Stat 571 HW1

Dongyang Wang

2023-01-15

Question 1

Under the random intercept model, since $var(Y) = 1 = \theta + \sigma^2$ and $corr(Y_{ij}, Y_{ik}) = \rho = \frac{\theta}{\theta+\sigma^2}$, we solve the equations and get $\theta = \rho$ and $\sigma^2 = 1 - \rho$. In this way, we can generate the x, e, b seperately and use a linear relationship we choose to generate the y values, without the need to sample y directly but achieving the same results.

```
rm(list=ls())
set.seed(42)
library(mvtnorm)
library(lme4)
```

```
## Loading required package: Matrix
```

```
# For testing
m = 10
n = 5
p = 0.5

gen.one <- function(m,n,p){

  total = m*n

  # Set beta to 0.5
  beta1 = 0.5
  beta0 = 1

  # Get variance for Y
  #cov_mat = matrix(c(rep(p,total^2)), ncol = total)
  #for (i in 1:total){
  #  cov_mat[i,i] = 1
  #}

  # Generate the variables
  x = rnorm(total, 0, 1)
  b = rep(rnorm(m, mean = 0, sd = sqrt(p)),n)
  e = rnorm(total, mean = 0, sd = sqrt(1-p))
  y = beta0 + beta1*x + b + e

  # Linear regression
  linear_model = lm(y~x)
  linear_coef0 = coef(linear_model)[1]
  linear_coef1 = coef(linear_model)[2]
  linear_bias0 = linear_coef0 - beta0
```

```
  linear_bias1 = linear_coef1 - beta1
  conf_linear = confint(linear_model)
  linear_cov0 = as.numeric(conf_linear[1,1] < beta0 & beta0 < conf_linear[1,2])
  linear_cov1 = as.numeric(conf_linear[2,1] < beta1 & beta1 < conf_linear[2,2])

  # LMM
  lmm = lmer(y ~ x + (1|b))
  lmm_coef0 = fixef(lmm)[1]
  lmm_coef1 = fixef(lmm)[2]
  lmm_bias0 = lmm_coef0 - beta0
  lmm_bias1 = lmm_coef1 - beta1
  conf_lmm = confint.merMod(lmm, method = "Wald")
  lmm_cov0 = as.numeric(conf_lmm[3,1] < beta0 & beta0 < conf_lmm[3,2])
  lmm_cov1 = as.numeric(conf_lmm[4,1] < beta1 & beta1 < conf_lmm[4,2])

  return(c(linear_coef0, linear_coef1, linear_bias0, linear_bias1,
           linear_cov0, linear_cov1, lmm_coef0, lmm_coef1,
           lmm_bias0, lmm_bias1, lmm_cov0, lmm_cov1))
}
```

```
nsim = 500
m = seq(5,25,10)
n = seq(5,25,10)
p = seq(0.1,0.9,0.4)

results <- vector("list", length = length(m) * length(n))
for (i in 1:length(m)){

  for (j in 1:length(n)){
    result <- c()
    for (k in 1:length(p)){
      res <- replicate(nsim, gen.one(m[i],n[j],p[k]))
      mean_res <- rowMeans(res)
      result <- c(result, m[i],n[j],p[k],mean_res)
    }
    result <- matrix(result, ncol = 15, byrow = T)
    results[[(i-1)*length(m) + j]] = result
  }

}

#results
```

```
# For testing
m = 10
n = 5
p = 0.5

res <- replicate(10, gen.one(m,n,p))
mean_res <- rowMeans(res)
result <- c(result, mean_res)
```

```
#total_len = length(m) * length(n) * length(p)
#m_var = rep(m, total_len/length(m))
#n_var = rep(n, total_len/length(n))
```

```r
#p_var = rep(p, total_len/length(p))

result_df = do.call(rbind.data.frame, results)
#result_df_res = cbind(m_var, n_var,p_var,result_df)

colnames(result_df) = c("m", "n", "p", "linear_coef0", "linear_coef1",
                        "linear_bias0", "linear_bias1", "linear_cov0",
                        "linear_cov1", "lmm_coef0", "lmm_coef1",
                        "lmm_bias0", "lmm_bias1", "lmm_cov0", "lmm_cov1")
result_df
```

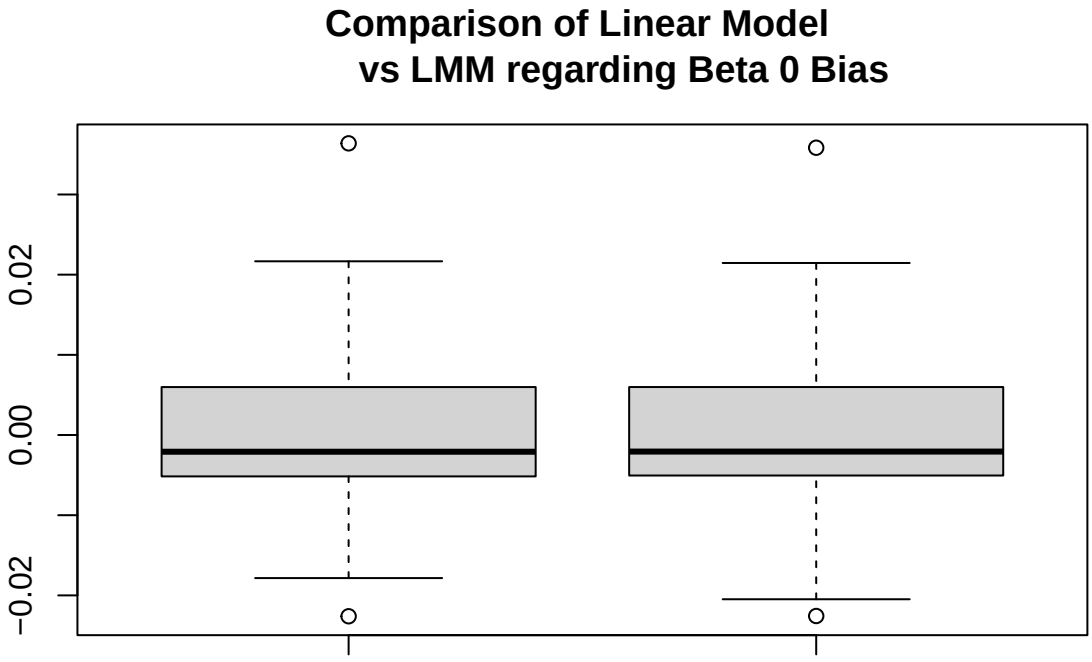
```
##     m  n   p linear_coef0 linear_coef1  linear_bias0  linear_bias1 linear_cov0
## 1   5  5 0.1    0.9986468    0.5082425 -0.0013532415  8.242540e-03       0.890
## 2   5  5 0.5    0.9864076    0.5190366 -0.0135923821  1.903660e-02       0.728
## 3   5  5 0.9    0.9821576    0.4960046 -0.0178424184 -3.995401e-03       0.612
## 4   5 15 0.1    0.9856528    0.5008348 -0.0143472450  8.347617e-04       0.794
## 5   5 15 0.5    1.0087853    0.5007236  0.0087853252  7.235574e-04       0.494
## 6   5 15 0.9    0.9773952    0.4969391 -0.0226047780 -3.060881e-03       0.378
## 7   5 25 0.1    1.0016917    0.4904754  0.0016916622 -9.524557e-03       0.708
## 8   5 25 0.5    1.0363773    0.5030022  0.0363772961  3.002203e-03       0.384
## 9   5 25 0.9    1.0025301    0.4963214  0.0025300855 -3.678551e-03       0.262
## 10 15  5 0.1    0.9967769    0.5022584 -0.0032230975  2.258427e-03       0.902
## 11 15  5 0.5    1.0216672    0.4991169  0.0216671656 -8.830510e-04       0.722
## 12 15  5 0.9    0.9857180    0.4949163 -0.0142819705 -5.083721e-03       0.600
## 13 15 15 0.1    0.9967974    0.4970163 -0.0032026217 -2.983686e-03       0.770
## 14 15 15 0.5    1.0117786    0.4976338  0.0117785745 -2.366228e-03       0.472
## 15 15 15 0.9    1.0041771    0.4973859  0.0041771443 -2.614056e-03       0.374
## 16 15 25 0.1    0.9979169    0.5005838 -0.0020831035  5.838435e-04       0.694
## 17 15 25 0.5    1.0120345    0.4994008  0.0120344665 -5.991628e-04       0.404
## 18 15 25 0.9    1.0190234    0.4994844  0.0190233856 -5.156342e-04       0.348
## 19 25  5 0.1    1.0035050    0.4960667  0.0035050481 -3.933300e-03       0.912
## 20 25  5 0.5    1.0001677    0.4997901  0.0001676773 -2.098639e-04       0.718
## 21 25  5 0.9    0.9957779    0.4929439 -0.0042220754 -7.056097e-03       0.630
## 22 25 15 0.1    0.9972542    0.5003075 -0.0027457673  3.074718e-04       0.786
## 23 25 15 0.5    0.9965238    0.5010894 -0.0034762249  1.089448e-03       0.514
## 24 25 15 0.9    0.9890055    0.4966740 -0.0109945299 -3.325951e-03       0.402
## 25 25 25 0.1    1.0077873    0.5029302  0.0077873421  2.930235e-03       0.674
## 26 25 25 0.5    0.9962206    0.4999372 -0.0037793867 -6.280942e-05       0.428
## 27 25 25 0.9    0.9939030    0.4979558 -0.0060970000 -2.044198e-03       0.300
##    linear_cov1 lmm_coef0 lmm_coef1     lmm_bias0     lmm_bias1 lmm_cov0
## 1        0.962 0.9988081 0.5046019 -0.0011919109  4.601917e-03    0.916
## 2        0.940 0.9867552 0.5098847 -0.0132448086  9.884719e-03    0.870
## 3        0.958 0.9795170 0.5011491 -0.0204829806  1.149146e-03    0.870
## 4        0.952 0.9858147 0.5033484 -0.0141853258  3.348410e-03    0.900
## 5        0.954 1.0086775 0.4990045  0.0086774556 -9.954690e-04    0.870
## 6        0.942 0.9774258 0.5012969 -0.0225742480  1.296890e-03    0.876
## 7        0.948 1.0018086 0.4907650  0.0018085513 -9.234961e-03    0.890
## 8        0.962 1.0358314 0.5038424  0.0358314278  3.842367e-03    0.878
## 9        0.934 1.0020499 0.5006975  0.0020498729  6.974751e-04    0.898
## 10       0.956 0.9965821 0.5014848 -0.0034179297  1.484755e-03    0.932
## 11       0.954 1.0214589 0.4986204  0.0214589154 -1.379578e-03    0.922
## 12       0.962 0.9858812 0.5008384 -0.0141187806  8.383753e-04    0.936
## 13       0.958 0.9968814 0.4980697 -0.0031186015 -1.930338e-03    0.924
## 14       0.956 1.0120030 0.4981338  0.0120030350 -1.866164e-03    0.918
```

```
## 15       0.948 1.0041808 0.4987060  0.0041807889 -1.293983e-03    0.926
## 16       0.952 0.9979373 0.4999476 -0.0020626559 -5.238571e-05    0.954
## 17       0.950 1.0119541 0.5011155  0.0119540973  1.115487e-03    0.928
## 18       0.940 1.0189045 0.4999841  0.0189044610 -1.593090e-05    0.948
## 19       0.960 1.0034399 0.4955323  0.0034398867 -4.467709e-03    0.942
## 20       0.960 1.0001321 0.4965407  0.0001320918 -3.459323e-03    0.932
## 21       0.958 0.9960071 0.4977568 -0.0039929165 -2.243159e-03    0.940
## 22       0.946 0.9972976 0.5001805 -0.0027023940  1.805489e-04    0.942
## 23       0.946 0.9965905 0.5005934 -0.0034095028  5.934107e-04    0.948
## 24       0.932 0.9890156 0.5000638 -0.0109844223  6.381073e-05    0.940
## 25       0.952 1.0077876 0.5022918  0.0077875581  2.291810e-03    0.932
## 26       0.978 0.9962168 0.4991240 -0.0037832251 -8.759581e-04    0.936
## 27       0.944 0.9938896 0.4997636 -0.0061103540 -2.363547e-04    0.926
##    lmm_cov1
## 1     0.944
## 2     0.940
## 3     0.936
## 4     0.946
## 5     0.944
## 6     0.952
## 7     0.942
## 8     0.970
## 9     0.960
## 10    0.946
## 11    0.946
## 12    0.942
## 13    0.958
## 14    0.956
## 15    0.952
## 16    0.950
## 17    0.924
## 18    0.954
## 19    0.948
## 20    0.946
## 21    0.936
## 22    0.942
## 23    0.954
## 24    0.960
## 25    0.950
## 26    0.954
## 27    0.948
```

