Chapter 15
PARSEC: An R Package for Poset-Based Evaluation of Multidimensional Poverty

Marco Fattore and Alberto Arcagni

Abstract The paper introduces PARSEC, a new software package implementing basic partial order tools for multidimensional poverty evaluation with ordinal variables. The package has been developed in the R environment and is freely available from the authors. Its main goal is to provide socio-economic scholars with an integrated set of elementary functions for multidimensional poverty evaluation, based on ordinal information. The package is organized in four main parts. The first two comprise functions for data management and basic partial order analysis; the third and the fourth are devoted to evaluation and implement both the poset-based approach and a more classical counting procedure. The paper briefly sketches the two evaluation methodologies, illustrates the structure and the main functionalities of PARSEC, and provides some examples of its use.

15.1 Introduction

PARSEC[1] is a new R (R Core Team 2012) package implementing basic partial order tools for multidimensional poverty evaluation with ordinal variables. Poset theory use overcomes the drawbacks of classical evaluation procedures, which prove scarcely effective and often inconsistent for handling ordinal data (Fattore et al. 2012). The poset-based approach has been primarily developed for poverty and material deprivation assessment (Fattore et al. 2011a,b), but it may be virtually applied to any kind of evaluation problem with ordinal variables, like assessing quality-of-life, well-being, or customer satisfaction. For the sake of completeness,

[1] PARtial orders in Socio-EConomics.

M. Fattore (✉) • A. Arcagni
Department of Statistics and Quantitative Methods, University of Milano-Bicocca, Milano, Italy
e-mail: marco.fattore@unimib.it

R. Brüggemann et al. (eds.), *Multi-indicator Systems and Modelling in Partial Order*,
DOI 10.1007/978-1-4614-8223-9_15,

PARSEC implements also the counting approach to multidimensional poverty evaluation, developed by the Oxford Poverty & Human Development Initiative (OPHI) group (Alkire and Foster 2011a). This procedure is gaining relevance at international level and may be used as a benchmark for the poset approach. The paper is organized as follows: Sect. 15.2 gives a brief sketch of the poset-based and OPHI evaluation procedures; Sect. 15.3 illustrates the main functionalities of PARSEC; Sect. 15.4 provides some scripts, showing PARSEC in action and giving some ideas of its performances; Sect. 15.5 discusses the improvements to be implemented in the next release of the package; Sect. 15.6 concludes and the Appendix provides a list of the functions currently available in PARSEC.

15.2 Poset-Based and Counting Evaluation Methodologies

Multidimensional poverty evaluation increasingly involves ordinal variables. This poses some critical methodological problems: (1) classical evaluation procedures based on variable aggregation are not directly applicable to ordinal data and (2) data are often truly multidimensional and variable interdependencies are too weak to achieve any dimensionality reduction, even conceptually. The scientific community is currently debating these issues and while some scholars stress the relevance of getting synthetic indicators anyway (e.g., for policy-making purposes), others argue their consistency and suggest relying on multidimensional dashboards (Ravaillon 2011). These difficulties are partly unavoidable, due to the complexity of the problems at hand; but they are also amplified by the use of unsuitable statistical tools, borrowed from classical multivariate analysis and based on linear algebra. Linear algebra tools break down when addressing ordinal variables and produce inconsistencies that may be mistaken for an intrinsic impossibility to get well-founded results. The use of partial order theory clarifies that this is not the case, it shows that the computation of synthetic indicators need not require variable aggregation and paves the way to alternative and consistent evaluation procedures. An example of the possibilities offered by partial order theory is provided by the poset-based methodology implemented in PARSEC and briefly introduced in the following. The methodology provides a consistent framework for ordinal evaluation problems, preserving the logic of classical procedures, but using partial order theory for data representation and information extraction. In the following, we limit ourselves to a very essential introduction to the methodology. More complete presentations can be found in Fattore et al. [2011a,b, 2012].

