micsr objects

July 9, 2025

micsr provides a special `micsr` class for fitted models, that is particularly convenient for models fitted by maximum likelihood.

Estimation

With **micsr**, models are estimated using the `micsr::maximize` function which is an interface for three optimization routines, selected using the `opt` argument:

- `"bfgs"` uses the `stats::optim` function with the `method` argument set to `"BFGS"`,
- `"nlm"` uses the `stats::nlm` function which uses the Newton-Ralphson algorithm,
- `"newton"` uses the `micsr::newton` function which is a fast implementation of the Newton-Ralphson algorithm.

For simple log-likelihood function, `"newton"` is the fastest way to estimate the model, otherwise, `"bfgs"` is the safer.

The first argument is the function that returns the log-likelihood of the model and further argument of this function can be passed using the `...` argument of `maximize`:

- `gradient`, `hessian` and `information` are booleans which enable to compute the gradient, the hessian and the information matrix that are returned as attributes of the result,
- `sum` is a boolean: if `FALSE` the result is a vector containing the individual contributions to the log-likelihood and the `gradient` attribute is a matrix containing the individual contributions to elements of the gradients,
- `opposite` is a boolean: if `TRUE`, the opposite of the function, the gradient and the hessian are returned; this is useful if the optimization function performs a minimization,
- `y`, `X` and `weights` are the vector of response, the matrix of covariates and a vector of weights.

Several functions are provided to estimate models already available in core **R** (the **stats** package) and recommended package.

- `binomreg` can be used to estimate the binomial model as an alternative to `stats::glm` function. An `"identity"` link is provided to estimate the linear probability model and two-part formulas can be provided to estimate the probit model with instrumental variables. In this case, several method of estimation are provided selected by the `method` argument: either `"ml"` for maximum-likelihood, `"twosteps"` for the two-step estimator and `"minchisq"` for the minimum χ^2 estimator,
- `poisreg` estimates different flavors of count models. The basic use of this function estimate the Poisson model as an alternative to the `stats::glm` function with `family = poisson(link = 'log')`. Mixing distribution can be introduced with the `mixing` argument which is `"none"` by default but can be set to `"lognorm"` to get the log-normal Poisson model and to `"gamma"` to get the NegBin model (the `vlink` argument is either set to `"nb1"` or to `"nb2"` to get the two flavors of the Negbin model, which can also be estimated using the `MASS::glm.nb` function),
- `weibreg` estimates the Weibull model that is normally estimated using the `survival::survreg` function with the default `"weibull"` `dist` argument. Using the `model` argument, one can use the "accelerated failure time" (`"aft"`) or the "proportional hazard" parametrization (`"ph"`). Moreover, the most used mixing survival model is obtained by setting the `mixing` argument to `TRUE` (in this case, the gamma mixing distribution is used).
- `ordreg` estimates the ordered model, also called, for the logit link the "proportional odds logistic regression). A link argument enables to estimate the model with a probit, logit or cloglog link. This model can be estimated using the `MASS::polr` function but `ordreg` deals with censored responses.
- `tobit1` estimates models for truncated responses, using either censored or truncated samples (using the `sample` argument); in the first case we get the so-called tobit or tobit1 model, in the second case the truncated model. The `left` and `right` arguments can be set to indicate the truncation points (the convenient default is 0 for the former and $+\infty$ for the latter). Several methods of estimations are provided and selected using the `method` arguments: `"ml"` for maximum likelihood, `"lm"` for the (biased) linear estimator, `"twostep"` for the two-steps consistent estimator and `"trimmed"` for the symmetric trimmed estimator. If a two-part formula is provided, by default, an instrumental variable is performed, either using maximum likelihood or the minimum χ^2 estimator. If the `scedas` argument is not null, but set either to `"exp"` or `"pnorm"`, the heterosckedastic tobit model is estimated.

All these functions have in common with `stats::lm` their first arguments: `formula`, `data`, `subset`, `weights`, `na.action`, `offset` and `contrasts`. Supplementary arguments include `opt`, `maxit` and `trace` to select the optimization method with a given maximum number of iterations and level of printing, `start` to indicate a vector of starting values and `check_gradient` to check the correspondence between the analytical and the numerical gradients.

