nonames2_hw9

```
# install.packages('e1071')
library(e1071)

# install.packages('scales')
library(scales)

library(ggplot2)

# install.packages('RColorBrewer')
library(RColorBrewer)

library(gridExtra)

# install.packages('cowplot')
library(cowplot)
```

```
##
## Attaching package: 'cowplot'
```

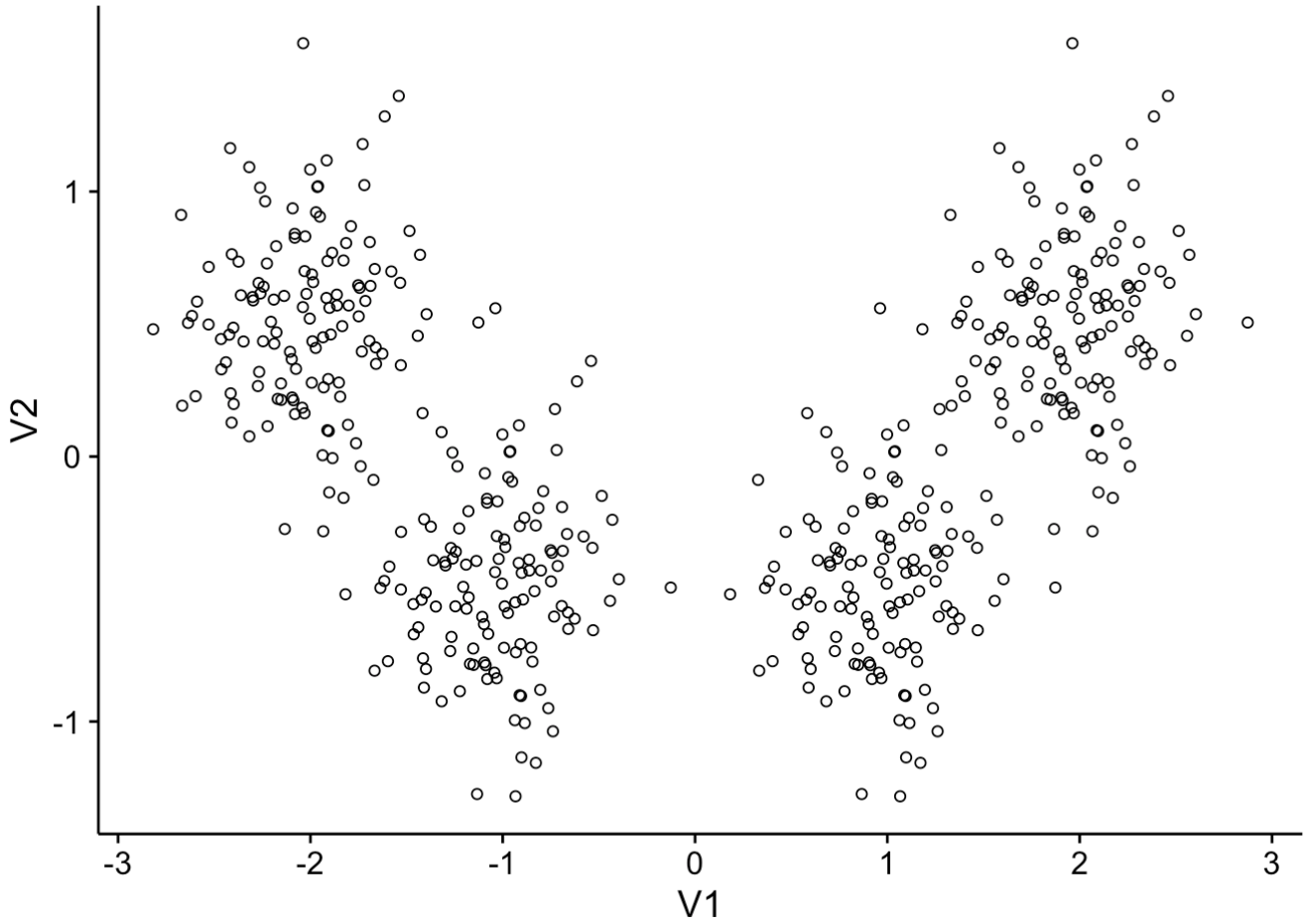
```
## The following object is masked from 'package:ggplot2':
##
##     ggsave
```

```
# install.packages('deldir', dependencies=TRUE)
library(deldir)
```

```
## deldir 0.1-14
```

In this problem set we will implement and apply the standard (batch) K-means algorithm, the online version, and the “soft” clustering procedures. The file cluster.dat contains a data set of p = 500 (2-dimensional) observations generated from four different Gaussians with four different means.

```
X <- t(as.matrix((read.table("cluster.dat", header = FALSE))))
ggplot(as.data.frame(X), aes(x = V1, y = V2)) + geom_point(shape = 1)
```



9.1 K-means Clustering (3 points) Write a program that implements the standard version of K-means clustering and partitions the given data set into K clusters. Repeat the clustering procedure for different initializations of the prototypes and K = 2, 3, 4, 5, 6, 7, 8. Include the following steps:

Initialization – • Set the initial prototypes wq randomly around the data set mean

```
# K <- 2 tmax <- 5

wq_matrix <- function(X, K) {
    wq <- matrix(0, nrow = K, ncol = ncol(X))
    set.seed(1234)
    for (i in 1:K) {
        wq[i, ] <- rnorm(ncol(X), mean(X))
    }
    return(wq)
}

# wq <- wq_matrix(X,K)
```

Optimization – Implement the k-means update (see lecture notes). Each iteration should contain the following two steps • assign all datapoints to their closest prototype

```
# calculate euclidian distrance, allgemein gültig for K > 2
euclidian_distances <- function(X, wq, K) {
    euclidian_matrix <- matrix(0, nrow = nrow(X), ncol = K)

    for (i in 1:nrow(X)) {
        for (j in 1:K) {
            euclidian_matrix[i, j] <- sqrt(sum((X[i, ] - wq[j,
                ])^2))
        }
    }
    return(euclidian_matrix)
}

# euclidian_distance_matrix <- euclidian_distances(X, wq, K)

# mq assign all datapoints to their closest prototype where
# the columns represent the prototype and the row equals the
# data point, allgemein gültig for K > 2
mq_matrix <- function(euclidian_distance_matrix) {
    mq <- matrix(0, nrow = nrow(euclidian_distance_matrix), ncol = ncol(euclidian_dis
tance_matrix))

    for (i in 1:nrow(euclidian_distance_matrix)) {
        c <- which.min(euclidian_distance_matrix[i, ])
        mq[i, c] <- 1
    }
    return(mq)
}

# mq <- mq_matrix(euclidian_distance_matrix)
```

• re-compute the new positions of the prototypes for this assignment

```
wq_update <- function(X, mq) {
    wq_u <- matrix(0, nrow = ncol(mq), ncol = ncol(X))

    for (i in 1:ncol(mq)) {
        for (j in 1:ncol(X)) {
            wq_u[i, j] <- (t(mq[, i]) %*% (X[, j]))/(sum(mq[,
                i]))
        }

    }
    return(wq_u)
}

# wq_1 <- wq_update(X,mq)
```

compute the positions of points belonging to the prototypes for this assignment

```
cluster_points <- function(X, K, mq) {
    cluster_points_list <- rep(list(matrix(1, nrow = nrow(X),
        ncol = X)), K)

    for (k in 1:K) {
        m <- matrix(mq[, k], nrow(mq), ncol = ncol(X))
        cluster_points_list[[k]] <- m * X
    }
    return(cluster_points_list)
}

# wq_1 <- wq_update(X,mq)
```

error function

```
error_function <- function(X, wq, mq, K) {
    s <- matrix(0, 1, ncol = K)

    for (q in 1:K) {

        for (a in 1:nrow(X)) {

            s <- s + mq[a, q] * (abs(X[a, ] - wq[q, ])^2)

        }

    }
    s <- (s/(2 * nrow(X)))
    return(s)
}
```

program that implements the standard version of K-means clustering and partitions the given data set into K clusters

```
k_means_clustering <- function(X, K, tmax) {

    # before get started, save the solution for each iteration in
    # memory
    wq_memory <- rep(list(wq_matrix(X, K)), tmax + 1)
    euclidian_distance_matrix_memory <- rep(list(euclidian_distances(X,
        wq_matrix(X, K), K)), tmax)
    mq_memory <- rep(list(matrix(1, nrow = nrow(X), ncol = K)),
        tmax)

    cluster_memory <- rep(list(matrix(1, nrow = nrow(X), ncol = K)),
        tmax)
    error_memory <- rep(list(matrix(0, nrow = 1, ncol = ncol(X))),
        tmax)

    # Initialization – initial prototypes wq randomly around the
    # data set mean
    wq_memory[[1]] <- wq_matrix(X, K)

    # Optimization – Each iteration should contain the fol-
    # lowing two steps
    for (t in 1:tmax) {

        # assign all datapoints to their closest prototype

        # - calculate euclidian distrance
        euclidian_distance_matrix_memory[[t]] <- euclidian_distances(X,
            wq_memory[[t]], K)
        # - mq assign all datapoints to their closest prototype
        mq_memory[[t]] <- mq_matrix(euclidian_distance_matrix_memory[[t]])

        # re-compute the new positions of the prototypes for this
        # assignment
        wq_memory[[t + 1]] <- wq_update(X, mq_memory[[t]])

        # calculate cluster points belonging to either one of the K
        # Cluster
        cluster_memory[[t]] <- cluster_points(X, K, mq_memory[[t]])

        error_memory[[t]] <- error_function(X, wq_memory[[t +
            1]], mq_memory[[t]], K)
    }

    return_list <- list(wq_memory = wq_memory, euclidian_distance_matrix_memory = euc
lidian_distance_matrix_memory,
        mq_memory = mq_memory, cluster_memory = cluster_memory,
        error_memory = error_memory)
    return(return_list)
}
```

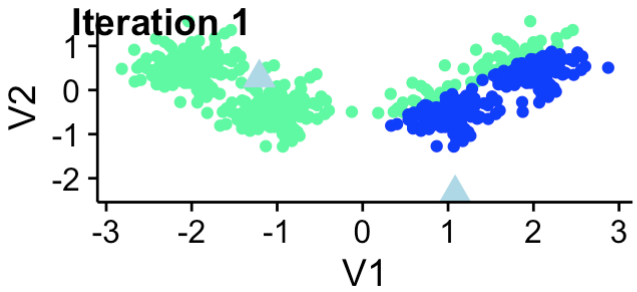
K-means Clustering with tmax = 5 and K = 2, 3, 4, 5, 6, 7, 8

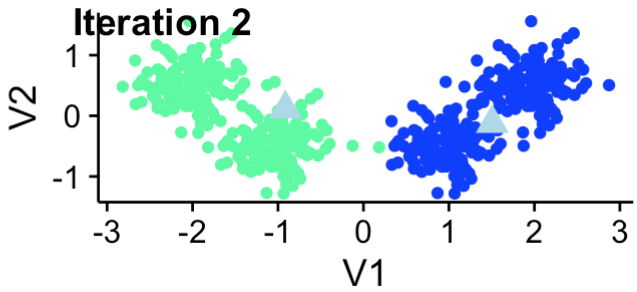
```
K_list <- list(2, 3, 4, 5, 6, 7, 8)
tmax <- 5

options(warn = -1)
cluster_list <- rep(list(k_means_clustering(X, K_list[[1]], tmax)),
    length(K_list))

for (k in 1:length(K_list)) {
    cluster_list[[k]] <- k_means_clustering(X, K_list[[k]], tmax)
}
options(warn = 0)
# cluster point solution for the first k in list (here K =2)
# for the third iteration and the position of data points in
# the second cluster
# head(cluster_list[[1]]$cluster_memory[[3]][[2]])
head(as.data.frame(cluster_list[[1]]$cluster_memory[[3]][[2]]))
```

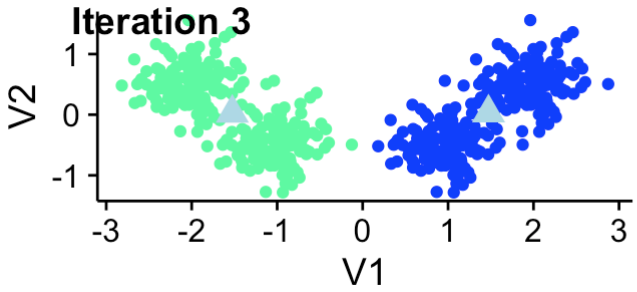
```
##          V1        V2
## V1 2.195194  0.119758
## V2 0.652289 -0.565997
## V3 1.029164 -0.078276
## V4 0.000000  0.000000
## V5 1.972860  0.830741
## V6 0.000000  0.000000
```

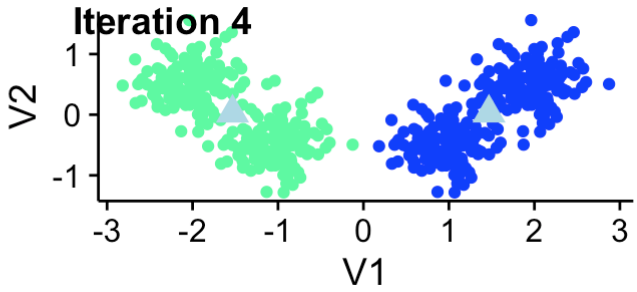
Visualization – (a) Visualize data points and prototypes for each iteration in a sequence of scatter plots.

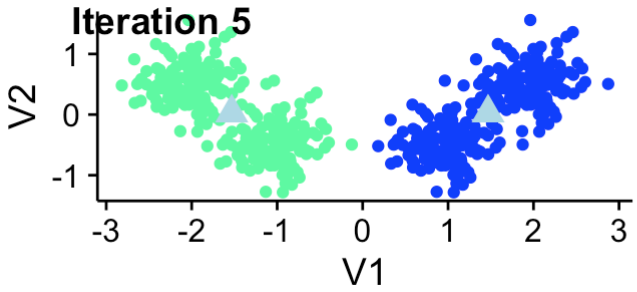
```
# For K = 2

col <- c("seagreen1", "blue", "green", "yellow", "red", "darkgoldenrod1",
    "gray3", "olivedrab")

plot_cluster <- function(cluster_list, K, t) {

    wq <- as.data.frame(cluster_list[[K]]$wq_memory[[t]])

    for (k in 1:length(cluster_list[[K]]$cluster_memory[[t]])) {
        s <- as.data.frame(cluster_list[[K]]$cluster_memory[[t]][[k]])
        s <- s[!(apply(s, 1, function(y) any(y == 0))), ]

        if (k == 1) {
            p <- ggplot() + geom_point(data = s, aes(x = V1,
                y = V2), color = col[k])
        } else {
            p <- p + geom_point(data = s, aes(x = V1, y = V2),
                color = col[k])
        }

    }

    p <- p + geom_point(data = wq, aes(x = V1, y = V2), shape = 17,
        size = 4, color = "lightblue")

    return(p)

}

# pink <- plot_cluster(1,1) pink

plot_cluster_iteration <- function(K) {

    K1 <- plot_cluster(cluster_list, K, 1)
    K2 <- plot_cluster(cluster_list, K, 2)
    K3 <- plot_cluster(cluster_list, K, 3)
    K4 <- plot_cluster(cluster_list, K, 4)
    K5 <- plot_cluster(cluster_list, K, 5)

    plot_grid(K1, K2, K3, K4, K5, labels = c("Iteration 1", "Iteration 2",
        "Iteration 3", "Iteration 4", "Iteration 5"), ncol = 2,
        nrow = 3)

}

plot_cluster_iteration(1)
```