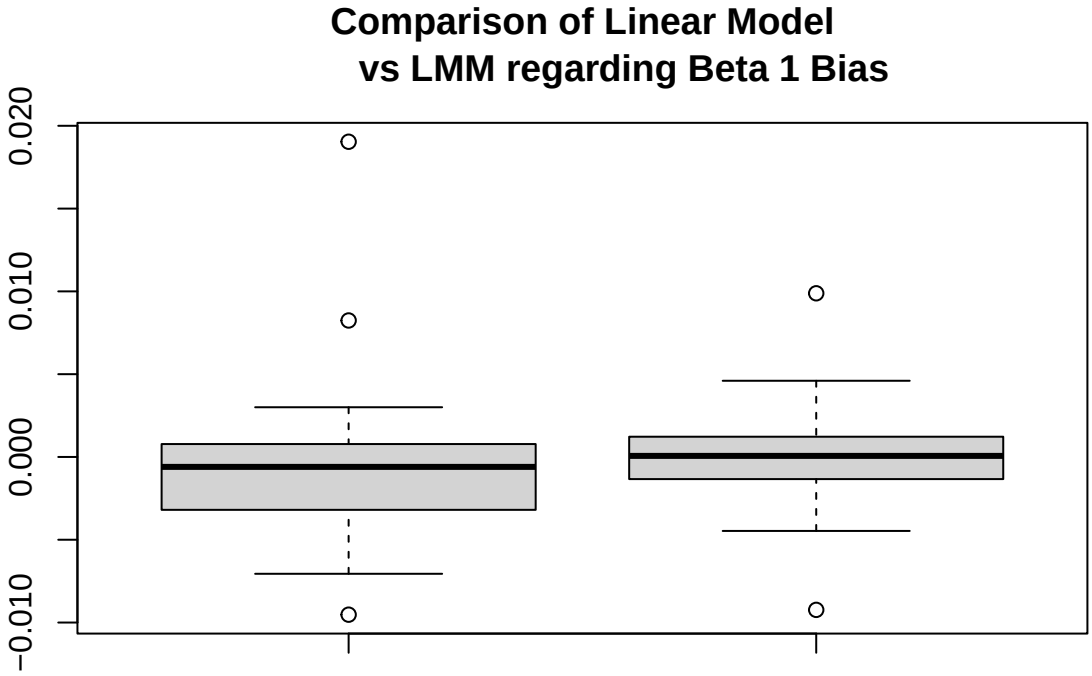
```
library(ggplot2)
attach(result_df)
```

```
## The following objects are masked _by_ .GlobalEnv:
##
##     m, n, p
```

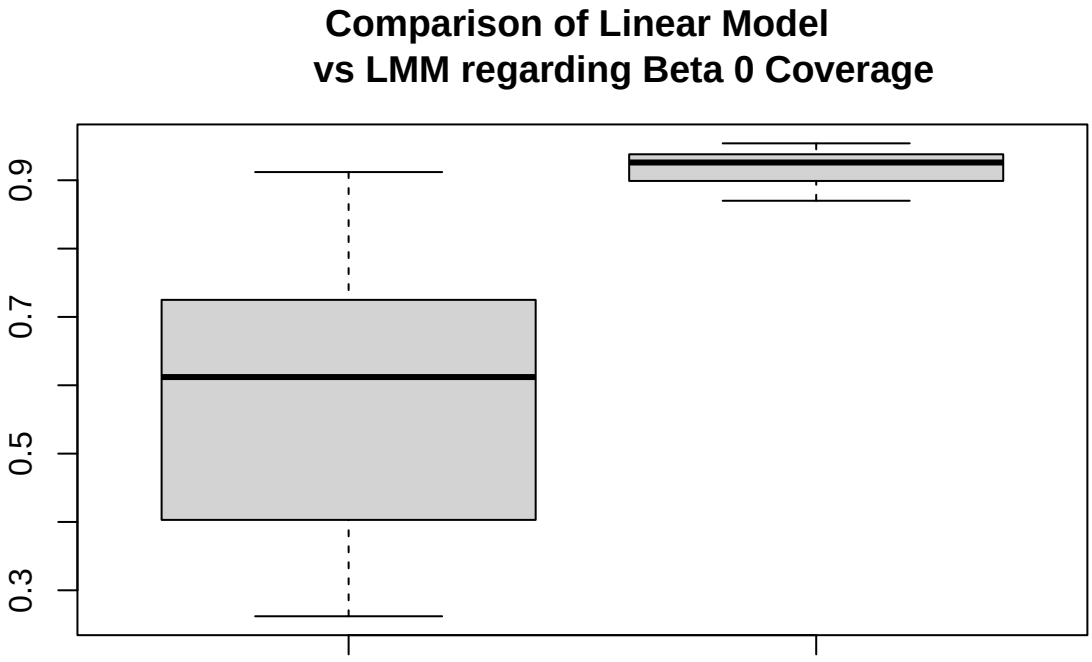
```
boxplot(linear_bias0, lmm_bias0, main = "Comparison of Linear Model
        vs LMM regarding Beta 0 Bias")
```



