```matlab



function [pVal, tThresh] = calcTperm(Y0, Y1, X, C, dimX)

origTval = calcT(X, [Y0; Y1], C, dimX);
tic
SAMPLE_SIZE0 = length(Y0);
SAMPLE_SIZE1 = length(Y1);

D = [Y0; Y1];

indices = 1:SAMPLE_SIZE0+SAMPLE_SIZE1;

I1 = combnk(indices, SAMPLE_SIZE0);
NR_PERMS = size(I1,1);
I2 = zeros(NR_PERMS,SAMPLE_SIZE1);
tstats = zeros(NR_PERMS, 1);

D1 = D(I1);
M = X*pinv(X'*X)*X';
ImM = (eye(size(M)) - M);
[n, ~] = size(X);
invXX = pinv(X'*X);
invXX_X = pinv(X'*X)*X';

for i=1:NR_PERMS
  I2(i,:) = setdiff(indices, I1(i,:));
  D2 = D(I2(i,:));
  %[~, ~, ~, STATS]= ttest2(D1(i,:), D2);
  %tstats(i) = STATS.tstat;

  Y = [D1(i,:)'; D2];
  %tstats(i) = calcT(X, Y, C, dimX);
  %t = calcT(X, Y, C, dimX);

  betaHat = invXX_X * Y;
  eHat = ImM * Y;
  variance = eHat'*eHat/(n - dimX);
  Sb = variance * invXX;
  tstats(i)  = (C' * betaHat)/sqrt(C' * Sb * C);

  %assert(t == tstats(i) );

end

% c

pVal = nnz(tstats > origTval)/NR_PERMS;

%  d
sortedTstats = sort(tstats);
tThresh = sortedTstats(floor(NR_PERMS * 95/100))

toc
end

function [pVal, tThresh] = calcTpermVect(Y0, Y1, X, C, dimX, I1, I2)
format long

origTval = calcT(X, [Y1; Y0], C, dimX);
%tic
```

```

D = [Y0; Y1];


NR_PERMS = size(I1,1);
%I2 = zeros(NR_PERMS,SAMPLE_SIZE1);

D1 = D(I1);
M = X*pinv(X'*X)*X';
ImM = (eye(size(M)) - M);
[n, ~] = size(X);
invXX = pinv(X'*X);
invXX_X = pinv(X'*X)*X';

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

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

%applyToGivenRow = @(func, matrix1, matrix2) @(row) func(matrix1(row, :), matrix2(row, :));
%applyToRows = @(func, matrix1, matrix2) arrayfun(applyToGivenRow(func, matrix1, matrix2), 1:size(matrix1,1))'

% Example
%myMx = [1 2 3; 4 5 6; 7 8 9];
%myFunc = @sum;

%I2 = applyToRows(@setdiff, indicesPS, I1);

%I22 = arrayfun(@(i) setdiff(indicesPS(i,:),I1(i,:)),1:size(indicesPS,1))';




D2 = D(I2);
YPS = [D1, D2];
betaHat2P = invXX_X * YPS';
eHatSP = ImM * YPS';
varianceP = sum(eHatSP .* eHatSP,1)'/(n - dimX);
invXXP22 = permute(repmat(invXX,[1,1,NR_PERMS]),[3 1 2]);
varianceP22 = repmat(varianceP, [1, 2, 2]);
Sb2P2 = permute(varianceP22 .* invXXP22, [2, 1, 3]);
%SbP22 = varianceP22 .* invXXP22;

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


pVal = nnz(tstatsP > origTval)/NR_PERMS;

%  d
sortedTstats = sort(tstatsP);
tThresh = sortedTstats(floor(NR_PERMS * 95/100));

%toc
end



function p11()
%% a
SAMPLE_SIZE = 25;
MU0 = 1;
MU1 = 1.5;
```

```matlab
MUError = 0;
STD_DEV = 0.25;

% set seed for random generator
rng(1);

% compute the new Y
Y0 = MU0 + MUError + STD_DEV .* randn(SAMPLE_SIZE, 1);
Y1 = MU1 + MUError + STD_DEV .* randn(SAMPLE_SIZE, 1);

% estimate the new means
muEst0 = mean(Y0);
muEst1 = mean(Y1);

% estimate the new std deviations
stdDevEst0 = std(Y0);
stdDevEst1 = std(Y1);

% check that they are close to the true values
tol = 0.1;
assert(abs(muEst0 - MU0) < tol);
assert(abs(muEst1 - MU1) < tol);
assert(abs(stdDevEst0 - STD_DEV) < tol);
assert(abs(stdDevEst1 - STD_DEV) < tol);

%% b

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

% null should be rejected, the samples come from different distributions
assert(H == 1);

%% c

% build matrices X and Y
X = [repmat([1 0], SAMPLE_SIZE,1); repmat([0 1], SAMPLE_SIZE,1)];

Y = [Y0;Y1];

C = [1; -1];
dimXc = 2; % it is not 3, as I had it before

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

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

% xi
betaTrue = [1; 1.5];

eTrue = Y - X*betaTrue;
% projection of e onto C(X)
eX = M * eTrue;
% xii
% projection of e onto error space
eE = (eye(size(M)) - M) * eTrue;

%% d

% X = 3x50, column space dim(X) = 2
X = [repmat([1 1 0], SAMPLE_SIZE,1); repmat([1 0 1], SAMPLE_SIZE,1)];

C = [0; 1; -1];
dimXd = 2;
```

```
calcAll(X, Y, C, dimXd);

%% e

%  X = 2x50, column space dim(X) = 2
X = [repmat([1 1], SAMPLE_SIZE,1); repmat([1 0], SAMPLE_SIZE,1)];

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



end


function [M, t] = calcAll(X, Y, C, dimX)

tol = 0.00001;
% ii
M = X*pinv(X'*X)*X';

% iii
Yhat = M * Y;

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

% cosine is almost zero, suggesting the vectors are perpendicular
cosYe = sum(Yhat' * eHat)/(norm(Yhat)*norm(eHat))

assert(abs(cosYe) < tol);

% iv
betaHat = pinv(X'*X)*X' * Y;

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

% vi

Sb = variance * pinv(X'*X)

std1 = sqrt(Sb(1,1));
std2 = sqrt(Sb(2,2));

% vii
U = null(C');

X0 = X * U;

% viii

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

Yhat0 = M0 * Yhat;
betaHat0 = pinv(X0'*X0)*X0' * Y;

r = 1;

YhatC = norm(Yhat - Yhat0); % additional error

F = (norm(Yhat - Yhat0)^2 / r) / variance;

