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. Parameter and nonparametric functions
2. User defined curves
3. Permit fitting for arbitrary number of groups
4. Updates to bootstrapping algorithm and introduction of permutation test
5. Automatic detection of paired tests based on subject identifier
6. Allows for non-homogenous sampling of data across subjects and groups
7. Introduce formula for bootstrapping difference function

8. Removal of auto-correlation assumption

9. bdots object inherits from data.table class

10. bdots is now stylized "bdots"

Outline

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.

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

(Note: for non-parametric functions, the θ values are associated with splines).

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 to determine the distribution of each group of curves and to specify regions of statistically significant differences, respectively. The algorithm for the bootstrapping steps 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 result is a $B \times p$ matrix (where p is the number of parameters) $\Leftarrow$ this might not be true, I'll leave it for now, but we technically don't need to prespecify that each subject has same number of parameters (ok, but actually we kind of do if θ_b is going to be the mean of each sample. This is fine.

The end results is a $B \times p$ matrix (where p is the number of parameters) containing a bootstrapped sample of the group distribution for θ. Each of these is used to construct a $B \times T$ matrix (T the number of time points), a collection of bootstrapped curves. Each column of this matrix represents a time point, t, from which we can compute the mean and standard deviation of the group curve at that time (wordy). $\Leftarrow$ this also might contain too specific of detail. they don't care that its a matrix. they care for each time point we can construct an estimate of the mean and sd. And really, even this is only from CI and group distribution. There could possibly be an option to skip computing this at all and just do the permutation test.

Next we attend to idenfiying regions in which a statistically significant difference between curves is present. To this end, . . . (stuff) . . . we construct a t-statistic at each time point between the two groups

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

This notation was clearly stolen from the FDA book. However, what we intend to articulate by $\overline{f}_1(t)$ is $\overline{f}_j(t) = \frac{1}{n_j}\sum_{i=1}^{n_j} f(\theta_i)$. In the case of a paired t-test, the test statistic is simply

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

In either case, once the collection of test statistics has been collected at each time point, we go about determining the null distribution. For each of P iterations, we perform the following:

1. Shuffle the labels assigning group membership for each subject
2. Recompute the test statistic $T(t)$, retaining the max of each iteration

The original collection of statistics, $T(t)$ are compared? at each point against the distribution of values collected during the permutation. Those exceeding a specified threshold are determined significant. Probably also something on paired permutation test

Again, go see the other paper I have to write that has even more detail (this one has too much I think) for a fuller description along with collection of diagnostics validating the changes made here.

3 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 Gauss, an exponential curve, polynomials of arbitrary degree, and (soon) splines. 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)

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.

4 Bootstrapping

Like the fitting function, the bootstrapping process has been consolidated to a single function,`bdotsBoot`.

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)

I'm actually going to write this section later because it's a massive pain in the ass

5 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

6 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.

7 Conclusion

Improvements made to the `bdots` package have drastically improved the ease of use of the package. 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.