Contents

Description of the “missreg” Library

Chris Wild and Yannan Jiang
University of Auckland, New Zealand

1 Introduction

This library gives functions for fitting arbitrary regression models to data subject to certain kinds of response-selective sampling and/or missingness that can be fitted by semi-parametric maximum likelihood. The class of likelihood functions that can be handled in this way is described in Section 3. Examples of members of the class can be found in Scott and Wild (1997, 2001), and Lawless *et al.* (1999).

The method applies to regression models in general and to quite a wide variety of missingness and selection schemes and we want to make this generality available. Unfortunately, there is a fatal conflict between the demands of an easily understandable user interface and the generality available here. We have taken a two tiered approach to overcome this. As a first tier, we have written specific functions to perform a limited number of specific types of regression with some specific missing data structures. Reading only Sections 2 (which concerns data structures) of this document should provide adequate preparation for using these. Details of the call sequences are given in the help files for the relevant function. All of these functions call a single general program engine. We have documented key parts of the wider system to enable a more sophisticated user to implement new models and/or new missingness structures. This latter use requires an understanding of the class of likelihood functions and missing data structures catered for and also of the program structure. Detailed discussions can be found in Section 3 and 4 of this document.

2 Functions for Specific Types of Analysis

2.1 Introduction

This package is concerned with estimation in the presence of response selection and/or certain kinds of missingness. To date, we have written functions for binary regression, for linear regression for continuous responses allowing for regression modelling of the scale as well as the location parameter, bivariate binary regression, and random intercepts models for clustered binary data (with arbitrary cluster sizes). We use the standard R/Splus formula language with some adaptations for models with more than one linear predictor as these necessitate, for example, the inclusion of several model formulae to describe the model.

Table 1: Leprosy data

	Scar=0		Scar=1		Total		Popn
Age[a]	Case	Control	Case	Control	Case	Control	Control
2.5	1	24	1	31	2	55	19312
7.5	11	22	14	39	25	61	17327
12.5	28	23	22	27	50	50	13172
17.5	16	5	28	22	44	27	10325
22.5	20	9	19	12	39	21	8026
27.5	36	17	11	5	47	22	5981
32.5	47	21	6	3	53	24	6479
Total					260	260	80622

[a]Age is age-group midpoint

2.1.1 Data structure

This package is concerned with the fitting of a regression of one or more Y-variables on one or more X-variables in the presence of certain missing value structures. Calls to analysis functions are of the general form:

```
    foo(formula, formula2, ..., xstrata = c("vname1","vname2", ...),
        obstype.name, data , xs.includes, ...)
```

where `formula` is of the usual form `y ~ x1 + x2 + ...` .

To understand the data structures catered for here, it is helpful to think in terms of a data frame in which columns are variables and rows (records) refer to data on the same "individual". For a subset of individuals (rows) we have data on all variables being used in the current analysis. For some individuals we may have data on all of the X-variables to be used in our regression but not on the Y-variables. There is another kind of variable sometimes present which we will call the V-variables. These are non-Y variables that we have data on for (essentially) all individuals used for the current analysis.

Example 1: *Leprosy data*

The leprosy data set described in Scott and Wild (1997, 2001), is reproduced in Table 1. It is included in the library as the data frame `leprosy1` shown in Figure 1. This data frame contains three variables: `leprosy`, `age`, and `scar` together with a frequency variable `counts` and a labelling variable called `obstype`.

For case-control sampled individuals we have data on all 3 variables. The case-control data makes up the first 28 rows of the data frame. The variable `obstype` labels these

leprosy	age	scar	counts	obstype
yes	2.5	no	1	retro
yes	2.5	yes	1	retro
yes	7.5	no	11	retro
yes	7.5	yes	14	retro
yes	12.5	no	28	retro
yes	12.5	yes	22	retro
yes	17.5	no	16	retro
yes	17.5	yes	28	retro
yes	22.5	no	20	retro
yes	22.5	yes	19	retro
yes	27.5	no	36	retro
yes	27.5	yes	11	retro
yes	32.5	no	47	retro
yes	32.5	yes	6	retro
no	2.5	no	24	retro
no	2.5	yes	31	retro
no	7.5	no	22	retro
no	7.5	yes	39	retro
no	12.5	no	23	retro
no	12.5	yes	27	retro
no	17.5	no	5	retro
no	17.5	yes	22	retro
no	22.5	no	9	retro
no	22.5	yes	12	retro
no	27.5	no	17	retro
no	27.5	yes	5	retro
no	32.5	no	21	retro
no	32.5	yes	3	retro
yes	2.5	NA	2	strata
yes	7.5	NA	25	strata
yes	12.5	NA	50	strata
yes	17.5	NA	44	strata
yes	22.5	NA	39	strata
yes	27.5	NA	47	strata
yes	32.5	NA	53	strata
no	2.5	NA	19312	strata
no	7.5	NA	17327	strata
no	12.5	NA	13172	strata
no	17.5	NA	10325	strata
no	22.5	NA	8026	strata
no	27.5	NA	5981	strata
no	32.5	NA	6479	strata

Figure 1: The leprosy data

rows as `"retro"` (for retrospective). We explain the labelling scheme in detail later. The population data on `leprosy` and `age` are represented by the last 14 rows of the data frame in terms of combinations of possible values and counts. The variable `obstype` labels these observations `"strata"`. For these rows there is only data on `leprosy` and `age` so `scar` has missing values (`NA`) for the last 14 rows. Because the case-control-study individuals are included in the population counts we will set `xs.includes=TRUE` when we call an analysis program rather than have to adjust the data frame to eliminate the double counting.

We will use the variable `leprosy` as our Y-variable. Note from Figure 1 that $Y =$ `leprosy` is available for all individuals. We are going to use `age` and `scar` as explanatory variables in models for `leprosy`. We have a single V-variable (a non-Y variable available for all individuals), namely, `age`.

■

Our semiparametric maximum likelihood methods can only cope with V-variables that are discrete. If you have data on continuous V-variables you will have to group them into class intervals. Discrete V-variables can be included in an analysis as so-called "`xstrata`" variables. Where data has been collected conditionally on Y-variables and some V-variables, any V-variables the data has been sampled on must be included in the `xstrata` list in order to obtain a valid analysis. Additional V-variables can be included in the list and this will increase the efficiency of analysis if they are closely related to X-variables. The same variable can appear both in the model as an X-variable and also as an `xstrata`-variable. Indeed that is the required construction if you want to take advantage of all-data information on a particular variable in the model. The one exception to this occurs when we have data for all individuals for all X-variables, as in Example 3 to follow. In this case we should not specify any `xstrata`. If a V-variable is included as a X-variable in the model but not specified as an `xstrata`-variable, the only information on the variable that will be used in the analysis is the values we have for those individuals for whom all Y- and X-variables in the model are fully observed.

Inside the program, strata are formed from all possible combinations of any `xstrata`-variables supplied to `foo()` and, in effect, the distribution of $\boldsymbol{x}$ is estimated nonparametrically within each stratum. Small counts of fully observed data in the cells formed by the cross-classification of `xstrata`-variables can be expected to cause problems. For this reason, analysis programs always print frequency tables of counts within strata.

Because we are allowing for both prospectively and retrospectively sampled data, and because the presence of random happenstance missing values may obscure the patterns described above, each record (row) must be labelled using the scheme described in Table

Table 2: Labels for observations by sampling and variable type.

`xstrata` variables[a]	`obstype` code	definition
None	`"uncond"`	$(\boldsymbol{y}, \boldsymbol{x})$ values sampled unconditionally[b]
	`"retro"`	$\boldsymbol{x}$ sampled conditionally on some or all of the Y-variables.
	`"xonly"`	unconditional[b] information on $\boldsymbol{x}$.
	`"y|x"`	$\boldsymbol{y}$ sampled conditionally on $\boldsymbol{x}$.
	`"strata"`	only the Y-variables observed.
Some supplied	`"uncond"`	$(\boldsymbol{y}, \boldsymbol{x})$ values sampled unconditionally[b] or conditionally on some or all of the `xstrata`[c] variables.
	`"retro"`	$\boldsymbol{x}$ sampled conditionally on some or all of the Y-variables, or on some or all Y-variables and some or all of the `xstrata` variables[c].
	`"xonly"`	unconditional[b] information on $\boldsymbol{x}$, or conditional on some or all of the `xstrata` variables.
	`"y|x"`	$\boldsymbol{y}$ sampled conditionally on $\boldsymbol{x}$.
	`"strata"`	only the Y and `xstrata` variables are observed.

[a] The same variable can appear in the model as an X-variable and also as an `xstrata` variable. Indeed that is required in order to take advantage of all-data information on a particular X-variable. The one exception to this occurs when we have data for all individuals for all X-variables. In this. case we should not specify any `xstrata`.

[b] Although we talk about "unconditional sampling" of $(\boldsymbol{y}, \boldsymbol{x})$, sampling can be conditional on some of the X-variables, but the conditioning set of variables must always be the same.

[c] When we talk about "sampling conditional on some or all of the `xstrata` variables", the conditioning set of variables must always be the same.

2 using a variable to be supplied in the call to `foo()` whose name is supplied through the argument `obstype.name` which has default `"obstype"`.

Some important details and qualifications:

1. *Population counts and* `"xs.includes"`
   For population based case-control studies, data on $\boldsymbol{y}$, $\boldsymbol{x}$ (and $\boldsymbol{v}$) for study individuals is supplemented by information on $\boldsymbol{y}$ and $\boldsymbol{v}$ for the whole population. Since the V-variables are discrete, this supplementary information will often be obtained via a cross-classification (e.g., published tables). This situation is handled correctly if the rows for study individuals are labelled `"retro"` and the rows relating to population counts are labelled `"strata"` (as in Example 1 above and in several examples which follow). Study individuals come from the population and thus are included in the population counts and thus are in the data set twice. Rather than requiring the user to separate these individuals out and reduce the population counts, the pro-

gram allows the user to specify that this is the case via setting `xs.includes` to `TRUE`.

Equivalently when we have two-phase sampling we use `xs.includes` set to `TRUE` or `FALSE` to indicate whether the rows of the data frame relating to first-phase sampling include or exclude those individuals selected at the second phase. Rows relating to first-phase sampled individuals should be labelled `"strata"`. If the second phase is unconditional we will label second-phase observations `"uncond"` rather than `"retro"`.

2. *Sampling strata versus analysis strata*
   As previously stated, strata are formed from all combinations of the V-variables named in `xstrata`. This stratification must be at least as detailed as any stratification used in sampling. More efficient analyses can be obtained if we have enough information so that the analysis stratification is finer than the stratification used in sampling.

3. *Estimation of the distribution of* $\boldsymbol{x}$ *and the* `"y|x"` *code*
   In standard regression fitting without missing data or response selection, the marginal distribution of $\boldsymbol{x}$ forms an orthogonal term in the likelihood that can be ignored when fitting regression models. This is not the case when we have missing data. In our semiparametric maximum likelihood approach, the marginal distribution of $\boldsymbol{x}$ is estimated nonparametrically. When there are no `xstrata` variables present, a single distribution is estimated for $\boldsymbol{x}$. When `xstrata` variables are present, a distribution for $\boldsymbol{x}$ is estimated separately within each stratum defined by all combinations of values of the `xstrata` variables. Data points labelled `"uncond"`, `"retro"` and `"xonly"` are used in the estimation of the distribution of $\boldsymbol{x}$. Data points labelled `"y|x"` are not used for this purpose (they contribute only to estimation of the regression parameters).

4. *Additional happenstance missingness*
   The analysis programs use case deletion to cope with the presence of small numbers of happenstance missing values occurring within the larger missingness structure. Thus: rows labelled `"uncond"`, `"retro"` and `"y|x"` will be deleted if $\boldsymbol{y}$ or the value of any X-variable in the model is missing; rows labelled `"xonly"` will be deleted if the value of any X-variable in the model is missing; and rows labelled `"xstrata"` will be deleted if $\boldsymbol{y}$ or the value of any `"xstrata"`-variable is missing. This latter tends to be inappropriate for `"xstrata"`-variables which are not also in the model. Here, missingness should usually be treated as just one more category (e.g. coded as `"miss"` rather than `NA`) when the variable is defined, prior to calling the function.

leprosy	age	scar	counts	obstype
yes	2.5	no	1	retro
yes	2.5	yes	1	retro
yes	7.5	no	11	retro
yes	7.5	yes	14	retro
yes	12.5	no	28	retro
yes	12.5	yes	22	retro
yes	17.5	no	16	retro
yes	17.5	yes	28	retro
yes	22.5	no	20	retro
yes	22.5	yes	19	retro
yes	27.5	no	36	retro
yes	27.5	yes	11	retro
yes	32.5	no	47	retro
yes	32.5	yes	6	retro
no	2.5	no	24	retro
no	2.5	yes	31	retro
no	7.5	no	22	retro
no	7.5	yes	39	retro
no	12.5	no	23	retro
no	12.5	yes	27	retro
no	17.5	no	5	retro
no	17.5	yes	22	retro
no	22.5	no	9	retro
no	22.5	yes	12	retro
no	27.5	no	17	retro
no	27.5	yes	5	retro
no	32.5	no	21	retro
no	32.5	yes	3	retro
yes	2.5	NA	2	strata
yes	7.5	NA	25	strata
yes	12.5	NA	50	strata
yes	17.5	NA	44	strata
yes	22.5	NA	39	strata
yes	27.5	NA	47	strata
yes	32.5	NA	53	strata
no	2.5	NA	19312	strata
no	7.5	NA	17327	strata
no	12.5	NA	13172	strata
no	17.5	NA	10325	strata
no	22.5	NA	8026	strata
no	27.5	NA	5981	strata
no	32.5	NA	6479	strata

(a) leprosy1

case	control	scar	age	obstype
1	24	no	2.5	retro
1	31	yes	2.5	retro
11	22	no	7.5	retro
14	39	yes	7.5	retro
28	23	no	12.5	retro
22	27	yes	12.5	retro
16	5	no	17.5	retro
28	22	yes	17.5	retro
20	9	no	22.5	retro
19	12	yes	22.5	retro
36	17	no	27.5	retro
11	5	yes	27.5	retro
47	21	no	32.5	retro
6	3	yes	32.5	retro
2	19312	NA	2.5	strata
25	17327	NA	7.5	strata
50	13172	NA	12.5	strata
44	10325	NA	17.5	strata
39	8026	NA	22.5	strata
47	5981	NA	27.5	strata
53	6479	NA	32.5	strata

(b) leprosy2

leprosy	age	scar	counts	obstype
yes	2.5	no	1	retro
yes	2.5	yes	1	retro
yes	7.5	no	11	retro
yes	7.5	yes	14	retro
yes	12.5	no	28	retro
yes	12.5	yes	22	retro
yes	17.5	no	16	retro
yes	17.5	yes	28	retro
yes	22.5	no	20	retro
yes	22.5	yes	19	retro
yes	27.5	no	36	retro
yes	27.5	yes	11	retro
yes	32.5	no	47	retro
yes	32.5	yes	6	retro
no	2.5	no	24	retro
no	2.5	yes	31	retro
no	7.5	no	22	retro
no	7.5	yes	39	retro
no	12.5	no	23	retro
no	12.5	yes	27	retro
no	17.5	no	5	retro
no	17.5	yes	22	retro
no	22.5	no	9	retro
no	22.5	yes	12	retro
no	27.5	no	17	retro
no	27.5	yes	5	retro
no	32.5	no	21	retro
no	32.5	yes	3	retro
yes	NA	NA	260	strata
no	7.5	NA	80622	strata

(c) leprosy3

Figure 2: The leprosy data in two permissable formats

2.2 Binary Regression (bin2stg)

The function `bin2stg` fits binary regression models using several links, the default being the logistic link. We will illustrate use of this function with some simple examples and default behaviour before describing more detailed features.

Example 1 *Leprosy data* (**continue**)

In Scott and Wild (1997, 2001) a logistic regression was performed of `leprosy` on `scar` and a transformation of `age`, namely $(\texttt{age} + 7.5)^{-2}$. The binary regression function is called `bin2stg`. The fitting syntax is as follows:

```
> data(leprosy1)
> leprosy1$age.trans <- 100 * (leprosy1$age + 7.5)^-2
> z1 <- bin2stg(leprosy ~ age.trans + scar, data=leprosy1, weights=counts,
                xstrata="age",xs.includes=TRUE)
```

Here, we have to tell the program to use age as an x-stratum variable and that the counts for the rows labelled strata include the counts for the rows labelled retro. It is equivalent to use age.trans as the x-stratum variable as both variables define the same set of groups. The output from summary(z1) is given below. The counts in the "Stratum Counts Report" reflect the data as it is used in the function (after removal of any random missing values). Thus, when xs.includes=TRUE the counts printed below for strata do not include those printed for retro observations.

```
> summary(z1)
Call:
bin2stg(formula = leprosy ~ age.trans + scar, weights = counts,
        xstrata = "age", data = leprosy1, xs.includes = TRUE)

Stratum Counts Report:
          xStrat     1      2      3      4      5      6      7
obstype y
retro   1            2     25     50     44     39     47     53
        2           55     61     50     27     21     22     24
strata  1            0      0      0      0      0      0      0
        2        19257  17266  13122  10298   8005   5959   6455

Model for prob of leprosy=yes (y=1)  given covariates

Key to x-Strat:
      1       2       3       4       5       6       7
 age2.5  age7.5 age12.5 age17.5 age22.5 age27.5 age32.5

loglikelihood = -3900.441   using 3 parameters

Wald Tests:
          Df   Chi Pr(>Chi)
age.trans  1 82.83  0.00000
scar       1  5.57  0.01827

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -4.4809     0.1144 -39.176  0.00000
age.trans    -4.0906     0.4495  -9.101  0.00000
scaryes      -0.4212     0.1785  -2.360  0.01827
```

Suppose we only had information on age for the case-control sampled individuals but had population counts of cases and controls. The data frame leprosy3 depicted in Figure 2(c) mimics this. The first 28 rows of leprosy3 are as for leprosy1. The last 2 rows of leprosy3 give population counts for cases and controls. This data set has no V-variables (i.e. no non-Y variables that have been observed for all individuals). Thus we do not specify any variables as xstrata.

```
> data(leprosy3)
> leprosy3$age.trans <- 100 * (leprosy3$age + 7.5)^-2
> z3 <- bin2stg(leprosy ~ age.trans + scar,data=leprosy3, weights=counts,
                xs.includes=TRUE)

> summary(z3)
Call:
bin2stg(formula = leprosy ~ age.trans + scar, weights = counts,
        data = leprosy3, xs.includes = TRUE)

Stratum Counts Report:
          xStrat     1
obstype y
retro   1          260
        2          260
strata  1            0
        2        80362

Model for prob of leprosy=yes (y=1)  given covariates

loglikelihood = -4949.26   using 3 parameters

Wald Tests:
          Df    Chi  Pr(>Chi)
age.trans  1 55.431 9.681e-14
scar       1  2.343 1.258e-01

Coefficients:
            Estimate Std. Error z value  Pr(>|z|)
(Intercept)  -4.5102     0.1600 -28.181 0.000e+00
age.trans    -4.3102     0.5789  -7.445 9.681e-14
scaryes      -0.3021     0.1974  -1.531 1.258e-01
```

The most obvious difference in output is in the "Stratum Counts Report". When no strata are defined, all observations are in the same stratum. The regression coefficients are very similar to what we saw in the previous analysis but the standard errors are somewhat larger reflecting the fact that the population information on age is no longer included in the analysis.

Note that if we use `leprosy1` without telling the program that `age` is an `xstrata`-variable, as below,

```
> summary(bin2stg(leprosy ~ age.trans + scar, data=leprosy1,
                   weights=counts, xs.includes=TRUE))
```

then the only information about `age` used is the information we have for fully observed individuals so that we get exactly the same results as when we obtained from `leprosy3`.

Specifying population proportions

In `leprosy3`, the case-control data is supplemented by a row giving the population counts for cases and a row giving the population count for controls. There are situations where we do not know population counts of cases and controls but have a pretty good idea of the percentage of cases in the population. We handle this situation in a very similar way. If we "knew" that the percentage of cases was approximately 0.003 and with a level of confidence roughly consistent with this having been obtained from a sample of $N = 100,000$ observations, use $N \times 0.003$ as the population count of cases and $N \times (1 - 0.003)$ as the population count of controls, and set `xs.includes=FALSE`. Use of a very large value of N corresponds to specifying the population percentage of cases precisely. Smaller values of N correspond to incorporating information about the population percertage of cases in an essentially Bayesian way.

The other standard data format

The data frame `leprosy1` is in the usual format expected by the analysis functions. However, in keeping with `glm()` for binary data, we also allow the format of `leprosy2` in Figure 2(b). The function call is now

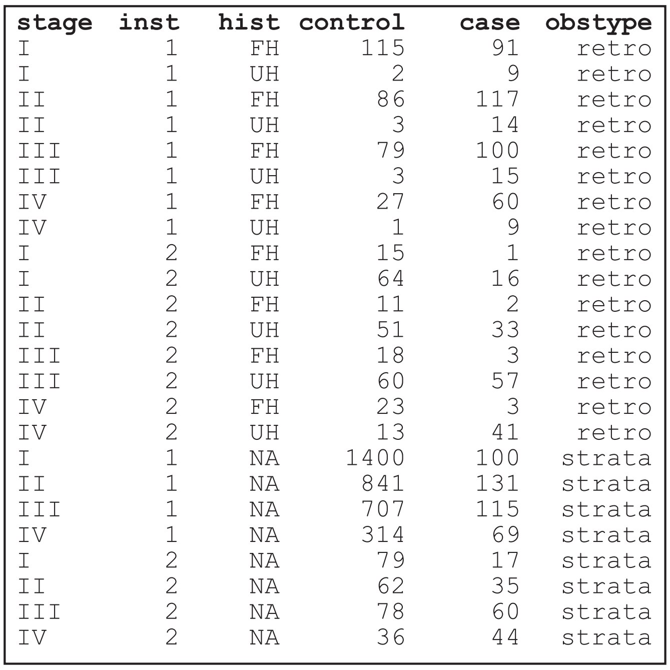
```
> data(leprosy2)
> leprosy2$age.trans <- 100 * (leprosy2$age + 7.5)^-2
> z2 <- bin2stg(cbind(case,control) ~ age.trans + scar, xstrata="age",
                 data=leprosy2, xs.includes=TRUE)
```

Running `summary(z2)` produces the same output as `summary(z1)` except for two lines which we print below.

```
Call:
bin2stg(formula = cbind(case, control) ~ age.trans + scar, xstrata = "age",
        data = leprosy2, xs.includes = TRUE)
. . . .

Model for prob of y=case (y=1)  given covariates
```

As we do not have a name for the resonse variable y, this second will suffice until we can think of a better solution.



stage	inst	hist	control	case	obstype
I	1	FH	115	91	retro
I	1	UH	2	9	retro
II	1	FH	86	117	retro
II	1	UH	3	14	retro
III	1	FH	79	100	retro
III	1	UH	3	15	retro
IV	1	FH	27	60	retro
IV	1	UH	1	9	retro
I	2	FH	15	1	retro
I	2	UH	64	16	retro
II	2	FH	11	2	retro
II	2	UH	51	33	retro
III	2	FH	18	3	retro
III	2	UH	60	57	retro
IV	2	FH	23	3	retro
IV	2	UH	13	41	retro
I	1	NA	1400	100	strata
II	1	NA	841	131	strata
III	1	NA	707	115	strata
IV	1	NA	314	69	strata
I	2	NA	79	17	strata
II	2	NA	62	35	strata
III	2	NA	78	60	strata
IV	2	NA	36	44	strata

Figure 3: Wilm's tumor data

Example 2: *Wilm's tumor data*

This data set was described in Breslow and Chatterjee (1999) as a balanced eight-strata data set. It has been included in the library as the data frame `wilms.sub` which is shown in Figure 3. We have data on `stage` (cancer staging), `inst` (institution) and case-control status for all study subjects and `hist` (histology classified as favourable or unfavourable) only for all cases and an equal number of randomly selected controls. We see that `stage` and `inst` are both V-variables (non-Y variables available for all individuals). Breslow and Chatterjee (1999) constructed data of the two-phase case-control form with a prospective first phase by only incorporating `inst` values only for a case-control sampled subset.

```
> data(wilms.sub)
> z4 <- bin2stg(cbind(case,control) ~ stage*hist, xstrata=c("stage",
                "inst"), data=wilms.sub, xs.includes=TRUE)


>  summary(z4)
Call:
bin2stg(formula = cbind(case, control) ~ stage * hist, xstrata = c("stage",
        "inst"), data = wilms.sub, xs.includes = TRUE)

Stratum Counts Report:
```

```
          xStrat    1    2    3    4    5    6    7    8
obstype y
retro   1         100  131  115   69   17   35   60   44
        2         117   89   82   28   79   62   78   36
strata  1           0    0    0    0    0    0    0    0
        2        1283  752  625  286    0    0    0    0

Model for prob of y=case (y=1)  given covariates

Key to x-Strat:
          1              2              3               4
 stageI:inst1  stageII:inst1 stageIII:inst1  stageIV:inst1
          5              6              7               8
 stageI:inst2  stageII:inst2 stageIII:inst2  stageIV:inst2

loglikelihood = -5330.041   using 8 parameters

Wald Tests:
           Df   Chi  Pr(>Chi)
stage       3 47.33 2.959e-10
hist        1 23.85 1.042e-06
stage:hist  3 11.78 8.169e-03

Coefficients:
                Estimate Std. Error  z value  Pr(>|z|)
(Intercept)      -2.7109     0.1080 -25.1019 0.000e+00
stageII           0.7707     0.1462   5.2733 1.340e-07
stageIII          0.7853     0.1512   5.1940 2.058e-07
stageIV           1.0459     0.1749   5.9809 2.219e-09
histUH            1.3756     0.2817   4.8836 1.042e-06
stageII:histUH    0.1007     0.3693   0.2727 7.851e-01
stageIII:histUH   0.4643     0.3522   1.3184 1.874e-01
stageIV:histUH    1.3682     0.4416   3.0984 1.946e-03
```

From above, we have fitted the model that Breslow and Chatterjee fitted. The printed `strata` counts again do not include the `retro` counts. Case-control status is modelled in terms of `stage` and `hist`. We want the all-individuals information on both `stage` and `inst`, although the latter is not in the model, to be used so both have been included as `xstrata` variables.

