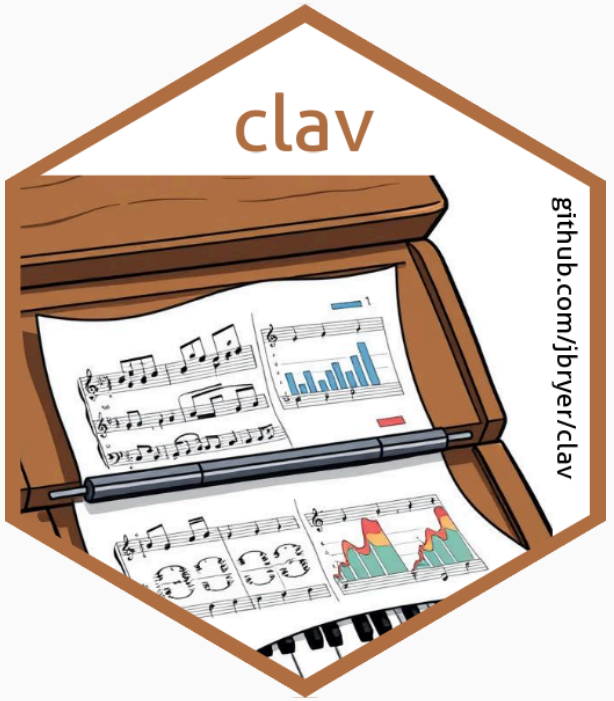
clav: Cluster Analysis Validation

Joint Statistical Meeting

Jason Bryer, Ph.D.

August 5, 2025



Overview

1. Motivation for this package.
2. Discussion of validation in the context of clustering analysis.
3. How to use the `clav` package for clustering analysis.
   a. Determine the optimal number of clusters.
   b. Validation the cluster solution.
   c. Exploring the relationship of clusters to other variables.
4. Shiny application.

This research was supported under grants P116F150077 and R305A210269 from the U.S. Department of Education. However, the contents do not necessarily represent the policy of the U.S. Department of Education, and you should not assume endorsement by the Federal Government.

Motivating Example

The Diagnostic Assessment and Achievement of College Skills (DAACS; www.daacs.net) is a suite of technological and social supports designed to optimize student learning.

Students complete assessments in self-regulated learning, writing, mathematics, and reading comprehension. They are then provided with immediate feedback in terms of one, two, and three dots (developing, emerging, and mastering, respectively) and receive customized strategies and resources based upon their results.

Prior research has shown that DAACS can improve the accuracy of predicting student success by 2% to 10% over non-DAACS models. However, those models have been **variable centric**.

In order to provide better information to help institutional staff/instructors we wish to define **profiles** using a **person centric** approach.

Data Source

Data for this study was collected as part of a large scale randomized control trial.

Online institution of predominately adult learners.

Competency based program where students complete a series of competencies that when combined form course credits.

Competencies are graded on pass/fail so success is measured by students completing the equivelent of 12 credits with 6 months.

Validation

Model validation is the process of estimating how well a model performs. This is often done by separating the data into two where one dataset is used to train the model and predictions are made with the second dataset.

Supervised Methods

Supervised models are models where the outcome, or *truth* is known. Common supervised methods include regression, classification, and object detection.

Unsupervised Methods

Unsupervised models are models where the outcome is not observed or known. Common unsupervised methods include clustering (e.g. k-means, latent profile analysis) and dimension reduction (e.g. exploratory factor analysis, principal component analysis).

Clustering

Clustering is a statistical procedure that groups observations that are similar across multiple variables. Whereas principal component analysis (PCA) and exploratory factor analysis (EFA) are variable centric (i.e. columns), clustering methods are observation centric (i.e. rows).

The `clav` package is designed to work with clustering algorithms. We will use k-means clustering here (using `stats::kmeans()`), but other methods do work.

The steps for clustering include:

1. Determine the number of clusters.
2. Validate the cluster solution.
3. Use the cluster assignments in other models.

Getting started

You can download the development version from Github:

```
remotes::install_github('jbryer/clav')
```

Load the package and data frame:

```
library(clav)
data("daacs", package = "clav")
cluster_vars <- c('Motivation', 'Metacognition', 'Strategies', 'Mathematics', 'Reading', 'Writing')
outcome_vars <- c('FeedbackViews', 'TermSuccess')
```

We will standardize our clustering variables:

```
daacs <- daacs |>
    dplyr::mutate(dplyr::across(dplyr::all_of(cluster_vars), clav::scale_this))
```

DAACS Variables

Clustering Variables

Self-Regulated Learning measures (Likert response data ranging from 0 to 4)

- `Motivation`
- `Metacognition`
- `Strategies`

Academic measures (students complete 18 to 24 items, scores range from 0 to 1)

- `Mathematics`
- `Reading`
- `Writing`

Outcome Variables

- `FeedbackViews` - number of feedback pages students access within the DAACS system.
- `TermSuccess` - whether the student successfully completed 12 credits within their first term.

