Stats 201B: Statistical Modeling and Learning
Problem Set 3

February 5, 2015

Due 16 February

Question 1

(25pt + 10pt) In this question, we use the file `brader.csv` which contains data from Brader, Valentino and Suhay (2008). The file includes the following variables for $n = 265$ observations:

- `immigr`: the outcome of interest – a four-point scale in response to "Do you think the number of immigrants from foreign countries should be increased or decreased?"
- `tone`: tone of the story treatment (positive or negative)
- `eth`: ethnicity of the featured immigrant treatment (Mexican or Russian)
- `ppage`: respondents' age
- `ppincimp`: respondents' income

Consider the following ordered logit model for an ordered outcome variable with four levels:

$$\Pr(Y_i \leq j \mid X_i) \;=\; \frac{\exp(\psi_j - X_i^\top \beta)}{1 + \exp(\psi_j - X_i^\top \beta)}$$

for $j = 1, 2, 3, 4$ and $i = 1, ..., n$ where $\psi_4 = \infty$ and $X_i = [\texttt{tone}_i \ \texttt{eth}_i \ \texttt{ppage}_i \ \texttt{ppincimp}_i]^\top$ (i.e. no intercept).

a. (5pt) Write down the likelihood function.

$$\begin{aligned} l_N(\beta, \tau) &= \log\left(\prod \left\{ logit(\psi_{Y_i} - X_i^t\beta) - logit(\psi_{Y_i - 1} - X_i^t\beta) \right\}\right) \qquad \text{since } Y_i = 1, 2, 3, 4 \\ &= \sum_i^N \left(\log\left\{ \frac{e^{\psi_{Y_i} - X_i^t\beta}}{1 + e^{\psi_{Y_i} - X_i^t\beta}} - \frac{e^{\psi_{Y_i - 1} - X_i^t\beta}}{1 + e^{\psi_{Y_i - 1} - X_i^t\beta}} \right\} \right) \end{aligned}$$

b. (10pt) Derive the score functions for β and ψ_j.

$$\frac{\partial l_N}{\partial \beta} = \sum_i^N \left(\frac{1}{logit(\psi_{Y_i} - X_i^t\beta) - logit(\psi_{Y_i-1} - X_i^t\beta)} [\frac{-e^{\psi_{Y_i} - X_i^t\beta} X_i}{(1+e^{\psi_{Y_i} - X_i^t\beta})^2} - \frac{-e^{\psi_{Y_i-1} - X_i^t\beta} X_i}{(1+e^{\psi_{Y_i-1} - X_i^t\beta})^2}] \right)$$

$$= \sum_i^N \left(\frac{-1}{logit(\psi_{Y_i} - X_i^t\beta) - logit(\psi_{Y_i-1} - X_i^t\beta)} [\frac{e^{\psi_{Y_i} - X_i^t\beta}}{(1+e^{\psi_{Y_i} - X_i^t\beta})^2} - \frac{e^{\psi_{Y_i-1} - X_i^t\beta}}{(1+e^{\psi_{Y_i-1} - X_i^t\beta})^2}] X_i \right)$$

$$\frac{\partial l_N}{\partial \psi_j} = \sum_i^N \left(\frac{1}{logit(\psi_{Y_i} - X_i^t\beta) - logit(\psi_{Y_i-1} - X_i^t\beta)} \right.$$
$$\left. [\frac{e^{\psi_{Y_i} - X_i^t\beta} 1\{Y_i = \psi_j\}}{(1+e^{\psi_{Y_i} - X_i^t\beta})^2} - \frac{e^{\psi_{Y_i-1} - X_i^t\beta} 1\{Y_i - 1 = \psi_j\}}{(1+e^{\psi_{Y_i-1} - X_i^t\beta})^2}] \right)$$

Since $1\{Y_i = \infty\} \equiv 0$, we may safely use the above notation for $j = 0, 1, \ldots, 4$.

c. (10pt) Using (a) and (b), calculate the maximum likelihood estimates of β and ψ_j and their standard errors via the `optim` function in R. Confirm your results by comparing them to outputs from the `polr` function in the `MASS` package.

The results of self-designed program:

```
brader = read.csv("E:\\brader.csv")
#logit link function
logit<-function(X) #X is a matrix
{
   return(1 - 1 / (1 + exp(X)))
}
#log-likelihood function
l.ordlogit.4<-function(param, X, Y)
{
  X = as.matrix(X)
  parNum = ncol(X)
  Y = matrix(Y, ncol = 1)
  param = matrix(param, ncol = 1)
  beta = param[1:parNum,,drop = FALSE]
  psi = param[(parNum+1):nrow(param),,drop = FALSE]
  p4 = log(1-logit(psi[3]-X[Y[,1]==4,]%*%beta))
  p3 = log(logit(psi[3]-X[Y[,1]==3,]%*%beta) - logit(psi[2]-X[Y[,1]==3,]%*%beta))
  p2 = log(logit(psi[2]-X[Y[,1]==2,]%*%beta) - logit(psi[1]-X[Y[,1]==2,]%*%beta))
  p1 = log(logit(psi[1]-X[Y[,1]==1,]%*%beta))
  penalty = 10^6*as.numeric(any(psi[2]>psi[3], psi[1]>psi[2]))
```

```
  l = sum(p1) + sum(p2)+ sum(p3)+sum(p4) - penalty
  return(l)
}
#process data
X = as.matrix(brader[,-1])
Y = as.matrix(brader[,1])
param = c(0.74, 0.17 , 0.010, 0.004,-1.6, 0.131, 1.3)
#calculate optimum value
orlogit = optim(par=param, fn=l.ordlogit.4, X=X, Y=Y, method="BFGS",
                control=list(fnscale=-1), hessian=TRUE)
orlogit$par
diag(-solve(orlogit$hessian))
> orlogit$par
[1]  0.749482509  0.166256087  0.009355862  0.004154183 -1.666480256  0.132527341
[7]  1.354225470
> diag(-solve(orlogit$hessian))
[1] 5.301158e-02 5.155771e-02 5.086789e-05 8.709295e-04 3.208288e-01 2.917577e-01
[7] 2.988210e-01
```

