Structure of a Zelig Model

Matt Owen

October 1, 2010

1 Introduction to Zelig

Zelig is a software package aimed at delivery and easy-to-use API for developing new statistical models, as well as, providing a framework to unify common ideas between such models. That is, Zelig offers methods for end-users to interact with a large variety of statistical models using identical syntax. This allows the focus of an arbitrary statistical model to be on the *quantities of interest*, rather than the decoding of mathematical objects.

2 Necessary Components of a Zelig Model

The following generic functions must be defined for a Zelig model to operate as expected.

2.1 The Original Fitted Model

Before a Zelig package can be written, a clear idea (and preferably an existing function) of the statistical model must be exist. That is, before Zelig can begin simulating *quantities of interest*, an existing statistical model must exist. Typical candidates for these models include logit, probit, normal, Gaussian, et cetera. In fact, given any statistical model, it is possible to create a Zelig object that will simulate the desired *quantities of interest* for a given data-set.

2.2 The `zelig2` Function

The `zelig2` function acts as the interface between the existing model and the Zelig object. Essentially, `zelig2` acts as the glue that binds the *zelig* object with that of the existing statistical model. *See the* `zelig2` *documentation for specific details.*

2.3 The `param` Function

The `param` function produces random samples from the statistical model created as a result from the `zelig2` function. That is, the `param` function takes random values from the distribution defined by the given, fitted statistical model. In addition to this, the developer may define explicitly the ancillary parameters and systematic components of the distribution. For a complete how-to of developing the `param` function, *see the param and parameters documents.*

2.4 The `qi` Function

The `qi` function simply simulates the *quantities of interest*. This function, in effect, is the heart and soul of any Zelig model. That is, the ability to simulate *quantities of interest* is a unifying concept between arbitrary statistical models. By coding the `qi` function, the developer can build statistical models that can be discussed and wielded in exactly the same fashion as any other Zelig model. For information on coding a `qi` function, *see the qi document*.

3 Optional Components of a Zelig Model

3.1 The `describe` Function

The `describe` function serves two purposes:

1. Generate the correct citation for the model
2. Specify the category of data-sets that work with the particular Zelig model

It is highly suggested that all developers, in the least, use the `describe` function to hardcode their authorship into the Zelig model. For more information on the `describe` function, *see the describe document*.

4 Advanced Zelig Model Components

In addition to the standard functions that are intended - and often necessary - for the Zelig developer to write, the advanced developer can do some true Zelig hacking.

4.1 The `setx` Function

The `setx` function analyzes various properties of the data-sets the end-user is working with. Typically this function is simply used to for specifying setting counterfactuals, as well as, summarizing useful data that will be used within the `qi` function. It is highly suggested that file remain untouched even by the most proficient Zelig developer.

4.2 The `sim` Function

The `sim` function prepares both the `zelig` and `setx` objects for use with the `param` and `qi` functions. If a model requires extensive hacking, it may be simpler to rewrite the entirety of the `sim` function, rather than trying to phrase the model in terms of `param` and `qi`. This approach should only be used as a last resort.

4.3 The `plot` Function

The `plot` function is a generic function that exists for almost every statistical model. Zelig will automatically be able to play your *quantities of interest* in most situation, however, if a different plot is desired, the developer simply needs to write the function named `plot.model_name.sim`, where "model_name" is the name of the model.