jlmerclusterperm: R (and Julia) package for cluster-based permutation tests on densely-sampled time series data

Introduction to CPA

Cluster-based permutation analysis (CPA) is a simulation-based, non-parametric statistical test of difference between groups in a time series (Maris and Oostenveld, 2007). It is suitable for analyzing **densely-sampled data**, commonly encountered in eye movement and brain signal research where behavioral and neural measures are collected at high sampling rates (Ito and Knoeferle, 2023; Pernet et al., 2015).

CPA is a popular choice of analysis when the research hypothesis is specified up to the *existence* of an effect over a window of time (as predicted by higher-order cognitive processes, for example) but agnostic to the *temporal details* of the effect, such as the precise moment of its emergence and its shape. This situation is common in the cognitive sciences; for example, a psychologist may design an eye-tracking experiment to test whether participants make more looks to the target image in a visual scene when listening to facilitatory versus conflicting auditory cues over the course of a few seconds. Here, the high density of the resulting data poses challenges for traditional parametric tests of difference in condition means such as t-tests and ANOVAs, which cannot overcome temporal auto-correlations and the multiple-comparisons problem.

CPA addresses these challenges by formalizing two notions of what it means for there to be a difference between groups – loosely, *lines* – in a time series:

1. The countable unit of effect (i.e., a **cluster**) is a contiguous span of *sufficiently large* differences between groups that are evaluated at each time point. These time-wise differences are quantified using a *proxy statistic* that is sensitive to both the magnitude and variability of difference between groups.
2. The degree of extremity of a cluster as a whole (i.e., the **cluster-mass statistic**) is the sum of the time-wise differences that contribute to the size of the cluster. This is the test-statistic for CPA.

Within the frequentist paradigm, CPA first identifies the *empirical* clusters in the observed time series data and tests the significance of their cluster-mass statistics against **permutations** of the data, yielding a distribution of simulated *null* clusters. The statistical significance of each empirical cluster identified in the data, then, is the probability of observing a cluster-mass from the null distribution that is as or more extreme as the empirical cluster-mass. Detecting a cluster whose cluster-mass is unlikely to be due to chance is in turn taken as evidence of difference between groups in the time series as a whole (cf. Sassenhagen and Draschkow, 2019).

Motivation for jlmerclusterperm

CPA's strengths lie in its flexibility: it can consume the entire stretch of the time series data without violating non-independence and overfitting to local temporal dynamics. The interpretability of its results is also helped by the fact that the researcher can choose the appropriate proxy statistic. This allows the cluster-mass statistic to be quantified using measures relevant to the research question, such as the t-statistics of regression terms.

However, these advantages are not without limitations. Namely, CPAs are **computationally expensive**. For example, an experiment involving 10-second stimuli with the response variable (e.g., looks to the target) averaged over 20ms bins will still yield 500 data points along the time dimension. In turn, the identification of clusters requires that many evaluations of the proxy statistic. This procedure is then repeated for each permutation of the data; assuming a thousand simulations, that's half-a-million tests of difference between groups.

Existing open-source implementations (R packages `eyetrackingR`, `permutes`, `permuco`, `clusterperm`, among others) celebrate the strengths of CPA's design, but do not prioritize overcoming its blatant performance bottlenecks in practice. For example, despite the "embarassingly parallel" problem of repeated simulations, there have been little serious attempts at integrating parallel processing at the appropriate level of implementation. Moreover, user-facing functions often treat CPA as a monolithic test despite its well-defined algorithmic steps; this design discourages researchers from fully leveraging CPA's flexibility and inspecting its assumptions on their data.

In face of such high costs to usability, researchers have remained overly cautious in the ways that they conduct CPAs. Researchers often sacrifice the temporal resolution of the data by selecting a smaller window of analysis and/or averaging the data over larger time bins. Researchers are also often forced to compromise on using less sensitive tests to compute the time-wise proxy statistics, such as aggregating the data by participant and conducting t-tests on participant means, as opposed to fitting (generalized) mixed-effects regression models which can account for such group-level conditional variances through partial pooling.

