Simulation

Sam Lee

Data-Generating Process for Simulation

Bootstrap samples by cluster

```
DGP <- function(G, clusters, Y, g.index, cluster=F){
  if(cluster){
    data = sapply(1:G, function(i){
      data.g = Y[(lag(g.index, default=0)+1)[i]:g.index[i]]
      #plot(data.g)
      data.new = sample(data.g, clusters[i], replace=T)
      #points(data.new, col='red')
      # (cov(women.cleaned$Z.ct1[(lag(g.index, default=0)+1)[i]:g.index[i]],
      #       data.new)/
      #     var(women.cleaned$Z.ct1[(lag(g.index, default=0)+1)[i]:g.index[i]])) %>%
```

```
      #   print()
      data.new
    })
    return(unlist(data))
  } else {
    return(sample(Y, length(Y), replace=T))
  }
}
#DGP(G, g.map$total, Y.tilde, g.map$g.index, cluster=bootstrap.cluster)


# Y.new = DGP(G, clusters, Y, g.index, T)
# # Y.new2 = DGP(G, clusters, Y, g.index, T)
# cov(Y.new, women.cleaned$Z.ct1)/var(women.cleaned$Z.ct1)
# # cov(Y.new2, women.cleaned$W.ct1)/var(women.cleaned$W.ct1)
# # cov(Y, women.cleaned$W.ct1)/var(women.cleaned$W.ct1)
# coefs = c()
# for(g in 1:G){
#   cluster = (lag(g.map$g.index, default=0)+1)[g]:g.map$g.index[g]
#   Y.new.g = Y.new[cluster]
#   coefs[g] = cov(Y.new.g, women.cleaned$Z.ct1[cluster])/
```

```
#  var(women.cleaned$Z.ct1[cluster])
# }
# mean(coefs)
```

Two-Stage Inference with Homoskedasticity Assumption

Linear Regression with Asymptotic Homoskedastic Normal Variance Assumption

```
ols <- function(X, Y, robust=F){
  n = nrow(X)
  k = ncol(X)-1
  beta.hat <- solve(t(X)%*%X)%*%t(X)%*%Y

  if(robust){
    u.hat = (X%*%beta.hat-Y)^2
    sigma <- matrix(0, k+1, k+1)
    for(i in 1:nrow(u.hat)){
      sigma <- sigma+(X[i,]%*%t(X[i,]))*u.hat[i]
    }
```

```
    V <- (n^2)/((n-k))*solve(t(X)%*%X)%*%sigma%*%t(solve(t(X)%*%X))
    #95% C.I. for delta
    beta.hat[1] + c(-1,1)*qnorm(0.975)*sqrt(n^-1*V[1,1])
  } else {
    epsilon <- sum((X%*%beta.hat-Y)^2)
    #DF correction
    sigma <- epsilon/(n-k)*(t(X)%*%X)
    V <- solve(t(X)%*%X)%*%sigma%*%solve(t(X)%*%X)
    #95% C.I. for delta
    beta.hat[1] + c(-1,1)*qnorm(0.975)*sqrt(V[1,1])
  }
}

ols(X, Y, robust=F)
```

```
[1] -1.583866 -0.131048
```

```
ols(X, Y, robust=T)
```

```
[1] -1.4865395 -0.2283743
```

```r
#Setup for Two-Stage linear Regression & Calculation of Cluster Robust Variance
G = women.cleaned %>% pull(countrycode) %>% unique() %>% length()
# X.tilde <- cbind(women.cleaned[,str_c("Z.ct", 1:lag.effect)], fixed.effects) %>% a
# Y.tilde <- women.cleaned[,str_c("W.ct", 1:lag.effect)] %>% as.matrix()
# delta.tilde.hat <- solve(t(X.tilde)%*%X.tilde)%*%t(X.tilde)%*%Y.tilde
# W.tilde <- X.tilde%*%delta.tilde.hat
#
# X.hat <- cbind(W.tilde, fixed.effects)
# k = ncol(X.hat)-1


g.map <- women.cleaned %>% group_by(countrycode) %>%
  summarize(
    total = n()
  ) %>% ungroup() %>% mutate(
    g.index = cumsum(total)
  )


Y.new = DGP(G, g.map$total, Y, g.map$g.index, cluster=T)
```