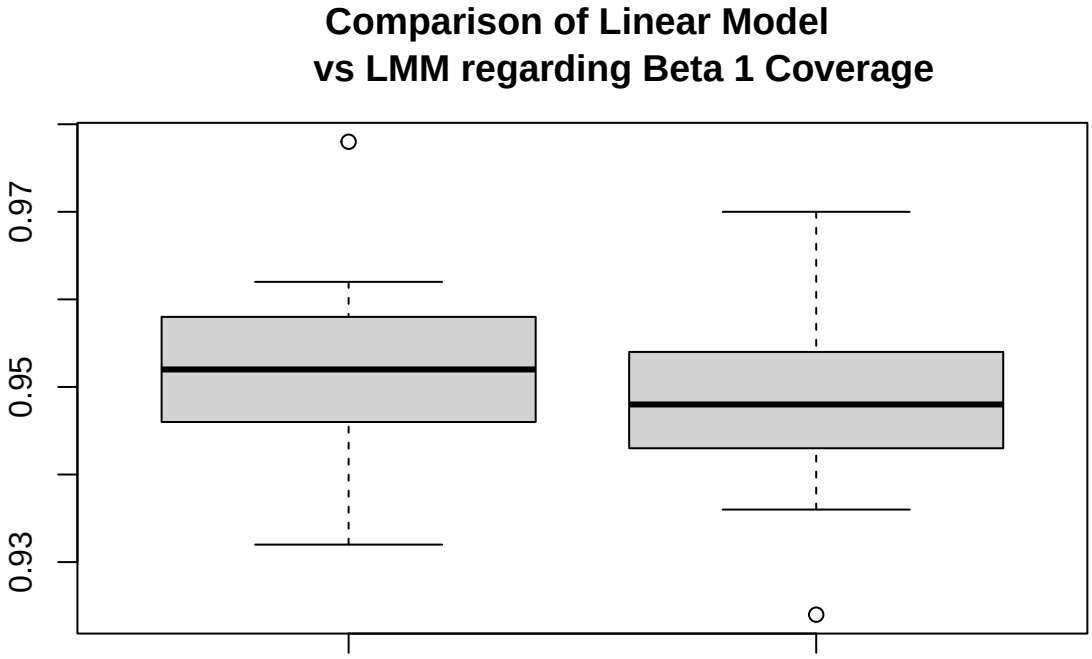

```
boxplot(linear_bias1, lmm_bias1, main = "Comparison of Linear Model
        vs LMM regarding Beta 1 Bias")
```



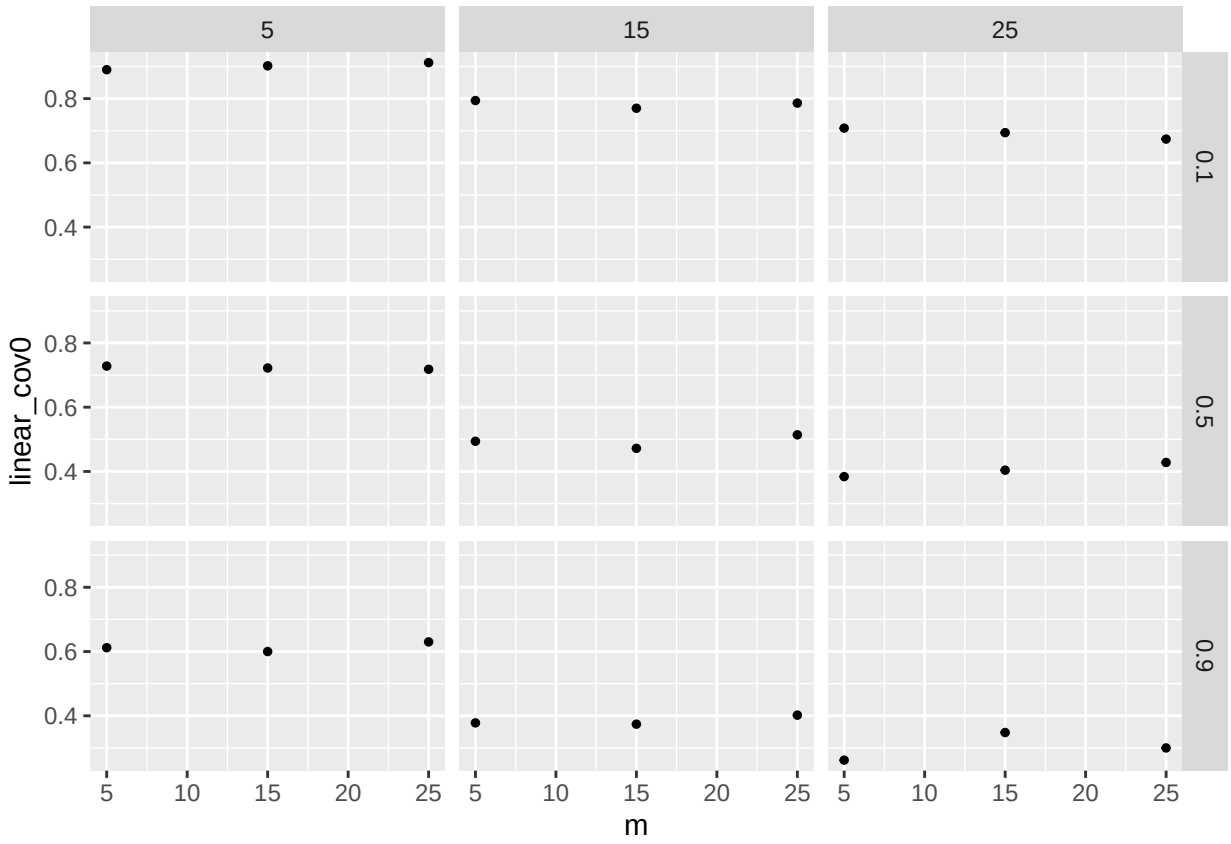

```
boxplot(linear_cov0, lmm_cov0, main = "Comparison of Linear Model
        vs LMM regarding Beta 0 Coverage")
```



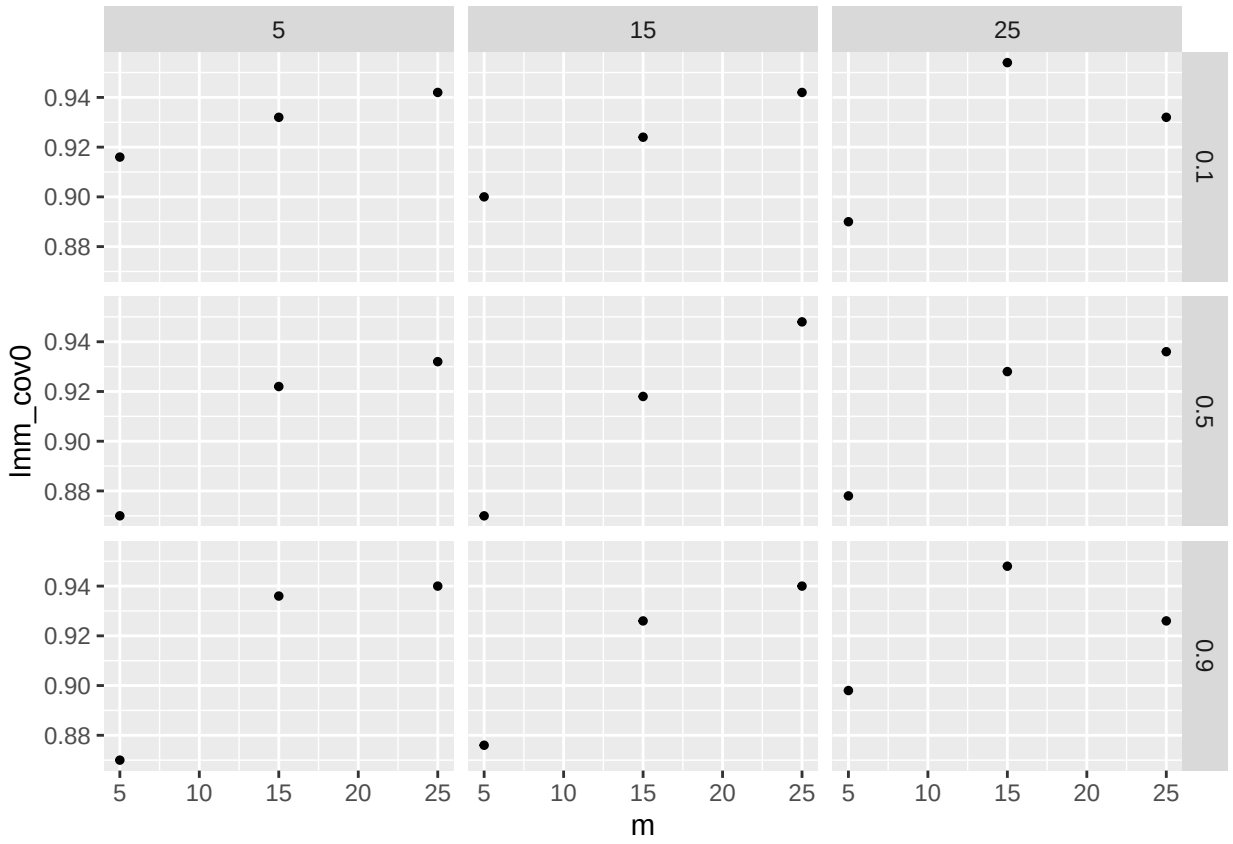

```
boxplot(linear_cov1, lmm_cov1, main = "Comparison of Linear Model
        vs LMM regarding Beta 1 Coverage")
```