Let $v_1,\ldots,v_k$ be k ordinal variables representing poverty dimensions (we assume that lower degrees of $v_1,\ldots,v_k$ represent higher deprivations). Each variable is recorded on a different scale, possibly with a different number of degrees $m_1,\ldots,m_k$. Each statistical unit in the population is scored against the k variables. The vector $\mathbf{p}=(p_1,\ldots,p_k)$ of k scores associated to a statistical unit is called a *profile*. The set of possible profiles P has cardinality $|P|=m_1\cdot m_2\cdot\ldots\cdot m_k$, even if some profiles may not

be observed within the population. The set P is naturally turned into a partially ordered set $(P, \lhd)$ putting

$$\mathbf{p} \lhd \mathbf{q} \Leftrightarrow p_i \leq q_i \ \forall i \in 1,\ldots,k \tag{15.1}$$

where $\mathbf{p}$ and $\mathbf{q}$ are elements of P. An *evaluation function* $E\ v\ a\ l(\cdot)$ is defined to assign a degree of poverty in $[0, 1]$ to each profile. In fact, the existence of incomparabilities among profiles leads quite naturally to describing poverty on a continuous scale (in a fuzzy spirit). Poset $(P, \lhd)$ is just a mathematical structure; as such, it conveys no information on the degree of poverty of its elements. To transform P in a tool for poverty evaluation, exogenous information is embedded into it choosing a *threshold* τ, that is, by selecting a minimal set of profiles[2] considered as poor and scored 1 by the evaluation function. The choice of the threshold allows the evaluation function to be extended to all of the profiles in P, according to the following procedure:

1. consider the set $L\ E$ of linear extensions of P;
2. for any linear extension $l \in L\ E$, assign poverty score 1 to all the profiles that are below an element of τ in l;
3. assign to profiles of P a final poverty score averaging the scores they get on the elements of $L\ E$.

In practice, given a profile $\mathbf{p}$, $E\ v\ a\ l(\mathbf{p})$ is computed as the relative frequency of linear extensions where $\mathbf{p}$ is below an element of the threshold. By construction, $E\ v\ a\ l(\cdot)$ scores to 1 all the profiles in τ or in the downset of τ and to 0 all the profiles in the intersection of the upsets of the elements of τ. All of the other profiles in P are assigned scores in $]0, 1[$. Finally, each statistical unit in the population is assigned the poverty degree of the profile it shares. Once the population has been assessed in this way, classical overall indicators may be computed, particularly the Head Count Ratio, here defined as the average degree of poverty in the population.

The poset-based methodology provides a radical alternative to classical aggregative procedures. Among the latter ones, the counting approach developed by the OPHI group (Alkire and Foster 2011a) is gaining more and more relevance and its application is spreading. One of its merits is to provide a general and unified framework for multidimensional poverty assessment, even if it suffers from drawbacks typical of aggregative methodologies (Fattore et al. 2012). Due to its importance, the OPHI procedure is implemented in PARSEC, also to provide a benchmark for the poset-based approach.[3] The OPHI procedure is conceptually quite simple. Let $v_1,\ldots,v_k$ be k poverty dimensions, as before. A set $c_1,\ldots,c_k$ of k cutoffs is exoge-

[2] In a multidimensional setting, the threshold need not be composed of just one profile, but may comprise several profiles, since the shapes of poverty can be different and incomparable. It may be proved that a threshold can be always chosen as an antichain (Fattore et al. 2011a).

[3] The OPHI approach can be applied also when cardinal variables are of concern, but here we limit the discussion to the ordinal case.

nously defined, identifying a different poverty threshold for each evaluation dimension. A statistical unit scoring a degree d_i lower than c_i is considered as deprived on dimension v_i. Statistical units are classified as definitely poor if the number of dimensions they are deprived on equals or exceeds an overall cutoff c, also to be defined exogenously. In practice, the OPHI approach defines a yes-or-no evaluation function (more precisely, it defines an *identification* function) and classifies statistical units in just two classes, the poor and the non-poor. The final output of the OPHI procedure comprises the Head Count Ratio, that is, the fraction of statistical units scored as poor, and the Adjusted Head Count Ratio, which is the product of the Head Count Ratio and the average number of deprivations suffered by poor statistical units. This last indicator is of interest since it helps to realize the severity of deprivation in a given population. A complete description and discussion of the methodology can be found in Alkire and Foster [2011a] and Alkire and Foster [2011b]. It is interesting to note that the OPHI approach can be cast in poset terms and can be seen as a special case of the poset-based methodology (Fattore et al. 2011b).

