Archetypal Analysis Module Guide

Andeas Holmer Bigom, Michael Alexander Harborg,

Marcus Presutti & Oliver Rosbæk Elmgreen

02466 | Project work - Bachelor of Artificial Intelligence and Data

20th June, 2022

Contents

1 About The Module

1.1 The Purpose

This module can be used to perform, evaluate and analyse Conventional Archetypal Analysis, Ordinal Archetypal Analysis and Response Bias Ordinal Archetypal Analysis. The module is developed by Andreas Bigom, Michael Harborg, Marcus Presutti and Oliver Elmgren in a collaboration between the Technical University of Denmark and Copenhagen Business School in order to enable students to analyse human questionnaire data in an effective and meaningful way.

1.2 Types of Archetypal Analysis Throughout the Module

This module is able to perform three different but fairly similar analyses, namely Conventional Archetypal Analysis, Ordinal Archetypal Analysis and Respnse Bias Ordinal Archetypal Analysis. The main features of the diferent models are;

Method	Abbr.	Features
`Conventional Archetypal Analysis`	`CAA`	Fast analysis, due to the lower amount of variables to perform optimization upon. Not suitable for datasets which contain noise or response bias. In theory not suitable for ordinal data, but is still able to produce a result on these.
`Ordinal Archetypal Analysis`	`OAA`	Slower than `CAA`, but is suitable for ordinal data and is able to model a global noise and response bias for the dataset. Not suitable for datasets, where the noise and response bias is subject specific.
`Reponse Bias Ordinal Archetypal Analysis`	`RBOAA`	Slower than `CAA`, same efficiency as `OAA`, but a hot start with `OAA` is greatly adviced, which makes it less time efficient. Is suitable for ordinal data and is able to model subject specific noise and response bias for each subject of the dataset. The most advanced analysis form of the module, and outperforms `CAA` and `OAA` in most cases.

2 Installation

Follow these steps to ensure a correct installation of the module.

2.1 Pip Installation of Module

To install the module as a package for you current environment of python, use the pip command. The version history, description and other information about the module can be found on `pipy.org` website at `https://pypi.org/project/AA-module/` for convenience. To install the module through pip, run the following command;

```
pip install AA-module
```

2.2 Requirements

In order to ensure that all the requirements of the dependencies of the module have been fulfilled, pip will go through each requirement at installation. The requirements are as follows;

Package	>=**version**
`Python`	3.9.7
`pandas`	Not Specified
`numpy`	Not Specified
`matplotlib`	Not Specified
`torch`	Not Specified
`scipy`	Not Specified
`sklearn`	Not Specified

3 Import and Initialization

To import the AA module for use in your script, run the following code:

```
from AAM import AA
```

To create an instance of the AAM module in your script, run the following code:

```
<instance_name> = AA()
```

4 Load Data

4.1 Load Data from CSV File

To load data into your AA instance from a local CSV file, run the following command:

```
<instance_name>.load_csv(filename: str, columns: list[int] = None, rows: int = None, mute: bool = False)
```

The method parameters for this method are the following:

Parameter	Description	Default Value
`filename`	A variable of type `str` describing the filename of the CSV file. Make sure that the CSV file is located in the CWD, else specify the alternative directory in this parameter.	This variable must be explicitly specified.
`columns`	A list of type int containing the indices of the columns to load from the CSV file.	None, thus all columns will be loaded.
`rows`	A variable of type int describing the amount of rows to load from the CSV file. The parameter will correspond to the index of the last row loaded.	None, thus all rows will be loaded.
`mute`	A variable of type bool describing whether the function should be executed without prompt messages in the console.	False, thus messages will be shown.

4.2 Load Data from a numpy.ndarray

To load data into your AA instance from a `numpy.ndarray`, run the following command:

```
<instance_name>.load_data(X: np.ndarray, columns: list[str])
```

The method parameters for this method are the following:

Parameter	Description	Default Value
`X`	A variable of type `numpy.ndarray` containing the data. X should have dimensions *(N,M)*, where *N = data-points* and *M = attributes*.	This variable must be explicitly specified.
`columns`	A list of type `str` containing the column names of the data. The list must be exactly in the same order as `X`.	This variable must be explicitly specified.

