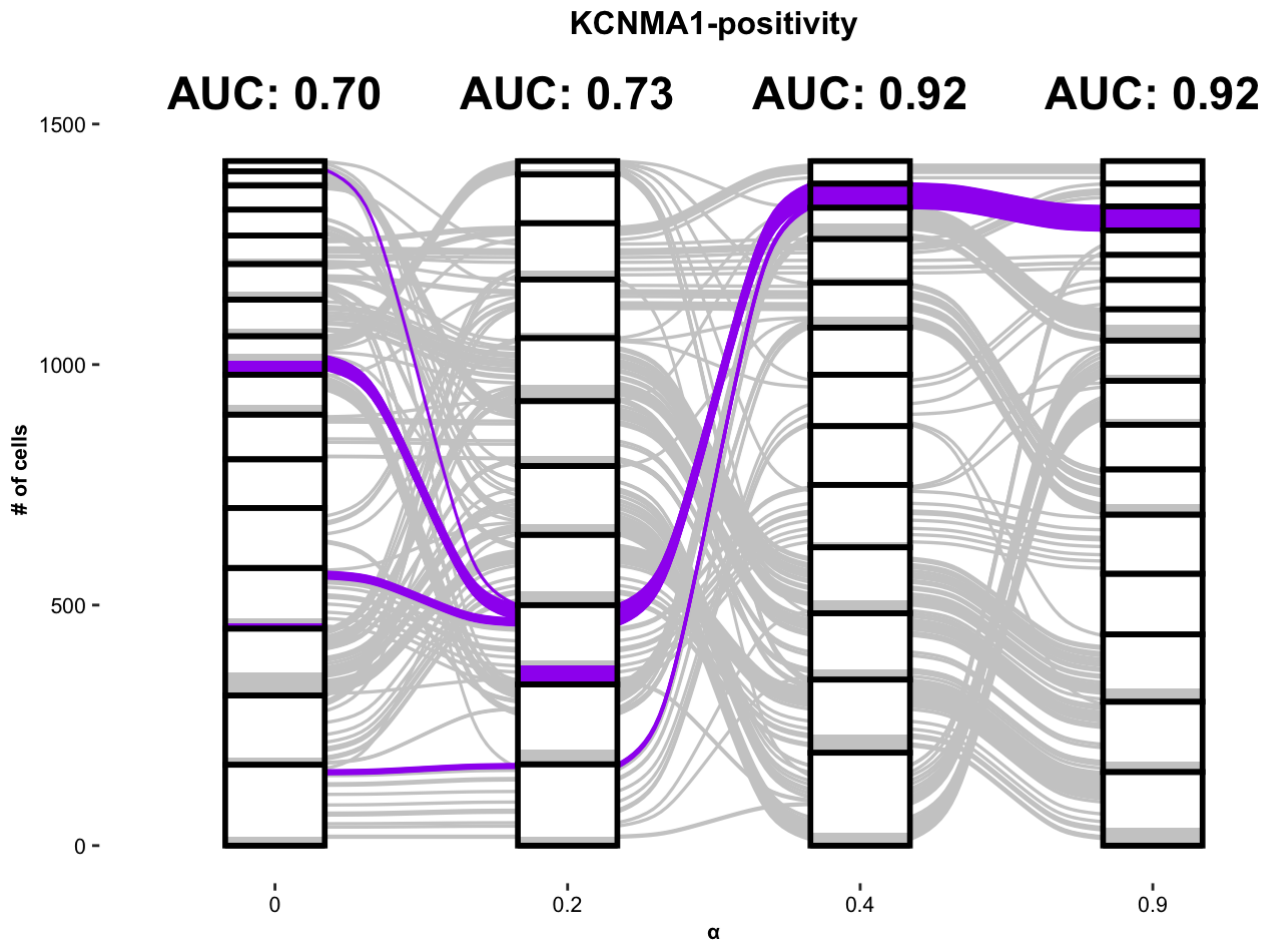
```
# get the transcriptome cluster that the is the most effective for our marker
at <- auc_table[alpha == 0 & rn == transcriptome_only]

# get our marked cluster members
cluster_members <- clust[alpha == 0 & Group == at[, Group], rn]

# retrieve marked cells
all_pos <- rownames(cm)[cm[, transcriptome_only] > at[, thresh]]

# check if the sign is flipped for the AUC
if (at[, direction == "less"])
  all_pos <- rownames(cm)[cm[, transcriptome_only] < at[, thresh]]

# get true positives (in cluster, positive)
tp <- intersect(cluster_members, all_pos)

# get false positives (out of cluster, positive)
fp <- all_pos[!all_pos %chin% cluster_members]

# generic plotting function, tracks any group of cells as a ribbon
p <- Plot_alluvia_track(clust[alpha %in% notable_alphas],
                          ids = list(fp, tp),
                          title = paste0(transcriptome_only, "-positivity"),
                          xlab = "\u03B1",
                          ylab = "# of cells",
                          label_nodes = FALSE,
                          border_size = 1,
                          flow_alpha = 1,
                          cols = c("grey80", "purple")
                        )

# put table in the same order as the plot
auc_table %<>% .[order(alpha)]

# AUC annotations
lab <- paste0("AUC: ", auc_table[rn == transcriptome_only]$auc %>%
  `/`(.,100) %>% sprintf(fmt = "%.2f"))

# add AUC annotations
p <- p +
  annotate("text", x = 1:4,
  y = nrow(bt) * 1.1, label = lab, fontface = "bold", size = 6)

# add text theming
p <- p + ttheme + theme(plot.title = element_text(size = 12))

p
```



