Introduction to clustPro

Christian D. Peikert

2019-05-14

clustPro

Install

`clustpro` is not on CRAN, so we'll use `devtools::install_github` to install.

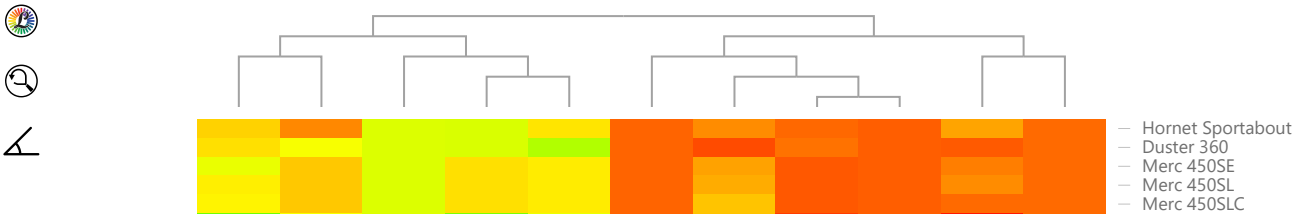
```
#devtools::install_github("cpeikert/clustpro")
```

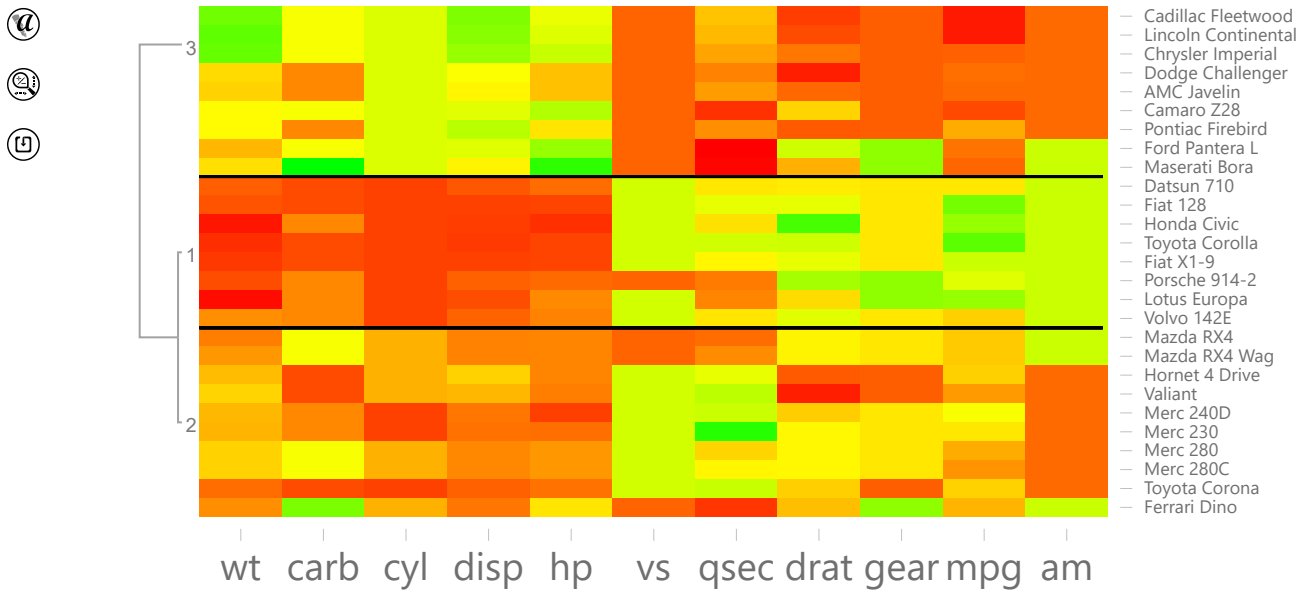
Examples

```
# suppressMessages is used to turn of loading package messages
suppressMessages(library('clustpro'))
unique_ids <<- paste0('vignette_id_',sprintf("%06d", 1:100000))

get_unique_id <- function(){
  id <- unique_ids[1]
  unique_ids <<- unique_ids[-1]
  return(id)
}
```

Basic Usage

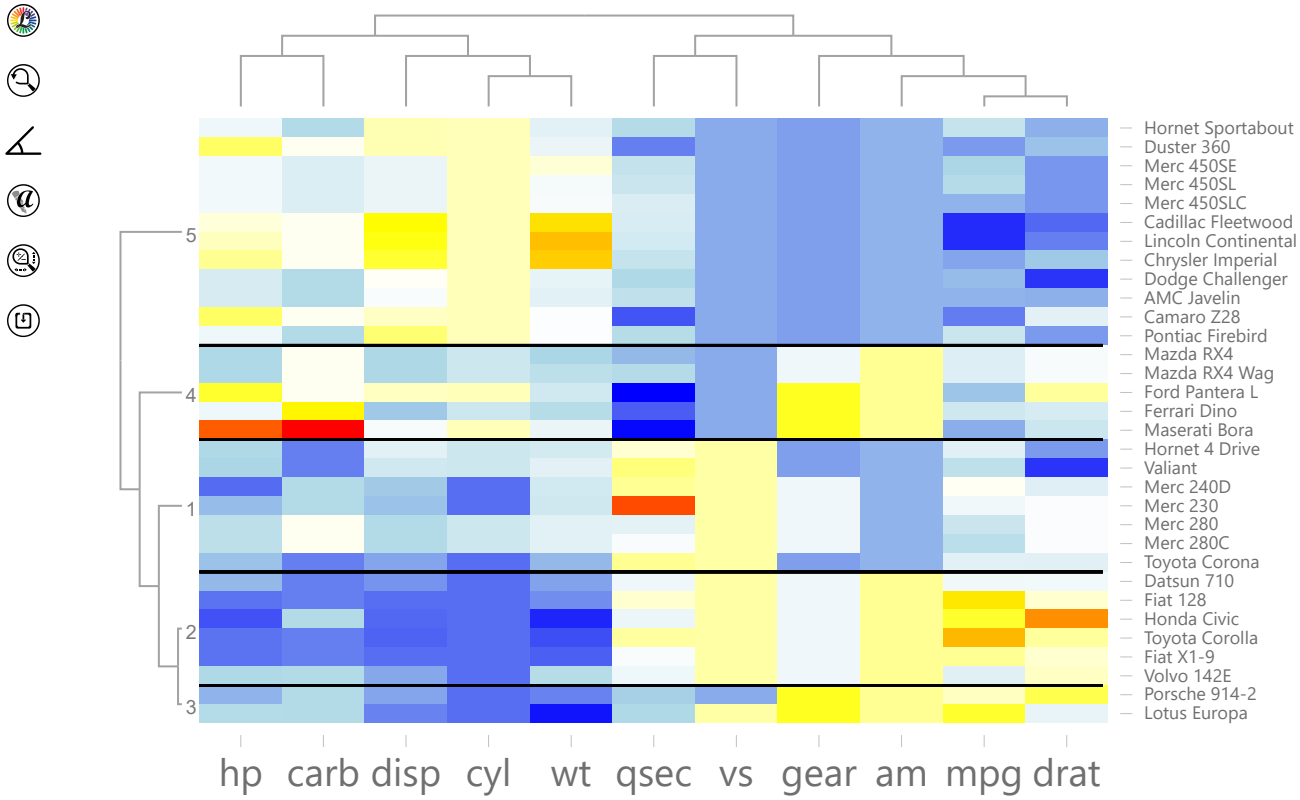
```
seed <- 1234
df_mtcars <- datasets::mtcars
df_data <- as.data.frame(scale(df_mtcars))

clustpro(matrix = df_data, elementId = get_unique_id())
```



