Building a t-SNE embedding

ADVANCED DIMENSIONALITY REDUCTION IN R

Federico Castanedo
Data Scientist at DataRobot

Introduction to t-SNE

- Published by Van der Maaten & Hinton in 2008
- Non-linear dimensionality reduction technique
- Works well for most of the problems and is a very good method for visualizing high dimensional datasets
- Rather than keeping dissimilar points apart (like PCA) it keeps the low-dimensional representation of similar points together

t-SNE method

1. Use **PCA** to reduce the input dimensions into a small number
2. Construct a **probability distribution** over pairs of original **high dimensional** records
3. Define a similarity **probability distribution** of the points in the **low-dimensional embedding**
4. **Minimize** the **K-L** divergence between the two distributions using gradient descent method

t-SNE in R

```r
library(Rtsne)
tsne_output <- Rtsne(mnist[, -1])
```

Modifying default parameters

```r
tsne_output <- Rtsne(mnist[, -1], PCA = FALSE, dims = 3)
```

Embedding Coordinates

Coordinates of the embedding

```
head(tsne_output$Y)
```

```
            [,1]      [,2]       [,3]
[1,]    5.651874 19.728930 -29.775616
[2,]   28.849641  1.487309  13.070709
[3,]   -6.767665 32.623485   2.336025
[4,] -28.991036  7.093971   4.314651
[5,]   31.776286  3.401418   8.544725
[6,]   20.192017 -2.069217   9.893550
```

