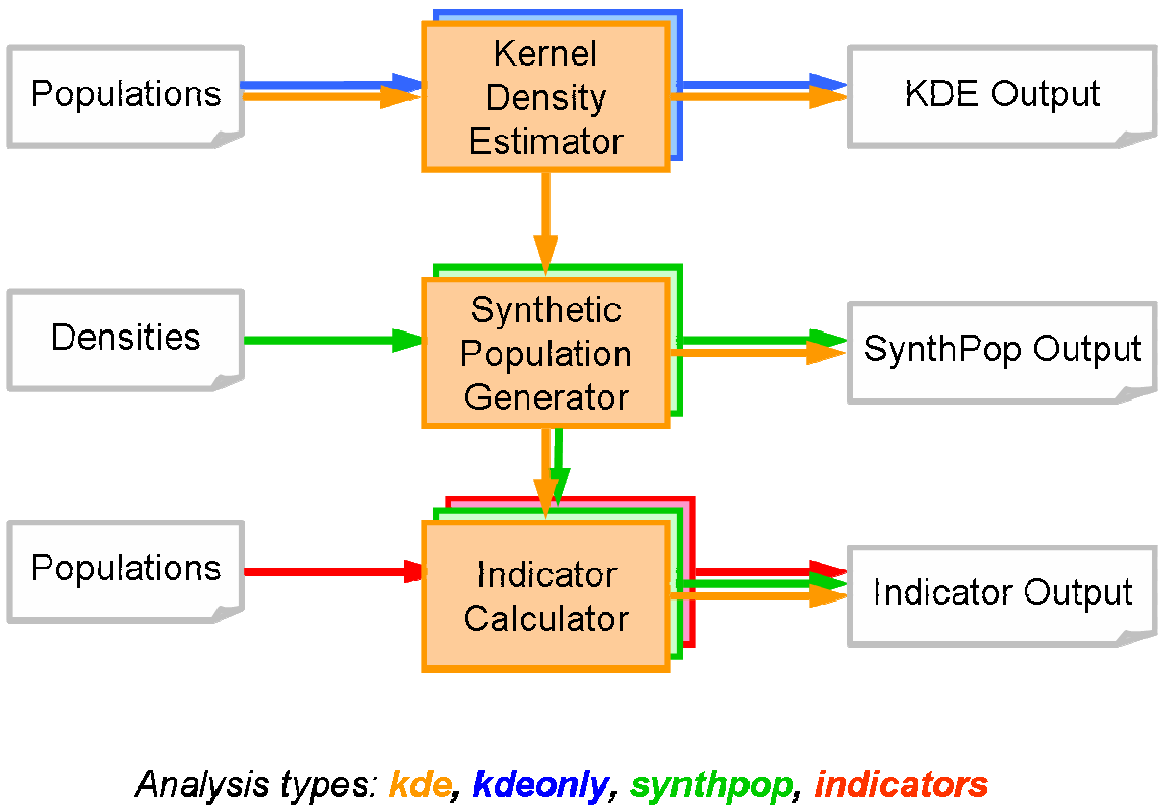
# KDE Tool Documentation

## *Program Flow*

The KDE Tool program allows a user to perform a series of mathematical operations on a set of input data as described in the following high-level block diagram:



The central column of boxes represents the primary stages of mathematical operations and the coloured lines represent the flow of information for the four mutually exclusive types of analysis that can be performed with the program. The grey file icons on the left are the possible inputs and the grey icons on the right are the outputs of each stage.

The orange line represents the most complete analysis type, called *kde*, in which a file containing a series of values called a ‘population’ is the primary input (note: see below for ways of operating on more than one input file at once). The values in the population are also referred to as ‘incomes’ in this document. The first stage of the pipeline performs Kernel Density Estimation to estimate the probability density underlying the population given. This density information is then passed to the second stage, in which a new, synthetic population is generated. The third stage of the pipeline computes the values of a number of indicators using the synthetic population from the second stage as input.

The blue line represents the *kdeonly* analysis type, in which KDE is performed on an input population to estimate the underlying probability density with no subsequent processing.

The green line represents the *synthpop* analysis type, in which the KDE step is bypassed with the user simply providing density information directly.

The red line represents the *indicators* analysis type, in which the user supplies a population (synthetic or not) to the indicator calculator directly.

## *Program Inputs*

Depending on the analysis type to be used, the program accepts either populations or densities:

- **Populations** are specified as singles column of numerical values: each value is a population member or income.
- **Densities** are specified as either two or three columns of numerical values (tab-delimited in the input files). The first column contains the incomes at which the density is reported. The second column reports the corresponding density, f(x). The third column is optional. It reports the cumulative density, F(x). If the third column is not present, the cumulative density values are found by integrating the probability density values.

