COVID-19 Project: RSKC Analysis for Gene Expression

Rachel Kwan and Jonathan Zaslavsky

2021-06-02

Relevant Packages

```
# Load packages
library(here) # To read in data from directory
library(tidyverse) # For ggplot2, dplyr
library(magrittr) # For set_colnames() and set_rownames()
library(ggpubr) # For making publication-ready plots based on ggplot
library(RSKC) # For RSKC clustering
library(Rtsne) # To run t-SNE (dimensionality reduction)

# Set the seed
set.seed(72613)
```

Import and Prepare Dataset

```
# Load full dataset
all_expression <- read.csv(here("Data", "COVID Data - 28 Genes - 1281 Samples from 16 Regions - No Trans

# Create a vector containing all of the names of the genes of interest.
genes <- as_vector(colnames(all_expression)[4:31])

# Create a data frame grouping each observation by the brain region and
# computing the average expression of each gene within a given brain region.
gene_expression <- all_expression %>%
  group_by(Region) %>%
  summarize_at(vars(all_of(genes)), mean) %>%
  rename(Brain.Region = Region)

# Use the brain regions to name the rows and remove the brain region column.
myExpression <- gene_expression %>%
  column_to_rownames("Brain.Region")
```

Perform Robust and Sparse K-Means Clustering (RSKC) and t-SNE Together

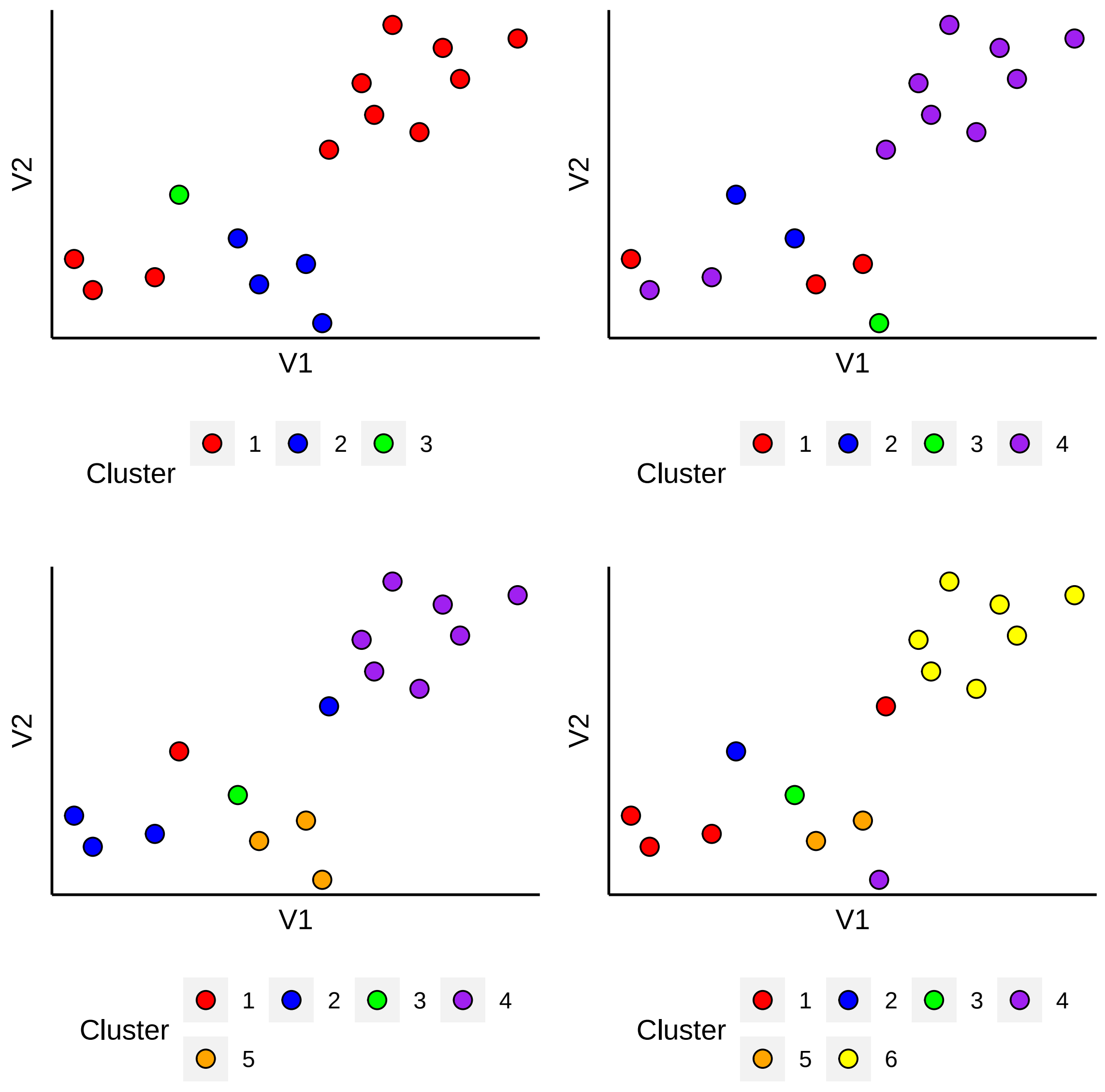
This while loop contains sections for RSKC, elbow plot, obtaining weighted data, and tSNE. Desmond's code was used as template for this while loop, in particular, for the RSKC and tSNE sections.

```r
set.seed(72613)

while (T) {

  # Assign desired number of clusters to 'clust_vect'.
  clust_vect <- c(3,4,5,6)

  # Assign an empty list to 'rskc_list'.
  rskc_list <- list()

  # Assign 6 colours to 'col_vect'.
  col_vect <- c("#FF0000",
               "#0000FF",
               "#00FF00",
               "#A020F0",
               "#FFA500",
               "#FFFF00")

  # Assign a value of 0 to 'counter'.
  counter <- 0

  # Assign an empty list to 'tsne_list'.
  tsne_list <- list()

  # Assign an empty list to 'weight_list'.
  weight_list <- list()

  # For 'i' -- the current number of clusters -- in 'clust_vect'...
  for (i in clust_vect) {
    # i = 3
    # Increment 'counter' with a value of 1.
    counter = counter + 1

    ###### RSKC ######

    # Perform RSKC for whatever-the-value-of-'i'-is many clusters using
    # 'myExpression', which has brain region as rows and gene_celltype as columns.
    # Assign RSKC's output as an entry in 'rskc_list'.
    rskc_list[[counter]] <- RSKC(myExpression,
                                 ncl = i,
                                 alpha = 0.1,
                                 L1 = sqrt(ncol(myExpression)))

    # Convert the row names of 'myExpression' to a column
    # called 'Brain.Region' and store it in 'gene_and_region'.
    gene_and_region <- myExpression %>%
      rownames_to_column("Brain.Region")

    # For the current object in 'rskc_list' convert the cluster labels
    # into characters, and assign them to a new column called 'cluster_labels'
    # in 'gene_and_region'.
    gene_and_region$cluster_labels <- rskc_list[[counter]]$labels %>%
      as.character()
```