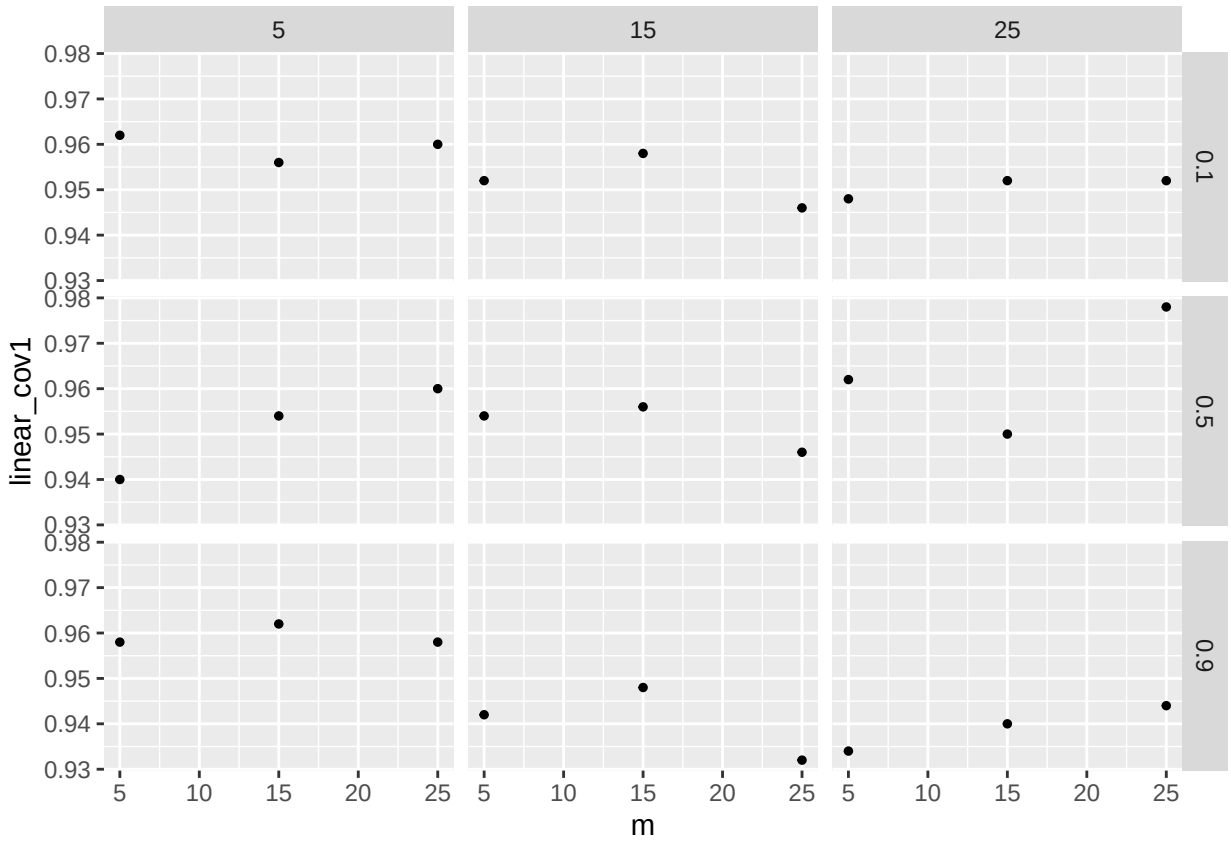

```
ggplot(data=result_df, aes(x=m, y=linear_cov0))+geom_point(lwd=1)+
  facet_grid(cols=vars(n),rows=vars(p))
```



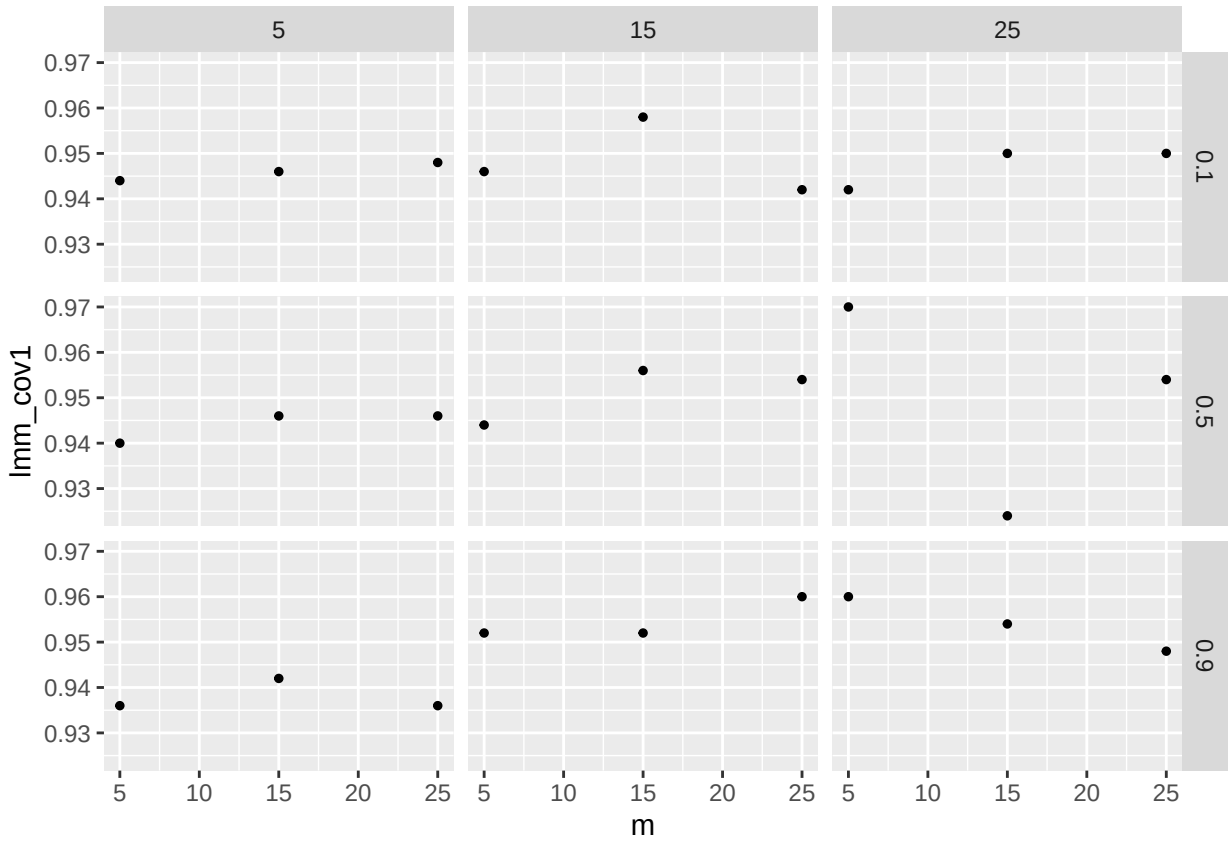

```
ggplot(data=result_df, aes(x=m, y=lmm_cov0))+geom_point(lwd=1)+
  facet_grid(cols=vars(n),rows=vars(p))
```




```
ggplot(data=result_df, aes(x=m, y=linear_cov1))+geom_point(lwd=1)+
  facet_grid(cols=vars(n),rows=vars(p))
```




```
ggplot(data=result_df, aes(x=m, y=lmm_cov1))+geom_point(lwd=1)+
  facet_grid(cols=vars(n),rows=vars(p))
```




```
detach(result_df)
```

In this simulation analysis, I set up 2 models. The simple model is the independent model of simple linear regression, which assumes independence between the response and the independent variable. The second model assumes correlation and is LMM. For the first part, the type I error is defined as the probability of rejecting the null hypothesis when it is true. In other words, given our null hypothesis provided by $\beta_0 = 1$ and $\beta_1 = 0.5$, we want to estimate how often the first model fails to cover those two true values with the 95 % confidence interval. As for the second model, I will check the bias (how much the estimation deviates from the true parameters) as well as the confidence interval coverage.

