Overview of the Kappalab R package

`http://www.polytech.univ-nantes.fr/kappalab`

February 23, 2006

The aim of this document is to provide an overview of the Kappalab R package and some help and references for first time users. Other very useful documents are : the Kappalab manual (`kappalab.manual.pdf`) describing in details the functions implemented in Kappalab, and the Kappalab class hierarchy (`kappalab.classes.png`) describing Kappalab's class hierarchy. The corresponding files are available form the Kappalab web page.

1 Description

Kappalab, which stands for "laboratory for capacities", is an S4 tool box for capacity (or non-additive measure, fuzzy measure) and integral manipulation on a finite setting. It contains routines for handling various types of set functions such as games or capacities. It can be used to compute several non-additive integrals: the Choquet integral, the Sugeno integral, and the symmetric and asymmetric Choquet integrals. An analysis of capacities in terms of decision behavior can be performed through the computation of various indices such as the Shapley value, the interaction index, the orness degree, etc. The well-known Möbius transform, as well as other equivalent representations of set functions can also be computed. Kappalab further contains seven capacity identification routines: three least squares based approaches, a method based on linear programming, a maximum entropy like method based on variance minimization, a minimum distance approach and an unsupervised approach grounded on parametric entropies. The functions contained in Kappalab can for instance be used in the framework of multicriteria decision making or cooperative game theory.

2 Prerequisites

Before being able to use Kappalab, it is necessary to have a basic knowledge of the R language and of the different concepts arising from the use of non-additive measures and integrals in the framework of multicriteria decision making and cooperative game theory.

A good reference to start learning R is *An Introduction to R* by W. N. Venables, D. M. Smith and the R Development Core Team, and is available from the R project site (`http://www.r-project.org`). As far as non-additive measures and integrals are concerned, a good starting point is *Fuzzy Measures and Integrals : Theory and Applications. Physica Verlag, 2000* by M. Grabisch, T. Murofushi and M. Sugeno, Eds. One may also download the numerous articles available from the Kappalab web page.

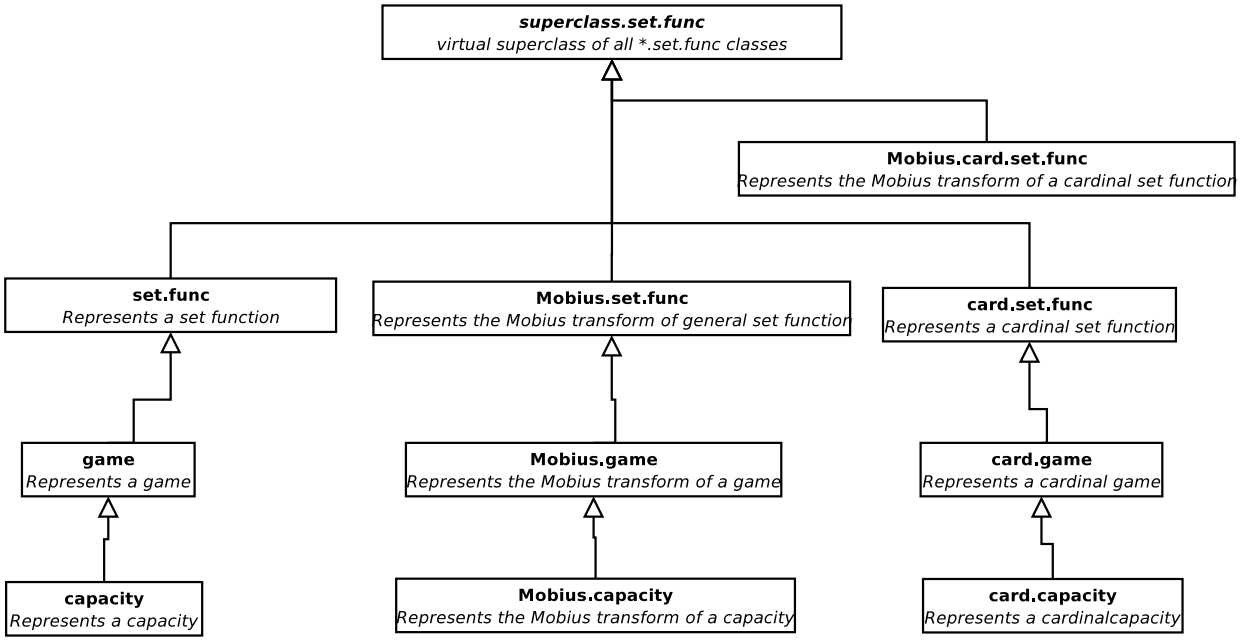

Figure 1: Simplified Kappalab's class hierarchy.

3 Structure of the Kappalab package

3.1 From a applicative perspective

Kappalab provides a set of high-level functions for non-additive measure and integral manipulation on a finite setting. These functions are written in C and using the S4 specification of the S language. From an applicative point of view, these functions can be mainly grouped into two classes : functions for non-additive measure *manipulation* and functions for non-additive measure (capacity) *identification*. For instance, in the framework of a multicriteria decision making problem, a natural way of using Kappalab would be first to use a function from the latter group in order to identify a capacity and the functions from the former group to analyze the capacity and the related integral in terms of decision behavior.