The program fundamentally operates on a single input population or density at a time, but has built-in facilities to accept a series of such sets as well as single files. All files must be in plain ASCII text format but they may have any three-letter extension (e.g. 'txt' or 'csv' or 'abc'). The input file types accepted are:

- **Single**: if this option is chosen when inputting either a population or a density, then a single text file is specified, as outlined above.
- **File-series**: if this option is chosen when inputting either populations or densities, the program will perform its analysis on a series of similarly-named files. The name of the file specified must contain a number and the rest of the files in the series must have the same name, with a different number for each file (e.g. input1.txt, input2.txt, input3.txt). Note on a limitation of this approach: the path to this file on disk must not contain any numbers!
- **Matrix**: this option represents a second way to specify a series of population. In this case, the file specified contains multiple tab-delimited columns, one column per population.

The input file is the only parameter which can accept a series of values in the primary modes of operation of the program. Please refer to the command-line interface section of this document for more information on running large numbers of analyses.

## *Program Outputs*

Each stage of the pipeline outputs a set of files to a directory in the same location as the input file. The following three sections detail each stage's output, but no matter which analysis type is used, the program will always output the following files:

- parameters.txt: each analysis can be fully described by a set of parameters, which is output in human-readable form in this file.
- parameters.xml: the parameter set (which internally to the program is called its 'state') is also output in machine-readable form in this file. This file can be loaded into back into the program to re-run the same analysis if desired. This file format is documented in the appendix to this document.

### KDE Output

- bandwidths.txt: lists the bandwidths used in KDE, if the *variable from file* option is not chosen. Each population's bandwidth is listed on its own line in the file.
- support.txt: lists the supports used in KDE in two columns, one line per population. If the *from file* option is chosen, a star (*) may appear is a third column if any of the exogenous points are outside of the 'feasible support' (see below).
- densityX.txt: lists the estimated density for the Xth population
- reflected-densityX.txt: lists the estimated density after reflection, if reflection is chosen as an option, for the Xth population

### SynthPop Output

- genstats.txt: lists the population size and representation ratio for every synthesized population
- synthpopX.txt: lists the synthetic population members for the Xth population

### Indicator Output

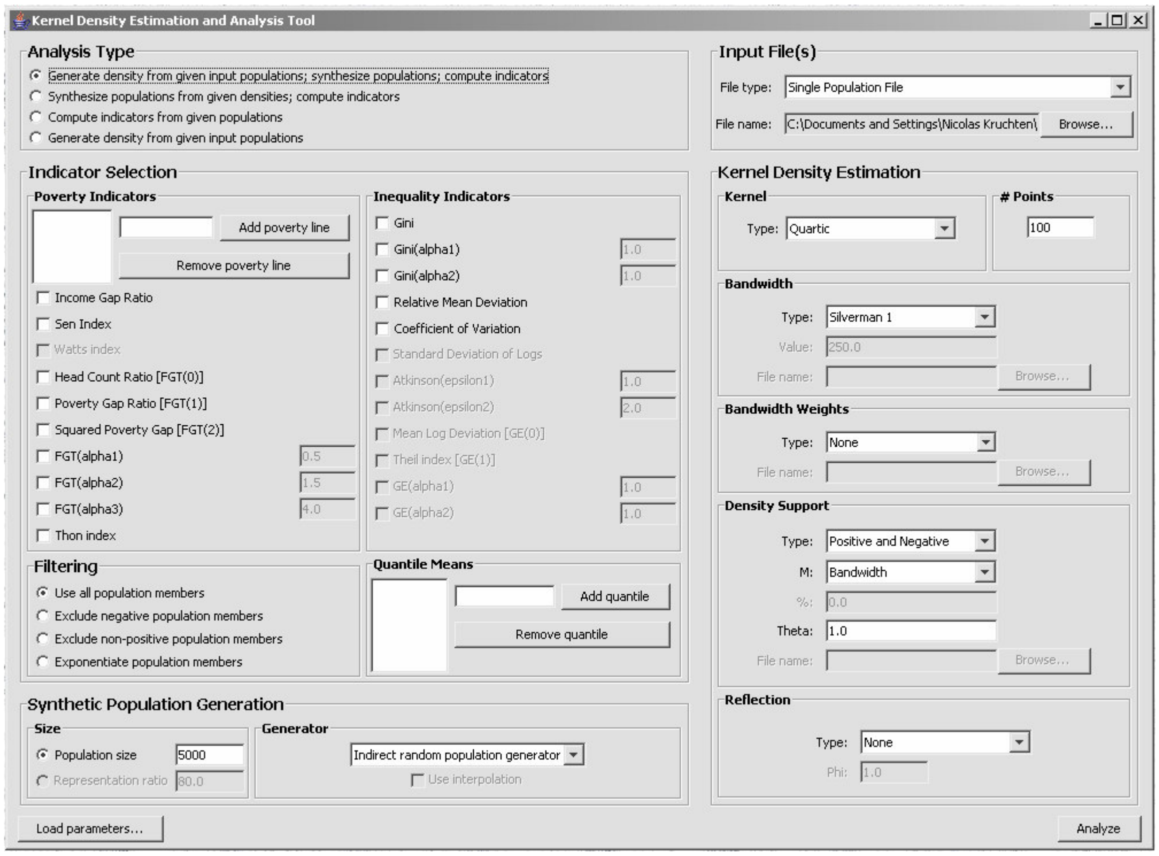
- PovertyIndicators-X.txt (where X is the poverty line used in computation): lists the computer values of the poverty indicators for every population for the given poverty line X
- InequalityIndicators.txt: lists the computed values of the inequality indicators
- quantile-X.txt (where X is the number of quantiles requested): lists the computer quantile means for the given quantile X