For completeness, I have included all the metrics for both models, namely beta estimations, biases, and confidence interval coverage for both parameters. As a result, given the table and box plots above, we can easily observe the following. As we can easily observe from the results above, the coverage for the LMM is significantly better than the results from the simple linear regression, which ignores the fact that there are some correlation in the data. The bias, on the other hand, is significantly lower for the LMM model both for beta 1. Also, for beta 0, the coverage of linear regression if off by a lot, but the LMM works pretty well. However, the coverage for beta 1 are about the same, and the bias for beta 0 are about the same. The reason is probably that since in both cases they are linear models, the slope beta 1 can be captured easily. But since the random effect b has been treated as some noise in the simple linear regression, the estimation for that would be a bit off.

Also shown on the last four graphs, we can see that the only clear trend that is here is as n increases, coverage for beta 0 of linear regression decreases significantly. Ohter than this, there is no clear relationship yet regarding coverage with respect to m,n,p in my result.

Question 2

```
# For testing
m = 10
n = 5
p = 0.5

gen.logi <- function(m,n,p){

  total = m*n

  # Set beta to 0.5
  beta1 = 0.5
  beta0 = 1

  # Generate the variables
  #x = rnorm(total, 0,  1)
  #b = rep(rnorm(m, mean = 2, sd = 3),n)
  #e = rnorm(total, mean = 0, sd = 5)
  #expo = beta0 + beta1*x + b + e
  #prob_y = exp(expo)/(1+exp(expo))
  #temp_y = runif(total)
  #y = ifelse(temp_y >= prob_y, 0, 1)

  # Generate the variables
  x = rnorm(total, 0, 1)
  b = rep(rnorm(m, mean = 0, sd = sqrt(p)),n)
  e = rnorm(total, mean = 0, sd = sqrt(1-p))
  expo = beta0 + beta1*x + b + e
  prob_y = exp(expo)/(1+exp(expo))
  temp_y = runif(total)
  y = ifelse(temp_y >= prob_y, 0, 1)

  # Linear regression
  logi_model = glm(y ~ x, family = "binomial")
  logi_coef0 = coef(logi_model)[1]
  logi_coef1 = coef(logi_model)[2]
  logi_bias0 = logi_coef0 - beta0
  logi_bias1 = logi_coef1 - beta1
  conf_logi = confint(logi_model)
  logi_cov0 = as.numeric(conf_logi[1,1] < beta0 & beta0 < conf_logi[1,2])
  logi_cov1 = as.numeric(conf_logi[2,1] < beta1 & beta1 < conf_logi[2,2])

  # LMM
  lmm = glmer(y ~ x  + (1 | b), family = binomial(link = "logit"),
              control = glmerControl(tolPwrss=1e-3))
  lmm_coef0 = fixef(lmm)[1]
  lmm_coef1 = fixef(lmm)[2]
  lmm_bias0 = lmm_coef0 - beta0
  lmm_bias1 = lmm_coef1 - beta1
  conf_lmm = confint.merMod(lmm, method = "Wald")
  lmm_cov0 = as.numeric(conf_lmm[2,1] < beta0 & beta0 < conf_lmm[2,2])
  lmm_cov1 = as.numeric(conf_lmm[3,1] < beta1 & beta1 < conf_lmm[3,2])
```

```
  return(c(logi_coef0, logi_coef1, logi_bias0, logi_bias1, logi_cov0, logi_cov1,
           lmm_coef0, lmm_coef1, lmm_bias0, lmm_bias1, lmm_cov0, lmm_cov1))
}
```

```
nsim = 500
m = seq(5,15,5)
n = seq(5,15,5)
p = 0.5

results1 <- vector("list", length = length(m) * length(n))
for (i in 1:length(m)){

  for (j in 1:length(n)){
    result <- c()
    for (k in 1:length(p)){
      res <- replicate(nsim, gen.logi(m[i],n[j],p[k]))
      mean_res <- rowMeans(res)
      result <- c(result, m[i],n[j],p[k], mean_res)
    }
    result <- matrix(result, ncol = 15, byrow = T)
    results1[[(i-1)*length(m) + j]] = result
  }

}
#results1
```

```
#total_len = length(m) * length(n) * length(p)
#m_var = rep(m, total_len/length(m))
#n_var = rep(n, total_len/length(n))
#p_var = rep(p, total_len/length(p))

result_df1 = do.call(rbind.data.frame, results1)
#result_df_res1 = cbind(m_var, n_var,p_var,result_df1)
result_df1
```

```
##   V1 V2  V3        V4        V5          V6          V7    V8    V9       V10
## 1  5  5 0.5 1.3821999 0.8387231  0.38219987  0.33872308 0.892 0.940 2.4539776
## 2  5 10 0.5 0.8686865 0.4863904 -0.13131352 -0.01360957 0.848 0.942 0.9541301
## 3  5 15 0.5 0.9005204 0.4693880 -0.09947964 -0.03061201 0.798 0.920 0.9778121
## 4 10  5 0.5 0.9073204 0.4599088 -0.09267960 -0.04009119 0.872 0.930 1.1506825
## 5 10 10 0.5 0.8884488 0.4528768 -0.11155125 -0.04712324 0.820 0.918 0.9550619
## 6 10 15 0.5 0.8629304 0.4404056 -0.13706957 -0.05959437 0.770 0.932 0.9297325
## 7 15  5 0.5 0.8629575 0.4671787 -0.13704254 -0.03282133 0.872 0.964 0.9389647
## 8 15 10 0.5 0.8530823 0.4505224 -0.14691768 -0.04947763 0.784 0.936 0.9231777
## 9 15 15 0.5 0.8685531 0.4245127 -0.13144691 -0.07548731 0.752 0.916 0.9426605
##         V11         V12         V13   V14   V15
## 1 1.5018784  1.45397757  1.00187844 0.944 0.972
## 2 0.5344783 -0.04586991  0.03447831 0.908 0.966
## 3 0.5105433 -0.02218793  0.01054326 0.916 0.934
## 4 0.5589458  0.15068250  0.05894579 0.916 0.952
## 5 0.4860590 -0.04493809 -0.01394101 0.908 0.928
## 6 0.4745528 -0.07026745 -0.02544717 0.896 0.956
## 7 0.5069594 -0.06103527  0.00695937 0.924 0.972
## 8 0.4865585 -0.07682232 -0.01344149 0.914 0.942
## 9 0.4596995 -0.05733951 -0.04030047 0.914 0.944
```

