On-line Appendix: Generalized Correlation and Kernel Causality with Applications in Development Economics

Hrishikesh D. Vinod*

first version, July 14, 2015

Abstract

This document describes on-line supplementary material explaining the data and R software to accompany my paper Vinod (2015) appearing in *Communications in Statistics - Simulation and Computation*. The aim of this supplement is to assist the reader in extending and implementing the ideas in my paper. Included R code can be used as a template allowing one to use my methods on any data of interest to the reader. My software here is not claimed to be efficient, but does work. Explanations of software steps are often included, but not claimed to be complete. Hence some familiarity with R is needed to use it properly. The software tools and their descriptions provided here need additional work to bring them up to the standard of an official R package.

1 Introduction

This document is intended to be read in conjunction with my paper Vinod (2015) published in *Communications in Statistics - Simulation and Computation* (CSSC). This document is focused only on implementation and data

*Vinod: Professor of Economics, Fordham University, Bronx, New York, USA 10458. E-mail: vinod@fordham.edu. I thank Shurong Zheng for giving me the R code for GMC. A version of the original paper (of which this is a supplement) was a keynote address to the 50th annual (Golden Jubilee) meeting of the Indian Econometric Society in Mumbai, December 2013. I thank two referees for several detailed and helpful suggestions.

analysis, and will not repeat the explained theory from the paper. Most R code snippets are named for cross-reference purposes. The names always start with "R-".

There is considerable R code provided here which can be developed into an R package. At the minimum the code can serve as a template for specific graphics or computational applications with minor modifications. The code is seen in red font ready to be copied and pasted by the user. All commands of the type 'require()' or 'library()' will not work unless the particular R package named in parentheses is first loaded at the R installation where the code is used.

The blue font is reserved for outputs from the code. The outputs of many code snippets is suppressed and /or abridged for brevity.

2 R software Initialization

While many R experts do not agree with me, I prefer to replace the R prompt (>) implicit in the R software with one space and the R continuation symbol (+) with two spaces. This change permits direct copy and paste from this file into any user's R console without modifications, avoiding confusion with the greater than and plus symbols also used in R. I also prefer to use the one stroke (=) symbol for assignment rather than (<-) requiring 4 strokes after including spaces commonly used in R.

```
#R-Init
rm(list=ls())  #clean up the R memory
seed=42
set.seed(seed)
print(c("seed=",seed),quote=FALSE)
options(prompt = " ", continue = "  ", width = 68,
useFancyQuotes = FALSE)
print(date())
```

It is recommended that the reader copy and paste this code named "R-Init" before each new project. The next section describes our data sources before use in the subsequent sections.

3 Details Regarding the Nine Variables in Economic Development (EcDev) Data

Our data are publicly available at the UN, (See `http://hdr.undp.org/en/data`), the World Bank `http://search.worldbank.org/data`, etc. The supplementary material to this paper allows the reader to reproduce our results. Some data and country codes follow the Association of Religion Data Archives (ARDA) `www.TheARDA.com`, with the Principal Investigator Jaime Harris.

1. GRO: the least squares annual growth rate in Gross Domestic Product (GDP) used with constant GDP per capita data in local currency units. Sample growth rates are China=7.9, India=3.6, Mexico=1.7 and USA=1.9.

2. INHDI: Inequality-adjusted Human Development Index.

   This measure adjusts a country's HDI score based on its scores on three additional indexes: the inequality-adjusted life expectancy index, the inequality-adjusted education index and the inequality-adjusted income index. Sample scores are Albania=0.627, Bangladesh=0.321, India=0.365, USA =0.799 and Pakistan =0.336.

3. GINI: Income Gini coefficient. This is a measure of the deviation of the distribution of income (or consumption) among individuals or households within a country from a perfectly equal distribution. A Lorenz curve plots the cumulative percentages of total income received against the cumulative number of recipients, starting with the poorest individual or household. The Gini index measures the area between the Lorenz curve and a hypothetical line of absolute equality, expressed as a percentage of the maximum area under the line. A value of zero represents absolute equality, a value of 100 absolute inequality. Sample scores are Albania=33, Bangladesh=31, India=36.8, Iran=38.5 and USA=40.6.

4. GII: A country's score on the Gender Inequality Index (GII), a composite index measuring loss in achievements in three dimensions of human development: reproductive health (measured by the maternal mortality ratio and the adolescent fertility rate), empowerment (measured by the female and male population with at least a secondary education and by

the female and male shares of parliamentary seats) and the labor market due to inequality between genders (measured by female and male labor force participation rates). These scores are high in countries with large Muslim populations as: Egypt=0.714 and India=0.748 and low in Westernized countries as Japan=0.273 and The Netherlands= 0.174. The USA=0.400 is somewhat higher than expected.

5. MPI: Multidimensional Poverty Index. This indicates the share of the population that is multidimensionally poor adjusted by the intensity of the deprivations. These deprivation data are not reported for rich countries. The scores are Albania=0.004, Bangladesh=0.291, India=0.296 and Pakistan=0.275. India is among countries showing the worst deprivations for her poor.

6. GEI: Governance Effectiveness Index from the World Bank ranging between -2.5 to 2.5. For examples: Australia=1.74, India=(-0.03), Pakistan =(-0.82), UK=1.55, Japan=1.35. It reflects perceptions of the quality of public services, the quality of the civil service and the degree of its independence from political pressures, the quality of policy formulation and implementation, and the credibility of the government's commitment to such policies. The data are available for 1996–2011 for over 200 countries. We use the 2011 data.

7. Entp: Entrepreneurship index is what the World Bank researchers call "New Business Density". A high density indicates prevalence of entrepreneurship in a country, calculated as the number of newly registered limited liability companies per 1,000 working-age people (those ages 15-64), World Bank (2013). For examples: Australia=6.17, India=0.09, Pakistan=0.03, UK=8.32, Japan=1.1.

8. ECI: Economic Complexity Index developed at MIT and explained in Hidalgo and Hausman (2009). The authors claim that the greater the number of complex exchanges in international trade of goods undertaken (manufactured) in a country, the higher is its economic growth potential. For examples: Brazil=0.244, India=0.247, Pakistan =(-0.398), Philippines=0.032, UK=1.558, Japan=2.316, reflecting that Pakistan does not do much of complex manufacturing and Philippines does more than Pakistan.

9. AidPC: Per Capita net official development assistance (ODA) consists of disbursements of loans made on concessional terms (net of repayments of principal) and grants by official agencies of the members of the Development Assistance Committee (DAC), by multilateral institutions, and by non-DAC countries to promote economic development and welfare. One divides by the midyear population estimate. It includes loans with a grant element of at least 25 percent (calculated at a rate of discount of 10 percent). Data are available online at: `www.oecd.org/dac/stats/idsonline`. World Bank population estimates are used for the denominator. For examples: Pakistan=20, India=3, Iraq=60, Bhutan=197, Kenya=59.

