Nogrod 1.1.2

Quick tutorial

Martin Wettstein

Nogrod, n [nogrod] 1. Python script to reshape and analyze datasets in text format prior to importing them to statistical software packages. 2. (from Sindarin: *hollow-bold*): Ancient dwarf city in the Blue Mountains, renowned for skilled craftsmen. Popular exports of the smiths in Nogrod were the Nauglamir and Angrist.

This is a beta version of Nogrod. Please contact m.wettstein@ikmz.uzh.ch or post an issue on Github if anything does not work as supposed.

Program, example data, and documentation downloadable from:

https://github.com/Tarlanc/Nogrod

Zürich, December, 2019

Contents

Chapter 1. Introduction

Nogrod is a script for Python 2.7 which uses the tk-interface of the current platform to enable quick and easy data processing and data manipulation. The script was designed to process and reshape data stored in textfiles (CSV or tabstopp separated) without the need to open them in statistical software packages. The program has a few easy to use routines which allow aggregation, combination, cross tabulation, and some basic analysis functions such as contingency analysis, for numerical and non-numerical data.

The program is based on the Angrist engine for content analysis data input and is specifically designed to read and process the data stored from Angrist. It may, however, be used for any kind of data which is stored in textfiles.

As of version 1.1, Nogrod may also be included as a module to any Python script. Upon inclusion, most of the functions of Nogrod may be called directly. The syntax is provided in the glossary in the appendix.

1.1. Example Data

In this quick tutorial a small and wholly artificial dataset is used to explain and demonstrate the basic functions of Nogrod. The data is designed to look a little like content analysis data on three US TV-networks and their quotation of political actors on three issues. All data (except for the actor affiliations) is fictional and solely for education purposes.

The whole example dataset may be found in the file 'Example_Data.txt'.

Table 1: Example Data for this quick reference. The table contains 14 variables and 19 cases as may be obtained by quantitative content analysis of evening news. The data is purely fictional and just meant for illustration purposes.

Show_ID	Station	Date	Actor	Actor_Affiliation	Quotation	Issue	Argument_Moral	Argument_Economy	Argument_Voter	Argument_Justice	Argument_Faith	Attack	Attack_Target
FN001	Fox	Mar 13	Obama, Barrack	Dem	Direct	Budget	1	1	0	0	0	0	
FN001	Fox	Mar 13	Van Hollen, Chris	Dem	Indirect	Budget	0	0	0	0	0	0	
FN001	Fox	Mar 13	Ryan, Paul	Rep	Direct	Budget	0	0	0	1	0	1	Obama, Barrack
FN001	Fox	Mar 13	Cruz, Ted	Rep	Direct	Obamacare	0	1	0	0	0	1	Obama, Barrack
FN002	Fox	Mar 14	Christie, Chris	Rep	Indirect	Obamacare	1	0	1	0	0	0	
FN002	Fox	Mar 14	Ryan, Paul	Rep	Direct	Budget	0	0	0	0	1	1	Biden, Joe
FN002	Fox	Mar 14	Kerry, John	Dem	Indirect	Budget	1	0	1	0	0	0	
FN003	Fox	Mar 20	Christie, Chris	Rep	Direct	Iran	0	0	0	0	1	1	Foreign Actor
PB001	PBS	Mar 13	Van Hollen, Chris	Dem	Direct	Budget	0	1	0	1	0	0	
PB002	PBS	Mar 15	Obama, Barrack	Dem	Indirect	Iran	0	0	0	0	0	1	Foreign Actor
PB002	PBS	Mar 15	Cruz, Ted	Rep	Direct	Iran	0	0	0	0	0	0	
PB003	PBS	Mar 20	Van Hollen, Chris	Dem	Direct	Budget	0	1	0	1	0	0	
PB003	PBS	Mar 20	Obama, Barrack	Dem	Indirect	Obamacare	0	0	0	0	0	0	
AB001	ABC	Mar 14	Ryan, Paul	Rep	Direct	Obamacare	0	0	0	0	1	0	
AB001	ABC	Mar 14	Kerry, John	Dem	Direct	Obamacare	0	0	0	1	0	0	
AB001	ABC	Mar 14	Cruz, Ted	Rep	Direct	Obamacare	0	0	0	0	1	0	
AB001	ABC	Mar 14	Van Hollen, Chris	Dem	Direct	Budget	0	1	0	0	0	1	Tea Party
AB002	ABC	Mar 15	Obama, Barrack	Dem	Direct	Iran	1	0	0	0	0	0	
AB002	ABC	Mar 15	Kerry, John	Dem	Indirect	Iran	0	0	1	0	0	1	Foreign Actor

In addition to this example data, there is a second table holding additional information on certain actors, which is used as an example for the merging function (add variables). This table is considerably small and just contains the functions of the actors (Table 2)

Table 2: Example Functions. This table contains additional information on the actors in Table 1. The data is actually accurate but by no means systematic. It is just meant for demonstration purposes.

Actor	Function
Obama, Barrack	President
Van Hollen, Chris	Representative
Ryan, Paul	Representative
Cruz, Ted	Senator
Christie, Chris	Governor
Kerry, John	Secretary of State

Since this table contains only few cases and just four different timestamps (March 13, 14, 15, and 20), this data would not be useful in time series analyses. For this reason, there is another example dataset which contains time series data. The table has three variables (see Table 3) and 2319 cases.

Table 3: Variables in time series example data

Variable	Description	Range
Time	Timestamps in seconds	30.12-319.88 Rate=8Hz
SkinCond	Skin Conductance in micro-Siemens for each measruement point	1.97-2.65 M=2.114; SD=.136
Heartrate	Heartrate in beats per minute, calculated once per second	55.45-80.00 M=67.44; SD=4.11

Since the whole table may not be shown here for space issues, the data is visualized in two line graphs for the two time series variables (see Figure 1).

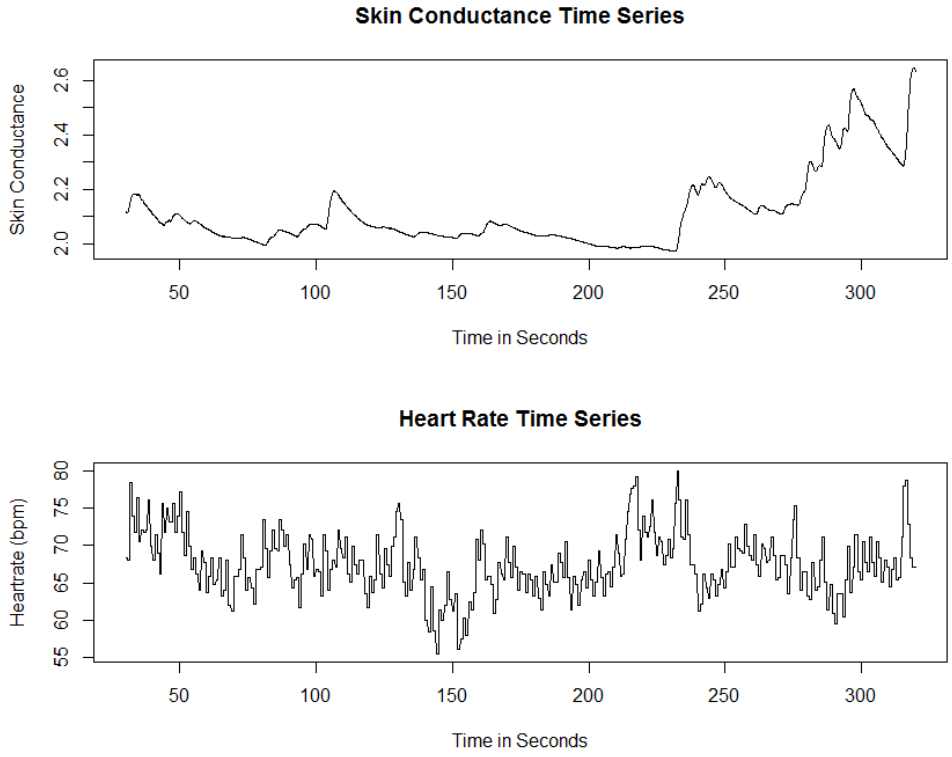


Figure 1: Time Series Data visualization.

Chapter 2. Handling of Nogrod

Nogrod was designed to offer an intuitive interface for data processing. Since each kind of data processing requires a set of parameters (such as input data, procedure, method, output format...), the user is required to provide some input on the data processing routine to be performed. As in Angrist, the user is asked to enter each of the parameters in a sequence of easy to answer questions with help texts. After each confirmation, Nogrod checks the validity of the parameters and provides the user with additional information on the data and potential risks when using a parameter which does not fit the data entered.

The screen of Nogrod always has the same structure (See Figure 2) to enhance usability and to keep it as simple and straightforward as possible.

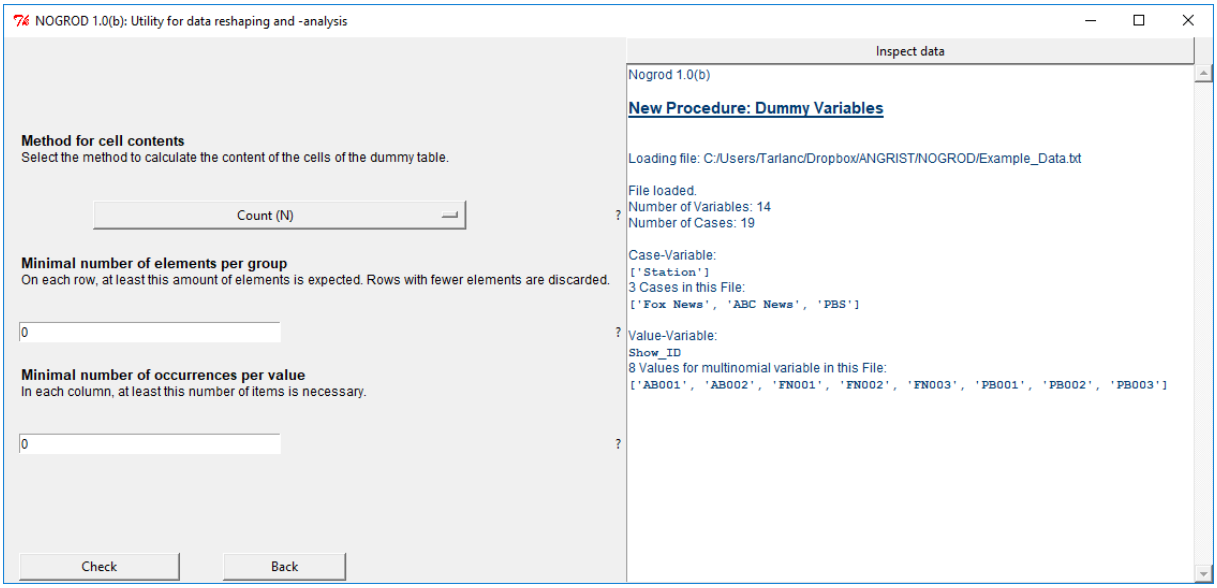


Figure 2: Screenshot from Nogrod on the left the parameters may be specified, on the right the tool offers additional information on the data and current processes.

In figure 1 you see the screen as it presents itself during the parameter entry for a dummy transformation. On the right hand side there is a console providing text output on the current data processing routine. In this example, the user has loaded a dataset with 14 variables and 19 cases. The user already selected a variable for cases with 3 different values and a multinomial variable to be transformed to dummies with eight values.

On the left hand side, the program asks for three inputs to be made for this data processing method. First, it asks for the method to be used in the dummy transformation. With the current entry the program outputs counts of occurrences of each value within each group. Second, the program asks for the minimal number of occurrences for each value and group to be included in the resulting dummy table.

Upon entering all data, you may confirm using the button 'Check' at the bottom of the left-hand side of the program. If a previous selection was wrong, you may also choose to go back by one page using the button 'Back'.

If insecure about the contents of the datafile, the button 'Inspect data' above the text output offers quick descriptive statistics on any variable in any dataset loaded at the moment. For some procedures, two or three datasets may be loaded at the same time.

Attention: Upon confirming the entries with 'Check', the program performs a set of preliminary analyses on the data to ensure the validity of all parameters. For large datasets this may take some seconds in which the check button appears sunken and the program does not react to any input. Please be patient and wait for the program to finish its task. After the last page, which usually includes the specification of an output file, the program performs the whole transformation. For large datasets this may take several minutes. Please be patient and don't force the program to close.

Upon completion of a data processing routine, the program asks whether you want to proceed with another routine. You may continue using the data you just created or use another dataset for analysis or transformation.

2.1. Style Sets

There are different style sets for the text display on the right-hand side of the dialog. In the settings (n_settings.ini) you may change the value of Styleset to `Default` (Darkblue on white), 'Tardis' (White on dark blue), 'Hacker' (Green system font on black), or 'Debug' (higlighted backgrounds for different style elements).

You may also change the orientation of the whole window from left- to right-handed by changing the value of `Layout` to 'Lefty' or 'Righty'.

2.2. Help texts

At the right-hand side of each question, a small question mark is visible. These question marks provide additional information on the parameter to be entered. Just click them to open a pop-up window with additional information (see Figure 3).

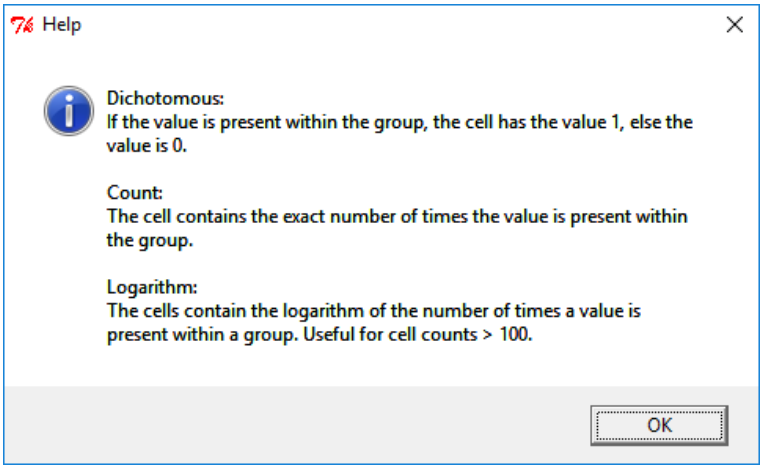


Figure 3: Help pop-up for the Method Variable

2.3. List selection

In most routines some parameters are entered as lists of objects you wish to include in the analysis. These lists may consist of variables, values, or cases. To enter these lists as quickly as possible you are presented with a list containing all possible selections and an empty list of selected items (see Figure 4).

You may add single items to the list at the bottom by selecting them and pressing 'Add Item'. The item will then be added to the bottom list but not removed from the list of possible selections. If you want to add all items to the selection you may press 'Add all'. If an item is added to the selection which you don't want to add, you may press 'Remove Item' to remove it from the selection again. This enables you to select all but one item by adding all items first and then

removing the one item you wish to exclude. By pressing 'Remove All', the selection list will be cleaned and you may start selecting again.

Please note that for some list selections it is necessary to include at least one element to the selection list. If no item is added, the programm will not accept the entry and prompt you to select an item.

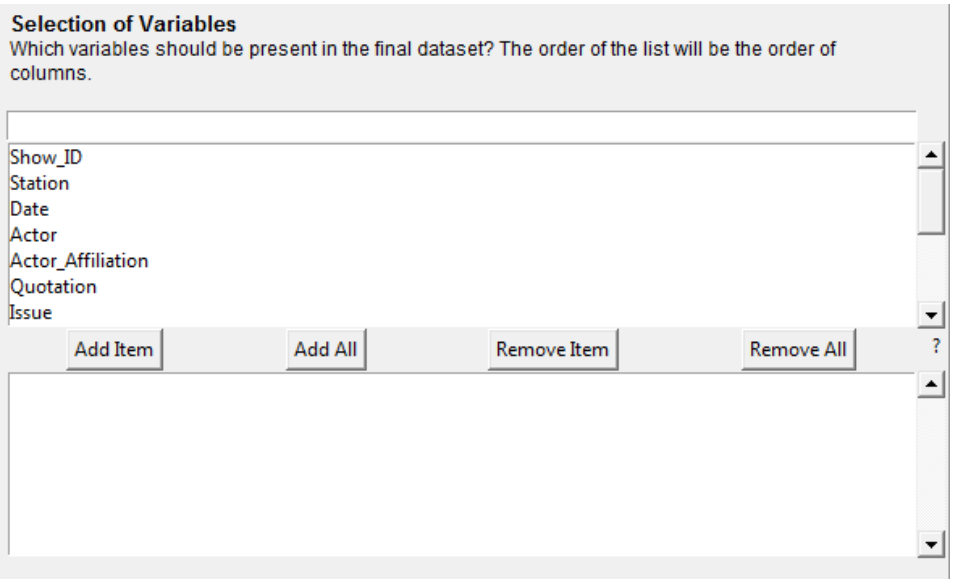


Figure 4: Screenshot of the list selection. In the top list all possible selections are presented. You may add and remove them to the selection list at the bottom using the four buttons between the lists

2.4. Starting Nogrod

Download Nogrod.zip from:

http://www.ipmz.uzh.ch/en/Abteilungen/Medienpsychologie/Recource/Nogrod.html

and unzip it to any location on your PC.

Download and install Python 2.7 (Not Python 3!) from http://www.python.org.

You may then start Nogrod by double-clicking nogrod.py.

Shortcut for Division Mediy Psychology&Effects:

Double-click the file:

G:\[IPMZ-intern]\Abteilung V\[Interner Bereich]\Forschung\Python_Programme\Nogrod.bat

2.5. File handling

For loading and storing files, Nogrod allows for two different ways. First, you may enter a filename directly to the textbox. If you want to enter the whole path, please use slashes (/) instead of backslashes (\) to separate directories. This applies to windows as well, which usually does not support slashes. Backslashes may not be recognized properly.

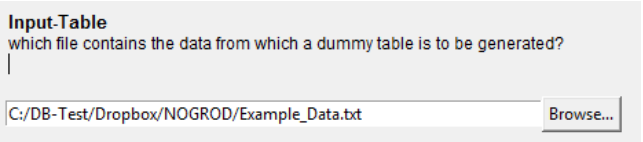


Figure 5: Input of filenames

If you don't know the exact filename you may click 'Browse...' to look up the file on your computer. Depending on the platform and version, the dialog box which opens may differ. It is usually identical to the dialogs used in programs on this computer (e.g. MS Word).

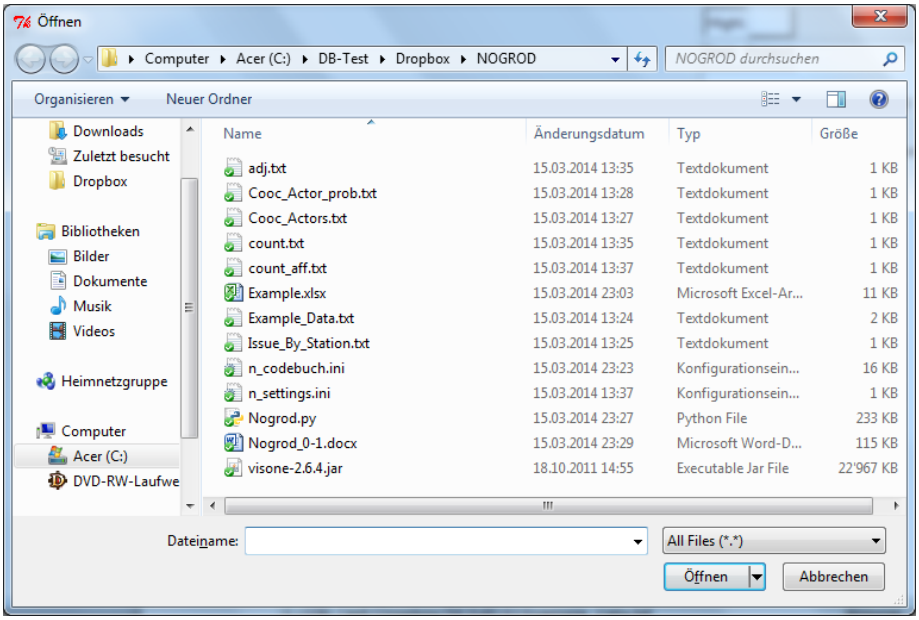


Figure 6: File dialog. This dialog may differ depending on the platform Nogrod is used on.

2.6. Header and Separator

When tables are stored in text files, they may have first line (a header) containing the name of each variable. This line should contain the same number of entries as each row below it and each name should only occur once. If any name occurs twice in a file, Nogrod will automatically rename the variables to make them unique by adding a number to each duplicate. If no header is present, each variable will be assigned a number.

Depending on the style of the text file, the columns (or cells within a row) are separated by a special character. In output files from Angrist, the separator is always a tabstopp. The same applies to text files which are created by copy&pasting values from an Excel table. In CSV files, the cell delimiter or separator is a semicolon (;). In BASIC tables, the separator is a comma (,) and sometimes a blank space is used to separate columns.

When opening or storing any file in Nogrod you are asked to specify whether this file has a header and which separator is used. Default values are 'Yes' and 'Tabstopp'.

Chapter 3. Data Transformation

The first family of functions is used for quick transformations of data tables. In this family, you can sort tables, rename and select variables. These functions are useful in data preparation before working with a dataset.

3.1. Sort table

This functon allows you to sort a table by one or several variables.

Applications:

This procedure may be used whenever you want to:

- Sort a table

Options:

Input Table
This table is a text document.

Sort cases by
You may select any number of variables to sort the dataset by. The first variable in the list has the highest weight.

Sort direction
You may choose the sorting to be ascending or descending.

Output File
The resulting table will be written in a text file just like the input table. You may enter a filename or browse the directories to specify a file.

3.2. Rearrange Variables

This functon allows you to rearrange the variables of a table to move the most used variables in the front of the dataset. You may also choose to leave out some variables to generate a slim table only carrying the necessary information.

Applications:

This procedure may be used whenever you want to:

- Change the order of variables
- Select a subset of variables
- Make copies of variables

Options:

Input Table
This table is a text document.

New List of variables

From all variables in the table you may select the new order of variables. Omitted variables will be omitted in the resulting table. Variables selected multiple times will get suffixes with ascending numbers (01, 02, 03...).

Output Table

The resulting table will be written in a text file just like the input table. You may enter a filename or browse the directories to specify a file.

3.3. Rename Variables

This functon allows you to rename a variable.

Applications:

This procedure may be used whenever you want to:

- Rename a variable

Options:

Input Table

This table is a text document.

Which variable should be renamed?

From all variables in the table, you may select one

New Name of the variable?

Enter a new name to a text box.

New name stored. Continue?

After renaming a variable, the new name is stored internally. You can then choose to rename other variables or save the table to a text file.

Output Table

The resulting table will be written in a text file just like the input table. You may enter a filename or browse the directories to specify a file.

Chapter 4. Merging Procedures

Merging procedures may be used to combine data from two different tables, both stored in textfiles, to one single table. Nogrod allows for two different ways of merging tables. First, you may add cases (rows) from another table to a working table which contains the same variables. Second, you may add variables (columns) from a key table to the working table by combining the cases over a common key variable.

4.1. Add Cases

If you want to combine two tables which contain (largely) the same variables you may use the merge procedure 'Add Cases'. You may open two tables simultaneously, select the variables which should be present in the resulting table and combine the two tables.

Note that you may combine tables with different variables. Some of them should, however, be present in both tables. For cases which come from a table which does not contain all variables of the output table, the values of the missing variables will be set to missing (blank cells).

Applications:

This procedure may be used whenever you want to:

- Combine fragments of the same dataset which were separated
- Combine two different datasets with the same variables
- Add some cases to an existing dataset which were stored somewhere else
- See Example: *Add cases from another dataset*

Options:

First Table

This table is a text document containing the first part of the table you wish to merge. In the end, the cases in this table will be at the top of the table, while the cases of the second table are at the end.

Second Table

This table is a text document containing the second part of the table you wish to merge. The second table should contain some variables which are present in the first table as well. If this is not the case, the script will work properly but the resulting table is hardly usable for further analysis as there will be many missing values.

Variables in the resulting table

From a list of variables which were present in either table, you may select any number of variables. Note that a variable which only occurs in one table will have missing values for all cases from the other table.

Order of the variables: In the resulting table, the order of the variables will be according to the order in the source tables. First, there are all variables which were present in the first table, then there will be all variables which were present in the second table only.

Hint: If you only want to add variables which are present in both tables, enter 'both' at the textbox above the list. The list will then be reduced to items which are present in both tables (unless you have a variable which contains the string 'both'. These will be shown as well. You may then press 'add all' to add all variables in the reduced list to the selection.

Output File

The resulting table will be written in a text file just like the source tables. You may enter a filename or browse the directories to specify a file.

4.2. Merge many tables with same variables

If you want to combine several tables which contain (largely) the same variables you may use the merge procedure 'Merge many tables'. In this procedure, you can combine whole folders of tables with the same variables.

Note that you may combine tables with different variables. For cases which come from a table which does not contain all variables of the output table, the values of the missing variables will be set to missing (blank cells).

Applications:

This procedure may be used whenever you want to:

- Combine split datasets
- Combine a list of different datasets with the same variables
- Merge data from different sources and the same variables

Options:

Directory

Choose the directory in which all tables are stored.

Select tables

From all files in this directory you may choose the tables you want to merge. Nogrod will then indicate in the text output the number and name of the variables in these tables.

Output File

The resulting table will be written in a text file just like the source tables. You may enter a filename or browse the directories to specify a file.

4.3. Add Variables

Another way to merge tables is the addition of variables to an existing table by combining the cases using one or several key variables. By this procedure information from a key table is inserted to a new variable of a main table. This may be used to add context information on recurring values of a variable in the key table or to add seperate measures for all cases of the main table which were stored in another table.

Applications:

This procedure is not suited to answer questions but it is really useful to reshape and combine existing datasets. The most common applications are:

- Combine two tables containing the same cases
- Combine two tables containing mutual variables but different cases
- Add additional information on specific values of a multinomial variable to a dataset
- Recode a variable according to special rules
- Attach dummy-variables produced with the above procedure to the source file (transform a multinomial variable to a set of dummy variables)
- See Example: *Add information on the actors to the table*

Options:

Main Table

This table contains the main dataset to which you wish to add information from a key table. At least one variable of this table has to contain values identical to one variable in the key table to perform matching. It may also contain several variables which may be used as key variables. The table does not have to be sorted.

Key Table

This table contains additional information which may be added to cases in the main table by means of a key variable. The key table may contain any number of variables of which you may add some or all to the main table. Neither this table nor the main table needs to be sorted.

Key variables in main table

You may select any number of key variables which are used to identify a case. Only cases with identical values on all key variables are matched. Cases for which no corresponding case exists in the key table are assigned missing values for all added variables.

Key variables in key table

As with the key variables in the main table you may select multiple variables. You must select the corresponding variables in the same order as the key variables in the main table. The key variables may be named differently in both tables.

If there are duplicates (i.e. cases with identical values on all key variables) in the key table, you will be warned. Only the last of the duplicates is matched with cases in the main table.

Variables to be added

When you have selected the key variables in both tables you may select any number of variables in the key table to be added to the main table. You may only add variables which are not among the key variables (as they are present in the main table anyway).

Output File

The resulting table will be written in a text file just like the source tables. You may enter a filename or browse the directories to specify a file.

Chapter 5. Subset Procedures

Subset procedures may be used to create a smaller table from an existing table, which does not contain all cases or variables. This may be useful for large tables or for tables with different categories of entries which should be separated prior to analysis.

5.1. Subset of cases

The first subset procedure allows you to select cases from a table, based on their values on certain variables. You may select any number of variables and any combination of values which should be present in the subset to be extracted. The dataset does not have to be sorted in any way prior to subset extraction.

Applications:

Especially with large datasets it may be required to select only some cases for further analysis. These cases may be defined by one or several variables. For the example dataset in this tutorial, automated subset extraction does not make much sense as there are only 19 cases to begin with, but for the sake of testing the procedure you may use it to:

- Select only shows which were issued on Fox News
- Select statements which contain an attack
- Select statements which were made by a democrat on March 14
- See Example: *Select statements of democrats containing an attack*

Options:

Input Table

This is the table which should be reduced to a subset. It should contain one or several multinomial variables defining groups of cases which may be extracted.

Grouping Variables

The grouping variables define the group which is to be extracted. You may select any number of variables wich are necessary to define the subset you wish to extract. The order of grouping variables is irrelevant for this method.

Grouping Values

From a list which contains all combinations of values of the selected grouping variables you may select the combinations which belong to the subset you wish to extract. You may select any number of combinations. Use the textbox above the list to filter the combinations.

Output File

The resulting table will be written in a text file just like the source tables. You may enter a filename or browse the directories to specify a file.

5.2. Split table

Like the subset of cases function, this function allows you to select cases from a table, based on their values on certain variables. Different from the subset function, however, new tables will be created for all different values. You may select any number of variables and any combination of values which should be present in the subset to be extracted. The dataset does not have to be sorted in any way prior to splitting.

Applications:

Especially with large datasets it may be required to split it to groups. You may use this function if you wish to:

- Generate multiple tables of subsets from a table.
- Make individual tables for male and female respondents
- Generate individual tables of content analysis data for each media outlet.

Options:

Input Table

This is the table which should be reduced to a subset. It should contain one or several multinomial variables defining groups of cases which may be extracted.

Grouping Variables

The grouping variables define the group which is to be extracted. You may select any number of variables wich are necessary to define the subset you wish to extract. The order of grouping variables is irrelevant for this method. Nogrod will indicate in the text output, how many groups there are and how many tables this will subsequentially generate.

Output File

The resulting tables will be written in text files just like the source table. You may enter a filename or browse the directories to specify a file. For each group, the filename will be completed with a suffix indicating the value of the group.

5.3. Random Subset of cases

The second function draws a random sample of cases from a list. You may select any size of the sample. The output table is not sorted in any way but contains the selected cases in a random sequence.

The randomized output of cases results in a sometimes handy side-effect of this function. You may randomize the order of cases in your dataset by drawing a random sample exactly the size of your dataset.

Applications:

In order to test and perform analyses which are very time-consuming, it may be required to divide large datasets to smaller random samples. This is especially the case for language processing or other expensive routines.

Options:

Input Table

This is the table which should be reduced to a subset. It should contain one or several multinomial variables defining groups of cases which may be extracted.

Number of Cases

You may enter any number of cases to be drawn from the table. If you enter a number which is higher than the total number of cases in the table, the only effect of this function will be to randomize the cases in your table.

Output File

The resulting table will be written in a text file just like the source tables. You may enter a filename or browse the directories to specify a file.

5.4. Subset of Variables

Another way for creating a subset from a table is the selection of certain variables. This function may prive useful for large datasets where only a small part of the variables is required in an investigation.

