ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/24137079

Generalized linear models: revision of glm

Article · February 1994

Source: RePEc

CITATIONS
20

READS
1,398

1 author:

Patrick Royston
University College London
401 PUBLICATIONS **56,778** CITATIONS

SEE PROFILE

All content following this page was uploaded by Patrick Royston on 27 July 2016.

The user has requested enhancement of the downloaded file.

STATA
TECHNICAL
BULLETIN

March 1994
STB-18

A publication to promote communication among Stata users

Editor

Sean Becketti
Stata Technical Bulletin
702 University Drive East
College Station, Texas 77840
409-696-4600
409-696-4601 FAX
stb@stata.com EMAIL

Associate Editors

J. Theodore Anagnoson, Cal. State Univ., LA
Richard DeLeon, San Francisco State Univ.
Paul Geiger, USC School of Medicine
Lawrence C. Hamilton, Univ. of New Hampshire
Joseph Hilbe, Arizona State University
Stewart West, Baylor College of Medicine

Subscriptions are available from Stata Corporation, email stata@stata.com, telephone 979-696-4600 or 800-STATAPC, fax 979-696-4601. Current subscription prices are posted at www.stata.com/bookstore/stb.html.

Previous Issues are available individually from StataCorp. See www.stata.com/bookstore/stbj.html for details.

Submissions to the STB, including submissions to the supporting files (programs, datasets, and help files), are on a nonexclusive, free-use basis. In particular, the author grants to StataCorp the nonexclusive right to copyright and distribute the material in accordance with the Copyright Statement below. The author also grants to StataCorp the right to freely use the ideas, including communication of the ideas to other parties, even if the material is never published in the STB. Submissions should be addressed to the Editor. Submission guidelines can be obtained from either the editor or StataCorp.

Copyright Statement. The Stata Technical Bulletin (STB) and the contents of the supporting files (programs, datasets, and help files) are copyright © by StataCorp. The contents of the supporting files (programs, datasets, and help files), may be copied or reproduced by any means whatsoever, in whole or in part, as long as any copy or reproduction includes attribution to both (1) the author and (2) the STB.

The insertions appearing in the STB may be copied or reproduced as printed copies, in whole or in part, as long as any copy or reproduction includes attribution to both (1) the author and (2) the STB. Written permission must be obtained from Stata Corporation if you wish to make electronic copies of the insertions.

Users of any of the software, ideas, data, or other materials published in the STB or the supporting files understand that such use is made without warranty of any kind, either by the STB, the author, or Stata Corporation. In particular, there is no warranty of fitness of purpose or merchantability, nor for special, incidental, or consequential damages such as loss of profits. The purpose of the STB is to promote free communication among Stata users.

The *Stata Technical Bulletin* (ISSN 1097-8879) is published six times per year by Stata Corporation. Stata is a registered trademark of Stata Corporation.

Contents of this issue

an1.1 STB categories and insert codes

Inserts in the STB are presently categorized as follows:

General Categories:

an	announcements	*ip*	instruction on programming
cc	communications & letters	*os*	operating system, hardware, & interprogram communication
dm	data management		
dt	data sets	*qs*	questions and suggestions
gr	graphics	*tt*	teaching
in	instruction	*zz*	not elsewhere classified

Statistical Categories:

sbe	biostatistics & epidemiology	*srd*	robust methods & statistical diagnostics
sed	exploratory data analysis	*ssa*	survival analysis
sg	general statistics	*ssi*	simulation & random numbers
smv	multivariate analysis	*sss*	social science & psychometrics
snp	nonparametric methods	*sts*	time-series, econometrics
sqc	quality control	*sxd*	experimental design
sqv	analysis of qualitative variables	*szz*	not elsewhere classified

In addition, we have granted one other prefix, *crc*, to the manufacturers of Stata for their exclusive use.

an41 STB office moving

The editorial office of the STB is moving from Kansas to New York. As this issue goes to press, the move is taking place, and the new address and telephone numbers are not available yet. Until the move is complete, submissions and other correspondence can be sent to the production office:

Stata Corporation
702 University Drive East
College Station, Texas 77840
409-696-4000 (voice)
409-696-4601 (FAX)

The address and telephone numbers for the new editorial office will be announced in the next issue of the STB.

sg16.5 Negative binomial regression

Joseph Hilbe, Dept. of Sociology, Arizona State University
FAX: 602-860-1446, EMAIL: atjmh@asuvm.inre.asu.edu

Poisson regression is the standard method used to analyze count data. However, many real life data situations violate the assumptions upon which the Poisson model is based. For instance, the Poisson model assumes that the mean and variance of the response are identical. This means that events occur within a period of observation at a constant rate; an event is equally likely at any point within the period. When there is heterogeneity in the data, it is likely that the Poisson model is overdispersed. Such overdispersion is indicated if the variance of the response is greater than its mean. We may also check for model overdispersion by submitting the data to a Poisson model and observing the χ^2-based or Deviance-based dispersion statistic. The model is Poisson-overdispersed if the dispersion value is greater than unity. The `glm` command (Hilbe 1993) provides the user with both dispersion values.

Negative binomial regression is most effectively used to model count data that violates the Poisson assumption of the equality of mean and variance. In effect, the model is based upon the premise that events enter a period of observation with a gamma distribution. Noting that the Poisson and gamma variances are μ and μ^2, respectively, the negative binomial is considered as a Poisson-gamma mixture distribution with a variance of $\mu + \mu^2$. This function may be reparameterized as $\mu + k\mu^2$ to allow a linear relationship in the second term. The parameter k can be regarded as a heterogeneity factor and is entered into the function as a known constant (Godambe 1991; Nelder 1993). In fact, the standard negative binomial formulation requires that k be fixed and independent of μ. If it is parameterized in any other manner, then the distribution is not a member of a GLM-type exponential family (Nelder 1993).

The glm command allows for the direct specification of the amount of heterogeneity thought to be present in the data by the use of the k() option with the log-linked negative binomial model, f(nb) l(pow) p(0). The default value of k is 1; but models typically require values ranging from .001 to 2. The goal is to adjust the value of k so that the χ^2-based or Deviance-based dispersion approximates 1, which is the optimal value for a Poisson model. Again, k represents the amount of positive heterogeneity in the otherwise Poisson count data.

As an example I shall create a negative binomial data set using a random number generator which allows for input of a mean variable, defined as the inverse of the link function, and a value for k. In the case of the log-linked negative binomial, the inverse link function is simply exp(lp), the same as for the Poisson, where lp is the linear predictor (see Note). I shall set the observations at 1000 and create two positive-valued random normal predictors, and example coefficients.

```
. set obs 1000
. gen x1=abs(invnorm(uniform()))
. gen x1=abs(invnorm(uniform()))
. gen byte b0=2
. gen b1=.5
. gen byte b2=3
. gen lp = b0 + b1*x1 + b2*x2
. gen mu = exp(lp)     /* inverse log link */
. rndnblx mu, k(.1)    /* Neg. Binomial random number generator */
```

Now model the resultant constrained NB random deviate using glm with the same value for k.

```
. glm xnb x1 x2, f(nb) l(pow) p(0) k(.1)
Iter 1 : Dev = 16788.2500
Iter 2 : Dev = 6303.8691
Iter 3 : Dev = 1848.6648
Iter 4 : Dev = 1086.5071
Iter 5 : Dev = 1062.9047
Iter 6 : Dev = 1062.8718
Iter 7 : Dev = 1062.8718
                                                No obs.    =      1000
Chi2       =  1034.838                          Deviance   =  1062.872
Prob>chi2  =  .1971762                          Prob>chi2  =  .0723221
Dispersion =  1.037952                          Dispersion =   1.06607
Negative Binomial distribution
Power link, power = 0
------------------------------------------------------------------------------
     xnb |      Coef.   Std. Err.       t     P>|t|      [95% Conf. Interval]
---------+--------------------------------------------------------------------
      x1 |    .476599   .0180338     26.428   0.000      .4412529    .5119452
      x2 |   2.997234   .0182415    164.308   0.000       2.96148    3.032987
   _cons |   2.029904   .0242727     83.629   0.000       1.98233    2.077478
------------------------------------------------------------------------------
```

The estimates are nearly the same as those used to create the data set; the fact that we only had 1000 observations accounts for the observed deviation. I have constructed a similar model with 50,000 observations; the χ^2-based dispersion is nearly identical to 1.0 and the coefficients are practically identical to the original values. Regardless, if we wanted to fine tune the model to accommodate even more heterogeneity, we can model the same data using a k-value of .1038 (.038 over the original .1). Note that little change occurs in the estimates and standard errors.

```
. glm xnb x1 x2, f(nb) l(pow) p(0) k(.1038)
Iter 1 : Dev = 16207.9512
Iter 2 : Dev = 6097.7090
 ..
Iter 7 : Dev = 1029.5723
                                                No obs.    =      1000
Chi2       =  1001.698                          Deviance   =  1029.572
Prob>chi2  =  .4522575                          Prob>chi2  =  .2307049
Dispersion =  1.004712                          Dispersion =   1.03267
Negative Binomial distribution
Power link, power = 0
------------------------------------------------------------------------------
     xnb |      Coef.   Std. Err.       t     P>|t|      [95% Conf. Interval]
---------+--------------------------------------------------------------------
      x1 |   .4764923   .0183481     25.970   0.000      .4405303    .5124544
      x2 |   2.997193   .0185529    161.549   0.000      2.960829    3.033556
   _cons |   2.030026   .0246486     82.359   0.000      1.981715    2.078337
------------------------------------------------------------------------------
```

Simulation studies have given support to the notion that the optimal NB model results when k is adjusted in such a manner that the χ^2-based dispersion, defined as

$$\sum \frac{(y-\mu)^2}{V(\mu)}$$

approximates 1.0. Usually one requires only two or three adjustments to determine the optimal model. If it is of interest, one may assess the statistical difference from the Poisson model by subtracting the NB deviance from the Poisson deviance and calculating the χ^2 probability with one degree of freedom.

Running the same model using Poisson regression results in estimates which are fairly close to those produced by the above NB model. However, as expected, the standard errors are much too narrow. This provides additional evidence that the predictor significance values may be overly optimistic when modeling overdispersed data with Poisson regression; i.e., values may indicate that predictors significantly contribute to the model when in fact they do not.

Stata's ML **nbreg** command also estimates a log negative binomial model. For large data sets the estimates and SEs are nearly identical. Running **nbreg** on the above data results in the following output:

```
. nbreg xnb x1 x2
Iteration 0:  Log Likelihood = -8318.3433
  ..
  ..
Iteration 16:  Log Likelihood = -5116.6982
Negative Binomial Regression                      Number of obs    =    1000
                                                  Model chi2(2)    = 4252.25
                                                  Prob > chi2      =  0.0000
Log Likelihood =  -5116.6981872                   Pseudo R2        =  0.2935
------------------------------------------------------------------------------
     xnb |      Coef.   Std. Err.       z    P>|z|      [95% Conf. Interval]
---------+--------------------------------------------------------------------
_lnmean  |
      x1 |    .476493   .0184595     25.813   0.000       .440313    .5126729
      x2 |   2.997193   .0186446    160.754   0.000      2.960651    3.033736
   _cons |   2.030026   .0248302     81.756   0.000       1.98136    2.078692
---------+--------------------------------------------------------------------
_lnalpha |
   _cons |  -2.26412   .0516951    -43.798   0.000      -2.36544   -2.162799
------------------------------------------------------------------------------
   alpha     .1039214   [_lnalpha]_cons = ln(alpha)
                  (LR test against Poisson, chi2(1) = 100898.6 P =  0.0000)
```

The results prove to be nearly the same. The foremost difference is time. On a 486DX 33MH with 16M RAM, the above **glm** run took 6.8 seconds whereas **nbreg** took a little over 7 minutes. For smaller data sets the extra time may be of no consequence; but there may be differences in estimates produced. Moreover, the **glm** negative binomial may be a more stable model when dealing with small data sets (Rogers 1993). With them, the ML method of both **nbreg** and comparable LIMDEP may at times estimate parameters and k (α) which still allow for some unaccounted overdispersion—based on the same methods used to assess Poisson overdispersion and hence justify NB modeling. To check for this, simply use the **nbreg** estimate of α for the value of k in **glm**. If the χ^2 dispersion is over 1.0, there is remaining overdispersion. Admittedly, altering k to get dispersion values closer to 1.0 does not substantially alter the estimates and SEs in most cases. But it does attempt to further reduce overdispersion in the model. Additionally, I always use the **glm** diagnostic capabilities when dealing with real data; e.g., with **gpredict** I can obtain negative binomial μ, η, hat, Pearson, deviance, standardized Pearson, and standardized deviance residual statistics. Residual analysis is requisite for proper model building and is well defined in the GLM context.

An advantage that **nbreg** has over the log negative binomial **glm** command is that the latter requires one to specify the amount of heterogeneity by use of the k option. It may take several runs to find a model which results in a χ^2 dispersion value approximating 1.0. Of course, when doing this k is being estimated from the dispersion value, which itself is based upon iteratively solved fit and variance values. In order to assist the user, I have developed a program which finds estimates by iteratively reducing overdispersion. k is initialized as the inverse of the Poisson dispersion. Used on the constructed example data, **glmnb** yields the following results:

(*Continued on next page*)

```
. glmnb xnb x1 x2
1:    Deviance:  5813.914  Alpha (k):  .0094763  Disp:   5.834937
2:    Deviance:  1707.187  Alpha (k):  .0552936  Disp:   1.684133
3:    Deviance:  1127.058  Alpha (k):  .0931218  Disp:   1.102079
4:    Deviance:  1040.316  Alpha (k):  .1026276  Disp:   1.015434
5:    Deviance:  1027.194  Alpha (k):  .1042116  Disp:   1.002339
6:    Deviance:  1025.206  Alpha (k):  .1044553  Disp:   1.000355
7:    Deviance:  1024.905  Alpha (k):  .1044923  Disp:   1.000054
8:    Deviance:  1024.859  Alpha (k):  .1044979  Disp:   1.000009
                                                 No obs.     =      1000
                                                 Poisson Dev =  104585.2
                                                 Neg Bin Dev =  1024.859
Alpha (k)   =  .1044988                          Prob > Chi2 =    0.0000
Chi2        =  997.0086                          Deviance    =  1024.859
Prob>chi2   =  .4939676                          Prob>chi2   =   .263347
Dispersion  =  1.000009                          Dispersion  =  1.027943
GLM: Negative binomial regression
------------------------------------------------------------------------------
     xnb |      Coef.   Std. Err.       t     P>|t|      [95% Conf. Interval]
---------+--------------------------------------------------------------------
      x1 |   .4764776   .0183939     25.904   0.000      .4404258    .5125293
      x2 |   2.997187   .0185982    161.155   0.000      2.960735    3.033639
   _cons |   2.030043   .0247034     82.177   0.000      1.981625    2.078462
------------------------------------------------------------------------------
```

The program takes less than 15 seconds to run and yields results which are more precise than the `glm` results above. I recommend its use when one has determined that the count data to be modeled is overdispersed. `gpredict` is compatible with `glmnb`.

