Accelerating Permutation Testing in Voxel-wise Analysis through Subspace Tracking: A new plugin for SnPM

Felipe Gutierrez-Barragan, Vamsi K. Ithapu, Chris Hinrichs, Camille Maumet, Sterling C. Johnson, Thomas E. Nichols, and Vikas Singh

http://felipegb94.github.io/RapidPT/

The paper

- **Problem:** Permutation testing is computationally expensive in voxel-wise analysis.
- **Solution:** A model that leverages the structure of the permutation testing matrix to reduce the computation runtime.
- **Contributions:**
  - *Theoretical framework + Algorithm*
  - *RapidPT:* A MATLAB toolbox for fast and robust permutation testing.
    - 2x – 38x faster than state of the art (SnPM toolbox) and 20x – 1000x faster than a naïve permutation testing implementation.
  - *SnPM Plugin:* RapidPT is incorporated into a widely used neuroimaging toolbox. Not a common contribution of new algorithms.

Schedule

Background: Problem

- Consider a study with *n* subjects from two groups (ex: sick vs. healthy, resting state vs. non resting state)
- For each subject we have a *v* dimensional measurement vector (**voxels**, genes, region of interest, etc)
- **Question:** Does the data display any interesting activity? If so, where is that “interesting” activity? How certain are we?
- **Goal:** Classify each voxel as either active or not active and associate a probability (alpha) to how certain we are of our classification of each voxel.

Background: Hypothesis Testing

- **Solution:** Hypothesis Testing does exactly what we want.
  - Calculate the probability that your claim/hypothesis is true.

Null Hypothesis vs. Alternate Hypothesis

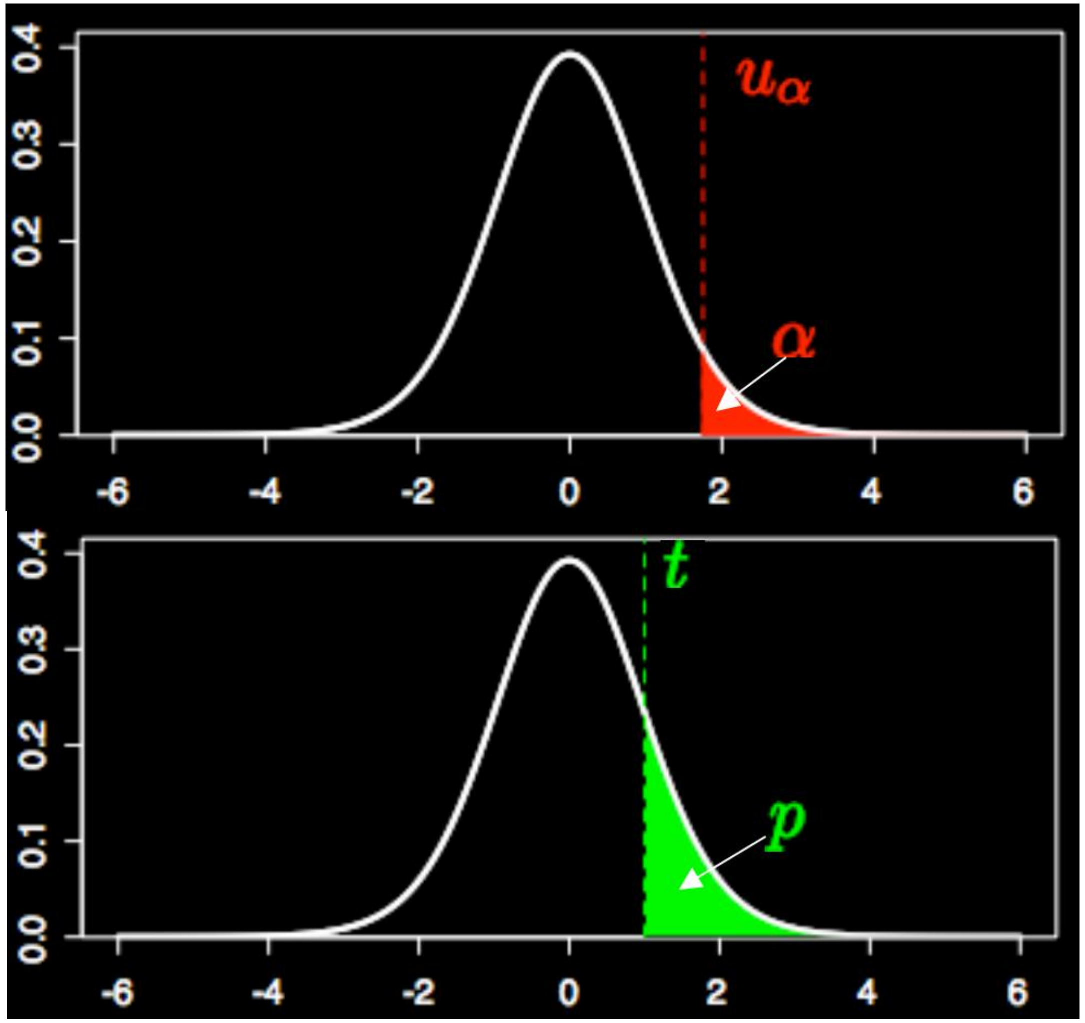
Inactive vs. Active

H_0 vs. H_a

- Basic idea:
  - Assume voxels from both groups come from the same distribution (null hypothesis).
  - Test if they do. If they do accept null, if they don’t reject null.

Background: Hypothesis Testing - Procedure

1. Choose appropriate test statistic (t-test, mean difference, etc).
2. State H_0 and H_a
3. Construct the null distribution for the test statistic. This is the distribution of the statistic given that H_0 is true. This can be done analytically in some cases.
4. State **alpha**
5. Compute the test statistic with the given data.
6. Calculate the probability of observing such a statistic given that H_0 is true **(p-value).**
7. Accept or reject H_0 .



Hypothesis Testing - Types of Error

	Actual Situation “Truth”	
Decision	**H_0 True**	**H_0 False**
Do Not Reject H_0	Correct Decision $1 - \alpha$	Incorrect Decision Type II Error β
Rejct H_0	Incorrect Decision Type I Error α	Correct Decision $1 - \beta$

$\alpha = P(Type\ I\ Error)$ $\beta = P(Type\ II\ Error)$

Multiple Testing Problem

- **FWER:** Probability of making at least Type I Error (False Positive).

$$P(\text{Making at least 1 error in m tests}) = 1 - (1 - \alpha)$$

$$P(\text{Making at least 1 error in m tests}) = 1 - (1 - \alpha)^m$$

  - Common Alpha = 0.05
  - For m = 100 → FWER = 0.99.
  - In neuroimaging m>100,000 tests. One for every voxel.
  - Just by chance we will reject the null MANY times.
- Hence, we want to somehow control for the FWER and keep it under a certain probability α_0.

Controlling FWER

- **Parametric Methods**
  - Assumptions about the data and its distribution.
  - **Bonferroni –** Conservative. Simply set $\alpha_0 = {}^{\alpha}/_{v}$ (*v = number of tests).*
  - **Random Fields Theory –** Estimate number of activation areas, effectively reducing the number of tests.

- **Nonparametric/Resampling Methods**
  - **Permutation Testing –** Empirically estimate the null distribution of the test statistics.
    - Exact control of the FWER.

Permutation Testing

- **Idea:** If the two groups do not differ, then I can permute the group/class labels and end up with approximately same set of test statistics.

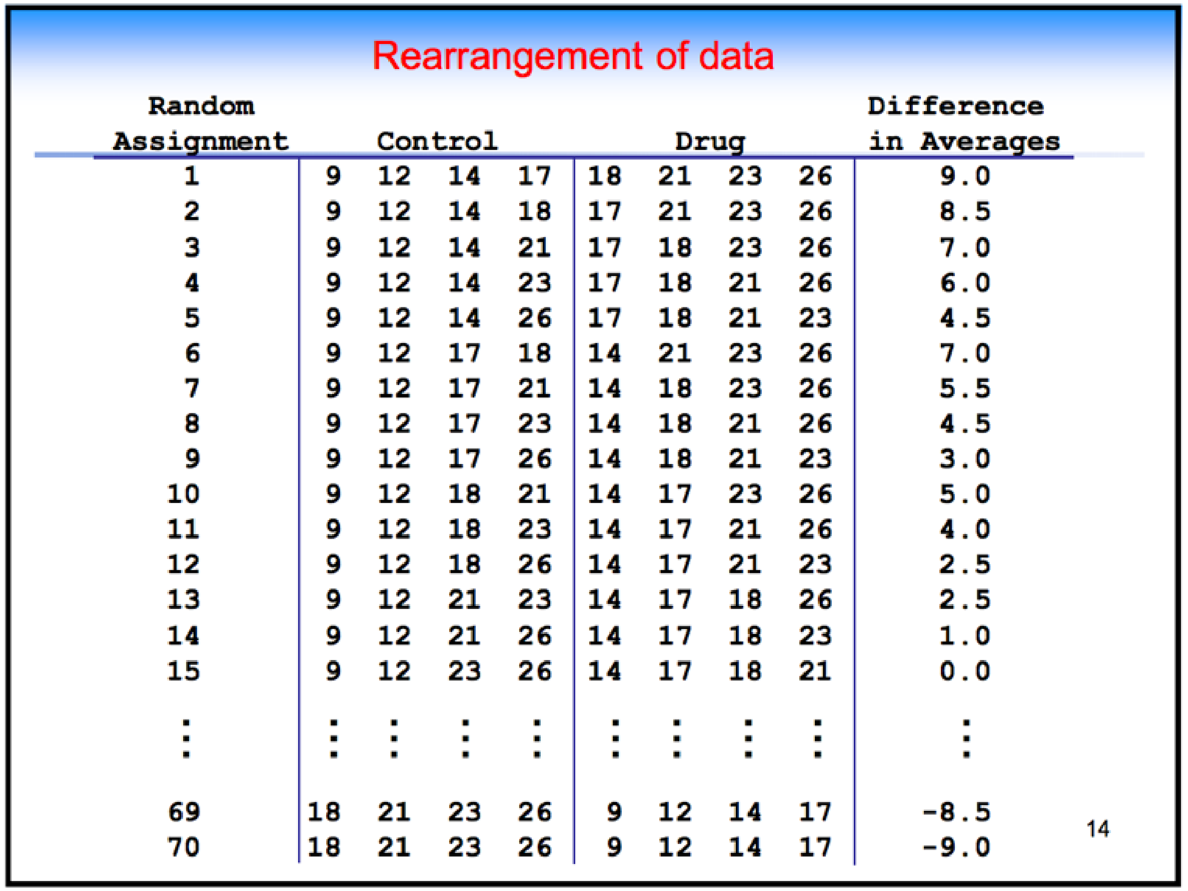
- **Procedure:**
  1. Re-label images (permute the labels of the images).
  2. Compute test statistic for each voxel
  3. Repeat N times.