The usage of this procedure is quite straightforward, as you only open a file and select a number of variables to retain. Therefore, examples are omitted, here.

Options

Input Table

This is the table which should be reduced to a subset. It should contain several variables, some of which you wish to extract.

Selection of Variables

From a list containing all variables in the dataset you may select the variables you wish to extract to retain in the subset.

Output File

The resulting table will be written in a text file just like the source tables. You may enter a filename or browse the directories to specify a file.

Chapter 6. Reshaping Procedures

Reshaping procedures may be used to transform the shape of tables. The main use of these reshaping procedures is the transformation of a multinomial Variable to a series of dichotomous variables (Dummy Table) or the transformation of a dummy table to one multinomial variable providing the same information (Slim Table). By either transformation, all information is retained but visualized in a slightly different way.

6.1. Dummy Table

The aggregation of cases containing a multinomial variable may be done using the dummy table routine. Dummy tables are large contingency tables of two multinomial variables where one variable denominates the groups (rows) and the other one the elements occuring within groups (columns) the cells provide information on whether the element occurred in a specific group (dichotomous) or how many times it occurred in a specific group (count data).

The result of a dummy transformation is a new dataset which contains one case per group and one variable for each value of the multinomial variable defining the elements. This dataset may then be used for further data processing or analysis.

Applications:

Dummy tables provide answers to a specific kind of questions: "(How many times / Where) does this value occur in these groups?"

With regards to the example dataset you might answer questions such as:

- On which issues does each speaker make statements?
- On which dates were the different issues covered?
- See Example:
- *Which issues were covered in each of the TV* shows?
- See Example: *How many times was each actor cited on each station?*

Options:

Input Table

This is a table containing at least two multinomial variables. One signifies the groups which are to be considered, the other one holds N different values which should be transformed to N dichotomous variables.

Grouping Variable

This variable contains the names of the groups within which the values of the multinomial variable should be found or counted. In extreme cases, the grouping variable may contain exactly one value which will result in a dummy table containing only one row or it may have a different value for each case which will result in a dummy table the same length as the input table.

The former case may be used to create a frequency table for a variable. The latter case may be used to simply transform a variable to a series of dummies.

Multinomial Variable

The multinomial variable may take at least two values. The values of this variable will constitute the columns of the resulting dummy table.

Method for cell contents

You may choose different methods to calculate the values of the cells of the resulting dummy variable.

- **Dichotomous**:
  The cells of the resulting table may contain 1 (the value is present in the group) and 0 (the value is not present in the group.
- **Count**:
  The cells of the resulting table contain the exact number of times a value is present within each group.
- **Logarithmic**:
  The cells of the resulting table contain the logarithm of the number of times a value is present within each group. This method is especially useful for large counts. In order to prevent log(0)-errors, the count data is increased by 1 to calculate the logarithm. Thus, the value for 0 occurrences is log(1)=0.

Minimal number of elements per group

You may select the minimal number of elements each group has to contain. Groups containing less elements are removed from the dummy table. This means that the sum of values in each row is at least as high as the number you select here.

Note that the sum of a row containing dichotomous values is naturally lower than the sum of a row containing count data. Depending on the method you may have to adjust the minimal number of elements.

Minimal number of occurrences per value

You may select the minimal number of occurrences for each value of the multinomial variable to justify a dummy variable. Values occurring less than the specified number are removed from the table. This means that the sum of each column is at least as high as this number.

As the minimal number of elements per groups and the minimal number of value occurrences have to be fulfilled at the same time, sparse rows and columns are removed until both requirements are met.

Note that the sum of a column containing dichotomous values is naturally lower than the sum of a column containing count data. Depending on the method you may have to adjust the minimal number of elements.

Dummy Output

The resulting dummy table will be written in a text file tables. You may enter a filename or browse the directories to specify a file.

6.2. Reshape to slim table

The inverse function of a dummy transformation is the transformation of a series of dichotomous or count variables to one single multinomial variable. There are essentially two ways of reshaping a dummy table to a slim table. First, you may choose to interpret the values of the dummy table as counts and reproduce a table in which the multiniomial variable appears the respective number of times in each case. Zeroes in the dummy table will result in no information in the slim table. Second, you may choose to interpret the values of the dummy table as values to be transferred to the slim table as is. In that case, two variables are created in the slim table, one holding the name of the original dummy variable, the other holding its value.

For illustration purposes, take the first few cases of the Example_Data as a dummy table to be transformed. By regarding the values of

Table: Transformation example: The dummy table may (left) may be transformed by interpreting the values as counts (middle) or values (right). The table on the right is incomplete for formatting reasons as it would have 25 lines.

Actor	Argument_Moral	Argument_Economy	Argument_Voter	Argument_Justice	Argument_Faith
Obama, Barrack	1	1	0	0	0
Van Hollen, Chris	0	0	0	0	0
Ryan, Paul	0	0	0	1	0
Cruz, Ted	0	1	0	0	0
Christie, Chris	1	0	1	0	0

Actor	Argument
Obama, Barrack	Moral
Obama, Barrack	Econom
Ryan, Paul	Justice
Cruz, Ted	Econom
Christie, Chris	Moral
Christie, Chris	Voter

Actor	Argument	Value
Obama, Barrack	Moral	1
Obama, Barrack	Econom	1
Obama, Barrack	Voter	0
Obama, Barrack	Justice	0
Obama, Barrack	Faith	0
Van Hollen, C.	Moral	0
Van Hollen, C.	Econom	0
Van Hollen, C.	Voter	0
...	...	...

Options:

Input Table

This is a table containing at least two dummy variables with dichotomous or count data. Additional variables are of no consequence and will be copied to the output table if not used in the reshaping process.

Variables to be condensed

From all variables in the dataset you may select the ones which should be condensed to one multinomial variable.

Type of Reshaping

Select whether the values should be interpreted as count data or values to be copied to the output table. In the latter case, two variables will be created.

Output File

The resulting dummy table will be written in a text file tables. You may enter a filename or browse the directories to specify a file.

Chapter 7. Aggregation Procedures

Aggregation procedures allow you to decrease the size of tables by aggregating cases or variables according to fixed rules. For all aggregation procedures aggregating cases you need a group variable which indicates the groups to which the cases are to be collapsed. You may then select the variables to be aggregated and the function to be used.

For the aggregation of values you may select a series of variables which are to be combined to one outcome variable. A small set of functions (e.g. mean score, sum, concatenation) may be used to aggregate the variables.

7.1. Aggregate Cases

The aggregation of cases to groups may be used when several variables have to be aggregated using a common rule (e.g. Sum of values). The cases will be combined to groups specified by the group variable and will contain one value for each variable selected for aggregation. This value is computed from the single values of all cases within the group by a rule you may select.

This data transformation is especially useful when handling relational datasets from content analyses where you may have several cases in one table which have to be aggregated to combine them with another table containing group specifics. In the example dataset you have several actors per news story. If you wish to combine this data to another table containing additional information on the news stories, you may aggregate the dataset first with the news story as a group and then merge the resulting dataset with another table.

You may also use this procedure to get mean scores or sums of values for certain groups without using statistical software. You may also use this procedure to reshape a dataset by copying multiple units of analysis next to each other on the same line in the dataset.

Applications

- See Example: *Which were the most frequent actors and issues on each station?*
- See Example: *Reshape the dataset to one show per line*

Options:

Input Table

This file contains a table with at least two variables. One variable defines groups and the other contains values which may be summed up or otherwise aggregated within these groups.

Grouping Variables

From a list containing all variables in the dataset you may select any number of variables defining the groups you wish to aggregate the dataset on. If you select more than one variable, each unique combination of values of these variables defines one group.

Variables to be aggregated

From a list containing all variables except for the grouping variables, you may select any number of variables which should be aggregated for each group.

Method for aggregation

Depending on the type of variable which is to be aggregated, you may select one of the following methods to calculate one value per variable per group. Please note that Sum, Mean, Maximum and Minimum are possible only for numerical data. The result of these methods is always a floating number.

- **Sum**: For each variable selected, the sum will be calculated for each group. Missing values are ignored.
- **Mean**: For each variable selected, the mean value will be calculated for each group. Missing values are ignored (i.e: mean(1,2, MISSING ,3) =2)
- **Maximum**: For each variable selected, the highest value will be selected for each group.
- **Minimim**: For each variable selected, the lowest value will be selected for each group.
- **Most Frequent**: For each variable selected, the most frequent value will be selected for each group. If there are two most frequent values, the choice will be the value which is lower (either numerically or in the alphabet)
- **First Value**: For each variable selected, the first value in the group within the dataset is selected.
- **Last Value**: For each variable selected, the first value in the group within the dataset is selected.
- **Transform Dataset to broad format**: Each variable selected will be multiplied to copy all values completely to one line in the new dataset. The number each variable is copied depends on the highest number of cases per group in the input dataset. When aggregating the example data in this way to get one show per line, each variable would be copied four times. See example

Note that you may only select one method for all variables selected. If you want to use one method for a set of variables and another method for another set, please do this separately and use the merge routine to combine the resulting datasets.

Output File

The resulting table will be written in a text file tables. You may enter a filename or browse the directories to specify a file.

7.2. Aggregate Variables (Calculation)

The aggregation of variables is a quick way to generate new variables from existing ones using five standard methods (sum, mean, min, max, and concatenate). You may use this procedure to create new variables without using statistical software packages or copy&pasting the data to Excel.

The range of the easy calculation proceture is deliberately limited to the five basic methods to keep it simple. For more sophisticated methods, you will have to use calculation programs (such as Excel) or statistical software.

Applications:

Easy calculation may be useful in cases where you need the sum or mean value of a certain variable for additional analyses. In these cases it may save time to perform the calculations directly in Nogrod instead of importing and exporting the data to an other program.

In the example data there is only a small number of numerical variables to use:

- Was there any arugment present?
- How many arguments are there in the statement?
- See Example: *How many arguments are there in the statement?*
- See Example: *Reshape* the dataset to one show per line
- *Here, the data is summarized to* the level of shows and all variables are aggregated by reshaping them to a broad view. For demonstration purposes and graphic limits, only three of the variables (Actor, Actor Affiliation, and Issue) are used.
  1. Select 'Aggregate Data' -> 'Aggregate Cases' as a procedure and load the example data.
  2. Select 'Show_ID' as the group variable.
  3. Add Actor, Actor_Affiliation, and Issue as variables to be aggregated. Select 'Transform Dataset to broad format' as method.

4. Choose an output file and confirm.

- The result is quite a broad table as all variables have been multiplied four times to make space for all data which was aggregated. The resulting table contains four copies of each variable as there is a maximum of four speakers in the shows of the example data. The aggregated lines are added in a chronological order, so Actor01 of AB001 is Ryan, followed by Kerry, Cruz, and Van Hollen. For Groups with less than four cases, missing values are entered to unused variables. As FN003 only has one actor, this case has nine missing values which might be used to specify actor 02, 03, and 04.

Reshape the dataset to one show per line
Here, the data is summarized to the level of shows and all variables are aggregated by reshaping them to a broad view. For demonstration purposes and graphic limits, only three of the variables (Actor, Actor Affiliation, and Issue) are used.
Select 'Aggregate Data' -> 'Aggregate Cases' as a procedure and load the example data.

5. Select 'Show_ID' as the group variable.
6. Add Actor, Actor_Affiliation, and Issue as variables to be aggregated. Select 'Transform Dataset to broad format' as method.
7. Choose an output file and confirm.

The result is quite a broad table as all variables have been multiplied four times to make space for all data which was aggregated. The resulting table contains four copies of each variable as there is a maximum of four speakers in the shows of the example data. The aggregated lines are added in a chronological order, so Actor01 of AB001 is Ryan, followed by Kerry, Cruz, and Van Hollen. For Groups with less than four cases, missing values are entered to unused variables. As FN003 only has one actor, this case has nine missing values which might be used to specify actor 02, 03, and 04.

Table 12: Result of the aggregation to broad format. The table is cut in two parts as it is too broad to be displayed on one line.

Show_ID	Actor01	Actor_ Affiliation01	Issue01	Actor02	Actor_ Affiliation02	Issue02
AB001	Ryan, Paul	Rep	Obamacare	Kerry, John	Dem	Obamacare
AB002	Obama, Barrack	Dem	Iran	Kerry, John	Dem	Iran
FN001	Obama, Barrack	Dem	Budget	Van Hollen, Chris	Dem	Budget
FN002	Christie, Chris	Rep	Obamacare	Ryan, Paul	Rep	Budget
FN003	Christie, Chris	Rep	Iran			
PB001	Van Hollen, Chris	Dem	Budget			
PB002	Obama, Barrack	Dem	Iran	Cruz, Ted	Rep	Iran
PB003	Van Hollen, Chris	Dem	Budget	Obama, Barrack	Dem	Obamacare

Actor03	Actor_ Affiliation03	Issue03	Actor04	Actor_ Affiliation04	Issue04
Cruz, Ted	Rep	Obamacare	Van Hollen, Chris	Dem	Budget

Ryan, Paul	Rep	Budget	Cruz, Ted	Rep	Obamacare
Kerry, John	Dem	Budget			

Which arguments were seen on which show?

Options:

Input Table

This file contains a table with more than one variable.

Variables to be aggregated

From a list of all variables in the dataset you may select any number of variables you wish to aggregate to one variable.

Method for calculation

Depending on the type of the variables you wish to aggregate you may select different methods for aggregation.

- **Sum of all values**: Calculates the sum of numerical variables
- **Mean of all values**: Calculates the mean of numerical variables
- **Maximum value**: Uses the highest value of the numerical variables
- **Minimum value**: Uses the lowest value of the numerical variables
- **Concatenate**: Paste the values together in one large string

The contatenation is the only method which may be used on string variables. If one of the variables contains string values in one of the other methods, these values are treated as missing values. In the calculation of sums, missing values are treated as 0. In the calculation of means, missing values are ignored, not adding to the number of values to calculate the mean of.

Name of output variable

The aggregation of variables results in an additional variable containing the results of the calculation. You may enter a name for this variable. It will be put at the end of the table.

Output File

The resulting table with the new variable will be written in a text file tables. You may enter a filename or browse the directories to specify a file. The table is identical (complete and in order) to the input table except for the additional variable at the end of the dataset.

7.3. Calculate Entropy within groups

A special way to aggregate the cases in your data for one multinomial variable is the calculation of the group entropy, meaning the diversity of values on a given variable within a group. Calculating the group entropy will result in one value for the variable selected.

This method may be used to determine the diversity of groups in your data, which may be useful for content analyses addressing the diversity (or focusedness) of media content.

As with dummy tables you may specify one variable which defines the groups and one multinomial variable which is used to determine the entropy.

Calculation:

The entropy of a group is calculated using this formula, where N is the number of unique elements in the group and M is the number of unique elements in the population. p_i is the proportion of element i in the group.

$$Entropy = -\sum_{i}^{N} p_i * log_M(p_i)$$

If we assume two different elements within a group where the first appears twice (p_1=0.66) and the second appears once (p_2=0.33), the entropy would be:

$$\boldsymbol{Entopy} = 0.66 * log_2(0.66) + 0.33 * log_2(0.33) = \mathbf{0.919}$$

The Entropy may assume any value between 0 (perfect uniformity) and 1 (perfect diversity).

Applications:

- See Example: *Actor Diversity across networks*
- See Example: *Diversity of the networks with respect to actor affiliation*

Options:

Input Table

This file contains a table with a grouping variable and a multinomial variable for which the entropy within each group is to be calculated.

Grouping Variable

From a list containing all variables in the dataset you may select the variable defining the groups. In order to be considered a group for the calculation of entropy, at least two cases have to belong to it. Groups consisting of only one element are assigned a missing value for entropy.

Multinomial Variable

From a list containing all variables but the grouping variable you may select the variable for which you want to calculate the entropy of values within each group. This variable should have at least two different values and there should be cases with the same value on this variable to result in useful output.

List of Reference

When calculating the entropy within a group, there are two different ways for this calculation. On one hand, you may calculate the entropy of the elements present in this group without considering other groups. On the other hand, you may see the elements within a group as a selection from a more extensive universe of elements. This selection itself decreases the entropy.

- **All elements across all groups**: This option calculates the entropy for each group using all possible values of the variable as population. This means that the population of possible values is the same for all groups and the entropy values are typically lower as no group is likely to contain all values.
- **Only the elements within each group (M==N)**: This option calculates the entropy for each group based on the elements which are present in this group. A group containing two different values has entropy of 1.0 as the values are equally distributed. Even if there are ten different values in another group.

Output file

The resulting table with the new variable will be written in a text file tables. You may enter a filename or browse the directories to specify a file.

Chapter 8. Recoding Procedures

Recoding procedures may be used to recode single variables accoriding to simple patterns. In this version of Nogrod, there are two different transformation procedures which may prove useful. First, there is a function to transform timestamps from one standardized format to another. Second, there is a transformation function which allows the grouping of scalar variables.

8.1. Transform Timestamps

The transformation of timestamps may be used to change the format of timestamps stored in a table to another format. For example, you may use them to recode German dates to American dates.

The function is straightaway and only asks you for an input variable which contains timestamps and an output variable in which the new timestamps should be written. Then you may select the input and output format of the timestamps.

Strptime patterns

Time formats for in- and output may be defined as strptime-patterns. These patterns are strings in which each component of a date or time are signified by placeholders. The placeholders are:

Sign	**Description**	**Example Output**
%a	Weekday as locale's abbreviated name.	Sun, Mon, …, Sat (en_US)
		So, Mo, …, Sa (de_DE)
%A	Weekday as locale's full name.	Sunday, Monday, …, Saturday (en_US)
		Sonntag, Montag, …, Samstag (de_DE)
%w	Weekday as a decimal number, where 0 is Sunday and 6 is Saturday.	0, 1, 2, 3, 4, 5, 6
%d	Day of the month as a zero-padded decimal number.	01, 02, …, 31
%b	Month as locale's abbreviated name.	Jan, Feb, …, Dec (en_US)
		Jan, Feb, …, Dez (de_DE)
%B	Month as locale's full name.	January, February, …, December (en_US)
		Januar, Februar, …, Dezember (de_DE)
%m	Month as a zero-padded decimal number.	01, 02 … 12
%y	Year without century as a zero-padded decimal number.	01, 02, … 99
%Y	Year with century as a decimal number.	0001, 0002, … , 9999
%H	Hour (24-hour clock) as a zero-padded decimal number.	01, 02, … , 23
%I	Hour (12-hour clock) as a zero-padded decimal number.	01, 02, … , 12

%p	Locale's equivalent of either AM or PM.	AM, PM (en_US)
		am, pm (de_DE)
%M	Minute as a zero-padded decimal number.	01, 02, … , 59
%S	Second as a zero-padded decimal number.	01, 02, … , 59
%f	Microsecond as a decimal number, zero-padded on the left.	000000, 000001, …, 999999
		Not applicable with time module.
%z	UTC offset in the form ±HHMM[SS] (empty string if the object is naive).	(empty), +0000, -0400, +1030
%Z	Time zone name (empty string if the object is naive).	(empty), UTC, IST, CST
%j	Day of the year as a zero-padded decimal number.	001, 002, …, 366
%U	Week number of the year (Sunday as the first day of the week) as a zero padded decimal number.	00, 01, …, 53
	All days in a new year preceding the first Sunday are considered to be in week 0.	
%W	Week number of the year (Monday as the first day of the week) as a decimal number.	00, 01, …, 53
	All days in a new year preceding the first Monday are considered to be in week 0.	
%c	Locale's appropriate date and time representation.	Tue Aug 16 21:30:00 1988 (en_US)
		Di 16 Aug 21:30:00 1988 (de_DE)
%x	Locale's appropriate date representation.	08/16/88 (None)
		08/16/1988 (en_US)
		16.08.1988 (de_DE)
%X	Locale's appropriate time representation.	21:30:00 (en_US)
		21:30:00 (de_DE)
%%	A literal '%' character.	%

Options:

Input Table

This file contains a table with one variable containing timestamps in any format which is to be converted to another format.

Input Variable

From a list of all variables in the dataset you may select the variable containing timestamps.

Output Variable

As a new variable is created by the transformation, you may specify the name of the new variable by entering it manually.

Format of input variable

There is a set of standardized timestamps from which you may select the timestamp format of the input variable.

- **Python Timestamp (string)**: This format is created by the time.ctime() function in python and has the form "Weekday Month Day Time Year" (e.g.: "Fri Mar 14 10:41:57 2014"). This timestamp may be found in Angrist outputs. Corresponds to the strptime string: `"%a %b %d %H:%M:%S %Y"`.
- **Python Timestamp (seconds)**: This format is created by the time.time() function in python and indicates the seconds which have passed since January 1 in the year 1970. This format is an international standard and may be found in website statistics and timestamps from various programs (e.g.: 1394790117.49).
- **Excel Timestamp (days)**: This format is used as an internal representation of dates in MS Excel and indicates the number of days which have passed since January 1, 1900. The time of the day is indicated by decimals (e.g.: 41712.40414).
- **German:** This format is the usual date format in middle Europe. It has the form: dd.mm.yyyy (e.g.: 14.03.2014). This corresponts to the strptime string: `"%m.%d.%Y"`
- **Open Format:** In addition to the predefined formats, you may enter any other format as a strptime pattern. Available patterns are explained above.

Format of output variable

Just like the input format you may also specify the output format. When both formats are selected, the text output at the right hand side of the interface shows an example transformation.

For the output format, one additional format is available:

- **Time of day**: The time of day has the form: hh:mm:ss (e.g.: 10:41:57). This corresponds to the strptime string: `"%H:%M:%S"`

Variable level

If the output format is numerical you may choose whether the timestamp is to be stored as integer or floating point number. If you choose integer values, the manner of transformation may be chosen. The integer may be created by cutting of the decimals (rounding down) or by algebraic rounding where 0.5 is rounded up to the next integer.

Cutting off decimals may be useful for the excel timestamp where the integer names the day and the decimals indicate the time of day. Here, algebraic rounding would transform 1pm on August 1 to August 2 which may not be intended.

Output File

The resulting table with the new variable will be written in a text file tables. You may enter a filename or browse the directories to specify a file.

8.2. Transform Scale to Groups

Another transforming function may be used to divide continuous scale variables to group variables. This may prove useful to partition timestamps in longitudinal data to phases or to partition numeric variables to quantiles.

Applications:

- Grouping of high-frequency timestamps to intervals
- Splitting normally distributed variables to quantiles
- Detecting confidence intervals and tails of distributions
- Detecting peaks in time series data

Options:

Input Table

This file contains a table with at least one scalar variable.

Scale Variable

From a list of all variables in the dataset you may choose the variable you wish to divide to groups. The variable should be metric and contain more different values than the groups you wish to create. Missing values result in missing group identification.

Output Variable

The transformation results in a new variable. You may enter any name for this variable.

Group Mode

There are different methods you may use to divide the scale variable. The optimal method depends on the type of the variable and the purpose of grouping (see Figure 7).

- **Equally sized groups**: This method divides the continuous variable to equally sized groups (quantiles). You may select four different options for this method:
    - Median: Two groups with equal amounts of cases will be produced
    - Tertiles: Three groups with equal amounts of cases will be produced
    - Quartiles: Four groups with equal amounts of cases will be produced
    - Quintiles: Five groups with equal amounts of cases will be produced
- **Equal steps between groups**: This method divides the continuous variable to groups with equal distance. You may set the steps manually by entering an integer or decimal value.
    - **Size of step:** You may enter the size of the steps manually. The steps may be entered as integer or floating point numbers.
- **Tails of the distribution**: This method divides the continuous variable to three steps, labeled -1, 0 and 1. The -1 group contains the lower tail of the distribution. The 1 group contains the upper tail of the distribution. The 0 group contains all values between the tails (the confidence interval). You may choose the size of the confidence interval.
    - **Size of the tails:** You may choose the size of the confidence interval and thus the size of the tails. The confidence interval and tails are always two-sided. This means that a confidence interval of 95% results in two tails each hilding 2.5% of the values.

Output file

The file in which the table with the new variable is written. The table corresponds ot the input table with one variable added at the end of the dataset.

Labels of the groups:

The labels of the groups depends on the method used for grouping.

- For quantiles the groups are numbered beginning with 1.
- For equal steps the groups are labeled with the lowest value within the group.

- For tails, the tails are labeled -1 and 1 while the confidence interval is labeled 0.

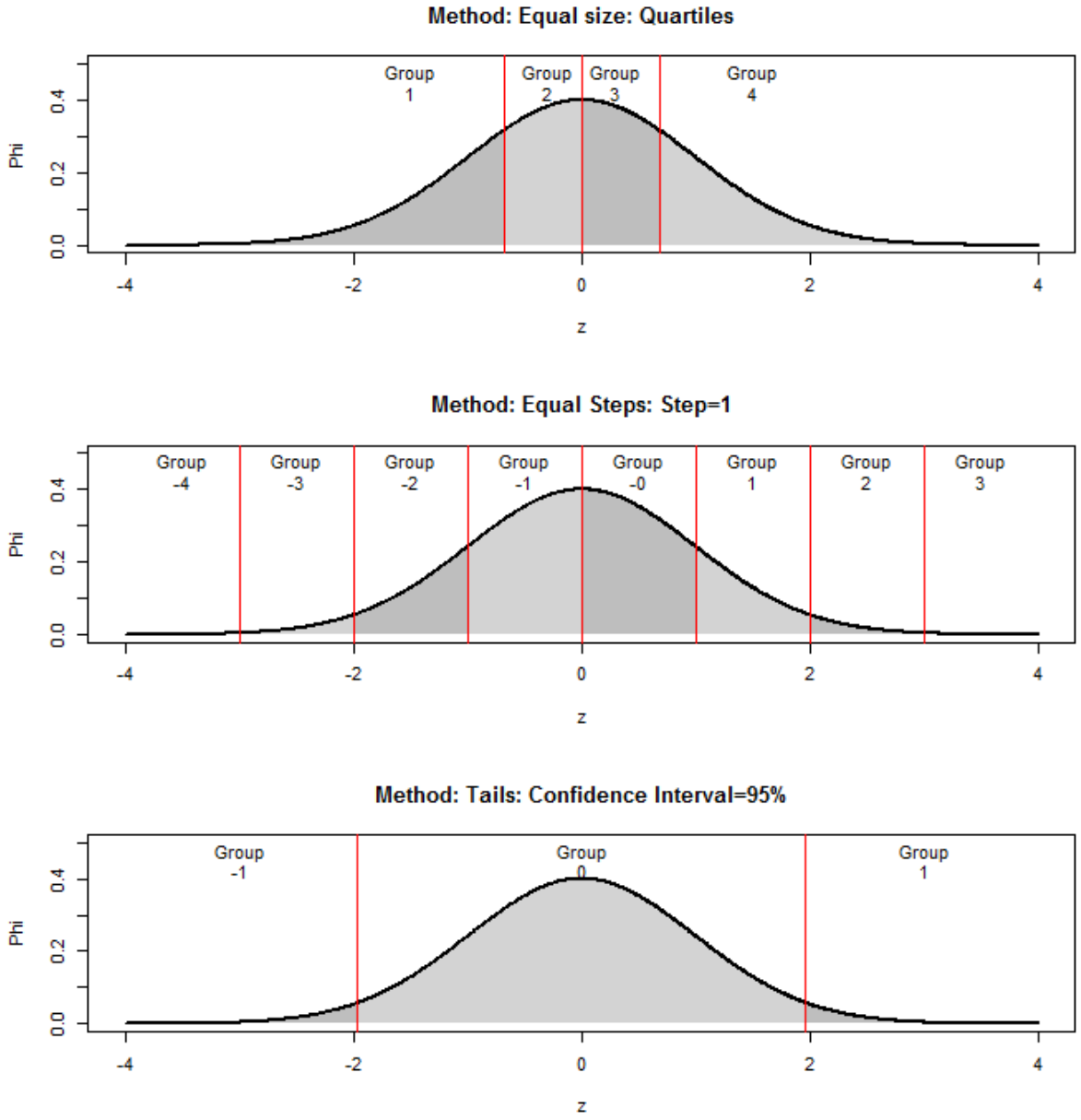


Figure 7: Visualization of the methods for grouping continuous variables. In this example, a normally distributed variable (standardized) was used. Note the labelling scheme for the different types of groups. For quantiles the groups are numbered, for equal steps the groups are labeled with the lowest value of the group. For tails, the tails are labeled -1 and 1 while the confidence interval is labeled 0.

Chapter 9. Co-Occurrence Analysis

Similar to dummy transformations, co-occurrence analyses are used to investigate the occurrence of values of a multinomial variable within groups defined by a second variable. Different from dummy transformations, however, the result is an NxN-Matrix providing information on how many times (or with which probability) two different values appear within the same group.

The starting point for co-occurrence analyses may be two multinomial variables as for dummy transformations or an already existing dummy table.

9.1. Co-Occurrence analysis from multinomial variables

Co-occurrence analyses of a multinomial variable require two different variables. First, you need to specify the group variable which contains the names of the groups. Then you have to indicate the multinomial variable whose values are tested for co-occurrence. You may then select different methods of how to express the co-occurrence of values within groups.

Applications:

Co-occurrrence analyses may be used to answer questions of the form: "(How frequently) do two values of this variable co-occur within groups defined by another variable?"

Using the example data, one might ask:

- How many times do the actors appear in the same issue of a TV show?
- How probable is the discussion of different issues on the same day?
- How likely is it that different arguments appear within the same statement of an actor?
- See Example: *How probable is the discussion of certain different issues on the same show?*

Options:

Input Table

The input table is a file which contains a table with at least two variables. As in dummy transformation, a grouping variable and a multinomial variable are required.

Group Variable

From a list containing all variables in the dataset you may choose the variable which defines the groups within which the co-occurrence of elements should be analyzed.

Multinomial Variable

From a list containing all variables in the dataset you may choose the variable which holds the elements that should be analyzed.

Method for calculating co-occurrence

For co-occurrence analyses a set of methods is available. These methods may be used to quantify the co-occurrence as measures of either closeness or distance:

- **Dichotomous**: The cells contain either 1 (these values co-occur in any group) and 0 (these values never co-occur)
- **Count**: The cells contain the exact number of groups the values co-occur in.

- **Probability of co-occurrence**: The cells contain the probability of co-occurrence measured as the product of the conditional probabilities of value A occurring in groups where B is presend and vice versa (P(A|B)*P(B|A)).
- **Percent Agreement**: The cells contain the number of groups where either both values are present or absent. Caution: For rare values this value is very close to 1 as they both don't appear in most groups.
- **Inverse Probability** of co-occurrence: The cells contain the negative logarithm of the probability of co-occurrence. This results in a measure of distance ranging from 0 (complete co-occurrence) to very large numbers. In order to prevent log(0)-errors, no co-occurrence is changed to 1 instance of co-occurrence in a dataset with twice the number of groups than present in the current data.
- **Sokal-Distance**: The cells contain the sokal-distance of two values which ranges from 0 to 1. 0 means perfect agreement, 1 means no co-occurrence.
- **Euclid Distance**: The cells contain the euclidian distance of two values.

