[Draw your reader in with an engaging abstract. It is typically a short summary of the document. When you’re ready to add your content, just click here and start typing.]

Askia Analyse Extensions

Help document

Support Askia Team

Table of Contents

[Table of documents 6](#_Toc51164319)

[1 Introduction 7](#_Toc51164320)

[2 First steps 8](#_Toc51164321)

[2.1 Installation 8](#_Toc51164322)

[2.1.1 Via AskiaSuite installer 8](#_Toc51164323)

[2.1.2 Separately 10](#_Toc51164324)

[2.2 Set options 10](#_Toc51164325)

[2.2.1.1 askiaAnalyse Options 10](#_Toc51164326)

[2.2.1.2 R and Python .bat parameters 11](#_Toc51164327)

[2.3 As a user 12](#_Toc51164328)

[2.4 As a developer 12](#_Toc51164329)

[3 Configuration 13](#_Toc51164330)

[3.1 AnalyseExtensions directory 14](#_Toc51164331)

[3.2 Xml configuration files 15](#_Toc51164332)

[3.2.1 The header 15](#_Toc51164333)

[3.2.2 The user parameters 15](#_Toc51164334)

[3.2.3 The global parameters 16](#_Toc51164335)

[3.2.4 The command-line 16](#_Toc51164336)

[3.3 R extension 17](#_Toc51164337)

[3.3.1 Package askiaranalysis 17](#_Toc51164338)

[3.3.1.1 Aim 17](#_Toc51164339)

[3.3.1.2 Updating the package 17](#_Toc51164340)

[3.3.2 Functions 18](#_Toc51164341)

[3.3.3 Exchange data with askiaAnalyse 19](#_Toc51164342)

[3.3.3.1 getDataFromAnalyse() 19](#_Toc51164343)

[3.3.3.2 sendDataToAnalyse() 19](#_Toc51164344)

[3.3.3.3 getQuestions() 19](#_Toc51164345)

[3.3.3.4 getResponses() 19](#_Toc51164346)

[3.3.4 Cleaning data 19](#_Toc51164347)

[3.3.4.1 CleanOpenQuestion() 19](#_Toc51164348)

[3.3.4.2 CleanNumericQuestion() 19](#_Toc51164349)

[3.3.4.3 CleanSingleQuestion() 19](#_Toc51164350)

[3.3.4.4 CleanMultipleQuestion() 19](#_Toc51164351)

[3.3.4.5 CleanAnalyseData() 19](#_Toc51164352)

[3.3.4.6 Other questions 19](#_Toc51164353)

[3.3.5 Useful functions 19](#_Toc51164354)

[3.3.5.1 shortcutsManagement() 19](#_Toc51164355)

[3.3.5.2 paramsManagement() 19](#_Toc51164356)

[3.3.5.3 cat0() 19](#_Toc51164357)

[3.3.5.4 now0() 19](#_Toc51164358)

[3.3.6 R analysis 20](#_Toc51164359)

[3.3.6.1 sentimentScoring() 20](#_Toc51164360)

[3.3.6.2 MaxDiffAggDesign() 20](#_Toc51164361)

[3.3.6.3 MaxDiffAnalysis() 20](#_Toc51164362)

[3.3.6.4 turfAnalysis() 20](#_Toc51164363)

[3.3.6.5 convWide2LongFormat() 20](#_Toc51164364)

[3.3.6.6 factoExploratoryAnalysis() 20](#_Toc51164365)

[3.3.6.7 factoImputeNA() 20](#_Toc51164366)

[3.3.7 Askia analysis 20](#_Toc51164367)

[3.3.7.1 AskiaSentimentScoring() 20](#_Toc51164368)

[3.3.7.2 AskiaMaxDiffAggDesign() 20](#_Toc51164369)

[3.3.7.3 AskiaMaxDiffAnalysis() 20](#_Toc51164370)

[3.3.7.4 AskiaTurfAnalysis() 20](#_Toc51164371)

[3.3.7.5 AskiaConvWide2LongFormat() 20](#_Toc51164372)

[3.3.7.6 AskiaExploratoryAnalysis() 20](#_Toc51164373)

[3.4 Python extension 21](#_Toc51164374)

[3.4.1 The virtual environment 21](#_Toc51164375)

[3.4.2 Package askiapythonanalysis 21](#_Toc51164376)

[3.4.2.1 getDataFromAnalyse() 21](#_Toc51164377)

[3.4.2.2 sendDataToAnalyse() 21](#_Toc51164378)

[3.4.2.3 isnumeric() 21](#_Toc51164379)

[3.4.2.4 savefig2pdf() 21](#_Toc51164380)

[3.4.2.5 clusterThresholdPlot() 21](#_Toc51164381)

[3.4.2.6 optimalNumberClusters() 21](#_Toc51164382)

[3.5 Create my first analysis 22](#_Toc51164383)

[3.5.1 R analysis 22](#_Toc51164384)

[3.5.2 Python analysis 22](#_Toc51164385)

[4 Analysis 23](#_Toc51164386)

[4.1 Before starting 23](#_Toc51164387)

[4.1.1 When uses an analysis? 23](#_Toc51164388)

[4.1.2 Some important information before starting. 24](#_Toc51164389)

[4.2 [1] Sentiment scoring 25](#_Toc51164390)

[4.2.1 Create an analysis 25](#_Toc51164391)

[4.2.2 Parameters 25](#_Toc51164392)

[4.2.2.1 VarSource 26](#_Toc51164393)

[4.2.2.2 WeightedAverage 26](#_Toc51164394)

[4.2.2.3 PolarityDataName 26](#_Toc51164395)

[4.2.2.4 PolarityDataUpdate 27](#_Toc51164396)

[4.2.2.5 ValenceDataUpdate 27](#_Toc51164397)

[4.2.2.6 OutputCSV 27](#_Toc51164398)

[4.2.3 Outputs 28](#_Toc51164399)

[4.2.4 Examples of external datasets 28](#_Toc51164400)

[4.3 [2] MaxDiff aggregate design table 29](#_Toc51164401)

[4.3.1 Create an analysis 29](#_Toc51164402)

[4.3.2 Parameters 29](#_Toc51164403)

[4.3.2.1 NumberExperiments 30](#_Toc51164404)

[4.3.2.2 NumberScreens 30](#_Toc51164405)

[4.3.2.3 NumberAlternatives 30](#_Toc51164406)

[4.3.2.4 RandomSeed 30](#_Toc51164407)

[4.3.3 Outputs 30](#_Toc51164408)

[4.4 [3] MaxDiff utilities 31](#_Toc51164409)

[4.4.1 Create an analysis 31](#_Toc51164410)

[4.4.2 Parameters 33](#_Toc51164411)

[4.4.2.1 MatrixVar 33](#_Toc51164412)

[4.4.2.2 MostVar 33](#_Toc51164413)

[4.4.2.3 LeastVar 33](#_Toc51164414)

[4.4.2.4 LoopLevel 33](#_Toc51164415)

[4.4.3 Outputs 33](#_Toc51164416)

[4.5 [4] Turf analysis 34](#_Toc51164417)

[4.5.1 Create an analysis 34](#_Toc51164418)

[4.5.2 Parameters 34](#_Toc51164419)

[4.5.2.1 VarSource 35](#_Toc51164420)

[4.5.2.2 MinCategories / MaxCategories 35](#_Toc51164421)

[4.5.2.3 VarWeighted 35](#_Toc51164422)

[4.5.3 Outputs 35](#_Toc51164423)

[4.6 [5] Convert data from wide to long format 36](#_Toc51164424)

[4.6.1 Create an analysis 36](#_Toc51164425)

[4.6.2 Parameters 36](#_Toc51164426)

[4.6.2.1 IdVar 37](#_Toc51164427)

[4.6.2.2 OtherVars 37](#_Toc51164428)

[4.6.2.3 VariablesName 37](#_Toc51164429)

[4.6.2.4 ValuesName 37](#_Toc51164430)

[4.6.3 Outputs 37](#_Toc51164431)

[4.7 [6] Exploratory analysis 38](#_Toc51164432)

[4.7.1 Create an analysis 39](#_Toc51164433)

[4.7.2 Parameters 40](#_Toc51164434)

[4.7.2.1 GroupVar1 – GroupVar9 40](#_Toc51164435)

[4.7.2.2 GroupNames 41](#_Toc51164436)

[4.7.2.3 SuppVar 41](#_Toc51164437)

[4.7.2.4 IdVar 41](#_Toc51164438)

[4.7.2.5 RowWeight 41](#_Toc51164439)

[4.7.2.6 CatLabels 42](#_Toc51164440)

[4.7.2.7 VarianceParams 42](#_Toc51164441)

[4.7.2.8 OtherParams 42](#_Toc51164442)

[4.7.2.9 OutputTables and OutputPlots 43](#_Toc51164443)

[4.7.3 Outputs 44](#_Toc51164444)

[4.7.4 Bad practices 44](#_Toc51164445)

[4.8 [7] Clustering (after PCA) 46](#_Toc51164446)

[4.8.1 Create an analysis 46](#_Toc51164447)

[4.8.2 Parameters 47](#_Toc51164448)

[4.8.2.1 LoopQuestion and ResultQuestion 47](#_Toc51164449)

[4.8.2.2 nComponents 47](#_Toc51164450)

[4.8.2.3 nClustersMin and nClustersMax 48](#_Toc51164451)

[4.8.3 Outputs 48](#_Toc51164452)

[4.8.4 How to improve my classifier? 48](#_Toc51164453)

[5 To update 49](#_Toc51164454)

[References 50](#_Toc51164455)

# Table of documents

[askiaAnalyse extentions 1 - Run an analysis with the extentions 12](#_Toc51157981)

[askiaAnalyse extentions 2 - Sentiment scoring 24](#_Toc51157982)

[askiaAnalyse extentions 3 - MaxDiff aggregate design table 28](#_Toc51157983)

[askiaAnalyse extentions 4 - MaxDiff utilities: loop/level calculated question 31](#_Toc51157984)

[askiaAnalyse extentions 5 - Maxdiff utilities: numeric calculated question 31](#_Toc51157985)

[askiaAnalyse extentions 6 - Turf analysis 33](#_Toc51157986)

[askiaAnalyse extentions 7 - Convert wide to long format 35](#_Toc51157987)

[askiaAnalyse extentions 8 - Exploratory analysis: loop/level calculated question 38](#_Toc51157988)

[askiaAnalyse extentions 9 - Exploratory analysis: numeric calculated question 39](#_Toc51157989)

[askiaAnalyse extentions 10 - Clustering: closed and numeric calculated questions 46](#_Toc51157990)

[Figure 1 - Askia Analyse Extensions global overview 13](#_Toc51157991)

[Figure 2 - AnalyseExtensions directory tree 14](#_Toc51157992)

[Screenshot 1 - AskiaSuite installer: Custom setup 8](#_Toc51157993)

[Screenshot 2 - AskiaSuite installer: Analyse R extensions features 9](#_Toc51157994)

[Screenshot 3 - Askia Suite installer: R and OpenJDK licences 9](#_Toc51157995)

[Screenshot 4 - AnalyseExtensions directory 10](#_Toc51157996)

[Screenshot 5 - askiaAnalyse Options for the extensions 11](#_Toc51157997)

[Table 1 - xml global parameters 16](#_Toc51157998)

[Table 2 - Exploratory analysis: output's availability 42](#_Toc51157999)

[Table 3 - Silhouette score interpretation [7] 45](#_Toc51158000)

# Introduction

Same as <https://blog.askia.com/askiaanalysis/analyse-5-5-2-jewels-in-the-crown-pt-1/> ?

# First steps

To be able to use the extensions and all the analysis – **in addition to askiaAnalyse** – you need:

* **AnalyseExtensions** directory: all the analysis scripts and miscellaneous files are inside
* **R**: for R analyses
* **Java**: used by R for exporting data into Microsoft Excel format
* **Python**: for Python analyses

NB: you don’t need to have Python if you plan to use only R analyses, same as you don’t need R (and Java) if you plan to use only Python analyses.

## Installation

You can install everything in one go via AskiaSuite installer or one by one separately (e.g. if you need some updates):

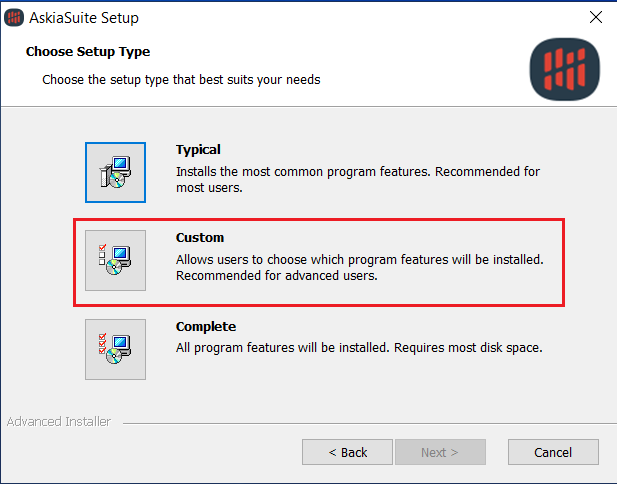
### Via AskiaSuite installer

The installer of AskiaSuite can be found here <https://installers.askia.com/helpdesk/askiasuite/5.5.3/>.

