

```

1
3
5 function [pVal, tThresh] = calcTperm(Y0, Y1, X, C, dimX)
7 origTval = calcT(X, [Y0; Y1], C, dimX);
8 tic
9 SAMPLE_SIZE0 = length(Y0);
10 SAMPLE_SIZE1 = length(Y1);
11 D = [Y0; Y1];
12
13 indices = 1:SAMPLE_SIZE0+SAMPLE_SIZE1;
14
15 I1 = combnk(indices, SAMPLE_SIZE0);
16 NR_PERMS = size(I1,1);
17 I2 = zeros(NR_PERMS,SAMPLE_SIZE1);
18 tstats = zeros(NR_PERMS, 1);
19
20 D1 = D(I1);
21 M = X*pinv(X'*X)*X';
22 ImM = (eye(size(M)) - M);
23 [n, ~] = size(X);
24 invXX = pinv(X'*X);
25 invXX_X = pinv(X'*X)*X';
26
27 for i=1:NR_PERMS
28     I2(i,:) = setdiff(indices, I1(i,:));
29     D2 = D(I2(i,:));
30     %[, ~, ~, STATS]= ttest2(D1(i,:), D2);
31     %tstats(i) = STATS.tstat;
32
33     Y = [D1(i,:)'; D2];
34     %tstats(i) = calcT(X, Y, C, dimX);
35     %t = calcT(X, Y, C, dimX);
36
37     betaHat = invXX_X * Y;
38     eHat = ImM * Y;
39     variance = eHat'*eHat/(n - dimX);
40     Sb = variance * invXX;
41     tstats(i) = (C' * betaHat)/sqrt(C' * Sb * C);
42
43     %assert(t == tstats(i));
44
45 end
46
47 % c
48
49 pVal = nnz(tstats > origTval)/NR_PERMS;
50
51 % d
52 sortedTstats = sort(tstats);
53 tThresh = sortedTstats(floor(NR_PERMS * 95/100))
54
55 toc
56 end
57
58 function [pVal, tThresh] = calcTpermVect(Y0, Y1, X, C, dimX, I1, I2)
59 format long
60
61 origTval = calcT(X, [Y1; Y0], C, dimX);
62 %tic

```

```

65 D = [Y0; Y1];
67
69 NR_PERMS = size(I1,1);
%I2 = zeros(NR_PERMS,SAMPLE_SIZE1);
71
D1 = D(I1);
73 M = X*pinv(X'*X)*X';
ImM = (eye(size(M)) - M);
75 [n, ~] = size(X);
invXX = pinv(X'*X);
77 invXX_X = pinv(X'*X)*X';

79 %indicesPS = repmat(indices, NR_PERMS, 1);

81 %I2 = arrayfun(setdiff, indicesPS, I1, 'UniformOutput', true)

83 %applyToGivenRow = @(func, matrix1, matrix2) @(row) func(matrix1(row, :), matrix2(
row, :));
%applyToRows = @(func, matrix1, matrix2) arrayfun(applyToGivenRow(func, matrix1,
matrix2), 1:size(matrix1,1))'
85
% Example
87 %myMx = [1 2 3; 4 5 6; 7 8 9];
%myFunc = @sum;
89
%I2 = applyToRows(@setdiff, indicesPS, I1);
91
%I22 = arrayfun(@(i) setdiff(indicesPS(i,:), I1(i,:)), 1:size(indicesPS,1))';
93
95
97 D2 = D(I2);
YPS = [D1, D2];
99 betaHat2P = invXX_X * YPS';
eHatSP = ImM * YPS';
101 varianceP = sum(eHatSP .* eHatSP,1)'/(n - dimX);
invXXP22 = permute(repmat(invXX,[1,1,NR_PERMS]),[3 1 2]);
103 varianceP22 = repmat(varianceP, [1, 2, 2]);
Sb2P2 = permute(varianceP22 .* invXXP22, [2, 1, 3]);
105 %SbP22 = varianceP22 .* invXXP22;

107 tstatsP = (C' * betaHat2P) ./ sqrt(C' * [squeeze(Sb2P2(1,:,:))*C, squeeze(Sb2P2
(2,:,:))*C]');

109
pVal = nnz(tstatsP > origTval)/NR_PERMS;
111
% d
113 sortedTstats = sort(tstatsP);
tThresh = sortedTstats(floor(NR_PERMS * 95/100));
115
%toc
117 end
119
121 function p11()
%% a
123 SAMPLE_SIZE = 25;
MU0 = 1;
125 MU1 = 1.5;