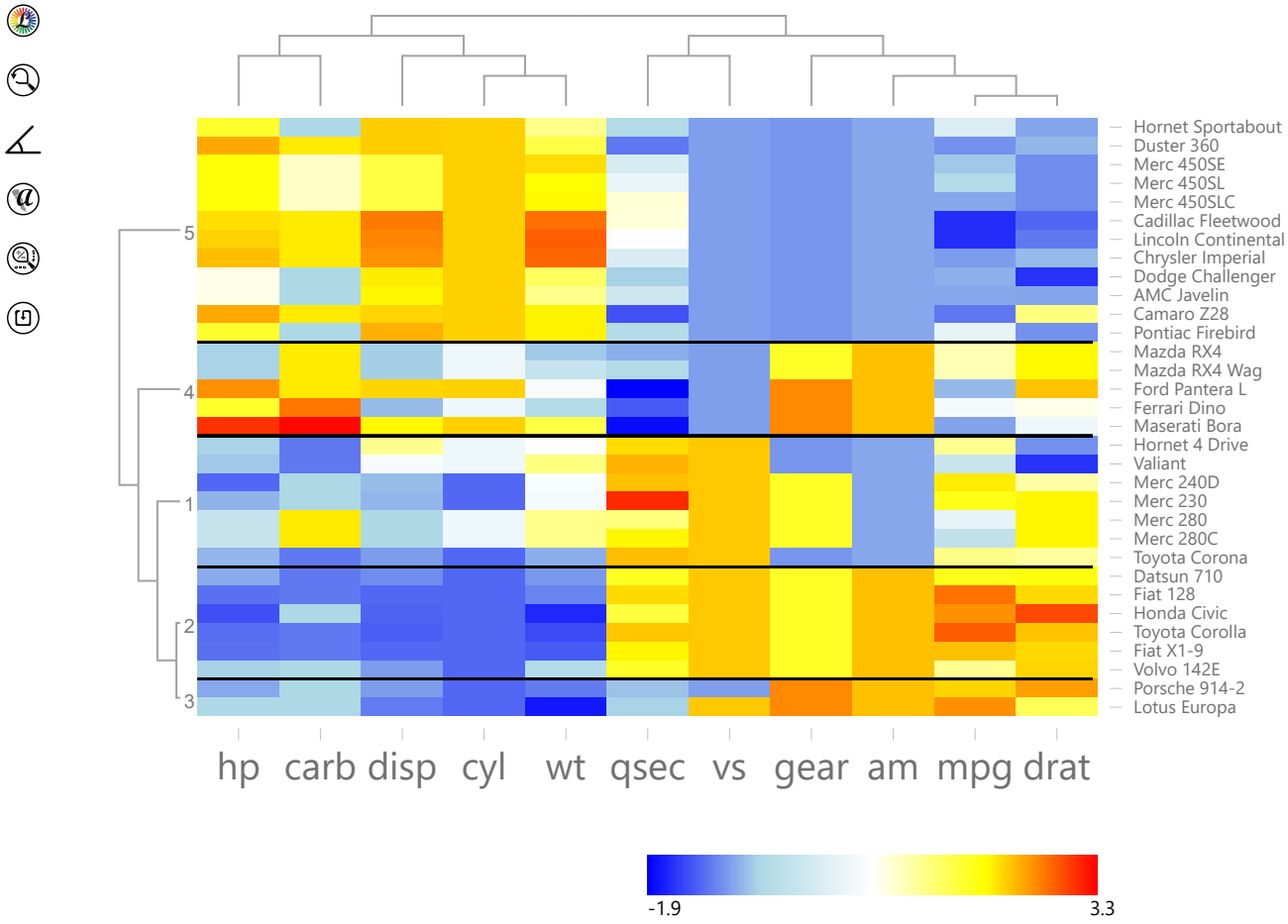


Color Setup

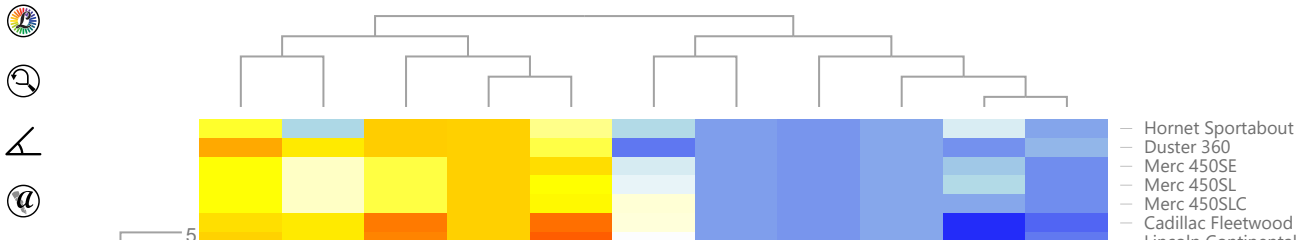
```
# setting of individual colors
  color_list <- c("blue", "lightblue", "white", "yellow", "red")
  color_legend01 <-
    clustpro::setHeatmapColors(data = df_data,
                     color_list = color_list,
                     auto = TRUE)

clustpro(matrix = df_data, color_legend = color_legend01, show_legend = TRUE ,seed = 1234, el
ementId = get_unique_id())
```