Variable Centric Approach

```
#>
#> Call:
#> lm(formula = FeedbackViews ~ Motivation + Metacognition + Strate
#>     Mathematics + Reading + Writing, data = daacs)
#>
#> Residuals:
#>     Min      1Q  Median      3Q     Max
#> -21.890  -9.120  -3.327   5.361 232.315
#>
#> Coefficients:
#>               Estimate Std. Error t value Pr(>|t|)
#> (Intercept)    16.8463     0.1742  96.729  < 2e-16 ***
#> Motivation     -1.1350     0.2113  -5.370 8.13e-08 ***
#> Metacognition  -1.3912     0.2295  -6.061 1.43e-09 ***
#> Strategies      1.6176     0.2430   6.657 3.03e-11 ***
#> Mathematics     1.6899     0.1908   8.858  < 2e-16 ***
#> Reading         0.8047     0.1943   4.141 3.50e-05 ***
#> Writing         2.7004     0.1848  14.614  < 2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 13.91 on 6369 degrees of freedom
#> Multiple R-squared:  0.07874,    Adjusted R-squared:  0.07788
#> F-statistic: 90.73 on 6 and 6369 DF,  p-value: < 2.2e-16
```

```
#>
#> Call:
#> glm(formula = TermSuccess ~ Motivation + Metacognition + Strateg
#>     Mathematics + Reading + Writing, family = binomial(link = "l
#>     data = daacs)
#>
#> Coefficients:
#>               Estimate Std. Error z value Pr(>|z|)
#> (Intercept)    0.795330   0.027933  28.473  < 2e-16 ***
#> Motivation    -0.002025   0.033495  -0.060  0.95179
#> Metacognition -0.094069   0.036811  -2.555  0.01060 *
#> Strategies     0.104083   0.038689   2.690  0.00714 **
#> Mathematics    0.236394   0.030138   7.844 4.38e-15 ***
#> Reading        0.295536   0.030575   9.666  < 2e-16 ***
#> Writing        0.168553   0.028544   5.905 3.53e-09 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> (Dispersion parameter for binomial family taken to be 1)
#>
#>     Null deviance: 7981.2  on 6375  degrees of freedom
#> Residual deviance: 7599.6  on 6369  degrees of freedom
#> AIC: 7613.6
#>
#> Number of Fisher Scoring iterations: 4
```

Finding Optimal Clusters

Finding the optimal number of clusters is generally a balance between optimal fit statistics, parsimony, and interpretability.

- Davies-Bouldin Index (1979) - DBI is a metric used to evaluate the quality of a cluster analysis by measuring the compactness of clusters and their separation from each other. A lower DBI indicates better clustering, with well-separated and compact clusters.
- Calinski-Harabasz Statistic (Caliński & Harabasz, 1974) - CH statistic measures the ratio of between-cluster variance to within-cluster variance, indicating how well-separated and compact the clusters are. Higher CH values generally indicate better clustering performance.
- Within group sum of squares (Thorndike, 1953) - WSS quantifies the dispersion of data points within each cluster, with lower WSS values indicating more compact and well-defined clusters.
- Silhoutte score (Rousseeuw, 1986) - The silhouette value is a measure of how similar an object is to its own cluster (cohesion) compared to other clusters (separation). The silhouette value ranges from −1 to +1, where a high value indicates that the object is well matched to its own cluster and poorly matched to neighboring clusters.
- Gap statistic (Tibshirani, Walther, & Hastie, 2001) - The Gap statistic works by comparing the within-cluster variation of the actual data to that of a null reference distribution, typically a uniform distribution. The *gap* is the difference between these two, and the optimal number of clusters is chosen where the gap statistic is maximized.
- Rand index (2012) - The Rand index measures how often pairs of data points are assigned to the same or different clusters in both partitions. A higher Rand Index indicates greater similarity between the two clusterings.

Finding Optimal Clusters (cont.)

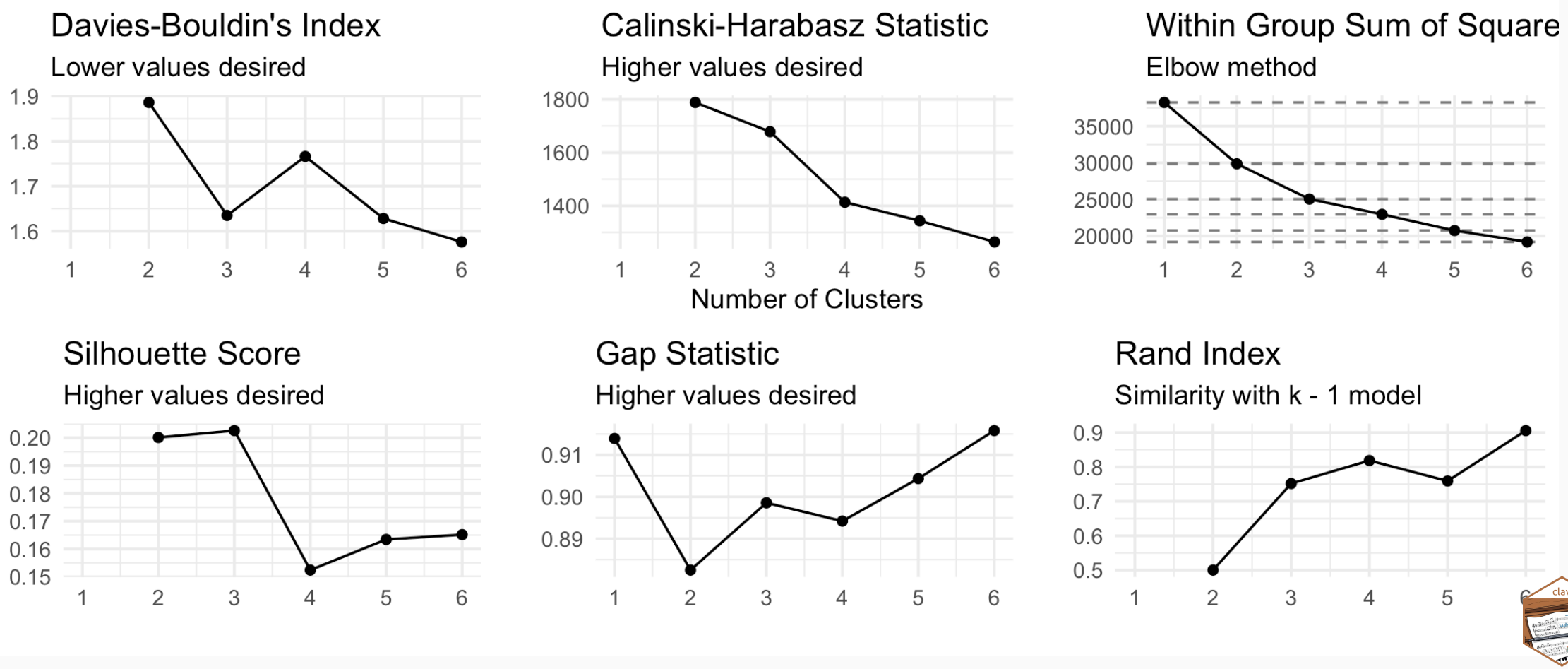
The `optimal_clusters` function will estimate the fit statistics for varying number of clusters. The default (`max_k`) is 9, but set to 6 here to reduce execution time.