```

```

127 MUErr0r = 0;
127 STD.DEV = 0.25;

129 % set seed for random generator
rng(1);

131 % compute the new Y
133 Y0 = MU0 + MUErr0r + STD.DEV .* randn(SAMPLE.SIZE, 1);
Y1 = MU1 + MUErr0r + STD.DEV .* randn(SAMPLE.SIZE, 1);
135
137 % estimate the new means
muEst0 = mean(Y0);
muEst1 = mean(Y1);
139
141 % estimate the new std deviations
stdDevEst0 = std(Y0);
stdDevEst1 = std(Y1);
143
145 % check that they are close to the true values
tol = 0.1;
147 assert(abs(muEst0 - MU0) < tol);
assert(abs(muEst1 - MU1) < tol);
149 assert(abs(stdDevEst0 - STD.DEV) < tol);
assert(abs(stdDevEst1 - STD.DEV) < tol);

151 %% b

153 % apply t-test, H should be 1
[H,P,CI,STATS] = ttest2(Y0, Y1);
155
157 % null should be rejected, the samples come from different distributions
assert(H == 1);

159 %% c

161 % build matrices X and Y
X = [ repmat([1 0], SAMPLE.SIZE,1); repmat([0 1], SAMPLE.SIZE,1) ];
163
Y = [Y0;Y1];
165
C = [1; -1];
167 dimXc = 2; % it is not 3, as I had it before

169 t = calcT(X, Y, C, dimXc);

171 M = calcAll(X, Y, C, dimXc);

173 % xi
betaTrue = [1; 1.5];
175
eTrue = Y - X*betaTrue;
177 % projection of e onto C(X)
eX = M * eTrue;

179 % xii
% projection of e onto error space
181 eE = (eye(size(M)) - M) * eTrue;

183 %% d

185 % X = 3x50, column space dim(X) = 2
X = [ repmat([1 1 0], SAMPLE.SIZE,1); repmat([1 0 1], SAMPLE.SIZE,1) ];
187
C = [0; 1; -1];
189 dimXd = 2;

```

```

    calcAll(X, Y, C, dimXd);
191
%% e
193
% X = 2x50, column space dim(X) = 2
195 X = [repmat([1 1], SAMPLE_SIZE,1); repmat([1 0], SAMPLE_SIZE,1)];

197 C = [0; 1];
    dimXe = 2;
199 calcAll(X, Y, C, dimXe);

201

203 end

205
function [M, t] = calcAll(X, Y, C, dimX)
207
    tol = 0.00001;
209 % ii
    M = X*pinv(X'*X)*X';
211
    % iii
213 Yhat = M * Y;

215 eHat = (eye(size(M)) - M) * Y;

217 % cosine is almost zero, suggesting the vectors are perpendicular
    cosYe = sum(Yhat' * eHat)/(norm(Yhat)*norm(eHat))
219
    assert(abs(cosYe) < tol);
221
    % iv
223 betaHat = pinv(X'*X)*X' * Y;

225 % v
    [n, ~] = size(X);
227 variance = eHat'*eHat/(n - dimX);

229 % vi

231 Sb = variance * pinv(X'*X)

233 std1 = sqrt(Sb(1,1));
    std2 = sqrt(Sb(2,2));
235
    % vii
237 U = null(C');

239 X0 = X * U;

241 % viii

243 M0 = X0*pinv(X0'*X0)*X0';

245 Yhat0 = M0 * Yhat;
    betaHat0 = pinv(X0'*X0)*X0' * Y;
247
    r = 1;
249
    YhatC = norm(Yhat - Yhat0); % additional error
251
    F = (norm(Yhat - Yhat0)^2 / r) / variance;
253