Analyzing the K-L divergence costs

K-L divergence after every 50th iteration

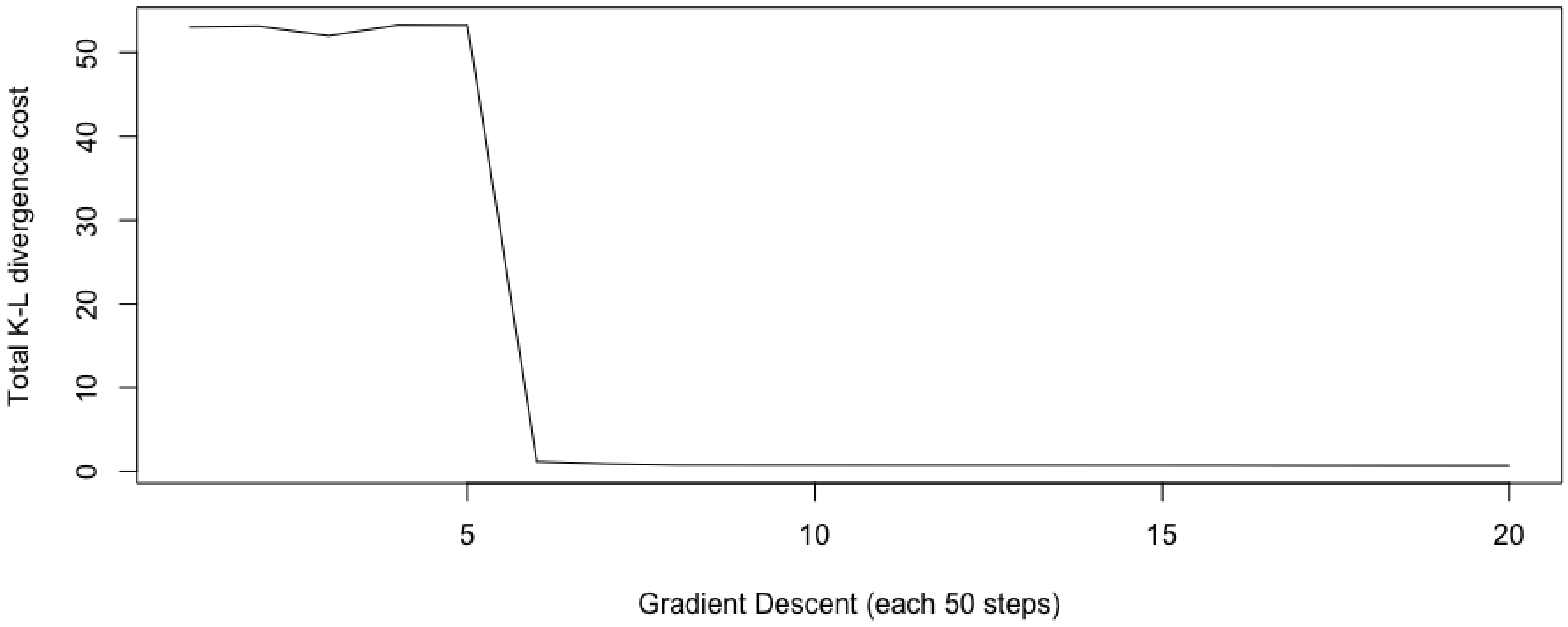
```
tsne_output$itercosts
```

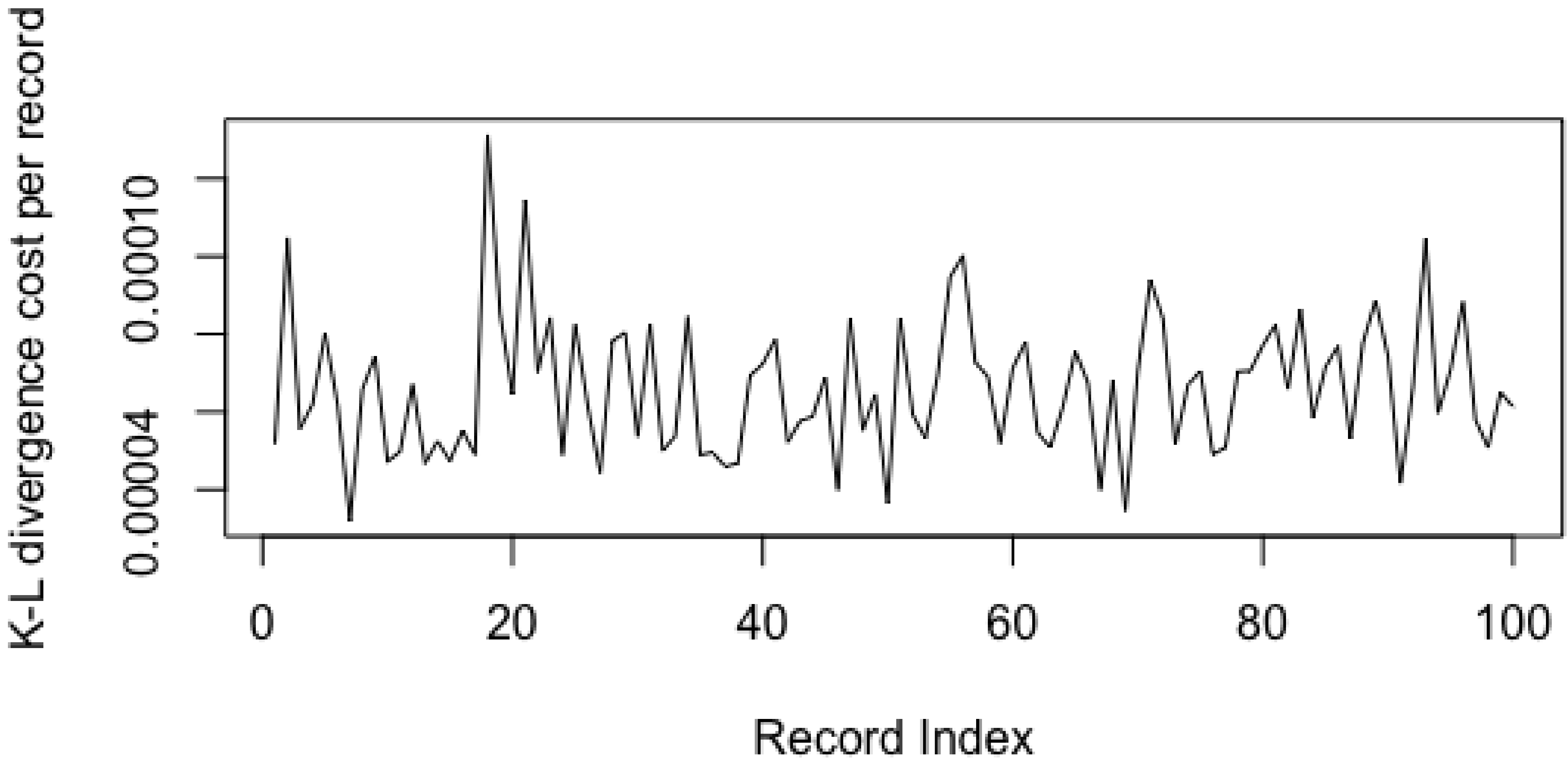
```
 [1] 114.644309 114.644291 109.257076  98.462824  96.049333   4.126860
 [7]   3.740483   3.520209   3.368242   3.253243   3.160782   3.083919
[13]   3.018355   2.961709   2.911557   2.866945   2.826918   2.790768
[19]   2.757847   2.727729
```

Cost of each record after last iteration

```
head(tsne_output$costs)
```

```
[1] 0.00005259133 0.00010117028 0.00005516008 0.00006157778
[5] 0.00007992530 0.00006642461
```


Total K-L divergence cost
0
10
20
30
40
50
5
10
15
20
Gradient Descent (each 50 steps)


K-L divergence cost per record
0.00004
0.00010
0
20
40
60
80
100
Record Index

