Feature Space Diagrams in R

Testing the methods on Wisconsin Breast Cancer Dataset

John Erickson

2022-12-12

Introduction

This is a first effort to implement "feature space diagrams" in R, inspired by https://towardsdatascience.com/escape-the-correlation-matrix-into-feature-space-4d71c51f25e5

```
library(ggbiplot)
library(lessR)
library(Hmisc)
library(corrplot)
library(gplots)
library(igraph)
library(tidyverse)
library(ggrepel)
```

Workflow

Our workflow is generally as described in the original post:

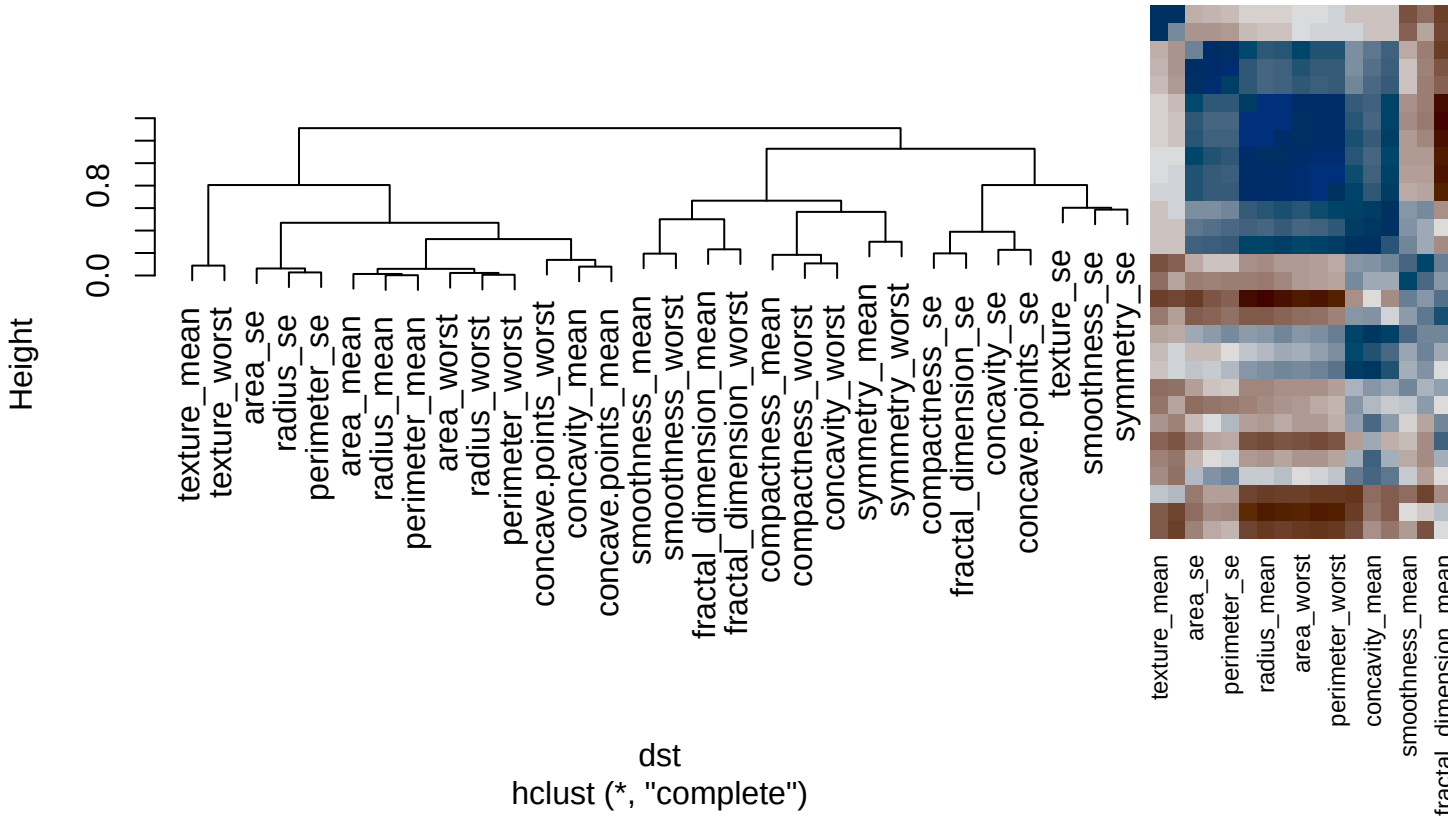
- Generate the correlation matrix
- Take the absolute value of correlation matrix and subtract each value from 1. The result is a distance matrix.
- Use PCA to reduce our NxN matrix to Nx2.
- Plot each feature's location using the two principal components.
- Use Feature Agglomeration to generate feature clusters.
- Color each feature by its cluster.
- Draw lines to represent relationships of at least r = 0.7 (or user's choosing)