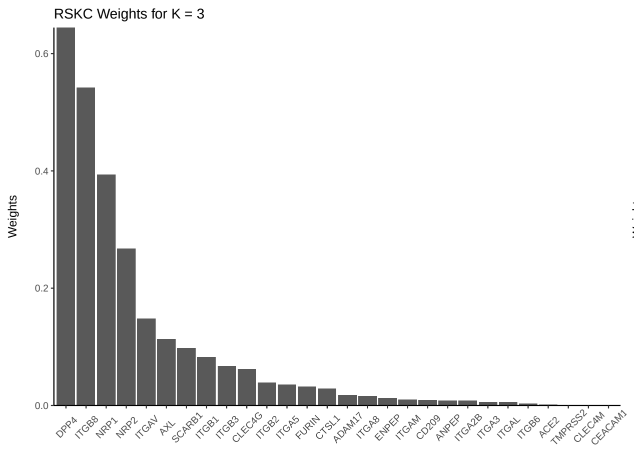
```r
# Order the weights for the current item in 'rskc_list' from largest
# to smallest, extract the names of the genes in this order,
# convert this object into a data frame, and store this info in
# an object 'weight_df' in a column called 'gene'.
weight_df <- sort(rskc_list[[counter]]$weights,
                  decreasing = T) %>%
  names() %>%
  as.data.frame() %>%
  rename('gene' = ".")

# Assign the ordered weights for the current item in 'rskc_list' into
# 'weight_df', in a column called 'weight',
weight_df$weight <- sort(rskc_list[[counter]]$weights,
                         decreasing = T) %>%
  unname()

# Impose a factor order on the contents of 'weight_df$gene' in
# the current order.
weight_df$gene <- factor(weight_df$gene,
                         weight_df$gene)

# Create a bar graph of the RSKC weights for each gene ordered from
# largest to smallest. Assign this graph to an object, 'weight_bars'.
weight_bars <- weight_df %>%
  ggplot(aes(x = gene, y = weight)) +
  theme_classic() +
  geom_bar(stat = 'identity') +
  theme(axis.text.x = element_text(angle = 45, vjust = 0.6)) +
  scale_y_continuous(expand = c(0,0)) +
  ggtitle(paste0("RSKC Weights for K = ",i)) +
  xlab("") +
  ylab("Weights\n")

# Assign 'weight_bars' as the current entry into 'weight_list'.
weight_list[[counter]] <- weight_bars

###### Apply weights from RSKC to myExpression ######

# Create vector of the weights obtained from RSKC and assign them to 'weights'.
# Make empty matrix 'weighted_expression' for new weighted exprression.
weights <- as.matrix(rskc_list[[1]]$weights)

# Multiply 'myExpression' columns containing gene_celltype by corresponding
# weights obtained from RSKC.
weighted_expression <- sweep(t(myExpression),
                             MARGIN = 1, weights,
                             `*`) %>%
  t()

###### tSNE (on weighted data) ######

# Run tsne on weighted expression scores, and assign to 'tsne'
set.seed(72613)
```