```

```
% ix
%C = [1; -1]
t = (C' * betaHat)/sqrt(C' * Sb * C);

%assert(abs(t^2 - F) < tol);

end


function p12()

%% a
SAMPLE_SIZE = 25;
MU0 = 1;
MU1 = 1.5;
MUError = 0;
STD_DEV = 0.25;

% set seed for random generator
rng(1);

% compute the new Y
Y0 = MU0 + MUError + STD_DEV .* randn(SAMPLE_SIZE, 1);
Y1 = MU1 + MUError + STD_DEV .* randn(SAMPLE_SIZE, 1);

Y = [Y0;Y1];

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

% null should be rejected, the samples come from different distributions
assert(H == 1);

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

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

X = [X, S];

C = zeros(SAMPLE_SIZE + 2, 1);
C(2) = 1;

dimX = 26; % because one dimension is lost due to the contrast

[M, t] = calcAll(X, Y, C, dimX)

end

function p12testT()

%% a
SAMPLE_SIZE = 25;
MU0 = 1;
MU1 = 1.5;
MUError = 0;
STD_DEV = 0.25;

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

  % compute the new Y
  Y0 = MU0 + MUError + STD_DEV .* randn(SAMPLE_SIZE, 1);
  Y1 = MU1 + MUError + STD_DEV .* randn(SAMPLE_SIZE, 1);

  Y = [Y0;Y1];

   Ycentered = Y - mean(Y);

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

   tvals(i) = STATS.tstat;
   tvals2(i) = STATS2.tstat;

end
end

function p21()

%% a
SAMPLE_SIZE0 = 6;
SAMPLE_SIZE1 = 8;
MU0 = 1;
MU1 = 1.5;
MUError = 0;
STD_DEV = 0.25;

% set seed for random generator
rng(1);

% compute the new Y
Y0 = MU0 + MUError + STD_DEV .* randn(SAMPLE_SIZE0, 1);
Y1 = MU1 + MUError + STD_DEV .* randn(SAMPLE_SIZE1, 1);

%% a

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

Tval = STATS.tstat
%% b

D = [Y0; Y1];

indices = 1:SAMPLE_SIZE0+SAMPLE_SIZE1;

I1 = combnk(indices, 6);
NR_PERMS = size(I1,1);
I2 = zeros(NR_PERMS,8);
tstats = zeros(NR_PERMS, 1);
meanDiffs = zeros(NR_PERMS, 1);

D1 = D(I1);


for i=1:NR_PERMS
  I2(i,:) = setdiff(indices, I1(i,:));
  D2 = D(I2(i,:));
  [~, ~, ~, STATS]= ttest2(D1(i,:), D2);
  tstats(i) = STATS.tstat;

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

