bdots

Abstract

The Bootstrapped Differences of Timeseries (bdots) was first introduced by Oleson (and others) as a method for controlling type I error in a composite of serially correlated tests of differences between two time series curves in the context of eye tracking data. This methodology was originally implemented in R by Seedorff 2018. Here, we revist that implementation, both improving the underlying theoretical components and creating a more robust implementation (that word twice) in R.

1 Introduction

Idk, introduction stuff

This paper is not intended to serve as a complete use guide to updates in the bdots package. Rather, the purpose is to showcase major changes and improvements to the package, with those seeking a more comprehensive treatment directed to the package vignettes.

Updates to the bdots package have been such that there is little resemblance to the original. Rather than taking a "compare and contrast" approach, we will first enumerate the major changes, followed by a general demonstration of the package use.

1. User defined curves
2. Permit fitting for arbitrary number of groups
3. Updates to bootstrapping algorithm and introduction of permutation test
4. Automatic detection of paired tests based on subject identifier
5. Allows for non-homogenous sampling of data across subjects and groups
6. Introduce formula syntax for bootstrapping difference function
7. bdots object inherits from data.table class
8. bdots is now stylized "bdots"

Bootstrapped differences in time series The high level motivation, abstracted from the particulars, is more or less as follows: we are often interested in comparing time series between two or more groups. A full(er) review of previous methods can be found in Seedorff, though we can limit the scope of interest to specifying that we are interested in "developing a statistical tool to (1) detect differences in two time series (as the VWP) and (2) to offer a precise characterization of the time window in which a difference occurs [**?**].

A typical instantiation of this problem occurs when we have two groups (or experimental conditions, etc.,) in which subjects in each group have an associated time series. It's assumed that each group has some distribution of associated functions in time, and we are interested in identifying windows in time in which these distributions are significantly different.

There are also situations in which we are interested in the difference of differences. I don't remember where or why this is done, some justification for doing this instead of some complicated F test, but I can find that later. Instead, let me offer an example. I'm already not going to like this example, so let me just put it anyways knowing that it will be deleted. Suppose we are interested in understanding how the color of a vehicle differentially impacts performance based on the vehicle type. We know that there is some difference between cars and trucks. Suppose then that we look at the difference between red cars and red trucks and then the difference between blue cars and blue trucks. If color does not mediate this difference, the difference between red cars and trucks should be the same as the difference between blue cars and trucks. This information can be determined if we look then at the difference between the differences.

The original bdots package was predicated on comparing differences between dense, highly correlated time series by first specifying functional forms and then performing statistical tests on each of the observed time points. With verison 2.0, this is no longer the case, and bdots is able to fit parameteric functions to any type of data observed in time. Along with methodological improvements, this expanded whatever has also replaced the original method of determining statistical significance in differences (autocorrelated adjustment to α) with a more robust permutation testing framework. In addition to methodology, a number of quality-of-life improvements have also been made, greatly simplifying syntax, creating more robust functions, and the addition of a number of useful methods for handling returned objects.

talk about how we are no longer just doing time series, as indicated with saccade method. we have now branched to the more general problem of comparing functional forms of observed data.

2 Methodology and Overview

There are major changes in the underlying methodology used in the bdots package, and we will briefly review the current methodology here (without explicit comparisons to the original). For those interested, please see some other article that I have to write.

Broadly, there are two steps to performing an analysis with the bdots pacakge: fitting the curves to observed data and bootstrapping differences between groups. The first step involves specifying an underlying curve, f, which may or may not be parametric. Along with the observed data y for each ith subject, bdots, via fitting with gnls, returns a set of parameters along with an estimate of their covariance.

$$F : f \times y_i \to N(\hat{\theta}_i, V_i), \tag{1}$$

where θ is a length-p vector representing the parameters of the function.

Once fits have been made, we are ready for testing the bootstrapped difference between curves. Once the groups of interest have been specified, two algorithms are implemented: a bootstrapping algorithm is used to determine the distribution of each group of curves, while permutation testing is used to specify regions of statistically significant differences. The algorithm for bootstrapping for each group is as follows:

1. For a group of size n, select n subjects from the group, *with replacement*
2. For each selected subject, draw a set of parameters from the distribution $\theta_i^* \sim N(\hat{\theta}_i, V_i)$. This permits us to account for within subject variability
3. For each of the resampled θ_i^*, find the bth bootstrap estimate for the group $\theta_b = \frac{1}{n}\sum_{i=1}^n \theta_i^*$
4. Perform this sequence B times

The end results is a $B \times p$ matrix containing a bootstrapped sample of the group distribution for θ. Each row of this matrix is used to create a $1 \times T$ vector representing f_θ evaluted at T time points. This results in a $B \times T$ matrix representing a collection of bootstrapped curves evaluated at each time point, in total representing a bootstrapped distribution of the curves.

Next we attend to idenfiying regions in which a statistically significant difference between curves is present. We begin by computing a t-statistic of the difference at each time point,

$$T(t) = \frac{|\overline{f}_1(t) - \overline{f}_2(t)|}{\sqrt{\frac{1}{n_2}\text{Var}(f_1(t)) + \frac{1}{n_2}\text{Var}(f_2(t))}}, \tag{2}$$

or, in the case of paired groups,

$$T(t) = \frac{\overline{f}_D(t)}{\sqrt{\frac{1}{n}\text{Var}(f_D(t))}}. \tag{3}$$

Next, we go about creating a null distribution against which to test our hypothesis that there is no difference between each group at each time point. We do this with permutation testing, and the algorithm is as follows: for two groups, with n_1 and n_2 subjects in each

1. Assign to each subject a label indicating group membership
2. Randomly shuffle the lables indicating membership, creating two new groups with n_1 and n_2 subjects in each
3. Recalculate the t-statistic, $T(t)$ and log the maximum

The collection of maximum values for $T(t)$ will serve as the null distribution against which to compare our observed $T(t)$.

