AIR Forum 2023: Integrating Cluster Analysis to Select Peer and Aspirant Comparison Groups

University of West Florida

Introduction

In this document, we will demonstrate the process of performing a cluster analysis on IPEDS data. Using this document as a guide, your office should also be able to perform a cluster analysis using R. To run this analysis, we will use the most up to date IPEDS data available in the IPEDS Data Center as of the drafting of the document (June 2023), labeling this data `ipeds_data_tbl`. Examples shown in this document will use our data file, but the user is free to use any IPEDS data or perhaps data from other locations to run their analyses.

The main objective of this analysis is to identify peer and aspirant comparison groups. Clustering is a powerful unsupervised learning technique that allows us to identify structure in our data. For our purposes, we will use the K-means clustering algorithm.

Load Required Packages

First, we need to load the required packages for our analysis:

- `tidyverse`: Collection of packages for data manipulation and visualization.
- `ggplot2`: Powerful tool for creating professional plots. Note: `ggplot2` is also part of `tidyverse`.
- `ggbiplot`: An extension of `ggplot2` for creating biplots.
- `recipes`: A package for pre-processing your data before analysis.
- `factoextra`: Simplifies the extraction and visualization of the output of exploratory data analysis.
- `tidyquant`: Used for financial analysis and the creation of visually apealing plots. In this analysis, we use `tidyquant` specifically for the `theme_tq()` function to modify the appearance of the cluster plot

The `ggbiplot` package is not hosted on CRAN. You can install it from GitHub using the `devtools` package.

```
# Install ggbiplot if it's not already installed
if (!require(ggbiplot)) {
  if (!require(devtools)) {
    install.packages("devtools")
    library(devtools)
  }
  devtools::install_github("vqv/ggbiplot")
}
```

Now, we load the packages:

```
# Load libraries
library(tidyverse) # for data manipulation and visualization
library(ggplot2) # for creating professional plots
```

```
library(ggbiplot) # for creating biplots
library(recipes) # for pre-processing your data
library(factoextra) # for visualizing the output of the cluster analysis
library(tidyquant) # for modifying the visualization of the output of the cluster analysis
```

Please replace the current "Load Required Packages" section in your Rmarkdown document with the above text and code. Remember to run the `devtools::install_github("vqv/ggbiplot")` code to install the `ggbiplot` package if you haven't done so already.

Load Data

Load the data file. The easiest way to do this is to save the data as a csv and load it into the session using the `read.csv` function and the `as_tibble` function to turn the `data frame` into a `tibble`. Note that to conduct a k-means cluster analysis, all fields included in the cluster portion of the analysis must be numeric fields. There are methods for handling categorical variables that are beyond the scope of this document.

```
ipeds_data_tbl <- read.csv(
  "[insert file path here].csv"
) %>%
  as_tibble()
```

Principal Component Analysis of the Variables

Principal Component Analysis (PCA) is a statistical technique that is used to emphasize variation and bring out strong patterns in a dataset. We use it to transform our original variables into a new set of variables called principal components. The first principal component explains the most variance in the data, the second principal component explains the second most, and so on. PCA helps to simplify the complexity in high-dimensional data while retaining trends and patterns. It does this by transforming the data into fewer dimensions, which can then be visualized.

In this analysis, we're performing PCA to gain insights into the relationships between our variables. PCA will allow us to reduce the dimensionality of our data, making it easier to visualize and interpret.

First, we set the seed to ensure the reproducibility of our results:

```
# Set seed for reproducibility
set.seed(100)
```

Next, we create a table of variable names, renaming them in a var_{} format for better visualization in our biplot. Note that I remove any identifier fields from the set of columns before running the PCA. You may have different or additional fields in your data, ensure that these are all removed (in your code, replace `-c("UnitID", "Institution.Name", "Sector")` with your fields that you will remove).

```
# Create variable table with renamed variables
vars_tbl <- tibble(
  vars_chr    = ipeds_data_tbl %>%
    select(-c("UnitID", "Institution.Name", "Sector")) %>%
    colnames(),
  varnums_chr = str_glue("var_{seq_along(vars_chr)}")
)
```