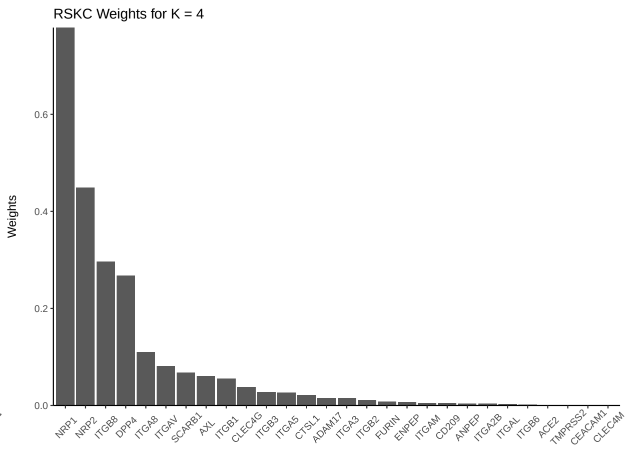
```
    tsne <- Rtsne(weighted_expression, perplexity = 5)

    # Create new df 'tsne_out' which contains the two dimensions obtained from tSNE
    # and corresponding regions
    tsne_out <- tsne$Y %>%
      data.frame(gene_and_region$Brain.Region) %>%
      rename(Brain.Region = gene_and_region.Brain.Region,
             V1 = X1, V2 = X2) #rename column

    # Merge 'tsne_out' with 'gene_and_region' according
    # to their shared 'Brain.Region' column, and assign to
    # 'tsne_genes_regions_clusts'.
    tsne_genes_regions_clusts <- merge(tsne_out,
                                       gene_and_region,
                                       by = "Brain.Region")

    # Create a tSNE scatter plot where each point is colour-coded according to
    # its designated RSKC cluster and assign this figure to 'tsne_scatter'.
    tsne_scatter <- ggplot(tsne_genes_regions_clusts,
                           aes(V1,
                               V2,
                               fill = cluster_labels)) +
      geom_point(shape = 21, size = 3) +
      scale_fill_manual(values = col_vect[1:i],
                        labels = 1:i,
                        name = "Cluster") +
      theme(axis.text.x = element_blank(),
            axis.ticks.x = element_blank(),
            axis.text.y = element_blank(),
            axis.ticks.y = element_blank(),
            panel.background = element_rect(fill = NA,
                                            colour = "white"),
            panel.border = element_blank(),
            axis.line = element_line(),
            legend.position = 'bottom',
            legend.background = element_rect(fill = NA,
                                             colour = NA),
            legend.title.align=0.5) +
      guides(fill=guide_legend(nrow = 2,
                               ncol = 4,
                               byrow = TRUE)) +
      labs(x="V1", y="V2")

    # Assign 'tsne_scatter' as an entry in 'tsne_list'.
    tsne_list[[counter]] <- tsne_scatter

  }

  # Break out of the while-loop when for-loop is done.
  break

}
```

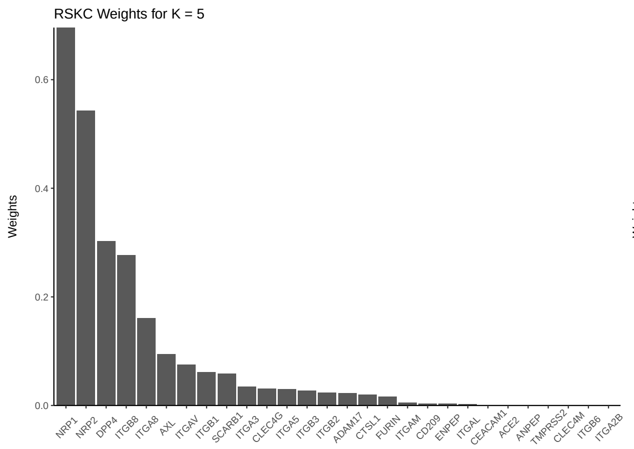
```r
# Make figures for all the plots obtained from the while loop above
# i.e. the RSKC scatter plots, the RKSC weights, and elbow plot
ggarrange(tsne_list[[1]], tsne_list[[2]],
          tsne_list[[3]], tsne_list[[4]],
          ncol = 2,
          nrow = 2)
```

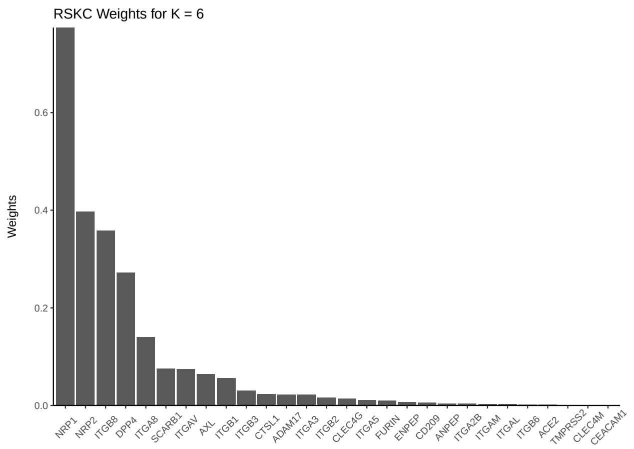


```r
ggarrange(weight_list[[1]], weight_list[[2]],
          weight_list[[3]], weight_list[[4]],
          ncol = 2,
          nrow = 2)
```


RSKC Weights for K = 3
Weights
0.0
0.2
0.4
0.6
DPP4
ITGB8
NRP1
NRP2
ITGAV
AXL
SCARB1
ITGB1
ITGB3
CLEC4G
ITGB2
ITGA5
FURIN
CTSL1
ADAM17
ITGA8
ENPEP
ITGAM
CD209
ANPEP
ITGA2B
ITGA3
ITGAL
ITGB6
ACE2
TMPRSS2
CLEC4M
CEACAM1


RSKC Weights for K = 4
Weights
0.0
0.2
0.4
0.6
NRP1
NRP2
ITGB8
DPP4
ITGA8
ITGAV
SCARB1
AXL
ITGB1
CLEC4G
ITGB3
ITGA5
CTSL1
ADAM17
ITGA3
ITGB2
FURIN
ENPEP
ITGAM
CD209
ANPEP
ITGA2B
ITGAL
ITGB6
ACE2
TMPRSS2
CEACAM1
CLEC4M