## *Program Interface Modes*

The program has two primary interface modes: a graphical user interface and a command-line interface. The latter has three sub-modes of operation: file execution, metafile processing and metafile execution.

## Graphical User Interface (GUI)

The program's GUI allows the user to specify in a single screen all of the various options and parameters that define the analysis pipeline to be used (the internal representation of this set of parameters is what is output into the two parameter files described above):


Kernel Density Estimation and Analysis Tool

Analysis Type
- [✓ Generate density from given input populations; synthesize populations; compute indicators]
- Synthesize populations from given densities; compute indicators
- Compute indicators from given populations
- Generate density from given input populations

Input File(s)
- File type: Single Population File
- File name: C:\Documents and Settings\Nicolas Kruchten\

Indicator Selection

Poverty Indicators
- Income Gap Ratio
- Sen Index
- Watts index
- Head Count Ratio [FGT(0)]
- Poverty Gap Ratio [FGT(1)]
- Squared Poverty Gap [FGT(2)]
- FGT(alpha1) 0.5
- FGT(alpha2) 1.5
- FGT(alpha3) 4.0
- Thon index

Inequality Indicators
- Gini
- Gini(alpha1) 1.0
- Gini(alpha2) 1.0
- Relative Mean Deviation
- Coefficient of Variation
- Standard Deviation of Logs
- Atkinson(epsilon1) 1.0
- Atkinson(epsilon2) 2.0
- Mean Log Deviation [GE(0)]
- Theil index [GE(1)]
- GE(alpha1) 1.0
- GE(alpha2) 1.0

Filtering
- [✓ Use all population members]
- Exclude negative population members
- Exclude non-positive population members
- Exponentiate population members

Quantile Means

Synthetic Population Generation
- Size: [✓ Population size] 5000
- Representation ratio 80.0
- Generator: Indirect random population generator
- Use interpolation

Kernel Density Estimation
- Kernel Type: Quartic
- # Points: 100
- Bandwidth Type: Silverman 1
- Value: 250.0
- Bandwidth Weights Type: None
- Density Support Type: Positive and Negative
- M: Bandwidth
- %: 0.0
- Theta: 1.0
- Reflection Type: None
- Phi: 1.0

Each stage of the analysis pipeline has a corresponding options pane in the GUI which the user can use to specify the various parameters that define that stage (see below for a description of the relevant mathematical operations and their parameters). The analysis type is specified at the top left, and the input file is specified at the top right. When the user is satisfied with the options chosen, the 'Analyse' button at the lower right will actually execute the analysis. The user can load a previously-generated XML parameter file with the button in the lower left (see above for information about the XML parameter file).

## Command Line Interface (CLI)

The program's CLI exists primarily to enable automated usage from external programs, and to run large numbers of similar analyses very simply. The CLI mode of operation is non-interactive, and the program does not output to the command line at all. In fact, the command line output stream is closed very rapidly, even before the program has finished running. All output is directed to a log file in the same directory as the executable called 'kde.log'. The user can be certain that the program has stopped running when its lock on the log file is released.

Anyone wishing to invoke the KDE Tool from an external program can easily do so by studying the appendix and having their external program write one or more XML parameter files or XML metafiles to disk before invoking the KDE Tool via some version of the ‘exec()’ function call available in most programming languages/environments.

