R Notebook

This is an R Markdown Notebook. When you execute code within the notebook, the results appear beneath the code.

Try executing this chunk by clicking the *Run* button within the chunk or by placing your cursor inside it and pressing *Ctrl+Shift+Enter*.

1. Import the data file Cereals.dat in the R environment.

```
cereal_data <- as.data.frame(read.table("Cereals.dat", header = TRUE))
```

2. Compute the Euclidean distance matrix for the cereals, using the information on calories and all seven cereal components, and standardizing the variables prior to the calculation of the distance matrix. You can use the R functions scale and dist for this purpose. Paste the distance matrix of the first 5 specimens into your report.

```
numerical_values <- cereal_data[,3:ncol(cereal_data)]
sd_numerical <- scale(numerical_values)
eucl_dist <- as.matrix(dist(sd_numerical, upper = T))
colnames(eucl_dist) <- cereal_data$Brand
rownames(eucl_dist) <- cereal_data$Brand
eucl_dist[1:5,1:5]
```

```
##                ACCheerios Cheerios CocoaPuffs CountChocula GoldenGrahams
## ACCheerios       0.000000 4.389923  1.8800970    1.8678873      2.413378
## Cheerios         4.389923 0.000000  5.5151628    5.4964619      4.869285
## CocoaPuffs       1.880097 5.515163  0.0000000    0.1512638      1.700201
## CountChocula     1.867887 5.496462  0.1512638    0.0000000      1.720269
## GoldenGrahams    2.413378 4.869285  1.7002007    1.7202688      0.000000
```