RSKC Weights for K = 5
Weights
0.0
0.2
0.4
0.6
NRP1
NRP2
DPP4
ITGB8
ITGA8
AXL
ITGAV
ITGB1
SCARB1
ITGA3
CLEC4G
ITGA5
ITGB3
ITGB2
ADAM17
CTSL1
FURIN
ITGAM
CD209
ENPEP
ITGAL
CEACAM1
ACE2
ANPEP
TMPRSS2
CLEC4M
ITGB6
ITGA2B


RSKC Weights for K = 6
Weights
0.0
0.2
0.4
0.6
NRP1
NRP2
ITGB8
DPP4
ITGA8
SCARB1
ITGAV
AXL
ITGB1
ITGB3
CTSL1
ADAM17
ITGA3
ITGB2
CLEC4G
ITGA5
FURIN
ENPEP
CD209
ANPEP
ITGA2B
ITGAM
ITGAL
ITGB6
ACE2
TMPRSS2
CLEC4M
CEACAM1

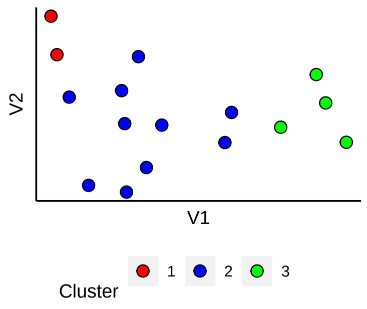
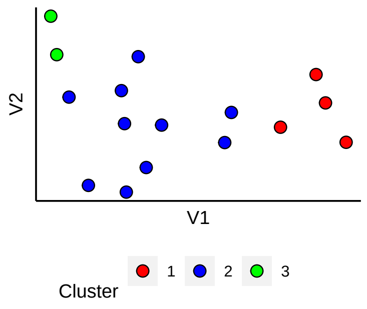
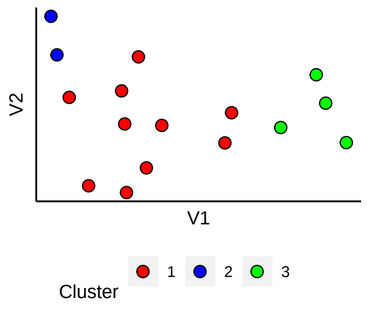
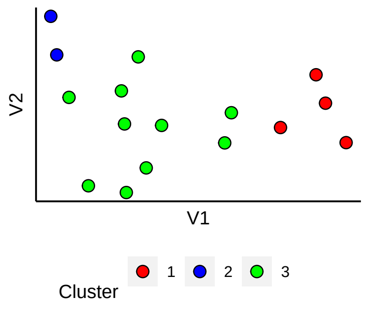
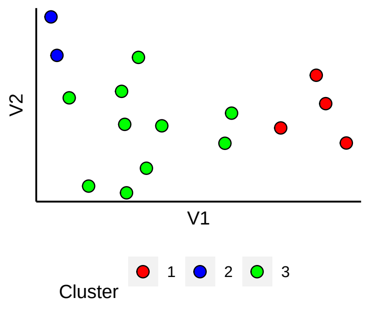
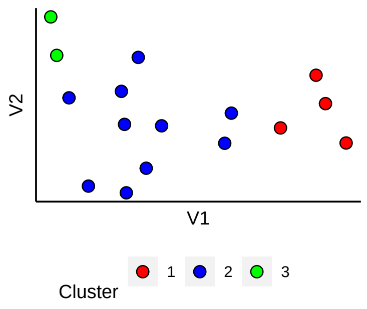
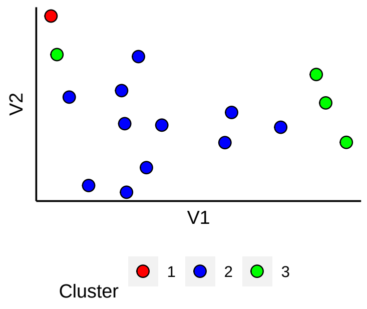
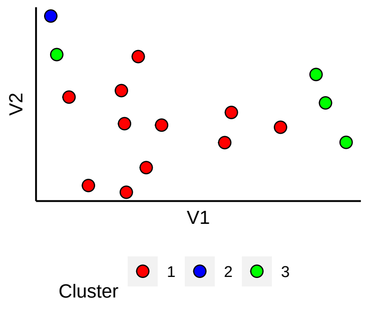
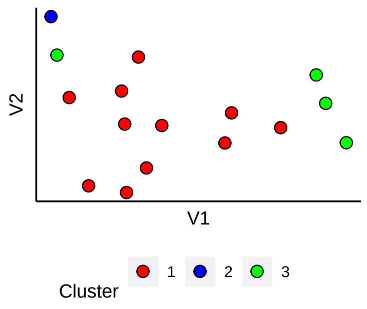
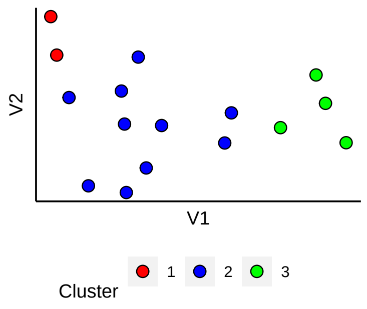
RSKC (10 Runs)

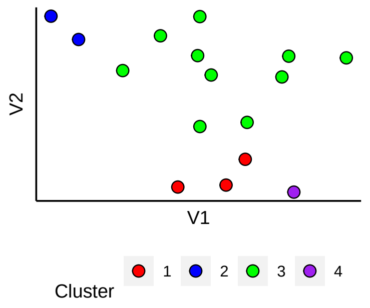
This chunk serves as a snapshot of the RSKC over 10 runs (i.e. different set.seed values) in order to visualize the variation in cluster label assignments for each observation. The 10 runs were carried out for each of $K = 3, 4, 5, 6$.