### File Execution Mode

In this mode, the program is passed in one or more XML parameter files (see documentation in appendix) as arguments and it simply executes the analysis on each. The command-line command to invoke this mode is:

```
C:/path/to/KDETool.exe [space-delimited list of paths to filenames]
```

### Metafile Processing Mode

In this mode, the program is passed a XML parameter metafile (see documentation in appendix) by way of arguments. It uses that information to generate a number of XML parameter files (see documentation in appendix) which can then be run using the file execution mode. This mode exists primarily as a quick check before using the metafile execution mode. The command-line command to invoke this mode is:

```
C:/path/to/KDETool.exe –processmetafile [filename] [optional integer]
```

If the optional integer is not specified, this mode will generate at most 100 XML files. If more are expected and desired, the second parameter should be the number of XML files expected. This is to prevent user or program errors causing very large numbers of files to be generated accidentally.

### Metafile Execution Mode

In this mode, which is very similar to the metafile processing mode above, the program does not output the XML parameter files, but runs the analyses directly, generating only the analysis outputs.

```
C:/path/to/KDETool.exe –executemetafile [filename] [optional integer (see above)]
```

## *Mathematical Details*

### Parameters used in the Kernel Density Estimation stage

Two comprehensive resources on non-parametric Kernel Density Estimation are: Silverman (1986)[1] and Pagan and Ullah (1999)[2].

[1] Silverman, B.W. (1986), *Density estimation for statistics and data analysis*, Chapman and Hall, New York.

[2] Pagan, A. and A. Ullah (1999), *Nonparametric econometrics*, Cambridge University Press, Cambridge

## Number of points (of estimation)

This is the number of observations at which the density is estimated. They are evenly spaced along the density support.

## Kernel

Six kernels are implemented in the KDE Tool: Gaussian, Uniform, Epanechnikov[3], Quartic, Triangular and Triweight.

## Bandwidth function

The bandwidths are the optimal "rule-of-thumb" bandwidths proposed by Silverman (1986, pages 45-47, Equations 3.28, 3.29 and 3.31). The standard deviation of the data is estimated with the (N-1) correction in the denominator for sample sizes smaller than 36. Given the theory of canonical kernels, each kernel has a distinct multiplicative constant in the formula of the bandwidth associated with it (Marron and Nolan, 1998)[4]. The user can also specify a fixed, constant bandwidth. Finally, the user can specify the bandwidths by providing a file or series of files formatted like the population input files, one bandwidth per population member. If the population input file type is *single*, then the bandwidth file must contain a single column of bandwidth values, otherwise it must either contain a single column of bandwidth values to be applied to all populations or the same number of columns of bandwidth values as there are populations.

## Bandwidth weights

The user can specify a set of multiplicative bandwidth weights. The resulting bandwidth is "adaptive" in the sense that it varies with the point of observation according to the vector of weights specified by the user. The weights are specified by through file input the same way as the bandwidth file option above.

## Density support

The density support is of the form [ $x_{\min} - \theta \times M$ , $x_{\max} + \theta \times M$ ] if density type is *positive and negative*, and (0, $x_{\max} + \theta \times M$ ] if density type is *positive only*, where $x_{\min}$ and $x_{\max}$ are the minimum and maximum observations in each input population, $\theta$ is a user-specified constant and M can be specified by the user to be any of the following three quantities:

- The bandwidth
- The standard deviation of the input population

[3] The Epanechnikov kernel corresponds to the "epan2" option in the "kdensity" STATA routine.

[4] J.S. Marron and D. Nolan (1988), Canonical kernels for density estimation, *Statistics and Probability Letters*, Vol. 7, Issue 3, pp 195-199.

- A percentage of the minimum income in the input population for the lower bound and a percentage of the maximum income in the input population for the upper bound.

The density support can also be specified by the user by loading a text file (if the support type is *from file*). The file format is the same as that of a population input file. If any of the exogenous points are outside of the endogenous 'feasible support' as would have been computed had the *from file* option not been chosen, the min or max value of the feasible support will be used, and the support.txt output file will indicate this occurrence.

## Reflection method

The two available options for boundary bias corrections are *partial* and *total* reflection. *Partial* reflection entails reflecting the data points in the interval [ $z_{\min}$ , $z_{\min} + \varphi \times h$ ], where $\varphi$ is user-specified. *Total* reflection entails reflecting the data points in the interval $[z_{\min},\ \mathrm{z}_{\max}]$ where $z_{\min}$ and $z_{\max}$ are the lowest and highest points of estimation from the density estimation analysis undertaken prior to reflection.

## Parameters used in Synthetic Population Generation stage

The synthetic population is a series of observations generated on the basis of the estimated densities. The following parameters are used in the generation of synthetic populations:

- **Size**: represents the number of observations in the synthetic population (see also the definition of the representation ratio below)
- **Generator**: Four generator methods are available: Deterministic, Deterministic with interpolation, Direct, Indirect. They are described below.

