Velocity Visualization Challenge

Jean Fan

7/30/2020

Introduction

In this tutorial, we will explore some of the challenges with visualizing RNA velocity results by walking through an RNA velocity analysis using MERFISH data of U2OS cells from Xia*, Fan*, Emanuel*, et al PNAS 2019.

Read in data

We will first read in expression count matrices for the whole cell and the nucleus respectively provided as Supplementary Tables 12 and 14. We will also create a cytoplasmic expression count matrix by subtracting the nucleus from the whole cell for downstream use. Here, each row is a gene (or blank) assayed by MERFISH, and each column is a cell.

```r
dir <- '../data/'
cell_gexp <- as.matrix(read.csv(paste0(dir, 'S12_cell_gexp.csv'), row.names=1))
print(cell_gexp[1:5,1:5])
```

```
##        B1_cell1 B1_cell2 B1_cell3 B1_cell4 B1_cell5
## A1CF          0        0        1        1        0
## A2M           4        2        1        1        1
## A2ML1         0        0        0        0        0
## A4GALT        6        3        4        2        3
## AACS         36       28       14       11       10
```

```r
nuc_gexp <- as.matrix(read.csv(paste0(dir, 'S14_nuc_gexp.csv'), row.names=1))
print(nuc_gexp[1:5,1:5])
```

```
##        B1_cell1 B1_cell2 B1_cell3 B1_cell4 B1_cell5
## A1CF          0        0        0        1        0
## A2M           1        1        0        0        0
## A2ML1         0        0        0        0        0
## A4GALT        0        1        1        1        0
## AACS          3        8        3        1        2
```

```r
cyto_gexp <- cell_gexp - nuc_gexp
print(cyto_gexp[1:5,1:5])
```

```
##        B1_cell1 B1_cell2 B1_cell3 B1_cell4 B1_cell5
## A1CF          0        0        1        0        0
## A2M           3        1        1        1        1
## A2ML1         0        0        0        0        0
## A4GALT        6        2        3        1        3
## AACS         33       20       11       10        8
```

In this MERFISH library, 9,050 genes that were labeled with the non-overlapping encoding probe strategy (rows 2 to 9,051) and the 1,000 genes were labeled with the overlapping encoding probe strategy (rows 9,280 to 10,279). The remaining are used a blank controls. We will limit our analysis to just the non-overlapping encoding probe strategy genes (ie. long genes) as the authors did in the original paper.

```
gene_info <- read.csv(paste0(dir, 'S1_codebook.csv'), header=FALSE, stringsAsFactors = FALSE)
long.genes <- gene_info[2:9051,1]
short.genes <- gene_info[9280:10279,1]
bad.genes <- gene_info[,1][grepl('Blank', gene_info[,1])]

length(long.genes)
```

```
## [1] 9050
```

```
length(short.genes)
```

```
## [1] 1000
```

```
length(bad.genes)
```

```
## [1] 2853
```

```
test.genes <- long.genes ## use subset of genes as in original paper
cell_gexp <- cell_gexp[test.genes,]
nuc_gexp <- nuc_gexp[test.genes,]
cyto_gexp <- cyto_gexp[test.genes,]
```

Cluster cells

Single-cell transcriptomic analysis enables the identification of novel cell types and cell states in a systematic and quantitative manner. To illustrate this, we will perform single-cell clustering analysis to identify cell populations based on the gene expression profiles of individual cells. Briefly, we filter out lowly expressed genes, performed batch correction, CPM and variance normalization, identify over-dispersed genes, and applied principal components (PC) analysis to identify 30 PCs that capture the greatest variance, and apply graph-based Louvain clustering in the PC space to identify cell clusters.

```
cd <- cell_gexp

## annotate batch
batch <- sapply(colnames(cd), function(x) strsplit(x, '_')[[1]][1])
batch <- factor(batch)
table(batch)
```

```
## batch
##  B1  B2  B3
## 645 400 323
```

