BIOSTAT 571 Homework 1

Wenxiao Gu

January 15, 2014

Contents

1 PR1

Main part of the question

$$
\begin{array}{rcl}
Z & \sim & \Gamma(k,\theta) \\
E(Z) & = & k\theta \\
Var(Z) & = & k\theta^2 \\
k & = & 1 \\
Y_i|x_i & \overset{ind.}{\sim} & \Gamma(1,\mu_i) \\
E(Y_i|x_i) & = & \mu_i, \text{ for } i = 1,2,...,n; \\
x_i & \sim & evenly\ distributed\ [-1,1]?\ (uniform?) \\
\mu_i & = & exp(\beta_0 + \beta_1 \cdot x_i)
\end{array}
$$

$$
\begin{array}{rcl}
(i) & Max-L & "model-based"\ in\ STAT\ 570 \\
(ii) & Quasi-L & "optimal"\ in\ STAT\ 570 \\
(iii) & Semi-Param. & "sandwich"\ in\ STAT\ 570
\end{array}
$$

1.1 Part (a)

Since $k = 1$, and $Y_i|x_i \overset{ind.}{\sim} \Gamma(1,\mu_i)$, then our model is correctly specified, because

$$
\begin{array}{rcl}
E(Y_i|x_i) & = & \mu_i = E\left[\Gamma(1,\mu_i)\right] \\
Mean\ of\ model & = & True\ mean.
\end{array}
$$

Thus the three settings of (i), (ii), and (iii), will all give the consistent estimates of β_1. For example of $\beta_0 = 1$ and $\beta_1 = 0.5$, the results from the three settings are stated as following,

	ML	Quasi-L	Semi-param
$\hat{\beta}_0$	.9949	.9949	.9949
$\hat{\beta}_1$	.6993	.6993	.6993

1.2 Part (b)

As we have stated in part (a), all of the three settings have correctly specified the mean model. Next, we look into the mean-variance relationship. The true mean-variance relationship for the Gamma distribution is $Var(Y_i|x_i) = \mu_i^2 = exp[2(\beta_0 + \beta_1 x_i)]$. However, the three settings have different settings of the mean-variance relationship, as following,

- The ML Poisson regression assumes

  $$Var(Y_i|x_i) = \mu_i;$$

  The mean-variance relationship in the ML Poisson regression has been incorrectly specified. So, this model is not guaranteed to yield asymptotically valid standard error estimates for β_1 and 95% confidence intervals with asymptotically correct coverage.

- The Qausi-Likelihood Poisson regression assumes

  $$Var(Y_i|x_i) = \phi \cdot \mu_i;$$

  The mean-variance relationship in the Qausi-Likelihood Poisson regression has been incorrectly specified. So,this model is not guaranteed to yield asymptotically valid standard error estimates for β_1 and 95% confidence intervals with asymptotically correct coverage.

- The Semi-Parameter Poisson regression only need a correct mean model. Since the mean model is correctly specified, sandwich standard errors for β_1 can be asymptotically valid standard error estimates and give 95% confidence intervals with asymptotically correct coverage.

1.3 Part (c)

In Part(c) and Part(d), I first make a large data set with the expansion of different simulation times, different sample sizes, and different true β_0's and β_1's.