We then apply these new variable names to our data:

```
# Apply new variable names to data
dclust_vars_pca_tbl <- ipeds_data_tbl %>%
  rename_with(
    function(x) vars_tbl %>% filter(vars_chr == x) %>% pull(varnums_chr),
    -c("UnitID", "Institution.Name", "Sector")
  )
```

Now, we're ready to perform PCA on our variables:

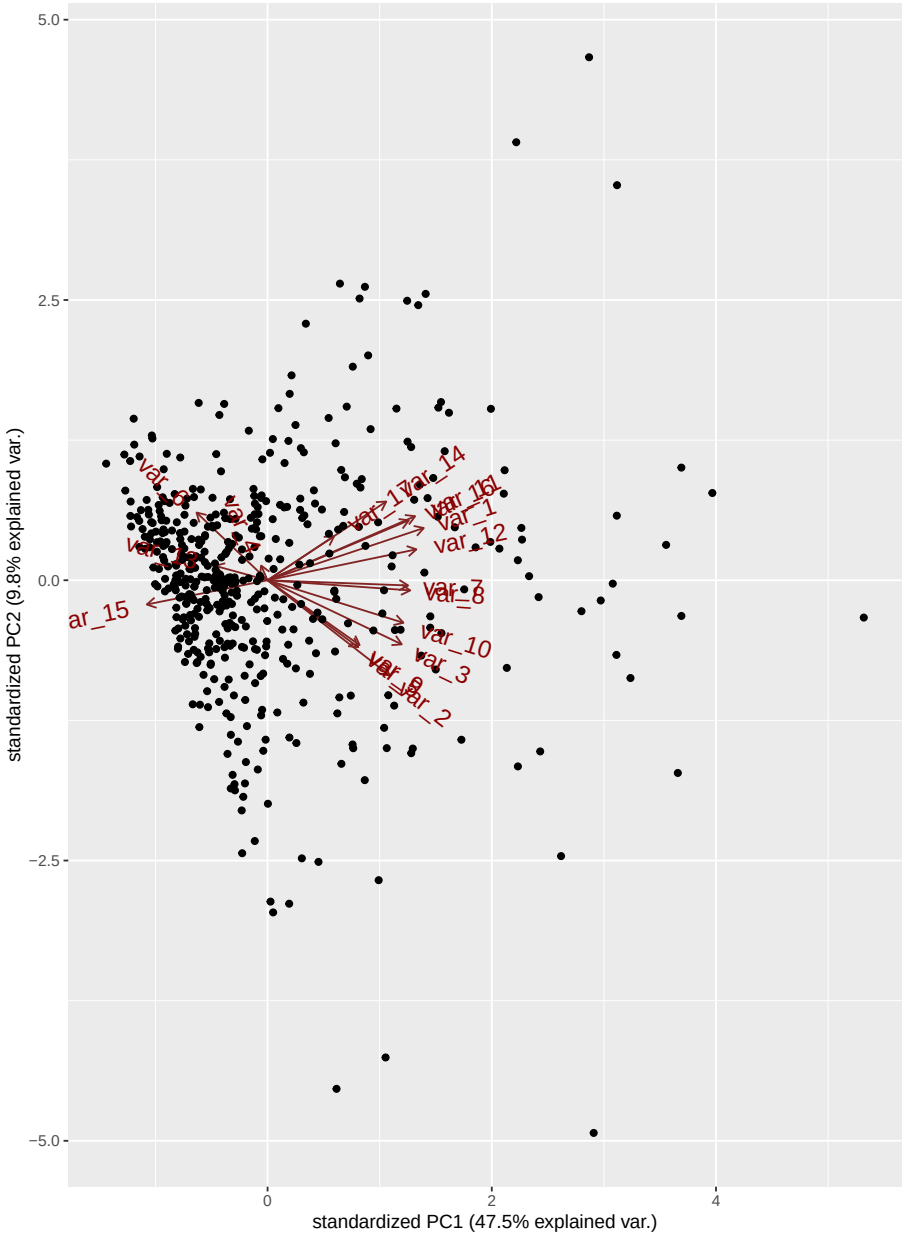
```
# Run PCA
dclust_vars_pca <- dclust_vars_pca_tbl %>%
  select(-c("UnitID", "Institution.Name", "Sector")) %>%
  prcomp(center = T, scale. = T)
```

We create a biplot of the PCA results using ggbiplot:

```
# Create biplot
dclust_vars_pca_biplot <- dclust_vars_pca %>%
  ggbiplot::ggbiplot(varname.size = 5, var.scale = 1)
```

Finally, let's display our biplot. This graphical representation will provide a clear view of how our variables are related and how well they are represented in the first two principal components:

```
# Display biplot
print(dclust_vars_pca_biplot)
```



Interpreting the Biplot

After generating a PCA biplot using `ggbiplot`, one of the key aspects we can examine is the relationships between variables. This is communicated through the direction and length of the arrows representing the variables.

