AutoNeta – A Shiny Implementation of a Semi-Automatic Data Transformation Manager

Tzviel Frostig, Barak Brill

15/08/2017

Contents

1. Introduction

When encountering real data, usually there is a need to pre-process the data before proceeding to analyze it, as it contains missing values, correcting typos and so on. While doing so, it can be beneficial to transform the data in order to make it symmetrical. Data analysis can be greatly simplified when the distribution of a variable (feature) is symmetric across subjects, variability is stable across different conditions, relationships are linear between variables of interest and models are additive rather than having a more complex structure[1]. The data analysis can performed if the assumptions do not hold, however it makes the models more complicated, less interpretable and it requires a certain level of expertise in statistical analysis.

As more data is collected, especially in the medical field, more non-experts are performing data analysis and are not always aware to the required assumptions. A tool which transforms the data in such a way that the assumptions are more likely to hold can be an asset.

The document is structured in the following way: Section 2 details the main idea behind a semi-automated data transformation manager, how transformations are performed and selected and issues regarding the algorithmic design and visualization that were resolved during development. Section 3 contains an overview of the software, describing the user interface and workflow. Subsection 4 describes the system state machine and MVC logic. Section 5 explains about the online availability and long term support of the software.

[1] Taken from 'The importance of non linear transformations use in medical data analysis' (2015)

2. Methodology – (Semi-Automatic Transformations)

When analyzing data, a statistician often needs to transform variables, mostly using monotone transformations (maintaining order between measurements). Selecting the proper transformation is an art as much as a science: requiring one to balance between model interpretability and model validity.

To automate the process, the semi-automatic transformation methodology is offered. Possible transformations for a variable are displayed to the statistician. The statistician is left with choosing the transformations, while reviewing a set of transformations concurrently. The mundane work of performing the transformations themselves is automated away – requiring minimal operator intervention.

Selection of transformations is based on displayed histograms, kernel density estimator (when appropriate) and the yule index for skewness.

2.1 Index

Currently only one index can be applied to the variables, the Yule index:

$$sk = \frac{0.5(m_3 + m_1) - m_2}{0.5 * (m_3 - m_1)},$$

where m_1, m_2 and m_3 are the lower quartile (i.e.: the 25th percent quantile), the median and the upper quartile (i.e.: the 75th percent quantile), respectively. For each variable, its own index is computed, before and after the transformations. The program can be extended to different types of indices.

2.2 Transformation Logic

Below is a list of the possible transformations for each variable type. Variable type definitions can be found in the appendix.

Type	**Seen Variable**	**Comments**	**Transformations**
Amount	$\boldsymbol{x \geq a}$	$'a' - lower\ bound$	$\log(x), \sqrt{x}, \frac{1}{x}, \frac{1}{\sqrt{x}}, x^2$
Count	$\boldsymbol{x \in [0, 1, 2, \dots,]}$		$\sqrt{x}, \log\left(x + \frac{1}{6}\right), \frac{1}{sqrt\left(x + \frac{1}{6}\right)}$
Ratio	$\boldsymbol{x = \frac{y}{z}}$		$, x^p - z^p, \log(x)$
Fraction	$\boldsymbol{x = \frac{y}{z}}$		$\log\left(\frac{x}{1-x}\right)$
Counted Fraction	N out of M $\boldsymbol{x = \frac{n}{m}}$	$'a' - Null$ $'b' - m$	$\log\frac{\left(x * b + \frac{1}{3}\right)}{\left(m - x + \left(\frac{1}{3}\right)\right)}$
Bounded Amounts	$\boldsymbol{x \geq a}$	$a \leq x \leq b$ $x^* = \frac{x-a}{b-a}$	$\log\left(\frac{x^*}{1-x^*}\right)$

Ranks/Bounded Counts (Required clarification)	$\boldsymbol{x} \in [\mathbf{0}, \mathbf{1}, \mathbf{2}, \ldots,]$	$'a' - n$ $'b' - m$	$\frac{x - a + \frac{1}{3}}{b - a + \frac{1}{3}}$
Difference	$\boldsymbol{x} \in \boldsymbol{R}$		None
Ordered Categories	$\boldsymbol{x} \in [\mathbf{0}, \mathbf{1}, \mathbf{2}, \ldots,]$		Cumulative entropy
To Reverse	$\boldsymbol{x} \in \boldsymbol{R}$	Decided by the user	$b - x$

Table 1 - Transformation table, which transformation fits to each type of variable.

The transformation logic was built in such a way that it is easy to update either types of variables, or transformations. All the transformations can be found in a single list, and for each type of data, the list of possible transformations is generated programmatically. A wrapper functions fits between the types of data in the variable definition file, to the function and call the transformations.

2.3 Data File and Variable Definitions file

The semi-automatic transformation method requires two files to be uploaded into the application. 1. The data file, which includes non – transformed variables. 2. The variable definition table.

The variable definition table includes the following columns: Variable name, a, b, Type, To Reverse. The columns guide the app on which transformation to suggest (see Table 1). When the file is loaded a battery of checks is performed to ensure its validity (see Figure 1).

2.4 Automatic generation of variable definitions file

Manually creating the variable definition file can be time consuming, we offer a shortcut to expedite the process. The user can upload only his data file, and the app will try to make a relevant guess regarding the type of the variable. The logic is as the following:

Condition	**Guess**
Not a number	'To Be Determined'
All numbers are equal	Constant
Number of unique value less equal than threshold (default 2)	Categories
Variable equals only complete numbers larger than 0	Count
Variable less than 1 larger than 0	Fraction
Variable is non-of the above, but is numeric	Amounts
Non-of the above	'To Be Determined'

Table 2 – Guessing variables based on conditions.

The placing in the table also dictates the importance of the condition, for example if the numbers are all equal but also complete numbers larger than 0, the guess would still be constant.