- After the procedure is done we will have the exact null distribution for each voxel, and we can proceed from step 4 of hypothesis testing .

Permutation Testing – Example, Single Test

	Control				Drug			
Expression	9	12	14	17	18	21	23	26
Average	13				22			

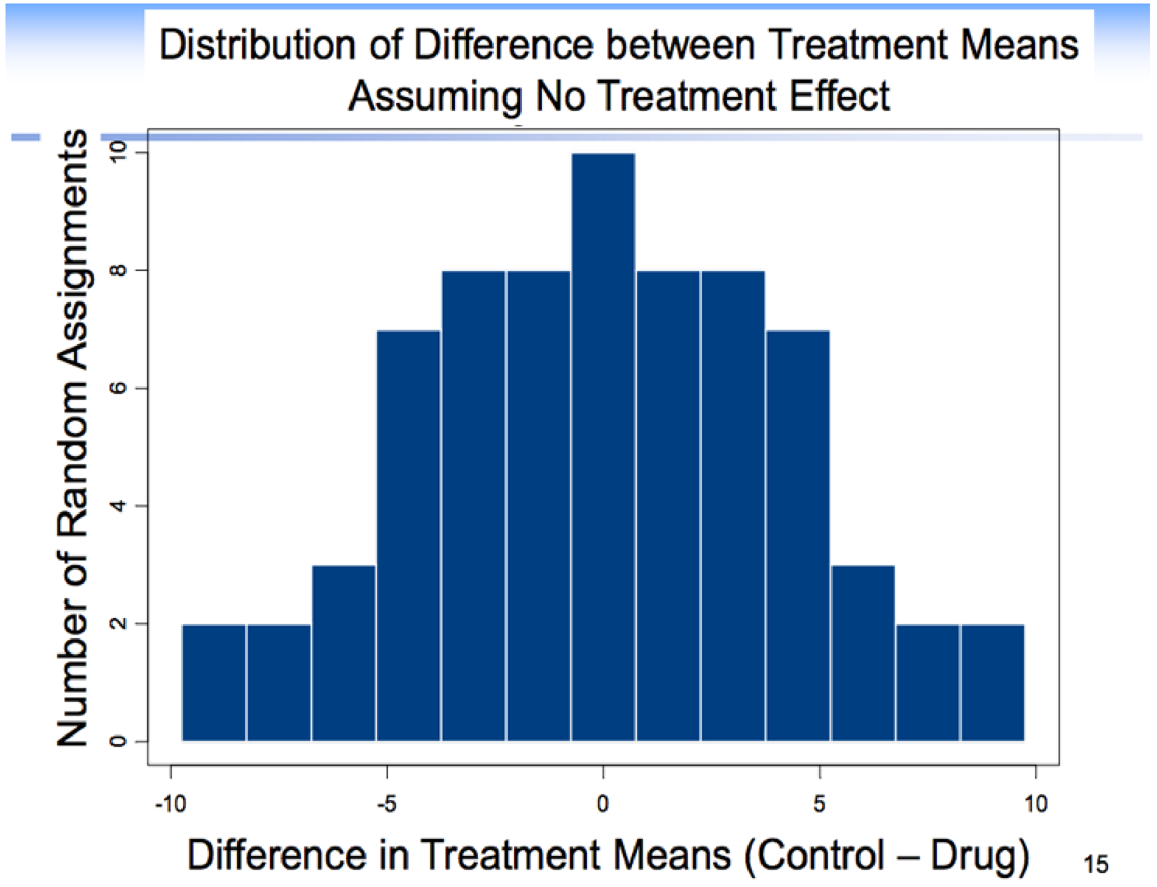
8 choose 4 = 70 possible permutations



Rearrangement of data

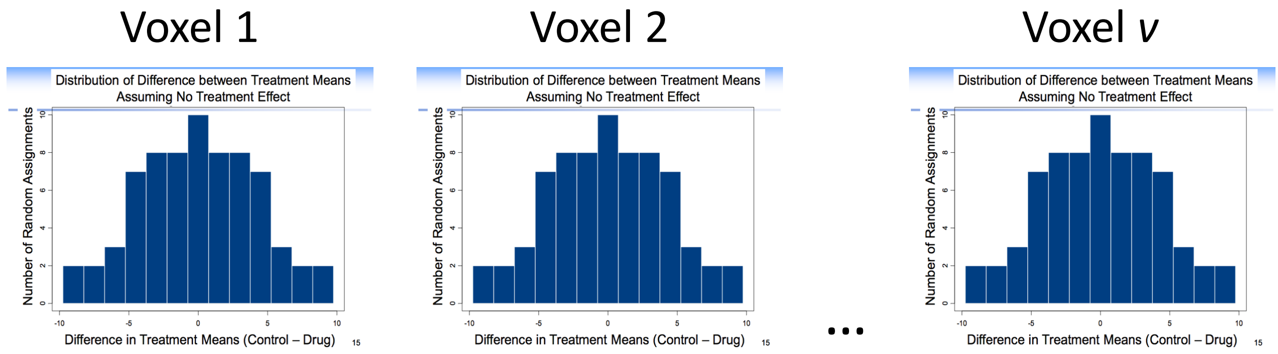
Random Assignment	Control				Drug				Difference in Averages
1	9	12	14	17	18	21	23	26	9.0
2	9	12	14	18	17	21	23	26	8.5
3	9	12	14	21	17	18	23	26	7.0
4	9	12	14	23	17	18	21	26	6.0
5	9	12	14	26	17	18	21	23	4.5
6	9	12	17	18	14	21	23	26	7.0
7	9	12	17	21	14	18	23	26	5.5
8	9	12	17	23	14	18	21	26	4.5
9	9	12	17	26	14	18	21	23	3.0
10	9	12	18	21	14	17	23	26	5.0
11	9	12	18	23	14	17	21	26	4.0
12	9	12	18	26	14	17	21	23	2.5
13	9	12	21	23	14	17	18	26	2.5
14	9	12	21	26	14	17	18	23	1.0
15	9	12	23	26	14	17	18	21	0.0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
69	18	21	23	26	9	12	14	17	-8.5
70	18	21	23	26	9	12	14	17	-9.0

14



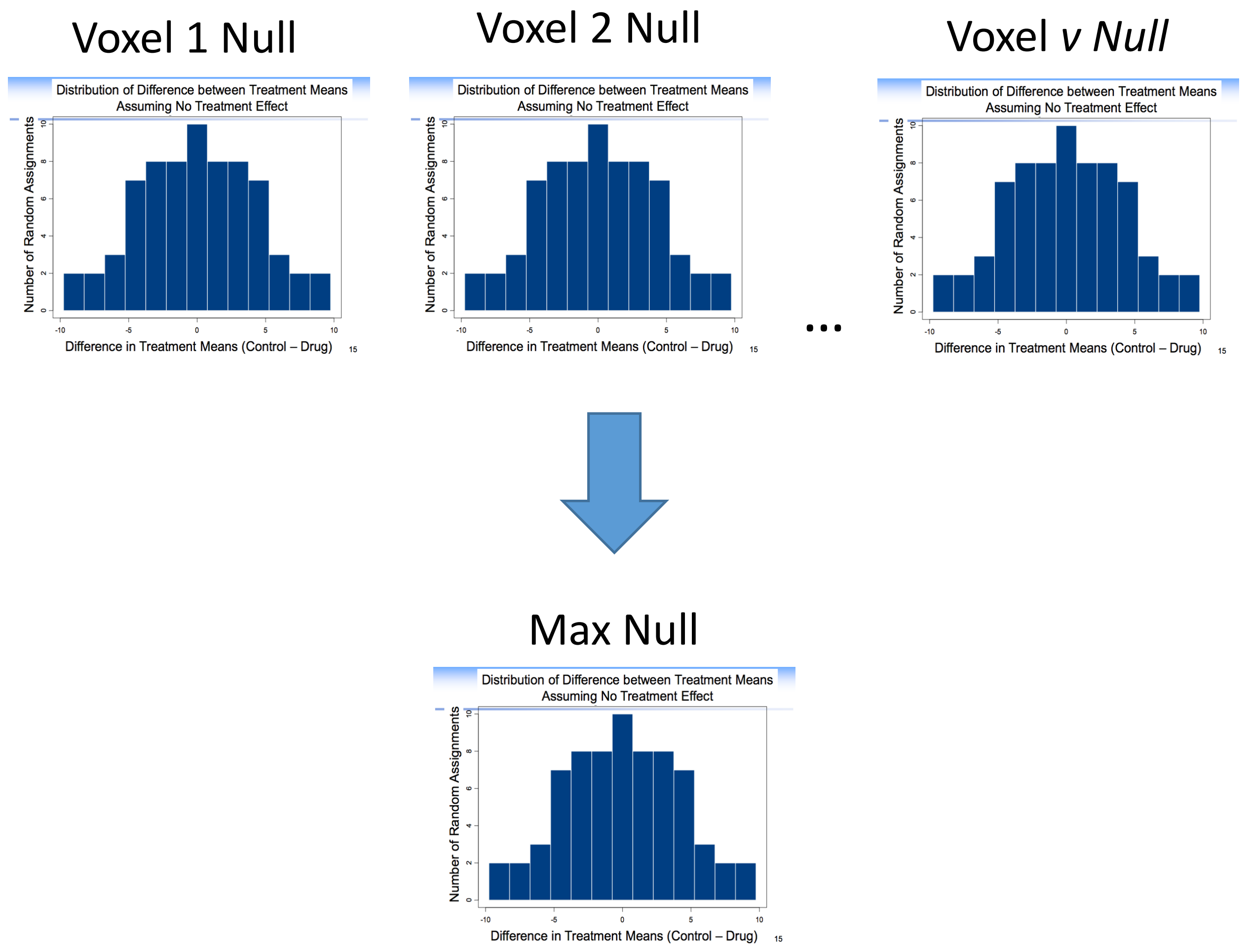
Permutation Testing Matrix

- For each voxel we will have one null distribution associated to it.
- The permutation testing matrix **T** = *v* * *L* matrix of test statistics.
- *Issue:* If v ~ 100,00, L ~10,000 the matrix is around 7.5 Gigabytes.