```

```

% ix
255 %C = [1; -1]
t = (C' * betaHat)/sqrt(C' * Sb * C);
257
%assert(abs(t^2 - F) < tol);
259
end
261

263 function p12()

%% a
265 SAMPLE_SIZE = 25;
267 MU0 = 1;
MU1 = 1.5;
269 MUErrror = 0;
STD.DEV = 0.25;
271
% set seed for random generator
273 rng(1);

275 % compute the new Y
Y0 = MU0 + MUErrror + STD.DEV .* randn(SAMPLE_SIZE, 1);
277 Y1 = MU1 + MUErrror + STD.DEV .* randn(SAMPLE_SIZE, 1);

279 Y = [Y0;Y1];

281 % apply t-test, H should be 1
[H,P,CI,STATS] = ttest(Y0, Y1);
283
% null should be rejected, the samples come from different distributions
285 assert(H == 1);

287 X = [ repmat([1 1], SAMPLE_SIZE,1); repmat([1 0], SAMPLE_SIZE,1) ];

289 S = [ eye(SAMPLE_SIZE); eye(SAMPLE_SIZE) ];

291 X = [X, S];

293 C = zeros(SAMPLE_SIZE + 2, 1);
C(2) = 1;
295
dimX = 26; % because one dimension is lost due to the contrast
297
[M, t] = calcAll(X, Y, C, dimX)
299
end
301
function p12testT()
303
%% a
305 SAMPLE_SIZE = 25;
MU0 = 1;
307 MU1 = 1.5;
MUErrror = 0;
309 STD.DEV = 0.25;

311 % set seed for random generator
rng(1);
313 tvals = zeros(100,1);
tvals2 = zeros(100,1);
315 for i=1:100

317 % compute the new Y

```

```

319 Y0 = MU0 + MUErrors + STD_DEV .* randn(SAMPLE_SIZE, 1);
    Y1 = MU1 + MUErrors + STD_DEV .* randn(SAMPLE_SIZE, 1);

321 Y = [Y0;Y1];

323 Ycentered = Y - mean(Y);

325 % apply t-test , H should be 1
    [H,P,CI,STATS] = ttest(Y0, Y1);
327 [H,P,CI,STATS2] = ttest2(Y0, Y1);

329 tvals(i) = STATS.tstat;
    tvals2(i) = STATS2.tstat;
331
end
333 end

335 function p21()

337 %% a
    SAMPLE_SIZE0 = 6;
339 SAMPLE_SIZE1 = 8;
    MU0 = 1;
341 MU1 = 1.5;
    MUErrors = 0;
343 STD_DEV = 0.25;

345 % set seed for random generator
    rng(1);

347 % compute the new Y
349 Y0 = MU0 + MUErrors + STD_DEV .* randn(SAMPLE_SIZE0, 1);
    Y1 = MU1 + MUErrors + STD_DEV .* randn(SAMPLE_SIZE1, 1);
351
353 %% a
    % apply t-test , H should be 1
355 [H,P,CI,STATS] = ttest2(Y1, Y0);

357 Tval = STATS.tstat
359 %% b
    D = [Y0; Y1];

361
363 indices = 1:SAMPLE_SIZE0+SAMPLE_SIZE1;

365 I1 = combnk(indices , 6);
    NR_PERMS = size(I1,1);
    I2 = zeros(NR_PERMS,8);
367 tstats = zeros(NR_PERMS, 1);
    meanDiffs = zeros(NR_PERMS, 1);
369
    D1 = D(I1);
371
373 for i=1:NR_PERMS
    I2(i,:) = setdiff(indices , I1(i,:));
375 D2 = D(I2(i,:));
    [~,~,~,STATS]= ttest2(D1(i,:), D2);
377 tstats(i) = STATS.tstat;

379 % c
    meanDiffs(i) = mean(D1(i,:)) - mean(D2);
381 end

