ARE213 Problem Set #2B

Peter Alstone & Frank Proulx

November 18, 2013

1 Problem #1

Table 1: Composite Treatment Group Summary

Statistic	N	Mean	St. Dev.	Min	Max
state	23	99.000	0.000	99	99
year	23	1,992.000	6.782	1,981	2,003
college	23	0.234	0.014	0.209	0.259
beer	23	1.507	0.074	1.394	1.670
primary	23	0.783	0.422	0	1
secondary	23	0.000	0.000	0	0
population	23	13,597.660	1,813.520	10,737.810	16,862.220
unemploy	23	6.085	1.124	3.855	8.014
fatalities	23	2,619.014	258.667	2,246.977	3,268.613
totalvmt	23	128,099.600	26,447.260	86,013.140	170,407.300
precip	23	2.502	0.289	1.990	3.104
snow32	23	0.143	0.058	0.013	0.270
rural_speed	23	63.443	6.568	55.000	72.886
urban_speed	23	59.184	5.858	55.000	67.138
logfatalpc	23	−1.643	0.168	−1.805	−1.189
sqyears	23	3,968,108.000	27,020.830	3,924,361	4,012,009

Table 2: Closest match for pre-policy fatalities: Alabama

Statistic	N	Mean	St. Dev.	Min	Max
state	23	1.000	0.000	1	1
year	23	1,992.000	6.782	1,981	2,003
college	23	0.170	0.029	0.131	0.220
beer	23	1.105	0.067	1.000	1.190
primary	23	0.174	0.388	0	1
secondary	23	0.304	0.470	0	1
population	23	4,185.794	209.389	3,918.533	4,501.862
unemploy	23	7.509	2.780	4.200	14.400
fatalities	23	1,036.957	88.042	839	1,189
totalvmt	23	44,826.090	10,109.350	27,852	58,637
precip	23	4.944	0.701	3.737	6.342
snow32	23	0.000	0.000	0	0
rural_speed	23	63.696	6.255	55	70
urban_speed	23	58.478	4.870	55	65
logfatalpc	23	−1.398	0.079	−1.543	−1.286
sqyears	23	3,968,108.000	27,020.830	3,924,361	4,012,009

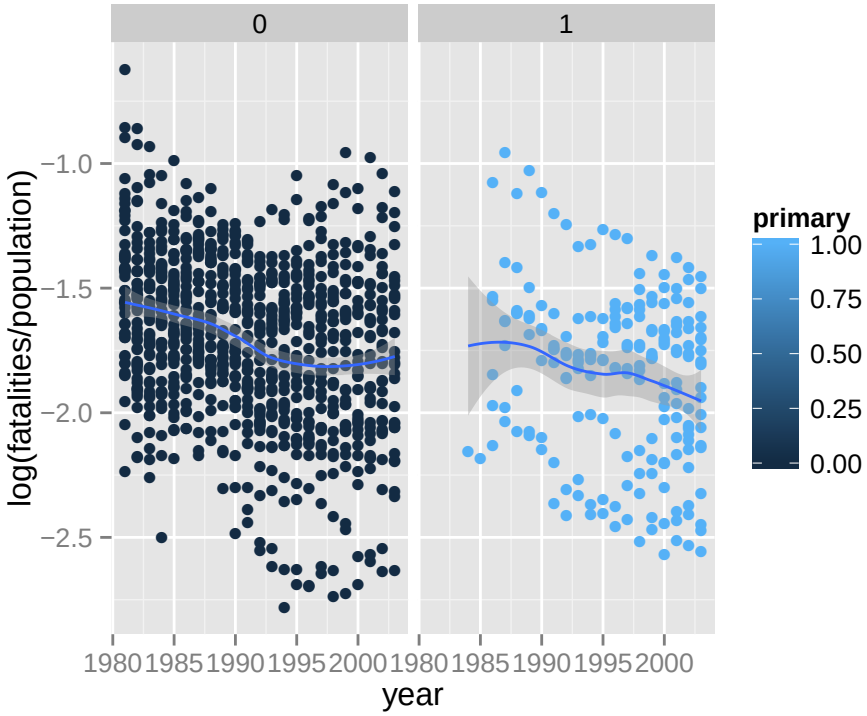

Figure 1: Year to year trends in the log of traffic fatalities per capita, divided by primary seatbelt law presence. A LOESS fit to each dataset is included for reference but is not necessarily indicative of the true underlying function.