Minimal number of elements per group

You may select the minimal number of elements each group has to contain. Groups containing less elements are removed from the dummy table. This means that the sum of values in each row is at least as high as the number you select here.

Note that the sum of a row containing dichotomous values is naturally lower than the sum of a row containing count data. Depending on the method you may have to adjust the minimal number of elements.

Minimal number of occurrences per value

You may select the minimal number of occurrences for each value of the multinomial variable to justify a dummy variable. Values occurring less than the specified number are removed from the table. This means that the sum of each column is at least as high as this number.

As the minimal number of elements per groups and the minimal number of value occurrences have to be fulfilled at the same time, sparse rows and columns are removed until both requirements are met.

Note that the sum of a column containing dichotomous values is naturally lower than the sum of a column containing count data. Depending on the method you may have to adjust the minimal number of elements.

Smallest Space Analysis

When choosing a method resulting in a distance measure (inverse probability, Sokal, Euclid), an additional output is possible. You may choose to export an R-script which calculates and displays the results of a Smallest Space Analysis (SSA) of the elements. In this analysis, the distances between each pair of elements is rescaled to two dimensions and visualized in a scatter plot.

The R-script may be executed in the statistical software project R using the package 'smacof'.

Output File

The resulting table with the new variable will be written in a text file tables. You may enter a filename or browse the directories to specify a file.

9.2. Co-Occurrence analysis from Dummy-tables

As an input for co-occurrence analyses you may also use pre-existing dummy tables. The co-occurrence of true values (1) within the cases is calculated if you choose to use this procedure. All methods are equal to the co-occurrence analysis using a multinomial variable.

Applications:

- See Example: *Which arguments are often mentioned together?*

Options:

Input Table

The input table is a file which contains a table with dummy variables.

Dummy Variables

From a list containing all variables in the dataset you may choose the variables which contain the dummy values. Since the dummies are regarded as dichotomous variables (present or absent in this case) all types of variables are transformed to dichotomous variables. For numerical values, zeroes and missing values are considered 'absent' while numbers are regarded as 'present'. For string variables, any value other than empty strings and missing values are considered 'present' while all missing values are 'absent.

Invariate variables may not be chosen.

Method for calculating co-occurrence

For co-occurrence analyses a set of methods is available. These methods may be used to quantify the co-occurrence as measures of either closeness or distance:

- **Dichotomous**: The cells contain either 1 (these values co-occur in any group) and 0 (these values never co-occur)
- **Count**: The cells contain the exact number of groups the values co-occur in.
- **Probability of co-occurrence**: The cells contain the probability of co-occurrence measured as the product of the conditional probabilities of value A occurring in groups where B is presend and vice versa (P(A|B)*P(B|A)).
- **Percent Agreement**: The cells contain the number of groups where either both values are present or absent. Caution: For rare values this value is very close to 1 as they both don't appear in most groups.
- **Inverse Probability** of co-occurrence: The cells contain the negative logarithm of the probability of co-occurrence. This results in a measure of distance ranging from 0 (complete co-occurrence) to very large numbers. In order to prevent log(0)-errors, no co-occurrence is changed to 1 instance of co-occurrence in a dataset with twice the number of groups than present in the current data.
- **Sokal-Distance**: The cells contain the sokal-distance of two values which ranges from 0 to 1. 0 means perfect agreement, 1 means no co-occurrence.
- **Euclid Distance**: The cells contain the euclidian distance of two values.

Smallest Space Analysis

When choosing a method resulting in a distance measure (inverse probability, Sokal, Euclid), an additional output is possible. You may choose to export an R-script which calculates and displays the results of a Smallest Space Analysis (SSA) of the elements. In this analysis, the distances between each pair of elements is rescaled to two dimensions and visualized in a scatter plot.

The R-script may be executed in the statistical software project R using the package 'smacof'.

Output File

The resulting table with the new variable will be written in a text file tables. You may enter a filename or browse the directories to specify a file.

Chapter 10. Social Network Visualization

Similar to tables created in dummy-transformations, Nogrod may create source files for social network visualizations in Visone (www.visone.info). These source files may then be opened and analyzed in Visone. The creation of these files is especially useful to illustrate the relation between variables.

For a social network structure, three sources of information are required. First, the subjects (nodes from which edges issue) and objects (nodes to which the edges point) have to be defined. Each of them may be stored in a multinomial variable. Second, there has to be some relation between the subjects and objects which may either be provided by a third variable indicating the relation or just be inferred from the number of times the subjects and objects co-occur within a case.

10.1. Social Network Visualization from two variables

In the standard case you have two variables (similar to dummy and co-occurrence analyses) which you would like to visualize as subjects and objects of a social network. A thir variable may be used to quantify or specify the relation between these variables.

Applications:

Social network visualizations answer questions of the type: "What kind of relation is there between values of different variables in this data?"

Based on the example dataset, one might ask:

- Who attacks whom?
- Who talks about which issues?
- Which shows cover which issues?

Options:

Input file

The input file contains a table with at least two multinomial variables defining the subjects and objects of the social network. A third variable may be used to quantify/identify the relation between subjects and objects.

Subject Variable

From a list containing all variables of the dataset you may choose the variable which holds the subject identifiers. In the social network the subjects are the nodes from which arrows issue.

Object Variable

From a list containing all variables of the dataset you may choose the variable which holds the object identifiers. In the social network the objects are the nodes toward which arrows point.

Identical values for subjects and objects

Choose whether the subjects and objects of the social network are coded by the same rules. If you choose 'yes' on this option, the subjects and objects are treated as one group of entities instead of two separate groups. This does not influence the way the network is visualized.

Relation variable

From a list containint all variables in the dataset you may choose the variable which defines the relation between subjects and objects. The relation may be numeric (e.g.: valence or count) or multinomial/string (e.g.: Type of relation).

Method for relation

The relation between subjects and objects may be calculated from the mere number of co-occurrence or from a third variable defining the relation.

- **Count**: The relation is the number of times a subject co-occurs with an object.
- **Inverse Count**: The number of times a subject and object co-occur (N) is inverted by the formula: 9/(1+N) which results in a distance of 9 for no co-occurrence and low distances for frequent co-occurrence.
- **Dichotomous**: The relation may only take the values 1 (present) and 0 (absent)
- **Type**: For multinominal relation variables, this method uses the last value of the relation variable for each pair as a value of their relation.
- **Mean Value**: Only if the relation variable is metric, this method will take the mean value of the relation variable for each pair as their relation.
- **Sum**: For metric relation variables only, this method calculates the sum of all relations.

Minimal occurrence

You may choose a minimal number of times a subject or object has to be present in the data to be counted in the analysis. Subjects and objects which occur less than the specified number of times are excluded from the visualization.

Output file

The file in which the adjacency matrix should be written. This file may be opened in Visone afterwards.

Output of node counts

The file in which a list of all nodes (subjects and objects) is written. For each node the number of occurrences is written in this output. You may merge this file with additional information on the nodes and import them in Visone.

Optimize the layout of your network

Even with the labels the network still looks bare and far from appealing. The nodes and links are of a drakish grey shade and much too small to contain the whole labels. It is therefore reasonable to add some cosmetics.

To change the appearance of the nodes you first select 'select all' from the menu 'nodes'. Then you pick 'properties..' from the same menu. You now see a small dialog with three tabs (general, label, attributes) with which you may change the appearance of the nodes. In the tab general you may change the shape of the nodes to rounded rectangles with a larger width to engulf the whole labels and a lighter color. In the tab 'label' you may then change the font and size of the labels. The tab 'attributes' is useless when all nodes are selected. When only one node is selected you may see all attributes of this node in that tab.

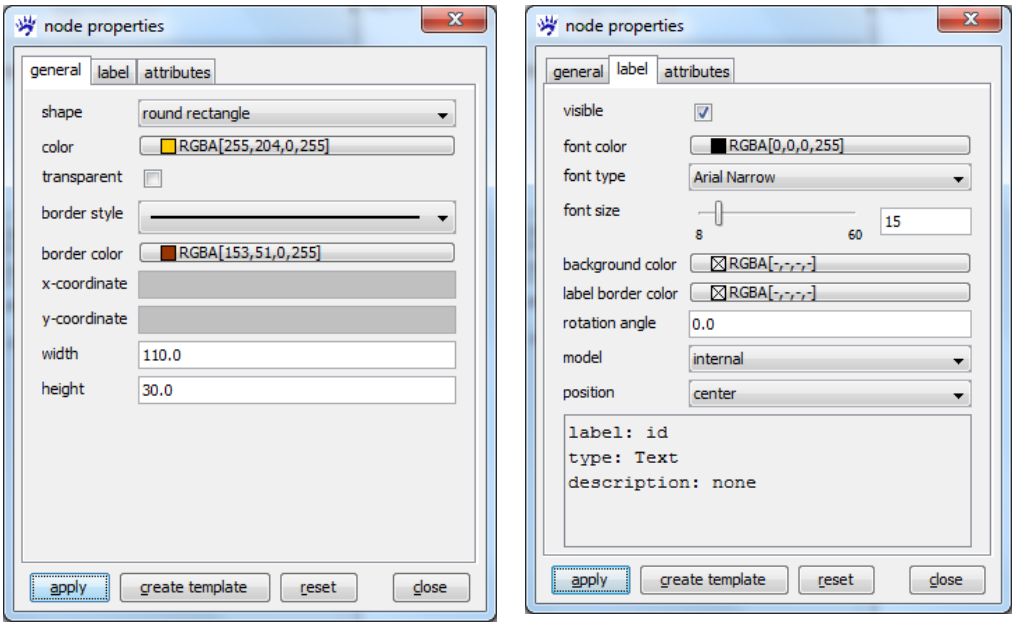


Figure 8: Node properties

Choosing the options as shown in figure 9, the network has a brighter and more friendly appearance (Figure 9). You may also change the shape and color of the links (arrows) with a similar properties dialog.

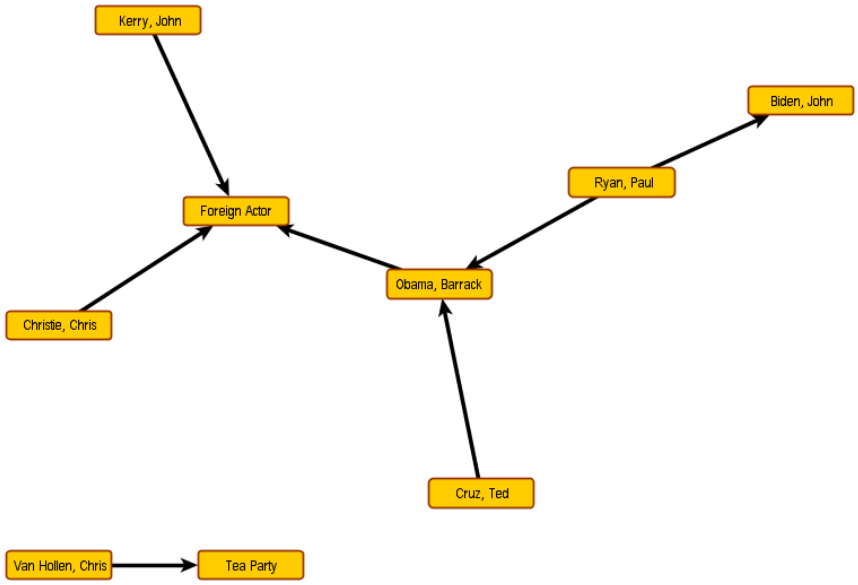


Figure 9: Appearance of the network after changing node and link properties.

If required you may also change the size or colour of nodes and the width or color of links according to their values. Remember that the number of times each node was present in the data is stored as an attribute 'Count' in this network.

To change the layout according to attributes, you may select the tab 'visualization' on the left-hand side of the visone window. In this tab you may change the layout and appearance automatically.

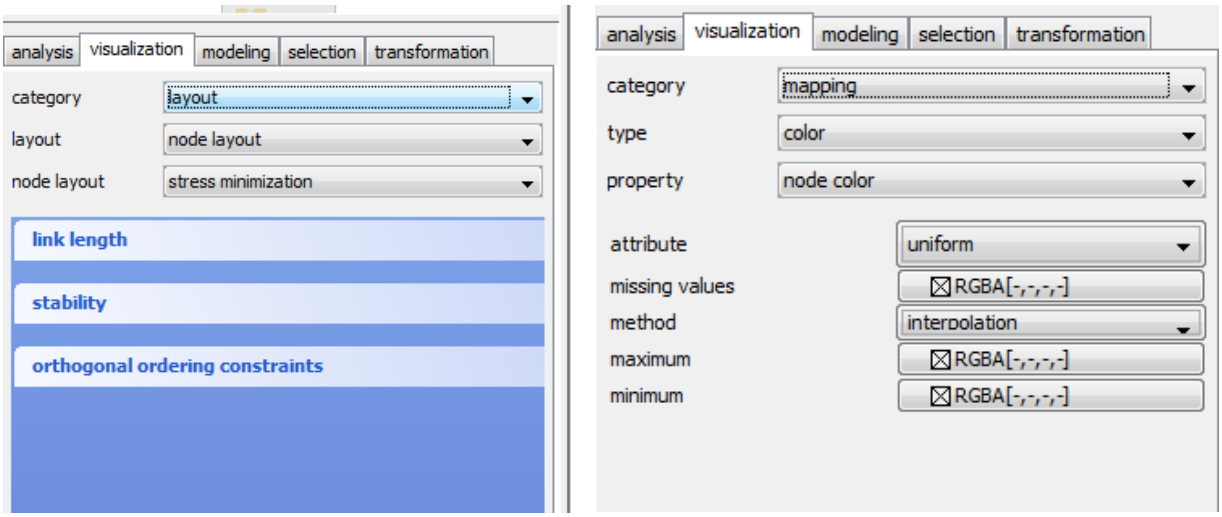
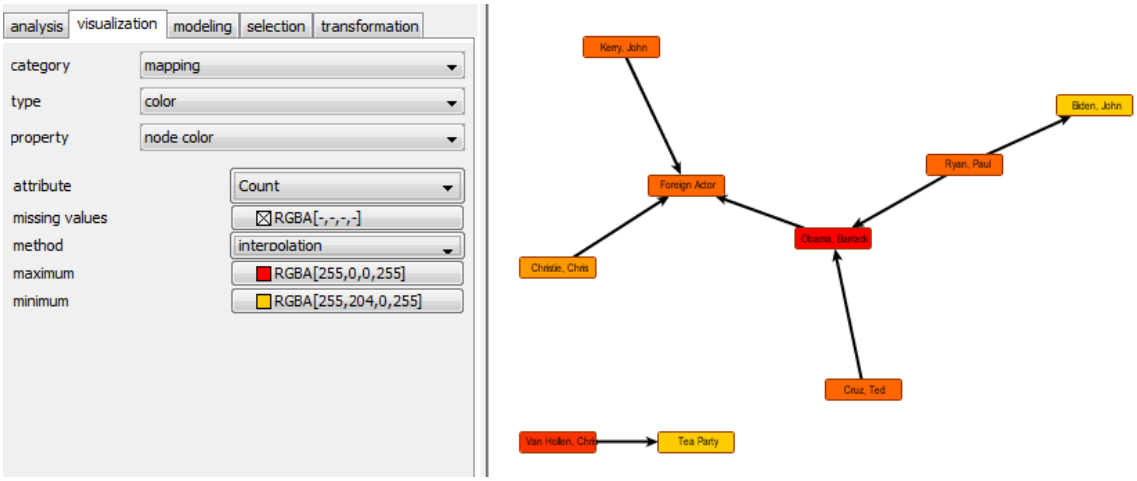


Figure 10: Visualization tab.

First, select 'mapping' from the dropdown menu 'category' the tab now changes its appearance and you may enter different parameters. Assume that wa want to change the colour of the nodes according to the number of times they appear in the data. Frequent nodes should be reddish, infrequent ones yellow (Figure 11a). Likewise, you may want to change the height of nodes according to their frequency (Figure 11b). You may also change the width of the links to the number of times an attack was present. This was stored in the adjacency matrix and has been imported to the network as 'csv value' (Figure 11c). In the current network, this has no effect, however, as all attacks are only present once.

a



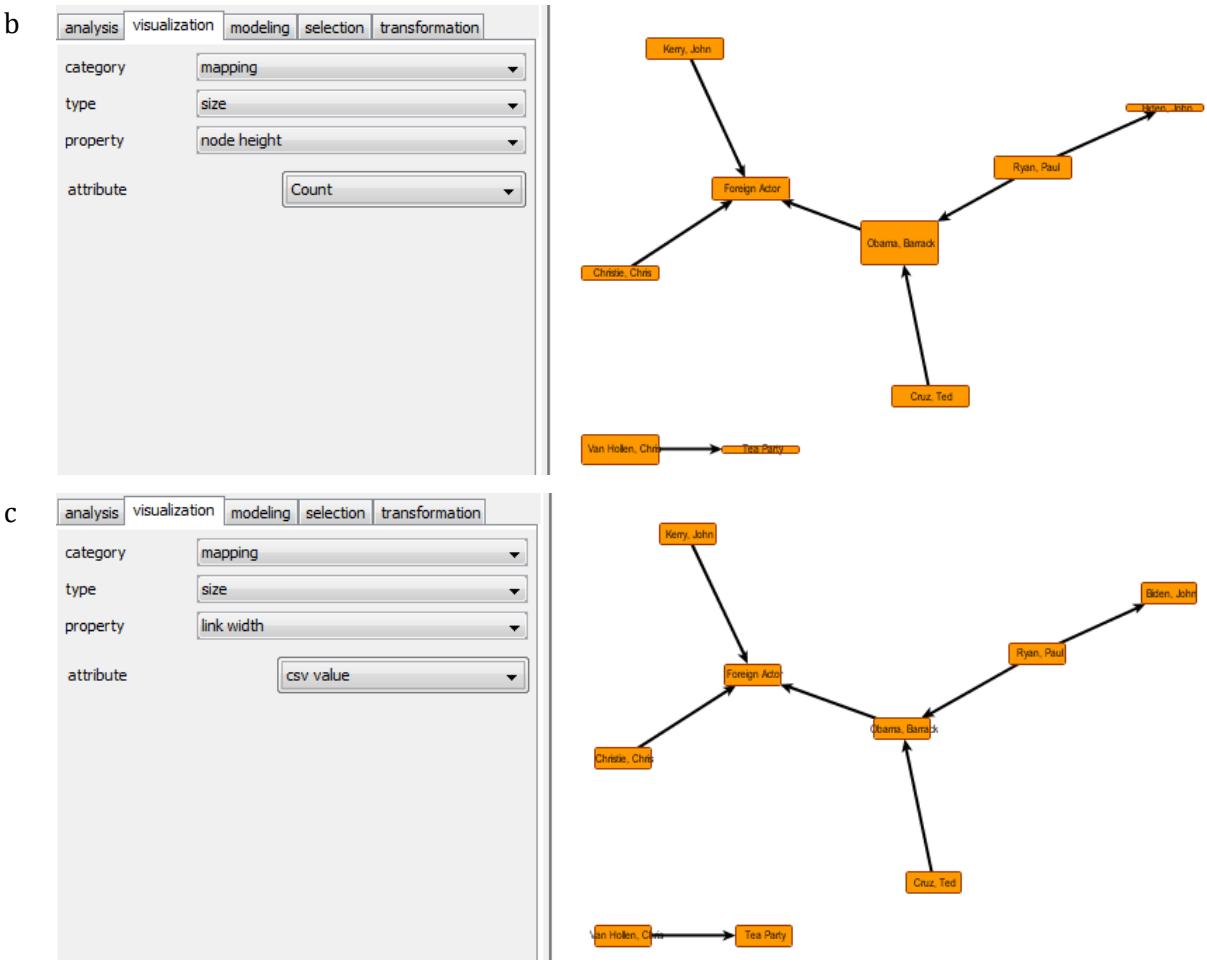
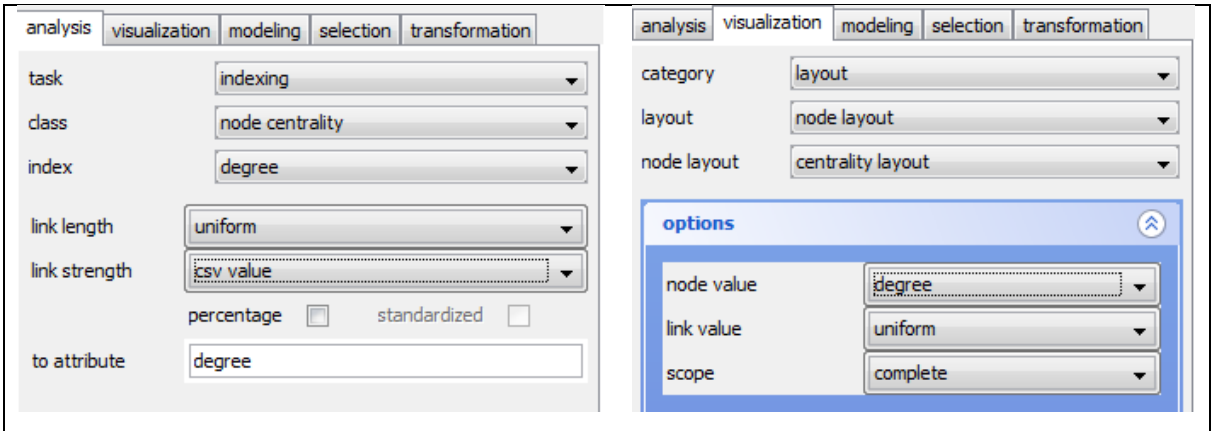


Figure 11: Data dependent properties of network appearance.

Network analysis

Using the tab 'analysis' on the left hand side of the window you may analyze the network. One useful procedure, which is present at the beginning, is the centrality calculation. Using this procedure, visone calculates a value for the centrality of nodes. The more links a node has to other nodes, the more central it is. When centrality is calculated, you may display the result using the 'layout' category of the 'visualization' tab (Figure 12).



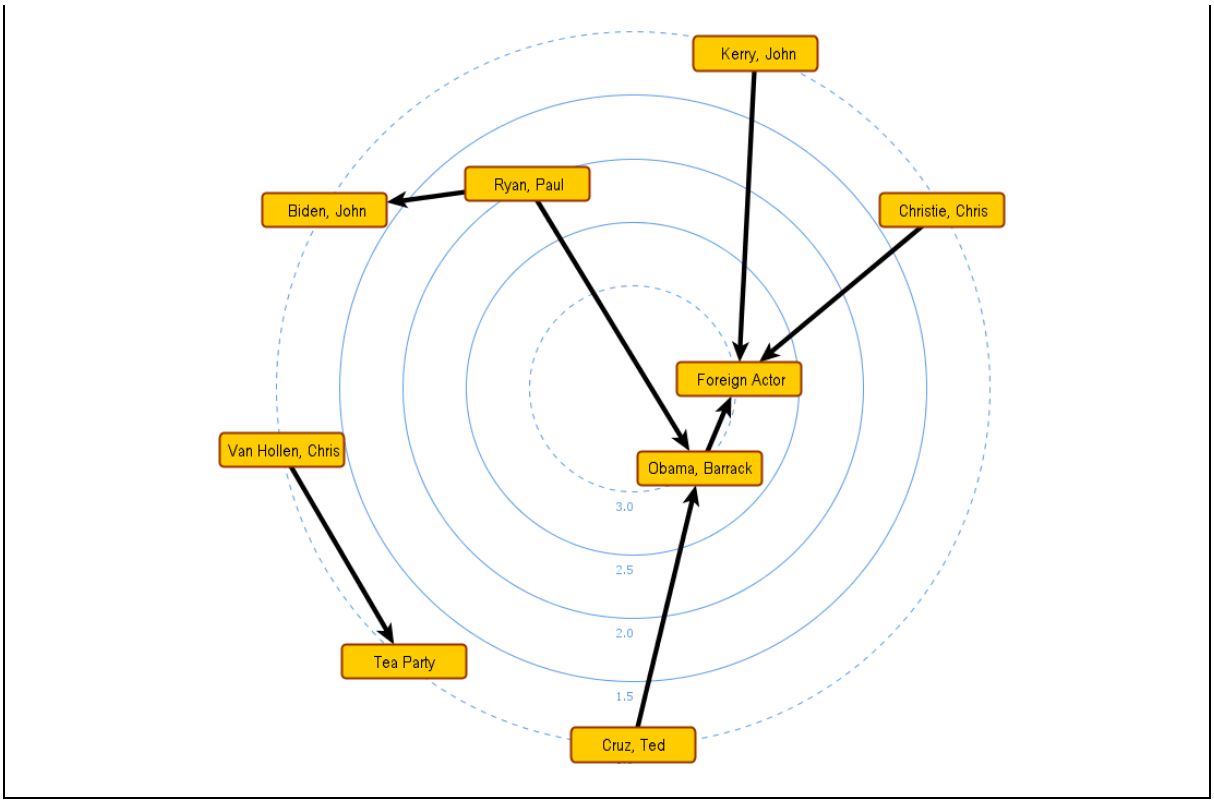


Figure 12: Calculate and display node centrality.

Chapter 11. Time Series Analysis

Time series analysis procedures allow the preparation and transformation of time series data. Time series transformations require a continuous numeric variable which is used as a timestamp.

11.1. Detect Peaks

Peak detection is a function which may be used to identify positive and negative peaks of time series data. Peaks are defined as the highest point of the time series which lies outside a confidence interval which may be chosen by the user. For all regions of the curve which lie outside the confidence interval, the highest (or lowest) point is detected.

A variable is added to the dataset which has the value 0 for most of the cases and assumes the value 1 for positive peaks and -1 for negative peaks. Values are considered to be a peak of the function if they lie outside a defined confidence interval and are higher or equal to both of their neighbors. This means that only the points of high peaks are given the values 1 or -1.

Applications:

- See Example: *Finding peak skin conductance during a movie*

Options:

Input Table

This file contains a table with time series data. It has to caintain at least one variable holding timestamps and another variable containing a time series of measurements for each timestamp.

Time Variable

From the complete list of variables in the dataset you may select the variable which holds the timestamps. Timestamps have to be numeric (seconds, hours, days...). If the timestamps have another format, use the function *Transform Timestamps* to transform timestamps to numeric format.

Each timestamp should be unique. If it is not unique, the last case bearing a certain timestamp will be used for calculation.

Measurement Variable

From a list of all variables you may select the variable holding the time series data. The variable has to be numeric and should be scaled and approximately normally distributed. Multinomial data may be read but result in nonsense results.

Output Variable

This function creates a new variable holding peak information. This variable has the value 0 for most values. For the highest point of positive peaks, the variable has the value 1, for the lowest point of negartive peaks it has the value -1.

Detect Peaks

You may choose which peaks you wish to detect. Positive peaks are values of the measurement variable lying above the confidence interval. Negative peaks are values which lie below the confidence interval.

Peak Threshold

As peaks are regions of the curve which lie outside a confidence interval. You may choose the width of the interval. You may also choose to find all peaks without any confidence interval. In this case all local maxima and minima of the curve are treated as peak. For curves containing noise this is not recommended.

Output File

The file in which the table with the new variable is written. The table corresponds ot the input table with one variable added at the end of the dataset.

11.2. Flatten moving Average

For time series data with moving averages, the detection of peaks or sudden changes in the average may be hard to detect. To amend for moving average, time series data may be decomposed to long-term moving averages and short-term peaks using this function.

The function takes time series data, a timestamp variable and an entered width for the gliding window to calculate the moving average. It then decomposes the time series data to a smooth curve of mean values and a flat curve of deviations from this mean curve. Depending on the size of the window, the variance of the time series data is distributed to the smooth curve or the deviations. For large windows, the smoothened curve exhibits less variation and the deviations exhibit more variation.

Applications:

- Smoothen a curve containing high-frequent noise
- Find peaks in a curve with moving average (in combination with procedure 'Find Peaks')
- Subtract the moving average from an unstable curve to obtain a curve with average 0
- See Example: *Find peak increases in skin conductance*

Options:

Input Table

This file contains a table with time series data. It has to caintain at least one variable holding timestamps and another variable containing a time series of measurements for each timestamp.

Time Variable

From the complete list of variables in the dataset you may select the variable which holds the timestamps. Timestamps have to be numeric (seconds, hours, days...). If the timestamps have another format, use the function *Transform Timestamps* to transform timestamps to numeric format.

Each timestamp should be unique. If it is not unique, the last case bearing a certain timestamp will be used for calculation.

Measurement Variable

From a list of all variables you may select the variable holding the time series data. The variable has to be numeric and should be scaled and approximately normally distributed. Multinomial data may be read but result in nonsense results.

Width of the gliding window

The moving average is computed using a gliding window with its center at the current position. You may specify the width of the movind window by entering a number. The units of the gliding window are the units of the time series variable.

Examle: If your time series data is available at 0.2 second steps and you enter a width of 3 seconds the average value at second 2.0 is calculated from all values between second 0.5 and 3.5.

Moving average Variable

This function creates a new variable holding the moving averages for each point in the time series. You may enter a name for this variable.

Peak Variable

This function creates a new variable holding the deviation from the moving average at each point of the time series. Adding this variable to the moving average variable, results in the exact value of the measurement variable at each point. The values in this variable indicate abrupt rise or decrease of the measurement variable.

Output File

The file in which the table with the new variable is written. The table corresponds ot the input table with one variable added at the end of the dataset.

11.3. Create Gliding Window

The gliding window procedure creates a gliding window on a time series dataset by copying cases to a number of groups. This procedure may be used to prepare a time series to aggregation.

The function creates a new dataset containing multiple copies of each case within the input table which may result in large files. Please be careful when handling large input files and consider using subsets for this kind of data transformation.

Options:

Input Table

This file contains a table with time series data. It has to caintain at least one variable holding timestamps and another variable containing a time series of measurements for each timestamp.

Time Variable

From the complete list of variables in the dataset you may select the variable which holds the timestamps. Timestamps have to be numeric (seconds, hours, days...). If the timestamps have another format, use the function *Transform Timestamps* to transform timestamps to numeric format.

Each timestamp should be unique. If it is not unique, the last case bearing a certain timestamp will be used for calculation.

Width of the gliding window

The moving average is computed using a gliding window with its center at the current position. You may specify the width of the movind window by seelcting a number. The units of the gliding window are the units of the time series variable.

Position of the identifier

A new variable will be created which contains the identifier for the gliding window each case belongs to. Since a case may belong to many gliding windows, each case is copied for each window it belongs to. The identifier of the gliding window is a value from within the window. You may choose which value should be used. You may choose the first element to name each window with the first time stamp which still belongs to the window. Respectively, you may choose the name to be the middle or last timestamp.