The parameters a and b are determined by the minimum and maximum of the variable respectively.

The user is strongly urged to verify and complete the variable definition by himself, as it not possible to guess all types of variables. Nevertheless, the above rules of thumb are a decent initial choice for the user to review.

2.5 Visualization

We present for each variable and transformation a histogram, and for continuous variables the density curve. The default values for the bin width are based on the Freedman–Diaconis rule[2]:

$$h_b^* = 2 * \frac{IQR(X)}{\sqrt[3]{n}}$$

We allow for the manipulation of the bin width for all transformations altogether, using a single slider (in order to avoid clutter). For each histogram, the bin width is updated based on $h_b^* * (Slider\ Value)$, which gives a reasonable result.

The kernel density estimation bandwidth rule is based on Silverman rule of thumb[3]:

$$h_d = 0.9 * \frac{m}{n^{\frac{1}{5}}}, \quad m = \min\left(\sqrt{Var(x)}, \frac{IQR(X)}{1.349}\right),$$

The manipulation is using a slider in a similar manner as in the bin width parameter.

[2] Freedman, David; Diaconis, Persi (December 1981). "On the histogram as a density estimator: L_2 theory" , *Probability Theory and Related Fields*. Heidelberg: Springer Berlin. **57** (4): 453–476. ISSN 0178-8051
3 Silverman, B. W. 1992. Density Estimation for Statistics and Data Analysis. London: Chapman & Hall. ISBN 9780412246203

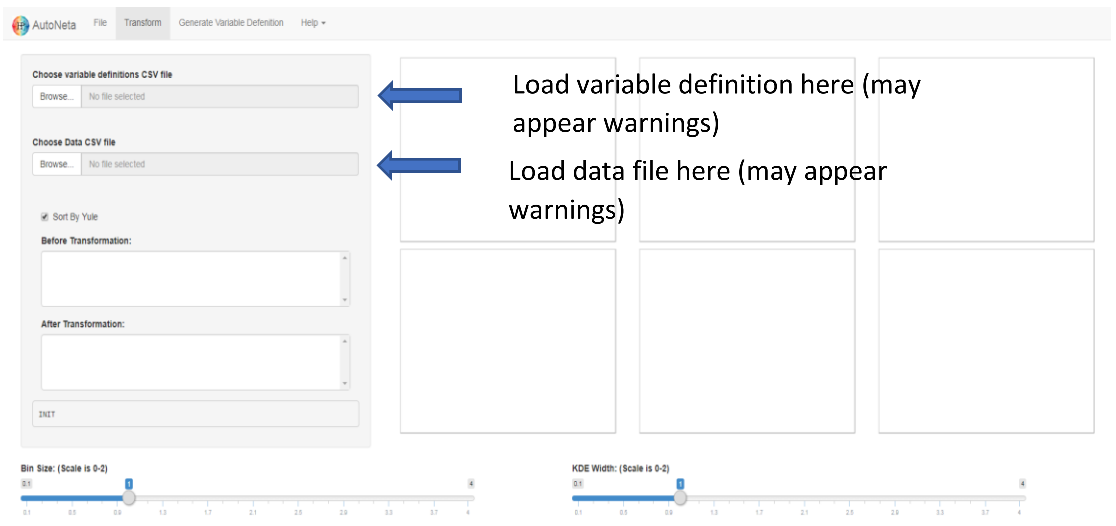
3. User Interface

Section 3.1 will demonstrate the use of the application using the ADNI data. Section 3.2 Describes software architecture.

3.1 Transformation Manger

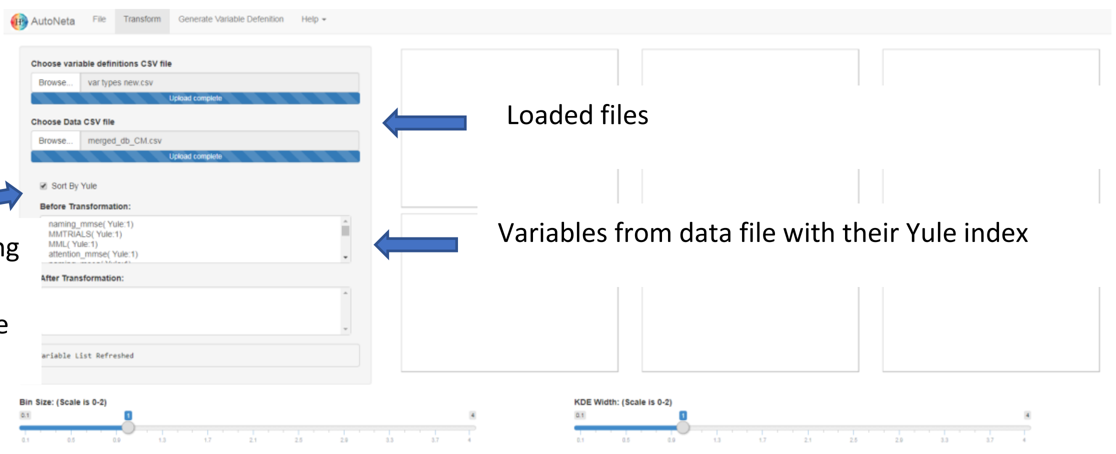
Loading variable definition and data file:

The application before loading files:



Load variable definition here (may appear warnings)

Load data file here (may appear warnings)