Example 3: *Trouser-trawl data*

The data included in the library as the data frame `trawl` and shown in Figure 4 is de-

caught37	**length**	**count**	**obstype**
NA	24	1	xonly
NA	25	1	xonly
NA	26	3	xonly
NA	27	14	xonly
NA	28	30	xonly
NA	29	49	xonly
NA	30	60	xonly
NA	31	49	xonly
NA	32	70	xonly
NA	33	108	xonly
NA	34	88	xonly
NA	35	84	xonly
NA	36	68	xonly
NA	37	37	xonly
NA	38	33	xonly
NA	39	12	xonly
NA	40	5	xonly
NA	41	6	xonly
NA	42	10	xonly
NA	43	1	xonly
NA	44	6	xonly
NA	45	2	xonly
NA	46	1	xonly
NA	47	0	xonly
1	24	0	retro
1	25	0	retro
1	26	0	retro
1	27	1	retro
1	28	5	retro
1	29	19	retro
1	30	29	retro
1	31	51	retro
1	32	91	retro
1	33	120	retro
1	34	118	retro
1	35	107	retro
1	36	78	retro
1	37	52	retro
1	38	40	retro
1	39	17	retro
1	40	17	retro
1	41	14	retro
1	42	10	retro
1	43	4	retro
1	44	6	retro
1	45	2	retro
1	46	5	retro
1	47	1	retro

Figure 4: Trouser-trawl data

scribed in Millar (1992). The idea is to estimate the catching qualities of a test net. Two nets were dragged behind a fishing trawler. One had a very fine-meshed end which let no fish escape. The other was a net with a mesh size that is being tested for its catching qualities. Our Y-variable is `caught37`. From the fine net we have a sample from the length distribution of the fish swimming into the nets, pr(x) say. Their value of `caught37` is missing (`NA`). These observations are labelled `xonly` because we have only observed the X-variable. From the test net we have a sample from the the distribution of the lengths of fish caught by the net, pr(length | caught) say. These observations are labelled `retro` and for them `caught37` has the value `1`.

We want to estimate $\text{pr}(Y = 1 \mid x) = \text{pr}(\text{caught} \mid \text{length})$. We can fit a logistic regression model as follows:

```
> data(trawl)

### Estimate pr(caught) assuming all fish over len=35 are caught:
# 265 out of 787 fish in fine net have length over 35  (caught37=NA)
# 353 out of 738 fish in test net have length over 35  (caught37=1)
# So 738 were caught from (estimate) 353*787/265 that entered
> phat <- 738 / (787*353/265)

> z5 <- bin2stg(caught37 ~ I(length-35), weights=count, data=trawl,
        start=c(log(phat/(1-phat)),0), Qstart=matrix(c(phat,1-phat)))

> summary(z5)
Call:
bin2stg(formula = caught37 ~ I(length - 35), weights = count, data = trawl,
    start = c(log(phat/(1 - phat)), 0), Qstart = matrix(c(phat, 1 - phat)))

Stratum Counts Report:
          xStrat   1
obstype y
retro   1        787
xonly   1          0

Observations of obstype==xonly
       xStrat
obstype   1
  retro   0
  xonly 738

Model for prob of caught37=1 (y=1) given covariates
```

```
loglikelihood = -11137.78   using 2 parameters

Wald Tests:
               Df   Chi  Pr(>Chi)
I(length - 35)  1 21.43 3.668e-06

Coefficients:
               Estimate Std. Error z value  Pr(>|z|)
(Intercept)      4.4740     1.1988   3.732 1.899e-04
I(length - 35)   0.9244     0.1997   4.629 3.668e-06
```

We have had to supply some starting values here as we do not have a good general starting value strategy for this data structure. The parameters `start` and `Qstart` in the function call are used to supply the starting values for $\boldsymbol{\theta}$, the regression coefficients, and for $\boldsymbol{Q}$ in which $Q_1 = \text{pr}(\boldsymbol{Y}=1)$ and $Q_2 = \text{pr}(\boldsymbol{Y}=0)$[1] respectively. When the logistic regression model is considered as in this example, the estimate of intercept is simply the logit of $\text{pr}(\boldsymbol{Y}=1)$ when the coefficient of X is set to zero.

Example 4: *Birthweight binary data*

This example gives a more real-world idea about usage of these functions in data analysis. The data frame `lowbirth.bin` is a subset of the data collected in the Auckland Collaborative Birthweight (ABC) study (see Thompson *et al.*, 2001). The ABC study was a case-control study in which, essentially, all births below the 10th percentile of weight (for gestational age and sex) were taken together with an equal-sized random sample of births above the 10th percentile. This data has been augmented here by routinely-collected data on all births over the study period. We will call the individuals selected into the case-control study *study individuals* and the rest *nonstudy individuals*. Data frame `lowbirth.bin` contains data on a subset of the variables for a 50% sample of individuals (after a number of deletions). The nonstudy controls have been further subsampled from the knowledge that approximately 1 in 9 controls were selected[2]. The data frame contains 1148 rows and 18 columns. The 18 variables are described in Table 3.

We will run a logistic regression on the response `sgagp` using explantory variables `mumht`, `bmi`, and the factors `ethnicdb`, `occ`, `hyper` and `smoke`. We will use a quadratic in `bmi`. Sampling into the case-control study was conditional on `sgagp` but not on any of the other variables. Other all-individuals variables can be used as `xstrata` variables in an

[1]See Section 3.2.3 to 3.2.4 for more detail.

[2]The original 50% sample contained 7215 nonstudy controls which were about twice as many controls in the data base as there should have been given the 1 in 9 sampling rate. As the information used by the exclusion rules for study babies was unavailable for the non-study babies, we randomly selected the correct number of controls to form a synthetic data set to use.

Table 3: Variables in `lowbirth.bin`

"All" individuals	
`instudy`	with 2 levels: `retro`=in-study (771 births), `strata` = out-of-study (2708 births, all controls)
`sgagp`	with 2 levels: `sga` (small) or `aga` (appropriate) for gestational age
`counts`	Frequency variable
`sex`	1=female or 2=male
`ethnicdb`	`A` (Asian), `E` (Euro.), `M` (Maori) or `P` (Pacifican)
`smokedb`	Smoking variable from database. Levels are: `Y`, `N`, and `miss` (About 35% are missing (`miss`) for both study and nonstudy births.)
`mstrat`	Marital status : 1=married, 2=defacto. 3=never married, 4=single separated, divorced or widowed
`mstratdb`	`mstrat`, but with levels 3 and 4 combined into `"3,4"`
`htstrat`	Class intervals of mother's height in cm: (0,1.55], (1.55,1.6], (1.6,1.65], (1.65,1.7], (1.7,2] and `miss` (About 49% are missing for both study and nonstudy births.)
`edstratdb`	mother's education, levels 1 to 4 and `miss` (About 40% are missing for both study and nonstudy births.)
Study individuals only (so the data on these are missing for the 2708 non-study births)	
occ	occupational group, 1 to 3 (3 is highest)
`mumht`	height of the mother in cm. (9 study individuals have `NA`)
`mumwt`	weight of the mother in kg. (17 study individuals have `NA`)
`bmi`	body mass index of the mother. (21 study individuals have `NA`)
`smoke`	smoking prior to pregnancy `Y` or `N` (6 study individuals have `NA`)
age1st	mother's age at first pregnancy
`hyper`	any hypertension (1=yes, 0=no)
`eductrm`	age mother left school: `<=15`,`16`, `17`, `18`, `>=19`. (4 study individuals have `NA`)

attempt to gain more efficiency. The simulations in Lawless et al. (1999) suggest that appreciable efficiency gains will only be made using V-variables that are identical to or are surogates of continuous variables in the model, and that most of those gains will probably be confined to those X-variables. We do not have any particularly good surrogates here. Nevertheless, we have included `ethnicdb` and `smokedb` as `xstrata`-variables for illustrative purposes.

```
> data(lowbirth.bin)
> z6 <- bin2stg(sgagp~mumht+bmi+I(bmi^2)+ethnicdb+factor(occ)+hyper+smoke,
          weights=counts, xstrata=c("ethnicdb","smokedb"),
          obstype.name=c("instudy"), data=lowbirth.bin, xs.includes=FALSE)

> summary(z6)
Call:
bin2stg(formula = sgagp ~ mumht + bmi + I(bmi^2) + ethnicdb + factor(occ) +
        hyper + smoke, weights = counts, xstrata = c("ethnicdb", "smokedb"),
        obstype.name = c("instudy"), data = lowbirth.bin, xs.includes = FALSE)

Observations deleted due to missing data:
 retro: 26 rows relating to 26 observations

Stratum Counts Report:
          xStrat   1   2   3   4   5   6   7   8   9  10  11  12
obstype y
retro   1         33  39  13  21  72  67   3  20   1  27  21  11
        2         13  76   4  39  31 106   5  33   2  21  12  15
strata  1          0   0   0   0   0   0   0   0   0   0   0   0
        2        102 414  52 190 259 933  40 288  16 185 104 125

Model for prob of sgagp=sga (y=1)  given covariates

Key to x-Strat:
                   1                     2                     3
ethnicdbA:smokedbmiss ethnicdbE:smokedbmiss ethnicdbM:smokedbmiss
                   4                     5                     6
ethnicdbP:smokedbmiss    ethnicdbA:smokedbN    ethnicdbE:smokedbN
                   7                     8                     9
   ethnicdbM:smokedbN    ethnicdbP:smokedbN    ethnicdbA:smokedbY
                  10                    11                    12
 ethnicdbE:smokedbY    ethnicdbM:smokedbY    ethnicdbP:smokedbY

loglikelihood = -4000.046   using 11 parameters
```

```
Wald Tests:
            Df    Chi  Pr(>Chi)
mumht        1 17.064 3.615e-05
bmi          1  8.507 3.537e-03
I(bmi^2)     1  5.381 2.035e-02
ethnicdb     3 15.432 1.483e-03
factor(occ)  2  7.784 2.040e-02
hyper        1 18.324 1.863e-05
smoke        1  4.261 3.900e-02

Coefficients:
               Estimate Std. Error   z value  Pr(>|z|)
(Intercept)   12.324846    2.75129  4.479663 7.476e-06
mumht         -0.056579    0.01370 -4.130812 3.615e-05
bmi          -35.087025   12.02962 -2.916719 3.537e-03
I(bmi^2)      49.662119   21.40839  2.319750 2.035e-02
ethnicdbE     -0.533751    0.19864 -2.686996 7.210e-03
ethnicdbM      0.001221    0.31550  0.003872 9.969e-01
ethnicdbP     -0.682218    0.26617 -2.563109 1.037e-02
factor(occ)2   0.347636    0.20076  1.731635 8.334e-02
factor(occ)3   0.742938    0.27329  2.718473 6.558e-03
hyper          1.297394    0.30308  4.280652 1.863e-05
smokeY         0.438023    0.21220  2.064206 3.900e-02
```

In introducing and motivating data structures we have seen a number of examples of the use of the binary regression function, `bin2stg` and described quite a few of its features. A detailed description of call sequence, output etc. is given in the help file for the function. Here we merely highlight two or three issues that have not been discussed previously.

The function `bin2stg` fits binary regression models using several links. Links are specified using the parameter `linkname` and the default is `"logit"`. Also implemented are `"probit"` and `"cloglog"`.

One of the most difficult issues to address with the range of data types allowed is that of starting values. To start the iterative process, starting values are required for both the regression coefficients (see `start`) and for the marginal response probablities within each combination of `xstrata` levels (see `Qstart`). For some data structures, we can automatically calculate sensible starting values. Outside these classes, the user will have to supply starting values.

If `Qstart` is `NULL` and `xs.includes` is `TRUE`, `Qstart` values are estimated using observations of observation type `strata` only. If `xs.includes` is `FALSE`, `Qstart` values are estimated using observations of observation type `uncond, retro` and `strata`. In all cases, the estimates are the proportions of those in x-stratum j with response i, $i = 1, 2$.

If `start` is `NULL`, the program constructs starting values for the regression coefficients by sending observations of types `uncond`, `retro`, or `y|x` to `glm()` with `retro` observations accompanied by appropriate offsets. It refuses to construct starting values if only one value of the Y-variable is observed (as in Example 3, trouser trawl data), or if the sample sizes are deemed too small. Under these circumstances, the user is required to supply starting values, cf. Example 3. In this example, we were able to apply particular knowledge/intuition about the problem to come up with starting values that proved adequate.

Other useful strategies include starting by fitting very simple models and supplying the results through `start` when fitting a more complex model. If the program finds a vector in `start` which it is not of the length expected from the model, it will match `names(`*start vector*`)` and the variable names in the model, extract corresponding elements of the start vector to use as starting values, and use zeroes as starting values for all other coefficients. When using this strategy, it is usually advisable to centre any continuous X-variable being added to the model. Otherwise adding the variable is likely to cause a big change in the intercept with the previously estimated intercept then often being too far from the solution, thus causing the algorithm to fail.

2.3 Linear Regression (locsc2stg)

Linear regressions can be fitted to the data structures described earlier using the function `locsc2stg`. This function caters for the *location and scale* class of models which are of the form

$$y = \eta_1 + \exp(\eta_2)\,U.$$

Here, η_1 and η_2 are linear predictors and U is an error variable. Currently implemented error distributions are the standard normal, the logistic, and Student's t-distribution with user-specified degrees of freedom. A model for the location parameter η_1 (e.g., the mean) is specified in the usual way for `lm()`, e.g., `y ~ x1 + x2`, supplied as the first formula in the call sequence. If `~ 1` is specified for the second formula in the call then η_2 is a constant, the log of the scale parameter. By using something like `~ x2 + x3` for the second formula, η_2 is modelled using a linear model in the covariates. It is sometimes attractive to use the fitted values of y from a previously fitted model (available in `$fitted`) in modelling the scale. Note that the second formula has no left-hand side.

The function allows for two types of information on the Y-variable for observations of type `strata`. One can use the actual y-values (when `method` is set equal to `"direct"`) or one can just use class-interval information on the y-values (when `method` is set equal to `"ycutmeth"`). In terms of what the Y-variable should look like on input to `locsc2stg`, the program behaves as if actual y-values are present. The user supplies a vector of cutpoints (in `yCuts`) dividing up $(-\infty, \infty)$. For the `"ycutmeth"`, the program sets up y-strata corresponding to the intervals so defined and determines stratum membership from determining which interval a given y-observation falls into. Thus, if you really do only have interval-level information on y for observations of type `strata`, you should use something like the centre of the interval as the y-value.

Why might one want to use only class-interval information on y (the `"ycutmeth"` method) when it is possible to use the actual y-values themselves (the `"direct"` method)? There are two reasons, storage and speed. The `"direct"` method utilizes very big arrays and these may become too large to be accommodated in memory for larger data sets. Processing these large arrays makes fitting even slower. The losses of efficiency in using the `"ycutmeth"` method may often not be great with judicious choice of y-cutpoints. (We are currently investigating this). Increased speed combined with relatively small losses in efficiency would make the `"ycutmeth"` method preferable in the more exploratory stages of model fitting.

If the program is to calculate starting values automatically, this is done using a weighted least squares fit using `lm()`. Here, `yCuts` information is used to reweight observations of type `"retro"` when calling `lm()`.

Compressing the data

Three arguments in the call are aimed at reducing storage when using the direct method. When `compactY=TRUE`, the program will round each y-value used for observations of type `strata` to the closest of `straty.maxnvals` equally spaced values spanning the range of y and compress these to distinct values and frequencies before fitting. This always makes a big difference to speed and storage. It appears that we can round these Y values fairly heavily without appreciable impact on the results. We are currently investigating this further. The current default rounds observations to one of 20 equally spaced values. Most of our simulations are rounded to 40 values with excellent results.

When `compactX=TRUE`, the program will round X-values for records with full explanatory variable information down to distinct combinations and frequencies before fitting. Substantial compression will occur only if there are not many X-variables and they all are coarsely discrete.

Further details about `yCuts`

In the simplest case, we assume that $-\infty < c_1 < c_2 < \ldots < c_k < \infty$ and supply the program with (the vector) `yCuts = c(`c_1`, `c_2`, ... , `c_k`)`. Note that the $\pm\infty$ endpoints are assumed and intervals of the form $(c_k < c_{k+1}]$ are employed. If a vector of cutpoints is supplied to the program, that set of cutpoints will be applied to y for every x-stratum. Sometimes, as in Example 5 below, different sets of cutpoints need to be applied in different `xstrata`. In this case, `yCuts` should be a matrix with the jth column containing the y-cutpoints corresponding to the jth x-stratum. We elaborate within the Example. The program allows for different numbers of y-cutpoints in different x-strata. One simply pads the bottom of any column of the `yCuts` matrix that is not full with `NA`'s.

Note that with the `"ycutmeth"` method, the stratification defined by the `yCuts` and `xstrata` on the $Y \times X$-space must be at least as fine as any stratification used in sampling. With a finer stratification a successful analysis incorporates more information about y and any `xstrata` variables for observations of type `strata`.

When data on y is rounded (as it always is) correct handling of what happens on the interval boundaries can be extremely important when there are substantial differences in sampling rates between strata. Otherwise some observations sampled at a low rate may be treated as if they were sampled at a high rate and vice-versa. The `"direct"` method is much more robust in this regard.

Example 5: *Continuous birthweight data*

This is a random 20% subset of data from the Auckland Collaborative Birthweight Study (after some simplifying deletions). As in data frame `lowbirth.bin`, the nonstudy controls in this data have also been subsampled with approximately 1 in 9 controls being selected. There are two additional variables that are not in the variable list from `lowbirth.bin` of Example 4, Table 3.

`gest`	Gestational age in weeks (38, 39, 40, 41)
`birthwt`	Birthweight in grams

Sampling was conditional on whether the birthweight was below or above a `birthwt` cutoff (our y variable). However, cutoff values for defining "small for gestational age" differed by `xstrata` defined by gestational age and sex as shown in below.

	Age			
Sex	38	39	40	41
Female	2550	2740	2900	3030
Male	2650	2840	3010	3140

In this example, the minimum useable stratification contains 16 $Y \times X$-strata defined by whether the birth falls below or above the relevant y (birthweight) cutpoint and the 8 combinations of levels of sex and gestational age. The x-strata are defined by the 8 combinations of levels of sex and gestational age. Because we are using different y-cutpoints in different x-strata, `yCuts` should be a matrix with 8 columns. The ordering of x-strata used by the program is critical so that, to ensure that the ordering of the columns of `yCuts` is the same as the ordering of the x-strata, it is safest to explictly form a single x-stratum variable (whose name is then communicated via `xstrata`) with levels coming in a known order before calling `locsc2stg()`.[3] Because our y-intervals are of the form $(-\infty, c]$ and (c, ∞), the matrix `yCuts` should have a single row.

We will now set up a single x-stratum variable (which we will call `sex.age`) and a `yCuts` matrix, and fit a model with a constant scale parameter to the response variable `birthwt`.

```
> data(lowbirth.ls)
> lowbirth.ls$sex.age <- interaction(lowbirth.ls$sex,lowbirth.ls$gest)
> yCuts <- matrix(c(2550,2650,2740,2840,2900,3010,3030,3140),nrow=1)
>
> z7 <- locsc2stg(birthwt~gest+mumht+bmi+ethnicdb+hyper+smoke, ~1,
                 yCuts=yCuts, xstrata=c("sex.age"), data=lowbirth.ls,
                 obstype.name=c("instudy"), xs.includes=FALSE,
                 method="ycutmeth")

> summary(z7)
Call:
locsc2stg(formula1 = birthwt ~ gest + mumht + bmi + ethnicdb + hyper +
          smoke, formula2 = ~1, yCuts = yCuts, xstrata = c("sex.age"),
          data = lowbirth.ls, obstype.name = c("instudy"),
          method = "ycutmeth", xs.includes = FALSE)

Observations deleted due to missing data:
 retro: 8 rows relating to 8 observations

Stratum Counts Report:
               xStrat  1   2   3   4   5   6   7   8
obstype yStrat
retro   1              7   8  13   7  27  18  12  13
        2             10   8  16  13  22  21  16  20
strata  1              0   0   0   0   0   0   0   0
```

[3]If the same set of y-cuts is being applied in each x-stratum (signalled by `yCuts` being a vector), order is not a problem and we can continue to define x-strata by setting `xstrata` to be a vector with variable names as we did previously with `bin2stg`.

```
      2             88  64 128 104 176 176 128 168

Key to x-Strat:
          1           2           3           4           5           6
sex.age1.38 sex.age2.38 sex.age1.39 sex.age2.39 sex.age1.40 sex.age2.40
          7           8
sex.age1.41 sex.age2.41

Key to the y-Strat:
     [,1]                [,2]                 [,3]
[1,] (-3.07e+04,2.55e+03] (-3.07e+04,2.65e+03] (-3.07e+04,2.74e+03]
[2,] (2.55e+03,3.46e+04]  (2.65e+03,3.46e+04]  (2.74e+03,3.46e+04]
     [,4]                 [,5]                [,6]
[1,] (-3.07e+04,2.84e+03] (-3.07e+04,2.9e+03] (-3.07e+04,3.01e+03]
[2,] (2.84e+03,3.46e+04]  (2.9e+03,3.46e+04]  (3.01e+03,3.46e+04]
     [,7]                 [,8]
[1,] (-3.07e+04,3.03e+03] (-3.07e+04,3.14e+03]
[2,] (3.03e+03,3.46e+04]  (3.14e+03,3.46e+04]

loglikelihood = -2683.785   using 10 parameters

Location Model:
Wald Tests:
         Df    Chi  Pr(>Chi)
gest      1 26.705 2.371e-07
mumht     1 16.255 5.536e-05
bmi       1  8.166 4.268e-03
ethnicdb  3 30.639 1.012e-06
hyper     1 36.780 1.323e-09
smoke     1  4.269 3.881e-02

Coefficients:
            Estimate Std. Error z value  Pr(>|z|)
(Intercept) -3716.93    963.584  -3.857 1.146e-04
gest          108.27     20.952   5.168 2.371e-07
mumht          14.50      3.598   4.032 5.536e-05
bmi            13.61      4.763   2.858 4.268e-03
ethnicdbE     220.22     63.493   3.468 5.237e-04
ethnicdbM     188.84    106.379   1.775 7.588e-02
ethnicdbP     435.47     79.935   5.448 5.100e-08
hyper        -620.56    102.324  -6.065 1.323e-09
smokeY       -116.12     56.203  -2.066 3.881e-02
```

```
Scale Model:
Coefficients:
  Estimate Std. Error    z value   Pr(>|z|)
   5.29675    0.04601  115.12151    0.00000
```

In forming the intervals the program takes "$-\infty$"= $\min(y) - 10 \times \text{range}(y)$ and "∞"= $\max(y) + 10 \times \text{range}(y)$.

Using the "direct" method with the results of the previous fit as strating values, we get:

```
> z8 <- locsc2stg(birthwt~gest+mumht+bmi+ethnicdb+hyper+smoke,~ 1,
        xstrata=c("sex.age"), data=lowbirth.ls, obstype.name=c("instudy"),
        xs.includes=FALSE, method="direct", start=z7$coefficients,
        compactX=TRUE, compactY=TRUE, straty.maxnvals=20)

> summary(z8)
Call:
locsc2stg(formula1 = birthwt ~ gest + mumht + bmi + ethnicdb + hyper + smoke,
          formula2 = ~1, xstrata = c("sex.age"), data = lowbirth.ls,
          obstype.name = c("instudy"), method = "direct",
          xs.includes = FALSE, compactX = TRUE, compactY = TRUE,
          straty.maxnvals = 20, start = z7$coefficients)

Observations deleted due to missing data:
 retro: 8 rows relating to 8 observations

Stratum Counts Report:
        xStrat
obstype  1   2   3   4   5   6   7   8
  retro   17  16  29  20  49  39  28  33
  strata  88  64 128 104 176 176 128 168

Key to x-Strat:
          1           2           3           4           5           6
sex.age1.38 sex.age2.38 sex.age1.39 sex.age2.39 sex.age1.40 sex.age2.40
          7           8
sex.age1.41 sex.age2.41

loglikelihood = -10275.40   using 10 parameters

Location Model:
```

```
Wald Tests:
         Df    Chi  Pr(>Chi)
gest      1 23.772 1.084e-06
mumht     1 21.789 3.044e-06
bmi       1 14.689 1.268e-04
ethnicdb  3 31.390 7.035e-07
hyper     1 40.521 1.945e-10
smoke     1  4.145 4.175e-02

Coefficients:
            Estimate Std. Error z value  Pr(>|z|)
(Intercept) -3541.35    925.557  -3.826 1.301e-04
gest           94.61     19.405   4.876 1.084e-06
mumht          16.33      3.499   4.668 3.044e-06
bmi            18.16      4.739   3.833 1.268e-04
ethnicdbE     180.33     59.322   3.040 2.367e-03
ethnicdbM     141.26    106.054   1.332 1.829e-01
ethnicdbP     437.16     80.835   5.408 6.372e-08
hyper        -654.90    102.880  -6.366 1.945e-10
smokeY       -114.99     56.480  -2.036 4.175e-02

Scale Model:
Coefficients:
  Estimate Std. Error    z value   Pr(>|z|)
    5.3466     0.0372   143.7251     0.0000
```

Refitting looking for a scale proportional to size effect by including the fitted values from the previous model in the scale model (divided by 1000 to keep the magnitude of the variable comparable with others we are using) we get

```
> z9 <- locsc2stg(birthwt~gest+mumht+bmi+ethnicdb+hyper+smoke,
         ~I(z8$fitted/1000), xstrata=c("sex.age"), data=lowbirth.ls,
         obstype.name=c("instudy"), xs.includes=FALSE, method="direct",
         start=z8$coefficients, Qstart=z8$Qmat, compactX=TRUE, compactY=TRUE)

> summary(z9)
Call:
locsc2stg(formula1 = birthwt ~ gest + mumht + bmi + ethnicdb + hyper + smoke,
        formula2 = ~I(z8$fitted/1000), yCuts = yCuts, xstrata = c("sex.age"),
        data = lowbirth.ls, obstype.name = c("instudy"), method = "direct",
        xs.includes = FALSE, compactX = TRUE, compactY = TRUE,
        start = z8$coefficients, Qstart = z8$Qmat)
```

```
Observations deleted due to missing data:
 retro: 8 rows relating to 8 observations

Stratum Counts Report:
        xStrat
obstype  1   2   3   4   5   6   7   8
  retro   17  16  29  20  49  39  28  33
  strata  88  64 128 104 176 176 128 168

Key to x-Strat:
          1           2           3           4           5           6
sex.age1.38 sex.age2.38 sex.age1.39 sex.age2.39 sex.age1.40 sex.age2.40
          7           8
sex.age1.41 sex.age2.41

loglikelihood = -10272.24   using 11 parameters

Location Model:
Wald Tests:
         Df    Chi  Pr(>Chi)
gest      1 28.438 9.676e-08
mumht     1 24.801 6.356e-07
bmi       1 15.860 6.820e-05
ethnicdb  3 35.656 8.853e-08
hyper     1 63.311 1.776e-15
smoke     1  3.546 5.969e-02

Coefficients:
            Estimate Std. Error z value  Pr(>|z|)
(Intercept) -4071.29    933.207  -4.363 1.285e-05
gest          103.69     19.444   5.333 9.676e-08
mumht          17.20      3.455   4.980 6.356e-07
bmi            18.37      4.613   3.982 6.820e-05
ethnicdbE     213.38     58.640   3.639 2.740e-04
ethnicdbM     156.33    106.997   1.461 1.440e-01
ethnicdbP     449.71     78.156   5.754 8.717e-09
hyper        -684.61     86.040  -7.957 1.776e-15
smokeY       -105.89     56.233  -1.883 5.969e-02

Scale Model:
Wald Tests:
                    Df   Chi Pr(>Chi)
```

```
I(z8$fitted/1000).2  1 6.177  0.01294

Coefficients:
                    Estimate Std. Error z value Pr(>|z|)
(Intercept).2         4.3589     0.3977  10.959  0.00000
I(z8$fitted/1000).2   0.2779     0.1118   2.485  0.01294
```

2.4 Linear to Binary Regression (linbin2stg)

It is not uncommon for a continuous outcome variable Y to be dichotomized and analysed using logistic regression. Moser and Coombs (*Statistics in Medicine* 2005, **23**) provide a method for converting the output from a standard linear regression analysis using the original continuous outcome Y to give much more efficient inferences about the same odds-ratio parameters being estimated by the logistic regression. However, these results apply only to prospective studies. When data has been obtained from response-selective sampling mechanisams such as case-control sampling, standard linear regression no longer produces valid estimates. A special-purpose software is therefore required.