The `cluster_fun` parameter defaults to `stats::kmeans`, but can other clustering functions can be used.

```
optimal <- optimal_clusters(daacs[,cluster_vars], max_k = 6)
optimal
#>   k      wss silhoutte       gap calinski_harabasz davies_bouldin rand_index
#> 1 1 38250.00        NA 0.9139183               NaN            NaN         NA
#> 2 2 29868.66 0.2001269 0.8825282          1788.684       1.886276  0.5002855
#> 3 3 25052.96 0.2025981 0.8985740          1678.549       1.634979  0.7516727
#> 4 4 22965.66 0.1524237 0.8942118          1413.590       1.766183  0.8185669
#> 5 5 20747.48 0.1634359 0.9043537          1343.662       1.628235  0.7590661
#> 6 6 19197.99 0.1651553 0.9158035          1264.308       1.576078  0.9055957
```

Finding Optimal Clusters (cont.)

```
plot(optimal)
```



Validating Clusters

For this example we are moving forward with a 5 cluster solution. The full details are available in Cleary, Bryer, and Yu, 2025.

Since there are no *known* clusters we cannot use methods typically used for supervised learning methods.

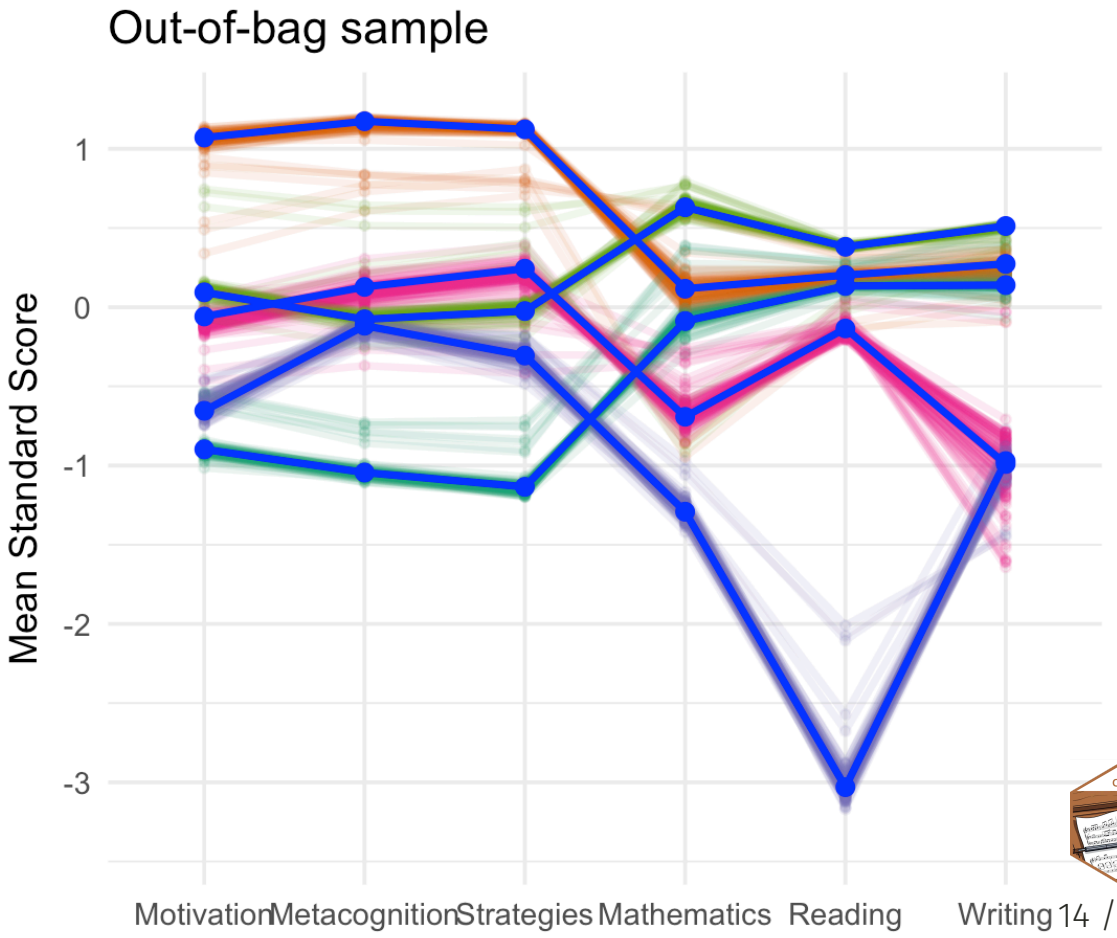
For cluster analysis, a valid cluster solution is one that is consistent.