As you can see, this marker loses fidelity with increasing alpha, suggesting its expression doesn't correlate with any particular lineage.

Let's plot an alpha-enhanced marker this time. We will **put in cluster, marked cells in purple** and **out of cluster marked cells in grey**.

```
# get the hybrid cluster that the is the most effective for our marker
at <- auc_table[alpha == last(notable_alphas) & rn == alpha_enhanced]

# get our marked cluster members
cluster_members <- clust[alpha == last(notable_alphas) & Group == at[, Group], rn]

# retrieve marked cells
all_pos <- rownames(cm)[cm[, alpha_enhanced] > at[, thresh]]

# check if the sign is flipped for the AUC
if (at[, direction == "less"])
  all_pos <- rownames(cm)[cm[, alpha_enhanced] < at[, thresh]]

tp <- intersect(cluster_members, all_pos)

fp <- all_pos[!all_pos %chin% cluster_members]

p <- Plot_alluvia_track(clust[alpha %in% notable_alphas],
                          ids = list(fp, tp),
                          title = paste0(alpha_enhanced, "-positivity"),
                          xlab = "\u03B1",
                          ylab = "# of cells",
                          label_nodes = FALSE,
                          border_size = 1,
                          flow_alpha = 1,
                          cols = c("grey80", "purple")
                        )

auc_table %<>% .[order(alpha)]

lab <- paste0("AUC: ", auc_table[rn == alpha_enhanced]$auc %>%
  `/`(.,100) %>% sprintf(fmt = "%.2f"))

p <- p +
  annotate("text", x = 1:4,
  y = nrow(bt) * 1.1, label = lab, fontface = "bold", size = 6)

p <- p + ttheme + theme(plot.title = element_text(size = 12))

p
```