Software design

Against this backdrop, the R package `jlmerclusterperm` (Choe, 2024) approaches CPA as first and foremost a computing problem, recognizing that performance and usability impose non-trivial constraints on practicing good statistics. The package specifically focuses on (mixed-effects) regression-based implementations of CPA and offers improvements in the speed of execution, modularity of function design, interpretability of (intermediate) results, and interoperability with related statistical software. We now discusses these considerations in turn.

Speed. Performance optimizations are brought about through two design choices. First, `jlmerclusterperm` integrates Julia in the backend via the `JuliaConnectoR` package, using a tightly-coupled custom Julia module built on top of the `GLM` and `MixedModels` libraries. This substantially improves the speed of fitting regression models for computing time-wise statistics compared to in R (e.g., via `lme4`). Second, the simulations, by default, run multi-threaded in the Julia subprocess, with appropriate infrastructural support including thread-safe RNGs. As a result, CPAs can comfortably scale up to tens of thousands of permutation samples.

Modularity. In addition to offering a one-stop function for CPA, `jlmerclusterperm` also exports a set of functions corresponding to each algorithmic step of the CPA that can be ran in sequence to the same effect. This encourages researchers to explore and validate CPA's performance on their data. For example, one can selectively diagnose the robustness of a particular cluster across different threshold values without repeating the entire CPA from scratch. This modular design is also important for pedagogy: it makes internal computations more transparent and emphasizes the fact that intermediate outputs are themselves debug-able, visualizable, and so on.

Interpretability. As is typical of research statistical software, the (intermediate) outputs of `jlmerclusterperm` are complex objects whose structural details are uninteresting to the average user. Thus, `jlmerclusterperm` uses lightweight `S3` classes to implement custom print methods via the `cli` package, returning carefully formatted console reports that highlight just the immediately useful information. Usability is further advanced through the ample use of progress bars, messages, early warnings for degenerate CPA specifications, among others.

Interoperability. `jlmerclusterperm` strives to seamlessly integrate with other tools in the statistical analysis ecosystem. For example, it extends methods from `generics` such as `glance()` and `tidy()` for collecting statistical objects as "tidy data" (Wickham, 2014) for a further inspection of results using familiar data-wrangling tools. The R-Julia interface is also helped by the option to return pointers to internal Julia objects before they are collected into R, allowing users to manipulate these lower-level objects in the Julia subprocess via `JuliaConnectoR`.

Conclusion

Development of `jlmerclusterperm` began in a seminar on eye movements research. This origin shaped its priorities, emphasizing accessibility and pedagogy at every turn. Beyond the design of the software itself, these values are promoted through its documentation, crafted to support researchers without expertise in statistical computing themselves. The documentation website <https://yjunechoe.github.io/jlmerclusterperm> hosts several tutorials covering on topics in computing (e.g., setting up a CPA to run asynchronously) as well as case studies that replicate prior published results with comparisons (in performance and in API) to existing software.

Since its inception, `jlmerclusterperm` has found a place in both eye-tracking and brain signal research. Its reliability is upheld through regular and rigorous testing on CRAN and GitHub Actions, proving itself resilient amid the finicky aspects of R-Julia integration. Future directions include supporting an extra spatial dimension in the permutation algorithm and offering native visualization capabilities.

References

Choe, J. (2024). *jlmerclusterperm: Cluster-Based Permutation Analysis for Densely Sampled Time Data* [R package version 1.1.4]. https://doi.org/10.32614/CRAN.package.jlmerclusterperm

Ito, A., & Knoeferle, P. (2023). Analysing data from the psycholinguistic visual-world paradigm: Comparison of different analysis methods. *Behavior Research Methods*, *55*(7), 3461–3493.

Maris, E., & Oostenveld, R. (2007). Nonparametric statistical testing of eeg- and meg-data. *Journal of Neuroscience Methods*, *164*(1), 177–190.

Pernet, C., Latinus, M., Nichols, T., & Rousselet, G. (2015). Cluster-based computational methods for mass univariate analyses of event-related brain potentials/fields: A simulation study. *Journal of Neuroscience Methods*, *250*, 85–93.

Sassenhagen, J., & Draschkow, D. (2019). Cluster-based permutation tests of meg/eeg data do not establish significance of effect latency or location. *Psychophysiology*, *56*(6).

Wickham, H. (2014). Tidy data. *Journal of Statistical Software*, *59*(10), 1–23.