Let's go practice!

ADVANCED DIMENSIONALITY REDUCTION IN R

Optimal number of t-SNE iterations

ADVANCED DIMENSIONALITY REDUCTION IN R

Federico Castanedo
Data Scientist at DataRobot

What is a hyper-parameter?

- A parameter whose value is not learned from the data and is set beforehand
  - Number of iterations
  - Perplexity
  - Learning rate
- Optimization criterium: K-L divergence

t-SNE stochastic nature

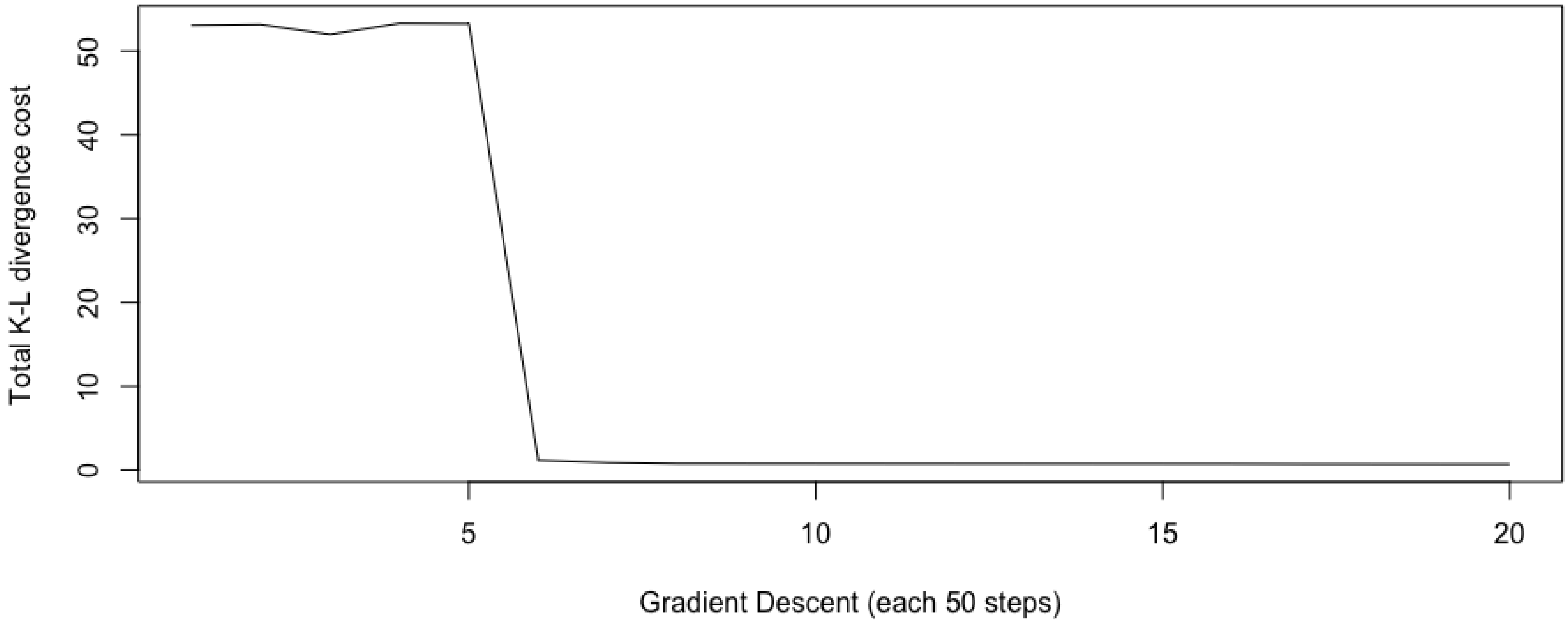
- The algorithm is non-deterministic:
  - Different executions with the same hyper-parameters will provide different results

```
set.seed(1234)
tsne_output_1 <- Rtsne(mnist[, -1], max_iter = 1500)

set.seed(1234)
tsne_output_2 <- Rtsne(mnist[, -1], max_iter = 1500)

identical(tsne_output_1, tsne_output_2)
```

```
TRUE
```