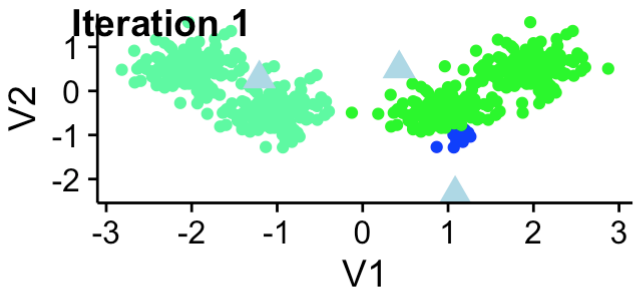


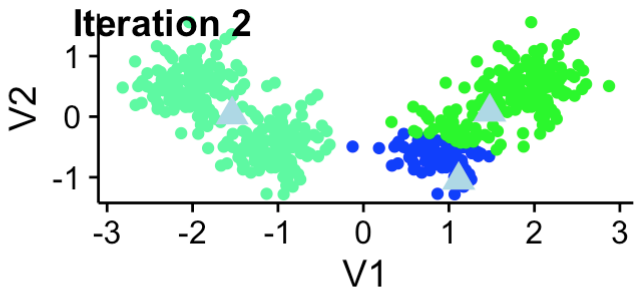


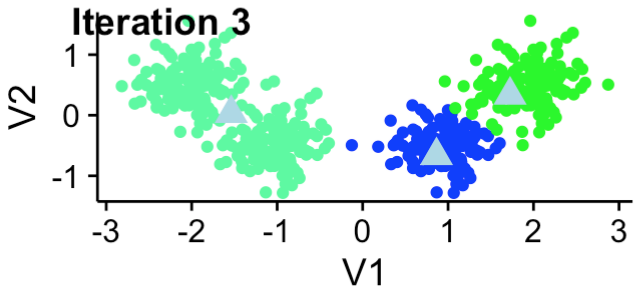


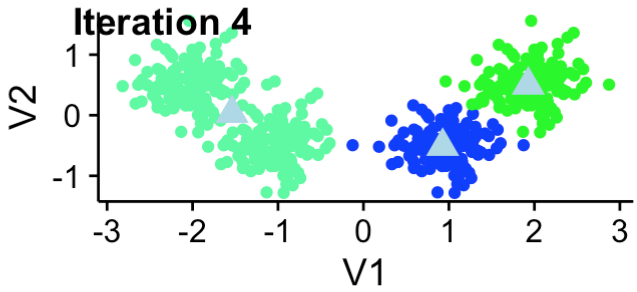


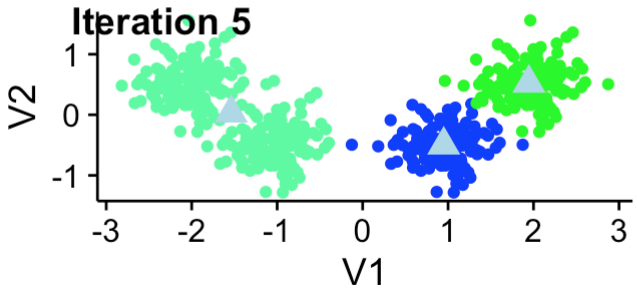

```
plot_cluster_iteration(2)
```



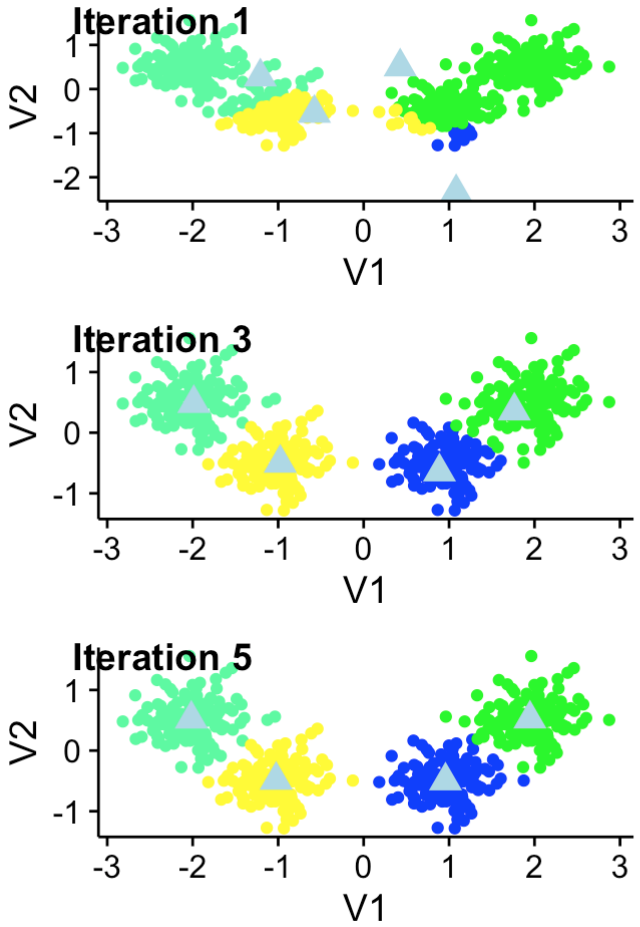


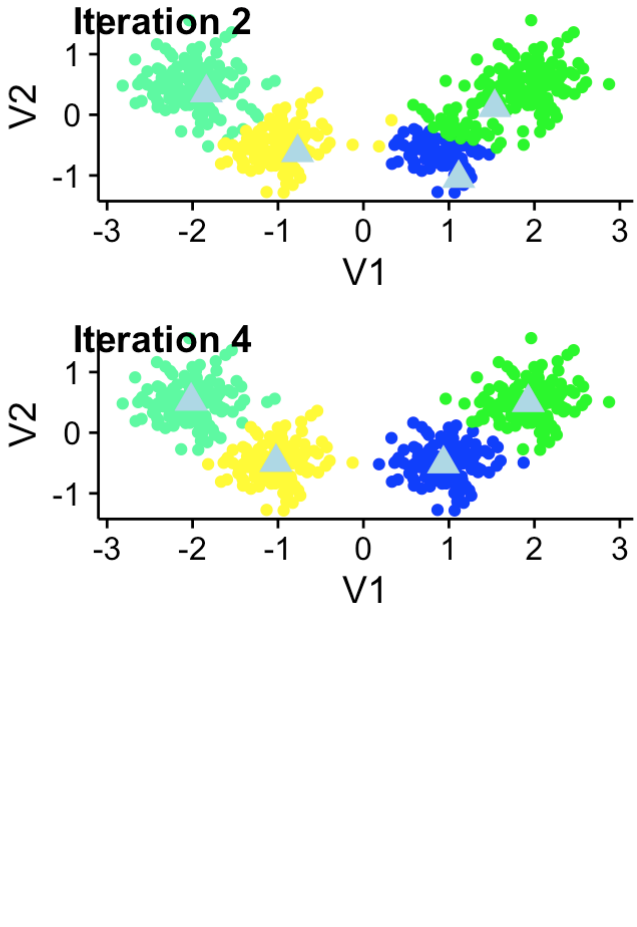




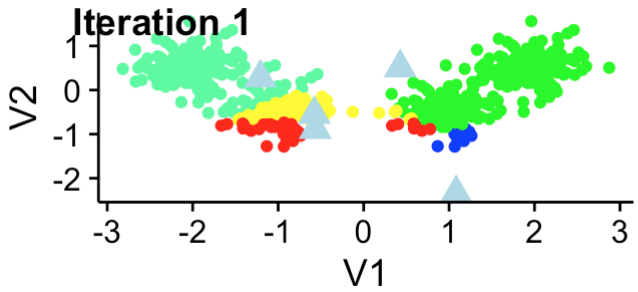


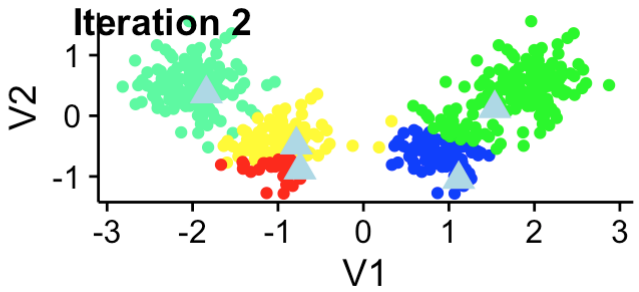
```
plot_cluster_iteration(3)
```

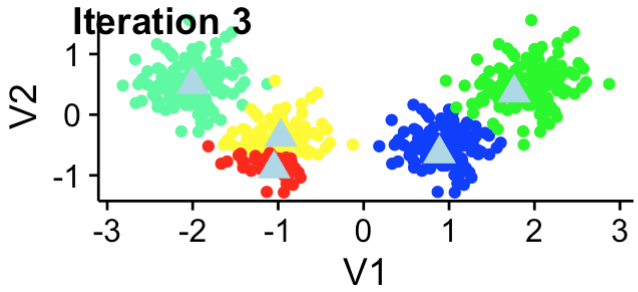


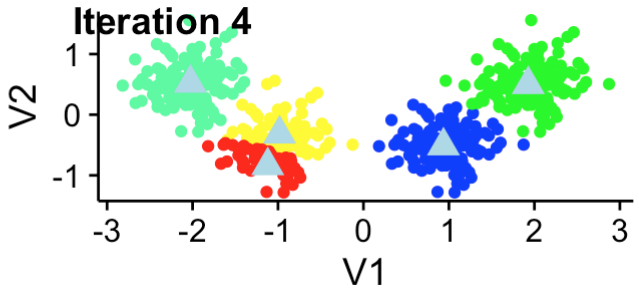


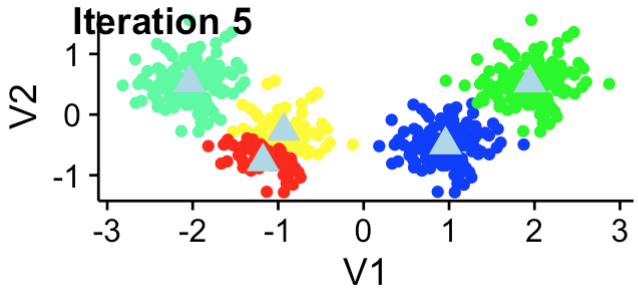

```
plot_cluster_iteration(4)
```



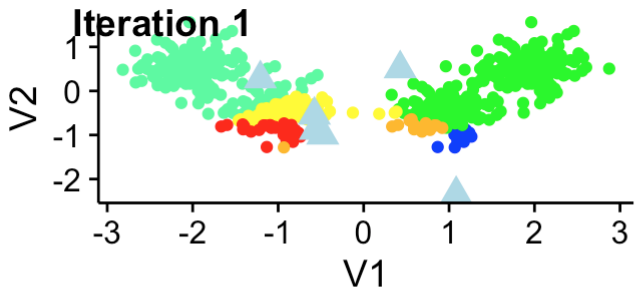


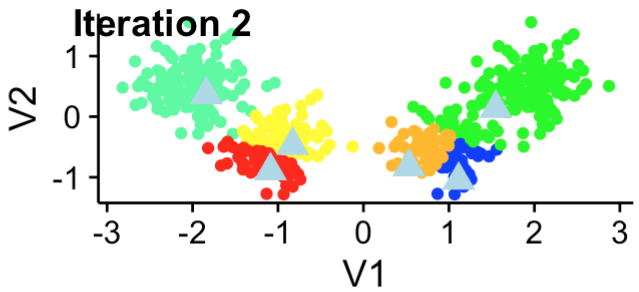


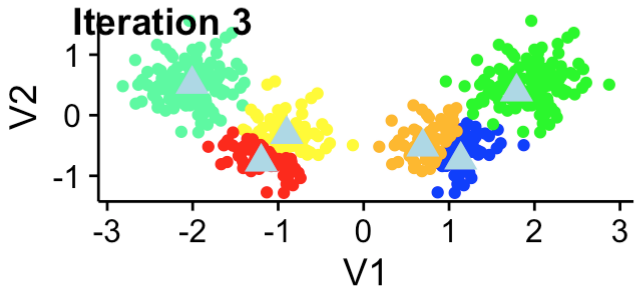


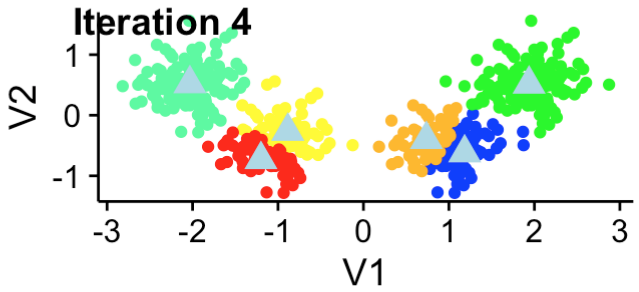


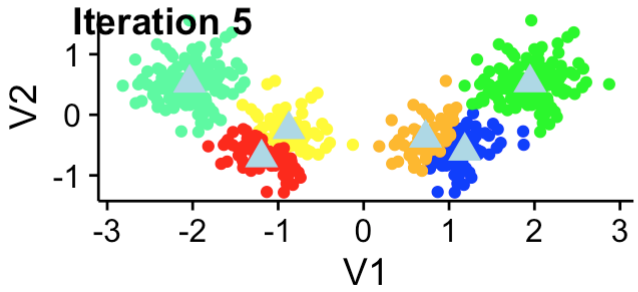

```
plot_cluster_iteration(5)
```



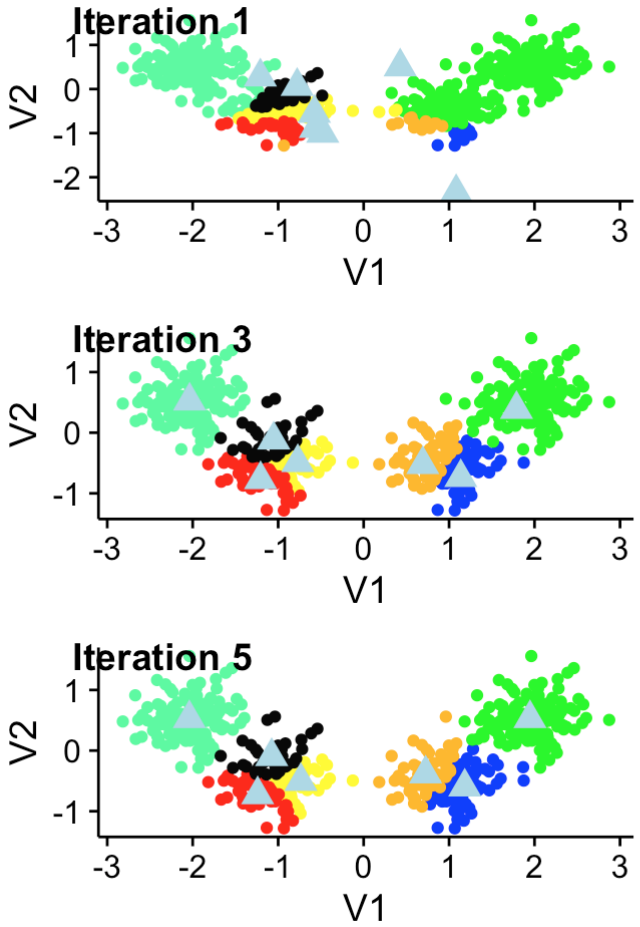




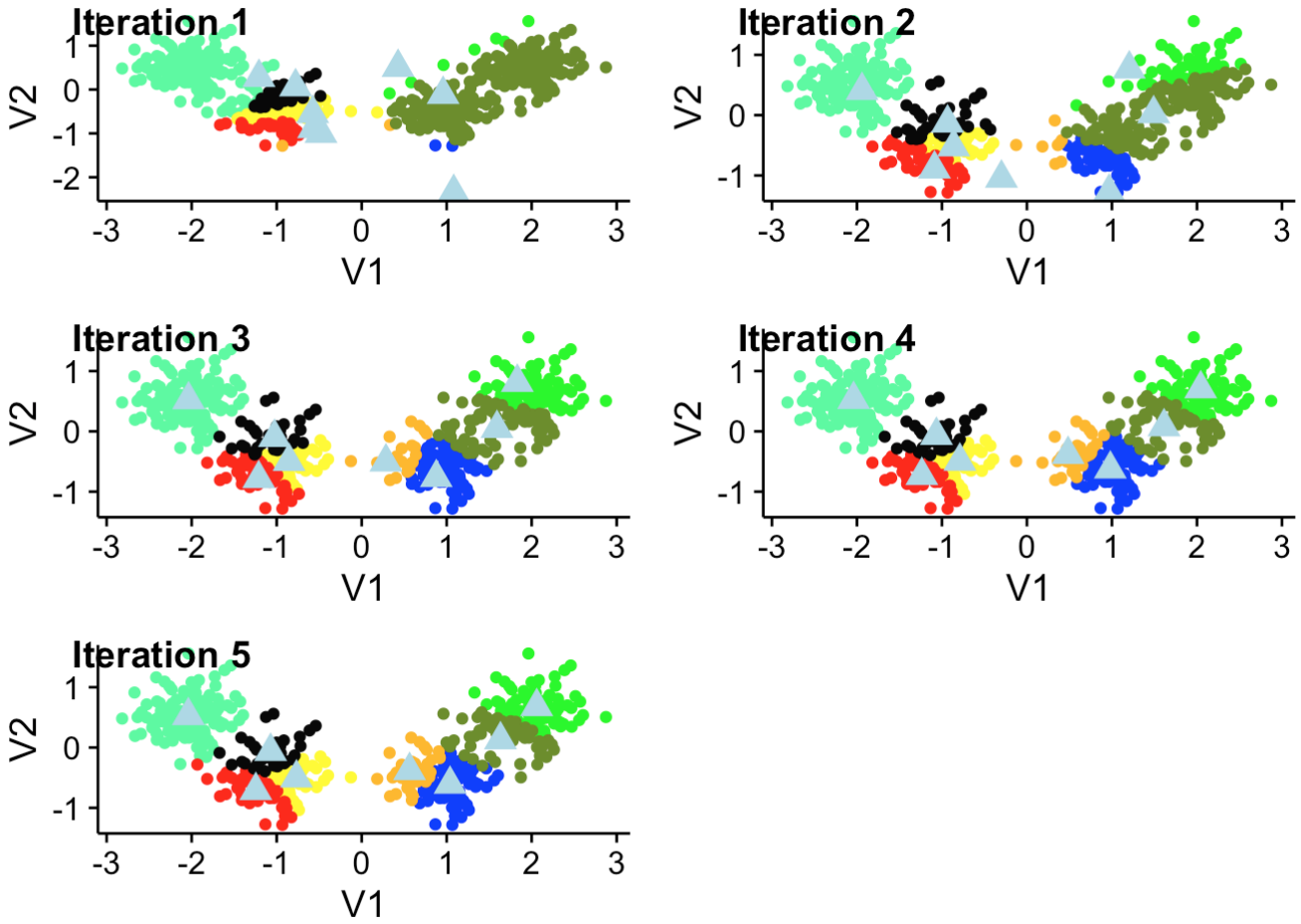




```
plot_cluster_iteration(6)
```

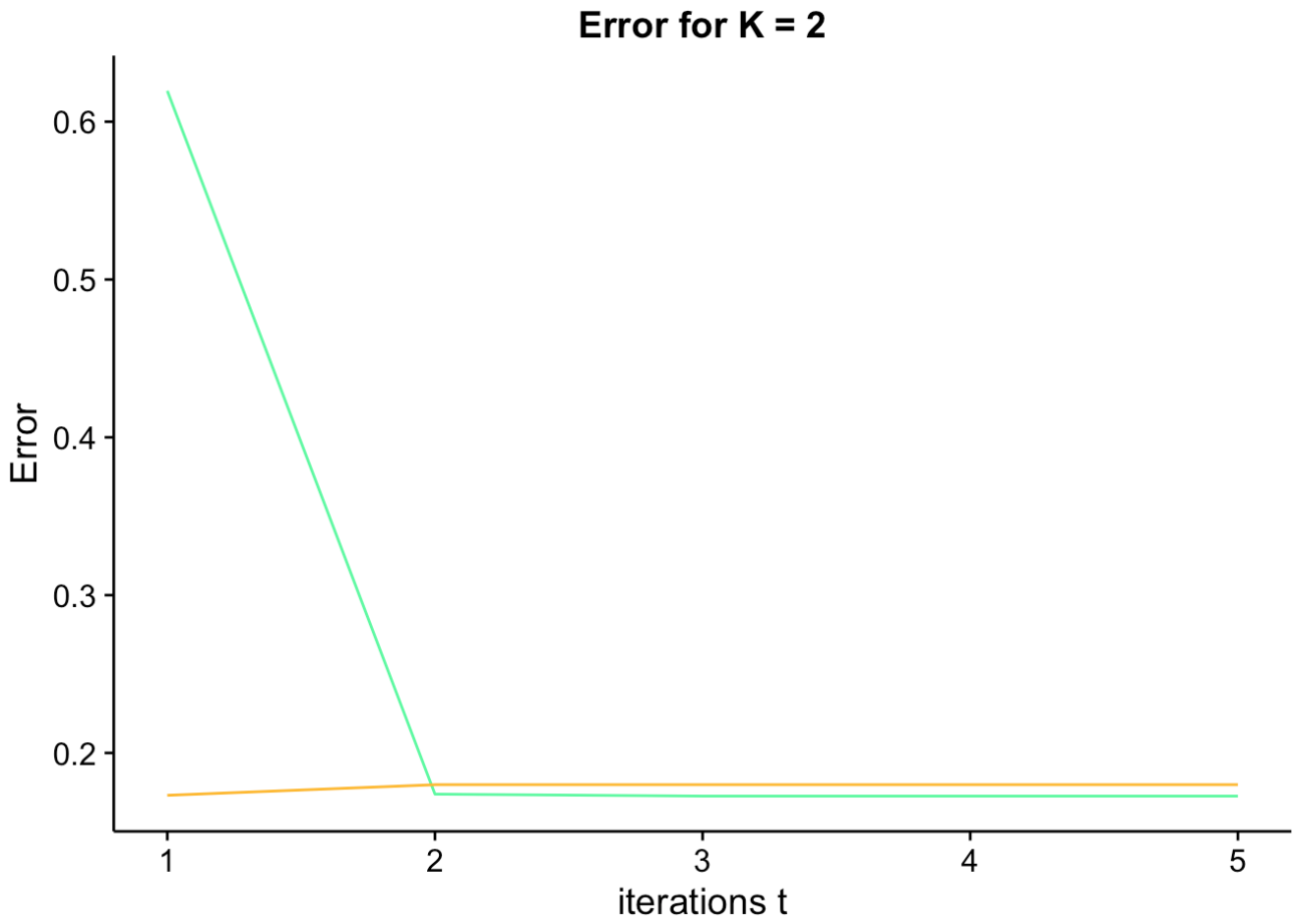


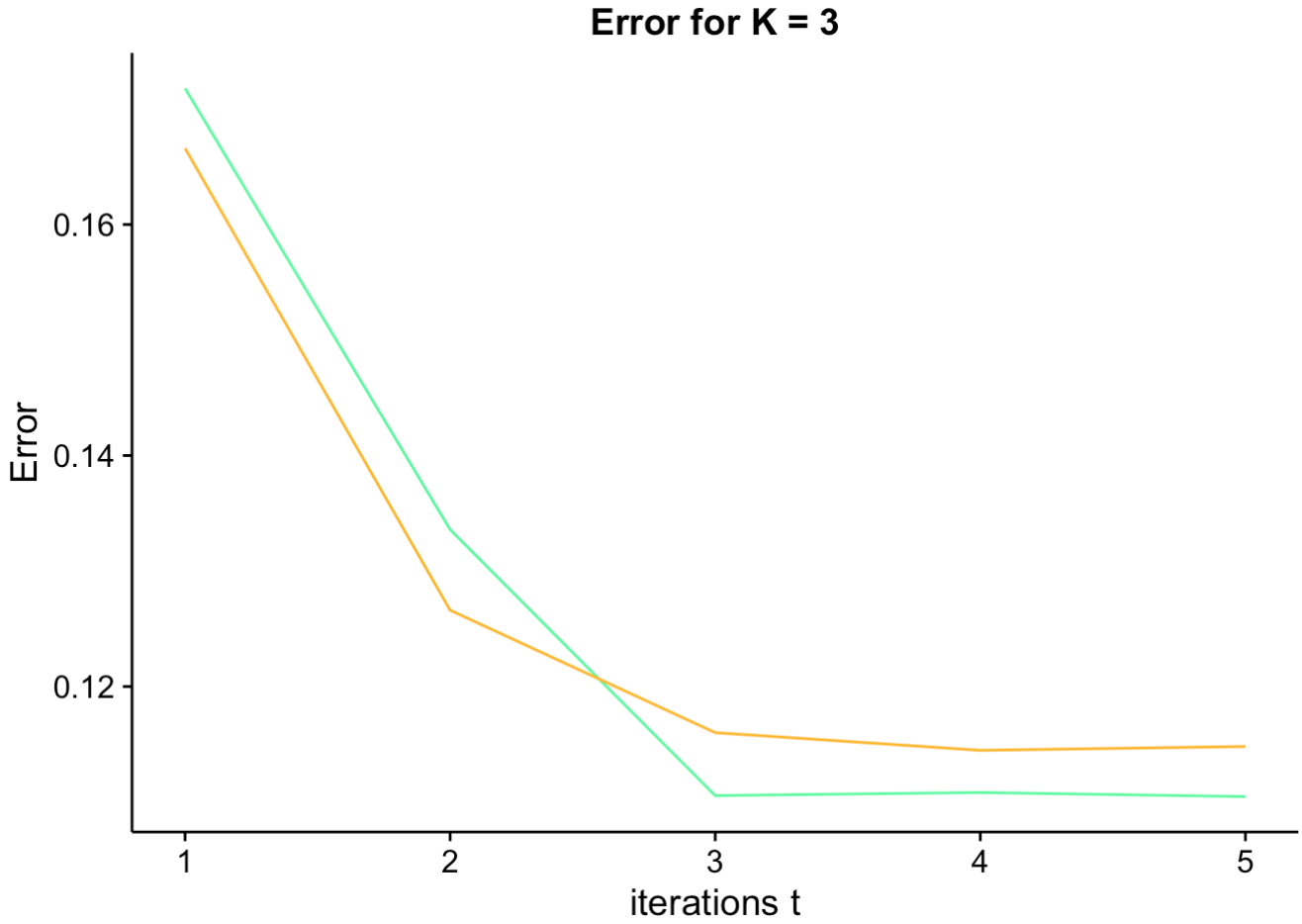

```
plot_cluster_iteration(7)
```



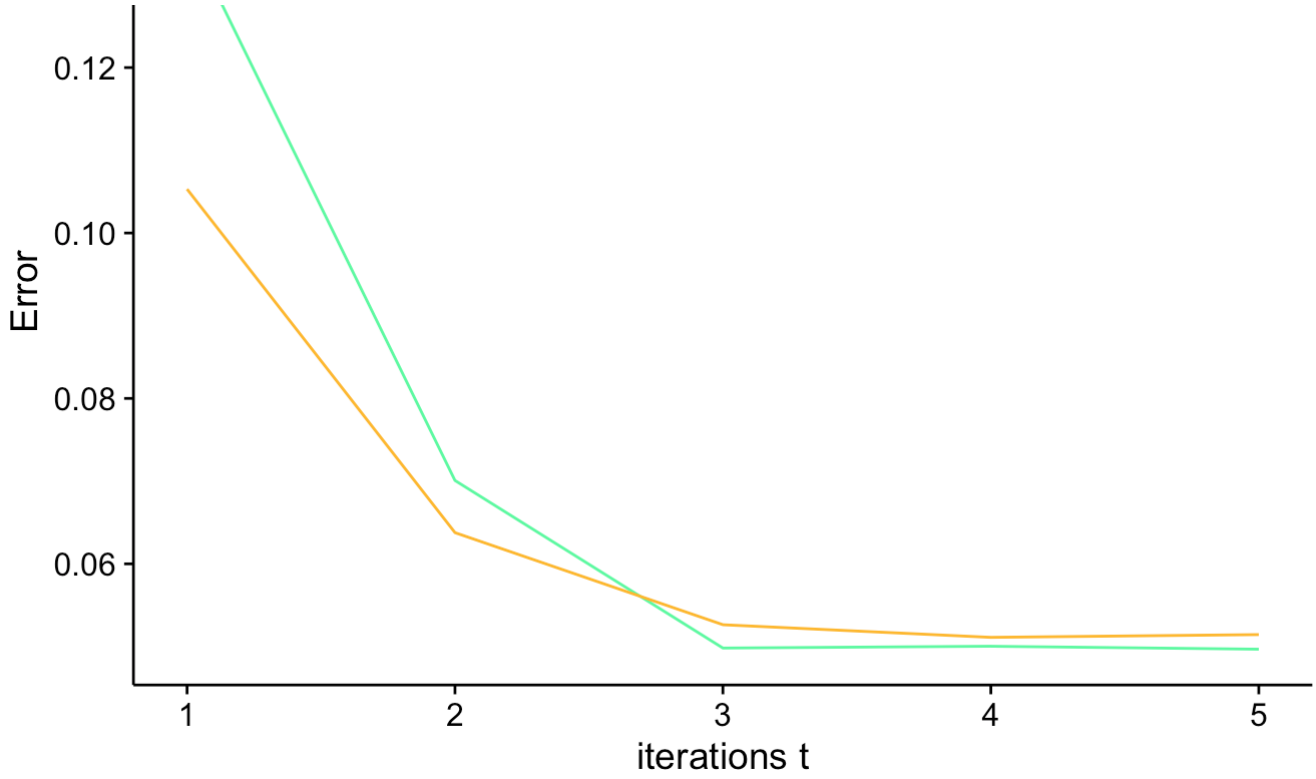
Visualization – (b) Plot the error function E against the iteration number t

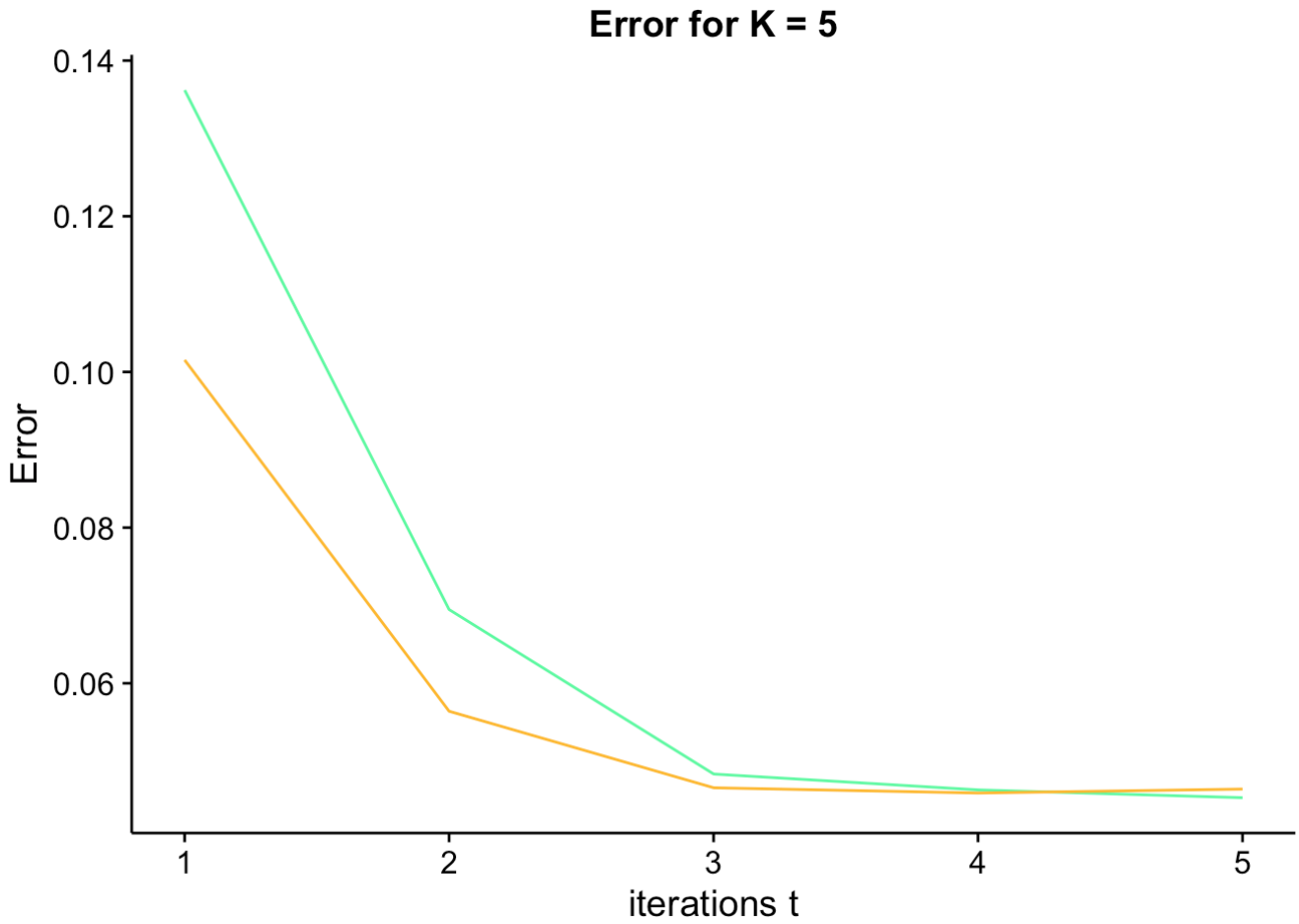
```
xy <- (matrix(0, nrow = tmax, ncol = 2))

for (k in 1:length(cluster_list)) {
    for (i in 1:length(cluster_list[[1]]$error_memory)) {
        xy[i, 1] <- cluster_list[[k]]$error_memory[[i]][, 1]
        xy[i, 2] <- cluster_list[[k]]$error_memory[[i]][, 2]
    }
    xy <- as.data.frame(xy)
    print(ggplot(data = xy) + geom_line(aes(x = 1:tmax, y = V1),
        color = "seagreen1") + geom_line(aes(x = 1:tmax, y = V2),
        color = "darkgoldenrod1") + labs(x = "iterations t",
        y = "Error") + ggtitle(paste0("Error for K = ", k + 1)))
}
```

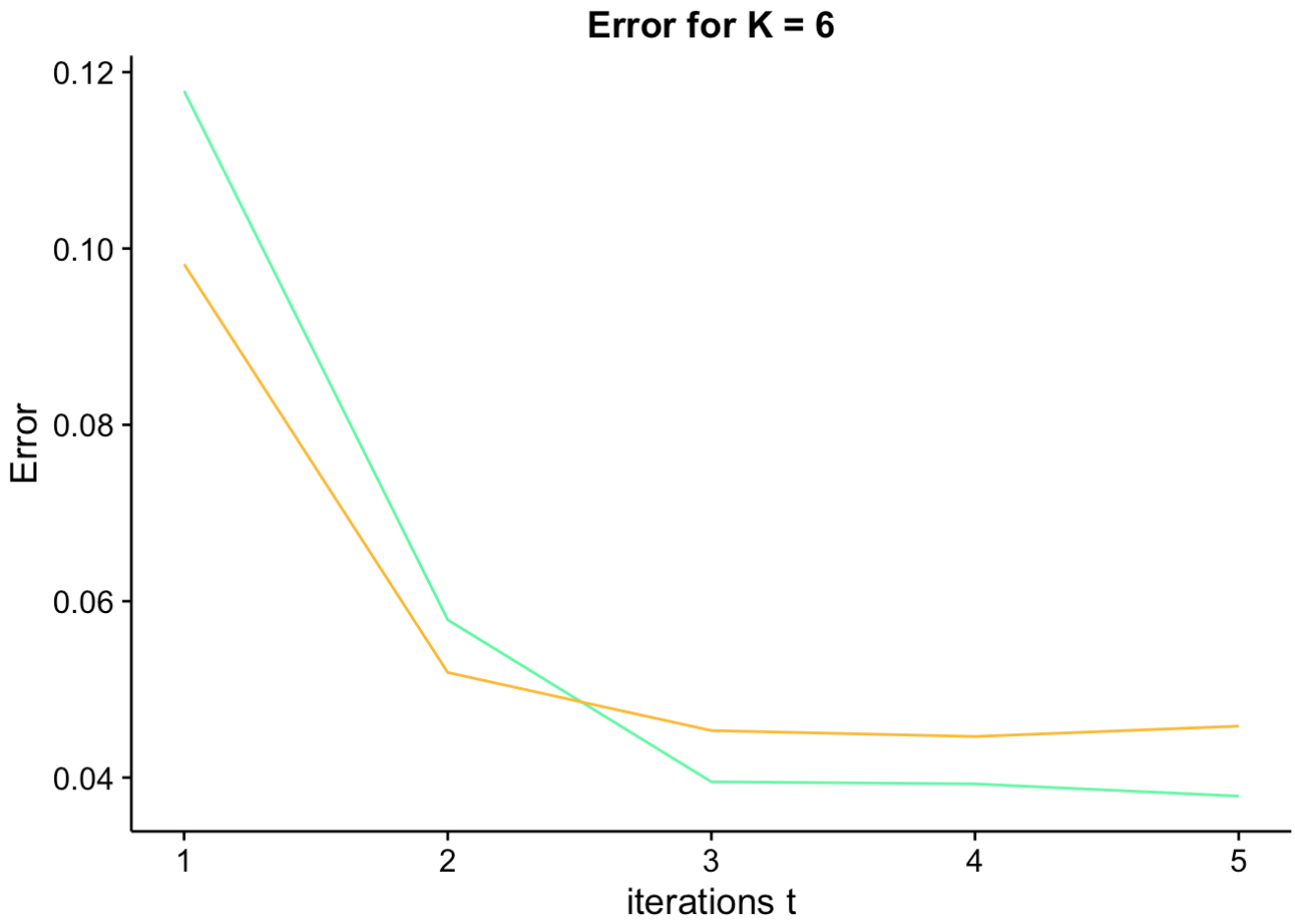

Error for K = 2
Error
0.6
0.5
0.4
0.3
0.2
1
2
3
4
5
iterations t

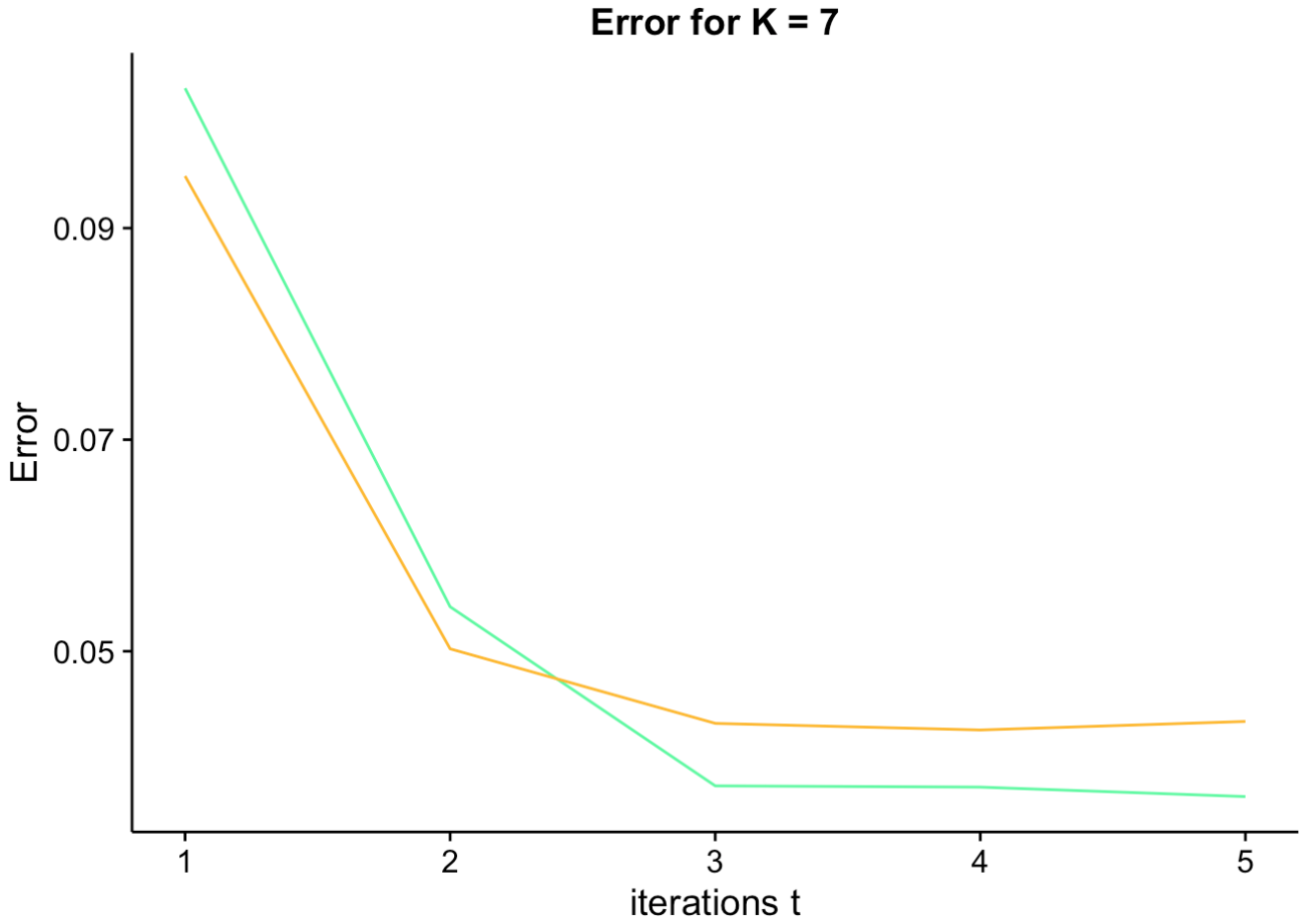

Error for K = 3
Error
0.16
0.14
0.12
1
2
3
4
5
iterations t

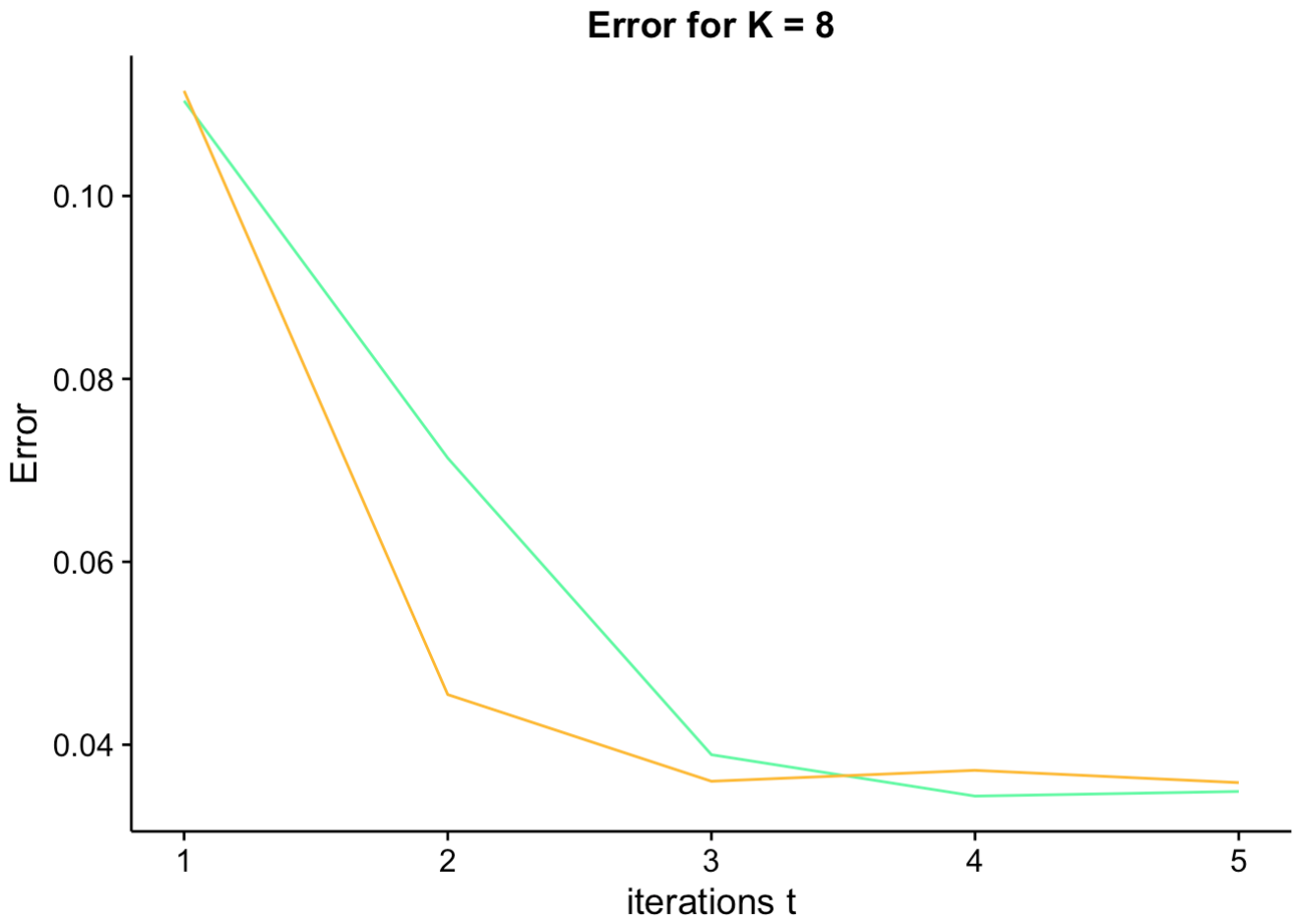

Error for K = 4
0.14


0.12
0.10
0.08
0.06
Error
1
2
3
4
5
iterations t


Error for K = 5
0.14
0.12
0.10
0.08
0.06
Error
1
2
3
4
5
iterations t


Error for K = 6
0.12
0.10
0.08
0.06
0.04
Error
1
2
3
4
5
iterations t


Error for K = 7
0.09
0.07
0.05
Error
1
2
3
4
5
iterations t

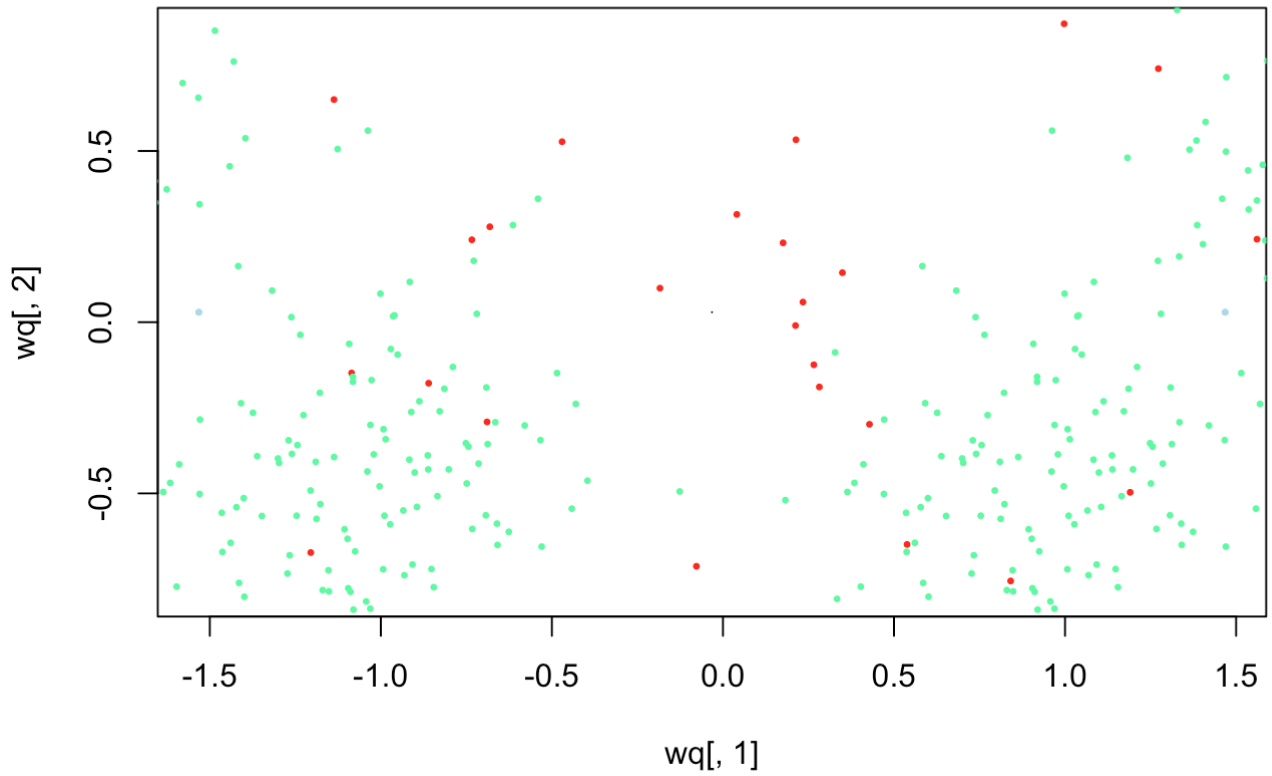


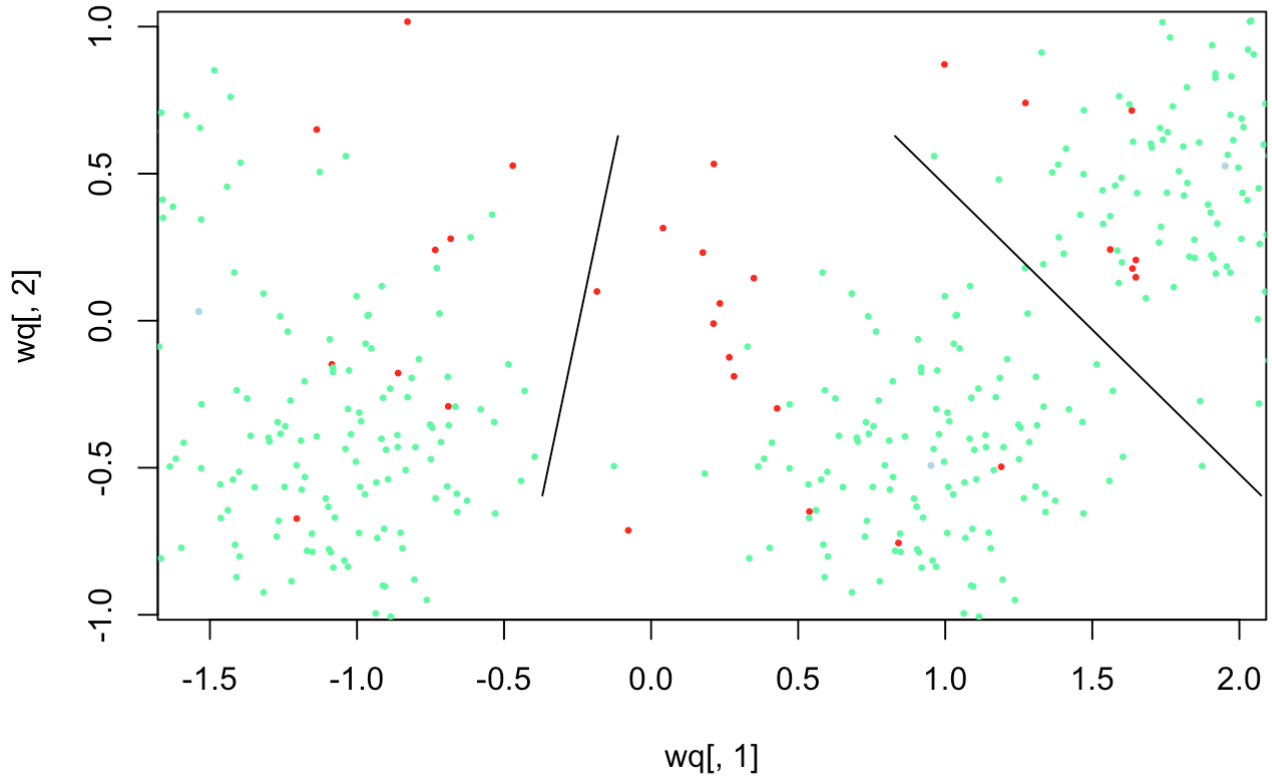

```
# plot_grid(E2,E3,E4,E5,E6,E7,E8, labels=c('K 2', 'K 3', 'K
# 4', 'K 5', 'K 6','K 7','K 8'), ncol = 2, nrow = 3)
```

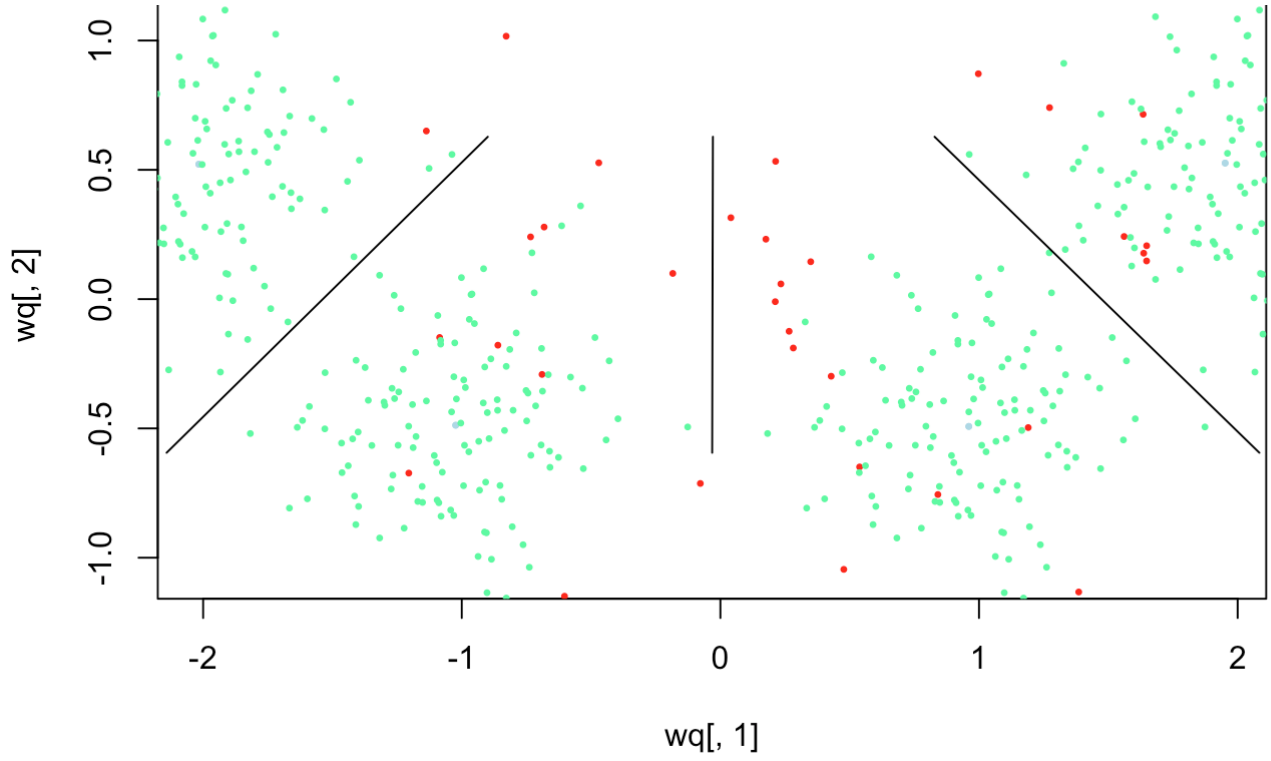
Visualization – (c) Create a plot (Voronoi-Tesselation) to show how the resulting solution assigns dif- ferent regions of input space (e.g. new data points x ∈ R2) to the different clusters.

