A Gauss Procedure to Estimate Panel-Corrected-Standard-Errors with Non-rectangular and/or Missing Data [1]

Robert J. Franzese, Jr.
Assistant Professor of Political Science, and
Faculty Associate, Center for Political Studies, Institute for Social Research,
The University of Michigan, Ann Arbor

Econometric analysis of time-series-cross-section (TSCS) data often presents the researcher with residuals which exhibit contemporaneous correlation (i.e., residuals from different cross-sections (CS's) in the same time period (TP) are correlated) in addition to the more usual time-series property of serial (or auto-) correlation and the typical cross-section property of heteroskedasticity. This non-sphericity in the error-covariance matrix implies that estimated coefficients are inefficient and that standard errors are inconsistent. The usual solution, prior to Beck and Katz (1995) was to estimate Feasible Generalized Least Squares in up to three steps, estimating a serial-correlation parameter for each cross-section, a error-variance parameter for each cross-section, and a number of contemporary correlation parameters. Asymptotically (in T), this solution is unbiased, consistent, and efficient in both its coefficient and standard-error estimates.

Beck and Katz (1995) show, however, that in more realistically sized data-sets the procedure tends to under-estimate standard errors seriously. In essence, the problem is that there are a large number of parameters in the error-covariance matrix to be estimated and relatively few observations are being leveraged to estimate them. While this is perfectly fine in that mythical *asymptopia*,[2] in practical samples it has a tendency to overstate one's certainty about coefficient estimates. That is, it tends to over-fit the data, or more precisely to over-fit patterns in the residuals, relative to what is truly optimal in a random sample of samples.

They suggest, therefore, that one deal with the serial-correlation property separately, transforming the data and/or including lag(s) prior to addressing the contemporaneous correlation.[3] Panel heteroskedasticity, perhaps, should be ignored; the implications of their findings regarding such non-constant variance by CS depend on the sample size and the researcher's distaste for inefficiency relative to standard-error inaccuracy. In any event, it may be treated in standard WLS or robust-standard-error ways, and, while they suggest the latter is usually safer for smaller data sets, the issue is separate from what should be done about contemporaneous correlation. These findings are treated in more detail in the *Political Analysis* (forthcoming). Finally, and most importantly for the present purposes, they show (*via* Monte Carlo experiments) that the efficiency gains from contemporaneous-correlation correction are typically small while standard-error bias is relatively large. Accordingly, they suggest estimating standard errors according to a robust procedure which incorporates potential contemporaneous-correlation information into the coefficient variance-covariance matrix without

[1] This is appendix MA4 of *The Political Economy of Over-Commitment* and is only a minor rewriting of my notice in *The Political Methodologist* (1996). Gratitude is due to Neal Beck for providing the Beck-Katz procedures, for other suggestions, and for general helpfulness. This piece has also benefitted from the helpful criticism of James Alt, Gary King, Kenneth Scheve, and Curtis Signorino. Special thanks to Curt for his assistance in optimizing the procedure for more minimal memory and time use and to Kenneth Scheve for finding an error in an earlier version.

[2] Actually it is fine only in one province of asymptopia: the one where the number of TPs goes to infinity.

[3] Actually, they make a stronger argument, based on the same reasoning and findings as their argument regarding contemporaneous correlation as described in the previous paragraph, that estimating a single serial-correlation parameter rather than N different parameters is probably optimal in small (*i.e.*, real) samples.

adjusting the coefficient estimate. They show that this procedure—controlling for serial correlation in some (simple) way prior to final estimation and relying on robust standard errors, i.e. Panel-Corrected Standard-Errors or PCSE's—has admirable small sample properties relative to alternatives.