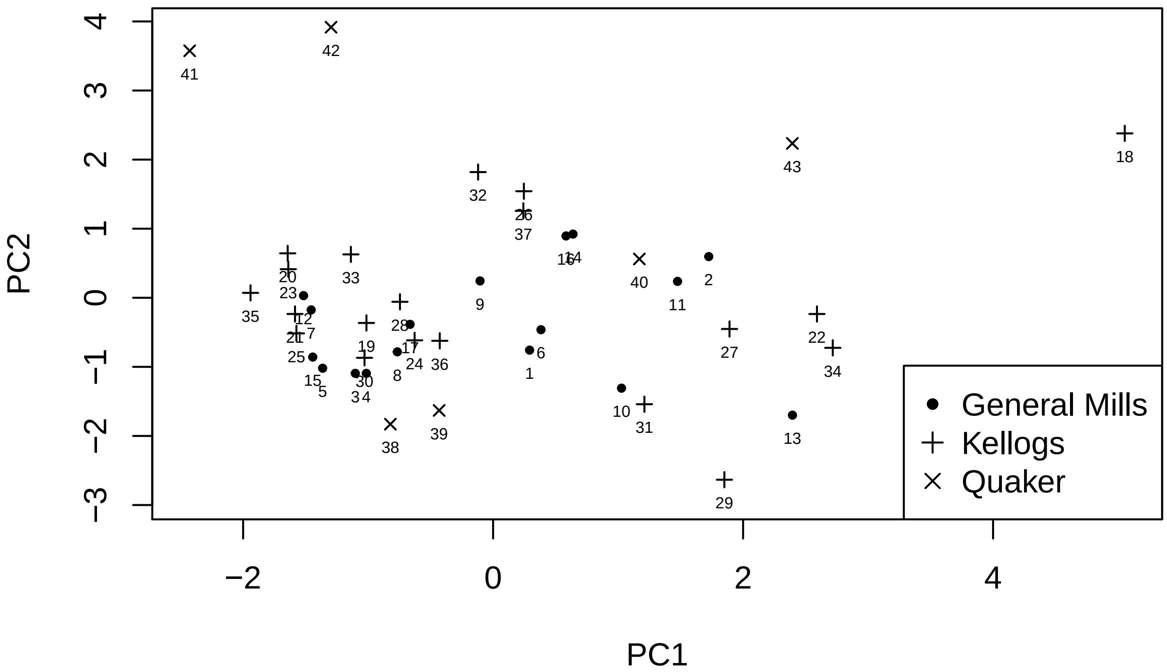
3. Perform a metric MDS of the data, using the cmdscale program. Plot the two-dimensional solution, and label the cereals with a number or abbreviated name. Use a different colour or symbol to label each cereal according to its manufacturer.

```
## 2D approx
multidim_scale <- cmdscale(eucl_dist, k = 2, eig = T, x.ret = T)
# multidim_scale$eig
multidim_scale$points
```

```
##                                   [,1]        [,2]
## ACCheerios                   0.2920929 -0.75770232
## Cheerios                     1.7253617  0.59636554
## CocoaPuffs                  -1.1023802 -1.09423804
## CountChocula                -1.0141779 -1.09422268
## GoldenGrahams               -1.3637169 -1.01992530
```

```
## HoneyNutCheerios         0.3830204 -0.46251880
## Kix                     -1.4553395 -0.17580753
## LuckyCharms             -0.7663923 -0.78292282
## MultiGrainCheerios      -0.1046586  0.24365371
## OatmealRaisinCrisp       1.0274180 -1.30885039
## RaisinNutBran            1.4756058  0.23736782
## TotalCornFlakes         -1.5165114  0.03102544
## TotalRaisinBran          2.3950722 -1.69910091
## TotalWholeGrain          0.6394065  0.92164978
## Trix                    -1.4427457 -0.85828746
## Cheaties                 0.5834790  0.89499876
## WheatiesHoneyGold       -0.6639221 -0.38460404
## AllBran                  5.0538101  2.37958311
## AppleJacks              -1.0134288 -0.36481097
## CornFlakes              -1.6433194  0.64307471
## CornPops                -1.5847842 -0.23452487
## CracklinOatBran          2.5909668 -0.23496220
## Crispix                 -1.6386340  0.41347394
## FrootLoops              -0.6279756 -0.61661365
## FrostedFlakes           -1.5737651 -0.51562217
## FrostedMiniWheats        0.2460563  1.54194630
## FruitfulBran             1.8913000 -0.45166110
## JustRightCrunchyNuggets -0.7452865 -0.05832696
## MueslixCrispyBlend       1.8501851 -2.63357241
## NutNHoneyCrunch         -1.0287339 -0.86926396
## NutriGrainAlmondRaisin   1.2105764 -1.54070672
## NutriGrainWheat         -0.1199946  1.81920111
## Product19               -1.1377649  0.62812821
## RaisinBran               2.7176797 -0.72472058
## RiceKrispies            -1.9407368  0.07067002
## Smacks                  -0.4266649 -0.62320011
## SpecialK                 0.2421091  1.25853930
## CapNCrunch              -0.8216995 -1.83011848
## HoneyGrahamOhs          -0.4317560 -1.63050275
## Life                     1.1700906  0.56149462
## PuffedRice              -2.4276015  3.57484827
## PuffedWheat             -1.2964803  3.91626288
## QuakerOatmeal            2.3942400  2.23450370
```

```r
multidim_scale_points_G <- multidim_scale$points[cereal_data$Manufacturer == "G",]
plot(multidim_scale_points_G[,1], multidim_scale_points_G[,2], xlab = "PC1", ylab = "PC2", pch = 20, xli
multidim_scale_points_K <- multidim_scale$points[cereal_data$Manufacturer == "K",]
points(multidim_scale_points_K[,1], multidim_scale_points_K[,2], pch = 3, cex = 0.75)
multidim_scale_points_Q <- multidim_scale$points[cereal_data$Manufacturer == "Q",]
points(multidim_scale_points_Q[,1], multidim_scale_points_Q[,2], pch = 4, cex = 0.75)
text(x = multidim_scale$points[,1], y = multidim_scale$points[,2], labels = seq(from = 1, to = nrow(cere
legend("bottomright", c("General Mills", "Kellogs", "Quaker"), pch = c(20, 3, 4))
```



4. Which pair of cereals is, according to the two-dimensional solution of the analysis, the most similar?

```
approx_dist <- as.matrix(dist(multidim_scale$points, upper = T))
approx_dist_non_matrix <- dist(multidim_scale$points)
which(approx_dist == min(approx_dist_non_matrix), arr.ind = T)
```

```
##                  row col
## Cheaties          16  14
## TotalWholeGrain   14  16
```

```
cereal_data$Manufacturer[[14]]
```

```
## [1] G
## Levels: G K Q
```

```
cereal_data$Manufacturer[[16]]
```

```
## [1] G
## Levels: G K Q
```

5. Which pair of cereals would you, according to the two-dimensional solution of the analysis, classify as most distinct?

```
which(approx_dist == max(approx_dist), arr.ind = T)
```

```
##            row col
## PuffedRice  41  18
## AllBran     18  41
```

```r
cereal_data$Manufacturer[[41]]
```

```
## [1] Q
## Levels: G K Q
```

```r
cereal_data$Manufacturer[[18]]
```

```
## [1] K
## Levels: G K Q
```

