Computation of the variance-covariance matrix

An example with the `Countr` package.

Tarak Kharrat[1] and Georgi N. Boshnakov[2]

[1]Salford Business School, University of Salford, UK.
[2]School of Mathematics, University of Manchester, UK.

Abstract

Computing standard errors and confidence intervals for estimated parameters is a common task in regression analysis. These quantities allow the analyst to quantify the certainty (*confidence*) associated with the obtained estimates. In `Countr` two different approaches have been implemented to do this job. First, using asymptotic MLE (Maximum Likelihood Estimator) theory, numeric computation of the inverse Hessian matrix can be used as a consistent estimator of the variance-covariance matrix, which in turn can be used to derive standard errors and confidence intervals. The second option available in `Countr` is to use bootsrap (Efron et al., 1979). In this document, we give the user an overview of how to do to the computation in `Countr`.

This vignette is part of package Countr (see Kharrat et al., 2019).

Before starting our analysis, we need to load the useful packages. On top of `Countr`, the `dplyr` package (Wickham and Francois, 2016) will be used:

```
library(Countr)
library(dplyr)
library(xtable)
```

1 Maximum Likelihood estimator (MLE)

1.1 Theory

Let $f(y, \mathbf{x}, \boldsymbol{\theta})$ be the probability density function of a renewal-count model, where y is the count variable, $\mathbf{x}$ the vector of covariate values and $\boldsymbol{\theta}$ the vector of coefficients to be estimated ($q \times 1$ vector). Define the log-likelihood by $\mathcal{L} = \sum_{i=1}^{n} ln f(y_i|\mathbf{x}_i, \boldsymbol{\theta}_i)$. Under regularity conditions (Cameron and Trivedi, 2013, see for example)[Section 2.3], the MLE $\hat{\boldsymbol{\theta}}$ is the solution of the first-order conditions,

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}} = \sum_{i=1}^{n} \frac{\partial ln f_i}{\partial \boldsymbol{\theta}} = 0, \tag{1}$$

where $f_i = f(y_i|\mathbf{x}_i, \boldsymbol{\theta}_i)$ and $\frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}}$ is a $q \times 1$ vector.

Let $\boldsymbol{\theta}_0$ be the *true* value of $\boldsymbol{\theta}$. Using MLE theory and assuming regularity conditions, we obtain $\hat{\boldsymbol{\theta}} \xrightarrow{p} \boldsymbol{\theta}_0$ and

$$\sqrt{n}(\hat{\boldsymbol{\theta}}_{ML} - \boldsymbol{\theta}_0) \xrightarrow{d} \mathcal{N}[\mathbf{0}, \mathbf{V}^{-1}], \tag{2}$$

where the $q \times q$ matrix $\mathbf{V}$ matrix is defined as

$$\mathbf{V} = - \lim_{n \to \infty} \frac{1}{n} \mathbb{E}\left[\sum_{i=1}^{n} \frac{\partial^2 \ln f_i}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}'} |_{\boldsymbol{\theta}_0} \right]. \tag{3}$$

To use this result, we need a consistent estimator of the variance matrix $\mathbf{V}$. Many options are available: the one implemented in `Countr` is known as the *Hessian estimator* and simply evaluates Equation 3 at $\hat{\boldsymbol{\theta}}$ without taking expectation and limit.

1.2 Implementation in `Countr`

The easiest way to compute the variance-covariance matrix when fitting a renewal-count model with `Countr` is to set the argument `computeHessian` to `TRUE` when calling the fitting routine `renewalCount()`. Note that this is the default behaviour and it will save the matrix in the returned object (slot `vcov`). Following standard practice in `R`, the matrix can be extracted or recomputed using the `vcov()` method. We show here an example with the fertility data.

```
data(fertility)
form <- children ~ german + years_school + voc_train + university + Religion +
year_birth + rural + age_marriage
gam <- renewalCount(formula = form, data = fertility, dist = "gamma",
                    computeHessian = TRUE,
                    control = renewal.control(trace = 0, method = "nlminb")
                    )
v1 <- gam$vcov
v2 <- vcov(gam)

all(v1 == v2)
```

```
[1] TRUE
```