$\beta = (0.749482509, 0.166256087, 0.009355862, 0.004154183)^t, \phi_1 = -1.666480256, \phi_2 = 0.132527341,$ $\phi_3 = 1.354225470$
The corresponding standard errors are $5.301158e{-}02, 5.155771e{-}02, 5.086789e{-}05, 8.709295e{-}04, 3.208288e-01, 2.917577e-01, 2.988210e-01$.
The results of polr:

```
brader.df = data.frame(immigr=as.factor(brader[,1]), tone=brader[,2],
                       eth=brader[,3], ppage=brader[,4], ppincimp=brader[,5])
polr(immigr~tone+eth+ppage+ppincimp,data = brader.df,
     method = "logistic", Hess = TRUE)
Call:
polr(formula = immigr ~ tone + eth + ppage + ppincimp, data = brader.df,
    Hess = TRUE, method = "logistic")

Coefficients:
       tone         eth       ppage    ppincimp
0.749445772 0.166225150 0.009344805 0.004149295

Intercepts:
       1|2        2|3        3|4
-1.6671092  0.1318298  1.3535054
```

```
Residual Deviance: 651.8892
AIC: 665.8892
```

Both results are the same. But self-designed program is not numerically stable so that a proper initial parameter must be found.

d. (10pt) **Bonus question.** The standard ordered logit model is sometimes called the *proportional odds model* because it assumes the effect of X_i is constant across levels on the odds ratio scale. One approach to relax this assumption is to allow the coefficients to vary across levels, i.e.,

$$\Pr(Y_i \leq j \mid X_i) \;=\; \frac{\exp(\psi_j - X_i^\top \beta_j)}{1+\exp(\psi_j - X_i^\top \beta_j)}$$

for $j = 1, 2, 3, 4$ and $i = 1, ..., n$ where $\beta_4 = 0$ (i.e. the fourth group is a reference group), and $\psi_4 = \infty$. For this model, derive the likelihood and score functions, and use `optim` to obtain the maximum likelihood estimates of β_j and ψ_j and their standard errors for the `brader` data.

We now extend the model in the following forms:

$$\begin{aligned}
&P(Y_i \leq 1 \mid X_i) = logit(\psi_1 - X_i^t\beta_1) \qquad \beta_1 = (\beta_{11}, \beta_{21}, \beta_{31}, \beta_{41})^t \\
&P(Y_i \leq 2 \mid X_i) = logit(\psi_2 - X_i^t\beta_2) \qquad \beta_2 = (\beta_{12}, \beta_{22}, \beta_{32}, \beta_{42})^t \\
&P(Y_i \leq 3 \mid X_i) = logit(\psi_3 - X_i^t\beta_3) \qquad \beta_3 = (\beta_{13}, \beta_{23}, \beta_{33}, \beta_{43})^t \\
&P(Y_i \leq 4 \mid X_i) = 1 \\
&\Rightarrow \\
&P(Y_i = 1 \mid X_i) = logit(\psi_1 - X_i^t\beta_1) \\
&P(Y_i = 2 \mid X_i) = logit(\psi_2 - X_i^t\beta_2) - logit(\psi_1 - X_i^t\beta_1) \\
&P(Y_i = 3 \mid X_i) = logit(\psi_3 - X_i^t\beta_3) - logit(\psi_2 - X_i^t\beta_2) \\
&P(Y_i = 4 \mid X_i) = 1 - logit(\psi_3 - X_i^t\beta_3)
\end{aligned}$$

The parameter vector is $par = (\beta_1^t, \beta_2^t, \beta_3^t, \psi_1, \psi_2, \psi_3)^t$.

$$\begin{aligned}
l(par \mid X, Y) = &\sum_{Y_i=1} \log\left[logit(\psi_1 - X_i^t\beta_1)\right] + \sum_{Y_i=2} \log\left[logit(\psi_2 - X_i^t\beta_2) - logit(\psi_1 - X_i^t\beta_1)\right] + \\
&\sum_{Y_i=3} \log\left[logit(\psi_3 - X_i^t\beta_3) - logit(\psi_2 - X_i^t\beta_2)\right] + \sum_{Y_i=4} \log\left[1 - logit(\psi_3 - X_i^t\beta_3)\right] \\
= &\sum_i^N \left(\log\left\{ \frac{e^{\psi_{Y_i} - X_i^t\beta_{Y_i}}}{1+e^{\psi_{Y_i} - X_i^t\beta_{Y_i}}} - \frac{e^{\psi_{Y_i-1} - X_i^t\beta_{Y_i-1}}}{1+e^{\psi_{Y_i-1} - X_i^t\beta_{Y_i-1}}} \right\} \right) \qquad \beta_0 = \beta_4 = 0 \quad \psi_0 = -\infty \quad \psi_4 = \infty
\end{aligned}$$

$$\frac{\partial l_N}{\partial \beta_j} = \sum_i^N \Bigg(\frac{-1}{logit(\psi_{Y_i} - X_i^t\beta_{Y_i}) - logit(\psi_{Y_i-1} - X_i^t\beta_{Y_i-1})} \\ [\frac{e^{\psi_{Y_i} - X_i^t\beta_{Y_i}} 1\{Y_i = \psi_j\}}{(1+e^{\psi_{Y_i} - X_i^t\beta_{Y_i}})^2} - \frac{e^{\psi_{Y_i-1} - X_i^t\beta_{Y_i-1}} 1\{Y_i - 1 = \psi_j\}}{(1+e^{\psi_{Y_i-1} - X_i^t\beta_{Y_i-1}})^2}] X_i \Bigg)$$

