# COMP0118: Coursework 2

Mingzhou Hu

February 2019

## Part 1

### Q1 (abc)

```
% set random seed
rng(3)
% size of sample 1
s1_size = 25;
% size of sample 2
s2_size = 25;
% set the means to g1 and g2
mean_s1 = 1;
mean_s2 = 1.5;
% stochastic error
e1 = 0.25*randn(s1_size, 1);
e2 = 0.25*randn(s2_size, 1);
% generate datasets
sample1 = mean_s1+e1;
sample2 = mean_s2+e2;

%% 1_(a)
% computer the mean for new datasets
new_mean_s1 = mean(sample1);
new_mean_s2 = mean(sample2);
% computer the std for new datasets
new_std_s1 = std(sample1);
new_std_s2 = std(sample2);

%% 1_(b)
% perform two-sample t-test
[hyp,p,ci,stats] = ttest2(sample1, sample2);

%% 1_(c)_(i)
% generate design matrix
oness = ones(s1_size,1);
zeross = zeros(s2_size,1);
design_mat_X = [oness,zeross;zeross,oness];
```

```
% dimension of col space of design matrix
dim_X = rank(design_mat_X);

%% 1_(c)_(ii)
% perpendicular projection operator Px corresponding to any C(X)
projection_X =
design_mat_X*inv(design_mat_X'*design_mat_X)*design_mat_X';
% check idempotence
proj_id = projection_X^2;
% check symmetric
proj_sy = projection_X';

%% 1_(c)_(iii)
% both groups
bothgroups = [sample1; sample2];
% use projection_X to determine projection_Y
projection_Yhat = projection_X*bothgroups;

%% 1_(c)_(iv)
% compute Rx = I - Px
Rx = eye(size(bothgroups,1))- projection_X;
% check idempotence
Rxsq = Rx^2;
% check symmetric
Rxt = Rx';

%% 1_(c)_(v)
% determine ehat, the projection of Y into the error space
ehat = Rx*bothgroups;

%% 1_(c)_(vi)
% determine the angle between ehat and Yhat.
angle = acos(dot(ehat,projection_Yhat));

%% 1_(c)_(vii)
% use derived formula to determine betahat
betahat = inv(design_mat_X'*design_mat_X)*design_mat_X'*bothgroups;

%% 1_(c)_(vii)
% estimate the variance of the stochastic component ehat
variance = (ehat'*ehat)/48;

%% 1_(c)_(ix)
% covariance matrix of estimated model parameters
```

```
covariance_mat = variance*inv(design_mat_X'*design_mat_X);
% use covariance_mat to determine std of model parameters
std_parameters = sqrt(covariance_mat(1,1));

%% 1_(c)_(x)
% derive contrast vector for comparing group differences in means
lambda = [1 -1]';
% reduced model X0 corresponding to the null hypothesis
X0 = ones(length(bothgroups),1);

%% 1_(c)_(xi)
% recalculate new betahat
new_betahat = inv(X0'*X0)*X0'*bothgroups;
% determine error via e = Y-X0B0
new_errorhat = bothgroups-X0*new_betahat;
% additional error as a result of placing the constraint
additional_error = new_errorhat - ehat;
% projection matrix for new design mat X0
projection_X0 = X0*inv(X0'*X0)*X0';
% calculate m and n-k for f-statistic
m = trace(projection_X-projection_X0);
n_k = trace(eye(length(bothgroups))-projection_X);
% estimate f-statistic of comparing X0 to X
SSE1 = new_errorhat'*new_errorhat;
SSE2 = ehat'*ehat;
F = ((SSE1 - SSE2)/m)/(SSE2/n_k);

%% 1_(c)_(xii)
% determine t-statistic to test whether one group has higher mean
t= (lambda'*betahat)/(sqrt(lambda'*covariance_mat*lambda));

%% 1_(c)_(xiii)
% ground truth beta composed of ground truth means
beta_gt = [1 1.5]';

%% 1_(c)_(xiv)
% compute the projection of the ground truth deviation e into C(X)
egtd = bothgroups - design_mat_X*beta_gt;
```