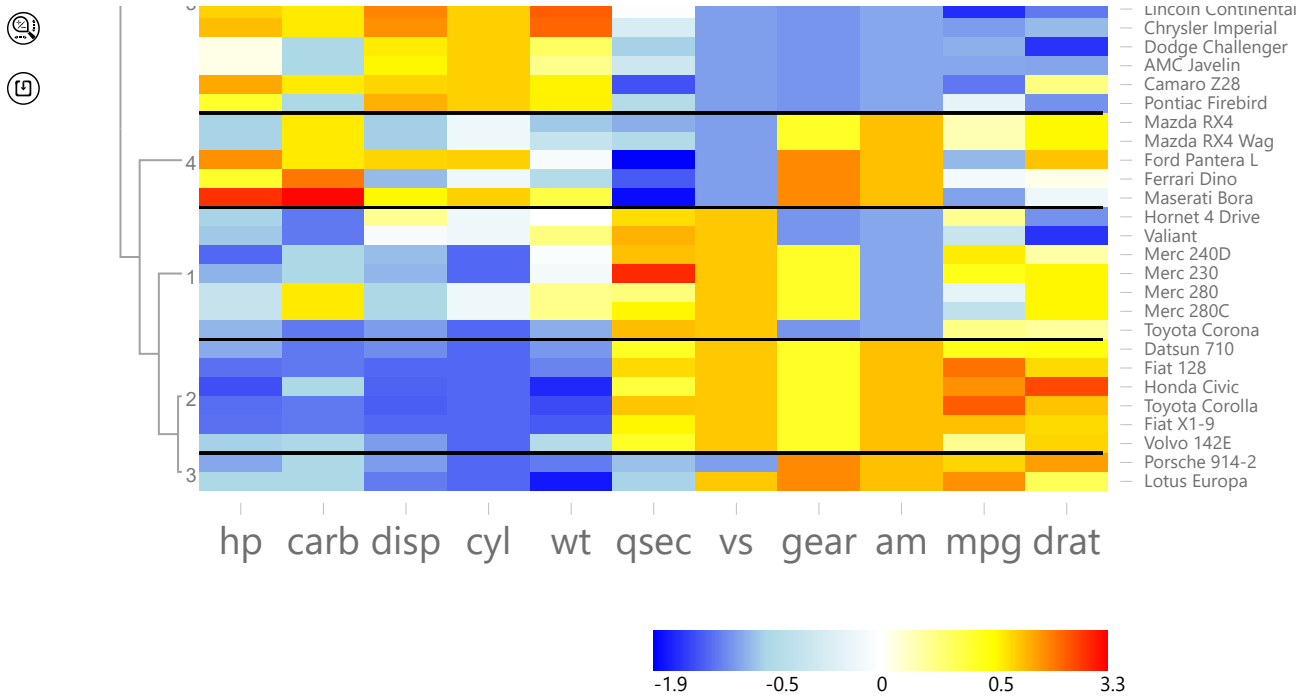



```
# setting of individual intervals
intervals <- c(-1.9,-0.5,0,0.5,3.3)
  color_legend02 <-
    clustpro::setHeatmapColors(data = df_data,
                     color_list = color_list,
                     intervals = intervals,
                     auto = FALSE)

clustpro(matrix = df_data, color_legend = color_legend02, show_legend = TRUE ,seed = 1234, el
ementId = get_unique_id())
```



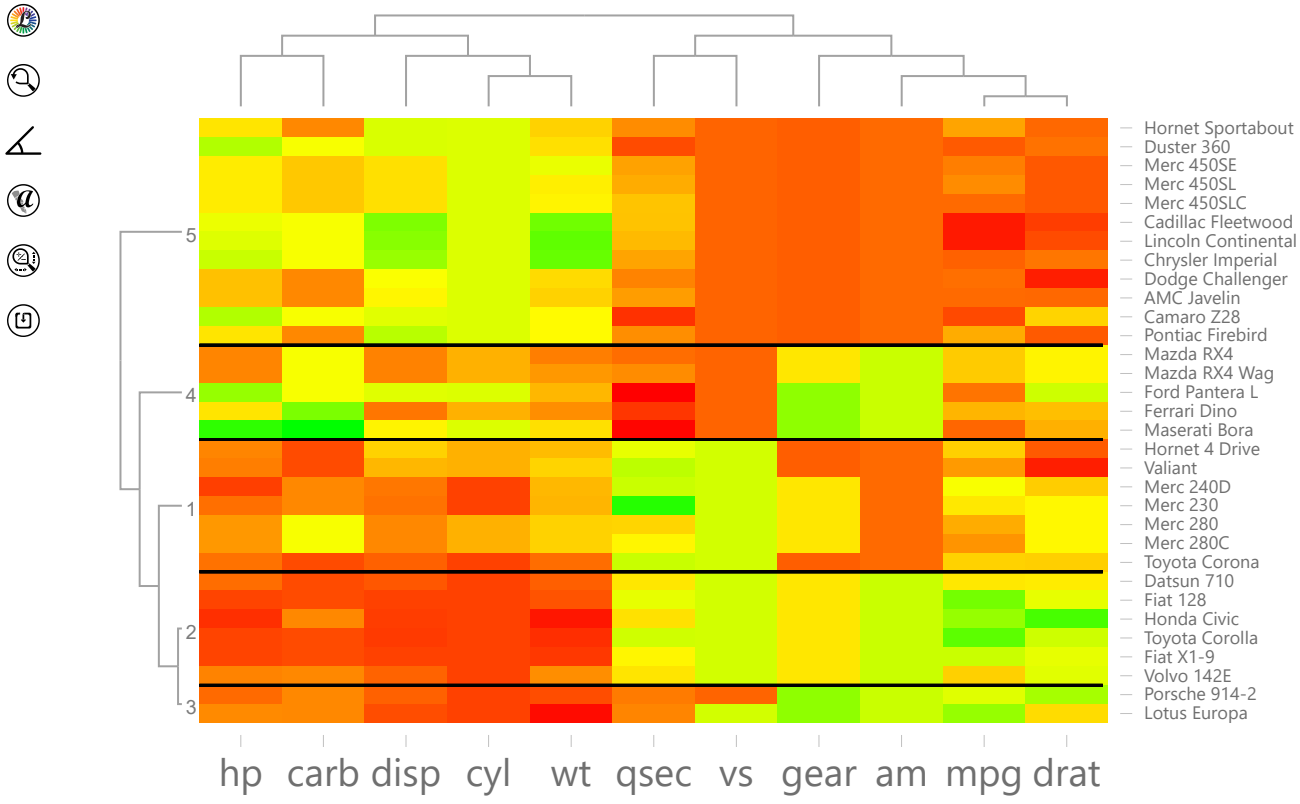

```
# setting of individual shown ticks in the legend
color_legend03 <- color_legend02
color_legend03$label_position <- c(-1.9,-0.5,0,0.5,3.3)

clustpro(matrix = df_data, color_legend = color_legend03, show_legend = TRUE ,seed = 1234, el
ementId = get_unique_id())
```