$$\frac{\partial l_N}{\partial \psi_j} = \sum_i^N \Bigg(\frac{1}{logit(\psi_{Y_i} - X_i^t\beta_{Y_i}) - logit(\psi_{Y_i-1} - X_i^t\beta_{Y_i-1})} \\ [\frac{e^{\psi_{Y_i} - X_i^t\beta_{Y_i}} 1\{Y_i = \psi_j\}}{(1+e^{\psi_{Y_i} - X_i^t\beta_{Y_i}})^2} - \frac{e^{\psi_{Y_i-1} - X_i^t\beta_{Y_i-1}} 1\{Y_i - 1 = \psi_j\}}{(1+e^{\psi_{Y_i-1} - X_i^t\beta_{Y_i-1}})^2}] \Bigg)$$

```
l.ordlogit.4.beta<-function(beta, X, Y, psi)
{
  beta = matrix(beta, ncol=1)
  X = as.matrix(X)
  Y = matrix(Y, ncol = 1)
  beta.1 = beta[1:4,,drop = FALSE]
  beta.2 = beta[5:8,,drop = FALSE]
  beta.3 = beta[9:12,,drop = FALSE]
  p4 = log(1-logit(psi[3]-X[Y[,1]==4,]%*%beta.3))
  p3 = log(logit(psi[3]-X[Y[,1]==3,]%*%beta.3)
           - logit(psi[2]-X[Y[,1]==3,]%*%beta.2))
  p2 = log(logit(psi[2]-X[Y[,1]==2,]%*%beta.2)
           - logit(psi[1]-X[Y[,1]==2,]%*%beta.1))
  p1 = log(logit(psi[1]-X[Y[,1]==1,]%*%beta.1))
  penalty = 10^6*as.numeric(any(psi[2]>psi[3], psi[1]>psi[2]))

  l = sum(p1) + sum(p2)+ sum(p3)+sum(p4) - penalty
  return(l)
}
l.ordlogit.4.psi<-function(psi, X, Y, beta)
{
  X = as.matrix(X)
  Y = matrix(Y, ncol = 1)
  beta.1 = beta[1:4,,drop = FALSE]
  beta.2 = beta[5:8,,drop = FALSE]
  beta.3 = beta[9:12,,drop = FALSE]
  p4 = log(1-logit(psi[3]-X[Y[,1]==4,]%*%beta.3))
  p3 = log(logit(psi[3]-X[Y[,1]==3,]%*%beta.3)
           - logit(psi[2]-X[Y[,1]==3,]%*%beta.2))
```

```
  p2 = log(logit(psi[2]-X[Y[,1]==2,]%*%beta.2)
           - logit(psi[1]-X[Y[,1]==2,]%*%beta.1))
  p1 = log(logit(psi[1]-X[Y[,1]==1,]%*%beta.1))
  penalty = 10^6*as.numeric(any(psi[2]>psi[3], psi[1]>psi[2]))

  l = sum(p1) + sum(p2)+ sum(p3)+sum(p4) - penalty
  return(l)
}

beta = c(0.892194514, -0.092005636,  0.011226191,  0.004547339,  0.815988147,
        0.112129601, 0.006075800,  0.020328998,  0.643340882,  0.455857262,
        0.009901077, -0.006459265)
psi = c(-1.6657782,  0.1511321,  1.3543657)
for(i in 1:1000)
{

  beta.optim = optim(par=beta, fn=l.ordlogit.4.beta, X=X, Y=Y, psi=psi,
                     method="BFGS", control=list(fnscale=-1), hessian = T)
  beta = beta.optim$par
  psi.optim = optim(par=psi, fn=l.ordlogit.4.beta, X=X, Y=Y, beta=beta,
                    method="BFGS", control=list(fnscale=-1), hessian = T)
  psi = psi.optim$par
}
beta.optim$par
diag(-solve(beta.optim$hessian))
psi.optim$par
diag(-solve(psi.optim$hessian))
> beta.optim$par
 [1]  0.933529158 -0.281408825  0.017481019  0.021859710  0.900995074  0.033472854
 [7]  0.009130671  0.029802613  0.623272607  0.310286816  0.008868095 -0.015931215
> diag(-solve(beta.optim$hessian))
 [1] 2.414161e-01 2.559858e-01 1.148406e-04 2.247434e-03 6.788319e-02 6.814716e-02
 [7] 3.445443e-05 6.682537e-04 5.954144e-02 6.049998e-02 3.210825e-05 6.275145e-04
> psi.optim$par
[1] -1.2837446  0.3891901  1.1139653
> diag(-solve(psi.optim$hessian))
[1] 0.05695386 0.01806734 0.01605114
```

If I calculate the parameters β and ψ in the same optim() function, there are warnings. So, I get optimum β and then get optimum ψ. In this way, I repeat the procedures for 1000 times.

The estimated values and errors are showed above.

Question 2

(30 pt.) Understanding the basics of maximum likelihood makes it easier to learn a wide range of models. Here you will learn about a model not covered in class. The *tobit model* is often used to model a censored outcome variable, where we only observe values of the outcome above a certain threshold τ while values below that threshold take on a single value, say 0. A classic example is labor market participation: you only observe a worker's wage if their potential earnings is high enough to be over minimium wage or otherwise employable.