Beck and Katz have also made available computer programs for calculating these PCSE's in RATS or GAUSS. However, both of their procedures constrain the data to be analyzed to be rectangular (*i.e.*, all CS's begin and end in the same year) and to have no missing data. This is a restriction in their programs, not in their econometric analysis. *I.e.*, in principle, PCSE's can be estimated in non-rectangular samples and/or samples with missing data. This is fortunate since almost every pooled data set I have encountered has at least one or the other complication. For example, in Chapter III, I had a data set with 20 CS's and up to 35 TP's, but sometimes as few as 20 and with a couple very unfortunately placed[4] missing data points, for a total of 550+/- usable observations, but if I had to rectangularize it somehow and remove breaks in the sample within a CS, it would have had at most 300 observations! An extreme example to be sure, but it is not rare to have to sacrifice 15% or more of one's data to rectangularize it and remove missing-data breaks. Of course, we would like to avoid this if possible. Accordingly, I have written a GAUSS procedure, *pcse.g*, which will accept such data and produce PCSE estimates along with the usual regression output. The procedure also adds several options which I think researchers commonly dealing with TSCS data may find useful.

Beck and Katz's PCSE estimate of coefficient standard errors is defined by:

$$(X'X)^{-1}X'\left(\frac{E'E}{T}\otimes I_t\right)X(X'X)^{-1}$$

E is the TxN (number of TP's by number of CS's) matrix of OLS residuals, $\otimes$ is the Kroenecker product operator, and I_t is an identity matrix of size T. Notice that E'E/T is just an estimate of the error covariances between cross-sections. The difficulty with missing values or non-rectangular data sets is that each element of E'E may not be based on fully T pairs of observations. My procedure solves this by substituting zeros for missing values in X and E and rectangularizing both also with zeros, and dividing each element of E'E by the number of (pairs of) observations on which it is actually based—*i.e.*, not counting the pairs where one or the other (or both) observation(s) is (are) missing. This is done by creating another TxN matrix, M, where each element is equal to one if the corresponding observation in E is not missing and zero if it is. Then, E'E divided element-by-element by M'M may replace E'E/T in the above formula and all is correct.

The procedure has been tested (in Gauss 386-i v3.1 and 3.2.13 running under DOS and a DOS Window under Windows 3.1 and Windows 95) as follows. First, I created some artificial data, rectangular and without missing values. X, the independent variable is random normal. e, the true error is also random normal and independent of X. Y, the dependent variable, was set equal to X+e. Beck's and Katz's GAUSS PCSE procedure (thank you to Neal Beck for the code) was then run on these data. In this sample, my procedure produces identical results. This proves only that my procedure works in perfect rectangular data sets if Beck's and Katz's does, but I consider that about as safe a bet as can be found. Testing how my procedure compares to Beck's in non-rectangular data-sets and/or those with missing values cannot be done because the latter cannot deal with such data to provide the baseline. I did perform a very crude test in this setting by, again generating random normal and independent data as above, only non-rectangular this time. The results are generally close to equal to the OLS results, and converge to it in larger samples, as we would expect if the procedure is working correctly. I fear that is the best that can be done by way of a comparison to the ideal.

[4] *I.e.*, toward the middle rather than toward the end of a CS's data.

More generally, the procedure seems to operate bug-free. (I have been using it extensively in a data set with 850+/- rows, but only 550+/- usable observations; the data are non-rectangular and have a few internal missing values). It has also been used/tested by Gary King's graduate statistics class without mishap following one important correction (gratitude to Ken Scheve).

Highlights of what the procedure does:

(1) Produce PCSE's in any data set where they are theoretically defined.

(2) Automatically generate and use CS and TP fixed-effect dummies if requested.[5]

(3) Estimate and incorporate Panel WLS information prior to applying PCSE's if requested.

> N.b., this is done in two stages. First, OLS is estimated. The residuals from that are squared and regressed on the CS indicators (less one of them) and a constant. The F-test from this regression can be seen as a test of homoskedasticity as the null against CS-specific variance (and the DW from this auxiliary regression can be seen as a test of ARCH(1) against the null of panel heteroskedasticity). The inverse of the square root of the fitted values from this auxiliary regression are then used as weights to transform the original data. OLS is run on the transformed data, which produces the Panel WLS results. The PCSE estimation is then applied to these results. The line in the procedure where the squared residuals are regressed on the CS indicators and related commands can be changed to a different model of error-variance if the researcher desires with minimal hard-wiring. They are flagged for easy recognition.