**Q1 (def)**

```
% set random seed
rng(3)
```

```
% size of sample 1
s1_size = 25;
% size of sample 2
s2_size = 25;
% set the means to g1 and g2
mean_s1 = 1;
mean_s2 = 1.5;
% stochastic error
e1 = 0.25*randn(s1_size, 1);
e2 = 0.25*randn(s2_size, 1);
% generate datasets
sample1 = mean_s1+e1;
sample2 = mean_s2+e2;

% %% 1_(d)
% % generate design matrix
% oness = ones(s1_size,1);
% zeross = zeros(s2_size,1);
% design_mat_X = [oness,oness,zeross;oness,zeross,oness];
% % dimension of col space of design matrix
% dim_X = rank(design_mat_X);
%
% % perpendicular projection operator Px corresponding to any C(X)
% projection_X =
design_mat_X*pinv(design_mat_X'*design_mat_X)*design_mat_X';
%
% % both groups
% bothgroups = [sample1; sample2];
% % use projection_X to determine projection_Y
% projection_Yhat = projection_X*bothgroups;
%
% % compute Rx = I - Px
% Rx = eye(size(bothgroups,1))- projection_X;
%
% % determine ehat, the projection of Y into the error space
% ehat = Rx*bothgroups;
%
% % use derived formula to determine betahat
% betahat =
pinv(design_mat_X'*design_mat_X)*design_mat_X'*bothgroups;
%
% % estimate the variance of the stochastic component ehat
% variance = (ehat'*ehat)/48;
%
```

```
% % covariance matrix of estimated model parameters
% covariance_mat = variance*pinv(design_mat_X'*design_mat_X);
% % use covariance_mat to determine std of model parameters
% std_parameters = sqrt(covariance_mat(1,1));
%
% % derive contrast vector for comparing group differences in means
% lambda = [0 1 -1]';
% % reduced model X0 corresponding to the null hypothesis
% X0 = ones(length(bothgroups),1);
%
% % determine t-statistic to test whether one group has higher mean
% t= (lambda'*betahat)/(sqrt(lambda'*covariance_mat*lambda));

%% 1_(e)
% generate design matrix
oness = ones(s1_size,1);
zeross = zeros(s2_size,1);
design_mat_X = [oness,oness;oness,zeross];
% dimension of col space of design matrix
dim_X = rank(design_mat_X);

% perpendicular projection operator Px corresponding to any C(X)
projection_X =
design_mat_X*pinv(design_mat_X'*design_mat_X)*design_mat_X';

%%
% both groups
bothgroups = [sample1; sample2];
% use projection_X to determine projection_Y
projection_Yhat = projection_X*bothgroups;

% compute Rx = I - Px
Rx = eye(size(bothgroups,1))- projection_X;

% determine ehat, the projection of Y into the error space
ehat = Rx*bothgroups;


% use derived formula to determine betahat
betahat = pinv(design_mat_X'*design_mat_X)*design_mat_X'*bothgroups;

% estimate the variance of the stochastic component ehat
variance = (ehat'*ehat)/48;
```

```
% covariance matrix of estimated model parameters
covariance_mat = variance*pinv(design_mat_X'*design_mat_X);
% use covariance_mat to determine std of model parameters
std_parameters = sqrt(covariance_mat(1,1));

%% 1_(d)_(iii)
% derive contrast vector for comparing group differences in means
lambda = [0 1]';
% reduced model X0 corresponding to the null hypothesis
X0 = ones(length(bothgroups),1);

%% 1_(d)_(iv)
% determine t-statistic to test whether one group has higher mean
t= (lambda'*betahat)/(sqrt(lambda'*covariance_mat*lambda));
```