The negative binomial model fits squarely within the GLM-type exponential family of distributions. The log-linked form of the distribution allows useful modeling of many types of overdispersed count data. However, if overdispersion results from autocorrelation, the event counts are not independent and hence the likelihood function is not equal to the product of the individual probability functions. A standard GLM-type model should not be used in this situation. But where there is independence and heterogeneity is not the result of longitudinal effects, the negative binomial may prove to be a powerful addition to an analyst's statistical toolbox.

Note

Walter Linde-Zwirble of Health Outcomes Technologies and I have developed a number of useful random number generators in Stata. One type simply generates a random deviate with a specified mean, shape and/or scale, etc. Most also allow the mean to be input as a variable per the example used in this article. RNGs include: t, F, χ^2, lognormal, binomial, negative binomial, beta binomial, Poisson, overdispersed Poisson, exponential, gamma, inverse Gaussian, Weibull, and a three parameter generalized logistic. Others may be forthcoming. Individuals interested in them may contact the author.

References

Godambe, V. P. (ed.) 1991. *Estimating functions*. New York: Oxford University Press. (In particular see Dean, Estimating equations for mixed Poisson models.)

Hilbe, J. 1993. sg16.2: GLM: A unified power-link based program including the negative binomial. *Stata Technical Bulletin* 14: 16–19.

——. 1994. Log negative binomial regression as a generalized linear model. Technical Report 26, Committee on Statistics, Graduate College, Arizona State University.

King, G. 1989. Variance specification in event count models: from restrictive assumptions to a generalized estimator. *American Journal of Political Science* 33: 762–84.

McCullagh, P. and J. A. Nelder. 1989. *Generalized Linear Models*, 2nd ed. New York: Chapman & Hall.

Nelder, J. and Y. Lee. 1992. Likelihood, quasi-likelihood and pseudolikelihood: some comparisons. *Journal of the Royal Statistical Society B* 54: 1, pp. 273–284.

Nelder, J. 1993. Generalized linear models with negative binomial or beta-binomial errors. unpublished manuscript.

Rogers, W. H. 1993. sg16.4: Comparison of `nbreg` and `glm` for negative binomial. *Stata Technical Bulletin* 16: 7.

sg22	Generalized linear models: revision of `glm`

Patrick Royston, Royal Postgraduate Medical School, London, FAX (011)-44-81-740-3119

Generalized linear models (GLMs) represent an important, modern and flexible approach to a wide range of data-analysis problems. Standard Stata 3.1 does not support such models in a general way, though several GLMs are provided by special commands, notably `logit/blogit` for logistic models, `poisson` for Poisson regression and `nbreg` for models with negative binomial errors (useful for "over-dispersed" Poisson data).

The arrival of Hilbe's (1993a) `glm` ado-file in the STB was therefore a significant event in the lives of many Stata users, particularly those familiar with the classic GLM tool, GLIM (Baker and Nelder 1985). Unfortunately, `glm` can produce meaningless results if a meaningless combination of options are specified (as a naive user might) and, worse, it can produce incorrect results in some other cases. In addition, certain choices made in implementation make `glm`, to my mind, sometimes unpleasant to use. A partial list of these problems (based on Hilbe 1993b) include

1. There is no default model, e.g. entering '`glm yvar xvar`' causes program failure.
2. No default link function is provided for each error structure, so if `link(`*link_function*`)` is not specified, the program fails.
3. The p-values from chi-square tests applied to the residual deviance and residual "chi-square" statistics are only valid for a small subset of error distributions but are given for all. For distributions with an estimated scale parameter (Gaussian, gamma, inverse Gaussian) they make no sense.
4. The program gives incorrect results if either `if` or `in` filters are used.
5. There is inadequate checking of user inputs, e.g. invalid families, links, etc., do not raise sensible error messages.
6. The wrong deviance is given for Poisson models in which at least one value of the dependent variable is 0.
7. `gpredict` has some major errors, e.g. with Gaussian data, certain residuals are calculated with the expected value of the dependent variable in the denominator. The latter could be zero.

[Royston listed an eighth problem in the original submission: "As supplied on the STB-16 disk, `glm` would not work at all, returning the mysterious error message 'STB not found'." This is my fault. References to the STB issue and insert number are added as comments to the top of programs before inclusion on the STB disk. In this case, the comment marker '`*`' was omitted.—Ed.]

Despite these problems, Hilbe provided the basis for a good GLM command.

glmr: a revision of glm

The main changes (from the user's perspective) made to `glm` in producing `glmr` are to the user interface—particularly defining link functions and in providing sensible default values—and in increased error-checking. Although `glm` and `glmr` appear otherwise quite similar, essentially, the program has been completely rewritten. I have not yet updated `gpredict` but plan to do so in a later insert. As a temporary measure, three of the most important variables created by `gpredict` may be obtained from `glmr` by using the `frvars` option.

Syntax

```
glmr depvar [varlist] [if exp] [in range] [weight] [, noconstant
     family(binomial [nvar|#] | gamma | gaussian | igaussian | nbinomial | poisson)
     link(power # | identity | log | logit | probit | cloglog | opower # | nbinomial)
     scale(x2 | dev | #) [ln]offset(varname) k(#) disp(#)
     frvars eform level(#) iterate(#) ltol(#) init(varname_μ) nolog ]
```

Description

`glmr` fits generalized linear models of *depvar* with the covariates `x` in *varlist*:

$$g\big(\mathrm{E}(depvar)\big) = \mathbf{xb}, \qquad depvar \text{ distributed according to user-specified family.}$$

The estimates produced are a function of the link function $g(\,)$ and the distribution family. (E$(\,)$ is the expectation operator above.) For instance, if *depvar* is assumed to have a Gaussian (normal) distribution and $g(\,)$ is assumed to be the identity function, we have,

$$\mathrm{E}(\textit{depvar}) = \mathbf{xb}, \qquad \textit{depvar} \text{ distributed Gaussian (normal)}$$

or linear regression. If $g(\,)$ is the natural log function and *depvar* assumed to be distributed Poisson, we have,

$$\log\bigl(\mathrm{E}(\textit{depvar})\bigr) = \mathbf{xb}, \qquad \textit{depvar} \text{ distributed Poisson}$$

or Poisson regression. Other combinations are possible.

While `glmr` can be used to estimate linear regression (and, in fact, does so by default), this should be viewed solely as an instructional feature; `regress` produces such estimates more quickly and (possibly) more accurately.

In any case, you specify the link function using the `link()` option and the distributional family using `family()`. The allowed link functions are

Link function	`glmr` option	comment
identity	`link(identity)`	equiv. to `link(power 1)`
log	`link(log)`	equiv. to `link(power 0)`
logit	`link(logit)`	equiv. to `link(opower 0)`
probit	`link(probit)`	
complementary log-log	`link(cloglog)`	
odds power	`link(opower #)`	e.g., `link(opower 1)`
power	`link(power #)`	e.g., `link(power -1)` or `link(power 2)`

The allowed distributional families are

Family	`glmr` option	comment
Gaussian (normal)	`family(gaussian)`	synonym: `family(normal)`
Inverse Gaussian	`family(igaussian)`	
Bernoulli/binomial	`family(binomial)`	see note
Poisson	`family(poisson)`	
Negative binomial	`family(nbinomial)`	
Gamma	`family(gamma)`	

Note: the binomial distribution can be specified as (1) `family(binomial)`, (2) `family(binomial #)`, or (3) `family(binomial` *varname*`)`. In case 2, # is the constant value of the binomial denominator. Specifying `family(binomial 1)` is the same as specifying `family(binomial)`; it indicates that *depvar* has the so-called Bernoulli distribution with values 0 and 1 only. In case 3, *varname* is the variable containing the binomial denominator, thus allowing the denominator to vary.

Not all combination of `family()` and `link()` make sense; when specifying these two options, you may choose among the following combinations:

	identity	log	logit	probit	cloglog	power	opower	nbinomial	default link
Gaussian	x	x				x			`identity`
binomial	x	x	x	x	x	x	x		`logit`
Poisson	x	x				x			`log`
gamma	x	x				x			`power -1`
inverse Gaussian	x	x				x			`power -2`
negative binomial	x	x				x		x	`nbinomial`

Notes: Default indicates the assumed link if `link()` is not specified, which in all cases is the so-called canonical link. The default link for the negative binomial, `link(nbinomial)`, is $\ln\bigl(\mathrm{E}(\textit{depvar})/\mathrm{E}(\textit{depvar}+k)\bigr)$.

Some `family()` and `link()` combinations result in models already estimated by Stata. These are

`family()`	`link()`	Other Stata estimation command
`gaussian`	`identity`	`regress` or `fit`
`binomial`	`logit`	`logit` or `logistic`
`binomial`	`probit`	`probit` (see note 1)
`poisson`	`log`	`poisson`
`nbinomial`	`log`	`nbreg` (see note 2)
`gamma`	`log`	`ereg` (see note 3)

Note 1: Coefficients will be the same; standard errors will be only asymptotically equivalent because probit is not the "canonical" link for the binomial family.

Note 2: While `glmr` can be used to estimate negative binomial regression, use of Stata's maximum-likelihood `nbreg` is probably preferred; see the `k()` option below.

Note 3: This requires specifying `scale(1)` (see options below). `glmr` cannot be used to estimate exponential regressions on censored data. As with probit, standard errors will be only asymptotically equivalent because log is not the "canonical" link for the gamma family.

Comparison to glm

1. All the bugs I noticed in `glm` have been fixed (but others may remain).
2. A `noconstant` option has been added, to provide GLMs with no constant term in the linear predictor.
3. The syntax for specifying the 'grouped' binomial distribution has been altered. The `group` option has been dropped. The binomial denominator is now indicated after the `binomial` phrase in `family(binomial)` and may be either a positive whole number or a variable. For example, if all the denominators were 10 you would enter `family(binomial 10)`, whereas if they were stored in a variable called `denoms` you would type `family(binomial denoms)`. `family(binomial)` is permitted, and is short for `family(binomial 1)`.
4. The syntax for specifying power link functions now requires you to put the value # of the power after the `power` phrase in `link(power)`. The `power(#)` option has been dropped. For example, `link(power 0.5)` gives a square-root link function.
5. The `exposure(`*exposure_var*`)` option has been renamed `lnoffset(`*lnoffset_var*`)` to indicate more clearly what it is doing. The term "exposure" is somewhat specific to epidemiology, whereas a log offset may be used appropriately in other circumstances too.

Options

`family(...)` specifies the distribution of *depvar*; `family(gaussian)` is the default. See description above.

`link(...)` specifies the link function; the default is dependent on the `family()` specified. See description above.

`noconstant` specifies that the linear predictor has no intercept term, thus forcing it through the origin on the scale defined by the link function.

`scale(x2|dev|#)` overrides the default scale parameter. By default, `scale(1)` is assumed for the discrete distributions (binomial, Poisson, negative binomial) and `scale(x2)` is assumed for the continuous distributions (Gaussian, gamma, inverse Gaussian).

`scale(x2)` means the scale parameter is set equal to the Pearson chi-square (or generalized chi-square) statistic divided by the residual degrees of freedom, as recommended by McCullagh and Nelder (1989) as a good general choice for continuous distributions

`scale(dev)` estimates the scale parameter as the deviance divided by the residual degrees of freedom; this provides an alternative to `scale(x2)` for continuous distributions and over- or under-dispersed discrete distributions.

`scale(#)` sets the scale parameter to #. For example, using `scale(1)` in `family(gamma)` models results in exponential-errors regression. Additional use of `link(log)` rather than the `family(gamma)` default `power(-1)` essentially reproduces Stata's `ereg` command if all the observations are uncensored.

`[ln]offset(`*varname*`)` specifies that *varname* contains values for an offset that is to be added to the linear predictor. `offset()` specifies the values directly. `lnoffset()` specifies the exponentiated values (logs of *varname* are added to the linear predictor). `lnoffset()` is most useful with Poisson-like data where *varname* records the person-years of exposure to some hazard and *depvar* records the observed number of "deaths."

`k(#)` may be specified only with `family(nbinomial)` models and, in such models, `k(1)` is the default. The value of `k()` enters the variance and deviance functions; typical values for `k()` in real data lie between .01 and 2. Negative binomial models are often used for data with an overdispersed Poisson distribution. To use `glmr` to estimate such models, one searches for a `k()` that results in the deviance-based dispersion being approximately 1. In the case of `link(log)`, use of the `nbreg` command is preferable since `nbreg` will estimate the entire model including `k()` by maximum likelihood and report appropriate confidence intervals; see the comments by Rogers (1993). For links other than log (including the canonical link), you will need to use `glmr`.

`disp(#)` multiplies the variance of *depvar* by # and divides the deviance by #. The resulting distributions are members of the so-called quasi-likelihood family; see McCullagh and Nelder (1989) for a detailed description. The option may be appropriate for use with moderately overdispersed binomial or Poisson data to adjust the standard errors of the regression coefficients, which otherwise are too small.

`frvars` adds three new variables to the data set:

`_mu`	the expected value of *depvar* according to the fitted model;
`_eta`	the estimated linear predictor;
`_dres`	the scaled deviance residuals.

Note: this option is provided as a stop-gap measure until a more sophisticated revised `gpredict` command for use after `glmr` is implemented.

Deviance residuals `_dres`, which are scaled by dividing by the square root of the scale parameter (see `scale()` above), are recommended by McCullagh and Nelder (1989) and by others as having the best properties for examining goodness of fit of a GLM. For example, they are approximately normally distributed if the model is correct. They may be plotted against the fitted values (`_mu` or `_eta`) or against a covariate to inspect the model's fit. Several other types of residuals—not yet implemented—are believed to be better for certain specific purposes and these will be included in the forthcoming revised `gpredict` command.

`eform` displays the exponentiated coefficients and corresponding standard errors and confidence intervals as described in [7] maximize. For `family(binomial) link(logit)` (i.e., logistic regression), exponentiation results in odds ratios; for `family(poisson) link(log)` (i.e., Poisson regression), exponentiated coefficients are rate ratios.

`level(#)` specifies the percent coverage for confidence intervals of the coefficients; see [4] estimate.