Consider the case where $\tau = 0$. In this case, the model can be written as follows.

$$Y_i = \begin{cases} Y_i^* & \text{if} \quad Y_i^* > 0 \\ 0 & \text{if} \quad Y_i^* \leq 0 \end{cases} \quad \text{where} \quad Y_i^* = X_i^\top \beta + \epsilon_i,$$

where $\epsilon_i \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sigma^2)$. Note that Y_i is the observed outcome variable and Y_i^* is the underlying latent variable, which is not (always) observable.

a. (10pt) Derive the likelihood and log-likelihood functions of the model for a simple random sample of size N.

$$Y_i = Y_i^* 1\{Y_i^* > 0\} \Rightarrow p(Y_i|X_i, \beta, \sigma) = \phi(\frac{X_i^t\beta}{\sigma})^{1\{Y_i>0\}}[1 - \Phi(\frac{X_i^t\beta}{\sigma})]^{1\{Y_i=0\}}$$

where ϕ, Φ are density function and distribution function of canonical Gaussian variable.

$$L(\beta, \sigma|X_i, Y_i) = \prod_i^N \left\{ \phi(\frac{X_i^t\beta}{\sigma})^{1\{Y_i>0\}}[1 - \Phi(\frac{X_i^t\beta}{\sigma})]^{1\{Y_i=0\}} \right\}$$

$$l(\beta, \sigma|X_i, Y_i) = \sum_i^N \left\{ \log[\phi(\frac{X_i^t\beta}{\sigma})]1\{Y_i > 0\} + \log[1 - \Phi(\frac{X_i^t\beta}{\sigma})]1\{Y_i = 0\} \right\}$$

b. (10pt) A natural quantity of interest for this model is the expected value of the outcome variable conditional on a set of predictor levels, i.e., $\mathbb{E}(Y_i \mid X_i = x)$. Prove that the exact expression for this quantity is the following.

$$\mathbb{E}(Y_i \mid X_i = x) = \Phi\left(\frac{x^\top\beta}{\sigma}\right) x^\top\beta + \sigma\phi\left(\frac{x^\top\beta}{\sigma}\right),$$

where $\phi(\cdot)$ and $\Phi(\cdot)$ denote the PDF and CDF of a standard normal random variable, respectively.

Hint: You may find the following property of the standard normal PDF helpful: $\partial\phi(z)/\partial z = -z \cdot \phi(z)$.

$$
\begin{aligned}
\mathbb{E}[Y_i|X_i] &= \int_0^\infty y \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-(y_i - X_i^t\beta)}{2\sigma^2}} dy \\
&= \int_0^\infty (y - X_i^t\beta) \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-(y_i - X_i^t\beta)}{2\sigma^2}} + X_i^t\beta \int_0^\infty \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-(y_i - X_i^t\beta)}{2\sigma^2}} dy \\
&= \int_{-\frac{X_i^t\beta}{\sigma}}^\infty \frac{z\sigma}{\sqrt{2\pi}} e^{\frac{z^2}{2\sigma^2}} dz + X_i^t\beta\Phi(\frac{X_i^t\beta}{\sigma}) \\
&= \frac{\sigma}{\sqrt{2\pi}} - e^{\frac{-z^2}{2}}|_{\frac{\mathbf{X}_i^t\beta}{\sigma}}^\infty + X_i^t\beta\Phi(\frac{X_i^t\beta}{\sigma}) \\
&= X_i^t\beta\Phi(\frac{X_i^t\beta}{\sigma}) + \sigma\phi(\frac{X_i^t\beta}{\sigma})
\end{aligned}
$$

c. (10pt) Nielsen (2013, *ISQ*) investigates the impact of human rights violations on the flows of foreign aid from donor states to recipient states. The file `nielsenaid.csv` contains a small subset of the original dataset and consists of the following variables measured for each donor-recipient dyad:

- `lneconaid` — log of economic aid commitments in U.S. dollars in 2000
- `l.physint` — physical integrity violations index (e.g. toture) in 1999
- `l.polity2` — Polity IV democracy score in 1999
- `l.lneconaid` — one-year lagged value of `lneconaid`
- `l.lnrgdp` — log GDP per capita in 1999 candidate
- `l.lnpop` – log population in 1999
- `colony` — indicator of former colony
- `recep` — recipient country name degree
- `donor` — donor country name

Fit the above tobit model, where

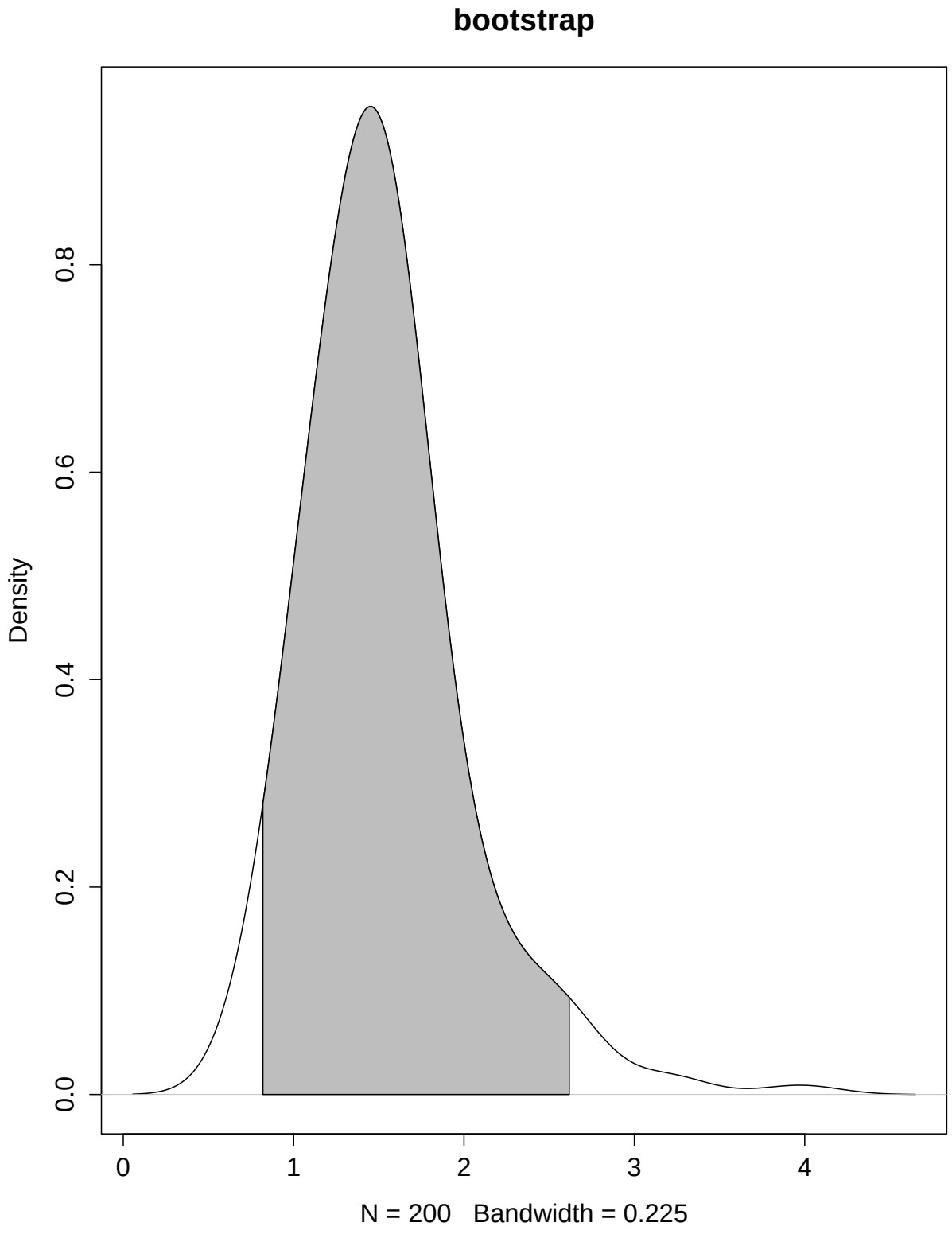
$$
\begin{aligned}
Y_i &= \texttt{lneconaid}, \\
X_i &= [\texttt{l.physint}, \texttt{l.polity2}, \texttt{l.lneconaid}, \texttt{l.lnrgdp}, \texttt{l.lnpop}, \texttt{colony}]^\top,
\end{aligned}
$$