The above `vcov()` method simply extracts the variance-covariance matrix if it has been computed at fitted. Otherwise, the function will re-compute it. The user can choose the computation method by specifying the `method` argument: `asymptotic` for numerical hessian computation or `boot` for the bootstrap method. In the latter case, user can customise the bootstrap computation as will be discussed in Section 2 by using the `...` argument.

Parameters' standard errors and confidence intervals can be computed by calls to the generic functions `se.coef()` and `confint()`. The hessian method can be specified by setting the argument `type = "asymptotic"`.

```
se <- se.coef(gam, type = "asymptotic")
se
```

```
              rate_          rate_germanyes        rate_years_school
          0.2523375               0.0590399                0.0265014
  rate_voc_trainyes      rate_universityyes    rate_ReligionCatholic
          0.0358390               0.1296149                0.0578032
rate_ReligionMuslim rate_ReligionProtestant          rate_year_birth
          0.0698429               0.0622954                0.0019471
     rate_ruralyes       rate_age_marriage                   shape_
          0.0311876               0.0053274                0.0710611
```

```
ci <- confint(gam, type = "asymptotic")
ci
```

```
                           2.5 %      97.5 %
rate_                   1.0621317  2.0512766
rate_germanyes         -0.3054815 -0.0740495
rate_years_school      -0.0202566  0.0836270
rate_voc_trainyes      -0.2141792 -0.0736929
rate_universityyes     -0.4000943  0.1079867
rate_ReligionCatholic   0.0924771  0.3190616
rate_ReligionMuslim     0.3857453  0.6595243
rate_ReligionProtestant -0.0149604  0.2292330
rate_year_birth        -0.0014721  0.0061603
rate_ruralyes          -0.0056353  0.1166177
rate_age_marriage      -0.0392385 -0.0183553
shape_                  1.3000440  1.5785983
```

One can validate the result obtained here by comparing them to what is reported in Winkelmann (1995, Table 1).

2 Bootstrap

2.1 Theory

The type of bootstrap used in `Countr` is known as *nonparametric* or bootstrap pairs. It is valid under the assumption that $(y_i, \mathbf{x}_i)$ is iid. The algorithm works as follows: (a) Generate a pseudo-sample of size n, $(y_l^*, \mathbf{x}_l^*)$, $l = 1, \ldots, n$, by sampling *with replacement* from the original pairs $(y_i, \mathbf{x}_i)$, $i = 1, \ldots, n$. (b) Compute the estimator $\hat{\boldsymbol{\theta}}_1$ from the pseudo-sample generated in 1. (c) Repeat steps 1 and 2 B times giving B estimates $\hat{\boldsymbol{\theta}}_1, \ldots, \hat{\boldsymbol{\theta}}_B$. (d) The bootstrap estimate of the variance-covariance matrix is given by $\hat{\mathbf{V}}_{Boot}[\hat{\boldsymbol{\theta}}] = \frac{1}{B-1}\sum_{b=1}^{B}(\hat{\boldsymbol{\theta}}_b - \bar{\boldsymbol{\theta}})(\hat{\boldsymbol{\theta}}_b - \bar{\boldsymbol{\theta}})'$ where $\bar{\boldsymbol{\theta}} = [\bar{\theta}_1, \ldots, \bar{\theta}_q]$ and $\bar{\theta}_j$ is the sample mean $\bar{\theta}_j = (1/B)\sum_{b=1}^{B}\hat{\boldsymbol{\theta}}_{j,b}$.