5 Create Data

5.1 Synthetically Created Data

To create synthetic data and store it in your AA instance, run the following command:

```
<instance_name>.create_synthetic_data(N: int = 1000, M: int = 10, K: int = 3, p:  int = 6,
sigma:  float = -20.0, rb:  bool = False, b_param:  float = 100.0, a_param:  float = 1.0,
sigma_dev:  float = 0, mute = False)
```

The method parameters for this method are the following:

Parameter	Description	Default Value
`N`	A variable of type `int` describing the number of subjects of the dataset.	1000
`M`	A variable of type `int` describing the number of attributes of the dataset.	10
`K`	A variable of type `int` describing the number of archetypes of the dataset.	10
`p`	A variable of type `int` describing the number of points on a likert scale of the dataset.	6
`sigma`	A variable of type `float` describing the noise of the dataset before softplus. -20 corresponds to very little noise and -1.5 corresponds to a lot of noise.	-20.0
`rb`	A variable of type `bool` describing whether the data should be created with response bias.	False
`b_param`	A variable of type `float` describing the value of the dirichlet distribution, which the response bias should be sampled from. 1 corresponds to a lot of response bias and 100 corresponds to very little.	100
`a_param`	A variable of type `float` describing the value of the dirichlet distribution, which the linear combination of archetypes of each subject should be sampled from. 1 corresponds to an equal sample.	1
`sigma_dev`	A variable of type `float` describing deviation of sigma for each subject of the dataset. 0 corresponds to no deviation.	0.0
`mute`	A variable of type bool describing whether the function should be executed without prompt messages in the console.	False, thus messages will be shown.

6 Perform and Get Archetypal Analysis

6.1 Perform Archetypal Analysis

To perform Conventional Archetypal Analysis, Ordinal Archetypal Analysis or Response Bias Ordinal Archetypal Analysis, run the following command:

```
<instance_name>.analyse(K: int = 3, p:  int = 6, n_iter:  int = 1000, early_stopping:  bool = True, model_type = "all", lr:  float = 0.01, mute:  bool = False, with_synthetic_data:  bool = False, with_hot_start:  bool = False)
```

OBS. when performing `RBOAA`, it is highly advicable that the `with_hot_start` parameter is set to true, as the `RBOAA` has too many variable to optimize upon to perform stable convergence, if the problem becomes large.

The method parameters for this method are the following:

Parameter	Description	Default Value
`K`	A variable of type `int` describing the number of archetypes to initiate the Archetypal Analysis with.	3
`p`	A variable of type `int` describing the number of points on a likert scale the data is described in.	6
`n_iter`	A variable of type `int` describing the number of iterations the algorithm should run for.	1.000
`early_stopping`	A variable of type `bool` describing whether the analysis should be performed with early stopping. Early stopping will stop the analysis before the number of iterations have been reached, if the loss has converged.	True
`AA_type`	A variable of type `str` describing the type of Archetypal Analysis to perform. This variable can take the following values; `"all"`, `"CAA"` (Conventional Archetypal Analysis), `"TSAA"` (Two Step Archetypal Analysis), `"OAA"` (Ordinal Archetypal Analysis) and `"RBOAA"` (Response Bias Ordinal Archetypal Analysis).	`"all"`
`lr`	A variable of type `float` describing the learning rate of the Adam optimizer used in the Archetypal Analysis.	`0.01`
`mute`	A variable of type `bool` describing whether the analysis should display guiding output in the console (such as a loading bar).	`False`
`with_synthetic_data`	A variable of type `bool` describing whether the analysis should be performed on synthetically created data, or data from a loaded dataset.	`False`
`with_hot_start`	A variable of type `bool` describing whether the analysis should be performed with hot start of model parameters. This variable is only relevant for `model_type` = `"RBOAA"`, as this model will hotstart from an `"OAA"`.	`False`

6.2 Get Archetypal Analysis Result

To get the analysis result element of an analysis of your AA instance, run the following command:

```
<instance_name>.get_analysis(model_type:  str = "CAA", result_number:  int = 0, with_synthetic_data:  bool = False)
```

The method parameters for this method are the following:

Parameter	Description	Default Value
`model_type`	A variable of type `str` describing the type of analysis to return. This variable can take the following values; `"CAA"`, `"OAA"`, `"RBOAA"`.	`"CAA"`
`result_number`	A variable of type `int` describing the index of the analysis to return. This variable is used when multiple analysis have been performed. The analysis at index `0` corresponds to the most recent analysis.	`0`
`with_synthetic_data`	A variable of type `bool` describing whether the analysis to return was performed on synthetically created data.	`False`

6.3 Archetypal Analysis Result Class

The Archetypal Analysis Result Classes contains the following attributes:

6.3.1 CAA result class

Attribute	Description
`A`	A matrix of analysis. Describes each subject as a linear combination of the archetypes.
`X`	X matrix of analysis. The dataset which the analysis was performed on.
`X_hat`	X_hat matrix of analysis. The approximation of the dataset from the analysis.
`n_iter`	The number of iterations the analysis was run for.
`loss`	The loss of each iteration of the analysis.
`Z`	Z matrix of the analysis. Describes the archetypes by the attributes.
`K`	Number of archetypes of the analysis.
`p`	Number of points on a likert scale of the data.
`time`	The time the analysis was run for.
`columns`	The columns the dataset.
`type`	The type the analysis.
`with_synthetic_data`	Boolean describing whether the analysis was performed on synthetic data or not.
`N`	Number of subjects of the data.

6.3.2 OAA and RBOAA result class

Attribute	Description
`A`	A matrix of analysis. Describes each subject as a linear combination of the archetypes.
`X`	X matrix of analysis. The dataset which the analysis was performed on.
`X_hat`	X_hat matrix of analysis. The approximation of the dataset from the analysis.
`n_iter`	The number of iterations the analysis was run for.
`loss`	The loss of each iteration of the analysis.
`Z`	Z matrix of the analysis. Describes the archetypes by the attributes.
`K`	Number of archetypes of the analysis.
`p`	Number of points on a likert scale of the data.
`time`	The time the analysis was run for.
`columns`	The columns the dataset.
`type`	The type the analysis.
`with_synthetic_data`	Boolean describing whether the analysis was performed on synthetic data or not.
`N`	Number of subjects of the data.
`b`	beta vector/matrix (response bias values) of analysis.
`sigma`	noise value/vector of analysis.
`X_tilde`	X matrix of dataset projected into the continuous space w.r.t. betas.
`Z_tilde`	Z matrix of dataset projected into the continuous space w.r.t. betas.

7 Save and Load Analysis Locally

7.1 Save Archetypal Analysis Locally

To save your Archetypal Analysis locally on your device, run the following command:

```
<instance_name>.save_analysis(filename: str = "analysis", model_type: str = "CAA",
result_number: int = 0, with_synthetic_data: bool = False, save_synthetic_data: bool = True)
```

The analysis is then saved to a folder named `results` (or `synthetic`$_{r}esultsif analysiswasperformedonsyntheticallycreatedd$
`OBS. if you get an error, it might be due to a lack of one of these two folders. In order to fix the error, create the folders with the exact names and place them in your current working directory.`
`The method parameters for this method are the following:`

Parameter	Description	Default Value
`filename`	A variable of type `str` describing the desired filename of the analysis.	`"analysis"`
`model_type`	A variable of type `str` describing the type of analysis to save. This variable can take the following values; `"CAA"`, `"OAA"`, `"RBOAA"`.	`"CAA"`
`result_number`	A variable of type `int` describing the index of the analysis to save. This variable is used when multiple analysis have been performed. The analysis at index `0` corresponds to the most recent analysis.	`0`
`with_synthetic_data`	A variable of type `bool` describing whether the analysis to save was performed on synthetically created data.	`False`
`save_synthetic_data`	A variable of type `bool` describing whether the synthetic data which the analysis was performed on should be saved alongside with the analysis. Only relevant if the `with_synthetic_data` parameter is set to `True`.	`False`

7.2 Load Locally Stored Archetypal Analysis

`To load a locally stored Archetypal Analysis, run the following command:`

```
<instance_name>.load_analysis(filename: str = "analysis", model_type: str = "CAA",
with_synthetic_data: bool = False)
```