Data Load

```
# TODO: User uploads data
# Load data
#boston <- read.csv("boston.csv", header = TRUE, fileEncoding="latin1")
WDBC <- read.csv("WDBC.csv", header = TRUE, fileEncoding="latin1")

# Ensure we have a matrix
#mydata <- na.omit(as.matrix(boston))
mydata <- na.omit(as.matrix(WDBC[,3:32]))
#mydata <- as.matrix(mtcars[,c(1:7,10,11)])
```

Initial Correlation Matrix

```
# Simple correlation matrix calculation
# TODO: user chooses method
mydata.cor <- cor(mydata, method = c("pearson"), use = "complete.obs")
#mydata.cor <- cor(mydata, method = c("kendall"), use = "complete.obs")
#mydata.cor <- cor(mydata, method = c("spearman"), use = "complete.obs")

# # Optional
# corrplot(mydata.cor)
#
# palette <- colorRampPalette(c("green", "white", "red")) (20)
# heatmap.2(x = mydata.cor, col = palette, symm = TRUE)

# Reorder correlation matrix based on: https://rdrr.io/cran/lessR/man/corReorder.html
mydata.cor.ro <- corReorder(mydata.cor)
```



Distance Matrix Calculation

Correlation can often be used as a distance metric for hierarchical clustering; see e.g. this discussion.