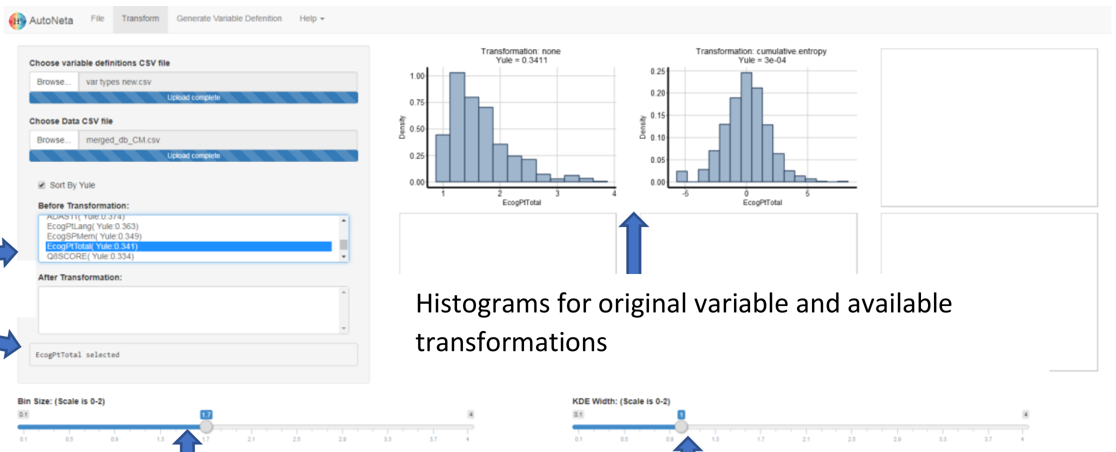
naming_mmse(Yule:1)
MMTRIALS(Yule:1)
MML(Yule:1)
attention_mmse(Yule:1)

ariable List Refreshed

Loaded files

Variables from data file with their Yule index

Sort by descending Yule index, or by sorting in data file



Choose variable definitions CSV file: var types new.csv

Choose Data CSV file: merged_db_CM.csv

☑ Sort By Yule

Before Transformation:

ADAS11(Yule:0.374)
EcogPtLang(Yule:0.363)
EcogSPMem(Yule:0.349)
EcogPtTotal(Yule:0.341)
QSSCORE(Yule:0.334)

After Transformation:

```
EcogPtTotal selected
```

Bin Size: (Scale is 0-2) 1.7

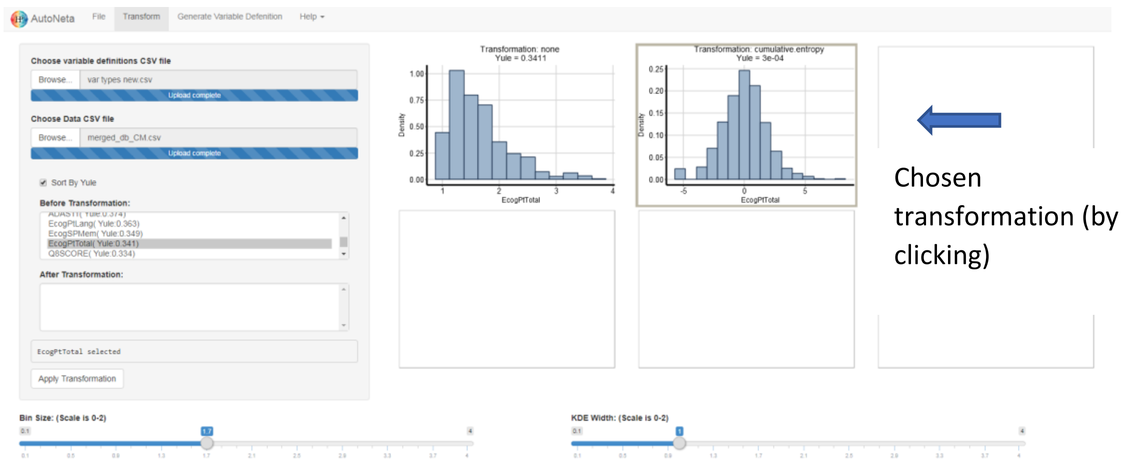
KDE Width: (Scale is 0-2)

Chosen variable

Histograms for original variable and available transformations

Control bin size for histogram

Control bandwidth for density estimation



Choose variable definitions CSV file: var types new.csv

Choose Data CSV file: merged_db_CM.csv

☑ Sort By Yule

Before Transformation:

ADAS11(Yule:0.374)
EcogPtLang(Yule:0.363)
EcogSPMem(Yule:0.349)
EcogPtTotal(Yule:0.341)
QSSCORE(Yule:0.334)

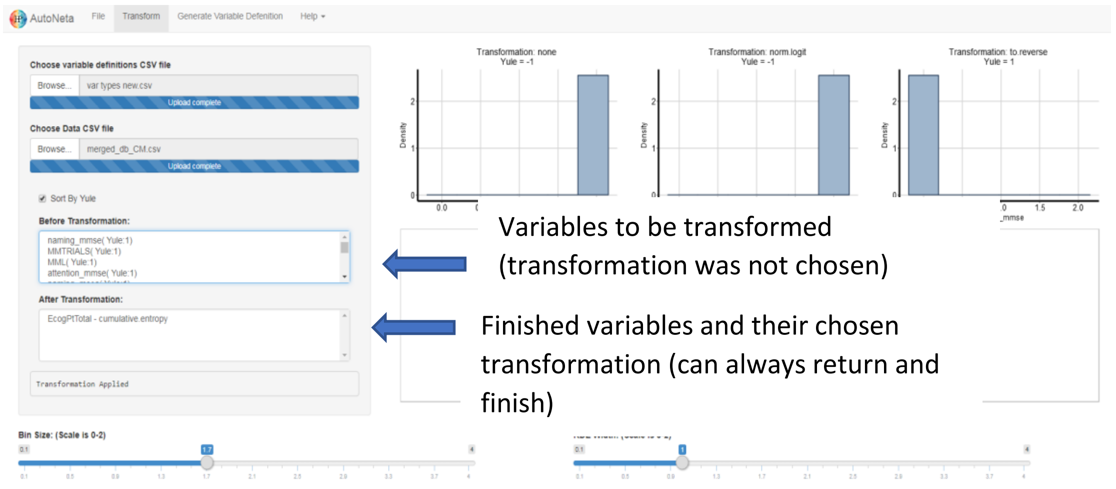
After Transformation:

```
EcogPtTotal selected
```