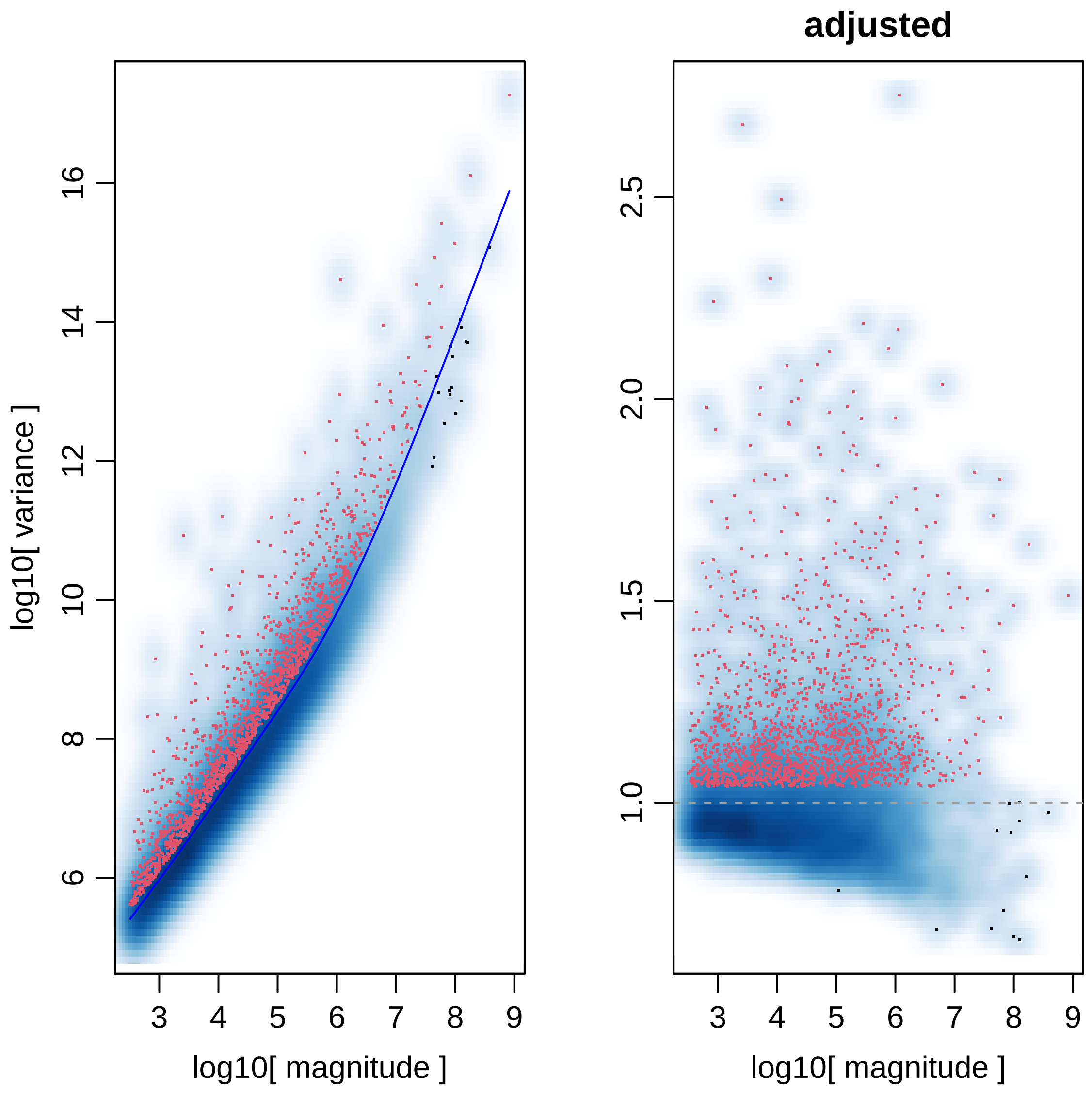
```
## remove lowly expressed
vi <- rowMeans(cd) > 1
table(vi)
```

```
## vi
## FALSE  TRUE
##  2941  6109
```

```
cd <- cd[vi,]

## batch correct (simply with ComBat here since no cell-type heterogeneity; otherwise recommend Harmony
```

```r
library(sva)
mat.bc <- ComBat(as.matrix(cd), batch[colnames(cd)])
mat.bc[mat.bc < 0] <- 0
mat <- mat.bc

## Use this helper package for some normalizations
library(MUDAN)
## CPM normalization
matcpm <- normalizeCounts(mat)
## Variance normalization
matnorm <- MUDAN::normalizeVariance(matcpm, details=TRUE, plot=TRUE)
```

```
## [1] "Calculating variance fit ..."
## [1] "Using gam with k=5..."
## [1] "1595 overdispersed genes ... "
```



```r
## Restrict to overdispersed genes
m <- log10(as.matrix(t(matnorm$mat[matnorm$ods,]))+1)
```