6. Is it possible to find a configuration of the 43 cereals in k dimensions that will represent the original distance matrix exactly? Why or why not? If so, how many dimensions would be needed to obtain this exact representation?

```r
## K dimension matrix
#Yes, 6
```

7. Report the eigenvalues of the solution, and calculate the goodness-of-fit of the two-dimensional solution.

```r
# multidim_scale$GOF # I just use this to check that the computed value matches

B <- t(multidim_scale$x) %*% multidim_scale$x
B.eigen <- eigen(B)
sum(B.eigen$values[1:2]) / sum(B.eigen$values)
```

```
## [1] 0.6996236
```

```r
B <- t(sd_numerical) %*% sd_numerical

B.eigen <- eigen(B)

eigenvalues <- B.eigen$values
eigenvalues
```

```
## [1] 106.997917  77.899995  74.290858  36.467901  20.886634  15.011995
## [7]   2.541141   1.903558
```

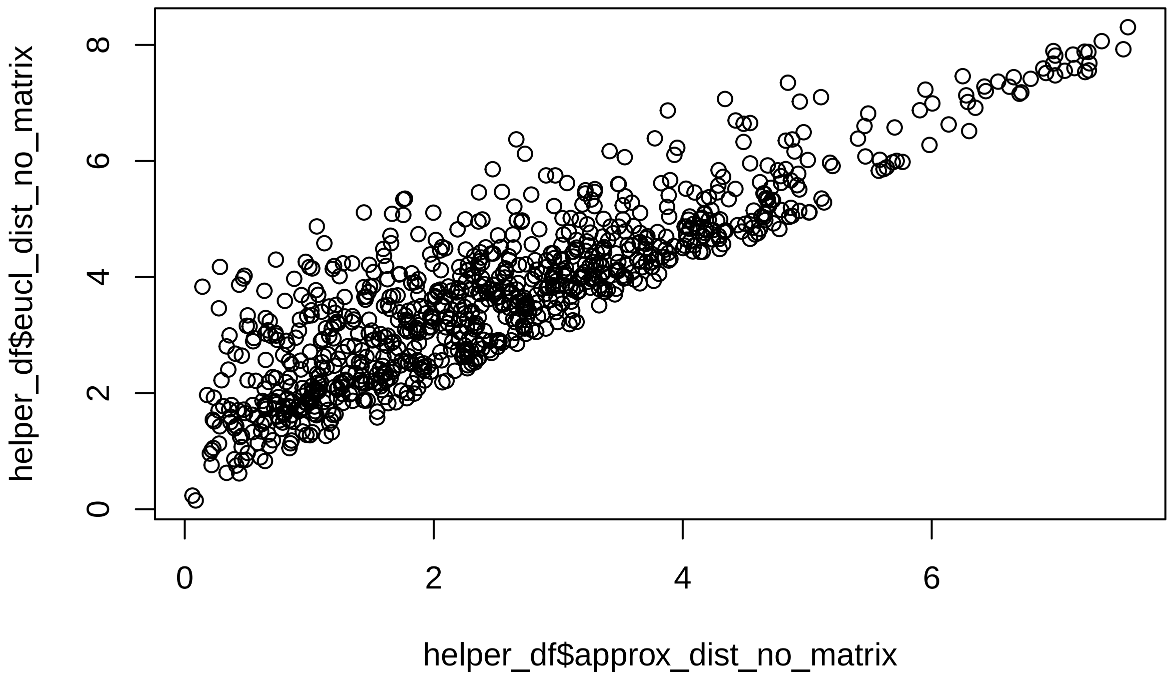
```r
GOF <- sum(B.eigen$values[1:2]) / sum(B.eigen$values)
GOF
```

```
## [1] 0.5502914
```

```r
## Are there 0 Eigen? Why not?
# No.
```

```r
## Plot distances and do regression.
eucl_dist_no_matrix <- dist(sd_numerical)
approx_dist_no_matrix <- dist(multidim_scale$points)
helper_df <- data.frame(cbind(approx_dist_no_matrix, eucl_dist_no_matrix))

plot(helper_df$approx_dist_no_matrix, helper_df$eucl_dist_no_matrix)
```




```r
linearMod <- lm(approx_dist_no_matrix ~ eucl_dist_no_matrix, data = helper_df)
summary(linearMod)
```

```
##
## Call:
## lm(formula = approx_dist_no_matrix ~ eucl_dist_no_matrix, data = helper_df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.6731 -0.4088  0.1261  0.5666  1.2678
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)         -0.79005    0.06698  -11.79   <2e-16 ***
## eucl_dist_no_matrix  0.89728    0.01674   53.59   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.75 on 901 degrees of freedom
## Multiple R-squared:  0.7612, Adjusted R-squared:  0.7609
## F-statistic:  2871 on 1 and 901 DF,  p-value: < 2.2e-16
```

```r
## Non-metric
non_metric <- MASS::isoMDS(eucl_dist)
```