Asymptotically ($B \to \infty$), the bootstrap variance-covariance matrix and standard errors are equivalent to their robust counterpart obtained by sandwich estimators. In practice, using $B = 400$ is usually recommended (Cameron and Trivedi, 2013, Section 2.6.4)}

2.2 Implementation in `Countr`

The bootstrap computations in `Countr` are based on the `boot()` function from the package with the same name (Canty and Ripley, 2017).

The variance-covariance matrix is again computed with the renewal method for `vcov()` by specifying the argument `method = "boot"`. The computation can be further customised by passing other options accepted by `boot()` other than `data` and `statistic` which are provided by the `Countr` code. Note that the matrix is only computed if it is not found in the passed `renewal` object. The bootstrap sample is actually computed by a separate function `addBootSampleObject()`, which computes the bootstrap sample and adds it as component `boot` to the renewal object. Functions like `vcov()` and `confint()` check if a bootstrap sample is already available and use it is. It is a good idea to call `addBootSampleObject()` before attempting computations based on bootstrapping. We show below how to update the previously fitted gamma model with 400 bootstrap iterations using the parallel option and 14 CPUs. if B is large and depending on how fast the model can be fitted, this computation may be time consuming.

```
gam <- addBootSampleObject(gam, R = 400, parallel = "multicore", ncpus = 14)
```

Once the object is updated, the variance-covariance matrix is computed by `vcov` in a straightforward way:

```
gam$vcov <- matrix()
varCovar <- vcov(gam, method = "boot")
```

This arranges the above results in a table (see Table 1):

```
capboot <- "Bootstrap variance-covariance matrix of model \texttt{gam}."
print(xtable(varCovar, digits = -1, caption = capboot, label = "tab:varCovar"),
      rotate.colnames = TRUE, floating.environment = "sidewaystable" )
```

Bootstrap standard errors are also very easy to compute by calling `se.coef()` with argument `type="boot"`. As discussed before, if the `boot` object is not found in the `renewal` object, users can customise the `boot()` call by passing the appropriate arguments in `"..."`.

```
se_boot  <- se.coef(gam, type = "boot")
se_boot
```

```
               rate_          rate_germanyes       rate_years_school
           0.2705321               0.0610739               0.0268098
  rate_voc_trainyes      rate_universityyes  rate_ReligionCatholic
           0.0375714               0.1257020               0.0564330
rate_ReligionMuslim rate_ReligionProtestant         rate_year_birth
           0.0684739               0.0608356               0.0019361
      rate_ruralyes       rate_age_marriage                  shape_
           0.0322879               0.0054811               0.1133149
```

Finally bootstrap confidence intervals can also be computed by `confint()` using the same logic described for `se.coef()`. Different types of confidence intervals are available (default is normal) and can be selected by choosing the appropriate type in `bootType`. We refer the user to the `boot` package (Canty and Ripley, 2017) for more information.

```
ci_boot  <- confint(gam, level = 0.95, type = "boot", bootType = "norm")
ci_boot
```

```
Bootstrap quantiles, type =  normal

                             2.5 %     97.5 %
rate_                    1.0105534  2.0710199
rate_germanyes          -0.3116866 -0.0722812
rate_years_school       -0.0176548  0.0874376
rate_voc_trainyes       -0.2211511 -0.0738740
rate_universityyes      -0.4072628  0.0854802
rate_ReligionCatholic    0.0985243  0.3197377
rate_ReligionMuslim      0.3871631  0.6555759
rate_ReligionProtestant -0.0133962  0.2250750
rate_year_birth         -0.0016933  0.0058961
rate_ruralyes           -0.0098069  0.1167594
rate_age_marriage       -0.0401915 -0.0187062
shape_                   1.1860640  1.6302501
```

We conclude this analysis by saving the workspace to avoid re-running the computation in future exportation of the document:

```
save.image()
```

References

Cameron, A. C. and Trivedi, P. K. (2013). *Regression analysis of count data*, volume 53. Cambridge university press.