Position Variable

This function creates a new variable holding the identifier of the gliding window. You may enter any name for this variable.

Output File

This is the file in which the table including the new variable is written. The table corresponds to the input table with one variable added at the end of the dataset and a multidude of copies for each case.

11.4. Detect Gaps in Timeseries

Especially when working with trace data or other behavioral data, it may be useful to tag gaps or interruptions in the observation to segment the data. Nogrod allows you to tag these gaps automatically by scanning the timestamps and indicating points where the distance between two neighboring timestamps exceeds a manually set threshold.

This function is useful, for example, for segmeting browser histories to actual sessions in which a person surfed the internet. By setting a threshold of 30 minutes, the browser history is segmented to chunks that end whenever there is inactivity for 30 minutes. The individual sessions may then be investigated.

Options :

Input Table

This file contains a table with time series data. It has to caintain at least one variable containing a time series of measurements. There may be missing values in the series. The time series should be integer or floating point numbers. If the data only contains a complex date, reformat the timestamps to seconds or the numeric format used in Excel.

The input table may also contain a grouping variable. If a grouping variable is present, the tags are sensitive to these groups and will handle time series within different groups separately.

Timeseries variable

The variable containing the timestamps in an integer or floating point format.

Group variable (optional)

The variable containing the group specifier (e.g.: Respondent ID). Time series from different groups will be handled separately.

Duration of a gap

In an open text box, specify the duration which should be considered a gap. Note, that the duration is in the same scale as your timestamp variable. If the timestamp variable is in seconds, the duration should also be provided in seconds. Only a number (integer or decimal) may be entered in this text box.

Sort by timestamps

Internally, the dataset will be sorted to find the gaps in the time series. Using a radiobutton you may indicate wether the dataset should be stored in the sorted order. If you select 'No', the output table has the same order as the input table. The decision does not have any effect on the detection of gaps.

Segment variable

In an open text box, specify the name of the new variable to be created. The new variable will contain the number of the segment as integer, starting from the earliest timestamp (in the alphanumerically first group).

Output File

This is the file in which the table including the new variable is written. The table corresponds to the input table with one variable added at the end of the dataset. Depending on the decision whether or not it should be sorted, the order of the table differs from the input table.

11.5. Normalize Timeseries

This function reformats time series data with unequal distances between measurement points to a continuous series with equidistant timestamps. The method by which the measurement for each unobserved moment should be computed may be specified.

This function offers a possibility to synchronize trace data or eye tracking data by transforming all observations to continuous data with a fixed resolution. This is especially useful when different observations are to be aggregated.

After defining the variables, Nogrod writes descriptive statistics on all groups in the text output. You may use this information to find abnormal groups and determine sensible choices for the scle used in the output.

Input Table

This file contains a table with time series data. It has to caintain at least one variable containing a time series of measurements. There may be missing values in the series. The time series should be integer or floating point numbers. If the data only contains a complex date, reformat the timestamps to seconds or the numeric format used in Excel.

The input table may also contain a grouping variable. If a grouping variable is present, the tags are sensitive to these groups and will handle time series within different groups separately.

Timeseries variable

The variable containing the timestamps in an integer or floating point format

Grouping variable (optional)

The variable containing the group specifier (e.g.: Respondent ID). Time series from different groups will be handled separately.

Relative Timestamps

Specify, whether the first timestamp (within each group) should be set to zero. If 'No' is selected, the original scale of timestamps is retained.

Interval length

The resolution of the output file. In the text output, the descriptives of all groups (or the complete time series) is displayed. You may use this information to decide for a sensible choice for the interval. Note, that a line for each step is created in the output table. Using few seconds if the length of the time series is several months will result in a huge table.

The descriptives both indicate the range of the time series and the mean duration of a step between two adjacent timestamps. It is recommended to use a number that is not too much larger than a mean step and not so small as to create thousands of observations.

Select Variables

You may select any number of variables that should be included in the output table. For each variable, a value will be computed at each point in time.

Aggregation method

You may select the most appropriate method to determine the state of the variables at unobserved points in time. The choices are:

- **Most proximal value**: If no measurement is present for a point in time, the nearest point in time is used as a proxy. The values of all variables from the nearest point in time are stored.
- **Most recent value**: If no measurment is present, the last value for each variable is stored. This option is useful for trace data and eye tracking data where the logfile contains the entry to a new page or area of interst. It may be assumed that this value is valid until the next entry.

- **Last mode value**: Just as the most recent value, this option uses the most recent measurements as a proxy. If there are several measurements between the last two points in time, however, it does not use the most proximal but the mode value of all values that were present in the interval.
- **Interpolate values**: This option is only possible for numeric values. The value at unboserved points in time is interpolated linearily from the most proximal measurements.

Output File

This is the file in which the normalized data should be written.

11.6. Pattern detection

The pattern detection for time series data uses a model pattern to find corresponding patterns in continuous time series data. The procedure may be used to find one pattern in one measurement series or any number of patterns to find in parallel measurement series.

The procedure uses a gliding window of variable length to detect, for any point in the time series, the optimal correlation of pattern and measured data. The procedure, therefore, requires a continuous sequence of measured values, a simple pattern to be matched with, and a minimum and maximum length of the gliding window.

The procedure then calculates the correlation between pattern and measurement for each time point and each window length to store the highest possible correlation. There are three ways to store the data: The raw correlation (best correlation of the pattern, beginning at this point in time), dichotomous indication of a correlation higher than a selected value (i.e. Peaks), or the y-axis range of the optimal match found for this point in time.

Especially in the presence of noisy data, where small false hits may occur from chance, the third method is viable as it gives a measure of the strength of the signal correlating with the predefined pattern.

If more than one time series are scanned, you may choose between the arithmetic and geometric means of ranges of the time series under investigation. If you deal with time series of similar scale, arithmetic means is preferrable. For time series with very different scaling, use geometric means.

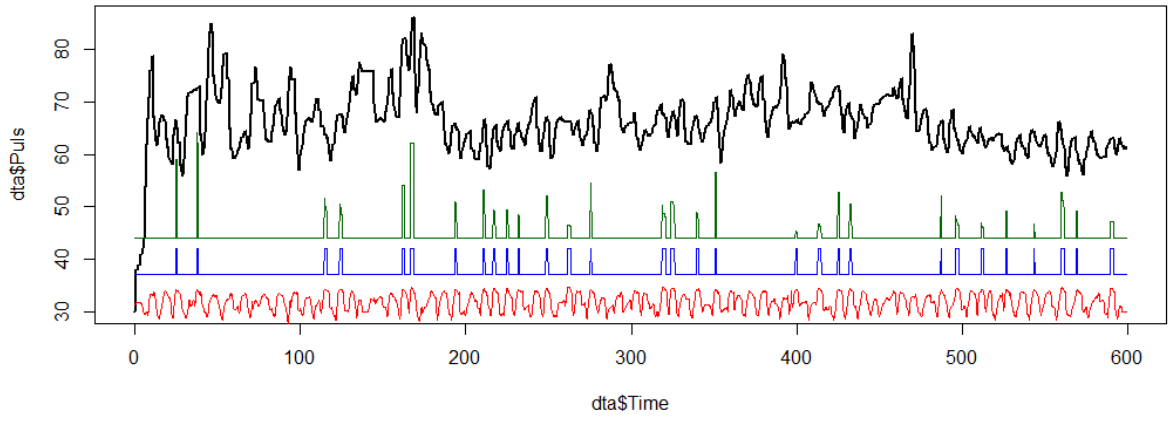


Figure: Example output for all three methods on one series. Red: correlation of the measurement sequence with a pattern `[1,1,0,1,1]`. Blue: indication of correlations higher than 0.8, Green: altitude of the patterns matched with a correlation above 0.8. You may see that some

correlations, although higher than 0.8, refer to quite small peaks in the measurement curve. These may be regarded as noise.

Options:

Input Table

This file contains a table with time series data. It has to caintain at least one variable containing a time series of measurements. There may be missing values in the series.

Pattern Table

A file containing the predefined pattern(s) in table format. Patterns are stored in columns with the name of the pattern being the name of the column (if header is present). Missing values in colums are ignored, so you may have patterns of different lengths in the same table. Below, you see an example of six different patterns (v-shape, sudden rise, peak, decline to baseline, constant rise, and constant decrease).

P101	P012	P010	P210	Prise	Psink
1	0	0	3	1	5
1	0	0	2	2	4
0	0	1	1	3	3
1	1	0	0	4	2
1	2	0	0	5	1
	3		0		

Measurement Variable(s)

The variable(s) containing the measured values in the time series file. The values are used in the order they appear in the file. Therefore, the file should be sorted by timestamp or be in the correct order by some other means.

Parallel Patterns

The patterns to be aligned with each measurement series. The order of patterns must match the order of measurement variables selected before. You may add the same pattern several times if you wish to find the same pattern in different timeseries.

Output Variable

A new variable will be created which contains values from the pattern detection procedure. You may choose any name. If it is already in the file, a new variable with suffix 01 will be created (or 02, 03, 04... if this is taken as well).

Minimum length of pattern

Indicate the minimum width of the gliding window. Depending on the scale of your data and the theoretical minimum length of the pattern, you may give an estimate, here. Note that the minimum length may not be smaller than the pattern itself. If you enter a value lower than the length of the pattern, it will be corrected to the length of the pattern afterwards.

Maximum length of pattern

Indicate the maximum length of the pattern to be detected. Here, again, rely on the theoretical boundaries of the pattern and the scaling of your data.

e.g. If you are looking for a pattern that may be between 3 and 6 seconds long and your data is measured in 4 Hertz, you should indicate a minimum length of 12 and a maximum length of 24.

Output method

You may choose from different kinds of output from the procedure. The text output in Nogrod indicates the distribution of the correlation measure to give you some indication on the values to be expected. The distribution also helps judging what a 'high' correlation should be in the case under investigation:

- Continuous Correlation: Store the maximum correlation with the pattern for each point in the time series. Note that the correlation does not depend on the size of the pattern.
- Dichotomous Indication: Store a dichotomous measure indicating whether a threshold correlation is reached.
- Altitude of matched patterns (arithm. mean): Store the y-axis range of the measurements with the best match only for points where a threshold is reached. This variable is essentially the product of the peaks variable and a variable indicating the y-range for each best pattern. This output mode has the highest information for peak analysis and allows to filter out noise. If more than one measurement variable is used, the altitudes of all curves are added up, and then divided by the number of measurements.
- Altitude of matched patterns (geom. mean): Store the y-axis range of the measurements with the best match only for points where a threshold is reached. This variable is essentially the product of the peaks variable and a variable indicating the y-range for each best pattern. This output mode has the highest information for peak analysis and allows to filter out noise. If more than one measurement variable is used, the altitudes of all curves are multiplied. The n-th root of the result is then returned with n equals the number of measurements

Threshold

If you selected dichotomous or altitude as an output method, you are asked to indicate a cut-off value for peaks. You may use the distribution of the correlation in the text output to choose a value. Note that correlations range between -1 and 1 and a high mean correlation may indicate a skewed distribution.

A value of 0.8 as a cut-off has proven to be a good starting point for most analyses.

Output File

This is the file in which the table including the new variable is written. The table corresponds to the input table with one variable added at the end of the dataset.

11.7. Synchronize Event Data

When combining time series data from different sources (e.g.: RTR, Eye Tracking, Trace Data,..) that are expected to be off by some ticks, it may be necessary to synchronize the timestamps before performing any analysis. The same applies to physiological measurements where the stimulus was not provided at exactly the same point for each respondent.

The function only takes dummy variables as input and assumes that there is one optimal alignment of timestamps where the agreement between the different time series is maximal. By changing the timestamps in an iterative process, it narrows down this optimal solution.

Options :

Input Table

This file contains a dummy table with a measurement (0 or 1) for a series of equidistant points in time.

Variables to be synched

From all dummy variables, select the ones that should be synchronized. Only the synchronized variables will be written in the output.

Maximal allowable frame shift

Set an upper limit for the frame shift. No difference between two measurement sequences may be larger than the shift allowed.

Measure of similarity

The agreement between two time series may be computed in different ways. Select the most appropriate calculation:

- **Percent agreement**: The agreement between two series is calculated as the percentage of points where both have the same value (0 or 1). This option is viable in cases where a 0 holds any meaning and two time series both being 0 are considered in synch.
- **Sokal distance**: The distance between two series is calculated by the number of times they differ when at least one of them has the value 1. This method ignores cases where both time series are 0. This method is useful in zero-inflated time series (e.g.: Event data).
- **Cohen's Kappa**: The distance between two series is calculated as percent agreement and corrected by per chance agreement. This measurement is useful when the binary variables are equally distributed on average but some of them may be skewed. In these cases, Kappa is more reliable than percent agreement.

Retain Time Stamp

If the synchronized data is to be rejoined with a larger dataset it may be useful to retain the time stamps. However, it has to be noted that the time stamps become unreliable in the course of this correction, as individual series are shifted several points in time to and fro. Thus, the timestamps do not reflect the exact moment of measurement anymore. Their deviation may amount to the maximal allowable time shift specified above.

Output File

This is the file in which the table of corrected time series is written. As columns are shifted in the course of the analysis, they may have leading or tailing missing values.

Chapter 12. Sequence Analysis

As a special case of time series analysis, Nogrod also contains a function to analyze a seqence of data to find patterns within the sequence for further analysis.

12.1. Find common sequences

The function for finding common sequences allows for the identification of sequence patterns within long sequences of data. This may be useful for time series data of nominal variables which indicate the state or actions of a person at each given time. As an example, the series of fixations of areas of interest in eye-tracking studies, or the sequence of issues in a news show may be investigated using this function.

The function is currently limited to the identification of sequences of four values.

Sequences with omissions:

The function does not only find coherent sequences but may also identify sequences with one omission. This means that there are four different sequences wich may be found in a sequence of length 4. In the sequence A-B-C-D, the first sequence of length three is ABC, the second is BCD. The fourth allows for one omission in the second place and is ACD. The fourth allows for one omission in the third place and is therefore ABD.

Keep in mind that the allowance of omissions lead to much larger data output.

Nomenclature of output:

Output of this function is a table containing all sequences starting from each case in the data. The sequences are thereby named according to the following system for a sequence A-B-C-D-E:

- Seq_2: AB, BC, CD, DE
- Seq_2a: AC, BD, CE
- Seq_3: ABC, BCD, CDE
- Seq_3a: ACD, BDE
- Seq_3b: ABD, BCE
- Seq_4: ABCD, BCDE
- Seq_4a: ACDE
- Seq_4b: ABDE
- Seq_4c: ABCE

The sequences denominated with a suffix -a, -b, and -c are sequences with one omission.

Depending on the mode of output, the sequences are written in one variable each which is named after the sequence (broad table), or as one list of sequences in one common variable with a second variable indicating the type of the sequence (long table).

Options:

Input Table

This file contains a table with sequence data. It has to caintain at least one variable containing the sequence of interest. It may also contain a time variable and a variable differentiating between different cases for which this sequence data is available. If a time variable and a case variable are inside, it does not have to be sorted. If only the sequence data is available, it should naturally be in correct temporal order.

Sequence Variable

The one column in the data containing the sequence of interest. The sequence variable may be any list of nominal values.

Time Variable

If desired, the sequence may be sorted by a time variable in the data. The time variable should contain numerical values indicating the time for each value in the sequence.

Group Variable

When more than one sequence is present in the data, a group variable may be indicated which differentiates the sequences. This may be the person for which the sequence was recorded, for example.

Sequence Length

You may choose a length of 2, 3, or 4 values. The shorter sequences are automatically added, so that a length of 4 values also exports the sequences of langth 2 and 3.

Omission of one value

You may choose whether or not omission should be allowed. Be aware that omissions may double the output.

Output mode

You may choose whether the output table should be long (containing all sequences in one variable and multiplying the cases) or broad (equal amount of cases and one variable for each type of sequence).

Output File

This is the file in which the table including the new variable is written. The output table only contains the sequence variable, group variable, time variable and the sequences. All other information in the source table is not part of the output.

12.2. T-Pattern Analysis

The temporal pattern (T-Pattern) analysis was first introduced by Magnusson (2000) to identify patterns in the sequence of events in the presence of noise and slight variations. The basic idea behind the T-Pattern analysis is to find events which follow other events more often that would be expected in a random series of events. If any of these duplets are found in the data, they form events of themselves which may be part of a higher order pattern.

The result of the analysis is an account of found patterns, their number of occurrence and their location within the data.

In Nogrod, it is possible to perform a T-Pattern analysis for multiple parallel sequences and to take non-linear timestamps into account. This means that missing data or short bursts of high-frequency activity do not pose a problem to the analysis and that several sequences may be analyzed in one go to find common patterns in all of them.

12.3. Grammar Induction

Grammar induction is a method of sequence analysis in which multiple sequences are analyzed to identify common patterns. In contrast to t-pattern analysis, grammar induction does not calculate the probability of any given pattern to appear in a sequence but identifies the most strongly treaden path through a series of symbols in the sequences.

Based on the general idea of grammar induction, a pattern recognition algorithm (yet unnamed) is incoroporated in Nogrod to identify common patterns. The grammar induction algorithm is based on the ADIOS algorithm written by Solan et al. (2005) and was slightly adapted to the needs and particular shape of eye tracking data. In particular, the algorithm is able to use a

preliminary routine to collapse repeating patterns before trying to find common sequences. This step renders two sequences with different repetition cycles of the same pattern identical.

References

Solan, Z., Horn, D., Ruppin, E., & Edelman, S. (2005). Unsupervised learning of natural languages. *Proceedings of the National Academy of Sciences of the United States of America*, *102*(33), 11629–11634. https://doi.org/10.1073/pnas.0409746102

The algorithm in short:

1) Generating sequences from data

The input format is a table containing one variable representing the symbols which were encountered. In eye tracking data this might be the area of interest for each fixation, in log-files of web-browsers this might be the URLs called upon at any given point of time. In addition to this variable, a group variable (e.g. Subject, Session, Day...) may be used to differentiate multiple sequences within the data. Also, a variable containing timestamps may be used to sort the dataset prior to analyzing.

The algorithm reads the symbols, their group identifiers and timestamps and generates fixed sequences for later analysis.

In noisy data (e.g. Eye tracking), a minimum length of consecutive symbols may be defined. Symbols appearing less than this minimum length are not counted as a symbol. If, for example, the minimum length of consecutive symbols is 2, the sequence [1,1,1,1,1,5,3,3,3,3] would be transferred to [1,1,1,1,1,3,3,3,3]. The single symbol '5' would be regarded as a glitch, as it just appears once.

2) Collapsing repetitive patterns

If desired, repetitious patterns of any given lengths may be collapsed. In this step, small sequences repeating themselves are considered as one repeating sequence. The repeating sequence is replaced by a new symbol expressing the repetitious pattern. The sequence 'aus gegebenen anlässen' would be transformed to 'aus (ge)*b(en)* anlä(s)*en'.

The user may define a minimal and maximal length of repeating sequences. If the minimal length is set to 2, for example, the example above would just translate to 'aus (ge)*b(en)* anlässen' as the repeating 's' is just one symbol.

3) ADIOS light

The next step in the algorithm is a slim version of the ADIOS algorithm. First, all sequences are transformed to a graph where each symbol is a node. The nodes are linked by directed edges counting the successions. If symbol A, for example, is followed by symbol B in three cases, the link A->B would have the value 3. The resulting complex network is then checked for treaden paths. A treaden path is a path through the nodes using links with high counts.

A pattern is found when both the beginning and the end of the path are fanning out, meaning that there is no link with high counts at either side which would allow for a continuation of the pattern. Since it is impossible to define absolute numbers for the required weight of links, the links are evaluated in relation to the path of the pattern already found. If there is a sudden drop in the probability of the paths when extending it further, the pattern is considered complete.

A coefficient Eta is used to define the drop of probability necessary. Eta is set to 0.9 by default and needs to be lower than 1. Lower Eta result in shorter and fewer patterns.

When patterns are found, they are checked for large patterns subsuming smaller ones. If any subsumption is found the smaller patterns are removed.

4) Rewrite the sequences
When all patterns have been found, the original sequences are rewritten, replacing sequences by their identifier. The resulting sequences are then written in the output.

Options:

Input Table
This file contains a table with sequence data. It has to caintain at least one variable containing the sequence of interest. It may also contain a time variable and a variable differentiating between different cases for which this sequence data is available. If a time variable and a case variable are inside, it does not have to be sorted. If only the sequence

Symbol Variable
The one column in the data containing the sequence of interest. The sequence variable may be any list of nominal values.

Group Variable
When more than one sequence is present in the data, a group variable may be indicated which differentiates the sequences. This may be the person for which the sequence was recorded, for example.

Time Variable
If desired, the sequence may be sorted by a time variable in the data. The time variable should contain numerical values indicating the time for each value in the sequence.

Exclude glitches
Here, the minimal length of consecutive symbols may be set. By default, this setting is set to 1, counting all symbols in the sequence. If set to higher values, symbols appearing less than that value in a row are not counted as symbols at all. The glitches are removed before collapsing repetitions.

Aggregate repetitious patterns
You may choose wheter or not to collapse repetitions of the same symbol (or series of symbols) to one single symbol expressing a repetitious pattern. If you choose to do so, you may also choose the minimal and maximal lengths of series you want to check for repetitions.

Minimal length of repeating n-gram
If collapsing repetitions was chosen, you may select the minimal length of n-gram to be checked for repetitions. The most reasonable choice, here, is 1.

Maximal length of repeating n-gram
If collapsing repetitions was chosen, you may select the maximal length of n-gram to be checked for repetitions. You may choose any value but a length of 3-5 symbols is usually most reasonable. Longer repetitious series already count toward patterns.

Minimal length of sequence to be extracted
Here, you may choose, how long a pattern has to be in order to be counted as a pattern. Shorter patterns will not be extracted and will not appear in the results, even if present. The most reasonable choice is 3, as sequences of 2 elements are quite frequent and not actually meaningful.

Minimal length of sequence to be extracted

Here, you may choose, how long a pattern may be in order to be counted as a pattern. Longer patterns will not be extracted and will not appear in the results, even if present. Actually, this value may be set high, if desired, as long patterns are very infrequent and will not be found anyway.

Eta-Coefficient

You may choose a value for Eta. It has to be lower than 1 and should be lower than 0.9. The lower it is, the harder it is to find sequences. If you get too many sequences, just lower Eta. Leave it at 0.9 for the first shot.

Output File

This is the file in which the results are written. The results contain a detailed log of all actions taken and results found. It also includes the transformed input sequences.

Chapter 13. Cluster Analysis

As there is no statistical software yet which has an implementation of cluster analysis of count data with outlier omission, this function was added to Nogrod.

13.1. Cluster Analysis of Count data (HECANE)

This method for cluster analysis uses hierarchical bottom-up clustering to find homogeneous clusters of large quantities of items and rejects all items not belonging to any of the homogeneous clusters. The result thus contains only a part of the items entered to the analysis. All other items are omitted.

HECANE is a bottom-up hierarchical cluster analysis aggregating all elements and clusters to one sinlge hierarchical structure, using cluster centre distance to combine clusters. In the process of the routine, each step of aggregation is evaluated. If the step is harmful to the homogeneity of the solution, it is tagged for postprocessing. Upon completion ogf the tree, all harmful steps are undone to isolate homogeneous clusters from the tree.

The result of the analysis is a collection of homogeneous clusters which could not be linked to their nearest neighbor without scrificing their homogeneity. The condition for steps to be counted as hamful is thereby: If the combination of a cluster with center C and radius R to the nearest element with distance D results in a new cluster with radius R' exceeding the length of D, the step is considered harmful. This condition only applies if the next step would be the inclusion of an outlyer or another homogeneous cluster with a high distance to C.

The resulting set of clusters has been shown to exclude randomly distributed elements and other sources of noise in count data matrices (vgl. Wettstein, Submitted). It is therefore an ideal procedure for clustering natural language and content analysis data where some elements (stop words or frequent categories) have to be excluded from the solution.

Nogrod performs the cluster analysis and gives a series of outputs containing fine-grained documentation of the process, a dendrogram, and a dataset with cluster scores to be included in statistical software packages.

Options

Input Table

This file contains a table with multiple variables constituting a count data matrix. There may be other variables in the dataset as well, as the user is asked to manually select the variables constituting to the matrix.

Count Variables

From a list of all variables, you may select the ones you wish to include in the cluster analysis. In the text output, a summary of each variable is provided which includes the variable level and the number of missing values. For the cluster analysis, only variables with numerical data and without missing values should be included as cases with missing / string values are excluded.

Standardization

If desired, you may choose to standardize table prior to the analysis. The columns of the table will be normalized in the progress of the content analysis, in order to level out the influence of frequent elements, you may preprocess the table by standardizing the rows. It is recommended not to use standardization.

Output FIle

Specify the file in which the summary of results is written. This file will contain the contents and radius of each of the emerging clusters, as well as their mutual correlations.

Additional Outputs

There is a series of additional outputs which may be provided.

- Smallest Space Analysis: If you select this option, an additional output is created containing an R-Script to perform a SSA using the smacof package in R. The analysis provides a visualization of the elements and their distance in the final solution and may be used for exploatory qualitative analysis and interpretation of the output.
- Dendrogram: This output is again an R-Script to visualize the results. With this script, a dendrogram of the cluster analysis may be visualized to investigate the relative postion of emerging clusters.
- Detailed clustering history: In this output, all decisions and calculations of the clustering process are documented for debuggin purposes. If you don't trust the cluster solution, you may investigate this file.
- Cluster loadings for all items: In this output, the distance of all elements and all cases to the cluster centers is provided. You may use this table for additional analyses of second-order clusters in statistical software.
- Complete Table of cluster vectors: In this output, all vectors of the analysis are provided. You may use them for additional factor- and cluster analyses using different methods.

Reference:

Wettstein, M. (Submitted). Identifying Clusters amid Noise: Hierarchical Exploratory Cluster Analysis with Term Exclusion. *Communication Methods and Measures*

13.2. K-Means Cluster Analysis

In K-means Cluster Analysis, the algorithm defines K random cluster centers and optimizes the solution iteratively to find the optimal place for the cluster centers which satisfy the condition that all cases belong to the cluster with the closest cluster center and all cluster centers are in the exact mean position of all elements within the cluster.

Options

Input Table

This file contains a table with multiple variables to be used for clustering. There may be other variables in the dataset as well, as the user is asked to manually select the variables constituting to the matrix.

Cluster Rows or Columns

Choose whether the rows or columns of the dataset should be clustered.

Number of Clusters

In a text box you may either enter a single number (e.g.: 3) or a range of clusters (e.g.: 2-10) to be extracted. The cluster analysis is performed once for each number of cluster centers. The text output informs you on the range of clusters selected. Check the text output to see whether your input was parsed correctly.

Cluster Variables

From all variables of the dataset select the variables relevant for clustering.

Standardize data

You may select to standardize (z-standardization) or normalize (scale to range [0;1]) the columns, rows, or the complete table. If your variables are scaled differently, column standardization or normalization is strongly recommended.

Output Table

Specify the file in which the results are written. The results are the input table with added variables for each number of clusters. These additional variables indicate the cluster each case belongs to..

13.3. Multigroup Cluster Analysis

Multigroup cluster analysis works exactly the same as the k-means analysis with the addition of one parameter. You may select a grouping variable which divides the dataset. The cluster analysis is then computed for both groups simultaneously, leading to roughly the same cluster centers for each group.

In MGCA, the text output in Nogrod is more extensive, as there are additional informations on pooled and un-pooled solutions of the cluster analysis. Also, the procedure is iterative, leading to longer computation time.

Chapter 14. Additional analysis procedures

14.1. Analysis of Entropy

The analysis of entropy compares the overall entropy of a nominal variable in a dataset to the entropy within groups of this dataset. It thereby analyes whether the distribution within the groups is comparable to the overall distribution or whether it is biased.

An analysis of entropy may be used to compare the distribution of issues in specific newspapers to the overall distribution of issues or to compare the distribution of actors between TV stations.

Options

Input Table
This file contains a table with at least one nominal variable and one group variable.

Multinomial Variables
From all variables in the dataset, choose the ones you wish to compute the entropy of.

Grouping variable
From all variables in the dataset, choose the one indicating the groups.

Options
You may choose to bootstrap the results to get confidence intervals for the entropy or to compare the results to a completely random assignment of groups. Since the entropy is expected to decrease if you group the data, a comparison to random groups offers thresholds for relevant bias within groups.

Output File
The output is a report on the analysis of entropy. Select a file in which to write that report.

14.2. Analysis of Attention and Focus

Based on Wettstein (2015), Nogrod offers a direct analysis of Attention and Focus of content analysis data. The function uses the pattern detection algorithm of Nogrod to scan content data for different types of attention peaks.

Options

Input Table
This file contains a table with dummy variables indicating the occurrence of sub-issues and actors.

Date Variable
From all variables in the dataset, choose the one indicating the date of the text.

Date Format
Choose the correct date format. When you choose a format, Nogrod directly assesses the format and tries to get the correct interpretation of each timestamp. In the text output, a summary of valid and invalid timestamps is displayed. By the invalid cases you may guess the correct format of the timestamps.

Number of text variable
From all variables in the dataset, choose the one which inidicates the number of texts on each date. If each line in the dataset is a text, let Nogrod count them itself.

Weighting factor

If you want to weigh the texts in your data, you may choose the weighting factor.

Issue variables

From all variables in the dataset, choose the ones indicating the occurrence of issues.

Actor variables

From all variables in the dataset, choose the ones indicating the occurrence of actors.

Length of gliding window

Choose a length for the gliding window in which to unvestigate the entropy of issues and actors. 7 days is strongly recommended, as this levels out the sundays in your data.

Direction of gliding window

Choose whether the indicator of the gliding window is the first or the last element. If you choose the first day of the window, the entropy of the days following the key date is computed. If you choose the last day, the entropy of the days previous to the key date is used. It is recommended to use the last day, since this gives you the date on which the actual event happened.

Output File

The output consists of the three time series (Attention, Issue Focus, Actor Focus). The text output in Nogrod will indicate the days on which specific attention peaks started and tries to guess the nature of these peaks.

This function is still under construction and requires some additional work until the text output is intuitively interpretable. But I am working on it.

Chapter 15. Data visualization

In this set of functions, you find some very crude ways to quickly visualize your data. The pictures are not nice but using them you don't have to open a file in Excel or R to get a quick look at the data.

15.1. Heat Map

This functon displays a heat map of large matrices (e.g. Dummy tables, Count data, Content analysis data) to show you the overall distribution of content.

Applications:

This procedure may be used whenever you want to:

- Get an impression of large matrixes containing numerical data.

Options:

Input Table
This table is a text document.

Numeric variables
From all variables in the dataset, you may choose the ones containing numeric values to display.

Sort Variables and Cases by Value
If you choose to sort the table, the rows and columns are sorted by their margins.

