STAT 6337 Project 2

Lakshmipriya Narayanan

Problem 1

Normal distribution with $\mu =$ and $\sigma = 1$: From the above output tables, we can see that the observed type I errr rate is close to the nominal 5% for all combinations of sample sizes.

n_1	n_2	TypeI error rate, α
10	10	0.057
30	30	0.051
10	30	0.042

Uniform distribution with $a = 0$ and $b = 1$: The results above are similar to the results generated for standard normal distribution but, there are some deflections from the standard normal distribution.

n_1	n_2	TypeI error rate, α
10	10	0.053
30	30	0.071
10	30	0.052

T distribution with degrees of freedom, $df = 1$: The above table tells us that the samples with this distribution deviate from the 5% level because the t distribution has heavier tails unlike the normal distribution.

n_1	n_2	TypeI error rate, α
10	10	0.018
30	30	0.015
10	30	0.039

Problem 2

See code (Obtained a p -value of 1 – which is incorrect)

Problem 3

(a):

Fitted model: $\text{SBP} = 86.66 + 1.78 \cdot \text{BMI}$
Confidence Interval for slope, $\text{BMI} = [1.66, 1.91]$
Unbiased estimator of $\sigma = (\text{Root MSE})^2 = 21.12489^2 = 446.26$

(b)

Interval estimates for mean response : $[83.329, 90.004]$
Interval estimates for future response of Avg BMI : $[98.1236, 167.641]$
Interval estimates for future response of Q1 BMI : $[93.2034, 162.724]$

Intervals using WORKING HOTELLING METHOD (By hand):
$b_0 = 86.66$, $b_1 = 1.78$, $X_i = 25.84 = \bar{X}$, $S_{XX} = 74265.26$, $n = 4415$, $S = 21.12$, $\sqrt{2 \cdot F_{1-\alpha,2,n-2}} = \sqrt{2 \cdot F_{1-0.05,2,4413}} = \sqrt{2 \cdot 19.49} = 6.25$

Interval estimates for mean response :
Confidence band $= b_0 + b_1 \cdot X_i \pm \sqrt{2 \cdot F_{1-\alpha,2,n-2}} \cdot S \cdot \sqrt{\frac{1}{n} + \frac{(X_i - \bar{X})^2}{S_{XX}}}$
$= 86.66 + 0 \pm 6.25 \cdot 21.12 \cdot \sqrt{\frac{1}{4415} + 0}$ because $X_i - \bar{X} = 0$ and $b_1 = 0$.
Therefore, confidence band $= 86.66 \pm 1.986 = [84.674, 88.646]$

Interval estimates for future response of Avg BMI :
Confidence band $= b_0 + b_1 \cdot X_i \pm \sqrt{2 \cdot F_{1-\alpha,2,n-2}} \cdot S \cdot \sqrt{\frac{1}{n} + \frac{(X_i - \bar{X})^2}{S_{XX}}}$
$= 86.66 + 1.78 \cdot 25.84 \pm 6.25 \cdot 21.12 \cdot \sqrt{\frac{1}{4415} + 0}$ because $X_i - \bar{X} = 0$.
Therefore, confidence band $= 132.65 \pm 1.986 = [130.664, 134.636]$

Interval estimates for future response of first quartile BMI :
$X_i = 23.09$, the rest of the calculations are same as above Confidence band $= b_0 + b_1 \cdot X_i \pm \sqrt{2 \cdot F_{1-\alpha,2,n-2}} \cdot S \cdot \sqrt{\frac{1}{n} + \frac{(X_i - \bar{X})^2}{S_{XX}}}$
$= 86.66 + 1.78 \cdot 23.09 \pm 6.25 \cdot 21.12 \cdot \sqrt{\frac{1}{4415} + \frac{(23.09-25.84)^2}{74265.26}}$
Therefore, confidence band $= 127.76 \pm 2.391 = [125.369, 130.151]$

From the above computed intervals, we can see that without the working-hotelling method, the intervals are more accurate and contain the estimates. The prediction intervals for future responses are more wider. The intervals estimated by the working- hotelling method are wider than the confidence intervals for a single mean response value.

(c)

MSR $= 237,560$ and MSE $= 446.261$. From the output ANOVA table, F = 532.33.
MSR and MSE will be similar when their expectations are equal, ie
$E(MSR) = \sigma^2 + \beta_1^2 + S_{XX} = E(MSE) = \sigma^2$ This happens when $\beta_1 = 0$ and when there is no linear relationship between SBP and BMI. A formal test for making this comparison is an F test with $H_0 : \beta_1 = 0$ vs the $H_A : \beta_1 \neq 0$.
Test statistic, $F_{obs} = 32.23$, p-value $= < 0.0001 \leq \alpha = 0.05$. Therefore, we can reject the H_0 and conclude that there is a linear relationship b/w SBP and BMI and that MSR and MSE are not similar.

(d):

Coefficient of determination, $R^2 = 0.1076$. This is not as high and this proportion indicates that the model we fit is not the best fit because high values indicate a better fit.
For simple linear regression, $R^2 = r^2$ (correlation coefficient). And $F = \frac{r^2 \cdot (n-2)}{1-r^2} = \frac{R^2 \cdot (n-2)}{1-R^2} = \frac{0.1076 \cdot (4415-2)}{1-0.1076} = 532.09 \approx 532.23$
Now, F $= t^2 \implies t = \sqrt{F}$. Therefore, t test statistic, t $= \sqrt{532.23} = 23.0701$ matches the t_{obs} for slope = 23.07.

