A quick walkthrough of the `spectralGOF` package

Jesse Shore and Benjamin Lubin

December 29, 2015

Introduction

This document provides a minimal introduction and tutorial for the `spectralGOF` package. Spectral goodness of fit is described in Shore and Lubin (2015)[1]

Downloading and installing `spectralGOF`

For now, `spectralGOF` is not on CRAN. To work with spectralGOF, you will first have to download the package source files to your computer. Please follow these steps to get started.

1. Download `spectralGOF_1.0.tar.gz` to your computer and keep track of where you save this file. The current version is available on Github at XXXXXXXXXXXXXXXXX.
2. Run the following line in R:

   ```
   install.packages(path_to_file, repos = NULL, type="source")
   ```

   where "`path_to_file`" is the location of the file on your computer. For example, on my computer, I replace "`path_to_file`" with `"C:/code/sgof/spectralGOF/spectralGOF_1.0.tar.gz"`.
3. Now load the package with

   ```
   require(spectralGOF)

   ## Loading required package:  spectralGOF
   ```

Now you should be ready to use the package.

[1] Jesse Shore and Benjamin Lubin (2015) "Spectral Goodness of Fit for Network Models," *Social Networks* **43**, October, 2015, pp 16-27

Using `spectralGOF` with a single ERGM

`spectralGOF` works on any fitted model that can be used to generate simulated networks. For simplicity, however, let's start with ERGMs. We will examine the "Faux Mesa High" data set that is included in the `sna` package. First, we will load the necessary packages and get set up.

```
suppressMessages(require(ergm))
suppressMessages(require(sna))
data("faux.mesa.high.withnumericattributes")
FMH <- faux.mesa.high #for convenience, set short names
n <- FMH$gal$n        #
```

We will begin by fitting a simple model, and then go on to measure its SGOF.

```
fit1<-ergm(FMH~edges+nodematch('Grade',diff=TRUE)+
                     nodematch('Sex',diff=TRUE)+
                     nodematch('Race',diff=TRUE,keep=c(1,3,4)))
```

Spectral goodness of fit (SGOF) is calculated by comparing simulated networks to the observed networks. Practically speaking, this means that we will need to decide how many simulated networks to work with. For published work, at least 1000 simulations should be used to calculate SGOF. Running 1000 simulations takes time, however, so we recommend using fewer simulations (for example, 100) to calculate the SGOF for preliminary or exploratory models.

```
#set the number of simulations to use
#when calculating SGOF:

nsim <- 100
```

Having fit the model, we can now assess its goodness of fit with the `SGOF` function. `SGOF` is the main function in the package. A simple way to use it is to pass a fitted ERGM as its first argument.

```
sgof1<-SGOF(fit1, nsim=nsim)

## [1] "fitting null model"
## [1] "simulating from null model"
## [1] "simulating from fitted ergm"
## [1] "0.216 (5th, 95th quantiles:0.03, 0.46)"
```

The `SGOF` function prints out the mean SGOF, as well as the 5th and 95th quantiles of SGOF. It also produces an `sgofobject`, which in this case is "`sgof1`."

Using `spectralGOF` with a multiple ERGMs

Usually, one works with multiple network models. If we simply follow the procedure above, we will fit and simulate from a null model in the process of finding the SGOF for each fitted model. This is not necessary, however, as we are just drawing from the same null distribution over and over again. It can save considerable time to draw one set of simulated networks from the null distribution and then pass them to the `SGOF` function as an argument. When `SGOF` receives a set of networks to use as the null distribution, it will skip the step of simulating them afresh.