Setting the optimal number of iterations

- `max_iter` : default value of 1000 iterations.
- Optimal number of iterations depends on the dataset.



Let's practice!

ADVANCED DIMENSIONALITY REDUCTION IN R

Effect of perplexity parameter in the t-SNE embedding

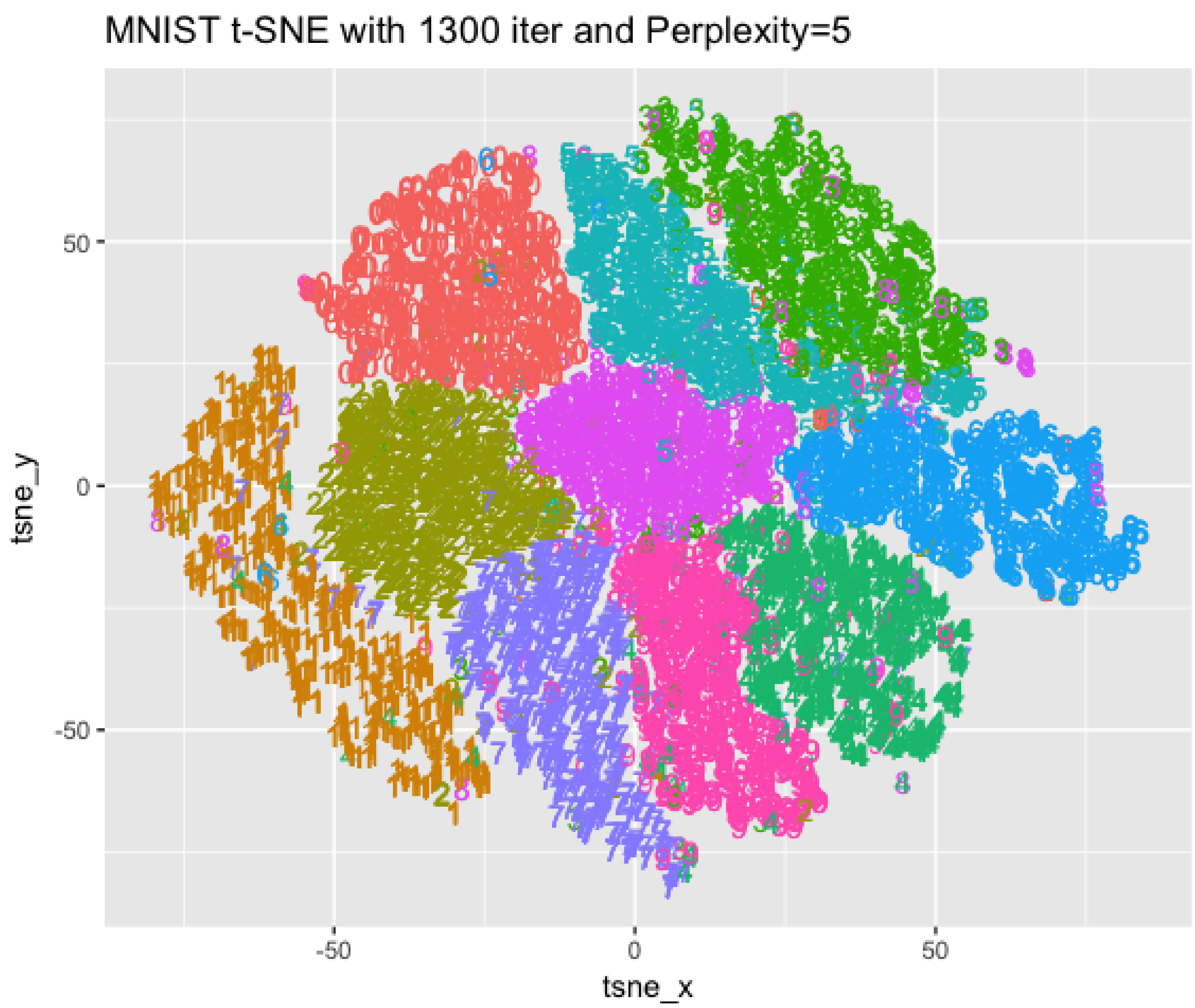
ADVANCED DIMENSIONALITY REDUCTION IN R

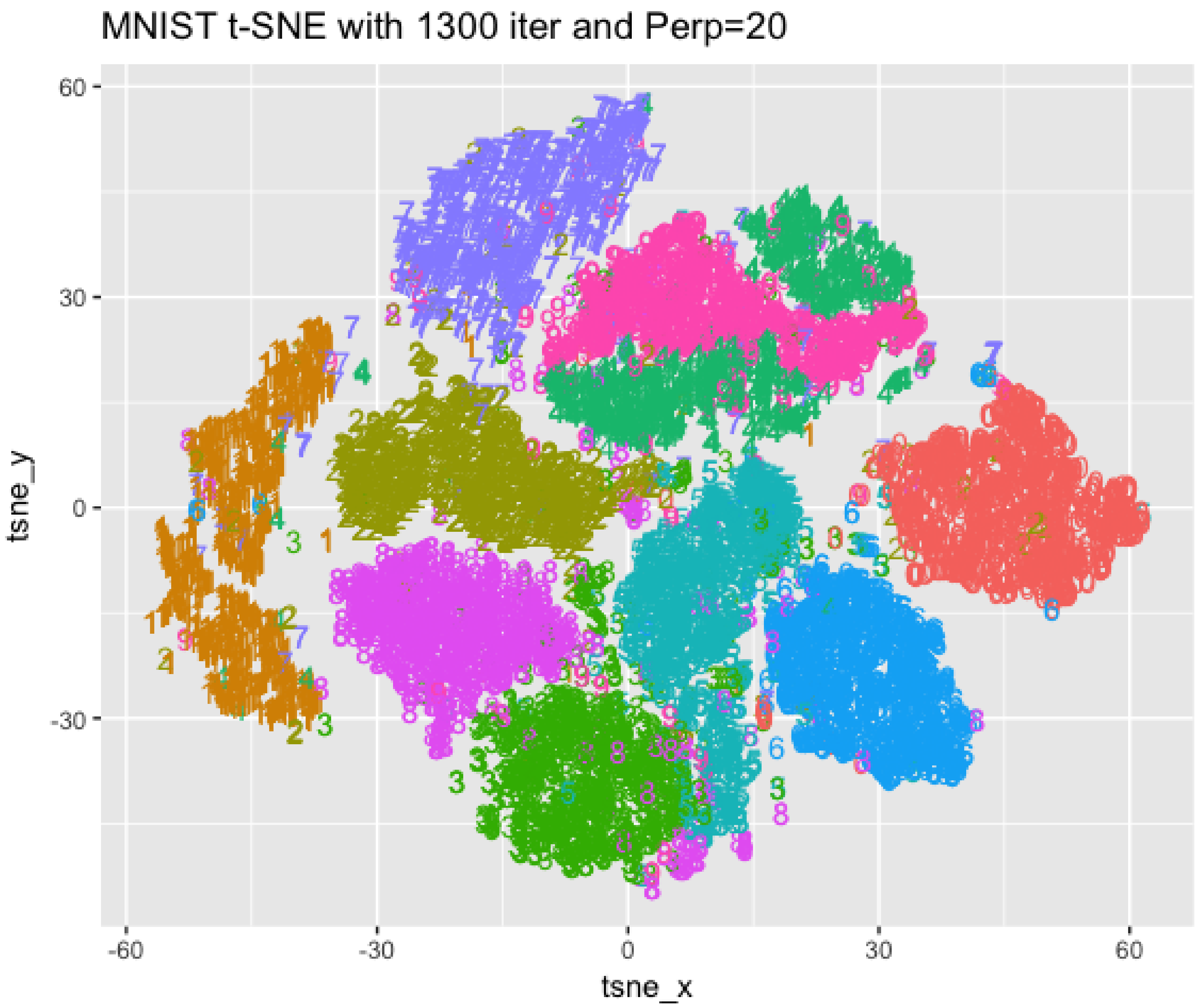
Federico Castanedo
Data Scientist at DataRobot

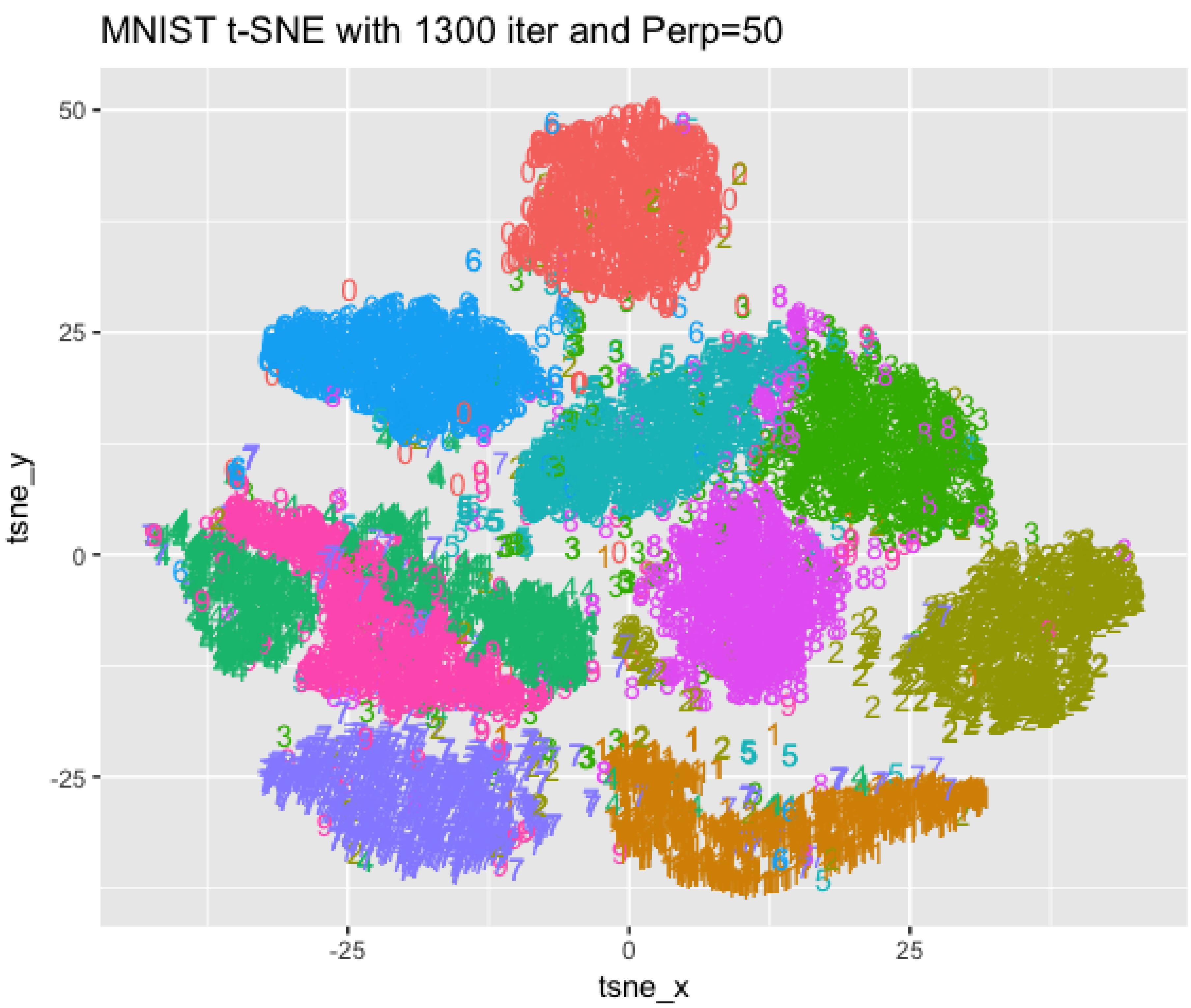
What is the perplexity parameter?