(e):

See 3(e) in RELEVANT SAS OUTPUTS FOR CONFIDENCE BANDS

The Bonferroni method uses t critical values and the confidence band is in the form of a rectangle where the bottom of the rectangle gives lower and upper bound for b_0 whereas the upper part of the rectangle gives the lower and upper bound for b_1.
Confidence band for $b_0 = [82.68, 90.64]$ and $b_1 = [1.63, 1.94]$. These values are very close to our original estimates obtained in our original model but slightly wider.

Using the ellipsoid, our intervals for b_1 is $[1.87, 1.95]$ whereas our interval for b_0 is $[82.58, 90.58]$. This ellipsoid interval is more wide than the Bonferroni interval. When we compare them with (a), both of these above intervals are wider but not as wide as a prediction interval would be.

Relevant SAS Outputs:

Problem 1:

distribution	n1	n2	type1_rate
N(0, 1)	10	10	0.057

distribution	n1	n2	type1_rate
N(0, 1)	30	30	0.051

distribution	n1	n2	type1_rate
N(0, 1)	10	30	0.042

distribution	n1	n2	type1_rate
t_1	10	10	0.018

distribution	n1	n2	type1_rate
t_1	30	30	0.015

distribution	n1	n2	type1_rate
t_1	10	30	0.039

distribution	n1	n2	type1_rate
Unif(0, 1)	10	10	0.053

distribution	n1	n2	type1_rate
Unif(0, 1)	30	30	0.071

distribution	n1	n2	type1_rate
Unif(0, 1)	10	30	0.052

Problem 2:

Obs	proportion_reject
1	1

Problem 3:

(a):

The REG Procedure
Model: MODEL1
Dependent Variable: SBP

Number of Observations Read	4434
Number of Observations Used	4415
Number of Observations with Missing Values	19

Analysis of Variance					
Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	1	237560	237560	532.33	<.0001
Error	4413	1969349	446.26082		
Corrected Total	4414	2206909			

Root MSE	21.12489	R-Square	0.1076
Dependent Mean	132.89320	Adj R-Sq	0.1074
Coeff Var	15.89614		

Parameter Estimates							
Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|	90% Confidence Limits	
Intercept	1	86.66684	2.02861	42.72	<.0001	83.32938	90.00430
BMI	1	1.78852	0.07752	23.07	<.0001	1.66099	1.91605

(b):

Interval estimates and future estimates for Average BMI:

Obs	AGE	TOTALCHOL	SBP	DBP	BMI	CIGSPERDAY	GLUCOSE	HEARTRATE	CVD	HYPERTENSION	yhat	lowerCI	upperCI	lowerPI	upperPI	residuals
1	36	226	124.0	76.0	25.84	20	70	75	0	1	132.882	132.359	133.405	98.1236	167.641	-8.8822
2	57	303	160.5	98.5	25.84	0	100	81	0	1	132.882	132.359	133.405	98.1236	167.641	27.6178
3	54	318	115.0	81.0	25.84	0	76	95	0	1	132.882	132.359	133.405	98.1236	167.641	-17.8822
4	63	228	141.0	82.0	25.84	20	81	82	1	1	132.882	132.359	133.405	98.1236	167.641	8.1178

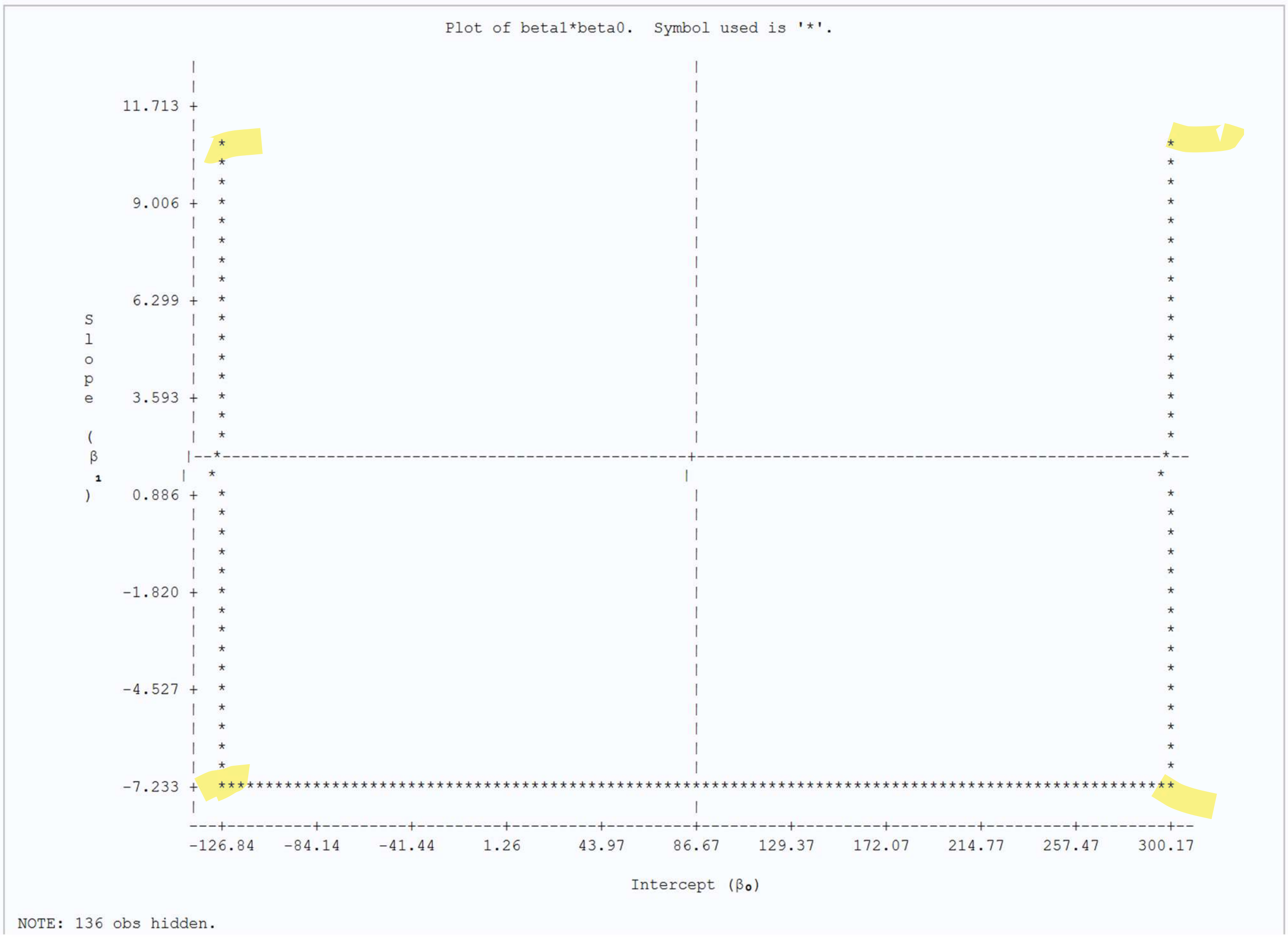
Interval estimates and future estimates for first quartile BMI:

Obs	AGE	TOTALCHOL	SBP	DBP	BMI	CIGSPERDAY	GLUCOSE	HEARTRATE	CVD	HYPERTENSION	yhat	lowerCI	upperCI	lowerPI	upperPI	residuals
1	39	269	97.0	64.0	23.09	20	67	82	0	1	127.964	127.334	128.594	93.2034	162.724	-30.9638
2	52	248	155.0	93.0	23.09	0	70	75	0	1	127.964	127.334	128.594	93.2034	162.724	27.0362
3	37	240	120.0	79.0	23.09	20	80	75	0	1	127.964	127.334	128.594	93.2034	162.724	-7.9638
4	54	272	132.5	91.0	23.09	5	78	70	0	1	127.964	127.334	128.594	93.2034	162.724	4.5362
5	35	231	150.0	90.0	23.09	20	72	83	0	1	127.964	127.334	128.594	93.2034	162.724	22.0362
6	44	195	118.0	86.0	23.09	0	75	70	0	0	127.964	127.334	128.594	93.2034	162.724	-9.9638
7	38	220	107.0	73.5	23.09	0	80	61	0	0	127.964	127.334	128.594	93.2034	162.724	-20.9638
8	43	232	122.0	70.0	23.09	20	77	67	0	1	127.964	127.334	128.594	93.2034	162.724	-5.9638
9	36	200	121.5	72.5	23.09	5	75	75	0	0	127.964	127.334	128.594	93.2034	162.724	-6.4638
10	64	330	108.0	82.0	23.09	0	80	85	0	0	127.964	127.334	128.594	93.2034	162.724	-19.9638
11	40	169	123.5	77.5	23.09	10	.	71	0	1	127.964	127.334	128.594	93.2034	162.724	-4.4638
12	50	229	121.0	85.5	23.09	0	75	63	0	0	127.964	127.334	128.594	93.2034	162.724	-6.9638
13	47	.	121.0	70.0	23.09	20	83	80	0	1	127.964	127.334	128.594	93.2034	162.724	-6.9638
14	43	210	138.5	95.5	23.09	0	.	75	0	1	127.964	127.334	128.594	93.2034	162.724	10.5362
15	52	222	108.0	70.0	23.09	30	61	72	0	0	127.964	127.334	128.594	93.2034	162.724	-19.9638
16	40	242	115.0	74.0	23.09	20	80	68	1	0	127.964	127.334	128.594	93.2034	162.724	-12.9638

(e):

Plot of ellipse and confidence bands for method 1 (Bonferroni method):

