COVID-19 Project: RSKC Analysis for Fidelity Scores

Rachel Kwan and Jonathan Zaslavsky

May 16, 2021

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)
library(factoextra) # For fviz_nbclust() and clustering analysis/visualization
library(reshape2) # For melt()
library(RColorBrewer) # For color palettes
library(gplots) # For heatmap.2()
library(dendextend) # For manipulating visual appearance of dendrograms

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

Import and Prepare Dataset

```
# Load full dataset
all_fidelity <- read.csv(here("Data", "ALL_Fidelity.csv"))

# Define a vector containing the five genes of interest.
genes_subset <- c("ACE2",
                  "DPP4",
                  "TMPRSS2",
                  "NRP1",
                  "ITGB3")

# Define a vector containing all SARS-CoV-2 related genes.
genes_full_list <- c("ACE2", "DPP4", "ANPEP", "CD209", "ENPEP", "CLEC4G",
                     "CLEC4M", "CEACAM-1", "TMPRSS2", "TMPRSS11d", "NRP1",
                     "CTSL", "ADAM17", "FURIN", "ITGA5", "ITGB3", "ITGA4",
                     "ITGB6", "ITGA7")

# Filter full dataset to include only the genes of interest and drop the
# columns that correspond to "Entrez" and "Alias".
```

```r
fidelity_subset <- all_fidelity %>%
  filter(Gene %in% genes_subset) %>% # Includes five genes
  # filter(Gene %in% genes_full_list) %>% # Includes full list of genes
  select(-c(2,3))

# Tidy the data frame by separating the previous columns into
# two new columns for the brain region and cell subtype. Assign each of the
# values to a new column for the fidelity scores. The tidy data's
# columns each correspond to a varaible and each observation is a new row.
fidelity_subset.long <-  pivot_longer(fidelity_subset, cols = -Gene,
                                      names_to = c("Brain.Region", "Cell.Subtype"),
                                      names_sep = "_",
                                      values_to = "Fidelity")

# Pivot the data frame to a wide view by assigning new columns corresponding to
# the gene_celltype for each brain region. Assign the fidelity scores as
# the values.
fidelity_subset.wide <- pivot_wider(fidelity_subset.long,
                                    names_from = c("Gene", "Cell.Subtype"),
                                    values_from = "Fidelity",
                                    names_sep = "_")

# Define a vector containing brain regions of interest. This list comes from the
# Oldham Lab website (https://oldhamlab.ctec.ucsf.edu/). Note: 'SC' contains
# NA values.
regions <- c("FCX", "PCX", "TCX", "LIM", "IN", "OCX","BF", "CLA", "AMY", "HIP",
             "STR", "GP", "DI", "MID", "PON", "MED", "CB", "WM") #, "SC")

# Choose only the desired brain regions from the data frame.
fidelity <- fidelity_subset.wide %>%
  filter(Brain.Region %in% regions) %>%
  slice(match(regions, Brain.Region)) %>% # Reorder rows to match entries in "regions" vector
  select(-contains("Percentile")) # Remove NA columns

# Use the brain regions to name the rows and remove the brain region column.
myFidelity <- fidelity %>%
  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. Dezi'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 the values of 2, 4, 6 and 8 to 'clust_vect'.
  clust_vect <- c(2,4,6,8)

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

```r
# Assign 8 colours to 'col_vect'.
col_vect <- c("#FF0000",
              "#0000FF",
              "#00FF00",
              "#A020F0",
              "#FFA500",
              "#FFFF00",
              "#A65628",
              "#F781BF")

# 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 = 2
  # Increment 'counter' with a value of 1.
  counter = counter + 1

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

  # Perform RSKC for whatever-the-value-of-'i'-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_list'.
  rskc_list[[counter]] <- RSKC(myFidelity,
                               ncl = i,
                               alpha = 0.1,
                               L1 = sqrt(ncol(myFidelity)))


