VeloViz Parameters

In this example, we will explore the different user-inputted parameters to VeloViz and their effects on the 2D embedding. To do this, we will create a cell cycle simulation with missing intermediates and create VeloViz graphs with various parameter values.

Create simulated cell cycle

```r
par(mfrow = c(1,2))
## make 2D circle like cell cycle + noise in 3D
set.seed(1)
x <- matrix(rnorm(1000),nc=2)
y <- x/sqrt(rowSums(x^2))
z = rnorm(500)

## make trajectory and order points counterclockwise
traj = y
angle <- atan2(traj[,2], traj[,1])
traj = traj[order(angle),]

## make observed and add noise
obs = traj
obs <- jitter(obs, amount = 0.25)
obs = cbind(obs,z)

## color by position in cycle and add cell labels
col = colorRampPalette(c(rainbow(10)))(nrow(obs))

labels <- paste0('cell', 1:nrow(obs))
rownames(obs) = labels
rownames(traj) = labels
names(col) = labels

plot(obs[,1:2], col=col, pch=16, main = "simulated cycle", xlab = "X", ylab = "Y")
points(obs[c(1,100,200,300,400,500),], pch = 4, col = "black")
text(obs[c(1,100,200,300,400,500),1:2]-0.1, labels[c(1,100,200,300,400,500)])


## make projected by rotating circle
angle.rot = pi*0.1 # angle to rotate by
exp = obs[,1:2]
exp[,1] = obs[,1]*cos(angle.rot) - obs[,2]*sin(angle.rot)
exp[,2] = obs[,2]*cos(angle.rot) + obs[,1]*sin(angle.rot)
zexp = rnorm(500) #3D
exp = cbind(exp,zexp)
rownames(exp) <- labels

## plot observed and expected
plot(obs[,1:2],col=col, pch=16,
     main = "simulated observed \nand projected", xlab = "X", ylab = "Y")
points(exp[,1:2],col=col)
arrows(obs[,1:2][,1],obs[,1:2][,2],exp[,1:2][,1],exp[,1:2][,2])
```
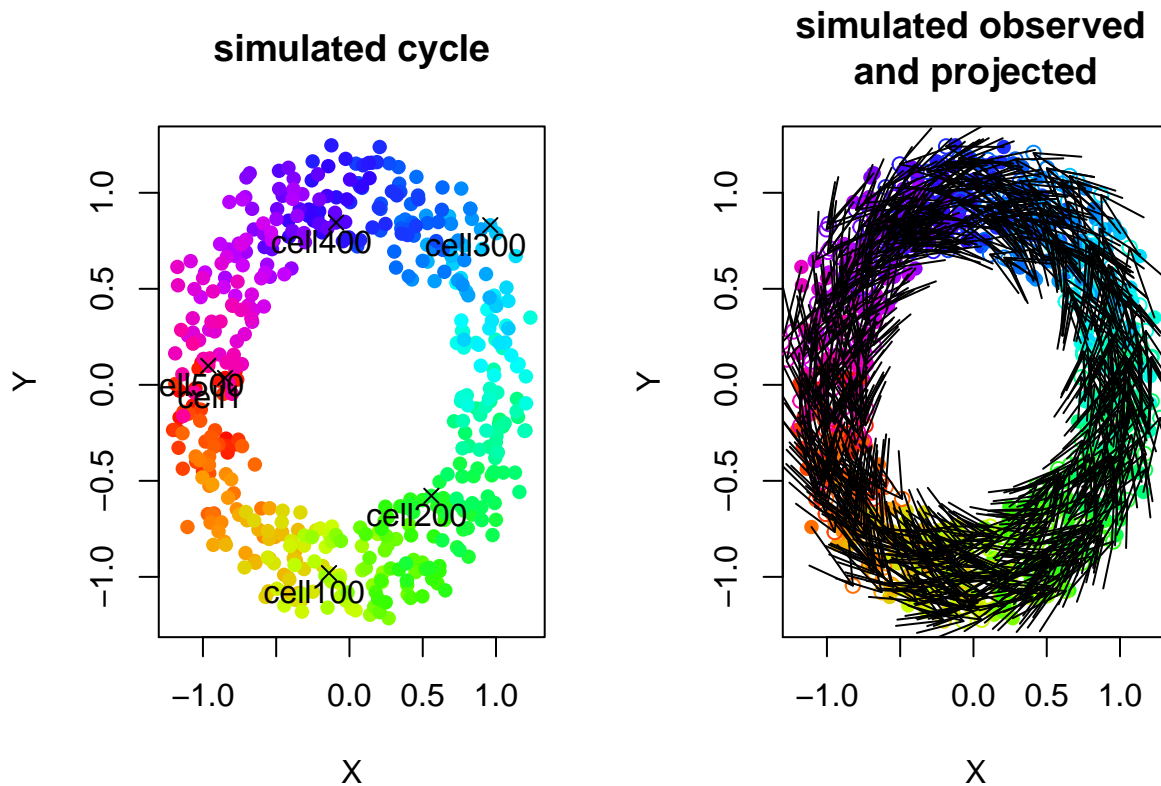
simulated cycle