**Q2(a)**

```
% set random seed
rng(3)
% size of sample 1
s1_size = 25;
% size of sample 2
s2_size = 25;
% set the means to g1 and g2
mean_s1 = 1;
mean_s2 = 1.5;
% stochastic error
e1 = 0.25*randn(s1_size, 1);
e2 = 0.25*randn(s2_size, 1);
% generate datasets
sample1 = mean_s1+e1;
sample2 = mean_s2+e2;

% perform two-sample t-test
[hyp,p,ci,stats] = ttest(sample1, sample2);
```

**Q2(b)**

```
% set random seed
rng(3);
% size of sample 1
```

```
s1_size = 25;
% size of sample 2
s2_size = 25;
% set the means to g1 and g2
mean_s1 = 1;
mean_s2 = 1.5;
% stochastic error
e1 = 0.25*randn(s1_size, 1);
e2 = 0.25*randn(s2_size, 1);
% generate datasets
sample1 = mean_s1+e1;
sample2 = mean_s2+e2;

% perform two-sample t-test
[hyp,p,ci,stats] = ttest(sample1, sample2);

% generate design matrix
oness = ones(s1_size,1);
zeross = zeros(s2_size,1);
identity = eye(s1_size);
design_mat_X = [oness,oness,identity;oness,zeross,identity];
% dimension of design matrix
dim_X = rank(design_mat_X);

% perpendicular projection operator Px corresponding to any C(X)
projection_X =
design_mat_X*pinv(design_mat_X'*design_mat_X)*design_mat_X';

% both groups
bothgroups = [sample1; sample2];
% use projection_X to determine projection_Y
projection_Yhat = projection_X*bothgroups;

% compute Rx = I - Px
Rx = eye(size(bothgroups,1))- projection_X;

% determine ehat, the projection of Y into the error space
ehat = Rx*bothgroups;


% use derived formula to determine betahat
betahat = pinv(design_mat_X'*design_mat_X)*design_mat_X'*bothgroups;

% estimate the variance of the stochastic component ehat
```

```
variance = (ehat'*ehat)/24;

% covariance matrix of estimated model parameters
covariance_mat = variance*pinv(design_mat_X'*design_mat_X);
% use covariance_mat to determine std of model parameters
std_parameters = sqrt(covariance_mat(1,1));

% derive contrast vector for comparing group differences in means
lambda = [0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]';
% reduced model X0 corresponding to the null hypothesis
X0 = ones(length(bothgroups),1);

% determine t-statistic to test whether one group has higher mean
t= (lambda'*betahat)/(sqrt(lambda'*covariance_mat*lambda));
```

# Part 2

**Q1 (abc)**

```
% set random seed
rng(10)
% size of sample 1
s1_size = 6;
% size of sample 2
s2_size = 8;
% set the means to g1 and g2
mean_s1 = 1;
mean_s2 = 1.5;
% stochastic error
e1 = 0.25*randn(s1_size, 1);
e2 = 0.25*randn(s2_size, 1);
% generate datasets
sample1 = mean_s1+e1;
sample2 = mean_s2+e2;

%% 1_(a)
% perform two-sample t-test
[hyp,p1,ci,stats] = ttest2(sample1, sample2);
original_tstatistic = stats.tstat;
original_meandiff = mean(sample1)-mean(sample2);
```

```matlab
%% 1_(b)_(i)
% both groups
bothgroups = [sample1; sample2];
% construct an one-dimensional array D
D=bothgroups';

%% 1_(b)_(ii)&(iii)
% construct all the valid permutations of D
C1 = combnk(D, s1_size);
% initialise C2
C2 = zeros(length(C1),s2_size);

% initialise array to store all t-statistics
tstatistic_array = zeros(1, length(C1)); % initialise array to store
all t-statistics
for i=1:length(C1)
    % make sure there is no repeat for valid permutations
    C2(i,:) = setdiff(D, C1(i,:));
    % compute the t-statistics for all the membership permutations
    [hyp,p2,ci,stats] = ttest2(C1(i,:), C2(i,:));
    tstatistic_array(i) = stats.tstat;
end
% construct empirical distribution of t-statistic
figure,
hist(tstatistic_array, 50);
title('The empirical distribution of t-statistic');

%% 1_(b)_(iv)
% determine the p-value by finding the percentage of the permutations
with a t-statistic greater
% than and equal to that of the original labeling.
p_value_b = nnz(tstatistic_array <=
original_tstatistic)/numel(tstatistic_array);

%% 1_(c)
% repeat (b) but rather than using the t-statistic, use the
difference between
% the means as the test statistic
mean_diff_array = zeros(1, length(C1));
for i=1:length(C1)
   mean_diff = mean(C1(i,:)) - mean(C2(i,:));
   mean_diff_array(i) = mean_diff;
end
figure,
```

```
hist(mean_diff_array, 50);
title('The empirical distribution of the difference between means');
% p-value computed with mean difference
p_value_c = nnz(mean_diff_array <=
original_meandiff)/numel(mean_diff_array);
```