15.3 The Structure of PARSEC and Its Main Functionalities

PARSEC is organized in four main sections, each comprising a set of functions for specific tasks:

1. Data management.
2. Basic poset analysis.
3. Poset-based evaluation.
4. OPHI counting approach.

In the following we give a brief account of each section, referring to the Appendix for a complete list of the functions currently available in the package.

This set of functions is used to build partial orders, possibly out of original data. Function `var2prof` allows the user to specify an arbitrary number of ordinal variables, each coded with a different scale, and produces the list of all the profiles built on them. It is very useful for building posets from scratch. It is also possible to assign a weight to each profile (usually the number of units sharing the profile). Function `pop2prof` extracts all the unique profiles out of a population of statistical units assessed against a set of ordinal variables. It also assigns to each observed profile the correspondent absolute frequency. Once the set of profiles is obtained, it can be turned into a partial order according to expression (15.1). The square binary matrix (usually labeled Z) representing the partial order (i.e., the incidence matrix of the corresponding Hasse diagram) is obtained through function `getzeta`.

The functions of this section manage posets and allow the investigation of their basic features. PARSEC represents posets in matrix terms, so many of its functions rely on matrix calculus. Through functions like `binary`, `reflexivity`, `antisymmetry`, and `transitivity`, one may check whether the input square matrix

is binary and represents a reflexive, antisymmetric, or transitive relation. Checking these properties jointly, functions `is.preorder` and `is.partialorder` verify whether the input matrix represents a preorder or a partial order. Often, it is useful to handle directly the cover relation generating the partial order. The cover relation matrix may be obtained from the partial order matrix using `incidence2cover`, while `cover2incidence` performs the opposite (which is useful, since often it is easier to specify a partial order through the cover relation). Maximal and minimal elements of a poset are directly obtained invoking `maximal` and `minimal`. The heights of poset elements are obtained through `heights`, and similar functions exist to compute the depths and the levels of poset elements (see Patil and Taillie 2004 for appropriate definitions). The poset-based evaluation methodology draws upon the concepts of antichain, downset and upset. PARSEC thus provides functions to get the set of incomparable elements of a given poset element (`incomp`), to check whether a list of elements forms a downset (`is.downset`) or an upset (`is.upset`), to return the downset or the upset generated by a given set of elements (`downset` and `upset`, respectively) and to identify the antichain generating a downset (`gen.downset`) or an upset (`gen.upset`).

PARSEC implements the poset-based approach to evaluation through function `evaluation`. Given the partial order matrix and the selected threshold, `evaluation` returns the evaluation function, the rank distribution of the profiles and the frequency distribution of the distances (rank differences) between a profile and the threshold.[4] Given the number of statistical units sharing each profile, the Head Count Ratio can then be easily obtained. Function `evaluation` is based on a pre-compiled C implementation of the Bubley–Dyer algorithm for (almost) uniform sampling of linear extensions (Bubley and Dyer 1999). Even in medium size posets (e.g., 40–50 elements), the computation of the evaluation function requires sampling several of hundred million linear extensions, a task that an interpreted scripting language like R could only accomplish in a very long time. The pre-compiled C routine decreases dramatically the computation time and allows the evaluation methodology to be applied to larger posets, composed of some hundreds elements. The next section provides some examples of the use of `evaluation` combined with other PARSEC functions; some tests are also presented to give an idea of the package performances.