and estimate $\mathbb{E}[Y_i \mid X_i = x]$ and its asymptotic 95% confidence interval, where x equals the medians for all predictors except `colony`, and the mode for `colony`. (For this question, you may ignore possible error correlation across observations, such as clustering. You may use a canned procedure to obtain the MLE of β and σ as well as their sampling variance matrix; but for **extra credit**, calculate those via `optim` on your own.)

```
nielsenaid = read.csv("E:\\nielsenaid.csv")
l.tobit<-function(param, X, Y)
{
  X = as.matrix(X)
```

```
  Y = as.matrix(Y)
  param = as.matrix(param)
  beta = param[-nrow(param),,drop=FALSE]
  std = param[nrow(param),,drop=TRUE]
  l = sum(log(dnorm(X[Y[,1]>0, ]%*%beta/std))) +
    sum(log(1-pnorm(X[Y[,1]==0,]%*%beta/std)))
  return(l)
}
X = as.matrix(nielsenaid[,3:8])
Y = as.matrix(nielsenaid[,2])
param = c(rnorm(6), 10)
op =  optim(par=param, fn=l.tobit, X=X, Y=Y, method="Nelder-Mead",
      control=list(fnscale=-1), hessian=TRUE)


itera = 200
mean.pd = rep(NaN, itera)
X.pd = matrix(c(colMeans(X)[-6], 0), nrow=1)
i=0
N = nrow(X)
while(i <= itera)
{
  index = sample(N, N, replace = T)
  X.resample = X[index,]
  Y.resample = Y[index,]
  param = c(rnorm(6), 10)
  par.re = optim(par=param, fn=l.tobit, X=X, Y=Y, method="BFGS",
                 control=list(fnscale=-1))$par
  beta.re = as.matrix(par.re[1:6])
  std.re = par.re[7]
  mean.pd[i] = pnorm(X.pd%*%beta.re/std.re) * (X.pd%*%beta.re) +
    std.re*dnorm(X.pd%*%beta.re/std.re)
  i=i+1
}
mean(mean.pd)
plot(density(mean.pd, bw = 0.225), main = "bootstrap")
> mean(mean.pd)
[1] 1.534984
```



The mean is 1.534984.

Question 3

(20 pt.) Consider a supervised learning problem in which you sample Y_i and X_i from $p(X, Y)$. You assume an additive noise model such that: $Y_i = f(X_i) + \epsilon_i$, and $\mathbb{E}[\epsilon_i|X_i] = 0$. We wish to come up with $\hat{f}(X)$ that is the best approximation to $f(X)$ in some sense.

a. (8pt.) First choose your loss function, $L(Y_i, \hat{f}(X_i))$ to be squared loss. Show that the minimizer of this loss is found by choosing $\hat{f}(X_i) = \mathbb{E}[Y_i|X_i]$. (It suffices to show this pointwise, i.e. conditional on a particular choice of X).

$$\mathbb{E}([Y_i - \widehat{f}(X_i)]^2) = \mathbb{E}([Y - \mathbb{E}(Y_i|X_i) + \mathbb{E}(Y_i|X_i) - \widehat{f}(X_i)]^2)$$
$$= \mathbb{E}([Y_i - \mathbb{E}(Y_i|X_i)]^2) + \mathbb{E}([\mathbb{E}(Y_i|X_i) - \widehat{f}(X_i)]^2) + \mathbb{E}([Y_i - \mathbb{E}(Y_i|X_i)][\mathbb{E}(Y_i|X_i) - \widehat{f}(X_i)])$$
$$= \mathbb{E}([Y_i - \mathbb{E}(Y_i|X_i)]^2) + \mathbb{E}([\mathbb{E}(Y_i|X_i) - \widehat{f}(X_i)]^2) + \mathbb{E}_{X_i}\left[\mathbb{E}\left([Y_i - \mathbb{E}(Y_i|X_i)][\mathbb{E}(Y_i|X_i) - \widehat{f}(X_i)]|X_i\right)\right]$$
$$= \mathbb{E}([Y_i - \mathbb{E}(Y_i|X_i)]^2) + \mathbb{E}([\mathbb{E}(Y_i|X_i) - \widehat{f}(X_i)]^2) \geq \mathbb{E}([Y_i - \mathbb{E}(Y_i|X_i)]^2)$$

We get equation, if and only if $\mathbb{E}([\mathbb{E}(Y_i|X_i) - \widehat{f}(X_i)]^2) = 0 \Rightarrow \mathbb{E}(Y_i|X_i) = \widehat{f}(X_i)$ a.s.

