Journal of Statistical Software

MMMMMM YYYY, Volume VV, Issue II. *http://www.jstatsoft.org/*

Flexible Regression Models for Count Data Based on Renewal Processes: The Countr Package

Tarak Kharrat
Salford Business School

Georgi N. Boshnakov
University of Manchester

Ian McHale
Salford Business School

Rose Baker
Salford Business School

Abstract

A new alternative to the standard Poisson regression model for count data is suggested. This new family of models is based on discrete distributions derived from renewal processes, i.e., distributions of the number of events by some time t. Unlike the Poisson model, these models have, in general, time-dependent hazard functions. Any survival distribution can be used to describe the inter-arrival times between events, which gives a rich class of count processes with great flexibility for modelling both underdispersed and overdispersed data. The R package **Countr** provides a function, `renewal()`, for fitting renewal count regression models and methods for working with the fitted models. The interface is designed to mimic the `glm()` interface and standard methods for model exploration, diagnosis and prediction are implemented. Package **Countr** implements state-of-the-art recently developed methods for fast computation of the count probabilities. The package functionalities are illustrated using a fertility dataset.

Keywords: renewal process, duration dependence, count data, Weibull distribution, convolution, Richardson extrapolation.

1. Introduction

Modelling a count variable (the number of events occurring in a given time interval) is a common task in many fields such as econometrics, social sciences, sports modelling, marketing, physics or actuarial science just to name a few. The standard approach is to use the Poisson model, where $Y|x \sim Poisson(\mathsf{E}(Y|x) = \exp(x^\top \beta))$. Here Y is predicted given covariates with values x, using regression coefficients β. This model was built around a one-to-one

correspondence between the count model (Poisson) and the distribution of the inter-arrival times (exponential). Perhaps this conceptual elegance contributed to its popularity. With this elegance comes some limitation: the Poisson model restricts the (conditional) variance to be equal to the (conditional) mean. This situation is rarely observed in real life data and among the thousands of alternatives proposed in the literature (see for example Winkelmann (2013) or Cameron and Trivedi (2013) for a review), only a few retain the correspondence between the count model and the timing process. This correspondence is not only a conceptual elegance but also offers the researcher the flexibility to model the aspect (counting or timing) that is perhaps known better (from the available data) and to draw conclusions (typically prediction) using the other. A classic example is the exponential distribution used in radioactive decay which leads to Poisson count model. Another very good example in the marketing context can be found in McShane, Adrian, Bradlow, and Fader (2008).

Another limitation of the Poisson model results from the memorylessness property of the exponential distribution. In fact, this property states that the probability of having an arrival during the next $[t, t + \Delta t]$ time period (where $t > 0$ and $\Delta t > 0$) is independent of when the last arrival occured. In many situations, this assumption is not realistic and the history of the process can be informative about future occurrences.

One way to incorporate the history of the process in the modelling process is to make the current probability of an occurence depend on the number of previous event occurences. These models are known as *occurence dependence* and they are said to display true contagion. Bittner, Nussbaumer, Janke, and Weigel (2007) gave a discrete time example where the probability of scoring a goal in soccer in the current unit of time depends on the number of goals scored previously. The modelling process resulted in a negative binomial distribution.

Another way to take advantage of the process history is to assume that the time since last observed event is informative about the probability of a future occurence. Inter-arrival times between events are still assumed to be independent and identically distributed but the hazard function, defined by $h(t) = \frac{f(t)}{1-F(t)}$ where $f(t)$ and $F(t)$ are the density and the cummulative probability function, is no longer a constant function of time (as in the exponential case) but is replaced by a time-varying function. These type of models display *duration dependence* where negative duration dependence is obtained by a decreasing hazard function (of time) and positive duration dependence by an increasing hazard function. As noted by Winkelmann (1995), *"Events are 'dependent' in the sense that the occurence of at least one event (in contrast to none) up to time t influences the occurence in $t + \Delta t$"*. This class of models is known as *renewal processes* and will form the main focus of this paper.

The key quantity when studying renewal processes (and time to event in general) is the hazard function. Not only does it fully characterize the inter-arrival timing distribution but it also relates to the type of dispersion observed in the corresponding count data. In particular, Winkelmann (1995) established that if the hazard function is monotonic, increasing (decreasing) hazard corresponds to count data with under-dispersion (over-dispersion); the constant hazard characterizing the exponential distribution corresponds to data with equi-dispersion. Therefore, allowing for a more flexible hazard function results in more flexible counting processes able to accomodate over-dispersed and under-dispersed, as well as equi-dispersed data.