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

  # 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,
```

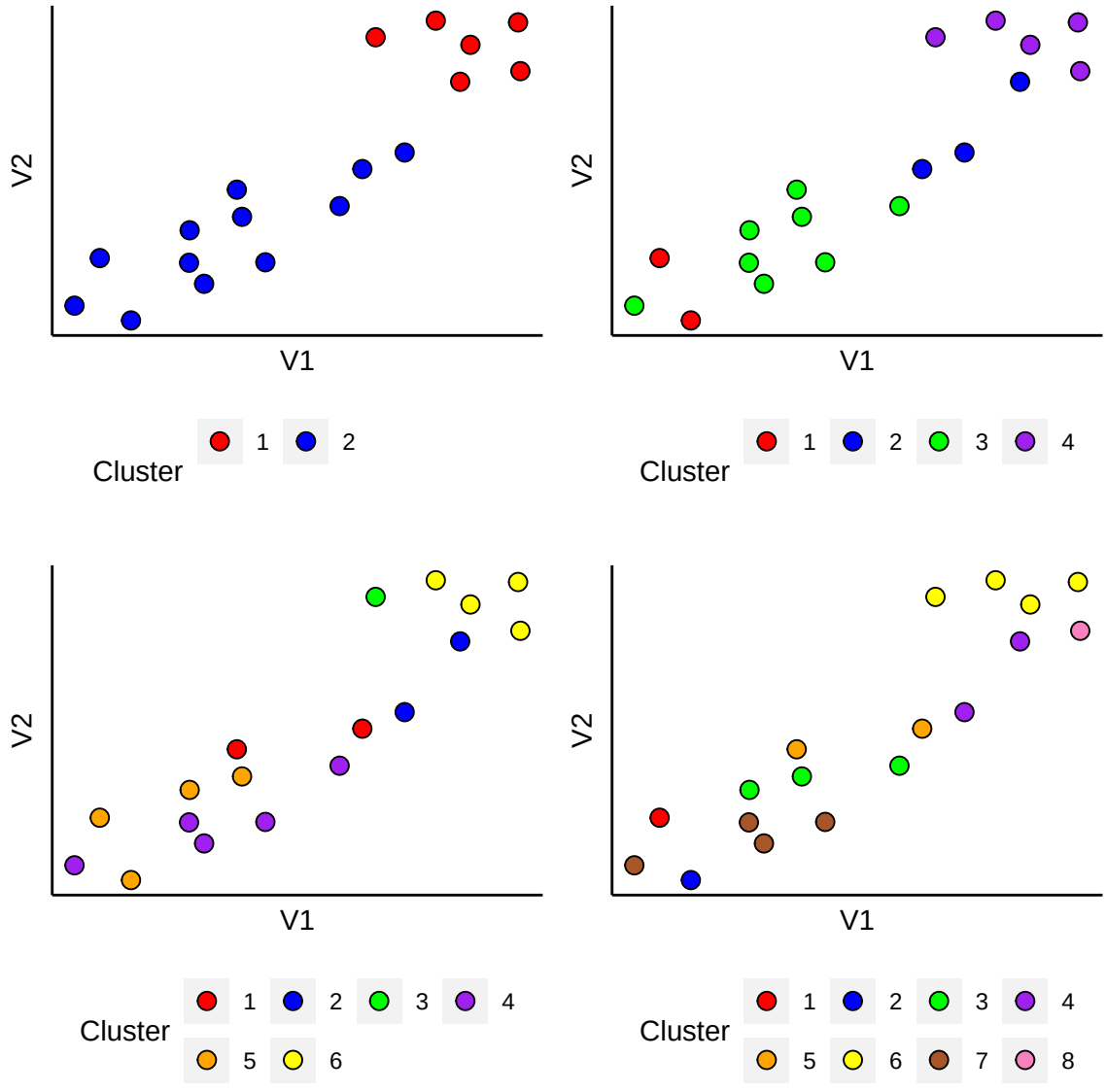
```
                    decreasing = T) %>%

  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


###### Elbow Plot ######

# For the purposes of an elbow plot, we want to run RSKC more times than
# specified in 'clust_vect' above, so create new vector 'elbow_clust_vect'
elbow_clust_vect <- c(2,3,4,5,6,7,8,9,10,11,12,13,14,15,16)

# Assign another empty list to 'elbow_rskc_list'.
elbow_rskc_list <- c()
```

```
  # Only produce elbow plot when we are on the last run of the while loop,
  # since we only need one elbow plot
  if (i == clust_vect[length(clust_vect)]){

    elbow_counter = 0

    for (n in elbow_clust_vect) {
      # n = 2
      # Increment 'counter' with a value of 1.
      elbow_counter = elbow_counter + 1

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

    # Create empty vector 'between_ss' to store weighted between sum of squares (WBSS)
    # values from RSKC output
    between_ss <- matrix(ncol = 1, nrow = length(elbow_clust_vect))

    # Use for loop to add WBSS values to 'between_ss'
    # Some RSKC outputs may give more than one WBSS value, so take the last one
    for (n in 1:length(elbow_clust_vect)){
      between_ss[n] <- elbow_rskc_list[[n]]$WBSS[length(elbow_rskc_list[[n]]$WBSS)]
    }

    # Create a new dataframe 'objective_function' with the WBSS values and
    # their corresponding number of clusters.
    objective_function <- data.frame(elbow_clust_vect, between_ss) %>%
      rename(k = elbow_clust_vect, WBSS = between_ss)

    # Create the elbow plot and assign it to 'elbow_plot'
    elbow_plot <- ggplot(objective_function, aes(x = k, y = WBSS)) +
      geom_line() +
      geom_point() +
      scale_x_continuous(breaks = elbow_clust_vect) +
      labs(x = "Number of Clusters",
           y = "Total Weighted Between Sum of Squares")