`iterate(#)` specifies the maximum number of iterations allowed in estimating the mode; `iterate(50)` is the default. You should rarely need to specify `iterate()`.

`ltol(#)` specifies the convergence criterion for the change in deviance between iterations; `ltol(1e-8)` is the default. While lower values would theoretically yield more precise numerical results, in practice 10^{-8} is believed to be small enough to approach machine precision.

`init(`*varname*$_\mu$`)` allows you to use the variable *varname*$_\mu$ as the initial estimate for the mean of *depvar*. This could be useful with models that produce convergence difficulties, for example, `family(binomial)` models with power or odds-power (`opower`) links.

Example

Users unfamiliar with the GLM concept may wonder why (or even whether) such models are useful. I hope to illustrate the flexibility of GLMs using a wider range of analyses in a later insert. For now, I give a single example of an analysis which cannot be carried out using standard Stata 3.1 commands.

The data in the file `flour.dta` (supplied on the STB-18 disk) are taken from an early insecticide experiment, and are given by Pregibon (1980). The variables are `ldose`, the log dose of insecticide, `n`, the number of flour beetles subjected to each dose and `r` the number killed. The aim of the analysis is to estimate a dose–response relationship between p, the proportion killed, and X, the (log) dose. Figure 1 shows the relationship between the proportion killed $\big($estimated as `p`=(`r`+0.5)/(`n`+1)$\big)$ and `ldose`.

The curve is sigmoid in shape, and possibly somewhat asymmetric. An obvious first attempt at modeling is linear logistic regression of p on `ldose`, that is to say, to take the logit of p and to represent the dose–response curve as a straight line in X:

$$\log\Big(p/(1-p)\Big) = \beta_0 + \beta_1 X$$

An alternative model, which gives asymmetric sigmoid curves for p, involves the so-called complementary log-log or cloglog function:

$$\log\Big(-\log(1-p)\Big) = \beta_0 + \beta_1 X$$

Figure 2 shows the relationship between the proportion killed, transformed by the logit and cloglog functions, and log dose. The cloglog transformation has resulted in a straight line. However, since the observations have a binomial distribution, it is not valid to use linear regression to estimate the parameters of the model. The logistic regression model can be estimated by using Stata's `blogit` command:

```
. blogit r n ldose
Logit Estimates                                   Number of obs =    481
                                                  chi2(1)       = 272.97
                                                  Prob > chi2   = 0.0000
Log Likelihood = -186.23539                       Pseudo R2     = 0.4229
------------------------------------------------------------------------------
_outcome |      Coef.   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
   ldose |   34.27034   2.912141     11.768   0.000       28.56265    39.97803
   _cons |  -60.71747   5.180713    -11.720   0.000      -70.87149   -50.56346
------------------------------------------------------------------------------
```

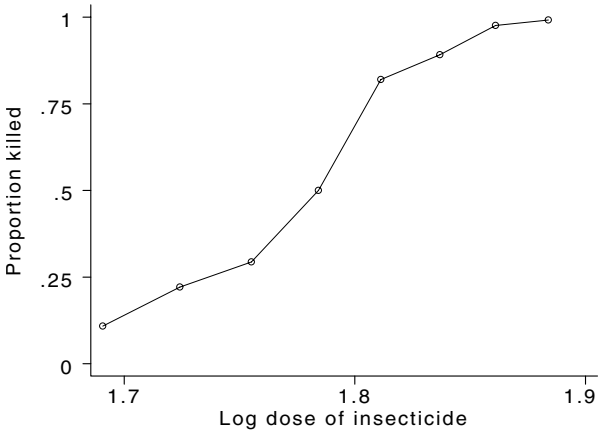

Figure 1

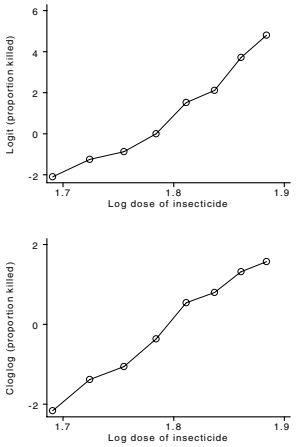

Figure 2

Here is the equivalent analysis using `glmr`:

```
. glmr r ldose, fam(bin n) link(l)
Iteration 1 : deviance =   11.4517
Iteration 2 : deviance =   11.2325
Iteration 3 : deviance =   11.2322
Iteration 4 : deviance =   11.2322
Residual df  =         6                  No. of obs =         8
Pearson X2   =   10.0268                  Deviance   =  11.23221
Dispersion   =  1.671133                  Dispersion =  1.872035
Binomial distribution, logit link
------------------------------------------------------------------------------
       r |      Coef.   Std. Err.       t     P>|t|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
   ldose |   34.27034   2.912135     11.768   0.000       28.56259    39.97809
   _cons |  -60.71748   5.180703    -11.720   0.000      -70.87159   -50.56336
------------------------------------------------------------------------------
```

The logit link `link(l)` need not in fact be specified, as it is the default link for the binomial family. Notice that `blogit` quietly expands the data set to 481 observations (the sum of the binomial denominators), whereas `glmr` works slightly differently and reports only 8 observations (the number of (`r`,`n`) pairs). Nevertheless, the parameter estimates from the two approaches are virtually identical.

There is no equivalent Stata command to estimate the cloglog model. However, `glmr` can do it:

```
. glmr r ldose, fam(bin n) frv link(c)
Iteration 1 : deviance =    3.5692
Iteration 2 : deviance =    3.4466
Iteration 3 : deviance =    3.4464
Iteration 4 : deviance =    3.4464
Residual df  =         6                  No. of obs =         8
Pearson X2   =  3.294678                  Deviance   =  3.446423
Dispersion   =   .549113                  Dispersion =  .5744038
Binomial distribution, cloglog link
------------------------------------------------------------------------------
```

```
       r |      Coef.   Std. Err.       t     P>|t|       [95% Conf. Interval]
---------+------------------------------------------------------------------
   ldose |   22.04118   1.799364     12.249   0.000        18.51445    25.56791
   _cons |  -39.57233   3.240289    -12.213   0.000       -45.92326   -33.2214
------------------------------------------------------------------------------
```

The `frvars` option used above creates three new variables: `_eta`, the "linear predictor" (here $\beta_0 + \beta_1 X$); `_mu`, the mean of the dependent variable; and `_dres`, the "scaled deviance residuals." The definition of deviance residuals varies according to the distribution, but if the model is correct they have approximately a normal distribution with mean 0 and standard deviation 1. They can be plotted against X or against the linear predictor in order to investigate the fit of the model graphically.

Figure 3 shows the observed and fitted proportions killed according to the cloglog-based model. The fit looks satisfactory. Figure 4 shows the scaled deviance residuals from the cloglog and logistic models, the latter also obtained by using `glmr`.

It is clear from the plots that the logit model gives the worse fit. Furthermore, its deviance is much higher, 11.23 compared with 3.45 for the cloglog model.

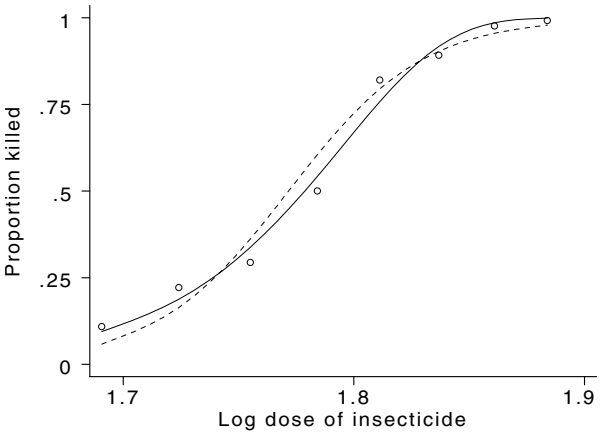

Figure 3

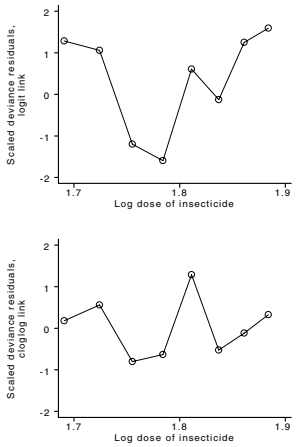

Figure 4

References

Baker, R. J. and J. A. Nelder. 1985. *The Generalised Linear Interactive Modelling System, release 3.77.* Oxford: Numerical Algorithms Group.

Hilbe, J. 1993a. sg16: Generalized linear models. *Stata Technical Bulletin* 11: 20–28.

——. 1993b. sg16.3: Quasi-likelihood modeling using an enhanced `glm` command. *Stata Technical Bulletin* 16: 6.

McCullagh, P. and J. A. Nelder. 1989. *Generalized Linear Models.* 2d ed. New York: Chapman & Hall.

Pregibon, D. 1980. Goodness of link tests for generalized linear models. *Applied Statistics* 29: 15–24.

Rogers, W. H. 1993. sg16.4: Comparison of `nbreg` and `glm` for negative binomial. *Stata Technical Bulletin* 16: 7.

sg22.1	Comment on Royston's revision of `glm`

Joseph Hilbe, Dept. of Sociology, Arizona State University
FAX: 602-860-1446, EMAIL: atjmh@asuvm.inre.asu.edu

I am pleased that Patrick Royston has discovered several errors that I overlooked in preparing the `glm` command (Royston 1994). Fortunately, these errors are easily changeable. His revision to `glm`, `glmr`, entails certain philosophical differences. Since many people have been using the `glm` command this past year, let me address some of these issues.

With respect to errors: the Poisson deviance is indeed incorrect if the response variable contains 0's. A one line fix ("`replace` `dev´=`mu´ if `y´==0`") following the Poisson deviance function line will remedy the matter. Standardized residuals for Gaussian data also need to be altered. I have compared output with SAS, SPLUS, XploRe, GAIM, GLIM4, and Xlisp-Stat. `glm` appears consistent in all other respects.

After some discussion with potential users, I intentionally wrote the program with no base default; i.e., '`glm yvar xvar`' will produce an error. We thought that at least a family should be declared. Poisson, gamma, and inverse Gaussian models do have canonical link defaults. However, the binomial models require one to specify a logit, probit, cloglog, or power link. In keeping with my pedagogic interest in using power links where possible, I forced all Gaussian models to use the power link form. This is perhaps inconvenient for the identity-linked normal model; but Stata's `fit` and `regress` commands, together with their slew of diagnostics, are certainly more powerful than the GLM approach for regular regression requirements. I suspect that all normal identity-linked regression will be modeled with Stata's in-house commands.

In `glm`, each GLM family has access to all power links. Some may be inappropriate for use; but they are available. SAS, GLIM4, and XploRe have also now taken this approach. Peter Lane, the author of the GLM facilities in GENSTAT, tells me that the next version will do the same. Here is an example why this is a wise tactic. Someone mentioned to me at one time that binomial power links would never have real life applications. However, I found Becker using the binomial log link for dealing with chain-binomial models (Becker 1989, p. 39). At the time he was forced to use GLIM's "`OWN`" command since it did not then allow for these types of models. I thought it best to let the user decide how best to deal with the appropriateness of links for the models being explored. Realize that writing `glm` from scratch was a rather monumental task. User input as the program traversed through different versions has been most helpful—and has been reflected in the command as it currently stands. At this point, error fixes, enhancements, modifications, and so forth are generally rather easy to accommodate.

Royston is perfectly correct in stating that χ^2-based p-values for the deviance and χ^2 summary statistics may not make sense for the continuous distributions. Rather than simply ignore them as I do, perhaps it may be best to delete them altogether for these types of models. However, realizing that most users will probably be using the binomial and Poisson (and now negative binomial) models, I simply attempted to standardize output.

I did note that when using the `if` or `in` options, an incorrect number of cases is displayed on output. This was not a problem in earlier versions and I am not sure how it crept into the current program. A simple "`` `if´ `in´ ``" added to line 92 of the code fixes the problem. The corrected `glm` is on the new STB diskette.

I like some of what Royston did with the `glm` command, especially incorporating the group option into the binomial family option and the power value into the link option. This method is directly taken from GLIM and may be more comfortable for users who have come from that background. I did not feel it necessary to follow GLIM's lead. Some may also like the default logit link for binomial models. I do not. When comparing logit with probit and cloglog models, all one needs to do in `glm` is type over the link on the command line. For me this is easier. This is exhibited in Royston's example where the `glm` and `glmr` commands for dealing with the same data are respectively:

```
glm r n ldose, f(bin) g l(c)
glmr r ldose, fam(bin n) link(c)
```

The output is nearly the same, except for the noticeable lack of p-values. To compare the model with a probit or logit specification one simply types a `p` or `l` in place of `c`. Royston has us do the same, except when you start with a logit model using the default. Then it's not as easy. But I would hope that anyone who is sophisticated enough to be using GLM methodologies will not find any of this very difficult.

P-values as discussed above have substantial modeling use when examining binomial, Poisson, and negative binomial data. They should not be deleted; nor should users be deprived of accessing any power link they desire. Except for fixing several above mentioned problem areas, making the listed modifications, and taking away several important modeling capabilities, Stata users will find that Royston has constructed `glmr` to be essentially the same `glm` program they have been using. The logic of the program is the same and the output a close approximation.

A `glm`-type program does greatly extend Stata capabilities and, from the communications I have received, has thus far enjoyed substantial use. The negative binomial addition to `glm` has been receiving the most attention—as far as I know, it was the first general GLM-type NB implementation available. I have provided the mathematical and modeling justification elsewhere (Hilbe 1994a) and have also prepared an article on its use that appears in this issue of the STB (Hilbe 1994b).

The original philosophy of the STB was to allow a forum in which Stata users may express opinions related to Stata, offer suggestions, or write programs which they may share with other users. In my opinion, the basic point of the STB is to enhance collegiality and to help Stata grow into a comprehensive statistical modeling tool. It is was not intended as a forum for competition. As always, I encourage any related comments or suggestions from users of the `glm` program as well as from those who require GLM capabilities.

References

Becker, N. 1989. *Analysis of Infectious Disease Data*. New York: Chapman & Hall.

Hilbe, J. 1994a. Log negative binomial regression as a generalized linear model, Technical Report 1, Committee on Statistics, Graduate College, Arizona State University.

——. 1994b. sg16.5: Negative binomial regression. *Stata Technical Bulletin* 18: 2–5.

Royston, P. 1994. sg22: Generalized linear models: revision of `glm`. *Stata Technical Bulletin* 18: 6-11.

sg22.2	Certifications of `glmr`

William Gould, Stata Corporation, FAX 409-696-4601

I have tested `glmr` by comparing its results to those produced by other Stata commands and, so far, I can report that `glmr` produces results that match Stata's other commands where it should. `glmr`'s `if` and `in` filters have been proven to work, `glmr` deals with missing values correctly, and all weighted estimates appear correct.