The returned objects, of class `micsr` includes:

- `coefficients`: the vector of coefficients,

- `model`: the model frame,
- `terms`: model terms,
- `value`: a vector of individual contribution to the log-likelihood,
- `gradient`: the matrix of individual contributions to the gradient, also called the estimating function,
- `hessian`: the hessian
- `info`: the estimation of the information matrix
- `fitted.values`: a vector of fitted values,
- `linear.predictors`: a vector of linear predictors, i.e., $X\hat{\beta}$
- `logLik`: a vector of length three containing the log-likelihood for the proposed, the saturated and the null models,
- `tests`: a vector of length three containing the three tests (the Wald, the score and the likelihood ratio tests) that "all the coefficients except the intercept are zero" (the equivalent of the F test for the linear model); these statistics can be used to compute different flavors of the coefficient of determination,
- `df.residual`: the number of fitted coefficients,
- `npar`: a named numeric vector, the names being the name of the group of coefficients and the value the number of parameter in the group,
- `est_method`: a character indicating the method of estimation, i.e., `ml` or `twostep`,
- `call`, `na.action`, `weights`, `offset`, `contrasts` and `xlevels` (as in `lm` objects),
- `family` a family object (as in `glm`),
- `check_gradient`: if the `check_gradient` argument is `TRUE`, the result is a list containing, for the gradient and the hessian, the maximal absolute value of the difference between the analytical and the numerical gradient and hessian; attribute `"gradient"` is a matrix containing the numerical and the analytical gradient and attributes `"anal_hess"` and `"num_hess"` contain the analytical and the numerical hessians.

Subsets of coefficients

Often, especially for advanced models, the whole set of fitted parameters can be splitted in several groups. For example, while fitted by maximum likelihood a limited dependent model (probit or tobit):

```
library(micsr)
bank_msq <- ivldv(federiv ~ eqrat + optval +  mktbk +
                      perfor + dealdum | . - eqrat - optval +
                      no_emp + no_subs + no_off,
                  data = federiv, method = "ml")
```

fitted coefficients belongs to 4 different groups:

- the coefficients associated with covariates (the coefficients of main interest,

- the coefficients of the residuals of the endogenous variables,
- the coefficients of the instruments,
- the parameters of the cholesky decomposition of the covariance matrix for the endogenous variables.

This information is stored in the `npar` element of the `micsr` object:

```
bank_msq$npar
```

```
 covariates       resid instruments        chol
          6           2          14           3
attr(,"default")
[1] "covariates" "resid"
```

It's a named list of integers containing the number of coefficients for each groups, in their order of appearance in the `coefficient` element of the `micsr` object. It may have a `"default"` attribute which indicates which subset should be selected by default. The selection of a subset of coefficients is performed by the `select_coef` function:

```
select_coef(bank_msq)
## (Intercept)       mktbk      perfor     dealdum       eqrat      optval
##           1           2           3           4           5           6
##   rho_eqrat  rho_optval
##           7           8
select_coef(bank_msq, subset = c("resid", "chol"))
##     rho_eqrat    rho_optval   eqrat|eqrat  optval|eqrat optval|optval
##             7             8            23            24            25
```

`select_coef` is called inside the `coef` and the `vcov` methods for `micsr` objects:

```
coef(bank_msq)
##   (Intercept)         mktbk        perfor       dealdum         eqrat
## -3.0493980864  0.0007126479  4.6919607187  0.8644131095 20.7983728258
##        optval     rho_eqrat    rho_optval
##  0.0887552265 -0.4478945968 -0.1407430396
coef(bank_msq, subset = c("resid", "chol"))
##     rho_eqrat    rho_optval   eqrat|eqrat  optval|eqrat optval|optval
##   -0.44789460   -0.14074304   56.80434859   -0.01092933    0.14746163
vcov(bank_msq, subset = c("resid"))
##                rho_eqrat    rho_optval
## rho_eqrat   0.0021818615 -0.0005167884
## rho_optval -0.0005167884  0.0045537894
```

Another way to select a subset of coefficients is to provide a regular expression for the `grep` argument. For example, to get all the coefficients that contains `"Intercept"`:

```
vcov(bank_msq, subset = "all", grep = "Intercept")
##                             (Intercept) instr_eqrat_(Intercept)
## (Intercept)                0.0026070311           -5.640990e-04
## instr_eqrat_(Intercept)   -0.0005640990            1.259446e-03
## instr_optval_(Intercept) -0.0001772582           -2.817664e-20
##                          instr_optval_(Intercept)
## (Intercept)                          -1.772582e-04
## instr_eqrat_(Intercept)              -3.381197e-20
## instr_optval_(Intercept)              1.259446e-03
```

The `npar` function extracts the number of the whole set of coefficients or of specific subsets:

```
npar(bank_msq)
## [1] 25
npar(bank_msq, subset = c("resid", "chol"))
## [1] 5
```

Estimating function, hessian and individual contribution to the log-likelihood

The estimating function is a $N \times K$ matrix containing the derivatives of the N elements of the likelihood function with respect to the K-lenght parameter vector. Its column-sum is the gradient that should be close to 0 at the optimum. The `sandwich::estfun` function is a generic that extract this information and several methods are provided for different objects. Some of these method are quite tedious as the matrix is not stored in the object containing the fitted models. For models of class `micsr`, this matrix is stored in the `gradient` element and the method is just: `x$gradient`. This matrix is particularly useful to compute the variance-covariance matrix estimator based on the outer-product of the gradient or on sandwich formula.