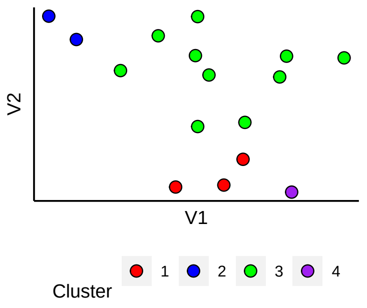
```
set.seed(72613)

while (T) {

  # Assign the values of 3, 4, 5, 6 to 'clust_vect'.
  clust_vect <- c(3,4,5,6)

  # Assign empty lists to 'rskc.results.list' and 'rskc.weighted.list'
  # to store the results and the RSKC weighted data frames that result
  # from the clustering.
  rskc_results_list = list()
  rskc_weighted_list = list()

  # Assign 7 colours to 'col_vect'.
  col_vect <- c("#FF0000",
                "#0000FF",
                "#00FF00",
                "#A020F0",
                "#FFA500",
                "#FFFF00",
                "#A65628")

  # Assign an empty list to 'tsne_list_3', 'tsne_list_4', 'tsne_list_5',
  # and 'tsne_list_6'.
  tsne_list_3 <- list()
  tsne_list_4 <- list()
  tsne_list_5 <- list()
  tsne_list_6 <- list()

  # Assign an empty list to 'weight_list'.
  weight_list <- list()

  # Create empty data frames to store the cluster assignments and cluster weights
  # for each of the 10 runs.
    rskc_region_labels_3 = data.frame("Region" = rownames(myExpression))
    rskc_region_labels_4 = data.frame("Region" = rownames(myExpression))
    rskc_region_labels_5 = data.frame("Region" = rownames(myExpression))
    rskc_region_labels_6 = data.frame("Region" = rownames(myExpression))
    rskc_region_weights = data.frame("Case" = colnames(myExpression))

  # For 'i' -- the current number of clusters -- in 'clust_vect'...
  for (i in clust_vect) {

    ###### RSKC (10 Runs) ######

    # Create a vector of seeds for all 10 runs.
    set.seed(72613)
    x = rdunif(10, a = 1, b = 1000000)
```

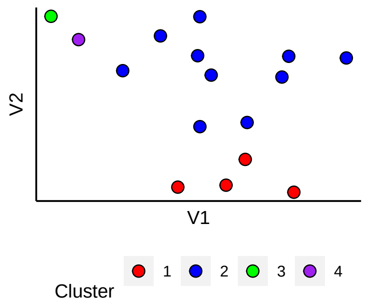
```r
for (counter in 1:10) {

  # Set the seed.
  set.seed(x[counter])

  # Perform RSKC for whatever-the-value-of-''-is many clusters using
  # 'myFidelity', which has brain region as rows and gene_celltype as columns.
  # Assign RSKC's output as an entry in 'rskc_results_list'.
  rskc_results_list[[counter]] <- RSKC(myExpression,
                                       alpha = 0.1,
                                       ncl = i,
                                       L1 = sqrt(ncol(myExpression)))

  # Use the following if statements to add the cluster assignments for run i to
  # the corresponding 'rskc_region_labels_3', 'rskc_region_labels_4', 'rskc_region_labels_5',
  # or 'rskc_region_labels_6'.
  if (i == 3){
    rskc_region_labels_3[counter+1] <- rskc_results_list[[counter]]$labels
    colnames(rskc_region_labels_3)[counter+1] <- paste("Run_", counter, sep = "")
  }

  if (i == 4){
    rskc_region_labels_4[counter+1] <- rskc_results_list[[counter]]$labels
    colnames(rskc_region_labels_4)[counter+1] <- paste("Run_", counter, sep = "")
  }

  if (i == 5){
    rskc_region_labels_5[counter+1] <- rskc_results_list[[counter]]$labels
    colnames(rskc_region_labels_5)[counter+1] <- paste("Run_", counter, sep = "")
  }

  if (i == 6){
    rskc_region_labels_6[counter+1] <- rskc_results_list[[counter]]$labels
    colnames(rskc_region_labels_6)[counter+1] <- paste("Run_", counter, sep = "")
  }

  # Add the variable weights for run i to the 'rskc_region_weights'
  rskc_region_weights[counter+1] <- rskc_results_list[[counter]]$weights
  colnames(rskc_region_weights)[counter+1] <- paste("Run_", counter, sep = "")

  # For the current object in 'rskc_list' convert the cluster labels
  # into characters, and assign them to a new column called 'cluster_labels'
  # in 'gene_expression'.
  gene_expression$cluster_labels <- rskc_results_list[[counter]]$labels %>%
    as.character()

  ###### Apply weights from RSKC to myExpression ######

  # Create vector of the weights obtained from RSKC and assign them to 'weights'.
  # Make empty matrix 'weighted_expression' for new weighted expression.
  weights <- as.matrix(rskc_results_list[[1]]$weights)

  # Multiply 'myExpression' columns containing gene_celltype by corresponding
# weights obtained from RSKC.
weighted_expression <- sweep(t(myExpression),
                             MARGIN = 1, weights, `*`) %>%
  t()

###### tSNE (on weighted data) ######

# Run tsne on weighted expression scores, and assign to 'tsne'
set.seed(72613)
tsne <- Rtsne(weighted_expression, perplexity = 5)

# Create new df 'tsne_out' which contains the two dimensions obtained from tSNE
# and corresponding regions
tsne_out <- tsne$Y %>%
  data.frame(gene_expression$Brain.Region) %>%
  rename(Brain.Region = gene_expression.Brain.Region,
         V1 = X1,
         V2 = X2) #rename columns

# Merge 'tsne_out' with 'gene_expression' according
# to their shared 'Brain.Region' column, and assign to
# 'tsne_genes_regions_clusts'.
tsne_genes_regions_clusts <- merge(tsne_out,
                                   gene_expression,
                                   by = "Brain.Region")

# Create a tSNE scatter plot where each point is colour-coded according to
# its designated RSKC cluster and assign this figure to 'tsne_scatter'.
tsne_scatter <- ggplot(tsne_genes_regions_clusts,
                       aes(V1,
                           V2,
                           fill = cluster_labels)) +
  geom_point(shape = 21, size = 3) +
  scale_fill_manual(values = col_vect[1:i],
                    labels = 1:i,
                    name = "Cluster") +
  theme(axis.text.x = element_blank(),
        axis.ticks.x = element_blank(),
        axis.text.y = element_blank(),
        axis.ticks.y = element_blank(),
        panel.background = element_rect(fill = NA,
                                        colour = "white"),
        panel.border = element_blank(),
        axis.line = element_line(),
        legend.position = 'bottom',
        legend.background = element_rect(fill = NA,
                                         colour = NA),
        legend.title.align=0.5) +
  guides(fill=guide_legend(nrow = 2,
                           ncol = 4,
                           byrow = TRUE)) +
  labs(x="V1", y="V2")
```