`The analysis is then loaded from a folder named results (or synthetic results if analysis was performed on synthetically created data.) in your current working directory.`
`The method parameters for this method are the following:`

Parameter	Description	Default Value
`filename`	A variable of type `str` describing the filename of the analysis.	`"analysis"`
`model_type`	A variable of type `str` describing the type of analysis to load. This variable can take the following values; `"CAA"`, `"OAA"`, `"RBOAA"`.	`"CAA"`
`with_synthetic_data`	A variable of type `bool` describing whether the analysis to load was performed on synthetically created data.	`False`

8 Visualize Analysis

8.1 Visualize Analysis through Plots

To visualize the Conventional Archetypal Analysis, Ordinal Archetypal Analysis or Response Bias Ordinal Archetypal Analysis, run the following command:

```
<instance_name>.plot(model_type: str = "CAA", plot_type: str = "PCA_scatter_plot", title: str = "", save_figure: bool = False, filename: str = "figure", result_number: int = 0, attributes: list[int] = [1,2], archetype_number: int = 1, types: dict = {}, weighted: str = "equal_norm", subject_indexes: list() = [1], attribute_indexes: list(tuple()) = [], with_synthetic_data: bool = False )
```

OBS. if you loop over plots with the module, multiple datasubsets might appear stacked on top of one another. If this happens, import matplotlib.pyplot as plt and write the line plt.clf() at the end of each loop.

The method parameters for this method are the following:

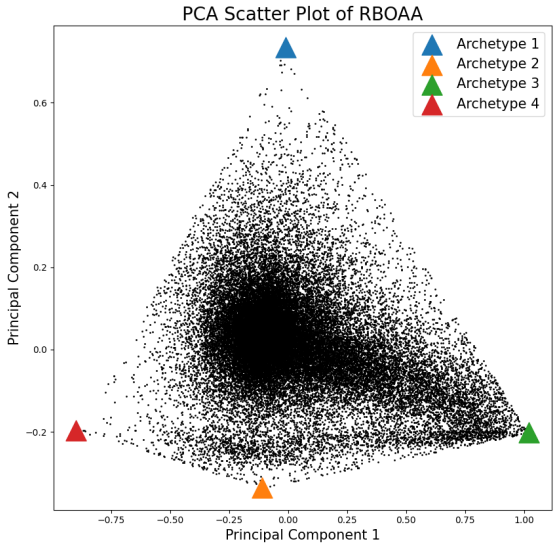
Parameter	Description	Default Value
`model_type`	A variable of type `str` describing the type of Archetypal Analysis to visualize. This variable can take the following values; `"CAA"`, `"OAA"` and `"RBOAA"`.	`"CAA"`
`plot_type`	A variable of type `str` describing the type of plot to visualize the analysis by. This variable can take the following values; `"PCA_scatter_plot"`, `"attribute_scatter_plot"`, `"loss_plot"`, `"typal_plot"`, `"mixture_plot"`, `"barplot"` and `"barplot_all"`	`"PCA_scatter_plot"`
`title`	A variable of type `str` describing the title of the plot. If not specified, then the plot will have a default title.	`""`
`save_figure`	A variable of type `bool` describing whether the figure should be saved locally on your device.	`False`
`filename`	A variable of type `str` describing the name of the file, if the figure should be saved locally on your device.	`figure`
`result_number`	A variable of type int describing the index of the analysis to visualize. This variable is used when multiple analysis have been performed. The analysis at index `0` corresponds to the most recent analysis.	`0`
`attributes`	A list of type `int` describing the indices of the two attributes to plot if the `plot_type` `"attribute_scatter_plot"` is chosen. This parameter should only be specified if the `plot_type` is defined as `"attribute_scatter_plot"`.	`[0,1]`
`archetype_number`	A variable of type `int` describing the index of the archetype which should be visualized if the `plot_type` `"barplot"` is chosen. This parameter should only be specified if the `plot_type` is defined as `"barplot"`.	`0`
`types`	A variable of type `dict` describing the individual types indented for the visualization as keys and their attributes columns indexes as type `list[int]` as values; `{"type_1": [1,2,3], "type_2": [4,5,6], "type_3": [7,8,9]}`. This variable should only be specified if the `plot_type` is defined as `typal_plot`.	`{"type 1": [1],"type 2": [2]}`
`weighted`	A variable of type `str` describing the method used for weighting the types of the visualization. This variable can take the following values; `"none"`, `"equal"`, `"norm"`, `"equal_norm"`. This variable should only be specified if the `plot_type` is defined as `typal_plot`.	`"equal_norm"`
`subject_indexes`	A variable of type `list()` describing the indexes of the subjects which should be used for the plot. Only applicable for the `"pie_chart"` and `"attribute_distribution"`.	`[1]`
`attribute_indexes`	A variable of type `list(tuple())` describing the conditions, on which the data should be filtered on for the plot. The variable should be specified on the following form; `[(attribute_1, condition, value_1),(attribute_2, condition, value_2)...]` (e.g. `[("Country", "=", "DE"),("age", ">", "18")]`). Only applicable for the `"pie_chart"` and `"attribute_distribution"`.	`[]`
`with_synthetic_data`	A variable of type `bool` describing whether the analysis which should be plotted, was performed on synthetically created data.	`False`