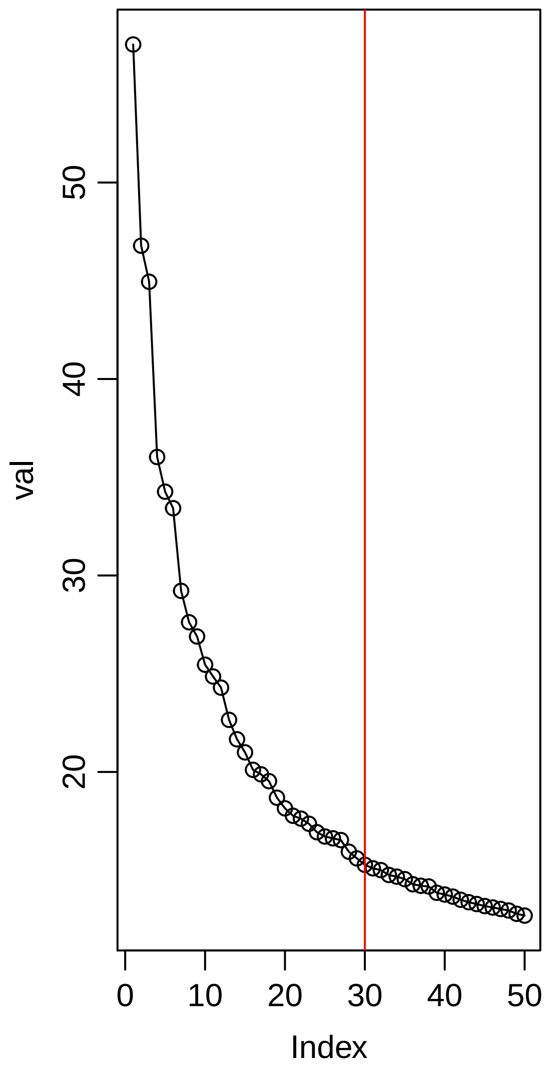
```
## PCA (slow)
#pcs <- prcomp(m)
## PCA (fast)
pca <- RSpectra::svds(
  A    = m,
  k    = 50,
  opts = list(
    center = TRUE, scale = FALSE, maxitr = 2000, tol = 1e-10
  )
)

## look at elbow plot to check what is reasonable number of pcs
val <- pca$d
plot(val, type="l")
points(val)
N <- 30
abline(v=N, col='red')
pcs <- pca$u[, 1:N]
rownames(pcs) <- colnames(mat)
colnames(pcs) <- paste0('PC', 1:N)
head(pcs)
```

```
##                    PC1           PC2          PC3         PC4         PC5
## B1_cell1 -0.0457922798  0.0005980665 -0.022031583  0.04309666  0.03698755
## B1_cell2 -0.0228822325 -0.0216632741 -0.026479726  0.04250357  0.00475602
## B1_cell3 -0.0033840021 -0.0500960762 -0.003137223  0.04035714 -0.04105824
## B1_cell4 -0.0159526579 -0.0267659369  0.023143928 -0.02173884  0.04240392
## B1_cell5 -0.0175046289 -0.0354836846  0.034938929  0.03968486  0.05204453
## B1_cell6  0.0009718162  0.0106313315 -0.029913382  0.01328155  0.04658692
##                   PC6          PC7          PC8          PC9         PC10
## B1_cell1 -0.023831456 -0.027227394 -0.016628774  0.005131082 -0.002817000
## B1_cell2 -0.020985244 -0.026215243  0.026327658  0.008885560  0.001918018
## B1_cell3 -0.051058309  0.025551571  0.009495523 -0.021725263 -0.012483053
## B1_cell4  0.016970813 -0.005469715  0.015078807  0.022419538  0.021724925
## B1_cell5  0.009303093 -0.020872628 -0.008264843  0.022127374  0.024581595
## B1_cell6  0.048547751  0.022777163 -0.018812971 -0.002653500  0.037510966
##                   PC11         PC12        PC13         PC14         PC15
## B1_cell1  0.0305229143  0.023887425  0.03941210  0.036273219 -0.004821876
## B1_cell2  0.0174677892  0.010465485  0.02508520  0.006699818 -0.000818187
## B1_cell3 -0.0341180540 -0.037733396 -0.03663732 -0.031625681 -0.066245610
## B1_cell4  0.0002484977  0.027381764 -0.01069326 -0.025703168  0.006342009
## B1_cell5  0.0065693581 -0.003201829 -0.01887241 -0.028641784  0.012575952
## B1_cell6  0.0071746022  0.048226435 -0.04347635  0.004737364  0.012413291
##                  PC16         PC17         PC18          PC19          PC20
## B1_cell1 -0.004417937  0.025179293 -0.013453189 -0.0239069145 -0.0110872798
## B1_cell2 -0.003309139  0.040017758  0.003998216 -0.0038170799  0.0126655396
## B1_cell3  0.004726756 -0.042235934 -0.034319624  0.0374654400 -0.0363347094
## B1_cell4  0.016347678  0.018407967 -0.014031337  0.0106476982  0.0004552663
## B1_cell5  0.004073057  0.008740713 -0.034385376  0.0007487695 -0.0143922123
## B1_cell6 -0.024506584  0.007981309  0.021433812  0.0181994120 -0.0137539699
##                 PC21         PC22         PC23          PC24          PC25
## B1_cell1 -0.01874579 -0.007752857 -0.003792575 -0.0079610513  0.0253157719
## B1_cell2  0.01071589  0.004496315  0.001310413 -0.0001942339  0.0110916498
## B1_cell3 -0.02176667 -0.008306747 -0.018575464  0.0593148035  0.0007605246
## B1_cell4  0.01439974 -0.020692003  0.012647563 -0.0111559975 -0.0249090887
## B1_cell5 -0.01133269  0.012664173 -0.033524048  0.0407902159 -0.0834254504
## B1_cell6  0.03401394 -0.003747932 -0.050585273 -0.0494956249 -0.0481466786
##                  PC26         PC27         PC28         PC29        PC30
## B1_cell1  0.021419084 -0.030900425 -0.004087136  0.021763040 0.027827904
## B1_cell2  0.019823090 -0.019092859 -0.004905913  0.010459861 0.031774212
## B1_cell3 -0.032152350 -0.007657983 -0.016113588  0.009915578 0.013366145
## B1_cell4  0.027385001 -0.061552099  0.015231559  0.020504221 0.044374133
## B1_cell5 -0.023565807 -0.027170066 -0.011425105  0.005478136 0.052167673
## B1_cell6  0.004374927  0.017072950 -0.045979275 -0.007906799 0.004673195
```