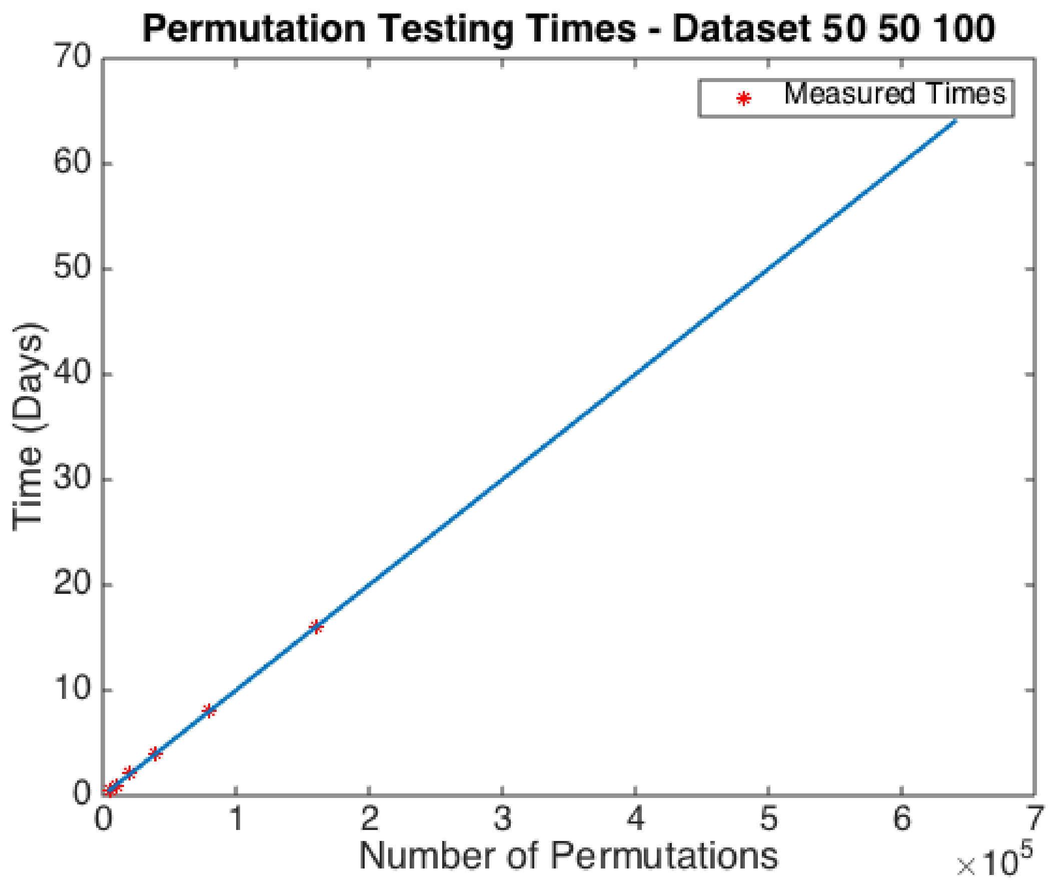


Maxnull Distribution

- We don’t have to keep track of all distributions.
- Simple *keep track of the maximum test statistic across voxels for each permutation.*
- Construct the maxnull distribution



We still have computational issues…

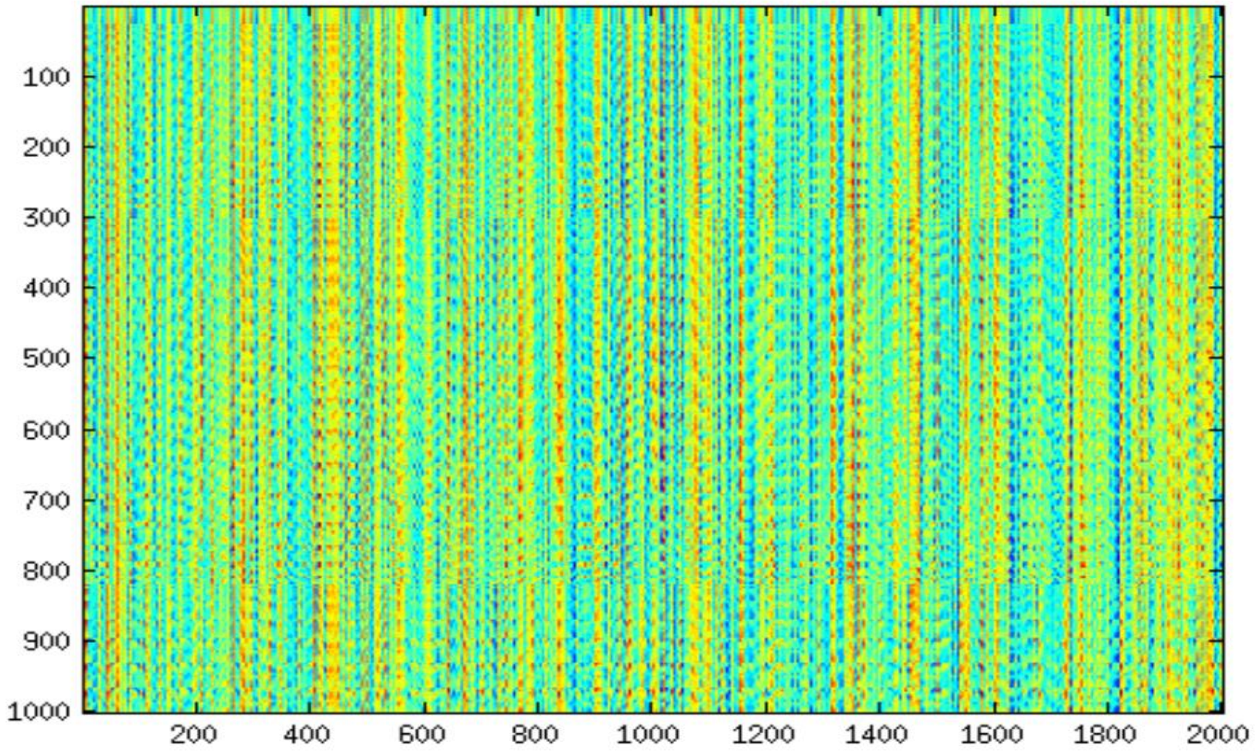


- The possible number of relabelings becomes huge as the number of subjects increases.
  - 30 choose 15 = 155,117,520
- So we compute only a subset of them.
- For each relabeling of the data we have to compute *v* test statistics and find the max!
- More permutations = we can compute lower p-values.
- Embarrassingly parallel problem, BUT not all neuroscientists have access to expensive hardware/supercomputers, and most of them prefer to work on their laptops.

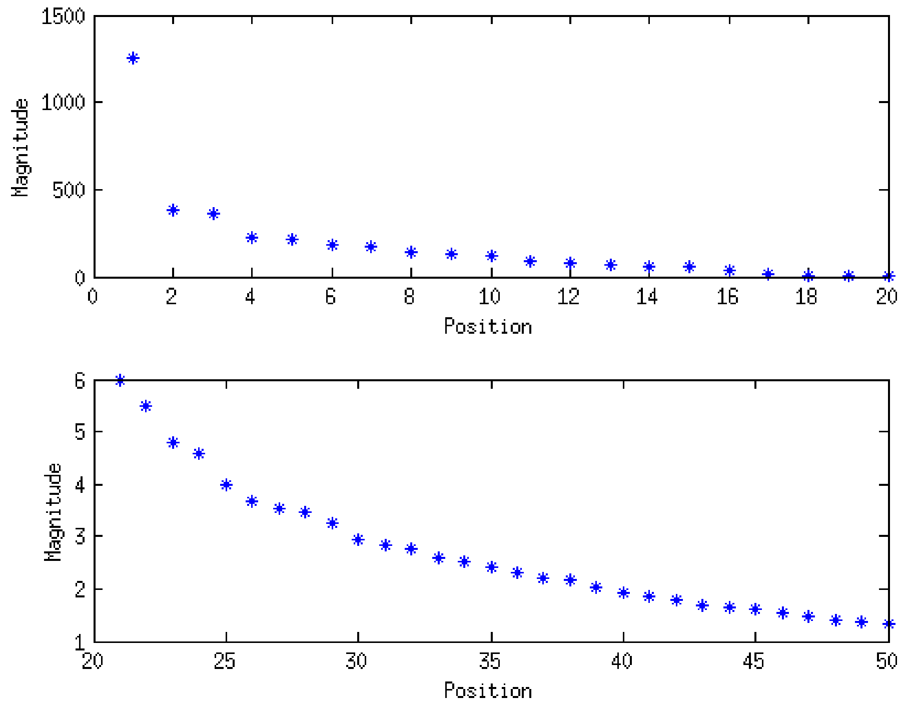
Idea: Look at the structure of *T*

- *T* is "highly structured" – A combination of low-rank signal and high-rank residual
- Example: MRI data 100 healthy vs. non-healthy. v = 1,000, L = 2,000

Permutation Testing Matrix *P*



Singular values of *P*



Core of RapidPT

- Many columns in *T* look similar to other columns as well as many rows look similar to other rows.
  - *Rank of T is constrained by the number of subjects.*
- If we compute a small number of entries of *T* we should be able to estimate the rest of it.
- Mathematically,