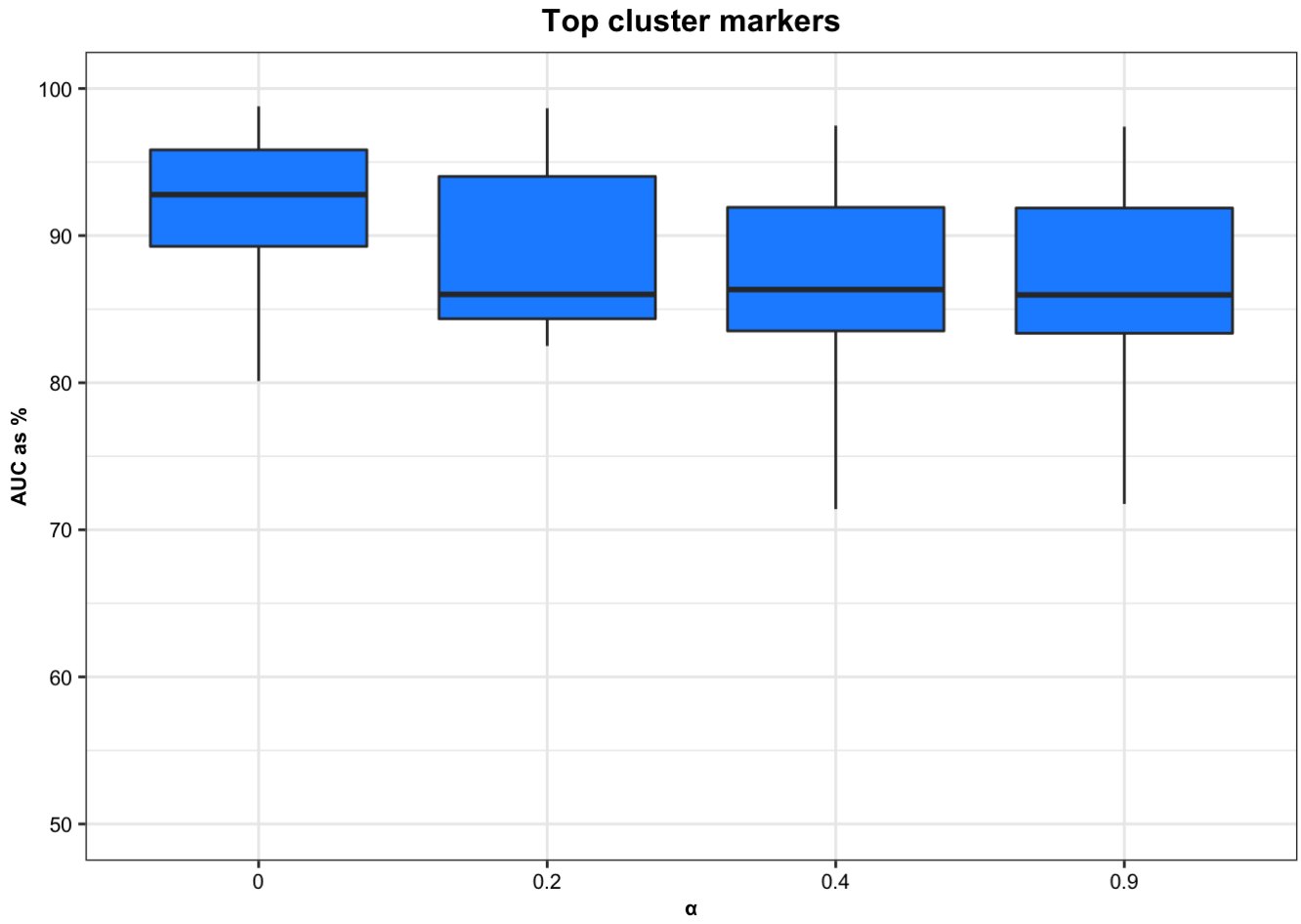


This alpha enhanced marker increases in fidelity with the addition of lineage information by increasing alpha, suggesting a unique expression level across one or more barcodes.