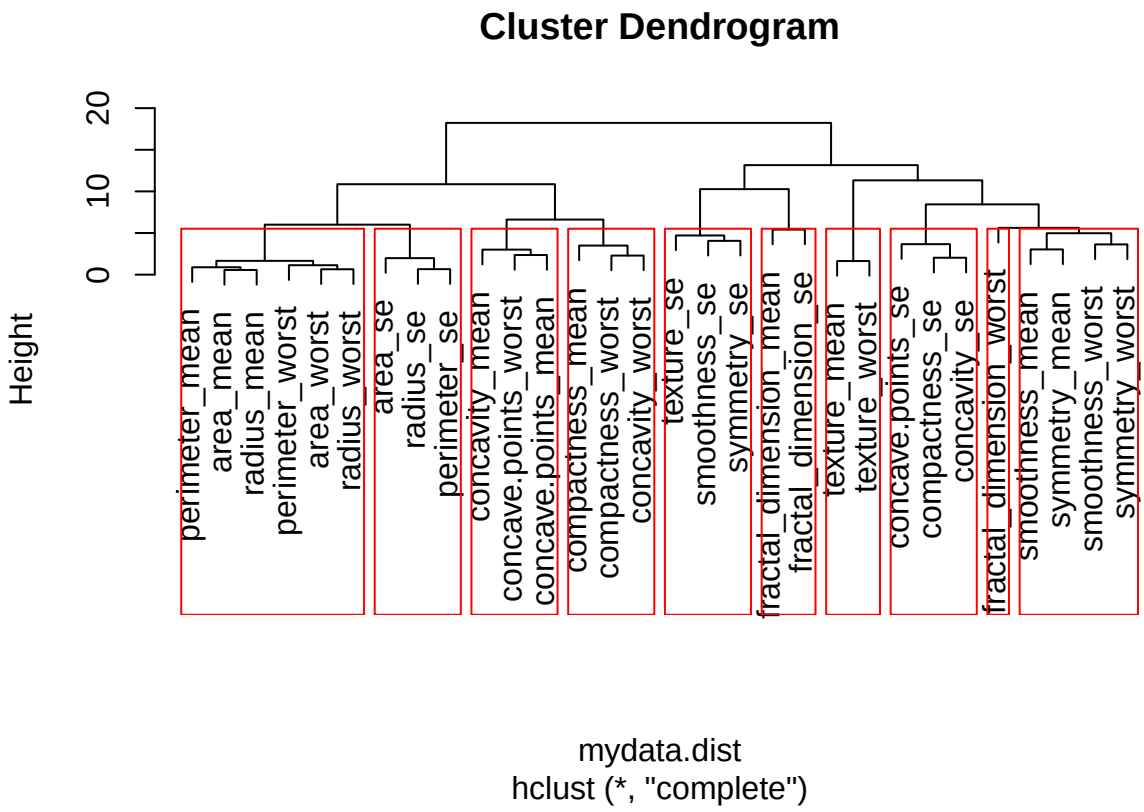
We're using the `dist()` function to provide a number of different distance options (default is *euclidean* distance).

```
# # Simplistic distance matrix (Absolute value et.al.)
# mydata.cor.ro.1 <- abs(mydata.cor.ro) - 1

# Distance Matrix via: https://stat.ethz.ch/R-manual/R-devel/library/stats/html/dist.html
mydata.dist <- dist(mydata.cor.ro, upper = TRUE, method = "manhattan")
```

```
mydata.dist.sym <- as.matrix(mydata.dist )
```

Hiearchical/Agglomerative Clustering (NEW)

```
# See eg:
# * https://metudatascience.github.io/datascience/clustering.html
# * https://uc-r.github.io/hc_clustering

# Distance calculated above
mydata.hclust <- hclust(mydata.dist)
plot(mydata.hclust)

# NOTE! These clusters are in numerical order, not feature order!
mydata.hclust.groups <- cutree(mydata.hclust, k=10) # Usually 3 is used for mtcars demos
rect.hclust(mydata.hclust, k=10, border="red")
```



PCA on Distance Matrix

```
# PCA
#mydata.dist.sym.pca <- prcomp(mydata.dist.sym,scale=TRUE)
mydata.dist.sym.pca <- prcomp(mydata.dist.sym) # Un-scaled gives results closer to source article

# Plottable version:
mydata.dist.sym.pca.plot <- as.data.frame(mydata.dist.sym.pca$x[,1:2])
```

Finding the Clusters

Write the cluster assignments and the feature names into the PCA data frame.

```
# Pull out our clusters (calculated)
mydata.dist.sym.pca.plot$cluster <- as.factor(mydata.hclust.groups)
mydata.dist.sym.pca.plot$name <- rownames(mydata.cor.ro)
```

Finding the Graph!