Bin Size: (Scale is 0-2) 1.7

KDE Width: (Scale is 0-2)

Chosen transformation (by clicking)



Choose variable definitions CSV file: var types new.csv

Choose Data CSV file: merged_db_CM.csv

☑ Sort By Yule

Before Transformation:

naming_mmse(Yule:1)
MMTRIALS(Yule:1)
MML(Yule:1)
attention_mmse(Yule:1)

After Transformation:

EcogPtTotal - cumulative.entropy

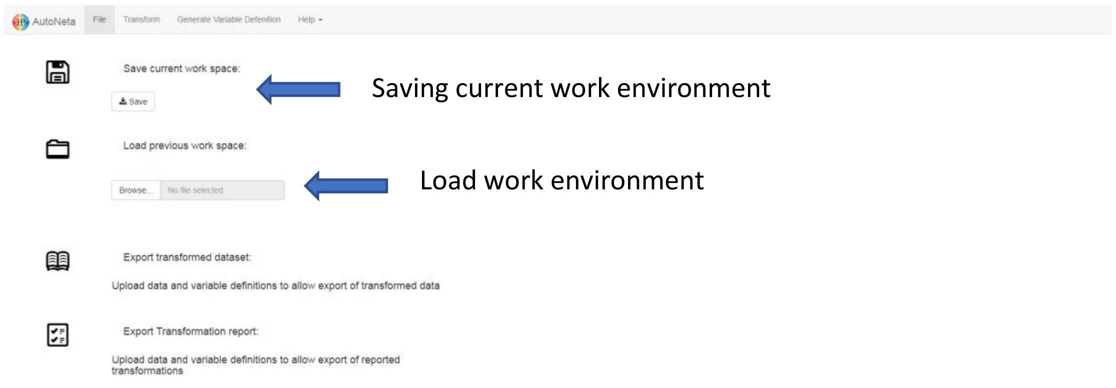
```
Transformation Applied
```

Bin Size: (Scale is 0-2) 1.7

Variables to be transformed (transformation was not chosen)

Finished variables and their chosen transformation (can always return and finish)

3.2 File



Save current work space:

Saving current work environment

Load previous work space:

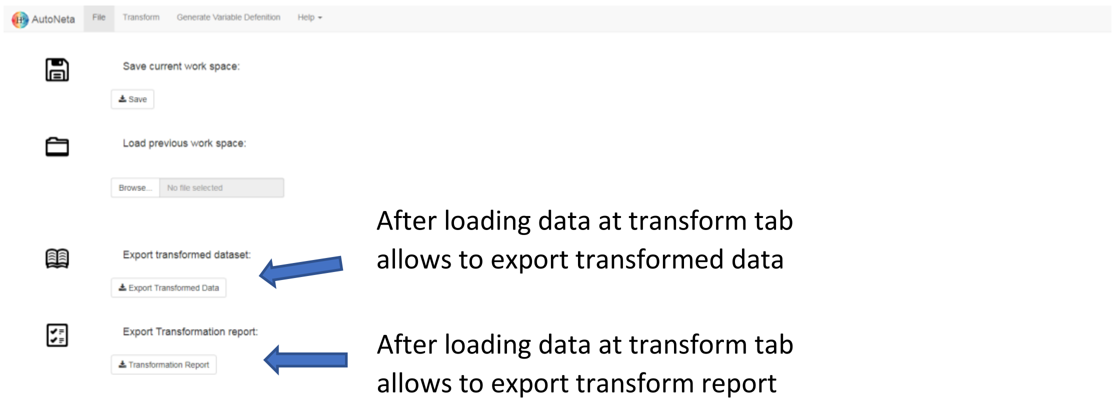
Load work environment

Export transformed dataset:

Upload data and variable definitions to allow export of transformed data

Export Transformation report:

Upload data and variable definitions to allow export of reported transformations



Save current work space:

Load previous work space:

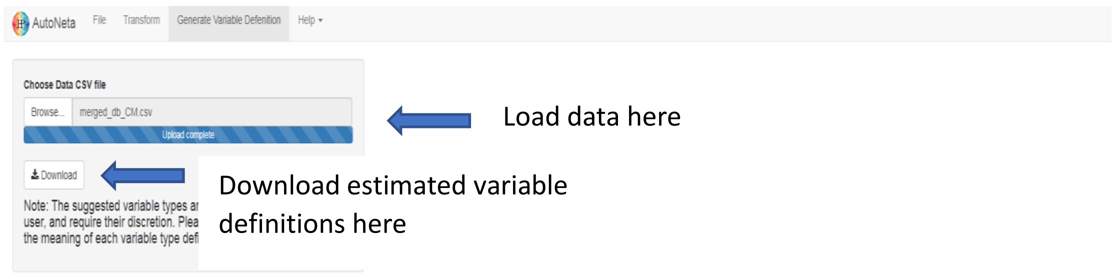
Export transformed dataset:

After loading data at transform tab allows to export transformed data

Export Transformation report:

After loading data at transform tab allows to export transform report

3.3 Variable Definition Generation



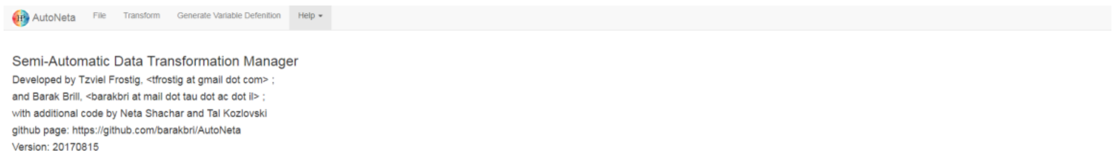
Choose Data CSV file

Load data here

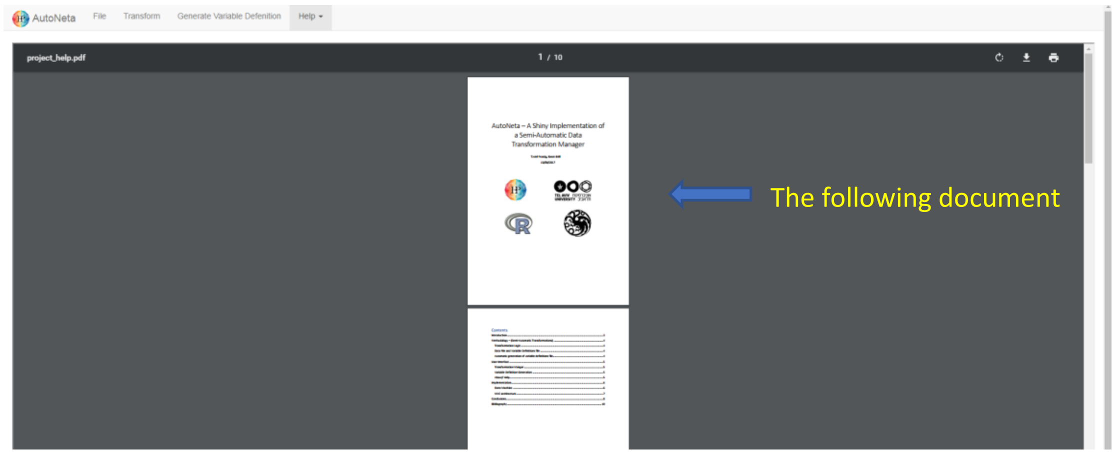
Download estimated variable definitions here

Note: The suggested variable types ar
user, and require their discretion. Plea
the meaning of each variable type defi

3.4 About/ Help



Semi-Automatic Data Transformation Manager
Developed by Tzviel Frostig, <tfrostig at gmail dot com> ;
and Barak Brill, <barakbri at mail dot tau dot ac dot il> ;
with additional code by Neta Shachar and Tal Kozlovski
github page: https://github.com/barakbri/AutoNeta
Version: 20170815


AutoNeta
File
Transform
Generate Variable Defenition
Help
project_help.pdf
1 / 10
AutoNeta – A Shiny Implementation of a Semi-Automatic Data Transformation Manager
The following document

4 Software Implementation

This section describes the structure of the application, in terms of state machine and architecture.

4.1 State Machine

Figure 1 describes the state machine. Additional details about **selected** states and actions are found below:

Transformation Manager Displayed – The transformation manager is displayed. The user may load datasets, variable definition files or a workspace from a previous session.

Check Var. Def. Validity – Variable definition is loaded from file. Values of a and b are checked to be legal and valid. Variable types are checked against a closed list of possible variable types.

MSG: Description of Var. Def. misfit – An informative message is displayed to the user, about which checks were not OK in the previous step.

Load dataset, init variables – The model (see next section) is fully initialized. Variables which are not purely numeric are excluded from transformation. Numeric variables must have at least 3 valid values, or they are excluded as well. Yule indices are computed for each variable, along with place holders for selected transformations are transformed data.

MSG: Description of Dataset misfit - An informative message is displayed to the user, about which checks were not OK in the previous step.

Check save file validity – A file containing dataset, variable definitions and transformed variables from a previous session is loaded. All graphical parameters (current lists and graphs) are resets. The save file contains the version of the Shiny App used to generate it. If versions are not identical (previously used session and current session) a warning message is displayed to the user.

Variable List is updated and displayed – the list of variables is divided into variables which have been transformed or not yet been transformed. For the 'before' list, variables are ordered by their original dataset order, or by descending value of absolute yule index (by checkbox). The 'after' list contains the list of transformed variables, along with the name of chosen transformation.

Check if variable is valid for transformations – The selection of a variable which is invalid for transformations (e.g., not numeric, does not have a valid row in Var. Def. file) triggers an informative message to the user. If the variable is valid for transformations, all possible transformations (according to Section 2) are displayed).

Transformation selected and applied – when a transformation is selected and applied – the variable is moved to the 'after' list, and both variable lists are updated. The variable presented is the new variable to top the 'before' list.

Save Transformed data to file – transformed dataset is saved to file.

Save transformation report to file – a report of the transofrmations performed for each variable (along with old and new Yule indices) is saved to file.