  }


  ###### Apply weights from RSKC to myFidelity ######

  # Create vector of the weights obtained from RSKC and assign them to 'weights'.
  # Make empty matrix 'weighted_fidelity' for new weighted fidelity scores.
  weights <- as.matrix(rskc_list[[1]]$weights)
  weighted_fidelity <- matrix(nrow = 18, ncol = 20)
```

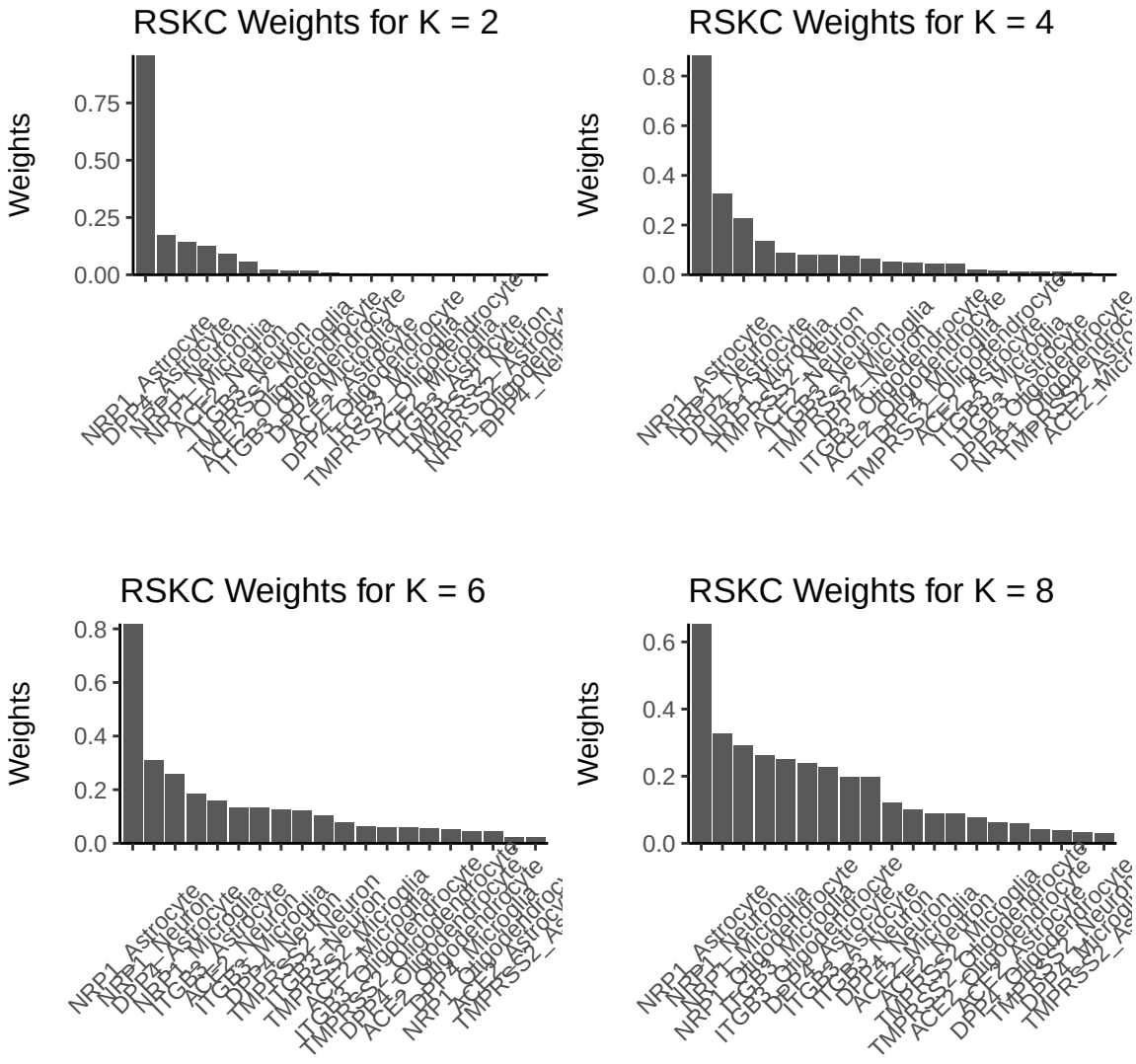
```r
# Multiply 'myFidelity' by corresponding weights obtained from RSKC.
for (n in 1:20){
  weighted_fidelity[,n] <- myFidelity[,n]*weights[n]
}

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

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


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

# 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'.
```

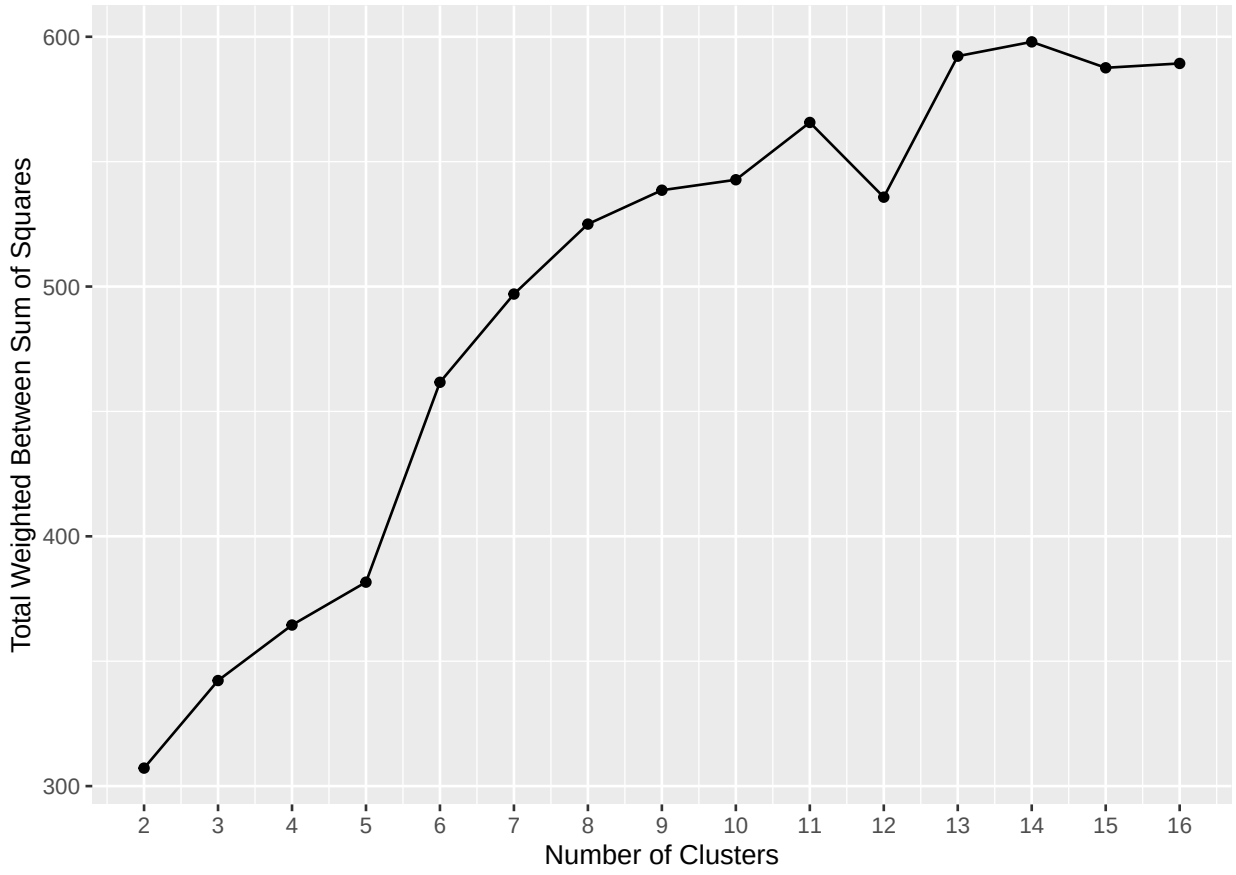
```
    tsne_list[[counter]] <- tsne_scatter