To obtain a null distribution for the SGOF, we can use the `SGOFnull` function. It requires the observed network in the form of a square adjacency matrix as its first argument. Because `FMH` is a `network` object (see documentation for `sna` and `ergm` packages), we can get the network adjacency matrix with `FMH[,]`.

```
fmhNull<-SGOFnull(FMH[,], nsim=nsim)

## [1] "simulating from null model"
```

The code above produces a `SGOFnull` object that includes the spectra from the simulations from the null model, as well information about the mean and 5th and 95th percentiles of the null SGOF. Usually, we will want to work directly with the spectra.

```
fmhNullSpectra<-fmhNull$allNullSpectra
```

Once we have these, we can pass them directly to the SGOF function to avoid re-creating them for each model.

```
#fitting a second model that omits race
fit2<-ergm(FMH~edges+nodematch('Grade',diff=TRUE)+
             nodematch('Sex',diff=TRUE))

#passing fmhNullSpectra to SGOF
sgof1 <- SGOF(fit1, nsim = nsim, nullDistribution = fmhNullSpectra)

## [1] "simulating from fitted ergm"
## [1] "0.2 (5th, 95th quantiles:0, 0.41)"

sgof2 <- SGOF(fit2, nsim = nsim, nullDistribution = fmhNullSpectra)

## [1] "simulating from fitted ergm"
## [1] "0.14 (5th, 95th quantiles:-0.07, 0.36)"
```

Functions providing further information

Several basic functions are provided that take an `sgofobject` as an argument and provide further information and functionality to the user. Most basically, **`summarySGOF`** prints out the basic information, plus the standard error of the mean as well as the 5th and 95th percentiles of SGOF for the null model.

```
summarySGOF(sgof1)

## ****************************************
## Spectral goodness of fit (SGOF) analysis
## ****************************************
##
## mean SGOF of fitted model: 0.2
##      5th percentile: 0.005
##      95th percentile: 0.409
##
## standard error of the mean: 0.0135
##
## Null model SGOF percentiles:
##      5th percentile: -0.162
##      95th percentile: 0.213
##
##
## ****************************************
```

A small standard error of the mean provides evidence that `nsim` is sufficiently large get a reliable estimate of the mean SGOF. If it is larger than 0.01, you should increase `nsim`, at least for work intended for publication.

The 5th and 95th percentile SGOF for the fitted model should *not* be interpreted as a confidence interval of the mean SGOF. Rather, they are percentiles of the whole distribution of SGOFs that are obtained from the whole distribution of `nsim` simulated networks. The range between the 5th and 95th percentiles of SGOF reflects the fact that the simulation process for generating networks from a fitted ERGM is stochastic in nature. Some simulated networks will be more similar to the observed network than others, which means that some will have a higher SGOF than others. The mean SGOF is the natural summary statistic for the distribution of SGOF, but these percentiles can be informative as well.

In addition to the percentiles of SGOF for the fitted model, we also have percentiles of SGOF for the null model. The null model percentiles give a sense of how close to a simple random graph the structure of the observed network is. The larger the 95th percentile SGOF of the null model, the more similar the initial network is to an Erdős-Rényi (AKA Bernoulli) random graph.

Visualizing spectral errors

The distance between an eigenvalue of the observed network and the corresponding eigenvalue of the spectrum from a simulated network drawn from a model of the observed network is a spectral “error.” There are errors between the observed and null spectra when the observed network has non-random[2] structure. By way of analogy to the typical use of R^2, the errors between the observed and null spectra constitute the structural “variation” to be explained by a fitted network model.