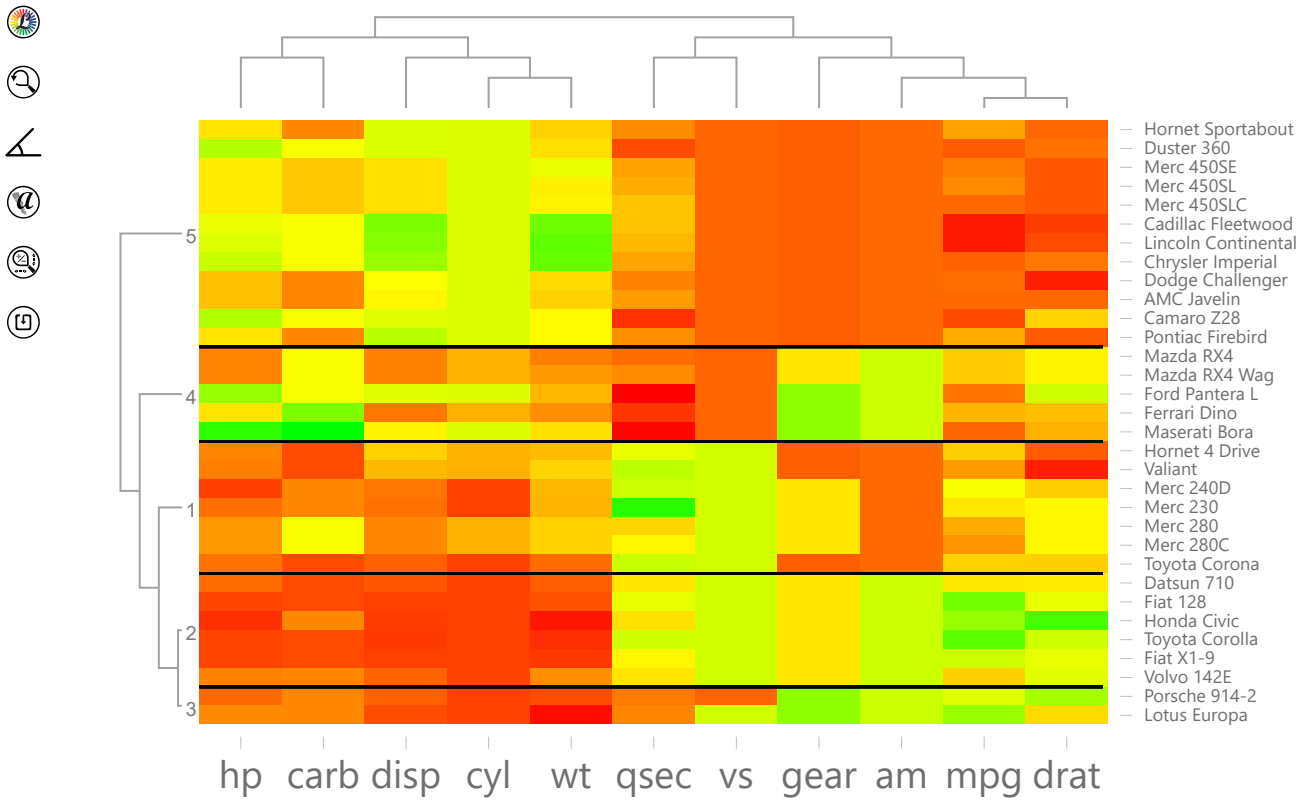


Adding Tooltips

```
# adding row id to the tooltips as well as a web link
  info_list <- list()
  info_list[['id']]  <- rownames(df_data)
  info_list[['link']] <-
    paste('https://www.google.de/search?q=/', rownames(df_data), sep = '')

clustpro(matrix = df_data,seed = 1234, tooltip = info_list, elementId = get_unique_id())
```



```
# adding all matrix values as strings to the tooltip list
  df_mtcars02 <- df_mtcars
  colnames(df_mtcars02) <- paste0('info_', colnames(df_mtcars02))
  data_columns <- colnames(df_data)
  info_columns <- colnames(df_mtcars02)
  df_data_extended <- cbind(df_data, df_mtcars02)

  if (!is.null(info_columns)) {
    temp_list <- lapply(info_columns, function(x) {
      df_data_extended[, x]
    })
    names(temp_list) <-
      sapply(info_columns, function(x)
        stringr:::str_match(x, 'info_(.*)')[2])

    info_list02 <- c(info_list, temp_list)
  }

clustpro(matrix = df_data,seed = 1234, tooltip = info_list02, elementId = get_unique_id())
```

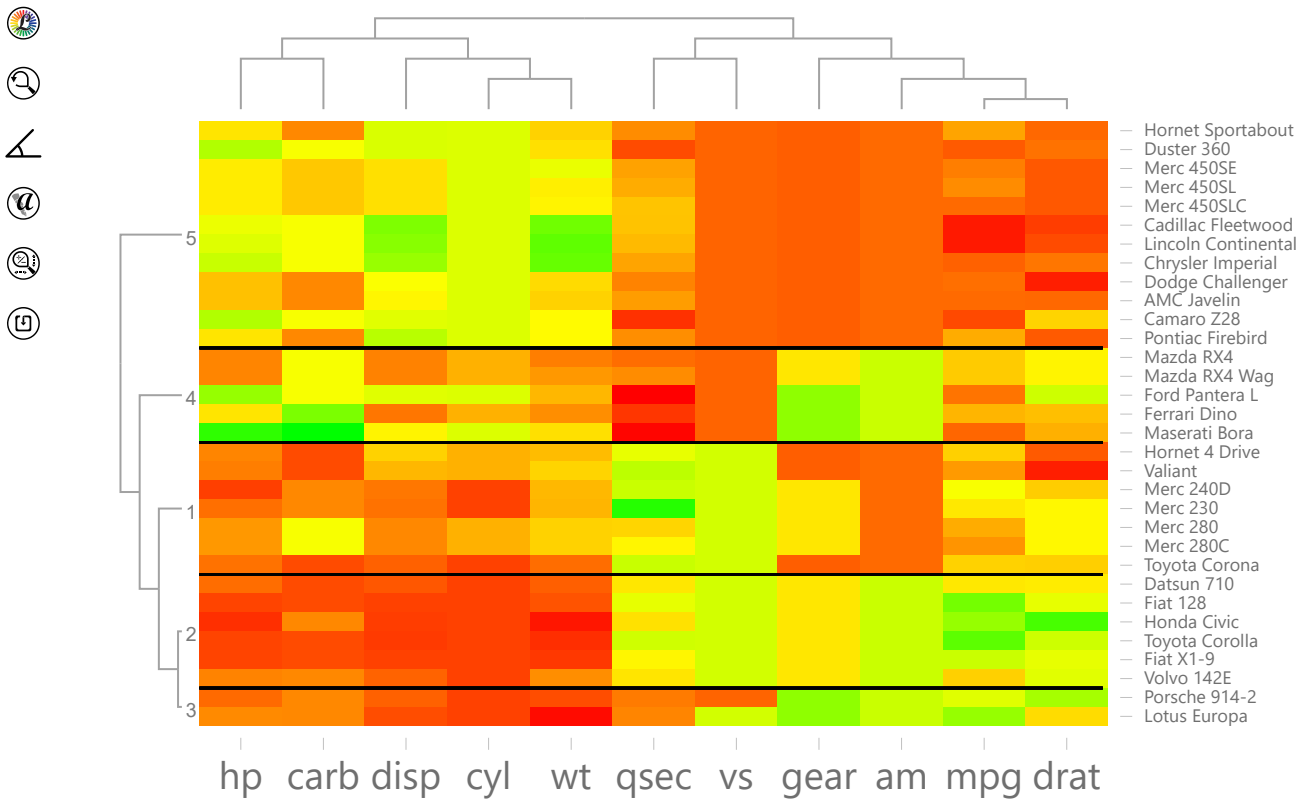


Clustering

```
# determine the best number of clusters using the DB-index. Minimal and maximal cluster size
 defining the rank of options.
db_object <- clustpro::get_best_k(matrix = df_data, min_k = 2, max_k = 10, method = 'kmeans',
seed = seed, cores = 4)

db_object$db_list
```

```
##    k     score withinerror
## 1  2 1.0506227   179.38258
## 2  3 1.0225064   133.28080
## 3  4 0.9386960    86.81903
## 4  5 1.1450902    81.75359
## 5  6 1.0717229    77.94270
## 6  7 0.8362264    64.69598
## 7  8 0.8271120    55.85866
## 8  9 0.8017703    43.52136
## 9 10 0.9180775    40.08108
```