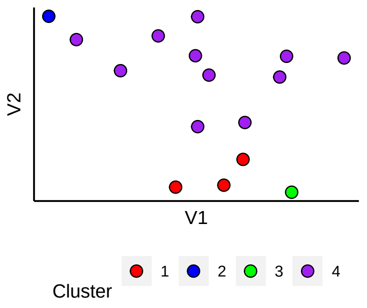
```
      # Use if statements to assign 'tsne_scatter' as an entry in 'tsne_list_3',
      # 'tsne_list_4', 'tsne_list_5', or 'tsne_list_6 depending on the current i.
      if (i == 3){
        tsne_list_3[[counter]] <- tsne_scatter
      }

      if (i == 4){
        tsne_list_4[[counter]] <- tsne_scatter
      }

      if (i == 5){
        tsne_list_5[[counter]] <- tsne_scatter
      }

      if (i == 6){
        tsne_list_6[[counter]] <- tsne_scatter
      }

    }

  }

  # Break out of the while-loop when for-loop is done.
  break

}
```

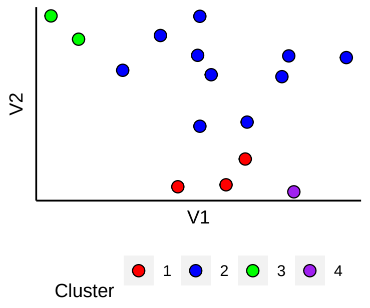

V2
V1
Cluster
1
2
3

V2
V1
Cluster
1
2
3

V2
V1
Cluster
1
2
3

V2
V1
Cluster
1
2
3

V2
V1
Cluster
1
2
3

V2
V1
Cluster
1
2
3

V2
V1
Cluster
1
2
3

V2
V1