Part 8: Plot marker fidelity

Let's quickly plot the overall AUCs of top cluster markers across alpha values:

```
# split the table by cluster and get the top marker
topmarkers <- auc_all %>% split(by = c("alpha", "Group")) %>%

  lapply(., function(t){

    return(t[auc == max(auc)])

  }) %>% data.table::rbindlist()

# make a boxplot of the auc of top markers per cluster
ggplot(topmarkers, aes(x = as.factor(alpha), y = auc)) +
  geom_boxplot(fill = "dodgerblue") +
  theme_bw() +
  ttheme +
  theme(plot.title = element_text(size = 12)) +
  ggtitle("Top cluster markers") +
  xlab("\u03B1") +
  ylab("AUC as %") +
  ylim(50,100)
```

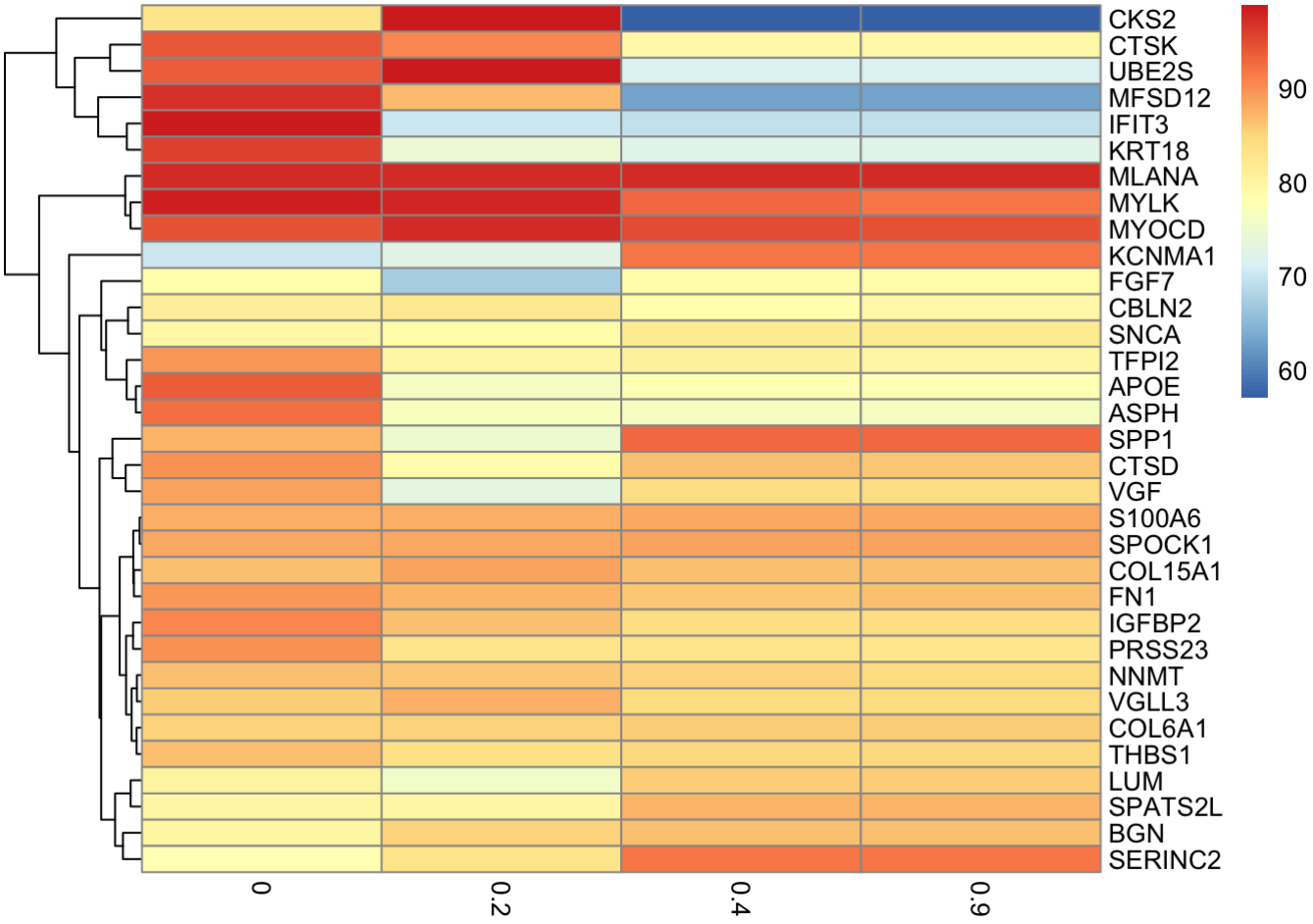


Overall marker fidelity is pretty much intact across alpha values. This is not always the case, as demonstrated in our analysis (link) of Jiang *et al. biorXiv* (2021) (https://www.biorxiv.org/content/10.1101/2021.08.08.455532v1). If there is a significant loss of fidelity, BarClusters may not be useful for identifying meaningful cluster markers and