- Balance attention between local and global aspects of the dataset
- A guess about the number of close neighbors
- In a real setting is important to try different values
- Must be lower than the number of input records

```
tsne_output <- Rtsne(mnist[, -1], perplexity = 50, max_iter = 1300)
```


MNIST t-SNE with 1300 iter and Perplexity=5
tsne_y
tsne_x
50
0
-50
-50
0
50


MNIST t-SNE with 1300 iter and Perp=20
tsne_y
tsne_x
60
30
0
-30
-60
-30
0
30
60


MNIST t-SNE with 1300 iter and Perp=50
50
25
0
-25
-25
0
25
tsne_y
tsne_x

Let's practice!

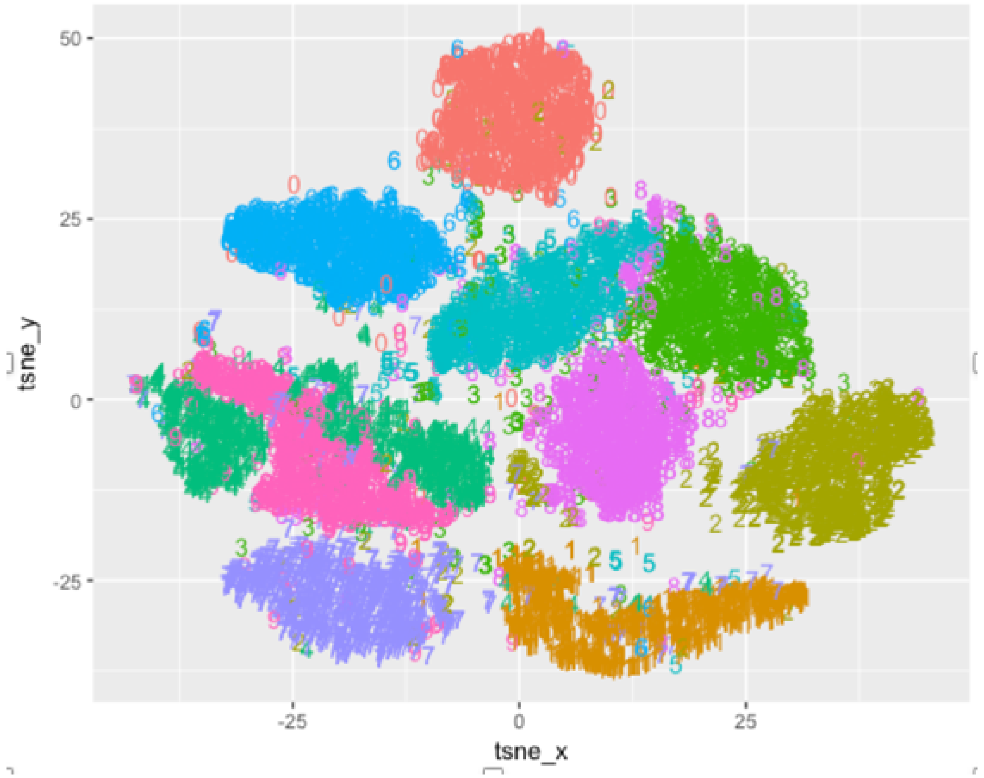
ADVANCED DIMENSIONALITY REDUCTION IN R

Using t-SNE embeddings in classification tasks

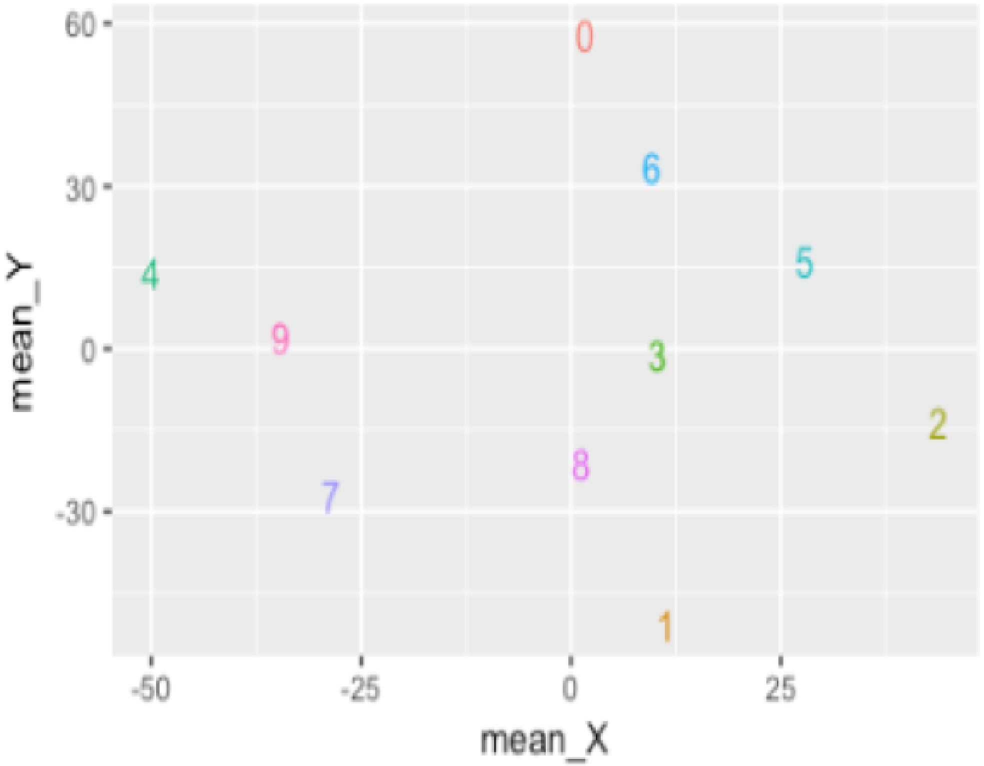
ADVANCED DIMENSIONALITY REDUCTION IN R

Federico Castanedo
Data Scientist at DataRobot

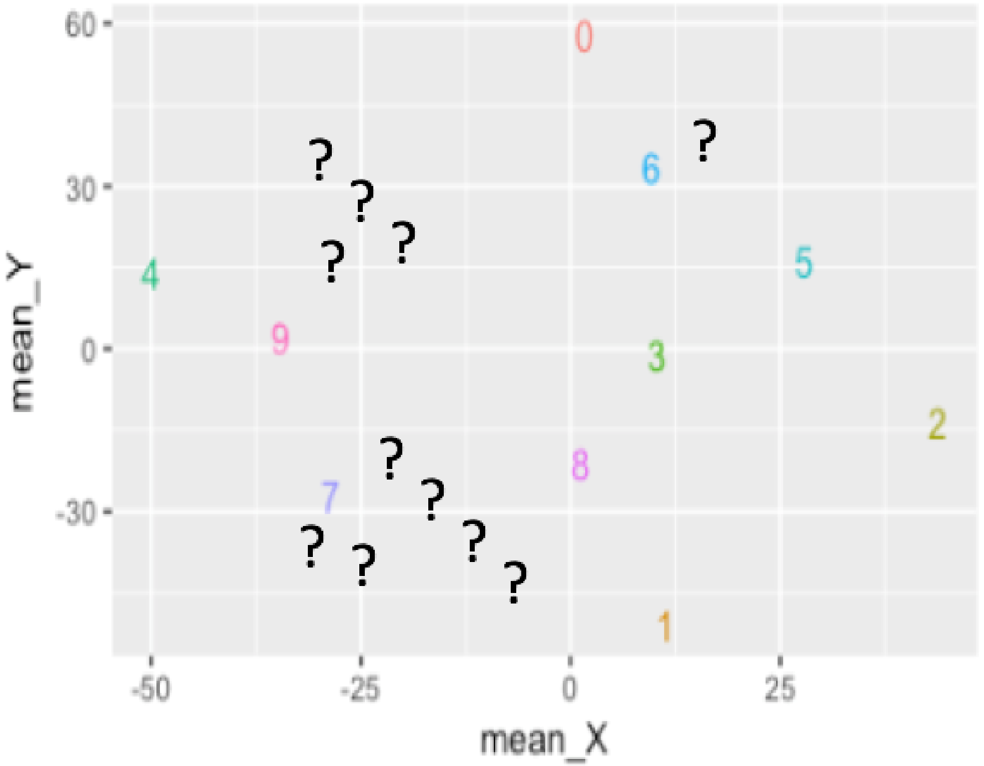
Classifying digits with t-SNE



Step 1. Build a t-SNE embedding using labeled data



Step 2. Compute the centroids of each label



Step 3. Classify un-labeled digits based on the Euclidean distance to the centroids