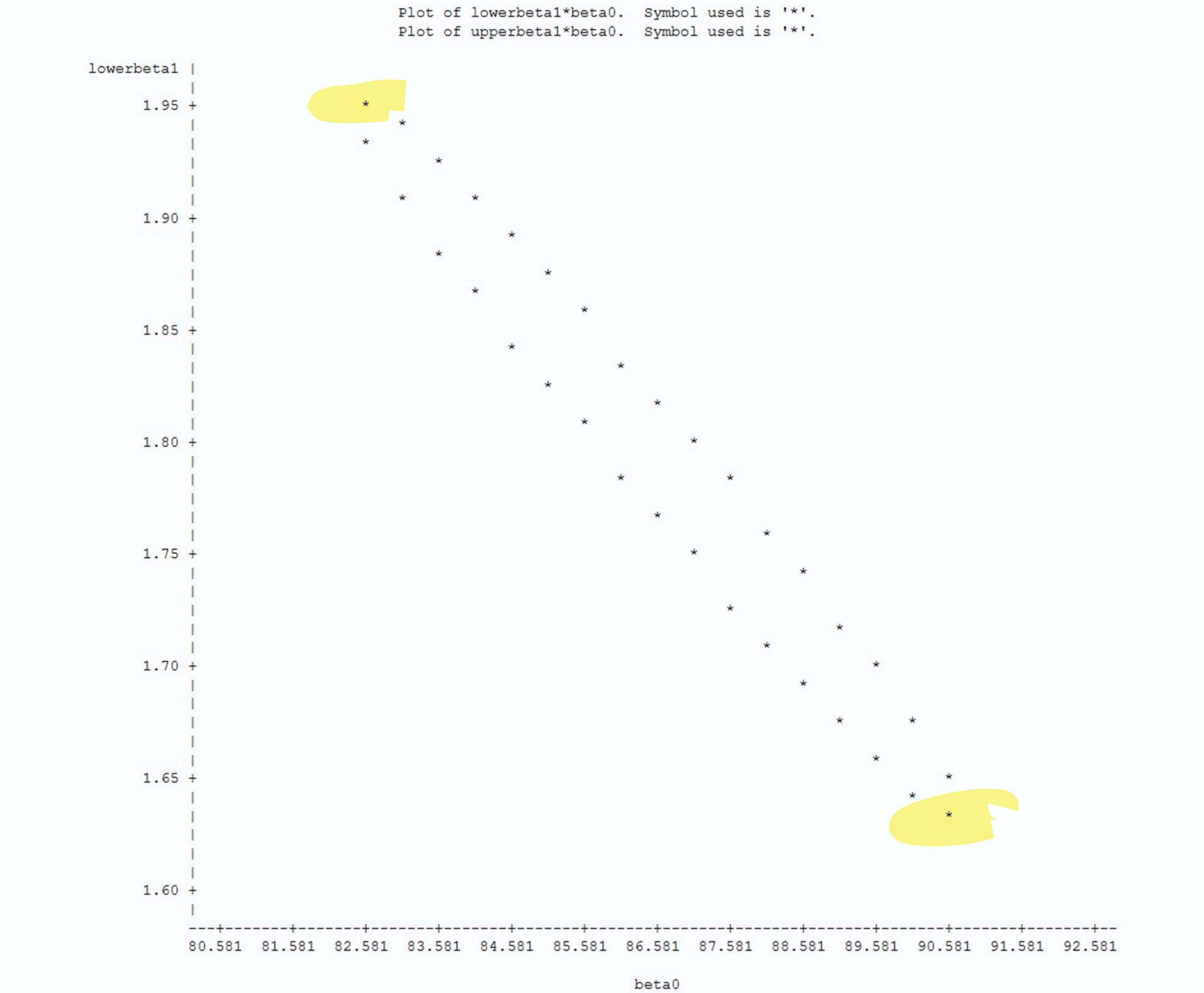
Obs	lower_b0	b0	upper_b0	lower_b1	b1	upper_b1
1	82.6898	86.6668	90.6439	1.63655	1.78852	1.94049



Plot of ellipse and confidence bands for method 2:

Obs	_TYPE_	_FREQ_	n	xbar	Sxx	s	b0	b1	sb1	sb0	F90	beta0	D	upperbeta1	lowerbeta1
1	0	4434	4415	25.8462	74265.26	21.1249	86.6668	1.78852	0.077518	2.02861	2056.18	80.5810	-5926746006.70	.	.
2	0	4434	4415	25.8462	74265.26	21.1249	86.6668	1.78852	0.077518	2.02861	2056.18	81.0810	-4013291385.44	.	.
3	0	4434	4415	25.8462	74265.26	21.1249	86.6668	1.78852	0.077518	2.02861	2056.18	81.5810	-2263777326.58	.	.
4	0	4434	4415	25.8462	74265.26	21.1249	86.6668	1.78852	0.077518	2.02861	2056.18	82.0810	-678203830.10	.	.
5	0	4434	4415	25.8462	74265.26	21.1249	86.6668	1.78852	0.077518	2.02861	2056.18	82.5810	743429103.99	1.95174	1.93370
6	0	4434	4415	25.8462	74265.26	21.1249	86.6668	1.78852	0.077518	2.02861	2056.18	83.0810	2001121475.68	1.93864	1.90905
7	0	4434	4415	25.8462	74265.26	21.1249	86.6668	1.78852	0.077518	2.02861	2056.18	83.5810	3094873284.99	1.92338	1.88658
8	0	4434	4415	25.8462	74265.26	21.1249	86.6668	1.78852	0.077518	2.02861	2056.18	84.0810	4024684531.90	1.90709	1.86513
9	0	4434	4415	25.8462	74265.26	21.1249	86.6668	1.78852	0.077518	2.02861	2056.18	84.5810	4790555216.43	1.89013	1.84435
10	0	4434	4415	25.8462	74265.26	21.1249	86.6668	1.78852	0.077518	2.02861	2056.18	85.0810	5392485338.57	1.87266	1.82408

Confidence region



NOTE: 16 obs had missing values.