The high-level capacity identification functions in Kappalab are : `entropy.capa.ident`, `least.squares.capa.ident`, `heuristic.ls.capa.ident`, `ls.sorting.capa.ident`, `ls.ranking.capa.ident`, `mini.dist.capa.ident`, and `mini.var.capa.ident`. More details and examples can be found in the document `kappalab.manual.pdf` available from the Kappalab web site. All the remaining functions mentioned in Kappalab's manual are intended for non-additive measure manipulation.

3.2 From an object-oriented point of view

The Kappalab package is written using the S4 specification of the S language which enables object-oriented programming. For instance, objects modeling *set functions*, *games*, and *capacities* can be created and manipulated in Kappalab. The corresponding classes are named `set.func`, `game` and `capacity` respectively. Recall that a game is simply a set-function vanishing at the empty set and that a capacity is a monotone game. This *inheritance relationship* is represented by the left branch of the simplified class hierarchy given in Figure 1.

The other classes represented in Figure 1 are :

- `Mobius.set.func`, `Mobius.game`, and `Mobius.capacity` : objects of these classes are meant to represent the Möbius transforms of already created objects of class `set.func`, `game`, and `capacity` respectively.
- `card.set.func`, `card.game`, and `card.capacity` : objects of these classes are meant to represent cardinal set functions, cardinal games, and cardinal capacities respectively.
- `Mobius.card.set.func` : objects of this class are meant to represent the Möbius transforms of already created objects of class `card.set.func`, `card.game`, and `card.capacity`.

A detailed description of these classes can be found in the document `kappalab.manual.pdf`.

In object-oriented programming, the type of the treatment that can be applied to an object depends on its class. For instance, one can compute the Shapley value of an object of class `set.func`, whether it is only a `set.func`, a `game` or a `capacity`. However, the Choquet integral can be computed with respect to a set function if the object representing it is at least a `game`. More details about the treatments that can be applied to objects in Kappalab can be found in the document `kappalab.classes.png`.

4 First steps

Once you are familiar with the basics of R, we suggest that you browse through the document `kappalab.manual.pdf` and run the examples given in the help of the functions. Note that this manual is directly available in R either in HTML or text format through the help command.

A first simple example :

- to start up R under Linux, type `R` in a terminal; under Windows, click on the appropriate icon,
- type `library(kappalab)` in the R terminal to load the library,
- type `html.start()` in the R terminal to start up the HTML documentation,
- in the browser, click on "Packages" and select "kappalab",
- click on "Choquet.integral-methods" and copy and paste the examples in the R terminal.

Other interesting examples can be found in the example sections of the classes `set.func`, `game`, `capacity`, `Mobius.set.func`, `Mobius.game`, `Mobius.capacity`, etc in the Kappalab manual.

5 A few words about the coding of set functions in Kappalab

We end this document by some information on the way objects of classes `set.func`, `game`, `capacity`, `Mobius.set.func`, `Mobius.game`, and `Mobius.capacity`, representing set functions and Möbius transforms, are coded in Kappalab. As can be seen from the document `kappalab.classes.png`, all such classes have an attribute `n`, which is meant to contain the cardinal of the set N on which the set function is defined, and an attribute `data`, which is a `vector` ("array") of size 2^n meant to contain the coefficients of the set function. Depending on the type of set function, the coefficients in `data` are given either with respect to the *binary* order or the *natural* order on 2^N.

By binary order, we refer to the following total ordering of the elements of 2^N :

$$\emptyset, \{1\}, \{2\}, \{1,2\}, \{3\}, \{1,3\}, \{2,3\}, \{1,2,3\}, \{4\}, \{1,4\}, \ldots, N$$

This order is obtained as follows. Consider the natural sequence of integers from 0 to $2^n - 1$, that is $0, 1, 2, \ldots, i, \ldots, 2^n - 1$, and its binary notation $[0]_2$, $[1]_2$, ..., $[i]_2$, ..., $[2n-1]_2$, which is (with n digits)

$$00\cdots000, 00\cdots001, 00\cdots010, \ldots, 11\cdots111.$$

To any number $[i]_2$ in binary notation corresponds a unique subset $I \subseteq N$ such that $j \in I$ if and only if the $(n+1-j)$-th digit in $[i]_2$ is 1.

By natural order, we refer to the following intuitive total ordering of the elements of 2^N :

$$\emptyset, \{1\}, \{2\}, \{3\}, \ldots, \{1,2\}, \{1,3\}, \{1,4\}, \ldots, \{1,2,3\}, \{1,2,4\}, \ldots, N.$$

In other terms, first is given the empty set, then the n singletons, then the $\binom{n}{2}$ pairs, then the $\binom{n}{3}$ 3-element subsets, ..., and finally N itself.

For objects of classes `set.func`, `game`, and `capacity`, the elements of the `data` attribute are stored with respect to the binary order, whereas for objects of classes `Mobius.set.func`, `Mobius.game`, and `Mobius.capacity`, they are stored with respect to the natural order.