4 R software for reading Economic Development (EcDev) Data

We use the country codes by ARDA available at `http://www.fordham.edu/economics/vinod/ARDAcountryCodes.xls`. The data are available in R format at: `http://www.fordham.edu/economics/vinod/DevEcData.Rdata`. If the Rdata file does not work an alternative version 'DevEc10.csv' a comma separated csv file is also available at the same Internet location. Its use is shown below.

The data for 250 countries contains lots of missing data (NAs). The code named "R-get-EcDev-data" loads entire data into user's R console for the chosen nine variables (abbreviations listed in the previous section) and country code.

```
#R-get-EcDev-data
UR="http://www.fordham.edu/economics/vinod/DevEc10.csv"
da=read.csv(file=UR, header=TRUE)
head(da,3)
tail(da,3)
attach(da)#enable access to names of variables
mtx=cbind(GRO, INHDI, GINI, GII, MPI, GEI, Entp, ECI, AidPC)
summary(mtx) #basic stats for 9 variables
```

The output from the above code allows one to briefly view the data by reporting the data for the first and last three countries only.

```
 head(da,3)
  CODE GRO INHDI GINI   GII   MPI   GEI Entp    ECI AidPC
1    1 1.9    NA   NA 0.797    NA -1.46 0.12     NA   231
2    3 2.2 0.627 33.0 0.545 0.004 -0.20   NA  0.087   111
3    4 1.1    NA 35.3 0.594    NA -0.66 0.19 -1.213     5
 tail(da,3)
    CODE  GRO INHDI GINI GII   MPI   GEI Entp   ECI AidPC
196  248 -0.7 0.656 28.2  NA 0.003 -0.15 1.96 0.644   190
197  249  0.0 0.693 36.9  NA 0.006  0.10   NA    NA   200
198  250   NA    NA   NA  NA    NA -0.49 0.70    NA   367
summary(mtx) #basic stats for 9 variables
      GRO             INHDI             GINI             GII
 Min.   :-3.000   Min.   :0.0980   Min.   :16.80   Min.   :0.1740
 1st Qu.: 0.600   1st Qu.:0.3110   1st Qu.:34.15   1st Qu.:0.3917
 Median : 1.600   Median :0.5085   Median :39.70   Median :0.5955
 Mean   : 1.658   Mean   :0.4992   Mean   :40.79   Mean   :0.5479
 3rd Qu.: 2.400   3rd Qu.:0.6627   3rd Qu.:47.23   3rd Qu.:0.6960
 Max.   :10.900   Max.   :0.8760   Max.   :74.30   Max.   :0.8530
 NA's   :16       NA's   :60       NA's   :54      NA's   :62
      MPI               GEI                Entp
 Min.   :0.00000   Min.   :-2.16000   Min.   : 0.000
 1st Qu.:0.01425   1st Qu.:-0.77000   1st Qu.: 0.525
 Median :0.11100   Median :-0.23000   Median : 1.300
 Mean   :0.18228   Mean   :-0.05192   Mean   : 3.024
 3rd Qu.:0.32300   3rd Qu.: 0.62750   3rd Qu.: 3.965
 Max.   :0.64200   Max.   : 2.25000   Max.   :27.670
 NA's   :96        NA's   :10         NA's   :100
      ECI
 Min.   :-1.90700
 1st Qu.:-0.66100
 Median :-0.09900
 Mean   :-0.00452
 3rd Qu.: 0.75900
 Max.   : 2.31600
 NA's   :77
```

The next task is to compute the usual correlation coefficients. I recommend using the package 'Hmisc' for this purpose. The output of the code

below named "R-plot-EcDev" consists of two figures. It is suppressed here for brevity. The 'corrgram' is printed in the journal article as "Figure 1: Corrogram for nine economic development variables."

```
#R-plot-EcDev
require(Hmisc)
mycor=rcorr(mtx)$r #$
require(corrplot)
corrplot.mixed(mycor, lower="number", upper="ellipse",
order="AOE") #AOE= angular order of eigenvectors
library(corrgram)
corrgram(mtx, order=TRUE,
main="Economic development correlation ellipses",
panel=panel.ellipse,
text.panel=panel.txt, diag.panel=panel.minmax)
```

5 Plotting Bivariate Density for EcDev data

Now we provide the R code for plotting Bivariate density seen as "Figure 2: Joint density between economic growth (GRO) and income in- equality (GINI)" in my CSSC paper.

One needs to do pair-wise deletion of missing data or NA's by using the code named "R-napair" providing my function called 'napair.'

The code named "R-napair-Dev-data" further below uses 'napair' function on EcDev data. It renames the pair-wise matched data as GRO2 and GINI2, while keeping the original GRO and GINI data intact, since the original data will have to be matched eventually with seven other variables.

```
#R-napair
napair=function(x,y){
#author:  H D Vinod, Fordham University, 2013
ava.x=which(!is.na(x))#ava means available
ava.y=which(!is.na(y))#ava means non-missing
ava.both=intersect(ava.x,ava.y)
list(newx=x[ava.both],#delete NAs from x
newy=y[ava.both])#delete NAs from y
}#end napair function
```

```
#R-napair-Dev-data
attach(da)
na1=napair(GRO,GINI)
GRO2=na1$newx; GINI2=na1$newy

#R-bivariate-density
library(sparr)#GET THIS PACKAGE first
cb2=cbind(GRO2,GINI2);xr1=min(GRO2);xr2=max(GRO2)
yr1=min(GINI2);yr2=max(GINI2)
bd=bivariate.density(data=cb2,xrange=c(xr1,xr2),
yrange=c(yr1,yr2), pilotH=1.4)
plot(bd, display="persp", phi = 30,
 theta = -30,  ticktype = "detailed", xlab="X", ylab="Y")
title("Bivariate density GRO and GINI")
```

The code named "R-bivariate-density" above does the actual plotting of the data after NAs are removed and series are matched.

6 Computing Generalized Correlation Matrix

The code named "R-gmcxy.np-function" below is my R function which in turn relies on the 'np' package to do kernel regressions in equations (1) and (2) in my CSSC paper. The function 'npregbw' suitably sets up the bandwidths.