In the biplot above, each arrow corresponds to a variable in our dataset (renamed as `var_{i}` for the purpose of visualization). The directions and lengths of the arrows in relation to the axes and each other gives us valuable information about the correlation between variables:

- If the arrows for two variables are close to each and form a small angle, this indicates a strong positive correlation between those two variables

- If the arrows for two variables are diametrically opposed (i.e., they roughly form a straight line with the origin), this indicates a strong negative correlation. In this case, as the value of one variable increases, the value of the other variable tends to decrease.
- If the arrows for two variables are orthogonal (i.e., they form a right angle), this indicates that the variables are not correlated. There is no specific linear relationship that as one variable increases, the other will either increase or decrease in a predictable manner.
- Variables with long arrows contribute more significantly to the variance in our data. These variables have a strong influence on the principal components and, consequently, the structure we observe in the plot.
- Variables with short arrows contribute less to the variance. They have less of an influence on the principal components and the overall structure of the data.

By studying the direction and length of the arrows in the biplot, we can obtain a detailed understanding of how variables in our dataset relate to each other and how they contribute to the overall variance in our data. This can guide our next steps in data analysis, including feature selection, cluster analysis, and predictive modeling.

Preprocessing the Data

When working with machine learning algorithms such as k-means clustering, preprocessing our data is a critical step. This is especially important when our variables are measured on different scales or have different variances. The k-means algorithm calculates distances between points to form clusters, and if variables are on different scales, those with larger scales can disproportionately influence the clustering process. As such, we need to ensure our variables are all on a similar scale.

To accomplish this, we'll be using the `recipes` package. This package allows us to specify a series of preprocessing steps to apply to our data using a recipe. The `recipes` package is a tidy approach to preprocessing, and it integrates well with the rest of our tidyverse pipeline.

We'll use two steps in our recipe:

1. **Centering**: This subtracts the mean of each variable, shifting the distribution so that it's centered around zero.
2. **Scaling**: This divides each variable by its standard deviation, converting our data to z-scores. This ensures that each variable has a mean of 0 and a standard deviation of 1.

After these steps, our data is preprocessed and ready for k-means clustering. All our numeric variables will be on the same scale, ensuring a fairer and more effective clustering process. Here is the code to accomplish this:

```
# Set seed for reproducibility
set.seed(100)

# Specify the recipe and apply it to our data
dclust_vars_prep_tbl <- recipe(
  UnitID ~ .,
  data = ipeds_data_tbl %>%
    select(-c("Institution.Name", "Sector"))
) %>%
  # Center all numeric variables
  step_center(all_numeric()) %>%
  # Scale all numeric variables
  step_scale(all_numeric()) %>%
```

```
  # Prepare the recipe
  prep() %>%
  # Apply the recipe to our data
  bake(ipeds_data_tbl)
```

Note again that I remove any identifier fields from the set of columns before preprocessing. As before, you will need to modify this code to ensure that only the columns being included in the cluster analysis are being preprocessed.

K-Means Clustering

Once our data is appropriately preprocessed, we can now move to the core of our analysis: k-means clustering. The goal of k-means clustering is to partition our data into `k` distinct, non-overlapping clusters, where `k` is a predefined number of clusters.