Function `count` implements[5] in a single call the computations involved in the OPHI counting approach (Alkire and Foster 2011a). Passing to `count` the profiles of the statistical units, the vector of cutoffs on the evaluation dimensions and the overall cutoff, a complete output is returned comprising the Head Count Ratio and the Adjusted Head Count Ratio. As already mentioned, it is easily seen that the OPHI approach can be considered as a special case of the poset-based methodology. The link between the two methodologies is given by function `count2threshold`

[4] Precisely, for any linear extension, the differences between the rank of the higher ranked element of the threshold and the ranks of the other profiles are computed.

[5] The OPHI approach can be applied also when cardinal variables are of concern. PARSEC implements the methodology for ordinal variables only.

which returns the threshold (in profile terms) generating the set of poor profiles identified by `count`. The returned threshold can be directly used in `evaluation`, to compare the results of the two methodologies.

15.4 Some Examples

In this section, we provide some application examples of PARSEC. First, we give a simulated example of multidimensional poverty evaluation using both the OPHI and the poset-based approaches, comparing the results. Then we illustrate the computation of the evaluation function through `evaluation` and show how increasing the number of sampled linear extensions leads to more accurate results. Finally, we test the performances of `evaluation` applying it to a sequence of posets of increasing complexity, comparing the computation times.

All of the computations described in the following were done on an Intel®; Core™2 Duo CPU E8400 @ 3.00GHz ×2, RAM 1.9 GB, equipped with Linux 32 bit operative system.

We simulate a multidimensional poverty evaluation process, on a poset of 40 profiles built out of three ordinal variables recorded on scales with number of degrees 4, 5, and 2, respectively. The simulation compares the OPHI and the poset-based approaches and is organized in three steps:

1. Data definition.
2. Implementation of the OPHI procedure.
3. Implementation of the poset-based procedure and comparison of the results.

Step 1: Data definition. To build the poset, we first define a vector `vs` whose length is the number of variables and whose components are the number of degrees of each variable

```
> vs <- c(4, 5, 2)
```

Vector `vs` is passed to function `var2prof` which generates all $4 \times 5 \times 2 = 40$ poset profiles, computing all the combinations of variable degrees. A vector `freq` of 40 randomized integer numbers is also passed to `var2prof` to simulate a distribution of frequencies on the profiles

```
> prof <- var2prof(vs, freq = rbinom(prod(vs), 100, .5))
```

Step 2: Implementation of the OPHI procedure. To apply the OPHI procedure to the simulated data, the vector `var_cut` of cutoffs and the overall cutoff `over_cut` are first defined

```
> var_cut <- c(1, 1, 1)
> over_cut <- 2
```

According to `var_cut`, a statistical unit is deprived on a variable if the corresponding degree is lower than 1, that is, if it scores 0; according to `over_cut`, a statistical unit is considered as definitely deprived if it scores 0 on at least two variables out of three. The OPHI methodology is now applied to data, calling function `count`

```
> countres <- count(prof, var_cut, k=over_cut)
```

Typing `countres$H` returns the Head Count Ratio

```
> countres$H
[1] 0.2190476
```

Step 3: Implementation of the poset-based procedure. To apply the poset-based evaluation procedure, partial order matrix `Z` is first built out of the profiles

```
> Z <- getzeta(prof)
```

Next a threshold must be defined. To compare the results of the two methodologies, the threshold is chosen to be that implicitly defined in the OPHI approach. This is achieved by typing

```
> threshold <- count2threshold(countres, prof, Z)
```

It turns out that the threshold is composed of profiles 300, 040, 001

```
 > prof$profiles[threshold,]
    Var1 Var2 Var3
P04    3    0    0
P17    0    4    0
P21    0    0    1
```

The evaluation function is computed through `evaluation`, passing to it `Z`, together with `threshold` and other two parameters, namely, an arbitrary linear extension `lin_ext` to initialize the Bubley–Dyer algorithm and the number of iterations `nit` to achieve an almost uniform sampling (the number of iterations depends upon the parameter `error`, i.e., the acceptable maximum total variation distance from a uniform distribution). The initializing linear extension is computed by typing

```
> lin_ext <- lingen(Z)
```

Selecting a maximum total variation distance from uniformity of 10^{-5}

```
> error <- 10^(-5)
```

`nit` is estimated using a formula of Karzanov and Khachiyan (see Bubley and Dyer 1999)

```
> nit <- floor(n^5*log(n)+n^4*log(error^(-1)))
> nit
[1] 407214345
```

where `n` is the number of profiles (here, 40). Finally calling

```
> eval <- evaluation(Z, lin_ext, nit, threshold)
```

the evaluation function is obtained (computation takes about 2 min).