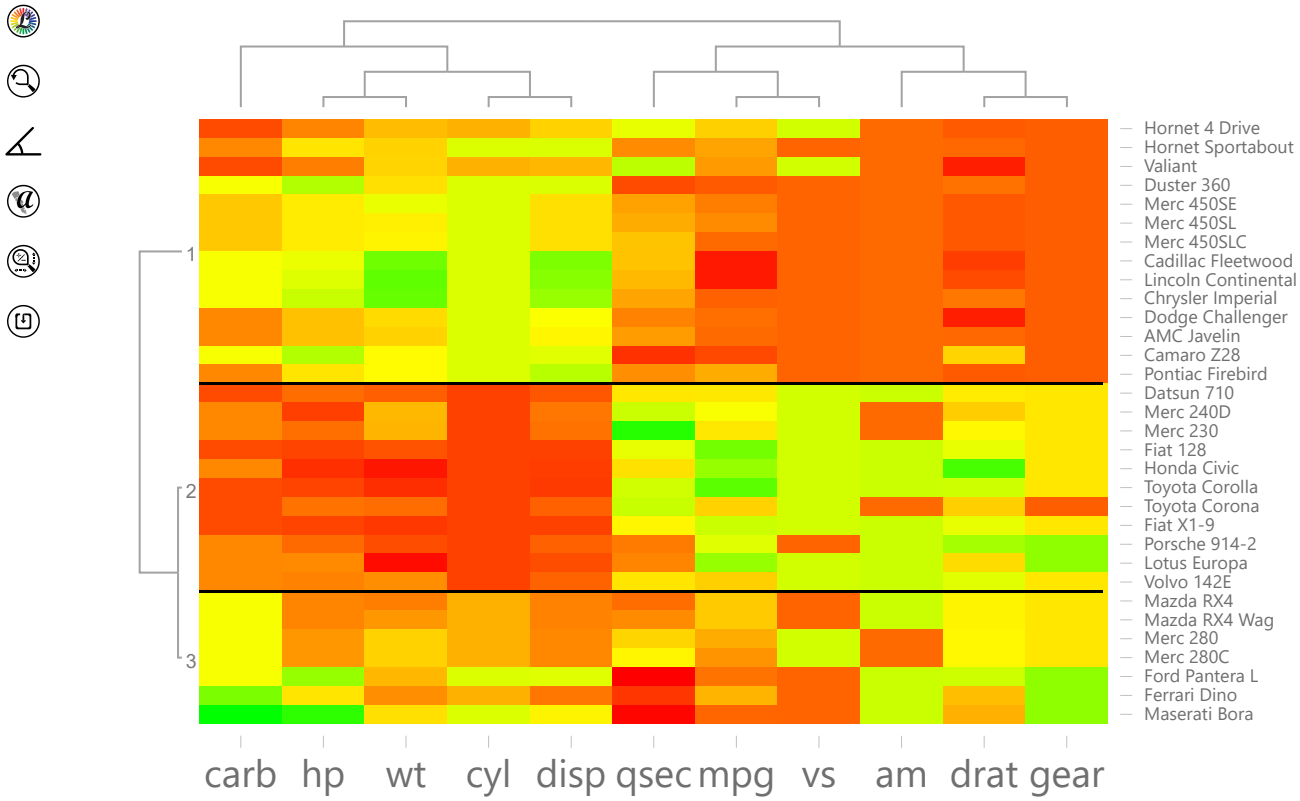
```
best_k <- db_object$best_k

# the best k is used to run the clustpro function with fixed k
clustpro(matrix = df_data, method = 'kmeans', fixed_k = best_k, seed = seed, elementId = get_
unique_id())
```



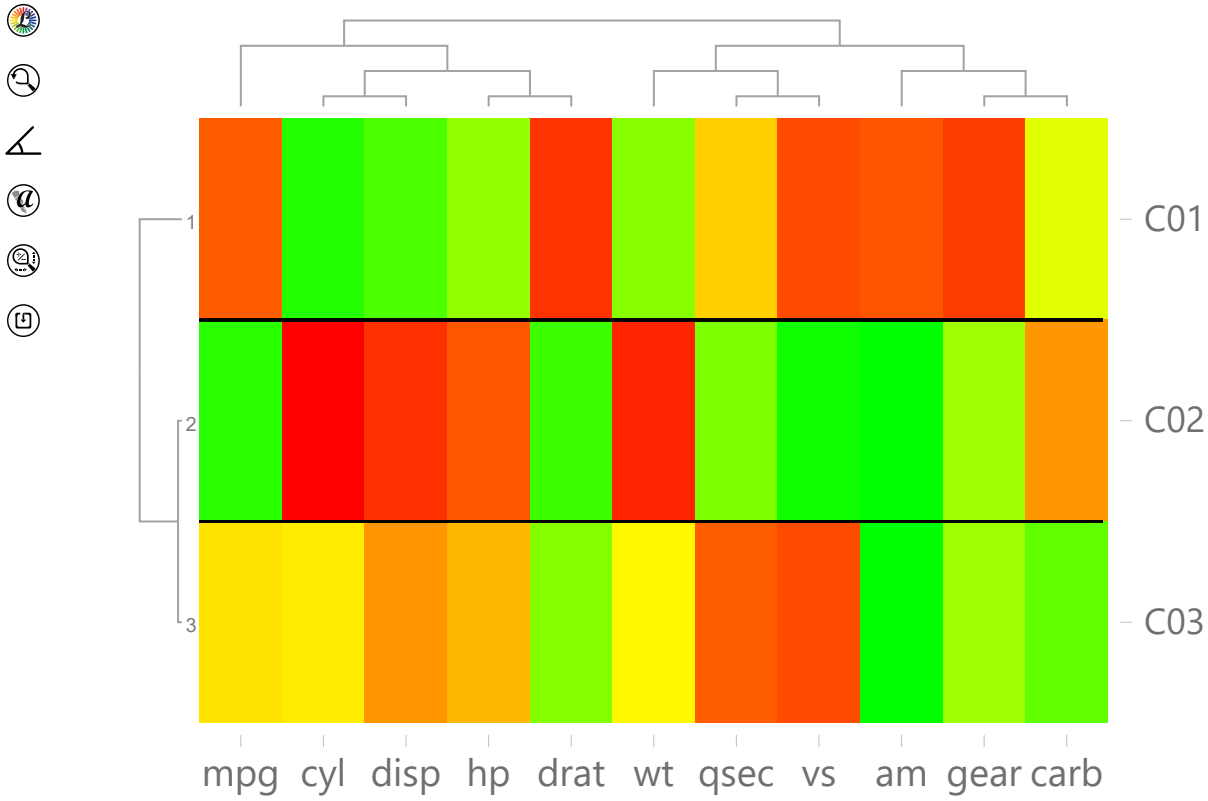
```
# it is also possible to use the fussy cmeans algorithm
db_object <- get_best_k(matrix = df_data, min_k = 2, max_k = 10, method = 'cmeans', seed = se
ed, cores = 4)

db_object$db_list
```

```
##    k     score withinerror
## 1  2 1.0625866   4.0707788
## 2  3 1.1855022   2.5647346
## 3  4 0.9386960   1.8042279
## 4  5 0.8624017   1.4018134
## 5  6 0.9049757   1.1330436
## 6  7 0.9157666   0.9441594
## 7  8 0.8676762   0.8419797
## 8  9 0.7791347   0.6044759
## 9 10 0.8977055   0.5098897
```

```
best_k <- db_object$best_k

clustpro(matrix = df_data, method = 'cmeans', fixed_k = best_k, seed = seed, elementId = get_
unique_id())
```




```
# large dataset can be simplified for visualisation. This means that own the mean value of a
 cluster is show in a single row. The tooltip included the number of grouped proteins. In the
json file all protein of a group a stored.
clustpro(matrix = df_data, method = 'cmeans', fixed_k = best_k, simplify_clustering = TRUE, s
eed = seed, elementId = get_unique_id())
```