The current version is **5.5.03.05**.

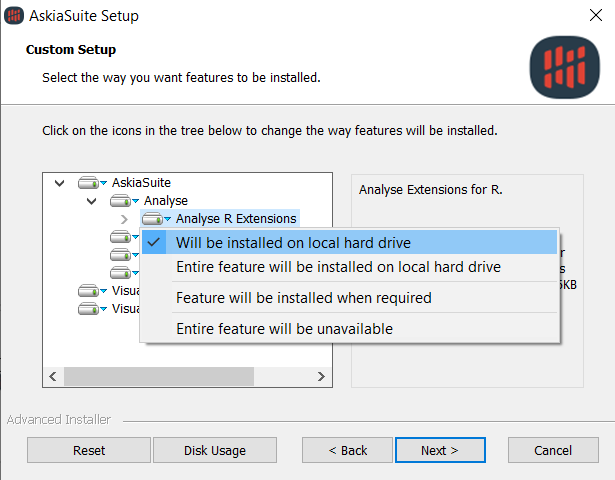
/!\ Note that this version doesn’t include an installer for Python.

From **Choose Setup Type** window, choose **Custom** (or **Complete**) to add extensions documents:



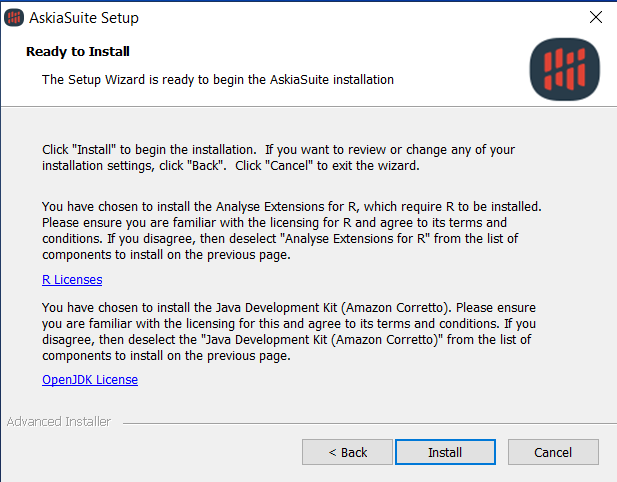
Screenshot 1 - AskiaSuite installer: Custom setup

From **Custom Setup** window, extend AskiaSuite>Analyse>Analyse R Extensions and be sure to select “**Will be installed on local drive**”:



Screenshot 2 - AskiaSuite installer: Analyse R extensions features

From **Ready to Install** window, be aware of both R and OpenJDK (for Java) licences:



Screenshot 3 - Askia Suite installer: R and OpenJDK licences

### Separately

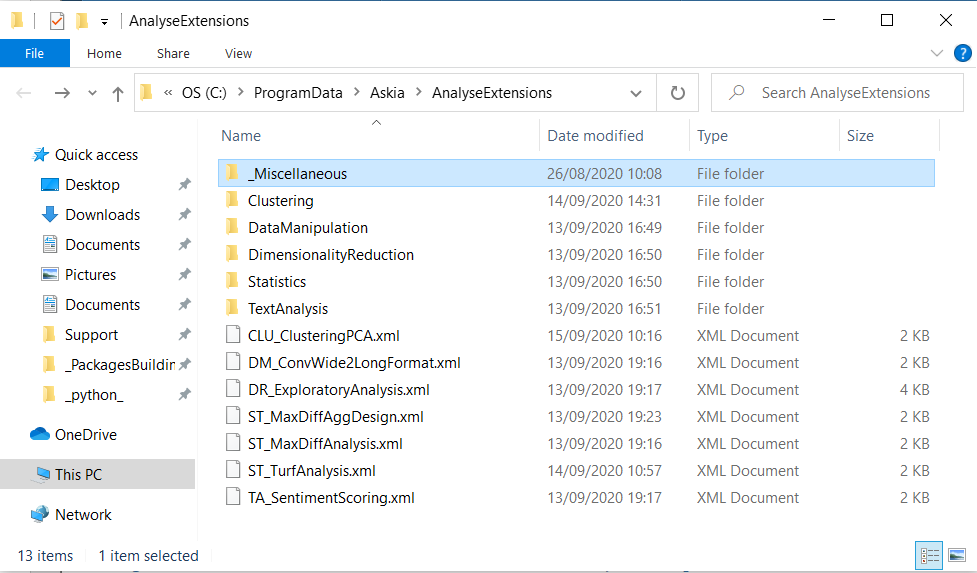
The **AnalyseExtensions directory** can be download from Askia GitHub: ...

**R version 3.6.3** is needed.

Any version of Java is suitable although **OpenJDK** is recommended.

**Python version 3.8.0** is needed.

Here’s AskiaExtensions directory after installing (from *C:\ProgramData\Askia\AnalyseExtensions*):



Screenshot 4 - AnalyseExtensions directory

## Set options

Now that you have installed the extensions features and before starting using the extensions, you’ll need to set some options which are: options into askiaAnalyse, parameters into required\_packages.bat (for R) and required\_modules.bat (for Python).

#### askiaAnalyse Options

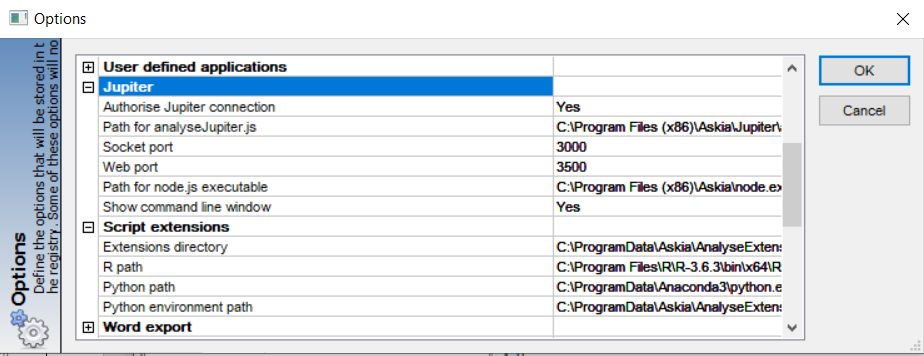
From askiaAnalyse > Tools > Options:

**Jupiter:**

* Authorise Jupiter connection: *Yes*
* Path for analyseJupiter.js: *C:\Program Files (x86)\Askia\Jupiter\app\jupiterAnalyse.js*
* Socket port: *3000*
* Web port: *3500*
* Path for node.js executable: *C:\Program Files (x86)\Askia\node.exe*
* Show command line window: *No* (this is more for debug, if you need to you can set its value to *Yes*)

**Script extensions:**

* Extensions directory: path to AnalyseExtensions directory
  + by default: *C:\ProgramData\Askia\AnalyseExtensions*
  + /!\ no “\” at the end
* R path: path to Rscript executable
  + by default: *C:\Program Files\R\R-3.6.3\bin\x64\Rscript.exe*
* Python path: path to python executable
  + by default: *C:\Program Files\Python38\python.exe*
* Python environment path: path to a folder where a virtual environment for python will be created
  + this is a folder you can create wherever you want, be aware that python modules will be installed inside it (so it has to be clean/empty)
  + by default: *C:\ProgramData\Askia\AnalyseExtensions\\_Miscellaneous\Python\_files\pythonvirtualenv*
  + /!\ no “\” at the end



Screenshot 5 - askiaAnalyse Options for the extensions

#### R and Python .bat parameters

For R and Python what you need is to install the packages (R) and modules (Python) needed for the analyses. There already are scripts doing that. What you need is to use the same values as the ones used from askiaAnalyse Options into the programs (.bat files) that’ll install all needs for R and/or Python extensions.

Install all R packages:

* from **\_Miscellaneous > R\_files > required\_packages.bat**:
* Rexe: same value as *[R path]*
* Rscript: path to the R script “required\_packages.R” (the one with the packages to install and also install them)
* Rpackages: path to the repertory with the local packages (*askiaranalysis\_1.1.0.tar.gz* and others)

Create a Python virtual environment and install all modules:

* from **\_Miscellaneous > Python\_files > required\_modules.bat**:
* pythonExe: same value as *[Python path]*
* venvDir: same value as *[Python environment path]*
* modulesTxt: path to the text file “required\_modules.txt” (the one with the modules to install)

You can launch those programs after having updated them.

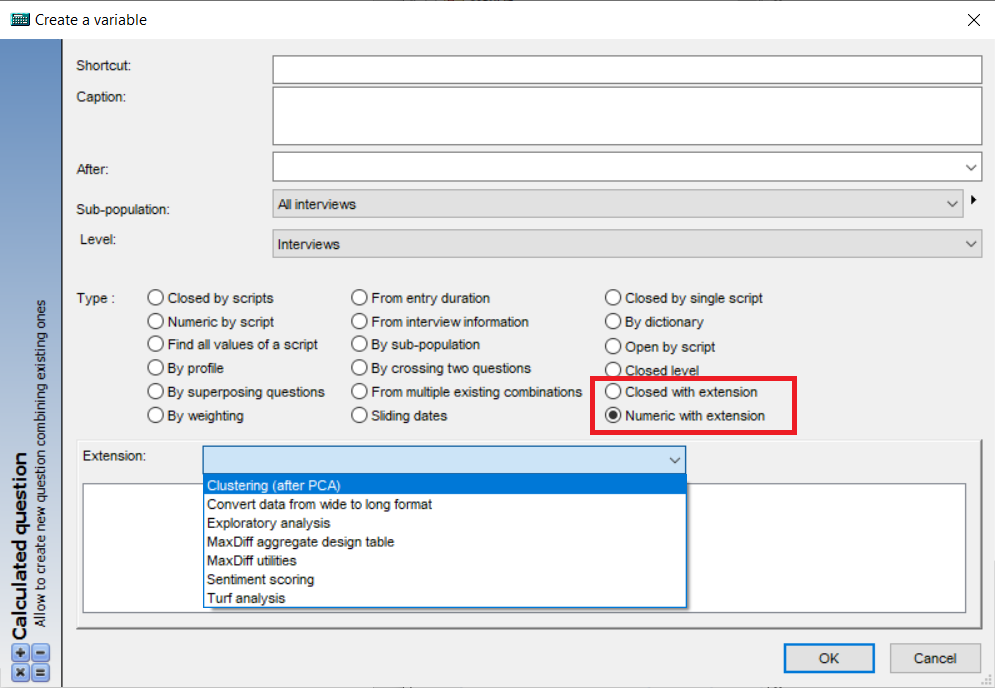
## As a user

You’re a user if your aim is only to use the available analyses from the extensions.

Now that everything has been set up correctly, you can start using the extensions and its analyses.

Open your QES with askiaAnalyse:

* from menu Edit > Create a variable…
* from Type: you can choose “Closed with extension” or “Numeric with extension”
* from Extension: choose the analysis you want to run



askiaAnalyse extentions 1 - Run an analysis with the extentions

These analyses are explained more in details in Analysis section.

## As a developer

You’re a developer if you want to update any analyses or to create a new one.

See the Configuration section to get more information about how the extensions work and how to create/update.

# Configuration

We’re going to explain here all-important documents, which are: AnalyseExtensions directory, the xml configuration files and the Miscellaneous files for R and Python.

Basically, how the extensions work:

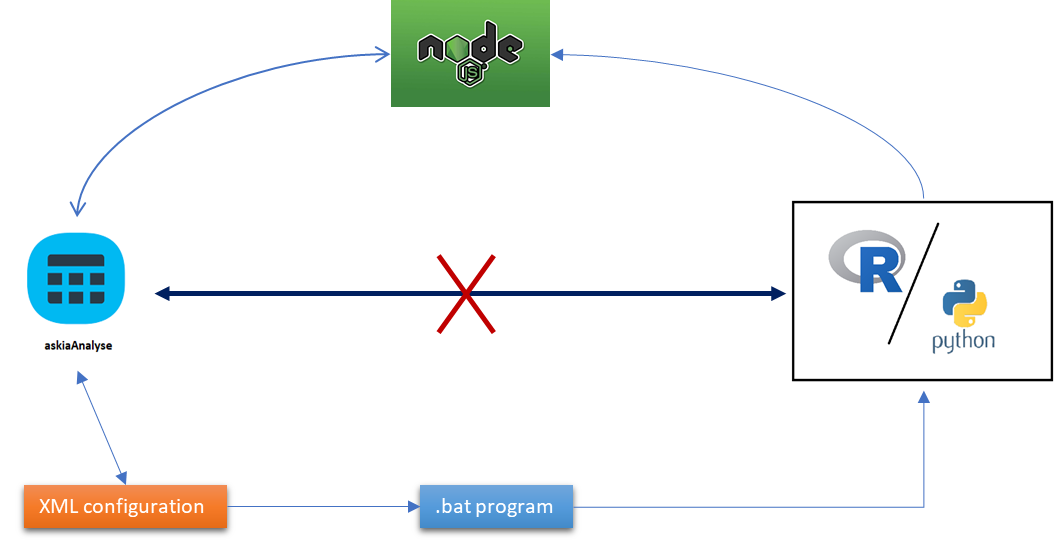


Figure 1 - Askia Analyse Extensions global overview

What we want is to be able to use R and Python from askiaAnalyse which is actually not possible in one step!