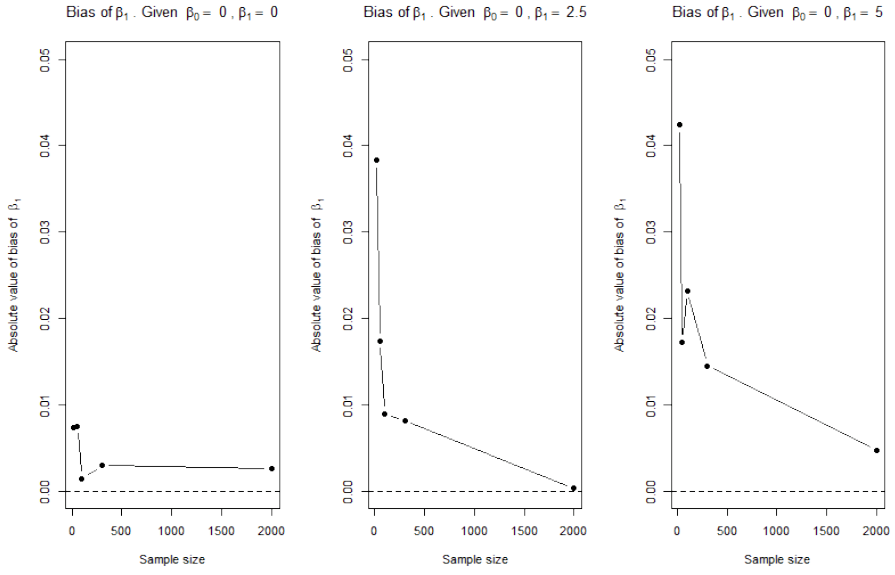
1.3.1 The set-up of the simulated data

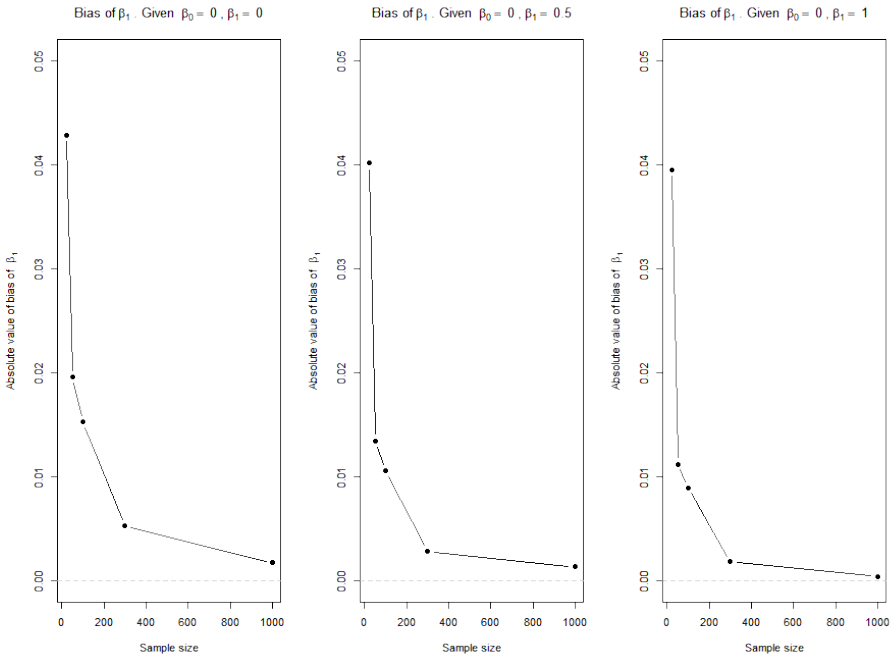
- There are $\beta_0 = 0, 0.5, 1, 2$, and $\beta_1 = 0, 0.5, 1, 2.5, 5$.
- The choices of sample sizes are 25, 50, 100, 300, 1000, and 2,000.
- The choices of number of simulations are 50, 100, 500 and 1,000.

1.3.2 The output of the simulated data

- $\hat{\beta}_0$'s and $\hat{\beta}_1$'s.
- The coverage of $\beta_0's$ and $\beta_1's$, under three settings: Maximum likelihood, Quasi likelihood, and semi-parametric.

In Part (a), we state that the three settings of (i), (ii), and (iii), will all give the consistent estimates of β_1. In other words, as the sample size gets larger, the estimate of β_1 should be approaching the true value of β_1. The following example shows that the consistency of β_1.