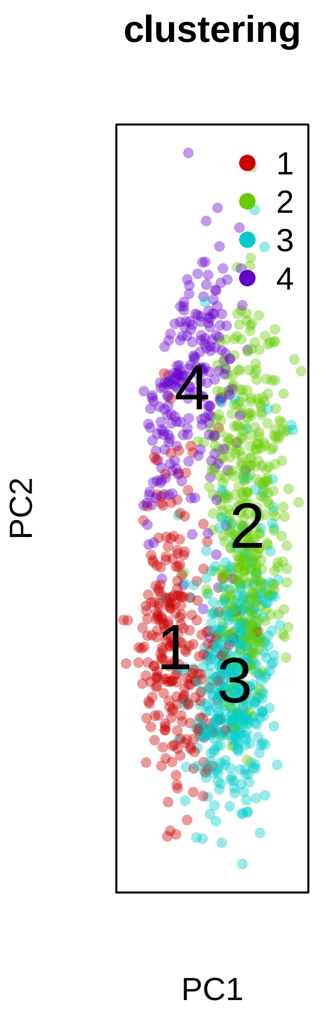
```
## use first two PCs as lower dimensional embedding
emb.test <- pcs[,1:2]

## Graph based community detection
com <- MUDAN::getComMembership(pcs,
                               k=300,
                               method = igraph::cluster_louvain)
```

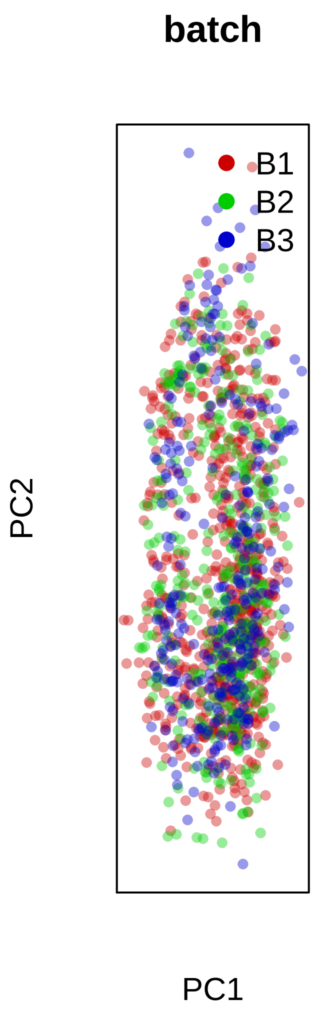
```
## [1] "finding approximate nearest neighbors ..."
## [1] "calculating clustering ..."
## [1] "WARNING"
## [1] "graph modularity: 0.121100754682579"
## [1] "identifying cluster membership ..."
## com
##   1   2   3   4
## 267 436 436 229
```

```
## Plot
par(mfrow=c(1,2), mar=rep(5,4))
```



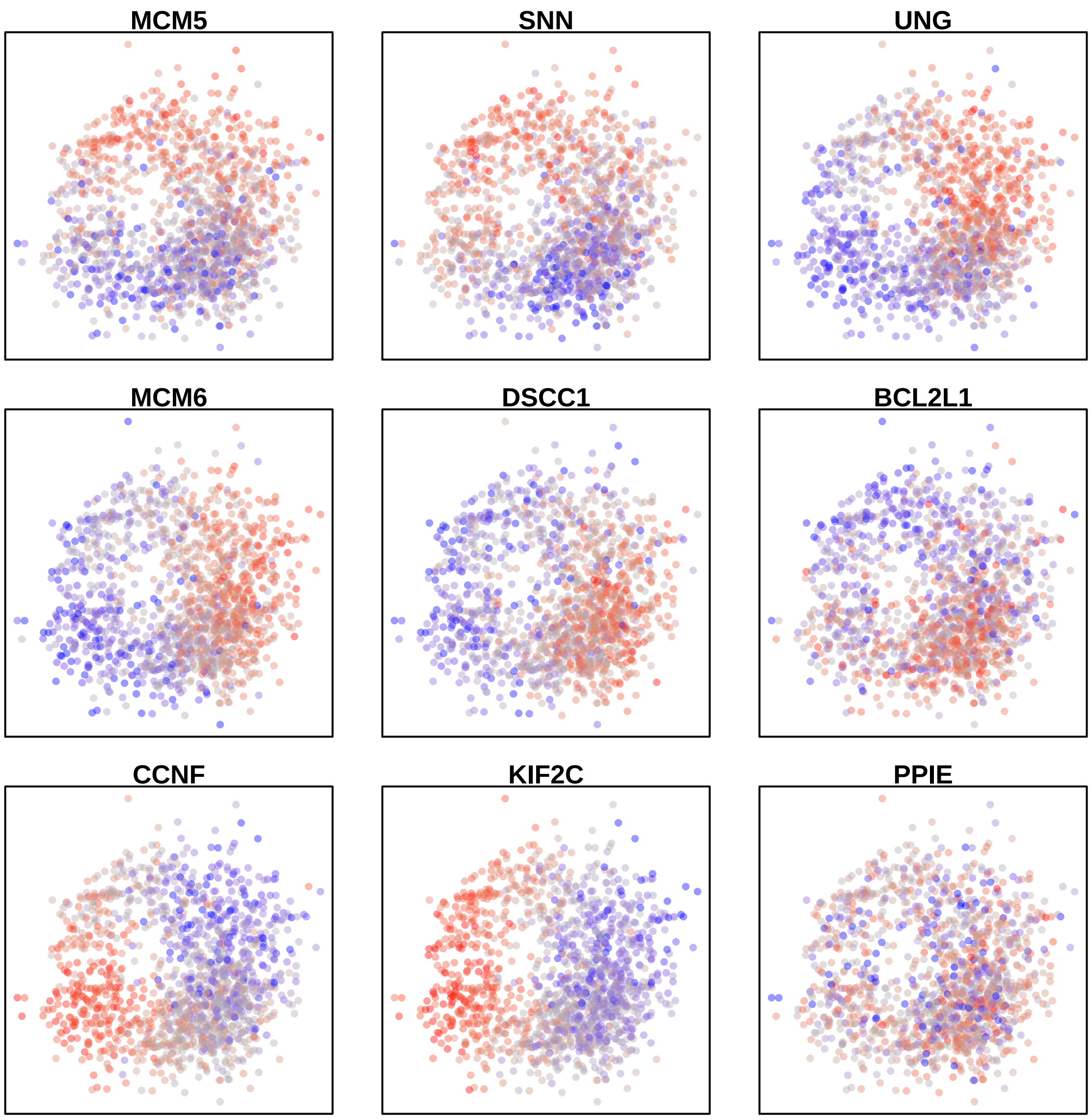
```
MUDAN::plotEmbedding(emb.test, groups=com,
            mark.clusters=TRUE, show.legend=TRUE,
            xlab='PC1', ylab='PC2', main='clustering',
            verbose=FALSE)
MUDAN::plotEmbedding(emb.test, groups=batch, show.legend=TRUE,
            xlab='PC1', ylab='PC2', main='batch',
            verbose=FALSE)
```