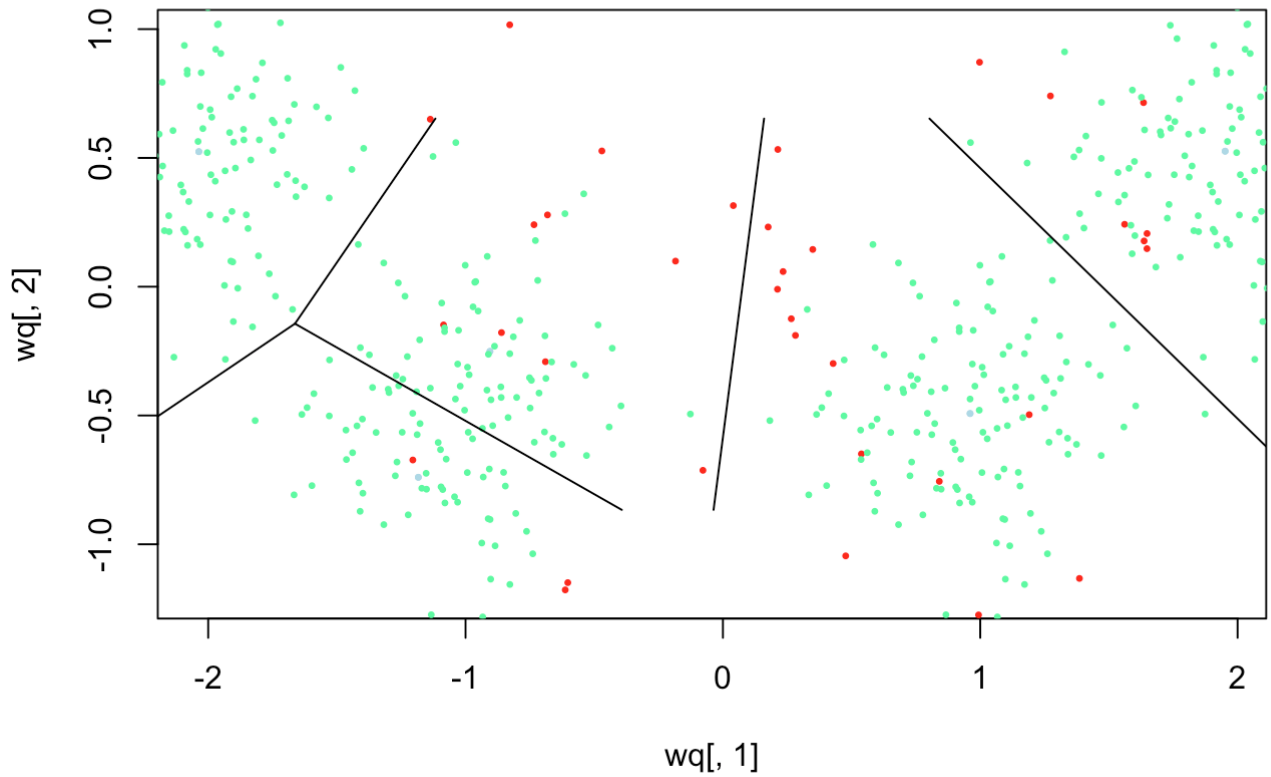
```
set.seed(2342)
X_new_x <- rnorm(50, mean(X), 1)
X_new_y <- rnorm(50, mean(X), 1)

for (k in 1:length(cluster_list)) {
    wq <- (matrix(0, nrow = length(cluster_list[[k]]$wq_memory[[tmax +
        1]]), ncol = 2))

    wq[, 1] <- cluster_list[[k]]$wq_memory[[tmax + 1]][, 1]
    wq[, 2] <- cluster_list[[k]]$wq_memory[[tmax + 1]][, 2]

    vtess <- deldir(wq[, 1], wq[, 2])
    plot(wq[, 1], wq[, 2], type = "n", asp = 1)
    points(wq[, 1], wq[, 2], pch = 20, col = "lightblue", cex = 0.5)
    points(X_new_x, X_new_y, pch = 20, col = "red", cex = 0.5)
    points(X[, 1], X[, 2], pch = 20, col = "seagreen1", cex = 0.5)
    plot(vtess, wlines = "tess", wpoints = "none", number = FALSE,
        add = TRUE, lty = 1)

}
```

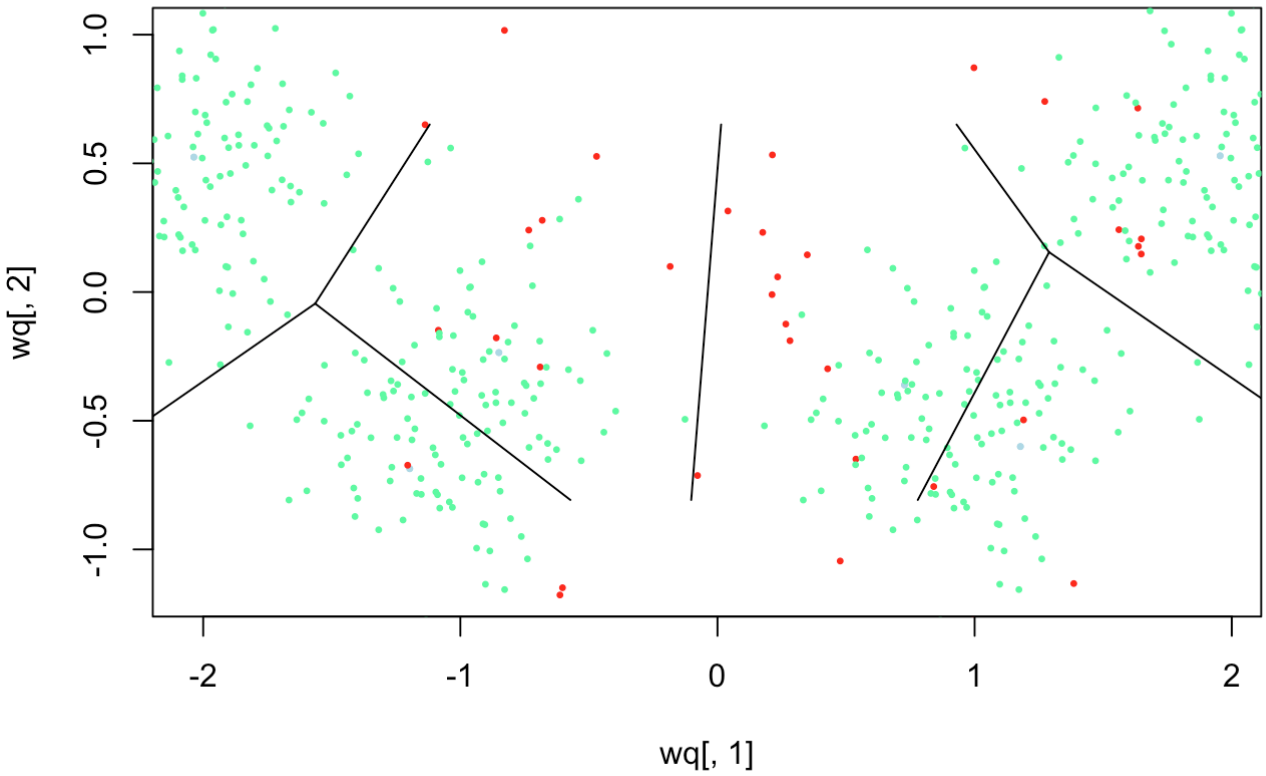
```
##
##      PLEASE NOTE:  The components "delsgs" and "summary" of the
##  object returned by deldir() are now DATA FRAMES rather than
##  matrices (as they were prior to release 0.0-18).
##  See help("deldir").
##
##      PLEASE NOTE: The process that deldir() uses for determining
##  duplicated points has changed from that used in version
##  0.0-9 of this package (and previously). See help("deldir").
```

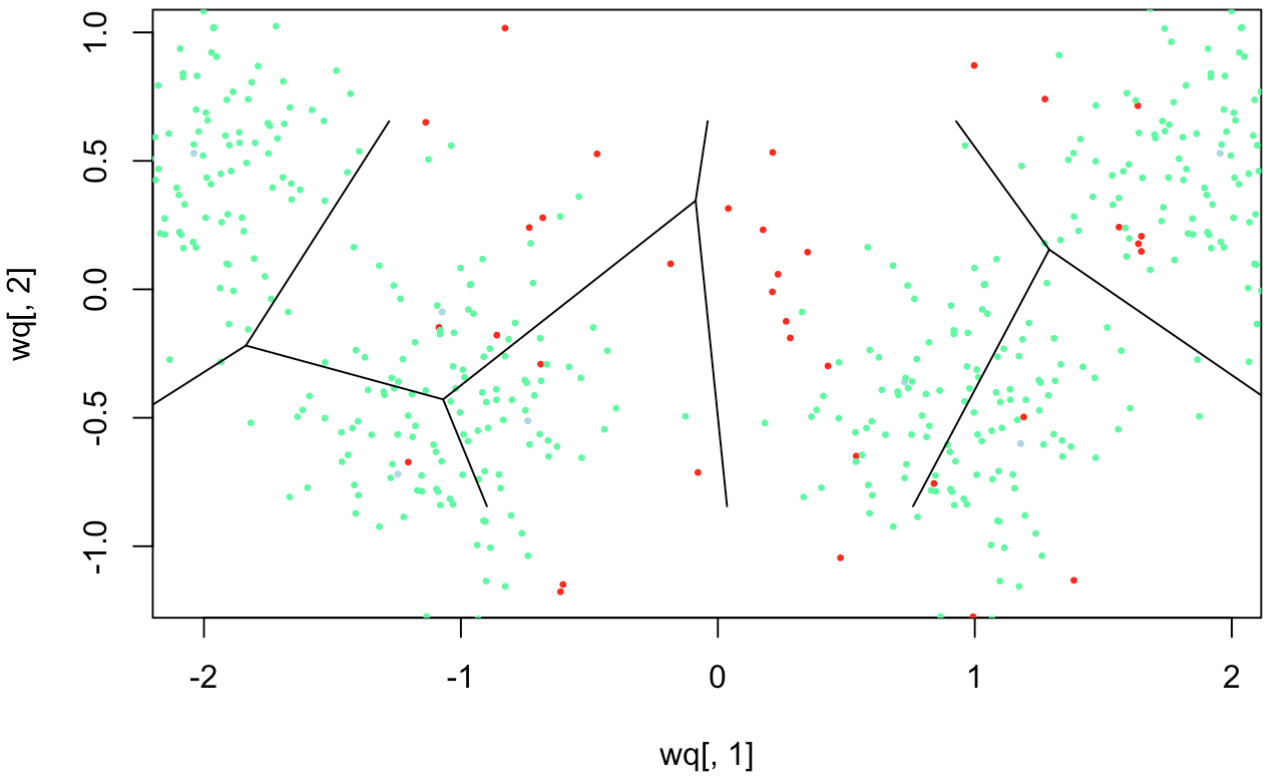

wq[, 2]
0.5
0.0
-0.5
-1.5
-1.0
-0.5
0.0
0.5
1.0
1.5
wq[, 1]

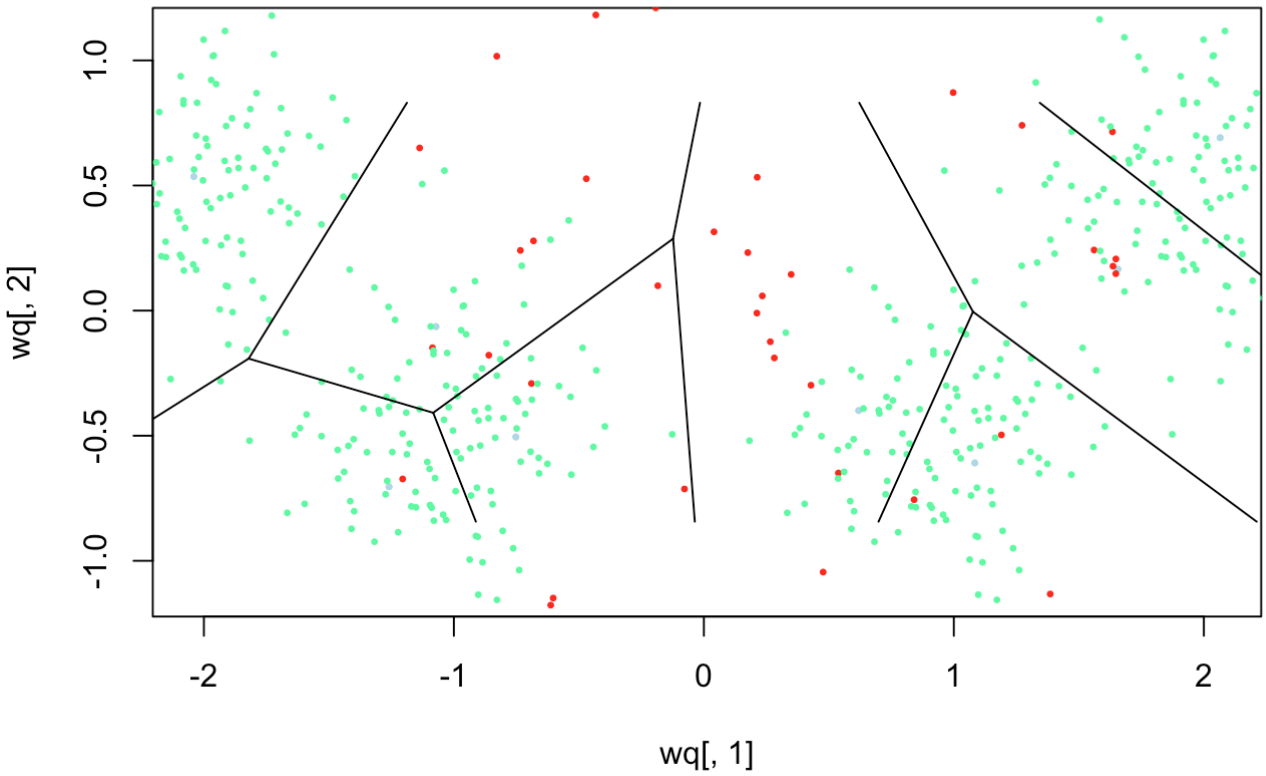

wq[, 2]
1.0
0.5
0.0
-0.5
-1.0
-1.5
-1.0
-0.5
0.0
0.5
1.0
1.5
2.0
wq[, 1]


wq[, 2]
1.0
0.5
0.0
-0.5
-1.0
-2
-1
0
1
2
wq[, 1]


wq[, 2]
1.0
0.5
0.0
-0.5
-1.0
-2
-1
0
1
2
wq[, 1]

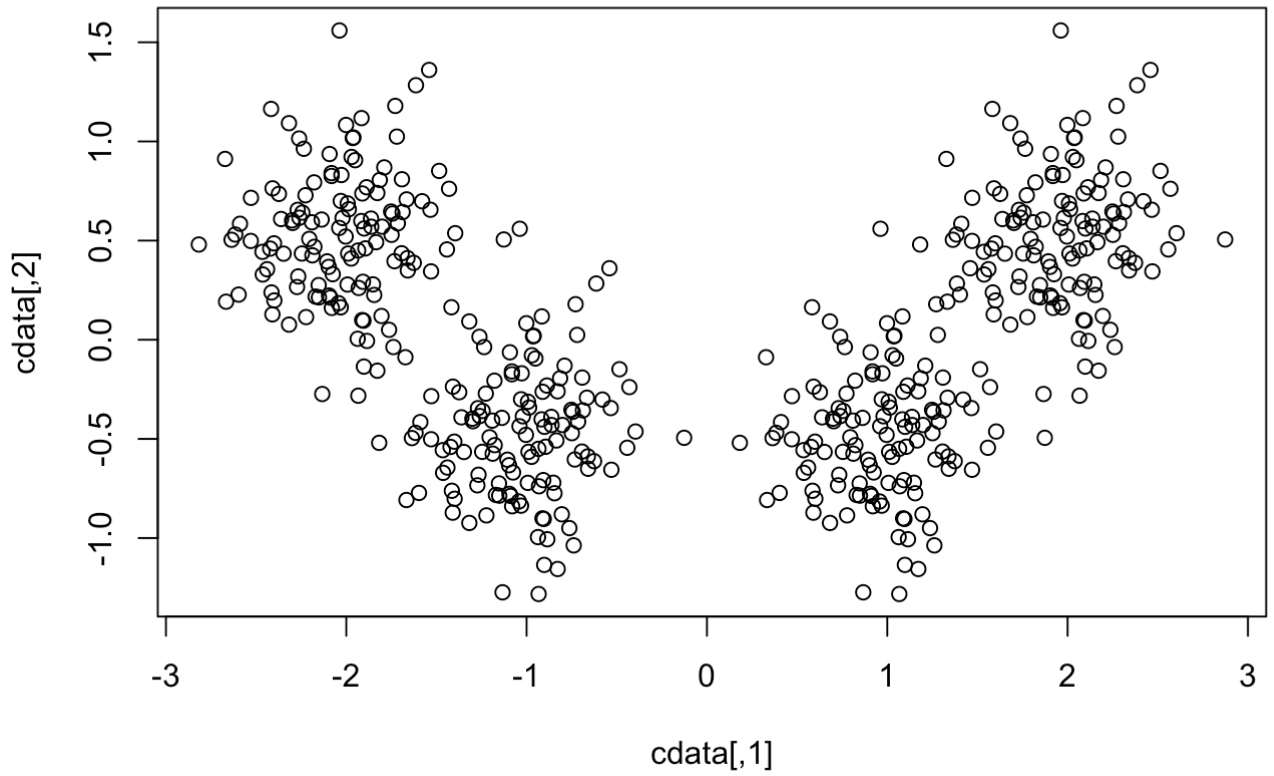

wq[, 2]
1.0
0.5
0.0
-0.5
-1.0
-2
-1
0
1
2
wq[, 1]


wq[, 2]
1.0
0.5
0.0
-0.5
-1.0
-2
-1
0
1
2
wq[, 1]



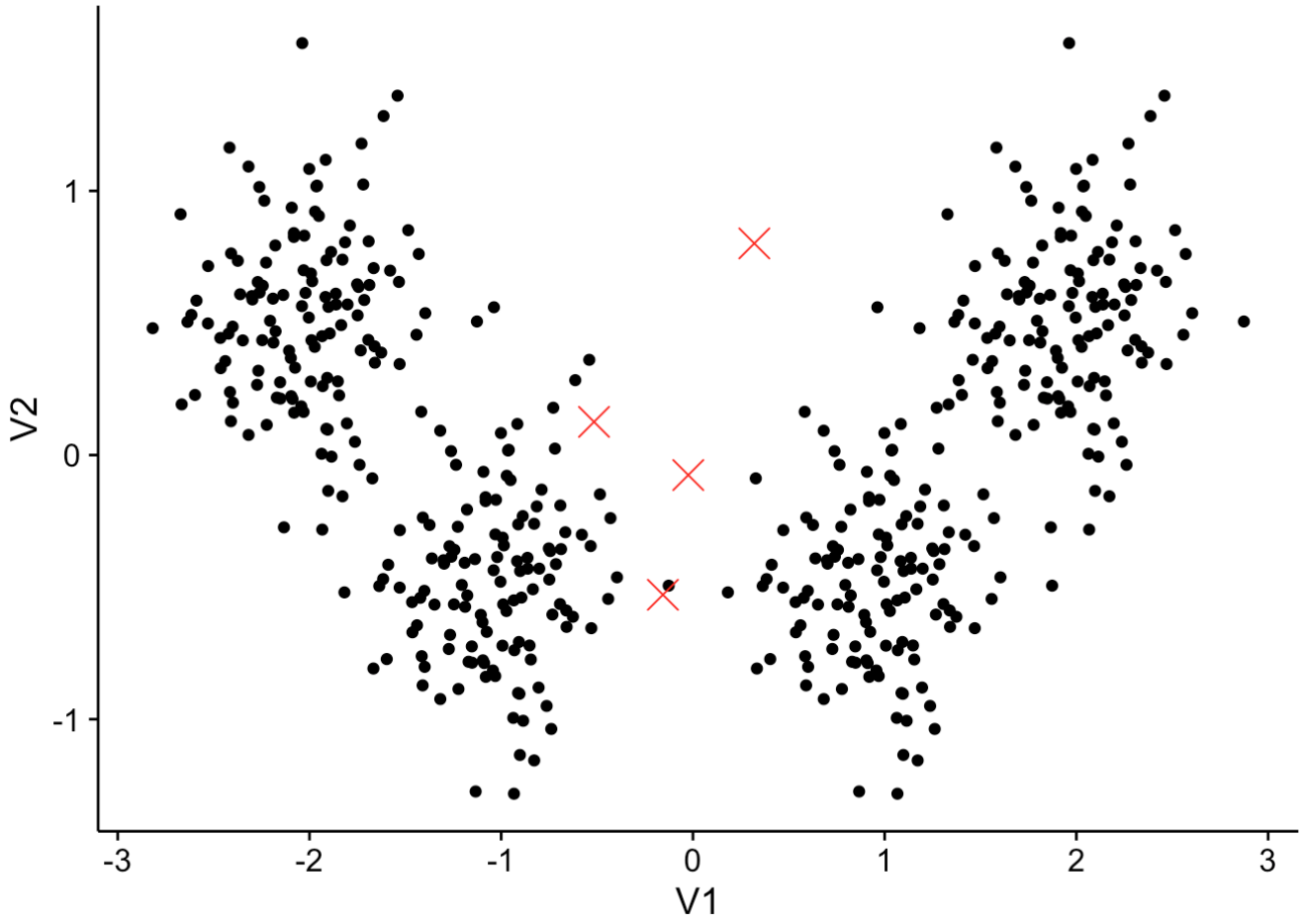
Exercise 9.2: Online k-means Clustering

```
# custom functions eucledian distance
e_distance <- function(p1, p2) {
    p1 <- t(as.matrix(p1))
    p2 <- t(as.matrix(p2))
    return(sqrt((p1[1, 1] - p2[1, 1])^2 + (p1[1, 2] - p2[1, 2])^2))
}

## read data
cdata <- read.csv("cluster.DAT", sep = "", header = FALSE)
cdata <- t(cdata)
plot(cdata)
```



Initialization:

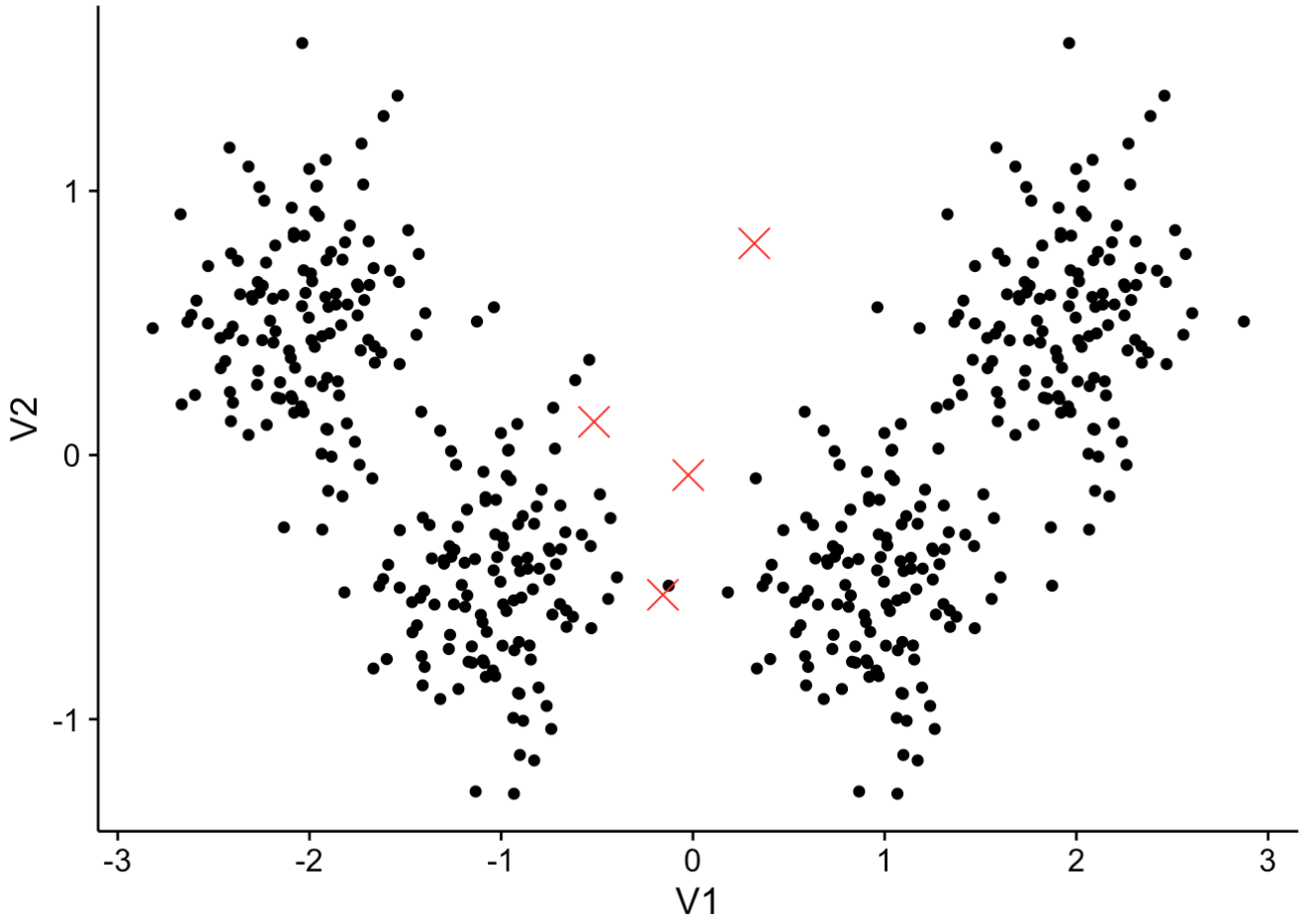
```
# col means
m1 <- mean(cdata[, 1])
m2 <- mean(cdata[, 2])

# initialize four prototypes initialize prototype matrix
W <- matrix(nrow = 4, ncol = 2)
set.seed(523)
W[, 1] <- rnorm(4, mean = m1, sd = 0.5)
W[, 2] <- rnorm(4, mean = m2, sd = 0.5)

# show initial prototypes
ggplot() + geom_point(data = as.data.frame(cdata), aes(x = V1,
    y = V2)) + geom_point(data = as.data.frame(W), aes(x = V1,
    y = V2), colour = "red", shape = 4, size = 5)
```




```
# initial learning step, max iterations
e0 <- 0.05
tmax <- 500
```

Optimization:

```
# col means
m1 <- mean(cdata[, 1])
m2 <- mean(cdata[, 2])

# initialize four prototypes initialize prototype matrix
W <- matrix(nrow = 4, ncol = 2)
set.seed(523)
W[, 1] <- rnorm(4, mean = m1, sd = 0.5)
W[, 2] <- rnorm(4, mean = m2, sd = 0.5)

# show initial prototypes
ggplot() + geom_point(data = as.data.frame(cdata), aes(x = V1,
    y = V2)) + geom_point(data = as.data.frame(W), aes(x = V1,
    y = V2), colour = "red", shape = 4, size = 5)
```




```
# initial learning step, max iterations
e0 <- 0.05
tmax <- 500
```

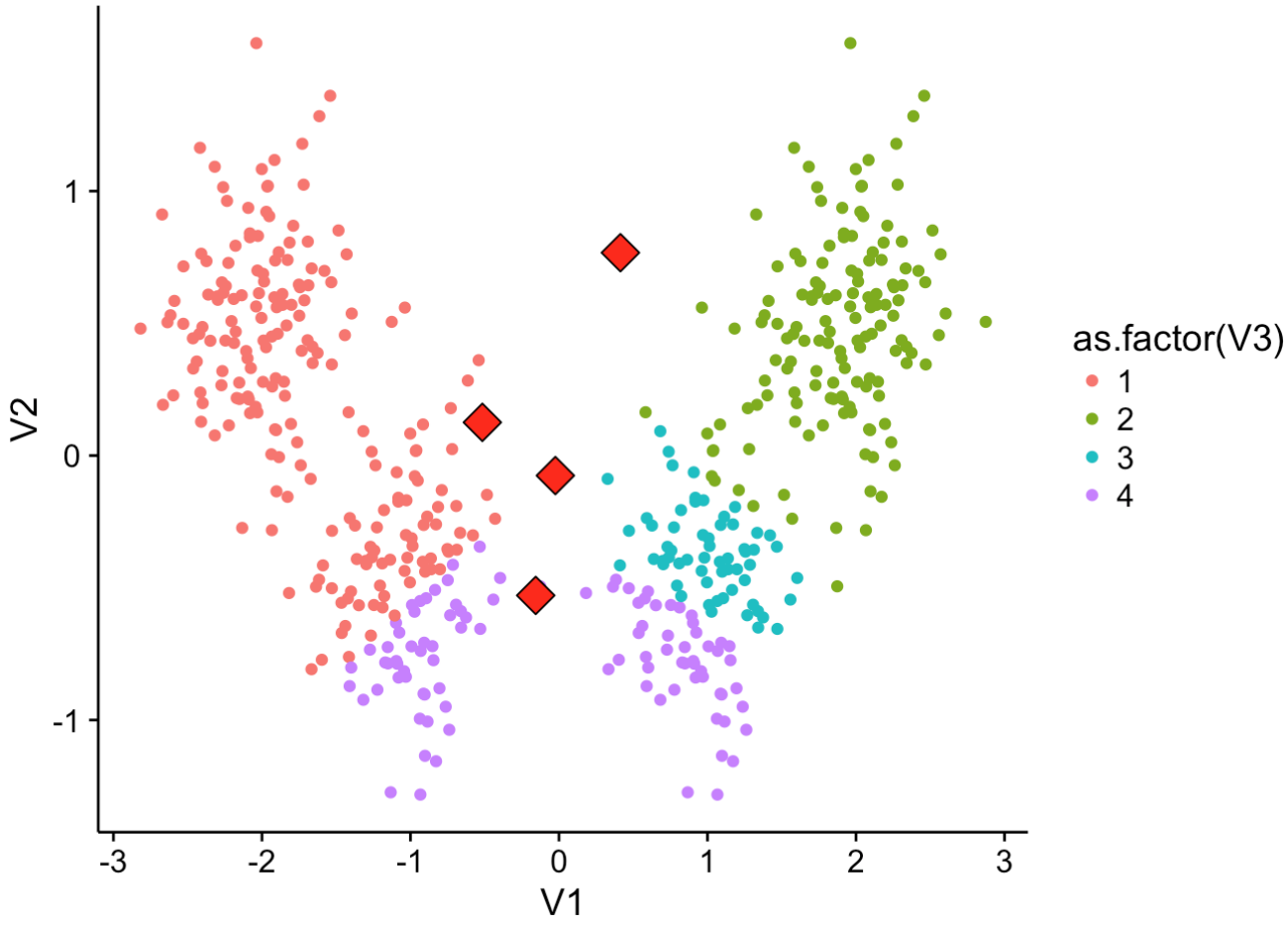
Clustering:

```
### FOR VISUALISATION PART ###### 3D-Arrays saves Prototypes at
### given times
W_evo <- array(dim = c(4, 6, 2))
# Given Times to save Prototypes: First, Final and four in
# between (83, 166, 250, 333)
plotcases <- c(1, 83, 166, 250, 333, 500)
# Assignment of the Data points at the given times
point_assignment <- matrix(nrow = 500, ncol = 6)
### END OF VISUALISATION PART ###

tau <- 0.75
k <- 4  #no of clusters

# prototype matrix: store assignment of prototypes for every
# data point here.  col 1 contains index of x, col 2 contains
# assigned prototype
p_assignment <- matrix(nrow = 500, ncol = 2)
p_assignment[, 1] <- 1:500
p_assignment[, 2] <- 0

# init of deltaw, e_old
deltaw <- 0
e_old <- e0

for (i in 1:500) {
    x <- cdata[i, ]  # chose data point i

    proto <- 0  #assignment variable for prototype

    # decide which e to use
    if (i <= (tmax/4)) {
        e_t <- e0
    } else {
        e_t <- e_old
    }

    # (re-)init distance vector
    distances <- matrix(nrow = k, ncol = 1)

    # calculate distance to every prototype
    for (j in 1:k) {
        distances[j, ] <- e_distance(x, W[j, ])
    }

    # search for prototype with minimizing eucledian distance and
    # store in matrix
    proto <- which.min(distances)
    p_assignment[i, 2] <- proto

    # calculate deltaw
    deltaw <- e_t * (x - W[proto, ])

    # recalculate prototype
    W[proto, ] <- W[proto, ] + deltaw

    e_old <- e_t
```

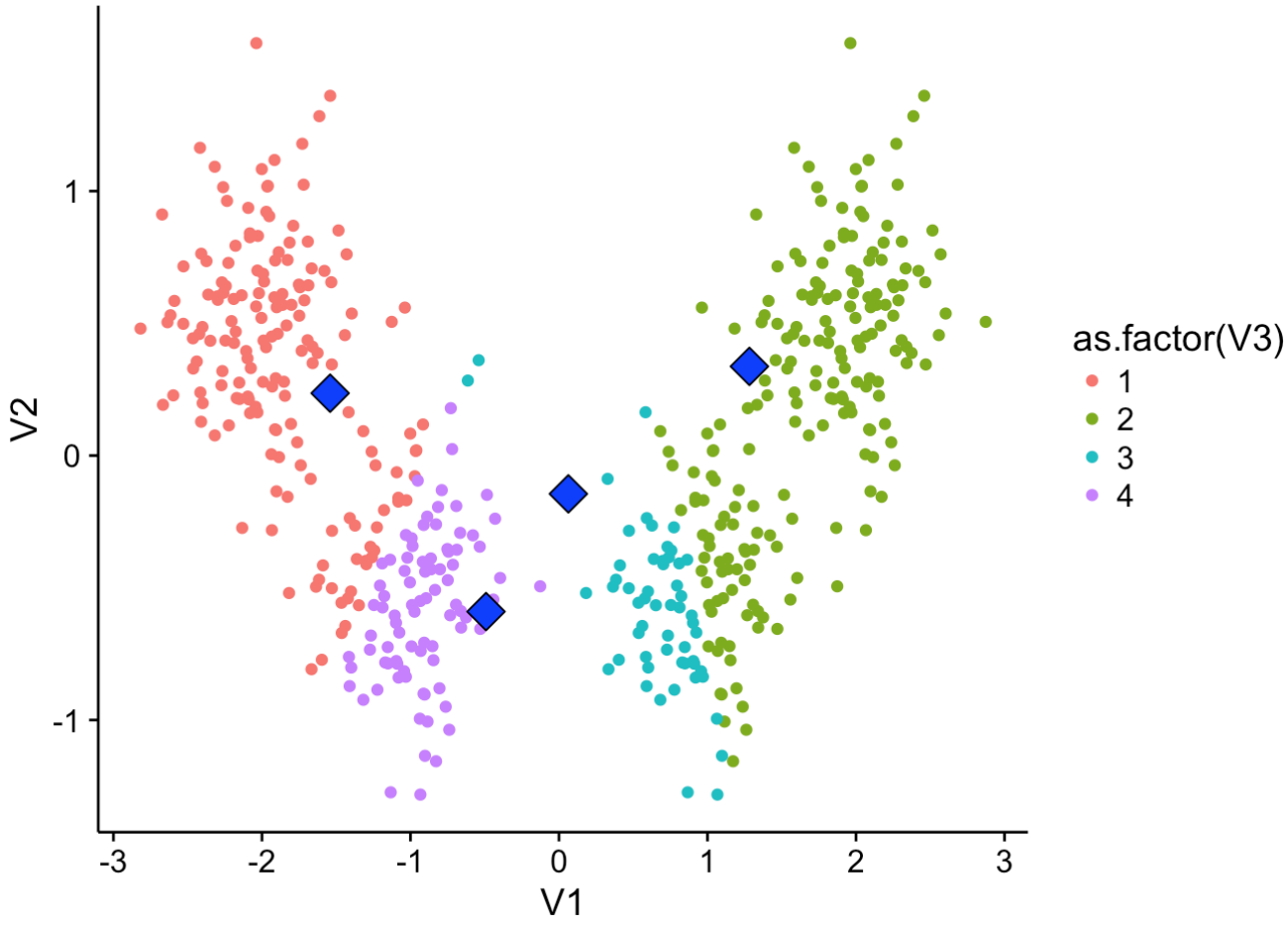
```
    # FOR VISUALISATION if iteration is to save for plotting,
    # save the Prototype of that iteration in the W_evo Array
    if (i %in% plotcases) {
        plotno <- which(plotcases == i)
        for (l in 1:4) {
            W_evo[l, plotno, ] <- W[l, ]
        }
        # assign every data point to its nearest prototype
        for (o in 1:500) {
            # calculate distance to every prototype
            dist <- matrix(nrow = 4)  # hier ist irgendwo ein falscher index -> wird
 immer zu 4 assignt
            for (q in 1:k) {
                dist[q, ] <- e_distance(cdata[o, ], W[q, ])
            }
            point_assignment[o, plotno] <- which.min(dist)
        }
    }


}
```

Visualization

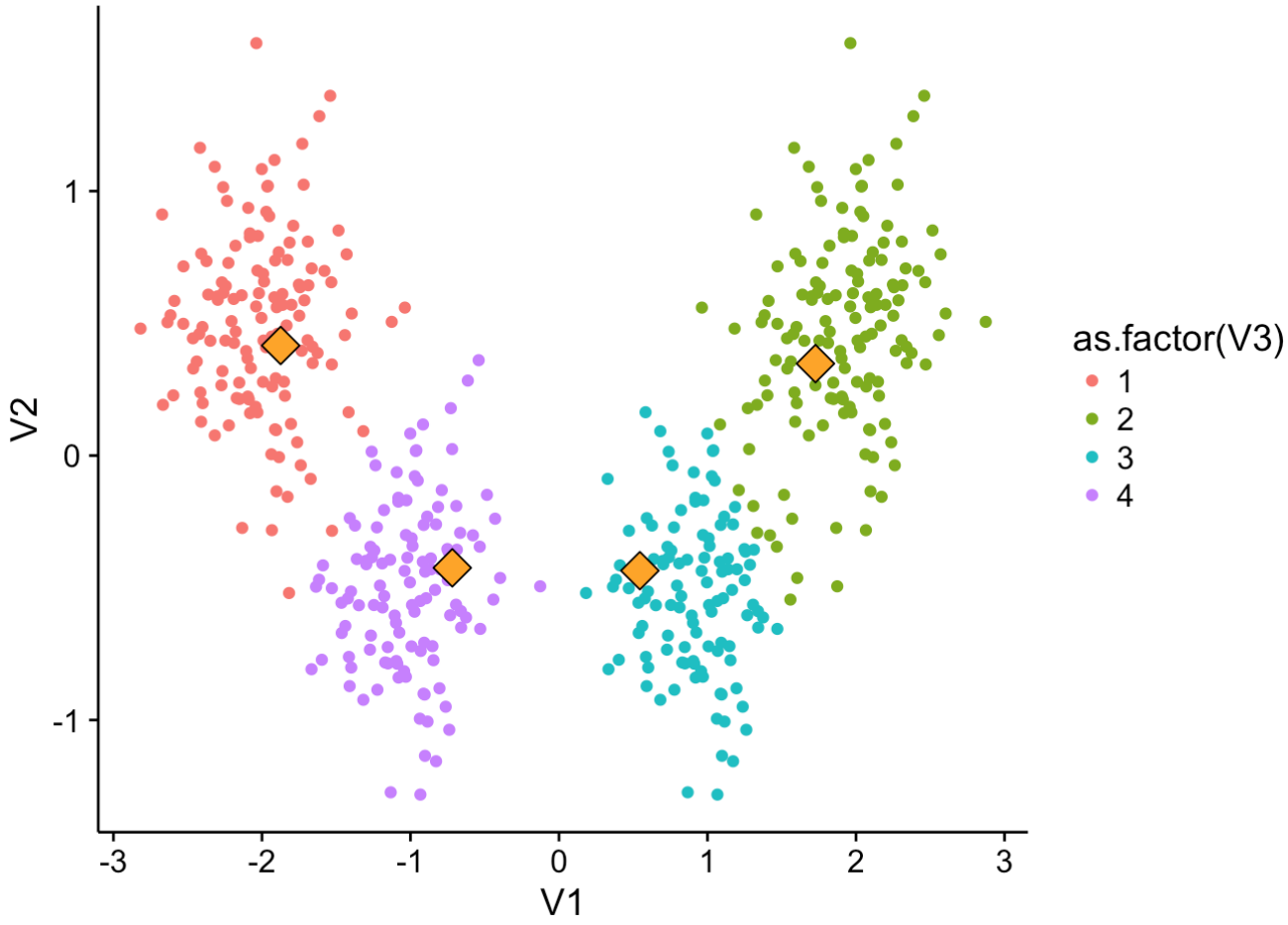
```
# plot 1: First iteration
WT1 <- W_evo[, 1, ]
ggplot() + geom_point(data = as.data.frame(cbind(cdata, point_assignment[,
    1])), aes(x = V1, y = V2, color = as.factor(V3))) + geom_point(data = as.data.fra
me(WT1),
    aes(x = V1, y = V2), fill = "red", shape = 23, size = 5)  #+
```