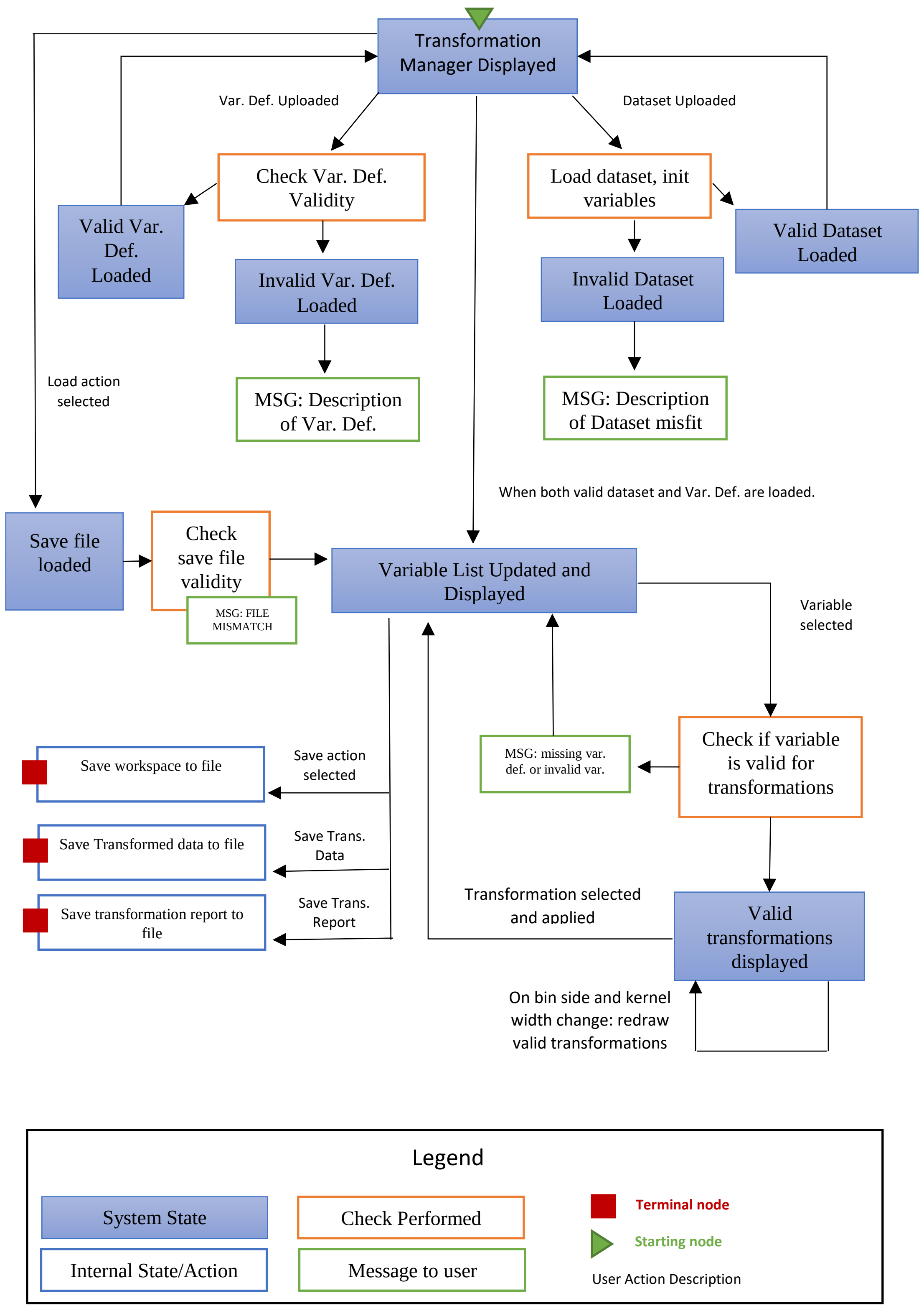
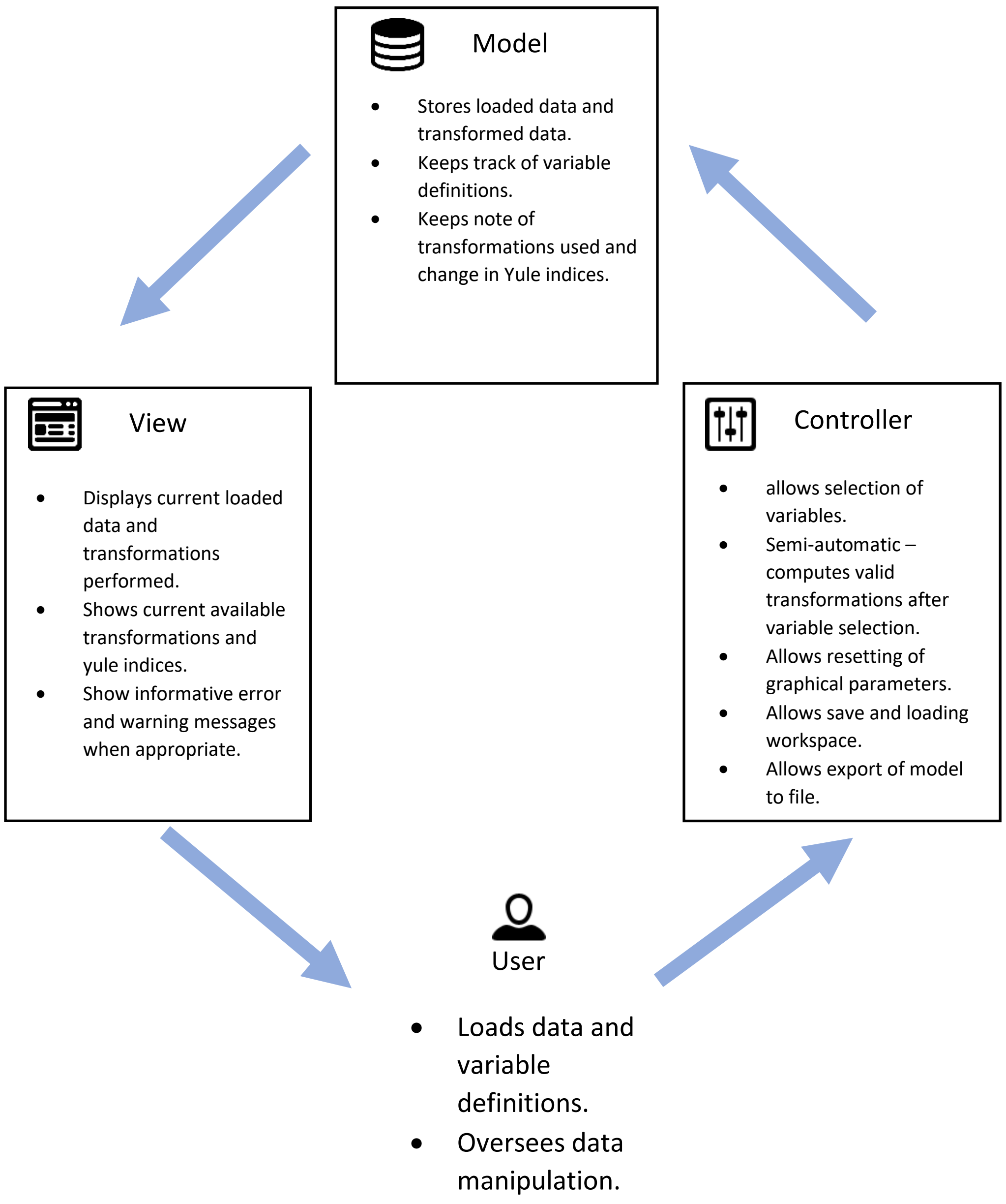