Using clustPro without Visualization

```
# run clustPro without visualization just in the console
clustpro(matrix = df_data, method = 'kmeans', seed = seed, useShiny = FALSE, elementId = get_
unique_id())
```

```
## $datatable
##                           hp       carb        disp        cyl
## Hornet Sportabout    0.41294217 -0.5030337  1.04308123  1.0148821
## Duster 360           1.43390296  0.7352031  1.04308123  1.0148821
## Merc 450SE           0.48586794  0.1160847  0.36371309  1.0148821
## Merc 450SL           0.48586794  0.1160847  0.36371309  1.0148821
## Merc 450SLC          0.48586794  0.1160847  0.36371309  1.0148821
## Cadillac Fleetwood   0.85049680  0.7352031  1.94675381  1.0148821
## Lincoln Continental  0.99634834  0.7352031  1.84993175  1.0148821
## Chrysler Imperial    1.21512565  0.7352031  1.68856165  1.0148821
## Dodge Challenger     0.04831332 -0.5030337  0.70420401  1.0148821
## AMC Javelin          0.04831332 -0.5030337  0.59124494  1.0148821
## Camaro Z28           1.43390296  0.7352031  0.96239618  1.0148821
## Pontiac Firebird     0.41294217 -0.5030337  1.36582144  1.0148821
## Mazda RX4           -0.53509284  0.7352031 -0.57061982 -0.1049878
## Mazda RX4 Wag       -0.53509284  0.7352031 -0.57061982 -0.1049878
## Ford Pantera L       1.71102089  0.7352031  0.97046468  1.0148821
## Ferrari Dino         0.41294217  1.9734398 -0.69164740 -0.1049878
## Maserati Bora        2.74656682  3.2116766  0.56703942  1.0148821
## Hornet 4 Drive      -0.53509284 -1.1221521  0.22009369 -0.1049878
## Valiant             -0.60801861 -1.1221521 -0.04616698 -0.1049878
## Merc 240D           -1.23518023 -0.5030337 -0.67793094 -1.2248578
## Merc 230            -0.75387015 -0.5030337 -0.72553512 -1.2248578
## Merc 280            -0.34548584  0.7352031 -0.50929918 -0.1049878
## Merc 280C           -0.34548584  0.7352031 -0.50929918 -0.1049878
## Toyota Corona       -0.72469984 -1.1221521 -0.89255318 -1.2248578
## Datsun 710          -0.78304046 -1.1221521 -0.99018209 -1.2248578
## Fiat 128            -1.17683962 -1.1221521 -1.22658929 -1.2248578
## Honda Civic         -1.38103178 -0.5030337 -1.25079481 -1.2248578
## Toyota Corolla      -1.19142477 -1.1221521 -1.28790993 -1.2248578
## Fiat X1-9           -1.17683962 -1.1221521 -1.22416874 -1.2248578
## Volvo 142E          -0.54967799 -0.5030337 -0.88529152 -1.2248578
## Porsche 914-2       -0.81221077 -0.5030337 -0.89093948 -1.2248578
## Lotus Europa        -0.49133738 -0.5030337 -1.09426581 -1.2248578
##                            wt        qsec         vs       gear
## Hornet Sportabout    0.227654255 -0.46378082 -0.8680278 -0.9318192
## Duster 360           0.360516446 -1.12412636 -0.8680278 -0.9318192
## Merc 450SE           0.871524874 -0.25112717 -0.8680278 -0.9318192
## Merc 450SL           0.524039143 -0.13920420 -0.8680278 -0.9318192
## Merc 450SLC          0.575139986  0.08464175 -0.8680278 -0.9318192
## Cadillac Fleetwood   2.077504765  0.07344945 -0.8680278 -0.9318192
## Lincoln Continental  2.255335698 -0.01608893 -0.8680278 -0.9318192
## Chrysler Imperial    2.174596366 -0.23993487 -0.8680278 -0.9318192
## Dodge Challenger     0.309415603 -0.54772305 -0.8680278 -0.9318192
## AMC Javelin          0.222544170 -0.30708866 -0.8680278 -0.9318192
## Camaro Z28           0.636460997 -1.36476075 -0.8680278 -0.9318192
## Pontiac Firebird     0.641571082 -0.44699237 -0.8680278 -0.9318192
## Mazda RX4           -0.610399567 -0.77716515 -0.8680278  0.4235542
## Mazda RX4 Wag       -0.349785269 -0.46378082 -0.8680278  0.4235542
## Ford Pantera L      -0.048290296 -1.87401028 -0.8680278  1.7789276
## Ferrari Dino        -0.457097039 -1.31439542 -0.8680278  1.7789276
## Maserati Bora        0.360516446 -1.81804880 -0.8680278  1.7789276
## Hornet 4 Drive      -0.002299538  0.89048716  1.1160357 -0.9318192
## Valiant              0.248094592  1.32698675  1.1160357 -0.9318192
## Merc 240D           -0.027849959  1.20387148  1.1160357  0.4235542
## Merc 230            -0.068730634  2.82675459  1.1160357  0.4235542
## Merc 280             0.227654255  0.25252621  1.1160357  0.4235542
```

