Problem Set 2

Clinton Tepper

Note: On the previous assignment you asked why I used the notation $\int (\cdot)\, p(s)\, ds$. Generally I find using a new variable as a dummy inside the integrand enhances clarity. If it makes the results less clear, let me know and I will stop using dummy variables.

Problem 1

1.

- Write down the total likelihood:

$$L\left(\hat{\theta}|X\right) = \prod_{i\in 1:n} \iota\left(\hat{\theta} \geq x_i\right) \frac{1}{\hat{\theta}}$$
$$l\left(\hat{\theta}|X\right) = \begin{cases} \sum_{i\in 1:n} \frac{1}{\hat{\theta}} & \hat{\theta} \geq \max\left\{x_i\right\}_1^N \\ -\infty & otherwise \end{cases}$$

- Since l is strictly decreasing for all $\hat{\theta} \geq \max\left\{x_i\right\}_1^N$, we have $\hat{\theta}_{MLE} = \max\left\{x_i\right\}_1^N$

2.

- The likelihood interval is defined as $LI = \left\{\kappa :\ L(\kappa) > cL\left(\hat{\theta}\right)\right\}$
- Denote $\hat{\theta} \equiv \max\left\{x_i\right\}_1^N$. Then:

$$\begin{aligned} cL\left(\hat{\theta}\right) =& L(\theta) \\ \frac{c}{\hat{\theta}^N} =& \frac{1}{\kappa^N} \\ \kappa =& \frac{\hat{\theta}}{c^{\frac{1}{N}}} \end{aligned}$$

- Plugging in, $\hat{\theta} = 2.85$ so $\kappa = 5.19$

3.

- Begin with the definition:

$$\begin{aligned} p\left(cL\left(\hat{\theta}\right) < L(\theta)\right) =& p\left(\frac{c}{\hat{\theta}^n} < \frac{1}{\theta^n}\right) \\ =& 1 - P\left(c^{\frac{1}{n}}\theta > max(X)\right) \end{aligned}$$

- Then, appealing to the CDF of the maximum of a uniform distribution ($P(max(X) < \tau) = \left(\frac{\tau}{\theta}\right)^N$):

$$\begin{aligned} P\left(c^{\frac{1}{n}}\theta > max(X)\right) =& \left(\frac{c^{\frac{1}{n}}\theta}{\theta}\right)^N \\ \rightarrow p\left(cL\left(\hat{\theta}\right) < L(\theta)\right) =& 1 - c \end{aligned}$$

Problem 2

1.

- This is an odds ratio of odds ratios.
- The numerator is the ratio of the probability of someone voting to the probability of someone not voting given that they had a high level of education and covariates w.
- The denominator is the ratio of the probability of someone voting to the probability of someone not voting given that they had low education and covariates w.
- The overall expression is the ratio of the odds ratio of voting given high education to the odds ratio given low education, all given covariates w.

2.

- This is simply plugging into the provided assumption $p\left(Y_i=1|X_i\right)=\frac{\exp\left(X_i'\beta\right)}{1+\exp\left(X_i'\beta\right)}$:

$$
\begin{aligned}
p\left(Y_i=1|T_i=1,W_i=w\right)=&\frac{\exp\left(\alpha+\gamma+\delta'w\right)}{1+\exp\left(\alpha+\gamma+\delta'w\right)}\\
p\left(Y_i=0|T_i=1,W_i=w\right)=&\frac{1}{1+\exp\left(\alpha+\gamma+\delta'w\right)}\\
p\left(Y_i=1|T_i=0,W_i=w\right)=&\frac{\exp\left(\alpha+\delta'w\right)}{1+\exp\left(\alpha+\delta'w\right)}\\
p\left(Y_i=1|T_i=0,W_i=w\right)=&\frac{1}{1+\exp\left(\alpha+\delta'w\right)}
\end{aligned}
$$

- Plugging in:

$$
\begin{aligned}
OR\left(w\right)=&\frac{p\left(Y_i=1|T_i=1,W_i=w\right)/p\left(Y_i=0|T_i=1,W_i=w\right)}{p\left(Y_i=1|T_i=0,W_i=w\right)/p\left(Y_i=0|T_i=0,W_i=w\right)}\\
=&\frac{\exp\left(\alpha+\gamma+\delta'w\right)}{\exp\left(\alpha+\delta'w\right)}\\
=&\exp\left(\gamma\right)
\end{aligned}
$$

- This allows us to interpret the estimated coefficient $\hat{\gamma}$ as an estimate of the log of the odds ratio of interest.

3.

- By the continuous mapping theorem, $\hat{\sigma}_n\xrightarrow{p}\sigma$ implies $\hat{\sigma}_n^2\xrightarrow{p}\sigma^2$
- As the standard error exists and $\hat{\gamma}_n\xrightarrow{p}\gamma$ and $\hat{\sigma}_n^2\xrightarrow{p}\sigma^2$, we can apply the central limit theorem:

$$\sqrt{n}\left(\hat{\gamma}_n-\gamma\right)\xrightarrow{d}N\left(0,\ \sigma^2\right)$$

- Hence the delta method provides the asymptotic distribution:

$$
\begin{aligned}
\sqrt{n}\left(g\left(\hat{\gamma}_n\right)-g\left(\gamma\right)\right)\xrightarrow{d}&N\left(0,\ \sigma^2 g'\left(\gamma\right)^2\right)\\
\sqrt{n}\left(e^{\hat{\gamma}_n}-e^{\gamma}\right)\xrightarrow{d}&N\left(0,\ \sigma^2 e^{2\gamma}\right)
\end{aligned}
$$

Problem 2

4.