Winkelmann (1995) was the first to comment on the usefulness of renewal process models and derived a count model based on gamma distributed inter-arrival times. The choice of

the gamma distribution was justified by computational necessity. In fact, the reproductive property of the gamma distribution (sums of independent gamma distributions are gamma distributed) leads to a simple form for the derived gamma count probability. McShane *et al.* (2008) derived a closed formula for the count probability of a renewal process based on Weibull inter-arrival times using series expansion. The same approach has been used by Jose and Abraham (2011) and Jose and Abraham (2013) to derive a counting process with Mittag-Leffler and Gumbel inter-arrival times, respectively.

Despite the attractive properties of count models based on renewal processes, their use is still limited in practice where Poisson, geometric and negative binomial are usually preferred. Perhaps the main reason is the lack of available software to easily fit this new type of models. The development of **Countr** (Kharrat and Boshnakov 2016), available from the Comprehensive R Archive Network (CRAN), is meant to fill in this gap and complete the practioners' toolbox for modeling count data in R (**?**).

The **Countr** package provides a function, `renewal()`, for fitting count regression models based on renewal distributions. It offers several built-in renewal distributions and supports custom distributions. The design of the fitting function (`renewal()`) and the methods that act on the object returned by it is meant to mimic the familiar user interface associated with a number of R modelling functions, especially `glm()` (Chambers, Hastie, and Pregibon 1990) from package **stats** (**?**) or `hurdle()` and `zeroinfl()` (Zeileis, Kleiber, and Jackman 2008).

The remainder of this paper is laid out as follows. In Section 2, we briefly review the fundamental relationship between a timing process and the resulting count model as well the renewal regression models considered in **Countr**. The package design is discusssed in Section 3 and several working examples are given in Section 4. Some closing remarks are addressed in Section 5. We conclude and discuss future work in Section 6.

2. Models

2.1. Count models and inter-arrival times

The distribution of non-negative integer valued discrete random variables, count distributions for short, can be used as the distribution of the number of events in a given time interval, and vice versa. A powerful method to specify count distributions then can be based on models of the times between the events.

Consider a stochastic process starting at time $t = 0$ which produces a sequence of events. Let τ_1 be the time of the first event and, in general, τ_k be the time between the $(k-1)$th and the kth event, $k \in \mathbb{N}$. The τ_k's are known as *inter-arrival times* or *waiting times*. The arrival time of the mth event is

$$a_m = \sum_{k=1}^{m} \tau_k, \qquad m = 1, 2, \ldots,$$

with cummulative probability function $F_m(t) = \mathsf{P}(a_m < t)$.

Let $N_t = N(t)$ denote the total number of events in $[0, t)$. For any fixed t (the observation horizon), N_t is the count variable we wish to model. We have $\mathsf{P}(N_t \geq m) = F_m(t)$ and $\mathsf{P}(N_t < m) = 1 - F_m(t)$, since $N_t \geq m$ if and only if the mth event occurs before time t.

Moreover, the probability, $P_m(t)$, for exactly m events before time t is

$$\begin{aligned} P_m(t) &\equiv \mathsf{P}(N_t = m) \\ &= \mathsf{P}(N_t \geq m) - \mathsf{P}(N_t \geq m+1) \\ &= F_m(t) - F_{m+1}(t) \end{aligned} \tag{1}$$

For fixed t, Equation (1) shows how a count distribution, $\{P_m(t),\ m = 0, 1, \ldots\}$, can be obtained from $\{F_m(t),\ m = 0, 1, \ldots\}$, which in turn can be specified flexibly by the inter-arrival distributions.