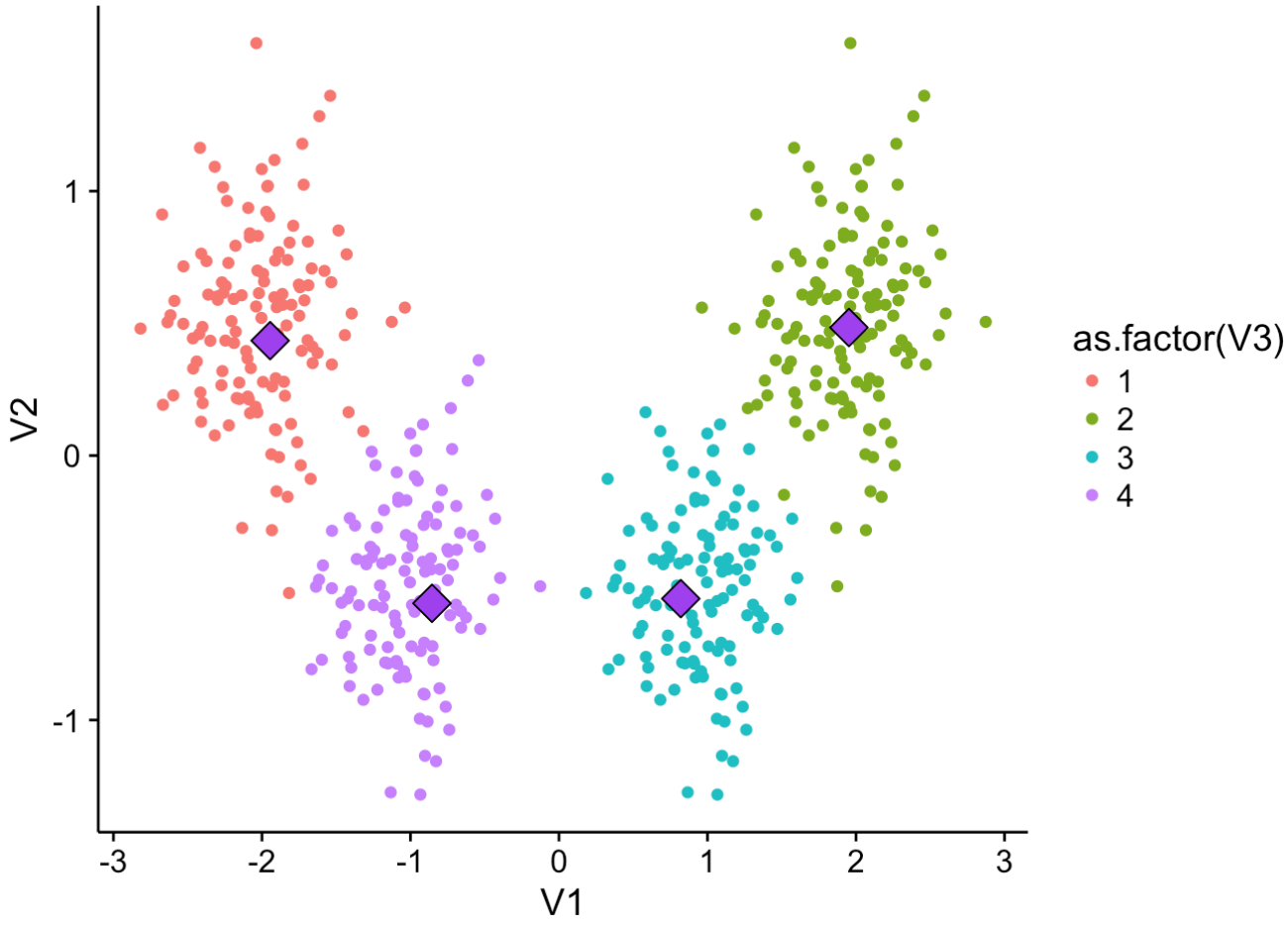

```
# plot 2: Iteration 83
WT2 <- W_evo[, 2, ]
ggplot() + geom_point(data = as.data.frame(cbind(cdata, point_assignment[,
    2])), aes(x = V1, y = V2, color = as.factor(V3))) + geom_point(data = as.data.fra
me(WT2),
    aes(x = V1, y = V2), fill = "blue", shape = 23, size = 5)
```



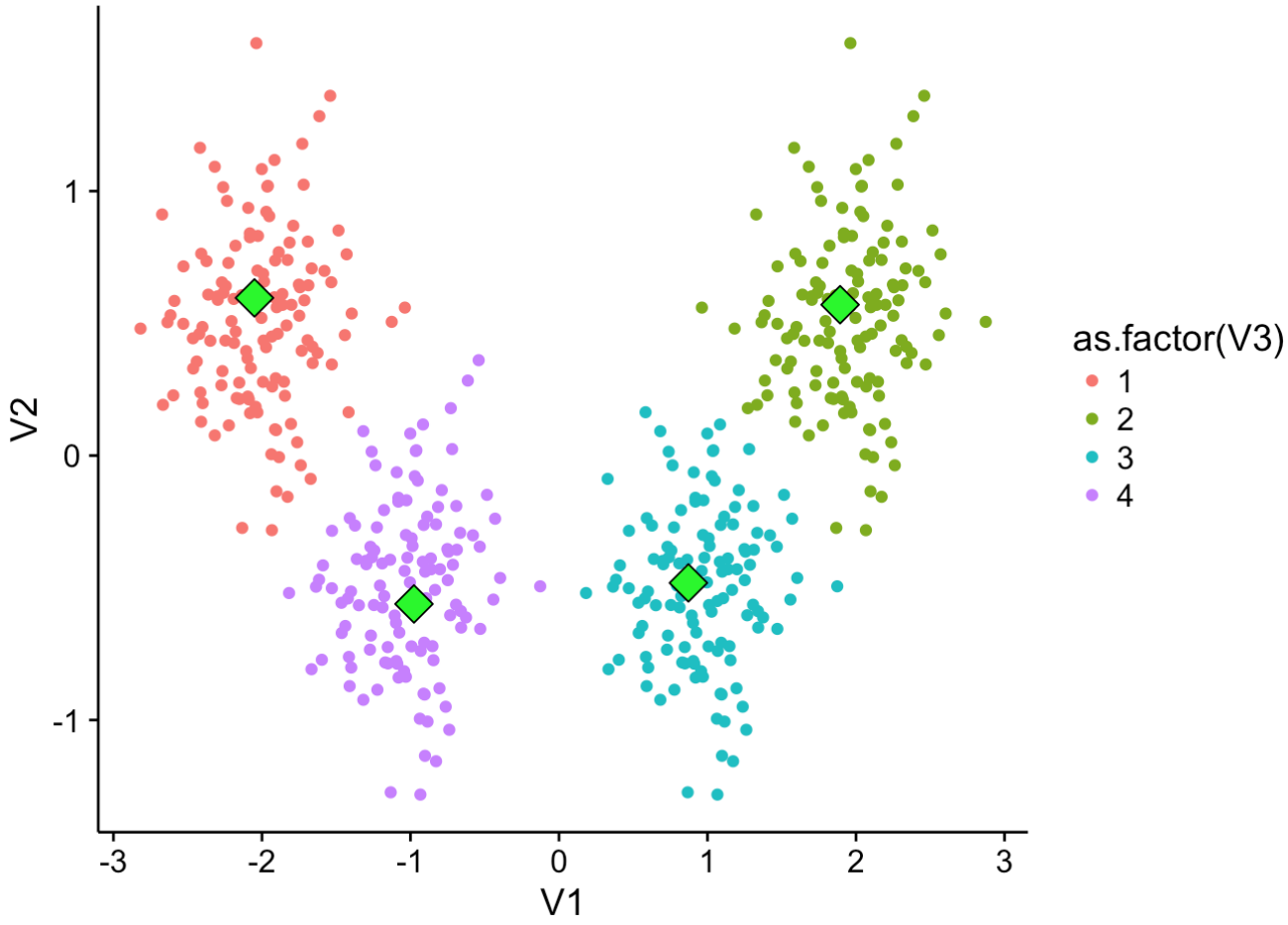

```
# plot 3: Iteration 166
WT3 <- W_evo[, 3, ]
ggplot() + geom_point(data = as.data.frame(cbind(cdata, point_assignment[,
    3])), aes(x = V1, y = V2, color = as.factor(V3))) + geom_point(data = as.data.fra
me(WT3),
    aes(x = V1, y = V2), fill = "orange", shape = 23, size = 5)
```



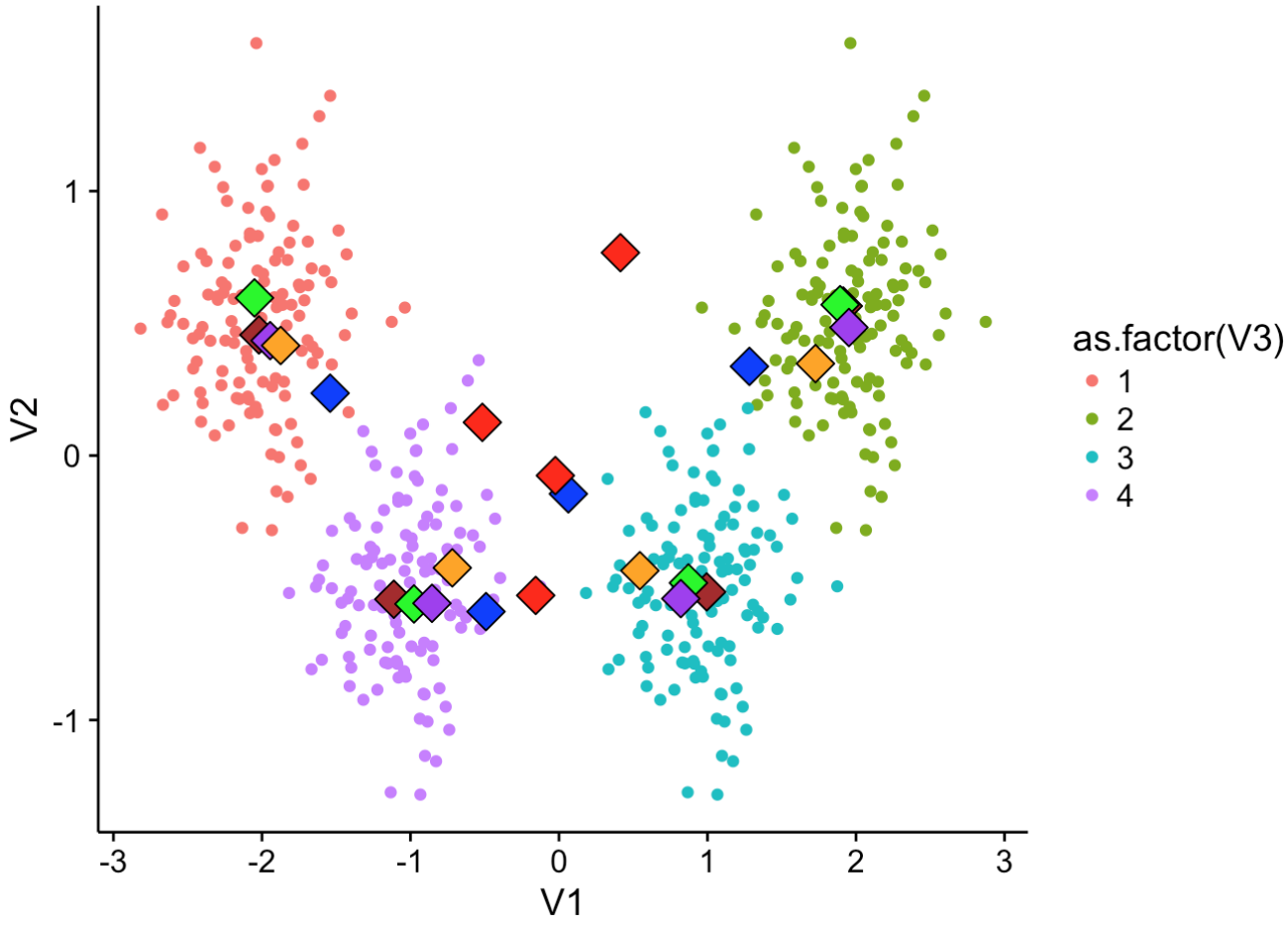

```
# plot 4: Iteration 250
WT4 <- W_evo[, 4, ]
ggplot() + geom_point(data = as.data.frame(cbind(cdata, point_assignment[,
    4])), aes(x = V1, y = V2, color = as.factor(V3))) + geom_point(data = as.data.fra
me(WT4),
    aes(x = V1, y = V2), fill = "purple", shape = 23, size = 5)
```




```
# plot 5: Iteration 333
WT5 <- W_evo[, 5, ]
ggplot() + geom_point(data = as.data.frame(cbind(cdata, point_assignment[,
    5])), aes(x = V1, y = V2, color = as.factor(V3))) + geom_point(data = as.data.fra
me(WT5),
    aes(x = V1, y = V2), fill = "green", shape = 23, size = 5)
```




```
# plot 6: Iteration 500
WT6 <- W_evo[, 6, ]
ggplot() + geom_point(data = as.data.frame(cbind(cdata, point_assignment[,
    6])), aes(x = V1, y = V2, color = as.factor(V3))) + geom_point(data = as.data.fra
me(WT6),
    aes(x = V1, y = V2), fill = "brown", shape = 23, size = 5) +
    geom_point(data = as.data.frame(WT5), aes(x = V1, y = V2),
        fill = "green", shape = 23, size = 5) + geom_point(data = as.data.frame(WT4),

    aes(x = V1, y = V2), fill = "purple", shape = 23, size = 5) +
    geom_point(data = as.data.frame(WT3), aes(x = V1, y = V2),
        fill = "orange", shape = 23, size = 5) + geom_point(data =
as.data.frame(WT2),
    aes(x = V1, y = V2), fill = "blue", shape = 23, size = 5) +
    geom_point(data = as.data.frame(WT1), aes(x = V1, y = V2),
        fill = "red", shape = 23, size = 5)
```



9.3 soft K-means

```
cluster_data <- read.table("cluster.dat")

cluster_data <- as.data.frame(t(cluster_data))
```

a)

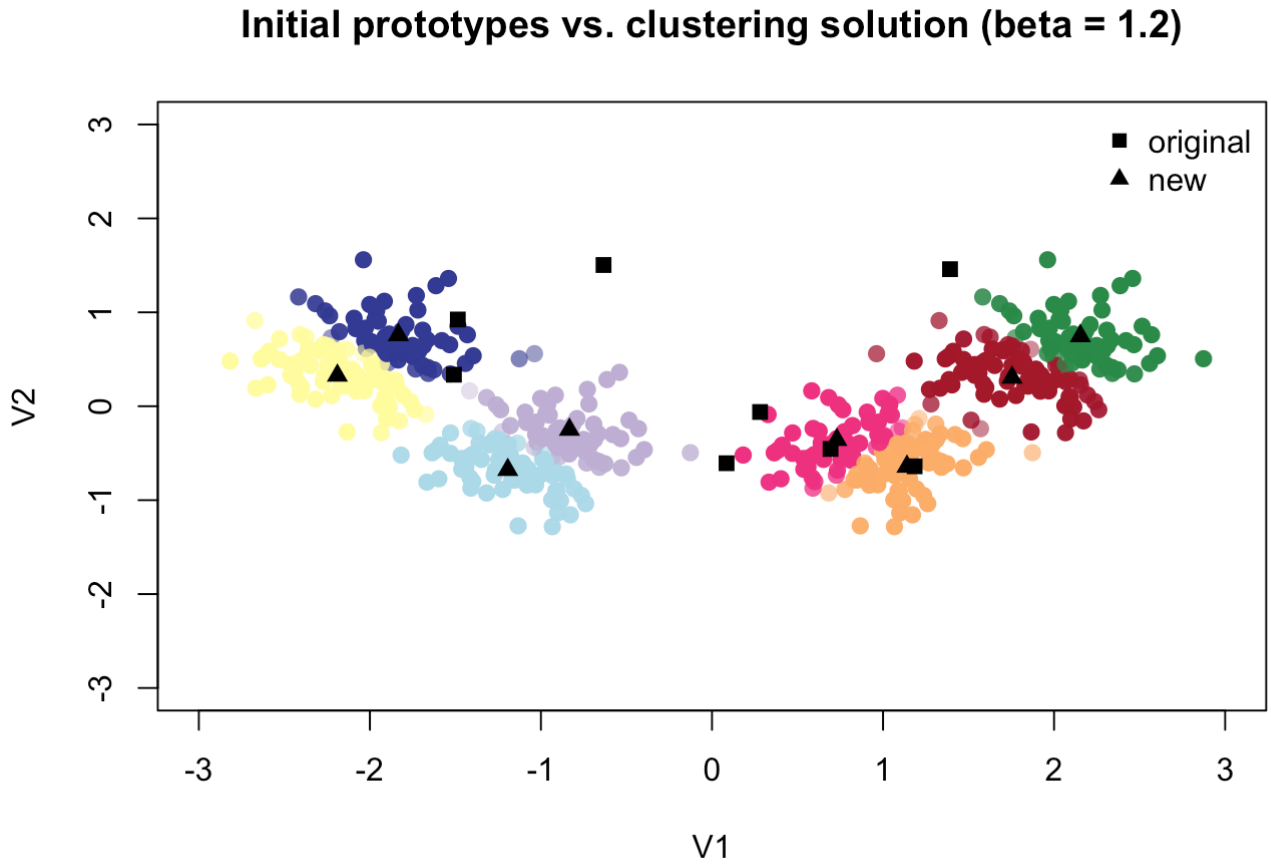
```
# number of initial prototypes
K = 8

data_mean <- data.frame(m1 = rep(mean(cluster_data$V1), K), m2 = rep(mean(cluster_dat
a$V2),
    K))

# set initial prototypes
initial_prototypes <- data_mean + matrix(runif(16, -1.5, 1.5),
    nrow = K)
# initial_prototypes <- matrix(runif(16, -3, 3), ncol = 2)

# plot initial prototypes plot(cluster_data, pch = 16, main =
# 'Initial prototypes') points(initial_prototypes, pch = 16,
# col = 'red', cex = 1.5)


# convergence tolerance
gamma = 0.001
```

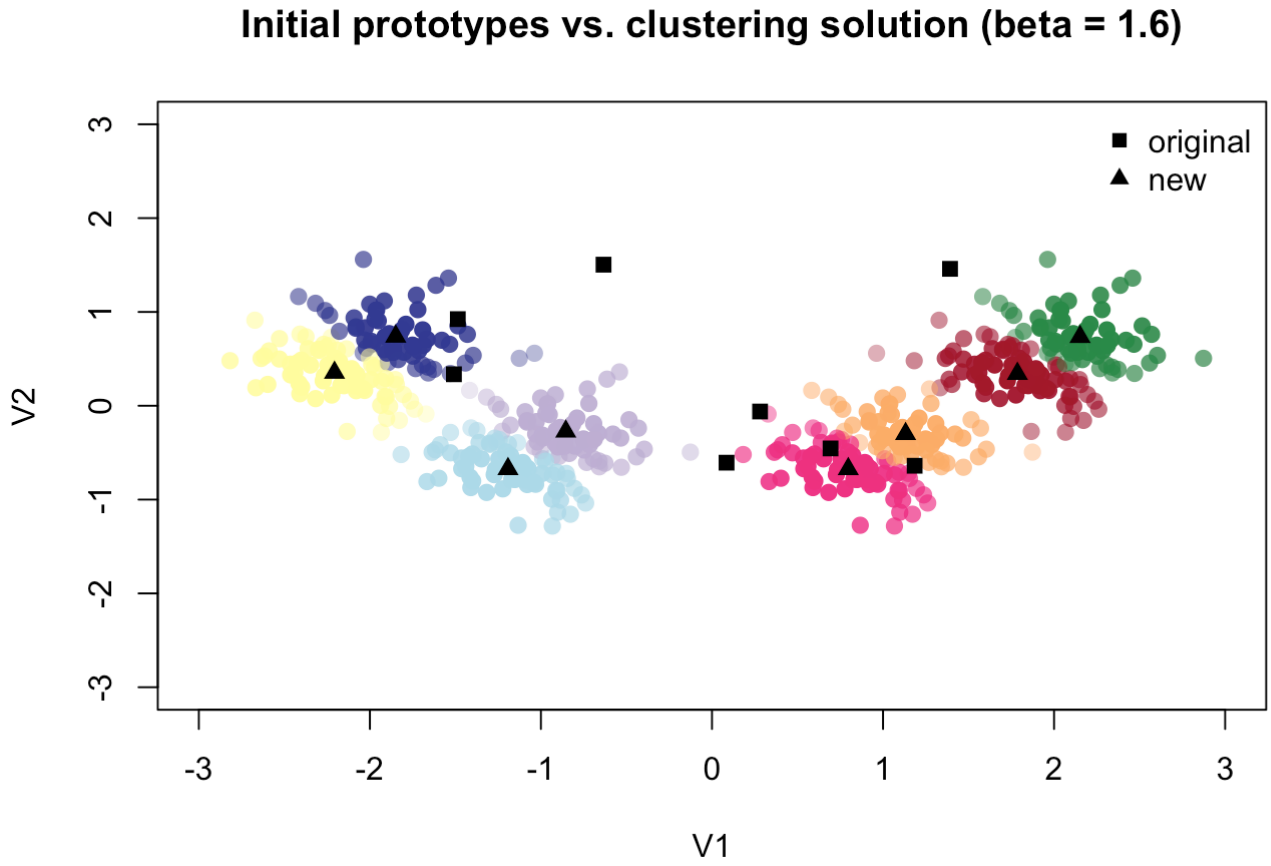
b)

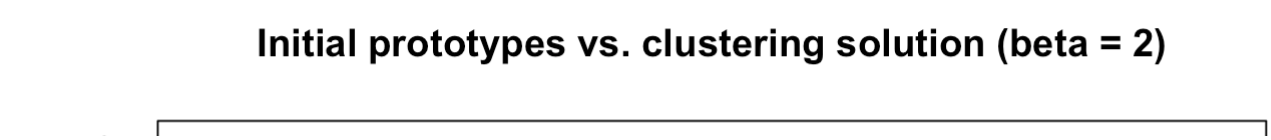
```
a_probs = matrix(nrow = nrow(cluster_data), ncol = K)

betas <- seq(from = 1.2, to = 20, by = 0.4)
beta = 1

w = as.matrix(initial_prototypes)


data_points = as.matrix(cluster_data)

assignment_probs = matrix(nrow = nrow(data_points), ncol = K)

diff = w

while (any(diff > gamma)) {

    for (alpha in 1:nrow(data_points)) {

        z = exp(-beta/2 * (rowSums((data_points[alpha, ] - w)^2)))
        n = sum(exp(-beta/2 * (rowSums((data_points[alpha, ] -
            w)^2))))

        assignment_probs[alpha, ] = z/n

    }

    w_new = (t(assignment_probs) %*% data_points)/colSums(assignment_probs)

    diff = abs(w_new - w)

    w = w_new
}

# plot(cluster_data, pch = 16, main = 'Initial prototypes',
# xlim = c(-3,3), ylim = c(-3,3))
# points(initial_prototypes[,1], initial_prototypes[,2], pch
# = 16, col = 'red', cex = 1.5) points(w_new[,1], w_new[,2],
# pch = 16, col = 'blue', cex = 1.5)
```

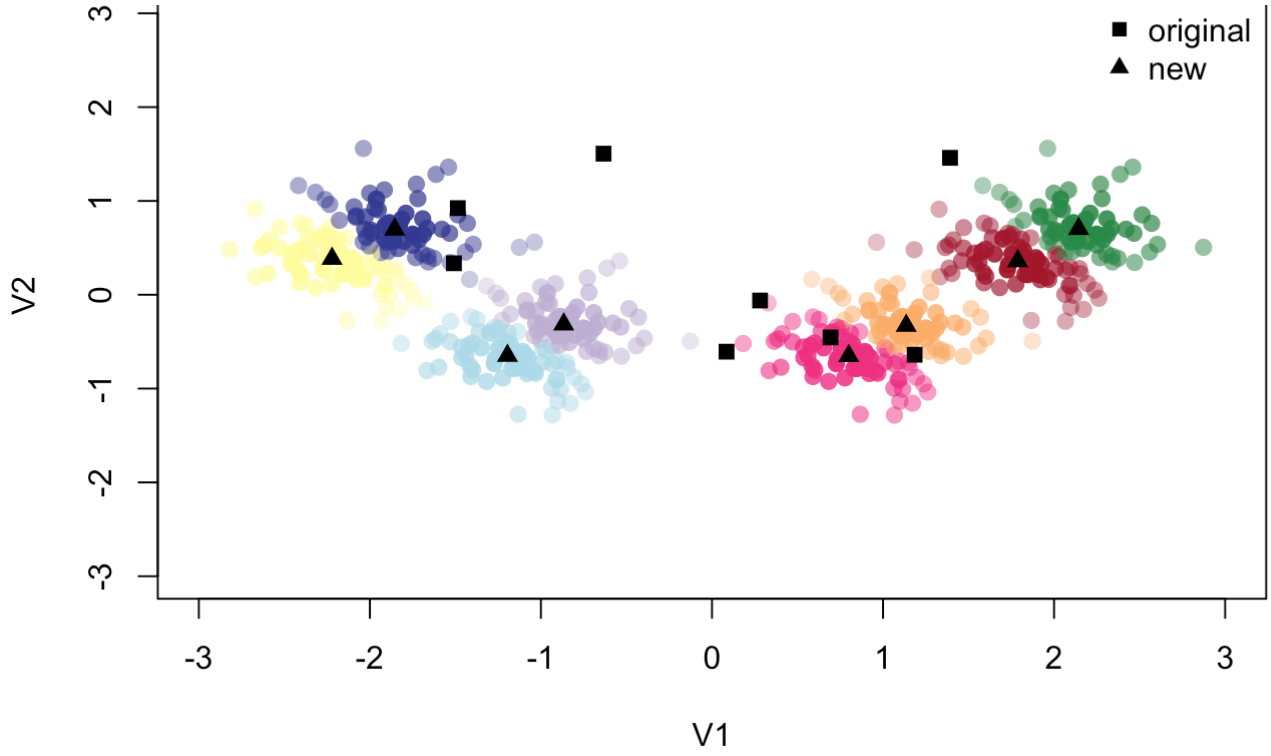
c)

```
cols = c("#313695", "#238b45", "#f0027f", "#abd9e9", "#fdae61",
    "#beaed4", "#ffff99", "#a50026")

par(mfrow = c(1, 1))

for (beta in betas) {

    cl = cmeans(data_points, dist = "euclidean", centers = initial_prototypes,
        m = beta)

    brightness = apply(cl$membership, 1, max)

    plot(cluster_data, pch = 16, main = paste("Initial prototypes vs. clustering solu
tion (beta = ",
        beta, ")", sep = ""), xlim = c(-3, 3), ylim = c(-3, 3),
        col = alpha(cols[cl$cluster], brightness), cex = 1.2)
    points(initial_prototypes[, 1], initial_prototypes[, 2],
        pch = 15, col = "black", cex = 1.1)
    points(cl$centers[, 1], cl$centers[, 2], pch = 17, col = "black",
        cex = 1.1)
    legend("topright", legend = c("original", "new"), pch = c(15,
        17), cex = 1, bty = "n")
}
```

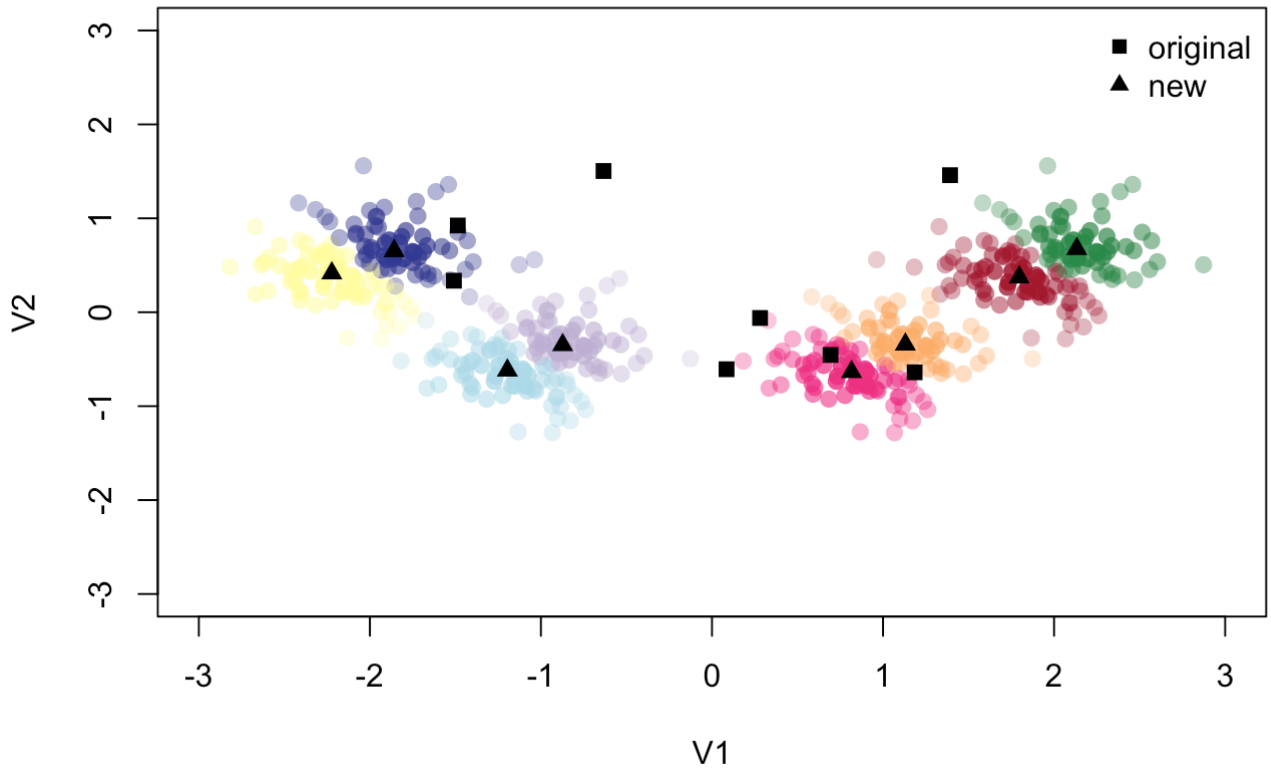

Initial prototypes vs. clustering solution (beta = 1.2)
original
new
V2
V1
-3
-2
-1
0
1
2
3

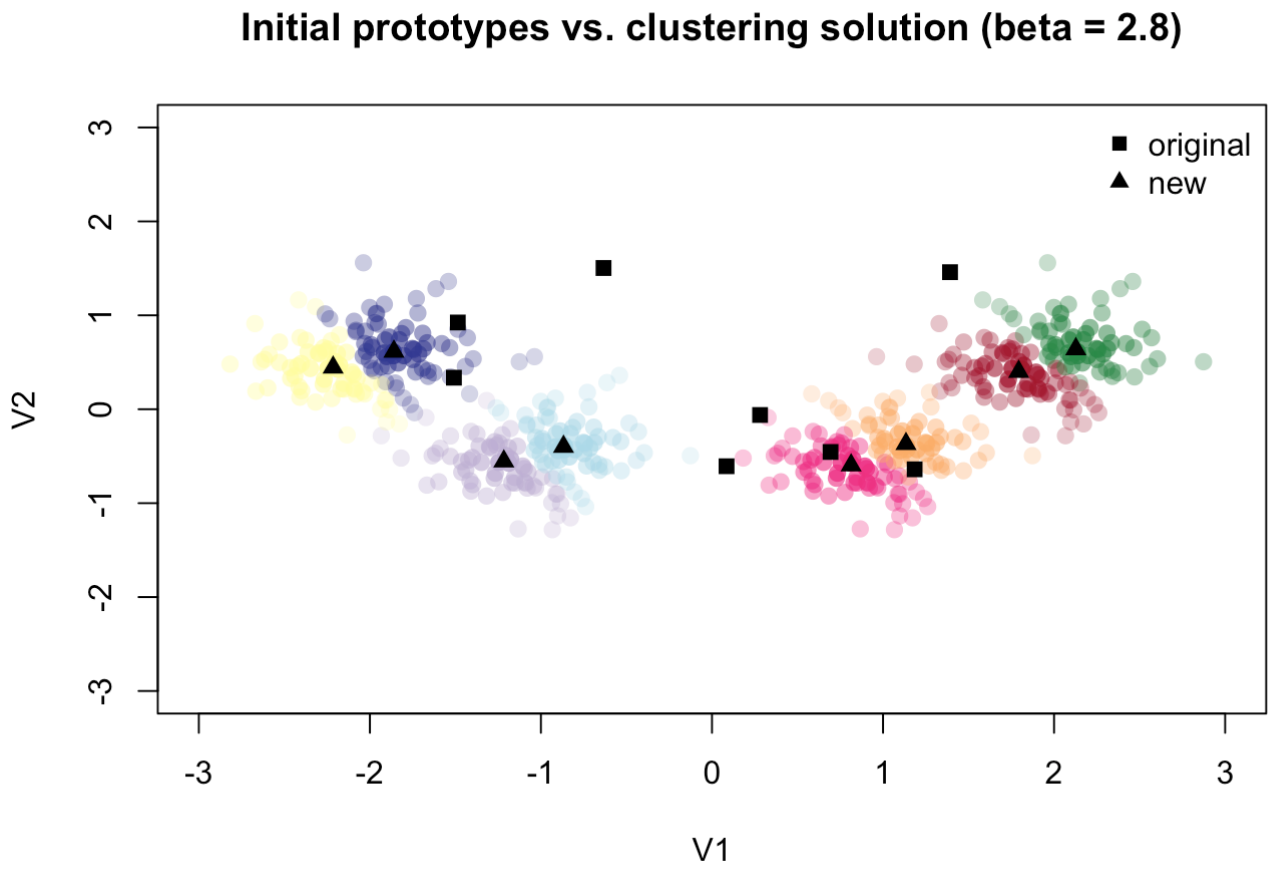

Initial prototypes vs. clustering solution (beta = 1.6)
original
new
V2
V1
-3
-2
-1
0
1
2
3

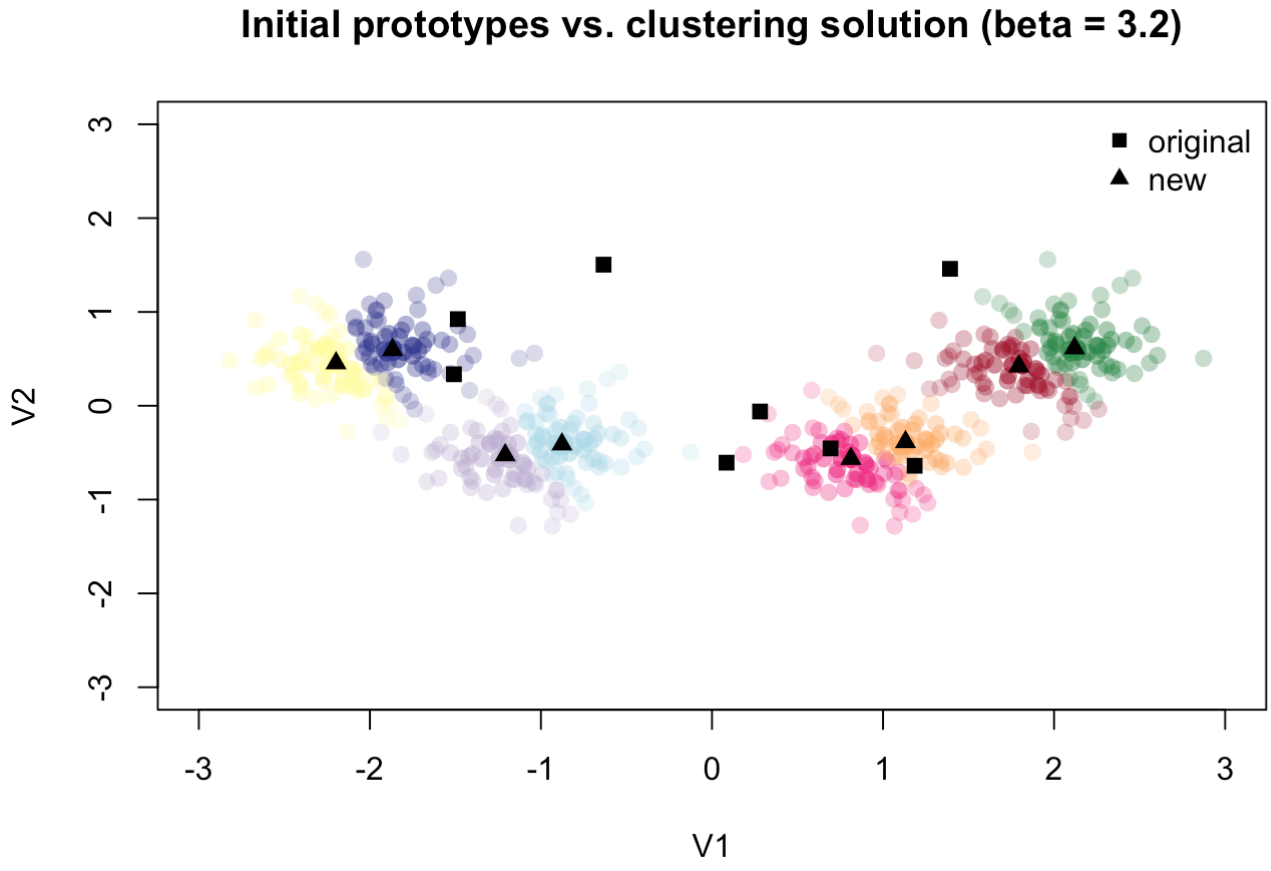

Initial prototypes vs. clustering solution (beta = 2)