  }

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

}
```

```
# 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)
```



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




Total Weighted Between Sum of Squares
600
500
400
300
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
Number of Clusters

RSKC (100 Runs)

Having previously selected the number of clusters to use for RSKC (i.e. 4), this portion of the code aims to evaluate the proportion of brain regions clustered together over 100 runs (i.e. 100 different set.seed values). The code is derived from previous work (i.e. Keon and Brendan).

```
# Create empty lists to store the results and the RSKC weighted data frames
# that result from the clustering.
rskc.results.list = list()
rskc.weighted.list = list()

# Create empty data frames to store the cluster assignments and cluster weights
# for each of the 100 runs.
rskc.region.labels = data.frame("Region" = rownames(myFidelity))
rskc.region.weights = data.frame("Case" = colnames(myFidelity))

# Create a vector of seeds for all 100 runs.
set.seed(72613)
x = rdunif(100, a = 1, b = 1000000)

for (i in 1:100) {

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

  # Perform RSKC clustering on the data; the number of clusters is selected
  # a priori.
  rskc.results.list[[i]] = RSKC(myFidelity,
                                alpha = 0.1,
                                ncl = 4,
                                L1 = sqrt(ncol(myFidelity)))

  # Add the cluster assignments for run i to the 'rskc.region.labels'.
  rskc.region.labels[i+1] = rskc.results.list[[i]]$labels
  colnames(rskc.region.labels)[i+1] = paste("Run_", i, sep = "")

  # Add the variable weights for run i to the 'rskc.region.weights'
  rskc.region.weights[i+1] = rskc.results.list[[i]]$weights
  colnames(rskc.region.weights)[i+1] = paste("Run_", i, sep = "")