8.2 Description of Plot Types

8.2.1 PCA_scatter_plot

```
<instance_name>.plot(model_type: str = "CAA", plot_type = "PCA_scatter_plot",
title: str = "", save_figure: bool = False, filename: str = "figure", result_number: int =
0, with_synthetic_data: bool = False)
```

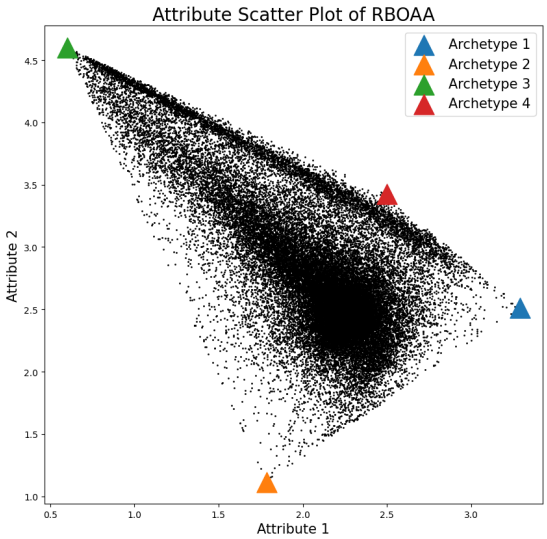
Plots the dataset projected into the first two principal components of a PCA conducted on the archetypes.



8.2.2 attribute_scatter_plot

```
<instance_name>.plot(model_type: str = "CAA", plot_type = "attribute_scatter_plot",
title: str = "", save_figure: bool = False, filename: str = "figure", result_number: int =
0, attributes: list[int] = [1,2], with_synthetic_data: bool = False )
```

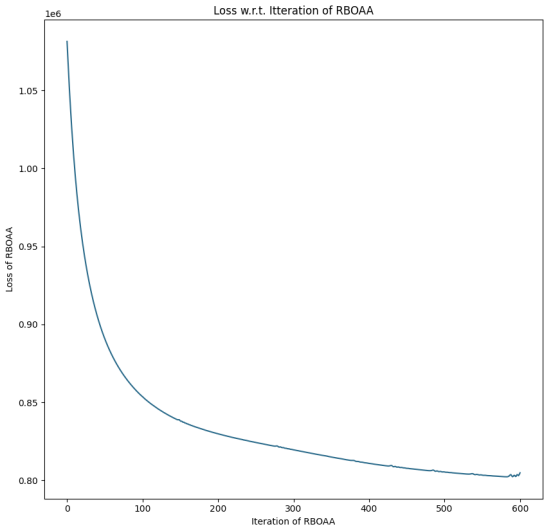
Plots two attributes of the dataset in a two dimensional scatter-plot. The two attributes can be specified in the attributes list.



8.2.3 loss_plot

Plots the loss of the analysis w.r.t. iterations. This plot can be used to unsure that the analysis has converged in the given number of iterations or after early stopping. Keep in mind, good convergenc does not necessarily mean a good result when using these methods.