Given that our measurements were performed on a single cell type, we previously noted these clusters to likely represent distinct cell states at different stages of the cell cycle. We can visualize expression of known cell-cycle markers to confirm this hypothesis. For example, MCM5 is an early G1 gene whereas KIF2C is an M-phase gene. PPIE is a house-keeping gene not related to cell-cycle.

```r
## Plot cell-cycle and house keeping genes noted in the manuscript
par(mfrow=c(3,3), mar=rep(1,4))
gs <- c('MCM5','SNN','UNG','MCM6','DSCC1','BCL2L1','CCNF','KIF2C','PPIE')
invisible(lapply(gs, function(g) {
    gexp = scale(log10(matcpm[g,]+1))[,1]
    gexp[gexp > 2] <- 2
    gexp[gexp < -2] <- -2
    MUDAN::plotEmbedding(emb.test, main=g, col=gexp, verbose=FALSE)
}))
```



Derive velocity model

Further understanding of these cell states will benefit from quantification of temporal changes of gene expression profiles across the cell cycle. However, like scRNA-seq analysis, MERFISH measurements capture only static snapshots in time. To address this limitation, we sought to place cells on a pseudotime axis by analysis of the RNA velocity, i.e. the time derivative of the gene expression state. As detailed in the original manuscript, we reasoned that RNA velocity could be inferred by distinguishing between nuclear and cytoplasmic mRNAs, leveraging the spatial information of transcripts obtained in our MERFISH measurements. We use these nuclear and cytoplasmic expression measurements as in situ analogues of unspliced and spliced mRNA expression measurements used in the original RNA velocity manuscript (La Manno et al, Nature 2018). Due to concerns of batch effects, we will limit analysis to one batch.

```r
library(velocyto.R)

## Color by cluster
cluster.label <- factor(com)
cell.colors <- MUDAN:::fac2col(cluster.label)
```

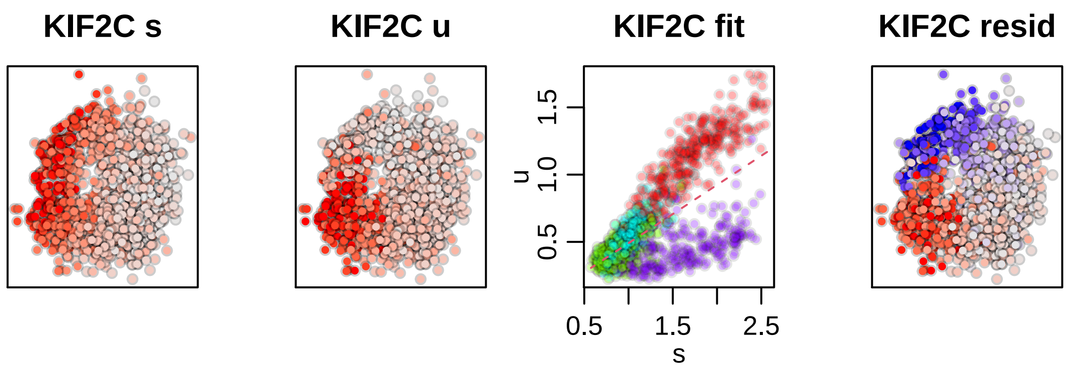
```r
## Can limit to one batch of cells
#subcells <- names(batch)[batch == 'B2']
## Or use all cells
subcells <- names(batch)
emat <- cyto_gexp[, subcells]
nmat <- nuc_gexp[, subcells]
cell.dist <- as.dist(1-cor(t(pcs[subcells,]))) ## cell distance in PC space
fit.quantile <- 0.05 ## 5th extreme quantile
## Velocity model
rvel.cd <- gene.relative.velocity.estimates(emat, nmat,
                                            deltaT=1, kCells=30,
                                            cell.dist=cell.dist,
                                            fit.quantile=fit.quantile)
```

```
## calculating cell knn ... done
## calculating convolved matrices ... done
## fitting gamma coefficients ... done. succesfful fit for 9050 genes
## filtered out 2045 out of 9050 genes due to low nmat-emat correlation
## filtered out 319 out of 7005 genes due to low nmat-emat slope
## calculating RNA velocity shift ... done
## calculating extrapolated cell state ... done
```