PROBLEM 1

```
/*Generate MC samples from Normal(0, 1), with the sample size n1=n2=10 */

data MC_Normn10;
call streaminit(5432); * Set seed for result reproduction;
do mcRun = 1 to 1000;  * Generate 1000 replicates of Monte Carlo simulation;
     do i = 1 to 20;  * Generate n1 + n2 = 20 independent samples with standard Normal distribution ;
        * Separate into two different sample groups for easy computation during ttest;
        if i <= 10 then sample_group = 1;
        else sample_group = 2;
        x10 = rand("normal", 0, 1);
        output;
     end;
end;
run;

* Perform two sided t-test to test mean for above simulated samples by for each
sample size(class variable);
ods select none;  * Suppress output because program takes a very long time to run;
proc ttest data=MC_Normn10;
by mcRun; /* This two-sided t test is performed for each monte carlo simulation for the
two independent samples */
class sample_group;
var x10;
ods output ttests = resultsn10; /* Save the results of the ttest into a dataset to use for
computing pvalues and error rate */
run;
ods select all;

* Calculate Type I error rate by computing number of tests that actually Reject H0;
data T1err;
set resultsn10;
where method = "Pooled"; *Perform pooled ttest because we assume equality of variances;
if probt < 0.05 then RH0 = 1; *Alpha = 0.05 is what we want to test the null hypothesis H0 for;
else RH0 = 0; * If pvalue < 0.05, then we reject H0;
run;

* Calculate the mean of the tests for which H0 was rejected to get the TypeI error rate;
proc means data=T1err mean noprint;
var RH0;
output out=type1_rate mean=type1_rate;
run;

* Store results for displaying the type 1 error rate;
data finalres_norm_10;
set type1_rate;
distribution = "N(0, 1)";
n1 = 10;
n2 = 10;
run;

* Print above results;
proc print data = finalres_norm_10 noobs;
var distribution n1 n2 type1_rate;
run;

*------------------------------------------------------------------------------------;

/*Generate MC samples from Normal(0, 1), with the sample size n1=n2=30. Same procedure followed as above */
data MC_Normn30;
call streaminit(5432); * Set seed for result reproduction;
do mcRun = 1 to 1000;  * Generate 1000 replicates of Monte Carlo simulation;
    do i = 1 to 60;  * Generate n1 + n2 = 60 independent samples with standard Normal distribution ;
       if i <= 30 then sample_group = 1;
       else sample_group = 2;
       x30 = rand("normal", 0, 1);
       output;
```

```
    end;
end;
run;

* Perform two sided t-test to test mean for above simulated samples by for each
sample size(class variable);
ods select none;  /* Suppress output because program takes a very long time to run */
proc ttest data=MC_Normn30;
by mcRun;
class sample_group;
var x30;
ods output ttests = resultsn30;
run;
ods select all;

* Calculate Type I error rate by computing number of tests that actually Reject H0;
data T1err;
set resultsn30;
where method = "Pooled";
if probt < 0.05 then RH0 = 1;
else RH0 = 0;
run;

proc means data=T1err mean noprint;
var RH0;
output out=type1_rate mean=type1_rate;
run;

* Store results for displaying the type 1 error rate;
data finalres_norm_30;
set type1_rate;
distribution = "N(0, 1)";
n1 = 30;
n2 = 30;
run;


proc print data = finalres_norm_30 noobs;
var distribution n1 n2 type1_rate;
run;


*-------------------------------------------------------------------------------------------;

/*Generate MC samples from Normal(0, 1), with the sample size n1 = 10 and n2 = 30.
Same procedure followed as above*/
data MC_Normn1030;
call streaminit(112344);
do mcRun = 1 to 1000;
   do i = 1 to 40;  * Generate n1 + n2 = 40 independent samples with standard Normal distribution ;
      if i <= 10 then sample_group = 1;
      else sample_group = 2;
      x1030 = rand("normal", 0, 1);
      output;
   end;
end;
run;

* Perform two sided t-test to test mean for above simulated samples by for each
sample size(class variable);
ods select none;  /* Suppress output because program takes a very long time to run */
proc ttest data=MC_Normn1030;
by mcRun;
class sample_group;
var x1030;
ods output ttests = resultsn1030;
run;
```

```
ods select all;

* Calculate Type I error rate by computing number of tests that actually Reject H0;
data T1err;
set resultsn1030;
where method = "Pooled";
if probt < 0.05 then RH0 = 1;
else RH0 = 0;
run;

proc means data=T1err mean noprint;
var RH0;
output out=type1_rate mean=type1_rate;
run;

* Store results for displaying the type 1 error rate;
data finalres_norm_1030;
set type1_rate;
distribution = "N(0, 1)";
n1 = 10;
n2 = 30;
run;

proc print data = finalres_norm_1030 noobs;
var distribution n1 n2 type1_rate;
run;
```

```
/*Generate MC samples from t distribution with 1 df, with the sample size n1=n2=10 */
data MC_tn10;
call streaminit(5); * Set seed for reproducing the results;
do mcRun = 1 to 1000;  * Generate 1000 replicates of Monte Carlo simulation;
     do i = 1 to 20;  * Generate n1 + n2 = 20 independent samples with standard t distribution ;
        * Separate into two different sample groups for easy computation during ttest;
        if i <= 10 then sample_group = 1;
        else sample_group = 2;
        z10 = rand("t", 1);
        output;
     end;
end;
run;

* Perform two sided t-test to test mean for above simulated samples by for each
sample size(class variable);
ods select none;  * Suppress output because program takes a very long time to run;
proc ttest data=MC_tn10;
by mcRun;
class sample_group;
var z10;
ods output ttests = resultsn10;
run;
ods select all;

* Calculate Type I error rate by computing number of tests that actually Reject H0;
data T1err;
set resultsn10;
where method = "Pooled";
if probt < 0.05 then RH0 = 1;
else RH0 = 0;
run;