Transformation Manager Displayed
Var. Def. Uploaded
Dataset Uploaded
Check Var. Def. Validity
Load dataset, init variables
Valid Var. Def. Loaded
Invalid Var. Def. Loaded
Invalid Dataset Loaded
Valid Dataset Loaded
MSG: Description of Var. Def.
MSG: Description of Dataset misfit
Load action selected
When both valid dataset and Var. Def. are loaded.
Save file loaded
Check save file validity
MSG: FILE MISMATCH
Variable List Updated and Displayed
Variable selected
Save workspace to file
Save action selected
Save Transformed data to file
Save Trans. Data
Save transformation report to file
Save Trans. Report
MSG: missing var. def. or invalid var.
Check if variable is valid for transformations
Transformation selected and applied
Valid transformations displayed
On bin side and kernel width change: redraw valid transformations
Legend
System State
Check Performed
Terminal node
Starting node
Internal State/Action
Message to user
User Action Description

Figure 1 - State machine

4.2 MVC architecture

The Model-View-Controller design architecture[4] is a contract between a data representation object, a set of handles used to manipulate the model and a visualization of the results. This perspective on software design allows one to specify and simplify the role of each software part.



Model
Stores loaded data and transformed data.
Keeps track of variable definitions.
Keeps note of transformations used and change in Yule indices.
View
Displays current loaded data and transformations performed.
Shows current available transformations and yule indices.
Show informative error and warning messages when appropriate.
Controller
allows selection of variables.
Semi-automatic – computes valid transformations after variable selection.
Allows resetting of graphical parameters.
Allows save and loading workspace.
Allows export of model to file.
User
Loads data and variable definitions.
Oversees data manipulation.

Figure 2: MVC diagram

[4] *Krasner, Glenn E.; Pope, Stephen T. (Aug–Sep 1988). "A cookbook for using the model–view controller user interface paradigm in Smalltalk-80". The Journal of Object Technology. SIGS Publications.* Also published as "A Description of the Model–View–Controller User Interface Paradigm in the Smalltalk-80 System" (Report), ParcPlace Systems.

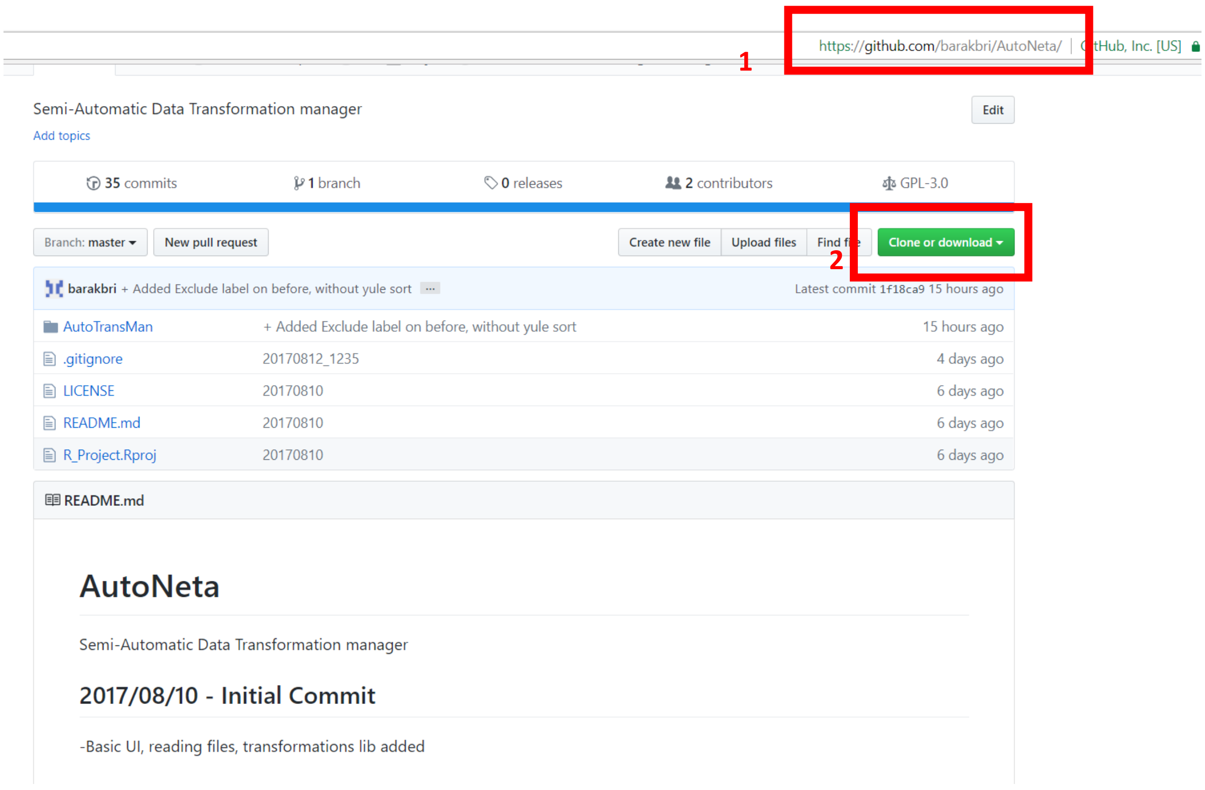
5. Online availability and long-term support

The software described above, along with it's available source code can be retrieved from a public GitHub repository: https://github.com/barakbri/AutoNeta/.