(4) GAUSS's *OLS.SRC* procedure produces incorrect Durbin-Watson statistics when there are missing values in the middle of the data (Aptech has been notified). The problem is that OLS removes missing values from the data eventually passed to the part of the procedure where estimates are made. This means that later, when any time-series properties (statistics) are analyzed (computed) some of the adjacent observations are no longer temporally adjacent. This implies, for example, that the Durbin-Watson statistic is not based on the correct ordered set of residuals. This problem is fixed prior to the reporting of DW-stats in output from my procedure, but it remains in OLS.SRC.

The text of the procedure (*PCSE.G*) follows. It can also be downloaded from the American University GAUSS Procedure Archive along with more detailed instructions on how to use it than are incorporated as comments into the procedure itself.

Text of the file PCSE.G :

```
@ Gauss Procedure to Estimate Linear Regression, Implementing
* Panel-Corrected Standard-Errors (PCSE) (Beck and Katz (APSR 1995))
* in the Presence of Non-rectangular and/or Missing Data
*
* Please cite "PCSE.G: A Gauss Procedure to Implement Panel-Corrected
* Standard-Errors in Non-Rectangular Data Sets."
*
* by Robert J. Franzese, Jr. (Harvard University) October 1995
* Revised 3/19/96
*
* Code is written and submitted for public, non-commercial use.
* The authors assumes no responsibility whatsoever for the
* performance of the procedure.
*
* Procedure handles non-rectangular data and/or missing observations,
* and (optional) creates and uses cross-section and/or time-period
* fixed effects, and (optional) estimates and uses panel weights (other
* linear variance models can be estimated if desired by editing (hardwiring)
* the line `vx = ...' (remarked in the procedure) A non-linear model could
* of course be substituted.
*
* AUTHOR'S NOTE:
* The assistance of Curt Signorino (Harvard University) in optimizing
* the procedure to reduce computer time and memory use is gratefully
* acknowledged. Gratitude also to Ken Scheve for discovering an error
* in the earlier version.
*
* The author was funded by a Mellon Foundation Dissertation
* Completion Grant during the writing of the procedure.
*
* FORMAT:
* {m,b,stb,zz,stderr,se,sigma,cx,rsq,rbsq,resid,dwstat,nobs,degfree,
* tstats,poft,ursq,urbsq,usigma,udw,ures,corre}
* =pcse(vnames,cnames,y,x,nocs,tstart,tend,weight,cdum,tdum);
```