The code named "R-gmcmtx0-function" provides my R function to compute the R^* (new asymmetric generalized correlation matrix) described in Section 2.5 and defined in equation (10) of my CSSC paper. The zero in the name 'gmcmtx0' is intended to distinguish it from a more CPU-time-consuming bootstrap version 'gmcmtx,' described later.

```
#R-gmcxy.np-function
require(np)
options(np.messages=FALSE)
gmcxy.np=function(x,y){
#np means we call the np library functions
bw=npregbw(formula=x~y,tol=0.1, ftol=0.1)
model=npreg(bws=bw, gradients=FALSE, residuals=TRUE)
corxy= model$R2
```

```
bw2=npregbw(formula=y~x,tol=0.1, ftol=0.1)
model2=npreg(bws=bw2, gradients=FALSE, residuals=TRUE)
coryx= model2$R2
list(corxy=corxy,coryx=coryx)}

#R-gmcmtx0-function
gmcmtx0=function(mym, nam=colnames(mym)){
# mym is a data matrix with n rows and p columns
# some NAs may be present in the matrix
p=NCOL(mym)
#print(c("p=",p))
out1=matrix(1,p,p)# out1 stores asymmetric correlations
for (i in 1:p){
x=mym[,i]
for (j in 1:p){
if (j>i){ y=mym[,j]
ava.x=which(!is.na(x))#ava means available
ava.y=which(!is.na(y))
ava.both=intersect(ava.x,ava.y)
newx=x[ava.both]
newy=y[ava.both]
c1=cor(newx,newy)
sig=sign(c1)
#begin non parametric regressions
bw=npregbw(formula=newx~newy,tol=0.1, ftol=0.1)
mod.1=npreg(bws=bw, gradients=FALSE, residuals=TRUE)
corxy= sqrt(mod.1$R2)*sig
out1[i,j]=corxy
bw2=npregbw(formula=newy~newx,tol=0.1, ftol=0.1)
mod.2=npreg(bws=bw2, gradients=FALSE, residuals=TRUE)
coryx= sqrt(mod.2$R2)*sig
out1[j,i]=coryx
}#end i loop
}#end j loop
}#endif
colnames(out1)=nam
rownames(out1)=nam
return(out1)}
```

Now we are ready to use the function 'gmcmtx0' on any data having T observations on p variables in columns, organized in the form of a $T \times p$ matrix.

Our first example is the European crime data from Section 3.3 of my CSSC paper.

```
#R-get-Crime-data
UR="http://www.fordham.edu/economics/vinod/PolicingSummary.csv"
da=read.csv(file=UR, header=TRUE)
summary(da)
attach(da)#total crime and police officers per 1000 population
pop=pop2008/10000
crim=as.numeric(crim2008)/as.numeric(pop)
off=as.numeric(Off2008)/as.numeric(pop)
na1=napair(crim,off)#remove missing data
crim=na1$newx; off=na1$newy

#R-call-gmcmtx0
gmcmtx0(cbind(crim,off))
```

```
 summary(da)
          Country      Off2008        crim2008     pop2008
 Austria       : 1   1,555  : 1   1,022,682: 1   Min.   :   35356
 Belgium       : 1   1,884  : 1   1,082,057: 1   1st Qu.: 2191810
 Bulgaria      : 1   10,743 : 1   1,112    : 1   Median : 8318592
 Cyprus        : 1   100,648: 1   1,266,165: 1   Mean   :19246569
 CzechRepublic: 1   11,018 : 1   1,377,854: 1   3rd Qu.:20635460
 Denmark       : 1   164,677: 1   104,758  : 1   Max.   :82217837
 (Other)       :23   (Other):23   (Other)  :23
  gmcmtx0(cbind(crim,off))
          crim       off
 crim 1.000000 0.9960115
 off  0.997196 1.0000000
```

Note that row variable X_i is the "effect" and the column variable X_j is the predictor or the "cause." The value of R^* in the column for crime is slightly larger than the value in the off column for police officers. It stands to reason that high crime causes larger police deployment.

Now consider the Internet advertising data of Section 3.4 of my CSSC paper. The code named "R-read-ad-R*" reads in the Internet advertising data and reports the R^* matrix computed by using the function 'gmcmtx0' given above.

```
#R-read-ad-R*
CTR=c( 0.0119, 0.0109,0.011, 0.0121,0.0096,0.0103,0.0105,0.0105,
0.0099,0.0122, 0.0108, 0.01, 0.0112,0.0097,0.0108,0.0114,0.0106,
0.0117,0.0112,0.0115, 0.0099, 0.0091,0.0107,0.0097,0.0127,0.0112,
0.0127,0.0101,0.0109,0.011, 0.0099, 0.0128,0.01,0.0109,0.011,
0.0109,0.0106,0.0098,0.0102,0.0106, 0.0117, 0.0099,0.011,0.0097,
0.0103,0.01, 0.0101,0.0091,0.0139,0.0103, 0.0132)
 CollegeGrad=c(0.19,  0.247, 0.235, 0.167, 0.266,
 0.327, 0.314, 0.25,  0.391, 0.223, 0.243,
 0.262, 0.217, 0.261, 0.194, 0.212, 0.258,
 0.171, 0.187, 0.229, 0.314, 0.332, 0.218,
 0.274, 0.169, 0.216, 0.244, 0.237, 0.182,
 0.287, 0.298, 0.235, 0.274, 0.225, 0.22,
 0.211, 0.203, 0.251, 0.224, 0.256, 0.204,
 0.215, 0.196, 0.232, 0.261, 0.294, 0.295,
 0.277, 0.148, 0.224, 0.219)
cor.test(CTR,CollegeGrad)
gmcmtx0(cbind(CTR,CollegeGrad))
```

The output of the command 'cor.test' is included first. It includes the t-test of the Pearson correlation coefficient, a confidence interval and p-value. It is followed by the estimated R^* matrix for the Internet advertising data.

```
        Pearson's product-moment correlation
data:  CTR and CollegeGrad
t = -5.6424, df = 49, p-value = 8.282e-07
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval: -0.7699822 -0.4255837
sample estimates:  cor = -0.6275642
                   CTR CollegeGrad
CTR          1.0000000  -0.7312316
CollegeGrad -0.7008299   1.0000000
```

Again, the row variable X_i is the "effect" and the column variable X_j is the "cause." The magnitude 0.73 exceeds 0.70 suggesting that college graduates are not easy customers of Internet advertising.

Consider the short data for 7 regions on housing bubble in Section 3.5 of my CSSC paper which can be manually entered.

```
GDPchang=c(18.6, 7.4, 7.4, 5.6, 5.1, 4.1, 3.4, 2.1, 2.0)
HomPrChg=c(14.6, 1.6, -14.1, 0.0,-5.4, -9.6,-4.4,-11.1, -4.6)
gmcmtx0(cbind(GDPchang, HomPrChg))
```

The output R^* matrix is given

```
          GDPchang HomPrChg
GDPchang 1.0000000 0.968136
HomPrChg 0.7838597 1.000000
```

Since the row variable X_i is the "effect" and the column variable X_j is the cause," HomPrChg has a stronger effect (0.968) on GDPchang than vice versa. This supports the notion that the housing bubbles during the particular time period may have hurt the growth in GDP.