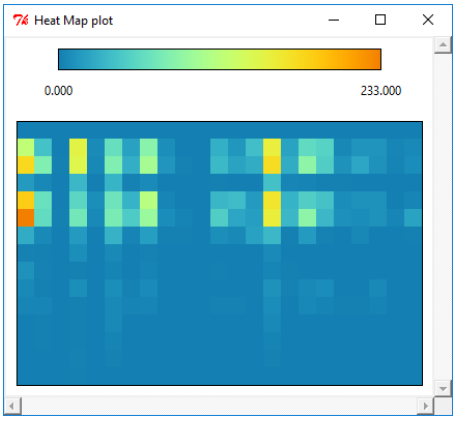
Maximal Size of the plotting region
Depending on your screen and the desired size of the output you may choose a maximal size.

Color Scheme
You can choose between three basic color schemes: Black and White (with black or white as the highest value) and red-blue (with red as highest value).

Logarithm
Indicate whether you want to logarithmize the values in the matrix. Advised for count data.

By clicking 'check' you can display the heat map, change the settings and display it again. If you wish to exit, just click 'Finished'.

Example Heat Map from count data.



15.2. X-Y Plot

This function plots a crude X-Y-Plot of data just like the one you get by prompting `plot(x,y)` in R. You may select two variables and plot either a line plot or a scatter plot. The data is sorted by the X variable to make the line plot look normal.

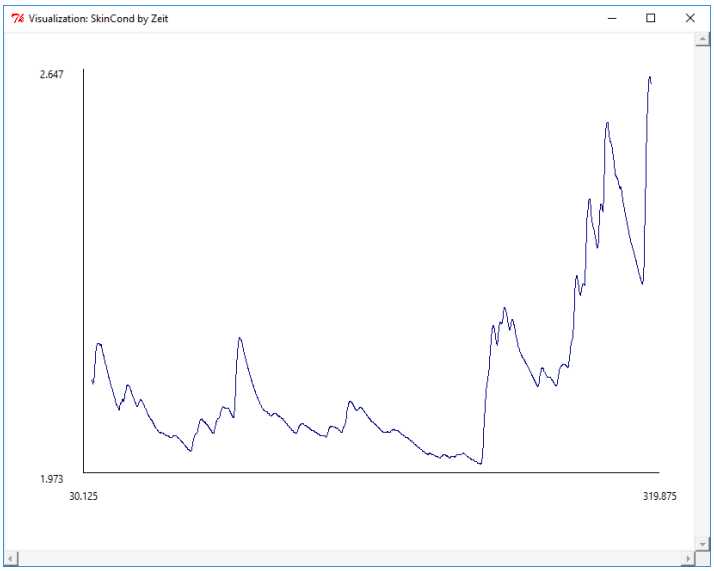
Input Table
This table is a text document.

X Axis
From all variables in the dataset, Select the one that stands for the horizontal axis.

Y Axis
From all variables in the dataset, Select the one that stands for the vertical axis.

Plot X-Y
You may choose to plot a Line plot or a scatter plot of the data. The plot will be displayed in a new window

Example Plot from the Example_Timeseries data:



Chapter 16. Reliability Testing

As Nogrod was designed to handle large amounts of data, mainly from automated and manual content analysis, it also includes functions for testing interrater (or intercoder) reliability.

16.1. Test interrater reliability

The function for testing interrater reliability allows for extensive tests of interrater reliability across large amounts of units of analysis and large sets of coders. It includes a growing sample of coefficients and speical functions which enable quick and thorough analysis of coder reliability.

In order for this function to work properly, the input data should come in a standardized form where the coding of each coder for each unit of analysis occupies one row (See Table 4). The number of categories (columns) is not limited. Missing values within the coding of an unit of analysis are allowed, so are missing units of analysis for a coder or missing coders for a given unit of analysis. The function only compares the information which is present.

Table 4: Input format for reliability tests.

Unit of analysis	Coder	Genre	Relevance	Politics	Economy
text1	joe	2	1	1	0
text1	anne	2	1	1	0
text1	fred	2	1	1	1
text2	joe	3	1	0	1
text2	anne	3	2	0	1
text2	fred	1	1	0	1
text2	jim	3	1	0	1
text3	joe	2	3	1	0
text3	anne	2	3	1	0
text3	fred	2	3	1	
text3	jim	3	3	0	
text4	joe	1	2	1	1
text4	jim	1	2	1	1

The function is programmed to calculate the most widely used coefficients for interrater reliability. In this chapter, a short introduction to each of the coefficients is offered in order to understand their background and calculation. For further information on the coefficients, please refer to the references offered at the end of this chapter.

Pairwise comparisons:

The most basic coefficients are based on the pairwise comparison of all coders among each other or against a correct solution (gold standard) for the texts that were rated. In order to calculate the coefficients for pairwise comparisons, cross tables are created which hold the information on the agreement between each pair of coders. For the purpose of an example, the category 'Genre' from Table 4: Input format for reliability tests.Table 4Table 4: Input format for reliability tests. is used to demonstrate the calculation of pairwise comparisons.

The cross tables obtained from comparing all coders are depicted in Figure 13.

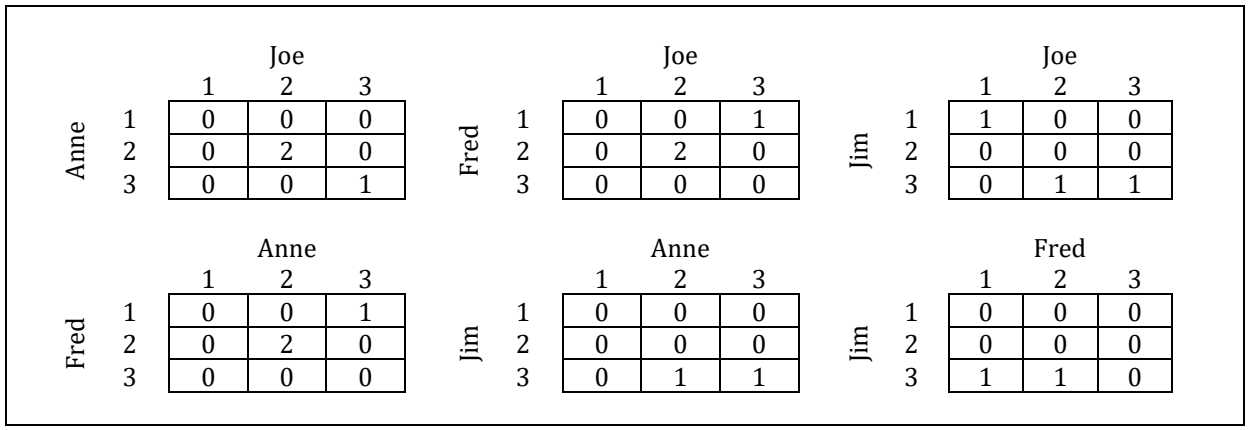


Figure 13: Cross tables for the variable 'Genre'.

From these crosstables, the pairwise agreement may be computed for each pair of coders (e.g.: Joe and Anne agree 100%, whereas Joe and Fred Agree by 66%). The most basic coefficient for pairwise comparison is calculated as the mean agreement across all coder pairs. This coefficient is called **Percent Agreement** or **Holsti Coefficient**. In this example, the coefficient would be 58.3%. For the Holsti coefficient, values above 80% are usually considered good agreement.

While the advantages of Holsti are its very easy calculation and interpretation, it has the disadvantage of being too liberal when it comes to unevenly distributed categories. In categories which have the same value in 90% of the cases, the chance of accidentally choosing the same value lies at 81%. Therefore, an agreement of 80% would not be considered very reliable in a real content analysis. In order to correct for this shortcoming, Kappa coefficients are often used in pairwise comparisons.

Kappa coefficients, such as **Cohen's Kappa** and **Soctt's Pi** are using the per chance agreement (Pc) of two coders, given the distribution of values, and indicate the distance of the true agreement (PA) versa the per chance agreement. In order to compute Kappa values, the following formula is used:

$$k = \frac{PA - Pc}{1 - Pc}$$

Kappa is exactly 0 when the actual agreement is equal as the per chance agreement. When the actual agreement is 100%, kappa equals 1. When the actual agreement is lower than the per chance agreement, kappa is negative. Values above 0.4 are generally considered acceptable, whereas values over 0.6 are considered good.

The only difference between Cohen's Kappa and Scott's Pi lies in the calculation of per chance agreement. For Cohen's Kappa, the per chance agreement is calculated individually for each pair of coders by summing up the suqared marginal chances for each category. In the above example, the per chance agreement for Joe and Fred would therefore be $(1/6)^2+(4/6)^2 + (1/6)^2 = 1/36 + 16/36 + 1/36 = 18/36 = 0.5$. With the actual agreement of Joe and Fred being 66%, Cohen's Kappa would equal: .166/.5 = 0.333.

For Scott's Pi, the per chance agreement is calculated by using the distribution of the values for the whole dataset. It is therefore constant for each pair of coders and allows for a direct comparison of pairwise coefficients. For Genre, the per chance agreement equals $(3/13)^2 + (6/13)^2 + (4/13)^2 = 0.361$. Therefore, the Scott's Pi for Joe and Fred would equal (0.666-.361)/(1-.361) = 0.478. In this case, Scott's Pi is slightly larger than Cohen's Kappa. In most cases, however, Cohen's Kappa is more liberal than Scott's Pi as the per chance agrement for a

badly agreeing pair of coders may be quite low. Scott's Pi is the most conservative and comprehensive coefficient for pairwise comparisons and should be preferred.

Note, that both Kappa coefficients are undefined for invariant variables because the per chance agreement of invariant variables equals 1.0. Therefore, kappa would result in a division by zero and may not be calculated. In Nogrod you may choose whether to set kappa for invariant variables to 1 (default) or a missing value.

Category comparisons

Slightly different from pairwise comparisons, variable coefficients such as **Krippendorff's Alpha** calculate the agreement on one category for any given team of raters. There are no pairwise comparisons in this calculation which means that there are no scores for pairs of coders or for single coders. The basic formula for behind Krippendorff's Alpha is the ratio of variance of a category within a unit of analysis and the overall variance of a category over all units of analysis.

To calculate Krippendorff's Alpha of M coders on N units of analysis, where the coding decision of Coder c on unit u is C(c,u) would therefore be:

$$\alpha = \frac{D_o}{D_e}$$

With D_o, being the mean distance within all units of analysis:

$$D_o = \frac{1}{N}\sum_{u}^{N}\frac{1}{M(M-1)}\sum_{c1}^{M}\sum_{c2}^{M} distance(C(c1,u),C(c2,u))$$

And D_e, being the mean distance between all coded values:

$$D_e = \frac{1}{N(N-1)+M(M-1)}\sum_{u1}^{N}\sum_{u2}^{N}\sum_{c1}^{M}\sum_{c2}^{M} distance(C(c1,u1),C(c2,u2))$$

The distance between two codes is thereby calculated differently for metric, ordinal and nominal variables.

- For nominal variables, the distance is 0 for equal values and 1 for different values.
- For ordinal variables, the distance is relative to the number of codings of the ranks between the ranks coded by the two coders. If they agree, the distance is 0.
- For metric variables, the distance equals the square distance of the values.

Coder comparisons

For large teams of coders, another coefficient has been introduced recently, which allows for the calculation of a score for each coder in the team. **Fretwurst Lotus** is computed as the mean agreement with the mode value of a category for a unit of analysis. This coefficient indicates in how many cases a coder is in agreeement with most of the other coders.

Since Fretwurst Lotus does not include the agreement among coders which do not agree with the mode value, it is very liberal and is higher than Holsti in most cases. It is almost impossible to reach a Lotus below 0.5 and in most cases it ranges from 0.8 to 1.0. Regression analyses have shown that Lotus is relative to Holsti with 94.6% of variance being explained by the formula Lotus = 0.363 + 0.643*Holsti.

Summary of coefficients:

Since all coefficients have their benefits and shortcomings, a short overview is offered here:

- Holsti: Easy to calculate but very liberal for unevenly distributed categories. Allows for comparison with one solution.
- Cohen's Kappa: Corrects for per chance agreement but hard to calculate and hard to interpret across large teams of coders. Allows for comparison with one solution.
- Scott's Pi: Easy to calculate and comprehensive for large teams of coders. Converges with Krippendorff's Alpha for large samples. Allows for comparison with one solution.
- Krippendorff's Alpha for nominal data: Slow computation but standard measure for content analyses in communication sciences.
- Krippendorff's Alpha for ordinal data: Very slow computation and hardly comprehensive.
- Krippendorff's Alpha for metric data: Slow computation but comprehensive and useful for rating data.
- Fretwurst Lotus: Very easy to calculate and gives information on the fit of single coders, but it is far too liberal to be taken serious.

It is therefore recommended to use Holsti, Scott's Pi and Krippendorff's Alpha together to get a comprehensive and detailed Analysis of interrater reliability.

References:

- Cohen, J. (1960). A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement*, *20*(1), 37–46. doi:10.1177/001316446002000104
- Fretwurst, B. (In Press). Intercoderreliabilität und Expertenvalidität bei Inhaltsanalysen: Erläuterungen zur Berechnung des Koeffizienten Lotus mit SPSS. In. W. Wirth, K. Sommer, M.Wettstein, J. Matthes (Eds.). [Fortschritte in der Inhaltsanalyse].
- Krippendorff, K. (2004). *Content analysis: An introduction to its methodology*. Thousand Oaks, Calif: Sage Publ.
- Scott, W. A. (1955). Reliability of Content Analysis: The Case of Nominal Scale Coding. *Public Opinion Quarterly*, *19*(3), 323–325.

Applications:

This function has only one application: Testing the agreement of a team of coders on a series of units of analysis.

Options:

Input Table

This file contains a table with reliability test data. The data should be formated as seen in Table 4.

Unit of analysis

From all variables in the file you may select the variable which denotes the unit of analysis. In the example in Table 4 this would be 'Unit of analysis'.

Coder identifier

From all variables in the file you may select the variable which denotes the coder. In the example in Table 4 this would be 'Coder'.

Test Variables

From all variables (except for the identifiers for coder and unit of analysis) you may select the variables for which a reliability test should be performed.

Test Cases

From all units of analysis within your file you may select the units which are to be included in the test.

Compare Coders

From all coders within your file you may select the coders whose decisions you want to compare.

Core Coder

You may choose one coder to test all other coders against when computing percent agreement (Holsti), Scott's Pi or Cohen's Kappa. If you want to compare each coder to all other coders, select 'No core coder'.

Core coders may be used when one of the coders is the project leader to whom all other coders need to be compared to ensure the validitiy of the decisions of each coder.

Method

There are a number of coefficients which may be calculated in a reliability test. You may choose one or several of these coefficients to be calculated. See detailed descriptions for each coefficient above.

Special Options

You may choose some special options for the reliability test. These options are:

- No Kappa value for invariant variables: In the case of invariant variables which are assigned the same value for all units of analysis by all coders, the chance agreement is 100% and may not be beaten by human coders. You may choose to omit these variables from the analysis.
- Count missing values as coding: In some cases it may be required to count a missing value as a valid code. By default, missing values are considered missing decisions and are omitted from the analysis. If this option is checked, all missing values are included in the analysis.
- Demand a minimum of 2 / 5 comparisons: Especially for datasets with few units of analysis it may be useful to omit comparisons where only the decision on one unit of analysis is compared.
- RICO: When choosing a 'Reliability if coder Omitted'-Test (cf. Scharkow 2012: 129), all coefficients will be computed for the complete team of coders. Subsequently, for each coder in the team, the coefficients are calculated for the team without this coder. If the reliability increases by omitting a single coder, this coder is harmful to the overall relaibility and should be trained again.

Output File

The Output file contains a summary on the test options you entered and a table containing the mean values of all coefficients across all variables and coders.

Report File

The Report contains more information than the output file since it also contains individual tables with summaries on all coefficients which are calculated. All tables are stored with tabulator spacers. The file is best viewed by copy&pasting its contents to an empty Excel sheet.

Table 5: Result of the reliability test from the example data shown above. While the reliability is satisfactory for most variables and good for 'Relevance' and 'Economy', the reliability of 'Genre' is quite low with an agreement <60% and Scott's Pi = 0.348.

	Alpha Metric	Alpha Nominal	Alpha Ordinal	Kappa	Lotus	PA	Pi
Economy	0.738	0.738	0.808	0.556	0.875	83.30%	0.64
Genre	0.374	0.583	0.374	0.198	0.834	58.30%	0.348
Politics	0.718	0.718	0.718	0.444	0.917	77.80%	0.531
Relevance	0.911	0.791	0.871	0.685	0.917	80.60%	0.696
Total	0.685	0.708	0.693	0.471	0.885	75.0%	0.554

Chapter 17. Text analysis functions in Nogrod

Nogrod includes a small but steadily growing set of text analysis functions you may use to quickly find patterns in your texts, identify duplicates, or train Machine Learning algorithms.

For reasons of portability, Nogrod will convert text files to latin-1 encoding, wherever possible. This also includes the option to transliterate Cyrillic texts to Latin characters and to store the texts in Latin alphabet.

Nogrod includes stemming algorithms for several languages, all adapted from the Snowball Stemmer. Currently supported languages:

- German
- English
- Italian
- French
- Dutch
- Polish

The standard format for corpora in Nogrod is a dataset with several variables, one of them holding the full texts as strings. Annotations of the text are usually encoded in the other variables that may hold informations on the designation, filename, path, genre, origin, or other attributes of the texts.

Functions for text analysis in Nogrod include:

- Support Vector Machines for Classification
- Duplicate detection

Planned funtions:

- Naive Bayes Classification
- Dictionary-Based annotation (e.g. Sentiment detection)
- Tools for handling texts, such as cutting text collections to small packages.

Chapter 18. Get Universe of Ngrams

For various application of text analysis with Nogrod, it is useful to base the analysis on a lexicon of known n-grams that are neither too rare nor too frequent. If you are using Nogrod for Machine Learning, for example, it would not be wise to include all possible words, bi- and trigrams as features of the analysis. This would inflate the training table and take too much time. It is more reasonable to select words that do occur in at least 1% of texts and not in all of them. Thereby, you exclude frequent stop words as well as n-grams that only occur once or twice in the corpus and are unlikely to contribute to the solution.

This function allows you to automatically extract the universe of ngrams that occur within given frequency bounds. You may also choose to stem the words with the included stemmer to facilitate later analyses of texts using stemming.

The function takes a folder with text files and some parameters and returns a list of n-grams that occur within the texts. For each n-gram, the share of texts it occurs in is noted.

Options:

Input Directory

You may select any directory containing textfiles on your computer. All textfiles will be rendered to n-grams in order to provide a list of all n-grams encountered in real texts. As this function will only have to be used once for each language, you may as well use a very large folder of textfiles and give the program some minutes to complete its task.

Include Subdirectories

In some cases, your texts may be organized in sub-directories. If you select 'yes', all subdirectories are taken into account. If you select 'no' only the chosen directory is used to find n-grams.

Encoding of Textfiles

Depending on the encoding, you may choose ANSI/Latin-1/ASCII or UTF-8. Other formats are currently not supported.

Language of texts

Select the language in which the texts are written. The function will use a snwoball stemmer to extract word stems from the words in the texts. This step in the function is strongly dependent on the language used. If you do not want to stem the words, just select ‘No stemming’.

Length of n-grams

Choose the maximum number of consecutive words you wish to extract from the texts. If you select 3, all words, bigrams and trigrams will be extracted. Please consider that selecting a higher number of consecutive words results in longer computation times. In most cases, trigrams are the maximum required to reach a satisfacory reliability.

Lowest (highest) word frequency?

You may choose up to which frequency of words a word should be discarded from the list. As words occurring in more than 99% of the texts are highly unlikely to occur more in texts of one category than another one, it is safe to discard all words occurring in less than 1% and more than 99% of the texts. You may, however, also choose to exclude words which occur in less than 5% or 10% or in more than 95% or 90%, respectively.

Output File

Choose any file in which the complete list should be written in the end. The file will contain a table with two variables. The first variable 'Ngram' contains the words and consecutive words. The second variable 'Share' contains the share of texts the n-gram was encountered in.

The output table is sorted alphabetically.

Chapter 19. Create a Corpus

Nogrod allows you to create a corpus either from a codesheet with textfiles or from a folder of textfiles. The format of the corpus will be a table with one column containing the content of all text files as string variables. Corpora may be used in the various text analysis functions of Nogrod.

19.1. From Codesheets and Textfiles

If only the codings for each text are available but there is not yet any column holding the correpsonding texts, the function 'Create corpus' may be used to combine textfiles with the table of codings. The textfiles may be in ANSI format or UTF-8 and should be collected in a folder on your computer. One column of the codesheet must contain the filename of the text which was classified. It may either just hold the name of the textfile if you have collected all text files in one folder or it may contain the full path leading to text files if they are distributed.

Note that the size of a corpus is almost equal to the size of all textfiles used in this corpus, as they are stored completely in the new variable. If you work with more than 10'000 texts it is advised to create several smaller corpora instead of one large corpus.

The corpus file will include all variables of the codesheets and one additional variable with a name to be specified that contains the contents of the corresponding text file.

Options:

Codesheets

The first input of this function is a table of codings for the texts. This table has to include one column identifying the name of the textfile associated with each line. This identifier will be used to combine the contents of a textfile with each line.

Textfiles

Here, you may specify the folder in which the textfiles are collected. The folder should contain all files specified in the codesheet. Any line containing a missing file will be removed from the corpus.

Name of Text variable

Here, you may specify a name for the newly created column containing all texts (e.g. 'Fulltext')

Name of Text ID

From all the columns of the codesheet you may select the column which contains the identifiers (names of the textfiles).

Output File

Enter a name for the corpus to be created. The table will be stored as a textfile with header and a separator.

19.2. From a folder of Textfiles

If there is no metadata available for the texts to be incorporated, you may just transform a folder of text files to one corpus. This is especially useful in cases you want to do an exploratory text analysis of a collection of texts.

The output of this function will be a table with two variables, one of them holding the file names, the other one the contents of the text files.

Options:

Textfiles

Here, you may specify the folder in which the textfiles are collected. Only files with the extension '.txt' are used in the generation of a corpus.

Include Subdirectories

In some cases, your texts may be organized in sub-directories. If you select 'yes', all subdirectories are taken into account. If you select 'no' only the chosen directory is used to find n-grams.

Encoding of Textfiles

Depending on the encoding, you may choose ANSI/Latin-1/ASCII or UTF-8. Other formats are currently not supported.

Name of Filenames Variable

Here, you may specify a name for a variable holding the filename of each text. If subdirectories are included, the full path will be stored in this variable.

Name of Text variable

Here, you may specify a name for the newly created column containing all texts (e.g. 'Fulltext')

Output File

Enter a name for the corpus to be created. The table will be stored as a textfile with header and a separator.

Chapter 20. Inspect Corpus

As of the current verstion, Nogrod offers one function that allows for a quick glance at the contents of a corpus. Further functions, especially offering text statistics, will be added in the future.

20.1. Find context of Regex

This function allows for the quick search of a regular expression within a corpus. The user may enter any number of regular expressions and display the context of these expressions in the open corpus.

The function is aimed at showing the occurrences of specific words at a glance to explore strange words popping up in the training tables of SVM or to validate dictionaries used for sentiment detection on a given corpus.

Options:

Corpus to be inspectd
Here, you may specify the corpus to be used. The corpus should include both the texts of the documents and their ID. If no ID is provided, Nogrod will just enumerate the texts in the order of the corpus table.

Name of the text variable
The variable holding the full texts of each document.

Identifier Variable
The variable holding the IDs of the texts in the corpus for later reference. If no ID is used, you may select 'Don't use an identifier'. Nogrod will then enumerate the texts to provide the line number in the output.

Expression to be searched
In an open text box you may enter any regular expression (using the re library in Python: https://docs.python.org/3/library/re.html) to be searched in the corpus.

Case Sensitivity
With a radio button, you may specify whether the expression is case sensitive. If you search for words that may appear at the beginning or in the middle of a sentence, case senstivitiy might suppress some of the results. If you are searching for names or fixed expressions which are always capitalized, it may help narrowing down the examples.

Upon confirming the regular expression with 'Check', Nogrod lists all strings matching the regular expression and their context (+/- 15 characters) in the text output. If there are more than 200 matches, the output is capped.

You may then change the regular expression and search again. The results are then added to the output.

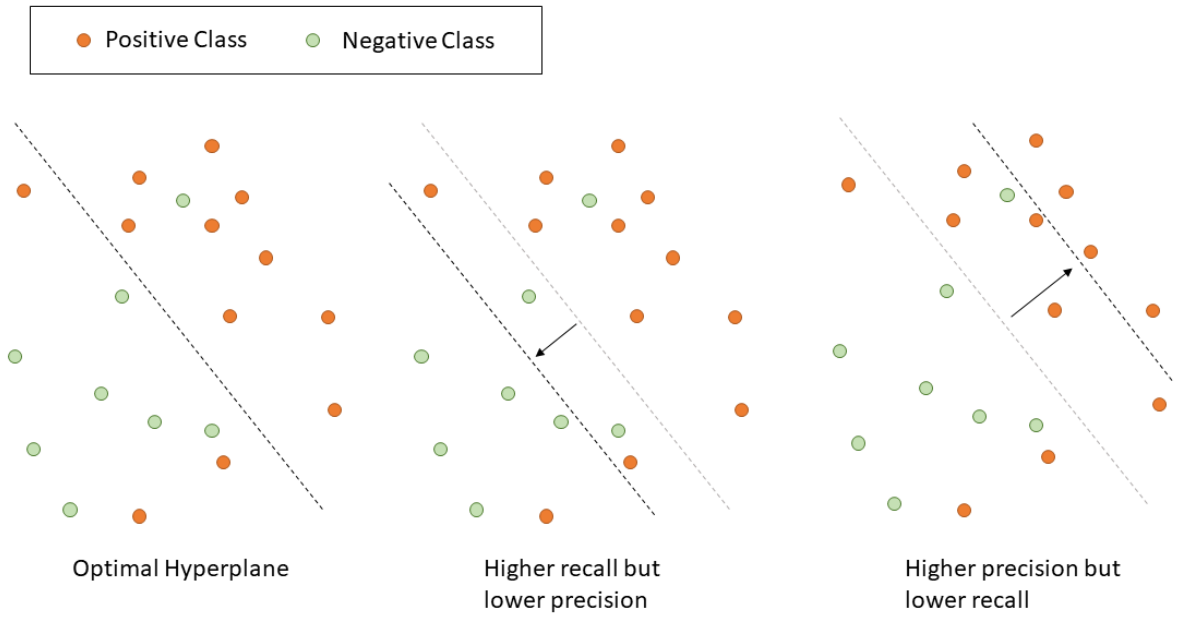
As several searches may be performed in sequence, Nogrod structures the output. Each search output begins with a header containing the regular expression, the number of documents, and the capping disclaimer. It always ends with a notice on the number of search results.

Note: Using 'Quick Tools' above the text output, you may always store the results to a text file. While this is possible for all text outputs, it is especially useful for this function.

Chapter 21. Support Vector Machines

A function to train, test, and apply support vector machines (SVM) is included in Nogrod. A SVM is a quick and multidimensionally linear approach to text classification that visualizes all documents as points within an N-dimensional feature space. It then searches for the plane that most efficiently separates the cases from different classes.

The SVM finds an optimal solution, that is, the solution with balanced precision and recall that has the highest F-score. By changing the intercept of the hyperplane, the model may be adjusted to optimize it for precision or recall. The test function in Nogrod allows you to inspect the effect of intercept changes on precision, recall, and F-score and choose the most appropriate value before classifying unknown documents.



Higher precision (fewer false positives) may be preferred in cases where representatives of the class are sought. Here, it does not matter whether some potentially positive cases are lost but a positive classification should be correct.

Higher recall (fewer false negative) may be preferred when texts on a specific topic are to be analyzed and the SVM just helps to discard the surely irrelevant texts. Here, it does not matter whether some negatives are classified as positive as they will be removed at some point in the subsequent analysis. It is more important not to miss anything relevant.

21.1. Training an SVM

In order to train an SVM, you only need a corpus with classified texts. The training algorithm may use words or n-grams as features and includes a stemming algorithm.

The training first reads all texts and extracts possible features. It then computes the optimal plane that separates the documents from different classes. From the slant of the plane in each

dimension (each feature), the algorithm then selects the most discriminant features. By default it uses the tom 2% of features. The strongest features are displayed in the text output.

In the text step, the use of each possible feature is tested by determining how much additional value it holds for the overall model fit. If a feature does not contribute anything to the solution, it is dropped from the model. Only relevant features are retained to compute the hyperplane. The relevant features are shown to the user as soon as they are found.

The optimal hyperplane (best F-score) is then computed and tested for precision and recall before it is written to a secified file in JSON format.

Options:

Corpus with Classification
Here, you may specify the corpus to be used. The corpus has to include both the texts of the documents and the classification that should be trained.

Name of the text variable
The variable holding the full texts of each document.

Name of the class variable
Name of the variable that specifies the class. Usually, this variable is binarily coded as 0 and 1.

Language of texts
Select the language in which the texts are written. The function will use a snwoball stemmer to extract word stems from the words in the texts. This step in the function is strongly dependent on the language used. If you do not want to stem the words, just select 'No stemming'.

Length of n-grams
Choose the maximum number of consecutive words you wish to count as features. If you select 3, all words, bigrams and trigrams will be extracted. Please consider that selecting a higher number of consecutive words results in longer computation times. In most cases, trigrams are the maximum required to reach a satisfacory reliability.

Lowest (highest) word frequency?
You may choose up to which frequency of words a word should be discarded from the list. As words occurring in more than 99% of the texts are highly unlikely to occur more in texts of one category than another one, it is safe to discard all words occurring in less than 1% and more than 99% of the texts. You may, however, also choose to exclude words which occur in less than 5% or 10% or in more than 95% or 90%, respectively.

Output File (json)
Enter a name for the training result to be created. The training result is written in a JSON file.

21.2. Testing and adjusting an SVM

The testing function uses the same options as the training function with one notable difference: A trained SVM is included by loading the respective JSON file.

After loading a corpus and an SVM JSON, Nogrod lists the relevant features and their slope in the text output for inspection. The tool also creates a visualization that illustrates the precision, recall, and F-score in relation to the intercept of the hyperplane. This information may be used to adjust this intercept and optimize the JSON for higher precision or recall.

Doring the test you may adjust the intercept manually and directly store it to the JSON file. Or you may leave it at the predefined value that originated from the training (optimal F-Score) or was adjusted in a previous modification.

Options

Corpus with Classification
Here, you may specify the corpus to be used. The corpus has to include both the texts of the documents and the classification that should be trained.

Name of the text variable
The variable holding the full texts of each document.