Software is available under the GNU Public License (GPL) V3.0

High software modularity allows for:

- **Continued development:** addition of indices, transformations, data types and themes for the plots.
- **Collaboration and open source development:** code is available, documented and extendable. Other users can view the code, report issues, request features and 'fork' from the current development branch, to create their flavors of the software.
- **Rapid Deployment:** software can be downloaded and run with several clicks (see in figure below).



Semi-Automatic Data Transformation manager

barakbri + Added Exclude label on before, without yule sort ... Latest commit 1f18ca9 15 hours ago

AutoTransMan	+ Added Exclude label on before, without yule sort	15 hours ago
.gitignore	20170812_1235	4 days ago
LICENSE	20170810	6 days ago
README.md	20170810	6 days ago
R_Project.Rproj	20170810	6 days ago

README.md

AutoNeta

Semi-Automatic Data Transformation manager

2017/08/10 - Initial Commit

-Basic UI, reading files, transformations lib added

Figure 3: Downloading the application and source code via GitHub

6. Summary

When one speaks of visualization in the field of statistics, one often refers to visualization of data or results. In this work, the workflow of the statistician itself was the subject of visualization. All tasks fit for automation (e.g. choosing valid transformations, sorting their relevance, selecting bin size and kernel width) were carried out by software. The statistician's sole role is selecting the transformation most suitable for the data. The automation of technical tedious tasks has turned the statistician from a piano tuner to an orchestra conductor, overseeing the proper analysis of data.

The art of building statistical models is an elegant and gentle one. Assumptions need to be verified and data carefully handed. Otherwise one might reach a model which is meaningless, a 'garbage in / garbage out ' scenario. As the age of Big Data enfolds, the analysis of datasets with hundreds of variables become common. The use of automation for the preprocessing of datasets does becomes mandatory, ensuring one does reach a scenario of 'big garbage in / big garbage out'.

Appendix

Taxonomy of variables and their appropriate transformations[5]

Amounts are measurements taking any positive value $y \geq 0$ (e.g. height, blood pressure). Some measurements are inherently bounded from below, i.e. $y \geq$ C, in which case $y - C$ is a valid amount. In order to arrive at a symmetric distribution for amounts, we can use any power transformation directly (for handling amounts that include zeros see Online supplement B).

Counts are counts of units, and therefore can only take integer values 0,1,2,…(e.g. number of adverse events, days under antibiotic treatment). Counts tend to be right-skewed and to have variance increasing with their mean, stemming from the Poisson nature of their distribution. The square root is a variance stabilizing transformation for Poisson counts and often achieves symmetry.

Ratio is one amount *x* divided by another amount *y*, r=x/y (e.g. $Speed = Distance/Time$; Protein to Creatinine ratio in the urine). Ratios often require log transformations. This then takes the form of $\log(r) = \log(x) - \log(y)$. Other power transformations can replace log, in the form of $x^p - y^p$, if the two elements are available.

Fraction is a ratio of one amount x to some other amount y, which is always larger, i.e. 0 ≤ r=x/y ≤ 1 (e.g. fraction of a blood vessel that is clotted). Usually deciding upon the property measured by the ratio r is equivalent to deciding to measure its complement 1-r (instead of the fraction of a blood vessel that is clotted, one could define the fraction that is open). $\text{Log}\left(\frac{r}{1-r}\right)$, or as it is called the 'logit' transformation, is the most useful (the widely used log odds for comparing morbidity).

Counted fraction is the fraction of counts, counting how many out of how many have some property (n out of m). It is therefore bound between 0 and 1. (e.g. the number of patients admitted on a day due to a specific symptom out of the total number of admitted patients that day). Here too, the most useful transformation is the logit transformation: $\log\left(\frac{n+1/3}{m-n+1/3}\right)$, we add a constant c=1/3 to avoid dividing by zero[2].

Amounts that are inherently bounded $a \leq y \leq b$, are essentially fractions, given by $0 \leq r = \frac{y-a}{b-a} \leq 1$ (e.g. the length of time a child sleeps per day). First write them as fractions and then transform them as discussed above.

[5] The importance of non-linear transformations use in medical data analysis. Shachar et al. (2015)

Counts that are inherently bounded $l \leq n \leq m$, are essentially counted fractions, given by $0 \leq r = \frac{n-l}{m-l} \leq 1$ (e.g. the number of correctly answered questions in a questionnaire), and as such, they should be handled in the same manner explained for counted fractions, $\log\left(\frac{n-l+1/3}{m-n+1/3}\right)$.

Difference in amounts, counts or fractions may take positive or negative values (e.g. the difference between the number of words forgotten in an immediate recall assignment and number of words forgotten in a delayed recall assignment). Differences are best handled by transforming the subtracted variables separately, be they amounts or counts. If the two variables that are differenced, are not available the difference variable should rarely be transformed, instead it can practically be only positive where it should be handled as amount.

Ranks are bounded counts of how many are below and equal the observation out of the total number ranked (e.g. the rank of a specific symptom descriptive word out of all descriptive words used). They will be treated exactly as bounded counts.

Ordinal variables are variables whose values are named categories that can be naturally ordered. (Everyday cognition assessment taking the values 1 thru 4). Each category can be assigned a numeric value (similar to Ranks with ties) depending on the proportion of cases in the category and below it, compared to the background of reference distribution