Step 1: generating t-SNE features

Perform t-SNE embedding

```
tsne <- Rtsne(mnist_10k[, -1], perplexity = 5)
```

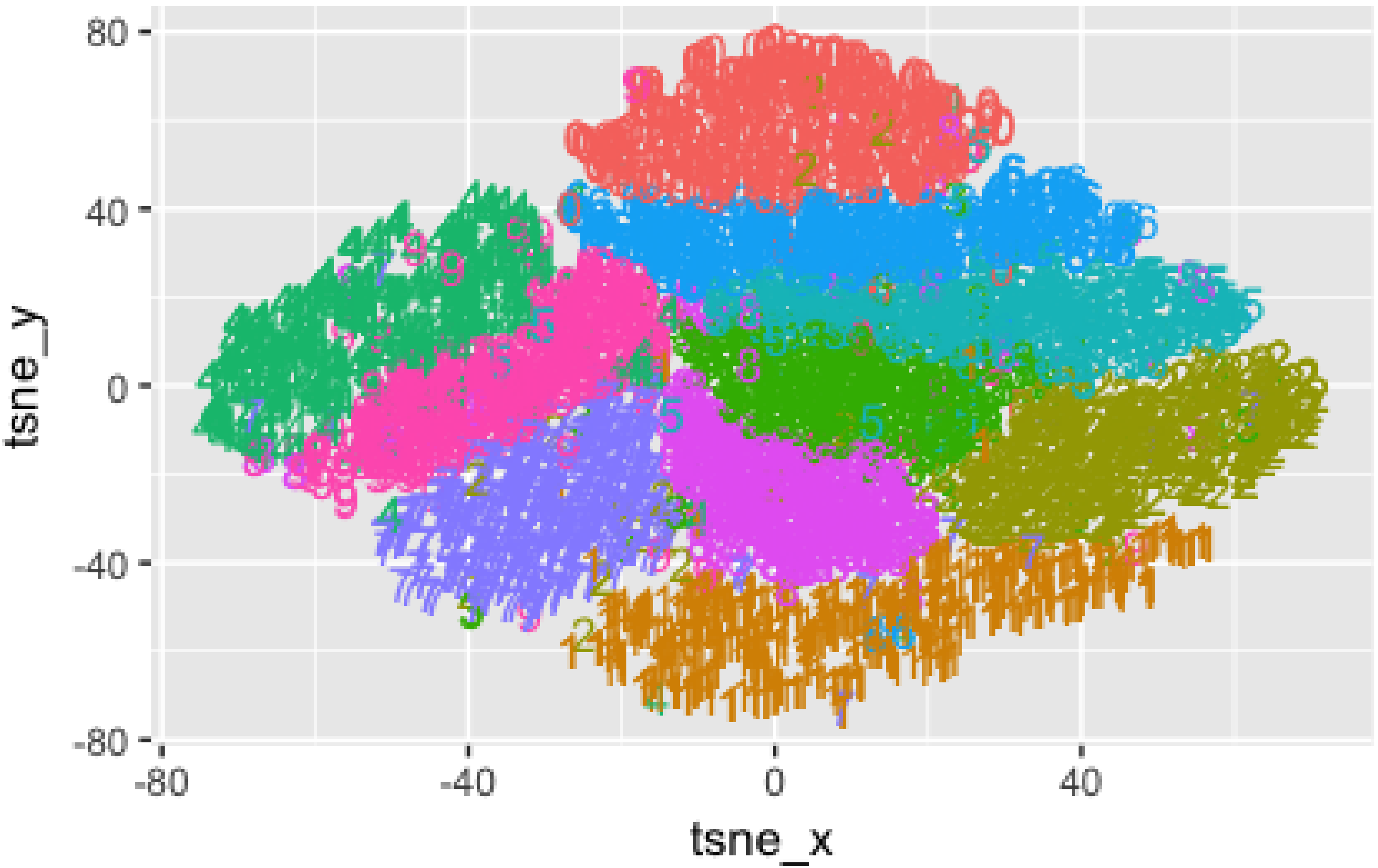
Generate data frame

```
tsne_plot <- data.frame(tsne_x= tsne_out$Y[1:5000,1],
                        tsne_y = tsne_out$Y[1:5000,2],
                        digit = as.factor(mnist_10k[1:5000,]$label))
```

Step 1: visualize obtained embedding

```
ggplot(tsne_plot, aes(x= tsne_x,
       y = tsne_y, color = digit)) +
       ggtitle("MNIST embedding of the first 5K digits") +
       geom_text(aes(label = digit)) + theme(legend.position="none")
```

MNIST embedding of the first 5K digits



Step 2: computing centroids

Get t-SNE coordinates

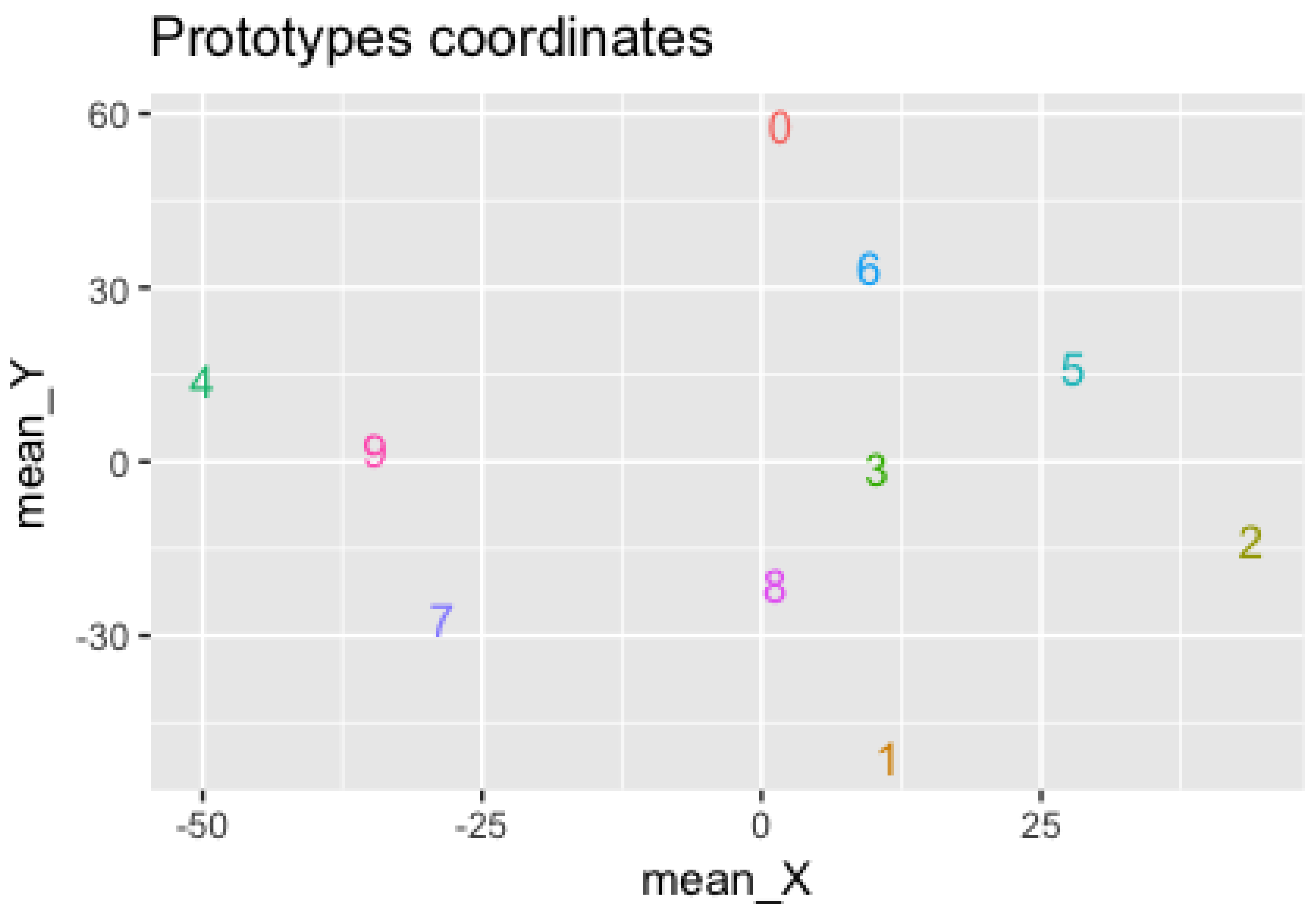
```
centroids <- as.data.table(tsne_out$Y[1:5000,])
setnames(centroids, c("X", "Y"))
centroids[, label := as.factor(mnist_10k[1:5000,]$label)]
```