```

```

383 % p-value using the t-statistic
pVal = nnz(tstats > Tval)/NR_PERMS;
385
hTstats = histogram(tstats,100);
387 xlabel('empirical distribution of the t statistic')
saveas(hTstats, 'report/figures/p21-b.eps');
389
%% c
391
meansDiffOrig = mean(Y1) - mean(Y0);
393
% p-value using the difference in group means as the statistic
395 pValMeans = nnz(meanDiffs > meansDiffOrig)/NR_PERMS;

397 hMeansStats = histogram(meanDiffs,100);
xlabel('difference of means statistic')
399 saveas(hMeansStats, 'report/figures/p21-c.eps');

401 %% d

403 % i
tstatsD = zeros(NR_PERMS, 1);
405 NR_PERMS_RAND = 1000;
perms = zeros(NR_PERMS_RAND,SAMPLE_SIZE0 + SAMPLE_SIZE1);
407
for i=1:NR_PERMS_RAND
409     perms(i,:) = randperm(SAMPLE_SIZE0 + SAMPLE_SIZE1);
D1 = D(perms(i,1:SAMPLE_SIZE0));
411 D2 = D(perms(i,SAMPLE_SIZE0+1:end));
[~,~,~, STATS]= ttest2(D1, D2);
413 tstatsD(i) = STATS.tstat;
end
415
% p-value approximation using a random sapling of 1000 permutations
417 pValD = nnz(tstatsD > Tval)/NR_PERMS_RAND; % p-value is zero for 1,000 runs, 3e-04
for 10,000 runs

419 % iii

421 dup_nr = 0;
for i=1:NR_PERMS_RAND
423     i
for j=i+1:NR_PERMS_RAND
425         if (permsEqual(perms(i,:), perms(j,:), SAMPLE_SIZE0))
dup_nr = dup_nr + 1;
427         fprintf('i:%d j:%d', i, j);
break;
429         end
end
431 end

433 % number of duplicate permutations
dup_nr
435
end

437
function eq = permsEqual(perm1, perm2, size1)
439
diffGroup1 = sum(abs(sort(perm1(1:size1)) - sort(perm2(1:size1))));
441 diffGroup2 = sum(abs(sort(perm1(size1+1:end)) - sort(perm2(size1+1:end))));

443 eq = (diffGroup1 + diffGroup2) == 0;
end

```

```

445 function p22()
447
448 RES = 40;
449 SUBJECTS = 8;
450 CPAdata = zeros(SUBJECTS, RES, RES, RES);
451 PPAdata = zeros(SUBJECTS, RES, RES, RES);
452
453 cpaI = [4,5,6,7,8,9,10,11];
454 ppaI = [3,6,9,10,13,14,15,16];
455
456 for s=1:SUBJECTS
457     filename = sprintf('glm/CPA%d_diffeo_fa.img', cpaI(s));
458     fid = fopen(filename, 'r', 'l'); % little-endian
459     data = fread(fid, 'float'); % 16-bit floating point
460     CPAdata(s, :, :, :) = reshape(data, [40 40 40]); % dimension 40x40x40
461
462     filename = sprintf('glm/PPA%d_diffeo_fa.img', ppaI(s));
463     fid = fopen(filename, 'r', 'l'); % little-endian
464     data = fread(fid, 'float'); % 16-bit floating point
465     PPAdata(s, :, :, :) = reshape(data, [40 40 40]); % dimension 40x40x40
466 end
467
468 fid = fopen('glm/wm_mask.img', 'r', 'l'); % little-endian
469 data = fread(fid, 'float'); % 16-bit floating point
470 wm_mask = reshape(data, [40 40 40]); % dimension 40x40x40
471
472 % a
473 [tVals, maxT] = partA(CPAdata, PPAdata, wm_mask, SUBJECTS, RES);
474
475 % b
476
477 % [pVals, maxP] = partB(CPAdata, PPAdata, wm_mask, SUBJECTS, RES);
478
479 [pVals, pVal, tThresh] = partBv2(CPAdata, PPAdata, wm_mask, SUBJECTS, RES);
480
481 plot_graphs()
482
483 end
484
485 function [tVals, maxT] = partA(CPAdata, PPAdata, wm_mask, SUBJECTS, RES)
486
487 X = [repmat([1 0], SUBJECTS, 1); repmat([0 1], SUBJECTS, 1)];
488
489 C = [1; -1];
490 dimX = 2;
491
492 tVals = zeros(RES, RES, RES);
493 matlabTVals = zeros(RES, RES, RES);
494 for i=1:RES
495     i
496     for j=1:RES
497         for k=1:RES
498             if (wm_mask(i, j, k) == 1)
499                 tic
500                 Y = [CPAdata(:, i, j, k); PPAdata(:, i, j, k)];
501                 tVals(i, j, k) = calcT(X, Y, C, dimX);
502
503                 [~,~,~,STATS] = ttest2(CPAdata(:, i, j, k), PPAdata(:, i, j, k));
504                 matlabTVals(i, j, k) = STATS.tstat;
505                 assert(abs(tVals(i, j, k) - matlabTVals(i, j, k)) < 0.00001);
506                 toc
507             end
508         end
509     end
510 end