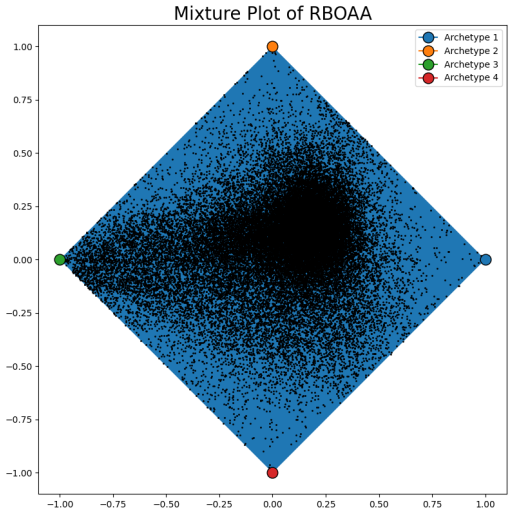
```
<instance_name>.plot(model_type: str = "CAA", plot_type = "loss_plot", title: str = "",
save_figure: bool = False, filename: str = "figure", result_number: int = 0,
with_synthetic_data: bool = False)
```



8.2.4 mixture_plot

```
<instance_name>.plot(model_type: str = "CAA", plot_type = "mixture_plot", title: str = "",
save_figure: bool = False, filename: str = "figure", result_number: int = 0,
with_synthetic_data: bool = False)
```

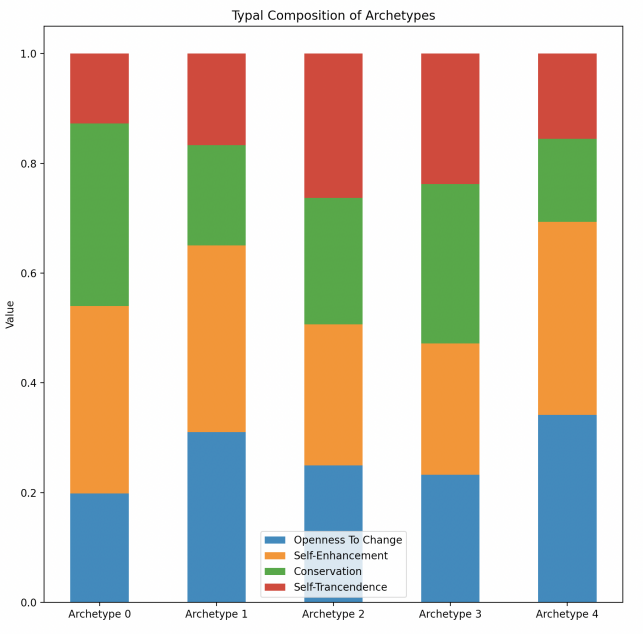
Plots a geometric figure with number of edges corresponding to the number of archetypes of the analysis. Inside the figure, the datapoints are projected w.r.t. their linear combination of the archetypes. That is, if many of the data-points are close to one archetype, then many of these datapoints can be describe mostly by this archetype.



8.2.5 typal_plot

```
<instance_name>.plot(model_type: str = "CAA", plot_type = "typal_plot", title: str = "",
save_figure: bool = False, filename: str = "figure", result_number: int = 0, types: dict =
{}, weighted: str = "equal_norm", with_synthetic_data: bool = False)
```

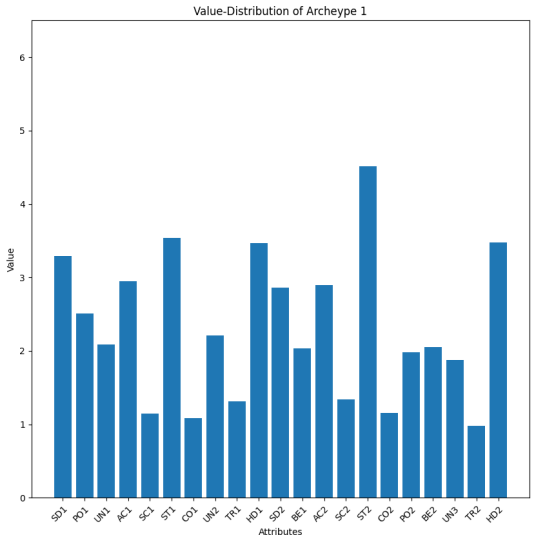
Plots the distribution of the types of each archetype. If an archetype has a large value of one type, then the archetype scores high values in this type. The types can be defined by the user as a dictionary. Keep in mind, when working on datasets, where 1 is Very much like me, the interpretat of the plot should be reversed.
The weighting of the plot, describes whether the bars should be normalized, equalized or normalized and equalized or just shown as raw cumulative values.