Upon active upregulation of a gene, we anticipate a rapid increase in nuclear mRNA counts, followed by an increase in cytoplasmic mRNA counts due to nuclear export until a new steady state is reached. Conversely, active downregulation in transcription would lead to a rapid reduction in nuclear mRNA counts as the nuclear export of mRNAs continues; the cytoplasmic mRNA will drop eventually because of the reduction in the nuclear RNA pool for export and the continued RNA degradation in the cytoplasm. We can plot a few example cell-cycle genes to see if this assumption holds in our data for the appropriate clusters.

```r
## Plot a few genes
gene.relative.velocity.estimates(emat, nmat,
                                 kCells = 30,
                                 fit.quantile = fit.quantile,
                                 old.fit=rvel.cd,
                                 show.gene='KIF2C',
                                 cell.emb=emb.test,
                                 cell.colors=cell.colors)
```

```
## calculating convolved matrices ... done
```

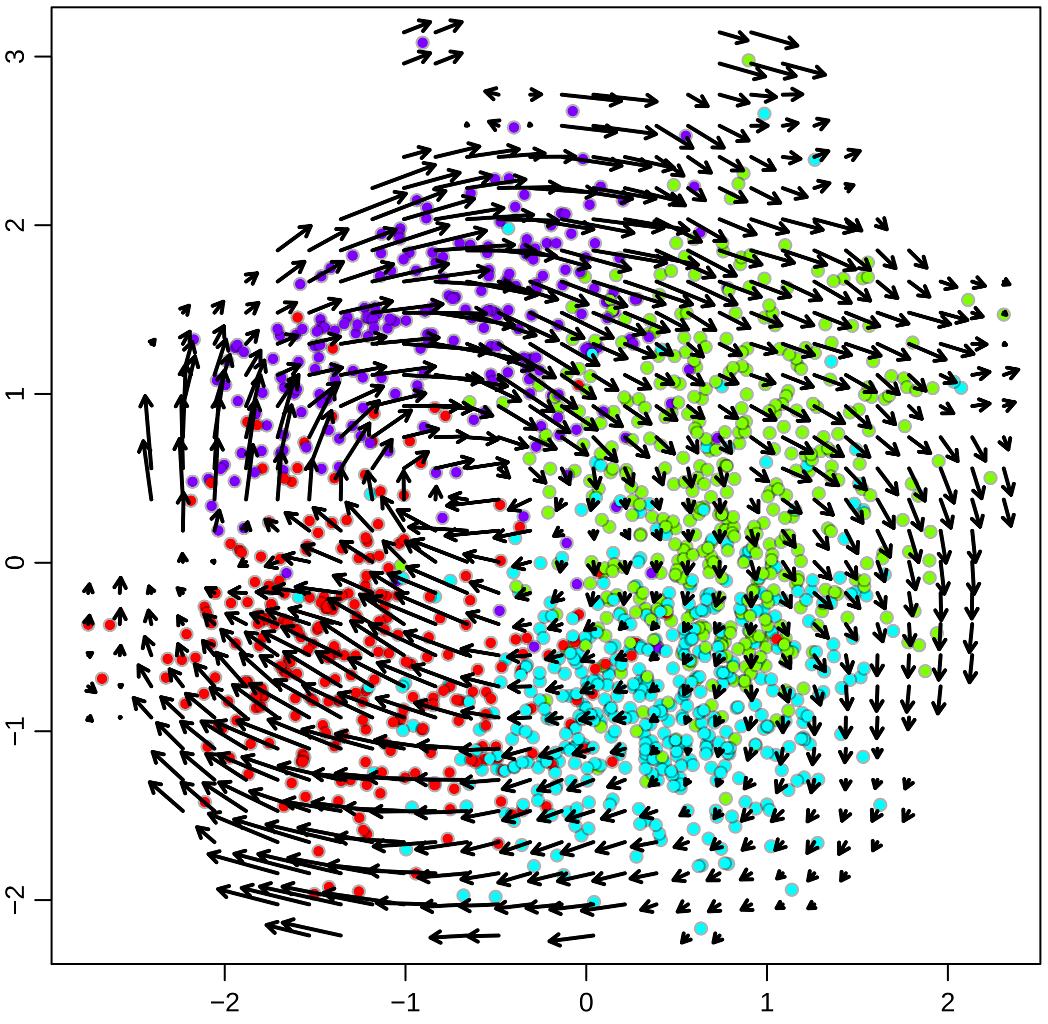


```
## [1] 1
```

Visualize velocity

The balance of nuclear and cytoplasmic mRNA abundance is, therefore, an indicator of the future state of cytoplasmic RNA abundance. We used this approach to determine the RNA velocity for each cell and projected these velocities as arrows. In this case, the RNA velocity arrows nicely recapitulate our expected cell cycle process.