$$T = UW + S$$

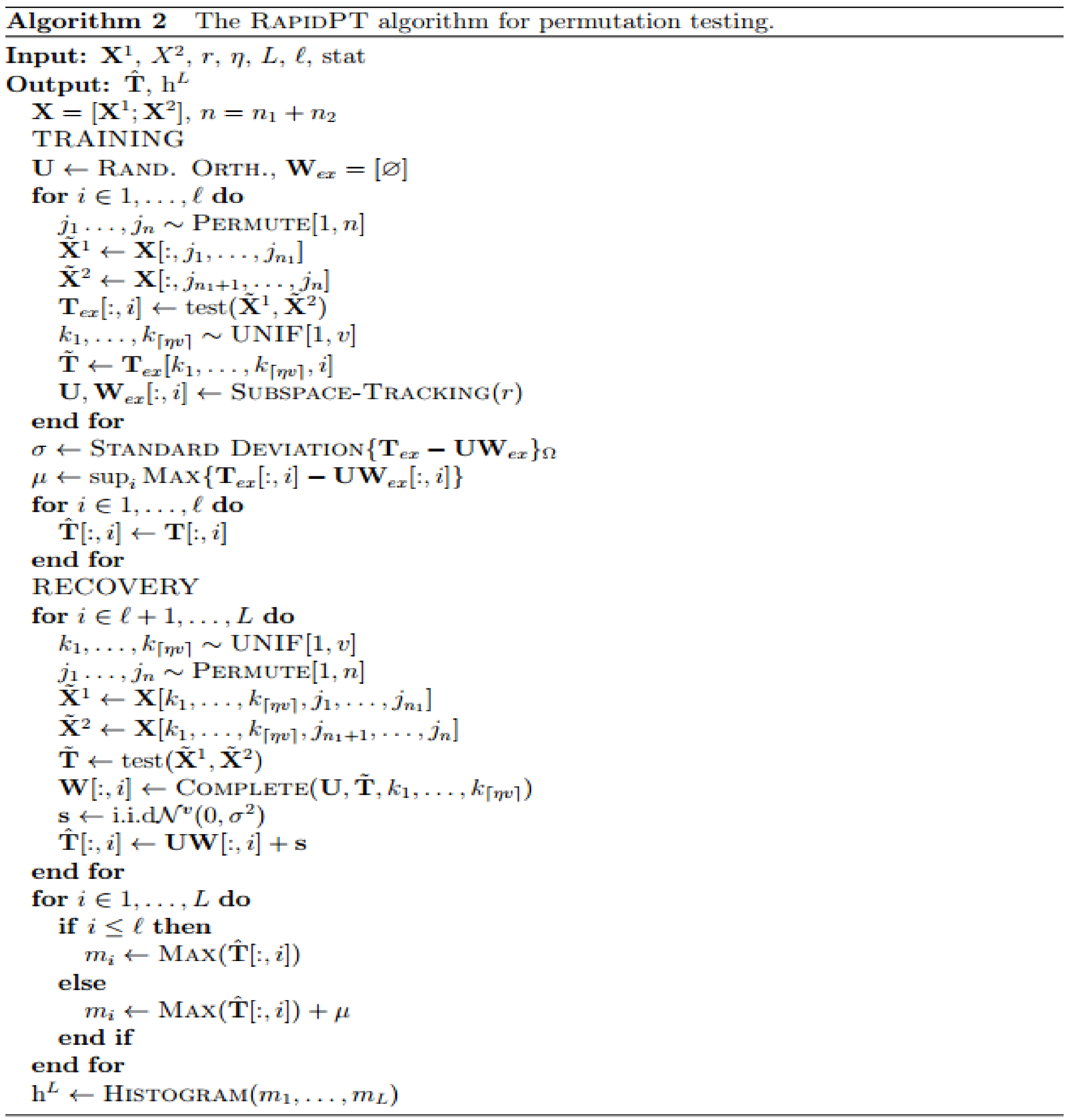
  - U is the low-rank basis of T.
  - W is the coefficient matrix
  - S is a high-rank random residual (noise).
- How many entries? In our experiments, subsampling <1% was enough

RapidPT Algorithm

- Divide the process into training and recovery.
- Training:
  - Calculate a few exact permutations ~ N
  - Estimate low-rank basis through subspace tracking
  - Estimate the residual

    S = T_exact – UW

- Recovery:
  - For each permutation Calculate a subset of the test statistics (eta) for a column of T.
  - Recover the rest through matrix completion.
  - Get max test statistic



Algorithm 2 The RapidPT algorithm for permutation testing.

```
Input: $\mathbf{X}^1$, $X^2$, $r$, $\eta$, $L$, $\ell$, stat
Output: $\hat{\mathbf{T}}$, $\mathrm{h}^L$
  $\mathbf{X} = [\mathbf{X}^1; \mathbf{X}^2]$, $n = n_1 + n_2$
  TRAINING
  $\mathbf{U} \leftarrow$ RAND. ORTH., $\mathbf{W}_{ex} = [\varnothing]$
  for $i \in 1, \ldots, \ell$ do
    $j_1 \ldots, j_n \sim$ PERMUTE$[1, n]$
    $\tilde{\mathbf{X}}^1 \leftarrow \mathbf{X}[:, j_1, \ldots, j_{n_1}]$
    $\tilde{\mathbf{X}}^2 \leftarrow \mathbf{X}[:, j_{n_1+1}, \ldots, j_n]$
    $\mathbf{T}_{ex}[:, i] \leftarrow \mathrm{test}(\tilde{\mathbf{X}}^1, \tilde{\mathbf{X}}^2)$
    $k_1, \ldots, k_{\lceil \eta v \rceil} \sim$ UNIF$[1, v]$
    $\tilde{\mathbf{T}} \leftarrow \mathbf{T}_{ex}[k_1, \ldots, k_{\lceil \eta v \rceil}, i]$
    $\mathbf{U}, \mathbf{W}_{ex}[:, i] \leftarrow$ SUBSPACE-TRACKING$(r)$
  end for
  $\sigma \leftarrow$ STANDARD DEVIATION$\{\mathbf{T}_{ex} - \mathbf{U}\mathbf{W}_{ex}\}_\Omega$
  $\mu \leftarrow \sup_i$ MAX$\{\mathbf{T}_{ex}[:, i] - \mathbf{U}\mathbf{W}_{ex}[:, i]\}$
  for $i \in 1, \ldots, \ell$ do
    $\hat{\mathbf{T}}[:, i] \leftarrow \mathbf{T}[:, i]$
  end for
  RECOVERY
  for $i \in \ell + 1, \ldots, L$ do
    $k_1, \ldots, k_{\lceil \eta v \rceil} \sim$ UNIF$[1, v]$
    $j_1 \ldots, j_n \sim$ PERMUTE$[1, n]$
    $\tilde{\mathbf{X}}^1 \leftarrow \mathbf{X}[k_1, \ldots, k_{\lceil \eta v \rceil}, j_1, \ldots, j_{n_1}]$
    $\tilde{\mathbf{X}}^2 \leftarrow \mathbf{X}[k_1, \ldots, k_{\lceil \eta v \rceil}, j_{n_1+1}, \ldots, j_n]$
    $\tilde{\mathbf{T}} \leftarrow \mathrm{test}(\tilde{\mathbf{X}}^1, \tilde{\mathbf{X}}^2)$
    $\mathbf{W}[:, i] \leftarrow$ COMPLETE$(\mathbf{U}, \tilde{\mathbf{T}}, k_1, \ldots, k_{\lceil \eta v \rceil})$
    $\mathbf{s} \leftarrow$ i.i.d$\mathcal{N}^v(0, \sigma^2)$
    $\hat{\mathbf{T}}[:, i] \leftarrow \mathbf{U}\mathbf{W}[:, i] + \mathbf{s}$
  end for
  for $i \in 1, \ldots, L$ do
    if $i \leq \ell$ then
      $m_i \leftarrow$ MAX$(\hat{\mathbf{T}}[:, i])$
    else
      $m_i \leftarrow$ MAX$(\hat{\mathbf{T}}[:, i]) + \mu$
    end if
  end for
  $\mathrm{h}^L \leftarrow$ HISTOGRAM$(m_1, \ldots, m_L)$
```

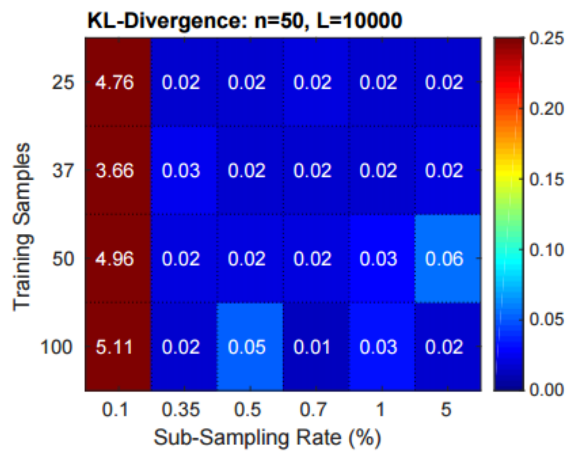
Evaluations Setup

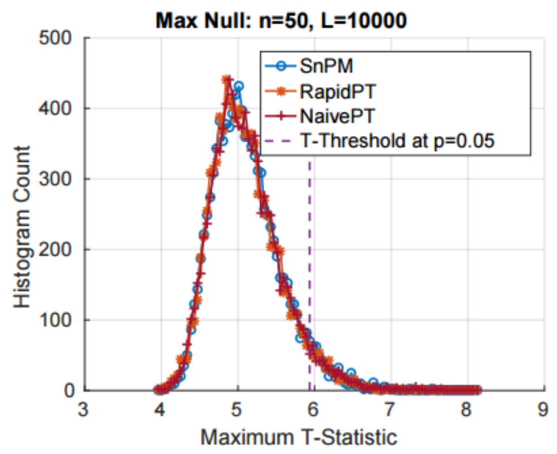
- **Data:** T1 MRIs from ADNI2 are used.
    - 601 subjects (259 AD and 342 CN)
    - SPM preprocessing is applied.
    - GM images with approx. 500,000+ voxels are extracted.
    - Multiple combinations of dataset sizes
- **Experiments:** Can we recover the Maxnull distribution?
    - Stability of hyperparameters – Sub-sampling rates, and training samples
    - Computational Speedups (RapidPT vs. SnPM, RapidPT vs. NaivePT)

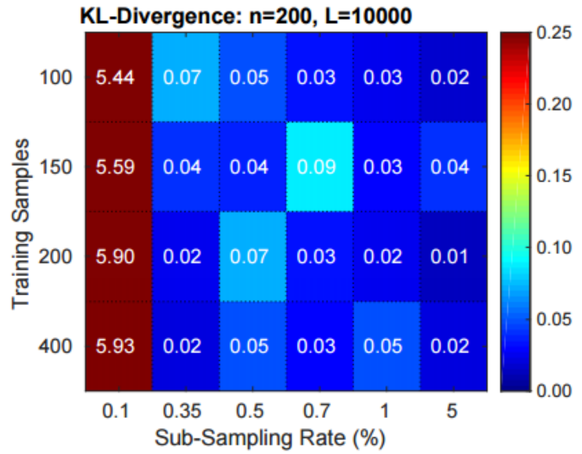
Recovered MaxNull Distribution Accuracy

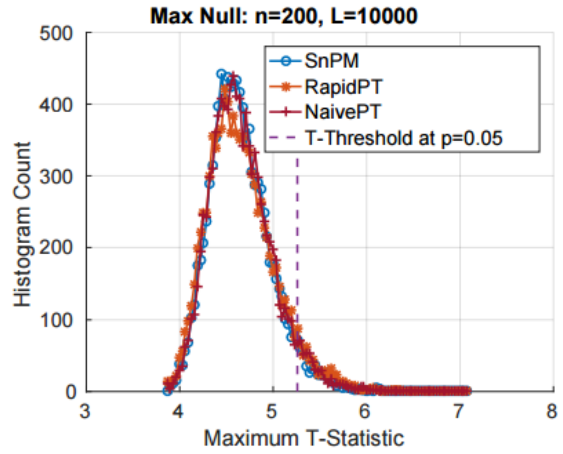
KL-Divergence: Measure of the difference between two distributions