* Calculate the mean of the tests for which H0 was rejected to get the TypeI error rate;
proc means data=T1err mean noprint;
var RH0;
output out=type1_rate mean=type1_rate;
run;

* Store results for displaying the type 1 error rate;
data finalres_t_10;
set type1_rate;
distribution = "t_1";
n1 = 10;
n2 = 10;
run;

* Print above results;
proc print data = finalres_t_10 noobs;
var distribution n1 n2 type1_rate;
run;

*---------------------------------------------------------------------------------------;

/*Generate MC samples from t distribution with 1 df, with the sample size n1=n2=30.
Same steps as above is repeated for a different variable and sample size */
data MC_tn30;
call streaminit(8122000);
do mcRun = 1 to 1000;
    do i = 1 to 60;  * Generate n1 + n2 = 60 independent samples with standard t distribution ;
```

```
        if i <= 30 then sample_group = 1;
        else sample_group = 2;
        z30 = rand("t", 1);
        output;
     end;
end;
run;

* Perform two sided t-test to test mean for above simulated samples by for each
sample size(class variable);
ods select none;  * Suppress output because program takes a very long time to run;
proc ttest data=MC_tn30;
by mcRun;
class sample_group;
var z30;
ods output ttests = resultsn30;
run;
ods select all;

* Calculate Type I error rate by computing number of tests that actually Reject H0;
data T1err;
set resultsn30;
where method = "Pooled";
if probt < 0.05 then RH0 = 1;
else RH0 = 0;
run;

proc means data=T1err mean noprint;
var RH0;
output out=type1_rate mean=type1_rate;
run;

* Store results for displaying the type 1 error rate;
data finalres_t_30;
set type1_rate;
distribution = "t_1";
n1 = 30;
n2 = 30;
run;

proc print data = finalres_t_30 noobs;
var distribution n1 n2 type1_rate;
run;


*-----------------------------------------------------------------------------------------;

/*Generate MC samples from t distribution with 1 df, with the sample size n1= 10 and n2=30.
Same steps as above is repeated for a different variable and sample size */
data MC_tn1030;
call streaminit(12378094);
do mcRun = 1 to 1000;
   do i = 1 to 40;  * Generate n1 + n2 = 40 independent samples with standard Normal distribution ;
      if i <= 10 then sample_group = 1;
      else sample_group = 2;
      z1030 = rand("t", 1);
      output;
   end;
end;
run;

ods select none;
```

```
proc ttest data=MC_tn1030;
by mcRun;
class sample_group;
var z1030;
ods output ttests = resultsn1030;
run;
ods select all;
```

```
data T1err;
set resultsn1030;
where method = "Pooled";
if probt < 0.05 then RH0 = 1;
else RH0 = 0;
run;
```

```
proc means data=T1err mean noprint;
var RH0;
output out=type1_rate mean=type1_rate;
run;
```

```
data finalres_t_1030;
set type1_rate;
distribution = "t_1";
n1 = 10;
n2 = 30;
run;
```

```
proc print data = finalres_t_1030 noobs;
var distribution n1 n2 type1_rate;
run;
```

```
/*Generate MC samples from Unif(0, 1), with the sample size n1=n2=10.
Same process as followed for Normal and t distribution */
data MC_Unifn10;
call streaminit(543210);
do mcRun = 1 to 1000;
     do i = 1 to 20;  * Generate n1 + n2 = 20 independent samples with standard Normal distribution ;
        if i <= 10 then sample_group = 1;
        else sample_group = 2;
        y10 = rand("uniform");
        output;
     end;
end;
run;

ods select none;
proc ttest data=MC_Unifn10;
by mcRun;
class sample_group;
var y10;
ods output ttests = resultsn10;
run;
ods select all;

data T1err;
set resultsn10;
where method = "Pooled";
if probt < 0.05 then RH0 = 1;
else RH0 = 0;
run;

proc means data=T1err mean noprint;
var RH0;
output out=type1_rate mean=type1_rate;
run;

data finalres_unif_10;
set type1_rate;
distribution = "Unif(0, 1)";
n1 = 10;
n2 = 10;
run;


proc print data = finalres_unif_10 noobs;
var distribution n1 n2 type1_rate;
run;

*-----------------------------------------------------------------------------------------;

/*Generate MC samples from Unif(0, 1), with the sample size n1=n2=30.
Same process as followed for Normal and t distribution */
data MC_Unifn30;
call streaminit(523672);
do mcRun = 1 to 1000;
    do i = 1 to 60;
       if i <= 30 then sample_group = 1;
       else sample_group = 2;
       y30 = rand("uniform");
       output;
    end;
end;
```

```
run;

ods select none;
proc ttest data=MC_Unifn30;
by mcRun;
class sample_group;
var y30;
ods output ttests = resultsn30;
run;
ods select all;

data T1err;
set resultsn30;
where method = "Pooled";
if probt < 0.05 then RH0 = 1;
else RH0 = 0;
run;

proc means data=T1err mean noprint;
var RH0;
output out=type1_rate mean=type1_rate;
run;

data finalres_unif_30;
set type1_rate;
distribution = "Unif(0, 1)";
n1 = 30;
n2 = 30;
run;

proc print data = finalres_unif_30 noobs;
var distribution n1 n2 type1_rate;
run;


*-----------------------------------------------------------------------------------------;

/*Generate MC samples from Unif(0, 1), with the sample size n1=n2=30.
Same process as followed for Normal and t distribution */
data MC_Unifn1030;
call streaminit(566578);
do mcRun = 1 to 1000;
   do i = 1 to 40;
      if i <= 10 then sample_group = 1;
      else sample_group = 2;
      y1030 = rand("uniform");
      output;
   end;
end;
run;

ods select none;
proc ttest data=MC_Unifn1030;
by mcRun;
class sample_group;
var y1030;
ods output ttests = resultsn1030;
run;
ods select all;

data T1err;
set resultsn1030;
where method = "Pooled";
```

```
if probt < 0.05 then RH0 = 1;
else RH0 = 0;
run;
```

```
proc means data=T1err mean noprint;
var RH0;
output out=type1_rate mean=type1_rate;
run;
```

```
data finalres_unif_1030;
set type1_rate;
distribution = "Unif(0, 1)";
n1 = 10;
n2 = 30;
run;
```

```
proc print data = finalres_unif_1030 noobs;
var distribution n1 n2 type1_rate;
run;
```

