Zelig Naming Conventions

Matt Owen

September 29, 2010

1 Mandatory Naming Conventions

When developing a Zelig package, it is important to write code in a style that is consistent with existing code and clearly understandable. The following specifies guidelines that are suggested for the any Zelig package, and strictly enforced for commits to Zelig's main trunk.

1.1 Function and Method Naming

The Zelig developer writes Functions and Class Methods as verbs in camel-case verbs with leading capital letters. That is, a function name must:

1. `be a verb`
2. `have a leading capital letter`
3. `contain only alphanumeric characters`
4. `use a capital letter to denote the beginning of a new word within the function name`
5. `treat all abbreviations as being lowercase`

For example, if the developer is writing a function that converts results to HTML, then the function should be named: `ResultsToHtml`.

1.2 Generic Function Naming

Generic functions, by Zelig's convention, are preferably single-word lowercase verbs. If it is impossible to phrase the generic function as a single-word, then it should be named as a verb in camel case with a leading lowercase letter. It is highly recommended that careful consideration be taken when naming generic functions. That is, a generic function name must:

1. `be a verb`
2. `have a leading lowercase letter`
3. `contain only alphanumeric characters`
4. `use a capital letter to denote the presence of a space between words`

5. `treat all abbreviations as being lowercase`

For example, if the developer is writing a function that stores class information in a log-file for a variety of datatypes, then the function be named: `store` or `writeToLog`.

1.3 Class Naming

Class names are preferably single-word, alphanumeric, and camel-case nouns. This convention may be loosely followed. The name of the constructor function and the class name must be identical. That is, a class name must:

1. `be a noun`
2. `contain only alphanumeric characters`
3. `match their function/constructors name`

For example, if the developer writes a class that represents a polygon, then it should be named: `polygon` or `Polygon`. The constructor function should be named to match this. Furthermore, classes should be kept in their own R file which is similarly named.

1.4 Variables

Variables follow very different conventions that functions, methods, classes, and generics. Few rules govern their naming, except that they are must be all lowercase and descriptive. That is, a variable name must:

- `contain only lowercase letters and dots`
- `be descriptive`
- `be longer than three letters long, unless it is an iterator`

For example,

1.5 Private Functions and Variables

Functions and variables that are intended to be hidden from the user follow all the rules of their visible counterparts, except they must begin with a leading dot. This is so that the export command will ignore them.

1.6 Operator Overloading

While in many circumstances operator overloading is a useful tool, it is highly discouraged in Zelig packages. Please store all overloaded operators in a file named `ZELIG_zzz_`*`model_class`*`_operators.R`, where *model_class* is the model's class. Note that, because operators do not begin with an alphabet character, they will be ignored by the standard construction of a `NAMESPACE` file. As a result, operators must be explicitly exported in the `NAMESPACE` file.

2 Suggested Conventions

2.1 S3 Object Orientation

While S4 objects offer a stricter style of object-oriented programming, it is still standard to write functions as S3 objects due to better support and consistency with core packages (all of which as S3 objects). Zelig can interface with S4 objects, however the results at times are unpredictable, and might only be suitable for truly savvy R developers.

2.2 Package Naming

It is suggested that packages depending on the Zelig library have ".zelig" appended to them. This will make clear that the syntax and structure of the package is compatible with that of other Zelig objects.

3 Synopsis

function verb, camel-case, leading capital letter

method *same as function*

generic function verb, alphanumeric, lowercase leading letter, preferably single word

variable noun with lowercase letters, numbers, and dots only

class object alphanumeric characters only. name must match constructor name and filename

hidden variable or function typical naming conventions but with leading dot

overloaded operators stored in `ZELIG_zzz_`*`model_name`*`_operators.R`