Name of the class variable
Name of the variable that specifies the class. Usually, this variable is binarily coded as 0 and 1.

Language of texts
Select the language in which the texts are written. The function will use a snwoball stemmer to extract word stems from the words in the texts. This step in the function is strongly dependent on the language used. If you do not want to stem the words, just select 'No stemming'.

Length of n-grams
Choose the maximum number of consecutive words you wish to count as features. This selection should be based on the trained SVM. If 3-grams were used in training, they should also be used in the test.

Hyperplane File
Specify the name of the JSON file in which the trained SVM was written.

Type of output column
Here, you may select what the output file should contain. It may either be the exact score of the document as idetfied by the SVM or the rounded score that corresponds to the category.

Name of the new variable
Pick a name for the variable that holds the SVM scores in the output table.

Hyperplane Intercept
As a default value, this parameter is set to the value stored in the JSON. If you want to change it, you may enter any numerical value in this textbox. Refer to the graphic output to make a reasonable choice.

Try or Test
When you have entered a value for the intercept, you may choose to either try this new intercept without any consequences or store it to the JSON and carry on with the test. If you try it, the text output will indicate the precision and recall in the test corpus using this intercept. You may try new intercepts any number of times.

When you decide to test, the intercept is written to the JSON file and the hyperplane is used to determine the score or classification of all documents in the text corpus.

Output File
Enter a name for the annotated corpus to be written. The table will contain an additional column with the computed scores.

21.3. Applying SVM to texts

The final goal of any classification algorithm is its application to unknown texts. Nogrod lets you apply the trained hyperplane to a new corpus or to a folder of textfiles. If you choose to apply it to a folder of text files, the function will first create a corpus and then continue with the analysis as if a corpus had been loaded.

The output of this function is an annotated corpus containing the contents of each text and its classification according to the SVM. If you choose to use metric scores as a result of the SVM, positive values correspond to 1 and negative ones to 0. If a document has an exact 0.0, it lies on the plane and its classification may not be inferred from the model. When storing dichotomous values, these cases are counted as "1".

Options

Input Text Corpus (only for classification in existing Corpus)
Specify the corpus within which duplicates should be identified.

Text Variable (only for classification in existing Corpus)
Name of the variable that holds the contents of the texts

Input Folder (only for detection in folders)
Name of the folder that holds all text files to be compared. It is possible to include sub-directories within this folder.

Encoding of Textfiles (only for detection in folders)
Depending on the encoding, you may choose ANSI/Latin-1/ASCII or UTF-8. Other formats are currently not supported.

Include Subdirectories (only for detection in folders)
In some cases, your texts may be organized in sub-directories. If you select 'yes', all subdirectories are taken into account. If you select 'no' only the chosen directory is used to find duplicates.

Language of texts
Select the language in which the texts are written. The function will use a snwoball stemmer to extract word stems from the words in the texts. This step in the function is strongly dependent on the language used. If you do not want to stem the words, just select 'No stemming'.

Length of n-grams
Choose the maximum number of consecutive words you wish to count as features. This selection should be based on the trained SVM. If 3-grams were used in training, they should also be used in the test.

Hyperplane File
Specify the name of the JSON file in which the trained SVM was written.

Type of output column
Here, you may select what the output file should contain. It may either be the exact score of the document as idetfied by the SVM or the rounded score that corresponds to the category.

Name of the new variable
Pick a name for the variable that holds the SVM scores in the output table.

Chapter 22. Find duplicates

Based on the work of Nicholls (2019), a plagiarism detection function is included in Nogrod. The function uses hashes of shingled n-grams to quickly compare large collections of texts for recurring elements.

The function may either be used on an existing corpus or on a folder of textfiles. If the latter option is selected, a corpus is first created from these textfiles before using the same options and algorithms as the first option.

The algorithm searches for ngrams that exist in different texts within the corpus and identifies these texts. The output is a list of text pairs that have a high overlap and might be considered duplicates.

Options

Input Text Corpus (only for detection in existing Corpus)
Specify the corpus within which duplicates should be identified.

Text Variable (only for detection in existing Corpus)
Name of the variable that holds the contents of the texts

Identifier Variable (only for detection in existing Corpus)
Name of each document that will be written to the result file. The identifier should be unique in order to identify the duplicates in the corpus.

Textfiles (only for detection in folders)
Name of the folder that holds all text files to be compared. It is possible to include sub-directories within this folder.

Encoding of Textfiles
Depending on the encoding, you may choose ANSI/Latin-1/ASCII or UTF-8. Other formats are currently not supported.

Include Subdirectories
In some cases, your texts may be organized in sub-directories. If you select 'yes', all subdirectories are taken into account. If you select 'no' only the chosen directory is used to find duplicates.

Length of shingled n-grams
You may choose how long the n-grams should be in order to detect overlap. If you select shorter n-grams the probability of false positives increases as standard phrases such as “the white house has announced” are considered overlaps. If you select longer n-grams, small deviations in texts, such as inserted adjectives may prevent duplicate detection.

Minimal overlap
In the text output Nogrod offers a table that indicates the occurrence of ovelaps in the corpus. In ascending order, the number of overlapping n-grams is listed and for each number of overlaps, a number of text pairs is indicated. In most cases, an overlap of 0-3 n-grams is most frequent with thousands of pairs. Depending on corpus and length of n-grams, more than 10 or 20 overlaps are rare. The table is trunctuated at 30.

From this table you may learn what a sensible lower threshold for duplicate detection in the given corpus might be. Enter the lowest number of overlaps to be counted as a duplicate in the text box.

Redundant output

The output is a file listing all pairs with an overlap above your lower bound. It is possible to use one line per pair or to use a redundant output with two lines per pair. In the redundant output, the paring A-B will be listed as well as the pairing B-A. This kind of output is useful if you want to identify single texts that have a high overlap with many others. Using a redundant output you do not have to look for this text in both the 'Text_1' and 'Text_2' column of the output but only in one of them.

Compute share of overlap

Especially if the length of texts varies, it may be useful to compute the share of the overlap instead of the absolute number. The share of the overlap is computed as the number of overlapping n-grams divided by the total number of n-grams for a given text. If this option is selected, the algorithm is slightly slower as it has to check the length of all texts but it may offer valuable information.

The shares are computed for both texts (Text 1 and 2) in the pair. Therefore, the shares may be used to infer whether a pairing is really a duplicate (two high shares) or whether one text merely subsumes the other (one high and one low share).

Output Table

Specify the table in which the information on duplicate pairs should be written. The table contains three or five columns, depending on whether shares are computed.

Chapter 23. Examples

23.1. Add cases from another dataset

For this procedure to work, two datasets should be present. As we only have one example dataset, we do something stupid here as we are just appending the table to itself. To do this, perform the following steps:

1. Select 'Merge Datasets' -> 'Add Cases from another table' from the menu
2. Select your first table ('Example_data.txt'), it contains a header and is separated by tabstopps
3. Select your second table ('Example_data.txt'), it contains a header and is separated by tabstopps

The text output now informs you on the size of the two datasets. As they are both the same, both of them contain 14 variables and 19 cases.

The output also informs you on the congruence of variables. It found 14 variables which are present in both files and none which are present only in one file.

4. In the list you see all variables and the information whether they are present in both tables or only in one. You may select the variables you wish to have in the output. For this example, just select Actor, Date, Issue and Quotation. Select them on the list at the top and press 'Add Item' to choose them. Then confirm your selection.
5. You may now choose a name for the output file and its format. (e.g.: 'Double-Table.txt'). Upon confirmation, the tool merges the datasets.

The text output now informs you on the size of the output which has 38 cases and only four variables.

The variables in the output table have the same order as in the input tables. If the names of the variables differ in both tables (i.e. if some variables only appear in one table) the table first contains the variables from the first table and then the variables which were only in the second table.

Table 6: Merged dataset when adding the example table to itslef.

Date	**Actor**	**Quotation**	**Issue**
Mar 13	Obama, Barrack	Direct	Budget
Mar 13	Van Hollen, Chris	Indirect	Budget
Mar 13	Ryan, Paul	Direct	Budget
Mar 13	Cruz, Ted	Direct	Obamacare
Mar 14	Christie, Chris	Indirect	Obamacare
Mar 14	Ryan, Paul	Direct	Budget
Mar 14	Kerry, John	Indirect	Budget
Mar 20	Christie, Chris	Direct	Iran
Mar 13	Van Hollen, Chris	Direct	Budget
Mar 15	Obama, Barrack	Indirect	Iran
Mar 15	Cruz, Ted	Direct	Iran
Mar 20	Van Hollen, Chris	Direct	Budget
Mar 20	Obama, Barrack	Indirect	Obamacare
Mar 14	Ryan, Paul	Direct	Obamacare
Mar 14	Kerry, John	Direct	Obamacare
Mar 14	Cruz, Ted	Direct	Obamacare
Mar 14	Van Hollen, Chris	Direct	Budget
Mar 15	Obama, Barrack	Direct	Iran
Mar 15	Kerry, John	Indirect	Iran
Mar 13	Obama, Barrack	Direct	Budget
Mar 13	Van Hollen, Chris	Indirect	Budget
Mar 13	Ryan, Paul	Direct	Budget

from here, the list repeats

Mar 13	Cruz, Ted	Direct	Obamacare
Mar 14	Christie, Chris	Indirect	Obamacare
Mar 14	Ryan, Paul	Direct	Budget
Mar 14	Kerry, John	Indirect	Budget
Mar 20	Christie, Chris	Direct	Iran
Mar 13	Van Hollen, Chris	Direct	Budget
Mar 15	Obama, Barrack	Indirect	Iran
Mar 15	Cruz, Ted	Direct	Iran
Mar 20	Van Hollen, Chris	Direct	Budget
Mar 20	Obama, Barrack	Indirect	Obamacare
Mar 14	Ryan, Paul	Direct	Obamacare
Mar 14	Kerry, John	Direct	Obamacare
Mar 14	Cruz, Ted	Direct	Obamacare
Mar 14	Van Hollen, Chris	Direct	Budget
Mar 15	Obama, Barrack	Direct	Iran
Mar 15	Kerry, John	Indirect	Iran

23.2. Add information on the actors to the table

For this example, a second table is needed. We therefore use the data in the example dataset Example_Functions.txt (see Table 2).

This information may then be added to the example dataset.

1. Select 'Merge two datasets'.
2. Choose the example data 'Example_Data.txt' as main table. Confirm and select 'Example_Functions' as key table.
3. Now the tables have to be linked by key variables. Choose 'Actor' as the key variable in the main table by adding them to the selection list.
4. Choose 'Actor' as the key variable in the key table.
5. From the variables remaining in the key table (which is only one) you may select the ones to be added to the main table. Select 'Function'
6. Choose an output file and confirm.

The result of this transformation is a new table which contains an additional column (Table 7)

Table 7: Merged dataset (omitting some variables to save space) with the additional variable. The value has been added to all rows of the dataset, according to the actor on this row

Show_ID	Station	Date	Actor	Actor_ Affiliation	Quotation	Issue	Function
FN001	Fox	Mar 13	Obama, Barrack	Dem	Direct	Budget	President
FN001	Fox	Mar 13	Van Hollen, Chris	Dem	Indirect	Budget	Representative
FN001	Fox	Mar 13	Ryan, Paul	Rep	Direct	Budget	Representative
FN001	Fox	Mar 13	Cruz, Ted	Rep	Direct	Obamacare	Senator
FN002	Fox	Mar 14	Christie, Chris	Rep	Indirect	Obamacare	Governor
FN002	Fox	Mar 14	Ryan, Paul	Rep	Direct	Budget	Representative
FN002	Fox	Mar 14	Kerry, John	Dem	Indirect	Budget	Secretary of State
FN003	Fox	Mar 20	Christie, Chris	Rep	Direct	Iran	Governor
PB001	PBS	Mar 13	Van Hollen, Chris	Dem	Direct	Budget	Representative
PB002	PBS	Mar 15	Obama, Barrack	Dem	Indirect	Iran	President
PB002	PBS	Mar 15	Cruz, Ted	Rep	Direct	Iran	Senator
PB003	PBS	Mar 20	Van Hollen, Chris	Dem	Direct	Budget	Representative
PB003	PBS	Mar 20	Obama, Barrack	Dem	Indirect	Obamacare	President
AB001	ABC	Mar 14	Ryan, Paul	Rep	Direct	Obamacare	Representative
AB001	ABC	Mar 14	Kerry, John	Dem	Direct	Obamacare	Secretary of State
AB001	ABC	Mar 14	Cruz, Ted	Rep	Direct	Obamacare	Senator
AB001	ABC	Mar 14	Van Hollen, Chris	Dem	Direct	Budget	Representative
AB002	ABC	Mar 15	Obama, Barrack	Dem	Direct	Iran	President
AB002	ABC	Mar 15	Kerry, John	Dem	Indirect	Iran	Secretary of State

23.3. Select statements of democrats containing an attack

For this example, two conditions have to meet. The values of two different variables have to take a certain value for a case to be selected.

1. Select the procedure 'Extract Subset from Data' -> 'Select Cases' and open the example dataset.
2. Select both variables which contain information required to extract this subset ('Actor_Affiliation' and 'Attack').

Upon confirming, the tool offers additional information on the values of these variables. The text output informs that there are four different combinations of values of these variables within the dataset:

```
File contains 14 variables
File contains 19 cases
Found 4 values for the variables ['Actor_Affiliation', 'Attack']. First 20:
       ['Actor_Affiliation=Dem; Attack=0', 'Actor_Affiliation=Dem;
       Attack=1', 'Actor_Affiliation=Rep; Attack=0', 'Actor_Affiliation=Rep;
       Attack=1']
```

From these values you may now select the value combinations which should be present in the subset.

3. Select the value 'Actor_Affiliation=Dem; Attack=1' and confirm.
4. Choose an output file and confirm.

Table 8: Subset of the data containing only cases where a democrat attacked an other actor. The argument variables were left out to save space.

Show_ID	Station	Date	Actor	Actor_ Affiliation	Quotation	Issue	Attack_Target
PB002	PBS	Mar 15	Obama, Barrack	Dem	Indirect	Iran	Foreign Actor
AB001	ABC	Mar 14	Van Hollen, Chris	Dem	Direct	Budget	Tea Party
AB002	ABC	Mar 15	Kerry, John	Dem	Indirect	Iran	Foreign Actor

23.4. How many times was each actor cited on each station?

In this short example, we try to find out how many times each of the actors was cited in each of the three different stations in the analysis. In order to answer this question, the following steps should be taken:

1. Open Nogrod and on the first page select 'Aggregate data' -> 'Create Dummy Table'. Confirm with 'Check'.
2. You are now asked for the input file. Browse the folder to find the file 'Example_Data.txt'. The dataset has a header and is separated by tabstopps. Select these options and confirm.

The console offers you an overview on the data just loaded. It should contain 14 variables and 19 cases. If you happen to have selected a wrong separator, the program would warn you.

3. Select the group variable 'Station' and the multinomial variable 'Actor' for this analysis. Set the method to be used to 'Count' as we are interested in how many times each actor appears on each station.

The console now offers some information on the variables selected. It informs you on the number of groups and values which will be used in the analysis. In addition, it provides the names of the groups and values. If there are more than 20 groups or values, only the first 20 are displayed.

```
File contains 14 variables
File contains 19 cases

Case-Variable: Station
3 Cases in this File. First 20: ['ABC News', 'Fox News', 'PBS']

Value-Variable: Actor
6 Values for multinomial variable in this File. First 20: ['Christie,
      Chris', 'Cruz, Ted', 'Kerry, John', 'Obama, Barrack', 'Ryan, Paul',
      'Van Hollen, Chris']
```

4. Optionally, you may now restrict the number of groups and values by adding a lower limit of appearances for both. If you only want to include actors which appear at least three times to exclude all unimportant actors, you may set the limit for values to '3'. In this small dataset, however, this would not make much sense. So just leave it and confirm with 'Check'.
5. You may now select an output file, either by writing a filename to the textbox or by browsing and setting a filename. It is highly recommended to leave the ouptut format at a textfile with header (with variable names) and tabstopps. You may, however, change it at will. Confirm with 'Check'.

The program now calculates the values for the dummy table and writes them in the output file. This happens instantaneously in this example as there are only 19 cases to evaluate. For datasets with several hundred or thousand entries, it may take some minutes.

The result of this transformation is a textfile containing a crosstable of sources and actors (Table 2). The first column is always labelled as '#Group' and contains values identical to the values which were present in the group variable.

Note that the names of the variables are not necessarily well formed. They may contain blanks, commata and any other special characters which were present in the values of the multinomial variable used for dummy aggregation. This is no problem for further data processing in Nogrod but it may pose a problem when importing the data to statistical software packages which do not allow special characters in variable names.

Table 9: Output of Dummy Transformation

#Group	Actor_Christie, Chris	Actor_Cruz, Ted	Actor_Kerry, John	Actor_Obama, Barrack	Actor_Ryan, Paul	Actor_Van Hollen, Chris
ABC News	0	1	2	1	1	1
Fox News	2	1	1	1	2	1
PBS	0	1	0	2	0	2

23.5. Which issues were covered in each of the TV shows?

For this question, you do not want the number of cases each issue appeared but just the shows that presented them. For the most part, the routine is handled as in the last example:

1. Select the file 'Example_Data.txt' as source file with header and tabstopp.
2. Select the 'Show_ID' as group variable and 'Issue' as multinomial variable, select 'Dichotomous' as method.
3. Type the name of a file for output and hit 'Check'.

The result may be seen in Table 10. You get a Dummy-Table which contains the value 1 for each show an issue was covered in.

Table 10: Dummy Table indicating the appearance of issues in TV shows

#Group	Issue_Budget	Issue_Iran	Issue_Obamacare
AB001	1	0	1
AB002	0	1	0
FN001	1	0	1
FN002	1	0	1
FN003	0	1	0
PB001	1	0	0
PB002	0	1	0
PB003	1	0	1

23.6. Which were the most frequent actors and issues on each station?

Here, the data is to be summarized on station level (three groups) and the actors and issues are to be aggregated to the most frequent value.

1. Select 'Aggregate Data' -> 'Aggregate Cases' as a procedure and load the example data.
2. Select 'Station' as the group variable.
3. Add 'Actor' and 'Issue' as variables to be aggregated. Select 'Most Frequent' as method.
4. Choose an output file and confirm.

The result of this transformation is a new table with three cases (for the networks) and four variables (table 10). The first variable is the group variable, which is the TV station in this example. The second variable provides the number of cases which belong to this group and were aggregated. Then, the aggregated variables themselves follow.

Table 11: Result of aggregation of actor and issue to station level.

Station	Number_of_Cases	Actor	Issue
ABC	6	Kerry, John	Obamacare
Fox	8	Christie, Chris	Budget
PBS	5	Obama, Barrack	Budget

23.7. Reshape the dataset to one show per line

Here, the data is summarized to the level of shows and all variables are aggregated by reshaping them to a broad view. For demonstration purposes and graphic limits, only three of the variables (Actor, Actor Affiliation, and Issue) are used.

8. Select 'Aggregate Data' -> 'Aggregate Cases' as a procedure and load the example data.
9. Select 'Show_ID' as the group variable.
10. Add Actor, Actor_Affiliation, and Issue as variables to be aggregated. Select 'Transform Dataset to broad format' as method.
11. Choose an output file and confirm.

The result is quite a broad table as all variables have been multiplied four times to make space for all data which was aggregated. The resulting table contains four copies of each variable as there is a maximum of four speakers in the shows of the example data. The aggregated lines are added in a chronological order, so Actor01 of AB001 is Ryan, followed by Kerry, Cruz, and Van Hollen. For Groups with less than four cases, missing values are entered to unused variables. As FN003 only has one actor, this case has nine missing values which might be used to specify actor 02, 03, and 04.

Table 12: Result of the aggregation to broad format. The table is cut in two parts as it is too broad to be displayed on one line.

Show_ID	Actor01	Actor_ Affiliation01	Issue01	Actor02	Actor_ Affiliation02	Issue02
AB001	Ryan, Paul	Rep	Obamacare	Kerry, John	Dem	Obamacare
AB002	Obama, Barrack	Dem	Iran	Kerry, John	Dem	Iran
FN001	Obama, Barrack	Dem	Budget	Van Hollen, Chris	Dem	Budget
FN002	Christie, Chris	Rep	Obamacare	Ryan, Paul	Rep	Budget
FN003	Christie, Chris	Rep	Iran			
PB001	Van Hollen, Chris	Dem	Budget			
PB002	Obama, Barrack	Dem	Iran	Cruz, Ted	Rep	Iran
PB003	Van Hollen, Chris	Dem	Budget	Obama, Barrack	Dem	Obamacare

Actor03	Actor_ Affiliation03	Issue03	Actor04	Actor_ Affiliation04	Issue04
Cruz, Ted	Rep	Obamacare	Van Hollen, Chris	Dem	Budget
Ryan, Paul	Rep	Budget	Cruz, Ted	Rep	Obamacare
Kerry, John	Dem	Budget			

23.8. Which arguments were seen on which show?

For this example, the argument variables have to be aggregated on show level. Since the presence of an argument in one statement means that it was on the show, you should pick 'Maximum' as a method.

1. Select 'Aggregate Dataset' as a procedure and load the example data.
2. Select 'Show_ID' as the group variable.
3. Add all 'Argument_'-variables as variables to be aggregated. Select 'Maximum' as method.
4. Choose an output file and confirm.

The result is a table containing 7 variables. First, again, the group variable and the number of cases per group are provided. Then, the maximum values for all selected variables are shown. Please note, that the maximum values are returned as decimal numbers.

Table 13: Arguments appearing on each show

Show_ID	Number_of_ Cases	Arugment_ Moral	Argument_ Economy	Argument_ Voter	Argument_ Justice	Argument_ Faith
AB001	4	0.0	1.0	0.0	1.0	1.0
AB002	2	1.0	0.0	1.0	0.0	0.0
FN001	4	1.0	1.0	0.0	1.0	0.0
FN002	3	1.0	0.0	1.0	0.0	1.0
FN003	1	0.0	0.0	0.0	0.0	1.0
PB001	1	0.0	1.0	0.0	1.0	0.0
PB002	2	0.0	0.0	0.0	0.0	0.0
PB003	2	00.	1.0	0.0	1.0	0.0

23.9. How many arguments are there in the statement?

For this example, the sum of all argument values for each case have to be calculated.

1. Select 'Aggregate Data' -> 'Aggregate Variables' as procedure and load the example dataset
2. Select all argument variables (Hint: You may type 'argu' to the textbox above the list to narrow the list to the argument variables. Then hit 'Add all' to add all these variables). Use the method 'Sum'
3. Select a name for the output variable (e.g. 'Sum_Argument')
4. Choose an output file and confirm.

The result of this calculation is a new variable (e.g. 'Sum_Argument') which contains the sum of all argument variables.

Table 14: Result of the calculation of the number of arguments in each statement. The new variable contains decimal numbers and is attached to the end of the table (some variables were removed to save space).

Show_ID	Station	Actor	Argument_ Moral	Argument_ Economy	Argument_ Voter	Argument_ Justice	Argument_ Faith	Sum_ Argument
FN001	Fox	Obama, Barrack	1	1	0	0	0	2.0
FN001	Fox	Van Hollen, Chris	0	0	0	0	0	0.0
FN001	Fox	Ryan, Paul	0	0	0	1	0	1.0
FN001	Fox	Cruz, Ted	0	1	0	0	0	1.0
FN002	Fox	Christie, Chris	1	0	1	0	0	2.0
FN002	Fox	Ryan, Paul	0	0	0	0	1	1.0
FN002	Fox	Kerry, John	1	0	1	0	0	2.0
FN003	Fox	Christie, Chris	0	0	0	0	1	1.0
PB001	PBS	Van Hollen, Chris	0	1	0	1	0	2.0
PB002	PBS	Obama, Barrack	0	0	0	0	0	0.0
PB002	PBS	Cruz, Ted	0	0	0	0	0	0.0
PB003	PBS	Van Hollen, Chris	0	1	0	1	0	2.0
PB003	PBS	Obama, Barrack	0	0	0	0	0	0.0
AB001	ABC	Ryan, Paul	0	0	0	0	1	1.0
AB001	ABC	Kerry, John	0	0	0	1	0	1.0
AB001	ABC	Cruz, Ted	0	0	0	0	1	1.0
AB001	ABC	Van Hollen, Chris	0	1	0	0	0	1.0
AB002	ABC	Obama, Barrack	1	0	0	0	0	1.0
AB002	ABC	Kerry, John	0	0	1	0	0	1.0

23.10. Diversity of the networks with respect to actor affiliation

For this example, the group variable is the network and the multinomial variable in question is the actor affiliation.

1. Select 'Aggregate Data' -> 'Calculate Entropy' as a procedure.
2. Choose the example data as input table (tabstopps and header) and confirm.
3. Select 'Station' as group variable and 'Actor_Affiliation' as multinomial variable.
4. For the method you may select either option. Since both affiliations appear in all networks, the result will be the same. It would only make a difference for values which are unevenly distributed across groups.
5. Choose an output file and confirm.

The resulting table always contains three Variables. The first variable denominates the group (value of the group variable). The second variable indicates the number of cases within each group. The third variable gives the entropy as floating number.

For this example you may note that the entropy is highest in Fox as republicans and democrats are distributed 5:3. This is almost an equal distribution. It is, however, low for PBS where the ratio is 1:4.

Table 15: Output for the calculation of actor affiliation entropy across groups

Station	N_Elements	Entropy_Actor_Affiliation
ABC	6	0.9183
Fox	8	0.9544
PBS	5	0.7219

23.11. Actor Diversity across networks

As in the example above, the diversity within networks has to be calculated here. In this example, however, the values in each group differ slightly as not all networks have the same population of actors. This means that it will make a difference which method you use here.

1. Select 'Aggregate Data' -> 'Calculate Entropy' as a procedure.
2. Choose the example data as input table (tabstopps and header) and confirm.
3. Select 'Station' as group variable and 'Actor' as multinomial variable.
4. Choose one of the options.
5. Choose an output file and confirm.
6.

For this example both methods were used to compare the results. As you may see below, the entropy scores differ strongly depending on the method used. When using all actors as a population, FOX has the highest entropy as it shows all actors and prefers none. ABC has a slightly lower entropy as one actor (Chris Christie) is missing. The lowest entropy may be observed in PBS which has several missing actors.

When using only the actors within each group as population, the entropy scores are much closer together. Other than with the full population, missing actors do not count when using this method. This results in higher scores for all groups (except for FOX where all actors are present).

Table 16: Result for entropy of actors using all actors as population.

Station	N_Elements	Entropy_Actor
ABC	6	0.8710490
Fox	8	0.9671320
PBS	5	0.5887621

Table 17: Results for entropy of actors using only the actors within each group as population.

Station	N_Elements	Entropy_Actor
ABC	6	0.9697239
Fox	8	0.9671320
PBS	5	0.9602297

23.12. How probable is the discussion of certain different issues on the same show?

Here, the probability of co-occurrence of issues within shows is to be calculated.

1. Select 'Co-Occurrence for a Multinominal Variable' and open 'Example_Data.txt' (with header and tabstopps)
2. Select 'Show_ID' as group variable and 'Issue' as multinomial variable. Select Probability of Co-occurrence as method.
3. Set no required minimum for groups and values.
4. Select a file for output and confirm.

Table 18: Result of the co-occurrence analysis of issues within shows. While Iran is never mentioned in the same show with any of the other issues, Budget and Obamacare have a large overlap. They co-occur in 80% of cases. The values in the diagonal are always 10

	Issue_Budget	Issue_Iran	Issue_Obamacare
Issue_Budget	1	0	0.8
Issue_Iran	0	1	0
Issue_Obamacare	0.8	0	1

23.13. Which arguments are often mentioned together?

In this example, you do not need a group variable and a multinomial variable but just the dummies provided in the variables 'Argument_xx'. These five variables are already dummy variables which may be used for an analysis of co-occurrence.

1. Select 'Co-Occurrence Matrix from Dummy Table' as procedure and load the example dataset.

The tool now gives a text output informing you on the suitability of each variable as a dummy variable:

```
File contains 14 variables
File contains 19 cases
Show_ID: Not suitable as Dummy Variable. Invariant.
Station: Not suitable as Dummy Variable. Invariant.
Date: Not suitable as Dummy Variable. Invariant.
Actor: Not suitable as Dummy Variable. Invariant.
Actor_Affiliation: Not suitable as Dummy Variable. Invariant.
Quotation: Not suitable as Dummy Variable. Invariant.
Issue: Not suitable as Dummy Variable. Invariant.
Arugment_Moral: Numeric Variable: 15 / 4
Argument_Economy: Numeric Variable: 14 / 5
Argument_Voter: Numeric Variable: 16 / 3
Argument_Justice: Numeric Variable: 15 / 4
Argument_Faith: Numeric Variable: 15 / 4
Attack: Numeric Variable: 12 / 7
Attack_Target: Non-Numeric Variable: 12 / 7
```

From this output you can judge that most variables are not suitable as dummies as they are invariant. The Argument variables, as well as 'Attack' and 'Attack_Target' are suitable as they contain different values. The numbers (e.g. 15/4) indicate the number of 0 and 1 in the variables

(e.g.: Attack has 12 zeroes and 7 ones). Note, that for string variables all strings are counted as '1', whereas all missings are counted as '0'.

2. Select all Argument variables from the list and add them to the list of variables to be used as dummies. Choose 'Count' as method.
3. Select a file for data output and confirm.

Table 19: Co-occurrence of arguments in the same statement. The diagonal is always the sum of occurrences of each variable

	Argument_ Economy	**Argument_ Faith**	**Argument_ Justice**	**Argument_ Voter**	**Arugment_ Moral**
Argument_Economy	5	0	2	0	1
Argument_Faith	0	4	0	0	0
Argument_Justice	2	0	4	0	0
Argument_Voter	0	0	0	3	2
Arugment_Moral	1	0	0	2	4

Example: Who attacks whom?

In the dataset there is a variable naming the target of attacks for each statement. If there is no attack, the value of this variable is missing. This data may be used to create a social network.

1. Select the procedure 'Create Social Network Visualization' and open the example data (with header and tabstopps)
2. The subject of this network will be the actors from variable 'Actor'. The objects are the actors from 'Attack_Target'. The subjects and objects are coded by the same rule. This means that Obama, Barrack as a target of an attack is the same node as Obama, Barrack attacking a foreign actor.
3. You might set a lower limit of appearances to reduce the number of nodes. With this example data, however, this is not recommended as we would lose most nodes. As we are just interested in the presence of attacks and do not have any detailed information on the attacks, there is no relation variable to select. The method 'Count' may be selected for this example. Any other method will do as well.
4. You now have to choose two output files. The first file contains the network information. This file will be needed to visualize it. The second file only contains the information how many times each node appeared in the data. This information may be included to the network.