- We have $X_i'\beta=\pi_i$, so the link function is given by $g\left(\mu\right)=\mu$.

5.

- We have

$$\begin{aligned} E\left[\varepsilon_i^2|X_i\right] =& E\left[\left(Y_i - X_i'\beta\right)^2|X_i\right] \\ =& E\left[Y_i^2|X_i\right] - E\left[Y_iX_i\beta|X_i\right] + \left(X_i'\beta\right)^2 \\ =& E\left[Y_i^2|X_i\right] - \left(X_i'\beta\right)^2 \end{aligned}$$

- Since $p\left(Y_i|X_i\right) \sim B\left(\pi_i\right)$, we have

$$\begin{aligned} E\left[\varepsilon_i^2|X_i\right] =& V\left[Y_i|X_i\right] + \pi_i^2 - \pi_i^2 \\ =& \pi_i\left(1-\pi_i\right) \end{aligned}$$

- Thus outside of some degenerate cases (e.g. π_i is a constant), the error terms are conditionally heteroskedastic.
- Per White 1980, if exogeneity holds such that $E\left[\varepsilon|X_i\right] = 0$, corrected estimators of the standard error are asymptotically consistent.

6.

- The likelihood is given by:

$$\begin{aligned} L\left(\theta|X\right) &= \prod_{i\in 1:P} \pi_i^{Y_i}\left(1-\pi_i\right)^{1-Y_i} \\ &= \prod_{i\in 1:P} \left(X_i\beta\right)^{Y_i}\left(1-X_i'\beta\right)^{1-Y_i} \end{aligned}$$

7.

- Let $n_Y = \sum Y_i$. Then the log likelihood is:

$$l\left(\theta|X\right) = \sum_{i\in 1:P}\left[Y_i ln\left(X_i'\beta\right) + \left(1-Y_i\right)ln\left(1-X_i'\beta\right)\right]$$

- The score is thus

$$\begin{aligned} S\left(\beta|X\right) = \nabla l\left(\theta|X\right) &= \sum_{i\in 1:P}\left[\frac{Y_i}{X_i'\beta}X_i - \frac{1-Y_i}{1-X_i'\beta}X_i\right] \\ &= \sum_{i\in 1:P}\left[\frac{Y_i\left(1-X_i'\beta\right) - X_i'\beta\left(1-Y_i\right)}{X_i'\beta\left(1-X_i'\beta\right)}X_i\right] \\ &= \sum_{i\in 1:P}\left[\frac{Y_i - X_i'\beta}{X_i'\beta\left(1-X_i'\beta\right)}X_i\right] \checkmark \end{aligned}$$

8.

- Under correct specification,

$$\beta_{MLE} \sim N\left(\beta,\ -E\left[H\right]^{-1}\right)$$

- But under correct specificaiton,

$$\begin{aligned} -E\left[H\right]^{-1} =& I_N\left(\beta|X\right)^{-1} \\ =& E\left[S\left(\beta|X_i\right)S\left(\beta|X_i\right)'\right]^{-1} \end{aligned}$$

- Then

$$\begin{aligned}
S\left(\beta|X_i\right)S\left(\beta|X_i\right)' &= \left[\frac{Y_i - X_i'\beta}{X_i'\beta\left(1-X_i'\beta\right)}\right]^2 X_iX_i' \\
E\left[S\left(\beta|X_i\right)S\left(\beta|X_i\right)'\right] =& E\left[\left[\frac{Y_i - X_i'\beta}{X_i'\beta\left(1-X_i'\beta\right)}\right]^2 X_iX_i'|X_i\right] \\
=& E\left[\left[\frac{Y_i - X_i'\beta}{X_i'\beta\left(1-X_i'\beta\right)}\right]^2 |X_i\right] X_iX_i' \\
=& \frac{V\left(\varepsilon|X_i\right)}{\left[X_i'\beta\left(1-X_i'\beta\right)\right]^2} X_iX_i' \\
=& \frac{X_i'\beta\left(1-X_i'\beta\right)}{\left[X_i'\beta\left(1-X_i'\beta\right)\right]^2} X_iX_i' \text{ (from Q5)} \\
=& \frac{X_iX_i'}{X_i'\beta\left(1-X_i'\beta\right)}
\end{aligned}$$

- Plugging in, we thus have

$$\beta_{MLE} \sim N\left(\beta,\ \left[\frac{X_iX_i'}{X_i'\beta\left(1-X_i'\beta\right)}\right]^{-1}\right)$$

Problem 3

9.

- The likelihood and log-likelihood are given by:

$$\begin{aligned}
L\left(\theta|X\right) &= \prod_{i\in 1:P}\Phi\left(X_i'\beta\right)^{Y_i}\left(1-\Phi\left(X_i'\beta\right)\right)^{1-Y_i} \\
l\left(\theta|X\right) &= \sum_{i\in 1:P}\left[Y_i\ln\left[\Phi\left(X_i'\beta\right)\right] + \left(1-Y_i\right)\ln\left(1-\Phi\left(X_i'\beta\right)\right)\right]
\end{aligned}$$

10. and 11.

(Sorry for the wacky R code. Julia is my main language.)