  # Create a list of data frames containing the clustering data multiplied by
  # the corresponding RSKC variable weights.
  rskc.weighted.list[[i]] = sweep(myFidelity, 2,
                                  rskc.results.list[[i]]$weights, "*")
}
```

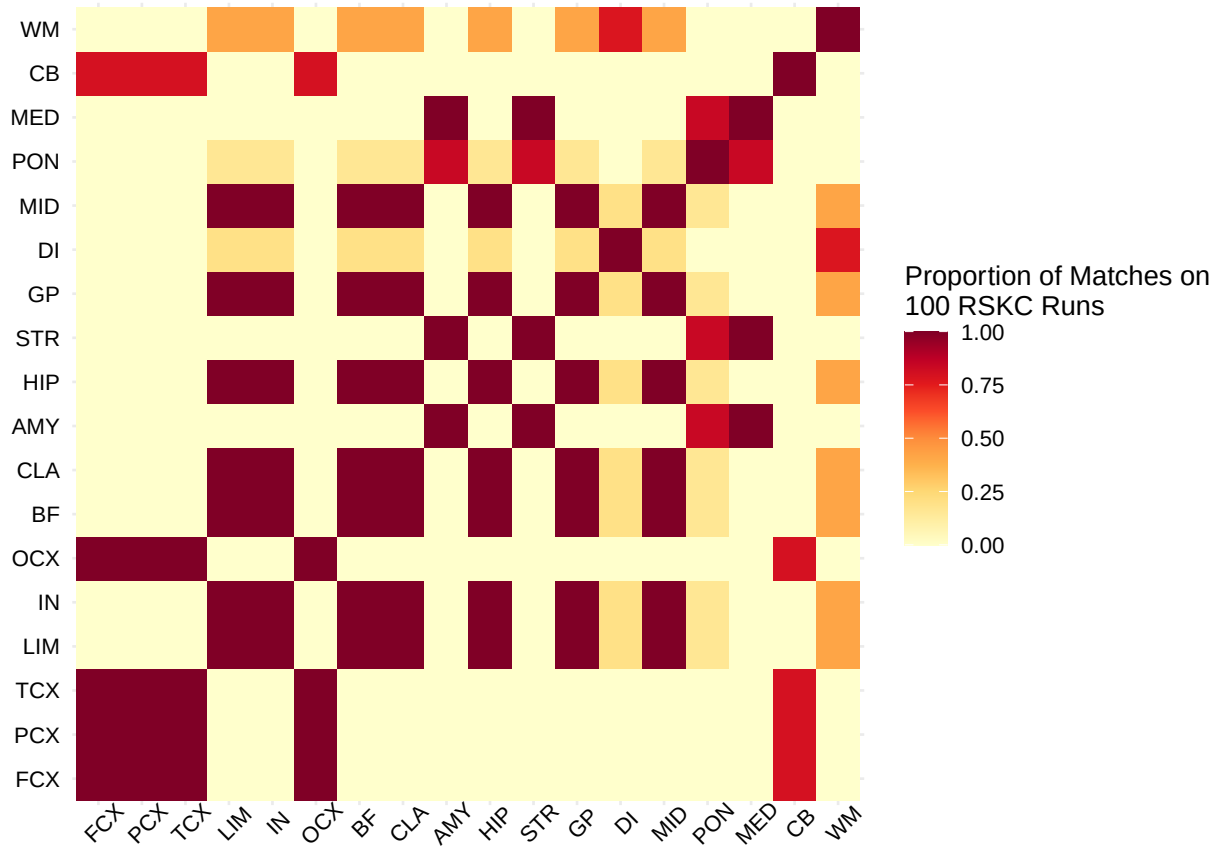
Calculating Proportion of Times Brain Regions are Clustered Together

```
# Transpose the data frame with the cluster labels from the 100 runs.
rskc.region.labels.t <-  rskc.region.labels %>%
  column_to_rownames("Region") %>%
  t() %>%
```

```
  data.frame()

# Create an empty 18x18 data frame
rskc.cluster.regions.wide <-  data.frame(matrix(ncol = 18, nrow = 18)) %>%
  # Set the column and row names as the regions (ordered alphabetically)
  set_colnames(regions) %>%
  set_rownames(regions)

# Add in the number of matches for each ith row/jth column combination to
# create the adjacency matrix.
for (i in 1:length(regions)) {
  for (j in 1:length(regions)) {

    rskc.cluster.regions.wide[i,j] = sum(rskc.region.labels.t[[regions[i]]] == rskc.region.labels.t[[reg

  }
}

# Convert the adjacency matrix to long format for various plotting purposes
rskc.cluster.regions.long <-  rskc.cluster.regions.wide %>%
  # Give the region row names their own column
  rownames_to_column("Region_1") %>%
  # Lengthen the data with melt() so we have three columns:
  # Region_1, Region_2 and the total number of matches.
  melt(id.vars = "Region_1", variable.name = "Region_2", value.name = "Matches") %>%
  # Convert the number of matches to a proportion
  mutate(Matches = Matches / 100)
```

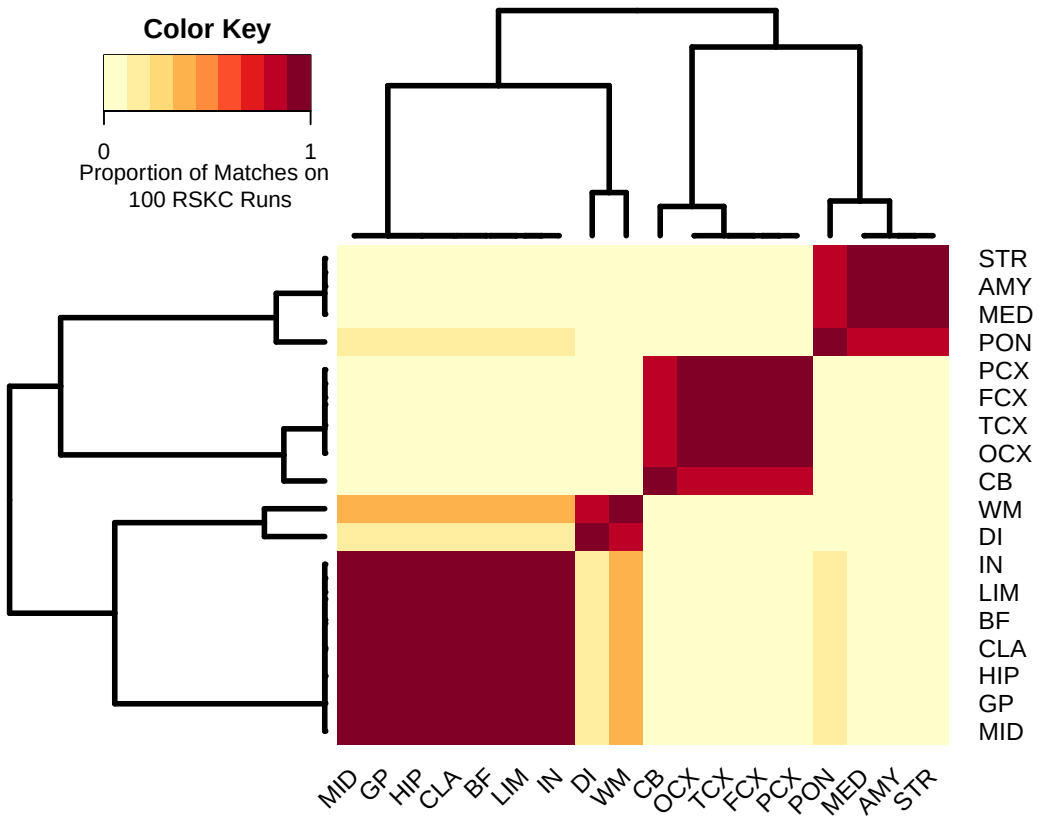
Heat Maps to Visualize Proportion of Shared Clusters