In summarizing the tests, I use the following shorthand:

tol. A squared measure of difference between two vectors or matrices defined as $\mathrm{trace}\big((\mathbf{A}-\mathbf{B})'(\mathbf{A}-\mathbf{B})\big)$.

coef. tol. The squared measure of difference between two coefficient vectors; coef. tol. 1×10^{-10} means the squared measure of tolerance was less than 1×10^{-10} in the example given.

VCE tol. The squared measure of difference between two variance–covariance matrix of the estimators; VCE tol. 1×10^{-10} means the squared measure of tolerance was less than 1×10^{-10} in the example given.

Exactly the same. The squared measure of tolerance is zero.

ETSR. "Exactly the same results," meaning exactly the same with respect to the coefficient vector and VCE matrix and exactly the same in terms of every result reported.

ETSRA..S. "Exactly the same results as dropping and estimating on the nonmissing subsample," meaning ETSR under the conditions stated.

The tests results are

1. Comparing `glmr` without the `family()` or `link()` options ("`glmr` linear regression") to `regress` resulted in identical results, coef. tol. 3×10^{-27}, VCE tol. 9×10^{-15}.
2. `glmr` linear regression with an `if` produces ETSR as dropping the excess observations and reestimating with `glmr`.
3. `glmr` linear regression with missing values among the independent variables produces ETSRA..S.
4. `glmr` linear regression with some missing values of the dependent variable produces ETSRA..S.
5. `glmr` linear regression with missing values of the dependent and independent variables produces ETSRA..S.
6. `glmr` linear regression with `aweight`s produces the same results as `regress`, coef. tol. 8×10^{-15} and VCE tol. 2×10^{-15}.
7. `glmr` linear regression with `aweight`s and `if` produces the same results as `regress`, coef. tol. 4×10^{-18} and VCE tol. 3×10^{-11}. (These differ from the tolerances reported in 6 only because the data was different.)
8. `glmr` linear regression with `aweight`s and missing values among the independent variables produces ETSRA..S.
9. `glmr` linear regression with `aweight`s and missing values among the dependent variable produces ETSRA..S.
10. `glmr` linear regression with `aweight`s and missing values among the weighting variables produces ETSRA..S.
11. `glmr` linear regression with `aweight`s and missing values among the dependent, independent, and weighting variables produces ETSRA..S.
12. `glmr` linear regression with `fweight`s produces ETSR as expanding the data and estimating without weights.
13. `glmr family(poisson)` (`link(log)` implied; "`glmr` Poisson regression") produces the same results as `poisson`, coef. tol. 4×10^{-17} and VCE tol. 4×10^{-17}.
14. `glmr` Poisson regression with `fweight`s produces ETSR as expanding the data and estimating without weights.

15. `glmr family(binomial)` (`link(logit)` implied; "`glmr` logistic regression") produces the same results as `logit`, coef. tol. 8×10^{-16} and VCE tol. 3×10^{-10}.

16. `glmr` logistic regression with `aweight`s produces the same results as `logit`, coef. tol. 5×10^{-14} and VCE tol. 9×10^{-11}.

17. `glmr` logistic regression with `fweight`s produces the same results as expanding the data and estimating without weights, coef. tol. 2×10^{-27} and VCE tol. 2×10^{-29}. Results are *exactly the same* for all other results reported; it is surprising but not concerning that all results are not ETSR.

18. `glmr` logistic regression with some `fweight`s of 0, missing, and positive integers produces the same results as expanding the data, dropping the 0-weight cases, and estimating without weights, coef. tol. 3×10^{-28} and VCE tol. 2×10^{-29}. Results are *exactly the same* for all other results reported. As in 17, it is surprising (but not concerning) that all results are not ETSR.

19. `glmr family(binomial) link(probit)` ("`glmr` probit regression") produces the same coefficient vector as `probit` (tol. 3×10^{-8}) but a slightly different VCE (tol. 2×10^{-2}).

20. `glmr` probit regression with `aweight`s produced the same coefficient vector as `probit` (tol. 1×10^{-10}) but a slightly different VCE (tol. 8×10^{-3}). This is not unexpected; probit is not the canonical link for the binomial family.

21. `glmr` probit regression with `fweight`s produced the same coefficient vector as `probit` (tol. 1×10^{-10}) but a slightly different VCE (tol. 2×10^{-3}), again as expected; see 20.

22. `glmr family(nbinomial link(log)` ("`glmr` negative binomial regression) produced the same results as `nbreg`, coef. tol. 2×10^{-11} and VCE tol. 2×10^{-13}. The alpha reported by `nbreg` was used to set `glmr`'s `k()` option.

23. `glmr` negative binomial regression with `lnoffset(`exposure`)` produced the same coefficient vector as `nbreg` (tol. 5×10^{-10}) and a slightly different VCE matrix (tol. 6×10^{-5}). The difference in the VCE is not concerning, see Rogers (1993).

24. `glmr` negative binomial regression with `offset(`lnexposure`)` produced the same coefficient vector as `nbreg` (tol. 5×10^{-10}) and a slightly different VCE matrix (tol. 6×10^{-5}). The difference in the VCE is not concerning, see Rogers (1993).

25. `glmr family(gamma) link(log)` ("`glmr` exponential regression") with option `scale(1)` produced the same coefficient vector as `ereg` (tol. 1×10^{-10}) but a different VCE, tol. 1.5. This is not unexpected; log is not the canonical link for the gamma family.

26. Various nonsense requests, such as attempting to estimate on 1 or 2 observations, etc., produced reasonable error messages.

27. `glmr family(binomial` *varname*`) link(logit)` produced the same result as `blogit`, coef. tol. 9×10^{-17} and VCE tol. 2×10^{-16}.

28. `glmr family(binomial` #`) link(logit)` produced the same result as `blogit`, coef. tol. 4×10^{-15} and VCE tol. 6×10^{-11}.

29. `glmr family(binomial` *varname*`) link(probit)` produced the same result as `bprobit`, coef. tol. 5×10^{-14} and VCE tol. 7×10^{-7}.

30. `glmr family(binomial` #`) link(probit)` produced the same result as `bprobit`, coef. tol. 2×10^{-14} and VCE tol. 3×10^{-6}.

31. `glmr family(binomial` *varname*`)` when varname contained odd values, such as negative numbers or numbers smaller than the dependent variable, produced reasonable error messages.

The test script, a Stata do-file, is provided on the STB diskette.

References

Rogers, W. H. 1993. sg16.4: Comparison of `nbreg` and `glm` for negative binomial. *Stata Technical Bulletin* 16: 7.

sg23	Semi-graphical determination of Gaussian components in mixed distributions

Isaías Hazarmabeth Salgado-Ugarte, Makoto Shimizu, and Toru Taniuchi,
University of Tokyo, Faculty of Agriculture, Department of Fisheries, Japan
FAX (011)-81-03-3812-0529, EMAIL isalgado@tansei.cc.u-tokyo.ac.jp

Mixed distributions—mixtures of normal distributions—arise frequently in biological and ecological data, and the analysis of these mixed distributions is an important topic in fisheries science, particularly when studying species in subtropical and tropical areas. Many samples of fish lengths for these species exhibit multiple modes. Since Petersen (1892), it has been recognized that these modes are evidence that the sample contains a mixture of several generations of fish. Many species have a short spawning period that occurs once a year. In these cases, the modes can be identified with a particular generation. The correspondence between modes and age groups is particularly clear for younger age groups, but is less marked in the older ones due to overlapping of the size distributions as each generation approaches the maximum size of the species.

The analysis of size frequency data supplies information on age and growth that is necessary for stock assessment. The difficulty of estimating ages by other more direct means, such as hard-parts reading or mark-recapture studies, helps to account for the importance of size frequency analysis. Even when direct methods are available, size frequency analysis provides validation of other estimates.

A variety of methods, both graphical and analytical, have been proposed for analyzing mixed frequency distributions into their individual components. These methods include the use of probability paper (Harding 1949; Cassie 1954), graphical trial-and-error parabola fitting (Tanaka 1962), logarithmic differences (Bhattacharya 1967), and maximum likelihood (Hasselblad 1972). Other, more sophisticated procedures have been proposed for the case where the age-group modes describe a growth curve (Macdonald and Pitcher 1979; Schnute and Fournier 1980; Liu et al. 1989). The general problem, where the number of modes is not known in advance, remains an active topic of research.

Several computer programs are available for applying these different methods to one or more samples: ELEFAN (Gayanilo et al. 1989), LFSA (Sparre et al. 1989), MIX (Macdonald and Green 1988, 1992), and MULTIFAN (Fournier et al. 1990). This insert presents Stata ado-files that apply Bhattacharya's semi-graphical method (1967) for estimating the Gaussian components in size frequency data. In Bhattacharya's approach, the analyst need not know the number of components in advance. Our ado-files incorporate some of the modifications and suggestions of Pauly and Caddy (1985), Sparre et al. (1989), and Erzini (1990).

The next section of this insert gives an informal introduction to Bhattacharya's method. Then the method is applied to Tanaka's (1962) data on porgy length frequencies. These data are well adapted to analysis by Bhattacharya's method. The next section of the insert demonstrates the application of the method to a more difficult data set, Goeden's (1978) measurements of coral trout length. In this example, we smooth the data using, first, a nonlinear resistant smoother and, second, an adaptive kernel estimator.

Bhattacharya's method

Bhattacharya's method is designed to estimate well-separated Gaussian components in mixed distributions. The simplest case of a single Gaussian component is depicted in Figure 1. This figure was produced by drawing 10,000 pseudo-random numbers from a normal distribution with mean zero and standard deviation 2.0. The real line then was divided into intervals a unit wide and centered on the integers, and the frequencies, the numbers of observations in each interval, were recorded. In this example, there were 1,942 observations between $-1/2$ and 1/2 (centered on 0), 1,768 observations between 1/2 and 3/2 (centered on 1), and so on. The approximately bell-shaped curve in Figure 1 is the frequency polygon for these data. (See Scott (1985a, 1992) for accounts of the theory and applications of frequency polygons and histograms.)

Consider the smooth bell-shaped curve that describes the ideal normal distribution from which these data were drawn. Consider, in particular, the derivative of this normal curve with respect to the x-axis. The slope of the curve is positive to the left of the mode, zero at the mode, and negative to the right of the mode. As Bhattacharya shows, the logarithmic derivative declines approximately linearly. The points in Figure 1 that describe a rough line with negative slope are the logarithmic differences of adjacent frequencies along our empirical bell-shaped curve.

Bhattacharya's method begins by plotting these logarithmic differences. If the Gaussian components are well-separated, the plot will exhibit negatively sloped linear segments, one segment for each component. If the components overlap heavily, it may be difficult to detect all the components. In this case, the dominant components can be estimated and subtracted from the data, then the remaining components can be more easily recognized and estimated.

There are four steps in Bhattacharya's method. First, the logarithmic differences of successive frequencies are calculated. Second, these differences are plotted and the components are picked out by eye. This step is the graphical part of the method. Third, the line segment corresponding to each segment is estimated. Bhattacharya develops formulas for estimating the parameters of each component—the mean, standard deviation, and frequency or proportion of the sample due to each component—from the angles and intercepts described by the negatively sloped line segments. Finally, an estimated frequency polygon is generated by adding the estimates of each of the Gaussian components. The method can be reapplied to the residuals to detect Gaussian components that are obscured by overlapping or dominant components.

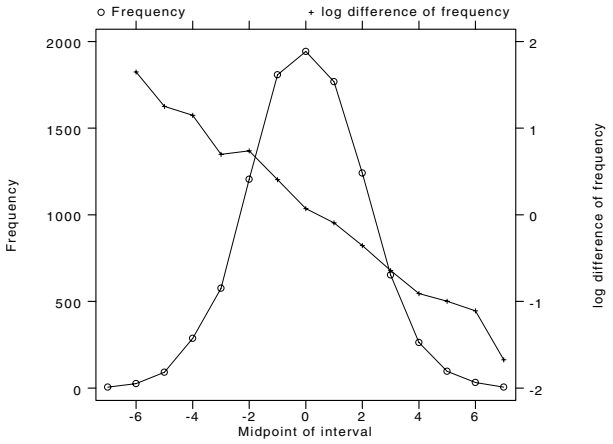

Figure 1: A single Gaussian component

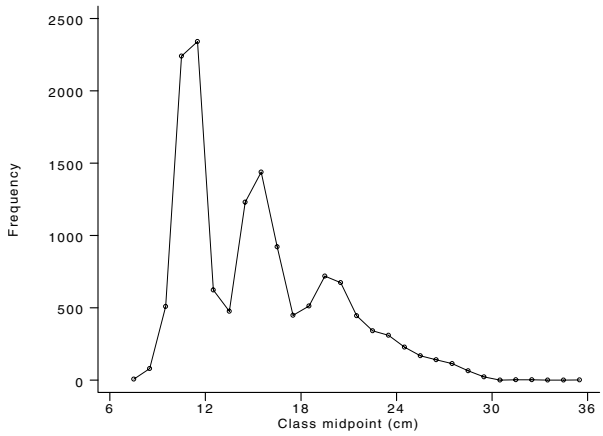

Figure 2: Frequencies of porgy length

Example: Analyzing the frequencies of porgy length

We have written four ado-files that correspond to the four steps in Bhattacharya's method. We demonstrate these ado-files on the well-known porgy (*Taius tumifrons*) data from Tanaka (1962). The measurements are grouped into 1 mm. intervals and the frequency of measurements in each interval is recorded. Table 1 lists the data and Figure 2 displays them as a frequency polygon. The data are stored in the file `porgy.dta` on the STB disk.

Table 1: Data

midpoint	frequency	midpoint	frequency	midpoint	frequency
7.5	7	17.5	448	27.5	114
8.5	79	18.5	512	28.5	64
9.5	509	19.5	719	29.5	22
10.5	2240	20.5	673	30.5	0
11.5	2341	21.5	445	31.5	2
12.5	623	22.5	341	32.5	2
13.5	476	23.5	310	33.5	0
14.5	1230	24.5	228	34.5	0
15.5	1439	25.5	168	35.5	1
16.5	921	26.5	140	36.5	0

The frequency polygon exhibits three distinct modes, then tails off. The modes for the shorter, thus younger, fish are more pronounced. The smoother tail on the right may result from overlapping modes.