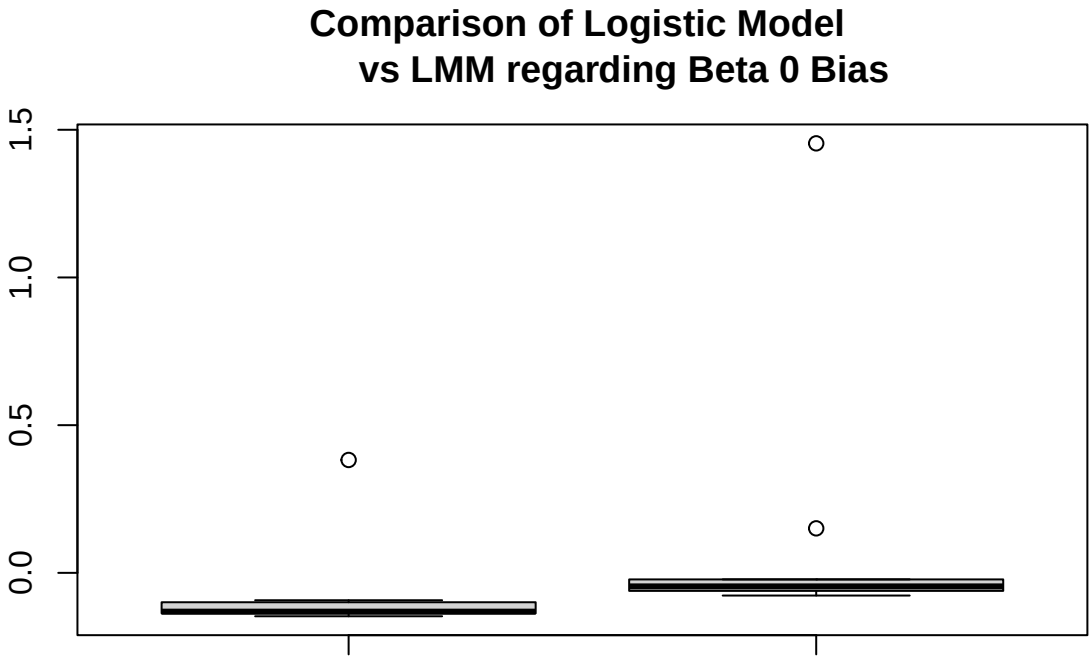
```
colnames(result_df1) = c("m", "n", "p", "logi_coef0", "logi_coef1",
                         "logi_bias0", "logi_bias1", "logi_cov0",
                         "logi_cov1", "lmm_coef0", "lmm_coef1",
                         "lmm_bias0", "lmm_bias1", "lmm_cov0", "lmm_cov1")
result_df1
```

```
##    m  n   p logi_coef0 logi_coef1  logi_bias0  logi_bias1 logi_cov0 logi_cov1
## 1  5  5 0.5  1.3821999  0.8387231  0.38219987  0.33872308     0.892     0.940
## 2  5 10 0.5  0.8686865  0.4863904 -0.13131352 -0.01360957     0.848     0.942
## 3  5 15 0.5  0.9005204  0.4693880 -0.09947964 -0.03061201     0.798     0.920
## 4 10  5 0.5  0.9073204  0.4599088 -0.09267960 -0.04009119     0.872     0.930
## 5 10 10 0.5  0.8884488  0.4528768 -0.11155125 -0.04712324     0.820     0.918
## 6 10 15 0.5  0.8629304  0.4404056 -0.13706957 -0.05959437     0.770     0.932
## 7 15  5 0.5  0.8629575  0.4671787 -0.13704254 -0.03282133     0.872     0.964
## 8 15 10 0.5  0.8530823  0.4505224 -0.14691768 -0.04947763     0.784     0.936
## 9 15 15 0.5  0.8685531  0.4245127 -0.13144691 -0.07548731     0.752     0.916
##   lmm_coef0 lmm_coef1   lmm_bias0   lmm_bias1 lmm_cov0 lmm_cov1
## 1 2.4539776 1.5018784  1.45397757  1.00187844    0.944    0.972
## 2 0.9541301 0.5344783 -0.04586991  0.03447831    0.908    0.966
## 3 0.9778121 0.5105433 -0.02218793  0.01054326    0.916    0.934
## 4 1.1506825 0.5589458  0.15068250  0.05894579    0.916    0.952
## 5 0.9550619 0.4860590 -0.04493809 -0.01394101    0.908    0.928
## 6 0.9297325 0.4745528 -0.07026745 -0.02544717    0.896    0.956
## 7 0.9389647 0.5069594 -0.06103527  0.00695937    0.924    0.972
## 8 0.9231777 0.4865585 -0.07682232 -0.01344149    0.914    0.942
## 9 0.9426605 0.4596995 -0.05733951 -0.04030047    0.914    0.944
```

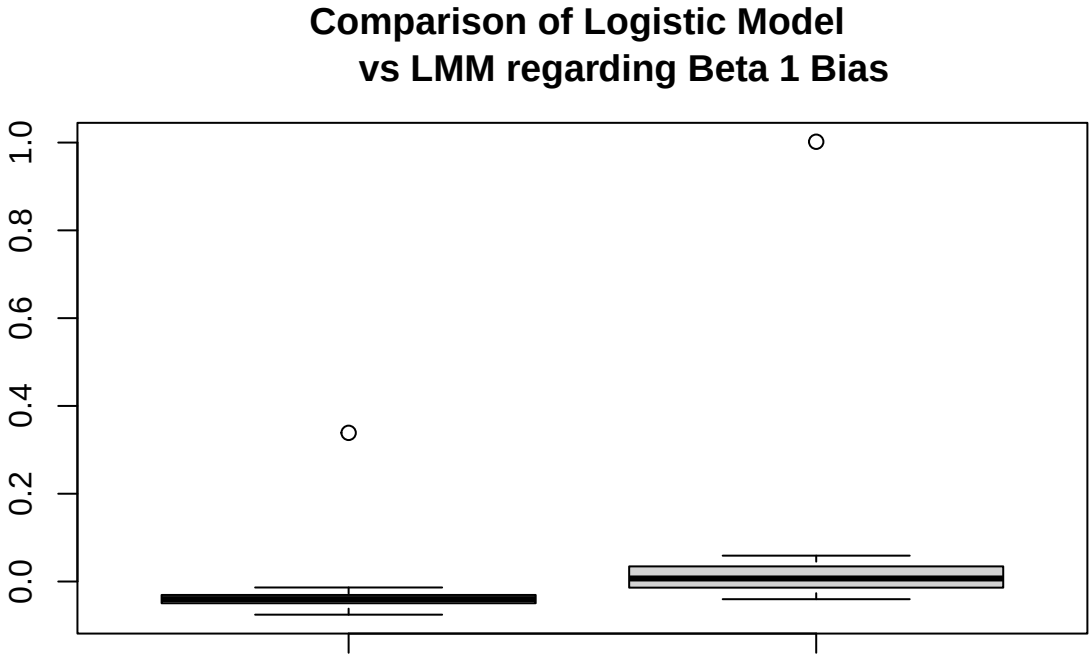
```
library(ggplot2)
attach(result_df1)
```

```
## The following objects are masked _by_ .GlobalEnv:
##
##     m, n, p
```

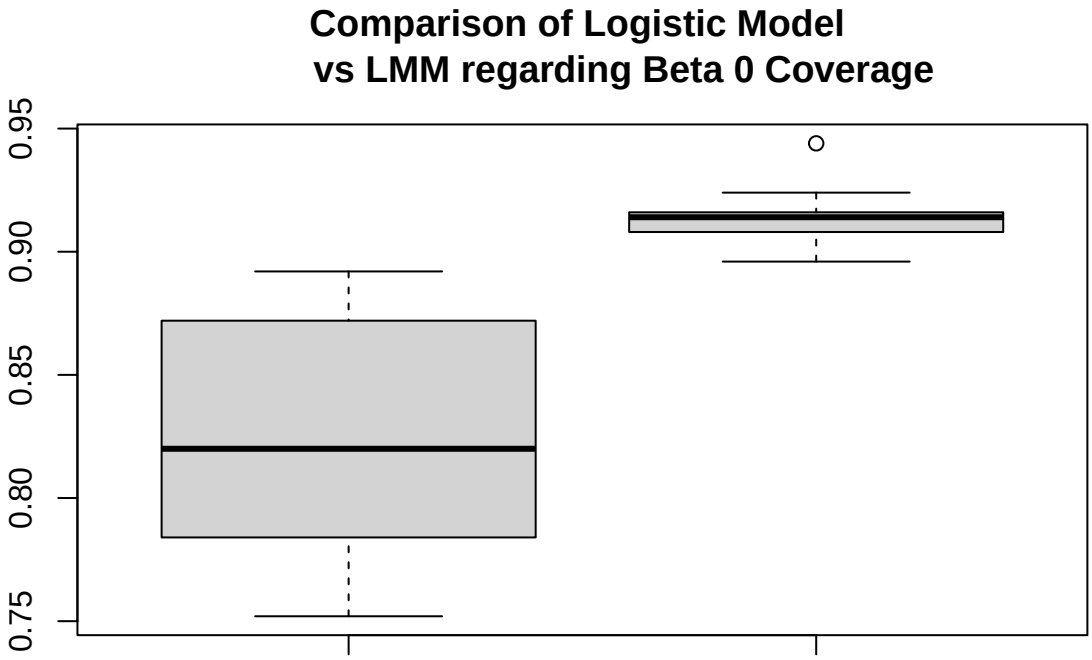
```
boxplot(logi_bias0, lmm_bias0, main = "Comparison of Logistic Model
        vs LMM regarding Beta 0 Bias")
```



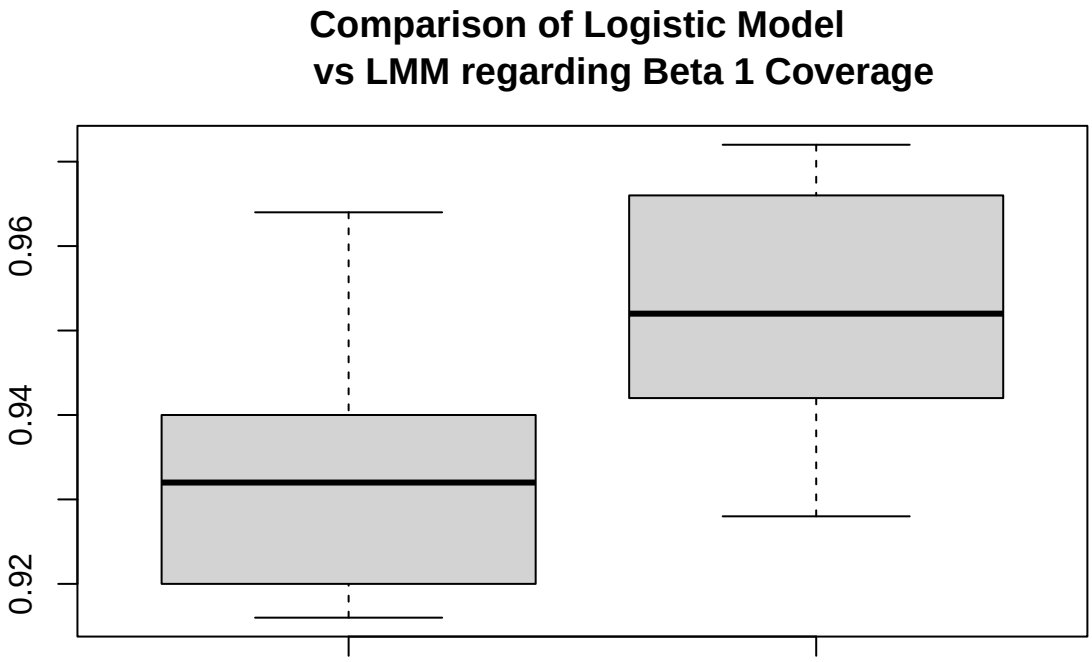

```
boxplot(logi_bias1, lmm_bias1, main = "Comparison of Logistic Model
        vs LMM regarding Beta 1 Bias")
```