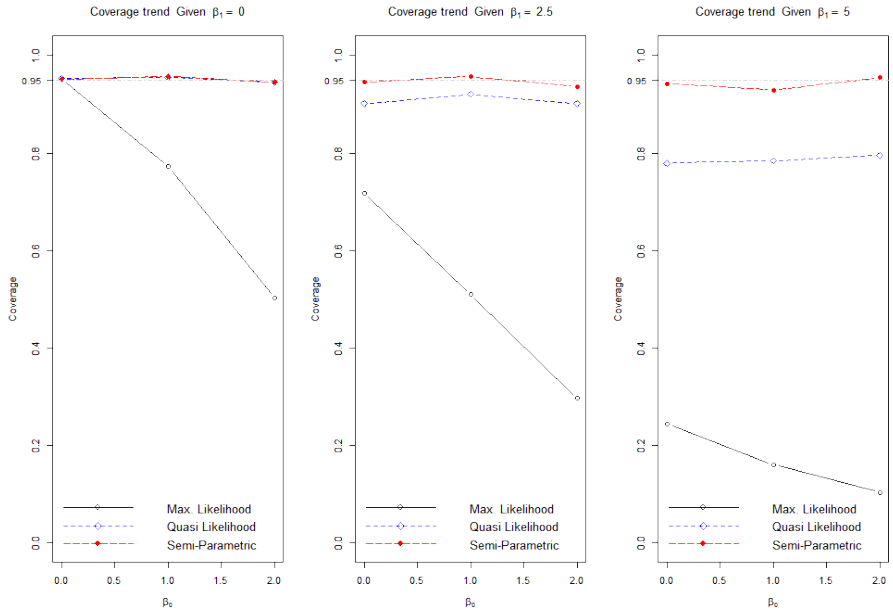




1.4 Part(d)

1.4.1 Simulation 1.

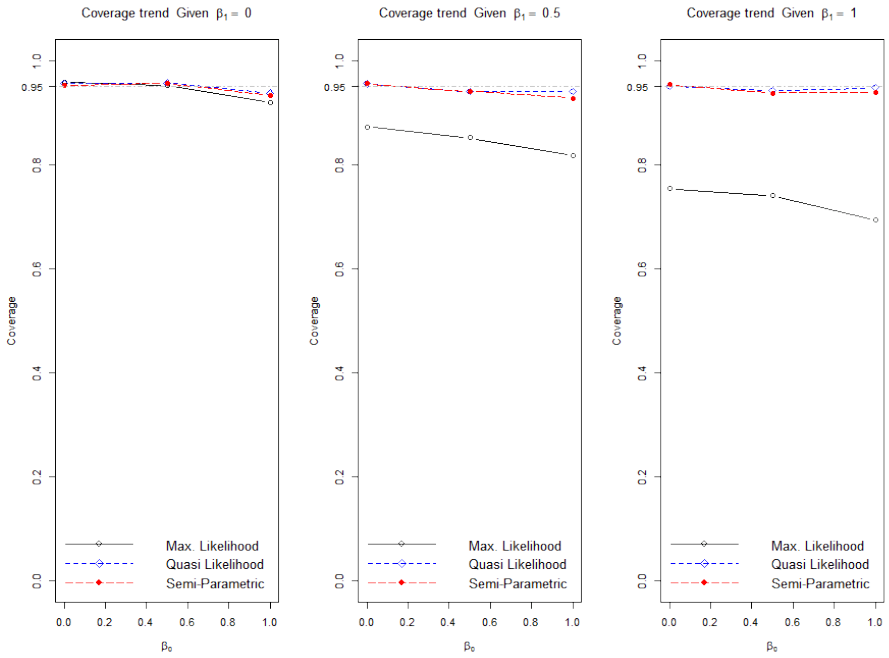
If $\beta_0 = 0$, and if $\beta_1 = 0$, then true mean-variance relationship for the Gamma distribution is $Var(Y_i|x_i) = \mu_i^2 = exp[2(\beta_0 + \beta_1 x_i)] = 1 = E(Y_i|x_i)$. Both true mean and true variance are equal to 1. In this case, the Maximum likelihood Poisson regression holds the true mean-variance relationship, as the Semi-parametric does. Thus, the model-based confidence intervals of the ML Poisson regression have the asymptotically correct coverage.



1.4.2 Simulation 2.

If $\beta_1 = 0$, then true mean-variance relationship for the Gamma distribution is $Var(Y_i|x_i) = \mu_i^2 = exp[2(\beta_0 + \beta_1 x_i)] = exp(\beta_0) \cdot exp(\beta_0) = exp(\beta_0) \cdot E(Y_i|x_i)$. Here we consider $\phi = exp(\beta_0)$ in the Quasi-likelihood Poisson regression.

Since the mean-variance relationship is correct if the constant is added. In this case, the Quasi-likelihood Poisson regression holds the true mean-variance relationship, as the Semi-parametric does. Thus, the confidence intervals of the Quasi-likelihood Poisson regression have the asymptotically correct coverage.