More specifically, let $\{\tau_k\}_{k \in N}$ be independent and identically distributed (iid) random variables with common density $f(\tau)$. In this case the process is called a *renewal process* (see Feller 1970, for a formal definition) and Equation (1) can be used to derive the following recursive relationship:

$$\begin{aligned} P_{m+1}(t) &= \int_0^t F_m(t-u)\,\mathrm{d}F(u) - \int_0^t F_{m+1}(t-u)\,\mathrm{d}F(u) \\ &= \int_0^t P_m(t-u)\,\mathrm{d}F(u), \qquad \text{for } m = 1, 2, \ldots, \end{aligned} \tag{2}$$

where $P_0(u) = S(u) = 1 - F(u)$ (a survival function). Equation (2) can be understood intuitively: the probability of exactly $m+1$ events occurring by time t is the probability that the first event occurs at time $0 \leq u < t$, and that exactly m events occur in the remaining time interval, integrated over all times u. $P_1(t), \ldots, P_m(t)$ can be generated in turn by evaluating this integral. Several methods implemented in **Countr** for the demanding task of computing $P_m(t)$ are summarized in Section 5.1. Readers interested in the computational details are referred to Baker and Kharrat (2016).

We use the term *count distribution* or *renewal count distribution* for the distribution of N_t and qualify it with the name of the inter-arrival distribution for a particular distribution of the inter-arrival times. For example, *Weibull count distribution* refers to the count model arising from a renewal process with inter-arrival times having a Weibull distribution.

2.2. Renewal regression models

The regression models fitted by **Countr** are in the spirit of the generalised linear models (McCullagh and Nelder 1989) and consist of two main components: a conditional distribution of the response variable (given the covariates, if any) and one or more linear equations relating parameters to covariates, possibly via link functions.

More formally, let Y be the response variable of interest, x a vector of covariates and $\mathcal{D}$ a renewal count distribution. We assume that

$$Y|x \sim \mathcal{D}(\boldsymbol{\theta}), \tag{3}$$

where $\boldsymbol{\theta}$ is the vector of the parameters of $\mathcal{D}$.

One or more parameters of the distribution may depend linearly on covariates via link functions. The equation for the kth parameter then is:

$$g_k(\theta_k) = x^\top \beta_k, \tag{4}$$

where g_k is the link function for the kth parameter, x the covariates and β_k the corresponing vector of regression parameters. Typically, covariates are related to a location parameter but it is helpful in some applications to be able to let other parameters depend on covariates.

We call these models *renewal regression models*. Note that, in general, the renewal distributions are not from the exponential family. For comparison, in standard generalised linear models (glm) the distribution is taken from the exponential family of distributions and the mean, transformed by a link function, is a linear combination of the covariates.

2.3. Choosing the inter-arrival time distribution

As discussed before, count models arising from renewal processes provide very flexible families of distributions. Perhaps the simplest way to use them is to simply ignore their connections to renewal theory. Several models can be tried and users can choose the model that provides the best fit to the data using standard goodness of fit tests (for example chi-squared) or compare information criteria such as the Akaike information criterion (AIC) or the Bayesian information criterion (BIC).

In some applications however, the researcher may have some information about the inter-arrival time process which can lead to a particular choice of model. For example, assume that a researcher is interested in modelling the number of occurences by some time horizon t. He has data on the observed count for a number, n, of individuals, together with a set of individual covariates $\boldsymbol{x}_i, i = 1, \ldots, n$. If data on time to first event are also available, the researcher can fit a parametric hazard model using package **flexsurv** (Jackson 2016), choose the parametric model that presents the best fit and use the associated renewal count family to model his data. This approach has been used in Kharrat (2016, Chapter 4).

3. Package design

The **Countr** package is available from CRAN `https://cran.r-project.org/web/packages/Countr` and can be installed using the standard R tools.

The main function in **Countr** is `renewal()`. It fits renewal regression models for count data using maximum likelihood. Several built-in distributions for the inter-arrival times are provided. These are `"weibull"`, `"gamma"`, `"gengamma"` and `"burr"`. User-defined distributions are also supported.

The `renewal()` function returns the fitted model as an object from S3 class `"renewal"`. The standard interface to the modelling functions is maintained, as much as possible. In particular, methods for `summary()`, `predict()`, `confint()`, `coef()` and similar functions are available.

The **Countr** package also exports functions for the computation of the probabilities associated with several renewal count models. The probability computations are rather intensive and are mostly implemented in C++ with the help of the **RcppArmadillo** (Eddelbuettel and Sanderson 2014) package. Several methods are provided offering various degrees of trade-off between speed and accuracy, see Section 5.1.