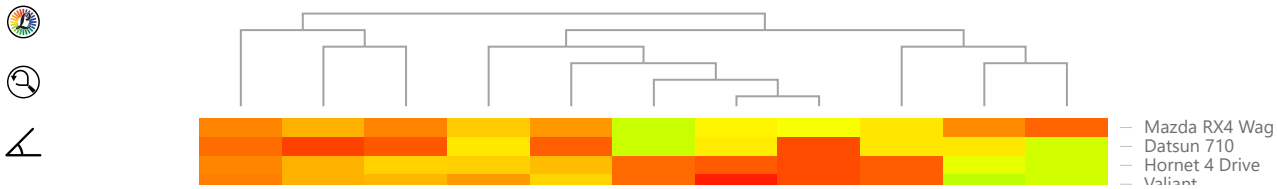
```
## Merc 280C            0.227654255  0.58829513  1.1160357  0.4235542
## Toyota Corona       -0.768812180  1.20946763  1.1160357 -0.9318192
## Datsun 710          -0.917004624  0.42600682  1.1160357  0.4235542
## Fiat 128            -1.039646647  0.90727560  1.1160357  0.4235542
## Honda Civic         -1.637526508  0.37564148  1.1160357  0.4235542
## Toyota Corolla      -1.412682800  1.14790999  1.1160357  0.4235542
## Fiat X1-9           -1.310481114  0.58829513  1.1160357  0.4235542
## Volvo 142E          -0.446876870  0.42041067  1.1160357  0.4235542
## Porsche 914-2       -1.100967659 -0.64285758 -0.8680278  1.7789276
## Lotus Europa        -1.741772228 -0.53093460  1.1160357  1.7789276
##                             am         mpg        drat cluster
## Hornet Sportabout   -0.8141431 -0.23073453 -0.83519779       5
## Duster 360          -0.8141431 -0.96078893 -0.72298087       5
## Merc 450SE          -0.8141431 -0.61235388 -0.98482035       5
## Merc 450SL          -0.8141431 -0.46302456 -0.98482035       5
## Merc 450SLC         -0.8141431 -0.81145962 -0.98482035       5
## Cadillac Fleetwood  -0.8141431 -1.60788262 -1.24665983       5
## Lincoln Continental -0.8141431 -1.60788262 -1.11574009       5
## Chrysler Imperial   -0.8141431 -0.89442035 -0.68557523       5
## Dodge Challenger    -0.8141431 -0.76168319 -1.56460776       5
## AMC Javelin         -0.8141431 -0.81145962 -0.83519779       5
## Camaro Z28          -0.8141431 -1.12671039  0.24956575       5
## Pontiac Firebird    -0.8141431 -0.14777380 -0.96611753       5
## Mazda RX4            1.1899014  0.15088482  0.56751369       4
## Mazda RX4 Wag        1.1899014  0.15088482  0.56751369       4
## Ford Pantera L       1.1899014 -0.71190675  1.16600392       4
## Ferrari Dino         1.1899014 -0.06481307  0.04383473       4
## Maserati Bora        1.1899014 -0.84464392 -0.10578782       4
## Hornet 4 Drive      -0.8141431  0.21725341 -0.96611753       1
## Valiant             -0.8141431 -0.33028740 -1.56460776       1
## Merc 240D           -0.8141431  0.71501778  0.17475447       1
## Merc 230            -0.8141431  0.44954345  0.60491932       1
## Merc 280            -0.8141431 -0.14777380  0.60491932       1
## Merc 280C           -0.8141431 -0.38006384  0.60491932       1
## Toyota Corona       -0.8141431  0.23384555  0.19345729       1
## Datsun 710           1.1899014  0.44954345  0.47399959       2
## Fiat 128             1.1899014  2.04238943  0.90416444       2
## Honda Civic          1.1899014  1.71054652  2.49390411       2
## Toyota Corolla       1.1899014  2.29127162  1.16600392       2
## Fiat X1-9            1.1899014  1.19619000  0.90416444       2
## Volvo 142E           1.1899014  0.21725341  0.96027290       2
## Porsche 914-2        1.1899014  0.98049211  1.55876313       3
## Lotus Europa         1.1899014  1.71054652  0.32437703       3
##
## $cobject
## K-means clustering with 5 clusters of sizes 7, 6, 2, 5, 12
##
## Cluster means:
##          mpg        cyl        disp         hp        drat          wt
## 1  0.1082193 -0.5849321 -0.44867013 -0.6496905 -0.04967936 -0.02346989
## 2  1.3178657 -1.2248578 -1.14415607 -1.0431424  1.15041823 -1.12736976
## 3  1.3455193 -1.2248578 -0.99260264 -0.6517741  0.94157008 -1.42136994
## 4 -0.2639188  0.3429602 -0.05907659  0.7600688  0.44781564 -0.22101115
## 5 -0.8363478  1.0148821  1.02385129  0.6924910 -0.88974768  0.90635862
##         qsec         vs         am       gear       carb
## 1  1.1854841  1.1160357 -0.8141431 -0.1573201 -0.4145882
## 2  0.6442566  1.1160357  1.1899014  0.4235542 -0.9157793
## 3 -0.5868961  0.1240040  1.1899014  1.7789276 -0.5030337
## 4  1.2404801 -0.8680278 -1.1899014 -1.2367783  1.4781451
```

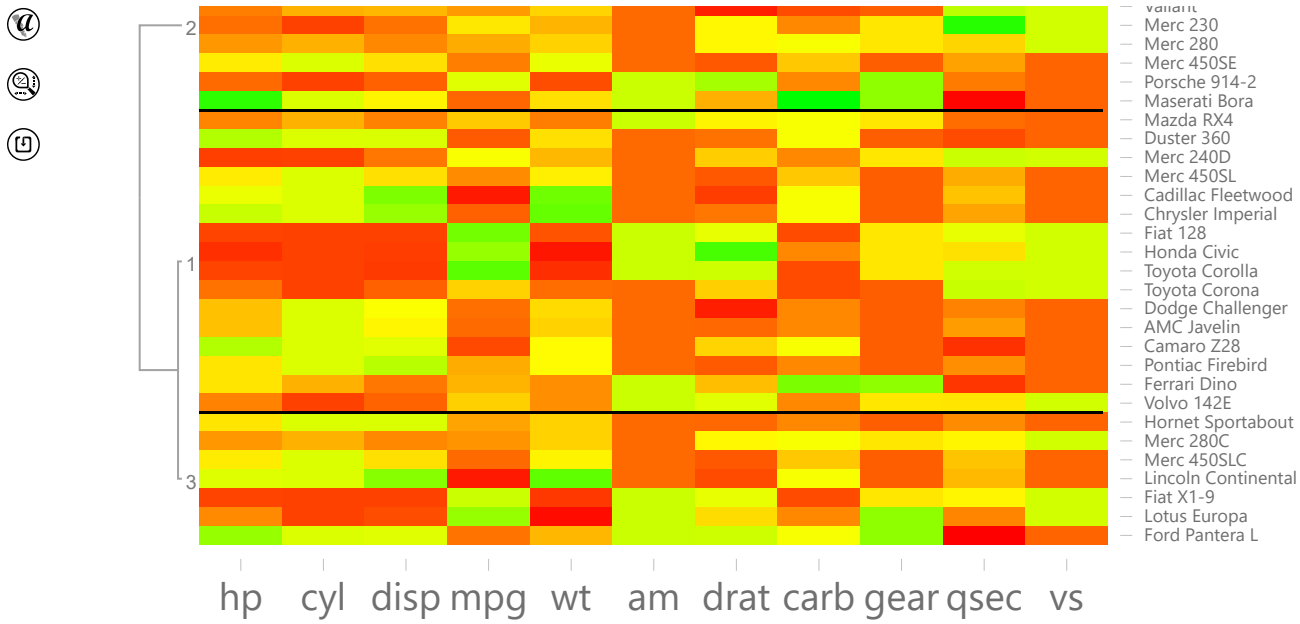
```
## 4 -1.2494801 -0.8680278  1.1899014  1.2367782  1.4781451
## 5 -0.3952280 -0.8680278 -0.8141431 -0.9318192  0.1676779
##
## Clustering vector:
##           Mazda RX4       Mazda RX4 Wag          Datsun 710
##                   4                   4                   2
##      Hornet 4 Drive   Hornet Sportabout             Valiant
##                   1                   5                   1
##          Duster 360           Merc 240D            Merc 230
##                   5                   1                   1
##            Merc 280           Merc 280C           Merc 450SE
##                   1                   1                   5
##          Merc 450SL         Merc 450SLC  Cadillac Fleetwood
##                   5                   5                   5
## Lincoln Continental   Chrysler Imperial            Fiat 128
##                   5                   5                   2
##         Honda Civic      Toyota Corolla       Toyota Corona
##                   2                   2                   1
##    Dodge Challenger         AMC Javelin          Camaro Z28
##                   5                   5                   5
##    Pontiac Firebird           Fiat X1-9       Porsche 914-2
##                   5                   2                   3
##        Lotus Europa      Ford Pantera L        Ferrari Dino
##                   3                   4                   4
##       Maserati Bora          Volvo 142E
##                   4                   2
##
## Within cluster sum of squares by cluster:
## [1] 21.287980  8.543145  3.280327 23.402761 23.083489
##  (between_SS / total_SS =  76.7 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
##
## $cluster_centers
##           mpg        cyl       disp         hp       drat           wt
## 5 -0.81145962  1.0148821  1.0027387  0.4858679 -0.9754689  0.605800492
## 4 -0.06481307 -0.1049878 -0.5706198  0.4129422  0.5675137 -0.349785269
## 1  0.21725341 -0.1049878 -0.5092992 -0.6080186  0.1934573 -0.002299538
## 2  1.45336826 -1.2248578 -1.2253790 -1.1768396  0.9322187 -1.175063881
## 3  1.34551931 -1.2248578 -0.9926026 -0.6517741  0.9415701 -1.421369943
##         qsec         vs         am       gear       carb
## 5 -0.2791079 -0.8680278 -0.8141431 -0.9318192  0.1160847
## 4 -1.3143954 -0.8680278  1.1899014  1.7789276  0.7352031
## 1  1.2038715  1.1160357 -0.8141431  0.4235542 -0.5030337
## 2  0.5071510  1.1160357  1.1899014  0.4235542 -1.1221521
## 3 -0.5868961  0.1240040  1.1899014  1.7789276 -0.5030337
##
## $col_dend_hclust
##
## Call:
## hclust(d = d_cols, method = hclust_method)
##
## Cluster method   : ward.D2
## Distance         : euclidean
## Number of objects: 11
```

```
## Number of objects: 11
##
##
## $row_dend_hclust
##
## Call:
## hclust(d = d_rows, method = hclust_method)
##
## Cluster method   : ward.D2
## Distance         : euclidean
## Number of objects: 5
```