PROBLEM 2

```
/* Create a pointer named FHS to the data file */
filename FHS "/home/u63986019/FramHeartStudy_data.csv";

DATA c; /* Assign name c to data */
INFILE FHS DSD FIRSTOBS = 2; /* Since the data is a CSV, use DSD FIRS
INPUT AGE TOTALCHOL SBP DBP BMI CIGSPERDAY GLUCOSE HEARTRATE CVD HYPE

/*Converting numerical variable TOTALCHOL to categorical by grouping
IF TOTALCHOL LE 206 then TCGRP = 1;
IF TOTALCHOL > 206 AND TOTALCHOL LE 234 then TCGRP = 2;
IF TOTALCHOL > 234 AND TOTALCHOL LE 264 then TCGRP = 3;
IF TOTALCHOL > 264 then TCGRP = 4;
RUN;

/*Get the quartiles to divide the TOTALCHOL variable into groups*/
proc univariate data = c;
var TOTALCHOL;
RUN;

/* 1f: Goodness of fit test to check if TOTAL CHOL follows normal dis
proc freq data = c;
tables TCGRP / chisq;
OUTPUT out= ChiSqResults CHISQ;
run;

/* Calculate and expected frequencies to obtain test statistic
for the MC simulation */
DATA expected;
SET observed END=last;
RETAIN total 0;
total + COUNT;
IF last THEN DO;
    expected = total / 4;  /* Equal expected frequencies for 4 groups
    DO i = 1 TO 4;
        TCGRP = i;
        OUTPUT;
    END;
END;
KEEP TCGRP expected;
RUN;

/* Compute the observed chi-square test statistic */
```

```
DATA chi_obs;
MERGE observed expected;
BY TCGRP;
chi_contrib = ((COUNT - expected)**2) / expected;
RUN;

PROC MEANS DATA=chi_obs NOPRINT SUM;
VAR chi_contrib;
OUTPUT OUT=chi_obs_sum SUM=chi_obs;
RUN;

/* MC Simulation */
data mc_m;
set TOTALCHOL;
call streaminit(54321);
do mcRun = 1 to 1000;
    do j = 1 to 4; /* Generate 1000 samples of size 4 of normal varia
        /* Generate n(mu0, s_d^2) random samples */
        TCGRP = rand("normal", mean(TOTALCHOL), 44.65110);
        output;
    end;
end;
run;

/* Calculate chi-square statistic for each replicate generates above
PROC FREQ DATA=mc_m NOPRINT;
TABLES mcRun*TCGRP / CHISQ OUT=mc_chi(KEEP=mcRun _PCHI_) OUTEXPECT;
RUN;

/* Calculate p-value and count rejections to obtain proportion of tes
ie H0: TOTALCHOL follows normal distribution */
DATA p_value;
SET mc_chi END=last;
RETAIN count 0 total 0;
IF _PCHI_ >= chi_obs THEN count + 1;
total + 1;
IF last THEN p_value = count / total;
RUN;

/* Calculate proportion of p-values less than significance level, alp
DATA reject_null;
SET mc_chi;
IF _PCHI_ < 0.05 THEN reject = 1;
```

```
ELSE reject = 0;
RUN;

PROC MEANS DATA=reject_null NOPRINT;
VAR reject;
OUTPUT OUT=reject_summary MEAN=proportion_reject;
RUN;

/* Print the p value of the simulation */
PROC PRINT DATA=reject_summary;
VAR proportion_reject;
RUN;
```