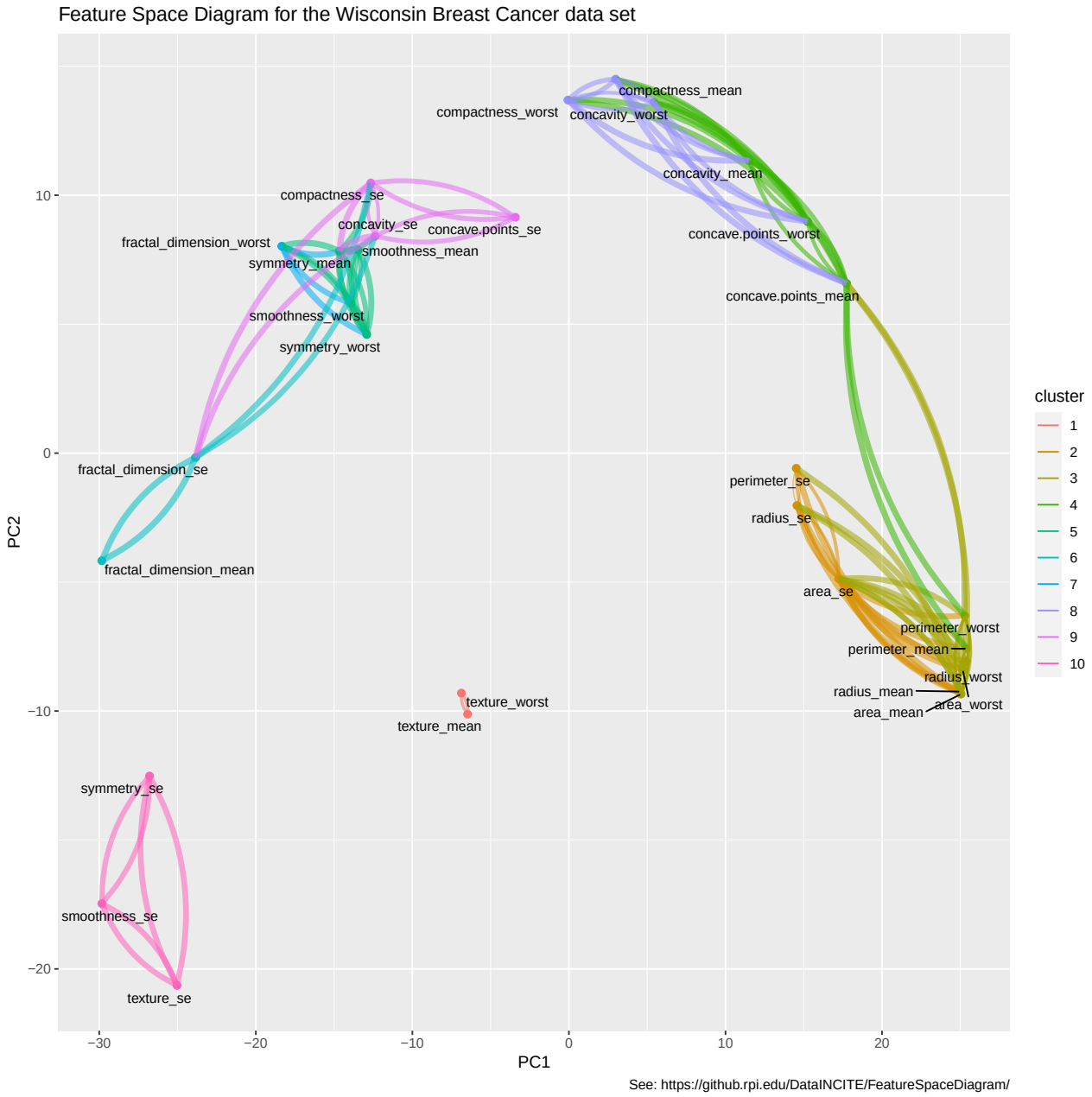
This is the hard part...

```
# Determine connectivity!
# This selects the nodes (features) we want to connect
threshold <- 5.5
selector <- ((abs(mydata.dist.sym) <= threshold ) * 1) # Filter out lines greater than threshold distan
for (i in 1:length(mydata.dist.sym.pca.plot$cluster)) {selector[,i] <- selector[,i] * as.numeric(mydata
diag(selector) <- 0

filtered <- abs(mydata.dist.sym)

filtered[ filtered <= threshold ] <- 0

# Get the function for re-scaling the line weights
# Line thicknesses will automatically range from 1-10
range <- c(1,10)
domain <- c(min(filtered[filtered > 0]),max(filtered))
line.lm <- lm(range~domain)

adjust <- function(x){
  # stupid lm trick to scale
    line.lm$coefficients[[2]] * x + line.lm$coefficients[[1]]
  }

# Repeat selector matrix, this time for strength of relationships
# Select the nodes (features) we want to connect
thickness <- ((abs(mydata.dist.sym) <= threshold ) * 1) # Filter out lines greater than threshold dista
diag(thickness) <- 0  # remove diagonal

# Use distances as thicknesses for the arcs we're keeping
for (i in 1:length(mydata.dist.sym.pca.plot$cluster)) {
  for (j in 1:length(thickness[,i])) {
    if (thickness[j,i] != 0 ) {
#       thickness[j,i] <- adjust(thickness[j,i] * abs(mydata.dist.sym[j,i]))
      thickness[j,i] <- thickness[j,i] * abs(mydata.dist.sym[j,i])
    } else { thickness[j,i] <- 0}
  }
}

diag(thickness) <- 0

# Switching to networks
# Create igraph structure
network <- graph_from_adjacency_matrix(selector, weighted = TRUE)
network.t <- graph_from_adjacency_matrix(thickness, weighted = TRUE)
```

```
# Gets us our edges!
mydata.edges <- get.data.frame(network) %>% # this is where "weight" is introduced
  dplyr::rename(Cluster = weight)

mydata.edges.t <- get.data.frame(network.t) %>% # this is where "weight" is introduced
  dplyr::rename(thickness = weight) %>%
#  mutate(thickness = as.integer(thickness))
  mutate(thickness = thickness)


# replace `from` with X1 and Y1, and `to` with X2 and Y2
# These will be our line segments!
mydata.segments <- mydata.edges %>%
  left_join(mydata.dist.sym.pca.plot, by=c("from"="name")) %>%
  dplyr::select(-cluster) %>%
  mutate(X1=PC1, Y1=PC2) %>%
  dplyr::select(-PC1, -PC2)

mydata.segments <- mydata.segments %>%
  left_join(mydata.dist.sym.pca.plot, by=c("to"="name")) %>%
  dplyr::select(-cluster) %>%
  mutate(X2=PC1, Y2=PC2) %>%
  dplyr::select(-PC1, -PC2)

mydata.segments <- mydata.segments %>%
  dplyr::rename(cluster = Cluster)

mydata.dist.sym.pca.plot$cluster <- as.factor(mydata.dist.sym.pca.plot$cluster)

# NEW: cluster used for coloring
mydata.segments$cluster <- as.factor(mydata.segments$cluster)
#mydata.segments$thickness <- as.integer(mydata.edges.t$thickness)
mydata.segments$thickness <- mydata.edges.t$thickness
```