- Need gradient for reliable optimization:

$$\nabla l\left(\theta|X\right) = \left[\frac{Y_i}{\Phi\left(X_i'\beta\right)} - \frac{\left(1-Y_i\right)}{1-\Phi\left(X_i'\beta\right)}\right]\phi\left(X_i'\beta\right)\beta$$

```r
require(ggplot2) #for graphs
require(parallel) #good for bootstrapping
require(data.table) #this and the below package are needed to work with data
require(knitr)
set.seed(11) #A seed for me



#holds constants and program parameters
CONST = list(
    NUM_ROWS = 200,
    NUM_COLS = 3,
    NUM_SAMPLES = 2000,
```

```r
    EPSILON = .Machine$double.eps, #machine precision
    NUM_WORKERS = max(round(detectCores() * .5), 2) #just a heuristic for multi-threading
)

#This is the probit likelihood function
llikelihoodProbit = function(b, Y, X) {
    epsilon = CONST$EPSILON
    argvec = X %*% b

    #avoid numerical issues with logs of small numbers
    pnorms = pmin(pmax(pnorm(argvec), epsilon), 1.0 - epsilon)

    #Use vectorized ifelse
    likes = ifelse(Y, log(pnorms), log(1 - pnorms))

    return(sum(likes))
}

#this is the gradient of the previous
llikelihoodProbitGrad = function(b, Y, X) {
    epsilon = CONST$EPSILON
    argvec = X %*% b
    pnorms = pnorms = pmin(pmax(pnorm(argvec), epsilon), 1.0 - epsilon)
    dnorms = dnorm(argvec)

    #Use vectorized ifelse
    premults = ifelse(Y, (1 / pnorms), - (1 / (1 - pnorms))) * dnorms

    #R's equivelent to broadcast
    grads = apply(X, MARGIN = 2, function(x) x * premults)
    return(colSums(grads))
}


probitModel = function(Y, X, suppressIntercept = FALSE) {
    #make the intercept as needed
    if (!suppressIntercept) {
        if (min(X[, ncol(X)]) != 1 || max(X[, ncol(X)]) != 1) X = cbind(X, rep(1, nrow(X)))
        }

    # Get some convenience constants
    R = nrow(Y)
    C = ncol(X)

    #initial value of b
    b = rep(1, C)

    #make single argument versions for optim
    ll = function(x) - 1.0 * llikelihoodProbit(x, Y, X)
    llgrad = function(x) - 1.0 * llikelihoodProbitGrad(x, Y, X)

    #call the optimizer
    opt = optim(b, ll, gr = llgrad, method = "BFGS", hessian = TRUE)
```

```r
    if (opt$convergence != 0) print("WARNING! Optimizer did not converge")

    #Efficient matrix inversion
    U = chol(opt$hessian)
    UInv = solve(chol(opt$hessian))
    Sigma = t(UInv) %*% UInv

    #form the info we want into a named list
    prob = list(B = opt$par, llikelihood = opt$value, varB = diag(Sigma), seB = diag(Sigma) ^ 0.5)
    return(prob)
}

#generates a test sample from the asymtotic distribution
testSample = function(R = CONST$NUM_ROWS, C = CONST$NUM_COLS, beta = 1 / (1:C)) {
    #pre-allocate
    X = matrix(rnorm(R * C), nrow = R, ncol = C)

    #create the Y vector
    Y = apply(X, 1, function(x) pnorm(x %*% beta))
    Y = rbinom(R, 1, Y)

    return(list(Y = Y, X = X))
}

#tests the model a single time and prints the results
testProbitModelOnce = function() {

    S = testSample()
    prob = probitModel(S$Y, S$X)
    print(prob)
}

testProbitModelOnce()
```