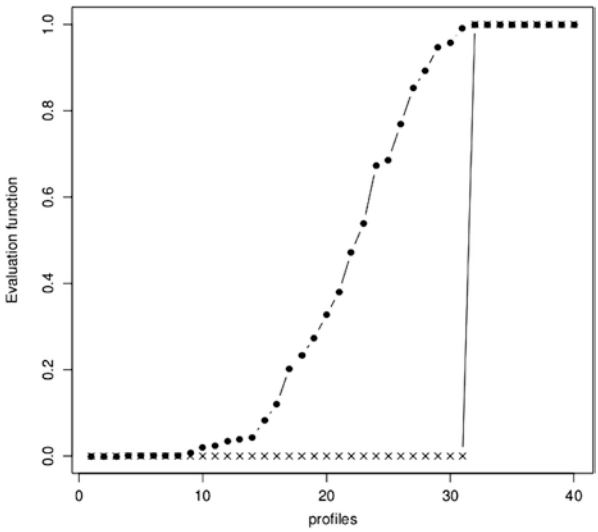

Fig. 15.1 Poset-based (*circles*) and OPHI (*crosses*) evaluation functions

The plot of the evaluation functions obtained from `count` and `evaluation` is depicted in Fig. 15.1 and is obtained by

```
 > ord <- order(eval$poorfreq)
> plot(eval$poorfreq[ord], type="b", pch=16,
    ylab="Evaluation function", xlab="profiles")
> lines(1:40, countres$Z_k[ord], type="b", pch=4)
```

The two evaluation functions coincide on profiles scored to 1 or 0 by the poset-based approach, but differ on all of the other profiles. This difference has a great impact on the Head Count Ratio (`hc`), which turns out to be much greater than that computed by `count`

```
>hc<-as.vector(eval$poorfreq%*%prof$freq/sum(prof$freq))
> hc
[1] 0.4623521
```

We now show how increasing the number of sampled linear extensions reflects on the estimation of the evaluation function. We consider the same poset and the same threshold introduced in the previous paragraph, running `evaluation` seven times, with increasing sample sizes of 10^3, 10^4, 10^5, 10^6, 10^7, 10^8, and 10^9 linear extensions. The computation times range from 0.001 s (sample of 10^3) to about 6 min (sample of 10^9) and increases almost linearly with sample size. The results are displayed in Fig. 15.2. In addition, Table 15.1 reports the distances (measured as the maximum point-wise absolute differences) of the first six evaluation functions from the last, computed on 10^9 linear extensions.

As can be noticed, the evaluation function estimated with 10^4 iterations is worse than that estimated with 10^3. This is due to randomness and to the fact the 10^3 and 10^4 are far below the number of linear extensions needed to approach a uniform distribution.

To check the computational performances of `evaluation` as poset complexity increases, we have run it on a sequence of nine posets, built as the product of