[5] See Smith (1995) for a cogent discussion of the substantive interpretation of models with each sort of fixed-effects.

```
*
* Procedure assumes data organized {[c1,t1],[c1,t2],...,c[2,t1],...,etc.}',
* that each cross-section has the same number of rows of the data matrix
* allotted to it, though some of those rows may be missing data, and that
* the first column of "indvars" is the constant. (Thus, if your data is non-
* rectangular, you must rectangularize it by putting missing values in. Any
* observations you wish to drop must be fed as missing in the dep var or at
* least one indep var.)
*
* Procedure returns: 22 total, in the following order:
 * m = (matrix) moment matrix (OLS/WLS)
 * b = (vector) coefficients (OLS/WLS)
 * stb = (vector) standardized coefficients (OLS/WLS)
 * zz = (matrix) variance-covariance of coefficients (OLS/WLS + PCSE)
 * stderr = (vector) coefficient standard-errors (OLS/WLS)
 * se = (vector) coefficient standard-errors (OLS/WLS + PCSE)
 * sigma = (scalar) std err of the estimate (WLS, same as usigma if OLS)
 * cx = (matrix) variance-covariance of X and Y (OLS/WLS)
 * rsq = (scalar) R-squared (WLS, same as ursq if OLS)
 * rbsq = (scalar) R-bar-squared (WLS, same as urbsq if OLS)
 * resid = (vector) residuals (WLS, same as ures if OLS)
 * dwstat = (scalar) Durbin-Watson statistic (WLS, same as udw if OLS)
 * nobs = (scalar) number of valid observations
 * degfree = (scalar) degrees of freedom of coefficient estimates
 * tstats = (vector) t-stats of test that coeffs=0 (OLS/WLS + PCSE)
 * poft = (vector) 2-sided prob of those t-tests (OLS/WLS + PCSE)
 * ursq = (scalar) R-squared (unweighted residuals used)
 * urbsq = (scalar) R-bar-squared (unweighted residuals used)
 * usigma = (scalar) std error of estimate (unweighted residuals used)
 * udw = (scalar) Durbin-Watson statistic (unweighted residuals used)
 * ures = (vector) unweighted residuals
 * corre = (matrix) Cross-section Correlation of Residuals

* Procedure Inputs: 10 in the following order:
 * vnames = character vector: variable names, last being dep var
 * cnames = character vector: cross-section unit names
 * y = vector: observations on dependent variable
 * x = matrix: obs. on independent variables including constant (first)
 * time and or c-s dummies can be added for you, omit them here
 * nocs = scalar: number of cross-sectional units
 * tstart = scalar: first time period (earliest from all c-s units)
 * tend = scalar: last time period (latest from all c-s units)
 * weight = scalar: 1 for WLS, 0 for OLS
 * default WLS model is panel weights, can change that directly by
 * hard-wiring the line `vx = ...' in this procedure
 * cdum = scalar: 1 = add c-s dummies, 0 = no c-s dummies
 * tdum = scalar: 1 = add time dummies, 0 = no yime dummies
 * if tdum=1 and cdum=1, procedure drops 1st time dummy @

proc (22) =
 pcse(vnames,cnames,y,x,nocs,tstart,tend,weight,cdum,tdum);

local i,j,t,cdums,tdums,denom,vcve,corre,zz,cm,valid,nindvar,nobs,
 wx,wy,vnam,m,b,stb,vc,stderr,sigma,cx,rsq,resid,dwstat,se,degfree,
 tstats,poft,results,mask,fmt,ures,ursq,urbsq,usigma,udw,tv,rbsq,
 sse,usse,nots,res2,vx,vare,minvar,wvare,wtv,res,resm,uresm,xm,ym,
 wxm,wym,tseq,rresm,rvalid,diagvcve,uresm1l,validm1l,resm1l;

/*declare vars local to procedure*/

format 12,6; /* resets default format for screen output */

nots = tend - tstart + 1; /* computes number of time periods */