We have developed a simple function based on `locsc2stg` which can provide both linear- and binary-based inferences as well as the estimated odds ratios and their 95% confidence intervals. The linear regression model we consider is of the form

$$y = \eta_1 + \sigma U,$$

where a single scale parameter σ is considered with logistic error distribution.

For a continuous outcome measure, say Y, a cut-off point needs to be provided to dichotomize Y into a binary response of interest. When stratfication is used with different cut-off points for different stratum, however, the intercept of binary-regression parameters cannot be easily estimated. Since this is not an issue for estimating the odds ratios, we will only output complete inferences when one cut-off point is employed.

Using the same Example 5, we can fit the following regression models using the simple function `linbin2stg`.

```
> data(lowbirth.ls)
> lowbirth.ls$sex.age <- interaction(lowbirth.ls$sex,lowbirth.ls$gest)
> yCuts <- matrix(c(2550,2650,2740,2840,2900,3010,3030,3140),nrow=1)

> z10 <- linbin2stg(birthwt~gest+mumht+bmi+ethnicdb+hyper+smoke,
                   yCuts=yCuts, xstrata=c("sex.age"), data=lowbirth.ls,
                   obstype.name=c("instudy"), xs.includes=FALSE)
```

```
> summary(z10)
Call:
linbin2stg(formula1 = birthwt ~ gest + mumht + bmi + ethnicdb +
  hyper + smoke, yCuts = yCuts, xstrata = c("sex.age"),
  data = lowbirth.ls, obstype.name = c("instudy"),
  xs.includes = FALSE)

Observations deleted due to missing data:
 retro: 8 rows relating to 8 observations

Stratum Counts Report:
               xStrat   1   2   3   4   5   6   7   8
obstype yStrat
retro   1               7   8  13   7  27  18  12  13
        2              10   8  16  13  22  21  16  20
strata  1               0   0   0   0   0   0   0   0
        2              88  64 128 104 176 176 128 168

Key to x-Strat:
          1           2           3           4           5           6
sex.age1.38 sex.age2.38 sex.age1.39 sex.age2.39 sex.age1.40 sex.age2.40
          7           8
sex.age1.41 sex.age2.41

Key to the y-Strat:
     [,1]                 [,2]                 [,3]
[1,] (-3.07e+04,2.55e+03] (-3.07e+04,2.65e+03] (-3.07e+04,2.74e+03]
[2,] (2.55e+03,3.46e+04]  (2.65e+03,3.46e+04]  (2.74e+03,3.46e+04]

     [,4]                 [,5]                 [,6]
[1,] (-3.07e+04,2.84e+03]  (-3.07e+04,2.9e+03] (-3.07e+04,3.01e+03]
[2,] (2.84e+03,3.46e+04]   (2.9e+03,3.46e+04]  (3.01e+03,3.46e+04]

     [,7]                 [,8]
[1,] (-3.07e+04,3.03e+03]  (-3.07e+04,3.14e+03]
[2,] ((3.03e+03,3.46e+04]   (3.14e+03,3.46e+04]

loglikelihood = -2683.785   using 10 parameters


Linear Location Model:
```

```
Wald Tests:
         Df    Chi  Pr(>Chi)
gest      1 26.705 2.371e-07
mumht     1 16.255 5.536e-05
bmi       1  8.166 4.268e-03
ethnicdb  3 30.639 1.012e-06
hyper     1 36.780 1.323e-09
smoke     1  4.269 3.881e-02

Linear Coefficients:
             Estimate Std. Error z value  Pr(>|z|)
(Intercept) -3716.932  963.58436  -3.857 1.146e-04
gest          108.271   20.95174   5.168 2.371e-07
mumht          14.504    3.59750   4.032 5.536e-05
bmi            13.609    4.76251   2.858 4.268e-03
ethnicdbE     220.217   63.49332   3.468 5.237e-04
ethnicdbM     188.837  106.37922   1.775 7.588e-02
ethnicdbP     435.467   79.93490   5.448 5.100e-08
hyper        -620.557  102.32422  -6.065 1.323e-09
smokeY       -116.125   56.20288  -2.066 3.881e-02
log(scale)      5.297    0.04601 115.122 0.000e+00

Binary Coefficients:
            Estimate Std. Error z value  Pr(>|z|)
(Intercept)       NA         NA      NA        NA
gest        -0.54220    0.10718  -5.059 4.223e-07
mumht       -0.07263    0.01838  -3.952 7.758e-05
bmi         -0.06815    0.02416  -2.821 4.791e-03
ethnicdbE   -1.10281    0.32144  -3.431 6.018e-04
ethnicdbM   -0.94566    0.53367  -1.772 7.640e-02
ethnicdbP   -2.18074    0.41491  -5.256 1.473e-07
hyper        3.10764    0.53181   5.843 5.112e-09
smokeY       0.58153    0.28321   2.053 4.004e-02

Odds Ratios for Binary Parameters:
             O.R. Lower C.I. Upper C.I.
gest       0.5815    0.47129     0.7174
mumht      0.9299    0.89704     0.9641
bmi        0.9341    0.89091     0.9794
ethnicdbE  0.3319    0.17678     0.6233
ethnicdbM  0.3884    0.13647     1.1055
ethnicdbP  0.1130    0.05009     0.2547
```

```
hyper     22.3682    7.88756    63.4336
smokeY     1.7888    1.02679     3.1163
```

Note that the intercept of binary coefficients was not estimated in the function due to different `yCuts` we used for different sex and gestational age of the child. If we used the mean of `yCuts` as a single cut-off point for all strata, the following output is provided with a complete inference.

```
> yCut1 <- mean(yCuts)
> yCut1
[1] 2857.5

> z11 <- linbin2stg(birthwt~gest+mumht+bmi+ethnicdb+hyper+smoke,
                    yCuts=yCut1, xstrata=c("sex.age"), data=lowbirth.ls,
                    obstype.name=c("instudy"), xs.includes=FALSE)

> summary(z11)
Call:
linbin2stg(formula1 = birthwt ~ gest + mumht + bmi + ethnicdb +
  hyper + smoke, yCuts = yCut1, xstrata = c("sex.age"),
  data = lowbirth.ls, obstype.name = c("instudy"),
  xs.includes = FALSE)

Observations deleted due to missing data:
 retro: 8 rows relating to 8 observations

Stratum Counts Report:
               xStrat   1   2   3   4   5   6   7   8
obstype yStrat
retro   1               8  10  16   7  21   9   3   4
        2               9   6  13  13  28  30  25  29
strata  1              14   6   8   1   0   0   0   0
        2              74  58 120 103 176 176 128 168

Key to x-Strat:
         1           2           3           4           5           6
sex.age1.38 sex.age2.38 sex.age1.39 sex.age2.39 sex.age1.40 sex.age2.40
         7           8
sex.age1.41 sex.age2.41

Key to the y-Strat:
     [,1]                  [,2]                  [,3]
[1,] (-3.07e+04,2.86e+03] (-3.07e+04,2.86e+03] (-3.07e+04,2.86e+03]
[2,] (2.86e+03,3.46e+04]  (2.86e+03,3.46e+04]  (2.86e+03,3.46e+04]
     [,4]                 [,5]                 [,6]
[1,] (-3.07e+04,2.86e+03] (-3.07e+04,2.86e+03] (-3.07e+04,2.86e+03]
[2,] (2.86e+03,3.46e+04]  (2.86e+03,3.46e+04]  (2.86e+03,3.46e+04]
     [,7]                 [,8]
[1,] (-3.07e+04,2.86e+03] (-3.07e+04,2.86e+03]
[2,] (2.86e+03,3.46e+04]  (2.86e+03,3.46e+04]

loglikelihood = -2727.675   using 10 parameters


Linear Location Model:
Wald Tests:
         Df    Chi  Pr(>Chi)
gest      1 14.010 1.818e-04
mumht     1 18.469 1.727e-05
bmi       1  8.599 3.363e-03
ethnicdb  3 31.180 7.791e-07
hyper     1 42.842 5.933e-11
smoke     1  4.877 2.722e-02

Linear Coefficients:
             Estimate Std. Error z value  Pr(>|z|)
(Intercept) -2781.676  1.004e+03  -2.771 5.592e-03
gest           78.329  2.093e+01   3.743 1.818e-04
mumht          15.759  3.667e+00   4.298 1.727e-05
bmi            14.511  4.949e+00   2.932 3.363e-03
ethnicdbE     212.957  6.428e+01   3.313 9.229e-04
ethnicdbM     166.595  1.095e+02   1.521 1.282e-01
ethnicdbP     447.018  8.173e+01   5.469 4.514e-08
hyper        -659.314  1.007e+02  -6.545 5.933e-11
smokeY       -128.444  5.816e+01  -2.208 2.722e-02
log(scale)      5.323  4.643e-02 114.644 0.000e+00

Binary Coefficients:
            Estimate Std. Error z value  Pr(>|z|)
(Intercept) 27.52130    4.94050   5.571 2.539e-08
gest        -0.38228    0.10306  -3.709 2.079e-04
mumht       -0.07691    0.01833  -4.196 2.712e-05
bmi         -0.07082    0.02450  -2.891 3.845e-03
ethnicdbE   -1.03931    0.31665  -3.282 1.030e-03
ethnicdbM   -0.81305    0.53500  -1.520 1.286e-01
```

```
ethnicdbP   -2.18161    0.41441  -5.264 1.407e-07
hyper        3.21770    0.51368   6.264 3.752e-10
smokeY       0.62685    0.28598   2.192 2.838e-02

Odds Ratios for Binary Parameters:
             O.R. Lower C.I. Upper C.I.
gest       0.6823    0.55751     0.8350
mumht      0.9260    0.89330     0.9598
bmi        0.9316    0.88795     0.9775
ethnicdbE  0.3537    0.19015     0.6579
ethnicdbM  0.4435    0.15542     1.2656
ethnicdbP  0.1129    0.05009     0.2543
hyper     24.9706    9.12379    68.3413
smokeY     1.8717    1.06858     3.2785
```

We see from the output that a unique cut-off point has been used for all strata and the intercept of binary coefficients was estimated.

Another important point to consider when dichotomizing the continuous outcome Y is how the cases are defined. The default option in the function is as follows.

$$pr(cases) = pr(Y^* = 1) = pr(Y \leq c),$$

When an upper tail is used to define the cases such as BMI over 30 for obesity, an argument `lower.tail=FALSE` needs to be specified in the function to get the correct inference for binary parameters.

```
> z12 <- linbin2stg(birthwt~gest+mumht+bmi+ethnicdb+hyper+smoke,
                    yCuts=yCut1, lower.tail=FALSE, xstrata=c("sex.age"),
                    data=lowbirth.ls, obstype.name=c("instudy"),
                    xs.includes=FALSE)

> summary(z12)
... ...

Binary Coefficients:
             Estimate Std. Error z value  Pr(>|z|)
(Intercept) -27.52130    4.94050  -5.571 2.539e-08
gest          0.38228    0.10306   3.709 2.079e-04
mumht         0.07691    0.01833   4.196 2.712e-05
bmi           0.07082    0.02450   2.891 3.845e-03
ethnicdbE     1.03931    0.31665   3.282 1.030e-03
ethnicdbM     0.81305    0.53500   1.520 1.286e-01
```

```
ethnicdbP     2.18161    0.41441   5.264 1.407e-07
hyper        -3.21770    0.51368  -6.264 3.752e-10
smokeY       -0.62685    0.28598  -2.192 2.838e-02

Odds Ratios for Binary Parameters:
             O.R. Lower C.I. Upper C.I.
gest      1.46562    1.19755     1.7937
mumht     1.07994    1.04184     1.1194
bmi       1.07339    1.02306     1.1262
ethnicdbE 2.82726    1.51995     5.2590
ethnicdbM 2.25477    0.79013     6.4344
ethnicdbP 8.86059    3.93287    19.9625
hyper     0.04005    0.01463     0.1096
smokeY    0.53427    0.30502     0.9358
```

We see that all binary parameters had opposite signs, and the estimated odds ratios switched from one side of one to another.

2.5 Bivariate Binary Regression (bivbin2stg)

In some studies, we are interested in modelling bivariate binary responses (Y_1, Y_2) and quantifying the association between them given covariates X. Methods have been written for such cases with data collected under different two-phase sampling schemes.

In this section, we will discuss how to carry out various semiparametric maximum likelihood analyses with bivariate binary data using the function `bivbin2stg`.

2.5.1 Sampling on Y_1

Suppose we have a binary response variable Y_1 taking values on 1 and 0 for individuals in the case and control populations respectively. In each Y_1 population, a random sample is taken with another response of interest Y_2 and potential covariates X being subsequently observed. The model of interest is $Pr(Y_1, Y_2|X)$ which, in principle, can be modelled arbitrarily.

A frequently used modelling approach for bivariate binary responses is to model the joint distribution of Y_1 and Y_2 given X by separately modelling the marginal distributions of $Pr(Y_1|X)$ and $Pr(Y_2|X)$, and the association between Y_1 and Y_2 given X (i.e. specifying 3 models). We will call methods which couple such models with semiparametric maximum likelihood estimation `spml1`. Currently implemented models for this method are the Palmgren, Bahadur and Copula models which we descirbe in the next paragraph.

In the Palmgren, Bahadur and our Copula models, a linear logistic model is used to model the two marginal probabilities of Y_1 and Y_2,

$$\begin{cases} \texttt{logit}\,\texttt{Pr}(\boldsymbol{Y}_1 = 1|\boldsymbol{X} = \boldsymbol{x}_1;\ \boldsymbol{\theta}_1) = \boldsymbol{x}_1^T\boldsymbol{\theta}_1 \\ \\ \texttt{logit}\,\texttt{Pr}(\boldsymbol{Y}_2 = 1|\boldsymbol{X} = \boldsymbol{x}_2;\ \boldsymbol{\theta}_2) = \boldsymbol{x}_2^T\boldsymbol{\theta}_2 \end{cases} . \tag{1}$$

The association between Y_1 and Y_2 given X is modelled in different ways.

- `Palmgren`: Association is modelled by log $\texttt{OR} = \boldsymbol{x}_3^T\boldsymbol{\theta}_3$, where `OR` is the conditional odds ratio of Y_1 and Y_2 given X. Here, Y_1 and Y_2 are conditionally independently if and only if $\log \texttt{OR} = 0$.
- `Bahadur`: Association is modelled by $\log\left(\dfrac{1+\boldsymbol{\rho}}{1-\boldsymbol{\rho}}\right) = \boldsymbol{x}_3^T\boldsymbol{\theta}_3$, where $\boldsymbol{\rho}$ is the conditional correlation coefficient between Y_1 and Y_2 given X. Here, Y_1 and Y_2 are conditionally independently if and only if the above log-ratio is zero.
- `Copula`: This model is implemented using Frank's family of bivariate distributions of the form

  $$\mathrm{Pr}(\mathrm{Y}_1 \le \mathrm{y}_1, \mathrm{Y}_2 \le \mathrm{y}_2) = \log_\alpha\left\{1 + \frac{(\alpha^{\mathrm{F}_1(\mathrm{y}_1)} - 1)(\alpha^{\mathrm{F}_2(\mathrm{y}_2)} - 1)}{\alpha - 1}\right\} \quad (\alpha \neq 1)$$

  where $\log_\alpha(t)$ denotes logarithm to the base $\boldsymbol{\alpha} > 0$. Here, $F_1(y_1)$ and $F_2(y_2)$ indicate the marginal distribution functions of Y_1 and Y_2 given covariates. Recall that their probabilities have been modelled in the form of (1) with binary Y_1 and Y_2 under our setting. The parameter $\boldsymbol{\alpha}$ has a natural interpretation as a rank based association parameter and allows for a full range of positive or negative correlation between the two variables.

  Association is modelled by $\boldsymbol{\gamma} = -\log(\boldsymbol{\alpha}) = \boldsymbol{x}_3^T\boldsymbol{\theta}_3$, where $\boldsymbol{\gamma}$ is treated as the association parameter in the model. The above distribution function becomes

  $$\mathrm{Pr}(\mathrm{Y}_1 \le \mathrm{y}_1, \mathrm{Y}_2 \le \mathrm{y}_2) = -\frac{1}{\gamma}\log\left\{1 + \frac{(\mathrm{e}^{-\gamma\mathrm{F}_1(\mathrm{y}_1)} - 1)(\mathrm{e}^{-\gamma\mathrm{F}_2(\mathrm{y}_2)} - 1)}{\mathrm{e}^{-\gamma} - 1}\right\}.$$

  Here, Y_1 and Y_2 are conditionally independently if and only if $\boldsymbol{\gamma} = 0$.

  Genest (1987) gave the following approximation for the population equivalent of Spearman's correlation coefficient as a function of $\boldsymbol{\gamma}$

  $$\boldsymbol{\rho}_s \approx (e^{-\gamma/2} - 1)^{-2}\,(1 - \gamma e^{-\gamma/2} - e^{-\gamma})$$

  based on continuous bivariate random variables. We will use it to obtain an initial estimate of $\boldsymbol{\gamma}$ when starting values for $\boldsymbol{\theta}_3$ are required.

All three models are appropriate for analyses when the joint modelling of $Y = (Y_1, Y_2)$ given X is required to explore the question of interest. In some situation, however, we carry out a secondary analysis based on the same type of data where we are only interested in the marginal distribution of Y_2 given X. When Y_2 is correlated with Y_1, conventional logistic regression will no longer provide consistent estimates for the parameters of interest. We can still apply the `spml1` method but the model of interest is now just $Pr(Y_2 = 1|X)$. The joint modelling of Y_1 and Y_2 is just one strategy to cope with the data collection mechanism, and it requires building two nuisance models, one for Y_1 and another for the association.

When we are only interested in $Pr(Y_2 = 1|X)$, another semiparametric maximum likelihood approach which we called the `spml2` method can be used. The basic idea is that the joint distribution of Y_1 and Y_2 given X is now estimated in terms of a conditional factorisation $Pr(Y_1|Y_2, X)Pr(Y_2|X)$ both treated parametrically. As both Y_1 and Y_2 are binary in our implementation, logistic models are used to separately model the two probabilities.

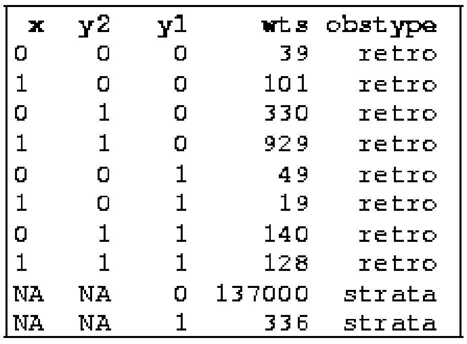
$$\begin{cases} \texttt{logit}\ \texttt{Pr}(\boldsymbol{Y}_1 = 1|\boldsymbol{Y}_2 = \boldsymbol{y}_2, \boldsymbol{X} = \boldsymbol{x}_1;\ \boldsymbol{\theta}_1) = \boldsymbol{x}_1^{*T}\boldsymbol{\theta}_1 \\ \\ \texttt{logit}\ \texttt{Pr}(\boldsymbol{Y}_2 = 1|\boldsymbol{X} = \boldsymbol{x}_2;\ \boldsymbol{\theta}_2) = \boldsymbol{x}_2^{T}\boldsymbol{\theta}_2 \end{cases} . \tag{2}$$

where $\boldsymbol{x}_1^* = (\boldsymbol{y}_2, \boldsymbol{x}_1)$. We call these the $Y_1|Y_2$-model and Y_2-model respectively. This approach requires only one nuisance model to be built, namely the $Y_1|Y_2$-model which typically requires inclusion of interactions between Y_2 and X-variables.

In the `missreg` library, the function we used to carry out semiparametric maximum likelihood analyses with bivariate binary data is called `bivbin2stg`. We see that both `spml1` and `spml2` methods employ more than one linear predictors, say η_1 and η_2 for Y_1-model and Y_2-model respectively in both (1) and (2), and η_3 for the association model if applicalble. The regression models for η_1 and η_2 are specified in the usual way for `glm()`, e.g., $y_1 \sim x_1 + x_2$ and $y_2 \sim x_1 + x_2$, supplied as the first and second formulas in the call sequence. If ~ 1 is specified in the call, the corresponding η is a constant. As η_3 is only considered in the `spml1` method with options for modelling, it is specified in the same way as the log-scale parameter for linear regression (e.g. $\sim x_1 + x_2$) with no left-hand side.

Example 6: *Cot-death data*

The original study was carried out in 1989 motivated by the comparatively high incidence of cot death (y_1) in New Zealand (Mitchell *et al.*, 1991). Because of its overall low incidence in the population, a case-control design was used to collect the data in order to find potnetial risk factors for this phenomenon. The data were subsequently reused in a secondary study aimed at identifying potential risk factors for the failure of immunization (y_2). Although the study examined many covariates, only one explanatory variable (x)



x	y2	y1	wts	obstype
0	0	0	39	retro
1	0	0	101	retro
0	1	0	330	retro
1	1	0	929	retro
0	0	1	49	retro
1	0	1	19	retro
0	1	1	140	retro
1	1	1	128	retro
NA	NA	0	137000	strata
NA	NA	1	336	strata

Figure 5: The Cot death data

is considered here which is the mother's marital status. The data frame for this study is shown in Figure 5.

We first fit the data using the Palmgren model, which is specified with the parameter `method` in the function call `bivbin2stg`. The fitting syntax is

```
> data(cotdeath)
> z13 <- bivbin2stg(y1~x, y2~x, ~x, weights=wts, data=cotdeath,
                    xs.includes=TRUE, method="palmgren")
```

and the following summary of regression output can be obtained:

```
> summary(z13)
Call:
bivbin2stg(formula1 = y1 ~ x, formula2 = y2 ~ x, formula3 = ~x,
           weights = wts, data = cotdeath, xs.includes = TRUE,
           method = "palmgren")

Stratum Counts Report:
               xStrat      1
obstype yStrat
retro   1             336
        2            1399
strata  1               0
        2          135601

Models for prob of joint distribution of y1 and y2 given covariates

loglikelihood = -15864.31   using 6 parameters
```

```
Y1-Model
Wald Tests:
  Df   Chi Pr(>Chi)
x  1 103.5        0

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)   -5.253     0.0854  -61.51        0
x             -1.278     0.1256  -10.17        0

Y2-Model
Wald Tests:
  Df    Chi Pr(>Chi)
x  1 0.2144   0.6433

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  2.12711     0.1673  12.711   0.0000
x            0.09137     0.1973   0.463   0.6433

Association Model
Wald Tests:
  Df   Chi Pr(>Chi)
x  1 4.697  0.03021

Coefficients:
             Estimate Std. Error z value  Pr(>|z|)
(Intercept)  -1.0857     0.2371  -4.579 4.675e-06
x             0.7743     0.3573   2.167 3.021e-02
```

Recall that when `xs.includes=TRUE`, the counts printed for `strata` in the `Stratum Counts Report` do not include those already printed for `retro`.

The coefficients tables for Y_1, Y_2 and the association models are listed sequentially right after the Wald tests for each of the models. When the Bahadur and Copula models are considered in the `spml1` method, we use `method=bahadur` and `method=copula` respectively in the `locsc2stg` function call. Because we have a saturated regression model (only one X-variable is considered) in this example, both methods give exactly the same output as that obtained from the Palmgren model for the two marginal probabilities. The results have been presented below.

```
> summary(z14)
```

```
Call:
bivbin2stg(formula1 = y1 ~ x, formula2 = y2 ~ x, formula3 = ~x,
      weights = wts, data = cotdeath, xs.includes = TRUE,
      method = "bahadur")
... ...

Association Model
Wald Tests:
  Df   Chi Pr(>Chi)
x  1 9.298 0.002294

Coefficients:
            Estimate Std. Error z value  Pr(>|z|)
(Intercept) -0.07167    0.01944  -3.686 0.0002280
x            0.06367    0.02088   3.049 0.0022942


> summary(z15)
Call:
bivbin2stg(formula1 = y1 ~ x, formula2 = y2 ~ x, formula3 = ~x,
     weights = wts, data = cotdeath, xs.includes = TRUE,
     method = "copula")
... ...

Association Model
Wald Tests:
  Df   Chi Pr(>Chi)
x  1 4.479  0.03432

Coefficients:
            Estimate Std. Error z value  Pr(>|z|)
(Intercept)   -2.589     0.7045  -3.676 0.0002372
x              1.938     0.9159   2.116 0.0343190
```

We now analyse this data using the `spml2` method. Recall that only two probability models are considered in this method.

```
> data(cotdeath)
> z16 <- bivbin2stg(y1~x*y2, y2~x, weights=wts, data=cotdeath,
          xs.includes=TRUE, method="spml2")

> summary(z16)
Call:
```

```
bivbin2stg(formula1 = y1 ~ x * y2, formula2 = y2 ~ x, weights = wts,
           data = cotdeath, xs.includes = TRUE, method = "spml2")

Stratum Counts Report:
               xStrat      1
obstype yStrat
retro   1               336
        2              1399
strata  1                 0
        2            135601

Models for prob of joint distribution of y1 and y2 given covariates

loglikelihood = -15864.31   using 6 parameters


Y1-Model
Wald Tests:
     Df    Chi  Pr(>Chi)
x     1 33.210 8.273e-09
y2    1 20.966 4.675e-06
x:y2  1  4.697 3.021e-02

Coefficients:
            Estimate Std. Error z value  Pr(>|z|)
(Intercept)  -4.3560     0.2129 -20.457 0.000e+00
x            -1.8989     0.3295  -5.763 8.273e-09
y2           -1.0857     0.2371  -4.579 4.675e-06
x:y2          0.7743     0.3573   2.167 3.021e-02

Y2-Model
Wald Tests:
  Df    Chi Pr(>Chi)
x  1 0.2144   0.6433

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  2.12711     0.1673  12.711   0.0000
x            0.09137     0.1973   0.463   0.6433
```

Again, as we have a saturated regression model, we obtain the same output for the model of interest as that obtained from the `spml1` method. We also note that the last two coefficients in the $Y_1|Y_2$-model (i.e. `y2` and `x:y2`) in this method are estimated identically as those in the association model obtained from the `spml1` method when the Palmgren model is used.