## Deterministic population generator

The synthetic population is generated directly and deterministically from the estimated densities by requiring that, up to rounding, the proportion of observations in the synthetic population that are identical to points of estimation should be equal to the density estimated at those points of estimation. For this generator, the user can specify either the size of the population or a representation ratio. Specifying the size of the population does not guarantee that all points of estimation will be represented in the synthetic population (due to rounding). In contrast, specifying a representation ratio equal to *RR* ensures that the synthetic population is as large as necessary such that *RR* percent of the points of estimation be replicated in the synthetic population.

## Deterministic population generator with interpolation

When choosing *use interpolation*, the user specifies that linear interpolation is to be performed between consecutive observations from the synthetic population. The interpolation is centered on the point of estimation. Let $\delta$ be the gap between two consecutive points of estimation $x_i$ and $x_{i+1}$, and let $N_i$ be the number of incomes equal to $x_i$ that will be added to the synthetic population under the scheme $N_i$ = (pop. size) x density x $\delta$. Then, regardless if $N_i$ is even or odd, the $N_i$ incomes will be equidistantly spread between ($x_i - \frac{\delta}{2}, x_i + \frac{\delta}{2}$). Each of these incomes will be $\frac{\delta}{N_i}$ apart. The smallest observation in the synthetic population will be $\left( x_i - \frac{\delta}{2} + \frac{\delta}{2N_i} \right)$ and the largest observation in the synthetic population will be $\left( x_i + \frac{\delta}{2} - \frac{\delta}{2N_i} \right)$. The linear interpolation at the lower bound of the estimated density support (say, given by a constant $A > 0$) works as follows: if $N_i$ is even, the observations in the synthetic population will be distributed in the interval [b, $A + \frac{\delta}{2}$), where b is the greater of 0 and $A - \frac{\delta}{2}$. More precisely, $\frac{N_i}{2}$ of the interpolated observations will be equidistantly distributed between [0, $A$] while the remaining $\frac{N_i}{2}$ interpolated observations will be equidistantly distributed in the interval [$A$, $A + \frac{\delta}{2}$). If $N_i$ is odd, then the observations will be distributed in the interval [0, $A + \frac{\delta}{2}$) according to the rule described above for any point inside the boundary support. Finally, at the upper bound, the highest possible interpolated observation will be $\left( x_{\max} + \frac{\delta}{2} \right)$ where $x_{\max}$ is the highest point of estimation.

## Indirect random population generator

The generator uses the inverse of the Cumulative Distribution Function $F_{KDE}^{-1}(x)$ to transform a uniform random variable on [0, 1] to a kernel density estimate random variable on the given density support. The procedure is outlined below:

1. Generate pseudo-random number $x_{u,i}$. The underlying probability density function is uniform on the interval [0, 1].
2. Find $x_{KDE,i} = F_{KDE}^{-1}(x_{u,i})$. If no analytical expression for $F_{KDE}^{-1}(x)$ is available, then iterate on $x_{KDE,i}$ until the condition $F_{KDE}(x_{KDE,i}) = x_{u,i}$ is met.
3. Add $x_{KDE,i}$ to the synthetic population.
4. Repeat until the synthetic population is full.

Notes: First, if analysis type *synthpop* is used and the estimated density is read from a file, the density is numerically integrated to find $F(x)$. Second, if analysis type *kde* is used, and kernel density estimation is performed, then $F(x) = \frac{1}{n}\sum_{i=1}^{n} F_k\left(\frac{x - x_i}{h(x_i)}\right)$ where $F_k(x)$ is the Cumulative Distribution Function of the chosen kernel.

## Direct random population generator

The generator is based on an algorithm given in "Non-Uniform Random Variate Generation" by Dr. Luc Devroye (page 765) and is also described on pages 678-679 in Hormann and Leydold (2000)[5].

## Poverty and Inequality indicators

The following poverty and inequality indicators can be computed by the KDE Tool:

- Poverty indicators[6]:
    - Poverty headcount ratio
    - Poverty gap ratio
    - Squared poverty gap
    - Income gap ratio
    - Up to three additional FGT($\alpha$) indicators with user-specified values of $\alpha$
    - Sen index
    - Watts index
    - Thon index
- Inequality indicators:
    - Gini index
    - Up to two generalized Gini($\alpha$) indices with user-specified values of $\alpha$
    - Relative mean deviation
    - Coefficient of variation
    - Standard deviation of logs
    - Mean log deviation
    - Theil index
    - Up to two additional GE($\alpha$) indicators with user-specified values of $\alpha$
    - Up to two measures from the Atkinson($\varepsilon$) class with user-specified values of $\varepsilon$

The formulae for the poverty and inequality indicators are provided in the second Appendix to this document.

[5] W. Hormann and J. Leydold (2000), "Automatic Random Variate Generation for Simulation Input", Proceedings of the 2000 Winter Simulation Conference.