The solution is to use other features to be able to do it:

* The xml configuration file
  + is used to set-up the analysis and its parameters,
  + and create the .bat program with then parameter’s values
* The .bat program file
  + will launch R/Python with their analysis’s scripts
  + and the parameter’s values
* From R/Python
  + run the script analysis with the parameter’s values
  + and go through NodeJS to get (send) data from (to) askiaAnalyse

Before starting some important vocabularies:

* A **Python module** is a Python script with Python functions
* A **Python package** is a folder with Python modules
* A **R package** is an archive with R functions (in reality it’s way more than that, but it’s just for sum-up)

The R (and Python) package(s) contain the functions needed by R (and Python) extensions for running the analyses.

## AnalyseExtensions directory

Here’s an overall view of AnalyseExtensions directory’s content:

folder

file

Legend:

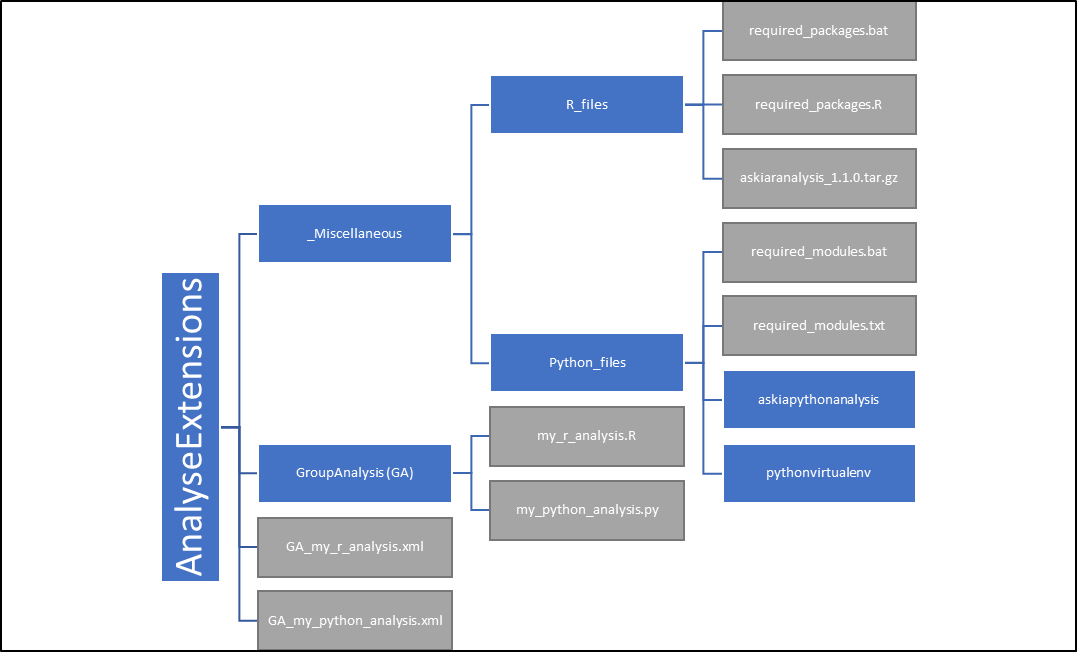


Figure 2 - AnalyseExtensions directory tree

To the root we have:

* Analyses xml files: define here the parameters needed for the analyses
* A group of analysis methods: will contain the R/Python scripts with the analyses
* \_Miscellaneous: miscellaneous files
  + R\_files contains:
    - the program (.bat) and the R script needed for install all R packages
    - *askiaranalysis\_1.1.0.tar.gz*: R package with the analyses
    - all other local packages
  + Python\_files contains:
    - the program (.bat) and the text file (with the modules) needed in order to create the Python virtual environment and install all the modules
    - askiapythonanalysis: Python package (with Python modules with the analyses)

The group of analyses available so far are: text analysis (TA), statistics (ST), data manipulation (DM), dimensionality reduction (DR) and clustering (CLU).

## Xml configuration files

The .xml are the files that being read by askiaAnalyse in order to create an analysis. Then, any analysis available in the extensions has been defined into the xml. You’ll need as many xml as analyses.

In a xml you need to define (in the same order):

* 1-The header, 2-The command-line, and 3-The user parameters

### The header

In the header you need first to your file as an xml and then you can define the name of the analysis and which program it’ll run.

* <?xml version=”1.0”, encoding=”Unicode” ?>
* <Process Name=””>
  + If R: <AppUsed ID=”1” Name=”R” StartCommand=”Rscript”/>
  + If Python: <AppUsed ID=”2” Name=”Python” StartCommand=”python”/>
  + The command-line section
  + The user parameters section
* </Process>

### The user parameters

A user parameter is declared into *<UserParameters>* and *<Parameter>* tags like that:

* *<UserParameters>*
  + *<Parameter>…</Parameter>*
    - … properties parameter 1
  + *<Parameter>…</Parameter>*
    - … properties parameter 2
* *</UserParameters>*

Here’s the list of the property’s tags needed to create a user parameter (“…” for where to put your value):

* *<Type>…</Type>*: define the type of the parameter
  + question or questions: allows to select 1 question or many questions
  + numeric or string: allows to use a numeric or a string value
* *<Constraints … />*: use a constraint (only for question/questions)
  + Open=”1”, Numeric=”1”, Date=”1”
  + Closed=”1”, Single=”1”, Scaled=”1”, Multiple=”1”
* *<DefValue>…</DefValue>*: define a default value
* *<Description>…<Description>*: the description of the parameter (text shown into askiaAnalyse)
* *<Name>…</Name>*: parameter’s name to use into the command-line

**You have to respect the same order as above**. Also Type, Description and Name are required.

NB1: Closed refers to Single, Scaled and Multiple questions.

NB2: to use many constraints, for example you can use *<Constraints Numeric=”1” Open=”1”/>* (but Type’s value must be **questions**).

### The global parameters

In addition to the user parameters, there are about ten global parameters that can be used into the command-line:

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **askiaAnalyse Options** | **Description** |
| 1 | CurrentQuestion |  | shortcut of the current calculated question |
| 2 | CurrentResponse |  |  |
| 3 | CurrentLevel |  |  |
| 4 | CurrentFilter |  |  |
| 5 | CurrentWeighting |  |  |
| 6 | StartURL |  |  |
| 7 | InputDir |  |  |
| 8 | OutputDir |  | path to the QES.dat folder directory |
| 9 | ScriptPath |  |  |
| 10 | AnalyseExtensionsPath | Extensions directory | path to AnalyseExtensions directory |
| 11 | JupiterHost | Web port | [http://localhost:[Web port]/](http://localhost:[Web%20port]/) |
| ?? | PythonEnv | Python environment path | path to python virtual environment directory |

Table 1 - xml global parameters

### The command-line

This is the most important part. You’re declaring here what’s going to be transcribed into the .bat program file (see Figure 1). The main code is declared into *<CommandLine>* tags with the possibility to add some commands before/after it.

Then you need in this order:

* *<PreCommandLine>…<PreCommandLine>*: command to do before running the analysis
* *<CommandLine>…<CommandLine>*: main code for running the analysis
* *<PostCommandLine>…<PostCommandLine>*: command to do after running the analysis

Into those tags you can use user and global parameters like that: << “??MyParameter??” >>.

By using *<CommandLine>*, the first command will be the value of *[R path]* or *[Python path]* (see section askiaAnalyse Options) depends on which program you’ve defined in *<AppUsed>* (see section The header).

With that the program will only launch R or Python software (in command-line). Then you need to add everything else you need (user and global parameters, R/Python analysis script, …) in order to run your analysis.

Example:

* *<CommandLine> “my\_r\_analysis.r” “??Param1??” “??Param2??” </CommandLine>*
* Will create a .bat file with:
  + *Rscript.exe “my\_r\_analysis.r” “value\_Param1” “value\_Param2”*

NB1: See [1] and [2] about how to run an R or Python script from the command-line.

NB2: inside a command-line tag, you can use only 1 command and it has to be on the same line. Yet you can use << & >> to allow the use of 2 commands in the same line. And can also use << ^ >> to separate the same command into many lines.

NB3: in order to use the Python virtual environment, you need to “activate” it before even launch Python itself (from *<CommandLine>*). To solve it use *<PreCommandLine>* tag to activate it.

## R extension

R extension allows askiaAnalyse to use the powerful R language in order to run analyses. What you need to run an analysis using R extension is to have the R script with the analysis into a group of analysis and called it from the xml.

Also, since some analyses can be quite heavy and may require dependencies from other packages, we decided to organise our work as a package in order to have a clean and stable R environment.

In that way, the R script with the analysis will only:

* load the package
* get xml parameter’s values from the command-line
* run the analysis by using the function from the package with the parameter’s values

### Package askiaranalysis

**The current version of askiaranalysis is 1.1.0.**

#### Aim

The package *askiaranalysis* has been developed for the extensions in order to:

1. Firstly, to provide R functions that allow R to communicate with askiaAnalyse. Hence, from R we can get/read data from or send/write data to askiaAnalyse.
2. Since the data from askiaAnalyse are not always in R format, so *askiaranalysis* also provide R functions that allow to clean/convert data from askiaAnalyse format to R format.
3. And also to provide also many analysis functions which need data in R format.

And by putting those 3 aims in common, we have the main aim of *askiaranalysis* package which is to provide – for each analysis – an R function that allows – in one go – to:

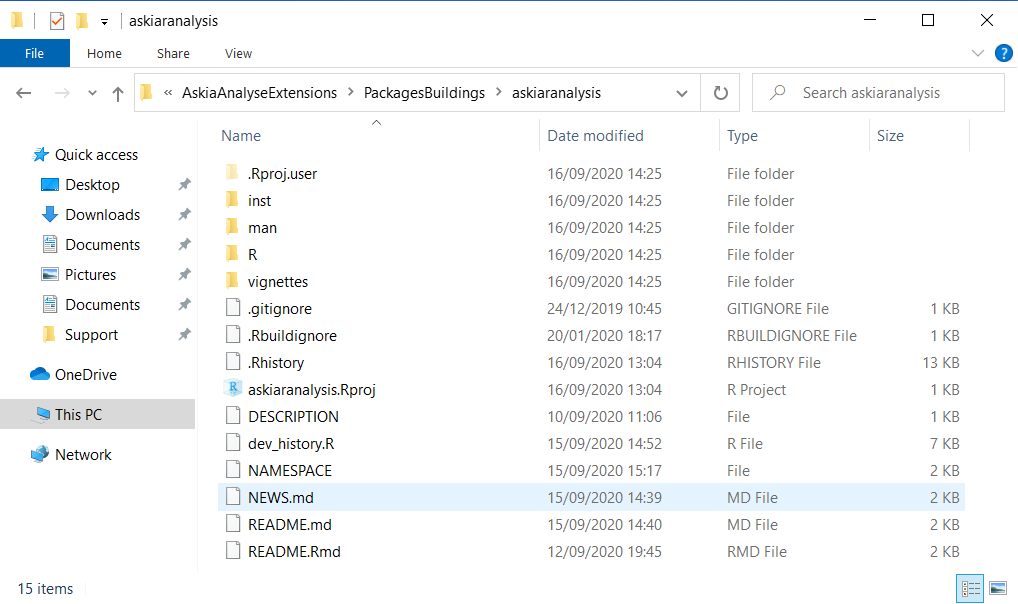
* 1-get data from askiaAnalyse,
* 2-convert those data to R format,
* 3-run an analysis,
* 4-send back the analysis result to askiaAnalyse.

#### Updating the package

This package has been created by following Think-R’s procedures (see [3] and [4]). So what you need to update it are RStudio+Rtools and askiaranalysis directory (contains all the documents for building *askiaranalysis* package).