Using pre-clustered Data

```
#randomly group matrix in 3 clusters
set.seed(seed)
clusterVector <- sample(1:3, nrow(df_data), replace = TRUE)
df_data_extended <- df_data
df_data_extended$clusterVector <- clusterVector

#compute mean matrix
mean_df_data <- aggregate(df_data_extended[,colnames(df_data)], list(df_data_extended$cluster
Vector), mean)
mean_df_data$Group.1 <- NULL

#determine dendrograms matrix
d_rows <- dist(mean_df_data, method = "euclidean") # distance matrix
row_dend_hclust <- hclust(d_rows, method = "ward.D2")
row_dend_nw <- ctc::hc2Newick(row_dend_hclust)
row_dend_nw <- gsub(":\\d+\\.{0,1}\\d*", "", row_dend_nw)
row_dend <- as.dendrogram(row_dend_hclust)

d_cols <- dist(t(mean_df_data), method = "euclidean") # distance matrix
col_dend_hclust <- hclust(d_cols, method = "ward.D2")
col_dend_nw <- ctc::hc2Newick(col_dend_hclust)
col_dend_nw <- gsub(":\\d+\\.{0,1}\\d*", "", col_dend_nw)
col_dend <- as.dendrogram(col_dend_hclust)

#order matrix in accordance with dendrograms
df_data_extended <- clustpro::order_dataframe_by_list(df_data_extended,order.dendrogram(row_d
end),'clusterVector')
clusterVector <- df_data_extended$clusterVector
df_data_extended$clusterVector <- NULL
df_data_extended <- df_data_extended[,order.dendrogram(col_dend)]

#run clustpro without clustering
clustpro(matrix = df_data_extended
         ,clusterVector = clusterVector
         ,perform_clustering = FALSE
         ,rows = row_dend_hclust
         ,cols = col_dend_hclust, elementId = get_unique_id())
```





Session info

```
sessionInfo()
```

```
## R version 3.5.1 (2018-07-02)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 10240)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=German_Germany.1252  LC_CTYPE=German_Germany.1252
## [3] LC_MONETARY=German_Germany.1252 LC_NUMERIC=C
## [5] LC_TIME=German_Germany.1252
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] clustpro_0.1.0.0 amap_0.8-16
##
## loaded via a namespace (and not attached):
##  [1] bitops_1.0-6            R2HTML_2.3.2
##  [3] bit64_0.9-7             doParallel_1.0.14
##  [5] webshot_0.5.1           fBasics_3042.89
##  [7] tools_3.5.1             R6_2.3.0
##  [9] rpart_4.1-13            DBI_1.0.0
## [11] BiocGenerics_0.28.0     lazyeval_0.2.1
## [13] colorspace_1.4-0        ade4_1.7-13
## [15] manipulateWidget_0.10.0 tidyselect_0.2.5
## [17] timeSeries_3042.102     bit_1.1-14
## [19] compiler_3.5.1          Biobase_2.42.0
## [21] scales_1.0.0            genefilter_1.64.0
## [23] spatial_7.3-11          stringr_1.3.1
## [25] digest_0.6.18           rmarkdown_1.11
## [27] pkgconfig_2.0.2         htmltools_0.3.6
## [29] stabledist_0.7-1        htmlwidgets_1.3
## [31] rlang_0.3.1             RSQLite_2.1.1
## [33] shiny_1.2.0             bindr_0.1.1
## [35] jsonlite_1.6            ctc_1.56.0
## [37] crosstalk_1.0.0         dplyr_0.7.8
## [39] RCurl_1.95-4.11         magrittr_1.5
## [41] kimisc_0.4              Matrix_1.2-15
## [43] Rcpp_1.0.0              munsell_0.5.0
## [45] S4Vectors_0.20.1        stringi_1.2.4
## [47] yaml_2.2.0              MASS_7.3-51.1
## [49] stable_1.1.3            plyr_1.8.4
## [51] grid_3.5.1              blob_1.1.1
## [53] parallel_3.5.1          promises_1.0.1
## [55] crayon_1.3.4            miniUI_0.1.1.1
## [57] lattice_0.20-38         splines_3.5.1
## [59] annotate_1.60.0         knitr_1.21
## [61] pillar_1.3.1            statip_0.2.0
## [63] codetools_0.2-16        rmutil_1.1.1
## [65] stats4_3.5.1            glue_1.3.0
## [67] XML_3.98-1.19           evaluate_0.12
## [69] clusterSim_0.47-3       bazar_1.0.10
## [71] httpuv_1.4.5.1          foreach_1.4.4
## [73] purrr_0.3.0             gtable_0.2.0
## [75] clue_0.3-56             assertthat_0.2.0
## [77] ggplot2_3.1.0           xfun_0.4
```

```
## [79] mime_0.6                xtable_1.8-3
## [81] e1071_1.7-0.1           later_0.7.5
## [83] class_7.3-14            survival_2.43-3
## [85] modeest_2.3.2           timeDate_3043.102
## [87] snow_0.4-3              tibble_2.0.1
## [89] iterators_1.0.10        AnnotationDbi_1.44.0
## [91] memoise_1.1.0           IRanges_2.16.0
## [93] bindrcpp_0.2.2          cluster_2.0.7-1
## [95] rgl_0.99.16
```