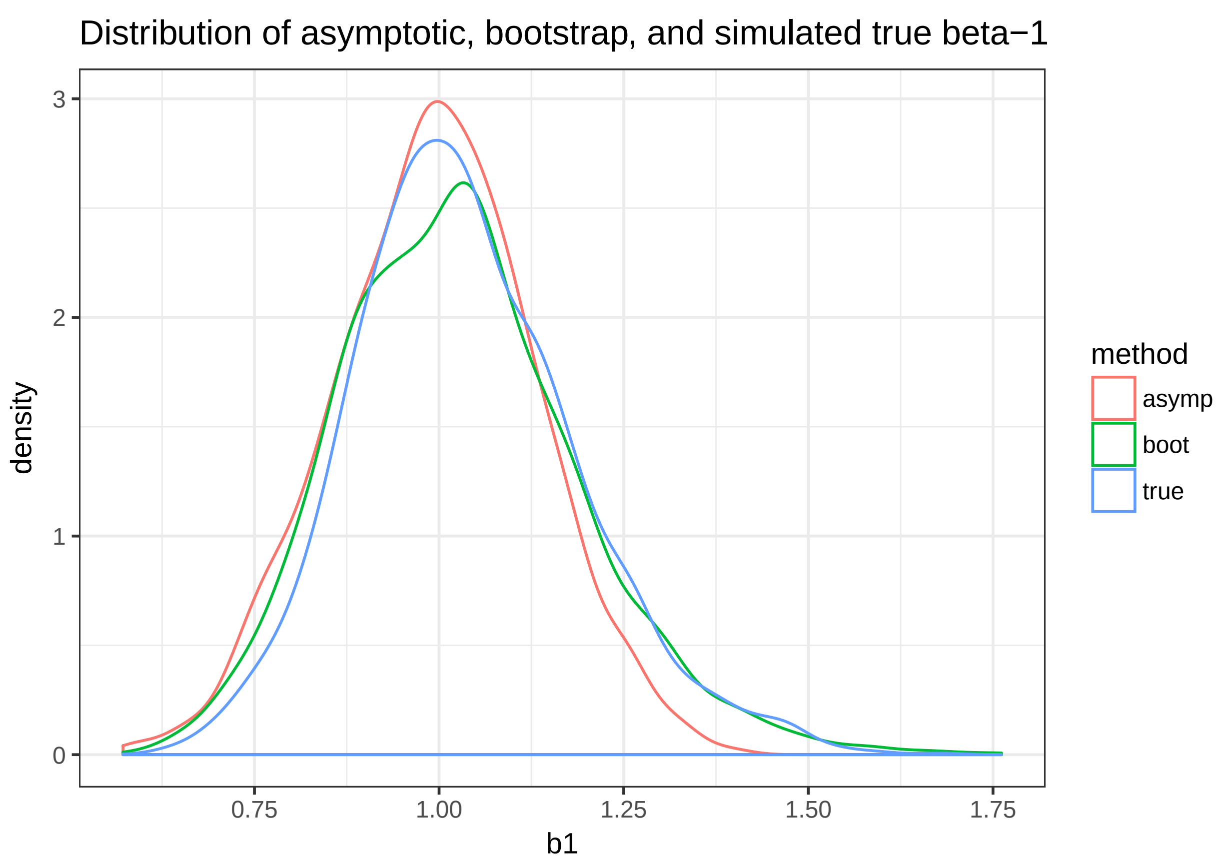
```
## $B
## [1]  1.294512138  0.429558179  0.302285828 -0.008555375
##
## $llikelihood
## [1] 85.56511
##
## $varB
## [1] 0.02820121 0.01650585 0.01472355 0.01217555
##
## $seB
## [1] 0.1679322 0.1284751 0.1213406 0.1103429
```

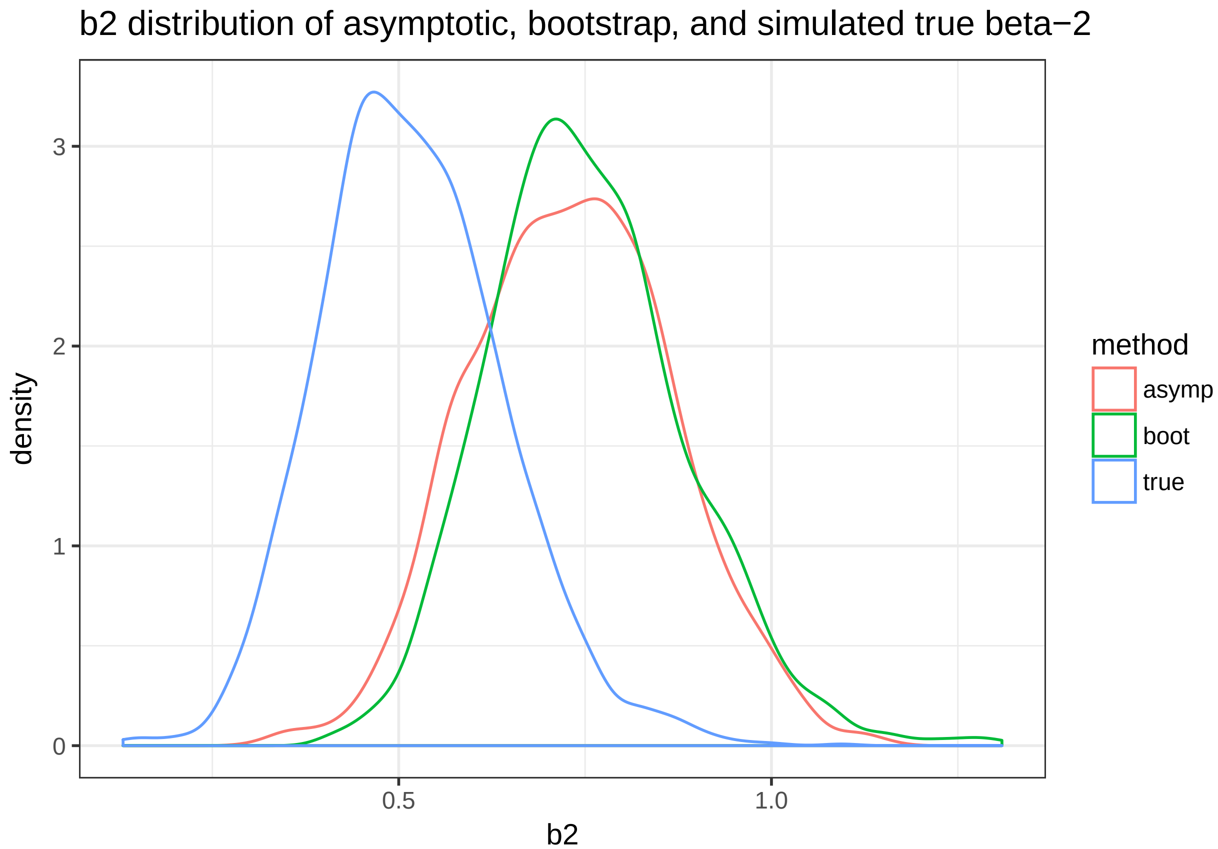
- Note that the true betas are 1.0, 0.5 and 0.33
- The results seem reasonably close to the true betas given the small sample size and binary nature of the dependent variable.
- From a frequentest standpoint, we cannot reject any of the true betas using the estimates.