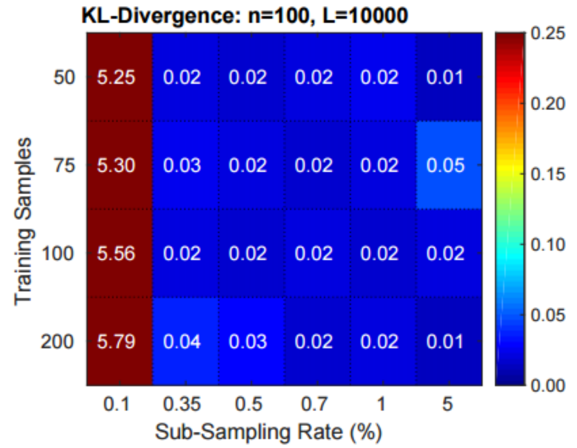
Datasets: 50, 100, 200 and 400 subjects

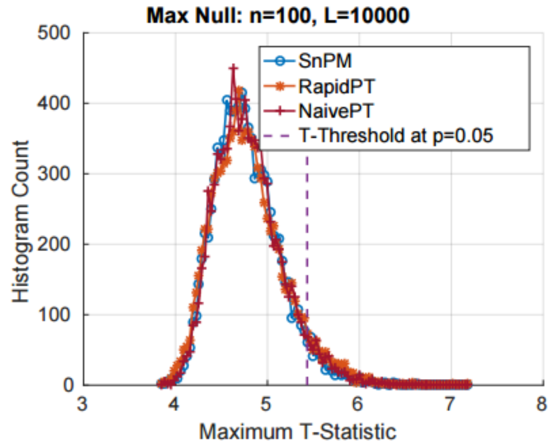


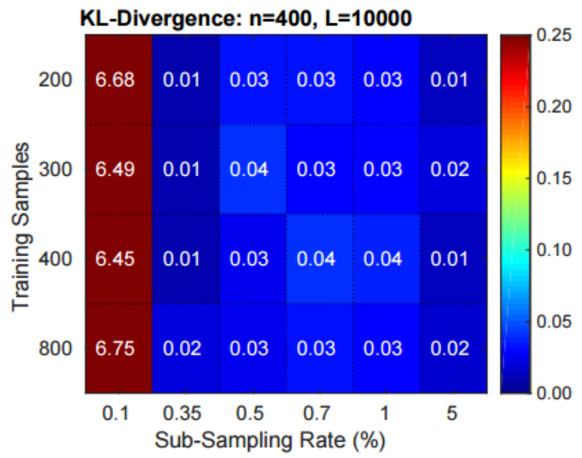


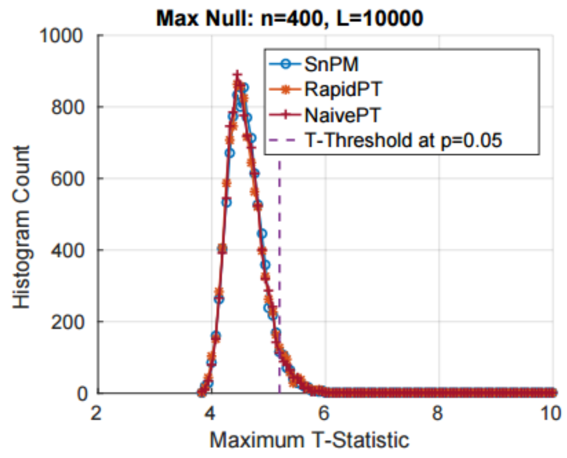












What Sub-Sampling Rate to Choose?

Low-rank matrix completion says that around ~rlog(d) entries are needed, where r is the column space rank and d is the ambient dimension.

$$\eta v \geq n log(v)$$

$$\eta \geq \frac{n log(v)}{v}$$

Empirically the above inequality worked well. However, we can sub-sample at lower rates

In our experiments, the number of subjects and number of voxels for each dataset were:

Number of Subjects and Number of Voxels: (n,v)			
(50,547783)	(100,558295)	(200,568086)	(400,574640)

Table 1: Number of subjects and number of voxels per subject on each dataset.

Therefore, the corresponding estimate of the minimum sub-sampling rate would be:

Minimum sub-sampling rate estimate: η_{min}			
$\approx 0.1206\%$	$\approx 0.2370\%$	$\approx 0.4665\%$	$\approx 0.9231\%$

Table 2: Number of subjects and number of voxels per subject on each dataset.

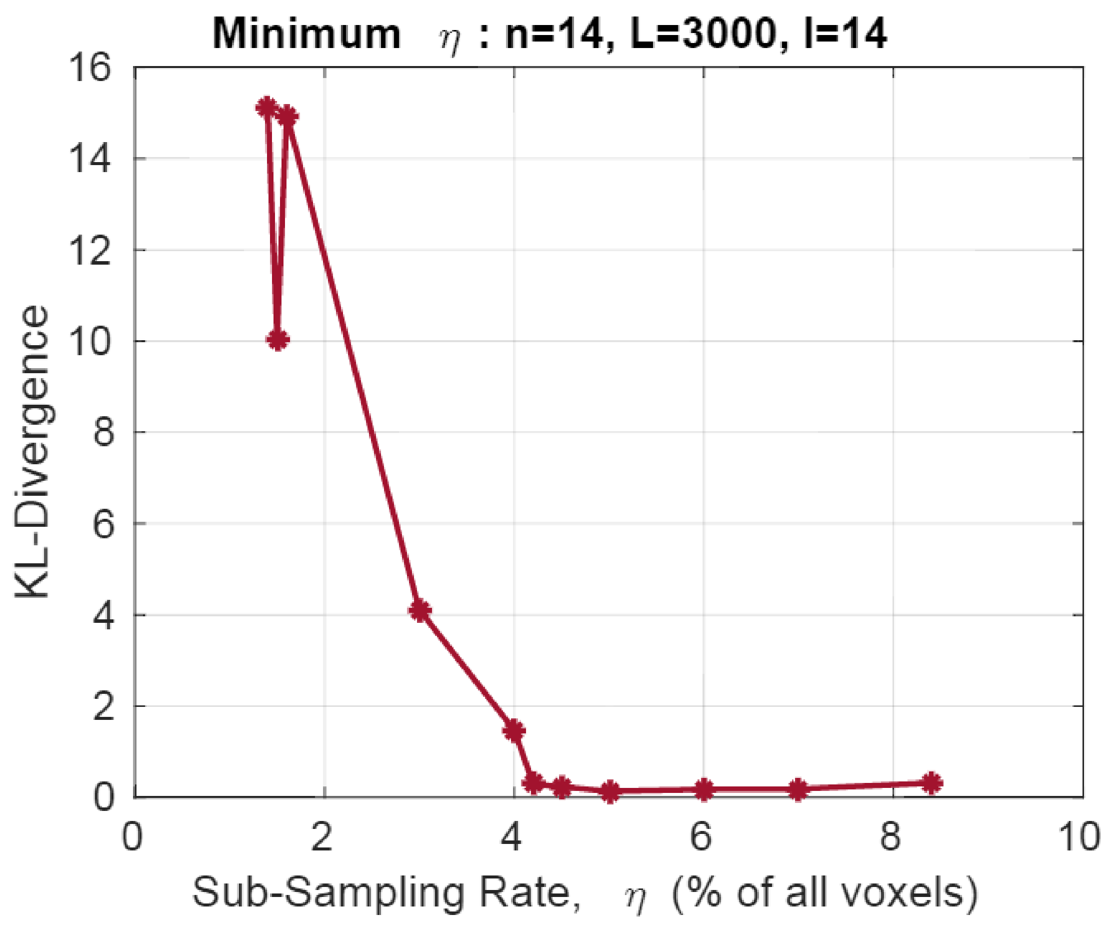
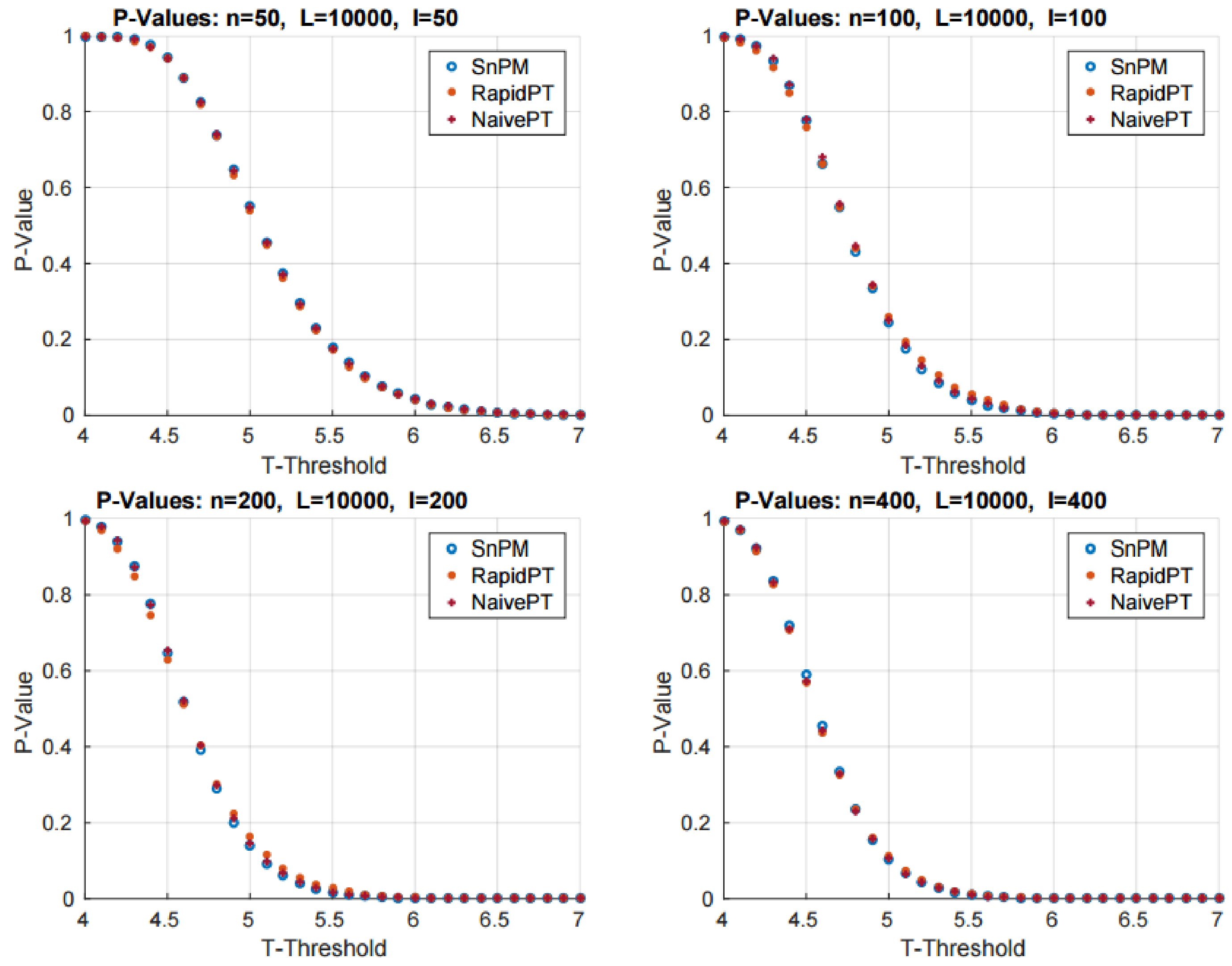
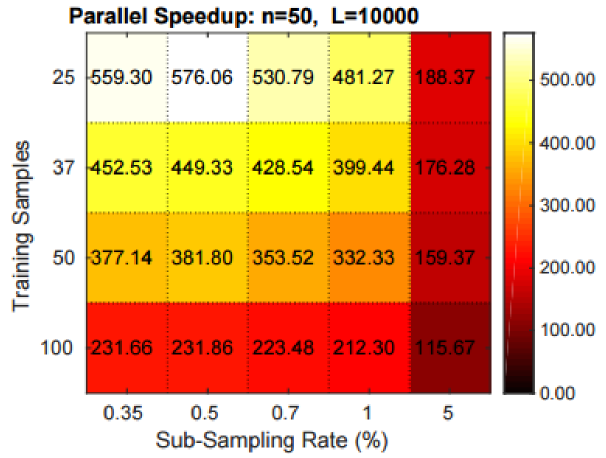


Figure 11: An experiment showing the effect of the sub-sampling rate (η) on the KL-Divergence.