this may suggest that your system has largely extrinsic determinants of cell fate. It may be worth continuing the analysis however, as you may identify a rare subset of cells that form their own cluster at high alpha that represent a state-fate correlation.

Now let's heatmap the maximum fidelity of all top cluster markers over alpha.

```
# get a table of just the genes in topmarkers and put columns in order
m2b <- aucw[rn %chin% topmarkers[, rn], .SD, .SDcols = c("rn", as.character(notable_alph
as))]

# convert data table to matrix
m <- dt2m(m2b)

# use pheatmap to make a heatmap, don't hierarchically cluster columns
pheatmap(m, cluster_cols = FALSE)
```

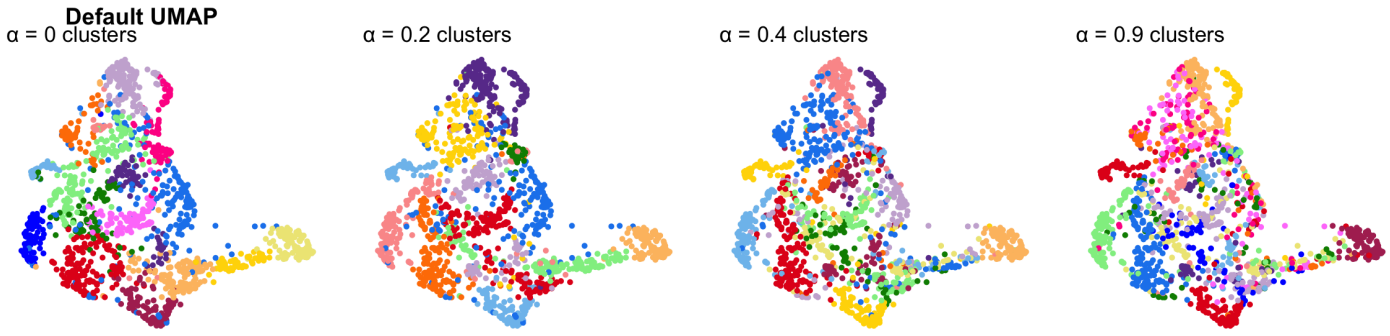


Clearly some of the markers like *CSK2* and *IFIT3* that are top cluster markers in transcriptome only clustering fall off when lineage information is added, and new markers, such as *KCNMA1* pop up. Thus, this alternative clustering scheme has incorporated the ground truth of lineage barcodes into our naive clustering approach and revealed alternative clusters and markers with retained fidelity.