8.2.6 barplot

```
<instance_name>.plot(model_type: str = "CAA", plot_type = "barplot", title: str = "",
save_figure: bool = False, filename: str = "figure", result_number: int = 0, archetype_number:
int = 1, with_synthetic_data: bool = False)
```

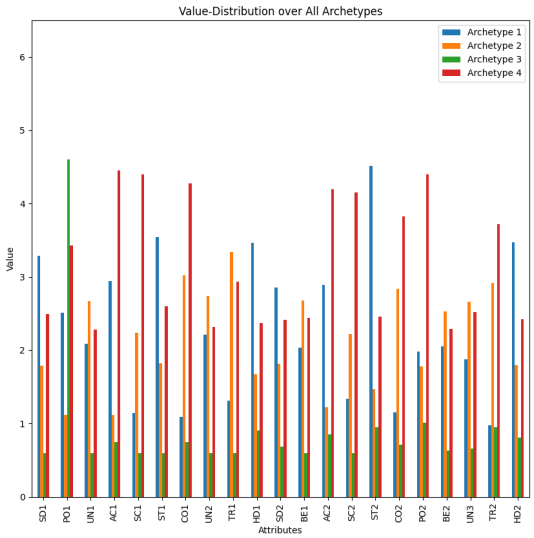
Plots a barplot of the values of each attribute of a specific archetype.



8.2.7 barplot_all

```
<instance_name>.plot(model_type: str = "CAA", plot_type = "barplot_all", title: str = "",
save_figure: bool = False, filename: str = "figure", result_number: int = 0,
with_synthetic_data: bool = False)
```

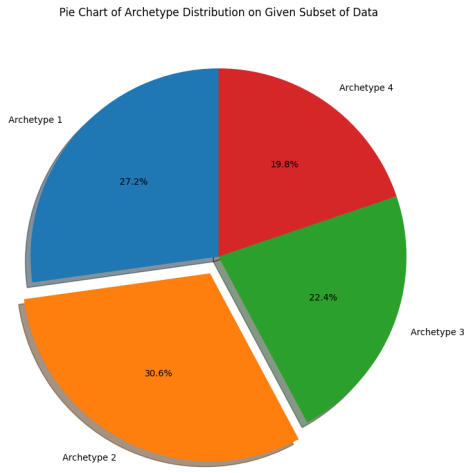
Plots a barplot of the values of each attribute of all archetypes. This plot can be unmanageable if the analysis contains many attributes and archetypes.



8.2.8 pie_chart

```
<instance_name>.plot(model_type: str = "CAA", plot_type = "pie_chart", title: str = "",
save_figure: bool = False, filename: str = "figure", result_number: int = 0, subject_indexes:
list() = [1], attribute_indexes: list(tuple()) = [], with_synthetic_data: bool = False)
```

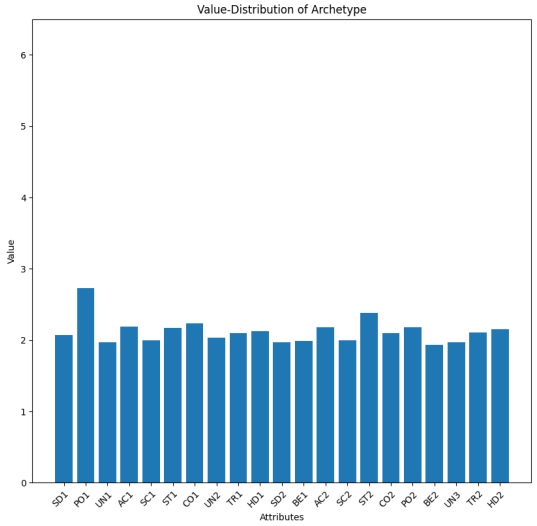
Plots a pie chart of the average distribution of archetypes of a subset of the dataset. If a dataframe (CSV file) has been loaded, the subset can be defined using the attribute_indexes parameter, else the subset can be defined using the subject_indexes parameter.