```
% p-value using the t-statistic
pVal = nnz(tstats > Tval)/NR_PERMS;

hTstats = histogram(tstats,100);
xlabel('empirical distribution of the t statistic')
saveas(hTstats, 'report/figures/p21_b.eps');

%% c

meansDiffOrig = mean(Y1) - mean(Y0);

% p-value using the difference in group means as the statistic
pValMeans = nnz(meanDiffs > meansDiffOrig)/NR_PERMS;

hMeansStats = histogram(meanDiffs,100);
xlabel('difference of means statistic')
saveas(hMeansStats, 'report/figures/p21_c.eps');

%% d

% i
tstatsD = zeros(NR_PERMS, 1);
NR_PERMS_RAND = 1000;
perms = zeros(NR_PERMS_RAND,SAMPLE_SIZE0 + SAMPLE_SIZE1);

for i=1:NR_PERMS_RAND
  perms(i,:) = randperm(SAMPLE_SIZE0 + SAMPLE_SIZE1);
  D1 = D(perms(i,1:SAMPLE_SIZE0));
  D2 = D(perms(i,SAMPLE_SIZE0+1:end));
  [~, ~, ~, STATS]= ttest2(D1, D2);
  tstatsD(i) = STATS.tstat;
end

% p-value approximation using a random sapling of 1000 permutations
pValD = nnz(tstatsD > Tval)/NR_PERMS_RAND; % p-value is zero for 1,000 runs, 3e-04 for 10,000 runs

% iii

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

% number of duplicate permutations
dup_nr

end

function eq = permsEqual(perm1, perm2, size1)

diffGroup1 = sum(abs(sort(perm1(1:size1)) - sort(perm2(1:size1))));
diffGroup2 = sum(abs(sort(perm1(size1+1:end)) - sort(perm2(size1+1:end))));