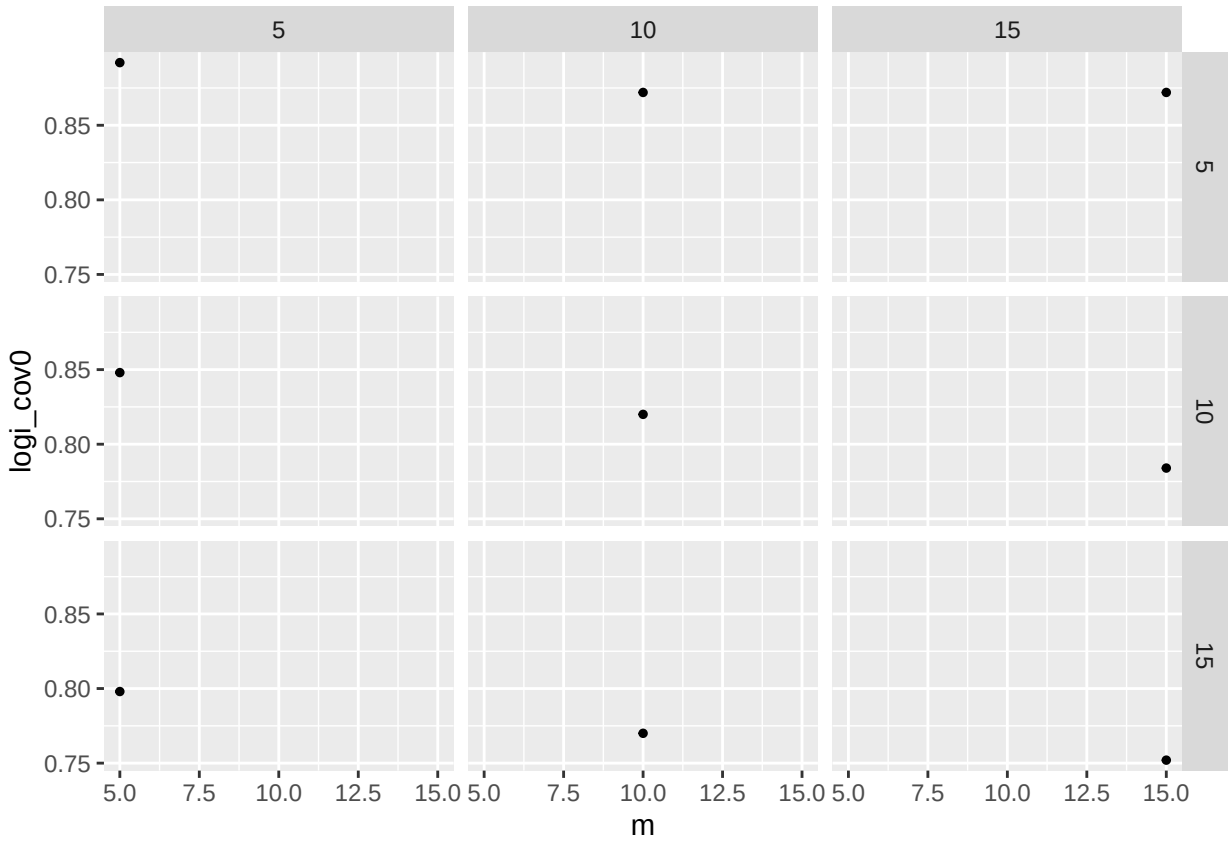

```
boxplot(logi_cov0, lmm_cov0, main = "Comparison of Logistic Model
        vs LMM regarding Beta 0 Coverage")
```



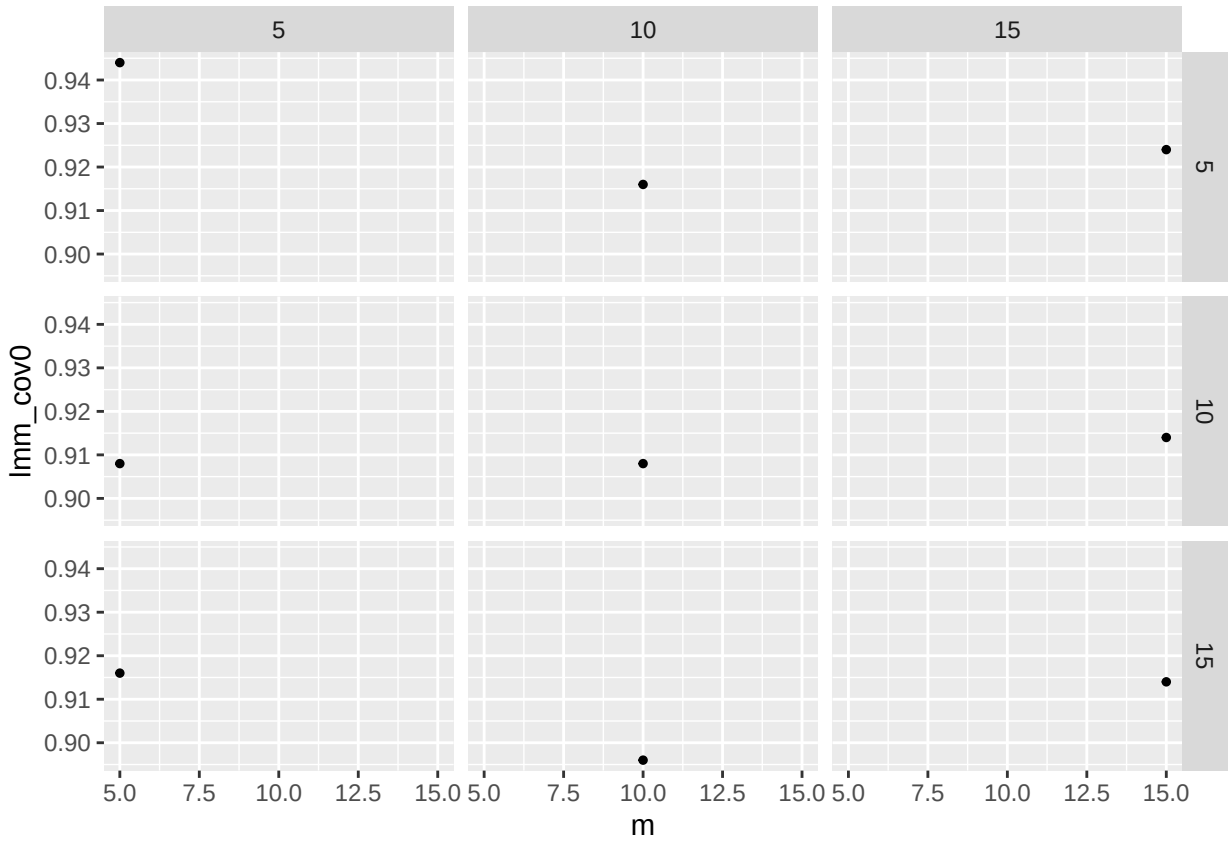

```
boxplot(logi_cov1, lmm_cov1, main = "Comparison of Logistic Model
        vs LMM regarding Beta 1 Coverage")
```