Cluster Robust Variance Function

```
#Partial out fixed effects
fixed.effects = X[,str_detect(colnames(X), "country|year")]
Y.tilde = (fixed.effects%*%
  solve(t(fixed.effects)%*%fixed.effects)%*%t(fixed.effects)%*%Y)-Y


covariates = setdiff(colnames(X), c(colnames(fixed.effects)))
X.tilde <- c()
for(covariate in covariates){
  X.tilde <- cbind(X.tilde, (fixed.effects%*%
  solve(t(fixed.effects)%*%fixed.effects)%*%t(fixed.effects)%*%
    X[,covariate])-X[,covariate])
}
colnames(X.tilde) <- c(covariates)


cluster.robust <- function(g.map, X, Y, cr=1){


  beta.hat <- solve(t(X)%*%X)%*%t(X)%*%Y
  k = ncol(X)-1
  n = nrow(X)
```

```
y.hat <- X%*%beta.hat

u <- y.hat-Y


g.start = lag(g.map$g.index, default=0)+1


if(cr == 1){
  group.s <- function(X,u.hat=u,g){
    X.g <- X[g,]
    u.g <- u.hat[g]

    s.g <- t(X.g)%*%u.g

    return(s.g%*%t(s.g))
  }

  cv.1.sigma <- matrix(0, k+1, k+1)
  for(i in length(g.map$g.index)){
    cv.1.sigma <- cv.1.sigma+group.s(X, g=g.start[i]:g.map$g.index[i])
  }
```

```
  CV.1 = ((G*(n-1))/((G-1)*(n-k)))*solve(t(X)%*%X)%*%
    cv.1.sigma%*%solve(t(X)%*%X)

  #95% Confidence Interval
  return(beta.hat[1,]+c(-1,1)*qt(0.975, G-1)*sqrt(CV.1[1,1]))
} else if (cr == 2){
  grave.s <- function(X,u.hat=u,g){
    X.g = X[g,]
    M = I(nrow(X.g))-X.g%*%solve(t(X)%*%X)%*%t(X.g)
    s.g = t(X.g)%*%matrixsqrtinv(M)%*%u.hat[g]
    s.g.prod <- (s.g)%*%t(s.g)
    #print(s.g.prod[1,1])
    return(s.g.prod)
  }

  cv.2.sigma <- matrix(0, k+1, k+1)
  for(i in length(g.map$g.index)){
    cv.2.sigma <- cv.2.sigma+grave.s(X, g=g.start[i]:g.map$g.index[i])
  }
```

```
  CV.2 = solve(t(X)%*%X)%*%
    cv.2.sigma%*%solve(t(X)%*%X)

  return(beta.hat[1,]+c(-1,1)*qnorm(0.975)*sqrt(CV.2[1,1]))
} else if(cr == 3){
  acute.s <- function(X,u.hat=u,g){
    X.g = X[g,]
    M = I(nrow(X.g))-X.g%*%solve(t(X)%*%X)%*%t(X.g)
    s.g = t(X.g)%*%solve(M)%*%u.hat[g]
    s.g.prod <- (s.g)%*%t(s.g)
    #print(s.g.prod[1,1])
    return(s.g.prod)
  }

  cv.3.sigma <- matrix(0, k+1, k+1)
  for(i in length(g.map$g.index)){
    cv.3.sigma <- cv.3.sigma+acute.s(X, g=g.start[i]:g.map$g.index[i])
  }

  CV.3 = ((G-1)/G)*solve(t(X)%*%X)%*%
```

```
    cv.3.sigma%*%solve(t(X)%*%X)


  # betags <- matrix(0, k+1, G)
  # for(g in 1:G){
  #   cluster = g.start[g]:g.map$g.index[g]
  #   betags[,g] <- solve(t(X)%*%X-t(X[-cluster,])%*%
  #                         X[-cluster,])%*%
  #     (t(X)%*%Y-t(X[-cluster,])%*%Y[-cluster,])
  # }
  # sigma <- (betags[,1]-beta.hat)%*%t(betags[,1]-beta.hat)
  # for(g in 2:G){
  #   sigma <- sigma+(betags[,2]-beta.hat)%*%t(betags[,1]-beta.hat)
  # }
  # CV.3 <- (G-1)/G*sigma


  return(beta.hat[1,]+c(-1,1)*qt(0.975, G-1)*sqrt(CV.3[1,1]))
 }
}
```

1. Cluster Robust Variance #1 (Liang-Zeger)

$$\text{CV}_1 : \frac{G(N-1)}{(G-1)(N-k)}(\hat{X}'\hat{X})^{-1}(\sum_{g=1}^{G}\hat{s}_g\hat{s}_g')(\hat{X}'\hat{X})^{-1}$$

$$\hat{s}_g = \hat{X}_g'\hat{u}_g$$

```
cluster.robust(g.map, X, Y, cr=1)
```

```
[1] -1.2150262 -0.4998876
```

2. Cluster Robust Variance #2

$$(\hat{X}'\hat{X})^{-1}(\sum_{g=1}^{G}\grave{s}_g\grave{s}_g')(\hat{X}'\hat{X})^{-1}$$

$$\grave{s}_g = \hat{X}'M_{gg}^{-1/2}\hat{u}_g; \quad M_{gg} = I_{N_g} - \hat{X}_g(\hat{X}'\hat{X})^{-1}\hat{X}_g'$$

```
cluster.robust(g.map, X.tilde, Y.tilde, cr=2)
```

```
[1] -1.8240812  0.1091673
```