Ullman et al (2021) proposed splitting the dataset and visually comparing the cluster solutions.

The `cluster_validation` implements this approach except that will split the dataset multiple times (default is 100). The clusters are estimated using the *training* data and cluster membership is predicted using that model with the *out-of-bag* (i.e. validation) sample.

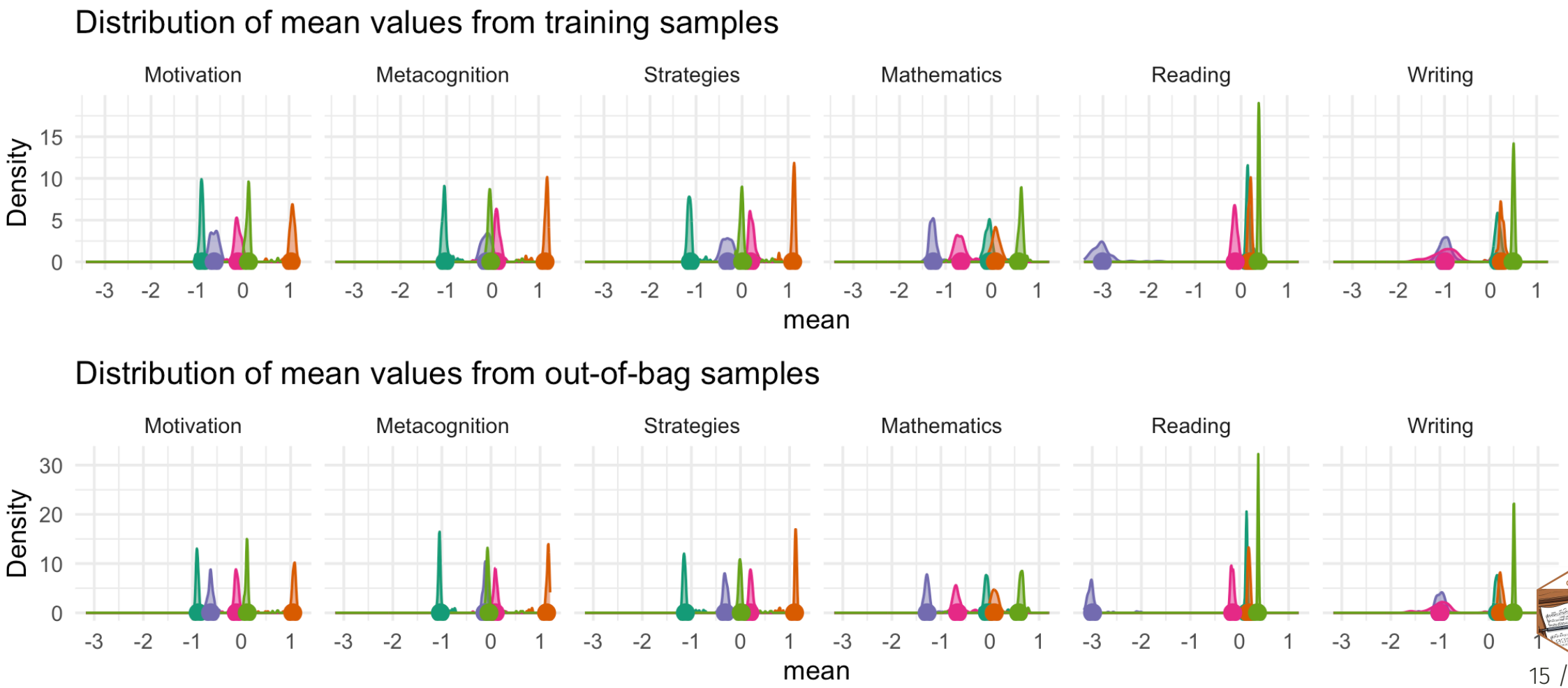
Validating Clusters

```
cv <- cluster_validation(df = daacs[,cluster_vars], n_clusters = 5)
plot(cv)
```



Distribution of Cluster Means

```
plot_distributions(cv, plot_in_sample = TRUE, plot_oob_sample = TRUE)
```



Bootstrapping

Alternatively we can using bootstrapping instead of mutually exclusive splits.

Bootstrapping is a procedure where we sample from our sample with replacement. Each bootstrap sample then has the same *n* as the original dataset.

The *out-of-bag* sample are observations that were not randomly selected for the training data.

Bootstrapping may be preferable when sample sizes are smaller.