Using the nine variables from 'EcDev' data in the matrix called 'mtx' above we can get the matrix of $r^*(X_i|X_j)$ defined by eq. (10) in the text of my CSSC paper by the following command. The generalized correlation matrix does not appear in Section 4 of the CSSC paper.

```
rs=gmcmtx0(mtx)
print(rs,digits=2)
```

The output given below is abridged to only two digits to fit the width of the page. The actual R^* matrix in the memory of R has more than eight digits than what is displayed here. Tables 5 and 6 of my CSSC paper displays additional digits for 36 unique pairs and explicitly identify the "cause" by name.

```
 print(rs,digits=2)
        GRO  INHDI   GINI   GII   MPI    GEI   Entp   ECI  AidPC
GRO    1.000  0.370 -0.061 -0.29 -0.28  0.380  0.317  0.40 -0.093
INHDI  0.385  1.000 -0.589 -0.94 -0.94  0.863  0.736  0.83 -0.511
GINI  -0.049 -0.566  1.000  0.68  0.62 -0.486  0.058 -0.53  0.134
```

```
GII   -0.307 -0.954  0.682  1.00  0.83 -0.864 -0.746 -0.86  0.593
MPI   -0.271 -0.956  0.289  0.82  1.00 -0.641 -0.613 -0.71  0.445
GEI    0.500  0.889 -0.436 -0.87 -0.62  1.000  0.733  0.81 -0.315
Entp   0.306  0.431  0.060 -0.46 -0.63  0.507  1.000  0.25  0.166
ECI    0.406  0.843 -0.557 -0.83 -0.70  0.811  0.514  1.00 -0.285
AidPC -0.074 -0.044  0.076  0.14  0.42 -0.044  0.919 -0.39  1.000
```

7 Software for Multivariate Tools for Ameliorating Misspecifications

This section provides software for Section 2.5 of my CSSC paper. It begins with the R^* matrix and computes its minors and cofactors, requiring R functions for those tasks. The minors of a matrix are given by my R function called 'minor' after removing row r and column c. The cofactors are signed minors and are given by the function called 'cofactor'. Both are given next.

```
#R-minor-function
minor=function(x,r,c){
 #x is n by p matrix we want its minor
 #after eliminating rth row and cth column
 n=nrow(x)
 p=ncol(x)
 myn=1:n
 myp=1:p
if (n<r) stop("n<r, minor undefined")
if (p<c) stop("p<c, minor undefined")
if (c<=0) stop("c<=0, minor undefined")
 newr=myn[-r]
 newc=myp[-c]
 out=x[newr,newc]
return(out)}

#R-cofactor-function
cofactor=function(x,r,c){
out=minor(x,r,c)*((-1)^(r+c))
return(out)}
```

Recall that eq. (16) of my CSSC paper defines the partial correlation coefficients for the R^* matrix. The function 'parcor.ijk' computes those partial correlation coefficients bet X_i and X_j after removing the effect of all X_k variables. The function reports a list of removed columns named 'myk'.

```
parcor.ijk=function(x,i,j){
 n=nrow(x)
 p=ncol(x)
 if (n<i) stop("n<i, parcor undefined")
if (p<j) stop("p<j, parcor undefined")
if (i<=0 | j<=0) stop("i OR j <=0, parcor undefined")
 myn=1:n
 myp=1:p
 myk=myp[c(-i,-j)]
 numij=det(cofactor(x,i,j))
 numji=det(cofactor(x,j,i))
deni=abs(det(cofactor(x,i,i)))
denj=abs(det(cofactor(x,j,j)))
ouij=(numij)/sqrt(deni*denj)
ouji=(numji)/sqrt(deni*denj)
list(ouij=ouij, ouji=ouji, myk=myk)}
```

Instead of using the general function 'parcor.ijk', it is convenient to use Cox-Vermuth function called 'partialc2' to get the betas or scale free partial regression coefficients described in my CSSC paper using the EcDev data.

```
#R-partialc2-function
partialc2= function(cor.matrix, digits = 3) {
# THE FORMULA IS FROM COX, D.R. & WERMUTH, N. 1993. LINEAR
# DEPENDENCIES REPRESENTED BY CHAIN GRAPHS. STATISTICAL SCIENCE
#8:204-283.
        a <- solve(cor.matrix)
        n <- length(a[1,  ])
        ans <- matrix(NA, n, n)
        for(i in 1:n) {
                for(j in 1:n) {
                 aiaj=a[i,i]*a[j,j]
                 myif=ifelse(aiaj>0,sqrt(aiaj),1)
                        ans[i, j] <- round((-1 *
```

```
                                    a[i, j])/myif,
                                    digits)}}
return(ans) }
```

The function 'getbeta' computes the betas or scale-free regression coefficients of two variables after removing the effect of additional variables. If the option 'rstar' is set FALSE, then Pearson correlations are used, which in tern require the function 'rcorr' of the package 'Hmisc'.

```
#R-getbeta-function
require(Hmisc)
getbeta=function(x,rstar=TRUE,verbo=TRUE,
nam=colnames(x))  {
#get beta coefficients from inverse of the corr matrix
#Input  x=data matrix  e.g. cbind(x,y,z)
p=NCOL(x)
if (p<3) stop(c("number of columns < 3",p))
if (rstar){
gm1=gmcmtx0(x)
if(verbo)print("r* matrix of generalized correlations")
if(verbo) print(gm1)
cr=gm1}
parco= partialc2(cr)
if (!rstar){
cr=rcorr(x)$r   #$
if(verbo)print("matrix of Pearson correlations")
if(verbo) print(cr)
parco= partialc2(cr)
}#end if !rstar
colnames(parco)=colnames(x)
rownames(parco)=colnames(x)
if(verbo)print("matrix of partial correlations")
if(verbo) print(parco)
# invert the correlation matrix
crinv=solve(cr)
if(verbo)print(" Inverse matrix ")
if(verbo) print(crinv)
pc1=partialc2(cr,digits=5)
```

```
pbeta=crinv # place to store
for (i in 1:p){
#diagonals of the output matrix has
# Rsquare when Xi is regressed on all others
#off diagonals have regression beta coefficients
for (j in 1:p){
ratio= (crinv[j,j]/crinv[i,i])
if(ratio<=0) print(c("neg",j,i,round(ratio,4)),quote=FALSE)
#myif=ifelse(ratio>0,sqrt(ratio),sqrt(abs(ratio)))
myif=ifelse(ratio>0,sqrt(ratio),NA)
pbeta[i,j]=pc1[i,j]*myif
} #end j loop
pbeta[i,i]= 1-(1/crinv[i,i])
} #end i loop
print("p,length(nam),ncol(pbeta),nrow(pbeta)")
print(c(p,length(nam),ncol(pbeta),nrow(pbeta)))
colnames(pbeta)=nam
rownames(pbeta)=nam
if(verbo)print("matrix of partial R-sq on diag; betas off-diag")
if(verbo) print(pbeta)
list(cr=cr, parco=parco, pbeta=pbeta)
}#end function
```