The hessian is the $K \times K$ matrix of second-derivatives of the log-likelihood function with respect with the parameters vector. Its stored in the `hessian` object of a `micsr` object. In sandwich estimators of the covariance matrix, the "meat" is $N(-H)^{-1}$ and it is how the `meat` method for `micsr` objects is defined.

The information matrix is either the variance of the gradient or the opposite of the expectation of the hessian. When the analytical computation of the information matrix is possible, it is stored in the `info` element of the `micsr` object.

The log-likelihood is the sum of N contributions: the individual contributions is a N-length vector, which is stored as the `value` element of the `micsr` object. This vector is particularly

usefull to compute Vuong's test.

Covariance matrix estimation

Three elements of the object can be used to compute different flavors of the covariance matrix of the coefficients:

- `gradient` to compute the outer-product of the gradient estimator,
- `hessian` to compute the hessian-based estimator,
- `info` to compute the information-based estimator.

These matrix are computed using the `vcov` method, which has a `vcov` argument that can be set to `"info"`, `"hessian"` or `"opg"`.

```
vcov(bank_msq, subset = "resid", vcov = "opg")
```

```
              rho_eqrat    rho_optval
rho_eqrat   0.0025591467 -0.0001606321
rho_optval -0.0001606321  0.0030387065
```

The vector of standard errors are often used, in particular in the table of coefficients. It can be obtained by taking the square root of the diagonal elements of the covariance matrix, or more simply by using `micsr::stder`.

```
sqrt(diag(vcov(bank_msq, subset = "resid")))
##  rho_eqrat rho_optval
## 0.04671040 0.06748177
stder(bank_msq, subset = "resid")
##  rho_eqrat rho_optval
## 0.04671040 0.06748177
```

Goodness of fit measures

For models fitted by maximum likelihood, most of the GOF statistics are based on the value of the objective function at the optimum. Three values of the log-likelihood are stored in the `logLik` element of a `micsr` object:

- the value at the optimum, for the fitted **model**,
- the value for the **saturated** model, ie., the model with no degrees of freedom,
- the value for the **null** model, ie., the model without any covariates.

```
pbt <- binomreg(mode ~ cost + ivtime + ovtime,
                data = mode_choice, link = 'probit')
pbt$logLik
```

```
    model saturated      null
-318.7685    0.0000 -370.6660
```

Any of these values can be extracted using the `logLik` method, which has a `type` argument:

```
logLik(pbt)
## 'log Lik.' -318.7685 (df=4)
logLik(pbt, type = "model")
## 'log Lik.' -318.7685 (df=4)
logLik(pbt, type = "saturated")
## 'log Lik.' 0 (df=842)
logLik(pbt, type = "null")
## 'log Lik.' -370.666 (df=1)
```

Comparing fitted models using the value of the log-likelihood is not relevant because introducing more covariates (even if they are irrelevant) will necessarily increase the value of the log-likelihood. **AIC** and **BIC** are two information measures that take into account the number of fitted parameters. The formulas are respectively:

- $-2\ln L + kK$
- $-2\ln L + K\ln N$

```
AIC(pbt)
## [1] 645.537
BIC(pbt)
## [1] 664.4802
AIC(pbt, type = "null")
## [1] 743.332
AIC(pbt, k = 5)
## [1] 657.537
```

The deviance is minus twice the difference between a model and the hypothetical saturated model. For a linear gaussian model, this is the sum of square residuals. The deviance method for `micsr` object has a `type` argument equal either to `"model"`(the default) or `"null"`. In the latter case, we obtain the "null deviance":

```
deviance(pbt)
## [1] 637.537
deviance(pbt, type = "null")
## [1] 741.332
```

Finally, the relevance of a proposed model can be addressed using a test that "all the coefficients except the intercept are zero", which is the equivalent of the **F** test for the linear regression model. Three tests can be conducted, either the Wald test, the score test and the likelihood ratio test. The values of the statistics are stored in the `test` element of the result:

```
pbt$test
##      wald     score        lr
##  80.74629  93.55961 103.79495
```

Summary

The `summary` method use the preceding infrastructure to compute the usual table of coefficients that contains the estimators, the standard errors, the statistics and the probability values. A subset of coefficients can be obtained using the `subset` argument and the covariance matrix is chosen using the `vcov` argument:

```
summary(bank_msq, subset = c("chol", "resid"), vcov = "opg")
```

```
Maximum likelihood estimation
                 Estimate  Std. Error  z-value Pr(>|z|)
rho_eqrat     -0.44789460  0.05061637  -8.8488  < 2e-16 ***
rho_optval    -0.14074304  0.05584664  -2.5202  0.01173 *
eqrat|eqrat   56.80434859  0.34566405 164.3340  < 2e-16 ***
optval|eqrat  -0.01092933  0.00716298  -1.5258  0.12706
optval|optval  0.14746163  0.00020104 733.4920  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

log-Likelihood: -902.15
```