■

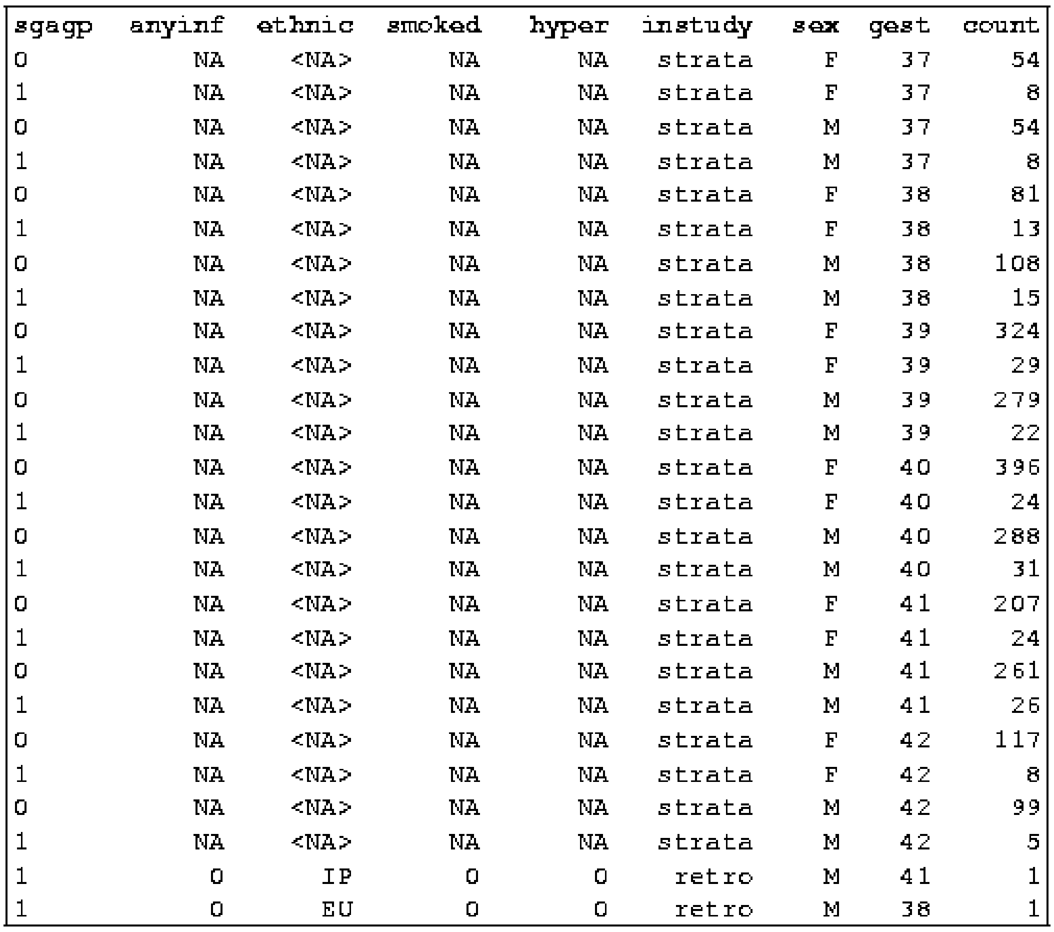
Under two-phase response-selective sampling, stratification is sometimes defined by not only the response (Y_1 here), but also some categorical V-variables other than the response that are fully observed. The second-phase data are then collected within each stratum cross-classified by both the response and V. We call it a "stratified" sample. Even if the actual data collection procedure didn't utilize this source of information, we find that introducing post-stratification on V can still be a good strategy to gain efficiency in analysis.

The function `bivbin2stg` can cope with this extra source of information using the argument `xstrata=c("vname1","vname2",...)` in the function call. The V-strata are then defined by the cross-classification of all variables listed in `xstrata`.

Example 7: *Placental infarctions in ABC study*

The data set for this example was derived from data collected in the ABC study described in Example 4. The incidence and associations of placental infarctions at term were investigated by Becroft *et al.* (2002) as one of the following secondary analyses in this study. Except for the original response variable (`sgagp`) that defines the baby's birthweight status classified as either "small for gestational age" (SGA) or "appropriate for gestational age" (AGA), we now have a new binary response of interest called `anyinf` which we describe in the next paragraph. If we define `sgagp` as Y_1 and `anyinf` as Y_2, Y_1 is fully observed for the whole first-phase population but Y_2 is not which has `NA`s for `strata` observations. Recall that the cutoff values for defining SGA babies differed by their gestational age and sex. Therefore, these two variables are also fully observed and available for defining the V-strata.

In particular, 509 placentas from women delivering SGA babies and 529 babies from women delivering AGA infants were examined using fixed protocols for infarctions. If any infarction was detected in the placenta, the baby was recorded as `anyinf=1` and `0` otherwise. The data frame `infarct` (see Figure 6) contains a subset of the data on these 1038 selected babies as well as artificial counts for `strata` observations generated using the knowledge that approximately 1 in 9 controls were selected from each V-stratum. Additional explanatory variables that are not in the variable list from Table 3 or have been modified, are described below.



sgagp	anyinf	ethnic	smoked	hyper	instudy	sex	gest	count
0	NA	<NA>	NA	NA	strata	F	37	54
1	NA	<NA>	NA	NA	strata	F	37	8
0	NA	<NA>	NA	NA	strata	M	37	54
1	NA	<NA>	NA	NA	strata	M	37	8
0	NA	<NA>	NA	NA	strata	F	38	81
1	NA	<NA>	NA	NA	strata	F	38	13
0	NA	<NA>	NA	NA	strata	M	38	108
1	NA	<NA>	NA	NA	strata	M	38	15
0	NA	<NA>	NA	NA	strata	F	39	324
1	NA	<NA>	NA	NA	strata	F	39	29
0	NA	<NA>	NA	NA	strata	M	39	279
1	NA	<NA>	NA	NA	strata	M	39	22
0	NA	<NA>	NA	NA	strata	F	40	396
1	NA	<NA>	NA	NA	strata	F	40	24
0	NA	<NA>	NA	NA	strata	M	40	288
1	NA	<NA>	NA	NA	strata	M	40	31
0	NA	<NA>	NA	NA	strata	F	41	207
1	NA	<NA>	NA	NA	strata	F	41	24
0	NA	<NA>	NA	NA	strata	M	41	261
1	NA	<NA>	NA	NA	strata	M	41	26
0	NA	<NA>	NA	NA	strata	F	42	117
1	NA	<NA>	NA	NA	strata	F	42	8
0	NA	<NA>	NA	NA	strata	M	42	99
1	NA	<NA>	NA	NA	strata	M	42	5
1	0	IP	0	0	retro	M	41	1
1	0	EU	0	0	retro	M	38	1

Figure 6: The Placental infarctions data

sex	F (female) or M (male).
gest	gestational age in weeks (37, 38, 39, 40, 41, 42).
ethnic	IP (Pacifican), MA (Maori), OA (Other Asian), OC (Chinese), OI (Indian) or RT (Others). (1 study individuals has NA)
smoked	smoking during pregnancy 1 or 0. (22 study individuals have NA)
agepreg	mother's age at this pregnancy. (21 study individuals have NA)

We first fit the data using the `spml1` method under the Palmgren model. The syntax is as follows.

```
> data(infarct)
> z17 <- bivbin2stg(sgagp~ethnic+smoked+hyper+mumwt+mumwtc2+agepreg,
                  anyinf~smoked+hyper+age1st, ~age1st, weights=count,
                  xstrata=c("sex","gest"), obstype.name="instudy",
                  data=infarct, xs.includes=TRUE, method="palmgren")
```

The variable named `mumwtc2` was obtained by squaring the differences between the variable `mumwt` and its mean value. Recall that it is usually advisable to centre any continuous X-variable being added to the regression model to prevent failure in algorithm. The two variables "`sex`" and "`gest`" are specified in `xstrata` to define addtional V-strata for analysis . If we look at the `Strata Counts Report` followed, we can see that there are in total 12 V-strata (named `xStrat` in the output) with their names clearly referenced in the following `key to x-Strat` report.

```
> summary(z17)
  ... ...

Observations deleted due to missing data:
 retro: 35 rows relating to 35 observations

Stratum Counts Report:
               xStrat  1  2  3  4   5   6   7   8   9  10  11  12
obstype yStrat
retro   1              8  8 13 15  29  22  24  31  24  26   8   5
        2              6  6  9 12  36  31  44  32  23  29  13  11
strata  1              0  0  0  0   0   0   0   0   0   0   0   0
        2             48 48 72 96 288 248 352 256 184 232 104  88

Models for prob of joint distribution of sgagp and anyinf given covariates

Key to x-Strat:
          1            2            3            4            5            6
sexF:gest37 sexM:gest37 sexF:gest38 sexM:gest38 sexF:gest39 sexM:gest39
```

```
          7             8             9            10            11            12
sexF:gest40 sexM:gest40 sexF:gest41 sexM:gest41 sexF:gest42 sexM:gest42

loglikelihood = -2622.846   using 18 parameters


Y1-Model
Wald Tests:
        Df    Chi  Pr(>Chi)
ethnic   6 13.118 4.120e-02
smoked   1 36.337 1.660e-09
hyper    1  5.584 1.812e-02
mumwt    1 21.637 3.295e-06
mumwtc2  1  5.704 1.692e-02
agepreg  1  7.250 7.091e-03

Coefficients:
             Estimate Std. Error z value  Pr(>|z|)
(Intercept)  2.357983  0.9910046  2.3794 1.734e-02
ethnicIP    -0.422015  0.3505923 -1.2037 2.287e-01
ethnicMA    -0.132421  0.4102123 -0.3228 7.468e-01
ethnicOA     0.500500  0.7698838  0.6501 5.156e-01
ethnicOC     0.710474  0.6494617  1.0939 2.740e-01
ethnicOI     1.388536  0.4655790  2.9824 2.860e-03
ethnicRT    -0.954606  0.9931126 -0.9612 3.364e-01
smoked       1.573849  0.2610906  6.0280 1.660e-09
hyper        0.856663  0.3625167  2.3631 1.812e-02
mumwt       -0.058104  0.0124914 -4.6515 3.295e-06
mumwtc2      0.001008  0.0004219  2.3884 1.692e-02
agepreg     -0.054777  0.0203441 -2.6925 7.091e-03

Y2-Model
Wald Tests:
       Df    Chi  Pr(>Chi)
smoked  1  4.616 0.0316760
hyper   1  4.673 0.0306457
age1st  1 14.577 0.0001345

Coefficients:
            Estimate Std. Error z value  Pr(>|z|)
(Intercept)  -5.1388    0.94657  -5.429 5.672e-08
smoked       -0.8927    0.41553  -2.148 3.168e-02
```

```
hyper          0.8471    0.39187   2.162 3.065e-02
age1st         0.1251    0.03277   3.818 1.345e-04

Association Model
Wald Tests:
       Df   Chi Pr(>Chi)
age1st  1 9.932 0.001624

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)   5.6390     1.7510   3.220 0.001280
age1st       -0.1979     0.0628  -3.152 0.001624
```

We then fit the other available models to the data. Their results are very close to one another but no longer identical for the marginal probabilities as they were in Example 6.

```
> summary(z18)
Call:
bivbin2stg(formula1 = sgagp ~ ethnic + smoked + hyper + mumwt + mumwtc2 +
           agepreg, formula2 = anyinf ~ smoked + hyper + age1st,
           formula3 = ~age1st, weights = count, xstrata = c("sex", "gest"),
           obstype.name = "instudy", data = infarct, xs.includes = TRUE,
           method = "bahadur")
... ...

loglikelihood = -2622.812   using 18 parameters


Y1-Model
Wald Tests:
        Df    Chi  Pr(>Chi)
ethnic   6 13.606 3.437e-02
smoked   1 35.185 2.998e-09
hyper    1  4.291 3.832e-02
mumwt    1 24.859 6.169e-07
mumwtc2  1  6.776 9.239e-03
agepreg  1  7.120 7.623e-03

Coefficients:
             Estimate Std. Error z value  Pr(>|z|)
(Intercept)  2.554869  1.0047152  2.5429 1.099e-02
ethnicIP    -0.415193  0.3485228 -1.1913 2.335e-01
ethnicMA    -0.240309  0.4169548 -0.5763 5.644e-01
```

```
ethnicOA      0.366071  0.7838269  0.4670 6.405e-01
ethnicOC      0.663465  0.6499321  1.0208 3.073e-01
ethnicOI      1.396151  0.4790277  2.9146 3.562e-03
ethnicRT     -1.773118  1.2845927 -1.3803 1.675e-01
smoked        1.562472  0.2634108  5.9317 2.998e-09
hyper         0.773174  0.3732522  2.0715 3.832e-02
mumwt        -0.061035  0.0122416 -4.9858 6.169e-07
mumwtc2       0.001084  0.0004166  2.6031 9.239e-03
agepreg      -0.054501  0.0204250 -2.6683 7.623e-03

Y2-Model
Wald Tests:
       Df    Chi  Pr(>Chi)
smoked  1  3.829 0.0503765
hyper   1  3.935 0.0473025
age1st  1 13.381 0.0002542

Coefficients:
            Estimate Std. Error z value  Pr(>|z|)
(Intercept)  -5.0894    0.99194  -5.131 2.886e-07
smoked       -0.8909    0.45529  -1.957 5.038e-02
hyper         0.8407    0.42383   1.984 4.730e-02
age1st        0.1248    0.03413   3.658 2.542e-04

Association Model
Wald Tests:
       Df   Chi Pr(>Chi)
age1st  1 9.727 0.001815

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  0.92807    0.31585   2.938 0.003300
age1st      -0.03269    0.01048  -3.119 0.001815


> summary(z19)
Call:
bivbin2stg(formula1 = sgagp ~ ethnic + smoked + hyper + mumwt + mumwtc2 +
           agepreg, formula2 = anyinf ~ smoked + hyper + age1st,
           formula3 = ~age1st, weights = count, xstrata = c("sex", "gest"),
           obstype.name = "instudy", data = infarct, xs.includes = TRUE,
           method = "copula")
```

```
... ...

loglikelihood = -2623.616   using 18 parameters


Y1-Model
Wald Tests:
        Df    Chi  Pr(>Chi)
ethnic   6 13.250 3.923e-02
smoked   1 35.181 3.004e-09
hyper    1  5.415 1.997e-02
mumwt    1 21.374 3.779e-06
mumwtc2  1  5.623 1.773e-02
agepreg  1  7.245 7.112e-03

Coefficients:
             Estimate Std. Error z value  Pr(>|z|)
(Intercept)  2.385893  1.0011318  2.3832 1.716e-02
ethnicIP    -0.435464  0.3522569 -1.2362 2.164e-01
ethnicMA    -0.124095  0.4173603 -0.2973 7.662e-01
ethnicOA     0.493857  0.7714617  0.6402 5.221e-01
ethnicOC     0.707135  0.6505054  1.0871 2.770e-01
ethnicOI     1.411132  0.4686395  3.0111 2.603e-03
ethnicRT    -0.917309  1.0020789 -0.9154 3.600e-01
smoked       1.555574  0.2622615  5.9314 3.004e-09
hyper        0.849607  0.3651212  2.3269 1.997e-02
mumwt       -0.058161  0.0125802 -4.6232 3.779e-06
mumwtc2      0.001012  0.0004266  2.3712 1.773e-02
agepreg     -0.055418  0.0205896 -2.6916 7.112e-03

Y2-Model
Wald Tests:
       Df    Chi  Pr(>Chi)
smoked  1  4.871 0.0273185
hyper   1  4.618 0.0316449
age1st  1 13.024 0.0003075

Coefficients:
            Estimate Std. Error z value  Pr(>|z|)
(Intercept)  -4.7754    0.89448  -5.339 9.360e-08
smoked       -0.9101    0.41237  -2.207 2.732e-02
hyper         0.8294    0.38596   2.149 3.164e-02
```

```
age1st        0.1133    0.03139   3.609 3.075e-04

Association Model
Wald Tests:
       Df   Chi Pr(>Chi)
age1st  1 7.312 0.006848

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  10.6714     3.9590   2.695 0.007029
age1st       -0.3722     0.1377  -2.704 0.006848


> summary(z20)
Call:
bivbin2stg(formula1 = sgagp ~ anyinf + ethnic + smoked + hyper + mumwt +
           mumwtc2 + agepreg + anyinf * age1st, formula2 = anyinf ~ smoked +
           hyper + age1st, weights = count, xstrata = c("sex", "gest"),
           obstype.name = "instudy", data = infarct, xs.includes = TRUE,
           method = "spml2")
... ...

loglikelihood = -2624.091   using 19 parameters


Y1-Model
Wald Tests:
              Df     Chi  Pr(>Chi)
anyinf         1  8.1344 4.343e-03
ethnic         6 13.1369 4.091e-02
smoked         1 34.4262 4.427e-09
hyper          1  4.8125 2.825e-02
mumwt          1 21.5138 3.513e-06
mumwtc2        1  5.2756 2.163e-02
agepreg        1  3.5806 5.846e-02
age1st         1  0.1547 6.941e-01
anyinf:age1st  1  7.5942 5.855e-03

Coefficients:
                Estimate Std. Error z value  Pr(>|z|)
(Intercept)    1.8766503  1.0888463  1.7235 8.479e-02
anyinf         4.8505593  1.7007012  2.8521 4.343e-03
```

```
ethnicIP      -0.4712683  0.3605822 -1.3070 1.912e-01
ethnicMA      -0.1411395  0.4409901 -0.3201 7.489e-01
ethnicOA       0.5006102  0.7839660  0.6386 5.231e-01
ethnicOC       0.6988698  0.6554965  1.0662 2.863e-01
ethnicOI       1.3916757  0.4804552  2.8966 3.773e-03
ethnicRT      -1.0146374  1.0064577 -1.0081 3.134e-01
smoked         1.6395796  0.2794395  5.8674 4.427e-09
hyper          0.8231381  0.3752222  2.1937 2.825e-02
mumwt         -0.0590960  0.0127409 -4.6383 3.513e-06
mumwtc2        0.0009963  0.0004338  2.2969 2.163e-02
agepreg       -0.0474487  0.0250755 -1.8922 5.846e-02
age1st         0.0110321  0.0280525  0.3933 6.941e-01
anyinf:age1st -0.1700654  0.0617126 -2.7558 5.855e-03

Y2-Model
Wald Tests:
       Df    Chi  Pr(>Chi)
smoked  1  5.100 0.0239231
hyper   1  4.661 0.0308616
age1st  1 11.780 0.0005986

Coefficients:
            Estimate Std. Error z value  Pr(>|z|)
(Intercept)  -4.5877    0.88995  -5.155 2.536e-07
smoked       -0.9368    0.41483  -2.258 2.392e-02
hyper         0.8441    0.39102   2.159 3.086e-02
age1st        0.1051    0.03062   3.432 5.986e-04
```

2.5.2 Sampling on whether or not $(Y_1 \neq 1, Y_2 \neq 1)$

In some situation, the case-control information on both Y_1 and Y_2 is available for the first-phase population. Subsampling into the second phase is dependent on whether or not $Y_1 \neq 1$ and $Y_2 \neq 1$, i.e., both belong to the control groups. Thus the Y-stratum is defined as either in group $(Y_1 \neq 1, Y_2 \neq 1)$ (named `yStrat=2`) or not in this group (named `yStrat=1`). This type of sampling occurs, for example, when there is a registry of disease cases with data collected on mother/daughter pairs. The pairs appearing in the registry are then supplemented by a sample of pairs where neither is a case (i.e. control/control).

The function `bivbin2stg` can deal with this situation using an extra argument `y1samp = FALSE` (the default is `TRUE`) in the function call. Only the `spml1` method has been implemented for this sampling scheme. As both Y_1 and Y_2 are fully observed at the first phase (they are treated as binary variables here), there should be no `NA`'s for `strata` observations except for some of the X-variables.

Example 8: *An artificial data set*

We now consider an artificial data set generated using this sampling scheme. The data frame is called `dat00` (see Figure 7) which contains six variables named `x, y1, y2, v, wts` and `obstype`. Only the binary variable `x`, the covariate of interest, contains `NA`s. Another binary variable `v` is fully observed and treated as the V-variable defining two strata. The total population size is $N = 10000$ with the total sample size of $n = 5000$.

The data was generated as follows. We first generated two binary response varibles Y_1 and Y_2 of length N using binomial distribution with $p = 0.2$. We then generated the binary x-variable using the same distribution with $p = 0.6$ if both Y_1 and Y_2 are zero, and $p = 0.4$ otherwise. The binary v-variable was generated in the same way with $p = 0.5$. If $v = 0$, we replaced it with a value of 2 indicating the second V-strata. Within each V-stratum, we randomly selected equal number of cases and controls (i.e. n/4) in correspondence to whether or not $(Y_1 \neq 1, Y_2 \neq 1)$. They became our `retro` observations. The rest were `strata` observations and all x-values for this part of data became "missing".

We first fit this data using the Palmgren model. The fitting syntax is as follows:

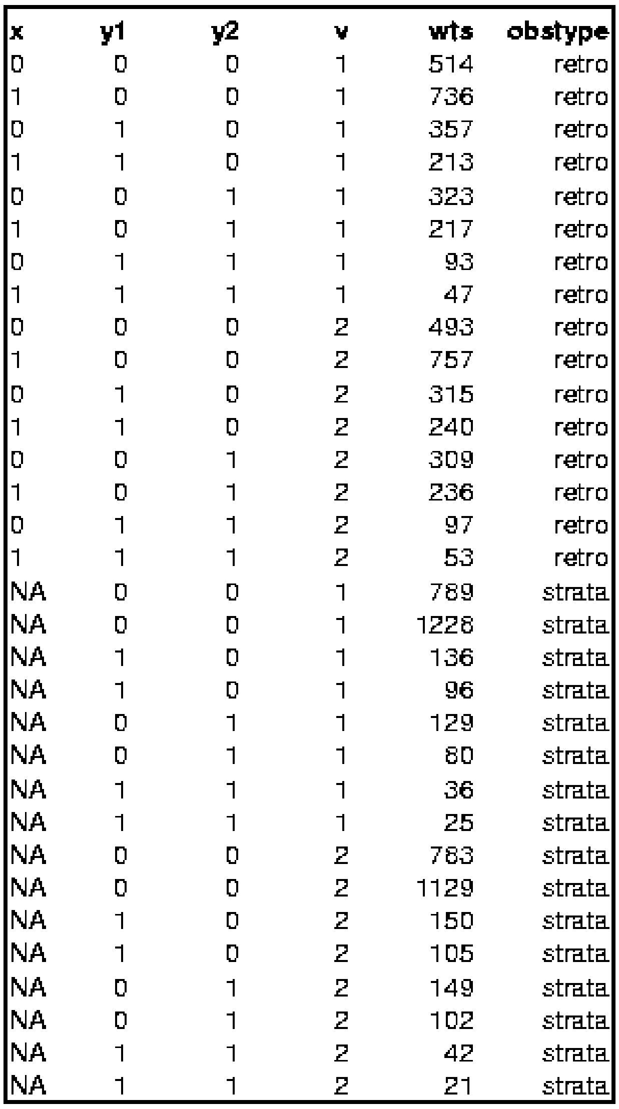
```
> data(dat00)
> z21 <- bivbin2stg(y1~x, y2~x, ~x, weights=wts, data=dat00,
    y1samp=FALSE, xstrata="v", xs.includes=FALSE, method="palmgren")
```

The argument `y1samp=FALSE` has been used specifically to distinguish from the previous sampling scheme (i.e. sampling on Y_1). The following summary of regression output is then obtained:

```
> summary(z21)
Call:
bivbin2stg(formula1 = y1 ~ x, formula2 = y2 ~ x, formula3 = ~x,
           weights = wts, xstrata = "v", data = dat00, xs.includes = FALSE,
           y1samp = FALSE, method = "palmgren")


Stratum Counts Report:
               xStrat    1    2
obstype yStrat
retro   1             1250 1250
        2             1250 1250
strata  1              502  569
        2             2017 1912


Models for prob of joint distribution of y1 and y2 given covariates
```



x	y1	y2	v	wts	obstype
0	0	0	1	514	retro
1	0	0	1	736	retro
0	1	0	1	357	retro
1	1	0	1	213	retro
0	0	1	1	323	retro
1	0	1	1	217	retro
0	1	1	1	93	retro
1	1	1	1	47	retro
0	0	0	2	493	retro
1	0	0	2	757	retro
0	1	0	2	315	retro
1	1	0	2	240	retro
0	0	1	2	309	retro
1	0	1	2	236	retro
0	1	1	2	97	retro
1	1	1	2	53	retro
NA	0	0	1	789	strata
NA	0	0	1	1226	strata
NA	1	0	1	136	strata
NA	1	0	1	96	strata
NA	0	1	1	129	strata
NA	0	1	1	80	strata
NA	1	1	1	36	strata
NA	1	1	1	25	strata
NA	0	0	2	783	strata
NA	0	0	2	1129	strata
NA	1	0	2	150	strata
NA	1	0	2	105	strata
NA	0	1	2	149	strata
NA	0	1	2	102	strata
NA	1	1	2	42	strata
NA	1	1	2	21	strata

Figure 7: An artificial data set

```
Key to x-Strat:
 1  2
v1 v2

loglikelihood = -47967.29   using 6 parameters


Y1-Model
Wald Tests:
  Df   Chi  Pr(>Chi)
x  1 64.63 8.882e-16

Coefficients:
            Estimate Std. Error z value  Pr(>|z|)
(Intercept)  -0.3227    0.04177  -7.727 1.110e-14
x            -0.5195    0.06462  -8.039 8.882e-16

Y2-Model
Wald Tests:
  Df   Chi  Pr(>Chi)
x  1 45.72 1.366e-11

Coefficients:
            Estimate Std. Error z value  Pr(>|z|)
(Intercept)  -0.4033    0.04231  -9.533 0.000e+00
x            -0.4389    0.06492  -6.761 1.366e-11

Association Model
Wald Tests:
  Df   Chi Pr(>Chi)
x  1 9.557 0.001992

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)   -1.387    0.09467 -14.653 0.000000
x              0.483    0.15624   3.091 0.001992
```

We see that `x` is highly significant in all three models. As in Example 6, we only have one binary explanatory variable `x`. Therefore, both the Bahadur and Copula models will give the same regression output as above for the two marginal probabilities of Y_1 and Y_2 except for the association model. We do not present them here.