Now we turn to recreating Table 7 of the CSSC paper. It is conveniently created by using the additional functions 'partialc2' given above and 'getParcor' given below.

```
#R-getParcor-function
#  NEEDS  gmcmtx0' and partialc2' in memory
require(Hmisc) #rcorr does pairwise deletion
#otherwise corr matrix will have NAs Nan etc problems
getParcor=function(x,rstar=TRUE,verbo=TRUE,
nam=colnames(x))  {
#get partial coeff from inverse of the correlation matrix
#Input  x=data matrix  e.g. cbind(x,y,z)
p=ncol(x)
if (p<3) stop("number of columns < 3")
if (rstar){
gm1=gmcmtx0(x)
```

```
if(verbo)print("r* matrix of generalized correlations")
if(verbo) print(gm1)
cr=gm1}#end if rstar
if (!rstar){
#require(Hmisc) #rcorr does pairwise deletion
cr=rcorr(x)$r  #$
if(verbo)print("matrix of Pearson correlations")
if(verbo) print(cr)
}#end if !rstar
#  INVERT correlation matrix
crinv=solve(cr)
if(verbo)print(" Inverse matrix ")
if(verbo) print(crinv)
pc1=partialc2(cr,digits=5)
par.cor=crinv # place to store partials
for (i in 1:p){
#diagonals of the output matrix has
# Rsquare when Xi is regressed on all others
#off diagonals have regression beta coefficients
for (j in 1:p){
par.cor[i,j]=pc1[i,j]
} #end j loop
par.cor[i,i]= 1-(1/crinv[i,i])
} #end i loop
print("p,length(nam),ncol(par.cor),nrow(par.cor)")
print(c(p,length(nam),ncol(par.cor),nrow(par.cor)))
colnames(par.cor)=nam
rownames(par.cor)=nam
if(verbo)print("partial R-sq on diag and partial correl off-diag")
if(verbo) print(par.cor)
#cr has R correlations or R* if rstar=TRUE
list(cr=cr, par.cor=par.cor)
}#end function getParcor
```

Now we are ready to apply the above tools to Economic Development data to create Table 7 of my CSSC paper illustrating the use of partial correlations and betas in multivariate situations. Table 7 has two vertical panels with the left panel for the three variables my3=(MPI, GRO, GINI)

and the right panel is for four variables my4=(MPI, GRO, GINI, GEI). Recall that the 'getbeta' function offers a choice of Pearson correlation by using the command 'g1=getbeta(my3,rstar=FALSE)' illustrated below. Of course, we are more interested in the new R^* and betas obtained from it obtained by the command: 'g2=getbeta(my3,rstar=TRUE)'. The four rows of the left panel on Table 7 of the CSSC paper are in the matrix 'myx'.

```
#R-left-panel-Table-7
my3=cbind(MPI, GRO, GINI)
g1=getbeta(my3,rstar=FALSE)
xg1=g1$pbeta[1,];xg1
g2=getbeta(my3,rstar=TRUE)
xg2=g2$pbeta[1,];xg2
g3=getParcor(my3,rstar=FALSE)
xg3=g3$par.cor[1,];xg3
g4=getParcor(my3,rstar=TRUE)
xg4=g4$par.cor[1,];xg4
myx=rbind(xg1,xg2,xg3,xg4);myx

#R-right-panel-Table-7 with 4 variables
my4=cbind(MPI, GRO, GINI, GEI)
g1=getbeta(my4,rstar=FALSE)
yg1=g1$pbeta[1,];yg1
g2=getbeta(my4,rstar=TRUE)
yg2=g2$pbeta[1,];yg2
g3=getParcor(my4,rstar=FALSE)
yg3=g3$par.cor[1,];yg3
g4=getParcor(my4,rstar=TRUE)
yg4=g4$par.cor[1,];yg4
myy=rbind(yg1,yg2,yg3,yg4);myy
ou1=cbind(myx[,2:3],myy[,2:3]);ou1
d1=abs(ou1[,1])-abs(ou1[,2]);d1
d2=abs(ou1[,3])-abs(ou1[,4]);d2
ou2=cbind(ou1[,1:2],d1,ou1[,3:4],d2)
require(xtable)
xtable(ou2,digits=4)
```

The output from the above code creates Table 7 in my CSSC paper, where the discussion of evidence supports Bhagwati's growth policies over Sen's

redistribution policies for economic development. It allows consideration of equation (22) (regressing MPI on GRO and GINI) which has two regressors illustrating multivariate regressions. By using partial correlations we can focus on the effect of one regressor at a time, removing the effect of remaining regressors. The right panel of Table 7 considers regression of MPI on GRO, GINI and GEI (i.e., three regressors). The partial correlations in the right panel remove the effect of two regressors.

8 Non-spherical Error Corrections

Correcting for autocorrelation and heteroscedasticity (non-spherical errors) is readilty done by using the following function called 'autohetero', which in turn needs a function 'sort.matrix' which is given first. The theory is described in Vinod (2010). The basic idea is to use generalized least squares.

```
#R-sort.matrix-function
sort.matrix =function(x,j){
y=x[sort.list(x[,j]),]
return(y)  }

#R-autohetero-function
autohetero = function (y, bigx){
#construct omega^ matrix with AR(1) and hetero with linear regr
#of sorted u^squared on time forcing thru origin
reg1=lm(y~bigx)
#print("OLS estimation")
#print(summary(reg1))
uhat=resid(reg1)
#print(acf(uhat,plot=F))
bigt= length(uhat)
bigt
tim=1:bigt
u2=cbind(tim, uhat^2)
soru2=sort.matrix(u2,2)
su2=soru2[,2] #second col has sorted u^2
# plot(tim,su2)
#first choice
```

```
#initialize
minf1=-9991
minf2=-9992
minf3=-9993
minf4=-9994
regt=lm(su2~tim+I(tim^2)) # quadratic case intercept present
fitu2=fitted(regt)
minf1=min(fitu2)
#if(minf1>=0) noquote(c("min fitted s^2 1+tim+tim^2",minf1))
if (minf1<0){
#second choice
regt=lm(su2~tim+I(tim^2)-1) # quadratic case NO intercept
fitu2=fitted(regt)
minf2=min(fitu2)
#if(minf2>=0) noquote(c("min fitted s^2~0+tim+tim^2",minf2))
}
if (minf1<0 && minf2<0) {
#third choice
regt=lm(su2~tim) # intercept present
fitu2=fitted(regt)
minf3=min(fitu2)
#if(minf3>=0) print(c("min fitted s^2 1+tim",minf3),q=F)
}

if (minf1<0 && minf2<0 && minf3<0) {
#4th choice
regt=lm(su2~tim-1) #force thru origin so min fitted value>0
fitu2=fitted(regt)
minf4=min(fitu2)
#if (minf4<=0) print("Error: heteroscedasticity regt fails")
}
#print(regt)
newu2=sort.matrix(cbind(soru2[,1],fitu2),1)
num=newu2[,2]
#  # # #
noquote("debug num diagonals")
#print(num)
```

```
#Now autocorrelaton correction
ac=acf(uhat,plot=F)
rho=ac$acf[2]  #$
if (bigt >200){
A=zapsmall(ARMAacf(ar=rho, ma=0, lag.max=bigt-1))}
#if lag.max=100,one gets 101x101 matrix since begin lag=0
A=ARMAacf(ar=rho, ma=0, lag.max=bigt-1)
omeg=toeplitz(A)
#print(omeg[1:3,1:3])
#NOW GLS
ei=eigen(omeg)
#print(ei)
bigv =ei$vec %*% diag(sqrt(1/ei$val)) %*% t(ei$vec)
#print(bigv[1:3,1:3])  #$
vy=bigv%*%y
vx=bigv %*%bigx
regg=lm(vy~vx)
#print("GLS estimation")
#print(summary(regg))
list(vy=vy,vx=vx)}
```

9 Sampling distribution of $\hat{\delta}$ using meboot

The maximum entropy bootstrap is implemented in the R package 'meboot'. It creates a large number (n999=999) of resamples of 'x' (similar data series) by the command 'xboot=meboot(x=x, reps=n999)$ensemble'. Now xboot has 999 columns of data. The following code needs 'gmcxy.np' from Section 6.