In **askiaranalysis > R**: there are all the R scripts with the functions uses for the analyses. Which are using a specific naming:

* A[n]\_xxx.R : for askia analysis functions
* B[n]\_xxx.R : for r analysis functions (per group of analyses)
  + B1\_xxx.R for TA, B2\_xxx.R for ST, B3\_xxx.R for DM and B4\_xxx.R for DR
* C[n]\_xxx.R : for exchange data with askiaAnalyse functions
* D[n]\_xxx.R : for cleaning data functions
  + D[n]\_xxx.R for each type of question and D1\_xxx.R for global cleaning
* E[n]\_xxx.R : other useful functions



Screenshot 6 - askiaranalysis directory

### Functions

As seen above, *askiaranalysis*’s functions are categorized like that:

* Exchange data with askiaAnalyse functions
* Cleaning data functions
* R analysis functions
* Useful functions
* Askia analysis functions: these functions are basically using all the functions up and works like that:
  + getting the parameters from the xml/command line and clean them
  + getting the data from askiaAnalyse (exchange data)
  + clean those data (cleaning data)
  + use the cleaned data to run the analysis (r analysis)
  + send the result to askiaAnalyse (exchange data)
  + export some other outputs

Note that only Askia analysis functions will be explained more in details. For all the other functions, you can use the help from R or help website. To open the help about the functions by opening the help website:

* from R: *askiaranalysis::open\_pkgdown()*
* from local: <file:///C:/Users/Admin/Documents/R/win-library/3.6/askiaranalysis/docs/index.html>

Exported and non-exported functions: by creating a function from into an r package, you can decide if you want to export that function or not. Only exported functions have help and are available after loading the package. In order to use a non-exported function you must use this special syntax (from R): *mypackage:::my\_non\_exported\_function()*.

In fact, for *askiaranalysis* package only the important functions were exported, the others (like useful functions) weren’t, but they’re still accessible by using the syntax above.

### Exchange data with askiaAnalyse

Here are the functions that allow R to communicate with askiaAnalyse. To sum-up:

* Get data from askiaAnalyse
* Send data to askiaAnalyse
* Get information about questions (name, type, …)
* Get information about a closed question’s responses (entryCode, caption, …)

#### getDataFromAnalyse()

#### sendDataToAnalyse()

#### getQuestions()

#### getResponses()

### Cleaning data

Here are the functions that going to clean/convert the data in askiaAnalyse format to R format. To sum-up:

* Clean open, numeric, single and multiple questions
* A global function that called the above functions in order to clean a whole dataset with different type of question

NB: so far, (with *askiaranalysis* v1.1.0), date and scaled questions don’t have cleaning function/method, so they shouldn’t be used for any analysis. Still, since date are considered as numeric and scaled as single, they can be treated like that even though that’s not the best way.

#### CleanOpenQuestion()

#### CleanNumericQuestion()

#### CleanSingleQuestion()

#### CleanMultipleQuestion()

#### CleanAnalyseData()

#### Other questions

### Useful functions

These are little – non-exported – functions that allows to:

* Clean a parameter with question’s shortcuts
* Get a parameter’s name and value specify into a string
* Print a message into the console window
* Get the current date-time

#### shortcutsManagement()

#### paramsManagement()

#### cat0()

#### now0()

### R analysis

Here are the analysis functions that need data in R format.

#### sentimentScoring()

#### MaxDiffAggDesign()

#### MaxDiffAnalysis()

#### turfAnalysis()

#### convWide2LongFormat()

#### factoExploratoryAnalysis()

#### factoImputeNA()

### Askia analysis

Here are the askia analyses functions. They are run directly from the extensions. You’ll get more details about them on Analysis part. Note that one askia analysis function can use/need many r analyses functions.

#### AskiaSentimentScoring()

#### AskiaMaxDiffAggDesign()

#### AskiaMaxDiffAnalysis()

#### AskiaTurfAnalysis()

#### AskiaConvWide2LongFormat()

#### AskiaExploratoryAnalysis()

## Python extension

Python extension allows askiaAnalyse to use the powerful Python language in order to run analyses. What you need to run an analysis using Python extension is to have the Python script with the analysis into a group of analysis and called it from the xml.

In order to have a clean and stable Python environment, the best way is to create a virtual environment where all modules required will be installed.

Note that all user functions have to be created into a module into *askiapythonanalysis* package.

In that way, the Python script with the analysis will only:

* load the package
* get xml parameter’s values from the command-line
* run the analysis by using the function from the package and the parameter’s values

### The virtual environment

Before lunch Python and run the analysis, we need to activate the virtual environment. That can be done with that command:

* *CALL "??PythonEnv??\Scripts\activate.bat"*
* Remind: *[PythonEnv]* is the global parameter refers to *[Python environment parth]* askiaAnalyse option.

### Package askiapythonanalysis

Into **\_Miscellaneous > Python\_files > askiapythonanalysis** are the modules (Python scripts) used for Python analyses. There are 3 of them (so far):

* Utils\_exchangeDataWithAnalyse: functions exchange data with askiaAnalyse
* Utils\_everyone: some useful functions
* Utils\_clustering: functions specific to clustering analyses

Here are the functions. More information are available through the scripts or by using Python help.

So far there’s only 1 analysis run with Python (see [7] Clustering (after PCA)). Note that unlike R, there is no functions for Python nor askia analyses. The whole analysis has been done in one and only script (**Clustering > AskiaClusteringPCA.py**) which is not the best.

#### getDataFromAnalyse()

#### sendDataToAnalyse()

#### isnumeric()

#### savefig2pdf()

#### clusterThresholdPlot()

#### optimalNumberClusters()

## Create my first analysis

Example of quick analysis.

### R analysis

### Python analysis

# Analysis

Here are all the analyses available to run from the extensions. So far, there are 7 of them:

1. Sentiment scoring (TA)
2. MaxDiff aggregate design (ST)
3. MaxDiff utilities (ST)
4. Turf analysis (ST)
5. Convert data from wide to long format (DM)
6. Exploratory analysis (DR)
7. Clustering (CLU)

And they each belong to one of these group of methods:

* TA : text analysis methods
* ST : statistics or survey questioning methods
* DM : data manipulation or data handling methods
* DR : dimensionality reduction methods
* CLU: clustering methods

Let’s see when and how you can run an analysis.

## Before starting

Before starting let’s have/remind some useful information.

### When uses an analysis?

Here are some general contexts about the need to use one of the analyses.

* You have open questions and don’t know what to do with them?
* Or you want to know the global opinion of your respondents?

The sentiment scoring analysis will allow you to quantify the affective states from the text into an open question.

* Use of MaxDiff?

Then MaxDiff analysis to get the result.

* You have multiple questions and don’t know what to with them?
* You want to measure the market reached of some products?
* You have opinions in some products and want to know which are the most liked?

Go see what turf analysis can do.

* You have your data into a QES but for some reason you need to have them in long format?

Then the conversion from wide to long format can do that for you.

* You have a huge questionnaire with many questions and you don’t exactly know where to start your analyses?
* You need to run a clustering but you have many closed questions and don’t know what to do?
* Or you just want to have an overview of your dataset?

The exploratory analysis can do all of that and even more.

* You have run a PCA an don’t know what’s next?
* You want to group your interviews to know which ones are the closer?

The cluster analysis includes some powerful methods able to do that.

### Some important information before starting.

You need to pay attention about the **distinction between questions (in askiaAnalyse) and variables (in R)**. A question can be a variable (e.g. numeric question) or not (e.g. multiple question is will have represents as many variables as it has responses). In fact, “question” will be used to refer a column in askiaAnalyse side and “variable” in R side.

Remind that date and scaled questions don’t have cleaning function/method so far, but can still being used into an analysis as numeric for date and single for scaled.

## [1] Sentiment scoring

This is a text analysis method, and its aim is to study and quantify the affective states and subjective information from an open question. The result is number value called sentiment score (or sentiment polarity) which can be positive, negative or zero (e.g. when someone has neutral opinion).

To calculate the sentiment score we need **polarity** or **valence shifter** information:

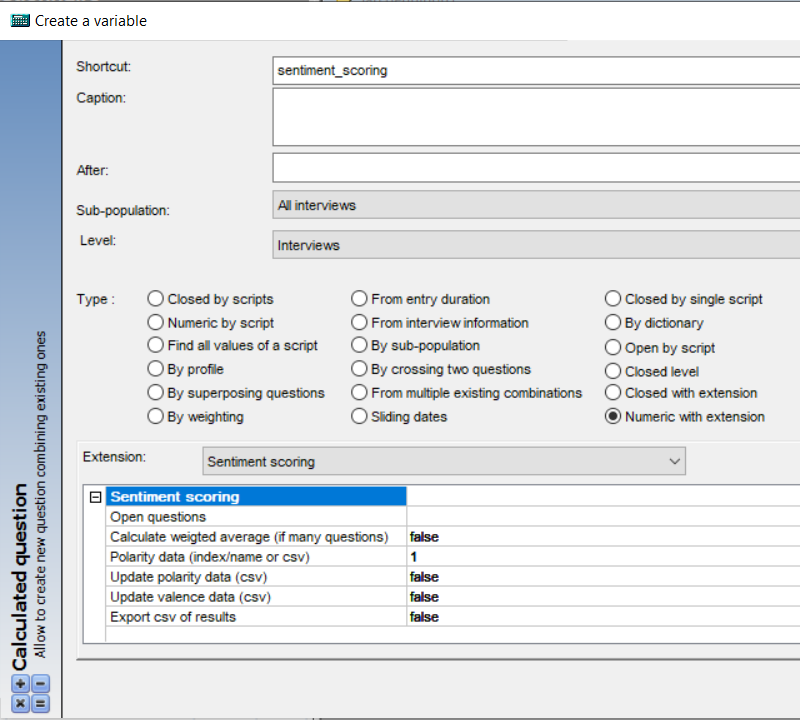
* Polarity is the score assigned to a word and can be positive (>0), negative (<0) or neutral (=0). A word with a polarity value is also called a **polarized word**. A polarity data is a dataset with words and their polarity values.
* Valence shifters are words that alter or intensify the meaning (the score) of polarized words. There is 4 kind of valence shifter (called **key**): **negator** (1), **amplifier** (2), **de-amplifier** (3) and **adversative conjunction** (4). A valence shifter data is a dataset with words and their key (its numeric value).

Example: let’s assume that the text << this is wonderful >> has 0.30 as sentiment score, then:

* << this is *not* wonderful >> will get -0.30, because “not” is a negator
* << this is *so* wonderful >> will get 0.50, because “so” is an amplifier

### Create an analysis

This is a **“Numeric with extension”** analysis and works with **only open questions (1 or many)**:



askiaAnalyse extentions 2 - Sentiment scoring

### Parameters

To run the analysis, we need to set the value of **6 parameters**, here’s their names and descriptions:

* VarSource: Open questions
* WeightedAverage: Calculate weighted average (if many questions)
* PolarityDataName: Polarity data (index/name or csv)
* PolarityDataUpdate: Update polarity data (csv)
* ValenceDataUpdate: Update valence shifters data (csv)
* OutputCSV: Export csv of results

#### VarSource

* Open questions

Select the open questions.

**Values possible:** use the … to select your open questions.

#### WeightedAverage

* Calculate weighted average (if many questions)

When you have selected more than 1 open question, you’ll get a sentiment score for each question. Hence the final sentiment score will be the mean between the scores of all the selected questions. There is 2 kind of average:

* Weighted: the score of each question is weighted with its number of words
* Arithmetic: the scores are not weighted (all have the same weight)

**Values possible:** you can use << **true** >>, << **yes** >>, or << **1** >> if you want to use the weighted average. Any other value means the use of the arithmetic average.

#### PolarityDataName

* Polarity data (index/name or csv)

As see below, a polarity data is a dataset with basically 2 columns: the 1st one with words and the 2nd with their polarity values. There are countless polarity data available on internet but actually only 15 are included in *askiaranalysis* package (called internal polarity data), and then available for this analysis.

List of internal polarity data:

* index: name
* 1: lexicon\_hash\_sentiment\_jockers\_rinker (default)
* 2: lexicon\_hash\_sentiment\_jockers
* 3: lexicon\_emojis\_sentiment
* 4: lexicon\_hash\_sentiment\_emojis
* 5: lexicon\_hash\_sentiment\_huliu
* 6: lexicon\_hash\_sentiment\_loughran\_mcdonald
* 7: lexicon\_hash\_sentiment\_nrc
* 8: lexicon\_hash\_sentiment\_senticnet
* 9: lexicon\_hash\_sentiment\_sentiword
* 10: lexicon\_hash\_sentiment\_slangsd
* 11: lexicon\_hash\_sentiment\_socal\_google
* 12: syuzhet\_nrc
* 13: syuzhet\_bing
* 14: syuzhet\_afinn
* 15: syuzhet\_syuzhet\_dict

The difference between these datasets is that they may not have the same words, or not have the same meaning/polarity for the same word, so the sentiment score will defer according to selected polarity data. You can get more information about them from R or internet.

You can also use you own polarity data (external polarity data) by specify the path to a csv file (with “word” and “polarity” as column names).