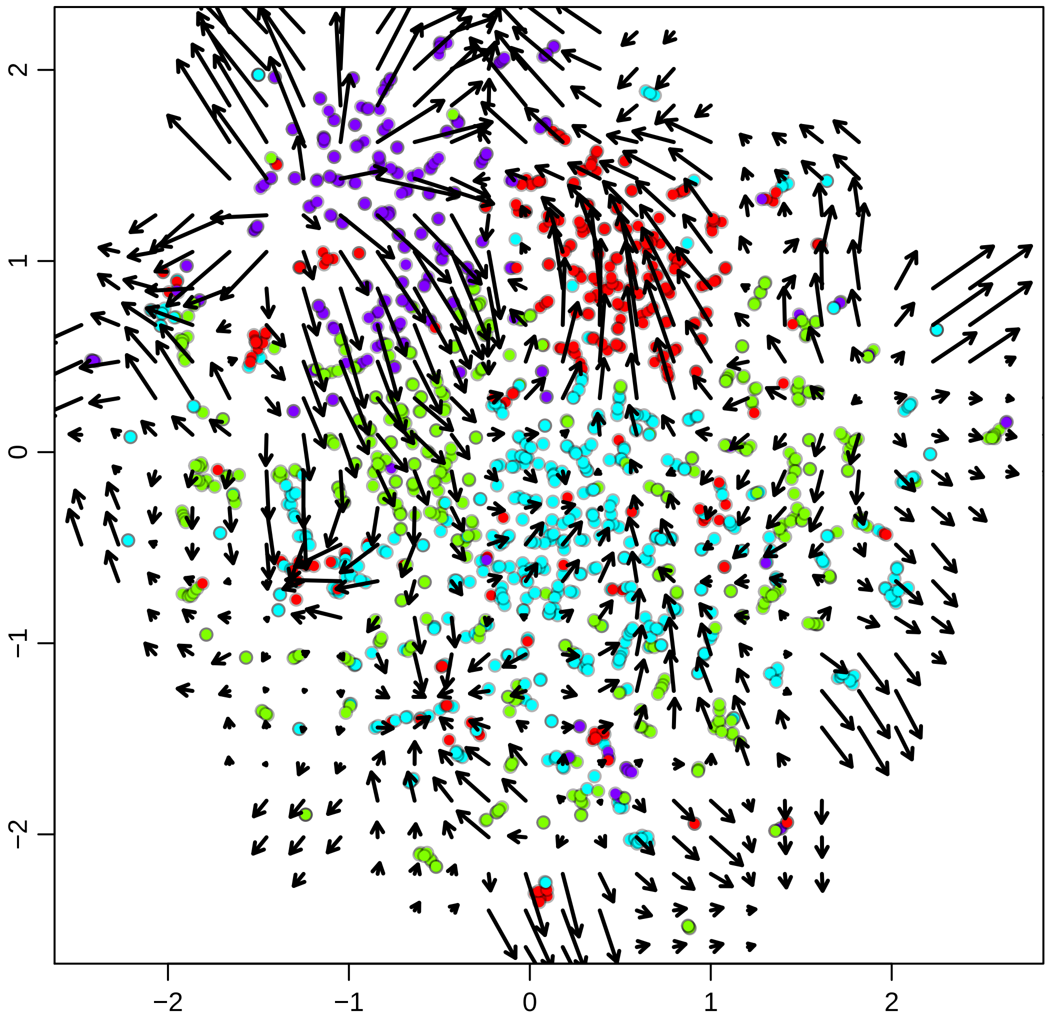
```
## Plot on PCs
show.velocity.on.embedding.cor(scale(emb.test), rvel.cd, n=100,
                               scale='sqrt', cell.colors=cell.colors,
                               cex=1, arrow.scale=1, show.grid.flow=TRUE,
                               min.grid.cell.mass=0.5, grid.n=30, arrow.lwd=2)
```



```
## delta projections ... sqrt knn ... transition probs ... done
## calculating arrows ... done
## grid estimates ... grid.sd= 0.1270606  min.arrow.size= 0.002541212  max.grid.arrow.length= 0.0597  d
```

However, RNA velocities can be visualized on a variety of low dimensional embeddings. Unfortunately, depending on how well these lower dimensional embeddings capture the underlying cellular transcriptional dynamics, we may end up with very different visual impressions. Let's try visuallizing RNA velocities on a few different embeddings to see what we mean.