12.

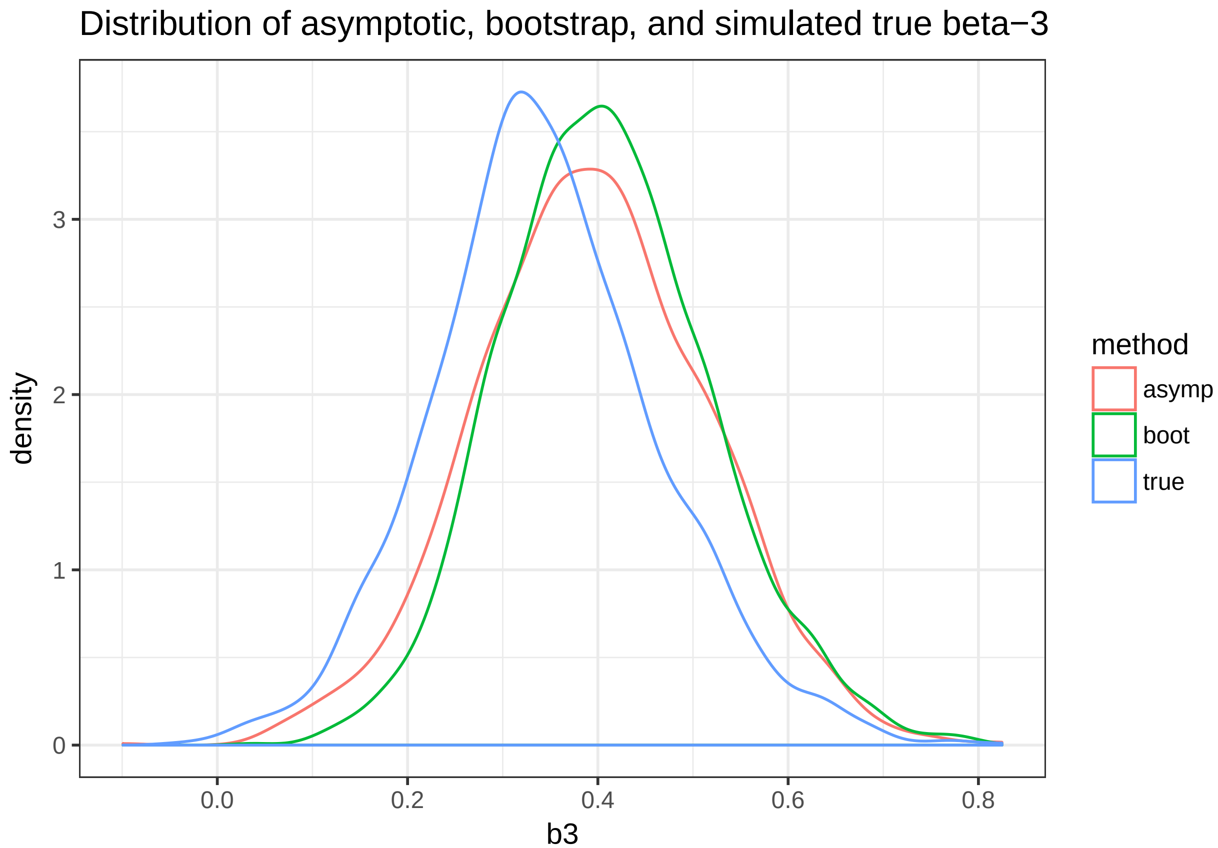
```r
#this generates a multi-variate bootstrap sample
bootSample = function(Y, X) {
    R = nrow(X)

    #first pick the rows we will sample
    sampledRows = sample(1:R, R, replace = TRUE)

    #sample the rows
    Y = sapply(sampledRows, function(r) Y[r])
    X = matrix(sapply(sampledRows, function(r) X[r,]), nrow = R, byrow = TRUE)

    return(list(Y = Y, X = X))
}

examineProbitDistributions = function(N = CONST$NUM_SAMPLES) {
    #maybe this will take a while, so lets multi-thread (process)
    cl = makeCluster(CONST$NUM_WORKERS)
    clusterExport(cl = cl,
        varlist = c("llikelihoodProbit", "llikelihoodProbitGrad", "probitModel",
        "testSample", "CONST", "bootSample"))

    #get the primary sample and model
    S = testSample()
    prob = probitModel(S$Y, S$X)
    betasAsymp = data.table(method = "asymp", b1 = rnorm(N, mean = prob$B[1], sd = (prob$varB[1] ^ 0.5))
        b2 = rnorm(N, mean = prob$B[2], sd = (prob$varB[2] ^ 0.5)),
        b3 = rnorm(N, mean = prob$B[3], sd = (prob$varB[3] ^ 0.5))
    )

    #Get the bootstrap samples and solve for the MLE
    bootSamples = parLapply(cl, 1:N, function(x) bootSample(S$Y, S$X))
    bootModels = parLapply(cl, bootSamples, function(s) probitModel(s$Y, s$X))
    betasBoot = data.table(method = "boot", b1 = sapply(bootModels, function(x) x$B[1]),
        b2 = sapply(bootModels, function(x) x$B[2]),
        b3 = sapply(bootModels, function(x) x$B[3]))

    #get the true samples
    trueSamples = parLapply(cl, 1:N, function(x) testSample())
    trueModels = parLapply(cl, trueSamples, function(s) probitModel(s$Y, s$X))
    betasTrue = data.table(method = "true", b1 = sapply(trueModels, function(x) x$B[1]),
        b2 = sapply(trueModels, function(x) x$B[2]),
        b3 = sapply(trueModels, function(x) x$B[3]))

    #combine into a ggplot2 friendly structure
    betas = rbind(betasAsymp, betasBoot, betasTrue)

    #plot the densities of the estimates
    p1 = ggplot(betas, aes(x = b1)) +
        geom_density(aes(group = method, color = method)) + theme_bw() +
        ggtitle("Distribution of asymptotic, bootstrap, and simulated true beta-1")
```

```
    p2 = ggplot(betas, aes(x = b2)) +
        geom_density(aes(group = method, color = method)) + theme_bw() +
        ggtitle("b2 distribution of asymptotic, bootstrap, and simulated true beta-2")

    p3 = ggplot(betas, aes(x = b3)) +
        geom_density(aes(group = method, color = method)) + theme_bw() +
        ggtitle("Distribution of asymptotic, bootstrap, and simulated true beta-3")
    print(p1)
    print(p2)
    print(p3)

    #cleanup
    stopCluster(cl)

}

system.time(examineProbitDistributions())
```