We've chosen to set `k` to 10, meaning we are looking to identify 10 distinct clusters in our data. Our team carefully decided on this number of clusters, taking into consideration the size of our dataset. We aimed for an average cluster size of approximately 50 institutions, ensuring a sufficiently large pool for subsequent fine-tuning. This allows us to effectively filter and select only the most relevant institutions, aligning with our specific institutional goals. It's worth noting that there are many statistical methods available for determining the statistically "optimal" number of clusters, such as the Elbow method or the Silhouette method. However, for this analysis, we've opted to set `k` to 10 directly.

```
# Set seed for reproducibility
set.seed(100)

# Set number of clusters
clustnum <- 10

# Run k-means clustering
kclust_obj <- dclust_vars_prep_tbl %>%
  select(-UnitID) %>%  # Exclude non-numeric variables
  dist() %>%  # Compute distance matrix
  kmeans(centers = clustnum, nstart = 50)  # Run k-means clustering
```

In the k-means clustering code block above, nstart refers to the number of random sets to be chosen. K-means is a heuristic algorithm and the results can depend on the initial random assignments of observations to clusters. By setting nstart to a number greater than 1 (in this case, 50), we can mitigate this issue.

To evaluate our clustering results, we calculate the R-squared value and the Pseudo F Statistic. The R-squared value represents the proportion of the total variance that is 'explained' or 'accounted for' by the between-cluster variance. The Pseudo F Statistic is another measure of cluster separation where higher values generally indicate better clustering.

```
# Calculate R-squared value
kclust_r2 <- kclust_obj$betweenss/kclust_obj$totss
kclust_r2
```

```
## [1] 0.8883857
```

```
# Calculate Pseudo F Statistic
kclust_pseudo_f <- (kclust_obj$betweenss/(clustnum - 1)) %>%
    magrittr::divide_by(kclust_obj$tot.withinss/(dim(dclust_vars_prep_tbl)[1] - clustnum))
kclust_pseudo_f
```

```
## [1] 416.5434
```

After running the k-means clustering, we computed the R-squared and Pseudo F statistic to evaluate the quality of our clustering. These statistics provide us with valuable information about how well our clustering has performed.

The R-squared value we obtained is 0.89, which means that approximately 89% of the total variance in our data is accounted for by the between-cluster variance. This is a high value, suggesting that our clusters are well separated and that each cluster is relatively homogeneous. In other words, institutions within the same cluster are similar to each other (in terms of the variables considered) and different from institutions in other clusters.

Our Pseudo F statistic value is 416.54, which is relatively high, indicating that our clusters are well separated. The Pseudo F statistic measures the ratio of the between-cluster variance to the within-cluster variance. Higher values generally indicate better clustering because they suggest that the between-cluster variance (i.e., the differences between clusters) is large relative to the within-cluster variance (i.e., the differences within each cluster).

Overall, these values suggest that our k-means clustering has performed well in separating our data into distinct, meaningful clusters. When you run your analysis, look for a high percentage for R-squared and a high Pseudo F statistic. However, it is important to consider these values in the context of our specific data and research objectives. They should not be interpreted in isolation, and we should also consider other aspects of our analysis, such as the substantive meaning and interpretability of our clusters.

Finally, we compute the eigenvalues from a Principal Component Analysis (PCA) of our preprocessed data. This allows us to understand the amount of variance explained by each principal component in our data. This will be beneficial when visualizing our clusters and interpreting our results.

```
# Compute eigenvalues
kclust_eigen_tbl <- dclust_vars_prep_tbl %>%
  select(-UnitID) %>%  # Exclude non-numeric variables
  prcomp(center = T, scale. = T) %>%  # Perform PCA
  factoextra::get_eigenvalue()  # Extract eigenvalues
kclust_eigen_tbl
```

```
##         eigenvalue variance.percent cumulative.variance.percent
## Dim.1  8.081501284      47.53824285                     47.53824
## Dim.2  1.664838586       9.79316815                     57.33141
## Dim.3  1.432424038       8.42602375                     65.75743
## Dim.4  1.212310984       7.13124108                     72.88868
## Dim.5  0.927462163       5.45565978                     78.34434
## Dim.6  0.765836560       4.50492094                     82.84926
## Dim.7  0.668207552       3.93063266                     86.77989
## Dim.8  0.522009495       3.07064409                     89.85053
## Dim.9  0.465732244       2.73960143                     92.59013
## Dim.10 0.384109252       2.25946619                     94.84960
## Dim.11 0.257461824       1.51448132                     96.36408
## Dim.12 0.171481760       1.00871624                     97.37280
## Dim.13 0.167194171       0.98349512                     98.35629
## Dim.14 0.101483464       0.59696156                     98.95326
## Dim.15 0.096449158       0.56734799                     99.52060
## Dim.16 0.077302091       0.45471818                     99.97532
## Dim.17 0.004195374       0.02467867                    100.00000
```

The first principal component (Dim.1) explains approximately 47.54% of the variance in the data. The second principal component (Dim.2) accounts for around 9.79% of the variance, and so on. It's noteworthy

that a large proportion of the variance is captured by the first few dimensions. By Dimension 5, about 78.34% of the variance in the data is already accounted for.