if rows(y) /= rows(x); /* this set of ifs checks if input valid */
 "ERROR: Different number of rows in X and Y";
 end;
elseif rows(y) /= nocs*nots;
 "ERROR: Rows in Y and X do not equal # of cs times # of t-p";
 end;
elseif rows(cnames) /= nocs;
 "ERROR: Number of c-s names does not equal number of c-s";
 end;
elseif rows(vnames) /= cols(x)+1;
 "ERROR: Number of variable names does not equal columns X + columns Y";
 end;
else; /* no errors in input detected, create an indicator for valid obs */
 valid = y~x; /*make valid the data matrix */
 valid = (valid./=miss(1,1)); /*set each element to 1 if not miss 0 else*/
 valid = prodc(valid'); /*creates column vector with 1 if valid obs */
 validm1l = missrv(lag1(valid),0); /* lag of valid */
 y = valid.*y+(1-valid).*"miss"; /*put "miss" in y for all missing*/
 y = miss(y,"miss") ; /* convert "miss" to missing value code */
 x = valid.*x+(1-valid).*("miss".*ones(1,cols(x))); /* ditto for x */
 x = miss(x,"miss") ;
endif;

/** Describing of Data Set Shape **/
nobs=sumc(valid); /* compute number of valid observations */
print ftos(nots,"Maximum Number of Time-Periods = %*.*lf",0,0);
print ftos(nocs,"Maximum Number of Cross-Sections = %*.*lf",0,0);
print ftos(nots*nocs,"NoT x NoC (possible observations) = %*.*lf",0,0);
print ftos(nobs,"Actual Number of Observations Valid = %*.*lf",0,0);
print ftos(nots*nocs-nobs,"Number of Invalid Observations = %*.*lf",0,0);

if cdum==1; /* create a matrix of c-s dummies if requested */
 /* delete constant and adjust variable names to match */
 vnames = vnames[2:rows(vnames),.]; /* delete var name for constant*/
 vnames = cnames|vnames; /* append c-s names to remaining var names */
 cdums = (eye(nocs).*.ones(nots,1)); /* create matrix of c-s dummies */
 cdums = valid.*cdums+(1-valid).*("miss".*ones(1,nocs)); /*put miss in*/
 cdums = miss(cdums,"miss");
 x=x[.,2:cols(x)]; /* remove constant from x matrix */
 x=cdums~x; /* c-s dummies are put at front of x-matrix */
elseif cdum /= 0;
 "ERROR: `cdum' argument must be a scalar, 0 (off) or 1 (on)";
 end;
endif;

/** generate time-period dummies **/

if tdum==1; /* create a matrix of time dummies if requested */
 /* delete constant if cdum hasn't already done so */
 /* append time dummies matrix to front of x and */
 /* adjust vnames to match the resulting x matrix */
 tdums = (ones(nocs,1).*.eye(nots)); /* create matrix of t-p dummies */
 tdums = valid.*tdums+(1-valid).*("miss".*ones(1,nots)); /* put miss in */
 tdums = miss(tdums,"miss");
 tseq = seqa(tstart,1,(tend-tstart+1));
 if cdum == 0 ;
 vnames = vnames[2:rows(vnames),.]; /* remove constant name */
 x = x[.,2:cols(x)]; /* remove constant */
 vnames = (0$+"T"$+ftocv(tseq,0,0))|vnames; /*append tnames to v-names*/
 x = tdums~x; /* append tdums to x-matrix */
 else ; /* if c-s dums, same, except trim first time-period dummy */
 vnames = trimr((0$+"T"$+ftocv(tseq,0,0)),1,0)|vnames;
 x = (trimr(tdums',1,0))'~x;
 endif;
elseif tdum /=0;
 "ERROR: `tdum' argument must be a scalar, 0 (off) or 1 (on)";
 end;
endif;

nindvar=cols(x); /* now compute total number of independent variables */
print ftos(nindvar,"Number of Independent Variables = %*.*lf",0,0);
"";
degfree = nobs-nindvar; /* compute degrees of freedom */

if weight==0; /* if weighting not requested, do OLS */
 "Using OLS in First and Second Stage...";
 “”;