```



```

509     end
510 end
511
512 save('tVals.mat', 'tVals', 'matlabTVals');
513
514 maxT = max(tVals(:));
515
516 end
517
518 function [pVals, maxP] = partB(CPAdat, PPAdat, wm_mask, SUBJECTS, RES)
519
520 %RES = 2;
521
522 X = [ repmat([1 0], SUBJECTS,1); repmat([0 1], SUBJECTS,1) ];
523
524 C = [1; -1];
525 dimX = 2;
526
527 pVals = zeros(RES, RES, RES);
528 tThresh = zeros(RES, RES, RES);
529 matlabPVals = zeros(RES, RES, RES);
530
531 SAMPLE_SIZE0 = 8;
532 SAMPLE_SIZE1 = 8;
533
534 indices = 1:SAMPLE_SIZE0+SAMPLE_SIZE1;
535
536 I1 = combnk(indices, SAMPLE_SIZE0);
537 NR_PERMS = size(I1,1);
538
539 I2 = zeros(NR_PERMS, SAMPLE_SIZE1);
540 for i=1:NR_PERMS
541     I2(i,:) = setdiff(indices, I1(i,:));
542 end
543
544
545 for i=1:RES
546     i
547     for j=1:RES
548         for k=1:RES
549             if (wm_mask(i,j,k) == 1)
550                 %tic
551                 Y0 = CPAdat(:, i, j, k);
552                 Y1 = PPAdat(:, i, j, k);
553
554                 [pVals(i,j,k), tThresh(i,j,k)] = calcTpermVect(Y0, Y1, X, C, dimX, I1, I2);
555
556                 [~, matlabPVals(i,j,k)] = ttest2(CPAdat(:, i, j, k), PPAdat(:, i, j, k));
557                 %toc
558             end
559         end
560     end
561 end
562
563 maxP = max(pVals(:));
564 save('pValsPerm.mat', 'pVals', 'matlabPVals', 'tThresh', 'maxP');
565
566 end
567
568 function [maxTs, pVal, tThresh] = partBv2(CPAdat, PPAdat, wm_mask, SUBJECTS, RES)
569
570 %RES = 2;

```

```

573 X = [ repmat([1 0], SUBJECTS,1); repmat([0 1], SUBJECTS,1) ];
575 C = [1; -1];
    dimX = 2;
577
SAMPLE_SIZE0 = 8;
579 SAMPLE_SIZE1 = 8;

581 indices = 1:SAMPLE_SIZE0+SAMPLE_SIZE1;

583 % make the permutations
    I0 = combnk(indices, SAMPLE_SIZE0);
585 %D0 = combnk(D, SAMPLE_SIZE0);
    NR_PERMS = size(I0,1);
587
    I1 = zeros(NR_PERMS,SAMPLE_SIZE1);
589 for i=1:NR_PERMS
        I1(i,:) = setdiff(indices, I0(i,:));
591 end

593 D0 = reshape(CPAdat, [SAMPLE_SIZE0 RES^3])';
    D1 = reshape(PPAdat, [SAMPLE_SIZE1 RES^3])';
595
    D = [D0, D1];
597 mask_lin= reshape(wm_mask, [1 RES^3]);

599 % b
    maxTs = zeros(NR_PERMS,1);
601 %NR_PERMS = 10;
    for p=1:NR_PERMS
603         p
            %ind0 = repmat(I0(p,:), [RES^3 1]);
605             %ind1 = repmat(I1(p,:), [RES^3 1]);

607         maxTs(p) = calcMaxTImages(D(:,I0(p,:)), D(:,I1(p,:)), mask_lin, X, C, dimX);
    end
609
    maxTOrig = calcMaxTImages(D0, D1, mask_lin, X, C, dimX);
611 % c
    pVal = nnz(maxTs > maxTOrig)/NR_PERMS;
613
    % d
615 maxTsSorted = sort(maxTs);
    tThresh = maxTsSorted(floor(NR_PERMS * 95/100));
617
    save('pValsPerm.mat', 'maxTs', 'pVal', 'tThresh', 'maxTOrig');
619
    end
621
function plot_graphs()
623
    load('tVals.mat')
625 maxT = max(tVals(:));
    load('pValsPerm.mat')
627 hMaxTs = histogram(maxTs,100);

629 xlabel('maximum T statistic')

631 hold on
    SP=maxT; %your point goes here
633 plot([SP SP],[0 700], 'r—o')
    hold on
635 SP=tThresh; %your point goes here
    plot([SP SP],[0 700], 'g—*')