```
# Factor the regions with levels corresponding to the specified brain region
# names from the 'regions' vector.
region.rskc.cluster.matches.ordered <-  rskc.cluster.regions.long %>%
  # Factor Region_1
  mutate(Region_1 = factor(Region_1, levels = regions)) %>%
  # Factor Region_2
  mutate(Region_2 = factor(Region_2, levels = regions))

# Plot the adjacency matrix as a heat map
ggplot(region.rskc.cluster.matches.ordered, aes(x = Region_1, y = Region_2, fill = Matches)) +
  geom_tile() +
  scale_fill_gradientn(colours = brewer.pal(n = 9, name = "YlOrRd")) +
  labs(x = NULL,
       y = NULL,
       fill = "Proportion of Matches on \n100 RSKC Runs") +
  theme_minimal() +
  theme(panel.border = element_blank(),
        panel.background = element_blank(),
        axis.text.x = element_text(colour = "black", angle = 45),
        axis.text.y = element_text(colour = "black"),
        axis.ticks = element_blank())
```




```
# Define a function to turn values out of 100 into proportions.
prop_function <- function(x){
  return(x/100)
}

# Convert all values in the 'rskc.cluster.regions.wide' matrix into
# proportions.
rskc.region.prop <- apply(rskc.cluster.regions.wide, 2, prop_function) %>%
  as.matrix()

# Assign a dendrogram for the matrix of matched clustering of RSKC regions.
rskc_dendro <- set(as.dendrogram(hclust(dist(rskc.region.prop))), "branches_lwd", 3)

heatmap.2(rskc.region.prop,
          scale = "none",
          col = brewer.pal(n = 9, name = "YlOrRd"),
          Rowv = rskc_dendro,
          Colv = rskc_dendro,
          key = TRUE,
          key.xlab = "Proportion of Matches on \n 100 RSKC Runs",
          trace = "none", # Remove the histogram trace from heat map
          density.info = "none", # Remove the histogram from color key
          srtCol = 45, # Rotate column labels on heat map
          margins = c(5, 10),
```