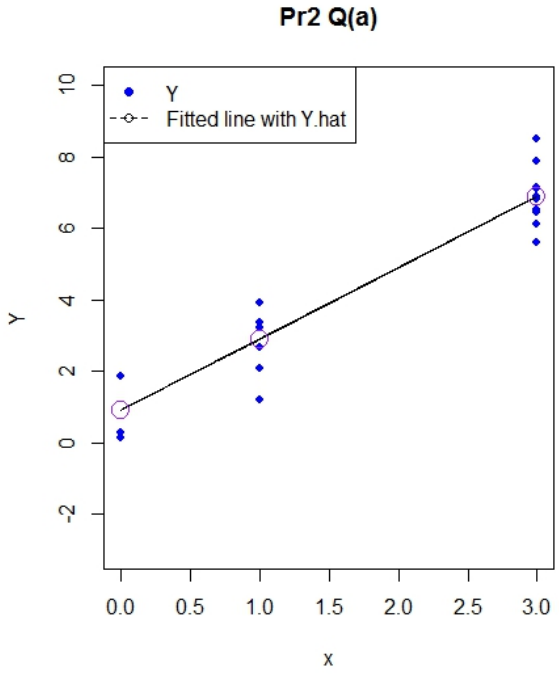
1.5 Part(e)

- The maximum-likelihood version of Poisson model assumes that the response Y has to be integer. However, the observed data of the response is not integer. The problem happens when calculating the maximum likelihood, because the Poisson distribution is for integers. But, I am not concerned about the validity of the maximum likelihood parameter estimates, because the results from the model-based results for both integer and non-integer data.
- When fitting the quasi-likelihood, there is no such messages, because the assumption of likelihood is not assumed. In this case, the relationship between mean and variance is assumed, and the dispersion parameter is of interest.

2 Pr2

2.1 Part(a)

From the graph below, I believe that the linear regression fits the data pretty well. In these data, the reason we can use linear regression is the obvious linear trend in the data. The mean at each x is linearly increasing from 0.6 to 6.8. The other reason is the equal variance of observed data at each x.



2.2 Part(b)

$$\mu(x) = \begin{cases} 0.6 & if\ x = 0 \\ 3.6 & if\ x = 1 \\ 6.8 & if\ x = 3 \end{cases}$$

$$\begin{aligned} I &= \int_x \left[\mu(x) - \beta_0 - x_i\beta_1\right]^2 \lambda(x)dx \\ &= \sum_x \left[\mu(x) - \beta_0 - x_i\beta_1\right]^2 \lambda(x) \\ &= \lambda\left\{\left[\mu(0) - \beta_0 - 0\cdot\beta_1\right]^2 + \left[\mu(1) - \beta_0 - 1\cdot\beta_1\right]^2 + \left[\mu(3) - \beta_0 - 3\cdot\beta_1\right]^2\right\} \\ &= \frac{1}{3}\left\{\left[0.6 - \beta_0\right]^2 + \left[3.6 - \beta_0 - \beta_1\right]^2 + \left[6.8 - \beta_0 - 3\beta_1\right]^2\right\} \end{aligned}$$

$$\begin{cases} \frac{dI}{d\beta_0} & = 2\left[\beta_0 - 0.6\right] + 2\left[\beta_0 + \beta_1 - 3.6\right] + 2\left[\beta_0 + 3\beta_1 - 6.8\right] = 0 \\ \frac{dI}{d\beta_1} & = 2\left[\beta_0 + \beta_1 - 3.6\right] + 6\left[\beta_0 + 3\beta_1 - 6.8\right] = 0 \end{cases}$$

$$\begin{cases} \hat{\beta}_0 & = 1 \\ \hat{\beta}_1 & = 2 \end{cases}$$

2.3 Part(c)

2.3.1 Assumption 1

$$\epsilon_i | x_i \sim N(\mu_i - \beta_0 - \beta_1 \cdot x_i, 1) \quad where\ \mu_i = \begin{cases} 0.6 - 1 - 2\times 0 = -0.4 & if\ x_i = 0 \\ 3.6 - 1 - 2\times 1 = 0.6 & if\ x_i = 1 \\ 6.8 - 1 - 2\times 3 = -0.2 & if\ x_i = 3 \end{cases}$$

As we see in the above distribution, the $\epsilon_i|x_i$ is not identically distributed. But all ϵ_i's given x_i are independent. Moreover,

$$\begin{aligned} E(\epsilon_i) &= E[x_i E(\epsilon_i|x_i)] \\ &= \sum_x \{x \cdot \mu\} \\ &= \frac{1}{3}[-0.4 + 0.6 - 0.2] \\ &= 0. \end{aligned}$$