b2 distribution of asymptotic, bootstrap, and simulated true beta−2
density
0
1
2
3
0.5
1.0
b2
method
asymp
boot
true




```
##    user  system elapsed
##    1.28    2.50   13.96
```

- The distribution of the bootstrap and asymtotic error seem reasonably close.
- Qualitatively, the median of the asymptotic and bootstrap distribution at least occurs within a reasonable part of the true beta distribution.
- Because the true distribution seems leptokurtic, I generally trust the bootstrap more in this situation.

Problem 4

13.

- Because the conditional mean of both specifications is the same, the MLE estimates of beta are consistent.

14.

- As discussed in the slides, the sandwich estimator does not simplify.
  - The estimator is thus distributed $\hat{\theta} \sim N\left(\theta,\, E\left[H^{-1}\right] E\left[S\left(\theta\right) S\left(\theta\right)'\right] E\left[H^{-1}\right]\right)$
- Only under correct specification does $I\left(\theta|X_i\right) = E\left[S\left(\theta|X_i\right) S\left(\theta|X_i\right)'\right]$
- PROOF (Univariate case, borrowing from MLE2_handout.pdf slides 8 and 9):

– First write down the square of the score, but using the true probability distribution to compute the expectation:

$$\begin{aligned}
E\left[S\left(\theta|Y_i\right)^2\right] &= \int S\left(\theta|Y_i\right)^2 q\left(Y_i|\theta\right) dY_i \\
&= \int \left[\frac{\partial lnp\left(Y_i|\theta\right)}{\partial\theta}\right]^2 q\left(Y_i|\theta\right) dY_i \\
&= \int \frac{1}{p\left(Y_i|\theta\right)^2}\left[\frac{\partial p\left(Y_i|\theta\right)}{\partial\theta}\right]^2 q\left(Y_i|\theta\right) dY_i
\end{aligned}$$

– Do the same for the Hessian (doesn't quite match up due to typo in bottom of slide 8):

$$\begin{aligned}
-E\left[H\left(\theta|Y_i\right)\right] =& -\int \frac{\partial^2 lnp\left(Y_i|\theta\right)}{\partial\theta^2} q\left(Y_i|\theta\right) dY_i \\
\frac{\partial^2 lnp\left(Y_i|\theta\right)}{\partial\theta^2} =& \frac{\partial}{\partial\theta}\left[\frac{1}{p\left(Y_i|\theta\right)}\frac{\partial p\left(Y_i|\theta\right)}{\partial\theta}\right] \\
=& \frac{-1}{p\left(Y_i|\theta\right)^2}\left(\frac{\partial p\left(Y_i|\theta\right)}{\partial\theta}\right)^2 + \frac{1}{p\left(Y_i|\theta\right)}\frac{\partial^2 p\left(Y_i|\theta\right)}{\partial^2\theta} \\
-E\left[H\left(\theta|Y_i\right)\right] =& -\int\left[\frac{-q\left(Y_i|\theta\right)}{p\left(Y_i|\theta\right)^2}\left(\frac{\partial p\left(Y_i|\theta\right)}{\partial\theta}\right)^2 + \frac{q\left(Y_i|\theta\right)}{p\left(Y_i|\theta\right)}\frac{\partial^2 p\left(Y_i|\theta\right)}{\partial^2\theta}\right] dY_i \\
=& E\left[S\left(\theta|Y_i\right)^2\right] - \int \frac{q\left(Y_i|\theta\right)}{p\left(Y_i|\theta\right)}\frac{\partial^2 p\left(Y_i|\theta\right)}{\partial^2\theta} dY_i \\
=& E\left[S\left(\theta|Y_i\right)^2\right] - \int \frac{q\left(Y_i|\theta\right)}{p\left(Y_i|\theta\right)}\frac{\partial^2 p\left(Y_i|\theta\right)}{\partial\theta^2} dY_i \checkmark
\end{aligned}$$

– Note if q=p we achieve the desired simplification.