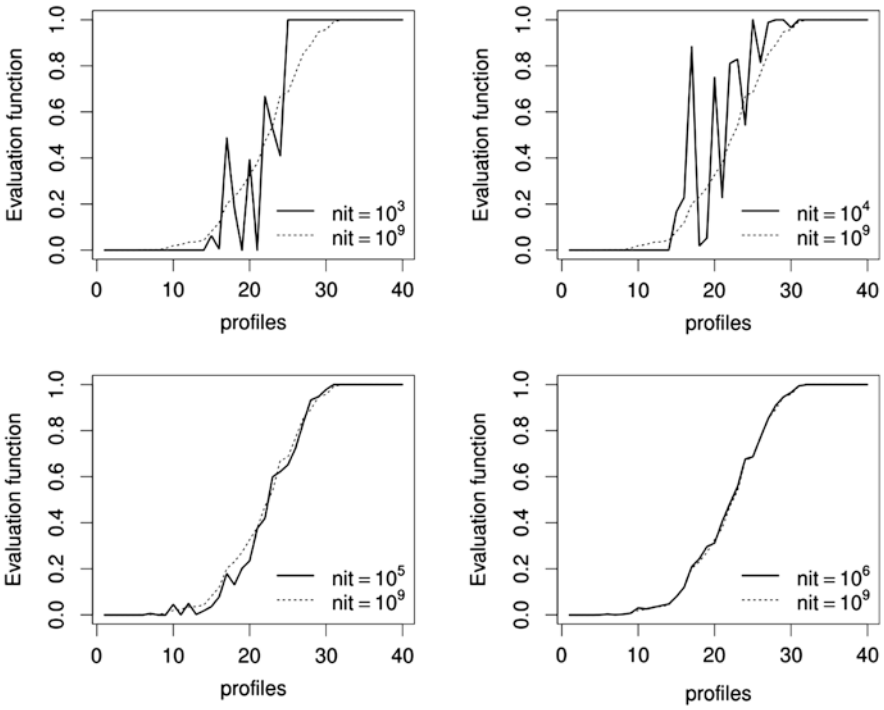

Fig. 15.2 Convergence of the evaluation function as number of iterations (nit) increases

Table 15.1 Distances from the evaluation function computed on a sample of 10^9 linear extensions

Sample size	10^3	10^4	10^5	10^6	10^7	10^8
Distance	0.378	0.682	0.100	0.026	0.017	0.005

Table 15.2 Computation time (in minutes) to run `evaluation` sampling 10^9 linear extensions, as poset complexity increases

Poset	2^2	2^3	2^4	2^5	2^6	2^7	2^8	2^9	2^{10}
Minutes	2	2	3	5	6	10	21	75	183

$2, 3, \ldots, 10$ two-element chains. The posets are labeled $2^2, 2^3, \ldots, 2^{10}$. In each case, we have extracted 10^9 linear extensions, with a fixed threshold composed of a single poset element. Before illustrating the results, it must be considered that `evaluation` computes various statistics on the input poset, some of which require the iterative execution of many `if-then` statements, in addition to the computational burden due to the core Bubley–Dyer algorithm. The computation time of these additional calculations depends critically, among other things, upon the cardinality and the structure of the poset, explaining the figures reported in Table 15.2. As can be seen, the computation time rapidly increases as the number of poset elements grows, reaching 183 min for poset 2^{10}, which is composed of 1,024 elements.

15.5 Planned Developments and Improvements

PARSEC is currently in its beta version. Some improvements are in order and will be implemented in the next release. They pertain to usability, to the addition of functionalities to handle large posets, and to the introduction of graphical capabilities.

Currently, PARSEC provides all the basic functions needed to implement the poset-based evaluation methodology introduced in the paper. However, to perform concrete computations some scripting is still needed by users, who must combine together different functions to obtain the required outputs. Although this allows for great flexibility, it may cause some problems to non-programmers. For this reason, some "macro" functions are being implemented to obtain the desired results in a single call, similarly to the `count` function. The aim is to reduce to a minimum the need for coding; similarly, a graphical user interface will be considered (see, for example, the R package Rattle Williams 2009), to assist non-expert users.