2.3.2 Assumption 2

Let $\delta = 1$ and $\Delta = 1,000,000,000$. For all i,

$$\begin{aligned} E\left(\left|\epsilon_i^2\right|^{1+\delta}\right) &= E\left(\left|\epsilon_i^2\right|^2\right) \\ &= E\left(\epsilon_i^4\right) \\ &= E[x_i E(\epsilon_i^4|x_i)] \\ &= \sum_x \left\{x \cdot \left[E(\epsilon^2)^2 + Var(\epsilon^2)\right]\right\} \\ &= \sum_x \left\{x \cdot \left[E(\chi^2)^2 + Var(\chi^2)\right]\right\} \\ &< \sum_x \left\{x \cdot \left[E(\chi_1^2)^2 + Var(\chi_1^2)\right]\right\} \\ &= \Delta \end{aligned}$$

2.3.3 Lemma 1

Since both assumption 1 and 2 hold, then the Lemma 1 holds. Thus $\hat{\beta}$ gives an consistent estimate of β.

2.4 Part(d)

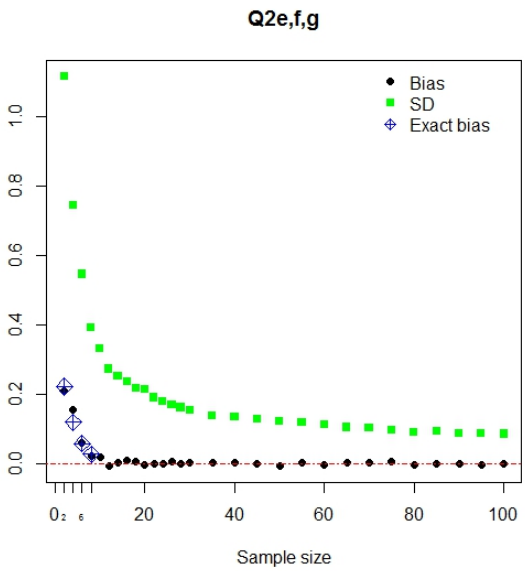
It is a biased estimate, because,

$$\begin{aligned} E(\hat{\beta}) &= E[(X^T X)^{-1} X E(Y|X)] \\ &= \beta + E\left[(X^T X)^{-1} X^T \epsilon\right] \\ &= \beta + (X^T X)^{-1} X^T E(\epsilon) \\ &\neq \beta \end{aligned}$$

2.5 Part(e, f)

The graph is in Part(g). Both standard deviation and bias decreases as the sample size goes larger. Because the estimate is consistent, then the bias goes smaller to zero.

2.6 Part(g)



The number of simulations is 1,500. The result of calculation is from R, shown below,

X	(0,1)	(0,3)	(1,3)
Y	(0.6,3.6)	(0.6,6.8)	(3.6,6.8)
$\hat{\beta}$	(0.6,3)	$(0.6, 2\frac{1}{15})$	(2,1.6)

And thus, the exact bias at n=2 is shown as below,

$$\begin{aligned} E(\hat{\beta}_1) &= \frac{1}{3}(3 + 2\frac{1}{15} + 1.6) = \frac{20}{9} \\ bias(\hat{\beta}_1) &= E(\beta_1) - \hat{\beta}_1 = \frac{20}{9} - 2 = \frac{2}{9} \end{aligned}$$

3 Pr3

Part(a) Our observed data would be a set of masses of top quarks. In a Bayesian view, we believe that the mass of a top quark is 173.3GeV, and we have 95% belief that the true mass is some value between 186.2 to 179.8 GeV. The prior candidate would be the distribution of the observed masses of top quarks.

Part(b) The application of the fertilizer is believed to be associated with increase in height of trees. The estimated increase of tree height is 20.0 cm, and we have 95% belief that the true increase is some value between 3.2 and 35.6 cm. The prior would be all the record of tree height increase in the past about the experiment of the fertilizer applied on the trees.

Part(c) We have 60% belief that it will rain in Seattle tomorrow.

Given a few days both at present and in the past, we predict the probability of rain tomorrow. This can be similar to a coin-tossing problem. If we assume the probability is a constant, and if we know the weather in the past few days, then we can use the Bayes' theorem to predict the weather tomorrow. So the prior candidate would be the weather information in the past week.