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

```

function p22()

RES = 40;
SUBJECTS = 8;
CPAdata = zeros(SUBJECTS, RES, RES, RES);
PPAdata = zeros(SUBJECTS, RES, RES, RES);

cpaI = [4,5,6,7,8,9,10,11];
ppaI = [3,6,9,10,13,14,15,16];

for s=1:SUBJECTS
  filename = sprintf('glm/CPA%d_diffeo_fa.img', cpaI(s));
  fid = fopen(filename, 'r', 'l'); % little-endian
  data = fread(fid, 'float'); % 16-bit floating point
  CPAdata(s,:,:,:) = reshape(data, [40 40 40]); % dimension 40x40x40

  filename = sprintf('glm/PPA%d_diffeo_fa.img', ppaI(s));
  fid = fopen(filename, 'r', 'l'); % little-endian
  data = fread(fid, 'float'); % 16-bit floating point
  PPAdata(s,:,:,:) = reshape(data, [40 40 40]); % dimension 40x40x40
end

fid = fopen('glm/wm_mask.img', 'r', 'l'); % little-endian
data = fread(fid, 'float'); % 16-bit floating point
wm_mask = reshape(data, [40 40 40]); % dimension 40x40x40

% a
[tVals, maxT] = partA(CPAdata, PPAdata, wm_mask, SUBJECTS, RES);

% b

%[pVals, maxP] = partB(CPAdata, PPAdata, wm_mask, SUBJECTS, RES);

[pVals, pVal, tThresh] = partBv2(CPAdata, PPAdata, wm_mask, SUBJECTS, RES);

plot_graphs()

end

function [tVals, maxT] = partA(CPAdata, PPAdata, wm_mask, SUBJECTS, RES)

X = [repmat([1 0], SUBJECTS,1); repmat([0 1], SUBJECTS,1)];

C = [1; -1];
dimX = 2;

tVals = zeros(RES, RES, RES);
matlabTVals = zeros(RES, RES, RES);
for i=1:RES
  i
  for j=1:RES
    for k=1:RES
      if (wm_mask(i,j,k) == 1)
        tic
        Y = [CPAdata(:,i,j,k); PPAdata(:,i,j,k)];
        tVals(i,j,k) = calcT(X, Y, C, dimX);

        [~,~,~,STATS] = ttest2(CPAdata(:,i,j,k),PPAdata(:,i,j,k));
        matlabTVals(i,j,k) = STATS.tstat;
        assert(abs(tVals(i,j,k) - matlabTVals(i,j,k)) < 0.00001);
        toc
      end
    end
  end
end

save('tVals.mat', 'tVals', 'matlabTVals');

maxT = max(tVals(:));

end

function [pVals, maxP] = partB(CPAdata, PPAdata, wm_mask, SUBJECTS, RES)

%RES = 2;

X = [repmat([1 0], SUBJECTS,1); repmat([0 1], SUBJECTS,1)];

C = [1; -1];
dimX = 2;

pVals = zeros(RES, RES, RES);
tThresh = zeros(RES, RES, RES);
matlabPVals = zeros(RES, RES, RES);

SAMPLE_SIZE0 = 8;
SAMPLE_SIZE1 = 8;

indices = 1:SAMPLE_SIZE0+SAMPLE_SIZE1;

I1 = combnk(indices, SAMPLE_SIZE0);
NR_PERMS = size(I1,1);

I2 = zeros(NR_PERMS,SAMPLE_SIZE1);
for i=1:NR_PERMS
  I2(i,:) = setdiff(indices, I1(i,:));
end


for i=1:RES
  i
  for j=1:RES
    for k=1:RES
      if (wm_mask(i,j,k) == 1)
        %tic
        Y0 = CPAdata(:,i,j,k);
        Y1 = PPAdata(:,i,j,k);

        [pVals(i,j,k), tThresh(i,j,k)] = calcTpermVect(Y0, Y1, X, C, dimX, I1, I2);

        [~,matlabPVals(i,j,k)] = ttest2(CPAdata(:,i,j,k),PPAdata(:,i,j,k));
        %toc
      end
    end
  end
end

maxP = max(pVals(:));
save('pValsPerm.mat', 'pVals', 'matlabPVals', 'tThresh', 'maxP');

end


function [maxTs, pVal, tThresh] = partBv2(CPAdata, PPAdata, wm_mask, SUBJECTS, RES)

%RES = 2;
```

```
X = [repmat([1 0], SUBJECTS,1); repmat([0 1], SUBJECTS,1)];

C = [1; -1];
dimX = 2;

SAMPLE_SIZE0 = 8;
SAMPLE_SIZE1 = 8;

indices = 1:SAMPLE_SIZE0+SAMPLE_SIZE1;

% make the permutations
I0 = combnk(indices, SAMPLE_SIZE0);
%D0 = combnk(D, SAMPLE_SIZE0);
NR_PERMS = size(I0,1);

I1 = zeros(NR_PERMS,SAMPLE_SIZE1);
for i=1:NR_PERMS
  I1(i,:) = setdiff(indices, I0(i,:));
end

D0 = reshape(CPAdata, [SAMPLE_SIZE0 RES^3])';
D1 = reshape(PPAdata, [SAMPLE_SIZE1 RES^3])';

D = [D0, D1];
mask_lin= reshape(wm_mask, [1 RES^3]);

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

    maxTs(p) = calcMaxTImages(D(:,I0(p,:)), D(:,I1(p,:)), mask_lin, X, C, dimX);
end

maxTOrig = calcMaxTImages(D0, D1, mask_lin, X, C, dimX);
% c
pVal = nnz(maxTs > maxTOrig)/NR_PERMS;

% d
maxTsSorted = sort(maxTs);
tThresh = maxTsSorted(floor(NR_PERMS * 95/100));

save('pValsPerm.mat', 'maxTs', 'pVal', 'tThresh', 'maxTOrig');

end

function plot_graphs()

load('tVals.mat')
maxT = max(tVals(:));
load('pValsPerm.mat')
hMaxTs = histogram(maxTs,100);

xlabel('maximum T statistic')

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