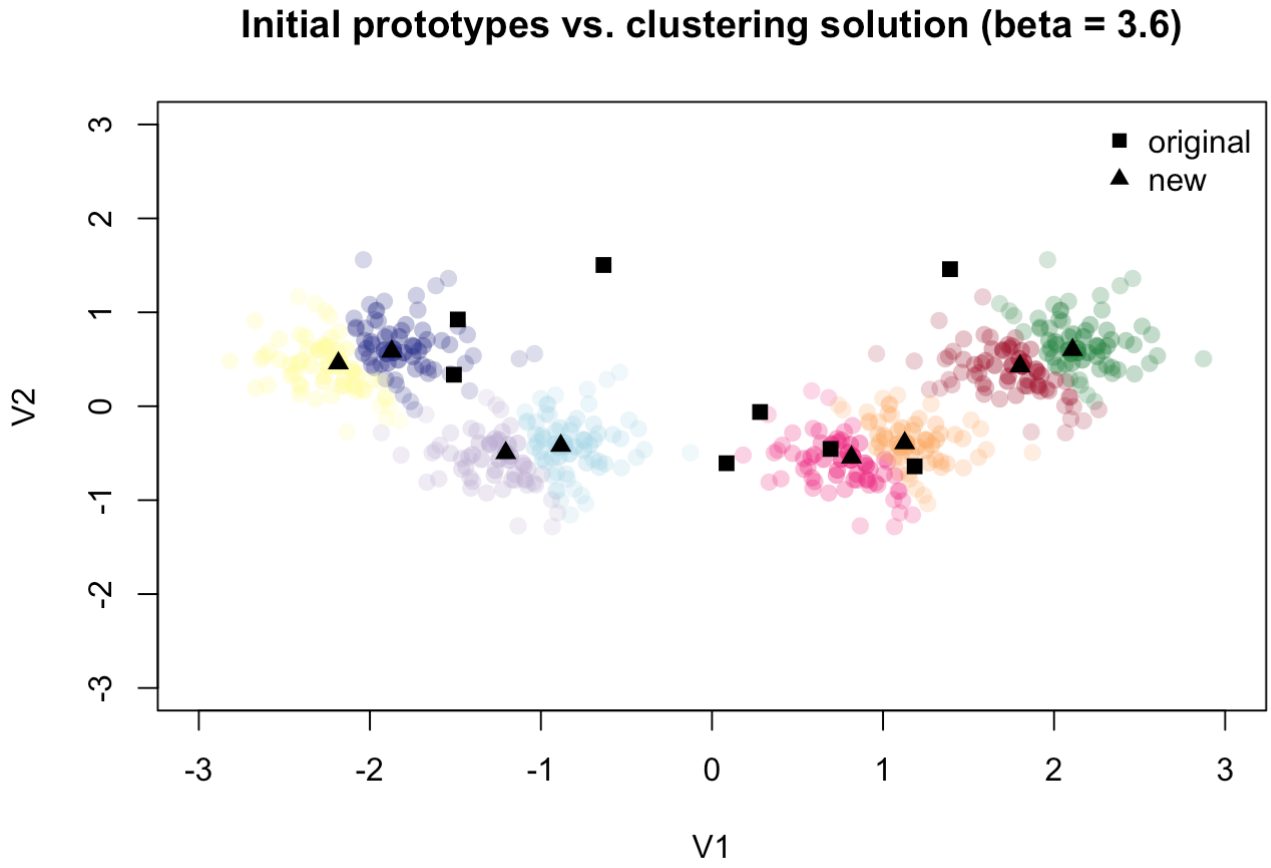


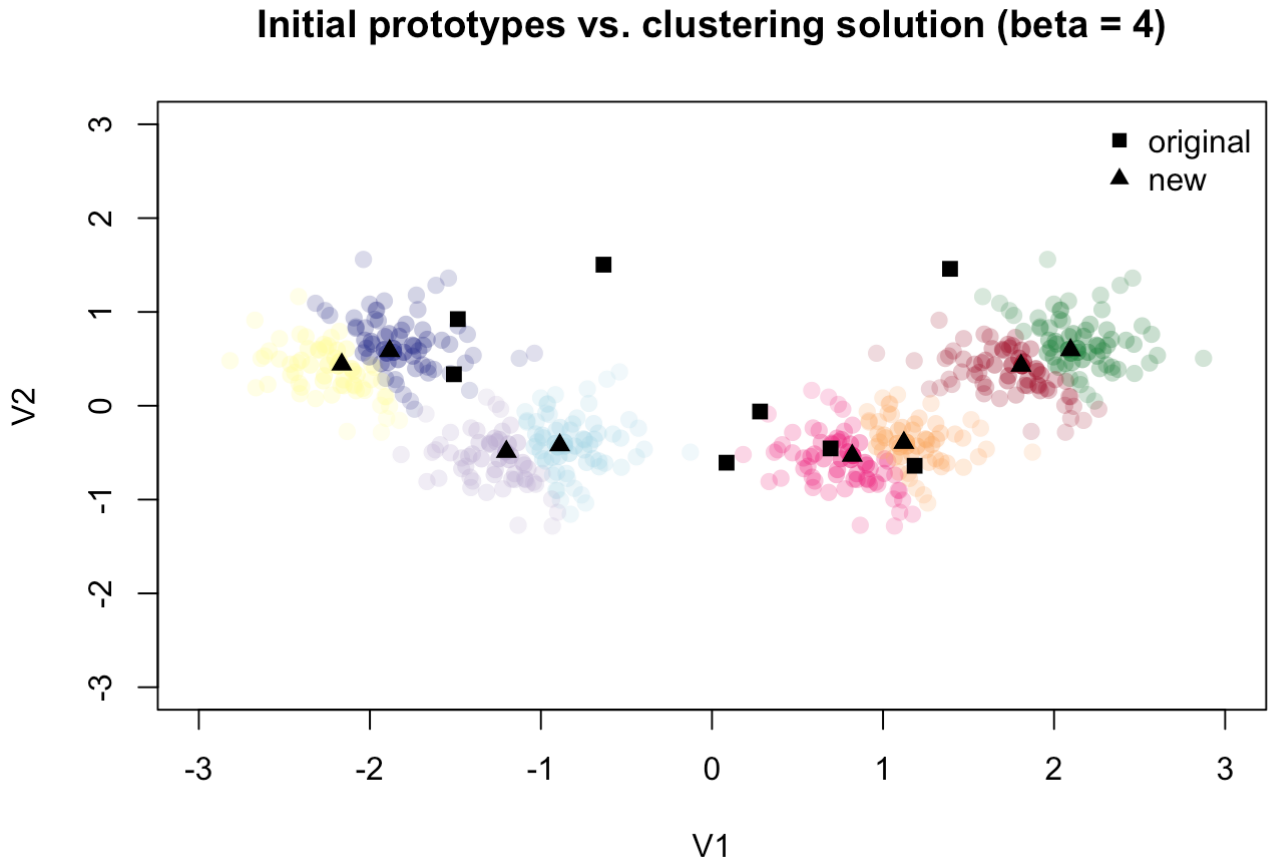
Initial prototypes vs. clustering solution (beta = 2.4)

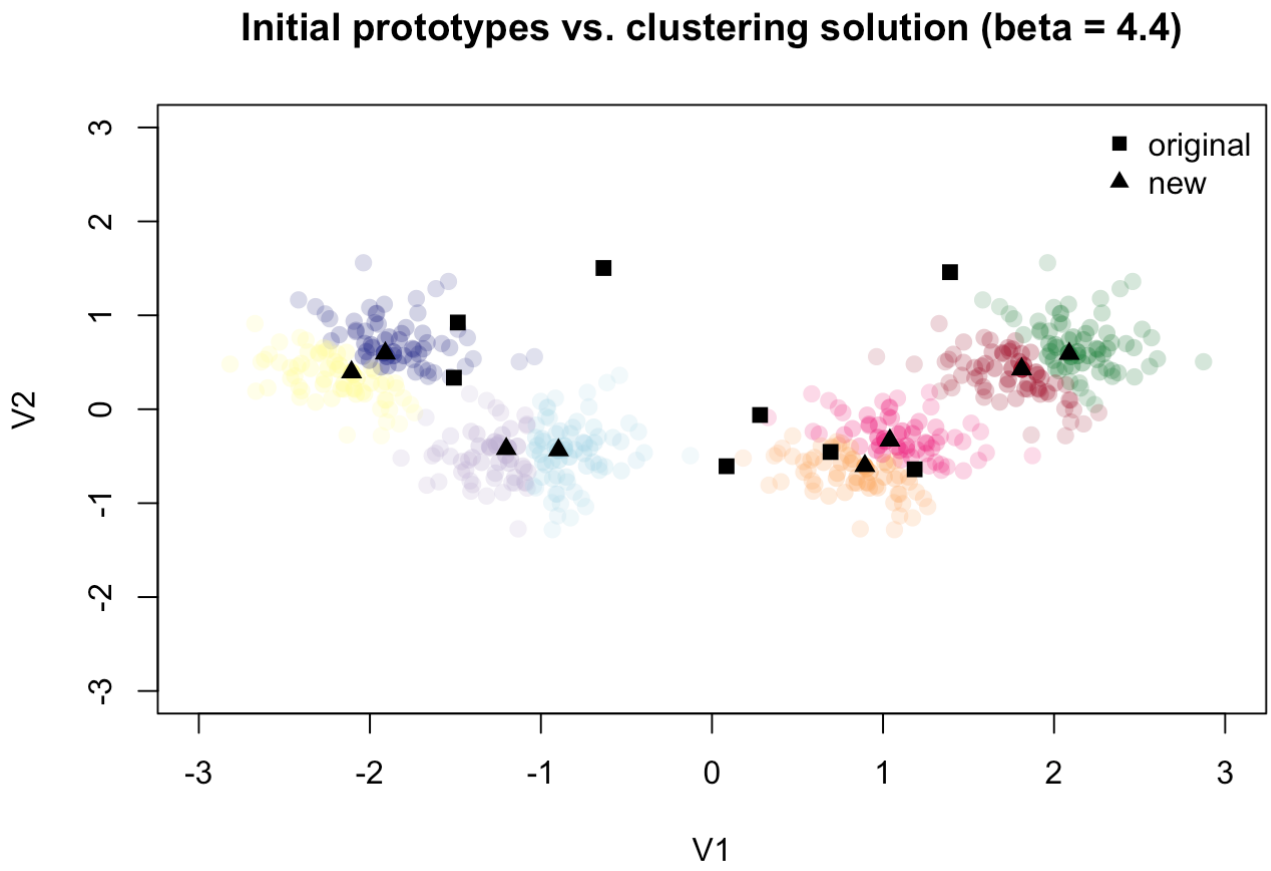


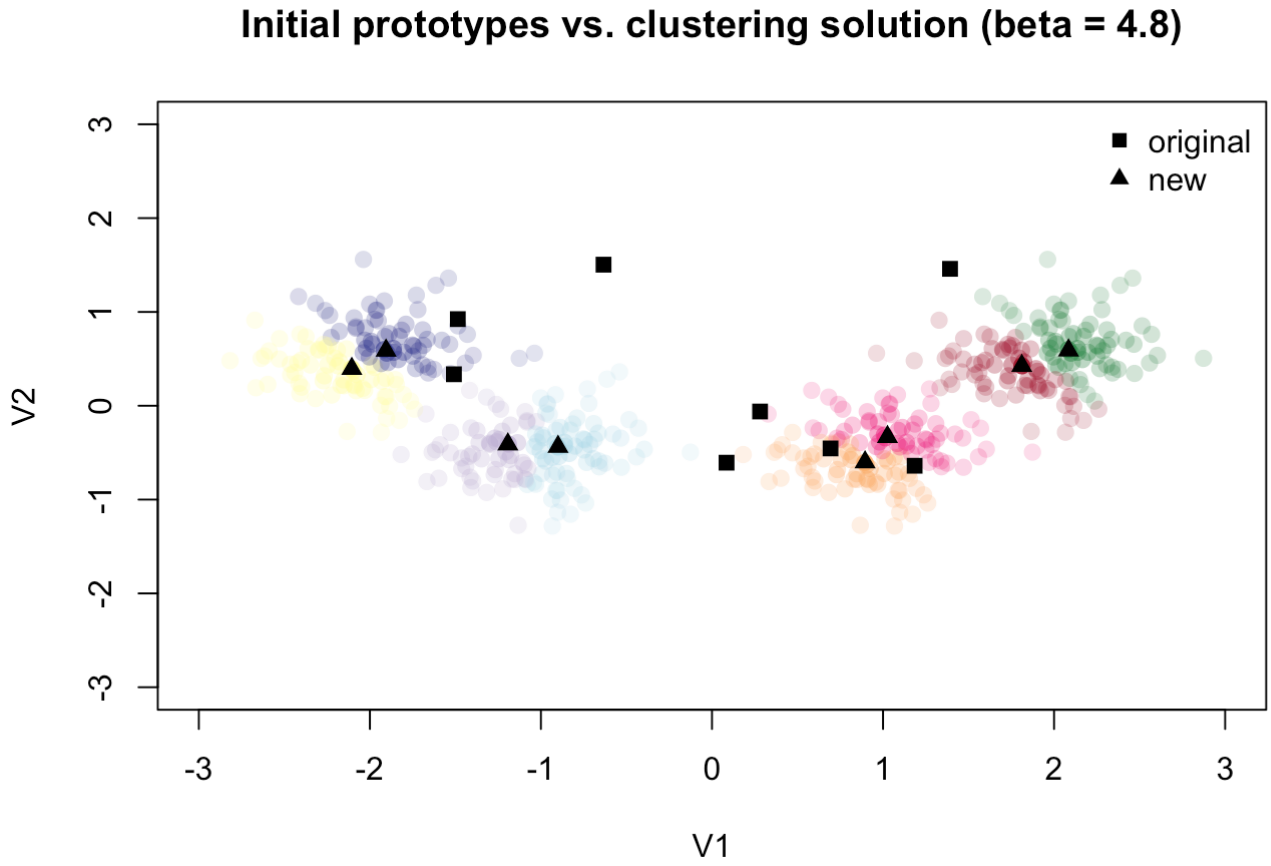

Initial prototypes vs. clustering solution (beta = 2.8)
original
new
V2
V1
-3
-2
-1
0
1
2
3

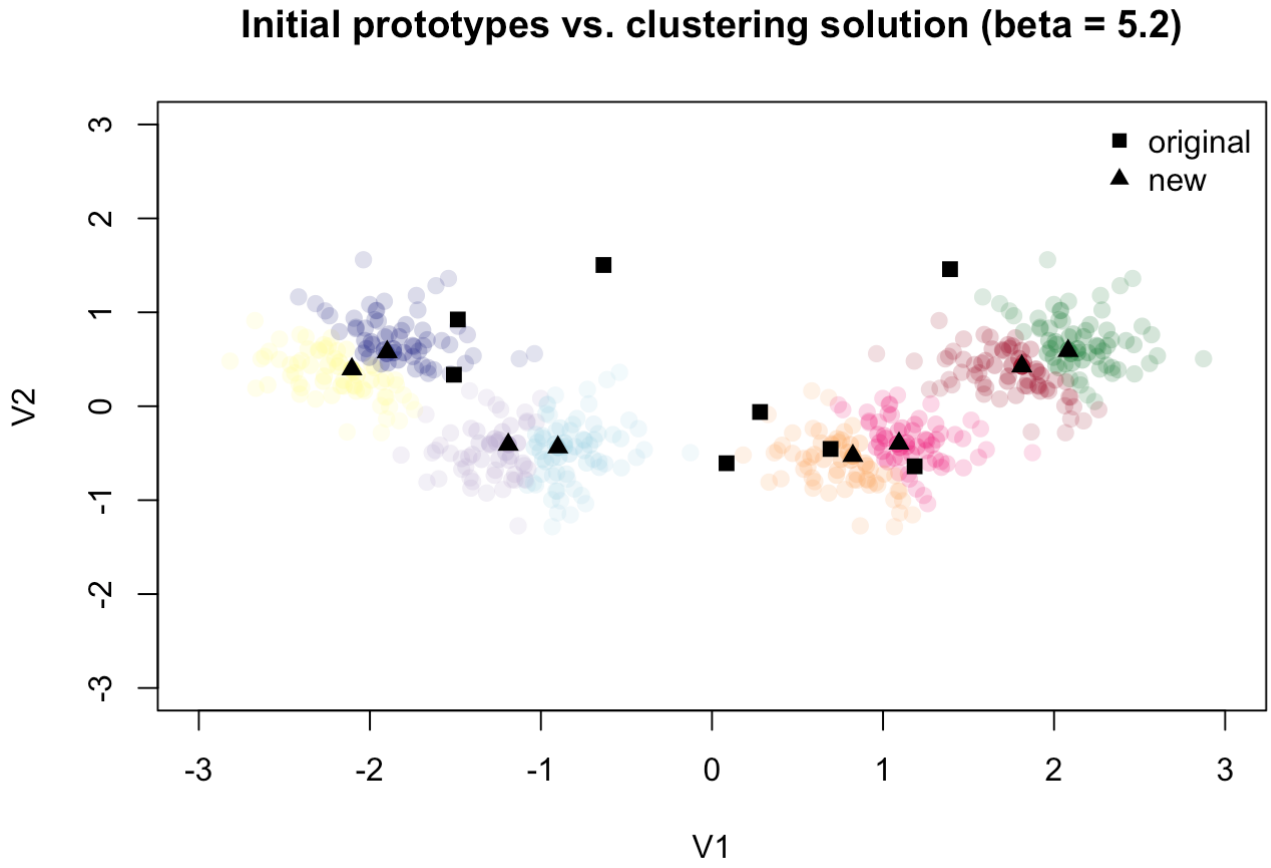

Initial prototypes vs. clustering solution (beta = 3.2)
original
new
V2
V1
-3
-2
-1
0
1
2
3

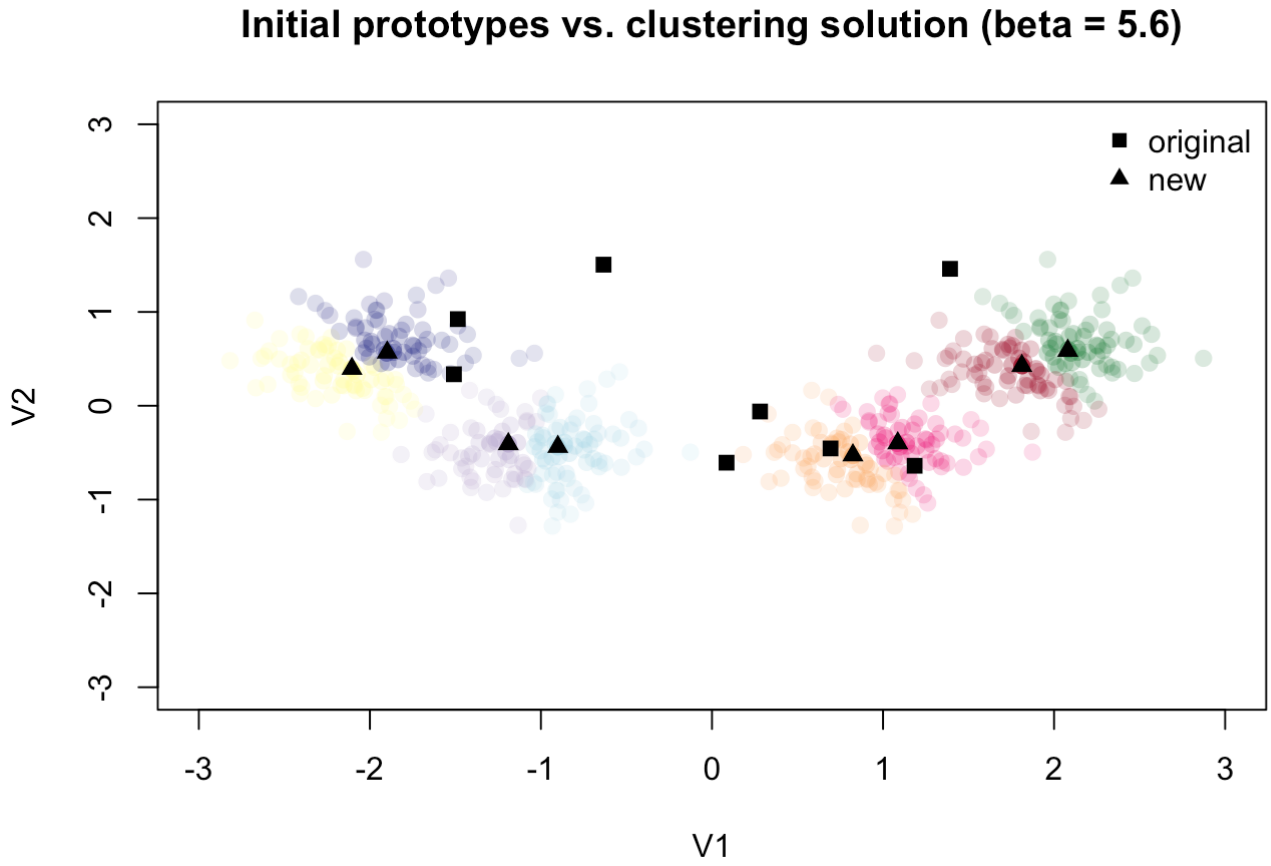

Initial prototypes vs. clustering solution (beta = 3.6)
original
new
V2
V1
-3
-2
-1
0
1
2
3

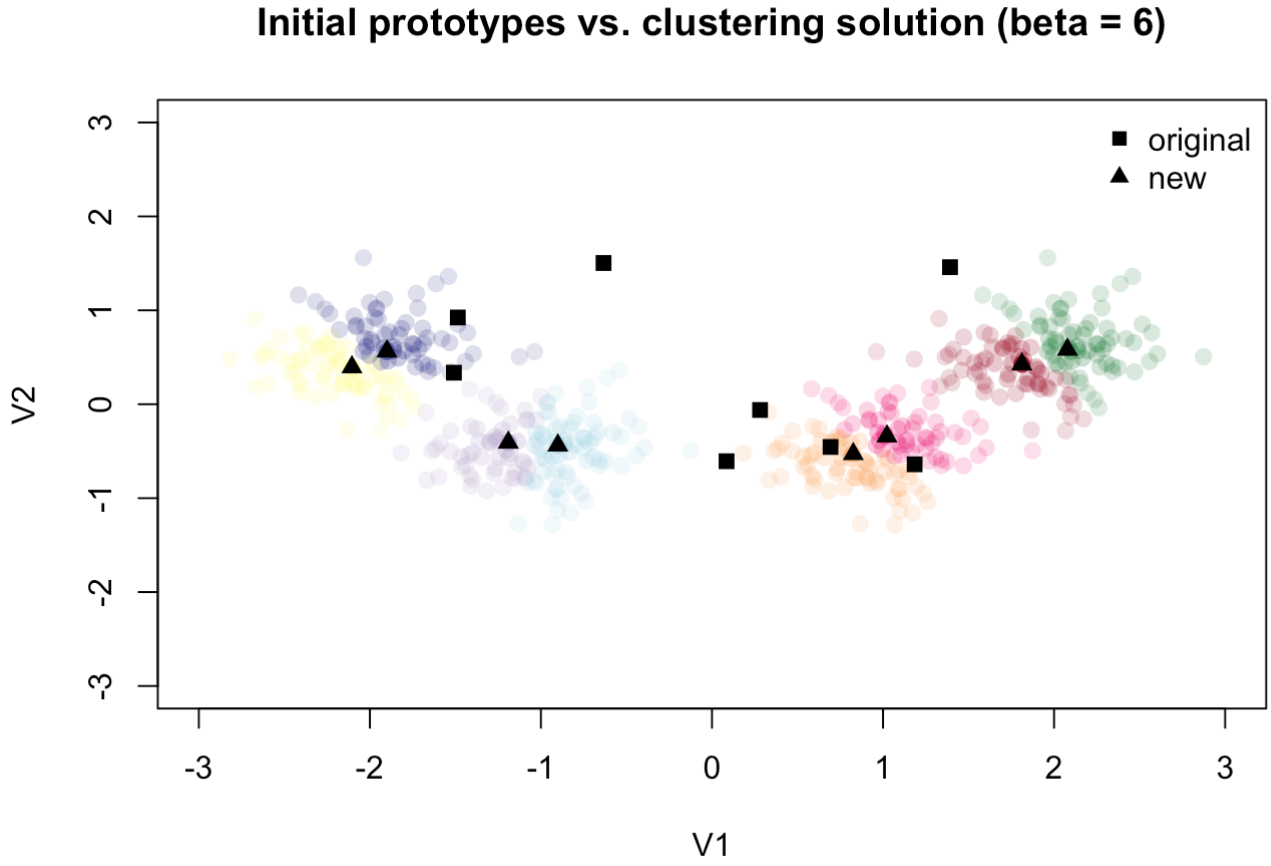

Initial prototypes vs. clustering solution (beta = 4)
original
new
V2
V1
-3
-2
-1
0
1
2
3

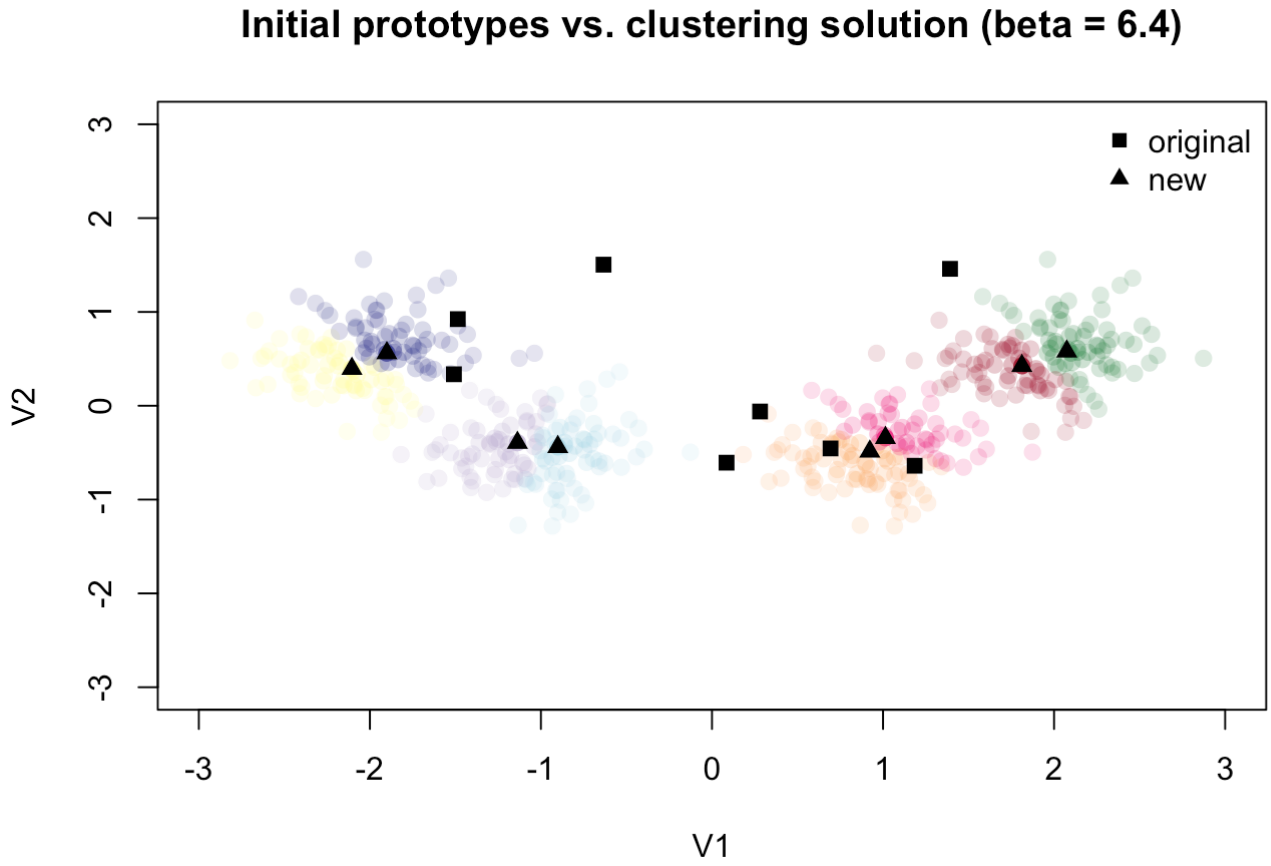

Initial prototypes vs. clustering solution (beta = 4.4)
original
new
V2
V1
-3
-2
-1
0
1
2
3

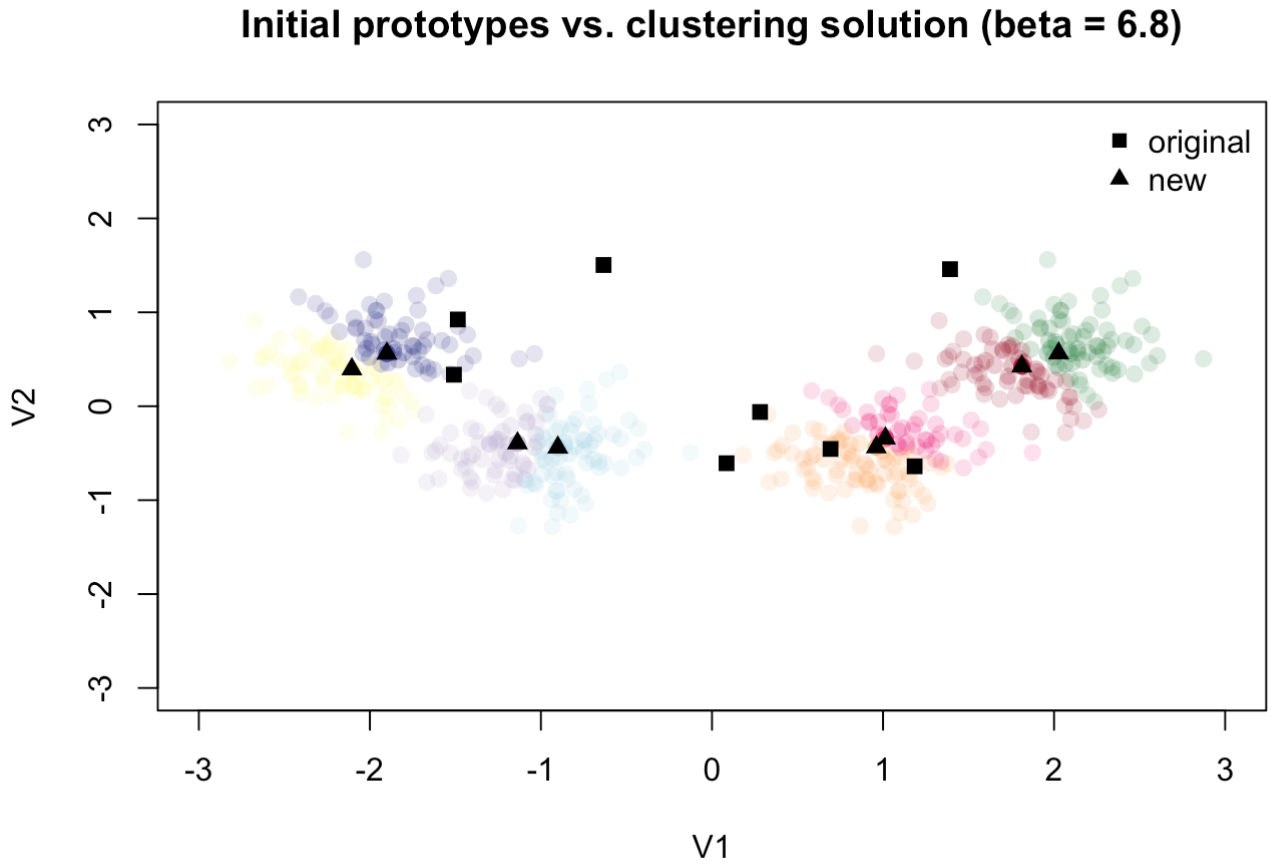

Initial prototypes vs. clustering solution (beta = 4.8)
original
new
V2
V1
-3
-2
-1
0
1
2
3

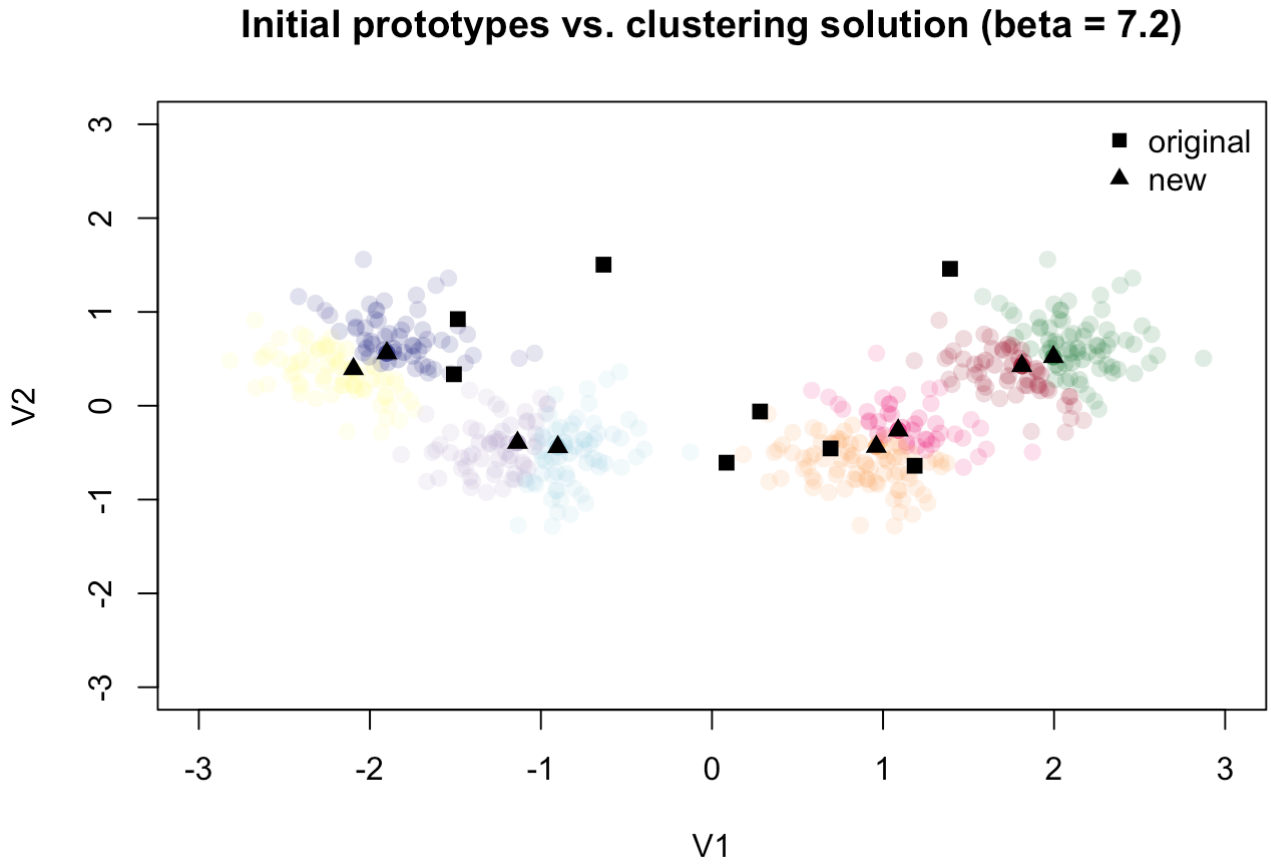

Initial prototypes vs. clustering solution (beta = 5.2)
original
new
V2
V1
-3
-2
-1
0
1
2
3

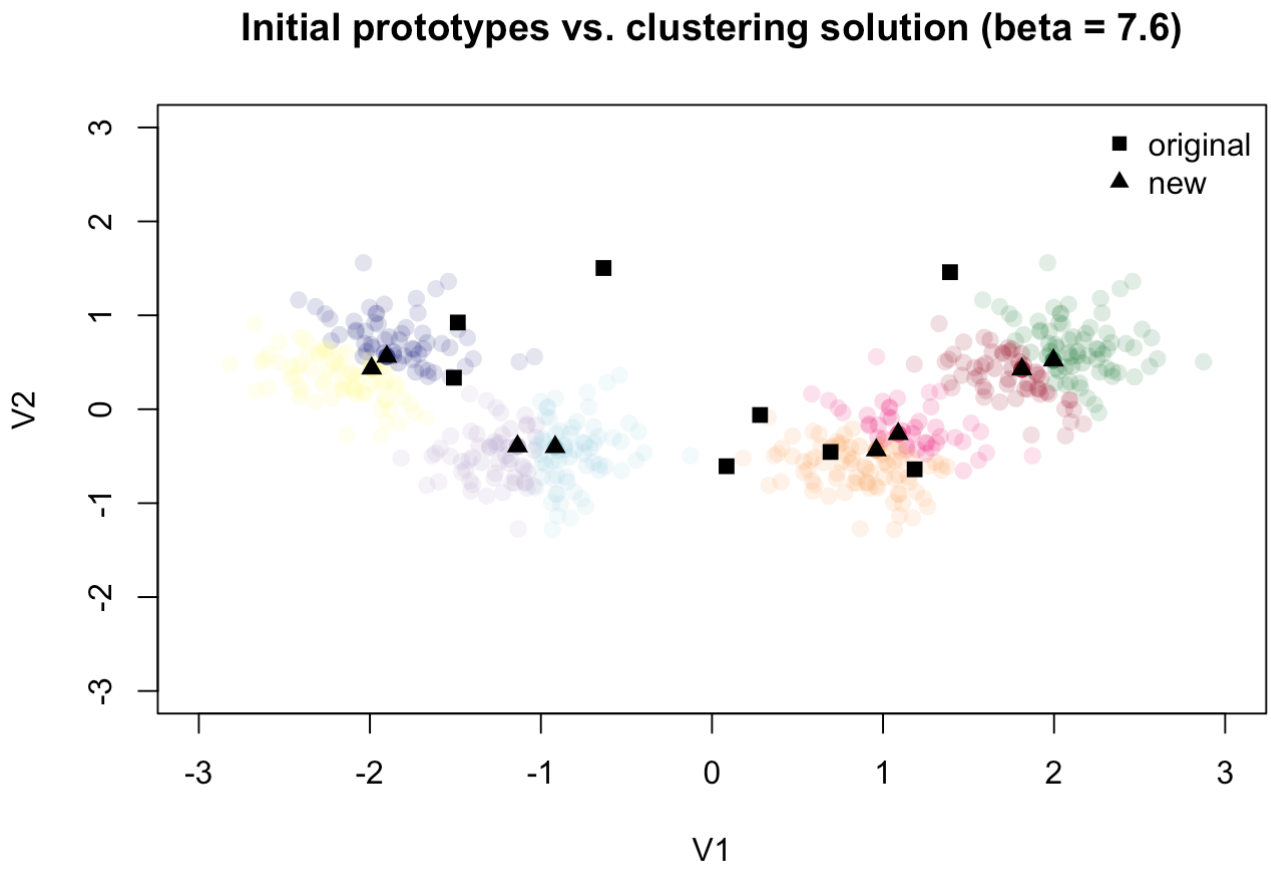

Initial prototypes vs. clustering solution (beta = 5.6)
original
new
V2
V1
-3
-2
-1
0
1
2
3