2.6 Bivariate Binary Linear Regression (bivlocsc2stg)

In this section, we consider the data collected under the same two-phase case-control sampling scheme but with Y_2 - the true response of interest - being a continuous variable that is fully observed at the second phase.

Since now Y_1 (the binary variable defining case-control status of all subjects) and Y_2 are different types of correlated measures, we use a similar approach as for bivariate binary regression called `spml2`. In particular, we model the joint distribution of Y_1 and Y_2 in terms of a conditional factorisation

$$pr(Y_1, Y_2, X) = pr(Y_1|Y_2, X; \theta_1)pr(Y_2|X; \theta_2)pr(X)$$

Both $pr(Y_1|Y_2, X)$ and $pr(Y_2|X)$ are modelled parametrically, using logistic and linear regression models respectively. As Y_2 is a continuous variable, the evaluation of $pr(Y_1|X)$ at the first phase in the loglikelihood can be a bit tricky.

The application of this model is very similar to those we have developed in Section 2.2 (Binary), 2.3 (Linear) and 2.4 (Bivariate Binary). Therefore, we will only introduce the syntax for the function call which is named as `bivlocsc2stg`.

Using the following R codes, we have created a simple sample data with only one covariate x. The interaction between Y_2 and x is particularly considered in the Y_1-model for generalisation. Categorical information of x based on defined cutoff points is used for stratification at the first phase. Using the defined parameter values, the proportion of cases in the population is approximately 10%.

```
### Data generation ###
N <- 5000
x <- rnorm(N)
eps <- rnorm(N)

theta2 <- c(0.5,1,0)
y2 <- theta2[1]+theta2[2]*x+exp(theta2[3])*eps

theta1 <- c(-3,-0.5,1,0.5)
eta1 <- theta1[1]+theta1[2]*y2+theta1[3]*x+theta1[4]*y2*x
p1 <- plogis(eta1)
y1 <- 1*(runif(N)<p1)

xcut <- c(-30,-1,0,1,30)
xstrata <- as.numeric(cut(x,xcut))
```

```
indca <- (1:N)[y1==1]
indct <- sample((1:N)[y1==0],length(indca))
ind <- sort(c(indca,indct))
rest <- (1:N)[-ind]
obstype <- rep("retro",N)
obstype[rest] <- "strata"
y2[rest] <- NA; x[rest] <- NA
dat <- data.frame(y1,y2,x,xstrata,obstype)

### Proportion of cases in population ###
prca <- length(indca)/N
prca
```

The results are as follows. We see that the parameter estimates are fairly close to their true values.

```
> z22 <- bivlocsc2stg(y1~y2*x, y2~x, ~1, xstrata="xstrata",
                      data=dat, xs.includes=FALSE)

> summary(z22)
Call:
bivlocsc2stg(formula1 = y1 ~ y2 * x, formula2 = y2 ~ x, formula3 = ~1,
 xstrata = "xstrata", data = dat, xs.includes = FALSE)

Stratum Counts Report:
               xStrat    1    2    3    4
obstype yStrat
retro   1               59   69  115  253
        2               86  204  156   50
strata  1                0    0    0    0
        2              688 1447 1394  479

Models for prob of joint distribution of y1 and y2 given covariates

Key to x-Strat:
       1        2        3        4
xstrata1 xstrata2 xstrata3 xstrata4

loglikelihood = -8258.786   using 7 parameters


Y1-Model
Wald Tests:
     Df    Chi  Pr(>Chi)
y2    1  60.48 7.438e-15
x     1 126.85 0.000e+00
y2:x  1 175.94 0.000e+00

Coefficients:
            Estimate Std. Error z value  Pr(>|z|)
(Intercept)  -2.9262    0.08479 -34.513 0.000e+00
y2           -0.6105    0.07850  -7.777 7.327e-15
x             1.2192    0.10825  11.263 0.000e+00
y2:x          0.4833    0.03644  13.264 0.000e+00

Y2-Model (Location)
Wald Tests:
  Df  Chi Pr(>Chi)
x  1 1292        0

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)   0.5565    0.03769   14.76        0
x             0.9851    0.02740   35.95        0

Y2-Model (Scale)
Coefficients:
  Estimate Std. Error    z value   Pr(>|z|)
  -0.01946    0.02185   -0.89056    0.37317
```

2.7 Random Effects Models for Clustered Binary Data (rclusbin)

2.7.1 Model description

One way in which this type of data can arise is as follows. In medical studies, we sometimes collect information on not only the individual sampled into a study (named "*probands*" afterwards) but also their family members. If we treat each family as an independent cluster, we have clustered binary data when the case-control status of all family members and their explanatory variables are recorded. As each family can have different number of persons, clusters can be of various sizes.

In this section, we consider random effects models in which cluster-specific random intercepts are used to account for correlations within clusters. Even if we only have information on the probands, we can still apply this model with the individual being thought of as a cluster of size one. Data sets that can be analysed under binary regression models can also be analysed under binary random intercept models.

Suppose a random effect a_i is specified for the i^{th}-cluster which has a Normal distribution with mean 0 and variance σ^2. We will work with $a_i = e^w \epsilon_i$ where $w = log(\sigma)$ and $\epsilon_i \sim N(0,1)$. The following probability model is considered:

$$Pr(\boldsymbol{y}_i|a_i, \boldsymbol{x}_i) = \prod_{j=1}^{J_i} Pr(\boldsymbol{y}_{ij}|a_i + \eta_{ij}), \quad \text{where } \eta_{ij} = \boldsymbol{x}_{ij}^T \beta.$$

Here, J_i indicates the total number of individuals in i^{th}-cluster and η_{ij} is the linear predictor related to the vector $\boldsymbol{x}_{ij}$. Thus,

$$Pr(\boldsymbol{y}_i|\boldsymbol{x}_i) = \int \prod_{j=1}^{J_i} Pr(\boldsymbol{y}_{ij}|e^w \epsilon_i + \eta_{ij}) \phi(\epsilon) d\epsilon = \int \left[\prod_{j=1}^{J_i} P_{ij} \right] \phi(\epsilon) d\epsilon$$

where $P_{ij} = Pr(\boldsymbol{y}_{ij}|e^w \epsilon_i + \eta_{ij})$. The parameters of interest are then denoted by $\vartheta = (\beta, w)$.

In the `missreg` library, the function written to carry out such analyses is called `rclusbin`, which can be used for data collected under both prospective and retrospective sampling schemes. Three different strategies have been particularly considered in the function to define appropriate Y-strata for retrospective sampling:

1. *Proband only*
   The Y-strata are defined based on the case-control status of the proband only.

2. *All controls*
   The Y-strata are defined based on the case-control status of all members in the same cluster. If any of the subjects in the cluster are cases, they belongs to $Y = 1$ and otherwise $Y = 0$.

3. *Gamma probabilities*
   The Y-strata are defined based on the case-control status of all members in the same cluster with probabilities depending on two gammas.

$$Pr(Ystrata = 1|i^{th}cluster) = \begin{cases} 0, & \sum_j y_{ij} = 0 \\ \gamma_1, & \sum_j y_{ij} = 1 \\ \gamma_2, & \sum_j y_{ij} > 1 \end{cases} .$$

Some important details are discussed below.

1. *Cluster/Intra-Cluster membership*
   Because we have clustered binary data and we assume that the input data frame consists of one row per individual, the specification of cluster membership is required in the function call using the parameter `ClusInd`. If there is only one individual in each of the clusters (e.g. when we analyse simple case-control data using a random intercepts model), `ClusInd` can be left as `NULL` (the default). The function automatically assigns a cluster variable `(1:n)`, which is of the same length as other variables in the data frame, whenever `ClusInd=NULL`.

   If some clusters contain more than one subject, the program assumes that the proband is the first one unless additional information is supplied via the argument `IntraClus`. When `IntraClus` is present, all subjects in the cluster will be sorted by their intra-cluster ID and the one with the smallest ID is treated as the proband.

2. *Extra V-stratification*
   As in other sections, V-stratification can be considered as either a real stratification above Y-strata, or a post-stratification aimed at increasing the efficiency of the method. Such information is again provided using the parameter `xstrata` in the function call. When we have strata, each subject is not only classified by their cluster membership but also the stratum they belong to. Therefore, how clusters and strata are related in the data becomes very important.

   The function `rclusbin` can currently only deal with the situation in which clusters are defined within strata (i.e. individuals in the same cluster must be in the same stratum). This puts limits on the V-variables that can be used. If they are not constant within a cluster, the program will stop and print out an error message.

3. *Specification of Weights*
   The `weights` variable used in `rclusbin` function call is treated in one of two ways depending upon whether there are any clusters of size greater than one. If no `ClusInd` values repeat, the programme decides all clusters are of size one. The `weights` variable gives the number of clusters=individuals with this set of variable values. If there are any repeated values of `ClusInd`, the programme treats each row as per individual rather than a cluster of size one. The `weights` variable so provided gives the number of individuals with this set of variable values within a cluster.

4. *Starting values provided by* `start/Qstart/sigma`
   In `rclubsin` function call, the parameter `start` is used to provide starting values for $\vartheta = (\boldsymbol{\beta}, w)$ with $w = \log(\sigma)$. A starting value of σ alone can also be provided

using the parameter `sigma` in the function call when the starting values for $\boldsymbol{\beta}$ are hard to obtain. As in other functions, the starting values for $\boldsymbol{Q}$, which is defined as the probabilities falling into each of the Y-strata in the population within each V-stratum[4], are provided via the argument `Qstart`.

Unfortunately, we do not have good strategies for obtaining an appropriate starting value for σ. If no starting value for σ is supplied, the program uses $\sigma = 0.5$ as the default. If the programme cannot converge in one fit, the user has to increase the number of iterations (the default is 20) in the function `mlefn` (See Section 4 for more detail) in the hope that the programme finally converges.

When we have observations of the type `"strata"`, there are some obvious strategies for obtaining starting values for $\boldsymbol{\beta}$ (based on weighted binary regression) and $\boldsymbol{Q}$ if the user has not provided one. We calculate the N-matrix with rows and columns in correspondence to Y- and V-strata using information collected on all `"strata"` observations.

For the case in which starting values for $\boldsymbol{\beta}$ and $\boldsymbol{Q}$ can also be calculated and sometimes work when there are no `"strata"` observations in the data frame, is when we have more than one individual in some of the clusters and the data are collected based on the case-control information of the probands. The population counts can be roughly estimated using the total number of cases and controls in the data regardless of their proband status. In all other cases, the user must supply starting values for both $\boldsymbol{\beta}$ and $\boldsymbol{Q}$.

In following three subsections, we will use examples to illustrate the type of analyses we carry out and explain the summary of results we obtain.

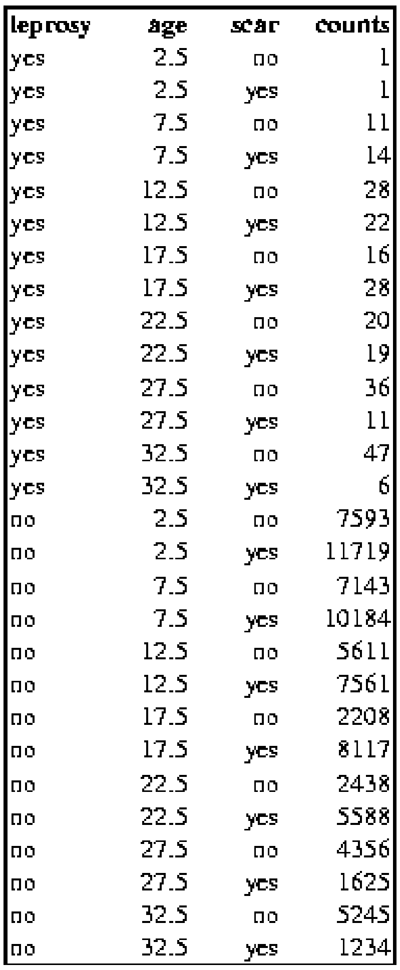
2.7.2 Prospective sampling

The function `rclusbin` can be applied to data collected prospectively, although its primary purpose is for retrospectively sampled data.

Example 9: *Prospective leprosy data*

Let's start from a simple example. Recall in Example 1 we used the leprosy data (see Figure 1) which were collected as a case-control sample with post-stratification on age groups. The covariate of interest `scar` was only observed for the cases and those controls sampled. We now consider the original prospective BCG scar data set from which the Example 1 data set was subsampled. These data, obtained from Clayton and Hills (1993,

[4]See Section 3.2.3 to 3.2.4 for details



leprosy	age	scar	counts
yes	2.5	no	1
yes	2.5	yes	1
yes	7.5	no	11
yes	7.5	yes	14
yes	12.5	no	28
yes	12.5	yes	22
yes	17.5	no	16
yes	17.5	yes	28
yes	22.5	no	20
yes	22.5	yes	19
yes	27.5	no	36
yes	27.5	yes	11
yes	32.5	no	47
yes	32.5	yes	6
no	2.5	no	7593
no	2.5	yes	11719
no	7.5	no	7143
no	7.5	yes	10184
no	12.5	no	5611
no	12.5	yes	7561
no	17.5	no	2208
no	17.5	yes	8117
no	22.5	no	2438
no	22.5	yes	5588
no	27.5	no	4356
no	27.5	yes	1625
no	32.5	no	5245
no	32.5	yes	1234

Figure 8: The prospective leprosy data

p. 230), are shown in Figure 8 with all variables being fully observed.

Because this is a prospective data set, tbe `obstype` variable should only contain `"uncond"` observations. Recall that we needn't specify any cluster variable in the function call when we apply the random intercept model with the individual being thought of as a cluster of size one. A cluster variable `(1:n)` is automatically generated in the function whenever `ClusInd=NULL` (the default) is specified. The fitting syntax is as follows:

```
> data(leprosypros)
> leprosypros$age.trans <- 100*(leprosypros$age+7.5)^-2
> leprosypros$obstype <- rep("uncond",dim(leprosypros)[1])


> z23 <- rclusbin(leprosy~age.trans + scar, weights=counts,
                    data=leprosypros, linkname="logit")
```

Here, we use the same regression model as in Example 1. The variable `counts` is provided as a `weights` variable to count for the number of clusters=individuals with the same set of covariable values. Recall that the function `bin2stg` fits binary regression models using several links which are specified using the parameter `linkname` and the default is `"logit"` (other choices are `"probit"` and `"cloglog"`). We have the same options here.

We found that the program could not converge from the default starting value of $\sigma = 0.5$ but converged much faster when a `sigma` value of greater than 1 was supplied in the call. When `simga=2.5` is specified, for example, the program converges in one fit and the following results are obtained:

```
> summary(z23)
Call:
 rclusbin(formula = leprosy ~ age.trans + scar, weights = counts,
          data = leprosypros, linkname="logit", sigma=2.5)



Cluster Size Report:
                xStrat     1
obstype Clusize
uncond  1              80882

Stratum Counts Report:
        xStrat     1
obstype
uncond         80882


Model for prob of leprosy=yes (y=1) given covariates

loglikelihood = -1644.876  using 4 parameters



Wald Tests:
          Df    Chi  Pr(>Chi)
age.trans  1 19.422 1.048e-05
scar       1  6.724 9.512e-03

Coefficients:
            Estimate Std. Error z value  Pr(>|z|)
(Intercept)  -6.9956     2.9744  -2.352 1.868e-02
```

```
age.trans    -4.5131     1.0241  -4.407 1.048e-05
scar         -0.6385     0.2462  -2.593 9.512e-03
logsigma      0.8843     0.6551   1.350 1.771e-01
```

In the `Cluster Size Report`, information is provided on how many clusters we have in the data for each `obstype` at various sizes within each V-stratum (named `"xStrat"` in the table). As in other functions, the next table `Stratum Counts Report` tells us how many clusters we have in the data for each `obstype` within each Y-stratum and V-stratum. Because we have a prospective sample here, there are no Y-strata and counts are only reported within each of the V-strata.

In fact, whether or not information on V-stratification is provided in a prospective data set like this, will make no difference to the regression results although the `Stratum Counts Report` is different with or without V-strata. Note that when observations with `obstype=="strata"` are considered, the data set is no longer treated as “prospective” in our definition.

From the table of `Coefficients`, we see that the term `logsigma` is non-significant (p-value is 0.177) in the model with an estimate of 0.8843, which gives $\widehat{\sigma} = 2.4$. When σ is close to zero and we fix its value between iterations, both $\widehat{\beta}$ and their standard errors are almost identical to those obtained using the function `bin2stg`.

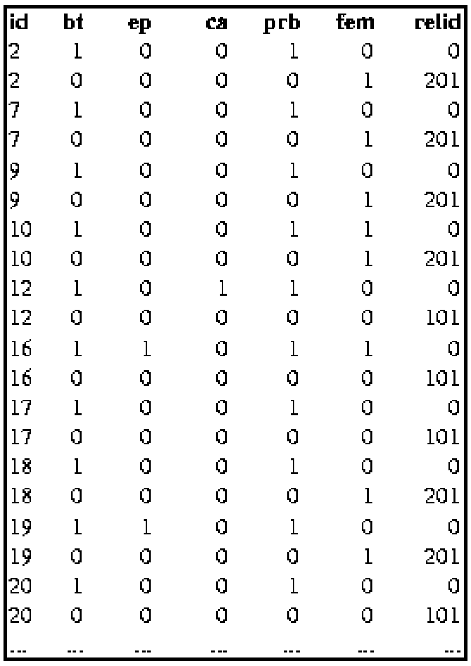
2.7.3 Sampling on probands only

When we have information on the case-control status of all probands (say $Y_{(1)} = 1$ and $Y_{(1)} = 0$), we take random samples from the $Y_{(1)} = 1$ and $Y_{(1)} = 0$ populations respectively with the possibility of extra stratification on other fully-observed categorical variables V. Recall that we always assume the clusters are defined within strata so that individuals in the same cluster must be in the same stratum. Information on other family members of those selected probands is then collected with both $\boldsymbol{Y}$ and $\boldsymbol{X}$, the covariates of interest, being fully observed.

Example 10: *Brain-cancer Pairs data*

The brain-cancer pairs data were constructed from data obtained by Wrensch *et al.* (1997). It contains 785 different clusters, all of size two, which are pairs of siblings and constructed by taking the proband and the sibling closest in age to him/her. There are 385 proband cases ($Y_{(1)} = 1$) and 400 proband controls ($Y_{(1)} = 0$) in the data set.

The data for the first 10 clusters are shown in Figure 9. Cluster membership is specified using the variable `"id"`. Note that each value of ID is repeated twice in the data frame. This indicates that we have two individuals in each cluster. The variable `"relid"`



id	bt	ep	ca	prb	fem	relid
2	1	0	0	1	0	0
2	0	0	0	0	1	201
7	1	0	0	1	0	0
7	0	0	0	0	1	201
9	1	0	0	1	0	0
9	0	0	0	0	1	201
10	1	0	0	1	1	0
10	0	0	0	0	1	201
12	1	0	1	1	0	0
12	0	0	0	0	0	101
16	1	1	0	1	1	0
16	0	0	0	0	0	101
17	1	0	0	1	0	0
17	0	0	0	0	0	101
18	1	0	0	1	0	0
18	0	0	0	0	1	201
19	1	1	0	1	0	0
19	0	0	0	0	1	201
20	1	0	0	1	0	0
20	0	0	0	0	0	101
...	...	...	...	...	...	...

Figure 9: The brain cancer data

is a within-cluster identifier specifying the intra-cluster membership. Recall we can use it to indicate the position of the proband in each cluster when necessary. The response of interest is "bt".

We now analyse this data set using the function `rclusbin` as in Example 9 with different values of arguments in the function call. The syntax is as follows.

```
> data(brainpairs)
> brainpairs$obstype <- rep("retro",dim(brainpairs)[1])

> z24 <- rclusbin(bt ~ ep + ca, ClusInd=id, IntraClus=relid,
                  data=brainpairs, retrosamp="proband")
```

Since the sub-sampling was based on the probands only, `retrosamp="proband"` needs to be specified in the function call which is the default one. Because there is no `obstype` variable in the original data frame, we need to specify one. We have set all observations to being of obstype "retro". The regression output is summarised below.

```
> summary(z24)
Call:
 rclusbin(formula = bt ~ ep + ca, ClusInd = id, data = brainpairs,
          retrosamp="proband", IntraClus = relid)
```

```
Cluster Size Report:
                xStrat   1
obstype Clusize
retro   2             785

Stratum Counts Report:
               xStrat   1
obstype yStrat
retro   1            385
        2            400
Cluster Counts Reports:


Model for prob of bt=1 (y=1) given covariates

loglikelihood = -5261.996  using 4 parameters


Wald Tests:
   Df     Chi  Pr(>Chi)
ep  1 16.6644 4.461e-05
ca  1  0.1739 6.766e-01

Coefficients:
            Estimate Std. Error z value  Pr(>|z|)
(Intercept) -5.07788     0.9099 -5.5810 2.392e-08
ep           1.57094     0.3848  4.0822 4.461e-05
ca           0.09496     0.2277  0.4171 6.766e-01
logsigma    -0.58624     1.2785 -0.4585 6.466e-01
```

Note that the program converges from the default starting values strategy. Both the `"ca"` and `"logsigma"` terms are non-significant in the model. If we look at the `Stratum Counts Report`, we see that there are two Y-strata defined by $Y_{(1)} = 1$ (`yStrat=1`) and $Y_{(1)} = 0$ (`yStrat=2`) respectively. There are no V-strata.

■

Recall that when all clusters are of size 1, sampling probands only simply gives us a usual case-control sample. We can either use the function `bin2stg` to fit ordinary binary regression models, or use the function `rclusbin` to fit a random intercept model using the same link.

Example 1 (continue): *Leprosy data*

In Example 1 we have analysed the leprosy data with an ordinary logistic regression model using the function `bin2stg`. We now re-analyse this data with the random intercepts version of this model using the function `rclusbin`. The fitting syntax is as follows.

```
> data(leprosy1)
> leprosy1$age.trans <- 100 * (leprosy1$age + 7.5)^-2
> z25 <- rclusbin(leprosy~age.trans + scar, weights=counts,
                  xstrata="age", data=leprosy1, xs.includes=TRUE,
                  start=c(-4.48,-4.09,-0.42,log(1)), retrosamp="proband")
```

Here, we have used the results obtained from `summary(z1)` in Example 1 as starting values for $\boldsymbol{\beta}$ and a positive value of 1 as starting value for σ. The programme could not converge in one fit which only took 20 iterations. The user is required to either increase the number of iterations[5], or refit the model using $\widehat{\vartheta}$ obtained in the last iteration as new starting values. The following summary of results is obtained:

```
> summary(z25)
... ...

Cluster Size Report:
                xStrat     1     2     3     4     5     6     7
obstype Clusize
retro   1                 57    86   100    71    60    69    77
strata  1              19257 17266 13122 10298  8005  5959  6455

Stratum Counts Report:
               xStrat     1     2     3     4     5     6     7
obstype yStrat
retro   1                 2    25    50    44    39    47    53
        2                55    61    50    27    21    22    24
strata  1                 0     0     0     0     0     0     0
        2             19257 17266 13122 10298  8005  5959  6455


Key to x-Strat:
     1       2       3       4       5       6       7
 age2.5  age7.5 age12.5 age17.5 age22.5 age27.5 age32.5
```

[5]The command used in the function call would be `"control=mlefn.control(niter=35)"`. See the help file for `mlefn.control` for more detail.

```
Model for prob of leprosy=yes (y=1) given covariates

loglikelihood = -3900.178  using 4 parameters


Wald Tests:
          Df    Chi  Pr(>Chi)
age.trans  1 20.284 6.674e-06
scar       1  3.277 7.027e-02

Coefficients:
            Estimate Std. Error z value  Pr(>|z|)
(Intercept)  -8.1519     3.2432  -2.514 1.195e-02
age.trans    -4.8132     1.0687  -4.504 6.674e-06
scar         -0.5792     0.3200  -1.810 7.027e-02
logsigma      1.0858     0.5158   2.105 3.530e-02
```

As in Example 9, the variable `counts` is used as a `weights` variable to provide the number of clusters=individuals with the same set of covariable values. If we compare the `Stratum Counts Report` with that provided by `bin2stg` (see `summary(z1)`), we find that they are identical. The `Coefficients` table shows a significant `logsigma` term with a p-value of 0.035.

If we refit the above data using the starting values `(-4.48,-4.09,-0.42,log(0.001))`[6] and fix the `logsigma` parameter so that it cannot change as iterations proceed, we obtain the following coefficents table:

```
Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -4.4809     0.1144 -39.179  0.00000
age.trans    -4.0906     0.4495  -9.101  0.00000
scar         -0.4212     0.1785  -2.360  0.01827
logsigma     -6.9078        NaN     NaN      NaN
```

We see that, ignoring the last row which corresponds to the fixed tiny value for σ, this coefficients table is identical to that presented in `summary(z1)` from the ordinary logistic regression fit. This is what we would expect. The loglikelihood has the same value, too.

[6]The last term indicates a very small σ value (=0.001) and we call it a `logsigma` parameter.

2.7.4 Sampling on all family members

Recall in Section 2.4 we talked about two sampling schemes. The second-phase data can be either collected based on the binary information of Y_1 (the original response of interest), or whether or not both Y_1 and Y_2 (the new response of interest) belong to the control groups. In previous subsection, we choose our sample based on the case-control status of the probands only. We now consider another sampling scheme that is similar to those we discussed in Section 2.4.2.

Consider a situation in which all cases of disease are centrally registered. We define two Y-strata. The first consists of clusters containing at least one case which can be identified from the register. The second contains clusters with no cases, i.e. $Y_{(1)} \neq 1, ..., Y_{(J_i)} \neq 1$. Here, J_i indicates the total number of individuals in i^{th} cluster. This sampling strategy is what we have called "All controls" in Section 2.5.1.

Example 11: *An artificial data set (2)*

We now consider another artificial data set. Suppose we have two individuals in each cluster with their case-control information on Y being fully observed. If both of them are controls, the corresponding cluster belongs to the second Y-stratum (`yStrat=2`). If any one of them is a case, the cluster belongs to the first Y-stratum with `yStrat=1`. There is no extra V-stratification and only one X-variable is considered.