The results of this transformation are two tables. The first table (adjacency matrix) looks like a co-occurence matrix with the exception that it is not symmetrical. In the adjacency matrix, the senders are the rows of the table, whereas the objects are the columns. A number in row 3 and column 5 would therefore mean that the element on row 3 has a link to the element in column 5.

In the current example, you may see that Joe Biden does not utter any attack on other persons but that he is attacked by Paul Ryan once. Similarly, Paul Ryan is never the object of attacks but twice the subject.

Table 20: Adjacency Matrix

_Case	Biden, Joe	Christie, Chris	Cruz, Ted	Foreign Actor	Kerry, John	Obama, Barrack	Ryan, Paul	Tea Party	Van Hollen, Chris
Biden, Joe	0	0	0	0	0	0	0	0	0
Christie, Chris	0	0	0	1	0	0	0	0	0
Cruz, Ted	0	0	0	0	0	1	0	0	0
Foreign Actor	0	0	0	0	0	0	0	0	0
Kerry, John	0	0	0	1	0	0	0	0	0
Obama, Barrack	0	0	0	1	0	0	0	0	0
Ryan, Paul	1	0	0	0	0	1	0	0	0
Tea Party	0	0	0	0	0	0	0	0	0
Van Hollen, Chris	0	0	0	0	0	0	0	1	0

The second table has only two columns. The first contains the ID of the nodes (variable _Case) in the adjacency matrix table, the second column indicates the number of times a node appeared (Table 21).

Table 21: Attribute Matrix with node counts. Even nodes not appearing in the social network are listed here, such as the missing value

id	Count
	12
Biden, John	1
Christie, Chris	2
Cruz, Ted	3
Foreign Actor	3
Kerry, John	3
Obama, Barrack	6
Ryan, Paul	3
Tea Party	1
Van Hollen, Chris	4

23.14. Finding peak skin conductance during a movie

As the example dataset does not contain any time series data which may be used to detect peaks, an additional dataset with time series data from physiological measurements is used in this example. The dataset 'Example_Timeseries.txt' contains the heartrate, electrodermal resistance and blood pressure of a subject watching a short movie.

To find moments with increased or decreased conductance, you may use the function Peak Detection

1. Select the procedure 'Detect Peaks' from 'Time Series' and open the example time series data (Example_Timeseries.txt) (with header and tabstopps)
2. Select the input variables. The variable 'Time' contains the timestamps in seconds. The variable 'SkinCond' contains skin conductance data. Choose any name for the new variable which will be created (e.g.: Peak_Skin).

3. In this example we look for both positive and negative peaks. The confidence interval may be set to low 80% which means that a peak will only be considered a peak if the value is among the highest or lowest 10%.
4. Select a name and format for the output file and confirm (e.g.: Skincond.txt).

The result is a table exactly corresponding the input table with one additional variable. The variable has the value 0 for most of the time but contains few cases with value 1 (positive peaks) and -1 (negative peaks).

Since the data is quite extensive, R is used to visualize the result. As you may see from the picture, the function did identify regions with increased and reduced skin conductance but does not offer much information on the curve. Especially since small peaks which are detectable by eye are not indicated as they are within the 80% confidence interval.

The code in R was:

```
dta <- read.table("../Skincond.txt", header=TRUE, sep="\t")
plot (dta$Time, dta$SkinCond, type="l", main="Skin Conductance with
      detected peaks", xlab="Time in Seconds", ylab="Skin Conductance")
points(dta$Time[dta$Peak_Skin>0],
      dta$Peak_Skin[dta$Peak_Skin>0]*.2+2.3,col="green",pch="^")
points(dta$Time[dta$Peak_Skin<0],
      dta$Peak_Skin[dta$Peak_Skin<0]*.2+2.3,col="red",pch="v")
```

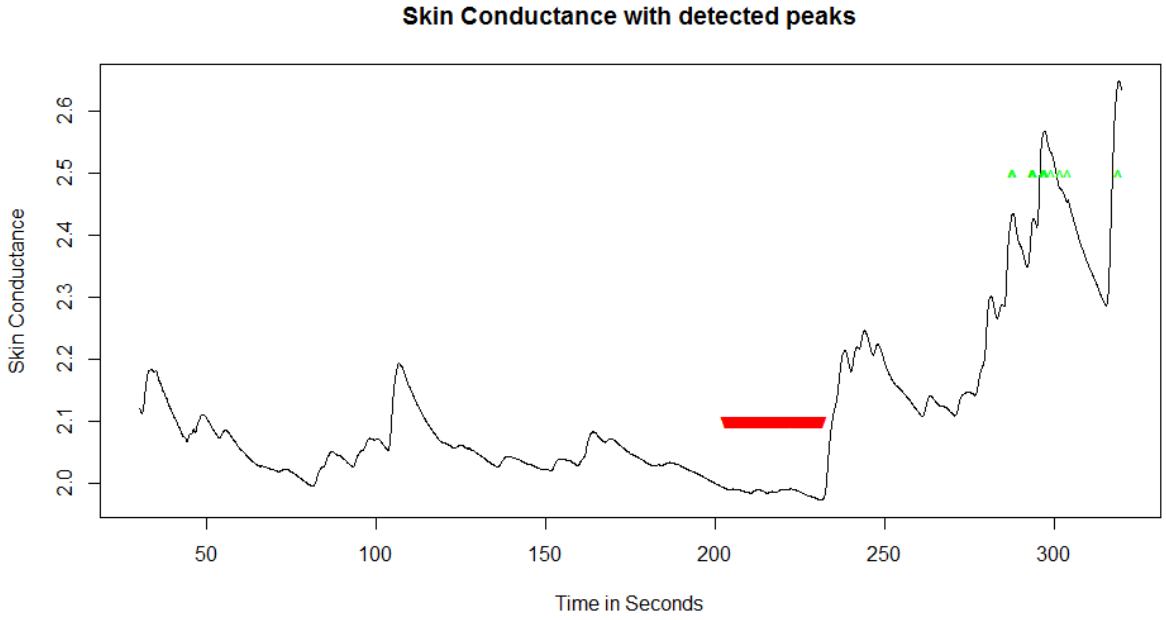


Figure 14: Skin conductance curve with indicated positive and negative peaks. Green triangles indicate positive peaks, red triangles indicate negative peaks.

23.15. Find peak increases in skin conductance

For this purpose, the same dataset as above is used, but the interest is slightly different. This time, the research interest does not concern the highest and lowest skin conductance over a long period of time, but small changes in skin conductance during this period. As the curve has been shown to exhibit a moving average, these changes may not be found by simply identifying high and low values. By subtracting the moving average from the curve, however, small changes may be observed easily.

1. Select the procedure 'Flatten moving average' from 'Time Series' and open the example time series data (Example_Timeseries.txt) (with header and tabstopps)
2. Select the input variables. The variable 'Time' contains the timestamps in seconds. The variable 'SkinCond' contains skin conductance data.
3. As Skin conductance takes some seconds to increase or decrease, it seems reasonable to set the gliding window for the moving average to 5-10 seconds.
4. Now you may enter two names for the two variables which are created (e.g.: Skin_MA and Skin_Dev)
5. Select a name and format for the output file and confirm (e.g.: Skincond.txt).

Again, the result is displayed in R as the data is too extensive and uninstructive as a table. To show the curves in one image and without overlap, the curves have been displaced vertically. The moving average (blue) is 0.1 points lower and the deviation of the moving average (red) is raised by 1.8 points in the graph. In the data, the moving average lies exactly on the time series data and the average of deviations is around 0.

The R code looks as follows:

```
dta <- read.table("../Skincond.txt", header=TRUE, sep="\t")
plot (dta$Time, dta$SkinCond, type="l", ylim=c(1.7,2.7), main="Skin
      Conductance with moving average and deviation", xlab="Time in
      Seconds", ylab="Skin Conductance", lwd=2)
lines(dta$Time, dta$Skin_MA-.1,col="blue")
lines(dta$Time, dta$Skin_Dev+1.8,col="red",pch="v")
```

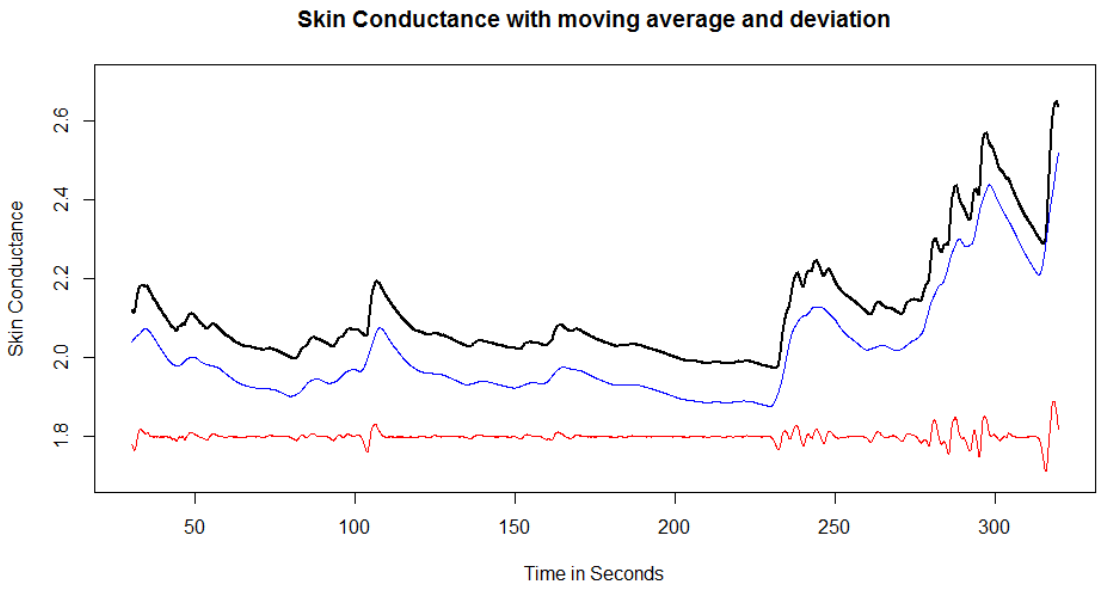


Figure 15: Graphical illustration of the moving average (blue) and deviation (red) of the skin conductance curve. The moving average was computed using a 5 second gliding window.

In a next step, the peaks of the deviation curve (now stripped from the moving average) may be used to detect the real peaks of the curve. As the curve only has positive peaks with intervals of low relaxation, only positive peaks are considered now.

6. Select the procedure 'Detect Peaks' from 'Time Series' and open the file created above (e.g.: Skincond.txt)
7. Select 'Time' and 'Skin_Dev' as input variables and an output variable (e.g.: Skin_Peaks).
8. Select positive peaks and a 90% confidence interval to consider only the top 5% of values.
9. Select a name and format for the output file and confirm (e.g.: Skincond.txt).

Again, the result is visualized in R. The code is as follows:

```
dta <- read.table("../Skincond.txt", header=TRUE, sep="\t")
plot (dta$Time, dta$SkinCond, type="l", main="Skin Conductance with real
      peaks", xlab="Time in Seconds", ylab="Skin Conductance")
points(dta$Time[dta$Skin_Peaks>0], dta$Skin_Peak[dta$Skin_Peaks>0]*.2+2.3,
      col="darkgreen",pch="^",cex=2)
```

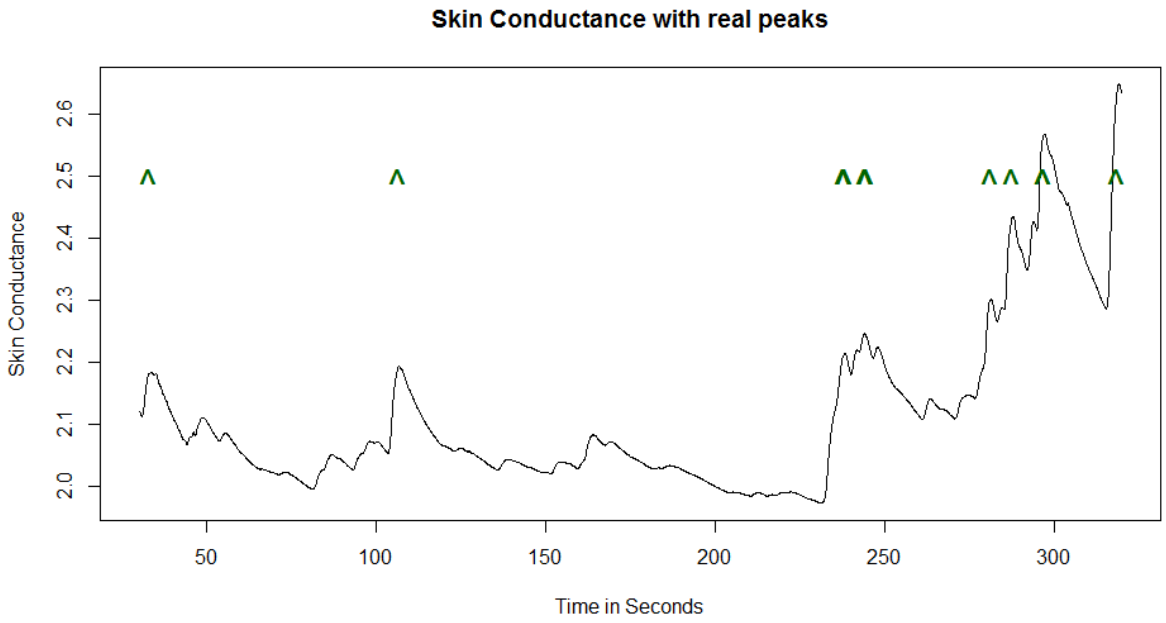


Figure 16: Detected peaks of the skin conductance curve when stripped of the moving average. It may be observerd that not all peaks have been found due to the high confidence interval of 90% which was selected.

In this example you may see that the routine detected large and small peaks independent from their baseline skin conductance. Whenever there is a peak increase, a peak is detected. The new variable may now be used to identify the exact time when sudden increase of skin conductance occurs.

Chapter 24. Glossary of Nogrod functions

Nogrod may be included as a module to any Python 2.7 script. As some functions bear very common names, such as 'aggregate' or 'get_data', it is recommended to import Nogrod as a whole (`import Nogrod`) instead of its components (`from Nogrod import *`). You may, however, also choose to import just some of the functions (e.g.: `from Nogrod import reltest`) if you do not need all of them but just want one basic function included.

Due to the interaction of functions within Nogrod, it is not advised to just copy+paste individual functions. This may result in complicated error handling.

Some notes on formats

The standard Nogrod dataset is a list (or Tuple) with at least two elements. The first position is the data dictionary. The second position is the correctly ordered list of variables.

A data dictionary is a dictionary with variable names as keys and the complete data vectors as values. The data vectors are lists of numeric or string elements.

A variable list is a list of the variables occurring in the dataset in correct order. As the keys of dictionaries tend to scramble, this list is required to keep the order of columns in the dataset. All variables in the variable list should be present as keys in the data dictionary. It is not necessary, however, to include all keys of the dictionary in the variable list. There may be some hidden vectors. These vectors, will be disregarded in any function the dataset is sent to (e.g.: They will not be sorted by sort_dataset and will not be written by write_dataset)

24.1. Basic Input/Output functions

get_dataset(fname, header=1, sep='\t')

This function loads a file, which is specified in fname and returns a list of four elements:

1) A data dictionary with variable names as keys and lists of cells as values. This is the standard representation of data in Nogrod.
2) A list of variables, sorted as they are in the original file.
3) A string containing a summary of the data import (usually the number of variables and cases)
4) A string containing error messages

The output of this function may be used as a standard dataset in Nogrod (i.e. directly used as input for analysis procedures).

Options:

`fname`	Filename. If path is omitted, the current working directory is assumed as path
`header`	Are there variable names in the first line of the table? 1: Yes, 0: No
`sep`	Cell separator. Default value is tabstop (\t)

write_dataset(data, filename, header=1, sep='\t') :

This function writes data to a file. The data has to be a list or tuple, containing at least two elements: Element 1: Data dictionary with variable names as keys and lists of cells as values (standard Nogrod data), Element 2: A list of variables to write to the file.

Options:

`data` A list or tuple. data[0] is a data dictionary, data[1] the list of variables to be written. This format is returned by most functions in Nogrod.

`fname` Filename. If path is omitted, the current working directory is assumed as path

`header` Should variable names be written to the first line of the table? 1: Yes, 0: No

`sep` Cell separator. Default value is tabstop (\t)

baum_schreiben(tdic, einr = 0, inherit = '', spc = ' ', trunc=0, lists=False)

This function takes a variable of type dict and returns a string, showing the whole contents of the variable as an indented tree of dictionaries. The function is recursive and can handle any number of nested sub-trees.

If the parameter `lists` is set to true, the function also expands nested lists and prints them one element per line. Since this would result in excessive outputs for datasets in Nogrod, this parameter is set False by default. If you want to print JSONS from APIs (e.g. Twitter, Tumblr, Facebook...) you may set it to True to expand all lists and all dictionaries inside.

Options:

`tdic` Dictionary to be printed. If `lists` is set True, it may also be a nested list instead of a dictionary.

`einr` level of the indentation (used for recursive function). If set to an integer value above 0, the whole tree will be indented by `einr` levels.

`inherit` inherited string (used only for recursive function. Do not change)

`spc` spacer used for the indentation. Default is four blanks. It may be changed to any other indentation (tabstop, hyphens, dots, etc.)

`trunc` trunctuation of large contents. If set to 0, the contents of each key are included completely in the output. If set to a numeric value, large contents are trunctuated to the number of characters specified.

Trunctuated entries are cut in the middle and get a remark at the end of the line indicating the number of characters they actually have.

`lists` Boolean parameter indicating whether lists are to be expanded as well. If there are several long (N>1000) lists in your dictionary, do not set this to true. It may get messy. This is the case for large Nogrod datasets.

24.2. Basic data transformation functions

sort_dataset(dset, variables, descend=False) :

This function takes a dataset and a list of variables to sort it. The function returns the sorted dataset.

The input and output format are standard Nogrod datasets, consisting of a data dictionary and a list of variables.

Options:

`dset`	A list or tuple. dset[0] is a data dictionary, dset[1] the list of variables in this dataset.
`variables`	A list of variable names or one single variable name in string format. The data is sorted by all variables. The first variable is the chief sorting key, the last one is the least important.
`descend`	Boolean variable specifying whether the data is sorted descending.

dummytab(data, cases,keylist,mode='dicho',min_case=0,min_anz=0):

This function takes two lists of equal length specifying groups and values to create a dummy table indicating which values appear in which groups. The function returns a standard dataset with data dictionary and variable list.

Note: There is also a second function, called `dummy()`, which takes only one grouping variable and returns a data table. This function is not recommended for use outside Nogrod and has mainly internal functions.

Options:

`data`	A data dictionary
`cases`	A list of variables specifying the groups or the name of the variable specifying the groups.
`keylist`	A list of values specifying the value for each case.
`mode`	Mode to compute the cells in the dummy table. • dicho: Dichotomous decision whether the value is in the group (1) or not (0) • anz: Number of times a value occurs in each group • log: Logarithmized number of occurrences
`min_case`	Minimum number of groups a value should occur in to be counted as a value. If set to 0, all values are counted.
`min_anz`	Minimum number of values a group should have to be counted as a group. If set to 0, all groups are counted.

delete_missing(data,varlist,n_allowed=0)

This function takes a data dictionary and a list of variables and returns a data dictionary where all cases with missing values on the specified variables are removed.

Options:

`data` A data dictionary

`varlist` A list of variables where missing values are not allowed

`allowed` Number of missing values allowed on these variables per case. (e.g.: If four out of five variables should have a value, allowed may be set to 1. If more than 1 values are missing, the case is removed)

aggregate(dset,key,var,method,weight=0,master=''):

Aggregate a dataset to groups specified in one or several key variables. A method for aggregation (summing, mean, etc.) can be specified for a list of variables.

Options:

`dset` Dataset in standard Nogrod format (`[data,variables]`) or a data dictionary.

`key` Name of the key variable or list of key variables. The key variables denominate the groups.

`var` List of variables to aggregate

`method` Method for data aggregation

- 'mean': Mean of values
- 'wmean': Weighted mean (requires zipped list in `liste`)
- 'sum': Sum of values
- 'wsum': Weighted sum (required zipped list in `liste`)
- 'sd': Standard deviation
- 'min', 'max', 'range': Minimum, Maximum and range of the values
- 'freq': Most frequent element
- 'first': First element
- 'last': Last element

`weight` Weighting factor. List of numeric values. If set to 0, the weight is a list of the value 1.0 with the length of the dataset.

stat_desc(values,weight=0)

Generate descriptive statistics of a list of values. The function returns a dictionary with complete descriptives:

- Val: List of numeric values
- Weight: List of weighting factor for each value
- N: Number of numeric values
- Sum: Sum of numeric values
- Err_Val: Non-numeric values
- Err_Weight: Non-numeric weighting factors

- Err_Both: Non-numeric values without valid weighting factor
- N_Total: Total number of values (numeric and non-numeric)
- M: Mean
- SD: Standard deviation
- Vari: Variance
- Min: Lowest numerical value
- Max: Highest numerical value
- Range: Range of the list (Max-Min)

Options:

`values` List of values (numeric, non-numeric, mixed)

`weight` List of weighting factors of equal length as `values`.

`allowed` Number of missing values allowed on these variables per case. (e.g.: If four out of five variables should have a value, allowed may be set to 1. If more than 1 values are missing, the case is removed)

24.3. Basic calculations for uni- and bivariate descriptives

calculate(liste,method)

Calculate one specific property of a list of values. The function returns one numeric value as a result.

Options:

`liste` List of values (numeric, non-numeric, mixed) or a zipped list of values and weighting factors.

`method` Value to be computed:

- 'mean': Mean of values
- 'wmean': Weighted mean (requires zipped list in `liste`)
- 'sum': Sum of values
- 'wsum': Weighted sum (required zipped list in `liste`)
- 'sd': Standard deviation
- 'min', 'max', 'range': Minimum, Maximum and range of the values
- 'freq': Most frequent element
- 'first': First element
- 'last': Last element

calculate_vectors(vectors,method,master=0)

Calculate new variable as a function of a list of variables or vectors. The function takes a list of list of values and returns a list of values.

This function may, for example, be used to add up two variables or get the maximum of five dummy variables.

Options:

`vectors` List of lists of values (numeric, non-numeric, mixed)

`method` Value to be computed:

- 'concat': Concatenate all values to one string variable.
- Any method that may be used in calculate()

calc_regression(x,y)

Simple and quick OLS regression analysis of two numeric vectors. The function takes two lists of numeric values and returns a list of two Values. The first value is the unstandardized slope (B), the second value is the intercept (A). Standard errors, explained variance and other results are not provided.

Options:

`x` Independent Variable. List of numeric values (no missings)

`y` Dependent Variable. List of numeric values (no missings)

calc_correlation(l1,l2)

Simple and quick correlation analysis of two numeric vectors. The function takes two lists of numeric values and returns the pearson correlation as floating point number.

Options:

`l1` First list of numeric values

`l2` Second list of numeric values

calc_entropy(liste,maxliste=[])

Quick calculation of the entropy of a list of elements. The function takes a list of elements and computes the entropy of these elements, either in relation to a known population of elements or in isolation. The function returns the Entropy (non-standardized!) as a floating point number.

Options:

`liste` List of values (any format) to be analyzed.

`maxliste` Optional list of all available values. If an element is present in `maxliste` but not in liste it is counted as an element with frequency 0. If no `maxliste` is provided, no element may have the frequency 0.

calc_chisquare_dummy(l1,l2,directed=0,cramersv=0)

Quick calculation of a Chi-Square coefficient of two dummy lists. The function takes two lists of dummy-coded values (1 and 0) and returns the chi-square coefficient as a floating point number. Using additional parameters, a directed or standardized coefficient may be returned.

Options:

`l1, l2` Two lists consisting of the values 1 and 0.

`directed` Integer of value 1 or 0. If set to 1, the Chi-square coefficient is returned with a sign indicating the dominant diagonal. If 11 and 00 are more frequent than 10 and 01, the sign is positive. If the off-diagonal is more prominent, it is negative.

`cramersv` Integer of value 1 or 0. If set to 1, Chi-square is standardized and Cramer's V is returned. The value of Cramer's V ranges between 1 and 0.

24.4. Data analysis

The data analysis procedures in Nogrod usually come with an extended output during computation. In the GUI version of Nogrod, this output is written in the text output window of the GUI. When imported to another program without predefined text output, these notes (progress bars, summaries, e.t.c.) are written in the console.

Parameter master

Each data analysis function has one parameter at the end, which is called `master`. Upon execution, the function will try to write the text in a TKinter text widget called `master.Artikel`. If no such widget exists, the text is written to the console by standard print command. If you wish to get the text output in a TKinter object, just create a window with a text widget called `Artikel`. You may then set the master parameter to this window to print all outputs nicely.

You may also define the different tags of this output (`'text'`, `'warning'`, `'progress'`, `'table'`, `'title'`) to adjust the fonts. Check the function `define_styleset()` in Nogrod for examples.

If you do not wish any output during the calculations but just want the result, you may set the parameter to silent: `master='silent'`. The output is then suppressed.

calc_ic_reliability(dset,uvar,cvar,varlist,unitlist=[],coderlist=[],core_cod='no_core', methods=['PA'],options=[],master='')

Calculate intercoder reliability from a table of codings. The table should contain one variable identifying the unit of analysis and one variable identifying the coder.

The result is a tuple with four elements:

- A short summary of the reliability test, giving the key results. The tables in this summary are printed with tab-spaced cells and can be imported in Excel.
- A long summary of the reliability test, giving all pairwise agreements, variable statistics, and much more information. All tables are printed with tab-spaced cells and can be imported in Excel.

- numres: A short dictionary with the overall results for each coefficient (mean values over all variables)
- nresults: A long dictionary with the results for all coefficients and all variables. The dictionary has the format `nresults[Variable][Coefficient]`. The coefficients are named as they are in the methods parameter. If there is a core coder, there are additional coefficients with the suffix `'_w_core'` indicating the mean agreement with the core coder on this variable. Note that the nresults dictionary may include additional coefficients which were not demanded (e.g.: if `'PA'` is computed, all Kappa measures are in it as well)

Options:

`dset` Codings. Dataset in standard Nogrod format (`[data,variables]`) or a data dictionary.

`uvar` Variable denoting the unit of analysis

`cvar` Variable denoting the coder

`varlist` List of variable names on which the test should be performed

`unitlist` Optional list of units of analysis to be included. If left empty, all units are included

`coderlist` Optional list of coders to be included. If left empty, all are included.

`core_cod` Optional name of the core coder. If left on the default `'no_core'`, there will be no comparison with a core coder.

`methods` List of coefficients to calculate. The coefficients are:

- `'PA'`: Percent agreement (Holsti)
- `'Kappa'`: Cohen's Kappa
- `'Kappan'`: Brennan and Prediger's Kappa
- `'Pi'`: Scott's Pi
- `'PRF'`: Precision, Recall, and F-Calue. Only to be used if there is a core coder
- `'Lotus'`: Fretwurst Lotus
- `'SLotus'`: Standardized Lotus
- `'Alpha Nominal'`: Krippendorff's Alpha for nominal variables
- `'Alpha Ordinal'`: Krippendorff's Alpha for ordinal variables
- `'Alpha Metric'`: Krippendorff's Alpha for metric variables

`options` List of special options for the reliability test:

- `'min2'`: A minimum of two codings is required to calculate any coefficient
- `'min5'`: A minimum of five codings is required to calculate any coefficient
- `'cm'`: Count Missings. If a coder has a missing value on a variable, this missing value is counted as a code.
- `'km'`: Missing Kappa if invariant. If there is no variance on a variable and the chance agreement is 100%, no kappa value is computed.

analyze_entropy(data,group,varlist, opt={'comp':0,'boot':0},report = '',master='')

Analysis of within and without entropy of a dataset. The function takes a data dictionary, a group variable and a list of variables to perform an analysis of entropy. It returns a detailed report in string format.

Options:

`data` Data dictionary

`group` Name of the group variable

`varlist` List of variables to be analyzed (conceptually, they should be nominal variables)

`opt` Dictionary with two keys:

- `'comp'` may have the value 1 and 0. If set to 1, the ratio of within and without entropy is compared to the case where all groups are randomly assigned.
- `'boot'` may have the value 1 and 0. If set to 1, the ratio of within and without entropy is bootstrapped (N=10'000). This may take some time.

`report` Optional header of the report to be returned in the end.

`master` TKinter object (see introduction to this chapter)

co_occurrence(data,dmode='dummy',method='anz',margin=1,prefix='',cases='',keyvar='',min_case=0,min_anz=0,dummylist=[],master='')

Co-occurrence analysis of either a dummy table or two variables, where one variable specifies the case and the other the values.

Co-occurrence analysis assumes that each of M cases (i.e. a row in a dummy table or all rows with identical case identifier) may have any number of N values (i.e. dummy-variables with value 1 or values of a nominal variable). For each pair of values, the analysis counts the cases in which both occur, only one occurs and none occur. It then computes a coefficient of co-occurrence for each pair and stores them in a matrix.

The output of the analysis is an NxN matrix indicating the coefficient of co-occurrence of values. The method to calculate this coefficient may be specified in the parameters.

The default values of the function are set to take a dummy table and return a matrix indicating for each pair of variables the number of rows they were both set to 1.

Options:

`data` Data dictionary containing either a dummy table or at least two variables

`dmode` Mode of the data. The mode may either be 'dummy' or 'nominal'.

- If set to `'dummy'` one case is considered one row in the table and a list of dummy variables is expected to indicate the presence or absence of values in this case.
- If set to `'nominal'`, two variables have to be specified. A group variable specifies the cases and a key variable holds the values to be associated with these cases.

`method` Method for the calculation of values in the co-occurrence table:

- `'dicho'`: Dichotomous co-occurrence. The value is 1 if values ever co-occur within a group and 0 if they don't.
- `'anz'`: Count of co-occurrences
- `'prob'`: mutual conditional probability of values (product of conditional probabilities)
- `'cprop'`: uni-directional conditional probability. The resulting co-occurrence matrix is not symmetrical. It indicates the probability that row occurs when column is present.
- `'einf'`: Proportional co-occurrence, percent agreement, Holsti.
- 'inv_prob': Inverted mutual probability (1/prob)
- `'sokal'`: Sokal distance between values
- `'eukl'`: Euclidian Distance between values
- `'chi1'`: Chi-Square
- `'chi2'`: Directional Cramer's V

`margin` Integer of value 1 or 0. If set to 1, margins are added to the table to indicate the overall occurrence of each value. The margins are calculated by computing the co-occurrence of each value with an element occurring exactly once in every case.

For mutual and conditional probabilities and simple co-occurrence, this measure gives the overall share of occurrences of each value. The euclidian distance is the number of cases a value does not occur in.