simulated observed and projected

Non-velocity based embedding on current expression

```r
pca = RSpectra::svds(A = obs, k=3,
                     opts = list(center = TRUE, scale = TRUE,
                                 maxitr = 2000, tol = 1e-10))
var = pca$d
pcs = pca$u
rownames(pcs) = labels

par(mfrow = c(2,2))
#PCA
emb.pca = pcs[,1:2]
plot(emb.pca, pch=16, main = "PCA", xlab = 'PC1', ylab = 'PC2', col = col)

#UMAP
set.seed(1)
emb.umap = uwot::umap(pcs, n_neighbors = 100L)
plot(emb.umap,pch=16, main = "UMAP", xlab = 'UMAP X', ylab = 'UMAP Y',col = col)

#tSNE
set.seed(1)
emb.tsne = Rtsne::Rtsne(pcs,
                        is_distance = FALSE, perplexity = 100,
                        pca = FALSE, num_threads =1, verbose = FALSE)$Y
plot(emb.tsne,pch=16, main = "t-SNE", xlab = 't-SNE X', ylab = 't-SNE Y',col=col)

#diffusion map
set.seed(1)
```
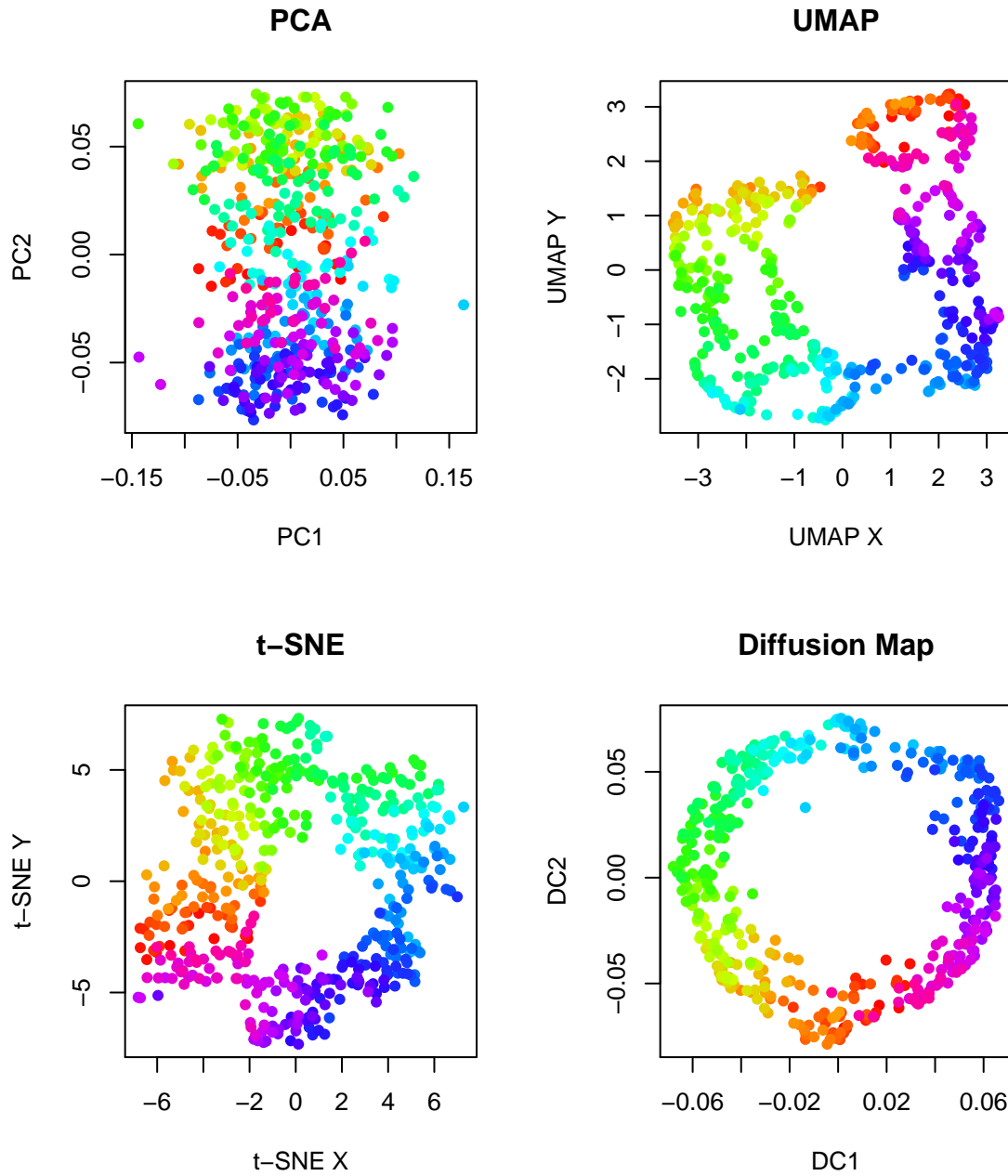
```
diffmap = destiny::DiffusionMap(pcs, k=50)
emb.diffmap = destiny::eigenvectors(diffmap)[,1:2]
plot(emb.diffmap,pch=16, main = "Diffusion Map", xlab = 'DC1', ylab = 'DC2',col=col)
```

PCA

UMAP


t−SNE

Diffusion Map

VeloViz Embedding

```
#, fig.width=7,fig.height=7
set.seed(1) # fig.width=6, fig.height=7
g = graphViz(observed = t(obs), projected = t(exp),
        k = 30, distance_metric = "L2", similarity_metric = "cosine",
        distance_weight = 1, distance_threshold = 1, similarity_threshold = 0.25,
        weighted = TRUE, remove_unconnected = TRUE,
        cell.colors = col, title = "VeloViz",
```

```
          plot = FALSE, return_graph = TRUE)
emb.veloviz = g$fdg_coords
plot(emb.veloviz, pch = 16, main = "VeloViz", xlab = '', ylab = '',col=col)
```