Compute centroids

```
centroids[, mean_X := mean(X), by = label]
centroids[, mean_Y := mean(Y), by = label]
centroids <- unique(centroids, by = "label")
```

Step 2: visualize centroids

```
ggplot(centroids, aes(x= mean_X, y = mean_Y, color = label)) +
  ggtitle("Centroids coordinates") + geom_text(aes(label = label)) +
  theme(legend.position = "none")
```



Prototypes coordinates
60
30
0
-30
mean_Y
-50
-25
0
25
mean_X
0
1
2
3
4
5
6
7
8
9

Step 3: classifying new digits

Get new examples of digits 4 and 9

```
distances <- as.data.table(tsne_out$Y[5001:10000,])
setnames(distances, c("X", "Y"))
distances[, label := mnist_10k[5001:10000,]$label]
distances <- distances[label == 4 | label == 9]
```

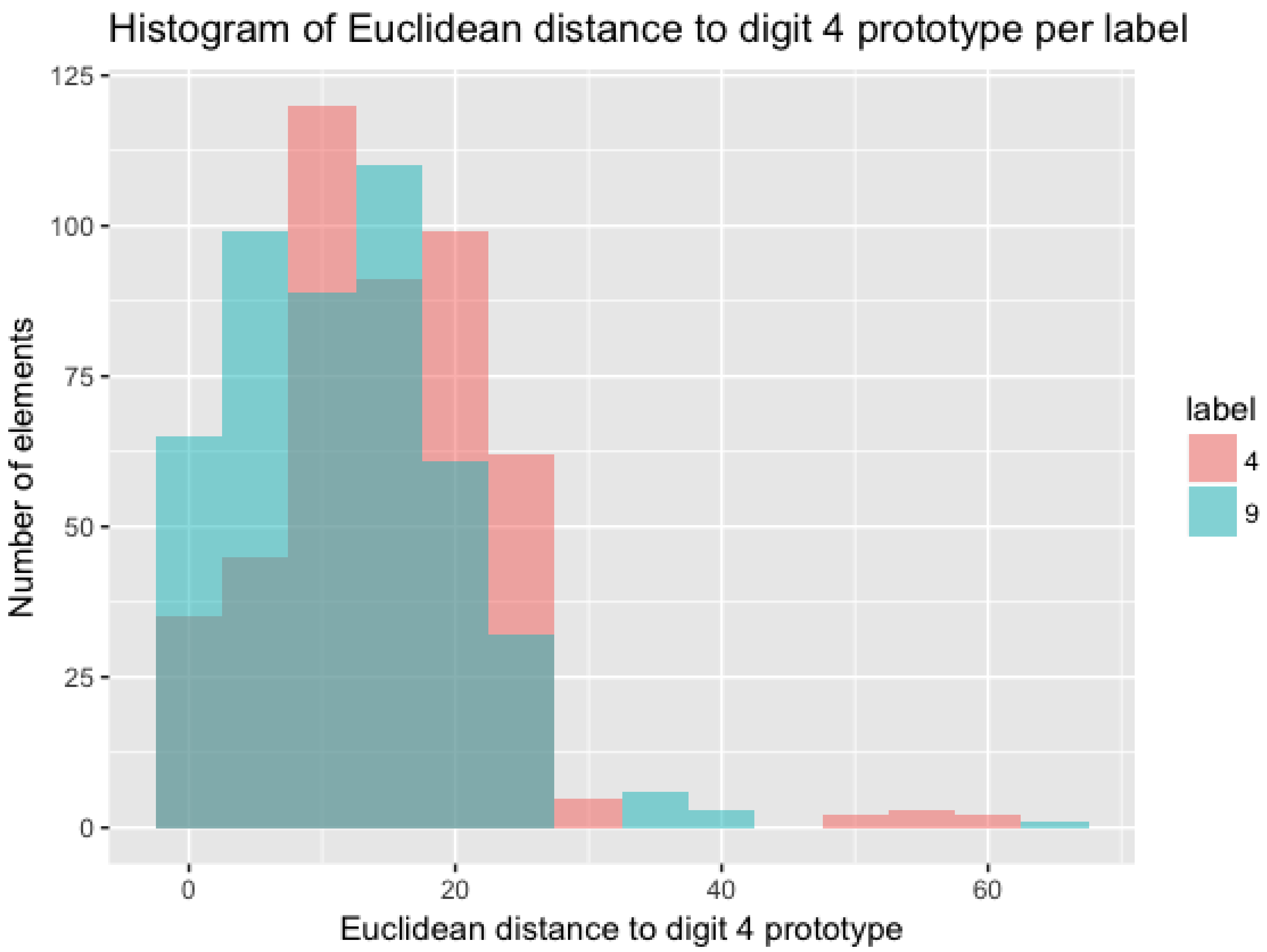
Compute the distance to the centroids

```
distances[, dist_4 := sqrt(((X - centroids[label==4,]$mean_X)
    + (Y - centroids[label==4,]$mean_Y))^2)]
```

Step 3: computing histogram

Plot distance to each centroid

```
ggplot(distances, aes(x=dist_4, fill = as.factor(label))) + geom_histogram(binwidth=5, a
    position="identity", show.legend = F)
```


Histogram of Euclidean distance to digit 4 prototype per label
Number of elements
125
100
75
50
25
0
0
20
40
60
Euclidean distance to digit 4 prototype
label
4
9

Lets practice!

ADVANCED DIMENSIONALITY REDUCTION IN R