Part 9: UMAP and Warp Factor the data

For the last part of our analysis, lets explore our barcodes and clusters in UMAP space. We will start by visualizing our clustering alphas in default UMAP space.

```
# move clusters to wide format so we can merge once
clw <- clust[alpha %in% notable_alphas] %>%
  dcast(rn ~ alpha, value.var = "Group")

# cluster column names
coln <- names(clw)[2:ncol(clw)] %>% paste("\u03B1 = ", ., sep = "")

# rename columns
names(clw)[2:ncol(clw)] <- coln

# default PCA based umap
um_tr <- umap_matrix(pca)

# merge clusters to umap coords
um_tr <- merge(um_tr, clw, by = "rn")

# loop over groupings to make umap plots
pl <- lapply(coln, function(cs){

  um <- um_tr %>% data.table::copy()

  um %>% data.table::setnames(cs, "group")

  p <- ggplot(um, aes(x = UMAP_1, y = UMAP_2)) +
    geom_point(aes(color = group), size = 0.3, alpha = 0.5) +
    scale_color_manual(values = c25) +
    ttheme +
    theme_void() +
    theme(legend.position = "none") +
    ggtitle(paste(cs, "clusters"))

  return(p)

})

# plot on a grid
cowplot::plot_grid(plotlist = pl, nrow = 1)
```