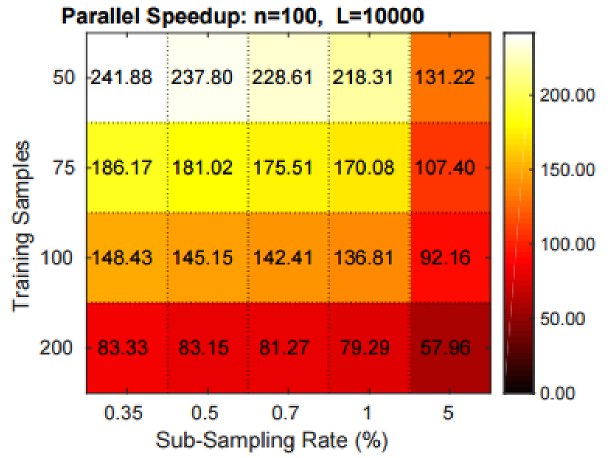
Calculated T-Threshold for a Given P-Value



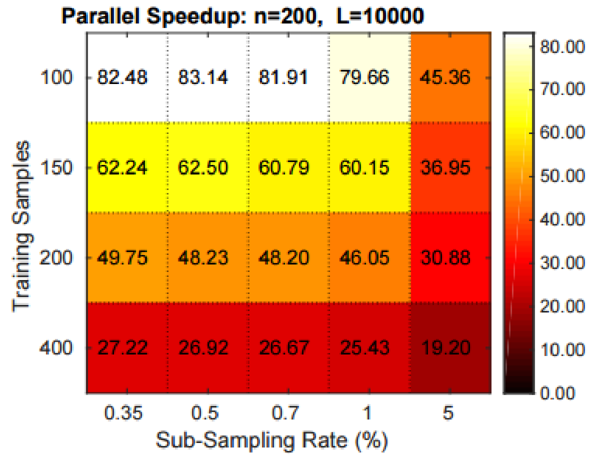
Speedup RapidPT vs. NaivePT



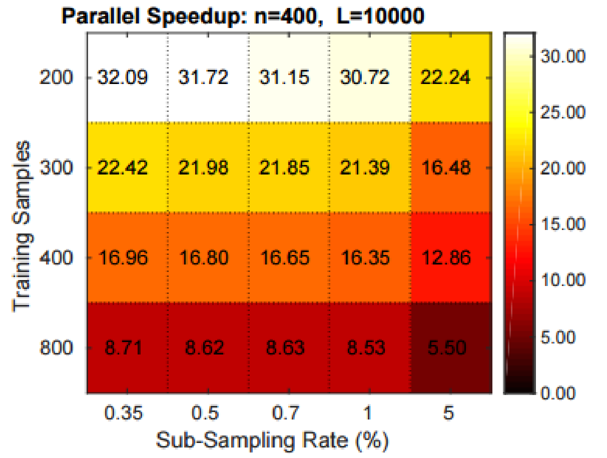
Training Samples	0.35	0.5	0.7	1	5
25	559.30	576.06	530.79	481.27	188.37
37	452.53	449.33	428.54	399.44	176.28
50	377.14	381.80	353.52	332.33	159.37
100	231.66	231.86	223.48	212.30	115.67



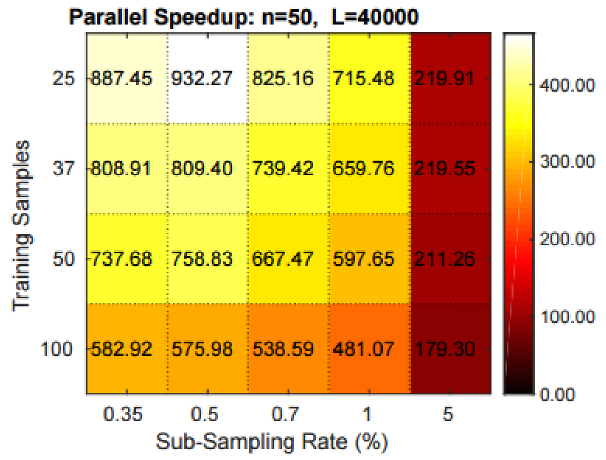
Training Samples	0.35	0.5	0.7	1	5
50	241.88	237.80	228.61	218.31	131.22
75	186.17	181.02	175.51	170.08	107.40
100	148.43	145.15	142.41	136.81	92.16
200	83.33	83.15	81.27	79.29	57.96



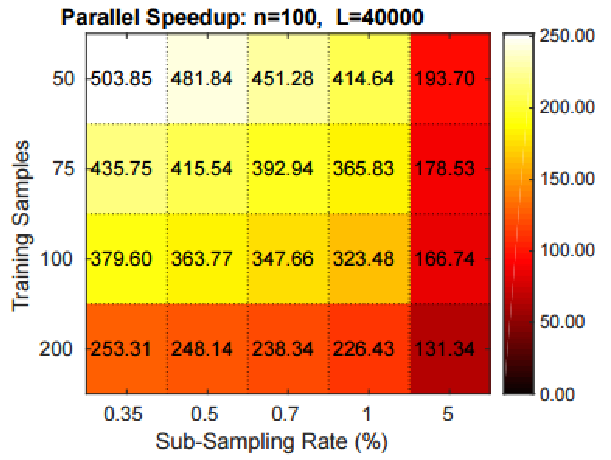
Training Samples	0.35	0.5	0.7	1	5
100	82.48	83.14	81.91	79.66	45.36
150	62.24	62.50	60.79	60.15	36.95
200	49.75	48.23	48.20	46.05	30.88
400	27.22	26.92	26.67	25.43	19.20



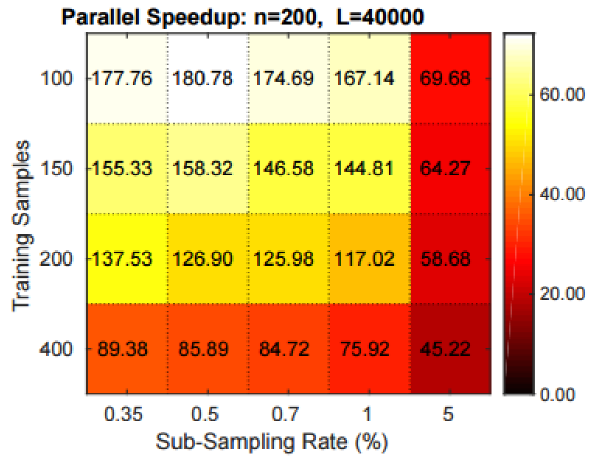
Training Samples	0.35	0.5	0.7	1	5
200	32.09	31.72	31.15	30.72	22.24
300	22.42	21.98	21.85	21.39	16.48
400	16.96	16.80	16.65	16.35	12.86
800	8.71	8.62	8.63	8.53	5.50



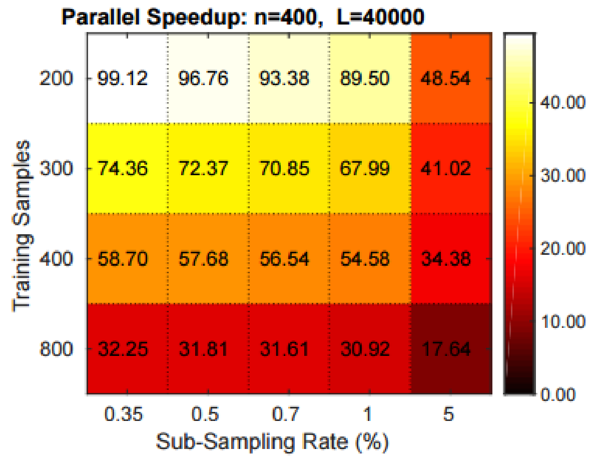
Training Samples	0.35	0.5	0.7	1	5
25	887.45	932.27	825.16	715.48	219.91
37	808.91	809.40	739.42	659.76	219.55
50	737.68	758.83	667.47	597.65	211.26
100	582.92	575.98	538.59	481.07	179.30



Training Samples	0.35	0.5	0.7	1	5
50	503.85	481.84	451.28	414.64	193.70
75	435.75	415.54	392.94	365.83	178.53
100	379.60	363.77	347.66	323.48	166.74
200	253.31	248.14	238.34	226.43	131.34

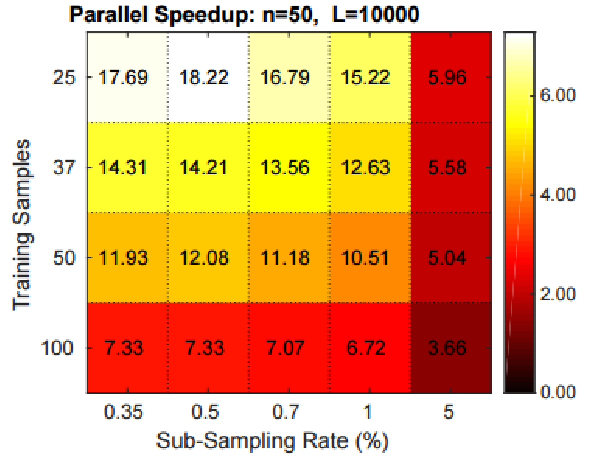


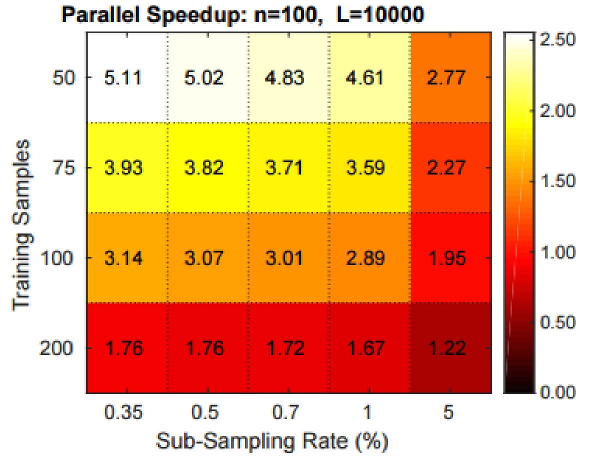
Training Samples	0.35	0.5	0.7	1	5
100	177.76	180.78	174.69	167.14	69.68
150	155.33	158.32	146.58	144.81	64.27
200	137.53	126.90	125.98	117.02	58.68
400	89.38	85.89	84.72	75.92	45.22