**Values possible:** as value for this parameter, you can use the index (<< **1** >>, << **2** >>, …) or the name (<< **lexicon\_hash\_sentiment\_jockers\_rinker** >>, …) of the polarity data to use one of the internal. Or the path to a csv to use an external polarity data (e.g. << **C:\my\_files\polarity\_dt.csv** >>). By default, the 1st internal polarity data is the one used.

#### PolarityDataUpdate

* Update polarity data (csv)

An updating polarity data is a polarity data used to update the main polarity data (see parameter *[PolarityDataName]*). It’s a csv file with “word” and “polarity” as column names.

Using an updating polarity data is useful when you want to use an internal polarity data but need to change the polarity values for some specific words.

For example, by using the 1st internal polarity data you’ll get 0.75 as polarity for “love”. If you want to use another polarity for that word, you need to specify the word and its new polarity on the csv file.

**Values possible:** path to the csv file (e.g. << **C:\my\_files\polarity\_update\_df.csv** >>). Any other value means no using an updating polarity data.

#### ValenceDataUpdate

* Update valence shifters data (csv)

Note that there is only 1 internal valence shifters data (*lexicon\_hash\_valence\_shifters*) and it is used by default.

As for updating internal polarity data by using a csv, you can also use a csv to update the internal valence shifters data. The csv file needs to have “word” and “polarity” as column names. However, on polarity column you’ll have to use the key to specify which type of valence shifter the word is.

Type of valence shifters:

* Key: type of valence shifter
* 1: negator
* 2: amplifier
* 3: de-amplifier
* 4: adversative conjunction

**Values possible:** path to the csv file (e.g. << **C:\my\_files\valence\_update\_df.csv** >>). Any other value means no using an updating polarity data.

#### OutputCSV

* Export csv of results

If you also want to export the sentiment score on csv file.

**Values possible:** you can use << **true** >>, << **yes** >>, or << **1** >> to export and any other value means no export.

### Outputs

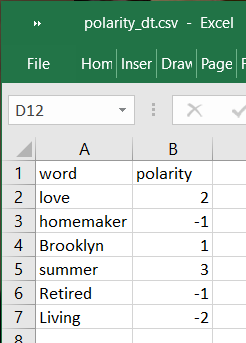
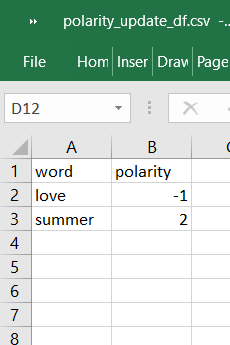
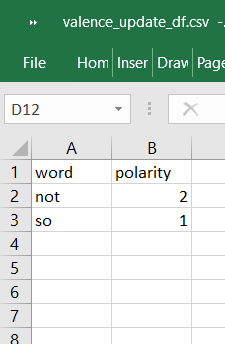
The results are sent back to askiaAnalyse into the numeric calculated question.

You can also have a csv file (into the .dat folder) if asked. Its name will be << Export\_SentimentScore\_[datetoday].csv >>.

### Examples of external datasets

Examples of external polarity and valence shifters data as csv files:

* Polarity data, updating polarity data, updating shifters data

## [2] MaxDiff aggregate design table

This is one of MaxDiff analysis (explained more in details on the next part). Its aim is to create an aggregate design which is used into askiaDesign when programming a MaxDiff questionnaire.

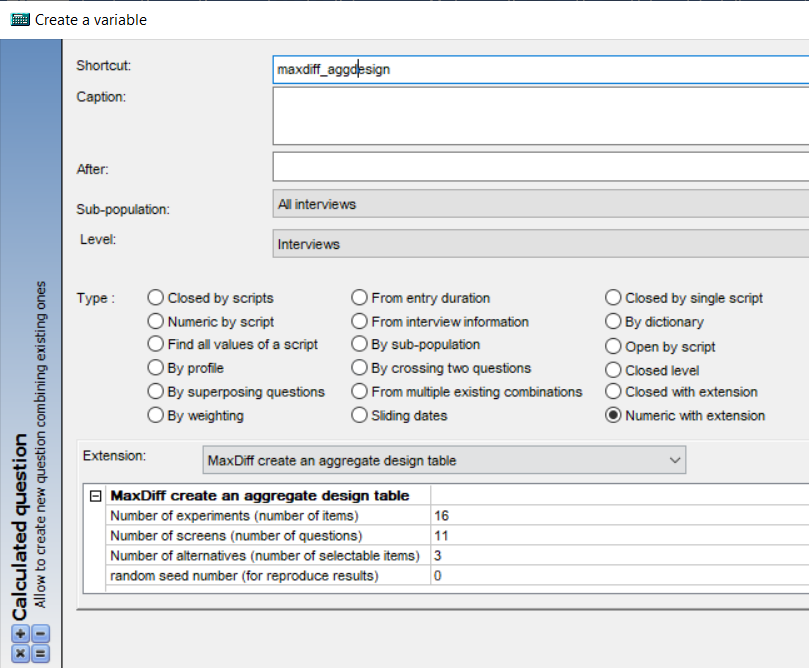
Find an example here: <https://support.askia.com/hc/en-us/articles/115005365885>

* Note that the values using into “Matrix” loop question are the data from the aggregate design table
* We can also call it the question storing the overall number of screens to be shown

NB: for this analysis, there are no data written back to askiaAnalyse as well as there is no need to have data from askiaAnalyse. We’re using askiaAnalyse’s interface just to get the values of the parameters.

### Create an analysis

It doesn’t matter if you’re using “Numeric with extension” or “Closed with extension” since no data are written back:



askiaAnalyse extentions 3 - MaxDiff aggregate design table

### Parameters

To create the aggregate design table **3 parameters (+1 optional)** are needed:

* NumberExperiments: Number of experiments (number of items)
* NumberScreens: Number of screens (number of questions)
* NumberAlternatives: Number of alternatives (number of selectable items)
* RandomSeed: Random seed number (for reproduce results)

#### NumberExperiments

* Number of experiments (number of items)

Also known as number of items, this is the number of attributes or statements you want to include overall in the MaxDiff design.

**Values possible:** from the example its value is << **16** >>

#### NumberScreens

* Number of screens (number of questions)

Also known as the number of questions or number of arrangements, this is the number of screens the respondent will see during the course of the MaxDiff section.

**Values possible:** from the example its value is << **11** >>

#### NumberAlternatives

* Number of alternatives (number of selectable items)

Also known as the number of selectable items, this is the number of options to choose between per screen.

**Values possible:** from the example its value << **3** >>

#### RandomSeed

* Random seed number (for reproduce results)

The aggregate design table is created via a process of randomization, then you’ll get different results each time you run the analysis. So, in order to be able to get the same results, you need to use a random seed number.

**Values possible:** using << **0** >> as value means no using any random seed number.

### Outputs

There is no data sent back to askiaAnalyse.

The aggregate design table is exported into a csv file:

<< Export\_MaxDiff\_AggregateDesign\_[NumberExperiments]\_[NumberScreens]\_[NumberAlternatives]\_[datetoday].csv >>

## [3] MaxDiff utilities

MaxDiff is a type of survey questioning approach in which respondents are presented with lists of items and are asked to indicate which in each list they like the most and which they like the least. It is also known as best-worst scaling, max diff, max-diff, maximum difference scaling, and Max Differential Scale Data. Typically respondents are asked to complete multiple similar tasks, where the options shown in each task varies according to an experimental design. [5]

This is the main analysis related to MaxDiff analysis which consists to calculate **MaxDiff utilities (data sent back to askiaAnalyse)** and some other statistics that are:

* utilities\_counts: individual-level counts for each respondent (using the logit model): **MaxDiff utilities**
* utilities\_ranks: individual-level ranks for each respondent (using the logit model)
* individual\_means: individual-level counts for each respondent
* mean\_ranks: overall counts for the whole sample using the arithmetic mean
* params: aggregate design table parameter's values

NB: individual refers to the interviews and level to the items (see parameter *[NumberExperiments]* from aggregate design table), then all “individual-level” statistics are table with:

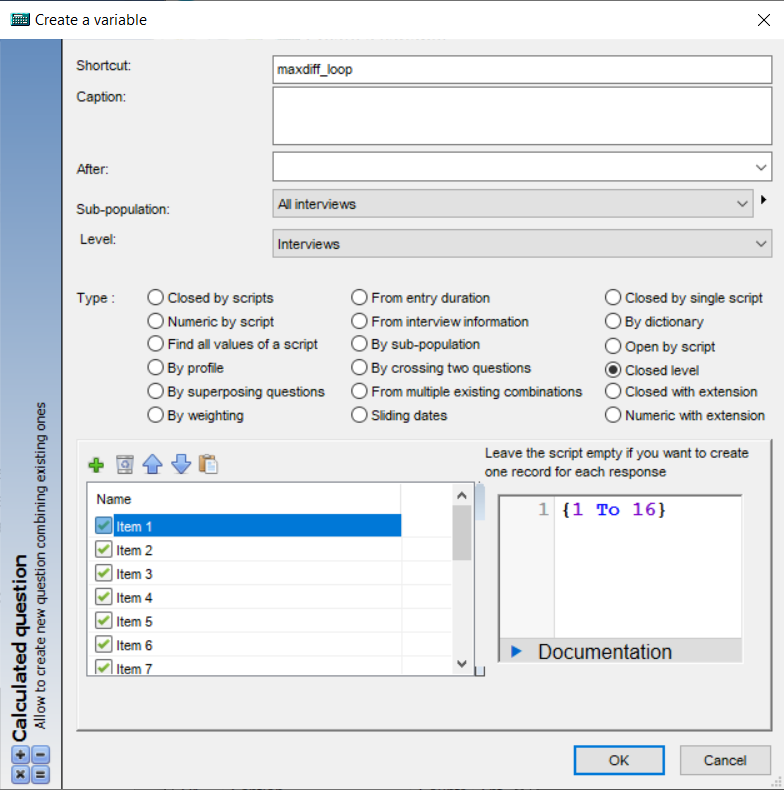
* as many rows as the number of interviews
* as many columns as the number of items

### Create an analysis

This is a “**Numeric with extension**” analysis.

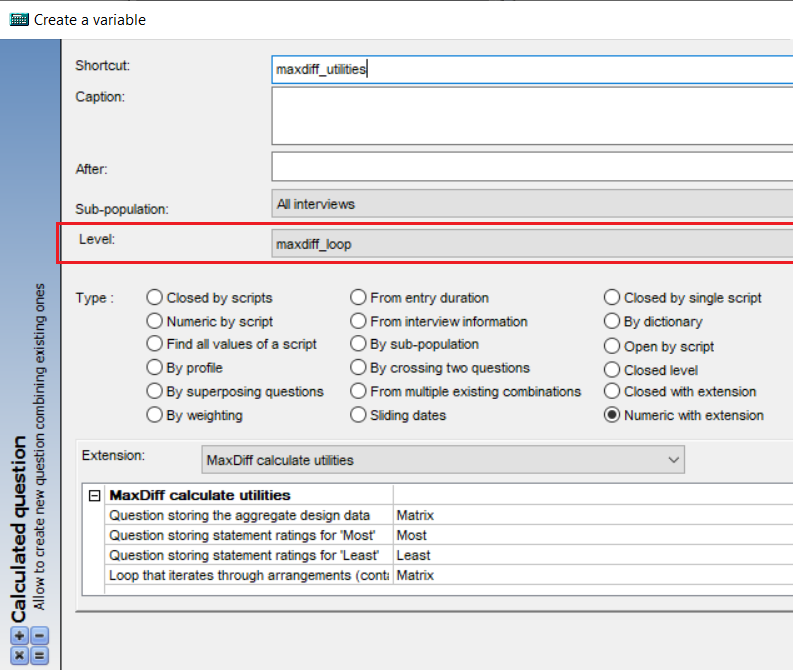
We saw that the main result from MaxDiff analysis (utilities) is a dataset with the items as columns. So, for 1 interview we’ll have as many values as items. Hence, to be able to receive many results for the same interview, we need to create a calculated loop/level question before the main calculated numeric question (the one that’ll run the analysis).

The loop calculated question will be created via “Closed level” method and it’ll have as many responses as items. The script part can be whether empty or set as << >> to create one record for each response:



askiaAnalyse extentions 4 - MaxDiff utilities: loop/level calculated question

For the numeric calculated question, you need to **set the previous loop question as Level**:



askiaAnalyse extentions 5 - Maxdiff utilities: numeric calculated question

### Parameters

To run MaxDiff analysis from R, we only need the aggregate design table and the data from the respondent. So, in order to re-create those datasets, we’ll need **4 parameters**:

* MatrixVar: Question storing the aggregate design data
* MostVar: Question storing statement ratings for 'Most'
* LeastVar: Question storing statement ratings for 'Least'
* LoopLevel: Loop that iterates through arrangements (containing Most and Least questions)

#### MatrixVar

* Question storing the aggregate design data

Here’s the question storing the values from the aggregate design table. Hence, this parameter refers to the loop question.