 __altnam = vnames; /* feeds vnames to OLS */
 __miss = 1; /* instructs OLS to omit missing data */
 __con = 0; /* instructs OLS not to add another constant */
 _olsres = 1; /* instructs OLS to compute residuals and DW-stat */
 __output = 0; /* set this to 1 if you want OLS results printed out */

{ vnam,m,b,stb,vc,stderr,sigma,cx,rsq,resid,dwstat }=OLS(0,y,x); /* OLS */

 rbsq = 1-(1-rsq)*((nobs-1)/degfree); /* compute r-bar-squared */
 res = y-x*b; /* compute nocs*nots by 1 vector of residuals */
 resm = missrv(res,0); /* an alternate resid vector with 0 for missing*/
 resm1l = missrv(lag1(resm),0); /* lag of resm */
 ures=res; /* unweighted residuals, same as OLS here */
 uresm = missrv(ures,0); /* alternate with zeros for missings */
 sse=resm’resm; /* compute sum of squared errors */
 usse=sse; /* unweighted sum of squared errors, same as OLS here */
 ursq=rsq; /* unweighted r-squared, same as OLS here */
 urbsq=rbsq; /* unweighted r-bar-squared, same as OLS here */
 usigma=sigma; /* unweighted std err of est, same as OLS here */

/** nb: GAUSS OLS computes incorrect DW when missing values present **/

 denom = sumc(valid.*validm1l.*resm^2); /* denom for DW is sum sq res */
 dwstat = sumc(valid.*validm1l.*(resm-resm1l)^2); /* numerator is... */
 dwstat = dwstat/denom; /*...sum of squared differences */
 udw=dwstat; /* unweighted DW-stat, same as OLS here */

 “First-stage done, now computing Panel V-Cov of Residuals...”;

 @ covariance of resid[i,t] with resid[j,t] is computed
 * taking care to divide sum of cross-products only by the
 * number of valid observations @

 rvalid = reshape(valid,nocs,nots)’; /* reshape valid to nots x nocs */
 rresm = reshape(resm,nocs,nots)’; /* ditto for residual vector */
 vcve = rresm’*rresm; /* generate nocs by nocs matrix of crossprods */
 denom = rvalid’*rvalid;
 denom = miss(denom,0);
 if ismiss(denom);
 “ERROR: A c-s unit exists w/o any observations or w/o “\
 “any observations in common with another c-s unit.”;
 “You must remove that c-s unit from the data passed to PCSE.”;
 end;
 endif;
 vcve = vcve./denom;

 “Done, now computing cross-section residual-correlation matrix...”;

 diagvcve = sqrt(diag(vcve));
 corre = vcve./(diagvcve*diagvcve’);

 “Done, now computing PCSE V-COV of b...”;
 xm = missrv(x,0); /* GAUSS doesn’t like to do m-algebra with missing */
 ym = missrv(y,0); /* obs, oblige it by replacing with 0. This has */
 /* no effect on x’x, just watch the denominators */
 tv = ym’ym-nobs*((sumc(ym)/nobs)^2); /* sum of squared variance of y */
 wtv = tv; /* sum of squared variance of weighted y, same for OLS */

 zz = (inv(xm’xm))*xm’(vcve.*.(eye(nots)))*xm*(inv(xm’xm));
 /*PCV-COV(b)*/

 se = sqrt(diag(zz)); /* PCSE’s */
 tstats = b./se; /* PC t-stats */
 poft = 2*cdftc(abs(tstats),degfree); /* 2-sided p of |PCSE t’s| */

elseif weight==1; /* if weighted least squares requested */
 “Using OLS in First Stage...”;
 “”;
 __altnam = vnames; /* feed OLS the relevant variable names */
 __miss = 1; /* instruct OLS to omit missings */
 __con = 0; /* instruct OLS not to add another constant */
 _olsres = 1; /* instruct OLS to compute residuals and DW-stat */
 __output = 0; /* change this to 1 if you want the OLS output */

{ vnam,m,b,stb,vc,stderr,sigma,cx,rsq,resid,dwstat }=OLS(0,y,x); /* OLS */

“Done, now calculating weighting series using OLS on OLS residuals squared”;

 res2=(y-x*b).*(y-x*b); /* compute nocs*nots by 1 vector of residuals */