The function 'pcause' computes P(cause) defined in eq. (9) of my CSSC paper giving the larger of the two rejection probabilities. The computed P(cause) lies in the range [0,1], where a larger value is more desirable for inference about the causal direction based on $\hat{\delta}$ using the meboot bootstrap.

```
#R-pcause-function from eq.(9) of my CSSC paper
require(meboot)
#Bring into R the function gmcxy.np from earlier section
pcause=function(x,y,n999=999){
```

```
#now find P(cause) using meboot
xboot=meboot(x=x, reps=n999)$ensemble
xb=x
yboot=meboot(x=y, reps=n999)$ensemble
yb=y
out.diff=rep(NA,n999)
out.corxy=rep(NA,n999)
out.coryx=rep(NA,n999)
for (i in 1:n999){
xb=xboot[,i]
yb=yboot[,i]
gm=gmcxy.np(xb,yb)
out.corxy[i]=gm$corxy
out.coryx[i]= gm$coryx
out.diff[i]=out.corxy[i]-out.coryx[i]
} #end of i loop
ou.nega=length(out.diff[out.diff<0])
ou.posi= length(out.diff[out.diff>0])
p.cause=max(ou.nega, ou.posi)/n999
list(out.corxy=out.corxy,out.coryx= out.coryx,
out.diff=out.diff,p.cause=p.cause)}
```

Note that 'out.diff' contains 999 estimates of $\hat{\delta}$ allowing us to approximate its sampling distribution.

10 Comprehensive Tables Displaying 36 Rows for 9 Variables

Note that our kernel causality criterion relies on the asymmetry of the generalized correlation matrix $r^*(i,j)$ to identify the causal variable. Since the row variable X_i is the "effect" and the column variable X_j is the "cause," we identify the cause depending on which matrix entry (i,j) or (j,i) is larger. EcDev data have a large 9×9 matrix and it is not convenient to manually compare such entries.

It is safer to let the computer choose all unique (i,j) pairs and explicitly identify the cause by comparing the indicated entries of the asymmetric matrix. Since $({}^9C_2 = 36)$, our EcDev data needs a comparison of 36 pairs.

Tables 4 to 6 of my CSSC paper do have 36 rows for each unique pair and the cause is identified and named in the third column by using a computer program. A typical program is given below.

If the bootstrap is used for inference, a much slower version of gmcmtx0 called gmcmtx needs to be used before we do the 36 pairs. Hence we begin with that code.

```
#Bring into R the function gmcxy.np from earlier section

#R-pcause-function version returning pcause only
pcause=function(x,y,n999=999){
#now find P(cause) using meboot
if (n999<=1){
p.cause=NA
return(p.cause)}
else {
xboot=meboot(x=x, reps=n999)$ensemble
xb=x
yboot=meboot(x=y, reps=n999)$ensemble
yb=y
out.diff=rep(NA,n999)
out.corxy=rep(NA,n999)
out.coryx=rep(NA,n999)
for (i in 1:n999){
xb=xboot[,i]
yb=yboot[,i]
gm=gmcxy.np(xb,yb)
out.corxy[i]=gm$corxy
out.coryx[i]= gm$coryx
out.diff[i]=out.corxy[i]-out.coryx[i]
} #end of i loop
ou.nega=length(out.diff[out.diff<0])
ou.posi= length(out.diff[out.diff>0])
p.cause=max(ou.nega, ou.posi)/n999
}#end of else
return(p.cause)}

#R-gmcmtx-function a slow CPU intensive version of gmcmtx0
```

```
gmcmtx=function(mym, n999=999){
# mym is a matrix with n rows and p columns
# some NAs may be present in the matrix
p=NCOL(mym);
#print(c("p=",p))
out1=matrix(1,p,p)# out1 has asymmetric correlations
out2=matrix(NA,p,p)#out2 diag has lengths of non-missing data
#out2 subdiag has usual p values
#out2 superdiag has P(cause) values
out3=matrix(NA,p,p)# super diag has non-missing data pairs
#out3  sub-diagonal has data lengths
#out3 super-diag has simple correlation coeff.
for (i in 1:p){
x=mym[,i]
for (j in 1:p){
if (j>i){ y=mym[,j]
ava.x=which(!is.na(x))#ava means available
ava.y=which(!is.na(y))#think of ava as non-missing
ava.both=intersect(ava.x,ava.y)
newx=x[ava.both]
newy=y[ava.both]
out2[i,i]=length(ava.x)
out3[i,i]=length(ava.x)
out2[j,j]=length(ava.y)
out3[j,j]=length(ava.y)
out3[i,j]=length(ava.both)
c1=cor.test(newx,newy)
sig=sign(c1$estimate)
out2[i,j]=c1$p.value #subdiagonal
out3[j,i]= c1$estimate
#begin non parametric kernel regressions $
bw=npregbw(formula=newx~newy,tol=0.1, ftol=0.1)
mod.1=npreg(bws=bw, gradients=FALSE, residuals=TRUE)
corxy= sqrt(mod.1$R2)*sig
out1[i,j]=corxy
bw2=npregbw(formula=newy~newx,tol=0.1, ftol=0.1)
mod.2=npreg(bws=bw2, gradients=FALSE, residuals=TRUE)
coryx= sqrt(mod.2$R2)*sig
```

```
out1[j,i]=coryx
#now find P(cause) using meboot
out2[j,i]=pcause(newx,newy,n999=999)
}#end i loop
}#end j loop
}#endif
list(out1=out1, out2=out2, out3=out3)}
#end function gmcmtx a bootstrap version of gmcmtx0
```