PARSEC represents poset by means of matrices and employs simple matrix computations (Patil and Taillie 2004) to address poset analysis. This makes the package quite easy to develop and to maintain and sufficiently effective for most real applications, but also makes PARSEC scarcely scalable. This could be a problem, when handling posets with several hundreds or thousands of elements. In this case, more sophisticated programming techniques will be needed, to effectively manage large sparse matrices. However, the main issue pertains to the way the evaluation function is computed. At present, the computation of the evaluation function is implemented by sampling linear extensions through the Bubley–Dyer algorithm. Although this is the fastest algorithm currently available, the number of linear extensions to be sampled and the computation time increase steeply with the complexity of the poset and in practice huge posets cannot be handled this way. In socio-economic applications, one may work with posets comprising many hundreds of elements, since statistical units are often assessed against ordinal variables coded in up to 10°. Evaluation function computation in this kind of partial orders is better addressed by using analytical formulas which provide approximations to mutual ranking probabilities. Different approximation formulas can be found in literature that can be used for our purposes, see, for example, De Loof et al. [2008], De Loof [2010]. Actually, available formulas are designed to approximate mutual ranking probabilities of two elements at a time, while the evaluation methodology requires computing the mutual ranking probability of an element with respect to an antichain. Thus, some adaptation is required before implementation.

Visualizing data is one of the most effective ways to ease user experience. In the near future, a set of functionalities will be implemented to give standard graphical representations to PARSEC outputs and to draw Hasse diagrams, projecting on their nodes various kinds of information, such as the value of the evaluation function and the corresponding number of statistical units, or inserting pictorial representations of the profiles [e.g., in the spirit of graphical representations available in the Kohonen package (Werhens and Buydens 2007)].

15.6 Conclusion

It is progressively clear to scholars and to decision-makers that addressing socio-economic issues in modern societies, and evaluation issues in particular, requires a change of paradigm. It is no longer the time to "aggregate and average," to produce macro-indicators that would fail to represent real societies and their structural dynamics. Instead, it is the time to represent and make explicit "shapes" and "patterns," "similarities" and "dissimilarities," "structural affinities" and "structural differences," and "nuances" and "complexities." Partial order theory surely plays an important role in this challenge and PARSEC can spread its use across the community of socio-economic scholars. PARSEC surely needs to be improved and extended in many directions, and we hope to get suggestions from users to fix possible bugs and to add new functionalities. In fact, it is our intention to transform PARSEC into an official and publicly available R package, publishing it on the CRAN web site. This task will be accomplished in the near future, after completing the test of the beta version. At present, PARSEC is freely available from the authors together with the technical documentation, for both Windows and Linux operating systems.

15.7 Appendix: Function List

Here, we list the functions currently available in PARSEC. The list is not for technical reference, but to give an idea of the scope and the capabilities of the package. The list is organized according to the four PARSEC sections.

15.7.1 Data Management

`var2prof`	Generates all possible profiles out of k ordinal variables. A vector of frequencies may also be passed, for subsequent use.
`pop2prof`	Reads a dataframe comprising statistical unit scores on k variables and extracts all unique profiles together with the corresponding frequencies.
`getzeta`	Generates the partial order matrix (i.e., the incidence matrix of the corresponding Hasse diagram) according to (1), from the profile list.
`popelem`	Associates each observed statistical unit with the index (i.e., the row or column of the partial order matrix) of the corresponding profile.

15.7.2 Basic Poset Analysis

`binary`	Checks whether a matrix is binary.
`reflexivity`	Checks whether a binary relation is reflexive.

`antisymmetry`	Checks whether a binary relation is antisymmetric.
`transitivity`	Checks whether a binary relation is transitive.
`is.preorder`	Checks whether a binary relation is a preorder.
`is.partialorder`	Checks whether a binary relation is a partial order.
`validate.partialorder`	Checks whether an input binary matrix defines a partial order and validates it as the incidence matrix of the corresponding Hasse diagram. If the input matrix does not represent a poset, the function returns which poset properties are not fulfilled.
`incidence2cover`	Builds a cover matrix from the partial order matrix.
`cover2incidence`	Builds a partial order matrix from the cover matrix.
`transitiveClosure`	Computes the transitive closure of a binary relation.
`upset`	Returns the upset of a set of elements.
`downset`	Returns the downset of a set of elements.
`is.upset`	Checks whether a set of elements of a poset is an upset.
`is.downset`	Checks whether a set of elements of a poset is a downset.
`gen.upset`	Returns the antichain generating the input upset.
`gen.downset`	Returns the antichain generating the input downset.
`incomp`	Returns the set of elements incomparable with a selected poset element.
`minimal`	Returns the minimal elements of a poset.
`maximal`	Returns the maximal elements of a poset.
`heights`	Returns the heights of the elements of the poset in the corresponding Hasse diagram.
`depth`	Returns the depths of the elements of the poset in the corresponding Hasse diagram.
`levels`	Returns the levels of the elements of the poset in the corresponding Hasse diagram.
`colevels`	Returns the colevels of the elements of the poset in the corresponding Hasse diagram.
`height.poset`	Returns the height of a poset.
`synopsis`	Gives a summary of the input poset features.