In the context of a PCA biplot, dimensions with larger eigenvalues will have a greater influence on the plot. This means that the first few dimensions (which account for the most variance) will have a greater impact on the separation and relative positioning of points (institutions) and vectors (variables) in the biplot. Thus, the biplot will primarily reflect the relationships and correlations among institutions and variables as captured by these leading dimensions.

Remember, in interpreting PCA results, it's important not just to consider the amount of variance explained by each component, but also the substantive meaning and interpretability of the components in the context of our data and research objectives.

Analyzing the Clusters - Visualization

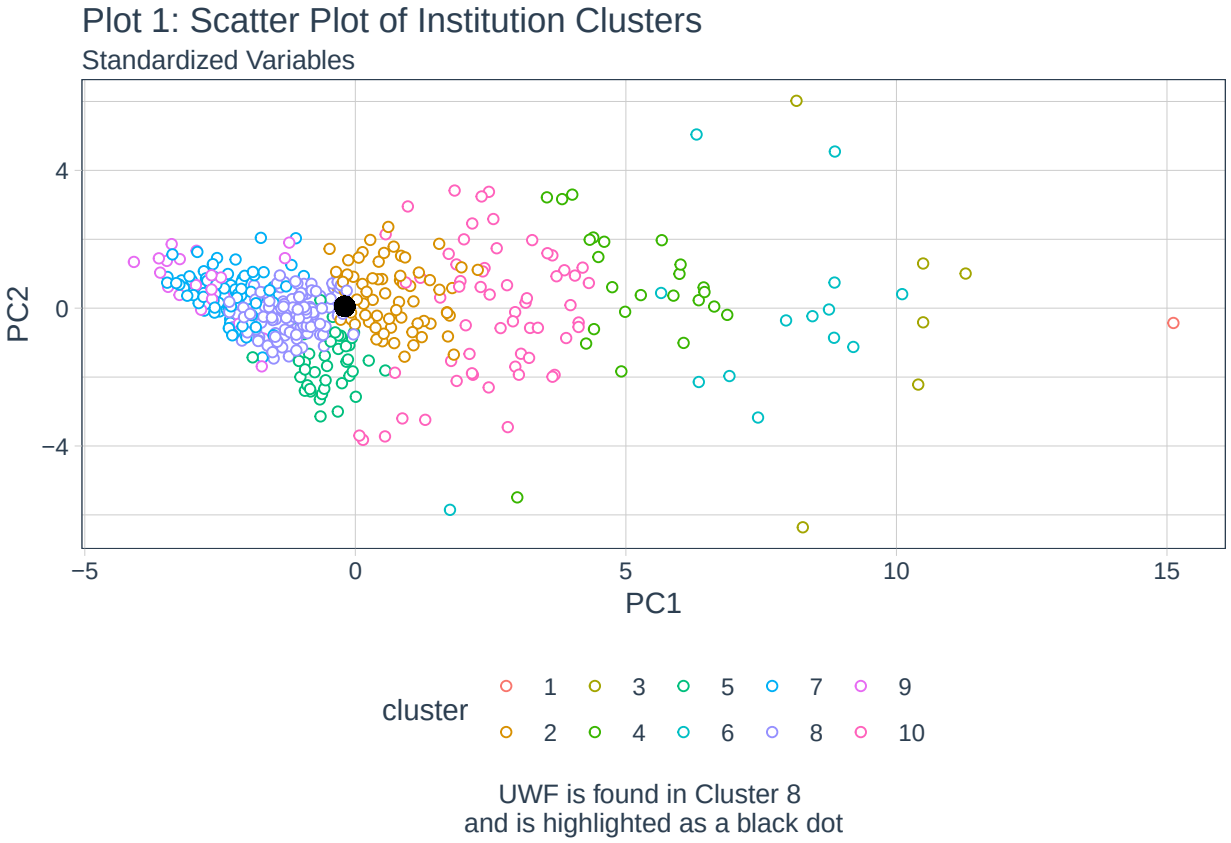
After running the k-means cluster analysis, a beneficial way to examine and understand the resulting clusters is through visualization. We are primarily interested in examining how the institutions group together based on the standardized variables used in the analysis.