[6] If no poverty line is specified by the user, then no poverty indicators are computed.

## Quantile means

This option enables the user to specify the numbers of quantiles of the synthetic population (for analysis types *kde* or *synthpop*) or input population (for analysis type *indicators*) for which quantile means are computed.

## Filtering

The filtering function enables the application of a transformation to the observations from the synthetic populations (for analysis types *kde* or *synthpop*) or input populations (for analysis type *indicators*) prior to computation of poverty and inequality indicators and/or quantile means. There are four self-explanatory transformations to the population members (i.e., the observations):

- Use all population members;
- Exclude negative population members;
- Exclude non-positive population members, and
- Exponentiate population members.

# Appendix: XML parameter file and XML parameter metafile documentation

## *General Context and Guidance*

The XML parameter file mentioned repeatedly in the documentation above is in fact an automatically-generated XML representation of a Java java.util.prefs.Preferences object, which is well-documented online, along with the XML schema for the XML file generated to serialize it. In cases of difficulty interpreting the current document, one could refer to the more general Preferences documentation for extra context.

Reference: http://java.sun.com/j2se/1.4.2/docs/api/java/util/prefs/Preferences.html

The KDE Tool stores all user choices for a given analysis run into an internal object called the State. The State is represented internally as a long list of key-value pairs stored inside a Preferences object. When a KDE Tool analysis run is executed, this Preferences object is output in 2 formats which contain the same key-value pair information: the parameters.txt file and the parameters.xml file. This knowledge should aid any reader familiar with the parameters.txt file in the generation and editing of the XML version. When the KDE Tool loads an XML parameter file, it simply sets its internal State object's key-value pairs according to the list found in the XML file.

The graphical user interface for the KDE Tool is basically just a human-friendly means of specifying the values of the State object, so experience with the KDE Tool GUI should make any reader well-prepared to understand the meaning of and interactions between the various key-value pairs listed in this document.

Finally, any Java-literate reader with access to the source code may find the classes in the 'com.kruchten.columbia.kde.state' package more compact and readable than this verbose document.

## *Basic Format of the XML Parameter File*

We can consider the XML parameter file to have three components: the header, the list and the footer. Neither the header nor footer should change from file to file. What follows is an example of the header, a generic list and the footer:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE preferences SYSTEM "http://java.sun.com/dtd/preferences.dtd">
<preferences EXTERNAL_XML_VERSION="1.0">
<root type="user"><map/><node name="com"><map/><node name="kruchten"><map/>
<node name="columbia"><map/><node name="kde"><map/><node name="state"><map>

