Computation of the variance-covariance matrix
An example with the Countr package.

Tarak Kharrat¹ and Georgi N. Boshnakov²

¹Salford Business School, University of Salford, UK.
²School 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 bootstrap (Efron et al., 1979). In this document, we give the user an overview of how to do the computation in Countr.

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

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

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

Let $\theta_0$ be the true value of $\theta$. Using MLE theory and assuming regularity conditions, we obtain $\hat{\theta} \overset{d}{\rightarrow} \theta_0$ and

$$
\sqrt{n}(\hat{\theta}_{ML} - \theta_0) \overset{d}{\rightarrow} \mathcal{N}(0, V^{-1}),
$$

where the $q \times q$ matrix $V$ matrix is defined as

$$
V = - \lim_{n \to \infty} \frac{1}{n} \mathbb{E} \left[ \sum_{i=1}^{n} \frac{\partial^2 \ln f_i}{\partial \theta \partial \theta'} | \theta_0 \right].
$$
To use this result, we need a consistent estimator of the variance matrix $V$. Many options are available: the one implemented in Countr is known as the Hessian estimator and simply evaluates Equation 3 at $\hat{\theta}$ 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()`.

```r
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)
```

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"`.

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

```r
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
```

```r
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 \((y_i, x_i)\) is iid. The algorithm works as follows: (a) Generate a pseudo-sample of size \(n\), \((y^*_l, x^*_l)\), \(l = 1, \ldots, n\), by sampling with replacement from the original pairs \((y_i, x_i)\), \(i = 1, \ldots, n\). (b) Compute the estimator \(\hat{\theta}_1\) from the pseudo-sample generated in 1. (c) Repeat steps 1 and 2 \(B\) times giving \(\hat{\theta}_1, \ldots, \hat{\theta}_B\). (d) The bootstrap estimate of the variance-covariance matrix is given by \(\hat{V}_{\text{Boot}}[\hat{\theta}] = \frac{1}{B-1} \sum_{b=1}^{B} (\hat{\theta}_b - \bar{\theta})(\hat{\theta}_b - \bar{\theta})'\) where \(\bar{\theta} = [\bar{\theta}_1, \ldots, \bar{\theta}_q]\) and \(\bar{\theta}_j\) is the sample mean \(\bar{\theta}_j = (1/B) \sum_{b=1}^{B} \hat{\theta}_{j,b}\).

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.

\[
\text{gam} \leftarrow \text{addBootSampleObject}(\text{gam}, R = 400, \text{parallel} = "\text{multicore}"`, ncpus = 14)\]
Once the object is updated, the variance-covariance matrix is computed by `vcov` in a straightforward way:

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

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

```r
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 "...".

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

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.

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

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

```r
save.image()
```
References


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

<table>
<thead>
<tr>
<th>shape</th>
<th>rate_age_marriage</th>
<th>rate_ruralyes</th>
<th>rate_year_birth</th>
<th>rate_voc_trainyes</th>
<th>rate_universityyes</th>
<th>rate_ReligionCatholic</th>
<th>rate_ReligionMuslim</th>
<th>rate_ReligionProtestant</th>
<th>rate_germanyes</th>
<th>rate_years_school</th>
<th>rate_vot_trainyes</th>
<th>rate_year_production</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.6E-03</td>
<td>1.2E-04</td>
<td>7.3E-02</td>
<td>1.3E-01</td>
<td>3.0E-02</td>
<td>1.3E-02</td>
<td>2.1E-01</td>
<td>1.3E-02</td>
<td>2.2E-02</td>
<td>6.9E-03</td>
<td>9.6E-03</td>
<td>2.3E-02</td>
<td>9.6E-03</td>
</tr>
</tbody>
</table>