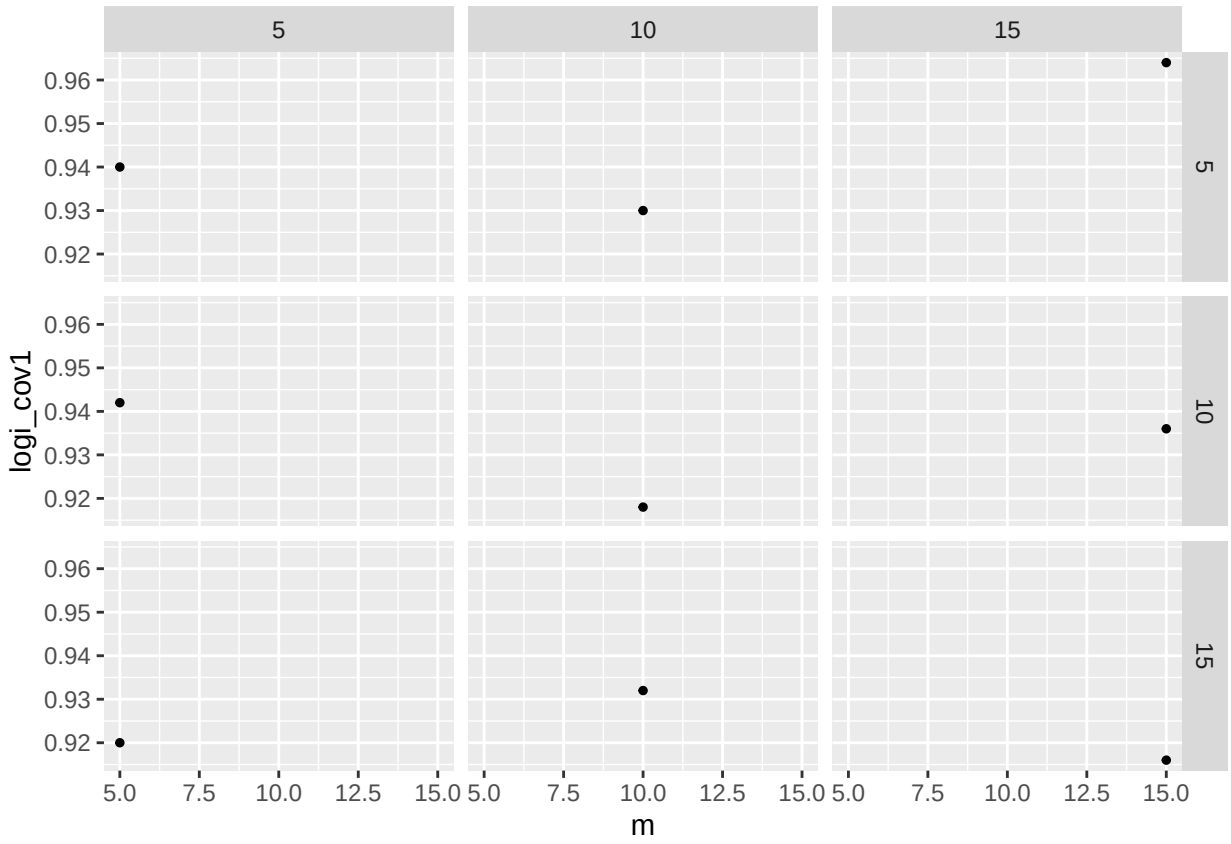

```
ggplot(data=result_df1, aes(x=m, y=logi_cov0))+geom_point(lwd=1)+
  facet_grid(cols=vars(m),rows=vars(n))
```



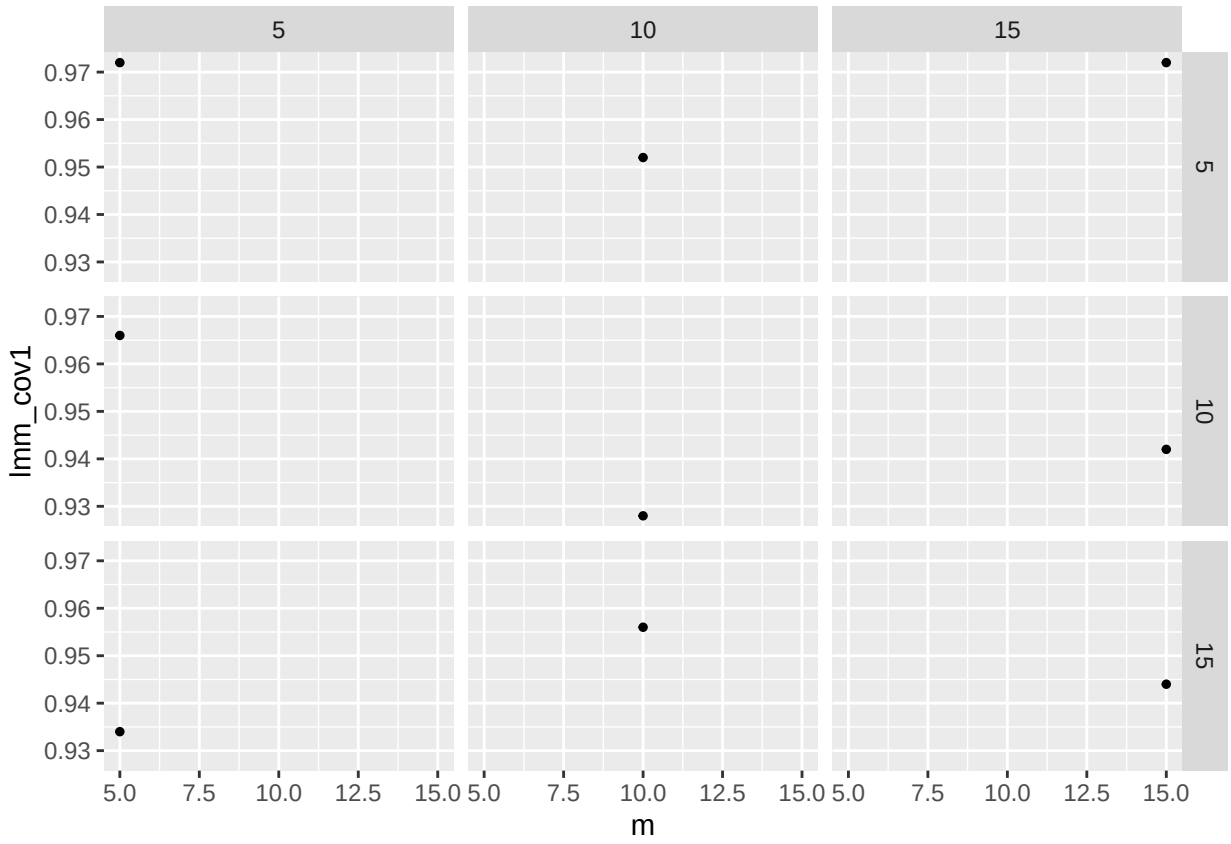

```
ggplot(data=result_df1, aes(x=m, y=lmm_cov0))+geom_point(lwd=1)+
  facet_grid(cols=vars(m),rows=vars(n))
```




```
ggplot(data=result_df1, aes(x=m, y=logi_cov1))+geom_point(lwd=1)+
  facet_grid(cols=vars(m),rows=vars(n))
```




```
ggplot(data=result_df1, aes(x=m, y=lmm_cov1))+geom_point(lwd=1)+
  facet_grid(cols=vars(m),rows=vars(n))
```




```
detach(result_df1)
```

Based on the results above, we have obtained similar insight as from the previous question. When the data is correlated, we should better use the correlated model rather than the independent model. This time, the coverage and biases for the GLMM is strongly preferable to those of the simple logistic model. The improvement in the beta 1 is potentially due to the non linear relationship, as compared with the linear case in question 1. There are no obvious trend in terms of m,n values though.

Question 3

If we do have multiple outcomes, we can definitely take a weighted average of the ourcome measures. But there are indeed a few pros and cons to this approach. The benefits of doing this weighted approach include: (1) easy to practice, (2) easy to explain, (3) contains all necessary/important information. To provide more detail, taking a weighted average makes things really easy. Since all we need to do is to do a bunch of calculations linearly, we save a lot of time compared with devising a complicated statistical model or gathering data. Furthermore, it is easy to let the audience know what the aggregate outcome variable contains and basically what information is in there with this approach. It's simply mixing the importance of all the different variables and make a comprehensive conclusion. Furthermore, it does not miss out some potential important information since everything is covered.

There are however some downsides to this approach. For example, (1) we do not know the proper ways to assign the weights; (2) many fields cannot be combined together since practically that would not make sense; (3) the association and causation relationships can be blurred. To provide more details, first I would emphasize that we do not know a priori how much weight to assign to each of the variables, all we can do is guess or make they weights equal. This, however, would very likely not reflect the reality: the true relationships might be that some outcomes have more play as a result of the independent variables while other outcomes are just subsidiary. Moreover, many fields are totally separate and cannot be linked together by force. We can not mix multiple outcomes and make them just one just for the convenience of computation

in most cases: for example, we can't mix a person's income, mortality, happiness all together just by using weights if those are the things we would like to model. In addition, given multiple outcomes, the associations between the independent variables and the outcome can really be blurred. Especially when there are a lot of outcome variables, since the outcome variables themselves might be related to each other.

Given the pros and cons discussed above, the best time to use this weighted approach is when the benefits can be best exemplified and the costs can be minimized. That is, when the multivariate outcomes are correlated in a manner that they carry similar pratical sense, and seem equally important in our research, and the relationships are pretty clear, we can take advantage of the ease of modelling in this manner. Otherwise, we should be very cautious when applying this approach.