The first step in Bhattacharya's method is to calculate logarithmic differences between successive frequencies. As a convenience, we have packaged this calculation in `diflogen.ado`. The syntax of `diflogen` is

`diflogen` *freqvar diflovar*

where *freqvar* is the existing variable that contains the frequencies (`freq` in this example) and *diflovar* is a new variable that will contain the logarithmic differences. We use the name `diflog` for this variable. Thus,

```
. diflogen freq diflog
```

The second step in Bhattacharya's method is to plot these logarithmic differences against the interval midpoints, searching for points that form negatively sloped straight lines. Each such straight-line segment indicates a separate Gaussian component. `bhatplot` displays such a graph, using observation numbers as plotting symbols. The syntax of `bhatplot` is

`bhatplot` *diflovar midpoivar* [`in` *range*]

Returning to our example, we type

```
. bhatplot diflog midpoi
```

The graph produced by `bhatplot` is shown in Figure 3. The `in` *range* option can be used to magnify the detail in any portion of the plot. We can distinguish at least four Gaussian components in Figure 3: one component is dominant in observations 3/5, one in observations 7/9, one in 12/14, and one in 20/22. There may be another component obscured in the jumble of points in observations 15/19.

The third (modified) step of Bhattacharya's method is to estimate the slopes and intercepts of the line segments corresponding to each component and to derive the parameters of the Gaussian components from these slope and intercept estimates. `bhatmesd` calculates these estimates, one component at a time. The syntax of `bhatmesd` is

`bhatmesd` *freqvar diflovar midpoivar* [`in` *range*]

The optional `in` *range* restricts the estimation to the range where one component is dominant. If this option is omitted, a single component is estimated using all the observations.

To estimate the first Gaussian component in the porgy data, we type

```
. bhatmesd freq diflog midpoi in 3/5
R-square = 0.9998          Adj R-square = 0.9996
Mean = 11.0480
s.d. = 0.8443
component size = 5783
```

The component size is the estimated number of observations attributable to this component. `bhatmesd` also displays a graph of the observed frequencies and the estimated Gaussian component. This graph is shown in Figure 4.

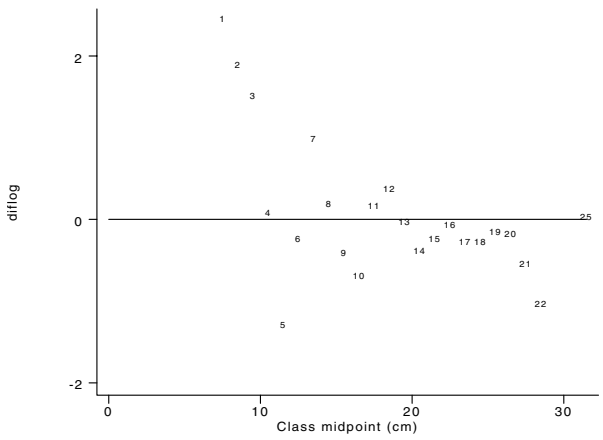

Figure 3: Bhattacharya's plot

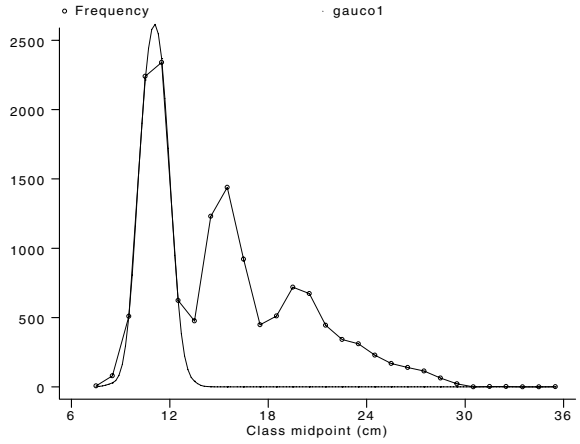

Figure 4: The first Gaussian component

If the fit is satisfactory (as in this example), we proceed to the fourth step of Bhattacharya's method, that is, we generate a new variable that contains the estimated Gaussian frequencies attributable to this component. We use `gaussgen` to calculate these estimates. The syntax of `gaussgen` is

`gaussgen` *midpoivar meanval sdval freqval comgaufre*

The *meanval*, *sdval*, and *freqval* are the estimates calculated by `bhatmesd`. For this example, we type

```
. gaussgen midpoi 11.048 0.8443 5783 gauco1
```

`gauco1` is a new variable that contains the estimated frequencies attributable to the first Gaussian component.

Now we estimate the other four components we spotted in the plot of logarithmic differences. To keep the example from growing too long, the graphs produced by `bhatmesd` are omitted in the listing below.

```
. bhatmesd freq diflog midpoi in 7/9
R-square = 0.9939          Adj R-square = 0.9878
Mean = 15.3153
s.d. = 1.1971
component size = 4498
(graph omitted)
. gaussgen midpoi 15.3153 1.1971 4498 gauco2
. bhatmesd freq diflog midpoi in 12/14
R-square = 0.9980          Adj R-square = 0.9960
Mean = 19.8759
s.d. = 1.6295
component size = 2988
(graph omitted)
. gaussgen midpoi 19.876 1.629 2988 gauco3
. bhatmesd freq diflog midpoi in 20/22
R-square = 0.9937          Adj R-square = 0.9875
Mean = 26.5694
s.d. = 1.5229
component size = 533
(graph omitted)
. gaussgen midpoi 26.569 1.523 533 gauco5
```

The last Gaussian component was labeled "5" rather than "4" because we suspect there is an obscured component between the last two estimated components. To search for this obscured component, we calculate the residuals from this fit. First we generate a new variable which is the sum of the estimated Gaussian components estimated so far. This variable is the "fitted" value.

```
. gen sumgaco=gauco1+gauco2+gauco3+gauco5
```

Next we subtract this fitted value from the actual frequencies to obtain the residual frequencies. We impose the restriction that the residual frequencies must be nonnegative.

```
. gen freq2=max(0,freq-sumgaco)
```

Now we apply Bhattacharya's method to these residuals to see if there are any additional Gaussian components.

```
. diflogen freq2 diflog2
(information on missing values omitted)
. bhatplot diflog2 midpoi
```

This graph, shown in Figure 5, displays evidence of an additional Gaussian component in observations 16/18. (The missing observations correspond to intervals with residual frequencies of zero. The residuals naturally have a large number of intervals with frequencies of zero.)

```
. bhatmesd freq2 diflog2 midpoi in 16/18
R-square = 0.9932          Adj R-square = 0.9865
Mean = 23.6216
s.d. = 1.1371
component size = 638
(graph omitted)
. gaussgen midpoi 23.622 1.137 638 gauco4
```

We can finally graph the observed frequencies and all the estimated Gaussian components by typing

```
. graph freq gauco1 gauco2 gauco3 gauco4 gauco5 midpoi, xlabel ylabel c(lssssss) s(p.....)
```

A more polished version of this graph is shown in Figure 6. Table 2 compares our estimates to those obtained by different authors using different methods.

Table 2: Estimates

Parameter	Estimation method	Components				
		1	2	3	4	5
Means	**Salgado-Ugarte, et al.**	**11.048**	**15.315**	**19.876**	**23.622**	**26.569**
	Buchanan-Wollaston	11.05	15.32	19.85	23.58	26.82
	Cassie	11.02	15.33	19.85	23.46	26.92
	Tanaka	10.99	15.26	19.84	23.50	26.82
	Bhattacharya	11.03	15.28	19.86	23.62	26.62
	Akamine	11.0	15.3	19.7	23.5	27.2
	MacDonald and Green	11.00	15.30	19.70	23.45	27.26
Standard	**Salgado-Ugarte, et al.**	**.844**	**1.197**	**1.629**	**1.137**	**1.523**
deviations	Buchanan-Wollaston	.844	1.161	1.412	1.212	1.443
	Cassie	.76	1.15	1.32	1.29	1.54
	Tanaka	.8	1.2	1.4	1.2	1.4
	Bhattacharya	.81	1.13	1.60	1.07	1.47
	Akamine	.87	1.14	1.43	1.55	1.19
	MacDonald and Green	.83	1.10	1.39	1.62	1.12
Proportions	**Salgado-Ugarte, et al.**	**.4005**	**.3115**	**.2069**	**.0442**	**.0369**
	Buchanan-Wollaston	.4072	.3110	.1860	.0642	.0316
	Cassie	.4049	.3164	.1788	.0693	.0307
	Tanaka	.4007	.3194	.1873	.0598	.0328
	Bhattacharya	.4065	.3067	.2087	.0420	.0361
	Akamine	.411	.305	.183	.077	.024
	MacDonald and Green	.4106	.3056	.1787	.0827	.0224

The differences between our results and Bhattacharya's are negligible and are due to our slightly different implementation (see the notes on calculations below).

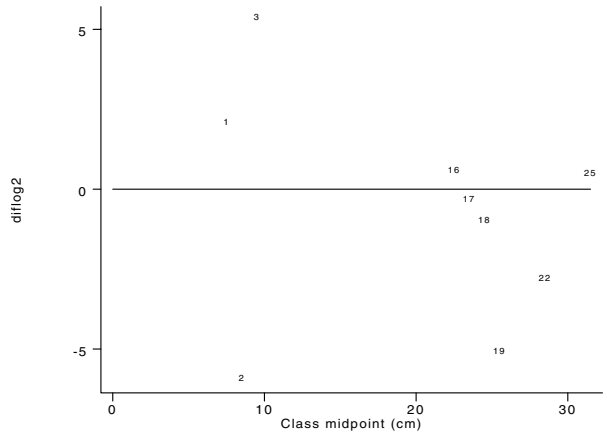

Figure 5: Log differences of residuals

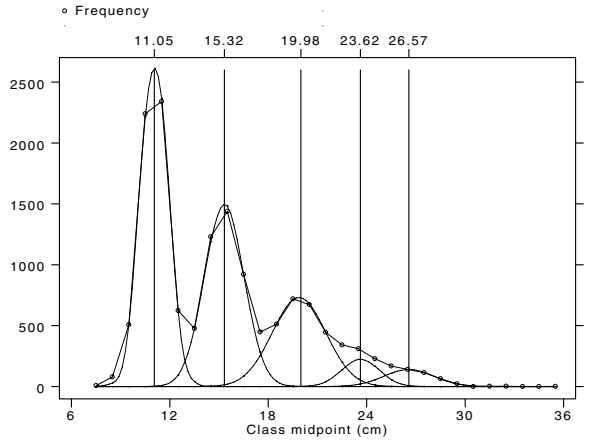

Figure 6: All the estimated components

Analyzing noisy data: nonlinear resistant smoothing

The frequency polygon for the porgy data is relatively smooth and the modes are clearly defined, the ideal case for Bhattacharya's method. With highly variable, or "noisy," data, the method breaks down. The logarithmic differences of a saw-toothed frequency polygon magnify the variability of the frequencies and obscure any negatively sloped segments. Bhattacharya's recommendation that narrow intervals be used for the frequency polygon tends to exacerbate this problem by increasing the variability of the individual frequencies.

Figure 7 displays the frequency polygon for the measurements of the lengths of coral trout (*Plectropomus leopardus*) obtained by Goeden (1978) on Heron Island in Australia. These data are stored in the file `trout.dta` on the STB disk. The 319 length observations were originally grouped in 5 mm. intervals. While these data show several clear modes, the frequencies are more variable than in the porgy data. The trout length frequencies exhibit the saw-toothed appearance characteristic of noisy data, in contrast to the much smoother appearance of the porgy length frequency polygon.

Bhattacharya's method is ineffective when applied unmodified to noisy data such as these. Figure 8 shows Bhattacharya's plot for the trout length data. It is difficult to discern any negatively sloped line segments in this plot, despite the obvious modes in the frequency polygon.

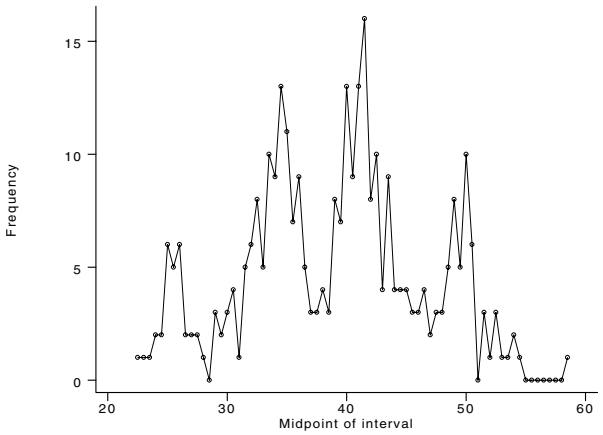

Figure 7: Frequencies of trout length

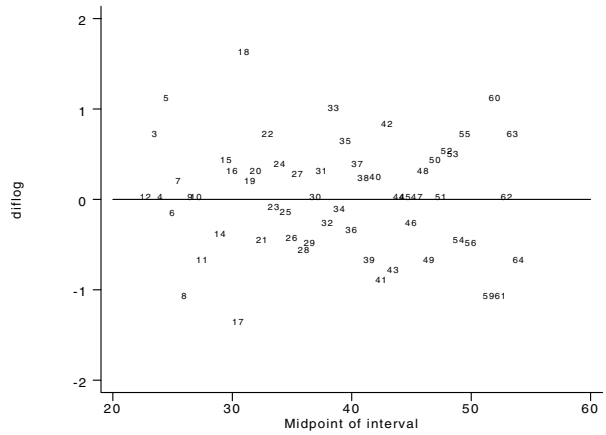

Figure 8: Bhattacharya's plot for trout lengths

The traditional method for reducing noise in histograms and frequency polygons is to increase the interval width. This width can be thought of as a smoothing parameter (Tarter and Kronmal 1978; Silverman 1986). However, increasing the interval width reduces information. Laurec and Mesnil (1987), for example, concluded that increasing the class interval length worsened mortality rate estimates derived from length frequency distributions. They recommended applying an explicit smoother to the data instead of increasing the interval width. In keeping with this suggestion, the ELEFAN program (Gayanilo, et al. 1989) includes routines to calculate moving averages of lengths 3 and 5 as a way of smoothing length frequency data. Taylor (1968) introduced the use of moving averages with length frequency data in connection with a trial and error procedure for Gaussian component identification.

Moving averages have disadvantages, however; they are extremely sensitive to outliers, they shift the peaks and valleys in the data, and they blur rapid transitions (Davis 1971, Tukey 1977, Velleman 1980, 1982, Goodall 1990, Hansen 1991, Salgado-Ugarte 1992). Thus, we prefer to smooth frequency values with a nonlinear resistant smoother recommended by Velleman (1982) for general use. The smoother is called 4254EH,twice. We use a modified version of the ado-files `sm4253eh` and `smtwice` written by Salgado-Ugarte and Curts-García (1992,1993) to calculate the smoothed frequencies. The same results can be obtained with Stata's `smooth` command ([5s] smooth).

```
. sm4253eh freq smoo1
. smtwice freq smoo1 smofreq
```

Figure 9 displays the raw trout length frequencies (the circles) and the smoothed version of the same data (the smooth line). The smoothed frequencies exhibit the same modes in approximately the same locations as the original data, but the roughness of the original frequency polygon is gone.