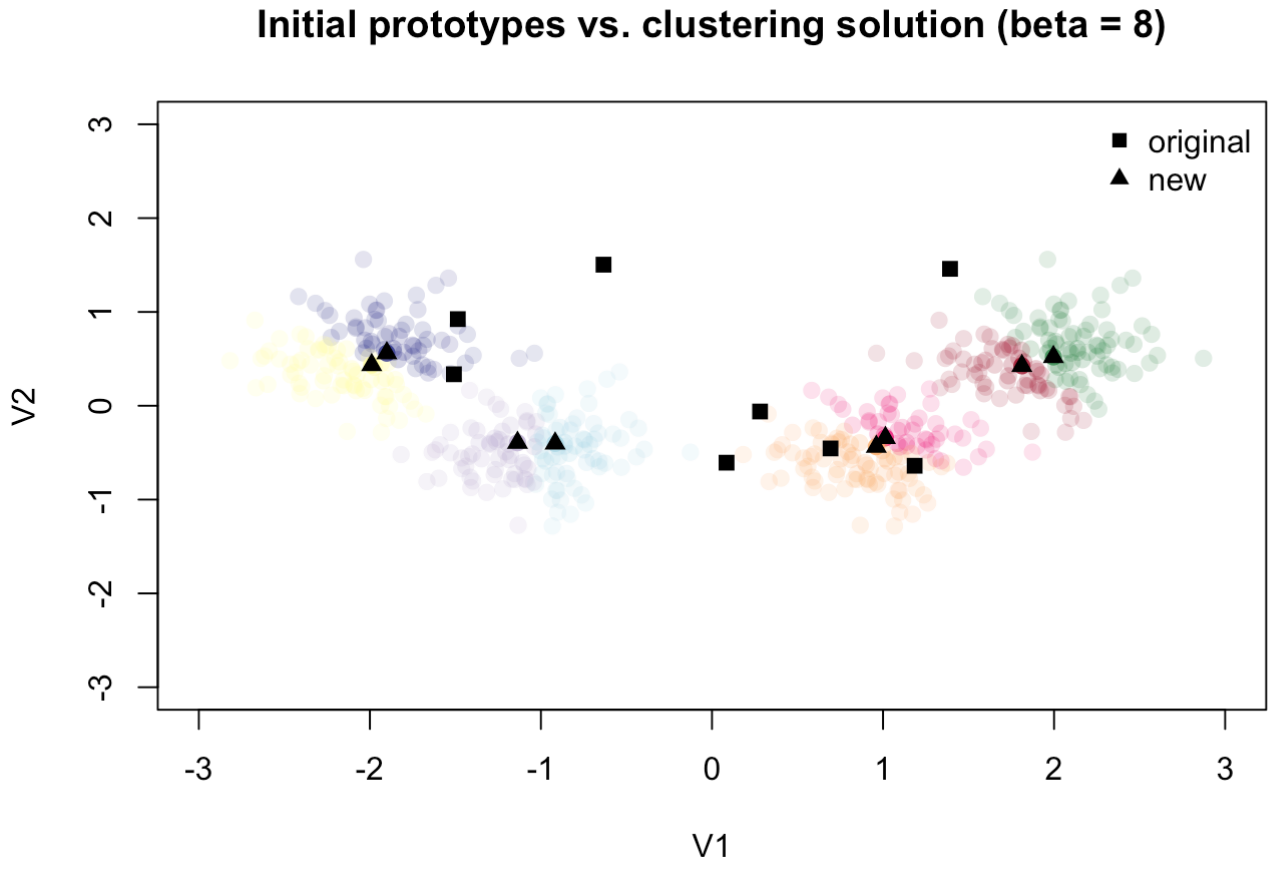

Initial prototypes vs. clustering solution (beta = 6)
original
new
V2
V1
-3
-2
-1
0
1
2
3

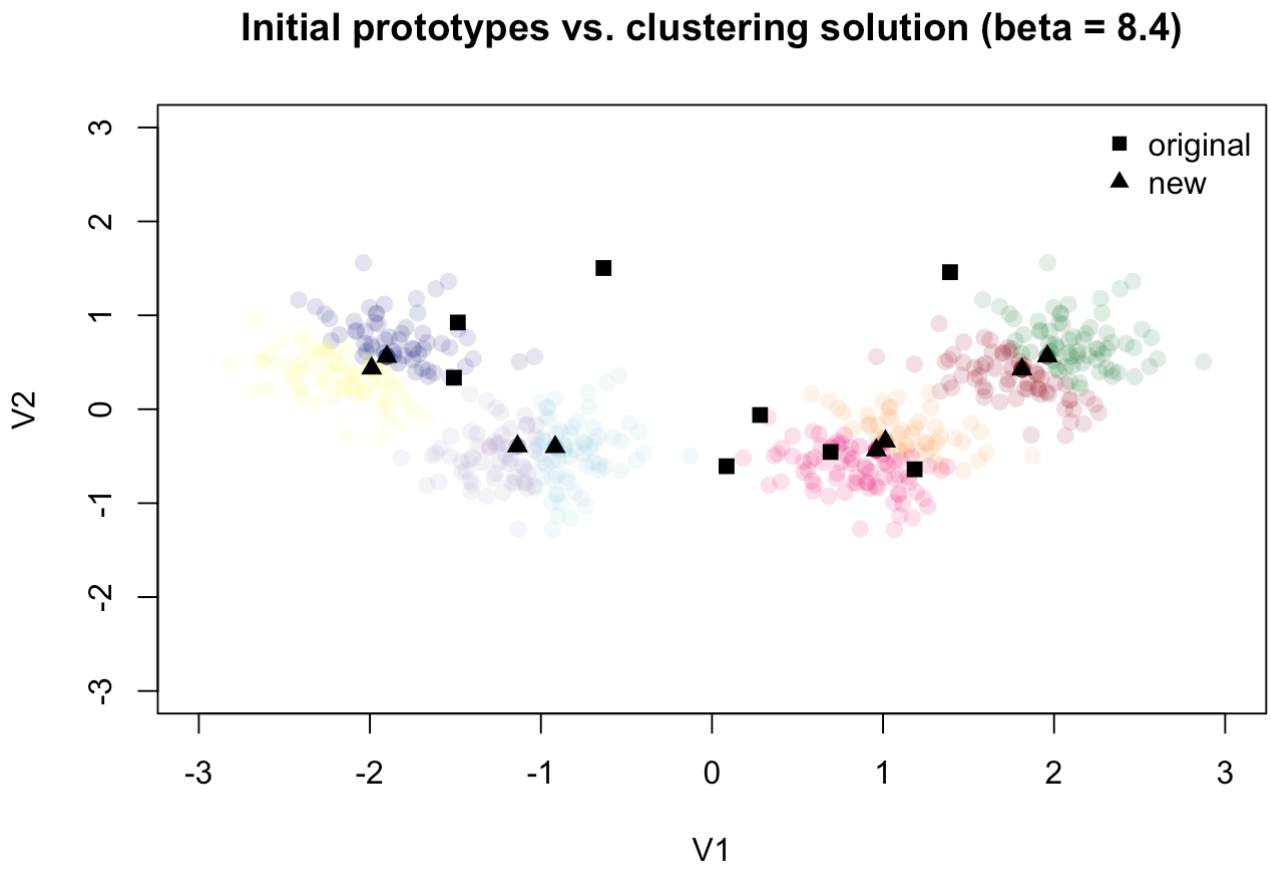

Initial prototypes vs. clustering solution (beta = 6.4)
original
new
V2
V1
-3
-2
-1
0
1
2
3

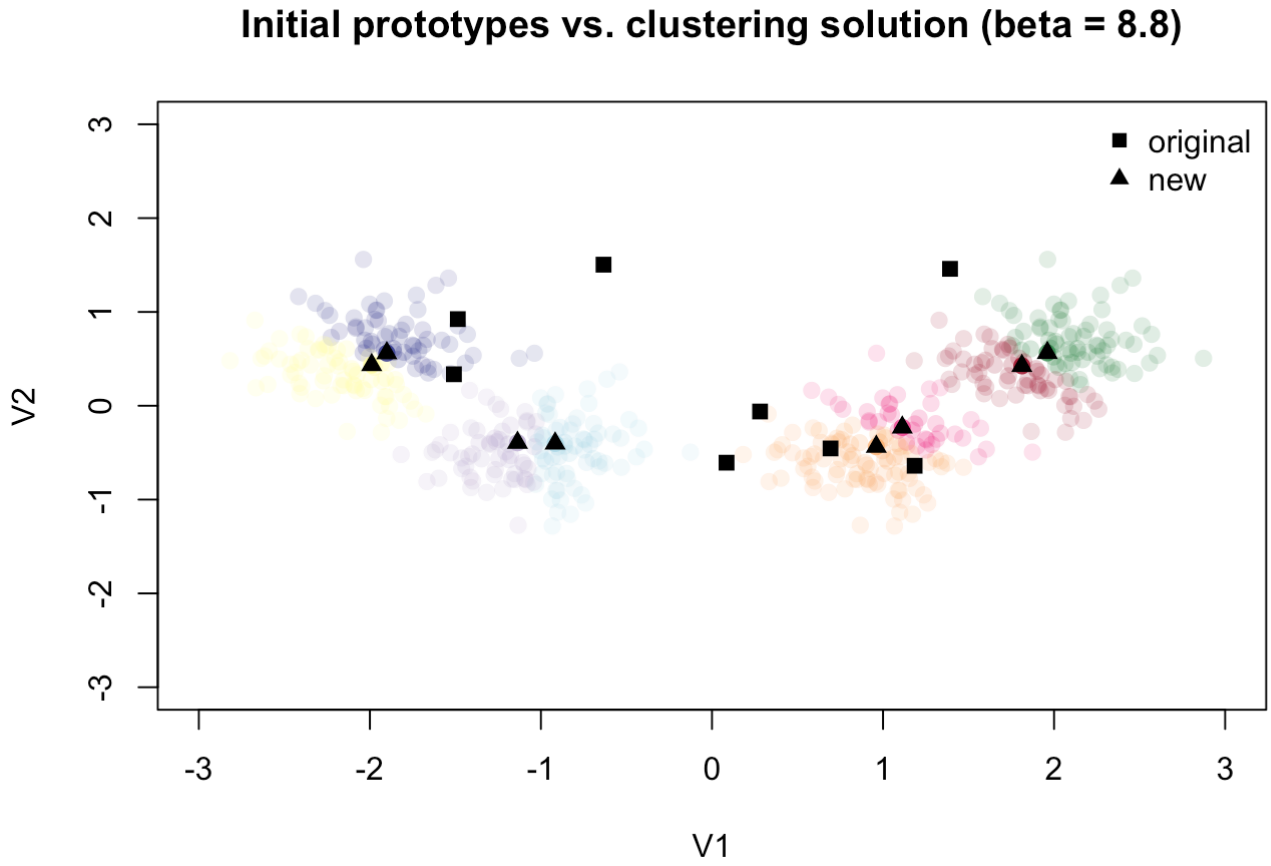

Initial prototypes vs. clustering solution (beta = 6.8)
original
new
V2
V1
-3
-2
-1
0
1
2
3

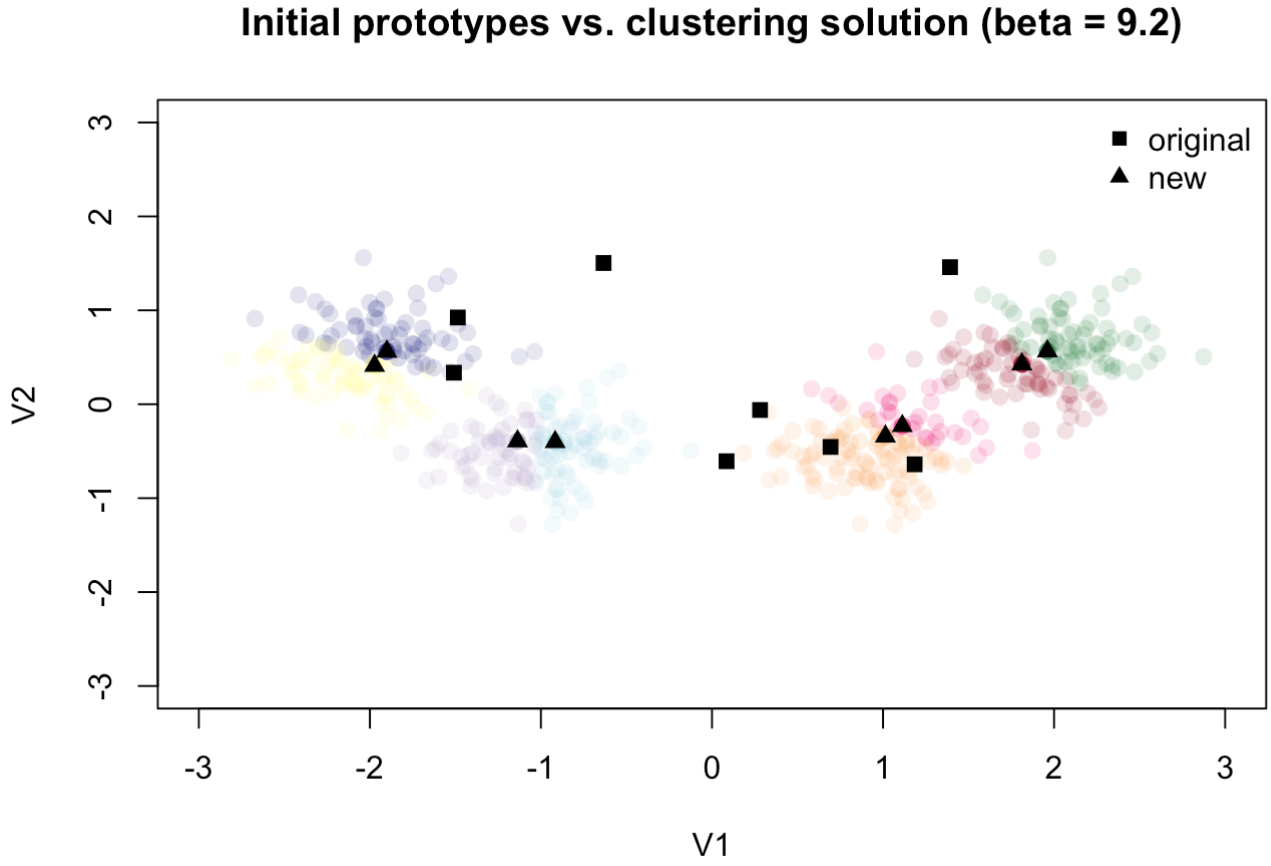

Initial prototypes vs. clustering solution (beta = 7.2)
original
new
V2
V1
-3
-2
-1
0
1
2
3

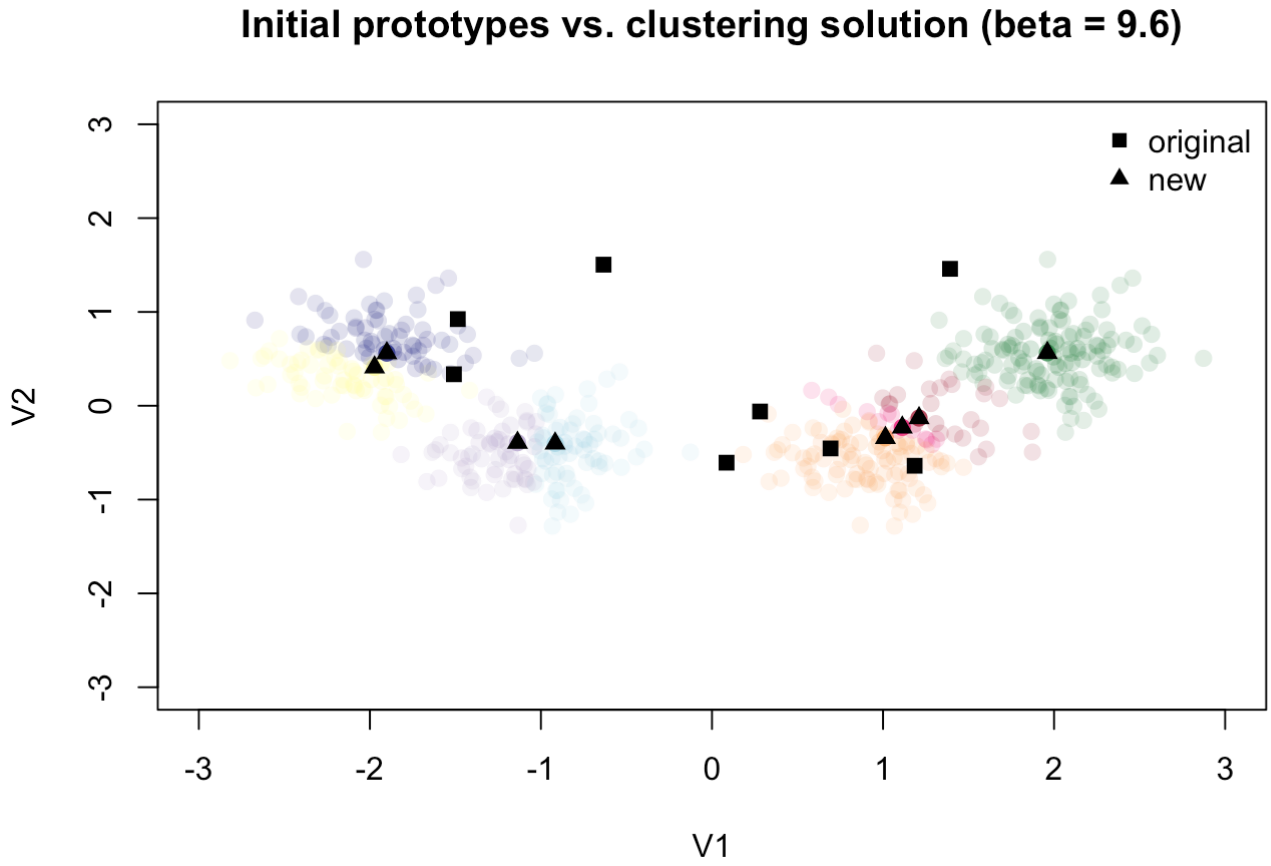

Initial prototypes vs. clustering solution (beta = 7.6)
original
new
V2
V1
-3
-2
-1
0
1
2
3

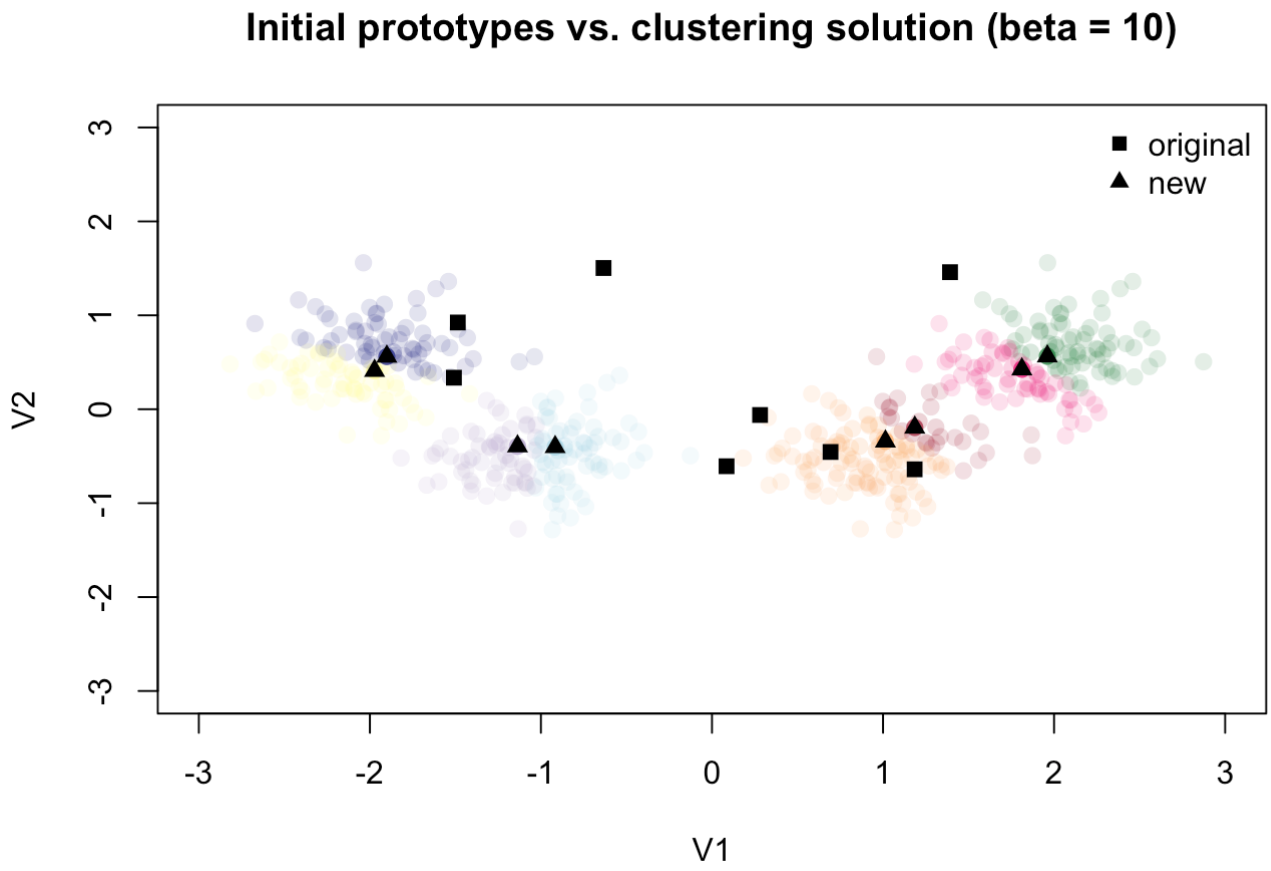

Initial prototypes vs. clustering solution (beta = 8)
original
new
V2
V1
-3
-2
-1
0
1
2
3

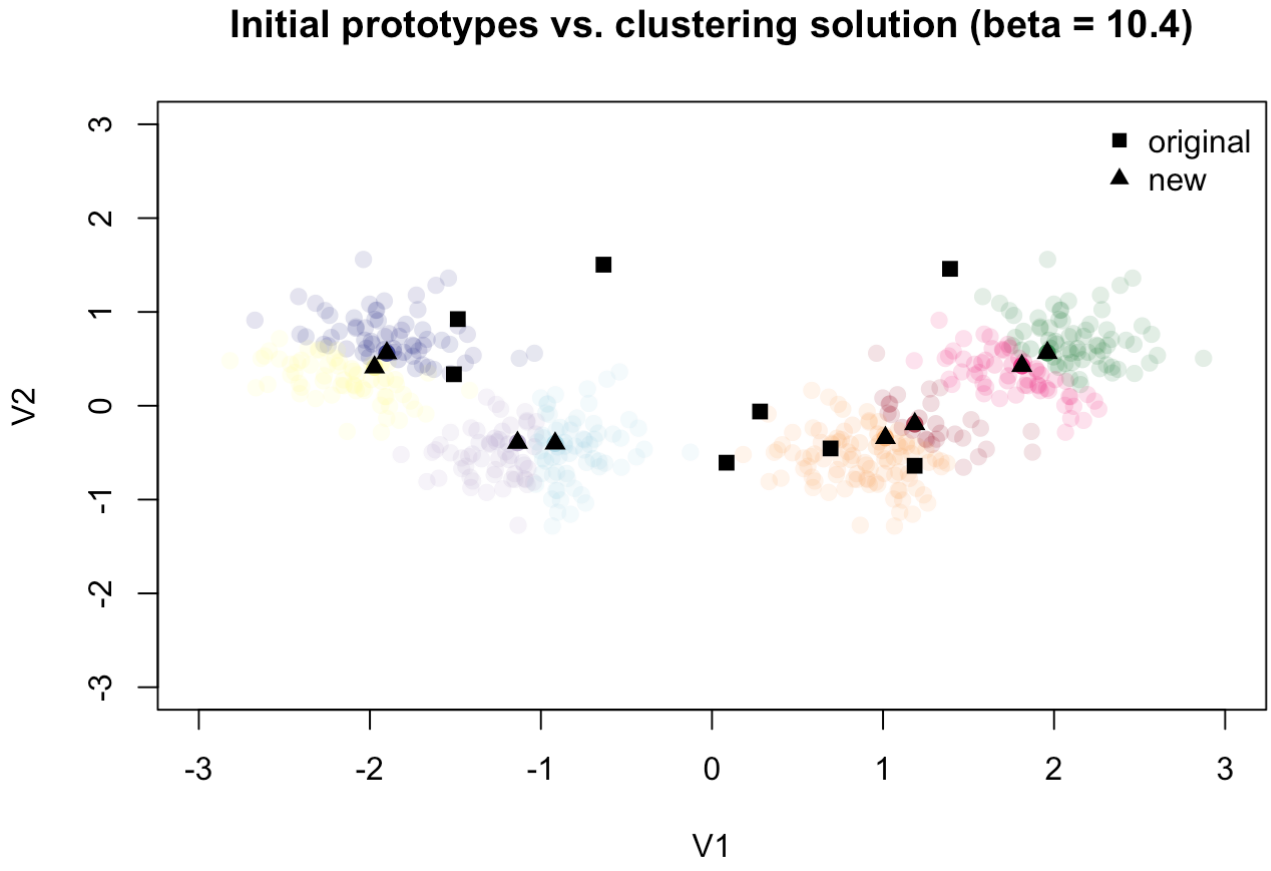

Initial prototypes vs. clustering solution (beta = 8.4)
original
new
V2
V1
-3
-2
-1
0
1
2
3

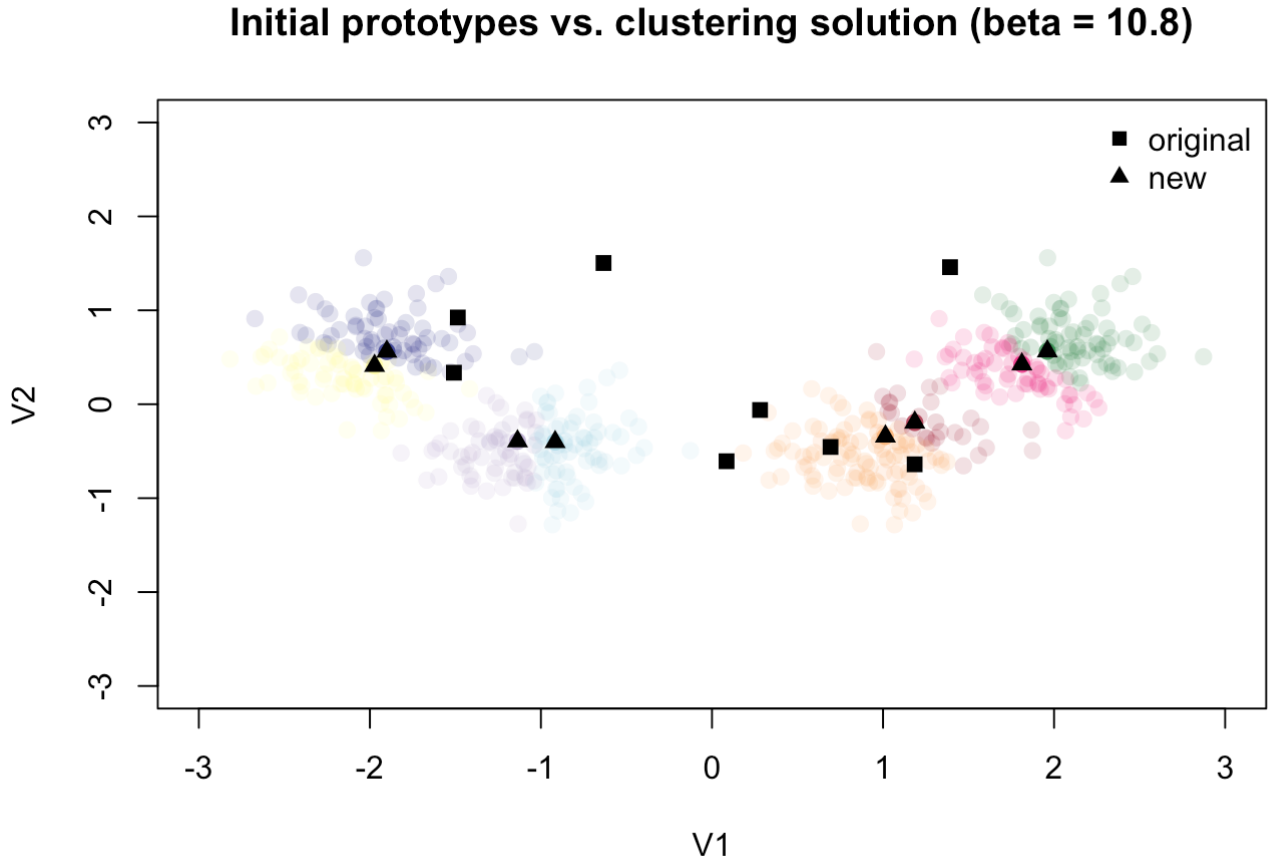

Initial prototypes vs. clustering solution (beta = 8.8)
original
new
V2
V1
-3
-2
-1
0
1
2
3

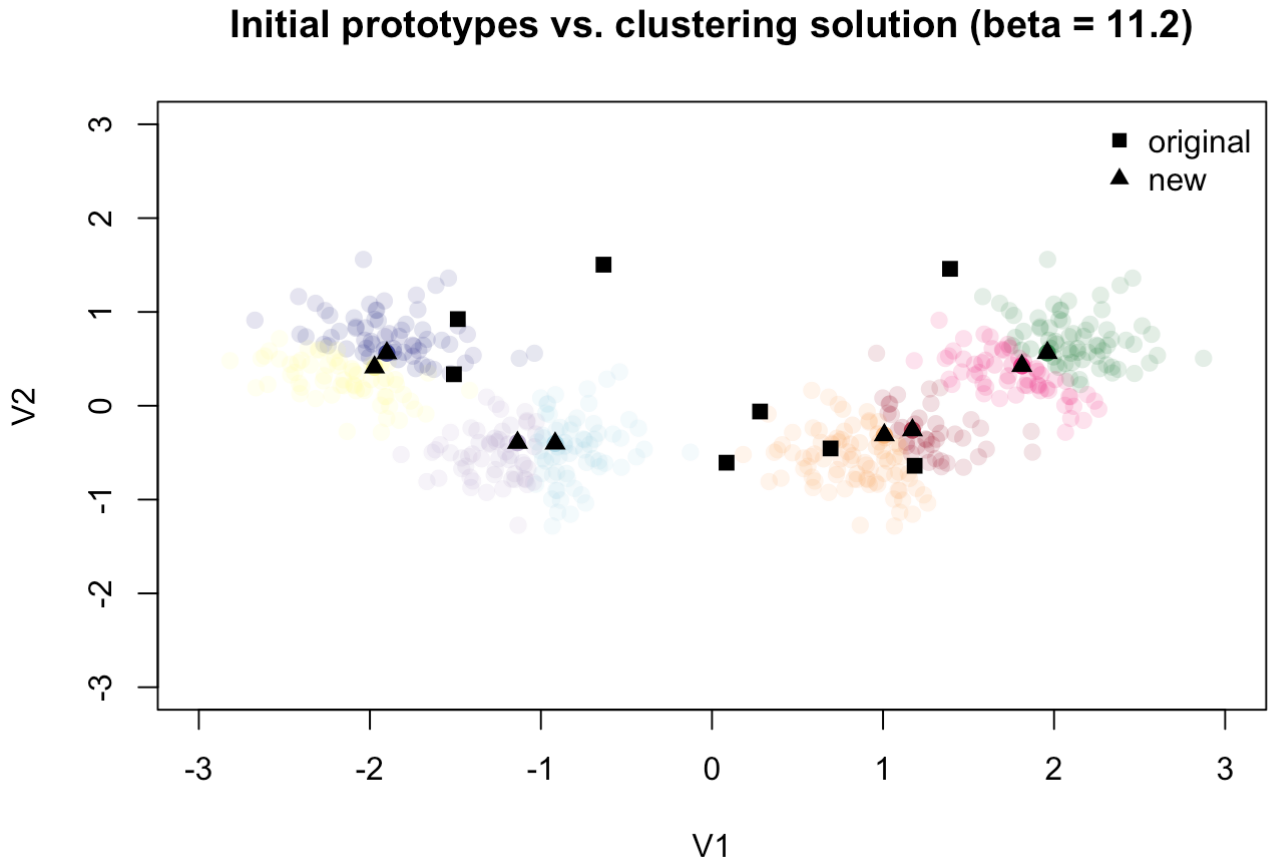

Initial prototypes vs. clustering solution (beta = 9.2)
original
new
V2
V1
-3
-2
-1
0
1
2
3

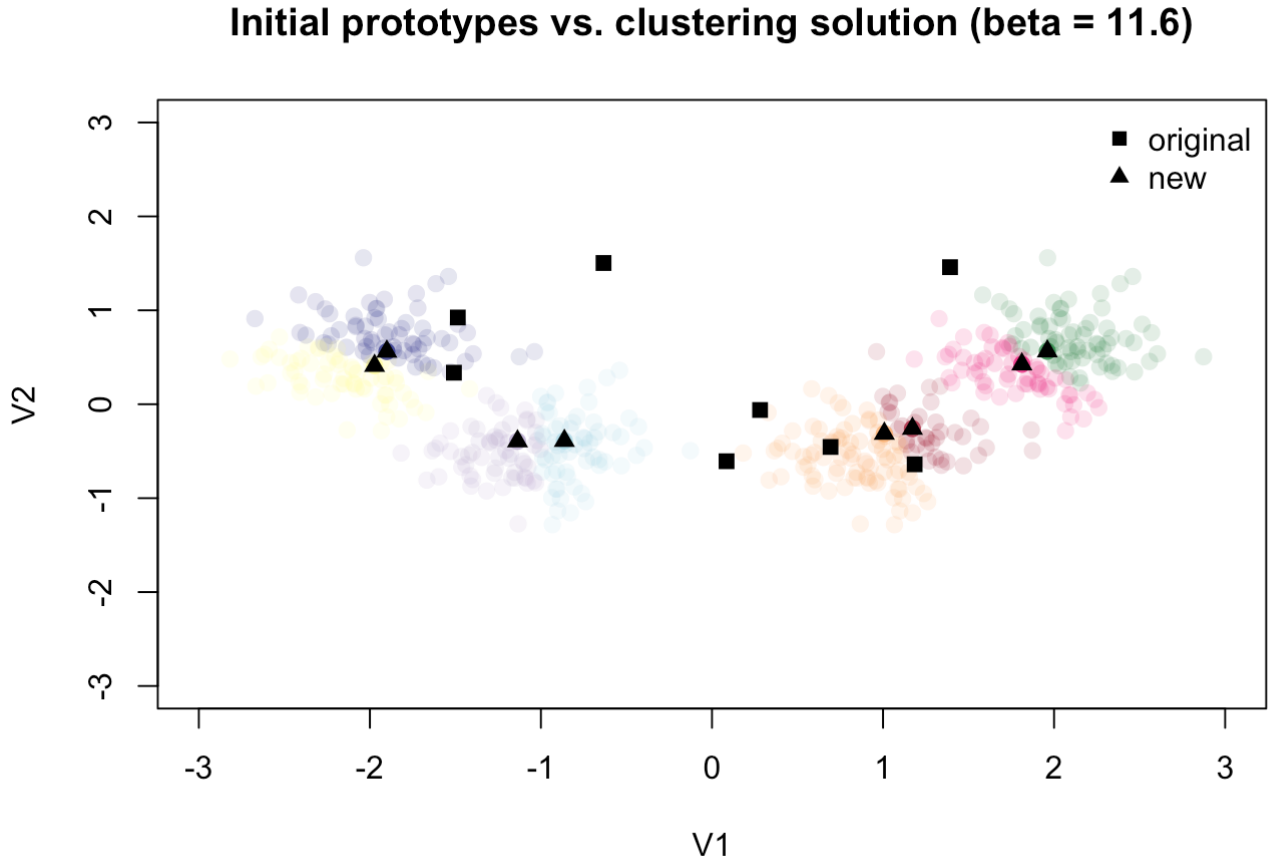

Initial prototypes vs. clustering solution (beta = 9.6)
original
new
V2
V1
-3
-2
-1
0
1
2
3