**Values possible:** use … to select the question.

#### MostVar

* Question storing statement ratings for 'Most'

Here’s the main question storing the overall data for “Most”.

**Values possible:** use … to select the question.

#### LeastVar

* Question storing statement ratings for 'Least'

Here’s the main question storing the overall data for “Least”.

**Values possible:** use … to select the question.

#### LoopLevel

* Loop that iterates through arrangements (containing Most and Least questions)

Here’s the loop/level question from which “Most” and “Least” are. **Same question as the one selected on *[MatrixVar]***.

**Values possible:** use … to select the question.

### Outputs

The results (utilities) are sent back to askiaAnalyse into the numeric calculated question.

Also, an Excel file with all the tables is created:

<< Export\_MaxDiff\_Utilities\_[NumberExperiments]\_[NumberScreens]\_[NumberAlternatives]\_[datetoday].xlsx >>

## [4] Turf analysis

TURF Analysis, an acronym for "Total Unduplicated Reach and Frequency", is a type of statistical analysis used for providing estimates of media or market potential and devising optimal communication and placement strategies given limited resources. TURF analysis identifies the number of users reached by a communication, and how often they are reached. […] TURF is also now used to provide estimates of market potential. [6]

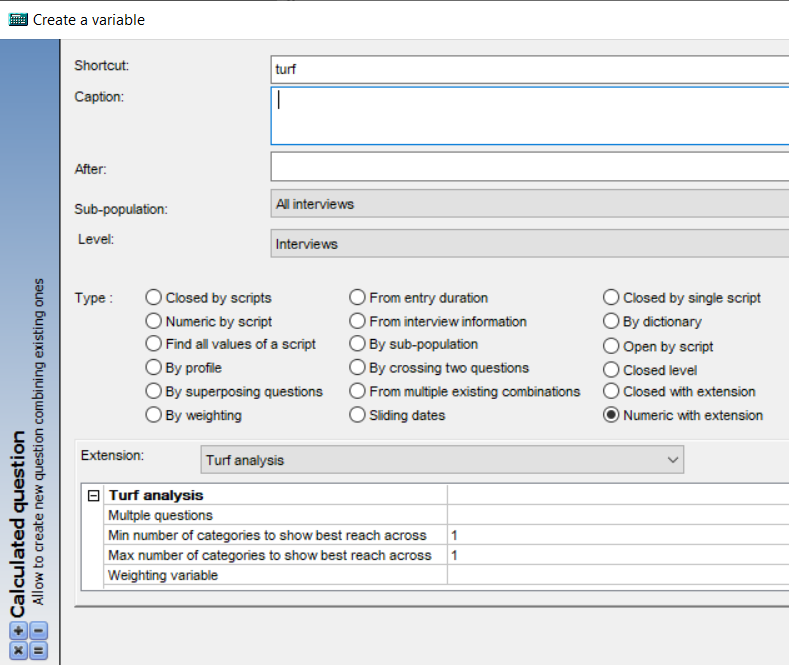
Basically, for turf what you need is a list of items (those for which you want to estimate the market potential) and know when they were reached. **So, in term of questionnaire we can considerate that context as a multiple question**. Its responses refer to the items and if the response has been selected by an interviewer, we can say that item was reached (for that interviewer).

As an example, it’s like when you have 20 different products to sell but you can put in your shop only 5 of them, then what you need is to know the best 5 products. So you’re going to launch your questionnaire with the 20 products and after getting the results, use turf analysis and specify that you want to test 5 products. What you’re going to get as result is table with all the combinations of 5 products and sorted from the best 5 to the worst 5.

NB: the result is a table with as many rows as combination tested. which is different to the interviews. Then there won’t be any data sent back to askiaAnalyse (only an output file).

### Create an analysis

This analysis can be run via “Numeric with extension” or “Closed with extension”. It needs **only multiple questions (1 or many)**:



askiaAnalyse extentions 6 - Turf analysis

### Parameters

To run this analysis **4 parameters** are needed:

* VarSource: Multple questions
* MinCategories: Min number of categories to show best reach across
* MaxCategories: Max number of categories to show best reach across
* VarWeighted: Weighting question

NB: categories refer to the question’s responses (or items)

#### VarSource

* Multple questions

Select your multiple questions.

**Values possible:** use the … to select your multiple questions.

#### MinCategories / MaxCategories

* Min number of categories to show best reach across
* Max number of categories to show best reach across

With this analysis you can test – at the same time – for more than 1 fixed number of items. To do that you need to specify a range by using different values of minimum and maximum numbers of categories (use the same value if you want to test for only 1 number of items).

Also, since you can select more than 1 multiple question, you need to specify as many values (for both min and max) as selected multiple questions (they can be separated by “,”).

NB: of course, the number of categories you want to test has to be between **1 to the maximum number of categories**.

**Values possible:** for example, you can use << 1,2 >> and << 3,2 >> if you had selected 2 questions and want to test 1 to 3 categories for the 1st one and only 2 categories for the 2nd.

#### VarWeighted

* Weighting variable

Select the weighting question. Has to be numeric.

**Values possible:** use the … to select your questions or empty of no weighting.

### Outputs

There are no data sent back to askiaAnalyse.

The result table has as columns:

* # categories tested: id of the combination of categories
* rchX: how many times the combination has reached (in %)
* frqX: how many times any product of the combination has reached (in %)
* categories: with “1” as value if the category is in the current combination (“0” if not)

The output is an Excel file with – for each question – as many tables as tested categories and some other information (name of the question, its number of responses and if a weighting question has been used).

<< Export\_TurfAnalysis\_[datetoday].xlsx >>

## [5] Convert data from wide to long format

This is not an analysis method but rather a data manipulation (handling) method. We’re using it to convert a dataset format to another, more precisely from wide (also askiaAnalyse data format) to long, which can be useful in some specific cases.

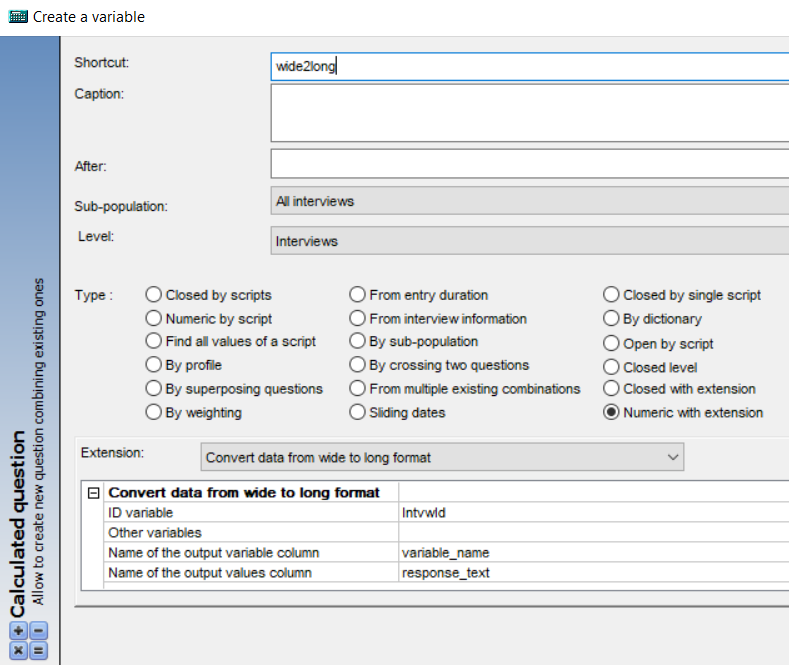
**Wide, or unstacked data** is presented with each different data variable in a separate column. […] **Narrow, stacked, or long data** is presented with one column containing all the values and another column listing the context of the value. [7]

Our datasets in long format will basically have 3 columns:

* The column with containing all the values (1)
* The columns listing the context which are:
  + A column with the question’s names (2)
  + A column with ID question’s values (3)

### Create an analysis

There won’t be data written back to askiaAnalyse then it doesn’t matter if you’re using a “Numeric with extension” or “Closed with extension” to use this method:



askiaAnalyse extentions 7 - Convert wide to long format

### Parameters

For this analysis 4 parameters are needed:

* IdVar: ID variable
* OtherVars: Other variables
* VariablesName: Name of the output variable column
* ValuesName: Name of the output values column

#### IdVar

* ID variable

Select the ID question.

**Values possible:** use … to select the question.

#### OtherVars

* Other variables

Select the questions need to be converted into the new format.

**Values possible:** use … to select the questions.

#### VariablesName

* Name of the output variable column

Set the name of the output column with the question’s names.

**Values possible:** by default << variable\_name >>.

#### ValuesName

* Name of the output values column

Set the name of the output column with the values.

**Values possible:** by default << response\_name >>.

### Outputs

There are no data sent back to askiaAnalyse.

The dataset in long format is exported into a csv file: << Export\_LongData\_[datetoday].csv >>.

## [6] Exploratory analysis

This is a dimensionality reduction method which consists to crystallising the important information in variables to a handful of richer “predictors” (or “latent”’ variables, also called “components”), therefore we will be able to gain more insight into the dataset and its underlying structure as well as making it easier to analyse. **The information in variables is also called variance or inertia**.

There are countless dimensionality reduction methods, but for this analysis we will use factorial methods and particularly methods based on PCA. There are 5 of them:

* **PCA**: principal components analysis: when **numerical variables**
* **MCA**: multiple correspondence analysis: when **categorical variables**
* **FAMD**: factor analysis of mixed data: when both **numerical and categorical variables**
* **CA**: correspondence analysis: **two variables** (create a contingency table)
* **MFA**: multiple factor analysis: when variables are brought together (or make sense) as **groups**

Special cases:

For the CA only two variables are needed. This is because a contingency table is needed to run the CA. And in order to create that table, 2 categorical variables are needed. If one (or both) of the variables is numerical then it’ll be converted to categorical.

For MFA, the variables must be specified separately per group. Also, the variables in the same group must have the same type (all numerical or all categorical), if not the numerical ones will be converted to categorical.

Type of variables and questions:

* numerical variables: numeric questions, also multiple, open and date questions
* categorical variables: single questions, also scaled questions

Note that multiple questions are converted to binaries, so they’ll be considered as numerical variables.

In order to use open questions (which are neither numerical nor categorical) into this analysis, we’re going to calculate the sentiment score of their text (see Sentiment scoring analysis) so they can be treated as numerical variables.

/!\ As see above (cleaning questions), date and scaled questions don’t have cleaning function/method, so they shouldn’t be used for this analysis. But it’s still possible to use them (not advised though) because:

* date questions are considered as numeric questions (using number of days from origin), so they can be used as numerical variables
* scaled questions are considered as single questions (using entryCode/caption instead of factor), so they can be used as categorical variables

NB: except for PCA, the number of variables will be higher than the number of questions (because closed question’s responses are most likely considered as variables).

This analysis also includes a powerful method for processing missing values. As long as you don’t have that many of missing values (in that case a pre-processing is needed), there is no need to worry about them.

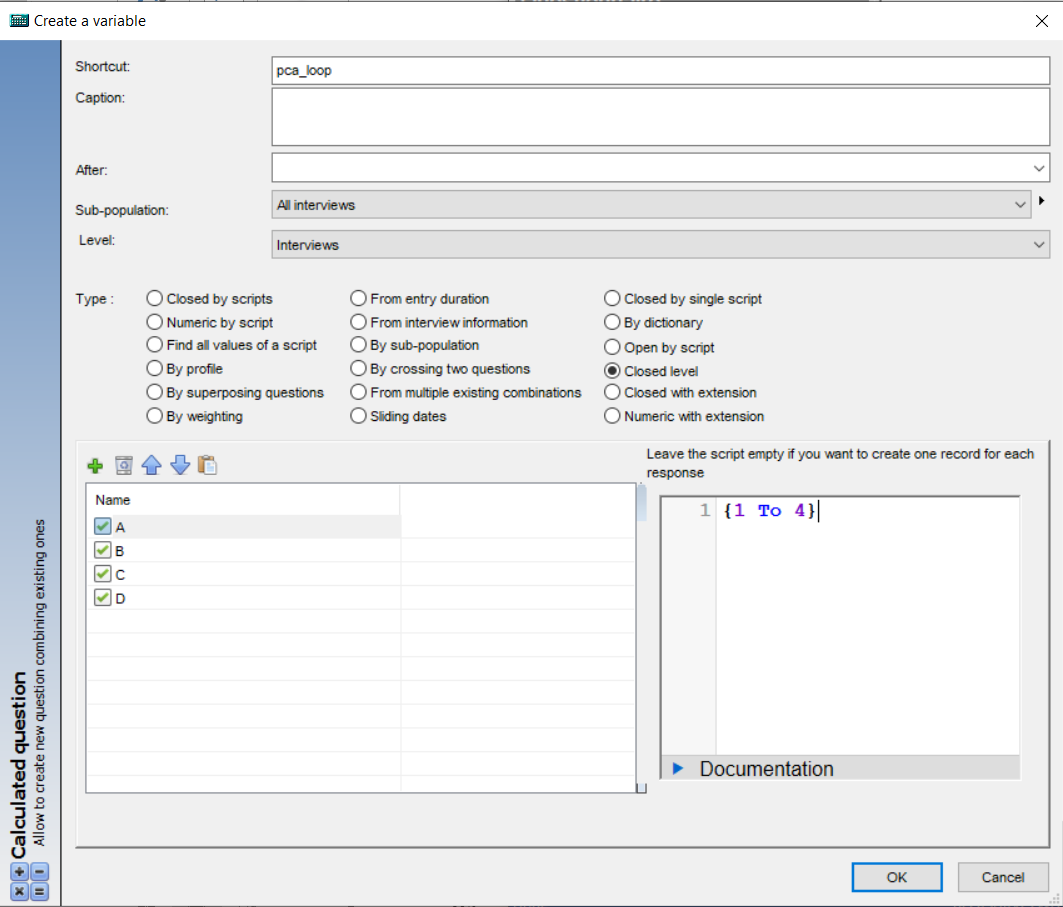
### Create an analysis

This is a **“Numeric with extension”** analysis and works with **all type of questions**:

The main result from a PCA is a new dataset with the components as columns. So for 1 interview, we can have 1, 2 or more values (depends on how many components we need). Also, note that the components are ordered by the quantity of information that they explain, then the 1st component is most important, …

Hence, to be able to receive many result values for the same interview, we need to create a calculated loop/level question before the main calculated numeric question (the one that’ll run the analysis).