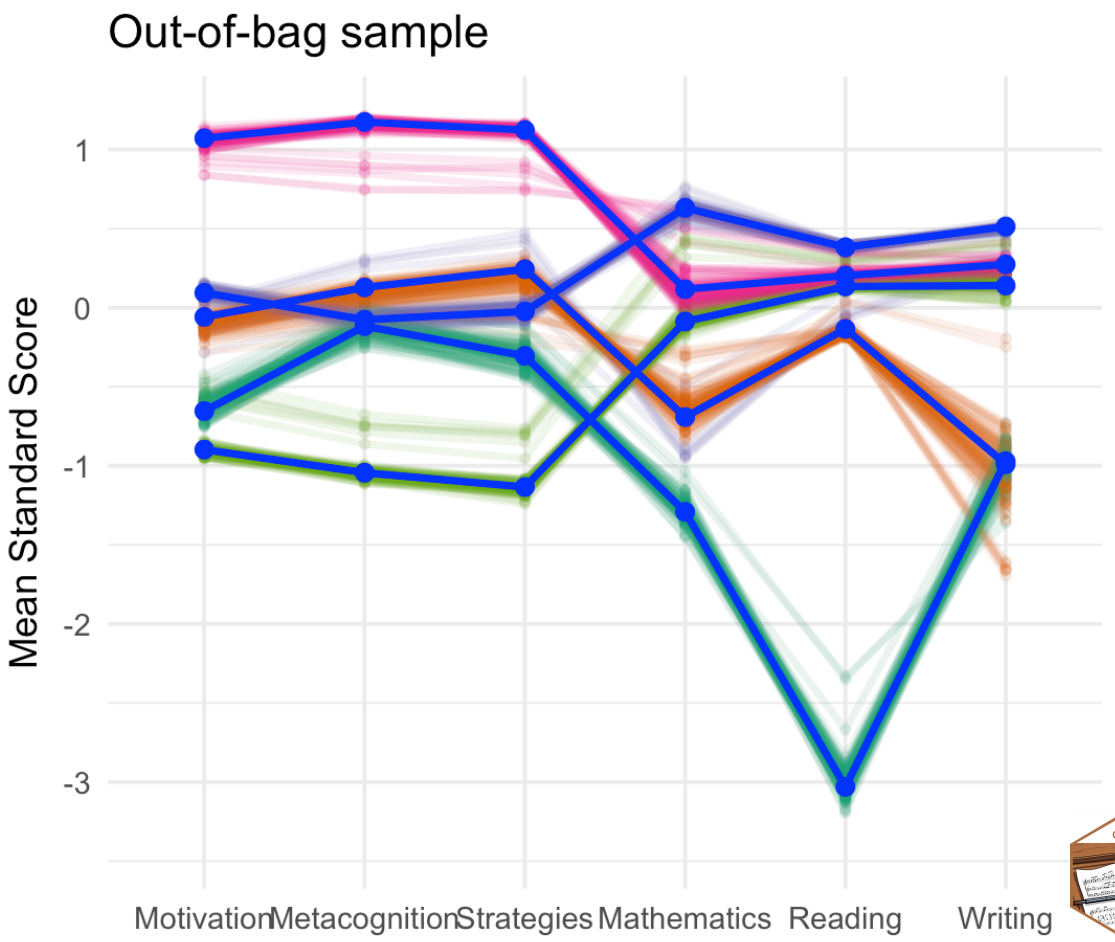
To use bootstrapping:

- Set `sample_size` to the number of observations.
- Set `replace = TRUE`

```
cv_boot <- cluster_validation(
    daacs[,cluster_vars],
    n_clusters = 5,
    sample_size = nrow(daacs),
    replace = TRUE)
```

Bootstrapping (cont.)

```
plot(cv_boot)
```



Retraining

In the previous examples we predicted cluster membership from the model trained with the training data.

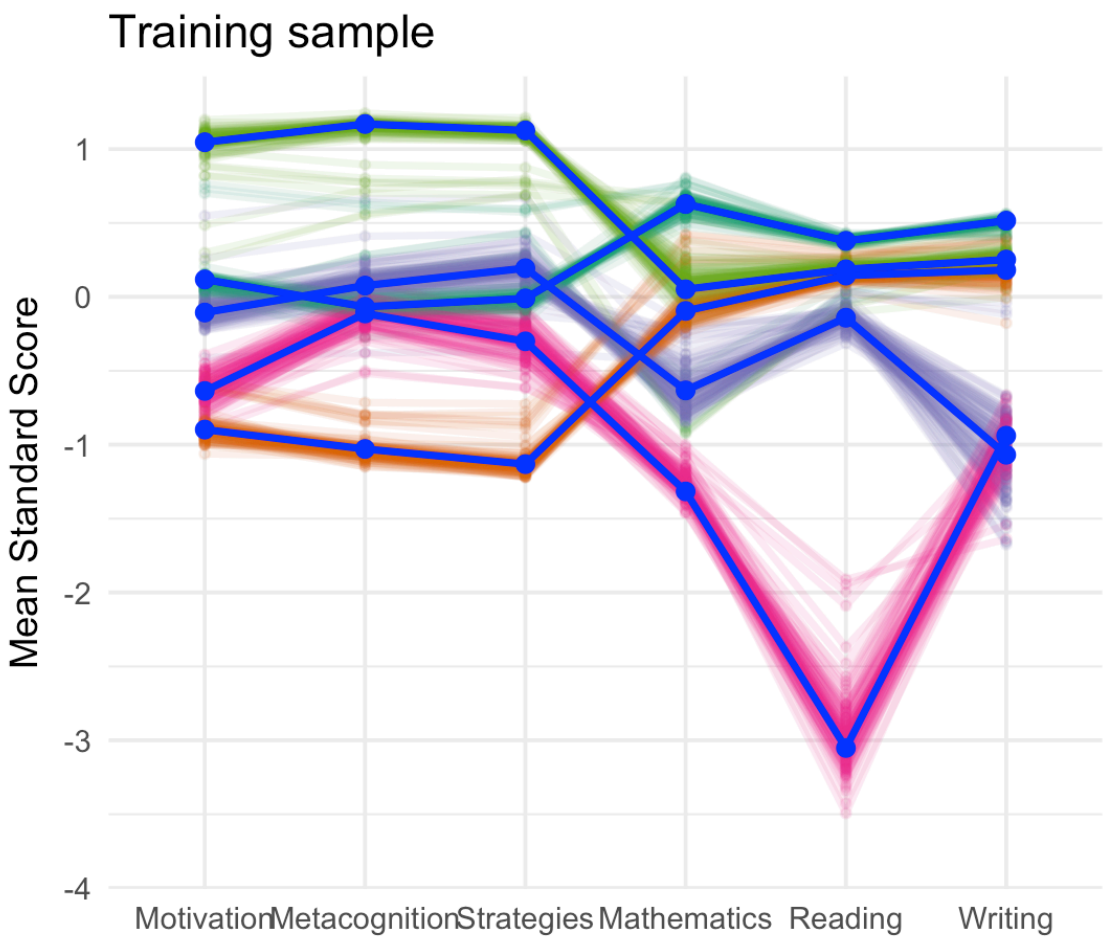
However, it is possible to compare the cluster solutions using two separate models: One trained with the training data and the other trained with out-of-bag (or validation) data.