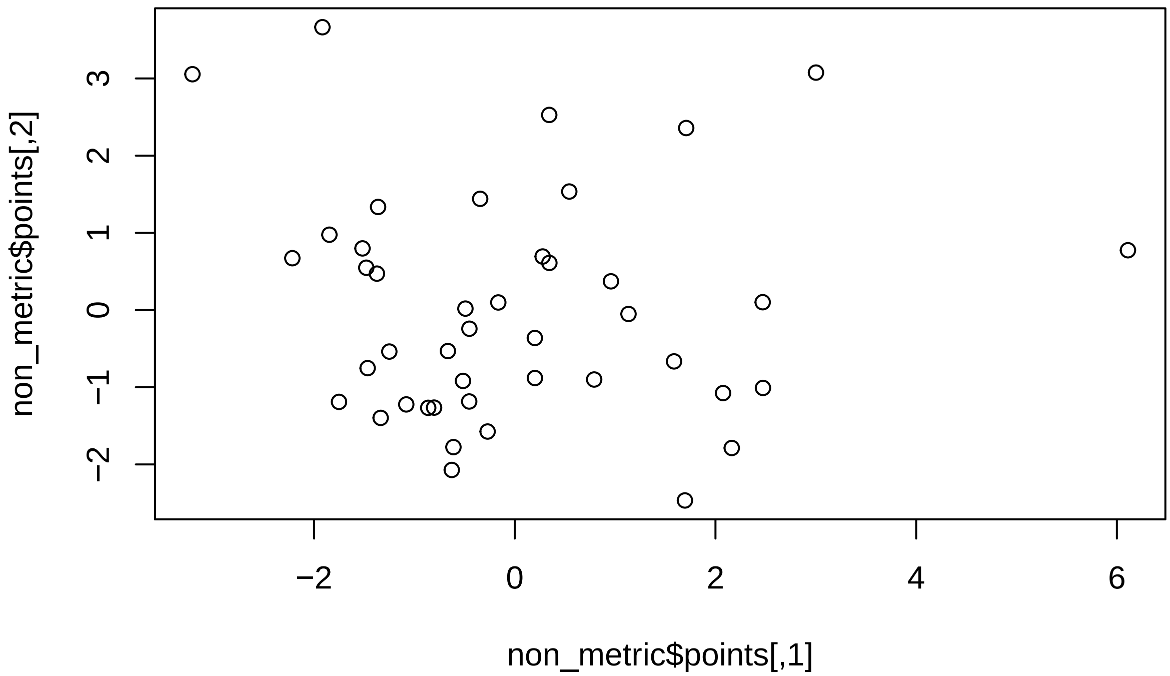
```
## initial  value 23.424101
## iter   5 value 17.264581
## final  value 17.035316
## converged
```

```r
# todo make plot nicer
non_metric
```

```
## $points
```