Let $N = 10000$ and $n = 1000$. We first generate a binary X-variable of length $2N$ from a Binomial distribution with $p = 0.5$. We then generate a random effect variable of length N from a Normal distribution with mean 0 and standard deviation 2 (i.e. $\sigma = 2$). As this variable should be constant within a cluster, we repeat its values twice assuming that the first N observations indicate the first subject in each cluster and the last N observations indicate the second subject in each cluster. The derived variable is of length $2N$ and used as the random effect $\boldsymbol{a}$ in the model. Let $\beta = (-3, 1)$. The response variable Y is next generated from a Binomial distribution with

$$p = \frac{\exp(\boldsymbol{a} + \boldsymbol{x}^T\beta)}{1 + exp(\boldsymbol{a} + \boldsymbol{x}^T\beta)}$$

At the next step, we define two Y-strata based on the case-control information of the two subjects in each cluster. Within each Y-stratum, we randomly choose $n/2$ clusters. Subjects in the chosen clusters are defined as `"retro"` observations. The rest are `"strata"` observations and their X values become *missing*. The data frame contains four variables: the response variable `y`, the explanatory variable `x`, the `obstype` variable and a variable specifying the cluster membership.

We now analyse this data using the function `rclusbin`. The fitting syntax is as follows.

```
> rdat00 <- data.frame(y=y, x=x, obstype=obstype, cluster=rep(1:N, 2))
> z26 <- rclusbin(y~x, ClusInd=cluster, data=rdat00, retrosamp="allcontrol")
```

Here we have used the parameter `retrosamp="allcontrol"` in the function call to distinguish from other sampling strategies. The results are then summarised below.

```
> summary(z26)
Call:
 rclusbin(formula = y ~ x, ClusInd = cluster, data = rdat00,
          retrosamp="allcontrol")



Cluster Size Report:
                xStrat    1
obstype Clusize
retro   2              1000
strata  2              9000

Stratum Counts Report:
               xStrat    1
obstype yStrat
retro   1              500
        2              500
strata  1             2248
        2             6752



Model for prob of y=1 (y=1) given covariates

loglikelihood = -13306.00  using 3 parameters



Wald Tests:
  Df   Chi Pr(>Chi)
x  1 53.31 2.85e-13

Coefficients:
            Estimate Std. Error z value  Pr(>|z|)
(Intercept)  -3.0791    0.12961 -23.756 0.000e+00
x             1.0439    0.14298   7.301 2.851e-13
logsigma      0.7447    0.05783  12.877 0.000e+00
```

We see that all terms are highly significant in the model. The parameter estimates are fairly close to their true values `(-3, 1, log(2))`.

■

We can also use pre-specified gamma probabilities to define Y-stratum for a cluster based on the case-control status of all members. The principle is that those clusters with more than one case are more likely to be in the first stratum (i.e. $Y = 1$) than those with one case only. When none of the members in a cluster are cases, they all belong to the second stratum with $Y = 0$. Therefore, γ_2 should be no less than γ_1 and normally assumed twice as large as γ_1 for simplicity.

When this sampling stragy is used, strata membership for each cluster must be specified in the function call using `ystrata`. The gamma probabilities should also be provided.

3 Semiparametric Maximum Likelihood

All functions we have implemented so far in the `missreg` library fit various regression models to data with the two-phase missingness structure by semiparametric maximum likelihood. If we want to know more about how the program engine works in each of the functions, we first need to understand the underlying method that these functions are based on.

Scott and Wild (2003) have given detailed description on semiparametric maximum likelihood approach for two-phase data. We will only represent some important ideas in this section.

3.1 Introduction

We consider a unified method for fitting essentially arbitrary regression models to a large class of two-phase missing data and/or response-selective sampling problems using semiparametric maximum likelihood. The profile likelihood methods discussed in Scott and Wild (1997), Lawless *et al.* (1999) and Neuhaus *et al.* (2002) have been developed and implemented. It began as the manuscript Lawless *et al.* referenced as giving the computational details of the profile likelihood approach. However, we have since discovered that we can substantially expand the range of applicability of the method with minimal increase in complexity. We begin by motivating a class of likelihood functions.

Suppose that $\boldsymbol{v}$ represents a set of variables containing easily obtainable information which is available for every individual under study. For our development, $\boldsymbol{v}$ must be discrete, whereas all other variables may be either discrete or continuous. Information on a (possibly multivariate) response variable $\boldsymbol{y}$, is available for at least a subset of individuals under study, and more "expensive" set of explanatory variables $\boldsymbol{z}$ is also available for a subset. We may wish to use a subset, $\boldsymbol{v}_1$, of the variables in $\boldsymbol{v}$ as explanatory variables in our model; other variables in $\boldsymbol{v}$ can play the role of informative surrogates for expensive covariates in $\boldsymbol{z}$. The object is to estimate the parameters $\boldsymbol{\theta}$ of $f(\boldsymbol{y} \mid \boldsymbol{z}, \boldsymbol{v}_1; \boldsymbol{\theta})$. Thus we have a parametric regression model for the conditional density of the response given explanatory variables. Data sources for our class of likelihoods can include observations on $\boldsymbol{z}$ and any or all of $(\boldsymbol{y}, \boldsymbol{v})$, $(\boldsymbol{y} \mid \boldsymbol{v})$, $(\boldsymbol{y}, \boldsymbol{z}, \boldsymbol{v})$, $(\boldsymbol{y}, \boldsymbol{z} \mid \boldsymbol{v})$, $(\boldsymbol{y} \mid \boldsymbol{z}, \boldsymbol{v})$, $(\boldsymbol{z} \mid \boldsymbol{y}, \boldsymbol{v})$, $(\boldsymbol{z}, \boldsymbol{v})$ and $(\boldsymbol{z} \mid \boldsymbol{v})$ where "|" refers to information obtained from conditional sampling.

Let $g(\boldsymbol{z}, \boldsymbol{v})$ denote the density of the covariates. With standard prospective sampling and no missing data, the likelihood factorises into a term involving $\boldsymbol{\theta}$ and a term involving $g()$ so that information on the distribution of the explanatary variables is orgthogonal to information on $\boldsymbol{\theta}$. Consequently, we do not need to model the covariate distribution. This is very convenient in practice because we often have many covariates of different types for it to be feasible to model their joint distribution in any realistic fashion. Unfortunately, with response-selective data, or most missing data mechanisms, information on $\boldsymbol{\theta}$ and $g()$ is no longer orthogonal and we are forced to use some sort of joint modelling. However, the practical need for methods which do not require parametric modelling of $g()$ is just as great. Thus, we consider semi-parametric methods in which the marginal distribution of $(\boldsymbol{z}, \boldsymbol{v})$ is left unspecified and estimated nonparametrically.

Recall that $\boldsymbol{v}$ is discrete and denote the distinct values of $\boldsymbol{v}$ that we observe $\widetilde{\boldsymbol{v}}_1, \ldots, \widetilde{\boldsymbol{v}}_S$. The class of likelihoods we consider first are those of the form

$$\prod_{s=1}^{S} \prod_{i: \boldsymbol{v}_i = \widetilde{\boldsymbol{v}}_s} f(\boldsymbol{y}_i \mid \boldsymbol{z}_i, \widetilde{\boldsymbol{v}}_{1s}; \boldsymbol{\theta})^{\Delta_{1i}} g_1(\boldsymbol{z}_i \mid \widetilde{\boldsymbol{v}}_s)^{\Delta_{2i}} \mathrm{pr}(\boldsymbol{y}_i \mid \widetilde{\boldsymbol{v}}_s; \boldsymbol{\theta})^{\Delta_{3i}}. \qquad (3)$$

Here, $\mathrm{pr}(\boldsymbol{y} \mid \boldsymbol{v}; \boldsymbol{\theta}) = \int f(\boldsymbol{y} \mid \boldsymbol{z}, \boldsymbol{v}_1; \boldsymbol{\theta}) dG_1(\boldsymbol{z}|\boldsymbol{v})$, Δ_{1i} and Δ_{2i} are 0/1 indicators, whereas Δ_{3i} can take values 0 and ± 1. Frequently, the i^{th} observation will contribute to all three terms in (3). A single profile likelihood method can cater for all likelihoods in this class. This enables general software to be written whereby a new regression model can be catered for simply by coding a new function to calculate $f(\boldsymbol{y} \mid \boldsymbol{z}, \boldsymbol{v}_1; \boldsymbol{\theta})$ and its derivatives.

3.2 Profile Likelihood

3.2.1 Preliminaries

We wish to maximise a likelihood of the form

$$L[\boldsymbol{\theta}, \{g(\cdot|\boldsymbol{v}_s)\}) = \prod_{s=1}^{S} \prod_{i:\boldsymbol{v}_i=\widetilde{\boldsymbol{v}}_s} f(\boldsymbol{y}_i \mid \boldsymbol{z}_i, \widetilde{\boldsymbol{v}}_{1s}; \boldsymbol{\theta})^{\Delta_{1i}} g(\boldsymbol{z}_i \mid \widetilde{\boldsymbol{v}}_s)^{\Delta_{2i}} \mathrm{pr}(\boldsymbol{y}_i \mid \widetilde{\boldsymbol{v}}_s; \boldsymbol{\theta})^{\Delta_{3i}}, \qquad (4)$$

where $\mathrm{pr}(\boldsymbol{y} \mid \boldsymbol{v}; \boldsymbol{\theta}) = \int f(\boldsymbol{y} \mid \boldsymbol{z}, \boldsymbol{v}_1; \boldsymbol{\theta}) dG(\boldsymbol{z}|\boldsymbol{v})$, as a function of $\boldsymbol{\theta}$ and the S conditional densities $g(\boldsymbol{z} \mid \widetilde{\boldsymbol{v}}_s)$. As is standard in semiparametric maximum likelihood, we treat these densities as discrete with all of the mass being placed at the observed $\boldsymbol{z}$ values.

We now make some helpful notational changes. We write $\boldsymbol{x} = (\boldsymbol{z}, \boldsymbol{v}_1)$ to include all covariates that are to be used in the model. When we consider only the sizes of probability atoms, $\mathrm{pr}(\boldsymbol{x}|\boldsymbol{v}) = \mathrm{pr}(\boldsymbol{z}, \boldsymbol{v}_1|\boldsymbol{v}) = \mathrm{pr}(\boldsymbol{z}|\boldsymbol{v})$. We note that the only remaining role of $\boldsymbol{v}$ in (4) is to divide the data set up into S strata with separate distributions of $\boldsymbol{x}$ to be estimated for each stratum.

Let $\mathcal{S}_s$ be the stratum containing all observations with $\boldsymbol{v}_i = \widetilde{\boldsymbol{v}}_s$. Our problem is now to maximise

$$\ell(\boldsymbol{\theta}, g_1, \ldots, g_S) = \prod_{s=1}^{S} \prod_{i \in \mathcal{S}_s} f(\boldsymbol{y}_i \mid \boldsymbol{x}_i; \boldsymbol{\theta})^{\Delta_{1i}} g(\boldsymbol{x}_i \mid \mathcal{S}_s)^{\Delta_{2i}} \mathrm{pr}(\boldsymbol{y}_i \mid \mathcal{S}_s; \boldsymbol{\theta})^{\Delta_{3i}}.$$

The log-likelihood is of the form $\ell(\boldsymbol{\theta}, g_1, \ldots, g_S) = \sum_s \ell_s(\boldsymbol{\theta}, g_s)$. Thus, the profile likelihood, in which we maximise out the g_s's for fixed $\boldsymbol{\theta}$, is of the form

$$\ell_P(\boldsymbol{\theta}) = \ell\{\boldsymbol{\theta}, \widehat{g}_1(\cdot; \boldsymbol{\theta}), \ldots, \widehat{g}_S(\cdot; \boldsymbol{\theta})\} = \sum_s \ell_s\{\boldsymbol{\theta}, \widehat{g}_s(\cdot; \boldsymbol{\theta})\} = \sum_s \ell_{P,s}(\boldsymbol{\theta}).$$

Thus, we need only to be able to solve the problem of obtaining the profile for $\boldsymbol{\theta}$ (and its derivatives) for a single stratum, i.e. for the case of

$$L(\boldsymbol{\theta}, g) = \prod f(\boldsymbol{y}_i \mid \boldsymbol{x}_i; \boldsymbol{\theta})^{\Delta_{1i}} g(\boldsymbol{x}_i)^{\Delta_{2i}} \mathrm{pr}(\boldsymbol{y}_i; \boldsymbol{\theta})^{\Delta_{3i}}. \qquad (5)$$

In terms of implementation in software, we need to write a function to find $\ell_P(\boldsymbol{\theta})$ for the case of (5). When we have multiple strata, we can send the data from each stratum in turn to that function and accumulate the results.

3.2.2 The direct approach

In essence, we will let $\delta_i = g(\boldsymbol{x}_i)$ and work with $\ell(\boldsymbol{\theta}, \boldsymbol{\delta})$ with $\boldsymbol{\delta} = (\delta_1, \delta_2, \ldots)$ as if it were an ordinary parametric likelihood. One of the problems with the method below, however, is that it often results in us working with some very large arrays. It is particularly important to keep the dimension of $\boldsymbol{\delta}$ as small as possible. Trapping replicate data points is one way of reducing the size of these arrays. Thus, we will write in terms of replicated data. Whether or not we do this makes no difference at all to the profile that we obtain for $\boldsymbol{\theta}$, but it can substantially reduce storage.

Let $A = \{i : \Delta_{1i} = 1\}$, $B = \{i : \Delta_{2i} = 1\}$, $C = \{i : \Delta_{3i} \neq 0\}$. Let $\widetilde{\boldsymbol{x}}_1, \dots, \widetilde{\boldsymbol{x}}_J$ be the unique values of $\boldsymbol{x}$ in B, m_j be the multiplicity of $\widetilde{\boldsymbol{x}}_j$ in B and $\delta_j = \mathrm{pr}(\widetilde{\boldsymbol{x}}_j)$. Let $\widetilde{\boldsymbol{y}}_1, \dots, \widetilde{\boldsymbol{y}}_K$ be the unique values of $\boldsymbol{y}$ in C (ignoring $\boldsymbol{x}$) and let $r_k = \sum_{i \in C, \boldsymbol{y}_i = \widetilde{\boldsymbol{y}}_k} \Delta_{3i}$. Note that r_k can be positive, negative or zero. On noting that $\mathrm{pr}(y) = \sum f(\boldsymbol{y}|\widetilde{\boldsymbol{x}}_j)\delta_j$, our log likelihood from (5) is

$$\ell(\boldsymbol{\theta}, \boldsymbol{\delta}) = \sum_A n_i \log f(\boldsymbol{y}_i \mid \boldsymbol{x}_i; \boldsymbol{\theta}) + \sum_j m_j \log \delta_j + \sum_k r_k \log \left\{ \sum_j f(\widetilde{\boldsymbol{y}}_k \mid \widetilde{\boldsymbol{x}}_j; \boldsymbol{\theta}) \delta_j \right\}. \quad (6)$$

Since the δ_j parameters have to satisfy the constraints $0 < \delta_j < 1$ and $\sum \delta_j = 1$, we reparameterize in terms of $\rho_j = \log(\delta_j/\delta_J)$ and work with $\ell(\boldsymbol{\theta}, \boldsymbol{\rho})$. With this parametrization $\delta_j = \exp(\rho_j)/\sum \exp(\rho_\ell)$ (with $\rho_J \equiv 0$) and the constraints are satisfied automatically. The profile loglikelihood is $\ell_P(\boldsymbol{\theta}) = \ell(\boldsymbol{\theta}, \widehat{\boldsymbol{\rho}}(\boldsymbol{\theta}))$, where $\widehat{\boldsymbol{\rho}}(\boldsymbol{\theta})$ solves $\partial\ell(\boldsymbol{\theta}, \boldsymbol{\rho})/\partial\boldsymbol{\rho} = \mathbf{0}$, and ℓ_P has score vector

$$\boldsymbol{U}_P(\boldsymbol{\theta}) = \left.\frac{\partial\ell(\boldsymbol{\theta}, \boldsymbol{\rho})}{\partial\boldsymbol{\theta}}\right|_{\boldsymbol{\rho} = \widehat{\boldsymbol{\rho}}(\boldsymbol{\theta})}$$

(see Seber and Wild, 1989, equation (2.69)) and information matrix

$$\boldsymbol{\mathcal{I}}_P = \left.\left\{ \boldsymbol{\mathcal{J}}_{\theta\theta} \ - \ \boldsymbol{\mathcal{J}}_{\theta\rho} \boldsymbol{\mathcal{J}}_{\rho\rho}^{-1} \boldsymbol{\mathcal{J}}_{\theta\rho}^T \right\}\right|_{\boldsymbol{\rho} = \widehat{\boldsymbol{\rho}}(\boldsymbol{\theta})}$$

written in terms of the blocks of the information matrix from $\ell(\boldsymbol{\theta}, \boldsymbol{\rho})$ (see Seber and Wild, 1989, just prior to equation (2.72)).

We have been applying a Newton-Raphson based algorithm to maximize the profile loglikelihood $\ell_P(\boldsymbol{\theta})$, i.e., $\boldsymbol{\theta}^{(a+1)} = \boldsymbol{\theta}^{(a)} + \boldsymbol{\mathcal{I}}_P^{-1} \boldsymbol{U}_P|_{\boldsymbol{\theta} = \boldsymbol{\theta}^{(a)}}$ for $a = 1, 2, \dots$. At each iteration, when $\boldsymbol{\theta}$ is updated to $\boldsymbol{\theta}^{(a+1)}$ we solve for the accompanying $\widehat{\boldsymbol{\rho}}(\boldsymbol{\theta}^{(a+1)})$ also by using Newton Raphson to maximise $\ell(\boldsymbol{\theta}^{(a+1)}, \boldsymbol{\rho})$ with respect to $\boldsymbol{\rho}$. Our "Newton-Raphson based algorithm" employs simple versions of the hill-climbing techniques discussed in Section 13.3.1 of Seber and Wild (1989) to improve robustness.

A clear disadvantage with the particular profile likelihood algorithm discussed in this section is that the parameter vectors can be very large indeed requiring substantial storage. We will find in the next section that, when $\boldsymbol{y}$ is discrete we can obtain a substantial reduction in dimensionality, and consequent increase in speed. Whereas $\boldsymbol{\rho}$ has dimension

$J-1$ where J is the number of distinct values observed for $\boldsymbol{x}$, in the case of binary regression models we can work with a nuisance parameter of dimension 1. The same sort of reduction can be made for continuous $\boldsymbol{y}$ where only class interval information on $\boldsymbol{y}$ is available for those data points for which $\boldsymbol{x}$ is not fully observed.

Before going on, however, we do note that when we have several strata, the method discussed in the current subsection never needs to store a $\boldsymbol{\mathcal{J}}_{\rho\rho}$ matrix for more than one stratum. These matrices are used to find the contribution of the current stratum to the overall information matrix $\boldsymbol{\mathcal{I}}_P$ and then discarded when we move on to process the data from the next stratum; see the discussion surrounding (5).

3.2.3 An extension to the class of likelihoods

There are important extensions of the likelihood class (3) which do not affect the essential nature of the maximization problem. Example 3 of Lawless *et al.* (1999) concerns failure time data in which whether of not a data point was fully observed depended on membership of strata defined in terms of both $\boldsymbol{y}$ and $\boldsymbol{x}$. They also used strata involving both $\boldsymbol{y}$ and $\boldsymbol{x}$ to avoid the problem of empty or near-empty strata. Neuhaus *et al.* (2002) concerns retrospectively sampled family data. Here, $\boldsymbol{y}$ records the set of binary responses for each member of a cluster (or family) and sampling is conditional upon observation of some specified pattern in the responses from a cluster. This data can also be supplemented in various ways, for example by knowledge of stratum sizes in the finite population from which the individuals were sampled.

We can expand (3) to cater for such examples as follows. Let $\boldsymbol{h}_{\boldsymbol{v}}(\boldsymbol{y},\boldsymbol{x})$ be a known function of the data, where we may use different functions for different $\boldsymbol{v}$. Imagine that observation depends upon the value of $\boldsymbol{h}$. We may, for example, sample conditionally upon $\boldsymbol{h}$ values and then observe $(\boldsymbol{y},\boldsymbol{x})$. We may also use a random mechanism by which $\boldsymbol{h}$ values are obtained according to some probability $\pi_4(\boldsymbol{h})$ and then $(\boldsymbol{y},\boldsymbol{x})$ are sampled obtained from the conditional distribution of $(\boldsymbol{y},\boldsymbol{x})$ given $\boldsymbol{h}$. We may supplement such data with finite population data, or data from a random sample of $\boldsymbol{h}$ values. Or we may may have data produced by a random process, observe the $\boldsymbol{h}$ values as they arise and then further observe $(\boldsymbol{y},\boldsymbol{x})$ with probability $\pi_5(\boldsymbol{h})$. In all of these situations, the likelihood is of the following form, generalised from (3), for suitably chosen Δ_{ji}s.

$$\prod_{s=1}^{S}\prod_{i:\boldsymbol{v}_i=\boldsymbol{v}_s} f(\boldsymbol{y}_i \mid \boldsymbol{z}_i,\boldsymbol{v}_{1s};\boldsymbol{\theta})^{\Delta_{1i}} g_1(\boldsymbol{z}_i \mid \boldsymbol{v}_s)^{\Delta_{2i}} \mathrm{pr}\{\boldsymbol{h}_{\boldsymbol{v}_s}(\boldsymbol{y},\boldsymbol{z}) = \boldsymbol{h}^{[i]} \mid \boldsymbol{v}_s;\boldsymbol{\theta}\}^{\Delta_{3i}},$$

where $\boldsymbol{h}^{[i]} = \boldsymbol{h}_{\boldsymbol{v}_s}(\boldsymbol{y}_i, \boldsymbol{x}_i)$. Considering the profiling problem for a single stratum, (5) becomes

$$L(\boldsymbol{\theta}, g) = \prod f(\boldsymbol{y}_i \mid \boldsymbol{x}_i; \boldsymbol{\theta})^{\Delta_{1i}} g(\boldsymbol{x}_i)^{\Delta_{2i}} \mathrm{pr}\{\boldsymbol{h}(\boldsymbol{y}, \boldsymbol{z}) = \boldsymbol{h}^{[i]} \mid \boldsymbol{v}_s; \boldsymbol{\theta}\}^{\Delta_{3i}},$$

where $\mathrm{pr}\{\boldsymbol{h}(\boldsymbol{y}, \boldsymbol{z}) = \widetilde{\boldsymbol{h}}; \boldsymbol{\theta}\} = \displaystyle\int_{\boldsymbol{h}(\boldsymbol{y},\boldsymbol{x})=\widetilde{\boldsymbol{h}}} dF(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\theta})dG(\boldsymbol{x})$, and (6), the formulation we use computationally, becomes

$$\ell(\boldsymbol{\theta}, \boldsymbol{\delta}) = \sum_A n_i \log f(\boldsymbol{y}_i \mid \boldsymbol{x}_i; \boldsymbol{\theta}) + \sum_j m_j \log \delta_j + \sum_k r_k \log \left\{ \sum_j \mathrm{pr}(\widetilde{\boldsymbol{h}}_k | \widetilde{\boldsymbol{x}}_j; \boldsymbol{\theta}) \delta_j \right\}, \quad (7)$$

where $\{\widetilde{\boldsymbol{h}}_k\}$ are the distinct values of $\boldsymbol{h}$ observed, that $\widetilde{\boldsymbol{h}}_k$ occurs with multiplicity r_k, and $\mathrm{pr}\{\widetilde{\boldsymbol{h}}_k \mid \boldsymbol{x}; \boldsymbol{\theta}\} = \displaystyle\int_{\boldsymbol{y}:\boldsymbol{h}(\boldsymbol{y},\boldsymbol{x})=\widetilde{\boldsymbol{h}}_k} dF(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\theta})$. The profile likelihood algorithm for the larger class of likelihoods in this subsection differs only from that in Section 3.2 only in that $\mathrm{pr}\{\widetilde{\boldsymbol{h}}_k \mid \boldsymbol{x}; \boldsymbol{\theta}\}$ replaces $f(\widetilde{\boldsymbol{y}}_k|\boldsymbol{x};\boldsymbol{\theta})$ in the last term of the loglikelihood. In computational terms, this means that to accomodate a new model $f(\boldsymbol{y} \mid \boldsymbol{x}; \boldsymbol{\theta})$ when $\boldsymbol{h}(\boldsymbol{y}, \boldsymbol{x})$ is more complicated than simply $\boldsymbol{h}(\boldsymbol{y}, \boldsymbol{x}) = \boldsymbol{y}$, an additional function must be written to evaluate $\mathrm{pr}\{\boldsymbol{h} \mid \boldsymbol{x}; \boldsymbol{\theta}\}$ and its derivatives with respect to $\boldsymbol{\theta}$.

3.2.4 The discrete partition version

When the $\boldsymbol{h}(\boldsymbol{y}, \boldsymbol{x})$ defines a finite partition of range of $(\boldsymbol{y}, \boldsymbol{x})$, a substantial reduction of the dimension of the maximisation problem can be achieved. If we maximise (7) with respect to $\boldsymbol{\delta}$ for fixed $\boldsymbol{\theta}$, using a Lagrange multiplier to cater for the constraint $\sum_j \delta_j = 1$, we find that $\widehat{\boldsymbol{\delta}}(\boldsymbol{\theta})$ satisfies the set of equations

$$\delta_j \;=\; m_j \Big/ \left\{ (m_+ + r_+) - \sum_k \frac{r_k \mathrm{pr}(\widetilde{\boldsymbol{h}}_k \mid \widetilde{\boldsymbol{x}}_j; \boldsymbol{\theta})}{\sum_{j'} \mathrm{pr}(\widetilde{\boldsymbol{h}}_k \mid \widetilde{\boldsymbol{x}}_{j'}; \boldsymbol{\theta}) \delta_{j'}} \right\}, \quad j = 1, \ldots, J, \quad (8)$$

where $m_+ = \sum_j m_j$ is the total number of observations in B and $R_+ = \sum r_k$. Suppose that the range of $\boldsymbol{h}(\boldsymbol{y}, \boldsymbol{x})$ is a finite set and that all possible values have been observed at least once. Thus $\sum_k \mathrm{pr}(\widetilde{\boldsymbol{h}}_k \mid \boldsymbol{x}; \boldsymbol{\theta}) = 1$ and it is straightforward to verify that the system (8) is satisfied when we write

$$\tilde{p}_k = (m_+ + r_+) - \frac{r_k}{Q_k}$$

and set

$$\delta_j = \frac{m_j}{(m_+ + r_+) - \sum_k r_k \frac{\mathrm{pr}(\widetilde{\boldsymbol{h}}_k | \widetilde{\boldsymbol{x}}_j; \boldsymbol{\theta})}{Q_k}} = \frac{m_j}{\sum_k \tilde{p}_k \mathrm{pr}(\widetilde{\boldsymbol{h}}_k \mid \widetilde{\boldsymbol{x}}_j; \boldsymbol{\theta})}, \quad (9)$$

where the set of Q_k's corresponding to nonzero r_k solve the system

$$Q_k = \sum_j \frac{m_j \mathrm{pr}(\widetilde{\boldsymbol{h}}_k \mid \widetilde{\boldsymbol{x}}_j; \boldsymbol{\theta})}{(m_+ + r_+) - \sum_k \frac{m_k}{Q_k} \mathrm{pr}(\widetilde{\boldsymbol{h}}_k \mid \widetilde{\boldsymbol{x}}_j; \boldsymbol{\theta})} = \sum_j \frac{m_j \mathrm{pr}(\widetilde{\boldsymbol{h}}_k \mid \widetilde{\boldsymbol{x}}_j; \boldsymbol{\theta})}{\sum_k \tilde{p}_k \mathrm{pr}(\widetilde{\boldsymbol{h}}_k \mid \widetilde{\boldsymbol{x}}_j; \boldsymbol{\theta})}. \tag{10}$$

We note that definition (9) together with (10) leads to $\sum_j \mathrm{pr}(\widetilde{\boldsymbol{h}}_k \mid \widetilde{\boldsymbol{x}}_j; \boldsymbol{\theta})\delta_j = Q_k$, and also that $\tilde{p}_k$ is a function of Q_k. The system of equations (10) is equivalent to the set of score equations in $\boldsymbol{Q}$ for fixed $\boldsymbol{\theta}$ from,

$$\begin{aligned} \ell^*(\boldsymbol{\theta}, \boldsymbol{Q}) &= \sum_A n_i \log f(\boldsymbol{y}_i \mid \boldsymbol{x}_i \; ; \; \boldsymbol{\theta}) - \sum_j m_j \log \left\{ \sum_k \tilde{p}_k(Q_k) \mathrm{pr}(\widetilde{\boldsymbol{h}}_k \mid \widetilde{\boldsymbol{x}}_j; \boldsymbol{\theta}) \right\} \\ &\quad + \sum r_k \log Q_k. \end{aligned} \tag{11}$$

The following relationships are important.