```

```

637 legend('maximum t-statistic for different permutations','maximum t-statistic among
        all voxels', 't-statistic threshold for p-value=5%', 'Location','northoutside')
639
640 set(gca, 'FontSize', 11);
641 %set(gca, 'Position', [100 100 800 600]);
642
643 %saveTightFigure(hMaxTs, 'report/figures/p22_b.eps');
644 %saveas(hMaxTs, 'report/figures/p22_b.eps');
645
646 end
647
648
649 function maxT = calcMaxTImages(D0, D1, wm_mask, X, C, dimX)
650
651
652 NR_PERMS = size(D0, 1);
653
654 M = X*pinv(X'*X)*X';
655 ImM = (eye(size(M)) - M);
656 [n, ~] = size(X);
657 invXX = pinv(X'*X);
658 invXX_X = pinv(X'*X)*X';
659
660 % prefixes: P - dimension of pixels/voxels, S - dimension of samples
661 YPS = [D0, D1];
662 betaHat2P = invXX_X * YPS';
663 eHatSP = ImM * YPS';
664 varianceP = sum(eHatSP .* eHatSP, 1)' / (n - dimX);
665 invXXP22 = permute(repmat(invXX, [1, 1, NR_PERMS]), [3 1 2]);
666 varianceP22 = repmat(varianceP, [1, 2, 2]);
667 Sb2P2 = permute(varianceP22 .* invXXP22, [2, 1, 3]);
668 %SbP22 = varianceP22 .* invXXP22;
669
670 tstatsP = (C' * betaHat2P) ./ sqrt(C' * [squeeze(Sb2P2(1, :, :))*C, squeeze(Sb2P2
        (2, :, :))*C]');
671
672 maxT = max(tstatsP .* wm_mask);
673
674 % pVal = nnz(tstatsP > origTval)/NR_PERMS;
675 %
676 % % d
677 % sortedTstats = sort(tstatsP);
678 % tThresh = sortedTstats(floor(NR_PERMS * 95/100));
679
680 %toc
681 end
682
683 function t = calcT(X, Y, C, dimX)
684
685 M = X*pinv(X'*X)*X';
686
687 betaHat = pinv(X'*X)*X' * Y;
688
689 eHat = (eye(size(M)) - M) * Y;
690 [n, ~] = size(X);
691 variance = eHat'*eHat/(n - dimX);
692 Sb = variance * pinv(X'*X);
693
694
695 t = (C' * betaHat)/sqrt(C' * Sb * C);
696
697 end

```

```

699
701 function saveTightFigure(h,outfilename)
% SAVETIGHTFIGURE(H,OUTFILENAME) Saves figure H in file OUTFILENAME without
703 % the white space around it.
%
705 % by ‘‘a grad student’’
% http://tipstrickshowtos.blogspot.com/2010/08/how-to-get-rid-of-white-margin-in.html
707
% get the current axes
709 ax = get(h, 'CurrentAxes');
711
% make it tight
ti = get(ax, 'TightInset');
713 set(ax, 'Position', [ti(1) ti(2) 1-ti(3)-ti(1) 1-ti(4)-ti(2)]);
715
% adjust the papersize
set(ax, 'units', 'centimeters');
717 pos = get(ax, 'Position');
ti = get(ax, 'TightInset');
719 set(h, 'PaperUnits', 'centimeters');
set(h, 'PaperSize', [pos(3)+ti(1)+ti(3) pos(4)+ti(2)+ti(4)]);
721 set(h, 'PaperPositionMode', 'manual');
set(h, 'PaperPosition', [0 0 pos(3)+ti(1)+ti(3) pos(4)+ti(2)+ti(4)]);
723
% save it
725 saveas(h,outfilename);
function unitTest()
727
SAMPLE_SIZE = 8;
729 MU0 = 1;
MU1 = 1.5;
731 MUErrror = 0;
STD_DEV = 0.25;
733
735 X = [repmat([1 0], SAMPLE_SIZE,1); repmat([0 1], SAMPLE_SIZE,1)];
737 C = [1; -1];
dimX = 2;
739
for i=1:10
741 i
Y0 = MU0 + MUErrror + STD_DEV .* randn(SAMPLE_SIZE, 1);
743 Y1 = MU1 + MUErrror + STD_DEV .* randn(SAMPLE_SIZE, 1);
745 tic
ans1 = calcTperm(Y0, Y1, X, C, dimX);
747 toc
749 tic
ans2 = calcTpermVect(Y0, Y1, X, C, dimX);
751 toc
753 assert(sum(abs(ans1 - ans2)) < 0.000000001);
755 end
757
759 end

```