The function 'cause36' computes the appropriate number of unique data pairs and reports the results. It needs the above code for 'gmcmtx'.

```
#R-cause36-function
require(meboot)
require(np)
options(np.messages=FALSE)
##NEEDS gmcmtx in memory
cause36=function(mtx,n999=999,dig=6){
#input matrix of data with p columns with colnames
#input dig=digits for rounding
#input n999 is the number of bootstrap replications
#output column1:2  names,  col3=cause,  col4=correlation
#outpout col 5= P(cause)
#output col 6=p-value,  col.7=generalized corij
#output col 8 =generalized corji
#output col 9 = name of the effect variable
n=NROW(mtx)
p=NCOL(mtx)
nam=colnames(mtx)
gm1=gmcmtx(mtx, n999=n999)
#print(gm1)  #print gmcmtx?
out1=gm1$out1
out2=gm1$out2
#out2 subdiag [i,j]has usual p values
#out2 super-diag [j,i] has P(cause) values
out3=gm1$out3
#out3  sub-diagonal [i,j] has data lengths #$
#out3 super-diag has simple correlation coeff.
n36=( p*(p+1)/2)-p
```

```
outcause=matrix(NA, nrow=n36,ncol=9)
ii=0
#following loop is such that j<=i, which means
#[i,j] will have i>j or sub-diagonal
for (i in 1:p){
for (j in 1:i){
if (i != j){
ii=ii+1
outcause[ii,1]=nam[i]
outcause[ii,2]=nam[j]
#now identify which is the cause in column 3
if (!is.na(out1[i,j]^2>out1[j,i]^2)){
if(out1[i,j]^2>out1[j,i]^2){
outcause[ii,3]=nam[j]
outcause[ii,9]=nam[i] #name of the effect variable
}}
if (!is.na(out1[i,j]^2<out1[j,i]^2)){
if(out1[i,j]^2<out1[j,i]^2){
outcause[ii,3]=nam[i]
outcause[ii,9]=nam[j]  #name of the effect variable
}}
#NOW simple correlations in column 4
outcause[ii,4]=round(out3[j,i],dig)
#outpout col 5= P(cause)
outcause[ii,5]= round(out2[j,i],dig)
outcause[ii,6]= round(out2[i,j],dig)
#output col 6=p-value,
# output col.7=generalized corij
#output col 8 =generalized corji
outcause[ii,7]= round(out1[i,j],dig)
outcause[ii,8]= round(out1[j,i],dig)
#ninth column has effect variable
} #end of if
}} #end of i and j loops
list(out1=out1, out2=out2, out3=out3,outcause=outcause)
} #end of function
```

The above code can be modified for various comparisons in Tables 5 and 6

of the CSSC paper as shown in Section 12.

11 Bar Chart of the Frequency of Cause Designations for Nine Variables

The 'table' command of R computes the frequencies and its result can be used for a bar plot using the following code named 'R-barchart-EcDev.'

```
#R-barchart-EcDev data, NEED cause36 in memory
require(Hmisc);mycor=rcorr(mtx)$r  #$
require(corrplot)
#c3=corrplot.mixed(mycor, lower="number", upper="ellipse",
order="AOE")
mtx2=mtx[,corrMatOrder(mycor,order="AOE")];head(mtx2)
#reorder data matrix by angular order of eigenvalues
c1=cause36(mtx2,n999=999)
tab1=table(c1$outcause[,3])#column 3 has names of causes #$
names(tab1)
require(lattice)
tab1b=sort(tab1)
barchart(tab1b,main="frequency of cause designations")
```

12 Code for Kernel Regression Robustness Checks

Recall that if the model regressing Y on X is more robust than the model regression X on Y we conclude that X kernel causes Y. Section 2.3 of my CSSC paper describes Kernel Regression Robustness Checks where the first robustness check uses a nonparametric test of dependence between residuals and regressors, where low dependence is desirable.

Table 5 of CSSC paper illustrates the results of this robustness check for EcDev data. The version of gmcmtx0 in the code named 'R-gmcmtx0npdep-function' is called 'gmcmtx0npdep.' The 0 in the name refers to fast version (without the meboot) and 'npdep' is the name of the function from the package 'np' used here to assess the dependence of regression residuals with

the regressor. The dependence is measured by S_ρ explained in Section 2.3 of my CSSC paper. Note that it is desirable to choose the model with the "smaller" value of S_ρ.

```
#R-gmcmtx0npdep-function
gmcmtx0npdep=function(mym, nam=colnames(mym)){
#author: H.D.Vinod, Prof of Economics, Forhdam Univ. NY Sept2013
# mym is a data matrix with n rows and p columns
# some NAs may be present in the matrix
p=NCOL(mym)
#print(c("p=",p))
out1=matrix(1,p,p)# out1 stores asymmetric correlations
out2=matrix(1,p,p)# out1 stores asymmetric corr wrt residuals
for (i in 1:p){
x=mym[,i]
for (j in 1:p){
if (j>i){ y=mym[,j]
ava.x=which(!is.na(x))#ava means available
ava.y=which(!is.na(y))
ava.both=intersect(ava.x,ava.y)
newx=x[ava.both]
newy=y[ava.both]
c1=cor(newx,newy)
sig=sign(c1)

#begin non parametric regressions
bw=npregbw(formula=newx~newy,tol=0.1, ftol=0.1)
mod.1=npreg(bws=bw, gradients=FALSE, residuals=TRUE)
corxy= sqrt(mod.1$R2)*sig
res1=mod.1$resid
#if(i==1)print(newy)
#if(i==1)print(res1)
cor.resy=npdeptest(newy,res1, method="summation",bootstrap=FALSE)
#if(i==1)print(cor.resy$Srho)
out1[i,j]=corxy
out2[i,j]=cor.resy$Srho
bw2=npregbw(formula=newy~newx,tol=0.1, ftol=0.1)
mod.2=npreg(bws=bw2, gradients=FALSE, residuals=TRUE)
```

```
coryx= sqrt(mod.2$R2)*sig
res2=mod.2$resid
cor.resx=npdeptest(newx,res2, method="summation",bootstrap=FALSE)
#if(i==1)print(cor.resx$Srho)
out1[j,i]=coryx
out2[j,i]=cor.resx$Srho
}#end i loop
}#end j loop
}#endif
colnames(out1)=nam
rownames(out1)=nam
colnames(out2)=nam
rownames(out2)=nam
list(out1=out1,out2=out2)}
```

The command 'gm1=gmcmtx(mtx,n999=n999)' in the above code for the function 'cause36' should be replaced by 'gm1=gmcmtx0npdep(mtx)' to create a Table similar to Table 5 in my CSSC paper.

Section 2.3 of the CSSC paper also describes another robustness check based on out-of-sample forecasts. The R code for that purpose is given next.

```
# Using out of sample 10% root mean square and scaled
require(np)
options(np.messages=FALSE)
require(akima)  #load this package
#R-stdze-function to standardize data matrix all columns
stdze=function(mtx){
stdx=function(x) (x-mean(x,na.rm=TRUE))/sd(x, na.rm=TRUE)
out=apply(mtx, 2, stdx)
return(out)}
```

The above standardization makes the mean zero and standard deviation unity for all columns. The following function computes root mean squared (rms) values defined in Section 2.3 of the CSSC paper. It uses the standardization function and then sorts on the first column, while carrying along all other columns.

```
#R-getrms-function computes root mean squared of errors
getrms=function(x,y,perc=10,reps=10){
```

```
# perc is percent data excluded for out of sample forecasting
#we sort on one variable and leave out perc=10% randomly chosen
oldmtx=stdze(cbind(x,y))
newmtx= oldmtx[sort.list(oldmtx[,1]),] #sorts on first column or x
n=length(x);n10=floor(n*perc/100);n90=n-n10
print("n,n90,n10,perc,reps")
print(c(n,n90,n10,perc,reps))
newx=newmtx[,1];newy=newmtx[,2]
outrms=rep(NA,reps)
for (ir in 1:reps){
n1=sort(sample(1:n)[1:n90])
n2= (1:n)[-n1]#define complementary set of numbers
x90=newx[n1];y90=newy[n1]#choose the sorted
x10=newx[n2];y10=newy[n2]#choose the complementary subset
bw=npregbw(formula=x90~y90,tol=0.1, ftol=0.1)#define bandwidth
mod.1=npreg(bws=bw, gradients=FALSE, residuals=TRUE)
#Now use x90 and fitted(y90) to estimate akima interpolation
#of y10 from x10 data
#where the correct y values are known.
ak1=aspline(x90, fitted(mod.1),xout=x10,method="improved")
#now compare interpolated with true y10
#err=(y10-ak1$y)/sd(y90) rescale not needed for standardized data
err=(y10-ak1$y)/sd(y90)
#find root mean squared error sqrt(sum(err^2))
outrms[ir]= sqrt(sum(err^2)/n10)
}#end loop for reps
rms=mean(outrms)
return(rms)}
```

