# Contributing Rapid Permutation Testing to SNPM

Felipe Gutierrez
Computer Sciences
University of Wisconsin-Madison
fgutierrez3@wisc.edu

## Abstract

*The goal of this project is to contribute, a novel permutation testing algorithm to perform Multiple Hypothesis Testing to the Statistical Non Parametric Mapping (SNPM) toolbox. Multiple Hypothesis Testing is a problem that needs to be addressed in many neuroimaging studies. Permutation testing is considered the most accurate method to do Multiple Hypothesis Testing, however, due to high computational costs, neuroscientists prefer not to use it. The algorithm developed by [0] reduces this computational burden by 50x, but it is still not widely used because of the number of technical details that come along with it. Therefore, the work for my final project will focus on implementing this rapid permutation testing algorithm in C++, creating Mex files, replicating results, and writing an extensive documentation such that it can be contributed to one of the major statistics toolboxes used by the neuroscience community, SNPM.*

## 1. Introduction

Suppose we are running a clinical trial to test a new drug. A group is composed of people that took the drug and people that did not. Brain imaging data gives a voxel map of the brain, and we are interested if there were any variations in each voxel due to the drug. Assume the probability of a significant variation appearing in some voxel is only 0.01 and we are interested for variations in a large number of voxels. Then the probability of seeing a variation in at least one voxel simply due to chance becomes very high because of the large number of voxels. In order to avoid this we want to control the Family Wise Error Rate (FWER). There exist two different methods to do this: The Bonferroni Correction and Permutation Testing.

**Bonferroni Correction**: The Bonferroni Correction is considered a conservative method because the idea behind it is to normalize the statistical significance by the number of hypothesis being tested. This means that in the examples above the statistical significance would be 0.01/"Number of Voxels". However, in this context since many of the voxels are highly correlated and the number of voxels is very large, the probability of getting a significant result becomes very small. The problem with this approach is that the Bonferroni Correction assumes that all the voxels are independent data, however they actually are highly dependent.

**Permutation Testing**: On the other hand permutation testing is desirable in this context because it is free of any distribution assumption. The idea behind it, is that if we have labeled high dimensional points, and a test statistic (t-statistic) which gives us information about the interaction between labeled groups. If we randomly permute the labels and recompute the t-statistic multiple times, which gives a the Global Null Distribution. Then for each permutation take the maximum t-statistic and create a histogram. If the t-statistic sampled from the original labels appears in the histogram, close to the mean, it means that there is a high probability that there was no difference between the groups (drug and no drug). However, if it is located on the tails of the histogram it means that there is a low probability that there was no difference between the groups.

In neuroimaging studies, the high dimensional points are each person in the study and the labels are: took the drug, did not take the drug. The t-statistic gives us information of the difference between a voxel of one group vs the other; giving us a t-statistic for each voxel.

The drawback with permutation testing is that the computational burden associated to it is high. Which makes it difficult to be used in practice. The recently developed algorithm by [0] decreases the computational burden by 50x. However, it is not being used by the neuroscience community due to the fact that it is unavailable in commonly used statistics toolboxes such as SNPM. The goal of this project is to implement this novel algorithm, document and format it such that it can be contributed to SNPM, which would have a high impact in future neuroimaging studies. The permutation testing implementation will act as a blackbox that perform multiple hypothesis testing which will abstract all the technical details associated with permutation testing and the novel algorithm.

## 2. Related Work

The rapid permutation testing algorithm [0] is currently fully implemented in Matlab. Which means that the performance of the computational intensive can be improved by simply re-implementing the algorithm in C++. Furthermore, the documentation associated to the currently available implementation is limited to a highly technical paper [0], which might not effectively communicate the big picture to the larger community who are not interested in the technical details. Therefore a detailed documentation that communicates why a scientist would want to use this algorithm when doing multiple hypothesis testing, will be developed. Finally, even though this algorithm could be considered state of the art, the implementation is not in a format such that it can be contributed to SNPM. This means that the new implementation will have to follow a certain format which is defined by the developers of SNPM.

## 3. Problem Description and Techniques You Want to Use

The goal of this project is to first implement the novel rapid permutation testing algorithm developed by [0] in C++. Then develop an extensive documentation that explains the big picture of this algorithm as well as the technical details associated with it. Finally, the new implementation will be formatted such that it can be contributed to SNPM.

The major technical challenge of this project is to create an efficient C++ implementation. The algorithm contains two sections where major matrix operations such as multiplication and inverse are done. This means that a C++ linear algebra library needs to be identified. Multiple linear algebra such as armadillo and LAPACK will be tested to see which one produces the best performance for this kind of problem.

Once the implementation is done, I will have acquired a good understanding of the algorithm which will allow me to properly document this program for both highly technical and non-technical users.

## 4. Experimental Evaluation

The new implementation will be tested by reproducing the results generated by [0]. Additionally, the performance between the different C++ linear algebra libraries tested will be compared, to be able to make an informed decision about which library to use.

## 5. Conclusion and Discussion

This project will have six main steps, and two future work steps that will most likely be reached towards the end of the semester:

1. Reproduce the results of [0] and understand the existing Matlab implementation.
2. Develop a C++ version using different linear algebra libraries.
3. Validate the results of the new program.
4. Create Mex functions and develop Matlab examples using them.
5. Write the documentation that will include the big picture as well as the more technical details.
6. Format the program such that it can be contributed to SNPM.
7. Future Work: Research techniques to improve the performance, such as a parallel implementation.
8. Future Work: Run the algorithm in other datasets and produce performance results compared to state of the art.

## References

[1] C. Hinrichs, V.K. Ithapu, Q. Sun, S.C. Johnson, V. Singh, Speeding Up Permutation Testing in Neuroimaging, Neural Information Processing Systems, 890-898, 2013. 1, 2