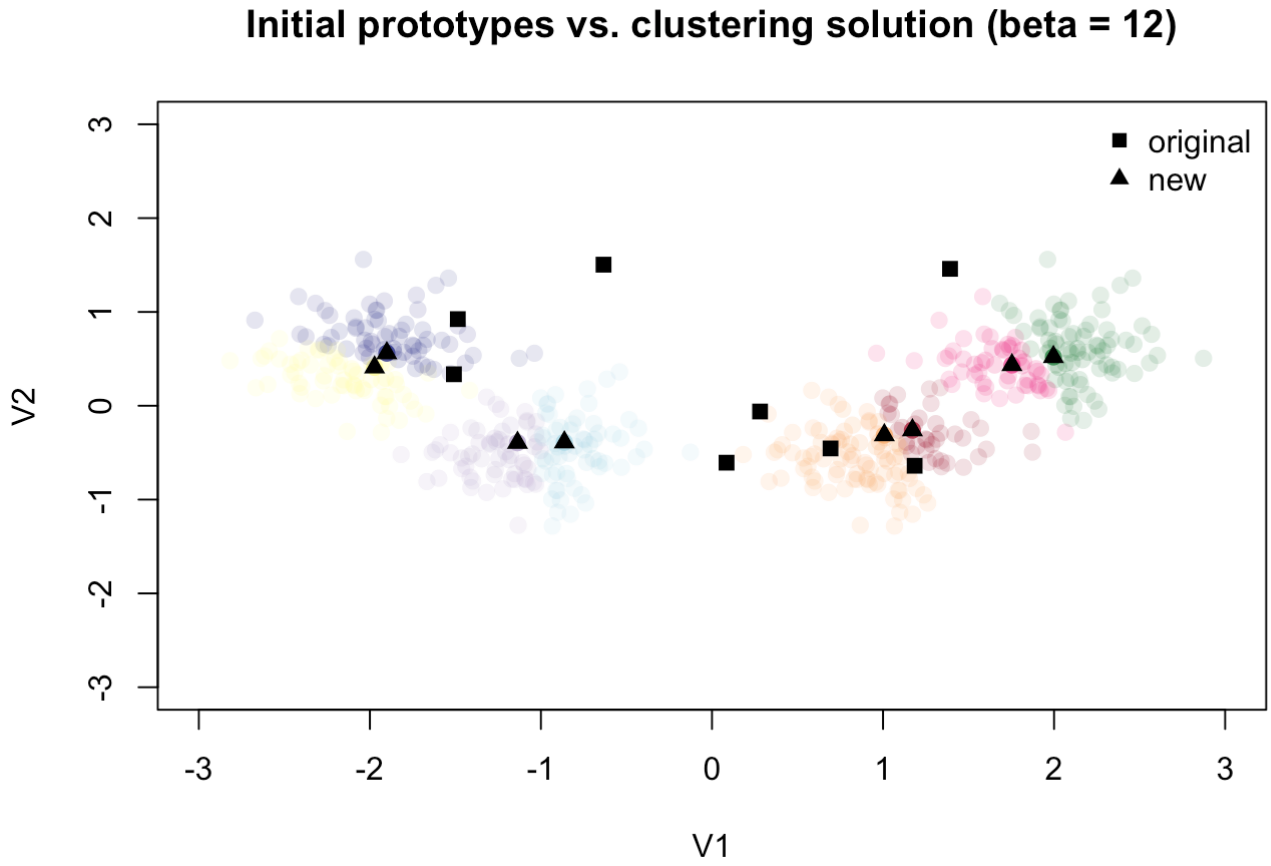

Initial prototypes vs. clustering solution (beta = 10)
original
new
V2
V1
-3
-2
-1
0
1
2
3

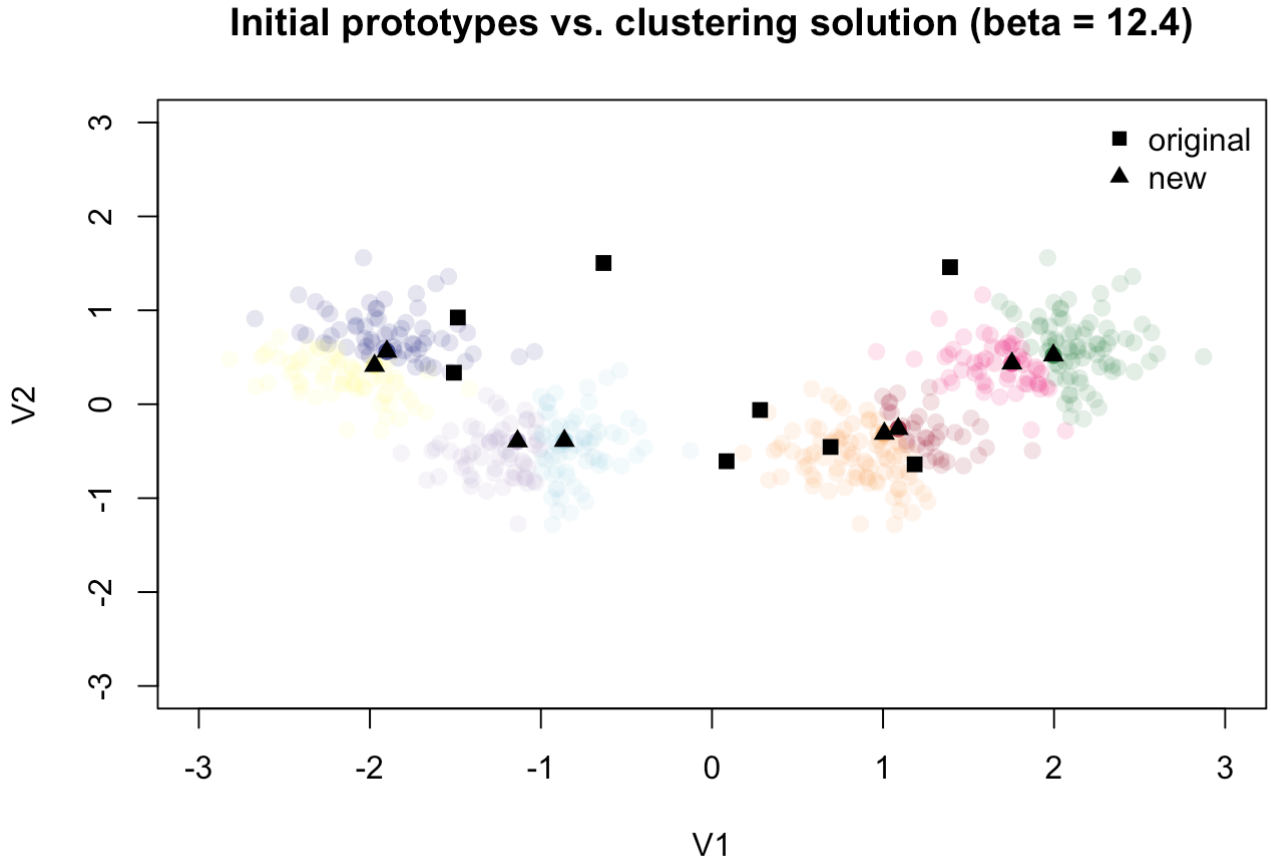

Initial prototypes vs. clustering solution (beta = 10.4)
original
new
V2
V1
-3
-2
-1
0
1
2
3

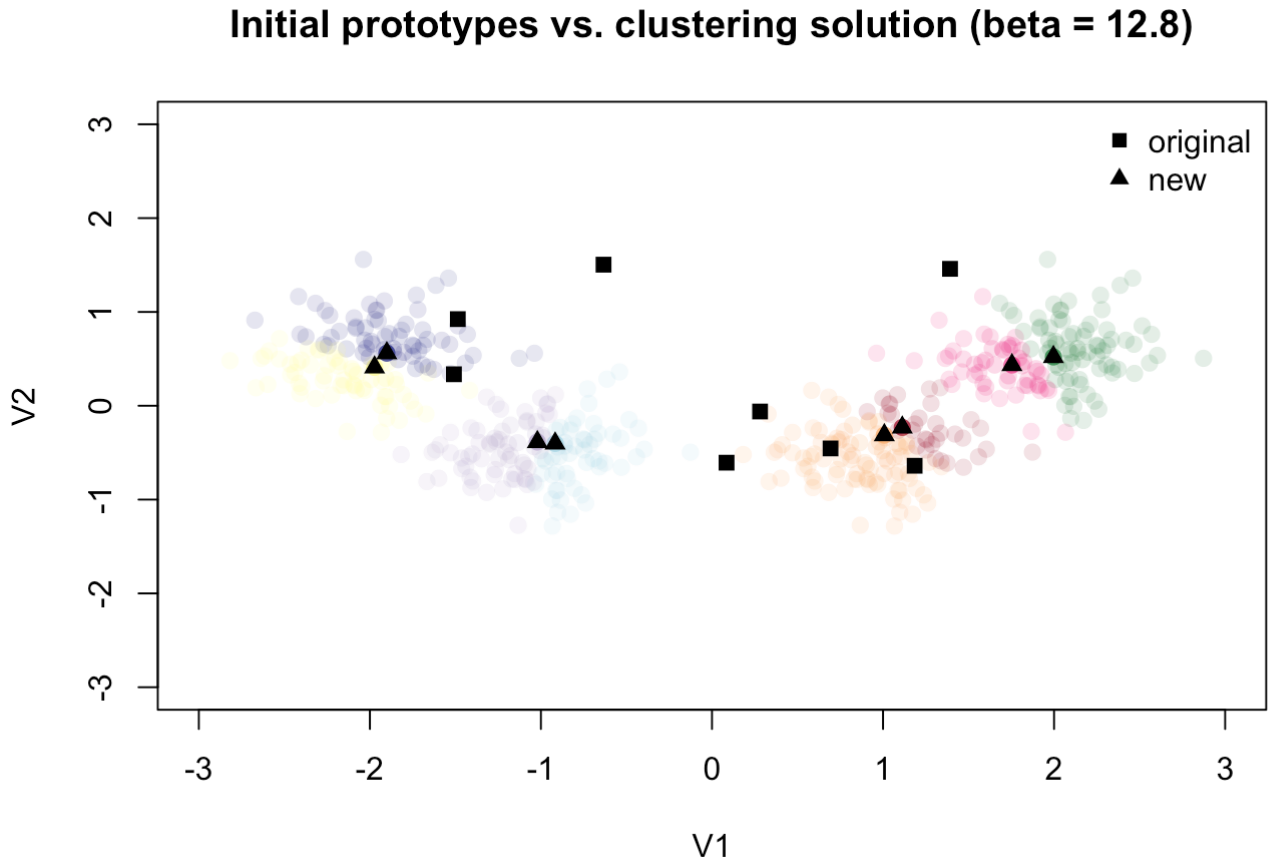

Initial prototypes vs. clustering solution (beta = 10.8)
original
new
V2
V1
-3
-2
-1
0
1
2
3

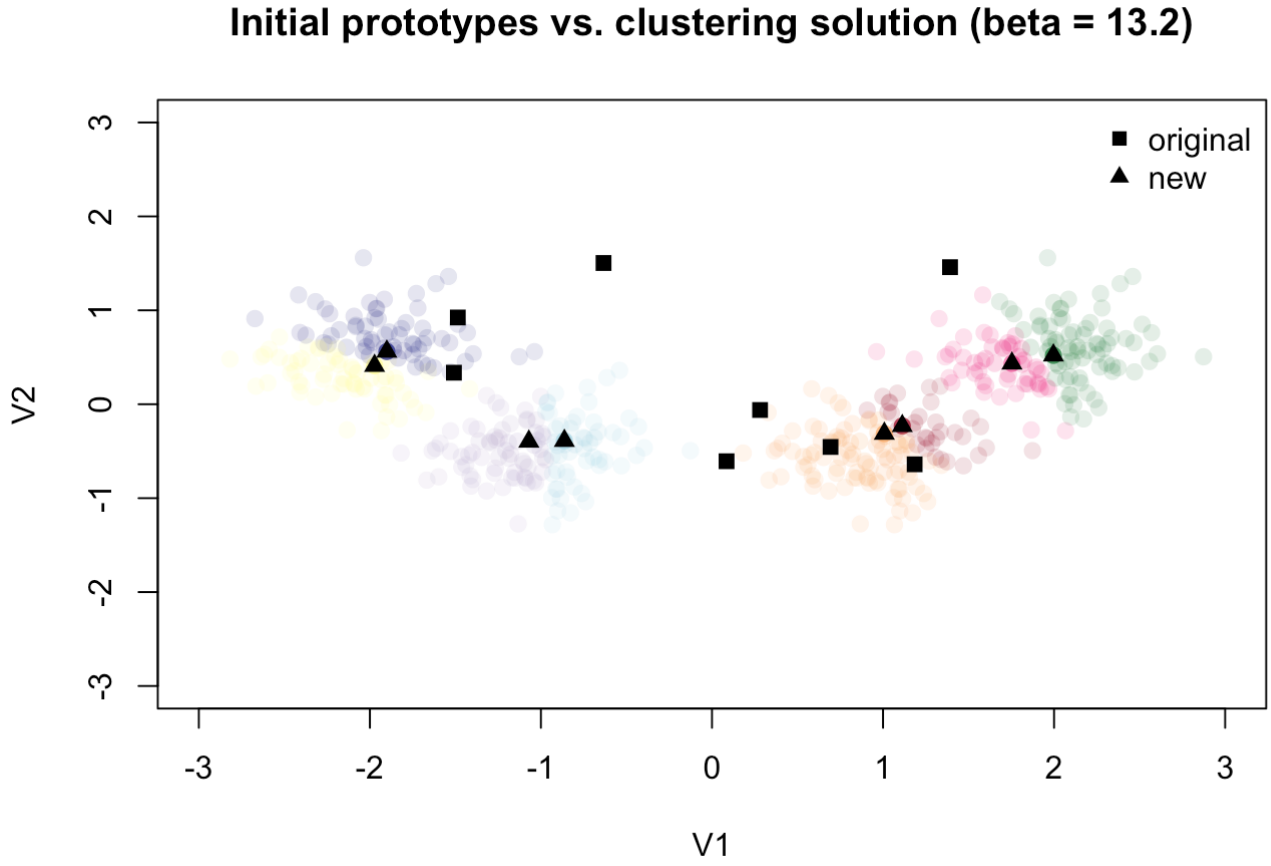

Initial prototypes vs. clustering solution (beta = 11.2)
original
new
V2
V1
-3
-2
-1
0
1
2
3

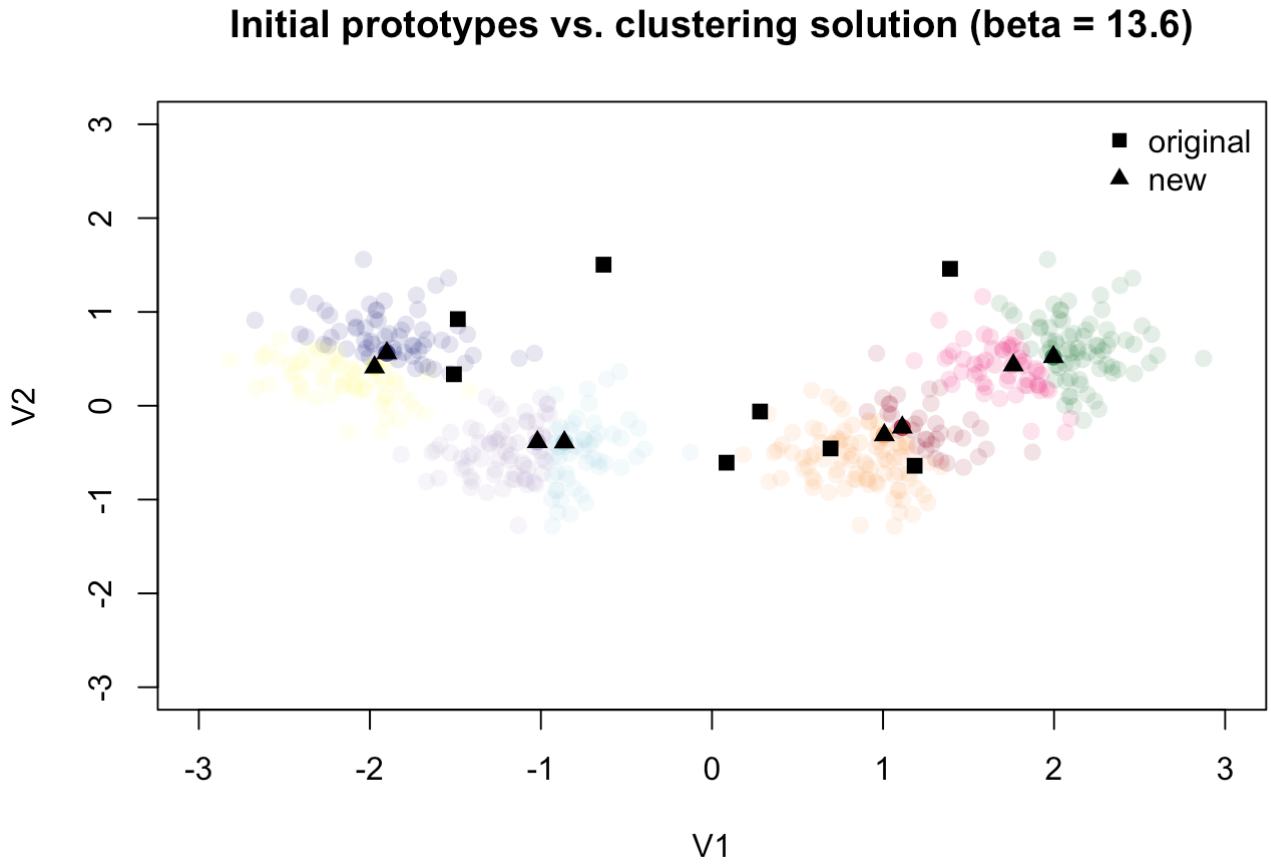

Initial prototypes vs. clustering solution (beta = 11.6)
original
new
V2
V1
-3
-2
-1
0
1
2
3

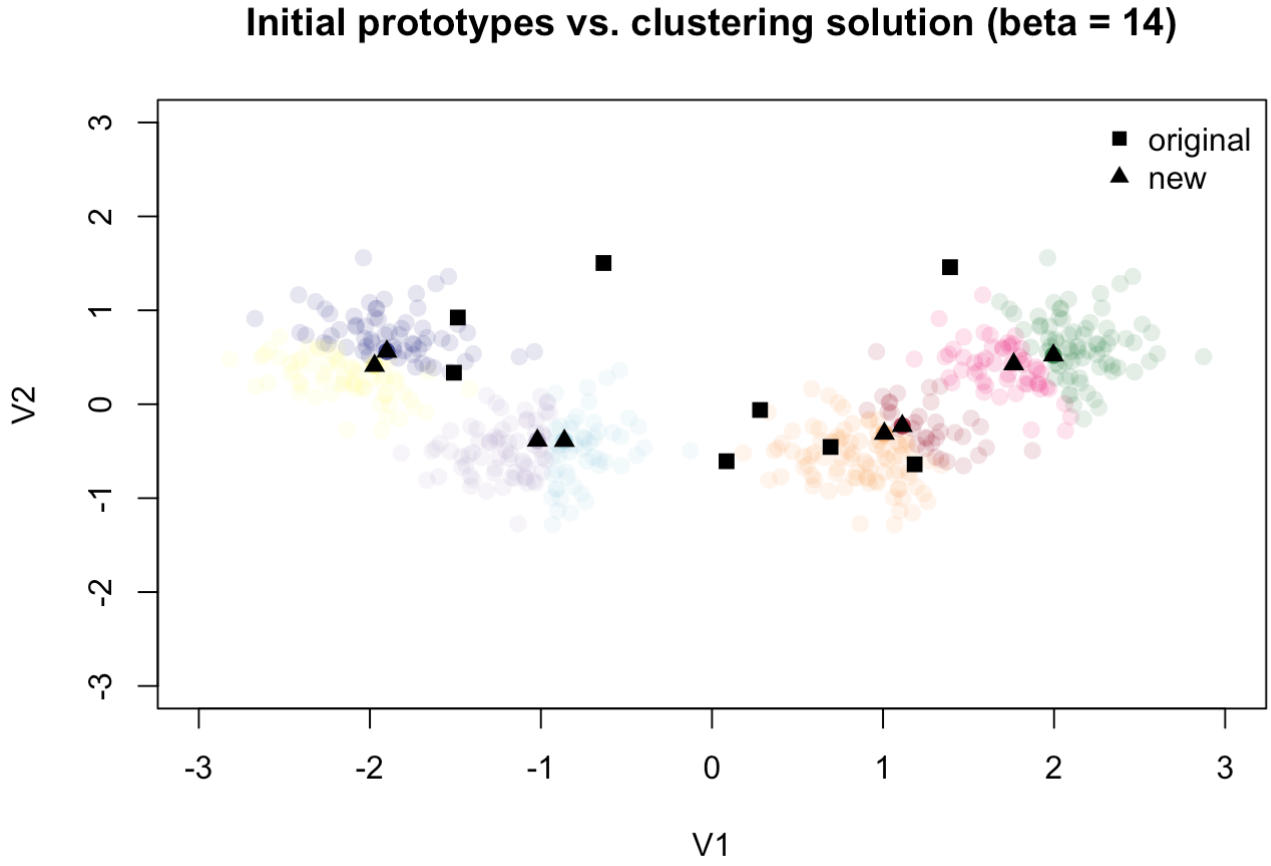

Initial prototypes vs. clustering solution (beta = 12)
original
new
V2
V1
-3
-2
-1
0
1
2
3

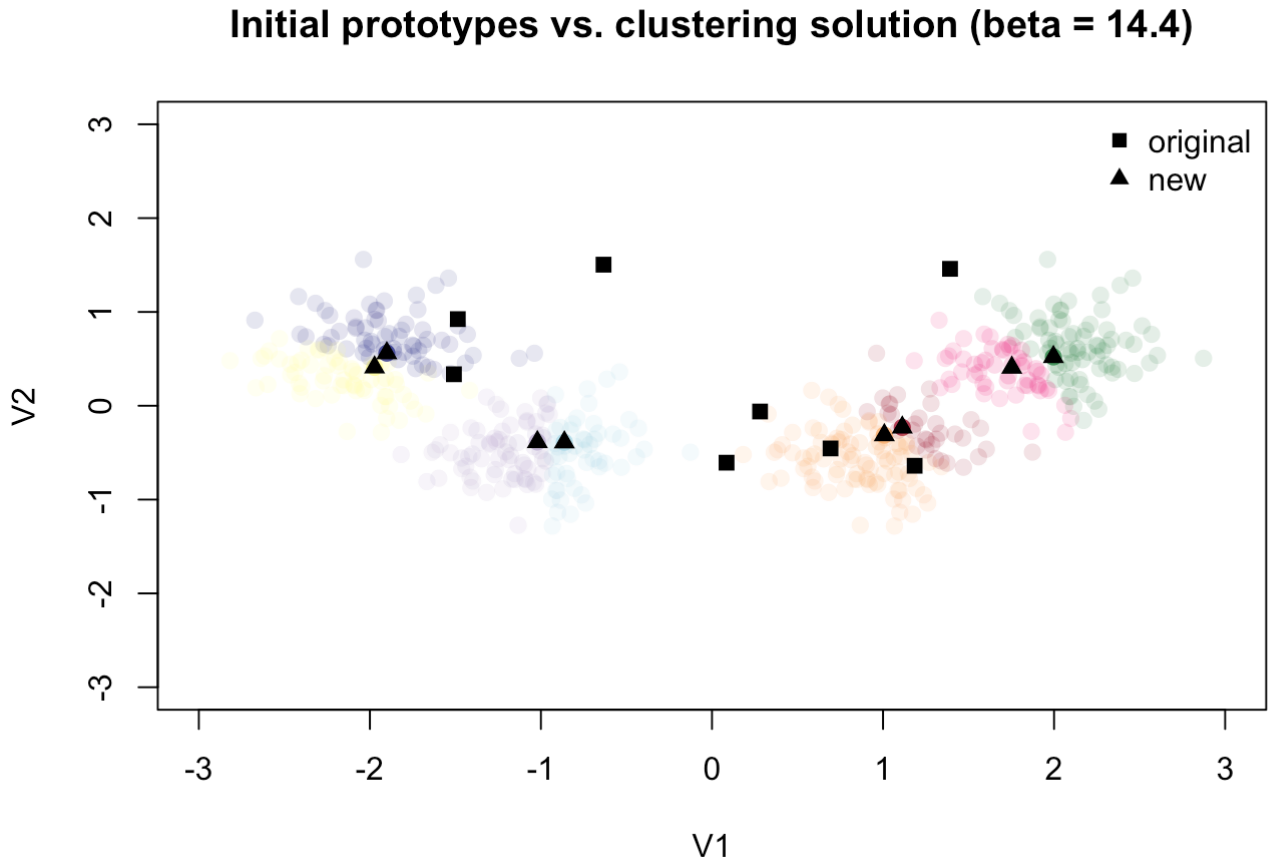

Initial prototypes vs. clustering solution (beta = 12.4)
original
new
V2
V1
-3
-2
-1
0
1
2
3

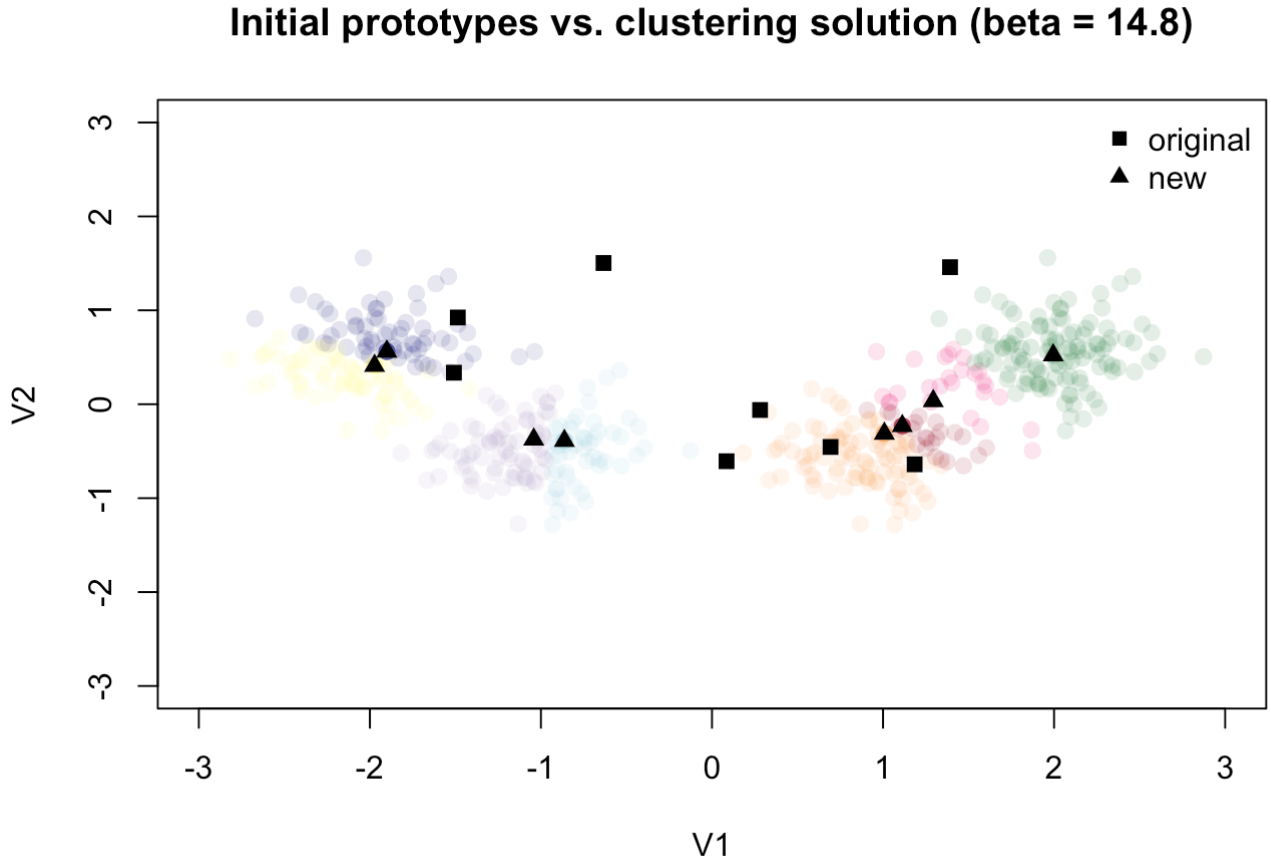

Initial prototypes vs. clustering solution (beta = 12.8)
original
new
V2
V1
-3
-2
-1
0
1
2
3

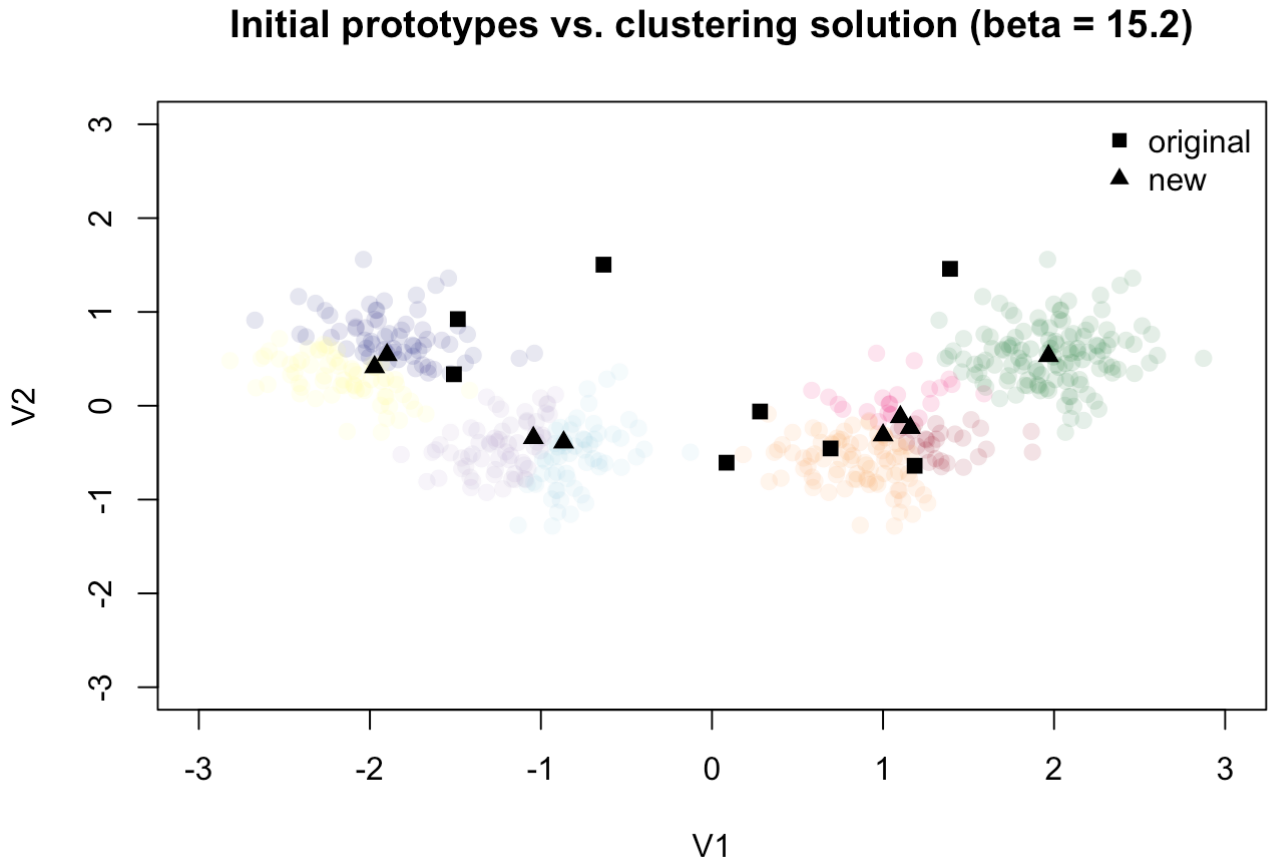

Initial prototypes vs. clustering solution (beta = 13.2)
original
new
V2
V1
-3
-2
-1
0
1
2
3

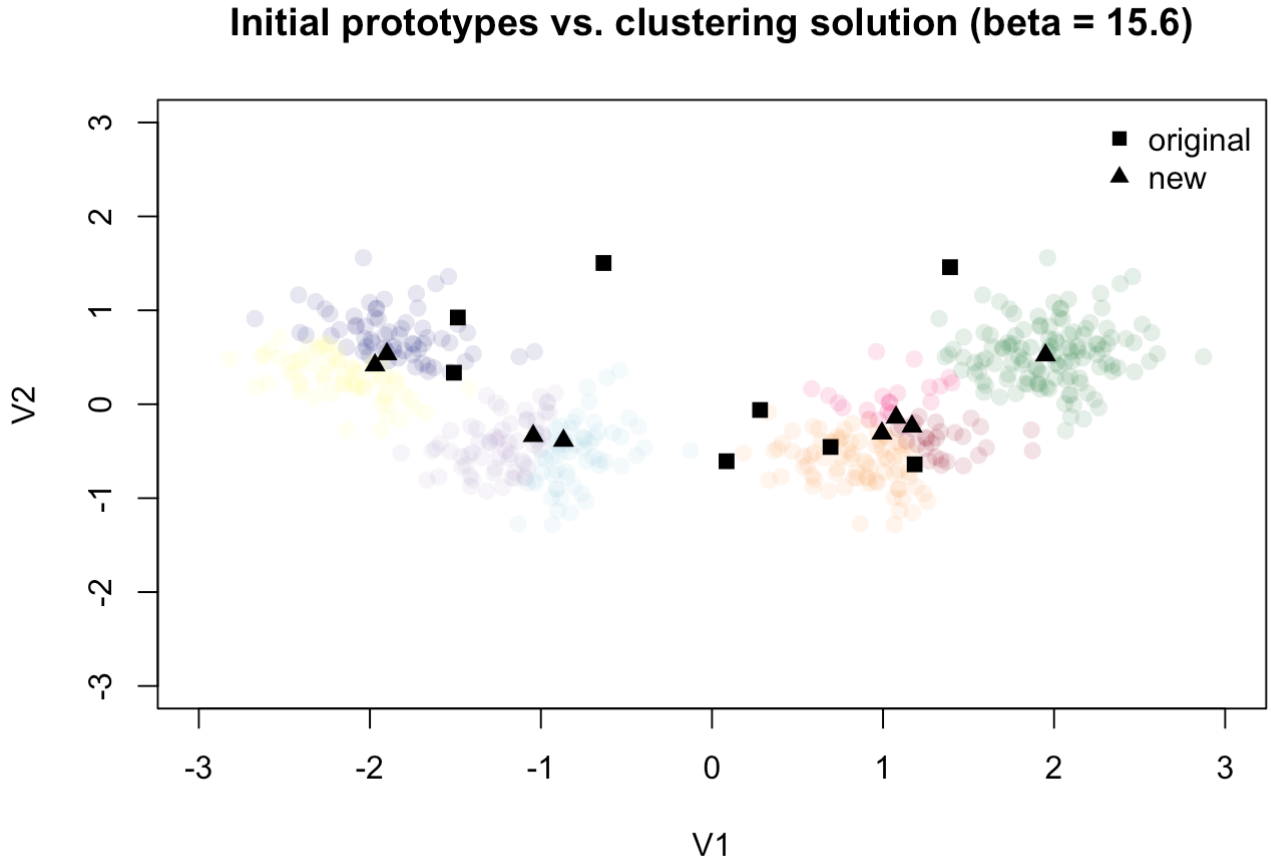

Initial prototypes vs. clustering solution (beta = 13.6)
original
new
V2
V1
-3
-2
-1
0
1
2
3

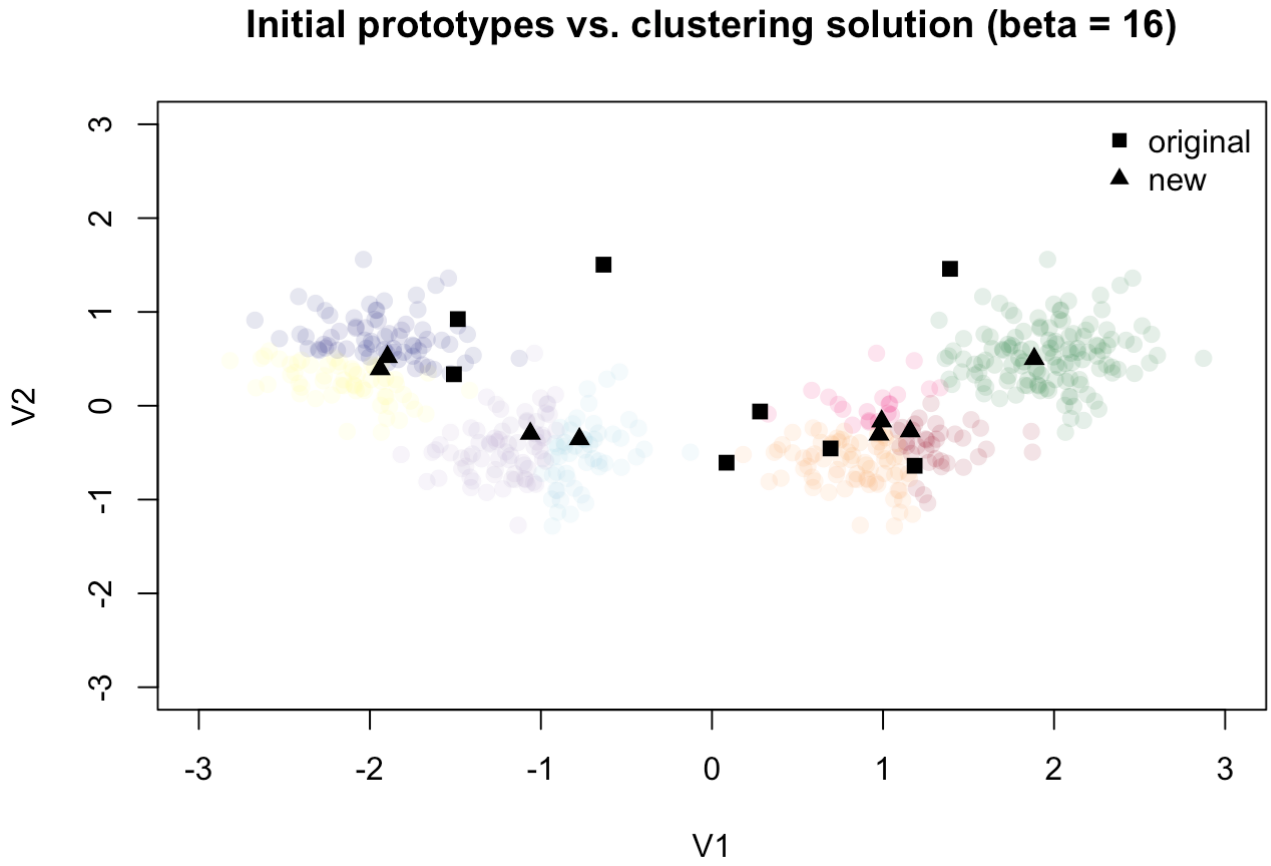

Initial prototypes vs. clustering solution (beta = 14)
original
new
V2
V1
-3
-2
-1
0
1
2
3