 if cdum==0; /* check to see if matrix of c-s dummies exists, if not...*/
 cdums = (eye(nocs).*.ones(nots,1)); /* create matrix of c-s dums */
 cdums = valid.*cdums+(1-valid).*(“miss”.*ones(1,nocs));
 cdums = miss(cdums,”miss”);
 endif;

@ Residual Variance Model can be changed by editing the line `vx = ...’ @

 vx = ones(nocs*nots,1)~cdums[.,2:nocs] ; /* ind vars in res var model */

@ The following process creates a weighting series by regressing the squares
* of the OLS residuals on the cross-section dummies (and whatever other
* explanitory variables you might have included in the preceding line.)
* The inverse of the square roots of the fitted values are the weights.
* Without other explanitors than the c-s dummies, this is exactly equivalent
* to panel-weighted least-squares @

 “----> n.b., F-stat is test of variance model against homoskedasticity”;
 “”;

@ if `vx= ...’ is changed, the following line must be altered to match @

 __altnam = “one”|cnames[2:nocs,1]|”SqOLSres”;
 __output = 1; /* 0 if you don’t want the res var model output */

 { vnam,m,b,stb,vc,stderr,sigma,cx,rsq,resid,dwstat }=OLS(0,res2,vx);

 vare = vx*b; /* make nocs*nots by 1 vector of fitted res variance */

@ The following if...endif checks for negative fitted values
* in the variance model, replacing them with the minimum positive value.
* (author’s note: substitution of a more appropriate model than a linear
* model of variance may be particularly appropriate if negative fitted
* values are common) They are not possible in the panel-weight set-up. @

“”;
“Calculating Weighting Series...”; /* Set negative wts to min positive */
 if minc(vare) <= 0;
 minvar = minc((vare.>0).*vare+(vare.<=0).*maxc(vare));
 vare = (vare.>0).*vare+(vare.<=0).*minvar;
 endif;

 wvare = ones(rows(vare),1)./sqrt(vare); /* weighting series */
 wx = wvare.*x; /* transform x to weighted x */
 wy = wvare.*y; /* transform y to weighted y */
 wxm = missrv(wx,0); /* create alternates wx and wy where miss are 0 */
 wym = missrv(wy,0);

“Done, now doing WLS using OLS on transformed variables...”;

 __altnam = vnames;
 __output = 0; /* change this to 1 if you want WLS output */

{ vnam,m,b,stb,vc,stderr,sigma,cx,rsq,resid,dwstat }=OLS(0,wy,wx); /*WLS*/

 res = wy-wx*b; /* nocs*nots vector of weighted residuals */
 resm = missrv(res,0); /* alternate with 0 for missing */
 resm1l = missrv(lag1(resm),0); /* lag of resm */
 rbsq = 1-(1-rsq)*((nobs-1)/degfree); /* weighted r-bar-squared */
 ures = y-x*b; /* calculate unweighted residuals */
 uresm = missrv(ures,0); /* alternate, 0 for missings */
 ym = missrv(y,0); /* alternate y with zeros for missings */
 tv = ym'ym-nobs*((sumc(ym)/nobs)^2); /* unweighted sum of var(y) */
 wtv = wym'wym-nobs*((sumc(wym)/nobs)^2); /* weighted sum of var(y) */
 ursq = (tv - uresm'uresm)/tv; /* unweighted r-squared */
 urbsq = 1-(1-ursq)*((nobs-1)/degfree); /* unweighted r-bar-squared */
 sse=resm'resm; /* sum of wwighted residuals variance */
 usse = uresm'uresm; /* unweighted sum of squared residuals */
 /* n.b.,in WLS, unweighted resids may not have mean 0 */
 usigma = sqrt((uresm'uresm)/degfree); /* unwtd std err of estimate */
                /* computing unweighted DW-stat */
 uresm1l = lag1(uresm); /* create lagged residual (alternate) */
 uresm1l = missrv(uresm1l,0); /* replace missings with zeros */