```
        key.xtickfun = function(){
          breaks = pretty(parent.frame()$breaks)
          breaks = breaks[c(1, length(breaks))]
          list(at = parent.frame()$scale01(breaks),
               labels = breaks)
        }
        )
```



RSKC (10 Runs)

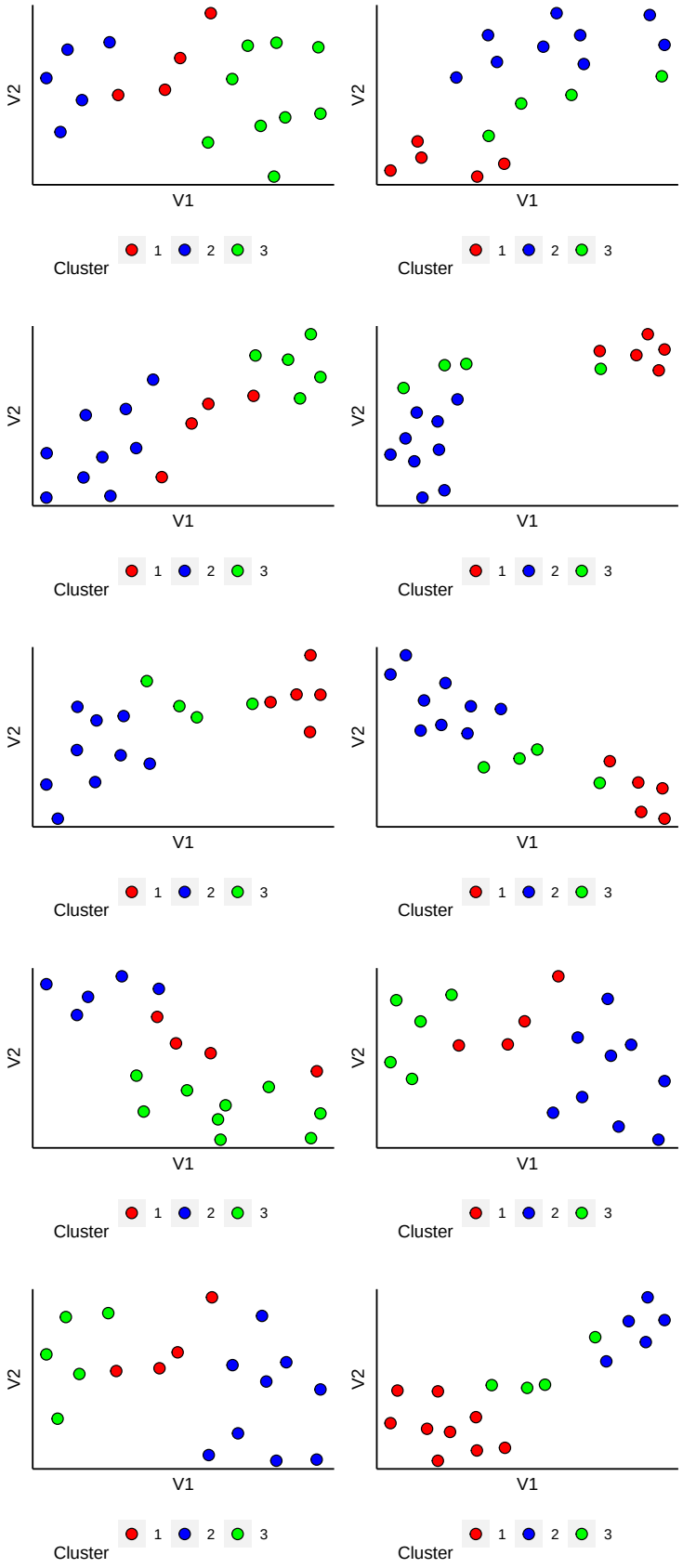
This chunk serves as a snapshot of the RSKC over 10 runs (i.e. different set.seed values) in order to visualize the variation in cluster label assignments for each observation. The 10 runs were carried out for each of $K = 3, 4, 5$.

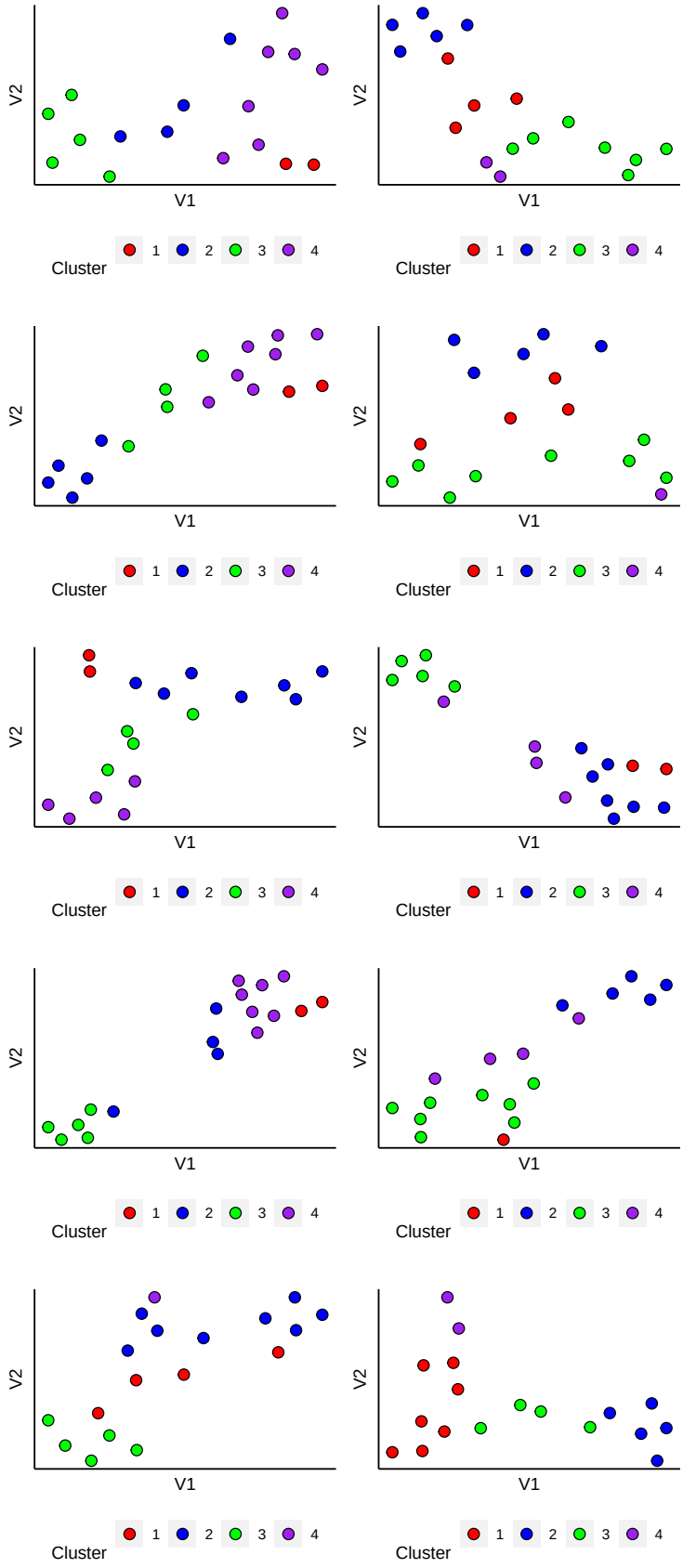
```
set.seed(72613)

while (T) {

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

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

  # Assign an empty list to 'tsne_list_3', 'tsne_list_4', and 'tsne_list_5'.
  tsne_list_3 <- list()
  tsne_list_4 <- list()
  tsne_list_5 <- 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(myFidelity))
    rskc_region_labels_4 = data.frame("Region" = rownames(myFidelity))
    rskc_region_labels_5 = data.frame("Region" = rownames(myFidelity))
    rskc_region_weights = data.frame("Case" = colnames(myFidelity))