4. Fitting renewal regression models

The examples below assume that the package is made available in the current session via

```
R> library("Countr")
```

The fertility dataset We illustrate the usage of **Countr** with the fertility data, first described in Winkelmann (1995, Section 5) and re-analyzed by McShane *et al.* (2008) and Baker and Kharrat (2016). The dataset is available when **Countr** is attached. It can also be loaded independently using `data()` e.g.

```
R> data("fertility", package = "Countr")
R> head(fertility)
```

```
  GERMAN EDU VOC UNI CATH PROT MUSL RURAL YEAR_OF_B AGEMARR Y
1      0   8   0   0    0    0    0     1        42      20 2
2      0   8   0   0    0    0    0     1        55      21 3
3      0   8   0   0    0    0    0     1        51      24 2
4      0   8   0   0    0    0    0     0        54      26 4
5      0   8   0   0    0    0    0     1        46      22 2
6      0   8   0   0    0    0    0     0        41      18 2
```

The `fertility` dataset contains information about a sample of 1,243 women who were over 44 years old in 1985 and answered the questions of the German Socio-Economic Panel. The responses are arranged in a data frame with one row for each person and 11 columns, coded as follows:

- `GERMAN` — German nationality, dummy variable.
- `EDU` — general education, measured as years of schooling.
- `VOC`, `UNI` — post-secondary education: vocational training (`VOC`) and/or university (`UNI`), dummy variables.
- `CATH`, `PROT`, `MUSL` — religion: catholic (`CATH`), protestant (`PROT`), muslim (`MUSL`), with other or none as reference group without its own column in the dataset.
- `RURAL` — rural, dummy variable.
- `YEAR_OF_B` — year of birth.
- `AGEMARR` — age at marriage.
- `Y` — number of children.

The response variable considered here is the number of children per woman (`Y`). The average number of children observed in this sample is 2.384 and variance of 2.33, so there is a hint for under-dispersion in the response variable's marginal distribution.

Renewal regression models are fitted with the function `renewal()`. It has been designed to mimic the GLM functionality in R. In fact, users familiar with `glm()` should recognize several common arguments in `renewal()`'s interface:

```
R> renewal(formula, data, subset, na.action, weights, offset,
+    dist = c("custom", "weibull", "weibullgam", "gamma", "gengamma", "burr"),
+    anc = NULL, convPars = NULL, link = NULL, time = 1.0,
+    control = renewal.control(...), customPars = NULL,
+    seriesPars = NULL, weiMethod = NULL,
+    computeHessian = TRUE, model = TRUE, y = TRUE, x = FALSE, ...)
```

where the first line contains the standard model-frame specifications, the last line controls some components of the returned object and the remaining arguments are specific to the renewal regression model.

The minimum required inputs are `formula` (an R formula), `data` (a data frame) and `dist` (a character string). Argument `formula` describes the model, `data` contains the values of the response and the covariates, while `dist` specifies the desired count model distribution.

The fitting process is based on maximum likelihood using optimization routines implemented in package **optimx** (Nash and Varadhan 2011). Users can customize different aspects of the fitting process and control what is returned but if the minimum inputs are provided the routine will work just fine.

The fitting process is discussed in more details in the following sections. Additional guidance can be found in the package documentation.

4.1. Specifying the count distribution

The count distribution is selected by specifying the distribution of the inter-arrival times. **Countr** currently provides four built-in distributions: the Weibull, the gamma, the generalized gamma (with Prentice (1974) parameterization), and the Burr type XII distribution. The Weibull-gamma distribution described in McShane *et al.* (2008) has also been implemented but we found the model to be numerically unstable and it should be used with care (see also the discussion in Baker and Kharrat 2016, Section 7.4).

For the `renewal()` function and other functions in the package that provide a choice, the desired inter-arrival distribution is specified by the argument `dist` as a character string, which should have one of the following values: `"weibull"`, `"gamma"`, `"gengamma"` or `"burr"`. Inter-arrival distributions defined by the user are also supported (and specified by `dist = "custom"`), see Section 4.4.

4.2. Specifying covariates

Covariates can be introduced using familiar R formula syntax. We will use the following formula in the examples using the `fertility` dataset:

```
R> regModel <- Y ~ GERMAN + EDU + VOC + UNI + CATH + PROT + MUSL +
+    RURAL + YEAR_OF_B + AGEMARR
```

As usual, the left-hand side of the formula supplied as the argument `formula` in a call to `renewal()` specifies the response variable. The right-hand side gives the covariates for the linear relationship to the (possibly transformed by a link function) corresponding parameter of the count distribution.