```

legend('maximum t-statistic for different permutations','maximum t-statistic among all voxels', 't-statistic threshold for p-value=5%','Location','northoutside')

set(gca,'FontSize',11);
%set(gca, 'Position', [100 100 800 600]);

%saveTightFigure(hMaxTs, 'report/figures/p22_b.eps');
%saveas(hMaxTs, 'report/figures/p22_b.eps');


end


function maxT = calcMaxTImages(D0, D1, wm_mask, X, C, dimX)


NR_PERMS = size(D0, 1);

M = X*pinv(X'*X)*X';
ImM = (eye(size(M)) - M);
[n, ~] = size(X);
invXX = pinv(X'*X);
invXX_X = pinv(X'*X)*X';

% prefixes: P - dimension of pixels/voxels, S - dimension of samples
YPS = [D0, D1];
betaHat2P = invXX_X * YPS';
eHatSP = ImM * YPS';
varianceP = sum(eHatSP .* eHatSP,1)'/(n - dimX);
invXXP22 = permute(repmat(invXX,[1,1,NR_PERMS]),[3 1 2]);
varianceP22 = repmat(varianceP, [1, 2, 2]);
Sb2P2 = permute(varianceP22 .* invXXP22, [2, 1, 3]);
%SbP22 = varianceP22 .* invXXP22;

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

maxT = max(tstatsP .* wm_mask);

% pVal = nnz(tstatsP > origTval)/NR_PERMS;
%
% %  d
% sortedTstats = sort(tstatsP);
% tThresh = sortedTstats(floor(NR_PERMS * 95/100));

%toc
end

function t = calcT(X, Y, C, dimX)

M = X*pinv(X'*X)*X';

betaHat = pinv(X'*X)*X' * Y;

eHat = (eye(size(M)) - M) * Y;
[n, ~] = size(X);
variance = eHat'*eHat/(n - dimX);
Sb = variance * pinv(X'*X);


t = (C' * betaHat)/sqrt(C' * Sb * C);

end
```

```


function saveTightFigure(h,outfilename)
% SAVETIGHTFIGURE(H,OUTFILENAME) Saves figure H in file OUTFILENAME without
%   the white space around it.
%
% by ‘‘a grad student”
% http://tipstrickshowtos.blogspot.com/2010/08/how-to-get-rid-of-white-margin-in.html

% get the current axes
ax = get(h, 'CurrentAxes');

% make it tight
ti = get(ax,'TightInset');
set(ax,'Position',[ti(1) ti(2) 1-ti(3)-ti(1) 1-ti(4)-ti(2)]);

% adjust the papersize
set(ax,'units','centimeters');
pos = get(ax,'Position');
ti = get(ax,'TightInset');
set(h, 'PaperUnits','centimeters');
set(h, 'PaperSize', [pos(3)+ti(1)+ti(3) pos(4)+ti(2)+ti(4)]);
set(h, 'PaperPositionMode', 'manual');
set(h, 'PaperPosition',[0 0  pos(3)+ti(1)+ti(3) pos(4)+ti(2)+ti(4)]);

% save it
saveas(h,outfilename);
function unitTest()

SAMPLE_SIZE = 8;
MU0 = 1;
MU1 = 1.5;
MUError = 0;
STD_DEV = 0.25;


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

C = [1; -1];
dimX = 2;

for i=1:10
    i
    Y0 = MU0 + MUError + STD_DEV .* randn(SAMPLE_SIZE, 1);
    Y1 = MU1 + MUError + STD_DEV .* randn(SAMPLE_SIZE, 1);

    tic
    ans1 = calcTperm(Y0, Y1, X, C, dimX);
    toc

    tic
    ans2 = calcTpermVect(Y0, Y1, X, C, dimX);
    toc

    assert(sum(abs(ans1 - ans2)) < 0.000000001);

end



end
```