<entry key="a" value="b"/>
<entry key="c" value="d"/>



</map></node></node></node></node></node></root></preferences>
```

The parameter file above contains the key-value pairs a=b and c=d, as per the two lines in the list portion of the file (the middle portion, delimited by the vertical whitespace).

The rest of this document lists the possible key-value pairs that can exist in a valid XML parameter file and how they should be used.

## *Key-Value Pairs*

The following list outlines allowed key-value pairs, their default values and any restrictions on them. If a key-value pair is omitted from the XML file, its default value will be used. If any filenames are missing or refer to files which do not exist, an error will be thrown.

- Key: 'analysis type'
  - Value: one of 'kdeonly', 'kde', 'synthpop' or 'indicators'
  - Default: kde

- Key: 'input type'
  - Value: one of 'single', 'fileseries' or 'matrix'
  - Default: 'single'

- Key: 'input file(s)'
  - Value: path to a file

- Key: 'kernel name'
  - Value: one of 'gaussian', 'uniform', 'epanechnikov', 'quartic', 'triweight', 'triangular'

- Key: 'bandwidth name'
  - Value: one of 'constant', 'silverman1', 'silverman2', 'silverman3', 'variablefromfile'
  - Default: 'constant'

- Key: 'bandwidth constant'
  - Value: real number
  - Default: 250.0
  - Note: required if 'bandwidth name' is constant, ignored otherwise

- Key: 'bandwidth filename'
  - Value: path to a file
  - Note: required if 'bandwidth name' is 'variablefromfile', ignored otherwise

- Key: 'bandwidth weights;
  - Value: one of 'true' or 'false'
  - Default: 'false'

- Key: 'bandwidth weightsfile'

    - Value: path to a file
    - Note: required if ‘bandwidth weightsfile’ is ‘true’, ignored otherwise

- Key: ‘support name’
    - Value: one of ‘positiveandnegative’, ‘positive’ or ‘file’
    - Default: ‘positiveandnegative’

- Key: ‘support theta’
    - Value: real number
    - Default: 1.0

- Key: ‘support M’
    - Value: one of ‘bandwidth’, ‘stddev’, ‘percentage’
    - Default: ‘bandwidth’

- Key: ‘support percentage’
    - Value: a real number
    - Default: 0
    - Note: required if ‘support M’ is ‘percentage’, ignored otherwise

- Key: ‘support resolution’
    - Value: an integer
    - Default: 100
    - Note: required if ‘support name’ is not ‘file’, ignored otherwise

- Key: ‘support filename’
    - Value: path to a file
    - Note: required if ‘support name’ is ‘file’, ignored otherwise

- Key: ‘reflector name’
    - Value: one of ‘off’, ‘total’ or ‘partial’
    - Default: ‘off’

- Key: ‘reflector phi’
    - Value: a real number less than or equal to 1.0
    - Default: 1.0
    - Note: required if ‘reflector name’ is ‘partial’, ignored otherwise

- Key: ‘synthesizer name’
    - Value: one of ‘deterministic’, ‘indirect’, ‘direct’
    - Default: ‘deterministic’

- Key: ‘synthesizer use rep ratio’
    - Value: ‘true’ or ‘false’
    - Default: ‘false’

  - Note: required if ‘synthesizer name’ is ‘deterministic’, ignored otherwise

- Key: ‘synthesizer rep ratio’
  - Value: integer between 0 and 100
  - Default: 80
  - Note: required if ‘synthesizer name’ is ‘deterministic’ and if ‘synthesizer use rep ratio’ is ‘true’, ignored otherwise

- Key: ‘synthesizer pop size’
  - Value: integer
  - Default: 500
  - Note: required if ‘synthesizer use rep ratio’ is ‘false’, ignored otherwise

- Key: ‘synthesizer use interpolation’
  - Value: ‘true’ or ‘false’
  - Default: ‘false’
  - Note: required if ‘synthesizer name’ is ‘deterministic’, ignored otherwise

- Key: ‘quantiles’
  - Value: comma delimited list of quantiles
  - Default: empty

- Key: ‘poverty lines’
  - Value: comma delimited list of poverty lines
  - Default: empty

- Key: ‘indicator filter’
  - Value: one of ‘all’, ‘onlypositive’, ‘onlynonnegative’, ‘exponentiated’

For the following keys, the value is either ‘true’ or ‘false’:

- 'indicator Income Gap Ratio active'
- 'indicator Sen Index active'
- 'indicator Watts index active'
- 'indicator Head Count Ratio [FGT(0)] active'
- 'indicator Poverty Gap Ratio [FGT(1)] active'
- 'indicator Squared Poverty Gap [FGT(2)] active'
- 'indicator FGT(alpha1) active'
- 'indicator FGT(alpha2) active'
- 'indicator FGT(alpha3) active'
- 'indicator Thon index active'
- 'indicator Gini active'
- 'indicator Gini(alpha1) active'
- 'indicator Gini(alpha2) active'
- 'indicator Relative Mean Deviation active'

- 'indicator Coefficient of Variation active'
- 'indicator Standard Deviation of Logs active'
- 'indicator Atkinson(epsilon1) active'
- 'indicator Atkinson(epsilon2) active'
- 'indicator Mean Log Deviation [GE(0)] active'
- 'indicator Theil index [GE(1)] active'
- 'indicator GE(alpha1) active'
- 'indicator GE(alpha2) active'

For the following keys, the value is a real valued parameter number:

- 'indicator FGT(alpha1) parameter'
- 'indicator FGT(alpha2) parameter'
- 'indicator FGT(alpha3) parameter'
- 'indicator Gini(alpha1) parameter'
- 'indicator Gini(alpha2) parameter'
- 'indicator Atkinson(epsilon1) parameter'
- 'indicator Atkinson(epsilon2) parameter'
- 'indicator GE(alpha1) parameter'