A link function (the function $g()$ in equation (4)) can also be specified via the `link` argument as a named list with the same length as the distribution's number of parameters. Each slot is named after the parameter it is linked to and takes the name of the desired link function as its slot value. Possible options for the link function are `"log"`, `"cauchit"`, `"cloglog"`, `"probit"`, `"logit"` and `identity` (default for user defined distributions). For example, for the Weibull distribution, one can choose `log` for the shape and the scale parameters as follows:

```
R> link_weibull <- list(scale = "log", shape = "log")
```

4.3. Fitting built-in models

When fitting built-in models, users are advised to work with the default setting and simply provide the required inputs. For example, the gamma model of Winkelmann (1995) can be fitted as follows:

```
R> gamModel <- renewal(formula = regModel, data = fertility, dist = "gamma",
+    control = renewal.control(trace = 0) )
```

To fit a count distribution without covariates, put 1 in its right-hand side. This fits a Weibull count distribution to `Y`:

```
R> weiCountA <- renewal(formula = Y ~ 1, data = fertility, dist = "weibull",
+    weiMethod = "series_acc", control = renewal.control(trace = 0) )
```

Almost any step of the computation can be customized in `renewal()` and options are provided to give the user control over the computation of the initial values, the numerical optimization algorithm, the method for computing the count probability and the returned values, among others.

User defined initial values

As usual in non-linear optimisation, for best results informed initial values should be provided whenever possible. For the built-in distributions which generalize the Poisson model, one strategy is to fit a Poisson glm model and use its parameter estimates as starting values for the numerical optimizer. This is the strategy adopted by `renewal()` when no initial values are provided.

```
R> IV <- glm(regModel, family = poisson(), data = fertility)
```

The initial values are passed to `renewal()` as a named numeric vector. For this, the names of the coefficients are needed. They have the form `par_covname`, where `covname` is the name of a covariate and `par` is the name of the distribution parameter to which it is linked. Intercepts are named `par_`. The names of the distribution parameters can be found by a call to `getParNames()`.

```
R> pars <- getParNames("weibull")
R> pars
```

```
[1] "scale" "shape"
```

Initial values for the Weibull count model of McShane *et al.* (2008) can be prepared as follows. We rename the coefficients of model `IV` obtained from `glm()` to link them to the first parameter of the Weibull distribution:

```
R> par1Values <- coef(IV)
R> names(par1Values) <- paste(pars[1], names(par1Values), sep = "_")
R> names(par1Values) <- gsub('\\(Intercept\\)', "", names(par1Values))
```

The Poisson model is a particular case of the Weibull model with shape parameter equal to one:

```
R> par2Value <- 1.0
R> names(par2Value) <- paste("shape", "", sep = "_")
R> start <- c(par1Values, par2Value)
```

Finally, the initial values are passed to `renewal()` through the `renewal.control()` routine that will run a sanity check on them before passing them to the optimizer:

```
R> weiModel <- renewal(formula = regModel, data = fertility, dist = "weibull",
+    control = renewal.control(trace = 0, start = start) )
```

Customizing the optimization routine

As mentioned above, `renewal()` maximizes the (log)-likelihood of the desired model by a call to `optimx()` from package **optimx** (Nash and Varadhan 2011). The default is to use `method = "nlminb"` with a maximum of 1000 iterations. Users can change this again through the `renewal.control()` routine. Any other option accepted by `optimx()` can also be passed in `renewal.control()`, e.g.,

```
R> weiModel <- renewal(formula = regModel, data = fertility, dist = "weibull",
+    control = renewal.control(trace = 0, method = "L-BFGS-B") )
```

Probability computation methods

Users can choose the method by which the count probabilities are computed. As discussed in more details in Section 5.1, there are two families of methods. First, there are series expansion methods specific to the Weibull-count models and selected via the `weiMethod` argument and second, there are convolution based methods that work with any inter-arrival distribution. If a series expansion method is selected, users can customize it by appropriately setting argument `seriesPars`. It is a list with components `terms` (the number of terms in the expansion), `iter` (the number of iterations in the accelerated series expansion algorithm) and `eps` (the convergence tolerance).