Cluster
1
2
3

V2
V1
Cluster
1
2
3

V2
V1
Cluster
1
2
3

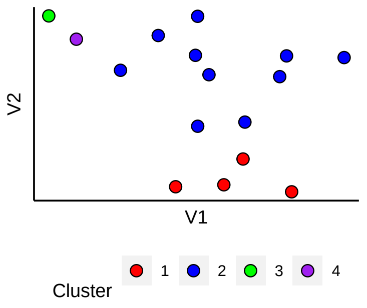

V2
V1
Cluster
1
2
3
4

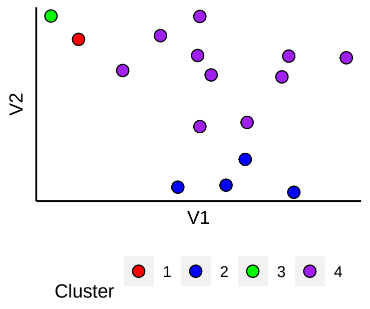

V2
V1
Cluster
1
2
3
4

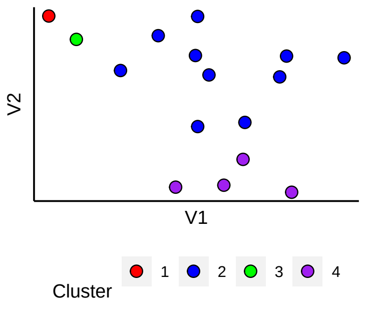

V2
V1
Cluster
1
2
3
4

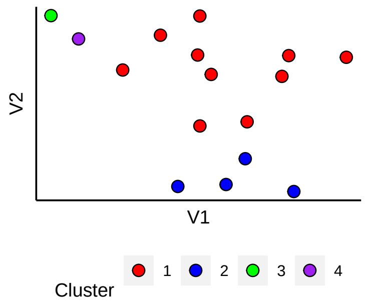

V2
V1
Cluster
1
2
3
4

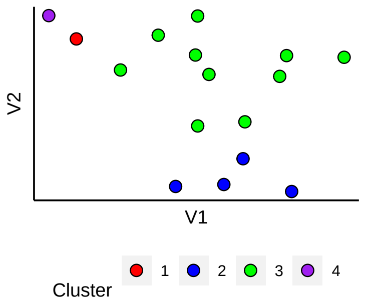

V2
V1
Cluster
1
2
3
4

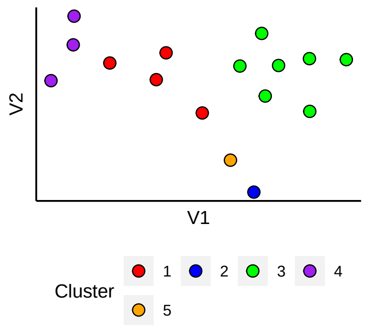
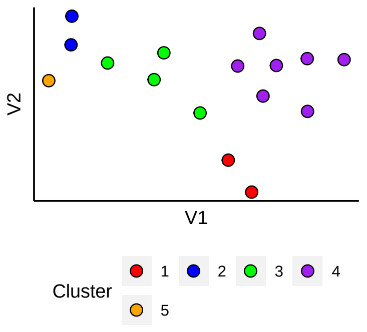
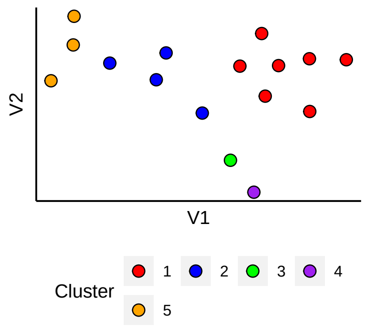
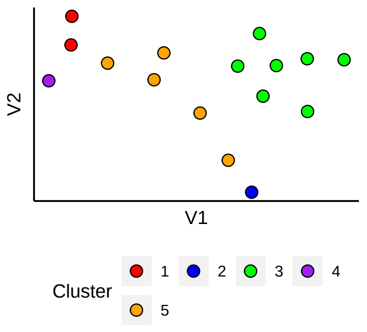
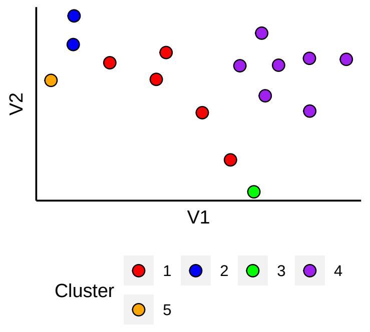
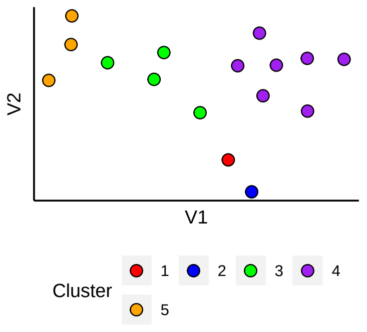
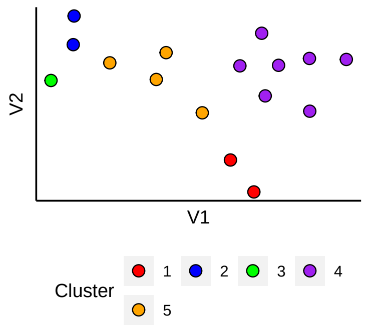
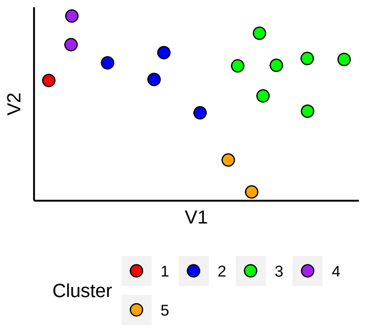
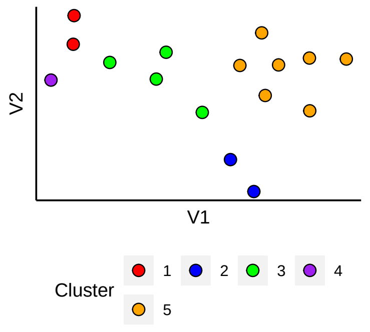
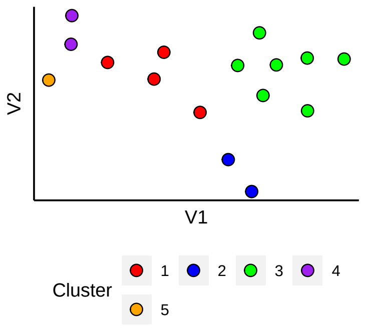