- 'indicator GE(alpha2) parameter'

## *The XML Parameter Metafile Format*

The XML parameter metafile format was developed to provide a compact way of storing the definition of a series of analyses rather a single analysis, as in the XML parameter file.

The metafile has exactly the same structure and format as the XML parameter file, except that it accepts multiple values for some of the keys. The metafile-related CLI modes of the program (see the documentation above for details on invoking those modes) operate on what can be called the 'feasible Cartesian product' of all of the values found in the metafile. By way of example, if the metafile contained the key-values:

a=b,c; d=e,f; g=h,i

and if a=b dictates that d is ignored and a=c dictates that g is ignored, then the output would be the following set of four files:

a=b; g=h        a=b; g=i        a=c; d=e        a=c; d=f

There are two ways of specifying more than one value in a metafile: enumeration and generation. In enumeration, the value for a key is a semi-colon-delimited list of values. In generation, the value for a key is a space-delimited triplet of numbers indicating min, max and increment. The values used are then:

min, min+increment, min+2*increment, min+3*increment, …, max

(inclusive of min, and max as well if max=N*increment with integer N).

The following keys accept enumerated values:

- input file(s)
- kernel name
- bandwidth name
- bandwidth constant
- bandwidth filename
- bandwidth weights
- bandwidth weightsfile
- support name
- support M
- support percentage
- support resolution
- support filename
- reflector name
- synthesizer name
- synthesizer use rep ratio
- synthesizer rep ratio
- synthesizer pop size
- synthesizer use interpolation
- indicator FGT(alpha1) parameter
- indicator FGT(alpha2) parameter
- indicator FGT(alpha3) parameter
- indicator Gini(alpha1) parameter
- indicator Gini(alpha2) parameter
- indicator Atkinson(epsilon1) parameter
- indicator Atkinson(epsilon2) parameter
- indicator GE(alpha1) parameter
- indicator GE(alpha2) parameter

Important notes: the poverty lines and quantiles are still comma-delimited, and that comma-delimited list is treated a single parameter value, rather than generating a new XML parameter file and/or output for each combination of poverty line and quantile! Please also note that given that the paths are semi-colon-delimited, one cannot use paths that contain semi-colons.

The following keys accept generated values:

- support theta
- reflector phi

Important notes: to use a single value for the generated fields, the user must still specify all three values, but if the increment is set to 0, then the min value will be used as the single value.

## Appendix: Formulae for poverty and inequality indicators

### *Poverty indicators*
*(where z=the poverty line)*

| INDICATOR | FORMULA |
|---|---|
| **Class of Foster, Greer and Thorbecke indices (FGT)** | $P_\alpha = \frac{1}{n}\sum_{i=1}^{n}\left[\frac{z-x_i}{z}\right]^\alpha 1\{x_i \le z\}$ where different values of $\alpha$ give rise to different poverty indicators (including the poverty Headcount Ratio, the Poverty Gap Ratio, the Squared Poverty Gap, etc.) |
| **Income gap ratio** | $I = \frac{z-\overline{x}_q}{z}$ where $\overline{x}_q = \frac{1}{q}\sum_{i=1}^{n} x_i 1\{x_i \le z\}$ |
| **Sen index** | $S = HCR \times [I + (1-I)\times G]$ where G is the Gini coefficient among the poor and is defined by: $G = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n}\left\|x_i - x_j\right\| 1\{x_i \le z\}}{2q^2\overline{x}_q}$ |
| **Watts index** | $WATTS = \frac{1}{n}\sum_{i=1}^{n}\ln(\frac{z}{x_i})\ 1\{x_i \le\ z\}$ |
| **Thon index** | $TH = \frac{2}{n(n+1)z}\sum_{i=1}^{q} g_i(n+1-i)$ where $g_i = z - x_i$ |

## Inequality indicators

| INDICATOR | FORMULA |
|---|---|
| **Gini coefficient** | $G = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n}\left\|x_i - x_j\right\|}{2n^2\bar{x}}$ |
| **"Generalized" Gini coefficient**<br>• The user can specify up to two values of $\alpha$ | $G(\alpha) = \frac{\left[\sum_{i=1}^{n}\sum_{j=1}^{n}\left\|x_i - x_j\right\|^{\alpha}\right]^{\frac{1}{\alpha}}}{2n^2\bar{x}}$ |
| **Relative mean deviation** | $RMD = \frac{1}{2n\bar{x}}\sum_{i=1}^{n}\left\|x_i - \bar{x}\right\|$ |
| **Coefficient of variation** | $CV = \frac{\sigma}{\bar{x}} = \frac{\sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2}}{\bar{x}}$ |
| **Standard deviation of logs** | $SD = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left[(\ln(x_i) - \overline{\ln(x_i)})\right]^2}$ |
| **Theil index (GE 1)** | $GE(1) = \frac{1}{n}\sum_{i=1}^{n}\frac{x_i}{\bar{x}}\ln(\frac{x_i}{\bar{x}})$ |
| **Mean log deviation (GE 0)** | $GE(0) = \frac{1}{n}\sum_{i=1}^{n}\ln(\frac{\bar{x}}{x_i})$ |
| **Atkinson class of measures**<br>• The user can specify values for $\varepsilon$ | $A_{\varepsilon} = 1 - \left[\frac{1}{n}\sum_{i=1}^{n}\left(\frac{x_i}{\bar{x}}\right)^{1-\varepsilon}\right]^{\frac{1}{1-\varepsilon}}$<br>When $\varepsilon = 1$, we have: $A_1 = 1 - \frac{\prod_{i=1}^{n}(x_i)^{\frac{1}{n}}}{\bar{x}}$ |