Building the Plot

```
# Adding line segments to PCA plot
p <- ggplot(mydata.dist.sym.pca.plot, aes(x=PC1, y=PC2)) +
  geom_point(aes(color=cluster, size=6), show.legend = FALSE) +
  geom_curve(aes(x=X1, y=Y1, xend=X2, yend=Y2, color=cluster, size=thickness, alpha=0.4), curvature=0.2
  guides(alpha = FALSE) +
  scale_size(range = c(0.1, 2), guide = guide_none()) +
  geom_text_repel(aes(label=name),hjust=1, vjust=0, size=3) +
#  labs(title="Feature Space Diagram for the Boston Housing data set") +
  labs(title="Feature Space Diagram for the Wisconsin Breast Cancer data set") +
#  labs(title="Feature Space Diagram for the classic `mtcars` data set") +
  labs(caption = "See: https://github.rpi.edu/DataINCITE/FeatureSpaceDiagram/")

p
```




```
ggsave(filename="WDBC_fsd.png", plot = p)
```

```
## Saving 10 x 10 in image
```

Sanity check: Biplot on the original dataset

Usually we review the feature space using a PCA biplot. Let's see how it compares!

```
mydata.pca <- prcomp(na.omit(mydata), center = TRUE, scale. = TRUE)

ggbiplot(mydata.pca) +
    xlim(-2.5,2.5) +
  ylim(-2.5,2.5)
```