For $L(Y, \widehat{f}(X)) = \sum_i^N \mathbb{E}([Y_i - \widehat{f}(X_i)]^2)$, based on above result, $\widehat{f}(X_i) = \mathbb{E}(Y_i|X_i)$ for $i = 1, \ldots, n$

b. (10pt.) Show that the expected generalization error, $\mathcal{R} = \mathbb{E}[(Y_i - \hat{f}(X_i))^2]$, can be decomposed into an irreducible error, a bias term, and a variance term. Furthermore, describe in plain English the meaning of variance and bias in this context. (Again, it suffices to do this pointwise, conditionally on X).

$$\mathbb{E}([Y_i - \widehat{f}(X_i)]^2) = \mathbb{E}([Y_i - \mathbb{E}(\widehat{f}(X_i)) + \mathbb{E}(\widehat{f}(X_i)) - \widehat{f}(X_i)]^2)$$
$$= \mathbb{E}([Y_i - \mathbb{E}(\widehat{f}(X_i))]^2) + \mathbb{E}([\mathbb{E}(\widehat{f}(X_i)) - \widehat{f}(X_i)]^2) + \mathbb{E}([Y_i - \mathbb{E}(\widehat{f}(X_i))][\mathbb{E}(\widehat{f}(X_i)) - \widehat{f}(X_i)])$$
$$= \mathbb{E}([Y_i - \mathbb{E}(\widehat{f}(X_i))]^2) + \mathbb{E}([\mathbb{E}(\widehat{f}(X_i)) - \widehat{f}(X_i)]^2) + \mathbb{E}_{X_i}\left(\mathbb{E}\left([Y_i - \mathbb{E}(\widehat{f}(X_i))][\mathbb{E}(\widehat{f}(X_i)) - \widehat{f}(X_i)]|X_i\right)\right)$$
$$= \mathbb{E}([Y_i - \mathbb{E}(\widehat{f}(X_i))]^2) + \mathbb{E}([\mathbb{E}(\widehat{f}(X_i)) - \widehat{f}(X_i)]^2)$$
$$= \mathbb{E}([Y_i - \mathbb{E}(\widehat{f}(X_i))]^2) + \mathrm{Var}(\widehat{f}(X_i))$$

Based on (a), we can treat $\mathbb{E}(\widehat{f}(X_i))$ as some $\widehat{f}(X_i)$

$$= \mathbb{E}([Y_i - \mathbb{E}(Y_i|X_i)]^2) + \mathbb{E}([\mathbb{E}(Y_i|X_i) - \mathbb{E}(\widehat{f}(X_i))]^2) + \mathrm{Var}(\widehat{f}(X_i))$$

The first term is the irreducible error, the second term is the bias error, and the third term is the variance.

The bias is the average of the difference between the true value and expected mean of some estimator. The variance is the fluctuations of some estimator around its expectation.

c. (2pt.) In general how does a learning algorithm's flexibility relate to the bias and variance terms described above?

In general, we calculate the sum of squared bias and variance, which is the mean squared error. Then we try to minimize it. For some case, we need an estimator which have small fluctuation, then we focus more on variance error, but the bias error should not be so large.

Question 4

Cross Validation for Polynomial Regression. (18 points)

Consider the following four data generating processes:

- DGP 1: $Y = -2 * \mathbb{1}_{\{X<-3\}} + 2.55 * \mathbb{1}_{\{X>-2\}} - 2 * \mathbb{1}_{\{X>0\}} + 4 * \mathbb{1}_{\{X>2\}} - 1 * \mathbb{1}_{\{X>3\}} + \epsilon$
- DGP 2: $Y = 6 + 0.4X - 0.36X^2 + 0.005X^3 + \epsilon$
- DGP 3: $Y = 2.83 * \sin(\frac{\pi}{2} \times X) + \epsilon$
- DGP 4: $Y = 4 * \sin(3\pi \times X) * \mathbb{1}_{\{X>0\}} + \epsilon$

X is drawn from the uniform distribution in [-4,4] and ϵ is drawn from a standard normal ($\mu = 0$, $\sigma^2 = 1$).

1. *(5 pts.)* Write a function to estimate the generalization error of a polynomial by k-fold cross-validation. It should take as arguments the data, the degree of the polynomial, and the number of folds k. It should return the cross-validation mean squared error.

```
# X is a n*1 matrix.
poly.cv <-function(X, Y, degree, k)
{
  #construct the polynomial matirx  degree = nrow(beta)-1
  X.matrix = matrix(1, nrow = nrow(X), ncol = 1)
  if(degree > 0)
  {
    for(i in 1:degree)
      X.matrix = cbind(X.matrix, X^i)
  }
  n = nrow(X)
  #divide data into k folds
  select.num = n %/% k
  index = 1:n
  fold = list()
  for(i in 1:(k-1))
  {
```

```
    num.within = sample(index, select.num, replace = F)
    fold = c(fold, list(num.within))
    index = setdiff(index, num.within)
  }
  fold = c(fold, list(index))
  error = rep(NaN, k)
  #calculate the test error for each fold
  for(i in 1:k)
  {
    X.train = X.matrix[-fold[[i]], , drop = F]
    Y.train = Y[-fold[[i]], , drop = F]
    beta = solve(crossprod(X.train), crossprod(X.train, Y.train))
    X.test = X.matrix[fold[[i]], , drop = F]
    Y.test = Y[fold[[i]], , drop = F]
    res = Y.test - X.test %*% beta
    sse = t(res) %*% res
    error[i] = sse[1,1] / (nrow(X.test))
  }
  return(mean(error))
}
```