3. Cluster Robust Variance #3 (Jackknife)

$$\frac{G-1}{G}(\hat{X}'\hat{X})^{-1}(\sum_{g=1}^{G}\acute{s}_g\acute{s}_g')(\hat{X}'\hat{X})^{-1}$$

$$\acute{s}_g = \hat{X}_g'M_{gg}^{-1}\hat{u}_g$$

```
cluster.robust(g.map, X, Y, cr=3)
```

```
[1] -4.145925  2.431011
```

Simulated CRVE Intervals

```
simulate.CRVE <- function(n, bootstrap.cluster=F){
  n.sims <- n
  sims.ci <- matrix(0, nrow=n.sims, ncol=2*5)
  for(i in 1:n.sims){
    #print(i)
    Y.new = DGP(G, g.map$total, Y, g.map$g.index, cluster=bootstrap.cluster)
    for(j in 1:5){
      if(j == 4){
        sims.ci[i,(j+(j-1)):(j+(j-1)+1)] = ols(X, Y.new, robust = F)
      } else if(j == 5){
        sims.ci[i,(j+(j-1)):(j+(j-1)+1)] = ols(X, Y.new, robust = T)
      } else {
        sims.ci[i,(j+(j-1)):(j+(j-1)+1)] =
          cluster.robust(g.map, X, as.matrix(Y.new), cr=j)
      }
```

```
    }
  }
  colnames(sims.ci) <- c("LZ.025", "LZ.975", "CR2.025", "CR2.975",
                         "JackKnife.025", "JackKnife.975",
                          "OLS.025", "OLS.975", "OLSR.025", "OLSR.0975")
  return(sims.ci)
}


coverage <- function(sims){
  coverages = c()
  k = 1
  for(i in 1:(ncol(sims)/2)){
    coverages[i] = sum(0 < sims[,k] & 0 < sims[,k+1] |
      (0 > sims[,k] & 0 > sims[,k+1]))/nrow(sims)
    k = k+2
  }
  names(coverages) = unlist(lapply(str_split(colnames(sims), "[.]"), first)) %>%
    unique()
  return(coverages)
}
```

```
simulate.CRVE(10) %>% round(2) %>% knitr::kable()
```

LZ.025	LZ.975	CR2.025	CR2.975	JackKnife.025	JackKnife.975	OLS.025	OLS.975	OLSR.025	OLSR.0975
-5.73	0.23	-14.66	9.16	-58.23	52.73	-9.62	4.12	-9.99	4.48
-0.13	3.45	-4.87	8.19	-23.99	27.32	-5.01	8.34	-4.53	7.85
-2.94	-1.48	-5.07	0.65	-14.97	10.55	-8.69	4.27	-8.99	4.56
5.60	6.07	4.90	6.77	0.87	10.80	-1.13	12.80	-0.89	12.56
-1.66	-1.46	-2.02	-1.10	-3.67	0.54	-8.40	5.27	-8.06	4.93
4.84	5.44	2.01	8.28	-21.30	31.59	-2.12	12.40	-2.21	12.50
-4.19	-3.58	-4.55	-3.22	-4.62	-3.15	-10.98	3.21	-10.90	3.13
-1.17	3.76	-7.53	10.13	-32.43	35.03	-5.68	8.28	-5.83	8.42
1.27	2.12	-0.17	3.56	-8.01	11.40	-5.52	8.91	-5.85	9.25
-1.45	1.71	-5.57	5.82	-21.53	21.79	-5.96	6.22	-5.27	5.53

Simulated CRVE Intervals with Clustered Bootstrap Sampling

```
sims.cluster = simulate.CRVE(1000, bootstrap.cluster = T)
```

```
sims.cluster %>%
  round(2) %>% head(10) %>% knitr::kable()
```

LZ.025	LZ.975	CR2.025	CR2.975	JackKnife.025	JackKnife.975	OLS.025	OLS.975	OLSR.025	OLSR.0975
-0.43	-0.40	-0.49	-0.35	-0.88	0.04	-1.27	0.44	-1.20	0.36
-0.05	0.13	-0.28	0.37	-1.24	1.32	-0.82	0.90	-0.88	0.97
-0.39	-0.26	-0.52	-0.13	-0.79	0.14	-1.22	0.57	-1.15	0.50
-0.26	-0.24	-0.37	-0.12	-1.36	0.87	-1.10	0.60	-1.14	0.65
0.06	0.34	-0.30	0.70	-1.71	2.12	-0.60	1.01	-0.56	0.96
-0.28	-0.24	-0.35	-0.17	-0.83	0.31	-1.02	0.50	-0.88	0.36
-0.36	-0.35	-0.43	-0.28	-0.95	0.24	-1.21	0.50	-1.16	0.45
-0.22	-0.14	-0.31	-0.05	-0.61	0.25	-1.04	0.68	-0.97	0.61
-0.25	-0.09	-0.43	0.10	-1.02	0.69	-0.90	0.56	-0.80	0.47
0.23	0.31	0.15	0.38	0.06	0.48	-0.48	1.02	-0.37	0.91

```
coverage(sims.cluster)
```

```
      LZ       CR2 JackKnife       OLS      OLSR
```

```
0.899      0.622      0.194      0.038      0.045
```