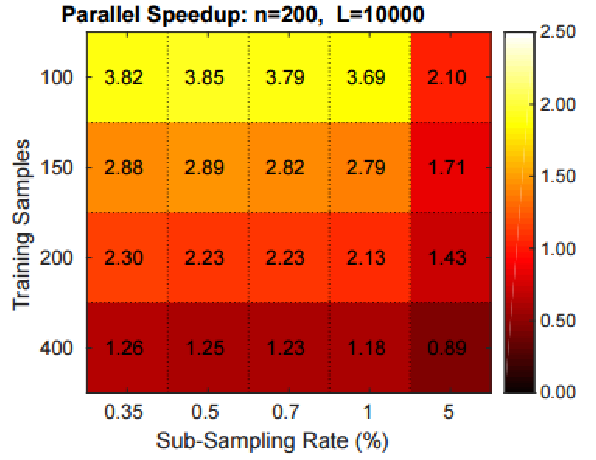


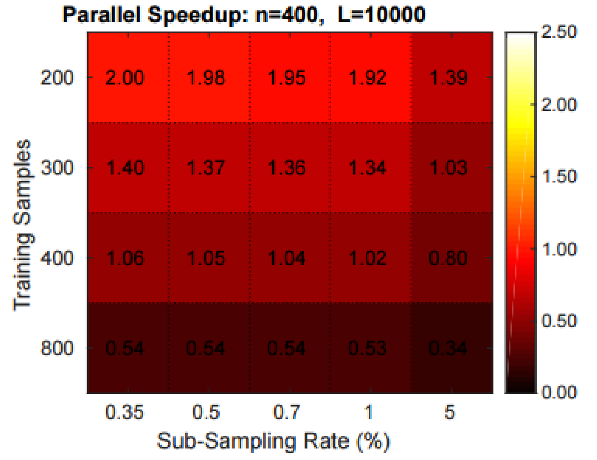
Training Samples	0.35	0.5	0.7	1	5
200	99.12	96.76	93.38	89.50	48.54
300	74.36	72.37	70.85	67.99	41.02
400	58.70	57.68	56.54	54.58	34.38
800	32.25	31.81	31.61	30.92	17.64