Appendix

```
# problem 1
one.sample =  function ( n.sample = 100, beta0 = 1, beta1 = 0.5, alpha  = 0.05, k = 1, out.choice = 2){
  x = seq(-1, 1, length = n.sample)
    mu = exp(beta0 + beta1*x)
  y = rgamma(n.sample, shape = k, scale = mu)

  fit.ML = glm(y ~ x, family = poisson)
    bhat.ML = coef(fit.ML)
    se.ML = summary(fit.ML)$coef[ ,2] #std error
    ci.ML = bhat.ML[2] + c(-1, 1)*qnorm(1-alpha/2)*se.ML[2]
    cov.ML = ifelse(beta1 > ci.ML[1] & beta1 < ci.ML[2], 1, 0)
  fit.QL = glm(y ~ x, family = quasipoisson)
    bhat.QL = coef(fit.QL)
    se.QL = summary(fit.QL)$coef[ ,2]
    ci.QL = bhat.QL[2] + c(-1, 1)*qnorm(1-alpha/2)*se.QL[2]
    cov.QL = ifelse(beta1 > ci.QL[1] & beta1 < ci.QL[2], 1, 0)

  ## The folloing Newton method is modified from Pr4 of HW4 Hey in 570.
  Xmat <- cbind(rep(1, length(x)), x)
  ## make vector of y & ni
  N <- n.sample
  ## Newton-Raphson
  beta.k <- coef(fit.ML) # intitial value
  repeat({
    mu.hat <- as.vector(exp(Xmat %*% beta.k))
    var.hat <-  mu.hat
    sumG <- crossprod(Xmat, y-mu.hat)
    Amat <- crossprod(Xmat, diag(var.hat)) %*%Xmat
    beta.kplus1 <- beta.k + solve(Amat)%*%sumG
    if (max(abs(beta.kplus1-beta.k))<1E-6) break()
    beta.k <- beta.kplus1
  })
  bhat.sandwich = beta.k
  ## End of Newton

  se.sandwich = sqrt(diag(vcovHC(fit.ML, type = "HC0")))
  #     print (se.sandwich)
  # #     se.1.sand =sqrt(vcovHC(fit.ML, type = "HC0")[2,2]) #A not Aobs
  #     print (se.1.sand)
  ci.sandwich = bhat.ML[2] + c(-1, 1)*qnorm(1-alpha/2)*se.sandwich[2]
  cov.sandwich = ifelse(beta1 > ci.sandwich[1] & beta1 < ci.sandwich[2], 1, 0)

  output = matrix(c(
             beta0.ML = bhat.ML[1],
             beta1.ML = bhat.ML[2],
             beta0.QL = bhat.QL[1],
             beta1.QL = bhat.QL[2],
             beta0.sand = bhat.sandwich[1],
             beta1.sand = bhat.sandwich[2],
             cov.ML,
             cov.QL,
             cov.sandwich
```

```
              ), nrow = 1)
  name = c("b0hat.ML", "b1hat.ML",
           "b0hat.QL", "b1hat.QL",
           "b0hat.sand", "b1hat.sand",
           "b1hat.ML.cov",
           "b1hat.QL.cov",
           "b1hat.sand.cov")

  output.chunk = matrix(output[1:6],nrow=2, ncol=3)
  output.useful = output[-(3:6)]
  colnames(output) = name
  colnames(output.chunk) = c("ML","Quasi-L","Semi-param")
  rownames(output.chunk) = c("b0hat","b1hat")
#   colnames(output.useful) = name[-(3:6)]
  out.list = list(compare.means = output.chunk, outputs = output.useful, output.all = output)
  return(out.list[[out.choice]])
  }

## Check the estimates (beta hats) from the three settings are the same.
compare.means = one.sample(n.sample = 100, beta0 = 1, beta1 = 0.5, out.choice = 1)
compare.means

## simulation function
n.sims= 20
n.sample = 200
beta0 = 1
beta1 =1
alpha  = 0.05
k = 1
out.choice = 2
sims <- function (n.sims = 50, n.sample = 200, beta0 = 1, beta1 = 0.5, alpha  = 0.05, k = 1 , out.choice
  out.tmr =  t(
    replicate(
      n.sims,
      one.sample(
        n.sample = n.sample, beta0 = beta0, beta1 = beta1, alpha = alpha, k = k, out.choice = out.choice
  output = apply (out.tmr, 2, mean)