VeloViz
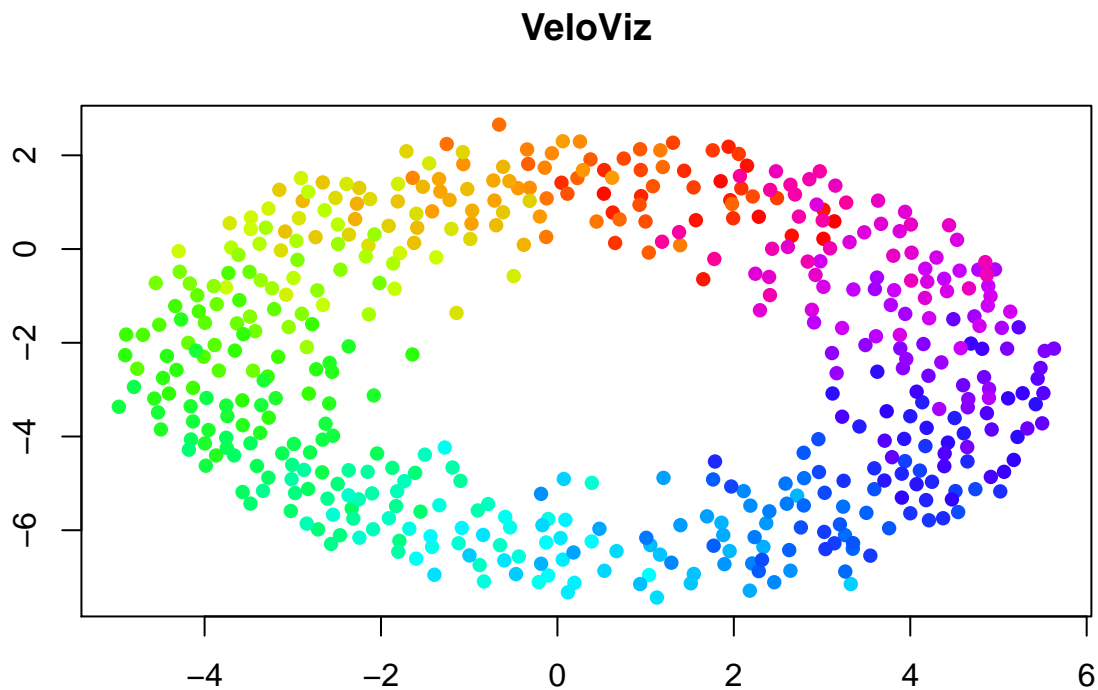
Incomplete Cycle

```
par(mfrow = c(1,2))
## cells to keep
labels.keep <- setdiff(labels, paste0('cell', 200:300))

## incomplete observed and expected
traj.missing = traj[labels.keep,]
obs.missing = obs[labels.keep,]
exp.missing = exp[labels.keep,]
col.missing = col[labels.keep]

plot(obs.missing[,1:2], col=col.missing, pch=16, main = "simulated cycle", xlab = "X", ylab = "Y")
points(obs.missing[c(1,100,200,300),], pch = 4, col = "black")
text(obs.missing[c(1,100,200,300),1:2]-0.1, labels.keep[c(1,100,200,300)])

## plot observed and expected
plot(obs.missing[,1:2],col=col.missing, pch=16,
     main = "simulated observed \nand projected", xlab = "X", ylab = "Y")
points(exp.missing[,1:2],col=col.missing)
arrows(obs.missing[,1:2][,1],obs.missing[,1:2][,2],exp.missing[,1:2][,1],exp.missing[,1:2][,2])
```