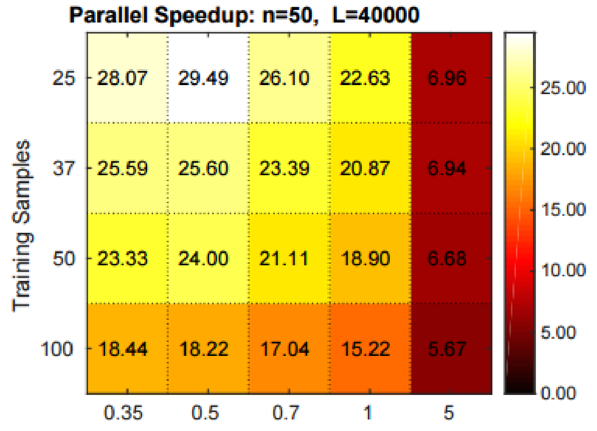
Speedup RapidPT vs. SnPM

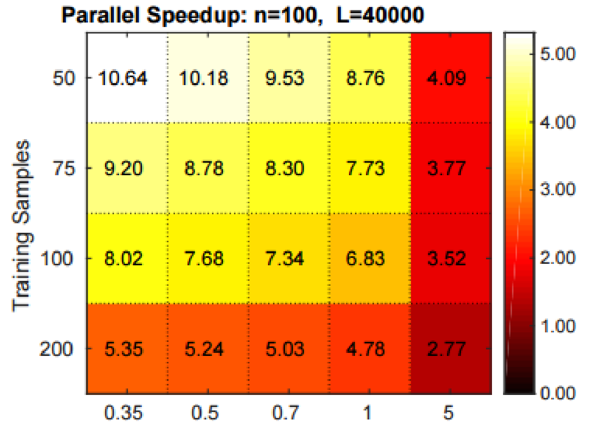


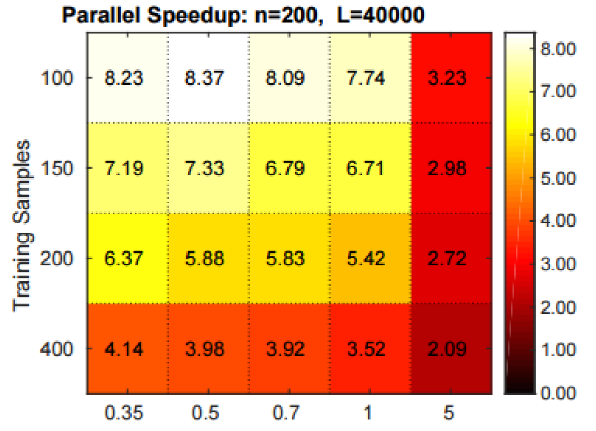


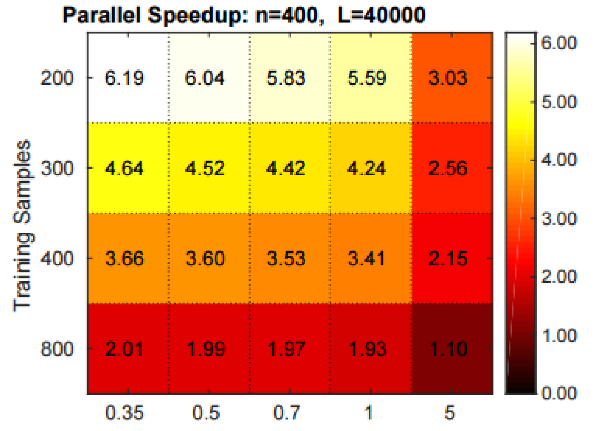




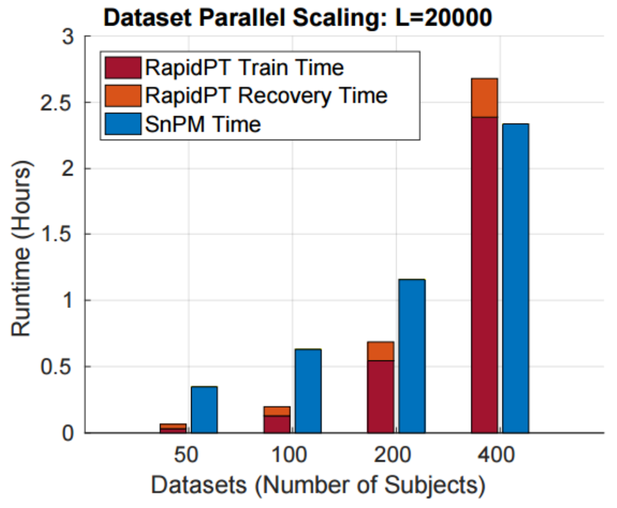


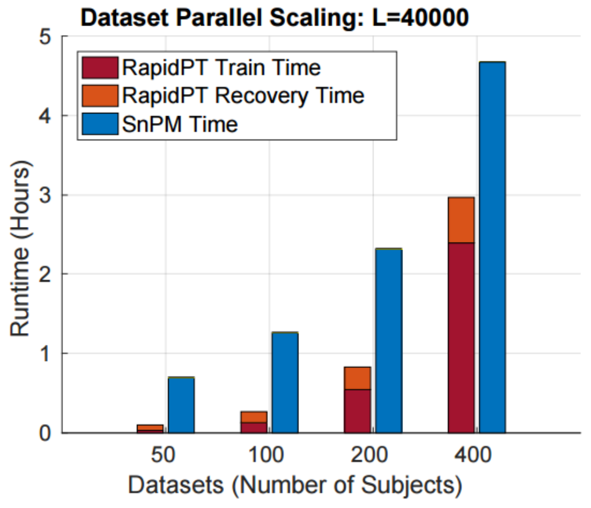


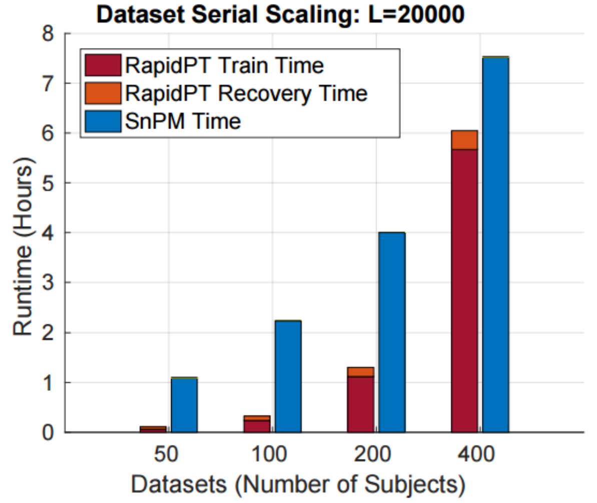


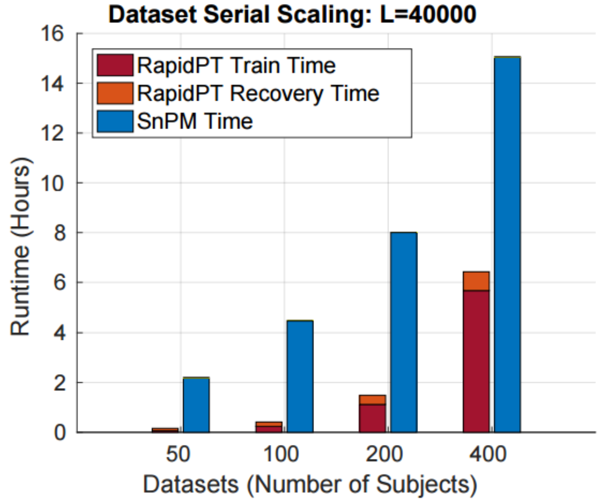


Timing RapidPT vs. SnPM

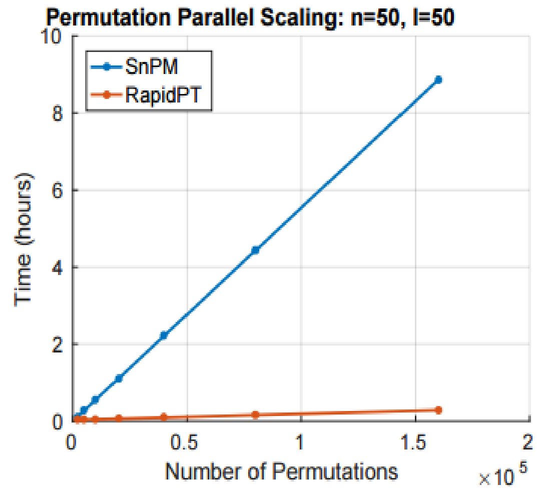


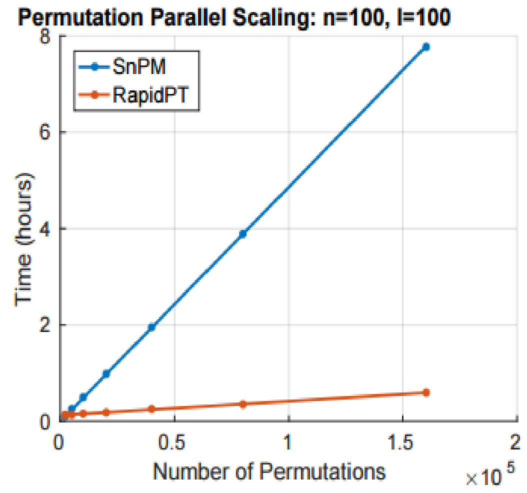


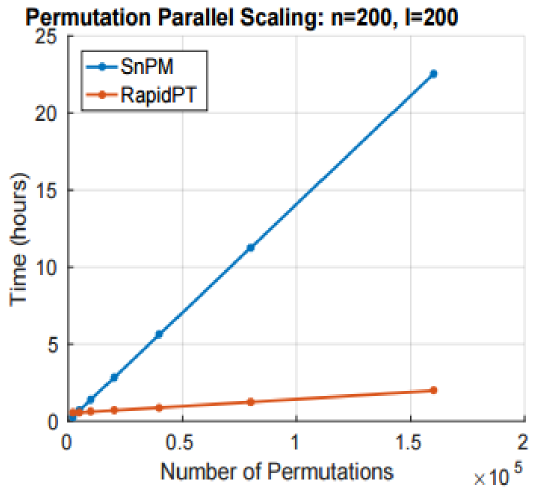


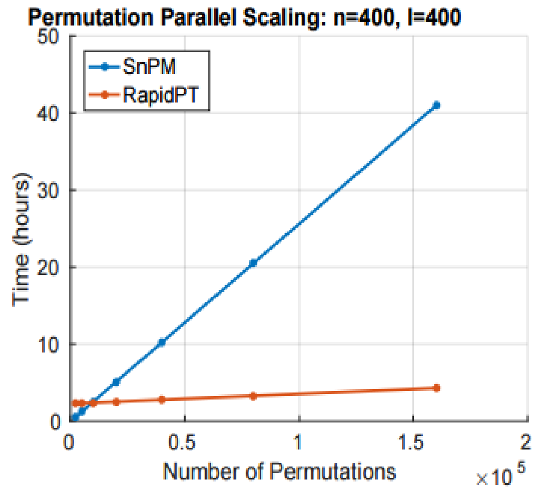


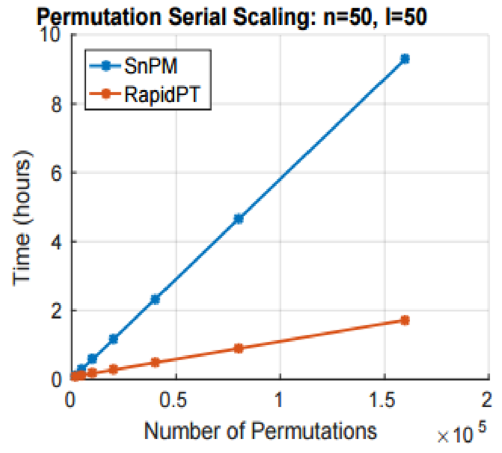
Scaling the number of permuations

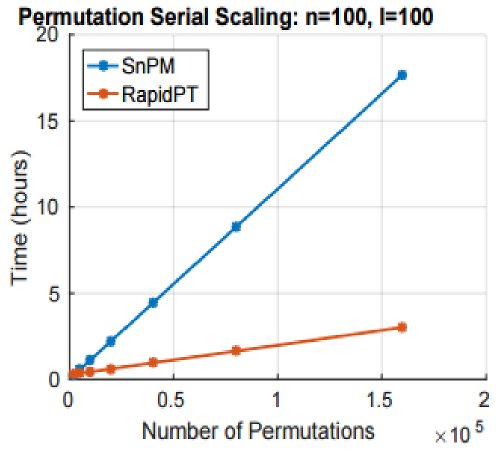


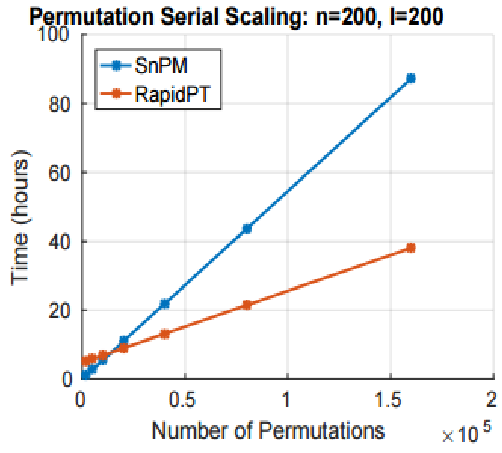


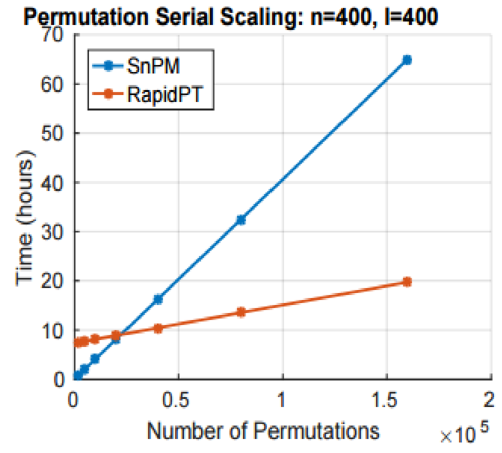










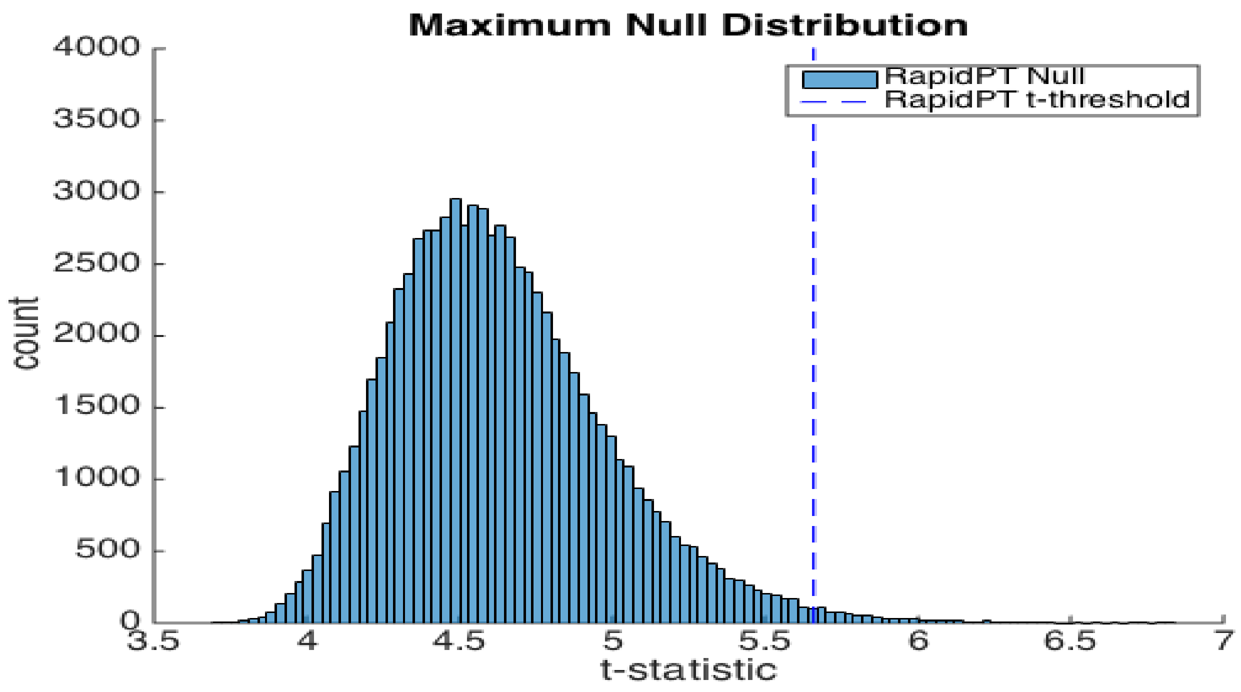


Discussion

- When to use RapidPT?
  - Many permutations. If you are doing > 10,000 permutations then RapidPT will highly benefit.
- When not to use it?
  - Permutation testing on small datasets take only a few seconds…
  - Large datasets >200 subjects, and < 10,000 permutations

RapidPT – Postprocess Example

```
% Get the outputs struct you obtained from RapidPT
load('~/PermTest/outputs/TwoSample_ADRC_200_200_400/rapidpt/outputs_80000_0
alpha = 0.01; % Significance level of 1 percent
tThresh_RapidPT = prctile(outputs.maxT, 100 - (100*alpha));
% Get the data
load('~/PermTest/data/ADRC/TwoSample/ADRC_400_200_200.mat');
[h,p,ci,stats]=ttest2(Data(1:200,:),Data(201:400,:),0.05,'both','unequal');
SampleMaxT = max(stats.tstat);
```



Acknoledgements and Website

- Vikas Singh
- Vamsi Ithapu
- Chris Hinrichs, Camille Maumet, Sterling C. Johnson, Thomas E. Nichols
- ADNI and ADRC

- Repository and project website:
  - https://github.com/felipegb94/RapidPT (Repository)
  - http://felipegb94.github.io/RapidPT/ (Website)