8.2.9 attribute_distribution

```
<instance_name>.plot(model_type: str = "CAA", plot_type = "attribute_distribution", title: str
= "", save_figure: bool = False, filename: str = "figure", result_number: int = 0,
subject_indexes: list() = [1], attribute_indexes: list(tuple()) = [], with_synthetic_data: bool
= False)
```

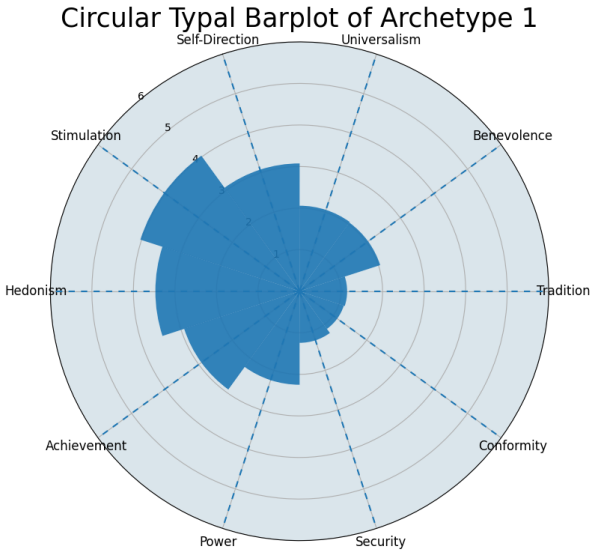
Plots a barplot of the average distribution of attributes of a subset of the dataset. If a dataframe (CSV file) has been loaded, the subset can be defined using the attribute_indexes parameter, else the subset can be defined using the subject_indexes parameter.



8.2.10 circular_typal_barplot

```
<instance_name>.plot(model_type: str = "CAA", plot_type = "circular_typal_barplot", title: str
= "", save_figure: bool = False, filename: str = "figure", result_number: int = 0, types: dict
= {}, with_synthetic_data: bool = False)
```

Plots a circular barplot of a specific archetype of specified types in the dataset.



9 Dataframes

If a CSV file has been loaded into the module and a analysis has been performed, then the module has automatically created both a Ranked Dataframe of 3 ranks and a Dataframe of linear combination values and stored them in the module.

9.1 Create Dataframe

To create a dataframe of the results and store it in your AA instance, or return it as a pandas.Datafra
run the following command:

```
<instance_name>.create_dataframe(model_type: str = "CAA", result_number: int = 0,
with_synthetic_data: bool = False, archetype_rank: int = 0, return_dataframe: bool = False)
```

The results will be appended to the existing loaded dataframe as the last columns of the dataframe. The method parameters for this method are the following:

Parameter	Description	Default Value
`model_type`	A variable of type `str` describing the type of analysis to create the dataframe from. This variable can take the following values; `"CAA"`, `"OAA"`, `"RBOAA"`.	`"CAA"`
`result_number`	A variable of type `int` describing the index of the analysis to create the dataframe from. This variable is used when multiple analysis have been performed. The analysis at index `0` corresponds to the most recent analysis.	`0`
`with_synthetic_data`	A variable of type `bool` describing whether the analysis to create the dataframe from was performed on synthetically created data.	`False`
`archetype_rank`	A variable of type `int` describing the number of ranked archeytpes which should be shown in the dataframe. If this variable is set to 0, then there will be no ranks, but insted the value of the linear combination of each archetype will be appended to each subject of the dataset.	`0`
`return_dataframe`	A variable of type `bool` describing whether the dataframe should be returned to the user as a pandas.Dataframe.	`False`

9.2 Get Dataframe

To get the dataframe element of your AA instance, run the following command:

```
<instance_name>.get_dataframe(ranked_dataframe:  bool = False)
```

The method parameters for this method are the following:

Parameter	Description	Default Value
`ranked_dataframe`	A variable of type `bool` describing whether the dataframe describing the ranks of the archetypes should be returned. If this variable is set to `False`, then the dataframe of the values of the linear combination of each archetype is returned.	`False`