simulated cycle

simulated observed and projected
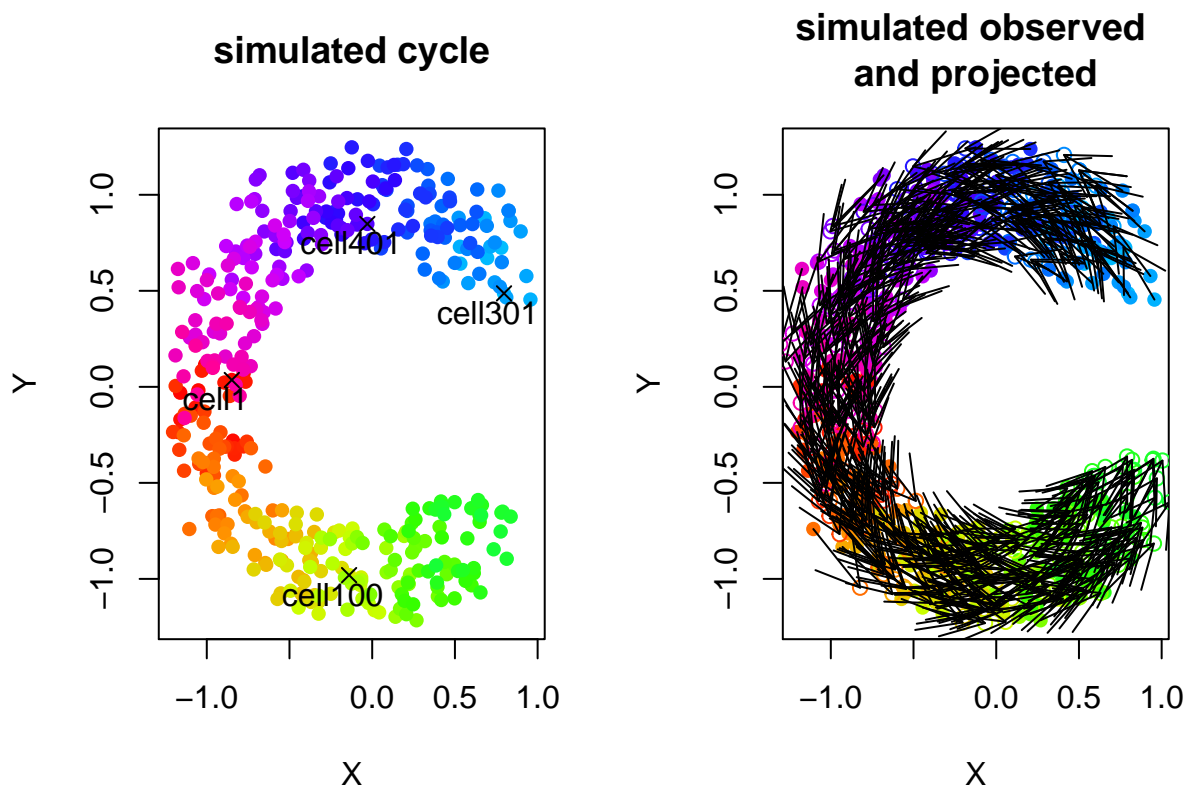
```r
pca = RSpectra::svds(A = obs.missing, k=3,
                     opts = list(center = TRUE, scale = TRUE,
                                 maxitr = 2000, tol = 1e-10))
var = pca$d
pcs = pca$u
rownames(pcs) = labels.keep

par(mfrow = c(2,2))
#PCA
emb.pca = pcs[,1:2]
plot(emb.pca, pch=16, main = "PCA", xlab = 'PC1', ylab = 'PC2', col = col)

#UMAP
set.seed(1)
emb.umap = uwot::umap(pcs, n_neighbors = 100L)
plot(emb.umap,pch=16, main = "UMAP", xlab = 'UMAP X', ylab = 'UMAP Y',col = col)

#tSNE
set.seed(1)
emb.tsne = Rtsne::Rtsne(pcs,
                        is_distance = FALSE, perplexity = 100, pca = FALSE,
                        num_threads =1, verbose = FALSE)$Y
plot(emb.tsne,pch=16, main = "t-SNE", xlab = 't-SNE X', ylab = 't-SNE Y',col=col)

#diffusion map
set.seed(1)
diffmap = destiny::DiffusionMap(pcs, k=50)
emb.diffmap = destiny::eigenvectors(diffmap)[,1:2]
plot(emb.diffmap,pch=16, main = "Diffusion Map", xlab = 'DC1', ylab = 'DC2',col=col)
```
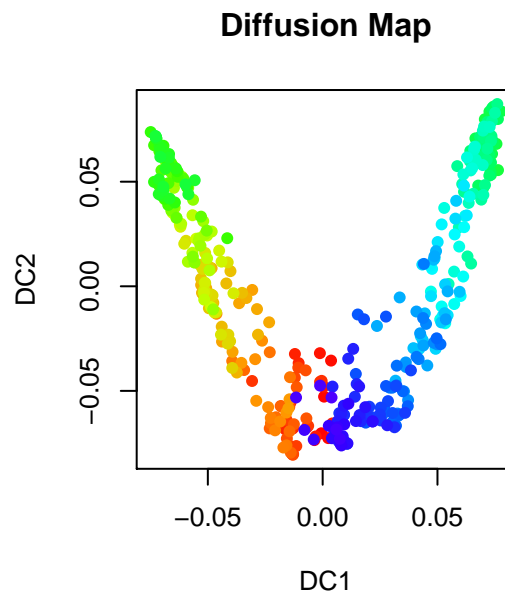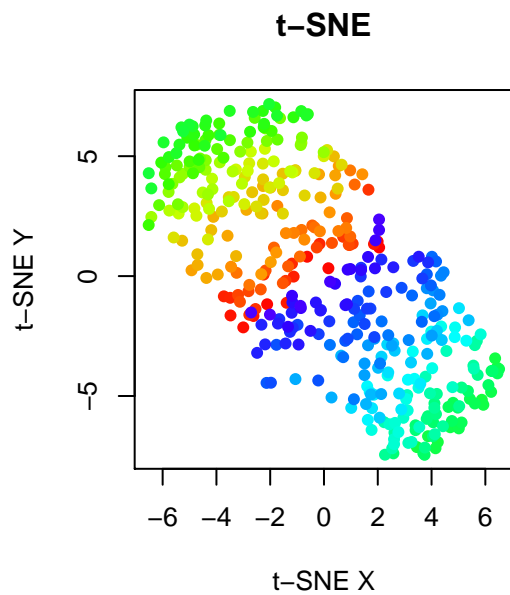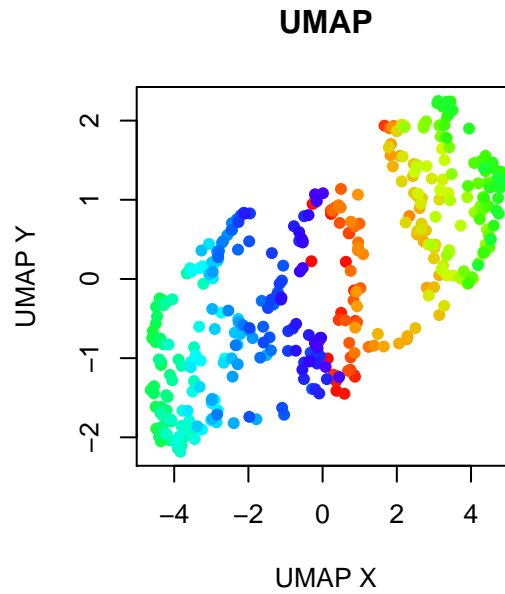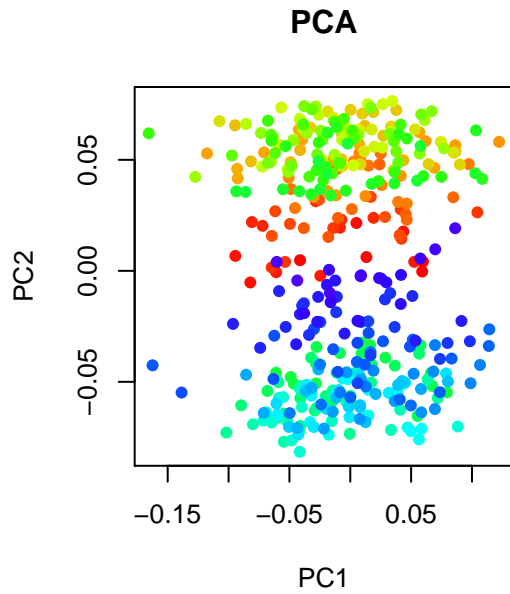
```
k = 30
distance.weight = 1
distance.threshold = 1
similarity.threshold = 0


set.seed(1)
g = graphViz(observed = t(obs.missing), projected = t(exp.missing),
        k = k, distance_metric = "L2", similarity_metric = "cosine",
        distance_weight = distance.weight, distance_threshold = distance.threshold,
        similarity_threshold = similarity.threshold, weighted = TRUE,
        remove_unconnected = TRUE,
        cell.colors = col, title = "VeloViz",
        plot = FALSE, return_graph = TRUE)
```

```
emb.veloviz = g$fdg_coords
plot(emb.veloviz, pch = 16, main = "VeloViz", xlab = '', ylab = '',col=col)
```
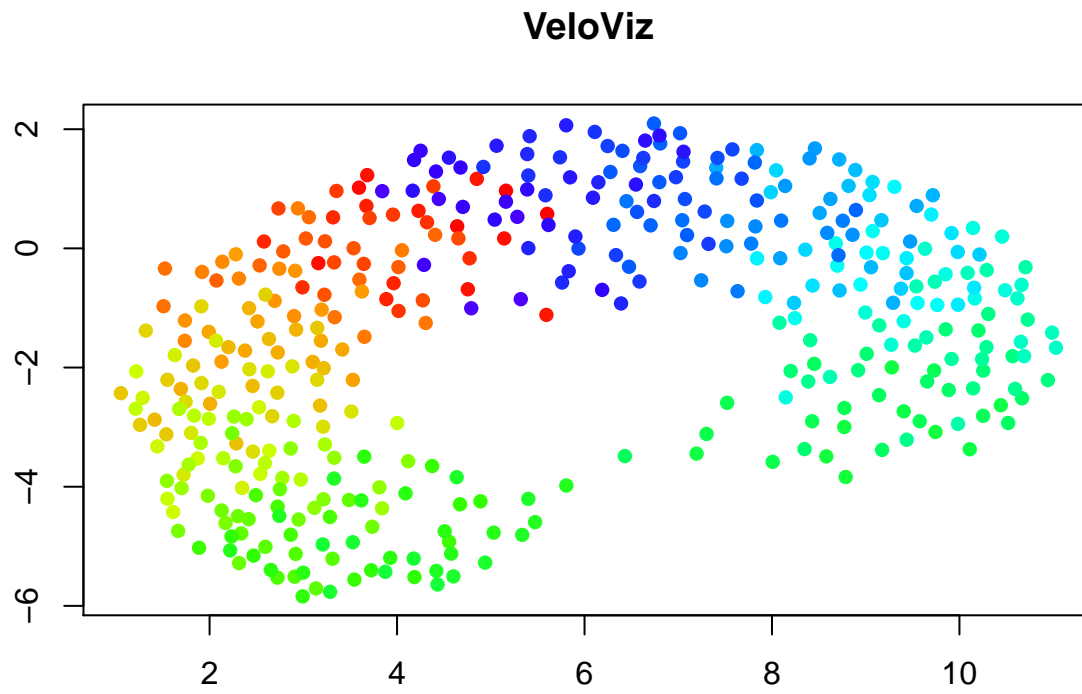
VeloViz