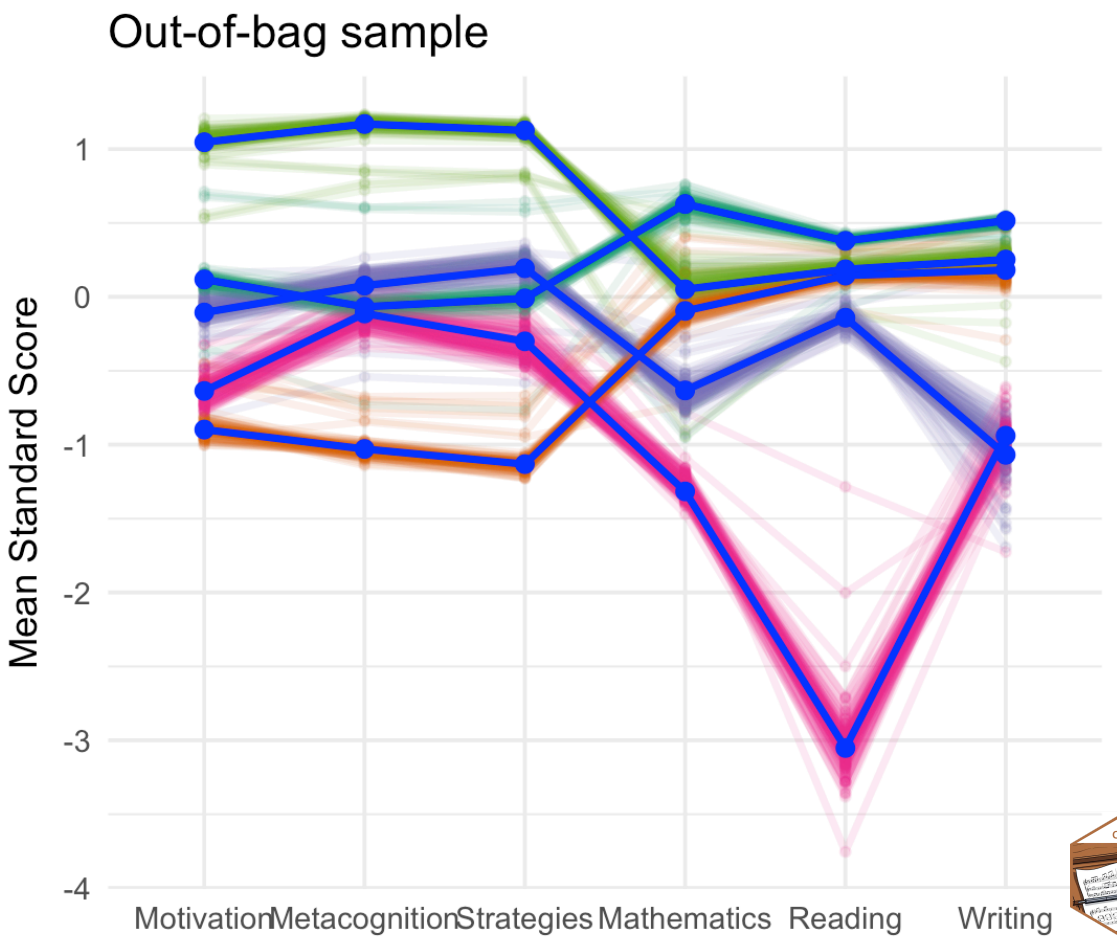
To use separate models for the two datasets, we need to define the `oob_predict_fun` parameter that implements cluster algorithm. In this example, we are wrapping the `stats::kmeans` function returning the cluster membership as a vector.

```
cv_retrain <- cluster_validation(
    daacs[,cluster_vars],
    n_clusters = 5,
    oob_predict_fun = function(fit, newdata) {
        stats::kmeans(newdata, 5)$cluster
    }
)
```

Retraining (cont.)