This notation was clearly stolen from the FDA book.

Previous implementations of `bdots` performed this test by adjusting the nominal α to account for correlation between subsequent tests in time, based on previous work in Oleson. Simluations presented in [cite] suggest that this was not optimial. `bdots` does still maintain this functionality, and an estimate of the adjusted α, along with adjusted p-values, can still be determined with the function, `somefun`.

—

In addition to whatever general uncertainy exists in the section above, we again now need to consider if we want each iteration of our bootstrap to be $f(\frac{1}{n}\sum \theta_{bi})$ or $\frac{1}{n}\sum f_i(\theta_{bi})$

3 Example Data

Maybe here choose a dataset and explain what it is, specifying the components of it so that we can use it in an ongoing example. That way (especially in bootstrap formula step) I can just say something like oh yeah Group1 values A and B and Group2 values with C and D. Or whatever.

4 Fitting Curves

The curve fitting process is performed with the `bdotsFit` function, taking the following arguments: (removing ‘cor‘ and numRefits)

```
bdotsFit(data, subject, time, y, group, curveType, cores, ...)
```

Curve functions Each of `subject`, `time`, `y`, and `group` are length one character vectors representing columns of the dataset used in `data`. New here is `curveType`, taking as an argument an R call to a particular curve, for example the four parameter logistic, `logistic()`. This is done to self-contain any additional arguments associated with the fitting curve, for example the concavity of the double Gaussian (`curveType = doubleGauss(concave = TRUE)`) or the number of knots in a piecewise spline (`curveType = splines(knots = 5)`). A number of curves are included with the `bdots` package, including those for the four-parameter logistic, the double Gaussian, an exponential curve, and polynomials of arbitrary degree. A detailed vignette on writing your own curves can be found with `vignette("bdots")` ($\Leftarrow$ actually it would be vignette("customCurves", "bdots") or browseVignette("bdots") to see, but I haven't decided which I want because I don't really like the name customCurves).

Notably, `bdots` can now fit curves to an arbitrary number of groups, so long as all have the same parametric specification.

Return object and generics The function `bdotsFit` returns an object of class `bdotsObj`, inheriting from class `data.table`. As such, each row of this object (object object object i need a new word) uniquely identifies one permutation of subject group (meaning if a subject in two groups, they get two rows). Included in this row are the subject identifier, group classification, summary statistics regarding the curves, and a nested gnls object. Not sure if this is worth including, most people won't use it and i can put it in the vignette.

Several methods exist for this object, including `plot`, `summary`, and `coef`, returning a matrix of fitted coefficients returned from `gnls`. One consequence of inheriting from `data.table`, we are able to utilize data.table syntax. Note, for example, the differences between `coef(fit)`, `coef(fit[group == "A",])`, and `coef(fit[group == "B",])`. That's pretty much it on the neat stuff you can do with this. Time to go to bootstrapping step.

Actually, there is one additional thing here that I might include as an aside for now – for part 2 of disseration, we are fitting curve to saccades instead of an "observed" function. In this case, R^2 ends up being kind of a dumb/silly metric. Same can be said for auto-correlation (in terms of it being relevant). Both of these things are included in the derivation of the `fitCode`. Have not yet decided how that will be handled.

5 Bootstrapping

Like the fitting function, the bootstrapping process has been consolidated to a single function, `bdotsBoot`. In addition to taking as an argument an object returned from `bdotsFit`, `bdotsBoot` also takes a formula argument, with syntax unique to the `bdots` package.. When comparing the bootstrapped differences of curves, there are two distinct types of analyses we may be interested in performing. First is a simple difference between the time series of two separate groups.

$$\texttt{y} \sim \texttt{group(A, B)}$$

This next part is kind of a weird aside, and I'm not sure yet where I want to put it. At least in the context of the visual world paradigm, and likely others, we find ourselves in situations in which two distinct types of difference curves are of interest: the difference between two group curves (say, A and B), and the difference of the difference between four group curves (say, the difference between the difference between condition 1 and 2 within group A and the difference between condition 1 and 2 in group B).

As curves for an arbitrary number of groups may be fit at once with the `bdotsFit` function, a formula syntax has been introduced to specify the differences sought. (possibly a better way to introduce/write this section)

6 Extensions? Plots? I don't know!

Should I include an example case where I am also able to demonstrate difference of difference syntax? It's already included in the vignette

Maybe do tumr, using expCurve and unequal time points. does not allow demonstration of difference of difference, though. Good plots to show include

1. plots of fits
2. plots of bootstrap with ci
3. difference curve, also with sig sections highlighted
4. Correlation with fixed variables across time (`bdotsCorr`)

7 Discussion

I'm not really sure what to include in the discussion. We don't need to compare it to other approaches for analyzing this data, as that's aleady been done. I can point to full methodology paper to see improvements in CI coverage and difference detection/power. Reporting should be fairly simple, an α is given which is used to set the treshold for permutation tests – nothing else needs to be done. The previous bdots package suggested reporting quality of individual fits that made up the bootstrapped curves, though that was based on R^2 and AR(1) status. The former of these is problem specific, the latter now irrelevant.

raff raff raff raff raff

8 Conclusion

Improvements made to the `bdots` package have drastically improved the ease of use of the package and the scope of the types of problems it is able to address. The consolidation of major components into two functions has also streamlined use. Quality of life improvements include multiple group fitting, formula syntax for bootstraps, tractable return objects, and others. Generics have also been good. The package is also now statistically correct. It has extended the types of data that can be accomodated, including heterogenous observations across time and the ability to construct user-specified curves. it really is a pretty neat package.