Changing VeloViz Parameters

To understand the effect of each of the parameters, let's first go through how the VeloViz graph is built to see where each of the parameters comes into play:

Once we have the current and projected transcriptional states in PC space, VeloViz calculates a composite distance for each cell pair. This composite distance has two components: a PC distance component, and a velocity similarity component. If we're considering the composite distance from Cell A to Cell B, the PC distance component measures how close a Cell A's projected state is to Cell B. The velocity similarity component measures how similar Cell A's velocity vector is to the vector representing the transition from Cell A to Cell B.

With these composite distances VeloViz creates a k-nearest neighbor graph by assigning `k` edges from each cell to the `k` cells with the minimum composite distances. These edges will have a weight corresponding to the composite distance if `weighted = TRUE`. We can change the relative importance of these two components by changing the `distance_weight` parameter. Setting `distance_weight` to 0, results in a graph that only uses velocity similarity to assign edges. Larger values of `distance_weight` place increasing relative importance on the PC distance component.

After assigning `k` out-edges to each cell, VeloViz removes some of these edges based on two threshold parameters, `distance_threshold` and `similarity_threshold`. The `distance_threshold` is a quantile threshold for the PC distance components of the composite distances. For example, setting `distance_threshold = 0.2` means that any edges where the PC distance component is not in the smallest 20% of PC distances will be removed from the graph; `distance_threshold = 1` includes all edges and does not prune based on distance. Note here that this is 20% of *all* computed PC distances, not just those in the k-nearest neighbor graph. The `similarity_threshold` specifies the minimum cosine similarity between the velocity vector and the cell transition vector for an edge to be included. For example, setting `similarity_threshold = 0` removes any edges where the velocity and cell transition vectors are orthogonal or less similar; `similarity_threshold = -1` includes all edges and does not prune based on similarity.