  return(output)
}
n.sims = 500
beta0 = 1
beta1 = 0.5
# sims (n.sims,  beta0 = beta0,beta1 = beta1)

sample.size.lst = c(20,50,100,300, 2000)
# sample.size.lst = c(40, 50)
# simulation.out = sapply(sample.size.lst, sims, n.sample = 200, beta0 = beta0, beta1 = beta1)
# simulation.out

simulation.number.lst = c(20, 200, 1000)

beta0.lst = c( 0, 0.5, 1, 2)
beta1.lst = c( 0, 0.5, 1, 2.5, 5)
```

```
nrows = length(simulation.number.lst)*length(sample.size.lst)*length(beta0.lst)*length(beta1.lst)
# as.data.frame(matrix(NA, nrow = nrows, ncol = 9), rownames = c(""))
pr1.data = data.frame(expand.grid(simulation.number = simulation.number.lst,
                        sample.size = sample.size.lst,
                        beta0 = beta0.lst,
                        beta1 = beta1.lst),
           b0hat=rep(NA,nrows) ,
           b1hat=rep(NA,nrows)  ,
           cov.ML=rep(NA,nrows) ,
           cov.sand=rep(NA,nrows) ,
           cov.QL=rep(NA, nrows) )
head( pr1.data )


copy.i.row = 0
## this may process for a while
for (beta1 in beta1.lst){
  for (beta0 in beta0.lst){
    for (n.sample in sample.size.lst){
      for ( n.sims in simulation.number.lst){
        copy.i.row = copy.i.row + 1
        pr1.data[copy.i.row,5:9] = sims (n.sims = n.sims,
                                         n.sample = n.sample,
                                         beta0 = beta0,
                                         beta1 = beta1,
                                         out.choice = 2)
        ## Show progress of the iterations
        perc = round(100*sum( pr1.data$simulation.number[1:copy.i.row])/sum(pr1.data$simulation.number,
        progr = paste("rows, ", perc), "%",sep="")
        print(
          paste("Processing ",
                copy.i.row,"/",
                nrows,
                progr,
                sep=""))
      }
    }
  }
}

head(pr1.data)

## Save the data.
write.table( pr1.data,  file = paste("571hw1pr1",".csv",sep=""),
             sep=",",
             row.names = FALSE)

data = pr1
q1c.plot = function (b0, b1.plot, data) {
  n.plot = length(b1.plot)
  fl.tmr = floor(sqrt(n.plot))
  ce.tmr = ceiling(n.plot/fl.tmr)
  par(mfrow=c(fl.tmr,ce.tmr))
```

```
  for (i in 1: n.plot){
    data.qc = subset(data, simulation.number == 1000 & beta0==b0 & beta1 == b1.plot[i])
    qc.tmr = which(data.qc$beta1==b1.plot[i])
    plot(data.qc$sample.size[qc.tmr],  abs(b1.plot[i]-data.qc$b1hat[qc.tmr]),pch=19, type="b",
         ylim = c(0,0.05),
         main = bquote("Bias of" ~ beta[1] ~ ". Given " ~ beta[0]== ~ .(b0) ~"," ~ beta[1]== ~ .(b1.plot
         ylab = expression(paste("Absolute value of bias of ", beta[1], sep="")),
         xlab = "Sample size")
    abline(h=0, lty=2, col = "grey")
  }

  for (i in 1: n.plot){
    data.qc = subset(data, simulation.number == 1000 & sample.size == 1000 & beta1 == b1.plot[i])
    qc.tmr = which(data.qc$beta1==b1.plot[i])
    plot(1,type="n",
         xlim = c(min(data$beta0),max(data$beta0)),
         ylim = c(0,1),
         main = bquote("Coverage trend" ~ " Given " ~ beta[1]== ~ .(b1.plot[i])),
         ylab = "Coverage",
         xlab = bquote(beta[0]))