  # 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)

    for (counter in 1:10) {

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

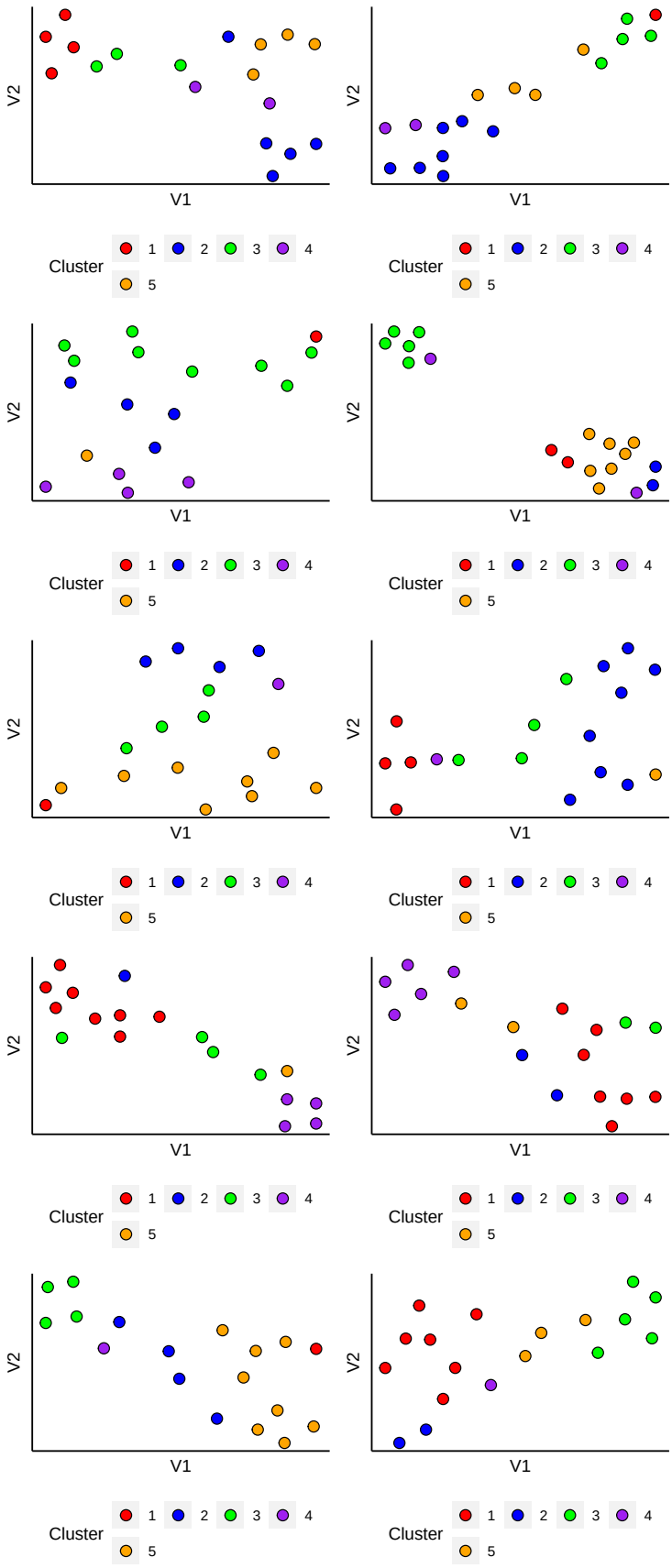
```
# Perform RSKC for whatever-the-value-of-''-is many clusters using
# 'myFidelity', which has brain region as rows and gene_celltype as columns.
# Assign RSKC's output as an entry in 'rskc_results_list'.
rskc_results_list[[counter]] <- RSKC(myFidelity,
                                     alpha =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

# Use the following if statements to add the cluster assignments for run i to
# the corresponding 'rskc_region_labels_3', 'rskc_region_labels_4' or 'rskc_region_labels_5'.
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 = "")
}

# 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 = "")

# Convert the row names of 'myFidelity' to a column
# called 'Brain.Region' and store it in 'gene_and_region'.
gene_and_region <- myFidelity %>%
  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$cluster_labels <- rskc_results_list[[counter]]$labels %>%

  as.character()

###### Apply weights from RSKC to myFidelity ######

# Create vector of the weights obtained from RSKC and assign them to 'weights'.
# Make empty matrix 'weighted_fidelity' for new weighted fidelity scores.
weights <- as.matrix(rskc_results_list[[1]]$weights)
weighted_fidelity <- matrix(nrow = 18, ncol = 20)

# Multiply 'myFidelity' by corresponding weights obtained from RSKC.
for (n in 1:20){
  weighted_fidelity[,n] <- myFidelity[,n]*weights[n]
}
```

```
# Run tsne on weighted fidelity scores, and assign to 'tsne'
tsne <- Rtsne(weighted_fidelity, perplexity = 5)

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

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

# 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")

# Use if statements to assign 'tsne_scatter' as an entry in 'tsne_list_3',
# 'tsne_list_4', or 'tsne_list_5' 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

      }

    }

  }

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

}
```


V2
V1
Cluster
1
2
3


V2
V1
Cluster
1
2
3
4


V2
V1
Cluster
1
2
3
4
5