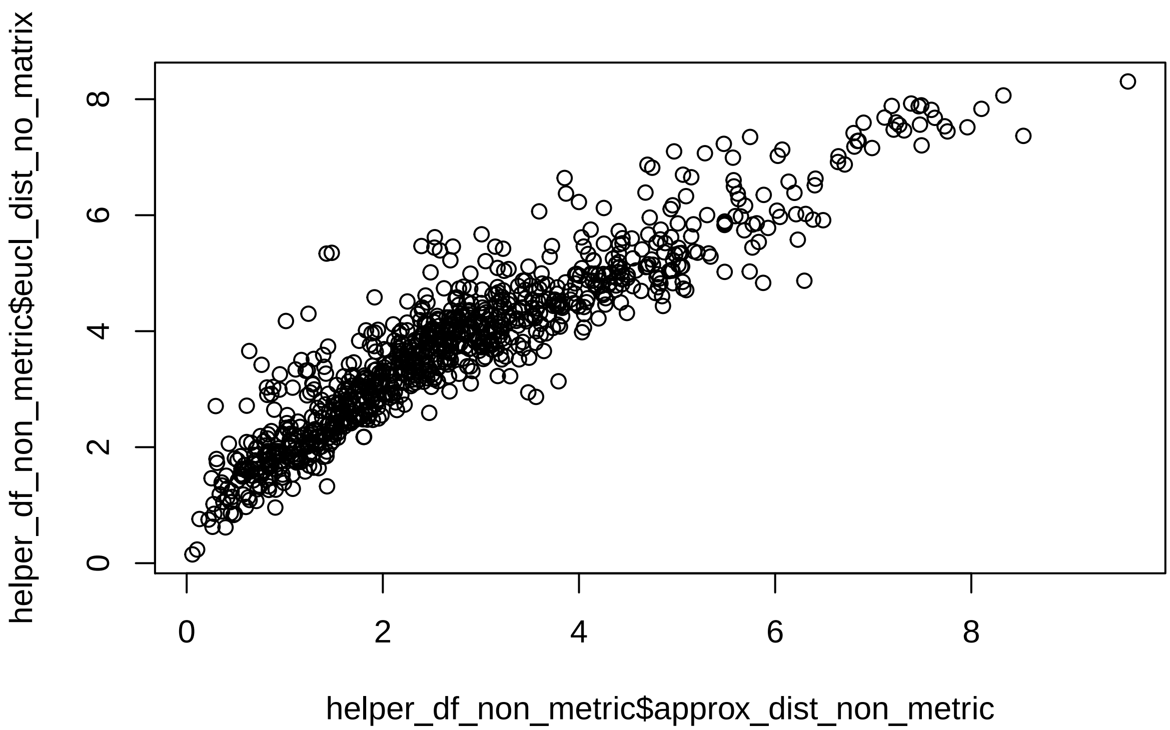
```
##                                [,1]        [,2]
## ACCheerios                0.2005806 -0.88061675
## Cheerios                  1.7084419  2.35732977
## CocoaPuffs               -0.8625012 -1.26657783
## CountChocula             -0.8039042 -1.26404028
## GoldenGrahams            -1.2501174 -0.53783559
## HoneyNutCheerios          0.1999211 -0.36191749
## Kix                      -1.4796079  0.54799703
## LuckyCharms              -0.5161610 -0.91808726
## MultiGrainCheerios       -0.1644365  0.09815065
## OatmealRaisinCrisp        0.7912075 -0.89984482
## RaisinNutBran             1.1333671 -0.05109934
## TotalCornFlakes          -1.3738189  0.47189588
## TotalRaisinBran           2.1621938 -1.78706823
## TotalWholeGrain           0.3446566  0.61033159
## Trix                     -1.0812367 -1.22234116
## Cheaties                  0.2778788  0.69338372
## WheatiesHoneyGold        -0.4523082 -0.24205162
## AllBran                   6.1104242  0.77471747
## AppleJacks               -1.3367020 -1.39694588
## CornFlakes               -1.8472647  0.97576752
## CornPops                 -1.7514072 -1.18971525
## CracklinOatBran           2.4704890  0.10144289
## Crispix                  -1.5175264  0.79857229
## FrootLoops               -0.4540746 -1.18457259
## FrostedFlakes            -1.4663497 -0.75223751
## FrostedMiniWheats         0.5430560  1.53498869
## FruitfulBran              2.0755709 -1.07572964
## JustRightCrunchyNuggets -0.4917421  0.01859970
## MueslixCrispyBlend        1.6957029 -2.46646373
## NutNHoneyCrunch          -0.6665476 -0.53141827
## NutriGrainAlmondRaisin    1.5871893 -0.66538050
## NutriGrainWheat          -0.3447366  1.44068816
## Product19                -1.3620348  1.33510101
## RaisinBran                2.4732957 -1.00905192
## RiceKrispies             -2.2163288  0.67165044
## Smacks                   -0.6280480 -2.07129654
## SpecialK                  0.3435431  2.52710283
## CapNCrunch               -0.6111739 -1.77569974
## HoneyGrahamOhs           -0.2705733 -1.57374071
## Life                      0.9584552  0.37200455
## PuffedRice              -3.2121530  3.05468439
## PuffedWheat              -1.9167401  3.66367824
## QuakerOatmeal             3.0015211  3.07564581
##
## $stress
## [1] 17.03532
```

```
plot(non_metric$points)
```




```r
## Check two closest
approx_dist_non_metric <- dist(non_metric$points)
helper_df_non_metric <- data.frame(cbind(approx_dist_non_metric, eucl_dist_no_matrix))
```

```r
## Regression
plot(helper_df_non_metric$approx_dist_non_metric, helper_df_non_metric$eucl_dist_no_matrix)
```