1. $\ell(\boldsymbol{\theta}, \widehat{\boldsymbol{\delta}}(\boldsymbol{\theta})) = \ell^*(\boldsymbol{\theta}, \widehat{\boldsymbol{Q}}(\boldsymbol{\theta})) = \ell_P(\boldsymbol{\theta})$.

2. $\ell^*(\boldsymbol{\theta}, \boldsymbol{Q})$ can be obtained from $\ell(\boldsymbol{\theta}, \boldsymbol{\delta})$ by substituting for the δ_js using (9) and ignoring contants. However, ℓ^* itself is not a loglikelihood. We elaborate later.

3. Both $\ell^*(\boldsymbol{\theta}, \boldsymbol{Q})$ and the system (10) depend upon only those Q_k for which $r_k \neq 0$. [Values of $r_k = 0$ arise for $\widetilde{\boldsymbol{h}}_k$ values for which all $(\boldsymbol{y}, \boldsymbol{x})$ are completely observed.]

4. If all possible values of $\boldsymbol{x}$ have been observed $Q_k = \sum_j \mathrm{pr}(\widetilde{\boldsymbol{h}}_k \mid \widetilde{\boldsymbol{x}}_j; \boldsymbol{\theta})\delta_j = \mathrm{pr}(\widetilde{\boldsymbol{h}}_k)$. Otherwise Q_k can be interpreted as estimating $\mathrm{pr}(\widetilde{\boldsymbol{h}}_k)$.

5. $\sum_k Q_k = 1$ since $\sum_k \mathrm{pr}(\widetilde{\boldsymbol{h}}_k \mid \boldsymbol{x}; \boldsymbol{\theta}) = 1$.

6. $\mathrm{Dim}(\boldsymbol{\delta}) = J$, the number of distinct observed $\boldsymbol{x}$ values, whereas $\dim(\boldsymbol{Q}) = K$, the number of distinct values of $\boldsymbol{h}$. For binary regression models, $K = 2$.

We stated that ℓ^* is not a log-likelihood. To illustrate, ℓ^* typically has a minimum rather than a maximum in $\boldsymbol{Q}$ when the nonzero r_ks are all positive as is the case with missing data problems, whereas when the nonzero r_Ks are negative as with unsupplemented retrospective sampling, ℓ^* typically has a maximum in $\boldsymbol{Q}$.

We have experimented with several reparameterizations of $\boldsymbol{Q}$ to take care of positivity and summation constraints including $Q_k = \exp(\rho_k)/\{1 + \sum \exp(\rho_\ell)\}$ and

$$Q_k = \exp(\xi_k)/\{1 + \exp(\xi_k)\} \quad \text{or} \quad \xi_k = \mathrm{logit}(Q_k).$$

The former leads to singular information matrices when two or more $r_k = 0$, which is not surprising given Note 2 above, whereas both parameterizations lead to the identical sets of derivatives when at most one $r_k = 0$. Thus, we routinely use $\ell^*(\boldsymbol{\theta}, \boldsymbol{\xi})$, where $\boldsymbol{\xi}$ contains only those ξ_k for which $r_k \neq 0$. Now,

$$\ell_P(\boldsymbol{\theta}) = \ell^*\{\boldsymbol{\theta}, \widehat{\boldsymbol{\xi}}(\boldsymbol{\theta})\},$$

where $\widehat{\boldsymbol{\xi}}(\boldsymbol{\theta})$ solves $\partial\ell(\boldsymbol{\theta}, \boldsymbol{\xi})/\partial\boldsymbol{\xi} = \mathbf{0}$. Despite ℓ^* not being a likelihood, we can still calculate the score vector and information matrix of $\ell_P(\boldsymbol{\theta})$ using

$$\boldsymbol{U}_P(\boldsymbol{\theta}) = \frac{\partial\ell(\boldsymbol{\theta}, \boldsymbol{\xi})}{\partial\boldsymbol{\theta}}\bigg|_{\boldsymbol{\xi}=\widehat{\boldsymbol{\xi}}(\boldsymbol{\theta})} \quad \text{and} \quad \boldsymbol{\mathcal{I}}_P = \left\{\boldsymbol{\mathcal{J}}_{\theta\theta} \;-\; \boldsymbol{\mathcal{J}}_{\theta\xi}\boldsymbol{\mathcal{J}}_{\xi\xi}^{-1}\boldsymbol{\mathcal{J}}_{\theta\xi}^{T}\right\}\bigg|_{\boldsymbol{\xi}=\widehat{\boldsymbol{\xi}}(\boldsymbol{\theta})}.$$

4 How the Program Engine Works

Recall that we have taken a two tiered approach in generating the `missreg` library. As a first tier, we have written specific functions to perform a limited number of specific types of regression with some specific missing data structures. All of these functions call a single general program engine. As a second tier, we have also documented key parts of the wider system to enable a more sophisticated user to implement new models and/or new missingness structures. This latter use requires an understanding of the class of likelihood functions and missing data structures catered for and also of the program structure.

In this section, we will discuss the program structure of those functions that are frequently used in the library and designed to be generalised easily. Descriptions are provided to enable people to develop functions to cater for models or sampling schemes in addition to those we have implemented so far.

4.1 Illustration using Ordinary Prospective Maximum Likelihood

To facilitate generality, the program consists of functions that perform general tasks that need to know nothing about how subtasks are performed. To enable subtasks to be performed in different ways, a general function's call sequence includes one or more arguments that are "user"-supplied functions will perform the subtasks. Since a parent function has no knowledge of what information a user-supplied function might require, we make heavy use of the "..." facility of Splus/R which enables a call to a high level function to pass information on down to any other functions that a parent function might call and on in turn to any functions that they might call.

The top of this tree is a general purpose Newton-method maximiser called `mlefn`. This function will maximise (or minimise or solve the score equations) of any function passed to it through the call argument `loglkfn`. The function being optimised is thought of as

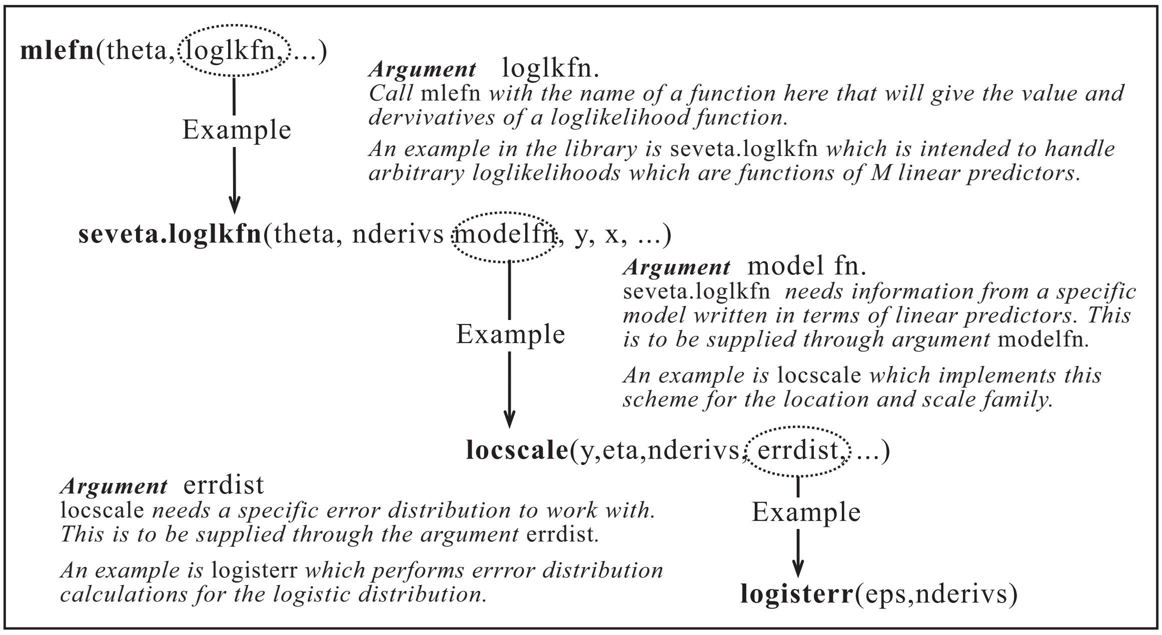

mlefn(theta, loglkfn, ...)
Example
Argument loglkfn.
Call mlefn with the name of a function here that will give the value and dervivatives of a loglikelihood function.
An example in the library is seveta.loglkfn which is intended to handle arbitrary loglikelihoods which are functions of M linear predictors.
seveta.loglkfn(theta, nderivs modelfn, y, x, ...)
Argument model fn.
seveta.loglkfn needs information from a specific model written in terms of linear predictors. This is to be supplied through argument modelfn.
An example is locscale which implements this scheme for the location and scale family.
Example
locscale(y,eta,nderivs, errdist, ...)
Argument errdist
locscale needs a specific error distribution to work with. This is to be supplied through the argument errdist.
An example is logisterr which performs errror distribution calculations for the logistic distribution.
Example
logisterr(eps,nderivs)

Figure 10: Maximum likelihood for models with M linear predictors

a loglikelihood function, though it need not be, but we use the "loglikelihood" language in our descriptions. The only other required argument of `mlefn` is `theta`, a vector of starting values for the regression parameters. Any function called as a specific `loglkfn` must be capable of taking a value of `theta` and supplying the loglikelihood value, the score vector (vector of first derivatives) and the information matrix (negative matrix of second derivatives).

The program structure is illustrated in Figure 10 in a simple prospective setting without the complication of missing values. This example also introduces the approach implemented in the library for models which involve several (1 or more) linear predictors.

Consider a standard independence-case loglikelihood of the form

$$\ell(\theta) = \sum_i \log f(\boldsymbol{y}_i \mid \boldsymbol{x}_i;\ \boldsymbol{\theta}). \tag{12}$$

We wish to implement maximum likelihood for the class of models for which $f(\boldsymbol{y}_i \mid \boldsymbol{x}_i; \boldsymbol{\theta})$ is of the form

$$f(\boldsymbol{y}_i \mid \boldsymbol{x}_i;\ \boldsymbol{\theta}) = f(\boldsymbol{y}_i \mid \eta_{i(1)},\ \ldots,\ \eta_{i(M)}), \tag{13}$$

with $\eta_{i(j)} = \boldsymbol{x}_{i(j)}^T\boldsymbol{\theta}$ $(j = 1, \ldots, M)$. Thus a likelihood in this class is a function of M linear predictors. Generalised linear models fall into this class with $M = 1$, bivariate binary regression models such as the Palmgren and Bahadur fall into this class with $M = 3$. Other examples are given in Yee and Wild (1996). A simple subclass, which we will use for illustrative purposes is the class of location and scale regression models ($M = 2$) written as

$$y_i = \eta_{i(1)} + e^{\eta_{i(2)}}\varepsilon_i$$

where the ε_i's are random errors from some distribution (e.g. normal, logistic, t, etc). Any single parameter can be treated as a linear predictor modelled using an intercept alone (i.e., without inclusion of explanatory variables). Allowing it to be a linear predictor gives additional modelling flexibility. For example, the scale parameter above can either be a single parameter or it can be modelled in terms of covariates.

If the loglikelihood is of the form (13), then for $j = 1, \ldots, M$ and $k = 1, \ldots, M$,

$$\begin{aligned} \frac{\partial \ell}{\partial \theta} &= \sum_i \sum_j \frac{\partial \log f(\boldsymbol{y}_i \mid \eta_{i(1)}, \ldots, \eta_{i(M)})}{\partial \eta_{(j)}} \, \boldsymbol{x}_{i(j)} \\ \frac{\partial^2 \ell}{\partial \theta \partial \theta^T} &= \sum_i \sum_j \sum_k \frac{\partial^2 \log f(\boldsymbol{y}_i \mid \eta_{i(1)}, \ldots, \eta_{i(M)})}{\partial \eta_{(j)} \partial \eta_{(k)}} \, \boldsymbol{x}_{i(j)} \boldsymbol{x}_{i(k)}^T \end{aligned}$$

Such calculations can be performed using vector arithmetic without looping over i. The other dimensions are typically small.

In Figure 10, `mlefn` is called with `loglkfn=seveta.loglkfn` which performs maximum likelihood for any M-linear predictor model. What `seveta.loglkfn` does is to translate `x` and `theta` into a matrix `eta` made up of M columns of values of the linear predictors. It then sends these to an user-specified function called `modelfn` in the call sequence which is being relied upon to supply the $\log f$ values and their derivatives with respect to the $\eta_{(j)}$'s. Last, `seveta.loglkfn` translates these back into derivatives with respect to $\boldsymbol{\theta}$. Thus, although there is a sense in which `seveta.loglkfn` handles all M-linear predictor models, it does so by passing the buck to `modelfn`. All `seveta.loglkfn` really does itself is to handle translation to and from linear predictors.

At the next level down in Figure 10, `seveta.loglkfn` is being called with `modelfn = locscale`. The function `locscale` allows `seveta.loglkfn` to handle the location and scale models described above which is a subclass of M-linear predictor models. Here, `locscale` calculates $\log f$ values and their derivatives with respect to the $\eta_{(j)}$'s for location and scale models specified up to an arbitrary error distribution which must be supplied via the argument `errdist`.

Next, `locscale` is being called with `errdist=logisterr`. This is the bottom level because the model is now completely specified. The complete call illustrated in Figure 10, omitting arguments which are peripheral to the current discussion, is

```
        mlefn(theta,loglkfn=seveta.loglkfn, modelfn=locscale,
                      errdist=logisterr, y=y, x=x)
```

In this call, the data `y` and `x` are available to all functions below `mlefn`. The covariate information `x` is only recognised and used by `seveta.loglkfn`, while the response information `y` is used by both `seveta.loglkfn` and `locscale`.

4.2 Implementation for Retrospective and Missing Data

To understand this material, it is necessary to first read Section 3.

4.2.1 The direct approach

This subsection draws heavily on the ideas in Section 3.2.1 to 3.2.3. We deal with loglikelihood which are a summation over strata of loglikelihoods of the form (7). More specifically, we deal with loglikelihood of the form

$$\begin{aligned}\ell(\boldsymbol{\theta},\boldsymbol{\delta}^{(1)},\ldots,\boldsymbol{\delta}^{(S)}) &= \sum_{s=1}^{S}\ell^{(s)}(\boldsymbol{\theta},\boldsymbol{\delta}^{(s)})\\ &= \sum_{s=1}^{S}\left[\sum_{A^{(s)}} n_i^{(s)}\log f(\boldsymbol{y}_i^{(s)}\mid \boldsymbol{x}_i^{(s)};\boldsymbol{\theta}) + \sum_j m_j^{(s)}\log\delta_j^{(s)}\right.\\ &+ \left.\sum_k r_k^{(s)}\log\left\{\sum_j \mathrm{pr}(\widetilde{\boldsymbol{h}}_k^{(s)}|\widetilde{\boldsymbol{x}}_j^{(s)};\boldsymbol{\theta})\delta_j^{(s)}\right\}\right], \qquad (14)\end{aligned}$$

where S is the total number of V-strata defined for subsampling. For an unstratified design, we have $S=1$. Here, s indexes the set of V-strata, $A^{(s)}$ indexes the set of complete $(\boldsymbol{x},\boldsymbol{y})$-values we have from stratum s, $n_i^{(s)}$ is the multiplicity of $(\boldsymbol{x}_i^{(s)},\boldsymbol{y}_i^{(s)})$. Sums over j range over the values of $\boldsymbol{x}$ observed in stratum s, $m_j^{(s)}$ is the multiplicity of $\boldsymbol{x}_j^{(s)}$, $\delta_j^{(s)}=\mathrm{pr}(\boldsymbol{x}_j^{(s)})$, $\boldsymbol{\delta}^{(s)}=(\delta_1^{(s)},\ldots,\delta_{J(s)}^{(s)})$, $Q_k^{(s)}=\mathrm{pr}(\widetilde{\boldsymbol{h}}(\boldsymbol{y},\boldsymbol{x})=\widetilde{\boldsymbol{h}}_k^{(s)})$ where the $\widetilde{\boldsymbol{h}}_k^{(s)}$'s are the distinct values of $\widetilde{\boldsymbol{h}}(\boldsymbol{y},\boldsymbol{x})$ observed in stratum s. Note that as strata are defined in terms of the $\boldsymbol{x}$-values, different $\widetilde{\boldsymbol{h}}(\boldsymbol{y},\boldsymbol{x})$ formulae can be used in different strata.

In the `missreg` library, the function `MLdirectInf` is specified as a `loglkfn` to return the value, score vector and information matrix at $\boldsymbol{\theta}$ for $\ell_P(\boldsymbol{\theta})=\ell(\boldsymbol{\theta},\widehat{\boldsymbol{\delta}}(\boldsymbol{\theta}))$ when the class of loglikelihoods (14) is considered. Note that this approach requires starting values for $\boldsymbol{\delta}$ as well as starting values for $\boldsymbol{\theta}$. The function `MLdirectInf` is at present set up for models of the form (13) which use $M\geq 1$ linear predictors. As $\ell_P(\boldsymbol{\theta})$ is a sum of individual terms $\ell^{(s)}(\boldsymbol{\theta}, \boldsymbol{\delta}^{(s)})$, we use a subfuntion `ML2directInf` to return the values, score vectors and information matrices at $\boldsymbol{\theta}$ for a $\ell^{(s)}(\boldsymbol{\theta}, \boldsymbol{\delta}^{(s)})$ $(s = 1, ..., S)$ within each of the V-strata. `MLdirectInf` is simply a tool to take input data, split the data by V-strata, send the part of the data set from a particular V-stratum off to `ML2directInf`, and finally accumulate (sum) the values, score vectors and information matrices returned from these strata to provide values for $\ell(\boldsymbol{\theta}, \widehat{\boldsymbol{\delta}}(\boldsymbol{\theta}))$ and its derivatives.

Recall in Section 3.2.2, we actually use the profile loglikelihood $l(\boldsymbol{\theta}, \widehat{\boldsymbol{\rho}}(\boldsymbol{\theta}))$ instead of $l(\boldsymbol{\theta}, \widehat{\boldsymbol{\delta}}(\boldsymbol{\theta}))$ in which $\boldsymbol{\delta}_j^{(s)} = \exp(\boldsymbol{\rho}_j^{(s)})/\sum_l \exp(\boldsymbol{\rho}_l^{(s)})$. With such parametrization, the constaints for $\boldsymbol{\delta}_j^{(s)}$ being a probability are satisfied automatically. The maximization of $\boldsymbol{\rho}$ for fixed $\boldsymbol{\theta}$ at each V-stratum is done within `ML2directInf`. A special subfunction `rhodirectInf` has been written to return the values of $\boldsymbol{\rho}^{(s)}$ and their derivatives at the current estimate of $\boldsymbol{\theta}$. Both `ML2directInf` and its subfunction `rhodirectInf` are automatically called by `MLdirectInf` when this function is used. The user wanting to implement new models does not have to know about any of this, however.

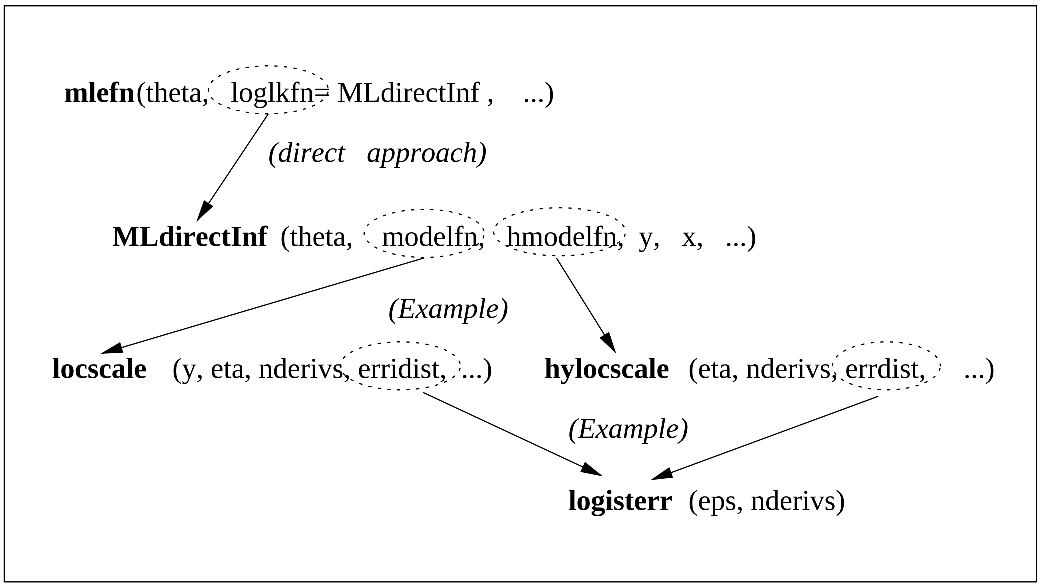
Similar to `seveta.loglkfn` we discussed in Figure 10, what `MLdirectInf` does is to translate $\boldsymbol{x}$ and $\boldsymbol{\theta}$ into a matrix η made up of M columns of values of the linear predictors. It then sends these to user-specified functions in the call sequence to supply the loglikelihood values and their derivatives with respect to the η's. Afterwards, `MLdirectInf` translates these back into derivatives with respect to $\boldsymbol{\theta}$. The difference now is that `MLdirectInf` needs two user-supplied functions which will provide values and their derivatives with respect to the η's for not only $f(\boldsymbol{y}_i^{(s)} \mid \boldsymbol{x}_i^{(s)}; \boldsymbol{\theta})$ but also $\mathrm{pr}(\widetilde{\boldsymbol{h}}_k^{(s)} | \widetilde{\boldsymbol{x}}_j^{(s)}; \boldsymbol{\theta})$ within each V-stratum. These are:

- `modelfn` which allows the `loglkfn` to handle a particular model in this class. It calculates the values and their derivatives with respect to the η's for both $f(\boldsymbol{y}_i^{(s)} \mid \boldsymbol{x}_i^{(s)}; \boldsymbol{\theta})$ and its logarithm.

- `hmodelfn` which calculates the values and their derivatives with respect to the η's for $\mathrm{pr}(\widetilde{\boldsymbol{h}}_k^{(s)} | \widetilde{\boldsymbol{x}}_j^{(s)}; \boldsymbol{\theta})$ under the same model as that specified in `modelfn`.

In principle, $\mathrm{pr}(\widetilde{\boldsymbol{h}}_k | \widetilde{\boldsymbol{x}}_j; \boldsymbol{\theta})$ and its derivatives can be obtained from the model and the missing data structure. In practice, however, this is too hard and we leave it to the user to program for her or his own special case.

The program structure is illustrated in Figure 11 using the location and scale class of models as an example. When the direct approach is considered, at the first step, the function `mlefn` is called with `loglkfn = MLdirectInf`. Recall that `MLdirectInf` handles all M-linear predictor models by passing the buck to two user-supplied subfunctions,



mlefn(theta, loglkfn= MLdirectInf , ...)
(direct approach)
MLdirectInf (theta, modelfn, hmodelfn, y, x, ...)
(Example)
locscale (y, eta, nderivs, erridist, ...)
hylocscale (eta, nderivs, errdist, ...)
(Example)
logisterr (eps, nderivs)

Figure 11: Direct method for models with M linear predictors

named `modelfn` and `hmodelfn`. Therefore, at the second step, we choose (for this example) the location and scale regression model using `modelfn = locscale` and `hmodelfn = hylocscale` in the call sequence. Next, both `locscale` and `hylocscale` are called with `errdist` which specifies an arbitrary error distribution. In this example, the function calls use `errdist = logisterr` which caters for the logistic error distribution[7]. The model is now completely specified at the bottom level.

The function for the location and scale class of models provided in the library for users is called `locsc2stg`. The main work of this function is to not only implement calls described above, but also deal with input data and model description, obtain starting values for $\boldsymbol{\theta}$, and organise the output. Such a function enables us to input data in a required format, pre-check the data for "random" missing values, report strata counts, output results in well-presented tables, etc.

Recall in Example 5 when the `"direct"` method was considered, the following commands were used to call the function `locsc2stg`.

```
> data(lowbirth.ls)
> z8 <- locsc2stg(birthwt~gest+mumht+bmi+ethnicdb+hyper+smoke,~ 1,
        xstrata=c("sex.age"), data=lowbirth.ls, obstype.name=c("instudy"),
        xs.includes=FALSE, method="direct", ... ...)
```

Inside `locsc2stg`, the function `mlefn` was invoked using `loglk = MLdirectInf` with both `modelfn` and `hmodelfn` specified simultaneously under this model. The command lines

[7]Other error distributions catered for include the standard Normal and Student-t distributions.

were as follows:

```
mlefn(theta, loglkfn=MLdirectInf, modelfn=locscale, hmodelfn=hylocscale,
      errdist=errdist, errmodpars=errmodpars, x=xarray, y=as.matrix(y),
      xStrat=as.numeric(xStrat), ... ...)
```

Note that the error distribution is specified using the argument `errdist` along with `errmodpars` which provides required parameter values for some of the error distributions, e.g. the degrees of freedom for a Students' t-distribution.