We utilize the `factoextra` package, which provides convenient functions for visualizing the results of cluster analyses. Specifically, `fviz_cluster` function can be used to visualize our k-means clusters.

```
# Extract the clustering results and join them to the original institution data
kclust_pts <- factoextra::fviz_cluster(
  kclust_obj,
  data         = ipeds_data_tbl %>% select(-c("UnitID", "Institution.Name", "Sector")),
  geom         = c("euclid")
) %>%
  .$data %>%
  as_tibble() %>%
  select(-name) %>%
  bind_cols(
    ipeds_data_tbl %>%
      select(UnitID, Institution.Name)
  )


# Plot the clustering results
factoextra::fviz_cluster(
  kclust_obj,
  data         = ipeds_data_tbl %>% select(-c("UnitID", "Institution.Name", "Sector")),
  geom         = c("point"),
  ellipse.type = "none",
  shape = 21,
  show.clust.cent = FALSE,
  xlab = "PC1",
  ylab = "PC2") +
  ggtitle(
      "Plot 1: Scatter Plot of Institution Clusters",
      subtitle = "Standardized Variables"
  ) +
  tidyquant::theme_tq() +
  ggplot2::labs(
    caption = str_glue(
      "UWF is found in Cluster {kclust_pts %>% filter(UnitID==138354) %>% pull(cluster)}
```

```
    and is highlighted as a black dot"
  )
) +
theme(
    plot.caption = element_text(
        size = 10,
        hjust = 0.5
    )
) +
geom_point(
  aes(
    x = kclust_pts %>% filter(UnitID == 138354) %>% pull(x),
    y = kclust_pts %>% filter(UnitID == 138354) %>% pull(y)
  ),
  size = 3,
)
```



This visualization provides a clear and concise view of our clustering results. Each institution is represented as a point on the plot, with the position of the point determined by the first two principal components. The coloring of the points indicates the cluster to which each institution is assigned. Our reference institution (UWF) is highlighted with a distinct marker for easy identification. With this plot, we can intuitively understand the grouping of the institutions and their relative positions in the data space.

Writing the Results to a CSV File

Having completed the clustering analysis, the final step is to write our results to a CSV file for further use. This will allow us to preserve our findings and refer back to them in the future. We include columns for the type of analysis and the year in which the analysis was conducted. This will facilitate tracking changes over time.

```
# Add Type column
kclust_pts <- kclust_pts %>%
  mutate(Type = "K-means")

# Add Year column
year_chr <- "2023"
kclust_pts <- kclust_pts %>%
  mutate(
    Year    = str_glue("{year_chr}") %>% as.character,
    cluster = as.character(cluster),
    UnitID  = as.character(UnitID)
  )
```

In addition to writing the results from this year, we also include data from a previous analysis conducted in 2022. By combining the datasets, we will be able to track how the clusters change from year to year. While this step is optional, it is highly recommended to establish a practice of keeping track of yearly updates to peer and aspirant institution lists.

```
# Read in the previous year's data
kclust_orig_tbl <- read_csv(
  "<Path to 2022 data file>/peer and aspirant previous clusters.csv"
) %>%
  as_tibble() %>%
  select(x, y, coord, cluster, UnitID, Institution.Name, Type, Year) %>%
  mutate(
    cluster = as.character(cluster),
    UnitID  = as.character(UnitID)
  )

# Combine the current and previous years' data
kclust_pts <- kclust_pts %>%
  bind_rows(kclust_orig_tbl)
```

Finally, we write our combined data to a CSV file. Be sure to replace with the path where you want the output file to be saved.

```
# Write the table to a CSV file
kclust_pts %>%
  write.table(
    "<Path to desired output location>/peer and aspirant with new clusters.csv",
    row.names = F,
    sep       = ","
  )
```

With our data saved, we can easily refer back to our clustering results and track how they evolve over time. The data is additionally now in an easy format to export to Tableau for use in a Peer and Aspirant Institution dashboard, allowing your institution to track your peer and aspirant institutions year over year.