**Q1 (d)**

```
% set random seed
rng(10);
% size of sample 1
s1_size = 6;
% size of sample 2
s2_size = 8;
% set the means to g1 and g2
mean_s1 = 1;
mean_s2 = 1.5;
% stochastic error
e1 = 0.25*randn(s1_size, 1);
e2 = 0.25*randn(s2_size, 1);
% generate datasets
sample1 = mean_s1+e1;
sample2 = mean_s2+e2;

% perform two-sample t-test
[hyp,p1,ci,stats] = ttest2(sample1, sample2);
original_tstatistic = stats.tstat;
original_meandiff = mean(sample1)-mean(sample2);

% both groups
bothgroups = [sample1; sample2];
% construct an one-dimensional array D
D=bothgroups';

%% d(i)
total_permutations = 1000;
tstatistic_array_d = zeros(1,total_permutations);
all_permutations = zeros(total_permutations, 6);
for i=1:total_permutations
    % use randperm to generate a random set of permutations of the
    % intergers
    random_permutations = randperm(14);
    all_permutations(i,:) = random_permutations(1:6);
```

```
    bothgroups = D(random_permutations);
    group1 = bothgroups(1:6);
    group2 = bothgroups(7:14);
    [hyp,p2,ci,stats] = ttest2(group1, group2);
    tstatistic_array_d(i) = stats.tstat;
end
figure,
hist(tstatistic_array_d,50);
title('The empirical distribution of t-statistic');
p_value_d = nnz(tstatistic_array_d
<=original_tstatistic)/numel(tstatistic_array_d);

%% (diii)
% sort the elements of each row of permutations
sort_permutations = sort(all_permutations(:,1:6),2);
% get the position of elments without repetition
[~, id1, ~]=unique(sort_permutations, 'rows');

extended_permutations = 1500;
tstatistic_array_diii = zeros(1,total_permutations);
all_permutations_iii= zeros(extended_permutations, 14);
for i = 1:extended_permutations
    % randperm generates random set of integer permutations
    % integers are indices for D
    random_permutations_iii = randperm(14);
    all_permutations_iii(i,:) = random_permutations_iii;
    bothgroups_iii = D(random_permutations_iii);
    group1_iii = bothgroups_iii(1:6);
    group2_iii = bothgroups_iii(7:14);
    [hyp,p3,ci,stats] = ttest2(group1_iii, group2_iii);
    tstatistic_array_diii(i) = stats.tstat;
end
% sort the elements of each row of permutations
sort_permutations_iii = sort(all_permutations_iii(:,1:6),2);
% get the position of elments without repetition
[~, id2, ~]=unique(sort_permutations_iii, 'rows');

new_permutations = sort_permutations_iii(id2,:);
new_permutations = new_permutations(1:total_permutations, :);
tstatistic_array_diii = tstatistic_array_diii(id2);
tstatistic_array_diii = tstatistic_array_diii(10:1010);

% p-value without repetition
p_value_diii = nnz(tstatistic_array_diii  <=
```

```
original_tstatistic)/numel(tstatistic_array_diii);
```