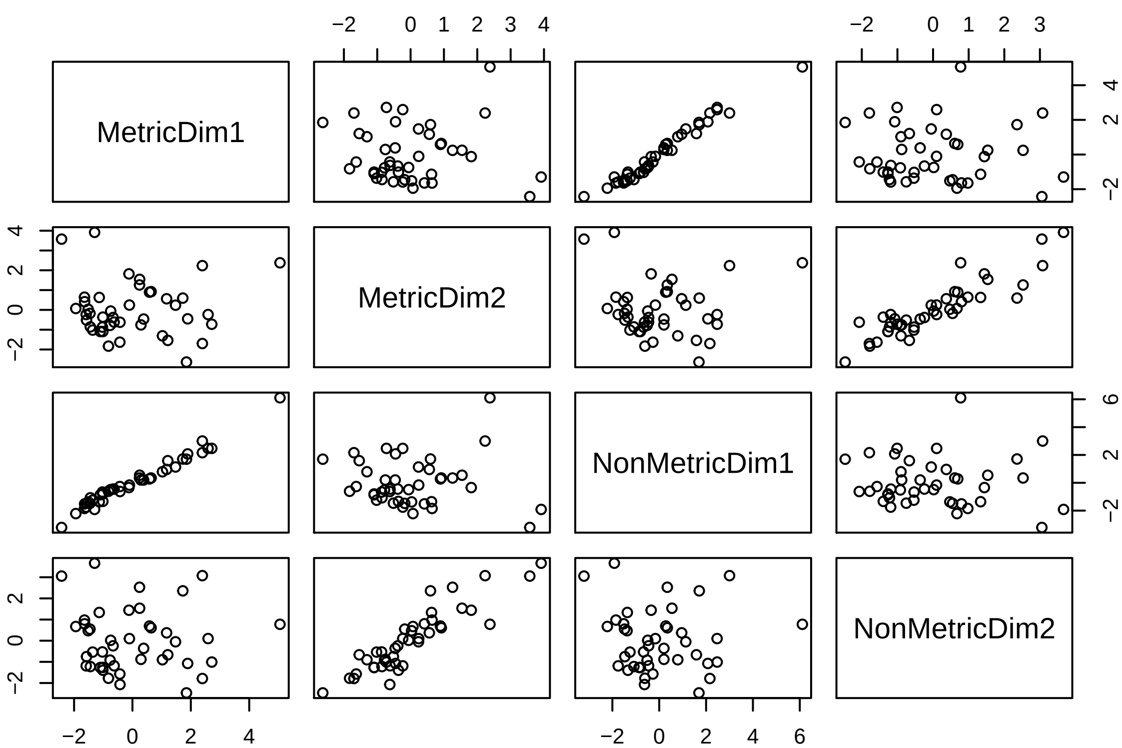

```r
linearMod_non_metric <- lm(approx_dist_non_metric ~ eucl_dist_no_matrix, data = helper_df)
summary(linearMod_non_metric)
```

```
##
## Call:
## lm(formula = approx_dist_non_metric ~ eucl_dist_no_matrix, data = helper_df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.95725 -0.31453  0.03293  0.33462  2.38912
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)         -1.02419    0.05490  -18.65   <2e-16 ***
## eucl_dist_no_matrix  1.01300    0.01373   73.80   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6148 on 901 degrees of freedom
## Multiple R-squared:  0.8581, Adjusted R-squared:  0.8579
## F-statistic:  5447 on 1 and 901 DF,  p-value: < 2.2e-16
```

Stress

Scatterplot

```r
helper_scatter <- data.frame(cbind(multidim_scale$points, non_metric$points))
colnames(helper_scatter) <- c("MetricDim1","MetricDim2","NonMetricDim1","NonMetricDim2")
plot(helper_scatter)
```




```r
round(cor(helper_scatter), digits = 3)
```

```
##               MetricDim1 MetricDim2 NonMetricDim1 NonMetricDim2
## MetricDim1         1.000      0.000         0.983        -0.028
## MetricDim2         0.000      1.000        -0.035         0.899
## NonMetricDim1      0.983     -0.035         1.000        -0.056
## NonMetricDim2     -0.028      0.899        -0.056         1.000
```

```
## Why sd?
```

Add a new chunk by clicking the *Insert Chunk* button on the toolbar or by pressing *Ctrl+Alt+I*.

When you save the notebook, an HTML file containing the code and output will be saved alongside it (click the *Preview* button or press *Ctrl+Shift+K* to preview the HTML file).