2 Appendix: Code Listings

```
# Econometrics helper functions for [R]
#
# Peter Alstone and Frank Proulx
# 2013
# version 1
# contact: peter.alstone AT gmail.com

# Category: Data Management -------------


# Category: Data Analysis ----------------

# Function: Find adjusted R^2 for subset of data
# This requires a completed linear model...pull out the relevant y-values and residuals and feed them to function
# [TODO @Peter] Improve function so it can simply evaluate lm or glm object, add error handling, general clean up.
adjr2 <- function(y,resid){
  r2 <- 1-sum(resid^2) / sum((y-mean(y))^2)
  return(r2)
} #end adjr2


# Category: Plots and Graphics -------------

## Function for arranging ggplots. use png(); arrange(p1, p2, ncol=1); dev.off() to save.
require(grid)
vp.layout <- function(x, y) viewport(layout.pos.row=x, layout.pos.col=y)
arrange_ggplot2 <- function(..., nrow=NULL, ncol=NULL, as.table=FALSE) {
  dots <- list(...)
  n <- length(dots)
  if(is.null(nrow) & is.null(ncol)) { nrow = floor(n/2) ; ncol = ceiling(n/nrow)}
  if(is.null(nrow)) { nrow = ceiling(n/ncol)}
  if(is.null(ncol)) { ncol = ceiling(n/nrow)}
  ## NOTE see n2mfrow in grDevices for possible alternative
  grid.newpage()
  pushViewport(viewport(layout=grid.layout(nrow,ncol) ) )
  ii.p <- 1
  for(ii.row in seq(1, nrow)){
    ii.table.row <- ii.row
    if(as.table) {ii.table.row <- nrow - ii.table.row + 1}
    for(ii.col in seq(1, ncol)){
      ii.table <- ii.p
      if(ii.p > n) break
      print(dots[[ii.table]], vp=vp.layout(ii.table.row, ii.col))
      ii.p <- ii.p + 1
    }
  }
}

robust <- function(model){ #This calculates the Huber-White Robust standard errors -- code from http://thetarzan.wordpress.com/2011/05/28/
```

```
    heteroskedasticity-robust-and-clustered-standard-errors-in-r/
50      s <- summary(model)
51      X <- model.matrix(model)
52      u2 <- residuals(model)^2
53      XDX <- 0
54
55      for(i in 1:nrow(X)) {
56          XDX <- XDX +u2[i]*X[i,]%*%t(X[i,])
57      }
58
59  # inverse(X'X)
60      XX1 <- solve(t(X)%*%X)
61
62  #Compute variance/covariance matrix
63      varcovar <- XX1 %*% XDX %*% XX1
64
65  # Degrees of freedom adjustment
66      dfc <- sqrt(nrow(X))/sqrt(nrow(X)-ncol(X))
67
68      stdh <- dfc*sqrt(diag(varcovar))
69
70      t <- model$coefficients/stdh
71      p <- 2*pnorm(-abs(t))
72      results <- cbind(model$coefficients, stdh, t, p)
73      dimnames(results) <- dimnames(s$coefficients)
74      results
75  }
76
77  ## Two functions for clustered standard errors below from: http://people.su.
        se/~ma/clustering.pdf -------
78
79  clx <-
80    function(fm, dfcw, cluster){
81      # R-codes (www.r-project.org) for computing
82      # clustered-standard errors. Mahmood Arai, Jan 26, 2008.
83
84      # The arguments of the function are:
85      # fitted model, cluster1 and cluster2
86      # You need to install libraries 'sandwich' and 'lmtest'
87
88      # reweighting the var-cov matrix for the within model
89      library(sandwich);library(lmtest)
90      M <- length(unique(cluster))
91      N <- length(cluster)
92      K <- fm$rank
93      dfc <- (M/(M-1))*((N-1)/(N-K))
94      uj  <- apply(estfun(fm),2, function(x) tapply(x, cluster, sum));
95      vcovCL <- dfc*sandwich(fm, meat=crossprod(uj)/N)*dfcw