15 and 16

```
require(ggplot2) #for graphs
require(sandwich) #standard error
require(parallel) #good for bootstrapping
require(data.table) #this and the below package are needed to work with data


set.seed(11) #A seed for me



#holds constants and program parameters
CONST = list(
    NUM_ROWS = 1000,
    NUM_SAMPLES = 10000,
    EPSILON = .Machine$double.eps, #machine precision
    NUM_WORKERS = max(round(detectCores() * .5), 2), #just a heuristic for multi-threading
    NBSIZE = 1 / 3

    )

#generates a test sample from the asymtotic distribution
binomSample = function(R = CONST$NUM_ROWS, nbsize = CONST$NBSIZE) {
```

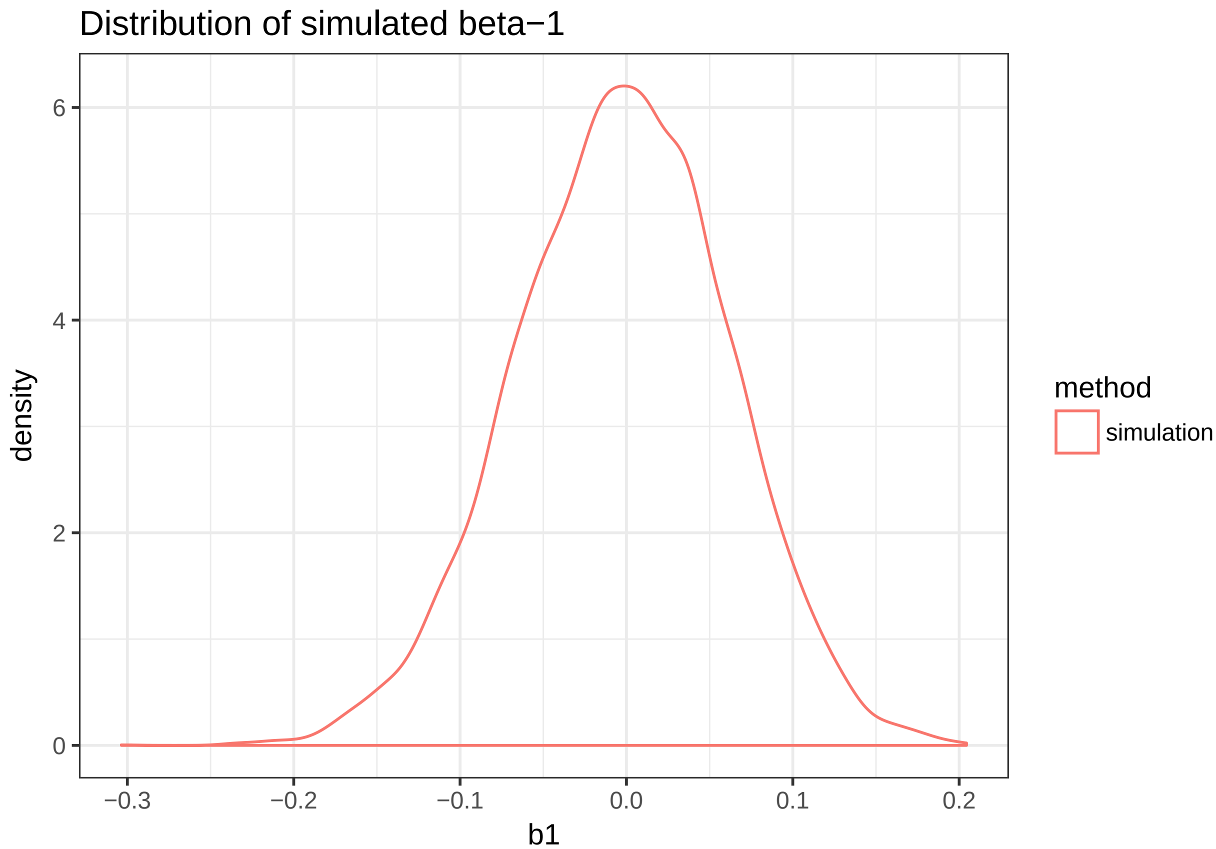
```r
    #pre-allocate
    X = rnorm(R)

    #create the Y vector
    Y = exp(X / 100)
    Y = rnbinom(R, size = CONST$NBSIZE, mu = Y)

    return(list(Y = Y, X = X))
}

glmPoisson = function(Y, X) {
    pois = glm(Y ~ X, family = poisson())
    beta = pois$coefficients

    #Get the standard SE
    AInv = vcov(pois)
    SE = diag(AInv) ^ 0.5
    names(SE) = names(beta)

    #Get the robust SE
    score = estfun(pois)
    B = t(score) %*% score
    AInvBAInv = AInv %*% B %*% AInv
    SERobust = diag(AInvBAInv) ^ 0.5
    names(SERobust) = names(beta)

    return(list(beta = beta, SE = SE, SERobust = SERobust))
}

compareModelsOnce = function() {
    S = binomSample() #get the main sample
    pois = glmPoisson(S$Y, S$X) #get the model output
    print(pois) #print it
}

compareModelsOnce()
```