V2
V1
Cluster
1
2
3
4

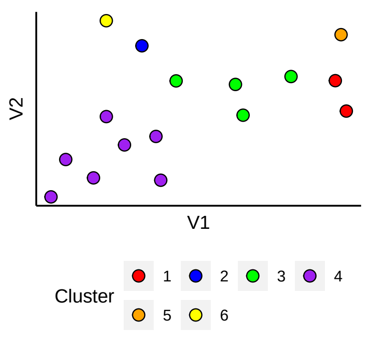

V2
V1
Cluster
1
2
3
4

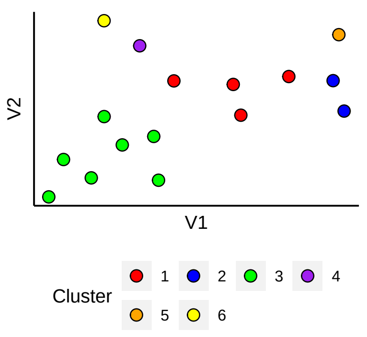

V2
V1
Cluster
1
2
3
4

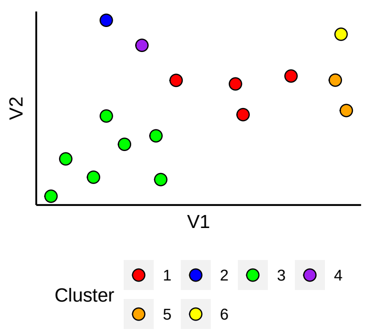

V2
V1
Cluster
1
2
3
4

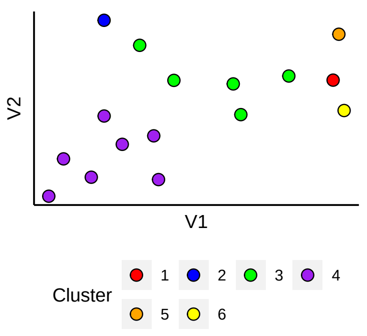

V2
V1
Cluster
1
2
3
4

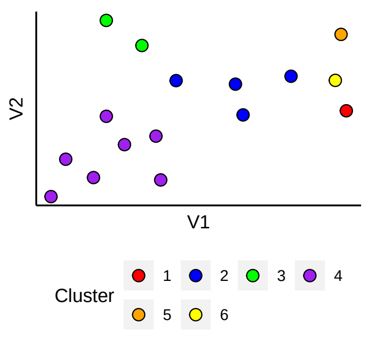

V2
V1
Cluster
1
2
3
4
5

V2
V1
Cluster
1
2
3
4
5

V2
V1
Cluster
1
2
3
4
5

V2
V1
Cluster
1
2
3
4
5

V2
V1
Cluster
1
2
3
4
5

V2
V1
Cluster
1
2
3
4
5

V2
V1
Cluster
1
2
3
4
5

V2
V1
Cluster
1
2
3
4
5

V2
V1
Cluster
1
2
3
4
5

V2
V1
Cluster
1
2
3
4
5

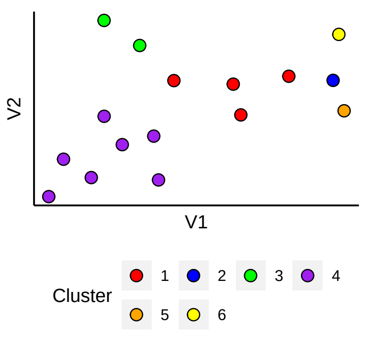

V2
V1
Cluster
1
2
3
4
5
6

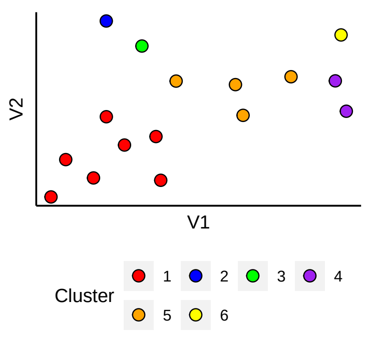

V2
V1
Cluster
1
2
3
4
5
6

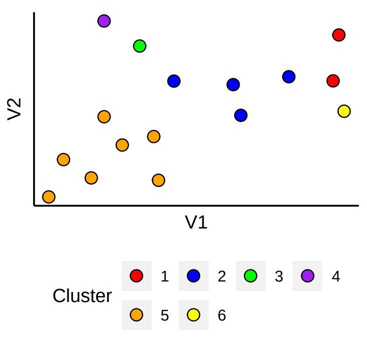

V2
V1
Cluster
1
2
3
4
5
6

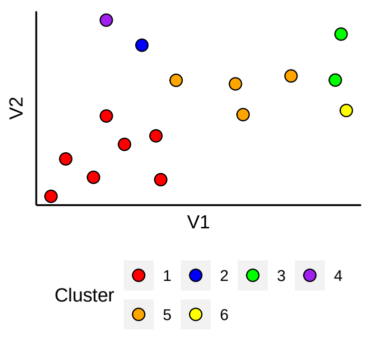

V2
V1
Cluster
1
2
3
4
5
6

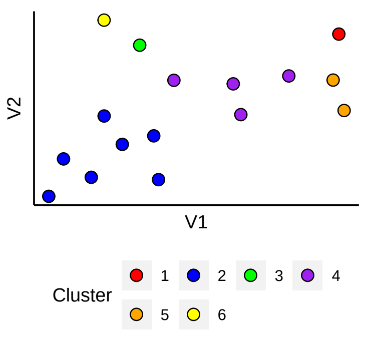

V2
V1
Cluster
1
2
3
4
5
6


V2
V1
Cluster
1
2
3
4
5
6


V2
V1
Cluster
1
2
3
4
5
6


V2
V1
Cluster
1
2
3
4
5
6


V2
V1
Cluster
1
2
3
4
5
6


V2
V1
Cluster
1
2
3
4
5
6