Bhattacharya's plot for the smoothed frequencies appears in Figure 10. The body of the data appears compressed because the logarithmic differences of the right tail of the frequency polygon are outliers, but the graph shows distinct, negatively sloped linear segments nonetheless.

We apply Bhattacharya's method to the smoothed frequencies to estimate the Gaussian components. All the graphs are omitted to conserve space.

```
. diflogen smofreq diflosm
. bhatmesd smofreq diflosm midpoi in 4/9
R-square = 0.9890          Adj R-square = 0.9862
Mean = 25.5166
s.d. = 1.0793
component size = 27
. gaussgen midpoi 25.52 1.08 27 gauco1
. bhatmesd smofreq diflosm midpoi in 21/28
R-square = 0.9509          Adj R-square = 0.9427
Mean = 34.2356
s.d. = 1.7593
component size = 97
. gaussgen midpoi 34.24 1.76 97 gauco2
. bhatmesd smofreq diflosm midpoi in 33/42
R-square = 0.9975          Adj R-square = 0.9971
Mean = 41.1522
s.d. = 1.8897
component size = 113
. gaussgen midpoi 41.15 1.89 113 gauco3
. bhatmesd smofreq diflosm midpoi in 53/57
R-square = 0.9901          Adj R-square = 0.9868
Mean = 49.5867
s.d. = 1.3343
component size = 44
. gaussgen midpoi 49.59 1.33 44 gauco5
```

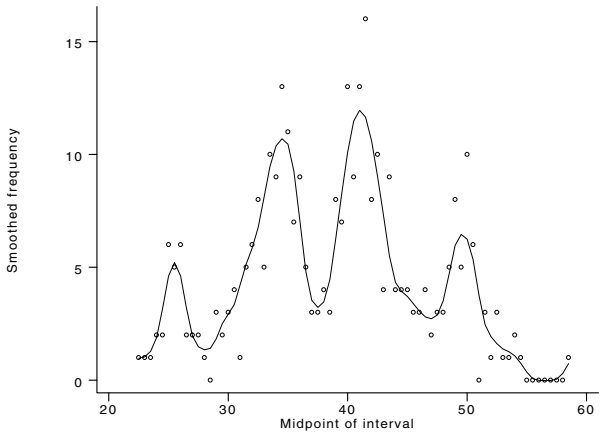

Figure 9: Smoothed frequencies

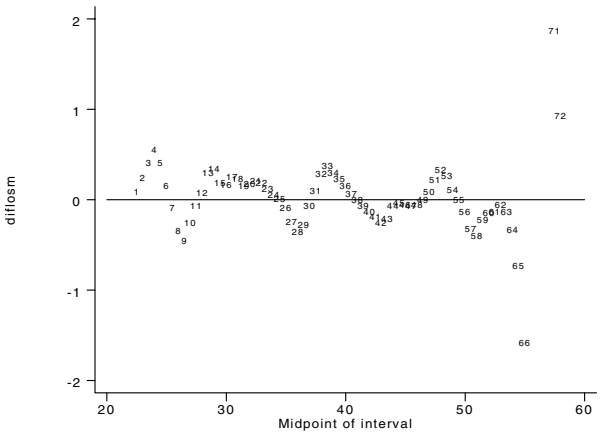

Figure 10: Bhattacharya's plot of the smoothed data

As in the previous example, smaller Gaussian components can be detected after the four dominant components are subtracted from the data. We estimate the remaining components below and graph them along with the raw and smoothed frequencies (Figure 11).

```
. gen sumgau=gauco1+gauco2+gauco3+gauco5
. gen adsmfre=max(0,smofreq-sumgau)
. diflogen adsmfre diflo2
```

```
. bhatmesd adsmfre diflo2 midpoi in 14/16
R-square = 0.9541          Adj R-square = 0.9083
Mean = 30.0343
s.d. = 1.2441
component size = 15
. gaussgen midpoi 30.03 1.24 15 gauco2a
. bhatmesd adsmfre diflo2 midpoi in 45/50
R-square = 0.9743          Adj R-square = 0.9679
Mean = 45.7868
s.d. = 1.1887
component size = 16
. gaussgen midpoi 45.79 1.18 16 gauco4
. bhatmesd adsmfre diflo2 midpoi in 62/65
R-square = 0.9528          Adj R-square = 0.9292
Mean = 53.4385
s.d. = 0.9793
component size = 6
. gaussgen midpoi 53.44 0.98 6 gauco6
. graph smofreq gauco1 gauco2a gauco2 gauco3 gauco4 gauco5 gauco6 midpoi,
> xlab ylab c(lsssssss) s(o.......)
```

Another approach: adaptive Gaussian kernel smoothing

The appearance of a histogram or, equivalently, a frequency polygon is sensitive to the placement of the origin (the left boundary of the first interval) and to the choices of the number and of the width of the intervals. Different selections of these parameters can give very different impressions of the distribution of a data set (Silverman 1986; Fox 1990; Salgado-Ugarte et al. 1993). Some authors recommend trying different numbers of intervals and interval widths to guard against misleading results (Sparre et al. 1989; Erzini 1990). No formal recommendations for choosing the placement of the origin have been proposed so far.

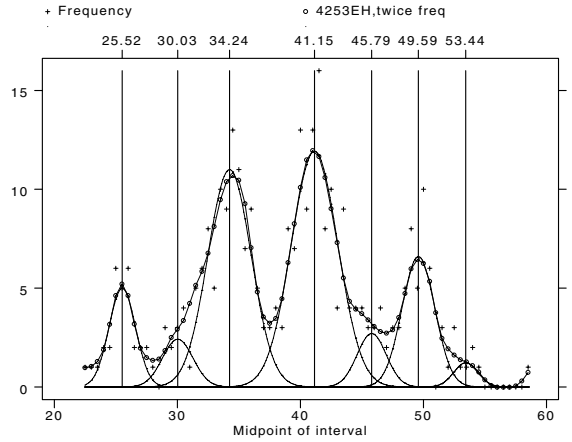

Figure 11: Estimated components

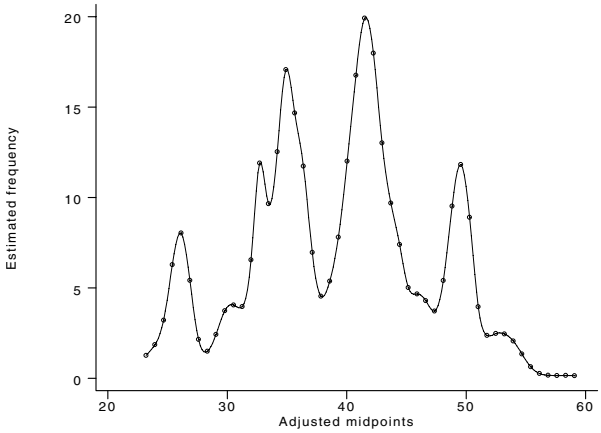

Figure 12: Smoothed frequencies

Statistical theory provides several approaches to choosing the number and width of intervals (Sturges 1926; Dixon and Kronmal 1965; Doane 1976; Velleman 1976; Scott 1979; Freedman and Diaconis 1981a, b; also see Emerson and Hoaglin 1983, Geiger 1991 and the *Stata Reference Manual, Release 3.1*, 1993, volume 1, pp. 206–208). The general rule of almost all the proposals is to determine the optimal number and width of intervals as a function of the sample size.

These techniques have all been developed for the case of a single underlying distribution. In mixed distributions, however, there are several Gaussian (or other) components, each with different parameters. The ideal number and width of intervals may be different for each component. Dominant components—components with many individuals—permit the use of a large number of small intervals; more sparsely populated components can support only a few, relatively wide intervals. The classical histogram uses a fixed interval width, hence it may do a poor job of portraying both dominant and lesser components. (See Fox 1990 for a discussion of this drawback of the histogram. Also see Wegman 1972 and Scott 1985b for interesting variations on the histogram). For example, applying Scott's (1985a) formulas to the trout data suggests that the frequency polygon group the data into seven intervals, each 50.36 mm. wide. These are too few intervals; they oversmooth the data and hide the multiple modes.

An alternative approach is the adaptive Gaussian kernel smoother described by Fox (1990) and Silverman (1986). This smoother does not depend on the placement of the origin, and it adjusts the interval width, making it narrow where observations are plentiful and wide where observations are scarce. As a result, the adaptive Gaussian kernel smoother reveals data details while simultaneously reducing noise.

We presented `adgkern`, a program to calculate adaptive Gaussian kernel smooths, in a previous insert (Salgado-Ugarte et al. 1993). It is beyond the scope of this insert to discuss this smoother in detail. The interested reader may consult our earlier insert for more information. The smoothed frequencies (`adgk25`) and the corresponding interval midpoints (`miadgk25`) are included in `trout.dta` along with the raw frequencies.

It is necessary to choose a smoothing parameter, h, to apply the adaptive Gaussian kernel smoother. We chose h by trial and error. We used the value suggested by Silverman's (1986) formula as an upper limit, then experimented with smaller values. The smoothed frequencies in Figure 12 were obtained with $h = 5$, our final choice. Note that the adaptive Gaussian kernel produces a smooth with more detail and greater separation of the modes than the smooth produced by 4253eh,twice.

The smoothed frequencies in Figure 12 were calculated by renormalizing the smooth to sum to the original number of observations. Then Bhattacharya's method was applied to obtain estimates of the Gaussian components. Again, we suppress all the graphs to conserve space.

```
. gen sumadgk=sum(adgk25)

. gen freqak5=adgk25*319/sumadgk(_N)

. diflogen freqak5 diflog5

. bhatmesd freqak5 diflog5 miadgk25 in 3/6
R-square = 0.9946          Adj R-square = 0.9919
Mean = 25.9906
s.d. = 0.9933
component size = 27

. gaussgen miadgk25 25.99 0.99 27 gauco1

. bhatmesd freqak5 diflog5 miadgk25 in 15/19
R-square = 0.9032          Adj R-square = 0.8709
Mean = 35.0485
s.d. = 1.5978
component size = 88

. gaussgen miadgk25 35.05 1.60 88 gauco2

. bhatmesd freqak5 diflog5 miadgk25 in 24/27
R-square = 0.9909          Adj R-square = 0.9864
Mean = 41.5637
s.d. = 1.5448
component size = 104

. gaussgen miadgk25 41.56 1.54 104 gauco3

. bhatmesd freqak5 diflog5 miadgk25 in 35/38
R-square = 0.9921          Adj R-square = 0.9881
Mean = 49.4266
s.d. = 1.0752
component size = 43

. gaussgen miadgk25 49.43 1.08 43 gauco5

. generate sumgauco = gauco1+gauco2+gauco3+gauco5
(23 missing values generated)

. generate freq2 = max(0,freqak5-sumgauco)

. diflogen freq2 diflog2

. bhatmesd freq2 diflog2 miadgk25 in 8/11
R-square = 0.9885          Adj R-square = 0.9828
Mean = 30.3391
s.d. = 1.2099
component size = 16

. gaussgen miadgk25 30.34 1.21 16 gauco2a

. bhatmesd freq2 diflog2 miadgk25 in 31/33
R-square = 0.9366          Adj R-square = 0.8732
Mean = 45.8540
s.d. = 1.1287
component size = 17
```

```
. gaussgen miadgk25 45.85 1.13 17 gauco4
. bhatmesd freq2 diflog2 miadgk25 in 41/43
R-square = 0.9993          Adj R-square = 0.9986
Mean = 53.0712
s.d. = 1.4614
component size = 12
. gaussgen miadgk25 53.07 1.46 12 gauco6
```

The estimated components are displayed in Figure 13.

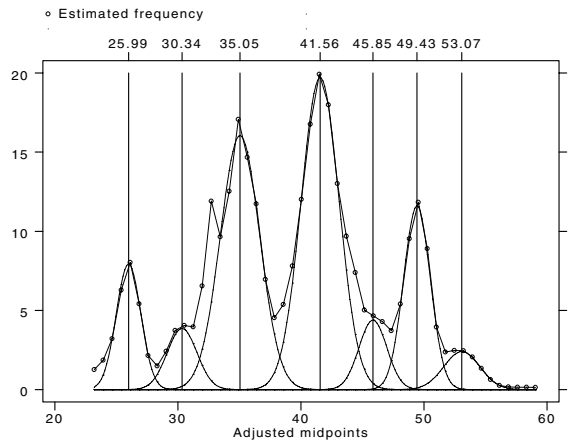

Figure 13: Estimated components

A comparison of the two smoothers

The results of both smoothing procedures are compared in Table 3. The estimated means are almost equal. The standard deviation and component size estimates tend to be larger when the histogram is smoothed using 4253eh,twice. The adaptive kernel estimator tends to produce more leptokurtic components in these data (the negatively sloped segments are steeper), and this factor accounts for the smaller standard deviation and size estimates compared to the results using 4253eh,twice.

Table 3: A comparison of estimates using different smoothers

	Mean		Standard deviation		Component size	
Gaussian component	4253EH, twice	Adaptive kernel	4253EH, twice	Adaptive kernel	4253EH, twice	Adaptive kernel
1	25.52	25.99	1.08	0.99	27	27
2a	30.03	30.34	1.24	1.21	15	16
2	34.24	35.05	1.76	1.60	97	88
3	41.15	41.56	1.89	1.54	113	104
4	45.79	45.85	1.19	1.13	16	17
5	49.59	49.43	1.33	1.08	44	43
6	53.44	53.07	0.98	1.46	6	12
				Total	318	307

As a practical matter, the data analyst is required to choose in advance the histogram parameters (origin, width, number of intervals) in order to use 4253eh,twice. These choices are avoided when using the adaptive Gaussian kernel smoother, moreover, this smoother adjusts the interval widths to account for variations in the concentration of data. The analyst must choose a smoothing parameter, however, in order to use the adaptive Gaussian kernel smoother.

As a final note, these smoothers produce similar estimates of the Gaussian components. As a consequence, either smoother should be acceptable if the estimates are to be used as initial values in a subsequent maximum-likelihood estimation of the component parameters.