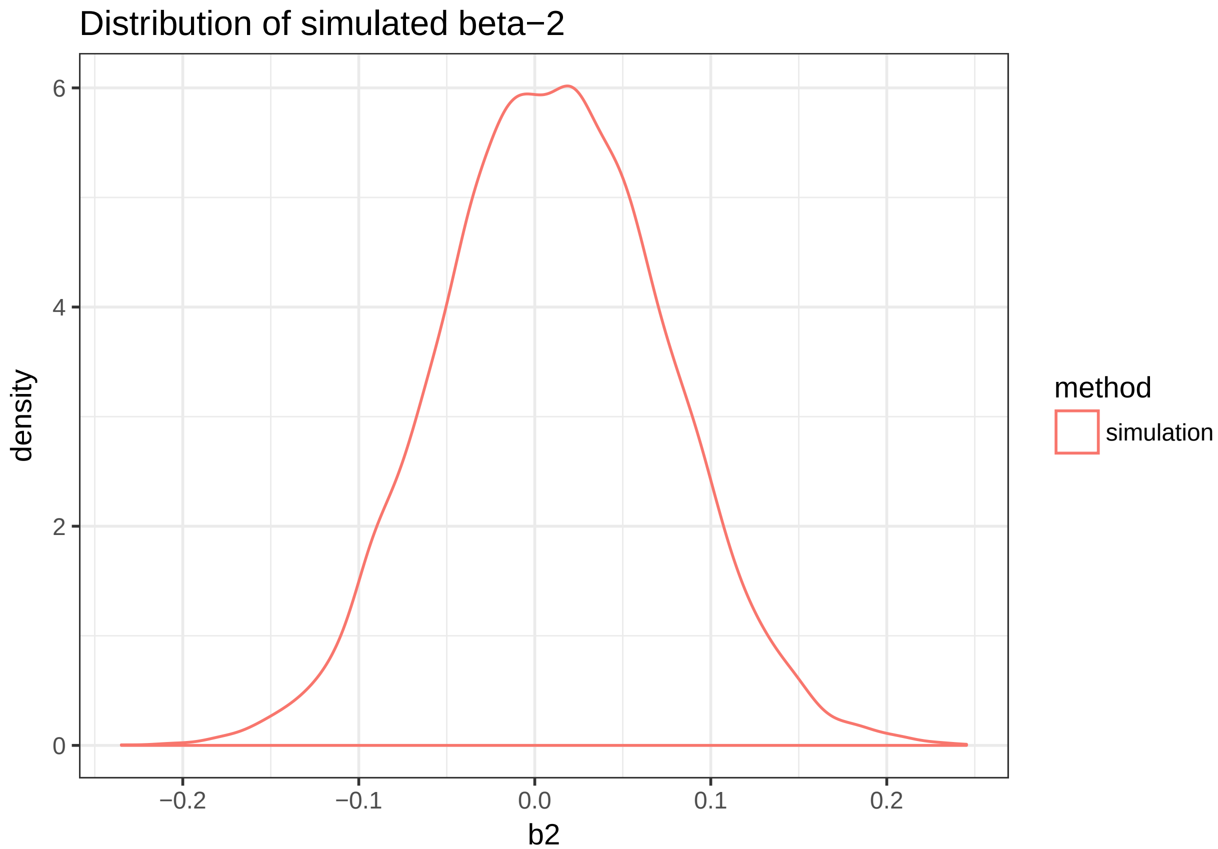
```
## $beta
## (Intercept)           X
## -0.01103343 -0.01367544
##
## $SE
## (Intercept)           X
##  0.03179847  0.03194003
##
## $SERobust
## (Intercept)           X
##  0.06358881  0.05710182
```

- As expected, the standard errors are higher using the more robust technique.

17.

```r
simulateModel = function(K = CONST$NUM_SAMPLES) {
    #prepare to parallelize
    cl = makeCluster(CONST$NUM_WORKERS)
    clusterExport(cl = cl, varlist = c("glmPoisson", "binomSample", "CONST"))
    clusterEvalQ(cl, require(sandwich))

    #first get the samples
    samples = parLapply(cl, 1:K, function(x) binomSample())
    simModels = parLapply(cl, samples, function(x) glmPoisson(x$Y, x$X))
    betas = data.table(method = "simulation", b1 = sapply(simModels, function(x) x$beta[1]),
        b2 = sapply(simModels, function(x) x$beta[2]))

    #plot
    p1 = ggplot(betas, aes(x = b1)) +
        geom_density(aes(group = method, color = method)) + theme_bw() +
        ggtitle("Distribution of simulated beta-1")

    p2 = ggplot(betas, aes(x = b2)) +
        geom_density(aes(group = method, color = method)) + theme_bw() +
        ggtitle("Distribution of simulated beta-2")

    print(p1)
    print(p2)

    cat("Cross-sectional standard deviation of b1: ", sd(betas[, b1]), "\n")
    cat("Cross-sectional standard deviation of b2: ", sd(betas[, b2]), "\n")

    #cleannup
    stopCluster(cl)
}


system.time(simulateModel())
```


Distribution of simulated beta−1
density
6
4
2
0
−0.3
−0.2
−0.1
0.0
0.1
0.2
b1
method
simulation




```
## Cross-sectional standard deviation of b1:  0.06374631
## Cross-sectional standard deviation of b2:  0.06375208
```

```
##    user  system elapsed
##    2.35    7.45   26.61
```

- The true standard errors seem reasonably close to the standard errors from the robust estimation technique.
- They are substantially more than the standard errors computed assuming correct specification.