PROBLEM 3

```
* Create a pointer named FHS to the data file;
filename FHS "/home/u63986019/FramHeartStudy_data.csv";

DATA c; /* Assign name c to data */
INFILE FHS DSD FIRSTOBS = 2; /* Since the data is a CSV, use DSD FIRSTOBS = 2*/
INPUT AGE TOTALCHOL SBP DBP BMI CIGSPERDAY GLUCOSE HEARTRATE CVD HYPERTENSION; /*Input names

* (a) Simple linear regression model of SBP on BMI.;
PROC REG DATA = c ALPHA=0.1;
MODEL SBP = BMI / CLB CLM CLI; /*Model is y = x; CLB: CI for Parameters; CLM: CI for Means;
/* Save predicted values (p) and residuals (r) in results dataset */
output out = results PREDICTED = yhat r = residuals LCL = lowerPI UCL = upperPI LCLM = lower
RUN;

/* (b): Interval estimates for the mean response (Y) and an individual future response for B
values equal to its average and first quartile. */
proc univariate data = c;
var BMI;
RUN;

/* Filter the results for BMI = 25.8 AVG AND BMI = 23.09 Q1 */
DATA filtered_results;
   SET results;
   IF BMI = 25.84 OR BMI = 23.09;
RUN;

/*Print the filtered results */
PROC PRINT DATA=filtered_results;
RUN;
```

```
/* (e): Simultaneous confidence sets for the regression coefficients using the two methods
discussed in class (Confidence ellipsoid and Bonferroni method). */

/*Calculate confidence band by confidence ellipsoid method --
 CODE PRIMARILY TAKEN FROM "ellipse" handout on eLearning */

/*Get sample mean, sample size and Sum of squares to calculate confidence intervals*/
PROC MEANS;
VAR BMI;
OUTPUT OUT=xdata
n=n
mean=xbar
css=Sxx;        /* save sample size */ /* mean of X_i */ /* sum of X_i**2 corrected for the
RUN;

*Fit a regression model for SBP and BMI as in (a) to obtain predictions;
PROC REG DATA=C OUTEST=est;
MODEL SBP = BMI;
RUN;

/*Obtain intercept, slope and root MSE to calculate confidence regions*/
DATA est; SET est;
s = _rmse_;             /* root MSE = estimated standard deviation */
b0 = intercept;         /* estimated intercept b0 */
b1 = BMI;               /* estimated slope b1 */
/* Create macro variable for b0 and b1 to plot the rectangle for bonferroni method and avoid
for b0 and b1*/
CALL SYMPUT('b0', b0);
CALL SYMPUT('b1', b1);
KEEP s b0 b1;
RUN;
```

```
/* Computation of ellipsoid and confidence band that make up the ellipsoid.*/
DATA ellipse; MERGE xdata est;

sb1 = s/sqrt(Sxx);              /* standard deviation of b1 */
sb0 = s*sqrt(1/n+xbar**2/Sxx); /* standard deviation of b0 */

F90=finv(0.90,2,n-2)*2*s**2;    /* 90% upper bound for the quadratic form */

DO beta0=b0-3*sb0 BY 0.5 TO b0+3*sb0; /* for a fixed value of beta0, solve for beta1 using t
 quadratic form of ellipse */
    D = (n*xbar*(beta0-b0))**2 - (n*xbar**2+Sxx)*(n*(beta0-b0)**2-F90); /*discriminant */
    if D < 0 then do; upperbeta1 = .; lowerbeta1 = .; end; /* discard beta0 values with D<0
    else do; upperbeta1 = b1+(n*xbar*(b0-beta0)+sqrt(D))/(n*xbar**2+Sxx);
    lowerbeta1 = b1+(n*xbar*(b0-beta0)-sqrt(D))/(n*xbar**2+Sxx); end;
OUTPUT;
END;

/*Print the new confidence bands*/
PROC PRINT data=ellipse (obs=10);
RUN;

/*Plot the resulting ellipsoid */
PROC PLOT DATA=ellipse; TITLE 'Confidence region';
PLOT (lowerbeta1 upperbeta1)*beta0 = '*' / overlay;
RUN;


/*Calculate confidence band by the bonferroni method -- CODE PRIMARILY TAKEN FROM "ellipse"
handout on eLearning. The only variation is that we changed the DO loop to accomodate a rect
changed the critcal region from F distribution to t distribution */

/* Calculate Bonferroni confidence region and create plot data to plot the rectangle */
```

```
DATA plot_data; MERGE xdata est;

sb1 = s/sqrt(Sxx);              /* standard deviation of b1 */
sb0 = s*sqrt(1/n+xbar**2/Sxx); /* standard deviation of b0 */

    /* Calculate Bonferroni critical value */
    alpha = 0.1;  /* 90% upper bound for the quadratic form */
    p = 2;  /* Number of parameters */
    t_crit = TINV(1 - alpha/(2*p), n-2);

    /* Calculate the corners of the resulting rectangle which give the confidence band */
    lower_b0 = b0 - t_crit * sb0;
    upper_b0 = b0 + t_crit * sb0;
    lower_b1 = b1 - t_crit * sb1;
    upper_b1 = b1 + t_crit * sb1;

    *Lay points on x and y axis (b0 and b1) to form a rectangles;
    /* Bottom edge */
    DO beta0 = lower_b0 TO upper_b0 BY (upper_b0 - lower_b0)/100;
        beta1 = lower_b1;
        OUTPUT;
    END;
    /* Right edge */
    DO beta1 = lower_b1 TO upper_b1 BY (upper_b1 - lower_b1)/100;
        beta0 = upper_b0;
        OUTPUT;
    END;
    /* Top edge */
    DO beta0 = upper_b0 TO lower_b0 BY -(upper_b0 - lower_b0)/100;
        beta1 = upper_b1;
        OUTPUT;
    END;
```

```
    /* Left edge */
    DO beta1 = upper_b1 TO lower_b1 BY -(upper_b1 - lower_b1)/100;
        beta0 = lower_b0;
        OUTPUT;
    END;
RUN;

/* Obtain Bonferroni confidence bands/intervals */
PROC PRINT DATA=plot_data (OBS=1);
VAR lower_b0 b0 upper_b0 lower_b1 b1 upper_b1;
RUN;

/* Plot the confidence rectangle */
PROC PLOT DATA=plot_data;
PLOT beta1*beta0='*' / HREF=&b0 VREF=&b1;
LABEL beta0 = 'Intercept' beta1 = 'Slope';
RUN;
```