96      coeftest(fm, vcovCL) }
97
98  mclx <-
99    function(fm, dfcw, cluster1, cluster2){
100     # R-codes (www.r-project.org) for computing multi-way
101     # clustered-standard errors. Mahmood Arai, Jan 26, 2008.
102     # See: Thompson (2006), Cameron, Gelbach and Miller (2006)
103     # and Petersen (2006).
104     # reweighting the var-cov matrix for the within model
```

```
105
106     # The arguments of the function are:
107     # fitted model, cluster1 and cluster2
108     # You need to install libraries ‘sandwich’ and ‘lmtest’
109
110     library(sandwich);library(lmtest)
111     cluster12 = paste(cluster1,cluster2, sep="")
112     M1  <- length(unique(cluster1))
113     M2  <- length(unique(cluster2))
114     M12 <- length(unique(cluster12))
115     N   <- length(cluster1)
116     K   <- fm$rank
117     dfc1  <- (M1/(M1-1))*((N-1)/(N-K))
118     dfc2  <- (M2/(M2-1))*((N-1)/(N-K))
119     dfc12 <- (M12/(M12-1))*((N-1)/(N-K))
120     u1j   <- apply(estfun(fm), 2, function(x) tapply(x, cluster1,  sum))
121     u2j   <- apply(estfun(fm), 2, function(x) tapply(x, cluster2,  sum))
122     u12j  <- apply(estfun(fm), 2, function(x) tapply(x, cluster12, sum))
123     vc1   <-  dfc1*sandwich(fm, meat=crossprod(u1j)/N )
124     vc2   <-  dfc2*sandwich(fm, meat=crossprod(u2j)/N )
125     vc12  <- dfc12*sandwich(fm, meat=crossprod(u12j)/N)
126     vcovMCL <- (vc1 + vc2 - vc12)*dfcw
127     coeftest(fm, vcovMCL)}
128
129 ## Function to compute ols standard errors , robust, clustered...
130 ## Based on http://diffuseprior.wordpress.com/2012/06/15/standard-robust-and
        -clustered-standard-errors-computed-in-r/
131 ols.hetero <- function(form, data, robust=FALSE, cluster=NULL,digits=3){
132   r1 <- lm(form, data)
133   if(length(cluster)!=0){
134     data <- na.omit(data[,c(colnames(r1$model),cluster)])
135     r1 <- lm(form, data)
136   }
137   X <- model.matrix(r1)
138   n <- dim(X)[1]
139   k <- dim(X)[2]
140   if(robust==FALSE & length(cluster)==0){
141     se <- sqrt(diag(solve(crossprod(X)) * as.numeric(crossprod(resid(r1))/(n
          -k))))
142     res <- cbind(coef(r1),se)
143   }
144   if(robust==TRUE){
145     u <- matrix(resid(r1))
146     meat1 <- t(X) %*% diag(diag(crossprod(t(u)))) %*% X
147     dfc <- n/(n-k)
148     se <- sqrt(dfc*diag(solve(crossprod(X)) %*% meat1 %*% solve(crossprod(X)
          )))
149     res <- cbind(coef(r1),se)
150   }
151   if(length(cluster)!=0){
152     clus <- cbind(X,data[,cluster],resid(r1))
153     colnames(clus)[(dim(clus)[2]-1):dim(clus)[2]] <- c(cluster,"resid")
154     m <- dim(table(clus[,cluster]))
155     dfc <- (m/(m-1))*((n-1)/(n-k))
156     uclust  <- apply(resid(r1)*X,2, function(x) tapply(x, clus[,cluster],
          sum))
```

```
157     se <- sqrt(diag(solve(crossprod(X)) %*% (t(uclust) %*% uclust) %*% solve
            (crossprod(X)))*dfc)
158     res <- cbind(coef(r1),se)
159   }
160   res <- cbind(res,res[,1]/res[,2],(1-pnorm(abs(res[,1]/res[,2])))*2)
161   res1 <- matrix(as.numeric(sprintf(paste("%.",paste(digits,"f",sep=""),sep=
          ""),res)),nrow=dim(res)[1])
162   rownames(res1) <- rownames(res)
163   colnames(res1) <- c("Estimate","Std. Error","t value","Pr(>|t|)")
164   return(res1)
165 }
```

../util/are213–func.R