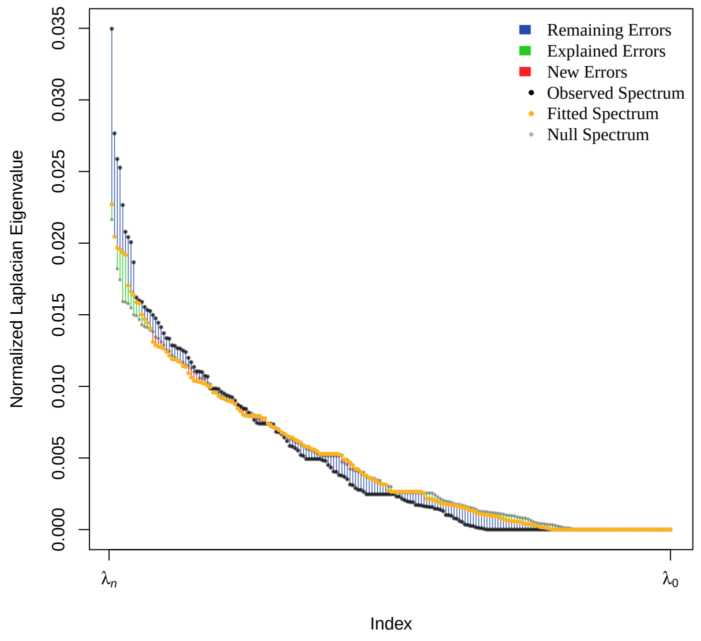
When everything is going perfectly, the spectrum of a fitted model is closer to the observed spectrum than the null spectrum is. In this case, we can say that the fitted model has “explained” a certain amount of error and left a certain amount of error remaining. However, things don’t always go perfectly. Network models can also introduce new error (the spectra from the simulations drawn from the fitted model can be further away from the observed network’s spectrum than those drawn from the null model); this can occur when the simulated networks drawn from the fitted model contain structure not present in the observed network.

To get a better sense of what is going on, beyond the mean SGOF summary statistic, it can be very helpful to visualize these spectral errors. The function `plotSGOFerrors` is provided for this purpose.

`plotSGOFerrors` takes an `sgofobject` as its first argument. If we set the `style` argument to `"original"`, the function plots the observed spectrum, as well as the spectra from the null and fitted models.

```r
plotSGOFerrors(sgof1, style="original", ptcex = 1.5)
```

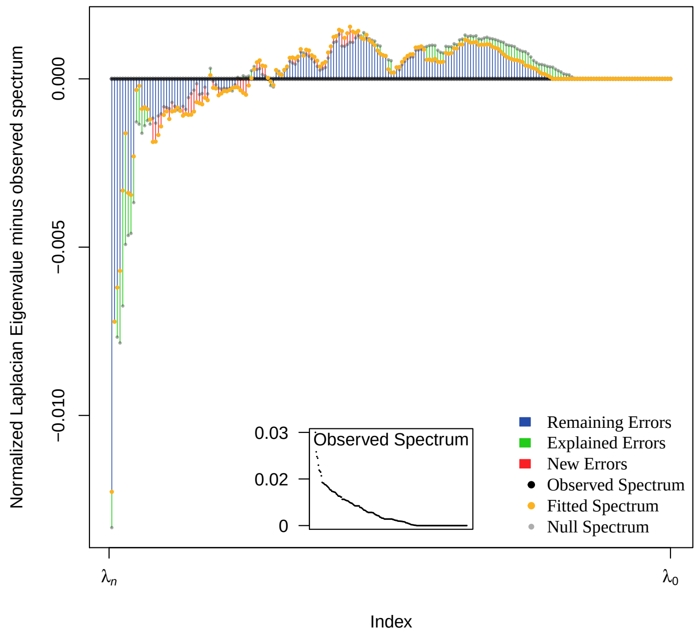
[2]in this case “random” means uniformly random locations of edges, as in the Erdős-Rényi random graph



The `"original"` style allows direct evaluation of the shape of the spectra, which can be useful as one becomes more familiar with the characteristic shapes associated with different network structures. However, it may not be easy to see how much error is explained in this style.

It is often easier to get a clear view of how much error is explained (or introduced) by the fitted model with the `"flat"` style. In the `"flat"` style, the observed spectrum is subtracted from the plotted spectra, which allows more full use of the plotting area.

```
plotSGOFerrors(sgof1, style="flat", ptcex = 1.5)
```



Calculating SGOF for other models

Because of the prevalence of ERGMs, we constructed the **SGOF** function to work conveniently with the output of the **ergm()** function. When we passed a fitted **ergm** as the first argument to **SGOF**, above, we were actually passing it to the **fittedERGM** argument. In other words, we could have typed the following:

```
SGOF(fittedERGM =  sgof1)
```

It is straightforward to use **SGOF** with other models as well by using other function arguments. If we are not passing a **fittedERGM** argument, we must provide values for the **networkModel** and **observedNetwork** arguments instead. However you arrived at your fitted model, at the purpose of the **networkModel** argument is to produce simulated networks drawn from that fitted model. The **networkModel** must therefore be a function that produces adjacency matrices of simulated networks with the same number of nodes as the observed network.