 validm1l = missrv(lag1(valid),0); /* ditto for valid */
 denom = sumc(valid.*validm1l.*uresm^2); /* denom for DW is sum sq res */
 udw = sumc(valid.*validm1l.*(uresm-uresm1l)^2); /* numerator is... */
 udw = udw/denom; /*...sum of squared differences */

/** nb: GAUSS OLS computes incorrect DW when missing values present **/

 denom = sumc(valid.*validm1l.*resm^2); /* denom for DW is sum sq res */
 dwstat = sumc(valid.*validm1l.*(resm-resm1l)^2); /* numerator is... */
 dwstat = dwstat/denom; /*...sum of squared differences */

"Done, computing PCSE V-COV of WLS residuals...";
 rvalid = reshape(valid,nocs,nots)'; /* reshape valid to nots by nocs */
 rresm = reshape(resm,nocs,nots)'; /* ditto for resids */
 vcve = rresm'*rresm; /* calculate nocs by nocs cross-prod of resids */
 denom = rvalid'*rvalid; /* calculate nobs for each element of vcve */
 denom = miss(denom,0);
 if ismiss(denom);
 "ERROR: A c-s unit exists w/o any observations or w/o "\
 "any observations in common with another c-s unit.";
 "You must remove that c-s unit from the data passed to PCSE.";
 end;
 endif;
 vcve = vcve./denom; /* cross-prods over nobs for each is covar matrix */

"Done, computing cross-section residual-correlation matrix...";

 diagvcve = sqrt(diag(vcve)); /* sq root of diag elements = std dev */
 corre = vcve./(diagvcve*diagvcve'); /* vcv over sd()sd() = corr */

"Done, computing PCSE V-COV of b..."; /* PCSE V-COV(b) with wtd resids */

 zz = (inv(wxm'*wxm))*wxm'*(vcve.*.(eye(nots)))*wxm*(inv(wxm'*wxm));
                /* PCV-COV */

 se = sqrt(diag(zz)); /* PCSE's from WLS */
 tstats=b./se; /* PC t-stats from WLS */
 poft = 2*cdftc(abs(tstats),degfree); /* 2-sided p of |PC t| */

else;

 "Weight parameter must be set as 0 (=off) or 1 (=on)";
 end;

endif;
"** PCSE Results **"; /* printing out the results pretty-like */
"";
print ftos(nobs,"Valid Cases: %*.*lf",20,0);;
print ftos(vnames[rows(vnames)]," Dependent Variable:%*.*s",20,0);
print ftos(nocs*nots-nobs,"Missing Cases:%*.*lf",20,0);;
print " Deletion Method: ";;
print "Listwise"; /** always uses listwise **/
if weight == 1;
 print ftos(wtv,"Wtd Total SS: %*.*lf",20,3);;
 print ftos(degfree," Degrees of Freedom:%*.*lf",20,0);
endif;
print ftos(tv,"Total SS: %*.*lf",20,3);;
print ftos(degfree," Degrees of Freedom:%*.*lf",20,0);
if weight == 1;
 print ftos(rsq,"Wtd R-Squared:%*.*lf",20,3);;
 print ftos(rbsq," Wtd Rbar-Squared: %*.*lf",20,3);
endif;
print ftos(ursq,"R-Squared: %*.*lf",20,3);;
print ftos(urbsq," Rbar-Squared: %*.*lf",20,3);
if weight == 1;
 print ftos(sse,"W Residual SS:%*.*lf",20,3);;
 print ftos(sigma," Wtd Std Error Est: %*.*lf",20,3);
endif;
print ftos(usse,"Residual SS: %*.*lf",20,3);;
print ftos(usigma," Std Error of Est: %*.*lf",20,3);
print ftos(udw,"DW-Stat: %*.*lf",20,3);;
print ftos(dwstat," Wtd DW-Stat: %*.*lf",20,3);
print;
print " PCSE Std Prob Sta"\
"ndardized Reg Std";
print "Variable Estimate Error t-value >|t| E"\
"stimate Error";
print "-------------------------------------------------------------"\
"------------------";
results = vnames[1:nindvar,1]~b~se~tstats~poft~stb~stderr;
mask = 0~1~1~1~1~1~1;
let fmt[7,3] = "-*.*s" 9 8 "*.*lf" 12 6 "*.*lf" 12 6 "*.*lf" 12 6 ""\
 "*.*lf" 10 4 "*.*lf" 12 6 "*.*lf" 12 6;
call printfm(results,mask,fmt);
"";

retp(m,b,stb,zz,stderr,se,sigma,cx,rsq,rbsq,res,dwstat,
 nobs,degfree,tstats,poft,ursq,urbsq,usigma,udw,ures,corre);

endp;
```