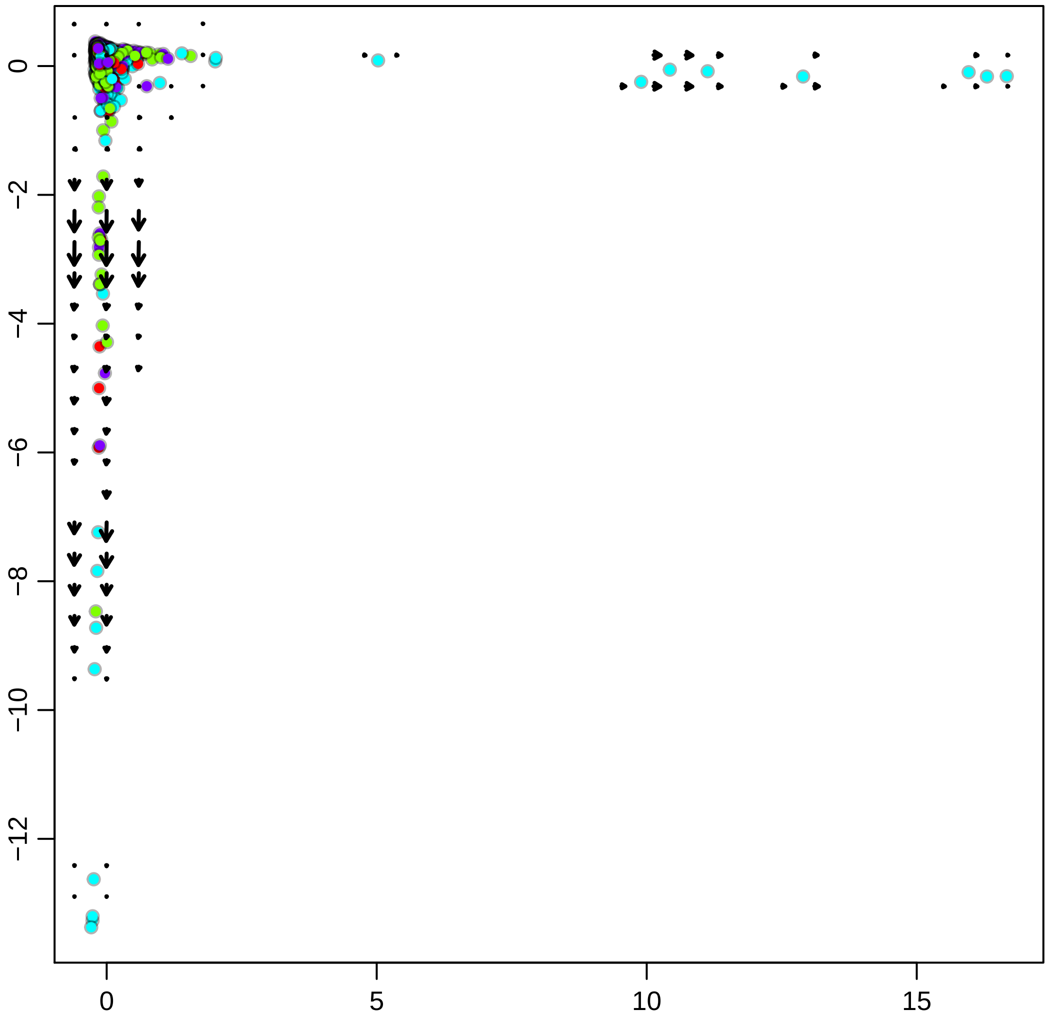
```
## tSNE
set.seed(1)
library(Rtsne)
emb.tsne <- Rtsne::Rtsne(pcs,
                    is_distance=FALSE,
                    perplexity=10,
                    num_threads=1,
                    verbose=FALSE)$Y
rownames(emb.tsne) <- rownames(pcs)

show.velocity.on.embedding.cor(scale(emb.tsne), rvel.cd, n=100,
                               scale='sqrt', cell.colors=cell.colors,
                               cex=1, arrow.scale=1, show.grid.flow=TRUE,
                               min.grid.cell.mass=0.5, grid.n=30, arrow.lwd=2)
```



```
## delta projections ... sqrt knn ... transition probs ... done
## calculating arrows ... done
## grid estimates ... grid.sd= 0.1401163  min.arrow.size= 0.002802326  max.grid.arrow.length= 0.0597  d
```

```
## diffusion mapping on 30 PCs
library(destiny)
emb.destiny <- destiny::DiffusionMap(pcs)
```

```r
emb.destiny <- eigenvectors(emb.destiny)[,1:2]

show.velocity.on.embedding.cor(scale(emb.destiny), rvel.cd, n=100,
                               scale='sqrt', cell.colors=cell.colors,
                               cex=1, arrow.scale=1, show.grid.flow=TRUE,
                               min.grid.cell.mass=0.5, grid.n=30, arrow.lwd=2)
```



```
## delta projections ... sqrt knn ... transition probs ... done
## calculating arrows ... done
## grid estimates ... grid.sd= 0.3836562  min.arrow.size= 0.007673124  max.grid.arrow.length= 0.0597  d
```

Try it out for yourself

1. Use `UMAP` to derive a lower dimensional embedding that is more faithful to long-range relationships between cells. Does this help? What is you change `UMAP` parameters?
2. Create a new approach to derive the lower dimensional embedding directly from RNA velocity estimates using force directed graphs. For each cell, use the RNA velocity model to predict the future transcriptional state. For each cell, identify the k-nearest neighbor of this predicted future transcriptional state and the currently observed cells (excluding the cell itself). Create a graph to represent this relationship and visualize the graph using forced-directed embedding.