Notes on the calculations

- `diflogen` calculates the difference of the natural logarithms of successive frequencies. In other words, the `diflog` is equivalent to typing

```
. generate diflovar = freqvar[_n+1] - freqvar
```

- When frequencies are zero, missing values will be generated. You may wish to add a small constant (1/6 for example) to the frequencies if there are many zeroes.
- `bhatmesd` uses `regress` to estimate the straight line described by the logarithmic differences of frequencies for a particular component, say the kth component. The mean (μ_k) and standard deviation (σ_k) of the kth component are estimated according to the following expressions (Pauly and Caddy 1985; Sparre et al. 1989, respectively):

$$\widehat{\mu}_k = 0.5d - a_k/b_k$$

and

$$\widehat{\sigma}_k = \sqrt{-d/b_k}$$

where d is the interval width and a and b are the intercept and slope, respectively, of the least square regression line. The size of the kth component is estimated by the expression (Bhattacharya 1967)

$$\widehat{N_k} = \sum y \Big/ \sum \widehat{P}$$

where the y are frequency values and P are the corresponding adjusted Gaussian probability values. The summation is carried out over the range specified by the `in` *range* option. In contrast, Bhattacharya (1967) only considered two points in the estimation of component size. If the linear trend is clear, the Gaussian component is relatively free of overlapping and this procedure provides reliable estimates of N_k. If points that deviate from the linear trend are included, the estimated component will not fit the frequencies well.

Acknowledgments

The first author is grateful to the Ministry of Education, Science and Culture of Japan and to the Universidad Nacional Autonoma de Mexico (FES Zaragoza and DGAPA) for their support.

References

Akamine, T. 1985. Consideration of the BASIC programs to analyze polymodal frequency distribution into normal distributions (in Japanese with English abstract). *Bulletin of the Japan Sea Regional Fisheries Research Laboratory* 35: 129–160.

Bhattacharya, C. G. 1967. A simple method of resolution of a distribution into Gaussian components. *Biometrics* 23: 115–135.

Buchanan-Wollaston, H. J. and W. C. Hodgson. 1929. A new method of treating frequency curves in fishery statistics, with some results. *Journal du Conseil International pour l'Exploration de la Mer* 4: 207–225.

Cassie, R. M. 1954. Some uses of probability paper for the graphical analysis of polymodal frequency distributions. *Australian Journal of Marine and Freshwater Research* 5: 513–522.

Cox, D. R. 1966. Notes on the analysis of mixed frequency distributions. *The British Journal of Mathematical and Statistical Psychology* 19: 39–47.

Davis, J. C. 1971. *Statistics and Data Analysis in Geology*. New York: John Wiley & Sons, 170–297.

Dixon, W. and R. A. Kronmal. 1965. The choice of origin and scale for graphs. *Journal of the Association for Computing Machinery* 12: 259–261.

Doane, D. P. 1976. Aesthetic frequency classifications. *The American Statistician* 30: 181–183.

Emerson J. D. and D. C. Hoaglin. 1983. Stem-and-leaf displays. In *Understanding robust and exploratory data analysis*, ed. D. C. Hoaglin, F. Mosteller, and J. W. Tukey, New York: John Wiley & Sons.

Erzini, K. 1990. Sample size and grouping of data for length-frequency analysis. *Fisheries Research* 9: 355–366.

Fournier, D. A., J. R. Sibert, J. Majkowski, and J. Hampton. 1990. MULTIFAN, a likelihood-based method for estimating growth parameters and age composition from multiple length frequency data sets illustrated using data for Southern bluefin tuna (*Thunnus maccoyii*). *Canadian Journal of Fisheries and Aquatic Sciences* 47: 301–317.

Fox, J. 1990. Describing univariate distributions. In *Modern Methods of Data Analysis*, ed. J. Fox and J. S. Long. Newbury Park, California: Sage Publications.

Freedman, D. and P. Diaconis. 1981a. On the histogram as a density estimator: L^2 theory, *Zeitschrift fur Wahrscheinlichkeitstheorie und verwandte Gebiete* 57: 453–476.

——. 1981b. On the maximum deviation between the histogram and the underlying density. *Zeitschrift fur Wahrscheinlichkeitstheorie und verwandte Gebiete* 58: 139–167.

Gayanilo, F. C., Jr., M. Soriano, and D. Pauly. 1989. *A draft guide to the complete* ELEFAN. Manila, Philippines: ICLARM Software, International Center for Living Aquatic Resources Management.

Geiger, P. 1991. gr1: Enhancing visual display using stem-and-leaf. *Stata Technical Bulletin* 1: 8–9.

Goeden, G. B. 1978. A monograph of the coral trout, *Plectropomus leopardus (Lacépède) Research Bulletin of the Fisheries Service Queensland* 1: 42 pp.

Good, I. J. and R. A. Gaskins. 1980. Density estimation and bump-hunting by the penalized likelihood method exemplified by scattering and meteorite data. *Journal of the American Statistical Association*, 75: 42–73.

Goodall, C. 1990. A survey of smoothing techniques. In *Modern Methods of Data Analysis*, ed. J. Fox and J. S. Long. Newbury Park, California: Sage Publications.

Hansen, K. M. 1991. Head-banging: robust smoothing in the plane. IEEE *Transactions on Geoscience and Remote Sensing* 29: 369–378.

Harding, J. F. 1949. The use of probability paper for the graphical analysis of polymodal frequency distributions. *Journal of the Marine Biological Association of the United Kingdom* 28: 141–153.

Hasselblad, V. 1966. Estimation of parameters for a mixture of normal distributions. *Technometrics* 8: 431–444.

Laurec, A. and B. Mesnil. 1987. Analytical investigations of errors in mortality rates estimated from length distribution of catches, In *Length-Based Methods in Fisheries Research:* ICLARM *Conference Proceedings 13*, eds. D. Pauly and G. R. Morgan. Manila, Philippines: International Center for Living Aquatic Resources Management.

Liu, Q., T. Pitcher, and M. al-Hossaini. 1989. Ageing with fisheries length-frequency data, using information about growth. *Journal of Fish Biology* 35: 169–177.

Macdonald P. D. M. and P. E. J. Green. 1988. *User's Guide to Program* MIX*: An Interactive Program for Fitting Mixtures of Distributions* Hamilton, Ontario Canada: Ichthus Data Systems.

Macdonald P. D. M. and T. Pitcher. 1979. Age-groups from size-frequency data: a versatile and efficient method of analyzing distribution mixtures. *Journal of the Fisheries Research Board of Canada* 36: 987–1001.

Pauly, D. and J. F. Caddy. 1985. A modification of Bhattacharya's method for the analysis of mixtures of normal distributions. FAO *Fisheries Circular* 781: 16.

Petersen, C. G. J. 1892. *Fiskenes biologiske forhold i Holbaek Fjord*, 1890–91 (in Danish). *Beretning fra den Danske Biologiske Station for 1890(91)* 1: 121–183.

Salgado-Ugarte, I. H. 1992. *El análisis exploratorio de datos biológicos* (in Spanish). *Fundamentos y aplicaciones* México: ENEP Zaragoza UNAM & Marc Ediciones. 89–120; 213–233. (in Spanish).

Salgado-Ugarte, I. H. and J. Curts-García. 1992. sed7: Resistant smoothing using Stata. *Stata Technical Bulletin* 7: 8–11.

——. 1993. sed7.2: Twice reroughing procedure for resistant nonlinear smoothing. *Stata Technical Bulletin* 11: 14–16.

Salgado-Ugarte, I. H., M. Shimizu, and T. Taniuchi. 1993. snp6: Exploring the shape of univariate data using kernel density estimators. *Stata Technical Bulletin* 16: 8–19.

Scott, D. W. 1979. On optimal and data-based histograms. *Biometrika* 66: 605–610.

——. 1985a. Frequency polygons: Theory and application. *Journal of the American Statistical Association* 80: 348–354.

——. 1985b. Average shifted histograms: effective nonparametric density estimators in several dimensions. *Annals of Statistics* 13: 1024–1040.

——. 1992. *Multivariate Density Estimation: Theory, Practice, and Visualization.* New York: John Wiley & Sons.

Schnute, J. and D. Fournier. 1980. A new approach to length-frequency analysis: growth structure. *Canadian Journal of Fisheries and Aquatic Sciences* 37: 1337–1351.

Silverman, B. W. 1981. Using kernel density estimates to investigate multimodality. *Journal of the Royal Statistical Society, Series B* 43: 97–99.

——. 1983. Some properties of a test for multimodality based on kernel density estimates. In *Probability, Statistics and Analysis*, eds. J. F. C. Kingman and G. E. H. Reuter. Cambridge: Cambridge University Press. 248–249.

——. 1986. *Density Estimation for Statistics and Data Analysis* London: Chapman & Hall.

Sparre, P., E. Ursin, and S. C. Venema. 1989. Introduction to tropical fish stock assessment: Part 1, Manual. *smrm FAO Fisheries Technical Paper* 306.1. Rome: smrm FAO. 57–123.

Stata Corporation. 1993. *Stata Reference Manual: Release 3.1.* 6th ed. College Station, TX.

Sturges, H. A. 1926. The choice of a class interval. *Journal of the American Statistical Association* 21: 65–66.

Tanaka, S. 1962. A method of analyzing a polymodal frequency distribution and its application to the length distribution of the porgy, *Taius tumifrons* (T. and S.). *Journal of the Fisheries Research Board of Canada* 19: 1143–1159.

Tarter, M. E. and R. A. Kronmal. 1976. An introduction to the implementation and theory of nonparametric density estimation. *The American Statistician* 30: 105–112.

Taylor, B. J. R. 1968. The analysis of polymodal frequency distributions. *Journal of Animal Ecology* 34: 445–452.

Tukey, J. W. 1977. *Exploratory Data Analysis.* Reading, Massachusetts: Addison–Wesley.

Velleman, P. F. 1976. Interactive computing for exploratory data analysis I: display algorithms. *Proceedings of the Statistical Computing Section* Washington, DC: American Statistical Association.

——. 1980. Definition and comparison of robust nonlinear data smoothing algorithms. *Journal of the American Statistical Association* 75: 609–615.

——. 1982. Applied nonlinear smoothing. In *Sociological Methodology*, ed. S. Leinhardt. San Francisco: Jossey-Bass.

Wegman, E. J. 1972. Nonparametric probability density estimation II: A comparison of density estimation methods. *Journal of Statistical Computation and Simulation* 1: 225–245.

sg24 The piecewise linear spline transformation

Constantijn Panis, RAND Corporation, EMAIL panis@rand.org

In STB-10 (Nov. 1992), Richard Goldstein explained restricted cubic splines and presented `spline.ado` to compute such spline transformations. The purpose of both the restricted cubic spline transformation and the piecewise linear spline transformation described in this article is to capture nonlinear relationships in data.

Suppose one wishes to assess the effect of a continuous variable, say, `age`, on an outcome of interest. Whether the effect is linear, or whether the variable requires some transformation is often an empirical question. Where nonlinearities are suspected, it is common practice to include a quadratic form (`age*age`) in addition to `age` itself. While this increases the flexibility of the model, it remains unclear whether the parabolic relationship holds true. A minor inconvenience is also that the value of the explanatory value where its effect is maximal (or minimal) is not immediately available, but needs to be computed from the estimated coefficients. A more satisfactory approach is to explore the functional form of the relationship by first estimating the model in a nonparametric fashion. The user can, for example, create dummy variables for single years of age, or for age categories, and examine the parameter estimates of these dummies to detect the pattern of the age effect. These categorizations, however, have the disadvantage that the observed pattern may seem erratic, especially if some age categories contain only few observations. Furthermore, in most cases, we tend to believe that the effect of an explanatory variable changes gradually, not stepwise, as it increases in value.

Spline transformations, either cubic or linear (or quadratic or otherwise) provide a way to estimate the relationship nonparametrically, while guaranteeing that the effect changes gradually and continuously as the explanatory variable increases in value. Goldstein shows how this is done using cubic splines. Interpretation of the transformed variables and the corresponding parameter estimates is extremely difficult—only by computing and graphing some sort of predicted value will the user get an idea of the underlying functional relationship. Coefficients on linear splines, by contrast, are extremely straightforward to interpret.

The effect of the explanatory variable is assumed to be piecewise linear on an arbitrary number of segments, and each coefficient represents the slope on a particular segment. The piecewise linear transformation is given by

$$\begin{aligned} v(1) &= \min(\texttt{age}, \mu_1), \\ v(2) &= \max\Big(\min(\texttt{age} - \mu_1, \mu_2 - \mu_1), 0\Big), \\ &\vdots \\ v(n+1) &= \max(\texttt{age} - \mu_n, 0), \end{aligned}$$

where $v(1)$ through $v(n+1)$ are transformations of `age`, each corresponding to one of $n+1$ segments around the n nodes μ_1 through μ_n. Each of the transformed variables increases linearly over its segment and is constant elsewhere; their sum is equal to `age` itself. Note that the number of transformed variables is equal to the number of nodes plus one. We present an ado-file, `lspline.ado`, to compute variables $v(1)$ through $v(n+1)$. The syntax is

`lspline` *varname stem* [*nodelist*] [`if` *exp*] [`in` *range*] [`,` *nodetype* `replace`]

where *varname* is the input variable (e.g., `age`); its transformed output variables ($v(1)$ through $v(n+1)$, above) are named *stem*1, *stem*2, et cetera, and receive sensible labels. Nodes are specified in the *nodelist*, or may be chosen automatically by `lspline` through a *nodetype* specification. Three special types of node locations are recognized: `mean`, `median` and `quartile`. Omission of a *nodelist* and specification of `median`, for example, results in a spline transformation with one node, at the median

value of the input variable. Similarly, `lspline` creates one node at the mean or three nodes at quartile values when `mean` or `quartile` are specified. The `replace` option needs to be given in case one or more *stem*`*` variables already exist.

An application to the age at first marriage

The National Longitudinal Survey of Youth is a panel data set with economic and demographic information on young adults in the United States. The survey started in 1979 with respondents aged 14 to 22, and has been repeated annually. The last available wave was fielded in 1991, when the respondents were 26 to 34 years old. We are interested in the age at first marriage for women. Our sample includes 5779 young women; by 1991, 4111 (71%) have married, and the remaining 1668 remain unmarried. The youngest bride reported that she was only 13 years old on her wedding day.

A natural way to analyze the transition into marriage is through a hazard model. To illustrate the power of detecting nonlinearities through linear splines, we take a sequential probit approach. Starting at each respondent's 13th birthday, we estimate a probit model for whether the woman got married during the next year. For example, a woman who marries at age 19 accounts for seven probits: six 'failures' and one 'success'.

