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

June Choe

University of Pennsylvania

Introduction to cluster-based permutation analysis

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 time series 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 is common in the cognitive sciences. For instance, a researcher might conduct an experiment measuring looks to the target (vs. a distractor) image in a visual scene over the course of several seconds. The high density of the resulting data (often thousands of samples per trial) is problematic for traditional parametric methods such as repeated t-tests and ANOVAs, which suffer from temporal autocorrelations and the multiple-comparisons problem (Piai et al., 2014).

CPA addresses these challenges by formalizing two notions of what it means for there to be a difference between groups—loosely speaking, *divergence between 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 group means.
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 cluster. This is the test-statistic for CPA.

In frequentist spirit, CPA first identifies the *empirical* clusters in a time series and tests the significance of their cluster-mass statistics against **permutations** of the data reprsenting the *null* distribution. Observing an empirical cluster whose cluster-mass statistic is unlikely to emerge under this permutation procedure is taken as evidence of difference between groups in the time series as a whole (cf. Sassenhagen and Draschkow, 2019).

Motivation

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 CPA to target specific contrasts relevant to the research question, e.g., via a custom contrast-coding of regression terms.

At the same time, there are practical barriers to doing CPA well. 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 yields 500 data points along the time dimension. In turn, identifying clusters in the data requires that many evaluations of the proxy statistic. This procedure is then repeated for each permutation of the data; assuming a thousand simulations, that totals up to half-a-million time-wise tests of difference, be it t-tests, ANOVAs, regressions, or another proxy.

Existing open-source implementations of CPA (R packages `eyetrackingR`, `permutes`, `permuco`, `clusterperm`, among others) celebrate the strengths of its design, but do not prioritize overcoming this blatant performance bottleneck in practice. For example, despite the "embarrassingly parallel" problem of independently simulated runs, there have been little serious attempts at integrating parallelization. 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 proxy statistics, such as by averaging the data across items within each 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 without sacrificing statistical power.

Design of jlmerclusterperm

Against this backdrop, the R package `jlmerclusterperm` 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 vastly improves the speed of fitting regression models to compute the proxy 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 permuted resamples.

Modularity. In addition to offering a one-stop function for CPA, `jlmerclusterperm` also exports a set of functions corresponding to each of its algorithmic steps 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 re-running 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 visualizable, debuggable, 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. Accessibility is further advanced through the ample use of progress bars, messages, and 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 directly manipulate these lower-level objects in the Julia subprocess.

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 on related topics in computing (reproducibility, asynchronous execution, etc.) 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, with over 10k downloads on CRAN. 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

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.

Piai, V., Dahlslätt, K., & Maris, E. (2014). Statistically comparing eeg/meg waveforms through successive significant univariate tests: How bad can it be? *Psychophysiology*, *52*(3).

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.