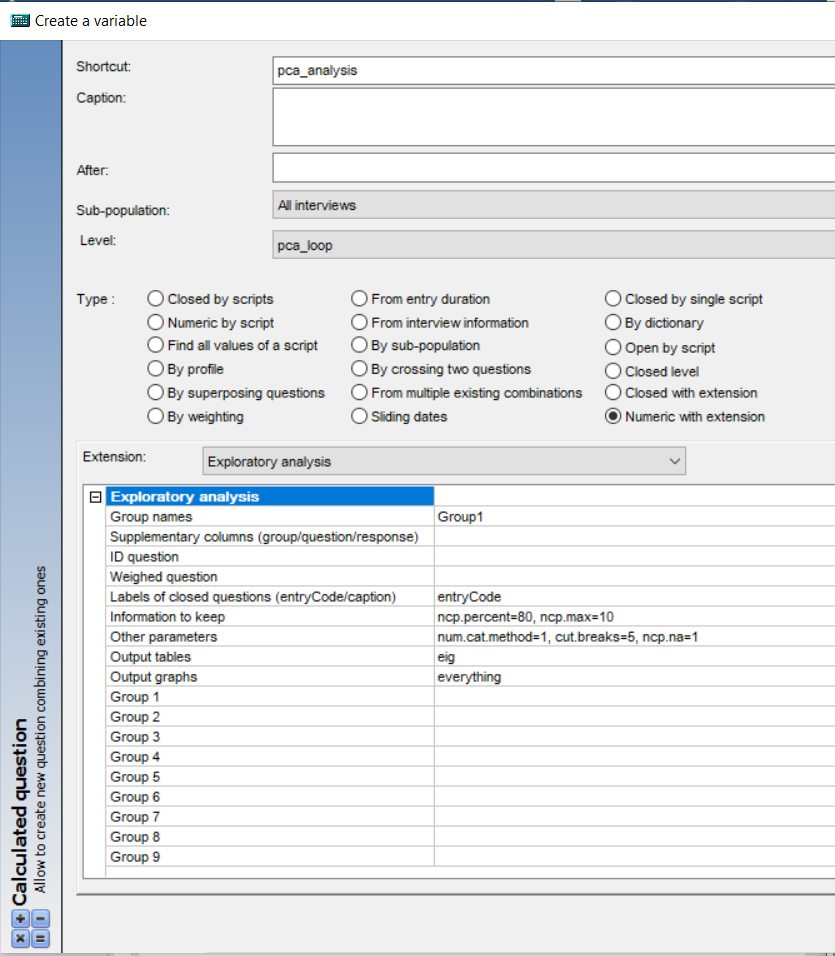
The loop calculated question will be created via “Closed level” method and it’ll have as many responses as components needed. The script part can be whether empty or set as << >> to create one record for each response:



askiaAnalyse extentions 8 - Exploratory analysis: loop/level calculated question

Set as many responses (their names don’t really matter) as many components you intend to keep from the PCA. In fact, each response will keep the data from a component. See the “Information to keep” on Parameters for more details.

For the numeric calculated question, **you need to set the previous question as Level**:



askiaAnalyse extentions 9 - Exploratory analysis: numeric calculated question

### Parameters

There are 10 parameters to set which are (name and description):

* GroupNames: Group names
* SuppVar: Supplementary columns (group/question/response)
* IdVar: ID question
* RowWeight: Weighed question
* CatLabels: Labels of closed questions (entryCode/caption)
* VarianceParams: Information to keep
* OtherParams: Other parameters
* OutputTables: Output tables
* OutputPlots: Output plots
* GroupVar1 --- GroupVar9: Group 1 to Group9

NB: there are 18 parameters in total, but << Group1, Group2, …, Group9 >> parameters are considered as an unique parameter.

#### GroupVar1 – GroupVar9

* Group1, …, Group9

This (there) is (are) the parameter (s) for selecting the questions used for the analysis. Many things (other parameters, analysis selected) will depend on how the questions are selected/grouped.

**If you select all the questions on the same group** (Group1, or any other one) then the program will run one of PCA, MCA, CA or FAMD analysis. Otherwise (**select questions into different groups**) then MFA will be run.

NB: meaningless to say that there is no use to select the same question into different groups. If so, the question will remain on the 1st group and get deleted from the others.

**Values possible:** use … to select your questions.

#### GroupNames

* Group names

Set manually the names of the groups of questions separated by “,”. If you don’t know, let it empty and the groups will be named as “Group1, Group2, …” (default naming). Of course, you need as many names as groups set before.

This parameter is useless when you have set questions for 1 group only.

**Values possible:** use for example << **customers, products, sales** >> if you have set 3 groups of questions. Or let it empty for using default naming.

#### SuppVar

* Supplementary columns (group/question/response)

Set manually the names of your supplementary columns. They can be:

* 1/ name of groups: if MFA, by selecting questions into different groups
* 2/ name of responses: if CA, by selecting only 2 questions on the same group
* 3/ name of questions: if PCA, MCA or FAMD, none of above conditions

**Values possible** (via examples)**:**

1/ (MFA) If I select my questions into 3 different groups (parameter *[GroupVar1-GroupVar3]*) and name them << customers, products, sales >> (parameter *[GroupNames]*), then if I want to use the questions into “customers” group as supplementary, I can set *[SuppVar]* parameter with << customers >> as value.

2/ (CA) If I select only 2 questions (into the same group), and one of that question has sex information (“1,2” as entryCode and “male,female” as caption), then if I want to use “male” as supplementary, I can set *[SuppVar]* parameter with << male >> (or << 1 >>, depends if *[CatLabels]* is set as caption or entryCode).

3/ (PCA,MCA,FAMD) Just need to set *[SuppVar]* with the shortcut of the supplementary questions.

Let the parameter empty if there is no supplementary column(s).

#### IdVar

* ID question

Select the question with ID values. This question can be numeric or closed (even open) as long as there are no duplicate values.

**Values possible:** use … to select your ID question (or empty if no ID).

#### RowWeight

* Weighed question

Select the question with weighting values. This question must be numeric.

**Values possible:** use … to select your weighting question (or empty if none).

#### CatLabels

* Labels of closed questions (entryCode/caption)

Set the label you want to use for closed question. Only 2 choices available: entryCode or caption.

NB: by using entryCode as label, the missing values (“don’t know” (-1) and “not asked” (-2)) will be considered (in R) as new responses and not real missing values.

**Values possible:** use << **caption** >> or << **entryCode** >>.

#### VarianceParams

* Information to keep

The quantity information/variance/inertia is related to the number of components. In fact, each component describes a specific quantity of information.

This parameter is used to set the quantity of information to keep. Put it another way, to set the number of components to keep. This parameter is managed by 2 sub-parameters:

* ncp.percent: minimum % of inertia to keep, a number between ]0;100]
* ncp.max: maximum number of components to keep, a number between [1;nb\_variables]

By setting ncp.precent you’ll get a specific number of components, but don’t know exactly how many (the answer will be available after the PCA). So, use ncp.max to specify a maximum number of components you’re allowing. If you don’t want to set any maximum use << ncp.max=Inf >>.

With those 2 sub-parameters you need to pay attention to the loop calculated question, especially its number of responses which will have the same job as ncp.max. In fact, each response will get the value of a component, so you can’t get more components than responses.

**Values possible:** as example use << **ncp.percent=100, ncp.max=Inf** >> to keep all the information. Also, its default value is << **ncp.percent=80, ncp.max=10** >>.

#### OtherParams

* Other parameters

Here are some sub-parameters useful in case you want to have more control on how the analysis is run. There are 3 of them:

* num.cat.method: method to use when you have both numerical and categorical variables
  + use << num.cat.method=1 >> for FAMD
  + use << num.cat.method=2 >> for CUT method (see below)
* cut.breaks: the number of cut/group needed for CUT method, which is used when:
  + << num.cat.method=2 >>
  + CA with numerical variable(s)
  + MFA with both numerical and categorical variables on the same group
* ncp.na: number of components in order to impute missing values
  + by using << ncp.na=NULL >> the program will choose its best value (**but it’s time consuming**)
  + the more missing values you have the higher its value should be
  + however, the higher its value is the longer the “imputing missing values” part will be

CUT is a method that converts a numerical variable to categorical by cutting/splitting/dividing the range of the numerical variable into intervals.

NB: by using CUT method (<< num.cat.method=2 >>) instead of FAMD (when you have both numerical and categorical variables, and on the same group), all numerical variables will be converted to categorical. Therefore, since there are only categorical variables in the dataset then an MCA will be run.

**Values possible:** its default value is << **num.cat.method=1, cut.breaks=5, ncp.na=1** >>.

#### OutputTables and OutputPlots

* Output tables
* Output graphs

Outputs are available for these indicators:

* eig: eigen values (i.e. inertia per components)
* coord: coordinates (data sent back to askiaAnalyse)
* cos2: square cosine
* contrib: contributions
* inertia: inertia
* correlations: correlations
* Lg: Lg coefficients
* RV: RV coefficients
* dist2: distance to the origin

Those indicators are produced for the individuals (ind) and/or the variables (var). Their availability depends on the type of output (table or plot) and the type of analysis:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Output name | | Output type | | Analysis | | | | |
| main | sub | table | plot | PCA | MCA | CA | FAMD | MFA |
| eig |  | OK | OK | OK | OK | OK | OK | OK |
| biplot |  | X | OK | OK | OK | OK | X | X |
| ind | all | X | OK | OK | OK | X | OK | OK |
| ind | coord | OK | OK | OK | OK | OK | OK | OK |
| ind | cos2 | OK | OK | OK | OK | OK | OK | OK |
| ind | contrib | OK | OK | OK | OK | OK | OK | OK |
| ind | inertia | OK | X | X | X | OK | X | X |
| var | coord | OK | OK | OK | OK | OK | OK | OK |
| var | cos2 | OK | OK | OK | OK | OK | OK | OK |
| var | contrib | OK | OK | OK | OK | OK | OK | OK |
| var | inertia | OK | X | X | X | OK | X | X |
| var | correlation | OK | X | X | X | X | X | OK |
| var | Lg | OK | X | X | X | X | X | OK |
| var | RV | OK | X | X | X | X | X | OK |
| var | dist2 | OK | X | X | X | X | X | OK |

Table 2 - Exploratory analysis: output's availability

NB: biplot is a plot with the coordinates of both individuals and variables.

To export a specific output, use this syntax: << main-name=sub-name >> (or << main-name >> if no sub-name). Example << eig, ind=coord, var=Lg >>.

Additional indicators for *[OutputPlots]*:

* you can use << x=1, y=2 >> to specify exactly for which components you want the plot to be drawn (here’s for the 1st and 2nd components)
* you can use << fmt=pdf >> to export all the plots on a same pdf file, or << fmt=png >> to export the plots on different png files.

Special cases: for MFA “var” refers to the groups, and for CA “ind”/”var” refer to the rows and columns from the contingency table.

**Values possible:** you can use << everything >> to export all of the available outputs. It the same as using:

* for *[OutputTables]*: << eig, ind=coord, ind=cos2, ind=contrib, ind=inertia, var=coord, var=cos2, var=contrib, var=inertia, var=correlation, var=Lg, var=RV, var=dist2 >>
* for *[OutputPlots]*: << eig, biplot, ind=all, ind=coord, ind=cos2, ind=contrib, var=coord, var=cos2, var=contrib, x=1,y=2, fmt=pdf >>

### Outputs

The results are sent back to askiaAnalyse into the numeric calculated question.