While a number of demographic characteristics (school enrollment, pregnancy) are very promising to predict the probability of getting married during the following year, we only take age into consideration. The propensity to get married may not be a linear function of age; exploratory analyses suggest that a large fraction of women marry between, say, the ages of 22 and 25, with lower fractions in the lower and higher age ranges. We therefore want to allow for a nonlinear relationship between age and the propensity to get married.

We present the results of a piecewise linear spline transformation of `age` with nodes at 18, 22 and 28 years, and contrast this with a quadratic specification in `age`.

```
. * Create spline transformations of age:
. lspline age aa 18 22 28
   aa1 = min(age,18)
   aa2 = max(min(age-18,22-18),0)
   aa3 = max(min(age-22,28-22),0)
   aa4 = max(age-28,0)
. describe
Contains data from marry.dta
  Obs: 63103 (max=139930)
 Vars:     7 (max=    40)
Width:    23 (max=    82)
  1. marry        byte   %8.0g            will marry this year
  2. id           int    %8.0g            respondent ID
  3. age          float  %9.0g            respondent age
  4. aa1          float  %9.0g            min(age,18)
  5. aa2          float  %9.0g            max(min(age-18,22-18),0)
  6. aa3          float  %9.0g            max(min(age-22,28-22),0)
  7. aa4          float  %9.0g            max(age-28,0)
Sorted by:  id
Note:  Data has changed since last save
. * The number of observations exceeds the number of respondents (5779)
. * because each respondent enters as many times as she has a birthday
. * since age 13 while not married.
. probit marry aa1 aa2 aa3 aa4, nolog
Probit Estimates                                  Number of obs =  63103
                                                  chi2(4)       =2538.34
                                                  Prob > chi2   = 0.0000
Log Likelihood = -13932.471                       Pseudo R2     = 0.0835
------------------------------------------------------------------------------
   marry |      Coef.   Std. Err.       t     P>|t|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
     aa1 |    .310451   .0102093     30.409   0.000       .2904408    .3304612
     aa2 |   .0328143    .006839      4.798   0.000       .0194099    .0462186
     aa3 |  -.0243593    .006322     -3.853   0.000      -.0367504   -.0119681
     aa4 |  -.0388698   .0214829     -1.809   0.070      -.0809763    .0032366
   _cons |  -6.906014   .1736961    -39.759   0.000      -7.246459   -6.565569
------------------------------------------------------------------------------
. * Then estimate the probability to get married using a quadratic form:
. generate age2=age*age
```

```
. probit marry age2 age, nolog
Probit Estimates                                  Number of obs =  63103
                                                  chi2(2)       =2330.81
                                                  Prob > chi2   = 0.0000
Log Likelihood = -14036.236                       Pseudo R2     = 0.0767
------------------------------------------------------------------------------
   marry |      Coef.   Std. Err.       t     P>|t|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
    age2 |  -.0132175   .0004413    -29.953   0.000      -.0140824   -.0123526
     age |   .6232345   .0189404     32.905   0.000       .5861112    .6603578
   _cons |  -8.459855   .1988649    -42.541   0.000       -8.84963   -8.070079
------------------------------------------------------------------------------

. * Where does age reach its maximum effect?
. display -0.5*_b[age]/_b[age2]
23.576134
```

Note that the estimated coefficients on spline variables provide direct insight into the shape of the age effect. Each coefficient represents the slope on a particular segment of the age range. Figure 1 shows the predicted values of the propensity to get married according to the spline and quadratic specifications. The quadratic model predicts that women are most likely to marry during the year following age 23.6. Constrained by its functional form, it appears to severely underestimate the probability of getting married at ages above 28.

There are no hard rules governing the choice of the number of nodes and their location. To minimize the degree of parametric structure imposed on the model, we started out with eight nodes at two-year intervals between ages 16 and 30. Stata's `test` command provides a very convenient and quick way to test whether adjacent slopes are significantly different from each other; a node may be eliminated if the hypothesis that the two surrounding slopes are the same is accepted. The desire to keep the model parsimonious may also prompt a reduction of the number of nodes. In most circumstances, we find that two to four nodes are adequate to capture nonlinear patterns.

Figure

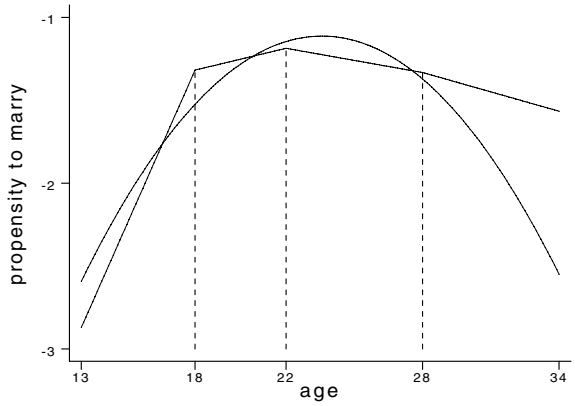

Figure 1

sts7.1	A library of time series programs for Stata (Update)

Sean Becketti, Stata Technical Bulletin, FAX 409-696-4601

In *sts7*, a library of time series programs for Stata was introduced (Becketti 1994). That insert described an approach to time series analysis that builds on Stata's core commands and on its extensibility. The insert also cataloged the programs in the time series library.

This update describes changes and additions to the time series library. An updated catalog of programs is also included. The updated library is available on the STB diskette. This update will be repeated in each issue of the STB. Consult the original insert for a general discussion of Stata's approach to time series analysis. As always, I actively solicit your comments, complaints, and suggestions.

New features

Expanded `lag` command: The `lag` command has been changed in two significant ways. First, the syntax of `lag` has been expanded to make it more convenient to lag and lead lists of variables. Second, the internal logic of `lag` has been modified to make `lag` "smarter" about combining operators when it names new variables.

Expanded syntax. The original syntax of the `lag` command was

`lag` [*#*] *varname* [, `suffix(`*str*`)`]

where the optional number after the command name indicated the number of lags (or leads, if negative) to generate. This syntax is still allowed, however an expanded syntax is now permitted as well. The new syntax is

`lag` [*#*] *varlist* [, `lags(`*#*[,*#*[,...]]`)` `suffix(`*str*[,*str*[,...]]`)`]

The optional number after the command name indicates the number of lags as before. Alternatively, the `lags()` option can be used. This option makes the syntax of the `lag` command similar to the syntaxes of the other time series commands. More importantly, the new syntax allows a list of variables names. Thus it is possible now to type

```
. lag 3 gnp money debt
```

to generate one through three lags each of the variables `gnp`, `money`, and `debt`. In fact, you can type

```
. lag gnp money debt, lags(2,-1,3)
```

to generate two lags of `gnp`, one lead of `money`, and three lags of `debt`. This expanded syntax should greatly reduce the amount of typing needed to prepare a data set of time series variables for analysis.

The `suffix()` option also has been generalized to handle variable lists. When this option is used, there must be at least as many suffixes specified as variables. (Any excess suffixes are ignored.) Thus

```
. lag gnp money debt, suffix(g,m,d)
```

generates `L.g`, `L.m`, `L.d`.

This expanded syntax will be extended to `dif`, `growth`, and `lead` soon.

Combining operators. In their original forms, `lag`, `lead`, `dif`, and `growth` did not always generate the shortest possible variable names for newly created variables. For example, the command '`lag D.gnp`' correctly generated a new variable named `LD.gnp`. However, '`lag L.gnp`' generated `LL.gnp`, *not* `L2.gnp`.

This anomaly has been corrected, and `lag` and `lead` now "add" lag and lead operators intelligently if they appear as the first operator in a name. This last clause is significant. The command '`lag LD.gnp`' generates `L2D.gnp`, but '`lag DL.gnp`' generates `LDL.gnp`, even though the contents of this variable are identical to `L2D.gpn`.

The reason for both the improvement and the remaining anomaly is a recently written utility program, `_addop`, that takes as input an operator and a variable name and returns as output the new variable name produced by applying the operator to the variable name. `_addop` is smarter than the original `lag` command, but it still doesn't understand all the rules of operator calculus. However, future improvements to `_addop` will now automatically improve the performance of `lag` and `lead`. The other programs that add operators to variable names (`dif`, `growth`, etc.) also will be modified soon to use `_addop`. When writing your own time series programs, I strongly recommend you use `_addop` to add operators to variable names.

`faketemp` added: The `faketemp` command has been added to the time series library (Hakkio 1994a). `faketemp` generates temporary variable names that can accept operators as leading characters without becoming confused with other temporary variables. `faketemp` will be used primarily by other time series programs.

New option and bug fix to `regdiag`: A new option, `time` has been added to `regdiag`. This option specifies a standard selection of time series diagnostics. This set includes Akaike's information criterion (`aic`), the ARCH test (`arch`), the Durbin–Watson test (`dw`), the LM (`lm`) and Q (`q`) tests for serial correlation, the Schwarz criterion (`sc`), and the test of the normality of the residuals (`normal`). In addition, the Durbin–Watson test would occasionally incorrectly report a missing value instead of the test statistic. This error has been corrected.

`tauprob` **added:** The `tauprob` command has been added to the time series library (Hakkio 1994b). `tauprob` calculates approximate asymptotic p-values for augmented Dickey–Fuller tests for unit roots and for Engle–Granger tests for cointegration.

Modified selection of diagnostics in `tsreg`: `tsreg` now reports the standard selection of time series diagnostics; that is, the set specified by the `time` option to `regdiag`.

A catalog of programs

The following table lists the user-level programs in the time series library. Each program's status is indicated by a letter grade. An 'A' indicates a program that is safe for general use. An 'A' program has been documented—*in its current form*—in the STB and follows all Stata guidelines for an estimation command, where relevant (see [4] estimate). A 'B' program produces accurate results, but either is not fully documented, not completely compatible with the time series syntax described above, or not in conformance with the guidelines for an estimation command. Most 'B' programs receive that grade because they have been revised significantly since they were last documented. A 'C' program is incomplete in significant ways but can be used safely by an advanced Stata user. A 'D' program has serious deficiencies, however its code may provide a useful model to advanced Stata users writing their own time series programs. An 'O' program is obsolete, that is, it has been superseded by a newer program. An 'O' program is retained if it is still be called by one or two user-level programs. There are currently no 'D' or 'O' programs.

Command	Status	Documentation	Description
`ac`	A	*sts1*	display autocorrelation plot
`chow`	C	—	perform Chow test for a shift in regression coefficients
`coint`	B	*sts2*	perform Engle–Granger cointegration test
`cusum`	B	—	perform CUSUM test of regression stability. (**Note:** this name conflicts with Stata's `cusum` command for binary variables.)
`datevars`	A	*sts4*	specify date variables
`dickey`	B	*sts2*	perform unit root tests
`dif`	A	*sts2*	generate differences
`dropoper`	A	*sts2*	drop operator variables
`findlag`	B	*sts2*	find optimal lag length
`findsmpl`	B	*sts4*	display sample coverage
`growth`	A	*sts2*	generate growth rates
`lag`	A	*sts2*	generate lags
`lead`	A	*sts2*	generate leads
`pac`	A	*sts1*	display partial autocorrelation plot
`pearson`	A	*sg5.1*	calculate Pearson correlation with p-value
`period`	A	*sts2*	specify period (frequency) of data
`quandt`	B	—	calculate Quandt statistics for a break in a regression
`regdiag`	B	*sg20*	calculate regression diagnostics
`spear`	A	*sg5.1*	Spearman correlation with p-value
`tauprob`	A	*sts6*	approximate p-values for unit root and cointegration tests
`testsum`	B	—	test whether the sum of a set of regression coefficients is zero
`tsfit`	A	*sts4*	estimate a time series regression
`tsload`	B	—	load an ad hoc time series equation into memory
`tsmult`	A	*sts4*	display information about lag polynomials
`tspred`	B	—	dynamically forecast or simulate a time series regression
`tsreg`	A	*sts4*	combined `tsfit`, `tsmult`, and `regdiag`
`xcorr`	A	*sts3*	calculate cross correlations

For more information on these programs, type '`help ts`' or '`help` *command-name*'.

Utilities for time series analysis

Writing programs for time series analysis presents a variety of challenges. In developing this library of programs, I had to write a pool of utility programs to interpret the time series options, to generate lags, to manipulate the list of variables in a lag polynomial, and so on. I recommend that you familiarize yourself with these utilities, if you wish to write your own time series programs. A list of some of the most frequently used utility programs follows.

Command	Description
_ac	calculate autocorrelations, standard errors, and Q-statistics
_addl	"add" a lag operator to a variable name
_addop	"add" an arbitrary operator to a variable name
_getrres	calculate recursive residuals for a regression model
_inlist	determine whether a token appears in a token list
_invlist	determine whether a varname appears in a varlist
_opnum	decode the operators (and their powers) in a varname
_parsevl	parse a varlist to replace abbreviations
_subchar	replace one character in a string with another
_ts_meqn	parse a time series command and generate lags
_ts_pars	parse a time series command into useful macros
faketemp	generate temporary variable names that can be lagged

Future developments and call for comments

As the comments above indicate, this library of time series programs is under constant revision and extension. Projects under development include programs to estimate rolling regressions, to estimate vector autoregressions, and to perform maximum-likelihood tests for cointegration. Older programs are being revised to bring them up to Stata's standards for estimation programs. A disadvantage of these constant revisions is the likelihood of inadvertently introducing errors into the programs. The advantage of constant revision is the ease and rapidity of fixing these errors and the steady increase in Stata's time series capabilities. I encourage you to alert me to any errors or inconveniences you find.

References

Becketti, S. 1994. sts7: A library of time series programs for Stata. *Stata Technical Bulletin* 17: 28–32.

Hakkio, C. 1994a. ip5: A temporary solution to a problem with temporary variable names. *Stata Technical Bulletin* 17: 8–10.

——. 1994b. sts6: Approximate p-values for unit root and cointegration tests. *Stata Technical Bulletin* 17: 25–28.

zz3.1	Computerized index for the STB	(Update)

William Gould, Stata Corporation, FAX 409-696-4601

The STBinformer is a computerized index to every article and program published in the STB. The command (and entire syntax) to run the STBinformer is `stb`. Once the program is running, you can get complete instructions for searching the index by typing `?` for help or `??` for more detailed help.

The STBinformer appeared for the first time on the STB-16 distribution diskette and included indices for the first fifteen issues of the STB. The STB-18 distribution diskette contains an updated version of the STBinformer that includes indices for the first *seventeen* issues of the STB. As the original insert stated, I intend to include an updated copy of this computerized index on every STB diskette. I encourage you to contact me with suggestions for changes and improvements in the program.

Reference

Gould W. 1993. Computerized index for the STB. *Stata Technical Bulletin* 16: 27–32.

View publication stats