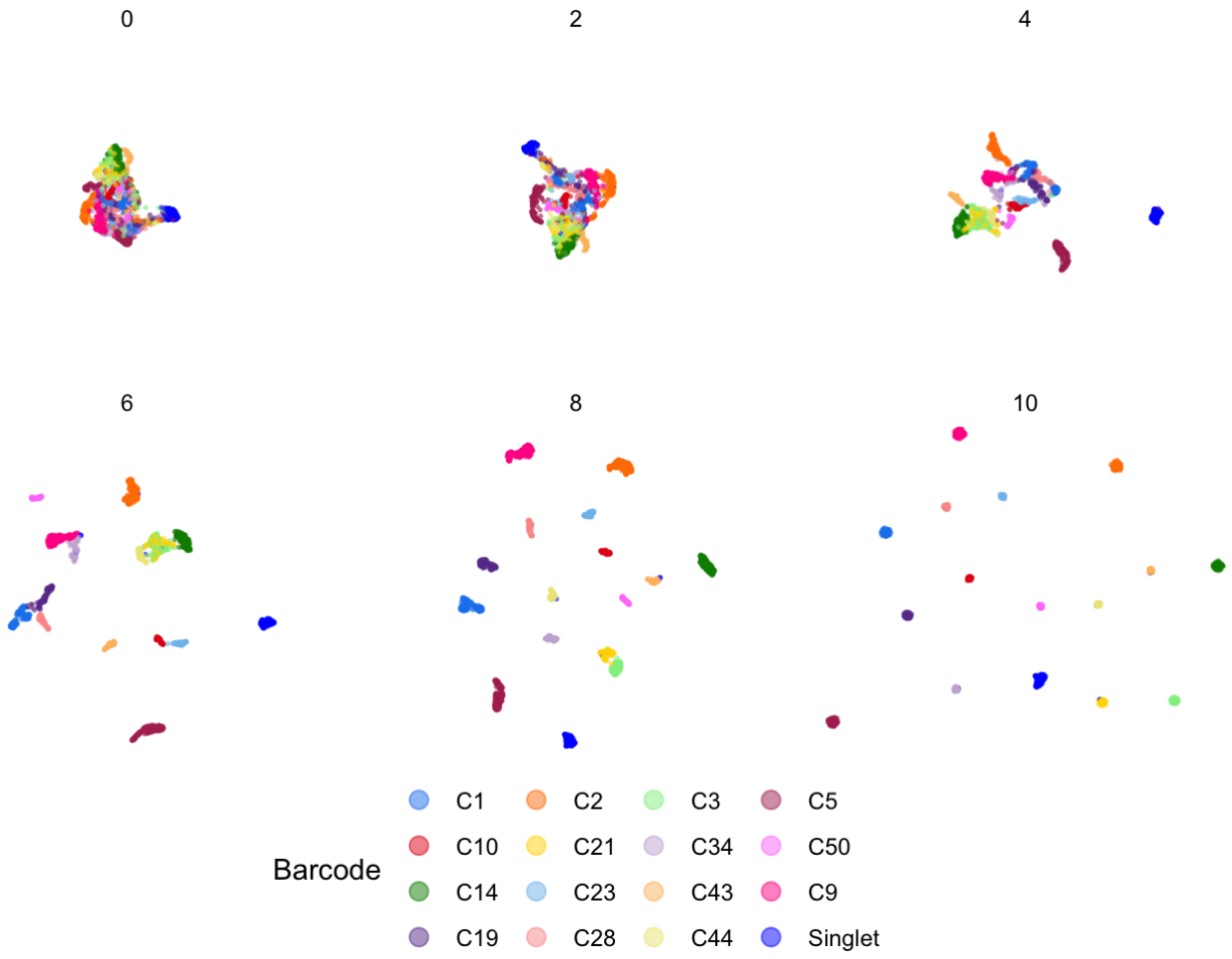


Now we know how our clusters look applied to the UMAP based only on the PCA. Let's warp the UMAP and look at barcode separation and decide where to set our warp factor and check our clusters again.

```
# lets do multiple warp factors
wfs <- c(0, 2, 4, 6, 8, 10)

# get our warped UMAPs
umaps <- lapply(wfs, function(s){

  uws <- engage_warp(pca, bt, s)

  return(uws)

}) %>% data.table::rbindlist()

# add our barcodes to the table
umaps <- merge(umaps, bt, by = "rn")

# color by barcode and put singlets into one category
umaps[, Barcode :=
        ifelse(rn %>% unique %>% length > 1, Barcode, "Singlet"),
        by = "Barcode"]

# make a plot faceted by warp factor
ums <- ggplot(umaps, aes(x = UMAP_1, y = UMAP_2)) +
    geom_point(aes(col = Barcode), size = 0.3, alpha = 0.5) +
    facet_wrap(~warp) +
    scale_color_manual(values = c25) +
    ttheme +
    theme_void() +
    theme(legend.position = "bottom")
```

Let's have a look at how our UMAP looks with increasing warp factor colored by barcode:

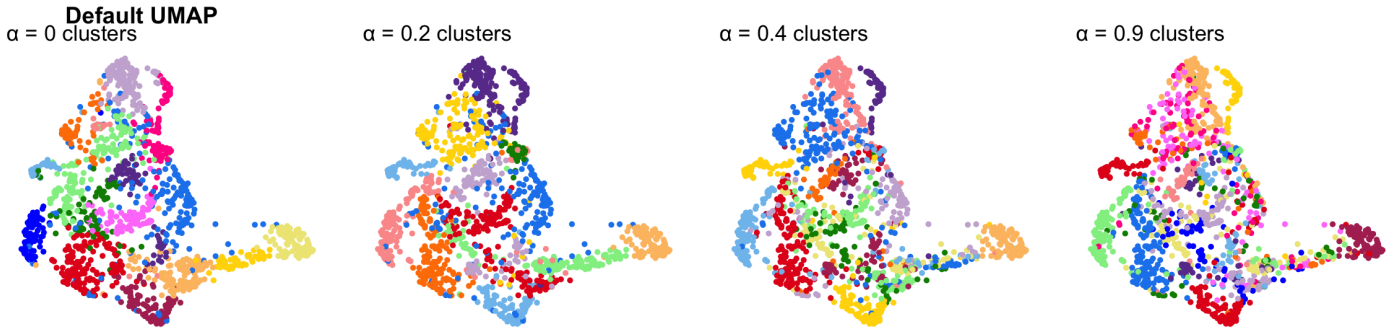
```
ums
```



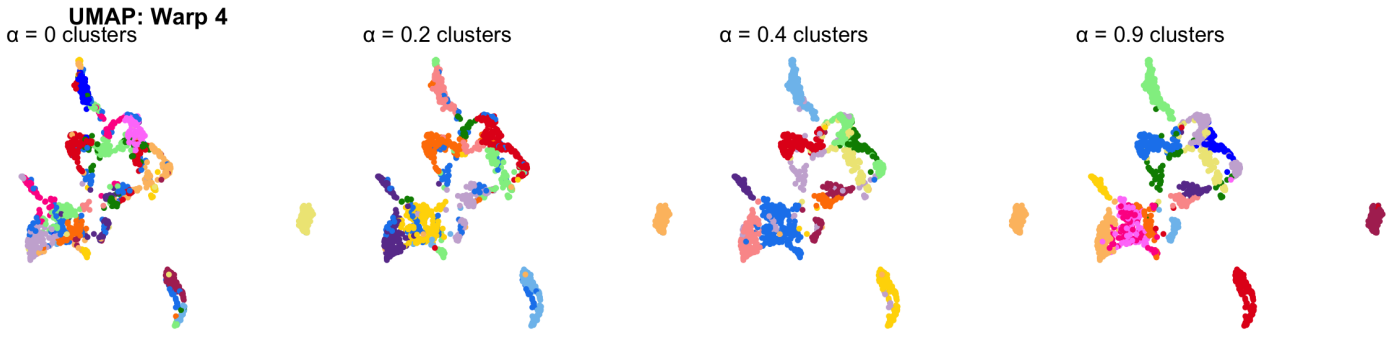
A warp factor of 4 looks interesting. It separates the barcodes nicely without too many barcodes splitting off completely. Lets look at our clusters at this warp factor.

```
# Umap at warp 4
um_wf4 <- engage_warp(pca, bt, s = 4)

um_wf4 <- merge(um_wf4, clw, by = "rn")

pl <- lapply(coln, function(cs){

  um <- um_wf4 %>% data.table::copy()

  um %>% data.table::setnames(cs, "group")

  p <- ggplot(um, aes(x = UMAP_1, y = UMAP_2)) +
    geom_point(aes(color = group), size = 1, alpha = 1) +
    scale_color_manual(values = c25) +
    ttheme +
    theme_void() +
    theme(legend.position = "none") +
    ggtitle(paste(cs, "clusters"))

  return(p)

})

cowplot::plot_grid(plotlist = pl, nrow = 1)
```

For direct comparison, here are the clusters on the original UMAP:



These are our clusters overlaid on the warped UMAP that we just generated:



Finally, let's look at how those two markers we identified appear in both UMAP settings.

```
# get UMI counts
umis <- cm[, c(alpha_enhanced, transcriptome_only)] %>%
  data.table::as.data.table(keep.rownames = TRUE) %>%
  melt(id.vars = "rn", value.name = "scaled UMI", variable.name = "Gene")

# add warp factor column to this table
um_tr[, warp := 0]

# combine table rows
um <- rbindlist(list(um_tr, um_wf4), use.names = TRUE, fill = TRUE)

# merge
um <- merge(um, umis, by = "rn")

um[, warp_factor := paste("Warp", warp)]

um %>% setkey(`scaled UMI`)

ggplot(um, aes(x = UMAP_1, y = UMAP_2)) +
  geom_point(aes(color = `scaled UMI`), size = 1, alpha = 1) +
  facet_wrap(warp_factor ~ Gene) +
  scale_color_gradient(low = "grey90", high = "darkblue") +
  ttheme +
  theme_void() +
  theme(legend.position = "bottom")
```

You can see how the transcriptome only marker expression is more spread out on the warped UMAP. The alpha enhanced marker is less spread out on the warped UMAP, and from comparison to the plots above you can see that the clusters also reconfigure to help make this a stronger marker.

Conclusion

Thank you for using the BarCluster tutorial! We generated our hybrid clusters, chose alpha values for analysis, visualized reorganization, identified markers, and modified the UMAP visualization to better reflect the incorporation of lineage barcode information. For more information, check out the package ReadMe (https://github.com/leeprichman/BarCluster) and our paper. (link)