Canty, A. and Ripley, B. D. (2017). *boot: Bootstrap R (S-Plus) Functions*. R package version 1.3-20.

Efron, B. et al. (1979). Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, 7(1):1–26.

Kharrat, T., Boshnakov, G. N., McHale, I., and Baker, R. (2019). Flexible regression models for count data based on renewal processes: The Countr package. *Journal of Statistical Software*, 90(13):1–35.

Wickham, H. and Francois, R. (2016). *dplyr: A Grammar of Data Manipulation*. R package version 0.5.0.

Winkelmann, R. (1995). Duration dependence and dispersion in count-data models. *Journal of Business & Economic Statistics*, 13(4):467–474.

	rate_	rate_germanyes	rate_years_school	rate_voc_trainyes	rate_universityyes	rate_ReligionCatholic	rate_ReligionMuslim	rate_ReligionProtestant	rate_year_birth	rate_ruralyes	rate_age_marriage	shape_
rate_	7.3E-02	5.0E-03	-5.7E-03	8.8E-04	1.7E-02	-7.2E-04	-3.5E-03	-6.3E-04	-1.8E-04	-2.0E-03	-3.6E-04	8.6E-03
rate_germanyes	5.0E-03	3.7E-03	-7.0E-04	-4.4E-04	1.8E-03	-9.5E-04	1.2E-03	-1.3E-03	-3.9E-05	-2.3E-04	6.8E-05	1.2E-05
rate_years_school	-5.7E-03	-7.0E-04	7.2E-04	-1.6E-04	-2.3E-03	-5.1E-05	1.4E-04	-4.9E-05	5.6E-06	1.2E-04	-1.9E-05	6.3E-05
rate_voc_trainyes	8.8E-04	-4.4E-04	-1.6E-04	1.4E-03	1.7E-03	5.5E-05	2.1E-04	9.6E-06	9.9E-06	3.5E-04	-3.0E-06	4.8E-04
rate_universityyes	1.7E-02	1.8E-03	-2.3E-03	1.7E-03	1.6E-02	5.2E-04	5.0E-04	5.3E-04	3.1E-06	4.0E-04	3.1E-05	4.1E-04
rate_ReligionCatholic	-7.2E-04	-9.5E-04	-5.1E-05	5.5E-05	5.2E-04	3.2E-03	1.6E-03	2.8E-03	-2.4E-06	2.1E-04	-3.1E-05	-6.7E-05
rate_ReligionMuslim	-3.5E-03	1.2E-03	1.4E-04	2.1E-04	5.0E-04	1.6E-03	4.7E-03	1.4E-03	-1.8E-05	3.8E-04	1.8E-05	8.1E-05
rate_ReligionProtestant	-6.3E-04	-1.3E-03	-4.9E-05	9.6E-06	5.3E-04	2.8E-03	1.4E-03	3.7E-03	1.7E-06	1.9E-04	-4.4E-05	-2.3E-04
rate_year_birth	-1.8E-04	-3.9E-05	5.6E-06	9.9E-06	3.1E-06	-2.4E-06	-1.8E-05	1.7E-06	3.7E-06	-2.1E-07	-2.3E-06	-2.4E-05
rate_ruralyes	-2.0E-03	-2.3E-04	1.2E-04	3.5E-04	4.0E-04	2.1E-04	3.8E-04	1.9E-04	-2.1E-07	1.0E-03	1.3E-05	1.3E-04
rate_age_marriage	-3.6E-04	6.8E-05	-1.9E-05	-3.0E-06	3.1E-05	-3.1E-05	1.8E-05	-4.4E-05	-2.3E-06	1.3E-05	3.0E-05	8.9E-05
shape_	8.6E-03	1.2E-05	6.3E-05	4.8E-04	4.1E-04	-6.7E-05	8.1E-05	-2.3E-04	-2.4E-05	1.3E-04	8.9E-05	1.3E-02

Table 1: Bootstrap variance-covariance matrix of model extttgam.