2. *(5 pts.)* Generate a dataset of size $n = 100$ from each of the four DGPs. For OLS fits of polynomials of orders $d = 0 \ldots 10$, calculate the in-sample MSE and the cross-validated MSE (with $k = 10$). For each dataset, plot the in-sample mean squared error and the cross-validated MSE as a function of d. For each dataset, what order polynomial has the best out of sample performance?

```
#calculate the in-sample mse.
poly.mse <-function(X, Y, degree)
{
  X.matrix = matrix(1, nrow = nrow(X), ncol = 1)
  if(degree > 0)
  {
    for(i in 1:degree)
      X.matrix = cbind(X.matrix, X^i)
  }
  beta = solve(crossprod(X.matrix), crossprod(X.matrix, Y))
  res = Y - X.matrix %*% beta
  sse = t(res) %*% res / (nrow(X) - degree -1)
  return(sse[1,1])
```

```
}

X = as.matrix(runif(100, min=-4, max=4))
epsilon = as.matrix(rnorm(100))
Y.1 = -2*(X<(-3)) + 2.55*(X>(-2)) - 2*(X>0) + 4*(X>2) - (X>3) + epsilon
Y.2 = 6 + 0.4*X - 0.36*X^2 + 0.005*X^3 + epsilon
Y.3 = 2.83 * sin(pi/2*X) + epsilon
Y.4 = 4 * sin(3*pi*X)*(X>0) + epsilon

#calcuate cv and mse for DGP1
cv.1 = rep(NaN, 11)
mse.1 = rep(NaN, 11)
k = 10
for(degree in 0:10)
{
  cv.1[degree+1] = poly.cv(X, Y.1, degree, k)
  mse.1[degree+1] = poly.mse(X, Y.1, degree)
}
plot(0:10, cv.1, pch = 1, ylim = c(0, 5), ylab="error", xlab="d")
points(0:10, mse.1, pch = 2)
legend(8.5, 5, pch=c(1,2), legend=c("cv", "mse"))
```

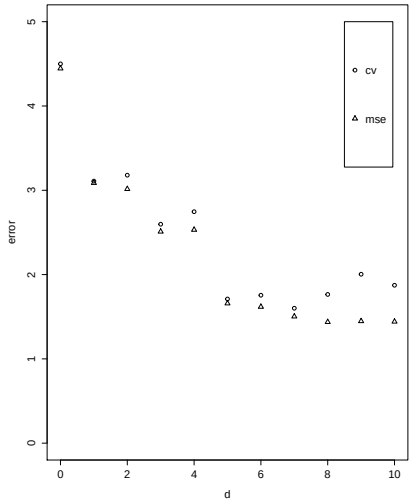


Figure 1: DGP1

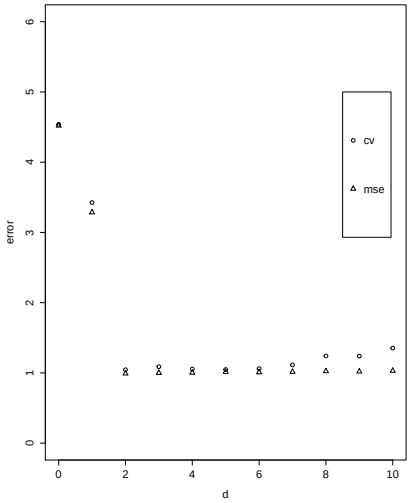


Figure 2: DGP1

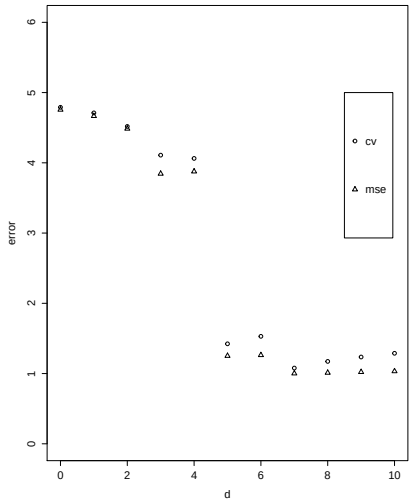


Figure 3: DGP3

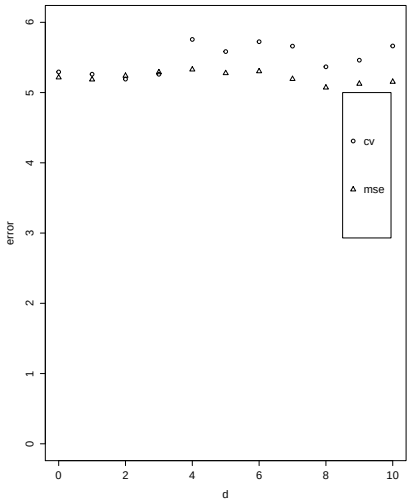


Figure 4: DGP4

The degrees are 5, 2, 5 and 0 respectively for DGP1, DGP2, DGP3 and DGP4.

3. *(3 pts.)* Fit a polynomial (using OLS) of the order selected in the previous part to each of the full datasets. Plot the estimated function together with the data.

```
poly.plot <- function(X, Y, degree)
{
  X.matrix = matrix(1, nrow = nrow(X), ncol = 1)
  if(degree > 0)
  {
    for(i in 1:degree)
```

```
      X.matrix = cbind(X.matrix, X^i)
  }
  beta = solve(crossprod(X.matrix), crossprod(X.matrix, Y))
  plot(X, Y)
  fit.Y = X.matrix%*%beta
  points(X[order(X),],fit.Y[order(X),], type ="l")
}
poly.plot(X, Y.1, 5)
poly.plot(X, Y.2, 2)
poly.plot(X, Y.3, 5)
poly.plot(X, Y.4, 0)
```