    lines(data.qc$beta0[qc.tmr], data.qc$cov.ML[qc.tmr], type = "b", lty = 1, pch = 1, col = "black")
    lines(data.qc$beta0[qc.tmr], data.qc$cov.QL[qc.tmr], type = "b", lty = 2,  pch = 5, col = "blue")
    lines(data.qc$beta0[qc.tmr], data.qc$cov.sand[qc.tmr], type = "b",lty = 5, pch = 19,  col = "red")
    abline(h=0.95, lty=4, col = "grey")
    axis(side=2, at=c(0.95), las=2, cex.axis=0.9)

  legend( x="bottomleft",
          legend=c("Max. Likelihood", "Quasi Likelihood", "Semi-Parametric"),
          col=c("black","blue","red"), lwd=1, lty=c(1,2,5),pch=c(1,5,19) ,cex=1.2, bty="n")

  }
  par(mfrow=c(1,1))
}
b1.plot = c(0,0.5,1)
b0 = 0
q1c.plot(b0, b1.plot, data)
```

```
# problem2
par(mfrow = c(1, 1))
x.lst = c(0, 1, 3)
size = 20
x = sample(x.lst, size, replace = T, prob = NULL)
mu = rep(0, size)
mu[which(x == 0)] = 0.6
mu[which(x == 1)] = 3.6
mu[which(x == 3)] = 6.8
Y = rnorm(size, mu, 1)

fit.ls = lm(Y ~ x)
plot(0, type = "n", xlim = c(0, 3), ylim = c(-3, 10), main = "Pr2 Q(a)", xlab = "x",
    ylab = "Y")
A <- matrix(c(x, Y), ncol = 2)
B <- matrix(c(x, fitted(fit.ls)), ncol = 2)
```

```r
# lines( A, col='red' )
points(A, col = "blue", pch = 19, cex = 0.8)
lines(B, col = "black", lty = 2)
points(B, col = "purple", pch = 1, cex = 2)

legend(x = "topleft", legend = c("Y", "Fitted line with Y.hat"), col = c("blue",
    "black"), lwd = 1, lty = c(NA, 2), pch = c(19, 1))
# b
coef(lm(c(0.6, 3.6, 6.8) ~ c(0, 1, 3)))

# e
sample.list = c(seq(2, 30, 2), seq(35, 100, 5))
# sample.list

do.one = function(size) {
    x.lst = c(0, 1, 3)
    x = sample(x.lst, size, replace = T, prob = NULL)
    mu = rep(0, size)
    ux = unique(x)
    x
    ux

    if (length(ux) == 1) {
        while (length(ux) == 1) {
            x = sample(x.lst, size, replace = T, prob = NULL)
            ux = unique(x)
        }
    }
    mu[which(x == 0)] = 0.6
    mu[which(x == 1)] = 3.6
    mu[which(x == 3)] = 6.8

    Y = rnorm(size, mu, 1)
    fit.ls = lm(Y ~ x)
    b1hat = coef(fit.ls)[2]
    return(b1hat)
}

n.sim = 1500
beta1 = 2
n.sample.list = length(sample.list)
result = matrix(ncol = 2, nrow = n.sample.list)
for (j in 1:n.sample.list) {
    size = sample.list[j]
    tmp = matrix(ncol = 1, nrow = n.sim)  # b1hat
    for (i in 1:n.sim) {
        tmp[i] = do.one(size)
    }
    result[j, ] = c(beta1 - mean(tmp), sd(tmp))
}
colnames(result) = c("bias", "SD")
head(result)
plot(sample.list, seq(min(-result[, 1]), max(result[, 2]), length.out = n.sample.list),
```

```
    type = "n", main = "Q2e,f,g", xlab = "Sample size", ylab = NA)
points(sample.list, y = -result[, 1], pch = 19, col = "black")
points(sample.list, , y = result[, 2], pch = 15, col = "green")
abline(0, 0, lty = 4, col = "red")
points(seq(2, 8, by = 2), y = c(0.22222, 0.119358, 0.05765, 0.02709), pch = 9,
    col = "blue", cex = 1.5)
axis(side = 1, at = c(2, 4, 6, 8), las = 1, cex.axis = 0.5)
legend("topright", c("Bias", "SD", "Exact bias"), col = c("black", "green",
    "blue"), pch = c(19, 15, 9), box.lwd = 0, bty = "n")
# g
coef(lm(c(0.6, 3.6) ~ c(0, 1)))
coef(lm(c(0.6, 6.8) ~ c(0, 3)))
coef(lm(c(3.6, 6.8) ~ c(1, 3)))
```