15.7.3 Poset-Based Evaluation

`lingen`	Returns a linear extension extracted by the input poset.
`linzeta`	Returns the (partial) order matrix of a linear extension, given the profile indexes of the original poset.
`evaluation`	From (1) the partial order matrix of a poset, (2) a linear extension of the poset, (3) the number of linear extensions to be sampled and (4) a threshold (as an antichain), the function returns (a) the estimated evaluation function, (b) the rank frequency distribution of each element of the poset, (c) the frequency distribution of the rank differences between each element of the poset and the higher ranked element of the threshold, and (d) the last linear extension sampled (that may be used to initialize other executions of the function).

15.7.4 OPHI Counting Approach

`count`	Given a population, the single variable cutoffs and the overall cutoff, the function implements the OPHI procedure and returns, among other results, (a) the indexes and the number of statistical units classified as poor, (b) the number of deprivations suffered by each statistical unit or by each profile, (c) the deprivation map (which observations or profiles are deprived on which dimensions), (d) the Head Count Ratio, (e) the Average Deprivation Share, and (f) the Adjusted Head Count Ratio.
`count2threshold`	Returns the profile threshold determining the poor profiles in the OPHI procedure.

References

Alkire S, Foster J (2011) Counting and multidimensional poverty measurement. J Public Econ 96(7–8):476–487

Alkire S, Foster J (2011) Understandings and misunderstandings of multidimensional poverty measurement. J Econ Inequal 9(2):289–314

Bubley R, Dyer M (1999) Faster random generation of linear extensions. Discrete Math 201: 81–88

De Loof K (2010) Efficient computation of rank probabilities in posets. Ph.D. dissertation. https://biblio.ugent.be/publication/874495

De Loof K, De Baets B, De Meyer H (2008) Properties of mutual rank probabilities in partially ordered sets. In: Owsinski JW, Brueggemann R (eds) Multicriteria ordering and ranking: partial orders, ambiguities and applied issues. Polish Academy of Sciences, Warsaw

Fattore M, Brueggemann R, Owsiński J (2011) Using poset theory to compare fuzzy multidimensional material deprivation across regions. In: Ingrassia S, Rocci R, Vichi M (eds) New perspectives in statistical modeling and data analysis. Springer, Berlin

Fattore M, Maggino F, Greselin F (2011) Socio-economic evaluation with ordinal variables: integrating counting and poset approaches. Stat Appl, Special Issue, 31–42

Fattore M, Maggino F, Colombo E (2012) From composite indicators to partial order: evaluating socio-economic phenomena through ordinal data. In: Maggino F, Nuvolati G (eds) Quality of life in Italy: research and reflections. Social indicators research series, vol. 48. Springer, Berlin

Patil GP, Taillie C (2004) Multiple indicators, partially ordered sets, and linear extensions: multicriterion ranking and prioritization. Environ Ecol Stat 11:199–228

R Core Team (2012) R: a language and environment for statistical computing. R Foundation for Statistical Computing, Vienna. ISBN 3-900051-07-0. http://www.R-project.org/

Ravaillon M (2011) On multidimensional indices of poverty. J Econ Inequal 9(2):235–248

Werhens R, Buydens LMC (2007) Self- and super-organizing maps in R: the kohonen package. J Stat Softw 21(5)

Williams GJ (2009) Rattle: a data mining GUI for R. R J 1/2:45–55