**Q2**

```
% load files
CPA_I = [4,5,6,7,8,9,10,11]; % identifier for CPA file names
PPA_I = [3,6,9,10,13,14,15,16]; % identifier for PPA file names
number_subjects = 8;
% group1 had 8 subjects, 64000=40*40*40
CPA = zeros(64000,number_subjects);
j = 0;
for i=drange(CPA_I)
    filename = sprintf('CPA%d_diffeo_fa.img',i);
    fid = fopen(filename, 'r', 'l'); % little-endian
    CPA_sub = fread(fid, 'float'); % 16-bit floating point
    j=j+1;
    CPA(:,j) = CPA_sub;
end
% group2 had 8 subjects, 64000=40*40*40
PPA = zeros(64000, number_subjects);
j = 0;
for i=drange(PPA_I)
    filename = sprintf('PPA%d_diffeo_fa.img',i);
    fid = fopen(filename, 'r', 'l'); % little-endian
    PPA_sub = fread(fid, 'float'); % 16-bit floating point
    j=j+1;
    PPA(:,j) = PPA_sub;
end
% wm_mask.img is an additional binary volume defining the ROI for
% statistical analysis
fid = fopen('wm_mask.img', 'r', 'l');
mask = fread(fid, 'float');
voxels = [CPA PPA];

%% 2_(a)
% we use the GLM in part1_1c Y = X1b1 + X2b2 + e.
% generate design matrix
oness = ones(number_subjects,1);
zeross = zeros(number_subjects,1);
design_mat_X = [oness,zeross;zeross,oness];
% contrast vector
lambda = [1 -1]';
% determine betahat
```

```
betahat = voxels*(inv(design_mat_X'*design_mat_X)*design_mat_X')';
% determine ehat
ehat = voxels-(betahat*design_mat_X');
% covariance matrix of estimated model parameters
variance = sum((ehat.*ehat), 2)./ 14;
t_statiatic =
(betahat*lambda)./sqrt(variance.*(lambda'*(inv(design_mat_X'*design_m
at_X))*lambda));
% compute t-statisti for ROI
ROItstatistic = t_statiatic.*mask;
% compute maximum t-statistic.
maxtstatistic = max(ROItstatistic);

%% 2_(b)
s1_size = 8;
s2_size = 8;
indices = 1:(s1_size+s2_size);
% construct all the valid permutations of indices
C1 = combnk(indices, s1_size);
% initialise C2
C2 = zeros(length(C1),s2_size);
for i=1:length(C1)
    % make sure there is no repeat for valid permutations
    C2(i,:) = setdiff(indices, C1(i,:));
end
bothgroups = [C1 C2];
% initialise array to store all max t-statistic
max_tstatistic_array = zeros((length(C1)),1);
% just keep the values in ROI
voxels = voxels.*mask;
for i=1:length(C1)
    voxels_now = voxels(:,bothgroups(i,:));
    betahat_now =
voxels_now*(inv(design_mat_X'*design_mat_X)*design_mat_X')';
    ehat_now = voxels_now-betahat_now*design_mat_X';
    % find std of error
    variance = sum((ehat_now.*ehat_now), 2)./ 14;
    tstatistic =
(betahat_now*lambda)./sqrt(variance.*(lambda'*(inv(design_mat_X'*desi
gn_mat_X))*lambda));
    max_tstatistic_array(i,1) = max(tstatistic);
end
figure,
hist(max_tstatistic_array,50);
```

```
title('The empirical distribution of max t-statistic');

%% (c)
p_value_d =
nnz(max_tstatistic_array >=maxtstatistic)/numel(max_tstatistic_array)
;

%% (d)
% determine maximum t-statistic threshold corresponding to p-value of
5%
threeshold=prctile(max_tstatistic_array, 95);
```