Next, we provide code for implementing robustness in terms of out-of-sample forecasts using a function similar to gmcmtx called gmcmtx0out3samp. The zero in its name refers to fast version without meboot. out3samp in its name refers to a version of out-of-sample forecasting which reports asymmetric root mean squared (rms) values for comparison of the two models.

```
#R-gmcmtx0out3samp-function 0=no boot,
# Using out of sample 10% root mean square and scaled
#version 3 standardizes data for forecasting out of sample
```

```
gmcmtx0out3samp=function(mym, perc,reps,nam=colnames(mym)){
# input mym = matrix with n rows and p columns
# input perc=percent data excluded as out-of-sample
#we sort on one variable and leave out perc=10% randomly chosen
p=NCOL(mym)
#print(c("p=",p))
out1=matrix(1,p,p)# out1 stores asymmetric correlations
out2=matrix(1,p,p)# out2 stores asymmetric rms values
for (i in 1:p){
x=mym[,i]
for (j in 1:p){
if (j>i){ y=mym[,j]
ava.x=which(!is.na(x))#ava means available
ava.y=which(!is.na(y))
ava.both=intersect(ava.x,ava.y)
newx=x[ava.both]
newy=y[ava.both]
c1=cor(newx,newy)#cor.test gives p-values etc.
sig=sign(c1)

#begin non parametric regressions
bw=npregbw(formula=newx~newy,tol=0.1, ftol=0.1)
mod.1=npreg(bws=bw, gradients=FALSE, residuals=TRUE)
corxy= sqrt(mod.1$R2)*sig
cor.resy=getrms(newx,newy,perc=perc,reps=reps)
#if(i==1)print(cor.resy$Srho)
out1[i,j]=corxy
out2[i,j]=cor.resy
bw2=npregbw(formula=newy~newx,tol=0.1, ftol=0.1)
mod.2=npreg(bws=bw2, gradients=FALSE, residuals=TRUE)
coryx= sqrt(mod.2$R2)*sig

#use getrms function defined above here #$
cor.resx=getrms(newy,newx,perc=perc,reps=reps)
#if(i==1)print(cor.resx)
out1[j,i]=coryx
out2[j,i]=cor.resx
}#end i loop
```

```
print(c(nam[i],nam[j]))
}#end j loop
}#endif
colnames(out1)=nam
rownames(out1)=nam
colnames(out2)=nam
rownames(out2)=nam
list(out1=out1,out2=out2)}
```

Now we apply the above robustness check to EcDev data having 9 columns and 36 unique pairs. This is done by using a revision of the function cause36 called 'cause036outsamp', which in turn calls the function 'gmcmtx0out3samp' above.

```
#R-cause36outsamp-function  results for n C p pairs
cause036outsamp=function(mtx,dig=6,perc=10,reps=20){
#author:H.D.Vinod,Prof of Economics, Forhdam Uni.NY Sept2013
#input matrix of data with p columns with colnames
#input dig=digits for rounding
print("out of sample prediction using root mean square")
print("Choose the cause where rms is smaller")
n=NROW(mtx)
p=NCOL(mtx)
print(c("n,p,digits",n,p,dig))
nam=colnames(mtx)
gm1=gmcmtx0out3samp(mtx,perc=perc,reps=reps)
print(gm1)
out1=gm1$out1
out2=gm1$out2
n36=( p*(p+1)/2)-p
outcause=matrix(NA, nrow=n36,ncol=8)
ii=0
#following loop is such that j<=i, which means
#[i,j] will have i>j or sub-diagonal
for (i in 1:p){
for (j in 1:i){
if (i != j){
ii=ii+1
```

```
outcause[ii,1]=nam[i]
outcause[ii,2]=nam[j]
#now identify which is the cause in column 3
if (!is.na(out1[i,j]^2>out1[j,i]^2)){
if(out1[i,j]^2>out1[j,i]^2){
outcause[ii,3]=nam[j]}}
if (!is.na(out1[i,j]^2<out1[j,i]^2)){
if(out1[i,j]^2<out1[j,i]^2){
outcause[ii,3]=nam[i]}}
#output col 4,5  have generalized corji
outcause[ii,4]= round(out1[i,j],dig)
outcause[ii,5]= round(out1[j,i],dig)

#Name rms room mean square based cause in column 6
if (!is.na(out2[i,j]>out2[j,i])){
if(out2[i,j]>out2[j,i]){
outcause[ii,6]=nam[i]}}
if (!is.na(out2[i,j]<out2[j,i])){
if(out2[i,j]<out2[j,i]){
outcause[ii,6]=nam[j]}}

#outpout col 7 and 8 have rms values
outcause[ii,7]= round(out2[i,j],dig)
outcause[ii,8]= round(out2[j,i],dig)
} #end of if
}} #end of i and j loops
namo=c("X","Y","Cause","r*xy","r*yx","cauRms","rmsxy","rmsyx")
colnames(outcause)=namo
return(outcause)
} #end of function

#R-out-of-sample rms for EcDev data.
require(xtable)
head(da,3)
mtx=da[,2:10]
outcause=cause036outsamp(mtx)
print(xtable(outcause))
```

13 Final Remarks

The discussion in this supplement is kept at a practical level to facilitate further enhancements, which can be both theoretical and practical. I hope that readers find the supplementary material containing R software useful. Suggestions to streamline, improve and convert the code into an R package are welcome, and may be sent by e-mail to `vinod@fordham.edu`.

References

Hidalgo, C. and Hausman, R. (2009), "The building blocks of economic complexity," Tech. Rep. 26, Proceedings of the National Academy of Sciences, URL `http://www.pnas.org/cgi/doi/10.1073/pnas.0900943106`.

Vinod, H. D. (2010), "Superior Estimation and Inference Avoiding Heteroscedasticity and Flawed Pivots: R-example of Inflation Unemployment Trade-Off," in "Advances in Social Science Research Using R," , ed. Vinod, H. D., New York: Springer, pp. 39–63.

— (2015), "Generalized Correlation and Kernel Causality with Applications in Development Economics," *Communications in Statistics - Simulation and Computation*, accepted Nov. 10, 2015, URL `Âăhttp://dx.doi.org/10.1080/03610918.2015.1122048`.

World Bank (2013), "Working for World Free of Poverty," Tech. rep., IBRD, Washington DC, URL `http://iresearch.worldbank.org/PovcalNet/index.htm?1`.