`master` TKinter object (see introduction to this chapter)

find_cluster(dset,varlist,outfile,add_outputs={'ssa':0,'dendro':0,'hist':0,'dist':0,'vector':0},row_std=0,table_std=0,master='')

Hierachical Exploratory Cluster Analysis with Noise Exclusion (HECANE) of a dataset containing count data.

The cluster analysis algorithm is optimized for finding homogeneous clusters in noisy count data, for example in data of content analyses. The function takes a dataset and a list of variables to be clustered and writes the report to a specified output file.

The algorithm may also generate additional outputs, such as R-Scripts for Smallest Space Analysis and Dendrograms, as well as detailed logs.

Options:

`dset` Dataset in standard Nogrod format (`[data,variables]`) or a data dictionary.

`varlist` List of variables to be clustered (the algorithm only clusters Variables)

`outfile` Name of the output file.

`add_outputs` Dictionary specifying additional outputs:

- `'ssa'`: If set to 1, an R script is generated that visualizes all relevant elements (noise exluded) and cluster centers in an SSA. Package smacof is required to display the data correctly.

- `'dendro'`: If set to 1, an R script is generated that visualizes the complete Dendrogram (inlcuding noise).
- `'hist'`: The history of added elements
- `'dist'`: Distance between elements and cluster centers. Also includes the distance between cases and cluster centers.
- `'vector'`: Complete vector table, adding the vectors of all clusters found in the analysis (even non-relevant ones) to the initial dataset. Standardized according to input.

`row_std` Integer of value 0 or 1. If set to 1, all rows are z-standardized.

`table_std` Integer of value 0 or 1. If set to 1, the table is z-standardized.

`master` TKinter object (see introduction to this chapter)

kmeans(data,varlist,direction=1,num=4,iterations = 100, precenter = {},master='')

K-means cluster analysis. The function takes data, a set of variables, and a number of initial cluster centers to perform a k-means cluster analysis and return the cluster informations on each cluster.

The result is a dictionary with the cluster numbers as keys and dictionaries as values. Each dictionary contains two elements: the coordinates of the Center of the cluster and its members.

Example result: Three clusters in four dimensions.

```
{0: {
    Center: [51.0, 8.0, 17.5, 11.5]
    Members: [1, 2]
    }
1: {
    Center: [1.2222222222222223, 0.1111111111111111, 0.1111111111111111,
       1.0, 0.3333333333333333]
    Members: [0, 7, 8, 9, 10, 11, 12, 13, 14]
    }
2: {
    Center: [37.5, 9.0, 9.5, 11.0]
    Members: [4, 5]
    }}
```

The clustering algorithm checks for the definite membership of clusters. If an element is equally distant to two cluster centers, it does not count this element toward any of the two.

Options:

`dset` Dataset in standard Nogrod format (`[data,variables]`) or a data dictionary

`varlist` List of variables to use in the clustering algorithm

`direction` Direction of clustering: 1: Cases are clustered. 2: Variables are clustered

`num` Number of cluster centers (K)

`iterations` Maximal number of iterations to prevent endless iteration. Usually, it finishes in less than 10 iterations.

`precenter` Dictionary specifying the cluster centers of the initial solution

`master` TKinter object (see introduction to this chapter)

calc_adios(data,symb,gvar='',tvar='',mini=0,rep=[1,3],length=[2,7],eta=.9,subst=1, master=''):

Grammar induction from sequence data, using an adaption of the ADIOS algorithm (Solan 2005). The function takes a data dictionary containing at least one variable representing the sequence to find repetitious patterns and common paths in this sequence.

It then returns a detailed report on the patterns in this sequence.

Options:

`data`	Data dictionary
`symb`	Name of the variable containing the symbol
`gvar`	Optional name of the variable containing the group. If left empty, the contents of the sequence variable are considered one large sequence
`tvar`	Optional name of the variable containing the timestamps. If left empty, the dataset is considered as sorted by time.
`mini`	Minimal length of repetitions to be counted as element of the sequence. If set to a value above 1, all elements of the sequence have to occur multiple times in a row to be counted. This setting is useful in Eye-Tracking data where glitches of 1-2 fixations are to be excluded and mini can be set to 3.
`rep`	List of two integers indicating the range of the length repeating n-grams to be collapsed before the analysis.
`length`	List of two integers indicating the range of the length of sequence patterns to be found in ADIOS
`eta`	Eta Parameter for ADIOS. Default is 0.9. Lower numbers lead to less identified patterns. The value has to be within the range [0,0.9]
`subst`	Handling subsuming patterns. Integer with value 1,2, or 0. If set to 1, the shorter pattern (the subsumed) is removed. If set to 2, the longer pattern (the subsuming) is removed. If set to 0, all patterns remain in the solution. Some may overlap.
`master`	TKinter object (see introduction to this chapter)

find_tpats(data,tvar,gvar,varlist,p_level=0.05,long_events=1,master='')

T-Pattern Analysis of sequence data with multiple groups. The function takes a data dictionary and a couple of event variables and returns a dictionary with individual t-patterns as keys and information on these patterns in subordinate dictionaries:

- A: First element of the pattern
- Anz: Number of patterns found in data
- B: Second element of the pattern
- Dist: Distance covered by each instance of the pattern.
- End: List of timestamps on which a t-Pattern ended. Each timestamp is a tuple with two elements: Time and Group

- Length: Length of each instance of the pattern
- 'List': List of elements in the t-Pattern in correct order
- Positions: List of positions of each element of the pattern in the data.
- 'Start': List of timestamps on which a t-Pattern started. Each timestamp is a tuple with two elements: Time and Group
- 'Tree': T-Pattern in tree format with nested lists
- Valid: Valid pattern
- d: Difference between shortest and longest pattern
- d0: Shortest distance between elements
- d1: Longest distance between elements
- p: probability of the pattern under the assumption of random distribution.

Options:

`data`	Data dictionary
`tvar`	Name of the variable holding timestamps
`gvar`	Name of the variable holding the group or participant identifiers
`varlist`	List of dummy variables holding the information of occurring events. Each variable stands for one type of event. The occurrences may overlap.
`p_level`	Level of significance for a pattern. It is strongly advised to start with low values if there are many cases in the data. Values below 0.00001 are not bad as a starting point.
`long_events`	Integer with values 1 or 0. If set to 1, repeating events of the same kind are not considered multiple instances but one long event. Set to 1 if events span several timestamps.
`master`	TKinter object (see introduction to this chapter)

References:

Solan, Z., Horn, D., Ruppin, E., & Edelman, S. (2005). Unsupervised learning of natural languages. *Proceedings of the National Academy of Sciences of the United States of America*, *102*(33), 11629–11634. https://doi.org/10.1073/pnas.0409746102

find_sequence(data,svar,tvar='tmp_Time',gvar='tmp_Group',slen=4,somit=1,mode=1,master='')

Crude and quick analysis of short patterns within a sequence. The algorithm finds 2, 3, and 4-grams with or without the omission of one element. Due to these strong restrictions, the algorithm is very fast but does not find anything but n-grams with maximally one omission.

The function returns a standard Nogrod dataset as a list of data and variable list.

Options:

`data`	Data dictionary containing the sequence variable
`svar`	Variable holding the sequence
`tvar`	Optional time variable. If set to 'tmp_Time', the dataset is assumed to be sorted correctly
`gvar`	Optional group variable. If set to 'tmp_Group', only one sequence is assumed
`slen`	Integer or string variable with values between 2 and 4, indicating the length of n-grams to be looked for.
`somit`	Integer with values 1 or 0. If set to 1, an omission of one element is allowed.
`mode`	Integer with values 1 or 2. If set to 1, one long table with two variables is generated. One variable indicates the type of the sequence (length of n-gram), the other holds the sequence. If set to 2, one variable is generated for each type of sequence. The dataset is not elongated.

mpdetection(seq,pat,minlen=5,maxlen=30,master='')

Parallel pattern detection in timeseries data. The function takes an array of sequences and prototypical patterns to identify the occurrence of the patterns in all sequences. This function may be used to scan time series data for prototypical changes, such as peaks, ditches, rising values, or more complex patterns.

The function returns four time series vectors

- A list of optimal correlations for each point in time
- A list of arithmetic means of altitudes of the pattern in each point in time
- A list of geometric means of altitudes of the pattern in each point in time
- A list of the length of the optimal fit in each point in time

Options:

`seq`	Dictionary with keys, numbered from 0 to the number of sequences to be scanned. Each key holds a list of numeric values representing the time series.
`pat`	Dictionary with keys, numbered from 0 to the number of sequences, holding the prototypical patterns to be found in each time series
`minlen`	Integer specifying the minimal length of patterns, measured in number of steps.
`maxlen`	Interger pecifying the maximal length of patterns, measured in number of steps.

create_visone(subj,obj,rel=[],method='all',min_anz=0,obj_is_subj='1',master='')

Create an adjacency matrix and importable information on nodes and links for Visone from two variables. The function takes two variables (subjects and objects) and assumes a relation

whenever they stand at the same position in the list (i.e. in the same case). If a relation variable is provided, the relation is quantified.

The output consists of three tables: An adjacency table in dataset format, a node table in dataset format and a link table in dataset format. All of them may be written to files using the `write_dataset()` function.

Options:

`subj` List of strings denoting the names of the subjects

`obj` List of strings denoting the names of the objects

`rel` Optional list of numeric or string values indicating the type of relation between the subject and object.

`method` Method to be used to compute the relation between subject and object:

- 'anz': Count of co-occurrence of subject and object
- 'dicho': Dichotomous value (1 or 0) indicating any relation between subject and object
- 'entf': inverted count, used as a measure of distance between subject and object
- 'mean': Mean value of the relation vector
- 'sum': Sum of all relation values
- 'typ': Type of relation. Most recent value of the relation variable
- 'all': Count, Dichotomous, Sum, and Mean values. All values are stored in an additional file holding the values of all links.

`min_anz` Minimal number of occurrences to count a node

`obj_is_subj` Integer of value 1 or 0. If set to 0, the identifiers of objects and subjects are not considered equal, even if they have the same value.

`master` TKinter object (see introduction to this chapter)

Chapter 25. Index of functions

25.1. Methods of the class 'Anzeige'

The methods of the class `Anzeige` are used to handle the GUI of Nogrod. All these methods are used to present and store information from the GUI. If Nogrod is opened as a library, these functions are irrelevant.

`abort(self)`: Defines the action to be taken if the abort-button is pressed

`anzeigen(self)`: Opens the text output

`ask(self)`: Defines the questions and buttons to be shown at each page

`ausblenden(self)`: Removes the text output

`back(self)`: Handles the action to be taken if the back-button is pressed

`buttons(self, check=1, abort=0, back=1, pause=0)`: Defines the visibility of buttons

`check_entries(self)`: Checks whether all entries on a page are valid

`cini_schreiben(self)`: Stores the settings and default values

`clean_all_tags(self,sel_tag=[])`: Removes tags from the text output

`clean_up(self,typ='all',pos=1)`: Removes elements from a page

`clean_up_all(self)`: Removes all current elements from the page

`codegetter(self,variabel,item)`: Finds the code corresponding to a label

`confirm_sheet(self,event=0)`: Calls for the selection of an Excel sheet when opening a workbook.

`cut_entry(self,wtup)`: Shortens a text entry

`cutback_list(self,liste,auffang=0,broadseek=0)`: Removes elements of a list that do not correspond to a query

`debug_on(self,event=0)`: Enables the debugging mode

`display_dendro(self, dendrogram, clusters)`: Opens a new window to present a dendrogram. Clusters may be colored.

`display_descriptives_add(self,optlist)`: Opens a new window to present descriptives of a variable

`display_heat_map(self, gridlist=[], w=0, h=0, ph=4, pw=4, legend=1, mode='bw', verbose=1)`: Opens a new window to show a heat map.

`display_line_plot(self,plotdic,width=800,height=600, verbose=1)`: Opens a new window to show a line plot.

`display_text(self, content = "No text to display")`: Opens a new window to display a scrollable text feed.

`export_data(self, dta_pos_all, varlist, filename,debug=0)`: Stores the contents of the storage dictionary to a tabspaced textfile.

`export_output(self,overspill=0)`: Stores the content of the text output to a textfile

`fuellen(self)`: Initiates the GUI

`getselect(self,textfeld)`: Import selected text from the text output to a question text box

`hide_review(self)`: Removes the review of elements within storage

`hilfe_zu(self,htext,event=0)`: Displays the help dialog with informations on a variable

`insecure(self,variabel,event=0)`: Marks a decision as insecure

`intronase(self)`: Clears loops from the history of viewed pages

`level_down(self,variable,ebene)`: Change the level of the current position within storage.

`level_up(self)`: Change the level of the current position within storage.

`list_add(self,spillover=0)`: Add element to the list of selected items in question_ladd()

`list_rem(self,spillover=0)`: Remove element from the list of selected items in question_ladd()

`load_cset(self,csettings)`: Load settings

`load_dset(self, fname, header, sep, dname='Data', vname='D_Var', designation='Main_Table')`: Load a dataset from a textfile.

`locate(self,l1,l2,l3)`: Describe the current location within storage

`mark(self,tag_id,ls=[])`: Tag elements in the text output

`message(self,m_id,m_type=1,var='Err_Msg',add='')`: Display a message to the user

`namegetter(self,variabel,item)`: Get the label of a selected element if only the code is known

`opendialog(self,mode='load',question_pos=1,defaultextension='.txt')`: Show a dialog box

`pause(self)`: Action to be taken when the 'Break' button is pressed

`question_bt(self, cb_var, question_pos)`: Question Type: Buttons

`question_cb(self, cb_var, question_pos,layout="hor",defval=0)`: Question Type: Check Boxes

`question_dd(self, cb_var, question_pos,width=40)`: Question Type: Drop Down Menu

`question_file(self, cb_var, question_pos, mode='load',defext='.txt')`: Question Type: Filename

`question_ladd(self,cb_var,liste,multi=1)`: Question Type: List from which several items may be selected

`question_ls(self,cb_var,liste,multi=1)`: Question Type: List to tag items

`question_lseek(self,cb_var,liste,multi=0)`: Question Type: Searchable list to tag items

`question_mark_units(self,cb_var,Einheit)`: Question Type: Mark elements in the text output

`question_menu(self, cb_var, question_pos)`: Question Type: Nested menu

`question_rating(self, cb_var, question_pos, scalelist=['disagree','','','','agree'], valuelist=['1', '2', '3', '4', '5'], defval='1')`: Question Type: Rating scale

`question_rb(self, cb_var, question_pos, layout='vert', defval='98')`: Question Type: Radio buttons.

`question_rbopen(self, cb_var, question_pos, layout='vert', defval='98')`: Question type: Rado buttons with one additional open textbox option

`question_sd(self, cb_var, question_pos,points=5,defval=0)`: Question type: Semantic differential scale

`question_sel_units(self,var,Einheit)`: Question type: Select elements from storage

`question_txt(self, cb_var, question_pos, width=40)`: Question type: Textbox (1 line)

`question_txt2(self, cb_var, question_pos, width=40, height=3, getselect=0)`: Question type: Textbox (several lines)

`rb_tamper(self)`: Command to be called whenever a radio button is changed

`remove_item(self,level,height)`: Remove item from review list

`reset_coding(self)`: Reset the storage

`select_sheet(self,sheetlist)`: Select sheet within an excel workbook

`set_window(self)`: Define the place of each component of the GUI

`show_descriptives(self)`: Open a new window to display variable descriptives

`show_parameters(self)`: Open a new window to display the current parameters of the program.

`show_path(self,event=0)`: Open a new window to display the page history

`show_review(self,level,rm=1,height=3)`: Display a review of elements in storage

`show_settings(self,event=0)`: Open a new window to display the settings dictionary

`show_storage(self,event=0)`: Open a new window to display the storage dictionary

`show_variables(self)`: Open a new window do display the variables in all open datasets

`show_verb(self,event=0)`: Open a new window to display the complete log

`store_var(self,variabel,pos=-1,setdef=1,store=1)`: Store the values entered on a question

`store_var_all(self,setdef=1)`: Store the values of all questions

`submit(self,overspill=0)`: Defines the actions to be taken upon submitting the parameter values on each page.

`test_styleset(self)`: Defines the style to be used in the text display.

`unit_confirm(self,Einheit,sel_tags=[])`: Confirms highlights from question_mark()

`unit_select(self,Einheit)`: Changes the level according to the element selected in question_sel_units()

`units_action(self,Einheit,action='none',sel_tags=[])`: Actions to be performed on units in question_sel_units()

`var_export(self, exp_file, dictionary, variabel,debug=0)`: Add the contents of a variable to a file export.

25.2. Callable functions

These functions are callable when Nogrod is included to scripts as a library. Some of them have the argument `master`, which specifies the GUI of class `Anzeige` linked to the analysis. If no GUI is opened, the argument may be omitted to pass all outputs to the standard I/O stream. If master is set to 'silent', all outputs are suppressed for this function.

`add_populism(data, master='')`: Function to recode content analysis data from NCCRIII

`add_varlist(vname, labels, codes=[], excludes=[], retain=0)`: Adds a list of new values to an existing variable in the codebook.

`agg_entropy(data,group,mvar,mode='1',master='')`: Aggregates groups and compute the entropy of each group.

`aggregate(dset,key,var,method,weight=0,master='')`: Aggregates groups of cases

`analyze_entropy(data, group, varlist, opt={'comp':0,'boot':0}, report = '', master='')`: Analysis of entropy. See section 14.1.

`artikelholen(ID)`: Loads the contents of a document.

`available(key)`: Returns true if the key exists in settings

`baum_schreiben(tdic, einr = 0, inherit = '', spc = '  ', trunc=0, lists=False)`: Represents a JSON as a nice, trunctuated string with line breaks and indented sub-levels.

`bereinigen(uml_string, lc=0,lb=0,uml=0,encod='latin-1')`: Transforms a string with unicode letters to a simpler encoding.

`binomial_odds(draws, p, critical)`: Computes the binomial probability.

`bootstrap_sample(liste,l=0)`: Draws a bootstrapping sample from a population

`calc_adios(data, symb, gvar='', tvar='', mini=0, rep=[1,3], length=[2,7], eta=.9, subst=1, master='')`: Grammar induction, see section 12.3

`calc_chisquare_dummy(l1,l2,directed=0,cramersv=0)`: Computes Chi-Square from two lists of values.

`calc_correlation(l1,l2)`: Computes the pearson correlation between two numerical lists

`calc_entropy(liste,maxliste=[])`: Computes the entropy of a list of elements.

`calc_ic_reliability(dset, uvar, cvar, varlist, unitlist=[], coderlist=[], core_cod='no_core', methods=['PA'], options=[], master='')`: Compute interrater reliability. See section 16.1.

`calc_regression(x,y)`: Compute a linear regression.

`calc_timecoef(x,y,dweight,currdate=0,master='')`: Support function for matching survey and content data.

`calculate(liste,method)`: Quick calculation function to compute means, sums and other statistics from a list of values which may contain strings and missing values.

`calculate_vectors(vectors,method,master=0)`: Computation for whole variables.

`cd_proof(var,master='')`: Tests whether a variable is a dummy

`check_data(data,varlist,master='')`: Tests whether data is correctly read from a file.

`check_dummytable(dataset)`: Tests whether variables are dummies.

`co_occurrence(data, dmode='dummy', method='anz', margin=1, prefix='', cases='', keyvar='', min_case=0, min_anz=0, dummylist=[], master='')`: Function to perform a co-occurrence analysis.

`collapse_repetitions(sequences,minlen,maxlen)`: Finds recurring ngrams and collapse them to placeholders.

`context(line, expression, span=10, case=0)`: Finds the context of a regular expression in a string.

`create_cluster_table(data, varlist, std, mg,master='')`: Initializes data for a cluster analysis

`create_coding_dic(data,uvar,cvar,varlist,unitlist=[],master='')`: Initializes data for a reliability test with calc_ic_reliability()

`create_corpus(dset,path,idvar,nvar='Fulltext',encod='latin-1',master='')`: Creates a corpus from a folder of textfiles.

`create_keydic(data,keyvars,varlist=[],master='')`: Transforms a table into a dictionary.

`create_ngrams(tokens,nlen=2,universe=[])`: Constructs ngrams from a list of tokens.

`create_visone(subj,obj,rel=[],method='all',min_anz=0,obj_is_subj=1,master='')`: Creates an adjacency matrix from two or three vectors.

`create_window(data,gvar,nvar,units=7,pos='1',master='')`: Creates a gliding window in time series data.

`critical_abort()`: Closes Nogrod GUI

`crosstab(l1,l2,mode='dicho')`: Creates a cross table from two variables

`curr()`: Returns the current subtree in storage.

`data_dim(dset)`: Returns the dimension of a dataset or table.

`define_styleset(styleset="Default")`: Function with all styleset definitions to be used in the GUI

`delete_missing(data,varlist,n_allowed=0)`: Removes missing values from a variable

`desparse(data,minc,mina,remvar=['#Group'],master='')`: Removes rows and colums with few non-zero values from a table.

`detect_gaps(data,tvar,nvar,length,gvar='',sorting=1,master='')`: Finds gaps in a time series.

`dim(dset)`: Redundant function for data_dim()

`display_dset(dset, sep='blank')`: Transforms a dataset to a string with separators to be written in the text output.

`display_table(table, sep="blank", cols_pre=[], rows_pre=[])`: Transforms a table to a string with separators to be written in the text output.

`distance(list1,list2,uni=0)`: Computes the Euclidian distance between two vectors.

`distmatrix(data,liste)`: Computes an Euclidian distance matrix from a dataset

`dset_dim(dset)`: Redundant function: data_dim()

`dummy(caslist,keylist,mode='dicho',min_case=0,min_anz=0,master='')`: Creates a dummy table from two variables.

`dummy_reshape(data,varlist,rvars,rtype='count',master='')`: Transforms a dataset from broad to lang format and vice versa.

`dummytab(data, cases, keyvar, mode='dicho', min_case=0, min_anz=0,master='')`: Creates a dummy table from a dataset.

`duplicate_shingling(tdata,idvar,tvar,ngl=5,master='')`: Identifies duplicate documents in a corpus and returns a report and a table of potential duplicates.

`event_agree_overall(ad)`: Computes the mean agreement and its standard deviation from event_agreement()

`event_agreement(l1,l2,lag,meas)`: Computes the agreement between two time series.

find_cluster(dset, varlist, outfile, add_outputs={'ssa':0, 'dendro':0, 'hist':0, 'dist':0, 'vector':0}, row_std=0, table_std=0, master=''): HECANE Cluster analysis. See section 13.1.

`find_peaks(svar,tvar=0,pdir='1',pthres=95,master='')`: Identifies peaks in a continuous longitudinal measurement.

`find_sequence(data, svar, tvar='tmp_Time', gvar='tmp_Group', slen=4, somit=1, mode=1, master='')`: Identifies recurrent patterns in a sequence of symbols. See section 12.1.

`find_tpats(data,tvar,gvar,varlist,p_level=0.05,long_events=1,master='')`: T-Pattern analysis. See section 12.2.

`flatten(nestlist)`: Transforms a nested list to a long list without sub-lists. Used in find_cluster()

`flatten_curve(tvar,svar,window=10,master='')`: Flattens a curve using moving average

`focus_timeseries(data,issvar,actvar,windir,winlen,master='')`: Function to identify instances of increased focus and attention. See section 14.2.

`generate_tdm(textlist, idlist=[], lang='none', ngrams=2, sparsity=[.01,.99], universe=[], weight="dicho", master='')`: Creates a Term-Document-Matrix from a list of documents.

`get_cluster(data,frames,mode='loading')`: Loads a cluster output

`get_codebook(filename)`: Loads the codebook with labels and questions for all menus in the GUI

`get_data(filename, header=1, sep='\t', varlist = [],verbose=2,master='')`: Loads a data table from a file.

`get_dataset(fname, header=1, sep='\t',master='')`: Loads a dataset from a file.

`get_directory(path='',ext=[])`: Returns the contents of a given folder.

`get_dta_level(liste,master='')`: Determines the most probable variable level of a list of values.

`get_node(prog,result)`: Supports the dendrogram function.

`get_pos(liste,element)`: Supports the dendrogram function.

`get_sep(s)`: Identifies the correct separator for data input.

`get_todo(fname)`: Loads a list of todo items from a file.

`get_unique(liste)`: Returns a sorted list of unique elements from a list.

`get_univ(flist, lang, ngram, sparsity=[.01,.99], encod='latin-1',master='')`: Returns the universe of n-grams from a list of textfiles.

`get_varnames(filename,header=1,sep='\t')`: Loads the variable names from a data table.

`get_words(fname, lang='en', laenge=2,encod='latin-1')`: Extracts words from a textfile.

`get_xlsx(filename, header=1, sheet=0, master='')`: Loads an xlsx workbook.

`getcaps(zeile)`: Identifies the capital letters in a string.

`gravity(data, cluster)`: Finds the cluster center in a cluster (HECANE).

`group_variable(liste,mode='equal',param=2,master='')`: Segments a list of values to groups.

`hash_texts(tid,fulltext,mode=1,ngl=5,master='')`: Computes a hash for a text for n-gram shingling.

`heat_color(sval,mode='bw')`: Transforms a metric value to a color code for display.

`inspect_variable(values)`: Returns the descriptives and type of a list of values.

`isPair(T)`: Support function for dendrogram.

`kmeans(dset,varlist,direction=1,num=4,iterations = 100, precenter = {},master='')`: Performs a k-means cluster analysis. See section 13.2.

`lcopy(liste)`: Returns a copy of a list. Sometimes necessary to bypass list equivalence of complex object in Python.

`lemmatize(zeile,lang='none')`: Lemmatizes a string of text using a Snowball stemmer.

`log(name,pos=1)`: Adds an entry to the internal log.

`log_settings(out_vars,outfile,append=1)`: Defines the internal log.

`make_fname(fname,ext='txt',suffix='')`: Transforms an entry to a valid filename and adds an extension if none was specified.

`match_nccr(sdata,mdata,wmode,dmode,cmode,suffix='',master='')`: Adds media data to survey data collected in NCCRIII

`maxdist(data,nestlist)`: Computes the maximal distance within a cluster in HECANE.

`maxHeight(T)`: Supports the dendrogram function.

`merge_data(data,keyvars,schldic,vadd=[],vretain=[],master='')`: Adds new variables to an existing table.

`merge_elong(d1,d2,v1,v2,varlist,master='')`: Concatenates two tables.

`merge_files(filelist,varlist=[],master='')`: Concatenates a directory full of tables.

`merge_ggcrisi(data1,data2,key1,key2,master='')`: Project specific function to merge tables in GGCRISI.

`mpdetection(seq,pat,minlen=5,maxlen=30,master='')`: Detects patterns in a sequence.

`multi_kmeans(data, dvar, varlist, direction, numrange, iterations, icenters, gv,master='')`: Performs a multigroup cluster analysis. See section 13.3

`naive_tokenizer(line, num=0)`: Tokenizes a string.

`nghash(ngram,ns=5)`: Computes the hash of a text using n-grams

`normalize_ts(dset, tvar, gvar, length, addvars=[], method='retain', relts='0', master='')`: Normalizes the timestamps in time series data with unequal distances between measurements. See section 11.5.

`printDendrogram(T, sep=2)`: Returns a string containing the dendrogram for output.

`recentpeak(date,values,threshold)`: Finds the most recent peak in a time series.

`reformat_mediause(data,master='')`: Supports the match_nccr() function.

`reformat_mediause_css(data,master='')`: Supports the match_nccr() function.

`reltest(codings, tvars, units, coders, kerncod='no_core', methods=['PA'], options=[], master='')`: Computes interrater agreement. See section 16.1.

`resource_path(relative_path)`: Supports functions using working directories.

`shinglehash(line,tid='none',prev={},ng=5)`: Computes a series of hashes for shingled n-grams in a text.

`sort_dataset(dset,variables,descend=False,master='')`: Returns a dataset sorted by one of several variables.

`sort_table(table,variables,mode=False)`: Returns a table sorted by one of several variables.

`standardize_rows(data)`: Returns a table in which the rows are z-standardized

`standardize_table(data)`: Returns a table in which all values are z-standardized

`stat_desc(values,weight=0,verbose=1)`: Computes descriptive statistics of a list of values.

`stat_frequencies(values)`: Returns a frequency table from a list of values.

`stat_type(values)`: Returns the type of a variable.

`stem(token,lang='eng',verbose=0)`: Snowball stemmer

`svm_prf(scores,true)`: Computes precision, recall and F from two binary variables.

`svm_prf_curve(tdm, rvec, classvec)`: Computes precision, revall, and F for different positions of the SVM hyperplane.

`svm_scores(tdm_in,rvec,features=[],cases=[],verbose=0)`: Computes the classification scores in an SVM classificaion.

`synch_events(data,varlist,frame,meas,master='')`: Synchronizes event data. See section 11.7

`synch_move(ad,v,l)`: Supports synch_events()

`synch_wiggle(ad,master='')`: Supports synch_events()

`tabulate(prog,variablen)`: Returns a table from a dictionary of data lines.

`tempout(d, fname = 'dict_temp.txt')`: Storage of a temporary dataset.

`textmine(linelist)`: Scans a text for metatext used in NCCRIII

`train_svm(tdm, classvec,master='')`: Trains an SVM classifier.

`transform_float(data,variablen=[])`: Transforms a series of variables in a dataset to float.

`traverse(T, h, isFirst)`: Supports the dendrogram function.

`tts(invalue,inf,outf,numf='dec',verbose=0,master='')`: Transforms timestamps from one format to another. See section 8.1

`univec(vector)`: Normalizes a numeric vector to length 1.

`verb(name,stage=2,nl=1)`: Writes a comment to the internal log.

`verbout(zeile,style='text',master='')`: Writes a line to the output. If set to master='silent', the output is suppressed. If master is set to a GUI, it is printed in the text output. Else the output is passed to the standard stream via print().

`write_data(data,init_varlist,filename,header=1,sep='\t',verbose=2)`: Writes a table to a textfile.

`write_dataset(data,filename,header=1,sep='\t')`: Writes a dataset to a textfile.

`write_dendrogram_R(prog,order,fname,plottitel='Clusteranalyse')`: Writes a dendrogram to a textfile.

`write_ssa(data,filename,rownames='',bildname='')`: Writes an R script to perform a Smallest Space Analysis using smacof.

`write_ssa_cluster(distmat, ssa_elemente, out_ssa, titel_bez='Proximity of Elements')`: Writes an R script to perform a Smallest Space Analysis using smacof.

`write_xlsx(data,varlist,filename,header=1)`: Writes a table to an xlsx workbook.

Chapter 26. References

Nicholls, T. (2019). Detecting Textual Reuse in News Stories, At Scale. *International Journal of Communication*, *13*, 4173–4197.