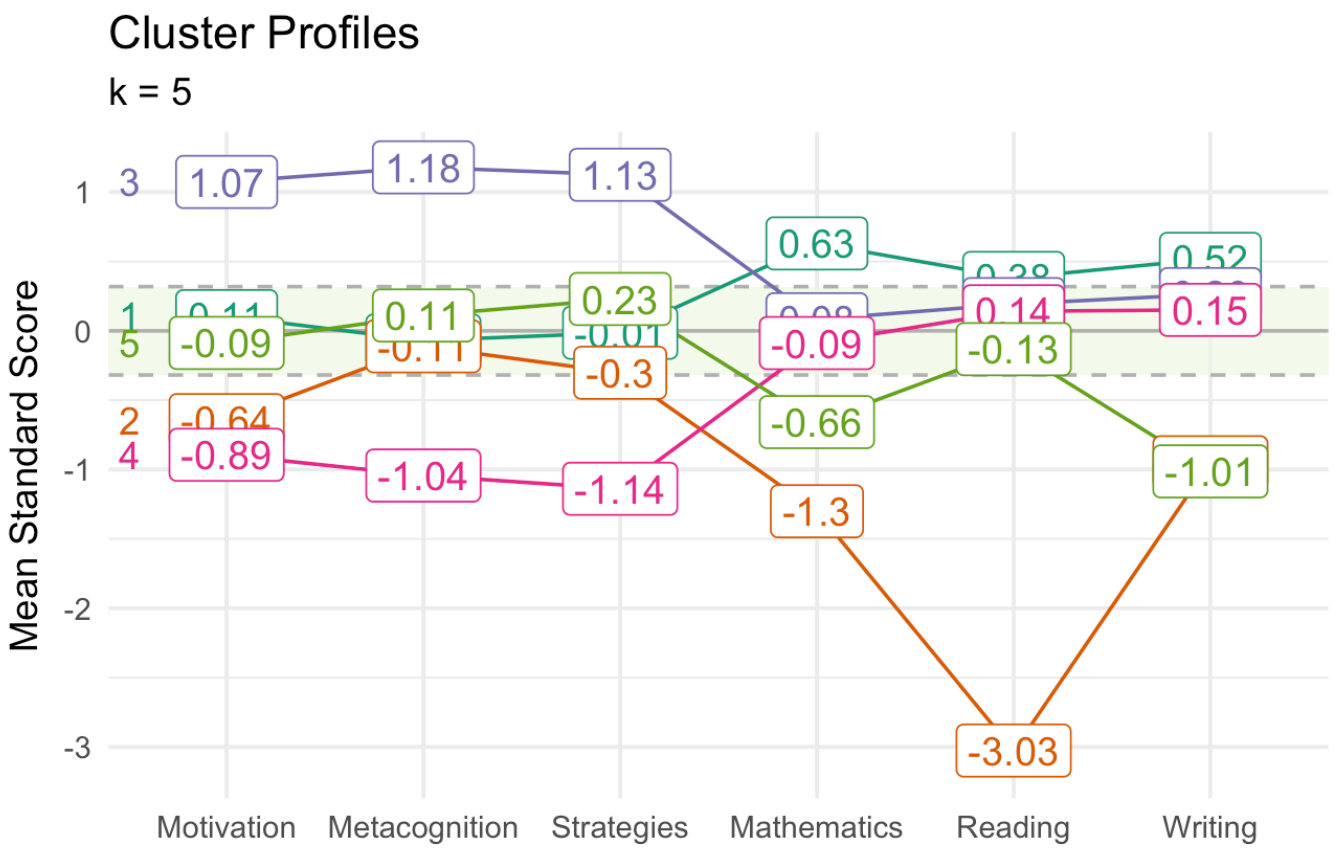
```
plot(cv_retrain)
```

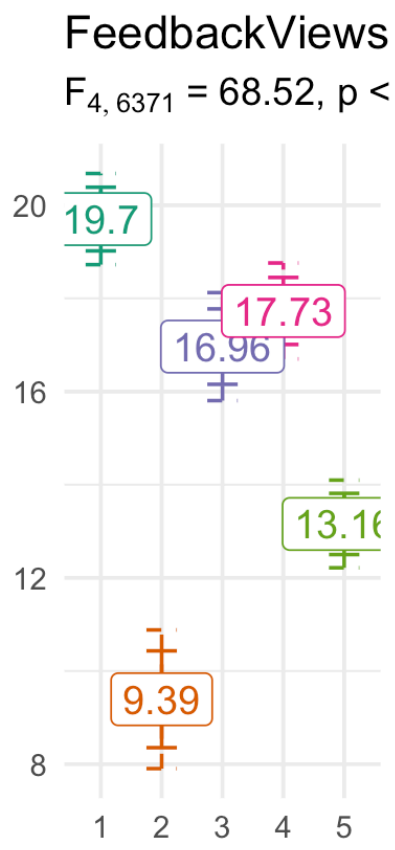


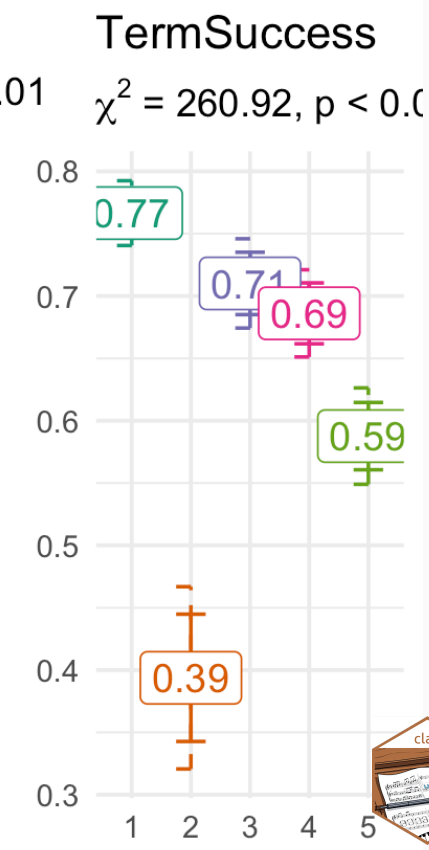


Profile Plots

```
fit <- stats::kmeans(daacs[,cluster_vars], centers = 5)
profile_plot(daacs[,cluster_vars], clusters = fit$cluster,
             df_dep = daacs[,outcome_vars], cluster_order = cluster_vars)
```