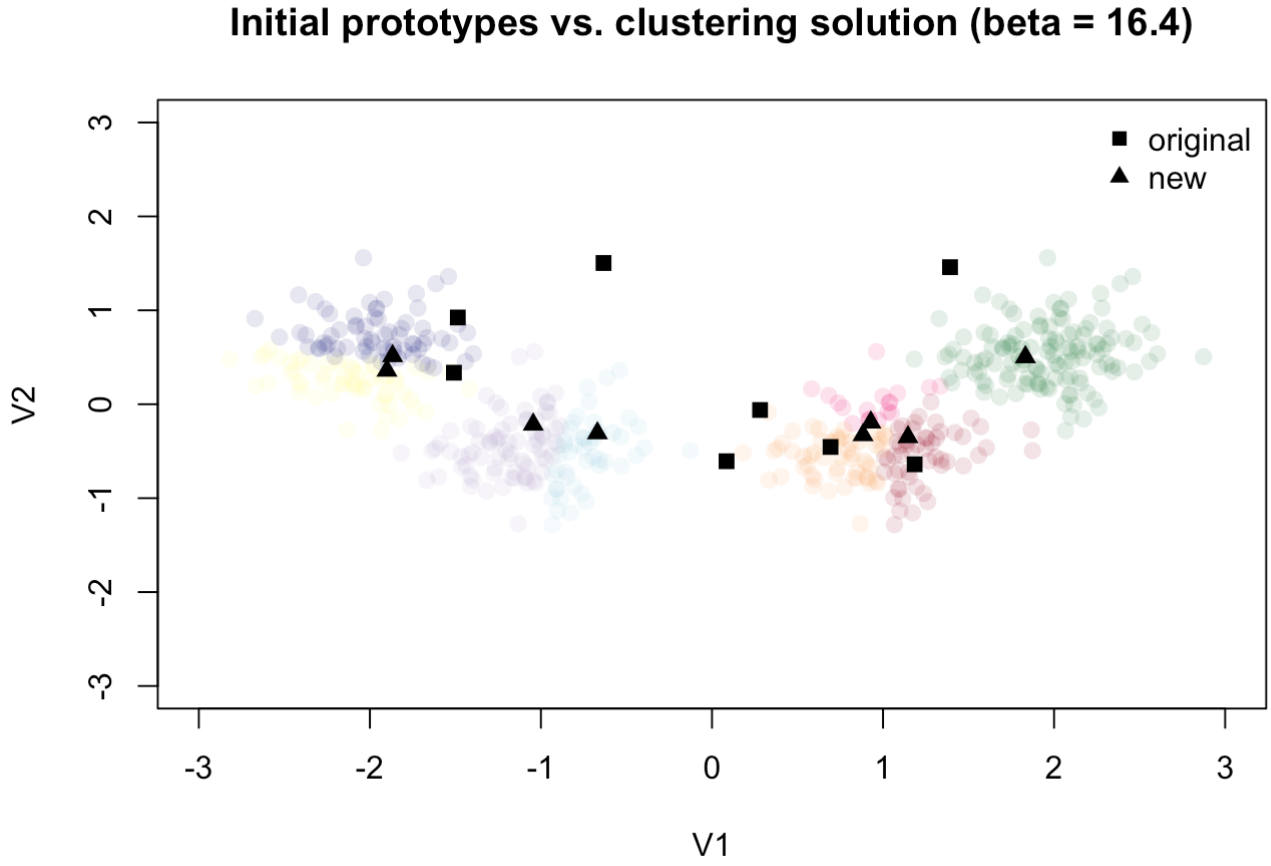

Initial prototypes vs. clustering solution (beta = 14.4)
original
new
V2
V1
-3
-2
-1
0
1
2
3

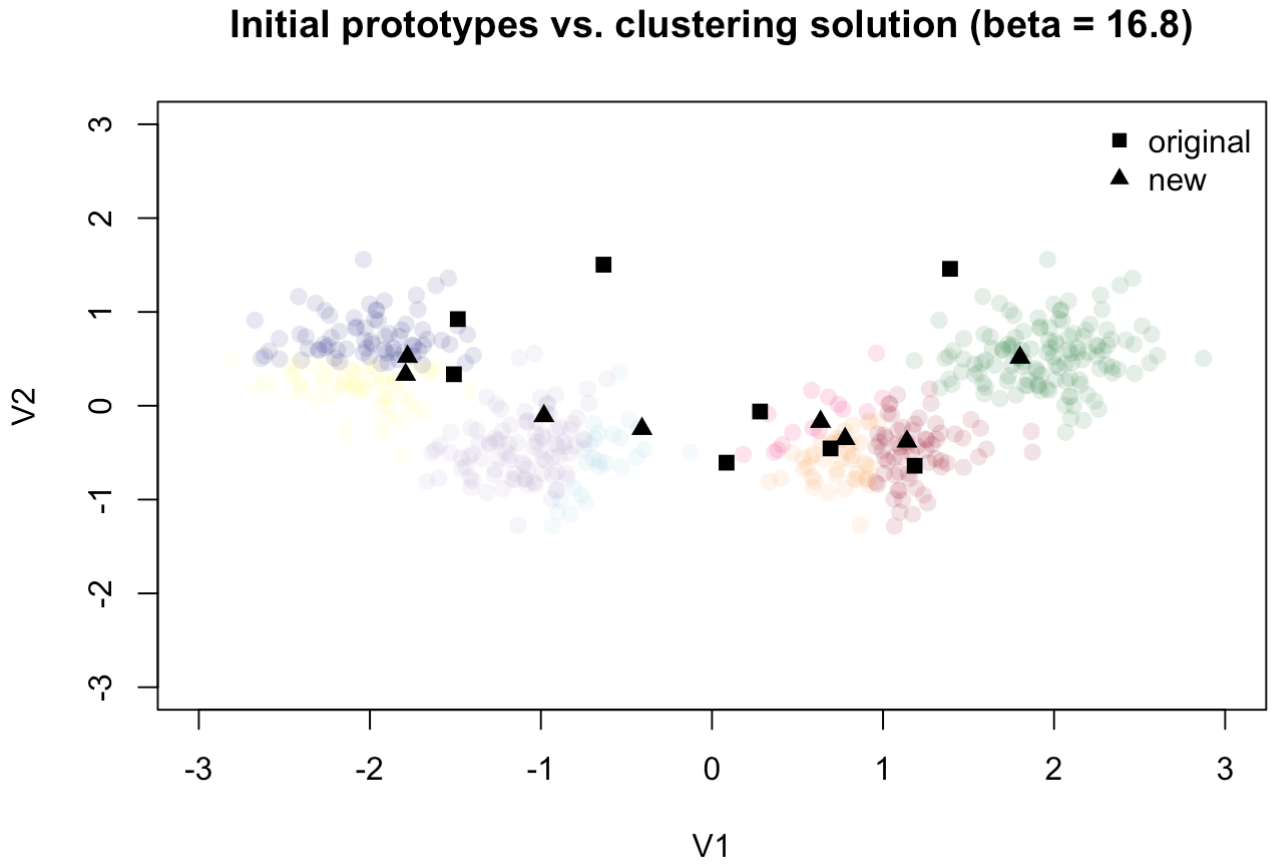

Initial prototypes vs. clustering solution (beta = 14.8)
original
new
V2
V1
-3
-2
-1
0
1
2
3

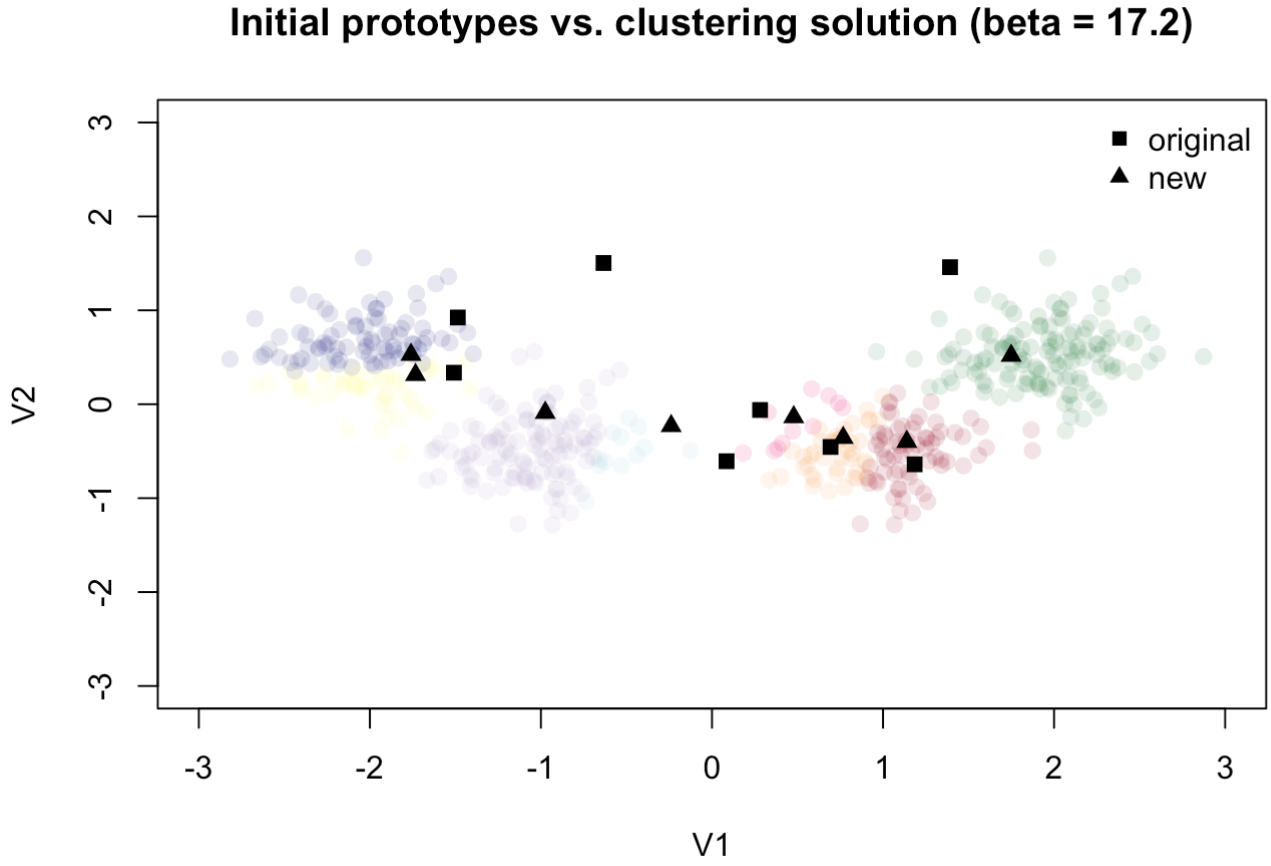

Initial prototypes vs. clustering solution (beta = 15.2)
original
new
V2
V1
-3
-2
-1
0
1
2
3

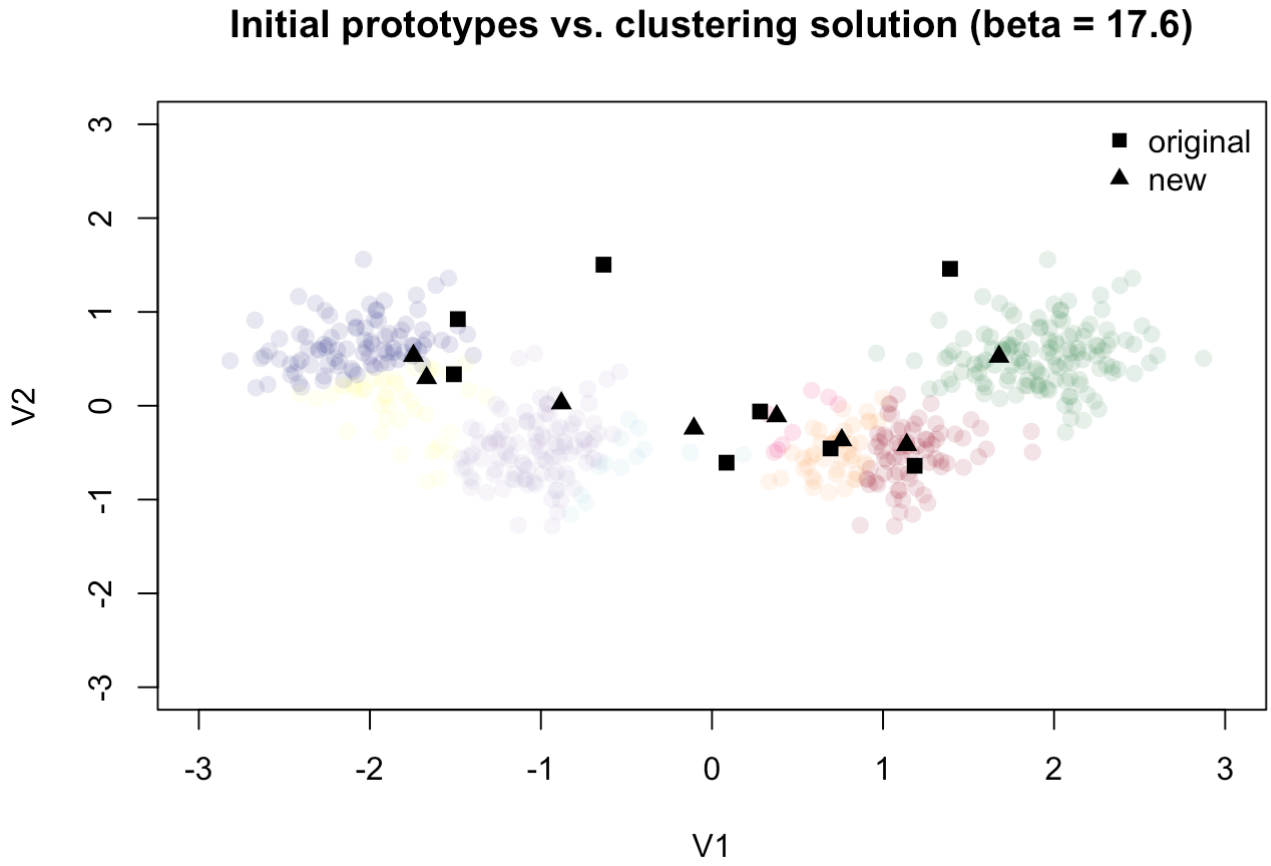

Initial prototypes vs. clustering solution (beta = 15.6)
original
new
V2
V1
-3
-2
-1
0
1
2
3

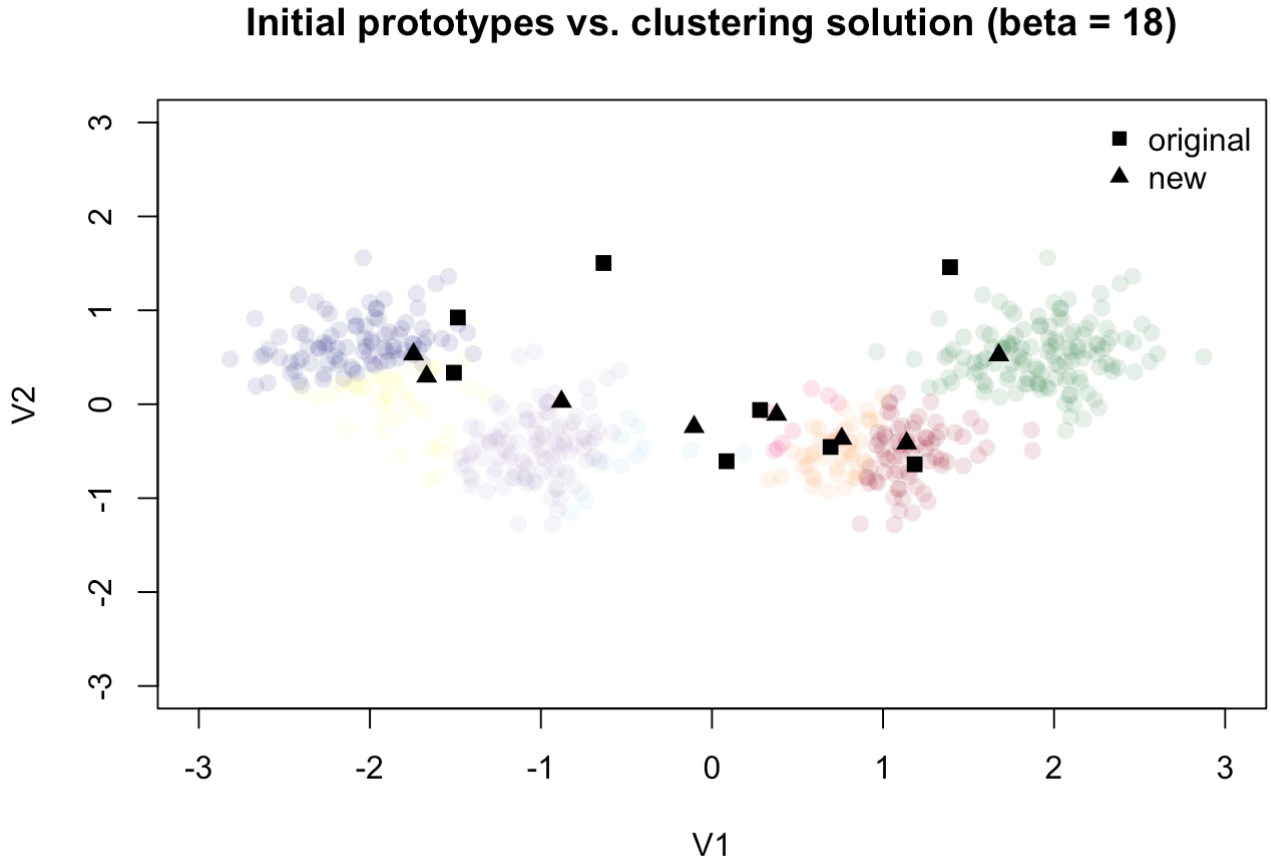

Initial prototypes vs. clustering solution (beta = 16)
original
new
V2
V1
-3
-2
-1
0
1
2
3

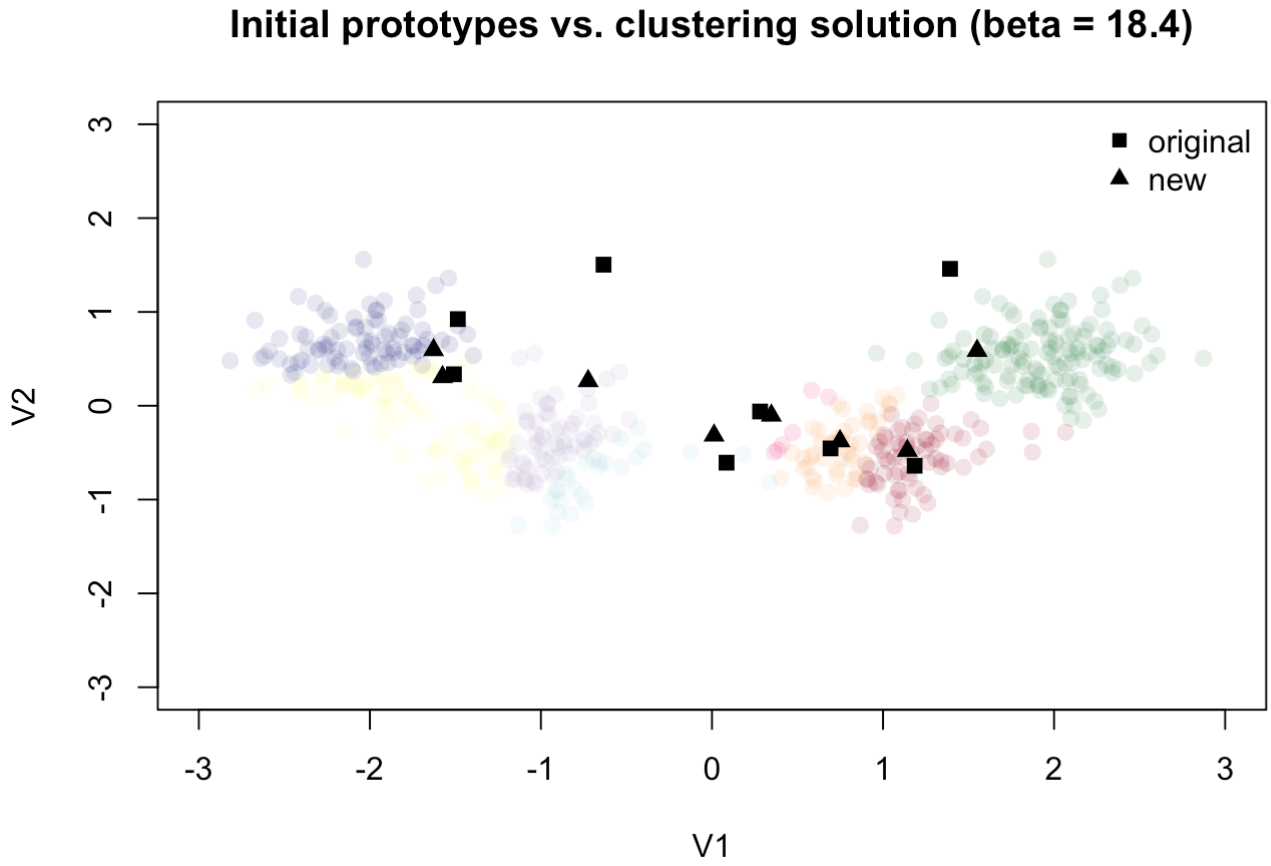

Initial prototypes vs. clustering solution (beta = 16.4)
original
new
V2
V1
-3
-2
-1
0
1
2
3

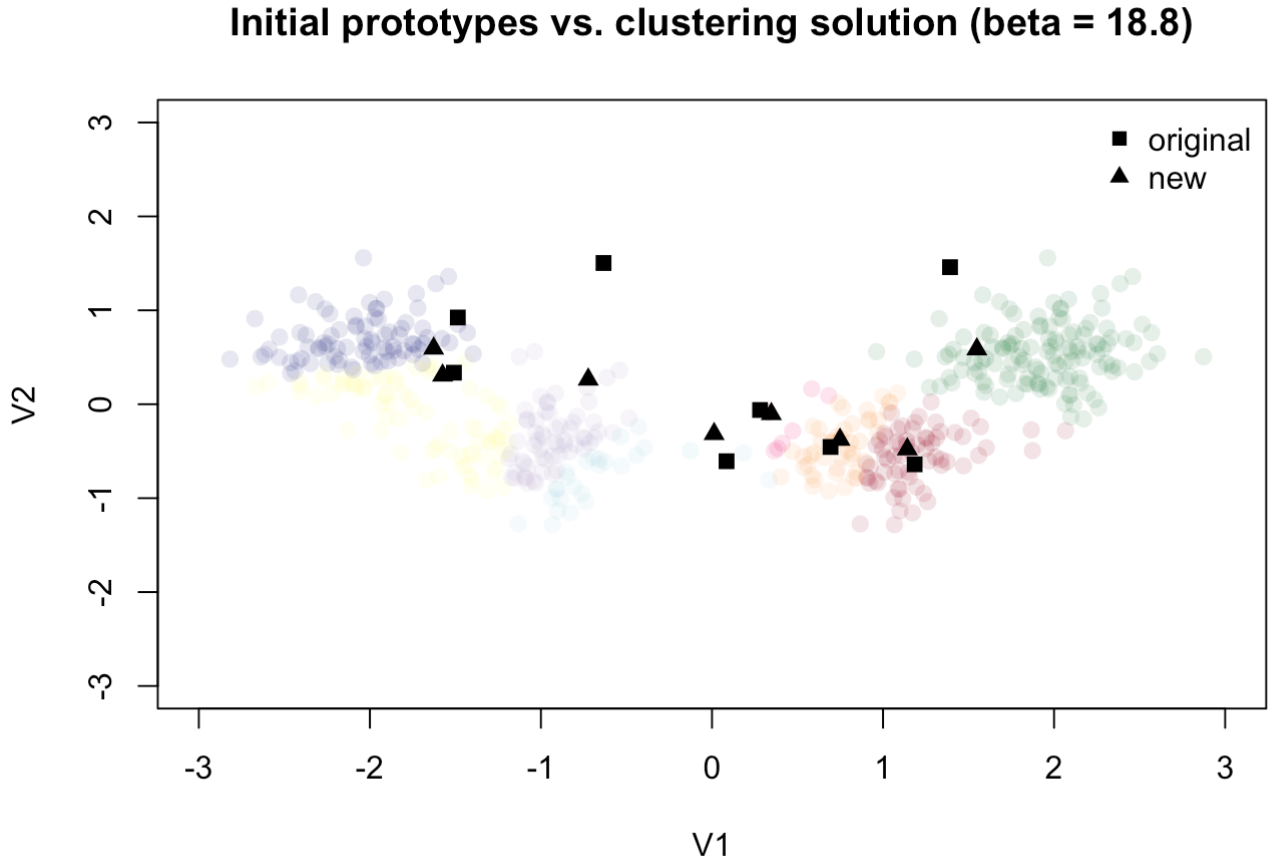

Initial prototypes vs. clustering solution (beta = 16.8)
original
new
V2
V1
-3
-2
-1
0
1
2
3

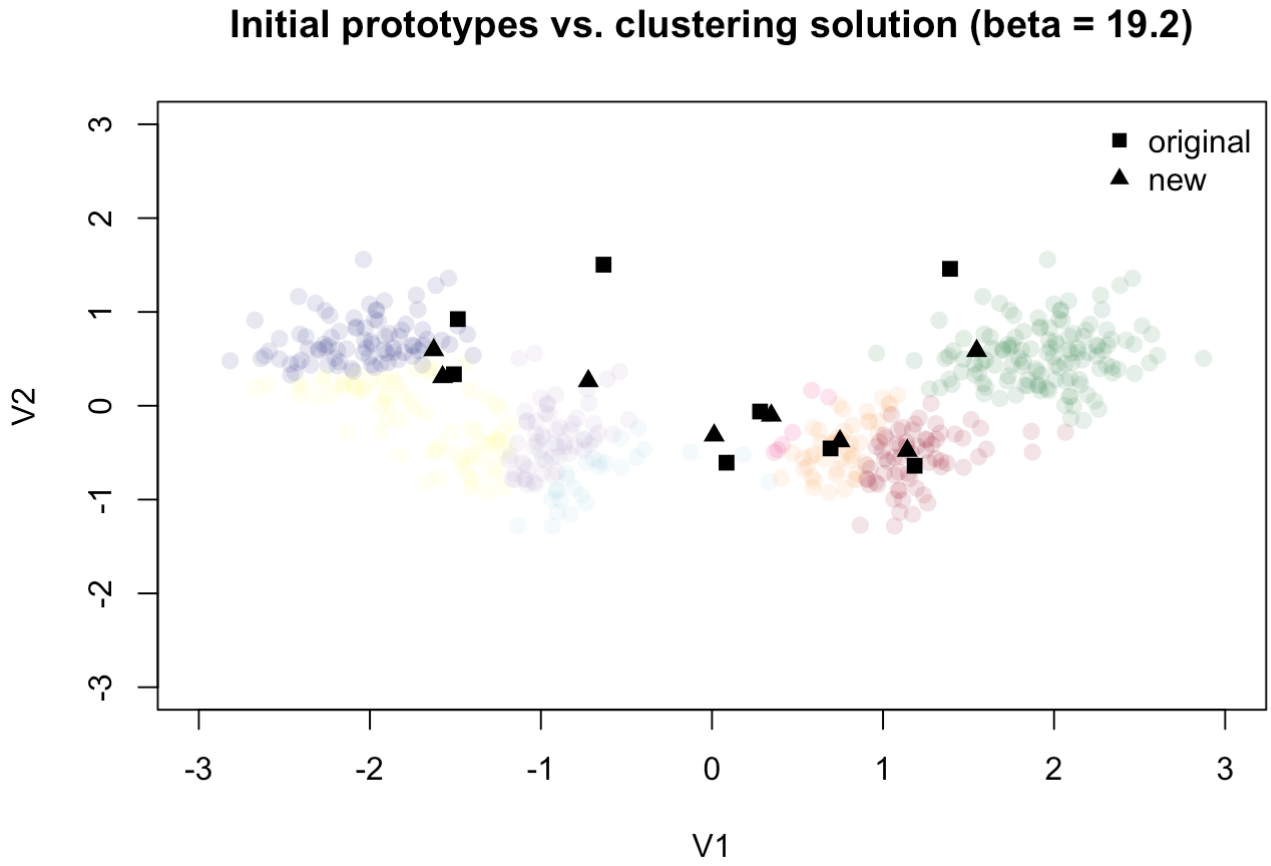

Initial prototypes vs. clustering solution (beta = 17.2)
original
new
V2
V1
-3
-2
-1
0
1
2
3

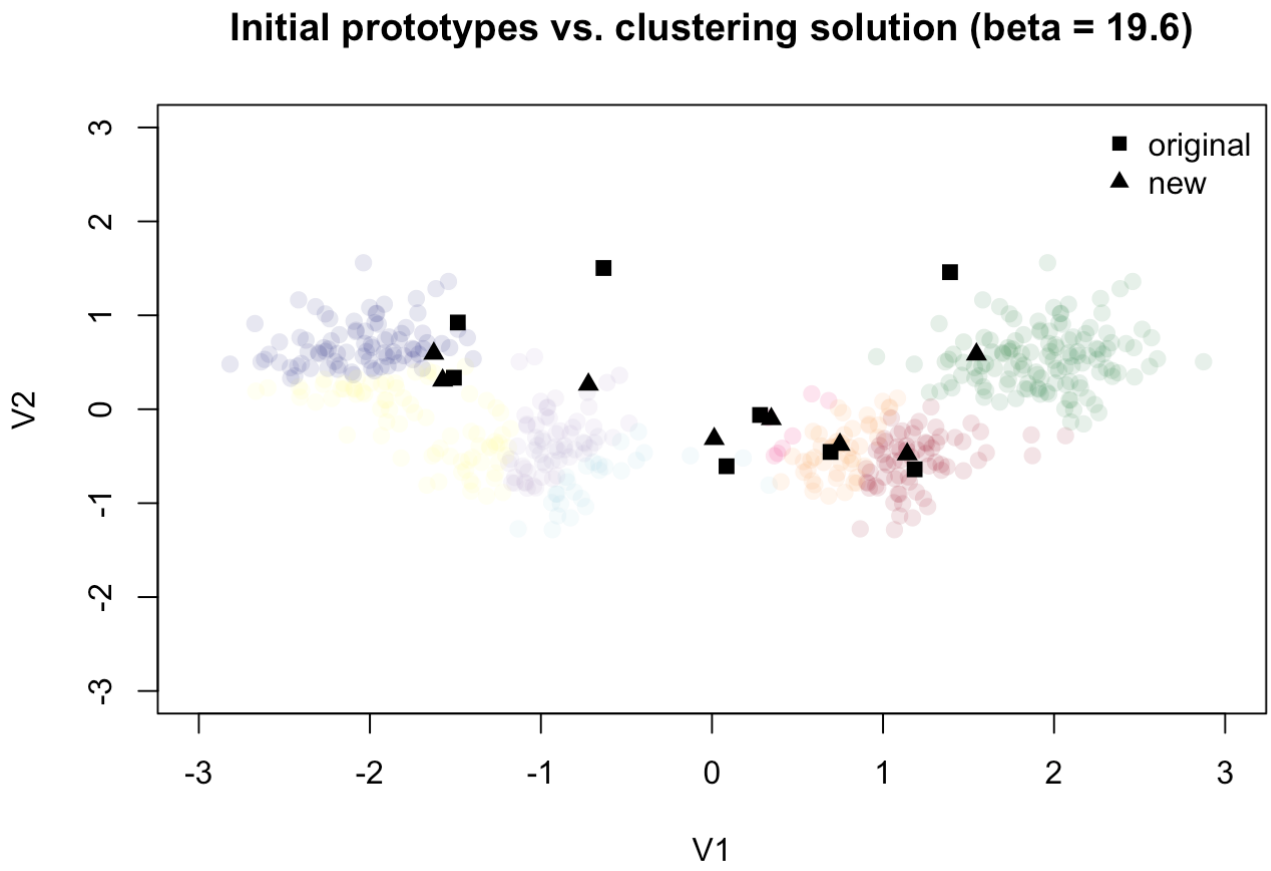

Initial prototypes vs. clustering solution (beta = 17.6)
original
new
V2
V1
-3
-2
-1
0
1
2
3

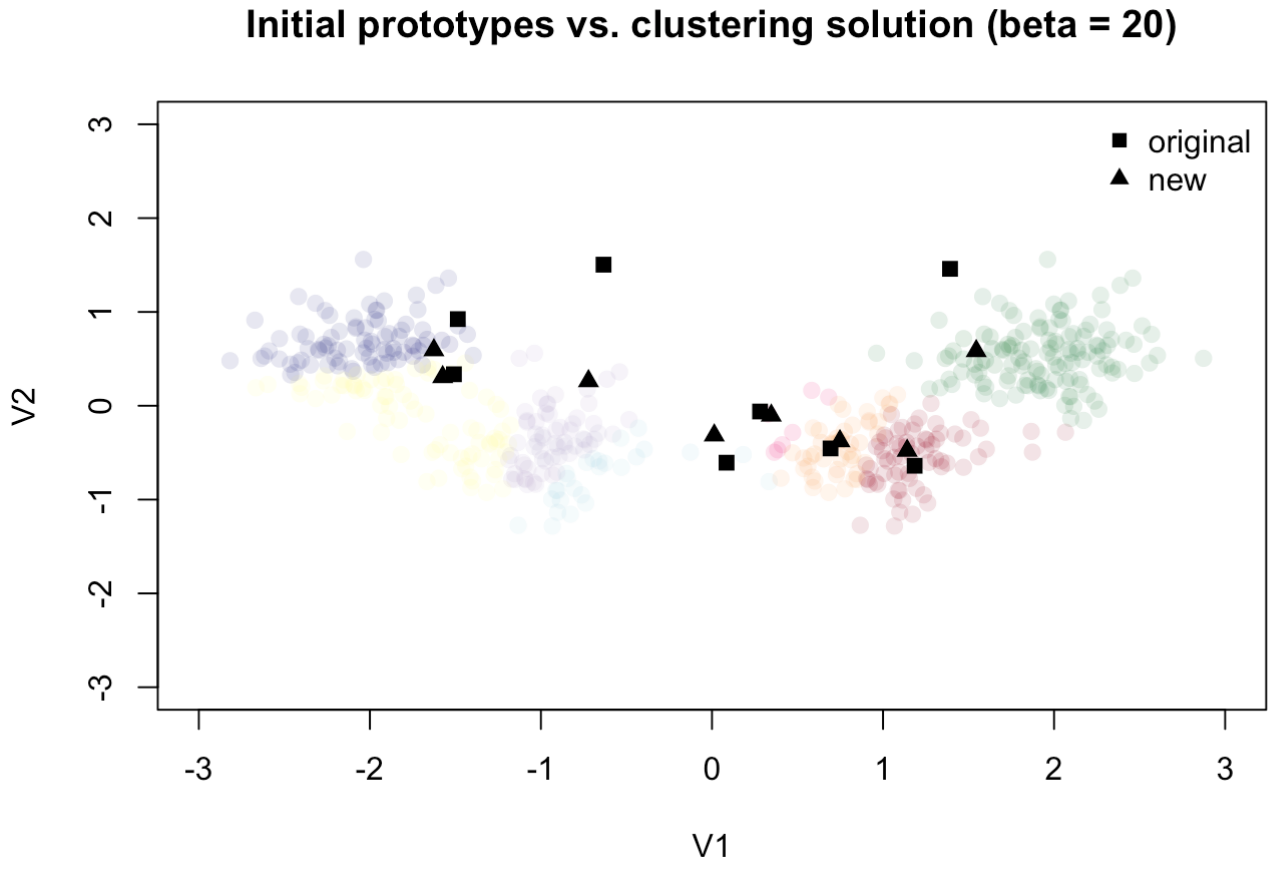

Initial prototypes vs. clustering solution (beta = 18)
original
new
V2
V1
-3
-2
-1
0
1
2
3


Initial prototypes vs. clustering solution (beta = 18.4)
original
new
V2
V1
-3
-2
-1
0
1
2
3


Initial prototypes vs. clustering solution (beta = 18.8)
original
new
V2
V1
-3
-2
-1
0
1
2
3


Initial prototypes vs. clustering solution (beta = 19.2)
original
new
V2
V1
-3
-2
-1
0
1
2
3






```
par(mfrow = c(1, 1))
```