For the tables, you’ll get an excel file with all tables asked: << Export\_ExploratoryAnalysis\_Tables\_[datetoday].xlsx >>

For the plots, you’ll get all the plots in one pdf << Export\_ExploratoryAnalysis\_Plots\_[datetoday].pdf >> or in different png << Export\_ExploratoryAnalysis\_Plots\_[main-name]\_[sub-name] \_[datetoday].png >>.

NB: Individuals refer to the interviews except for CA, for which individuals refer to the contingency table’s rows (1st variable’s responses). Also, variables refer to the contingency table’s columns which are the 2nd variable’s responses. That’s why there’s no data written back to askiaAnalyse for that analysis (number of interviews is not the same as the number of responses), so don’t forget to export its data to excel (with *[OutputTables]*: << ind=coord, var=coord >>).

### Bad practices

Here some bad practices to avoid. Some of them are just reminder.

ID question with duplicates or missing values

Empty questions or question with a unique value

Same question on different groups

Create the analysis outside a loop calculated question

Weighting question (as much as you can)

Good tips:

Do not hesitate to use entryCode as labels to keep use missing values as new responses (for closed questions) and see what you get, especially when you have a lot of them.

For ncp.na, usually a value between 1 to 5 is good enough.

To use the data from a CA, you need to get its data from the output excel file and create a new QES.

The CUT method is not the most proficient converting method, do not hesitate to convert some questions – if needed – by yourself.

Since you can decide what is the best number of components only after getting the analysis results, you can set the number of responses of the loop question quite high, then you’ll get the data anyway and you can decide if you still want to use all of them or less.

## [7] Clustering (after PCA)

**Cluster analysis** or **clustering** is the task of grouping a set of objects in such a way that objects in the same group (called a **cluster**) are more similar (in some sense) to each other than to those in other groups (clusters). [8]

To summarize clustering method’s main aim is to divide your interviews into different groups with similar profiles.

There are countless methods of clustering, but so far only 2 of have been included on this analysis:

* [K-means](https://en.wikipedia.org/wiki/K-means_clustering)
* [Agglomerative clustering](https://en.wikipedia.org/wiki/Hierarchical_clustering)

Also, after run your clustering you need to evaluate it in order to say if it’s a good classifier (general name of a clustering method) or not. In general terms we say that a good classifier is when:

* [1st rule] in the same group, the population is as similar as possible
  + minimize the intra-cluster variance
* [2nd rule] for any two group, they are as different as possible
  + maximize the inter-cluster variance

There are countless indicators allowing to evaluate a classifier, some prioritize the 1st or the 2nd rule, and other do a mix of both 1st and 2nd rule. We’ve decided to use the [Silhouette score](https://en.wikipedia.org/wiki/Silhouette_(clustering)) (or Silhouette coefficient) as indicator because:

* it mixes both 1st and 2nd rule
* its values are between -1 to 1
* it’s easier to interpret (the higher the score is, the better the classifier is)

|  |  |
| --- | --- |
| **RANGE OF SC** | **INTERPRETATION** |
| 0.71-1.0 | A strong structure has been found |
| 0.51-0.70 | A reasonable structure has been found |
| 0.26-0.50 | The structure is weak and could be artificial. Try additional methods of data analysis. |
| < 0.25 | No substantial structure has been found |

Table 3 - Silhouette score interpretation [9]

Both of clustering methods used here (k-means and agglomerative) have a big handicap which is **that you need to specify how many clusters you want** before even running it! In some special cases this can rather be an advantage (e.g. when you want a specific number of clusters). So in order to help the users, we allow him to use a range of values to test (instead of only one) and the program will choose itself the optimal number of clusters (the one with the higher silhouette score).

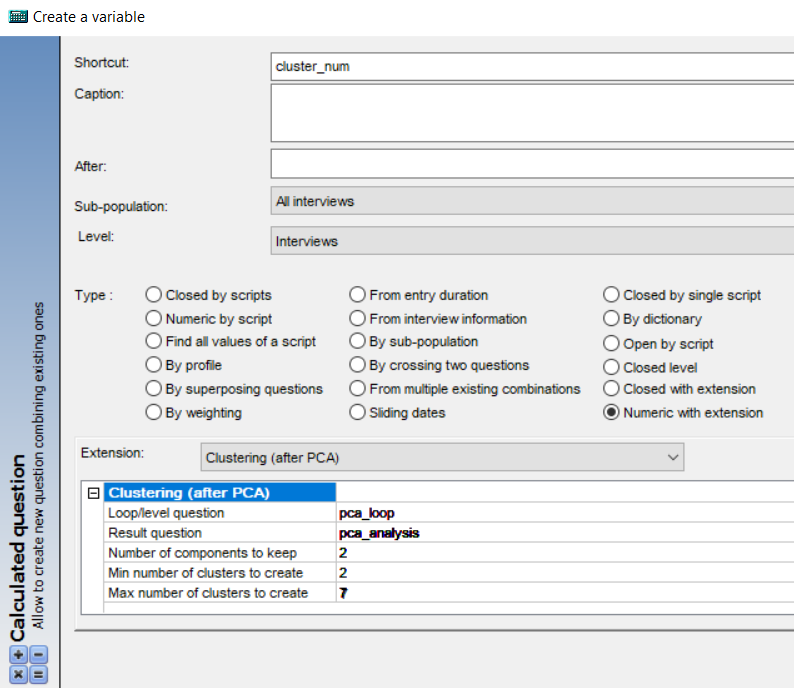
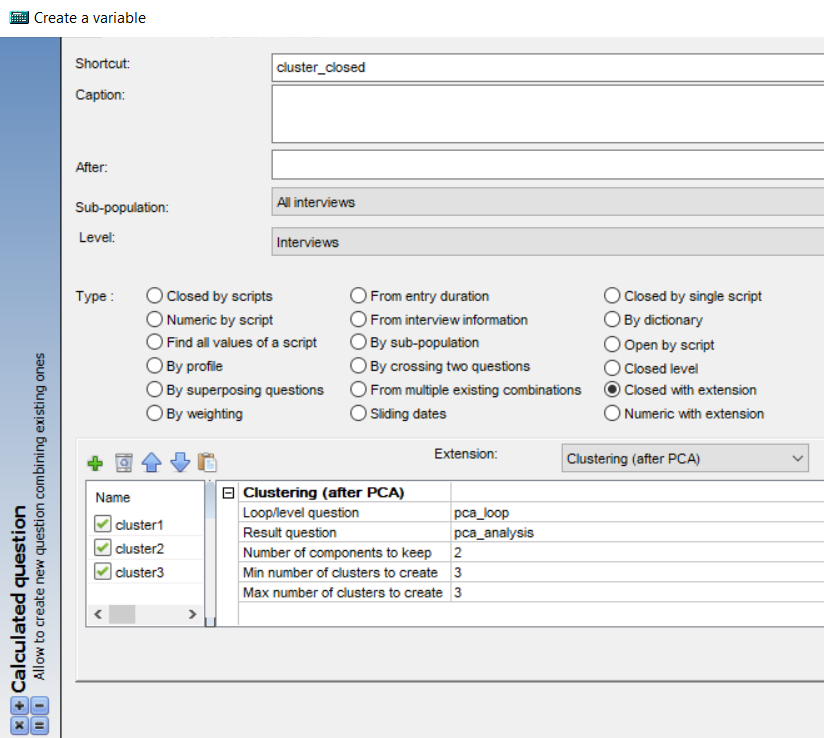
NB: this analysis is called Cluster **after PCA** (i.e. after Exploratory analysis) because it needs data from a PCA which are **structured as a numeric question into a loop question**. Thanks to that, you don’t really need data from a PCA as long as they are structured in the same way (numeric into loop).

### Create an analysis

The result from a clustering are labels that refer to which group an interview belongs to. In that way this should be launched as a “**Closed with extension**” analysis. However, since you need to set first the responses (as many as number of clusters you want) of your closed calculated question, this can be painful when you don’t know exactly the number of clusters you want.

What you can do is:

* Use “Closed with extension” when you know the number of clusters you want
* Use “Numeric with extension” otherwise, i.e. you have many values you want to test (also you can still convert that question to a Closed one)



askiaAnalyse extentions 10 - Clustering: closed and numeric calculated questions

### Parameters

There are **5 parameters** here:

* LoopQuestion: Loop/level question
* ResultQuestion: Result question
* nComponents: Number of components to keep
* nClustersMin: Min number of clusters to create
* nClustersMax: Max number of clusters to create

#### LoopQuestion and ResultQuestion

* Loop/level question
* Result question

Select here the data from the PCA by selecting the loop/level calculated question (*[LoopQuestion]*) and the numeric calculated question (*[ResultQuestion]*).

**Values possible:** use … to select your loop and numeric questions.

#### nComponents

* Number of components to keep

Remind from Exploratory analysis: the best number of components is decided with the help of the eigen values (inertia per components) which is obtained after the PCA.

In that way, it may happen that the number of components you want to use is less than the one you kept (same as the number of responses of the loop question). This parameter is here to fix that.

**Values possible:** use << **0** >> if you want to keep the data of all the components.

#### nClustersMin and nClustersMax

* Min number of clusters to create
* Max number of clusters to create

Use these parameters to specify the range of number of clusters you want to test. Use the same value for both of them in the case if you know exactly how many clusters you want.

**Values possible:** for example, you can use << 2 >> and << 5 >> if you want to find which is the number of clusters between 2,3,4,5.

### Outputs

The data from the clustering are sent back to askiaAnalyse.

Also, for each clustering method, there is pdf file with different plots inside:

* The plot with the evolution of the silhouette score per number of clusters
  + only if many values have been tested
* The silhouette analysis plot: a plot with the silhouette coefficients for each interview and the average silhouette score as threshold. With that we can that,
  + any interview over the threshold is well classed (he is in the good cluster)
  + any interview less than “0” can be considered as an outlier
* The plots with the interviews data and their cluster
  + plot 2D:1st and 2nd components
  + plot 3D: 1st, 2nd and 3rd components (if available)

The output files are named as:

* << Export\_Clustering\_Plots\_k-Means\_[datetoday].pdf >>
* << Export\_Clustering\_Plots\_Agglomerative-Clustering\_[datetoday].pdf >>

### How to improve my classifier?

As seen in Table 3, when you have a silhouette score between 0.26 to 0.5 (or even more) it’s advised to use another clustering method (which is not possible since only 2 are available so far).

There are other ways to improve a classifier. The main one is to improve the PCA by having more control into your data. **This is called data pre-processing**. For example:

* for numeric questions: convert them to single
* for multiple questions: converted them to single by using the 1st answered
* for single questions: merge the responses with few values to the big ones
* for date questions: use its month or season instead

You can use one or a mix of this tips and also you can do it for some specific questions (the most important) or all of them. This can be done upstream manually from askiaAnalyse.

# To update

# References

|  |  |
| --- | --- |
| [1] | Data Cornering, “How to run R scripts from the Windows command line (CMD),” Data Cornering, [Online]. Available: http://datacornering.com/how-to-run-r-scripts-from-the-windows-command-line-cmd/. [Accessed 14 09 2020]. |
| [2] | A. Sharma, “How to Run Python Scripts,” DataCamp, [Online]. Available: https://www.datacamp.com/community/tutorials/running-a-python-script. [Accessed 14 09 2020]. |
| [3] | V. Guyader, «Créer un package R en quelques minutes,» ThinkR, [En ligne]. Available: https://thinkr.fr/creer-package-r-quelques-minutes/. [Accès le 14 09 2020]. |
| [4] | S. Rochette, «Quand le developpement commence par la documentation,» ThinkR, [En ligne]. Available: https://rtask.thinkr.fr/fr/quand-le-developpement-commence-par-la-documentation/. [Accès le 14 09 2020]. |
| [5] | Displayr, “MaxDiff,” Displayr, [Online]. Available: https://docs.displayr.com/wiki/Category:MaxDiff#What\_is\_MaxDiff.3F. [Accessed 14 09 2020]. |
| [6] | Wikipedia, “TURF Analysis,” Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/TURF\_Analysis. [Accessed 14 09 2020]. |
| [7] | Wikipedia, “Wide and narrow data,” Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Wide\_and\_narrow\_data. [Accessed 14 09 2020]. |
| [8] | Wikipedia, “Cluster analysis,” Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Cluster\_analysis. [Accessed 14 09 2020]. |
| [9] | The Wayback Machine, “Partitioning Around Medoids (PAM),” Internet Archive, 02 10 2011. [Online]. Available: https://web.archive.org/web/20111002220803/http://www.unesco.org:80/webworld/idams/advguide/Chapt7\_1\_1.htm. [Accessed 14 09 2020]. |