K: Number of nearest neighbors

Now let's explore how changing the VeloViz parameters changes the embedding, starting with `k`:

```r
par(mfrow = c(2,3))
ks = c(2,5,10,30,50,100)

distance.weight = 1
distance.threshold = 1
similarity.threshold = 0

for (k in ks){

  set.seed(1)
  g = graphViz(observed = t(obs.missing), projected = t(exp.missing),
          k = k, distance_metric = "L2", similarity_metric = "cosine",
          distance_weight = distance.weight, distance_threshold = distance.threshold,
          similarity_threshold = similarity.threshold, weighted = TRUE,
          remove_unconnected = TRUE,
          cell.colors = col, title = "VeloViz",
          plot = FALSE, return_graph = TRUE)
  emb.veloviz = g$fdg_coords
  plot(emb.veloviz, pch = 16, main = paste("VeloViz: k = ",k), xlab = '', ylab = '',col=col)

}
```
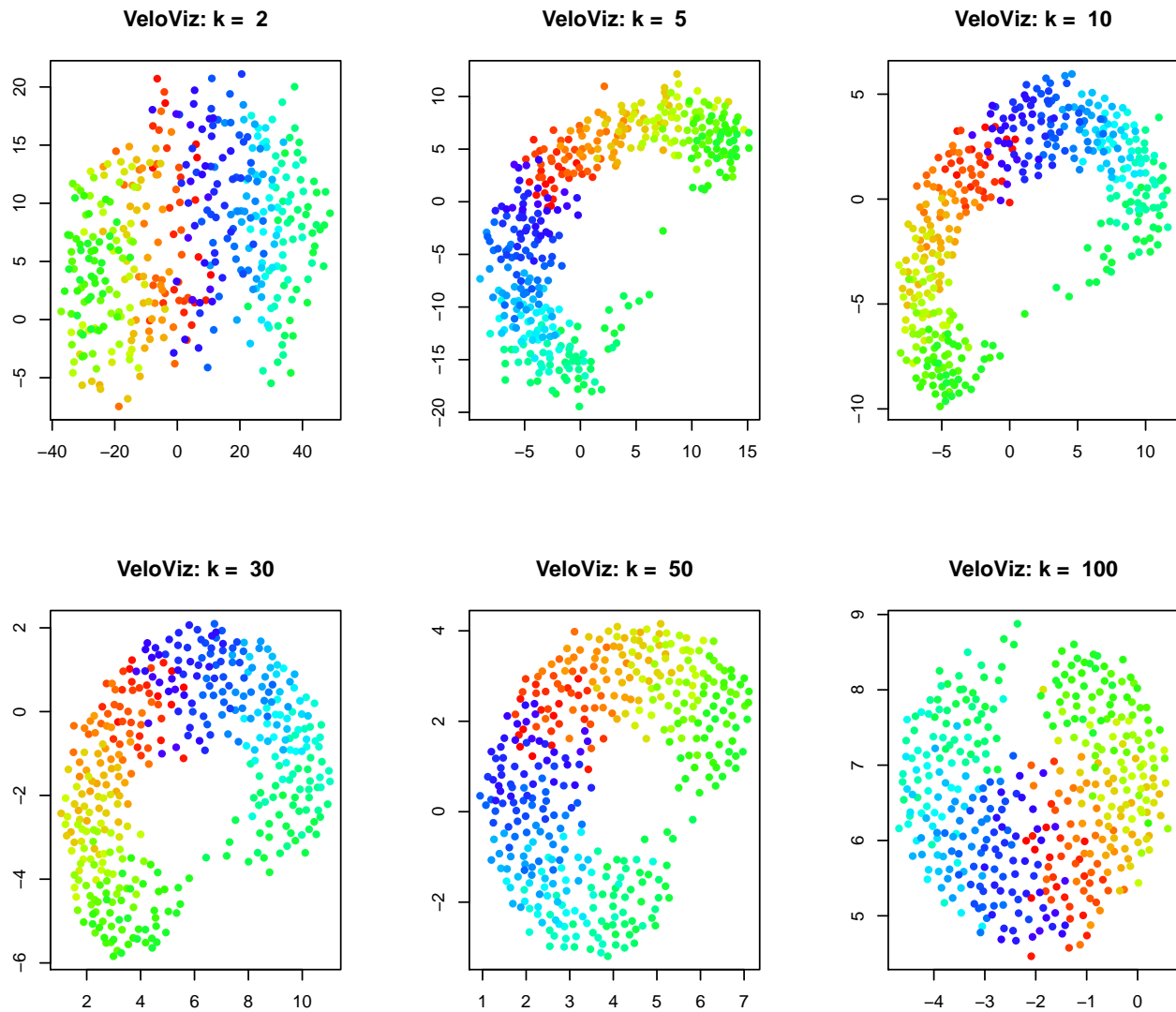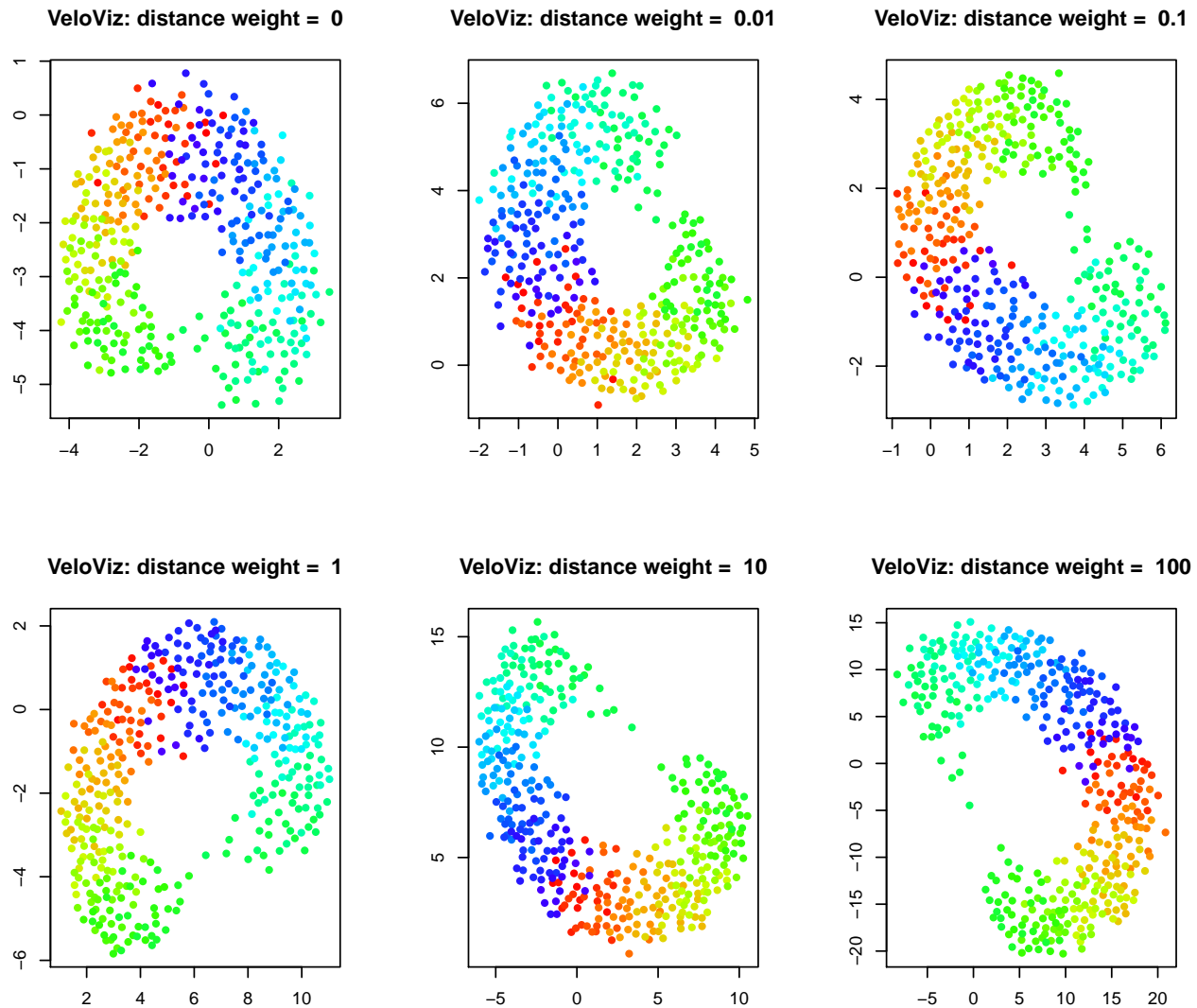
Distance weight:

```
par(mfrow = c(2,3))
dws = c(0,0.01,0.1,1,10,100)

k = 30
distance.threshold = 1
similarity.threshold = 0

for (distance.weight in dws){

  set.seed(1)
  g = graphViz(observed = t(obs.missing), projected = t(exp.missing),
          k = k, distance_metric = "L2", similarity_metric = "cosine",
          distance_weight = distance.weight, distance_threshold = distance.threshold,
          similarity_threshold = similarity.threshold, weighted = TRUE,
          remove_unconnected = TRUE,
          cell.colors = col, title = "VeloViz",
          plot = FALSE, return_graph = TRUE)
```
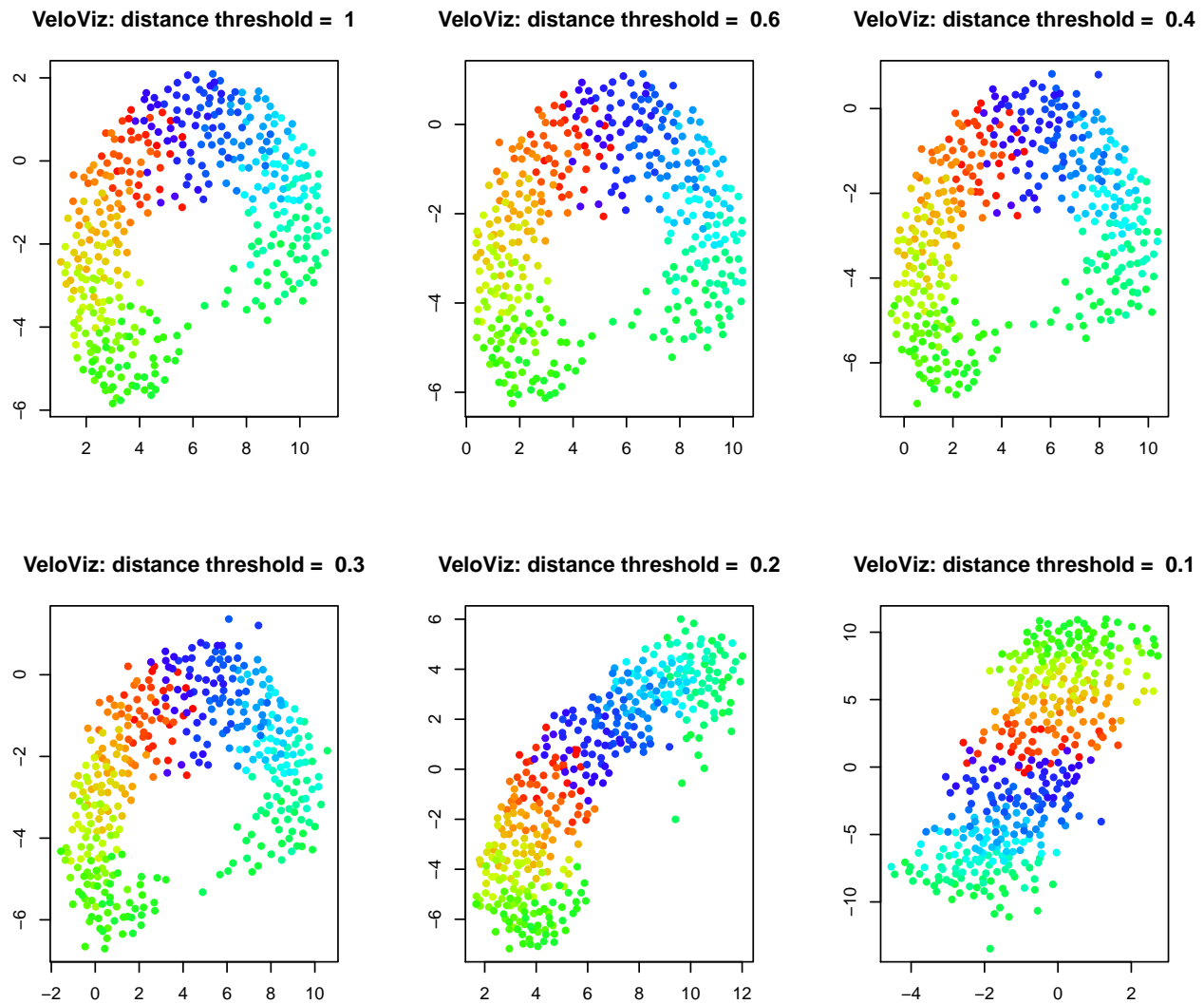
```
  emb.veloviz = g$fdg_coords
  plot(emb.veloviz, pch = 16, main = paste("VeloViz: distance weight = ",distance.weight), xlab = '', yl

}
```