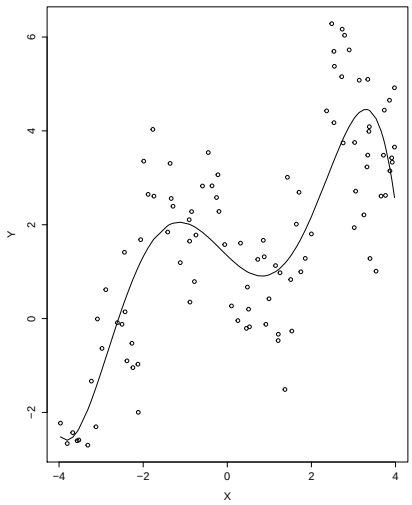


Figure 5: DGP1

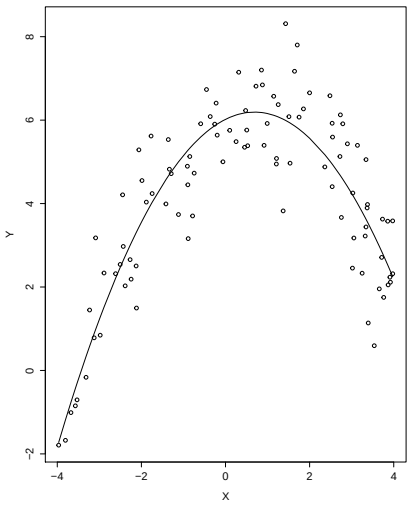


Figure 6: DGP1

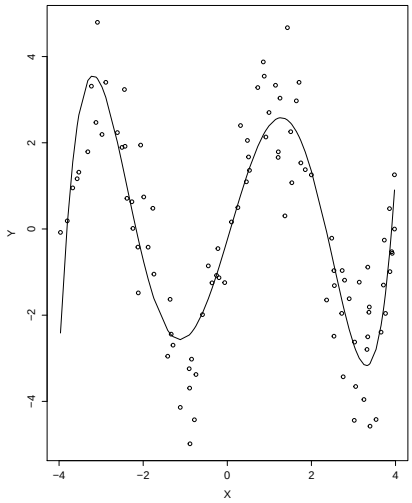


Figure 7: DGP3

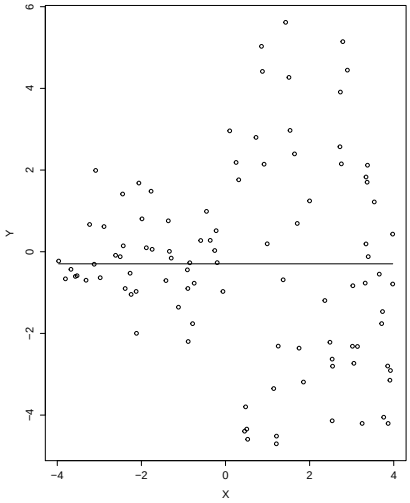


Figure 8: DGP4

4. *(5 pts.)* Generate a new sample of size $n = 10^4$ for each of the functions. What order of polynomial is now selected by cross-validation for each dataset? Explain why this is about the same as or different from what you got on the smaller sample.

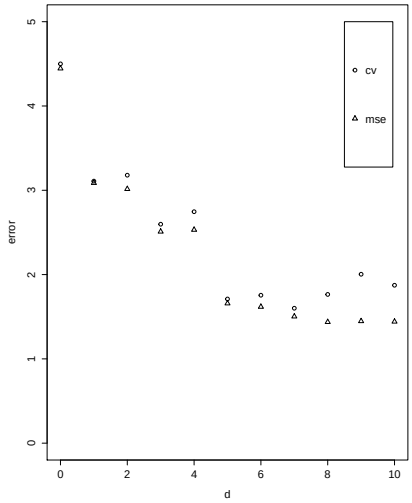


Figure 9: DGP1

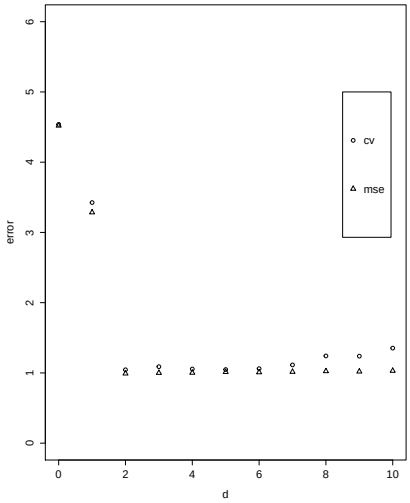


Figure 10: DGP1

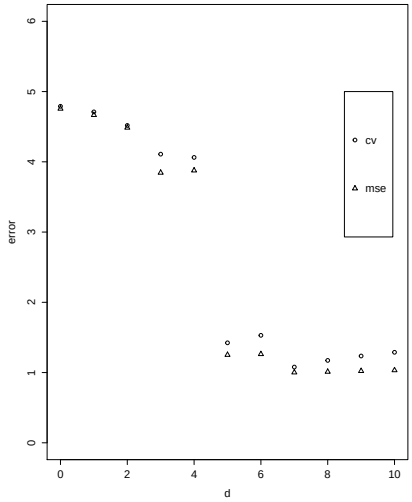


Figure 11: DGP3

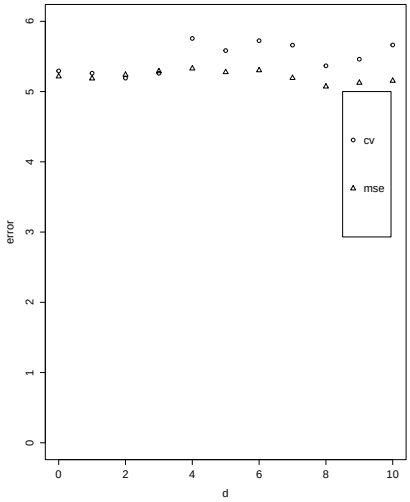


Figure 12: DGP4

The code is similar with problem 2, so I omit it. The degrees are 5, 2, 5 and 0 respectively for DGP1, DGP2, DGP3 and DGP4. When the sample size becomes larger, we gain more information so that I can fit the polynomial better. When the number of parameters is at most 5, samples, whose size is 100, are enough to estimate parameters with cross-validation.