4.2.2 The discrete partition version

This subsection draws heavily on the ideas in Section 3.2.4. Here, we deal with finding stationary values for "loglikelihoods" which are sums over strata of terms of the form (11). More specifically, we work with $\ell^*(\boldsymbol{\theta}, \boldsymbol{Q}) = \sum_{s=1}^{S} \ell^{*(s)}(\boldsymbol{\theta}, \boldsymbol{Q}^{(s)})$ where

$$\begin{aligned} \ell^{*(s)}(\boldsymbol{\theta}, \boldsymbol{Q}^{(s)}) &= \sum_{A^{(s)}} n_i^{(s)} \log f(\boldsymbol{y}_i^{(s)} \mid \boldsymbol{x}_i^{(s)};\ \boldsymbol{\theta}) + \sum_k r_k^{(s)} \log Q_k^{(s)} \\ &- \sum_j m_j^{(s)} \log \left\{ \sum_k \tilde{p}_k^{(s)} \mathrm{pr}(\widetilde{\boldsymbol{h}}_k^{(s)} \mid \widetilde{\boldsymbol{x}}_j^{(s)};\ \boldsymbol{\theta}) \right\}. \end{aligned} \tag{15}$$

Here, $\tilde{p}_k^{(s)} = (m_+^{(s)} + r_+^{(s)}) - \frac{r_k^{(s)}}{Q_k^{(s)}}$ where $m_+^{(s)} = \sum_j m_j^{(s)}$ and $r_+^{(s)} = \sum_k r_k^{(s)}$. Note that in practice we do not require the $\boldsymbol{x}_j^{(s)}$-values supplied to (15) to be distinct. This approach is normally used when there are considerably fewer $\widetilde{\boldsymbol{h}}_k^{(s)}$ values than $\boldsymbol{x}_j^{(s)}$ values, and requires starting values for $\boldsymbol{Q}$ as well as starting values for $\boldsymbol{\theta}$.

In the `missreg` library, this apporach is applied using `loglkfn = MLInf` in the `mlefn` function call. The function `MLInf` imposes even fewer restrictions on the models fitted than does the function `MLdirectInf`, and actually does very little. The subfunction `ML2Inf` (which does the real work) has been written to return the values, score vectors and information matrices at $\boldsymbol{\theta}$ for (15) within each V-stratum ($s = 1, ..., S$). `MLInf` just knows enough about the input data structure to split up the input data by stratum, send the data set for each stratum off to `ML2Inf`, and finally sum the values, score vectors and information matrices returned by `ML2Inf` across strata.

We can still use transformation $\boldsymbol{Q}_k^{(s)} = \exp(\boldsymbol{\rho}_k^{(s)}) / \sum_l \exp(\boldsymbol{\rho}_k^{(s)})$ as for $\boldsymbol{\delta}_j$'s to look after the constraints posed on $\boldsymbol{Q}_k^{(s)}$. Alternatively, we may use a logit transformation $\boldsymbol{Q}_k^{(s)} = \exp(\boldsymbol{\xi}_k^{(s)})/\{1 + \exp(\boldsymbol{\xi}_k^{(s)})\}$ which corresponds to $\boldsymbol{\xi}_k^{(s)} = \mathtt{logit}(\boldsymbol{Q}_k^{(s)})$ (see Section 3.2.4 for detail). Inside `ML2Inf`, two subfunctions `rhoInf` and `xiInf` have been written to return the values, score vectors and information matrices according to $\boldsymbol{\rho}$ and $\boldsymbol{\xi}$ at the current estimate of $\boldsymbol{\theta}$ respectively within each V-stratum. Either one can be chosen in the program. As in previous subsection, both `ML2Inf` and its subfunctions `rhoInf` and `xiInf` are automatically called by `MLInf` when this function is used. Again, those users who simply want to implement new models do not have to know about any of this.

In fact, `ML2Inf` is not restricted to models with M linear predictors which is the case for `ML2directInf`. It is for general models but the nature of the model is unspecified.

ML2Inf

In its default mode, `ML2Inf` returns the value, score vector and information matrix at $\boldsymbol{\theta}$ for $\ell^{*(s)}(\boldsymbol{\theta}, \widehat{\boldsymbol{Q}}^{(s)}(\boldsymbol{\theta}))$ which is of the form (15).

As we have mentioned, `ML2Inf` does not impose any restrictions on the models fitted either. Everything `Ml2Inf` knows about the model being fitted, it gets from the following two user-supplied functions in the call,

- `ProspModInf`, a function to be supplied by the user to supply values of

$$\sum_{A^{(s)}} n_i^{(s)} \log f(\boldsymbol{y}_i^{(s)} \mid \boldsymbol{x}_i^{(s)};\ \boldsymbol{\theta})$$

  and its derivatives with respect to $\boldsymbol{\theta}$.;

- `StratModInf`, a function to be supplied by the user to supply values of

$$\sum_j m_j^{(s)} \log \left\{ \sum_k \tilde{p}_k^{(s)} \mathrm{pr}(\widetilde{\boldsymbol{h}}_k^{(s)} \mid \widetilde{\boldsymbol{x}}_j^{(s)};\ \boldsymbol{\theta}) \right\}$$

  and its derivatives with respect to $\boldsymbol{\theta}$.

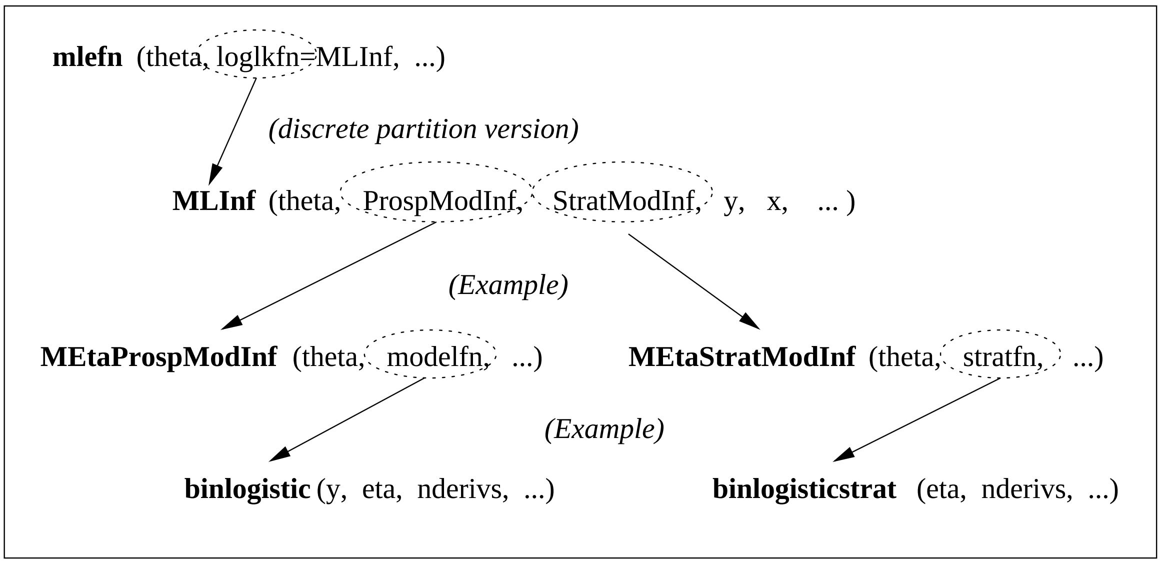
The functions we have written to be `ProspModInf` or `StratModInf` functions again handle classes of models rather than specific models. Only two examples of these functions are currrently available in the library. One is to implement models that are functions of M linear predictors. Another handles binary random effects models. Here, we will only discuss the use of the former.

In the `missreg` library, the functions `MEtaProspModInf` and `MEtaStratModInf` are specified as `ProspModInf` and `StratModInf` functions respectively when M-linear predictor models of the form (13) is considered. As they only handle this class of models, any specific model in this class is next specified using the following two subfunctions:

- `modelfn` which allows `ProspModInf` to handle a particular model in its class. It calculates the values and their derivatives with respect to the η's for both $f(\boldsymbol{y}_i^{(s)} \mid \boldsymbol{x}_i^{(s)}; \boldsymbol{\theta})$ and its logarithm.

- `stratfn` which allows `StratModInf` to handle the same model as that specified in `modelfn`. It calculates the values of $\text{pr}(\widetilde{\boldsymbol{h}}_k^{(s)}|\widetilde{\boldsymbol{x}}_j^{(s)}; \boldsymbol{\theta})$, its first derivative with respect to the η's, and the second derivative of $\sum_k \tilde{p}_k^{(s)} \text{pr}(\widetilde{\boldsymbol{h}}_k^{(s)} \mid \widetilde{\boldsymbol{x}}_j^{(s)};\ \boldsymbol{\theta})$ with respect to the η's.

Note that the differences between the function `stratfn` and the function `hmodelfn`, which is used in the direct approach, are not only their output but also the way the values of $\text{pr}(\widetilde{\boldsymbol{h}}_k^{(s)}|\widetilde{\boldsymbol{x}}_j^{(s)}; \boldsymbol{\theta})$ and their derivatives are calculated. Recall that $\widetilde{\boldsymbol{h}}_k^{(s)}$ defines group membership of $\boldsymbol{Y}$ in s^{th} $\boldsymbol{V}$-stratum when the discrete partion version is considered, even though $\boldsymbol{Y}$ is fully observed as a continuous variable.



mlefn (theta, loglkfn=MLInf, ...)
(discrete partition version)
MLInf (theta, ProspModInf, StratModInf, y, x, ...)
(Example)
MEtaProspModInf (theta, modelfn, ...)
MEtaStratModInf (theta, stratfn, ...)
(Example)
binlogistic (y, eta, nderivs, ...)
binlogisticstrat (eta, nderivs, ...)

Figure 12: Discrete partition version for models with M linear predictors

The program structure is illustrated in Figure 12 using binary regression models as a simple example. When the discrete partition version is considered[8], at the first step, the function `mlefn` is called with `loglkfn = MLInf`. Recall that `MLInf` is not restricted to models with M-linear predictors. The nature of the model is unspecified and passed to `MLInf` (and then `ML2Inf`) by two user-supplied functions given in the argument, namely `ProspModInf` and `StratModInf`. For M-linear predictor models of the form (13), for example, we choose the functions `MEtaProspModInf` and `MEtaStratModInf` in the library

[8]In fact, we consider only this approach in binary regression analyses as the dimention of $\boldsymbol{Q}$ in which $Q_k = \sum_j pr(\widetilde{\boldsymbol{h}}_k \mid \widetilde{\boldsymbol{x}}_j;\ \boldsymbol{\theta})$ is simply 2.

respectively.

Next, `MEtaProspModInf` requires a model function to handle a particular model in this class and `MEtaStratModInf` requires a stratum model function that is consistent with that specified in `MEtaProspModInf`. This is realized using two subfunctions named `modelfn` and `stratfn` in their function call. If we choose a binary logistic regression model, for example, we specify `modelfn = binlogistic` in `MEtaProspModInf` function call and `stratfn = binlogisticstrat` in `MEtaStratModInf` function call. The model is then completely specified.

As in previous subsection, we normally use a specially-written function (e.g. `bin2stg`) to carry out a particular analysis. The `mlefn` is then automatically invoked inside such model-specific function. In Section 2.2 when we analysed the Leprosy data, the fitting syntax was as follows:

```
> data(leprosy1)
> leprosy1$age.trans <- 100 * (leprosy1$age + 7.5)^-2
> z1 <- bin2stg(leprosy ~ age.trans + scar, data=leprosy1, weights=counts,
                 xstrata="age",xs.includes=TRUE)
```

Recall that the function `bin2stg` fits binary regression models using several links and the default is a logistic model (i.e. `"logit"`) which has been chosen here. The `mlefn` was then applied inside this function using one linear predictor with both `modelfn` and `stratfn` specified simultaneously under a logistic model. The command lines were as follows:

```
mlefn(theta, loglkfn=MLInf, ProspModInf=MEtaProspModInf,
      StratModInf=MEtaStratModInf, modelfn=binlogistic,
      hmodelfn=binlogisticstrat, x=xarray, y=as.matrix(y),
      xStrat=as.numeric(xStrat), ... ...)
```

We see that the model is complete when both `binlogistic` and `binlogisticstrat` are specified.

4.3 Function Generalisation for Specific Types of Analysis

In this subsection, we will find out in more detail how each type of analysis we have discussed in Section 2 can be generalised. More specifically, we will discuss how and what the user is required to do if he or she wants to implement new models in addition to those that have been implemented in the library.

We have to mention that these functions in the library for specific types of analysis were originally written to facilitate our research and were not specifically designed for easy generalisation. Some generalisation is possible, however, with relatively minor additions or alternations to the code.

4.3.1 `bin2stg`

The function `bin2stg` fits binary regression models using several links. The links are specified using the parameter `linkname` in the function call and the default is `"logit"`, the logistic model. Other implemented models are the probit (`"probit"`) and the complementary log-log (`"cloglog"`). If the user wants to use another link other than these three, he or she will need to write new `modelfn` and `stratfn` functions for each new link and also make a few appropriate modification on the `bin2stg` function itself.

Recall that when we carry out a binary regression analysis, only the discrete partion version is considered. This means that the user must first write two sub-model functions for `modelfn` and `stratfn` with requirement. Currently available functions in the library are as follows:

- **Logistic**: `modelfn = binlogistic`; `stratfn = binlogisticstrat`.
- **Probit**: `modelfn = binprobit`; `stratfn = binprobitstrat`.
- **Cloglog**: `modelfn = bincloglog`; `stratfn = bincloglogstrat`.

The user can look at how these functions are orginised and write his/her own `modelfn` and `stratfn` in a similar manner.

This is not all the user is required to do, however. Inside `bin2stg`, commands have been written to choose right `modelfn` and `stratfn` functions corresponding to the `linkname` specified in the call. This is done right before the function `mlefn` is invoked. If the users want to carry out analyses using the new link they have written, they have to add a few lines of code to allow for this link name.

Another thing attention should be drawn to is the way that the starting values of the parameters are obtained when `start=NULL` is used. The function `bin2stg` calculates appropriate starting values using `glm` under the links that the `glm` function can cope with. Also, as we normally have a retrospective data set, an offset is required in `glm` to obtain consistent parameter estimates. The user will have to add a line stating an appropriate value for the offset.

4.3.2 `locsc2stg`

The function `locsc2stg` carries out a linear regression anlaysis using the location and scale class of models.The function allows for two types of information on the Y-variable for observations of type `strata`: the actual y-values (i.e. the direct approach), or class-interval information on the y-values (i.e. the discrete partition version).

Different approaches will correspond to different program structures inside `locsc2stg`. When the direct approach is considered, we specify `modelfn = locscale` and `hmodelfn = hylocscale` in the `mlefn` function call inside `locsc2stg`. Otherwise, we specify `modelfn = locscale` and `stratfn = locscstrat1` in the `mlefn` function call for the discrete partition version.

Recall that the location and scale class of models is specified up to an arbitrary error distribution which must be supplied via the argument `errdistrn` in the `locsc2stg` function call. Currently implemented error distributions in the library cater for the Logistic (`"logistic"`, the default), standard Normal (`"normal"`) and Student-t (`"t"`) distributions. The program will then choose appropriate submodel functions for these error distributions with respect to the name specified in `errdistrn`.

When the functions `locscale` and `hylocscale` are used in the direct approach, they work with a specific error distribution which is supplied through the argument `errdist` in their function calls. The functions that have been written in the library to perform error distribution calculations are as follows:

- **Logistic**: `errdist = logisterr`;
- **Normal**: `errdist = normerr`;
- **Student-t**: `errdist = terr` with `errmodpars = 4` (the default *df*).

If the user wants to implement another error distribution, he/she needs to write a new `errdist` function as above and gives a name for that error distribution to be used in `errdistrn`.

When the functions `locscale` and `locscstrat1` are used in the discrete partition version, the error distribution is specified via the argument `errdist` in the `locscale` function call, but via the argument `errdistcdf` in the `locscstrat1` function call. The functions written for `errdistcdf` return the values and their derivatives with respect to the η's for the cumulative distribution function of an error distribution. They are currently implemented as follows:

- **Logistic**: `errdistcdf = logistcdf`;
- **Normal**: `errdistcdf = normcdf`;
- **Student-t**: `errdistcdf = tcdf` with `errmodpars = 4` (the default *df*).

If the user wants to implement another error distribution with the discrete partition version, he/she needs to write not only a new `errdist` function but also a new `errdistcdf` function as above.

Another thing attention should be drawn to is that when the starting values of the parameters are calculated in the program, the scale parameter estimate $\widehat{\sigma}$ (recall that $\sigma = e^{\eta_2}$) can be obtained using the `sigma` output from the summary of `lm` fit. As different error distributions might have different variances, this value should be next scaled to one unit of variance as the standard Normal distribution. Therefore, it is also the user's job to state the standard deviation of the error distribution he/she is interested via the vector `scalef` named inside the `locsc2stg` function. For the Logistic, standard Normal and Student-t distributions, for example, the values of `scalef` are 1.8, 1, and $\sqrt{df/(df-2)}$ respectively.

4.3.3 `bivbin2stg`

The function `bivbin2stg` carries out various semiparametric maximum likelihood analyses with bivariate binary data. One most frequently used approach is to model the joint distribution of Y_1 and Y_2 given X by separately modelling the marginal distributions of $Pr(Y_1|X)$ and $Pr(Y_2|X)$, and the association between Y_1 and Y_2 given X (i.e. specifying 3 models). Currently implemented models for this approach are the Palmgren (`"palmgren"`, the default), Bahadur (`"bahadur"`) and Copula (`"copula"`) models, which must be specified via the argument `method` in the `bivbin2stg` function call.

Similar to binary regression analysis, we only consider the discrete partition version here when both Y_1 and Y_2 are binary variables. This indicates that we first need to specify appropriate `ProspModInf` and `StratModInf` functions, and if necessary, specify `modelfn` and `stratfn` functions within that class of models.

In `bivbin2stg`, we have used `ProspModInf = MEtaProspModInf` and `StratModInf = MEtaProspModInf`. That is, we implement models that are functions of M linear predictors[9]. If the user wants to implement maximum likelihood for other classes of models, he/she has to write his/her own `ProspModInf` and `StratModInf` functions. If the new functions are written at the similar structure as that in `MEtaProspModInf` and `MEtaStratModInf`, the user also needs to write `modelfn` and `stratfn` functions to specify a particular model in this class.

For the Palmgren, Bahadur and Copula models, the following functions have been written

[9]With the Palmgren, Bahadur and Copula models, the number of linear predictors we use is three.

in the library to specify a particular model in the `mlefn` function call inside `bivbin2stg`. Which one should be used depends on not only the model we are interested, but also the sampling scheme:

- **Sampling on Y_1**: `stratfn = biv1pbc.strat`;
  - **Palmgren**: `modelfn = lpalmgrn`.
  - **Bahadur**: `modelfn = lbahad`.
  - **Copula**: `modelfn = lcopula`.
- **Sampling on ($Y_1 \neq 1$, $Y_2 \neq 1$)**: `stratfn = biv00pbc.strat`;
  - **Palmgren**: `modelfn = lpalmgrn`.
  - **Bahadur**: `modelfn = lbahad`.
  - **Copula**: `modelfn = lcopula`.

Again, the users can have a look about these functions and write their own in a similar manner.

Recall that the function `bin2stg` can fit binary regression models using several links. In principle, this should also be available when we carry out bivariate binary regression analyses. That is, we can model the marginal distributions of $Pr(Y_1|X)$ and $Pr(Y_2|X)$, and the association model between Y_1 and Y_2 given X using different links. So far we have only implemented logistic links.

Finally, when the starting values of the parameters are calculated, those used to explain the marginal distributions of $Pr(Y_1|X)$ and $Pr(Y_2|X)$ are obtained uniformly using `glm` with weights regardless of the `method` we choose. The parameters for the association model, however, are obtained in different ways. This is because the Palmgren, Bahadur and Copula models use the same marginal distributions but with different association model[10]. If the user wants to use a new `method` in which the joint distribution of Y_1 and Y_2 given X is modelled in a similar way, he/she is required to write a few more commands inside the function `bivbin2stg` to calculate the starting values of those parameters for the association model. If the new method models the joint distribution in a completely different way, the user needs to reorginize the way that the starting values are calculated.

4.3.4 `bivlocsc2stg`

Recall that this function models the joint distribution of Y_1 and Y_2 as a conditional factorisation of $pr(Y_1|Y_2, X; \theta_1)$ and $pr(Y_2|X; \theta_2)$. A logistic regression model is used for the

[10]See Section 2.4.1 for details.

binary outcome Y_1 and a location-scale model for the continuous outcome Y_2. Again, the discrete partition version is chosen here.

Since the linear location-scale model is used for $pr(Y_2|X)$, same error distributions are considered including `"logistic"`, `"normal"` and `"student-t"`.

There is only one `modelfn` required in `ProspModInf` which is called `lspm2locsc`. The new `StratModInf` function called `MEtaStratModInf.spml2locsc` works on its own and does not require `modelfn` or `stratfn` as input any more.

4.3.5 `rclusbin`

The function `rclusbin` fits random effects models for clustered binary data in which cluster-specific random intercepts are used to account for correlations within clusters. The same set of links are available as in `bin2stg` using the same `linkname` in the `rclusbin` function call. More specifically, the logistic (`"logit"`), probit (`"probit"`) and complementary log-log (`"cloglog"`) links are currently available.

As we have binary data, this function is also an implementation of the discrete partition version. When we handle binary random intercept models, we specify `ProspModInf = RClusProspModInf` and the function supplied as `StratModInf` in the `mlefn` function call depends on the following sampling schemes:

- **Proband only**: `StratModInf = RProbandStratModInf`;
- **All controls**: `StratModInf = R0StratModInf`;
- **Gamma probabilities**: `StratModInf = R1StratModInf`;

Note that no `StratModInf` function is required when we have a prospective sample.

Both `ProspModInf` and `StratModInf` require the same submodel function `modelfn` to work with. This is different from the structure we see in M linear predictor models in which both `modelfn` and `stratfn` are required. Currently available functions in the library are as follows:

- **Logistic**: `modelfn = binlogistic`.
- **Probit**: `modelfn = binprobit`.
- **Cloglog**: `modelfn = bincloglog`.

If the user wants to use another link other than the above three, he/she only need to write a new `modelfn` function. Again, the user will have to add a few lines of code in `rclusbin` to allow for this new link name.

The starting values of the parameters are calculated (if necessary) using the `glm` function with weights inside `rclusbin`. Therefore, an `offset` value is no longer required by the regression model in order to obtain consistent parameter estimates. As long as the link that the user is interested in can be handled by the function `glm`, there is no extra work required for the user to modify any command that are relevant.

REFERENCES

Becroft, D.M.O., Thompson, J.M.D. and Mitchell, E.A. (2002). The epidemiology of placental infarction at term, *Placenta*, **23**, 4:343–351.

Breslow, N.E. and Chatterjee, N. (1999). Design and analysis of two-phase studies with binary outcome applied to Wilms tumour prognosis. *Applied Statistics*, **48**, 457–468.

Clayton, D. and Hills, M. (1993). Statistical models in epidemiology. Oxford University Press, Oxford.

Genest, C. (1987). Frank's family of bivariate distributions. *Biometrika*, **74**, 3:549–555.

Jiang, Y., Scott, A.J. and Wild, C.J. (2006). Secondary analysis of case-control data. *Statistics in Medicine*, **25**, 1323–1339.

Jiang, Y. (2004). Semiparametric maximum likelihood for multi-phase response-selective sampling and missing data problems. Awarded Phd thesis in Statistics at the University of Auckland, New Zealand.

Lawless, J.F., Kalbfleish, J.D. and Wild, C.J. (1999). Semiparametric methods for response-selective and missing data problems in regression. *J. R. Statist. Soc.* B **61**, 413–38.

Le Cessie, S. & Van Houwelingen, J.C. (1994). Logistic regression for correlated binary data. *Appl. Statist.* **43**, 95–108.

Lee, A.J., McMurchy, L. & Scott, A.J. (1997). Re-using data from case-control studies. *Statist. in Med.* **16**, 1377–89.

Meester, S.G. & MacKay, J. (1994). A parametric model for cluster correlated categorical data. *Biometrics* **50**, 954–63.

Millar, R.B. (1992) Estimating the size-selectivity of fishing gear by conditioning on the total catch. *J. Amer. Statist. Assoc.* **87**, 962-968.

Mitchell, E.A., Scragg, R., Stewart, A.W., Becroft, D.M.O., Taylor, B.J., Ford, R.P.K., Hassall, I.B., Barry, D.M.J., Allen, E.M. and Roberts, A.P., (1991) Results from the first year of the New Zealand Cot Death Study. *New Zealand Medical Journal*, **104**, 71-76.

Neuhaus, J., Scott, A.J. and Wild, C.J. (2002). The analysis of retrospective family studies. *Biometrika*, **89**, 23-37.

Scott, A.J. and Wild, C.J. (1997). Fitting regression models to case-control data by maximum likelihood. *Biometrika* **84**, 57-71.

Scott, A.J. and Wild, C.J. (2001). Maximum likelihood for generalised case-conrol studies. *J. Statist. Planning and Inference* **96**, 3-27.

Scott, A.J. and Wild, C.J. (2003). Semiparametric maximum likelihood for two-phase sampling. *Unpublished manuscript.*

Seber, G.A.F and Wild, C.J. (1989) Nonlinear regression. *John Wiley and Sons, New York.*

Whittemore, A.S. (1995). Logistic regression of family data from case-control studies. *Biometrika* **82**, 57–67. (Correction: **84**, 989–90.)

Wrensch, M., Lee, M., Miike, R., Newman, B., Barger, G., Davis, R., Wiencke, J., & Neuhaus, J. (1997). Familial and personal medical history of cancer and nervous system conditions among adults with glioma and controls. *Am. J. Epidemiol.* **145**, 581–93.

Thompson, J.M.D., Clark,, P.M., Robinson, E., Becroft, DMO, Pattison, N.S., Glavish, N., Pryor, J.E., Wild, C.J., and Rees, B.A. and ., Mitchell, E.A. (2001). Risk factors for small-for-gestational-age babies: the Auckland Birthweight Collaborative (ABC) Study. *Journal of Paediatrics and Child Care*, **37**, 369–375.

Yee, T.W. and Wild, C.J. (1996). Vector generalised additive models. *Journal of the Royal Statistical Society* **B 58**, 481–493.