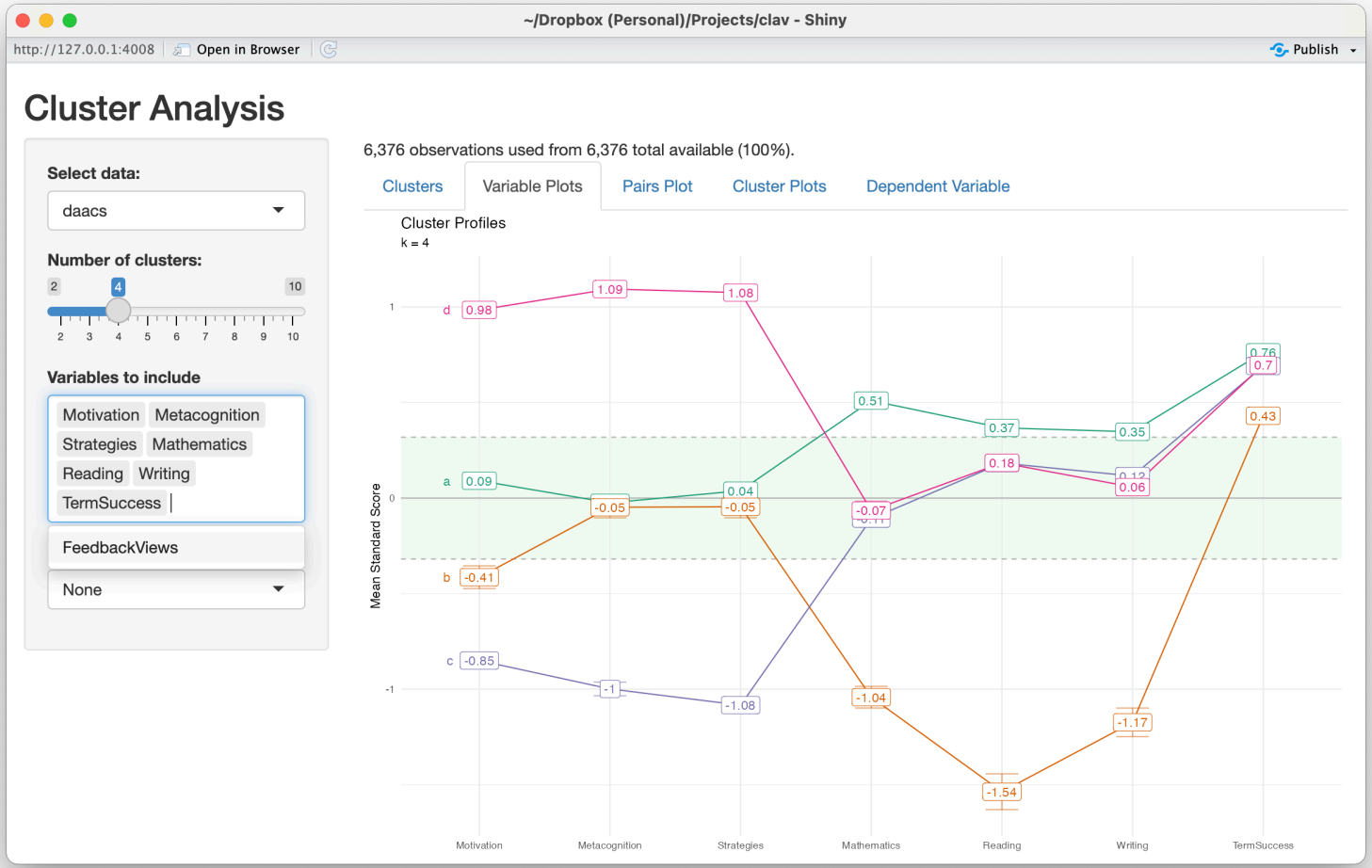






Shiny Application

```
clav::cluster_shiny(daacs = daacs) # NOTE: Can pass an arbitrary named parameters of data.frames
```



Deploying Application with Your Data (app.R)

```r
library(clav)                                                    # Load packages

data("daacs", package = "clav")                                  # Load some data frames
data("pisa2015", package = "clav")
pisa_usa <- pisa2015 |> dplyr::filter(country == "UNITED STATES")
pisa_can <- pisa2015 |> dplyr::filter(country == "CANADA")

data_frames <- list(                                             # The data_frames list object
    "DAACS" = daacs,                                             # contains the data.frames.
    "PISA USA" = pisa_usa,                                       # The name is required and is
    "PISA Canada" = pisa_can                                     # what the user sees.
)

server <- clav::clav_shiny_server                                # Copy the server and UI Shiny
ui <- clav::clav_shiny_ui                                        # functions from clav.

app_env <- new.env()                                             # Create a new empty environment.
assign("data_frames", data_frames, app_env)                      # Assign data_frames to app_env.

environment(server) <- as.environment(app_env)                   # Assigned the data_frames
environment(ui) <- as.environment(app_env)                       # object to the Shiny functions.

shiny::shinyApp(ui = ui, server = server)                        # Run the app
```

Thank you!

jason.bryer@cuny.edu
@jbryer
@jbryer@vis.social
github.com/jbryer/clav