VeloViz: distance weight = 0   **VeloViz: distance weight = 0.01**   **VeloViz: distance weight = 0.1**

VeloViz: distance weight = 1   **VeloViz: distance weight = 10**   **VeloViz: distance weight = 100**

Distance threshold:

```
par(mfrow = c(2,3))
dts = c(1,0.6,0.4,0.3,0.2,0.1)

k = 30
distance.weight = 1
similarity.threshold = 0

for (distance.threshold in dts){

  set.seed(1)
  g = graphViz(observed = t(obs.missing), projected = t(exp.missing),
        k = k, distance_metric = "L2", similarity_metric = "cosine",
```

```
          distance_weight = distance.weight, distance_threshold = distance.threshold,
          similarity_threshold = similarity.threshold, weighted = TRUE,
          remove_unconnected = TRUE,
          cell.colors = col, title = "VeloViz",
          plot = FALSE, return_graph = TRUE)
  emb.veloviz = g$fdg_coords
  plot(emb.veloviz, pch = 16, main = paste("VeloViz: distance threshold = ",distance.threshold), xlab =

}
```

VeloViz: distance threshold = 1   **VeloViz: distance threshold = 0.6**   **VeloViz: distance threshold = 0.4**

VeloViz: distance threshold = 0.3   **VeloViz: distance threshold = 0.2**   **VeloViz: distance threshold = 0.1**

Similarity threshold:

```
par(mfrow = c(2,3))
sts = c(-1,0,0.25,0.5,0.7,0.9)

k = 30
distance.weight = 1
distance.threshold = 1

for (similarity.threshold in sts){
```
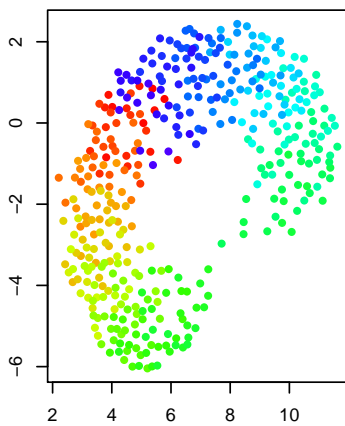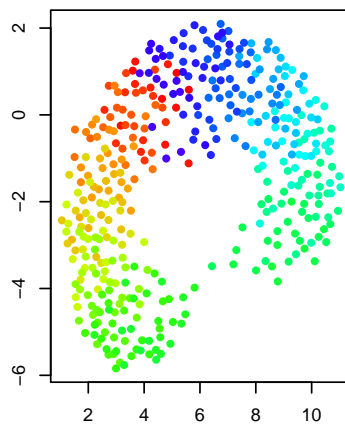
```
set.seed(1)
g = graphViz(observed = t(obs.missing), projected = t(exp.missing),
        k = k, distance_metric = "L2", similarity_metric = "cosine",
        distance_weight = distance.weight, distance_threshold = distance.threshold,
        similarity_threshold = similarity.threshold, weighted = TRUE,
        remove_unconnected = TRUE,
        cell.colors = col, title = "VeloViz",
        plot = FALSE, return_graph = TRUE)
emb.veloviz = g$fdg_coords
plot(emb.veloviz, pch = 16, main = paste("VeloViz: similarity threshold = ",similarity.threshold), xla
}
```