For example, if we wanted to assess the fit of the Watts-Strogatz small world model[3], we could first define a function that generates simulated networks (in

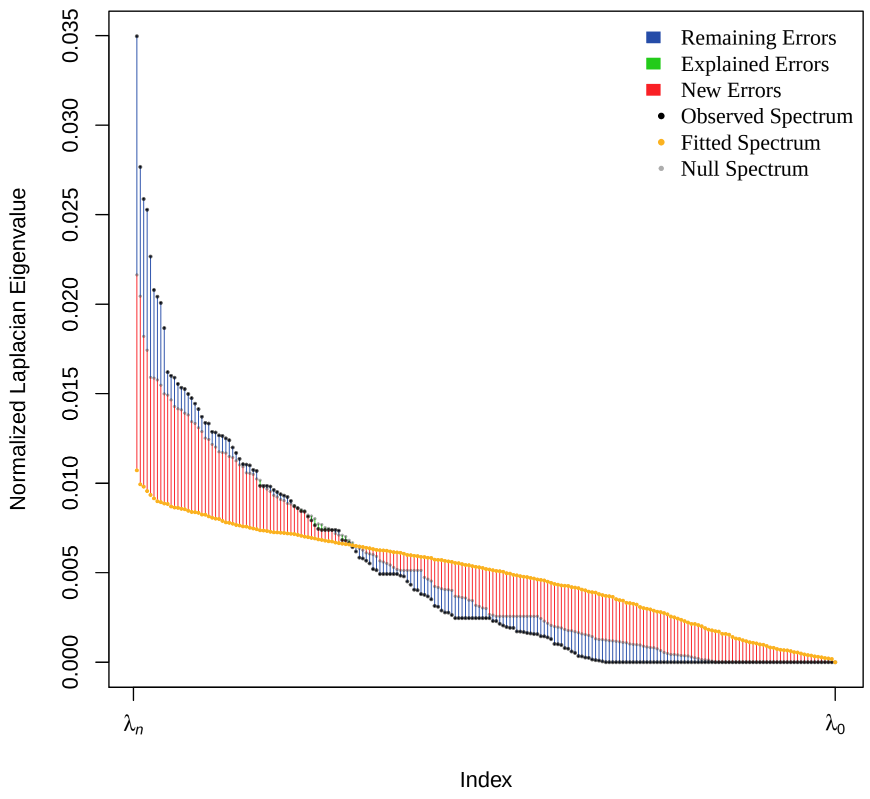
[3]This is a highly stylized model, and not intended as a realistic model of observed networks. We include it simply for demonstration.

the form of adjacency matrices). The `igraph` package includes a useful function for creating such networks, and we will use it here.

```
suppressMessages(require(igraph))

#define a function to create simulated networks
#drawn from the fitted model
SWModel<-function(){
  as.matrix(watts.strogatz.game(1,n,nei=2, p=.1)[,])
}

#pass this function to SGOF along with the observed network
SWSGOF<-SGOF(networkModel = SWModel, observedNetwork = FMH[,],
             nsim = nsim, nullDistribution = fmhNullSpectra)

## [1] "simulating from network model"
## [1] "-1.369 (5th, 95th quantiles:-1.42, -1.32)"
```

Clearly, this model fits very poorly! A look at the spectra reveals that a simple random graph (ie, the null model for `SGOF`) is a better model than the stylized Watts-Strogatz small world model.

```
plotSGOFerrors(SWSGOF, style="original")
```



Remaining Errors
Explained Errors
New Errors
Observed Spectrum
Fitted Spectrum
Null Spectrum
Normalized Laplacian Eigenvalue
0.000
0.005
0.010
0.015
0.020
0.025
0.030
0.035
λ_n
λ_0
Index