If a convolution method is to be used, users can customize it by using argument `convPars`. It is a list with optional components for choosing the number of steps in the convolution (`nsteps`), the convolution algorithm (`convMethod`), which can take one of the following values:

"conv_naive", "conv_direct" or "conv_dePril") and specify if Richardson extrapolation should be applied by setting the boolean slot extrap.

As an example we show below how to fit the Weibull model of McShane *et al.* (2008) using a series accelerated method based on the Euler and van-Wijngaarden transformations (Press 2007, Chapter 5) and user defined parameters.

```
R> weiModel <- renewal(formula = regModel, data = fertility, dist = "weibull",
+    weiMethod = "series_acc", control = renewal.control(trace = 0),
+    seriesPars = list(terms = 80, iter = 400, eps = 1e-10) )
```

Controlling the returned values

Following the R convention, users can decide whether the model frame (model), the model matrix (x) and the response (y) are returned in the output of renewal(). In addition, users can decide whether the variance-covariance matrix of the model should be computed by setting the boolean flag computeHessian (defaults to TRUE).

Regression models on the ancillary parameters

So far we have given examples of regression models in which the parameter regressed on was the location parameter, or more precisely, the first parameter of the count distribution. **Countr** offers the possiblity to specify covariates on the "ancillary" parameters (the ones that determine the shape, the variance or other higher moments) via the argument anc. anc is a named list containing formulas describing regression equations for the desired ancilliary parameters. Only the right-hand sides of the formulas are used here.

In the ancillary regression case, starting values have to be provided. We show below an example of fitting such a model for the generalized gamma inter-arrival times with covariates applied to all the distribution parameters. The names of the distribution parameters in this case are "mu" (location), "sigma" and "Q". Given the complexity of the optimization task, we solve the problem in two steps. First, we need to find informative initial values. In order to do that, we use the Poisson trick for the location parameter described earlier and set other regression coefficients to zero (except for the intercept which is set to one) and run a first optimization:

```
R> mu <- coef(IV)
R> names(mu) <- paste0("mu_", names(mu))
R> sigma <- Q <- c(1, rep(0, 10))
R> names(sigma) <- gsub("mu_", "sigma_", names(mu))
R> names(Q) <- gsub("mu_", "Q_", names(mu))
R> start <- c(mu, sigma, Q)
R> names(start) <- gsub('\\(Intercept\\)', "", names(start))
R> anc <- list(sigma = regModel, Q = regModel)
R> gengamModel_ext0 <- renewal(formula = regModel, data = fertility,
+    dist = "gengamma", anc = anc,
+    control = renewal.control(start = start, trace = 0),
+    computeHessian = FALSE)
```

Second, we use the values of the frst optimization as starting values in a second one. The `spg` algorithm was used in this second iteration:

```
R> start <- coef(gengamModel_ext0)
R> gengamModel_ext <- renewal(formula = regModel, data = fertility,
+    dist = "gengamma", anc = anc,
+    control = renewal.control(method = "spg", start = start, trace = 0),
+    computeHessian = FALSE )
```

Users can make sure that the optimization did converge by investigating the convergence status flag in object `gengamModel_ext`:

```
R> gengamModel_ext$converged

[1] TRUE
```

4.4. Custom inter-arrival distributions

Instead of using the built-in distributions in `Countr`, users can also specify their own inter-arrival parametric distributions. For this to work, the following information is required:

- names of the distribution parameters, a character vector.
- survival distribution function, a function with signature `function(tt, distP)`, where `tt` is a vector of class `"numeric"` of non-negative values and `distP` gives the values of the distribution parameters as a named list.
- the name(s) of the link function(s); different link functions may be used for the different parameters. If not specified, `identity` will be used.

The Weibull inter-arrival distribution is one of the built-in distributions but as an illustrative example here is how it could be specified as a custom distribution:

```
R> parNames <- c("scale", "shape")
R> sWei <- function(tt, distP) exp( -distP[["scale"]] * tt ^ distP[["shape"]])
R> link <- list(scale = "log", shape = "log")
```

Here `parNames` defines the names of the parameters, `"sWei"` computes the survival distribution function and `link` requests `log` link for both parameters (a common choice for positive parameters).

Initial values are very desirable for user defined distributions and are computed as discussed in 4.3.1.

```
R> pars <- coef(IV)
R> names(pars) <- gsub('\\(Intercept\\)', "", paste0("scale_", names(pars)) )
R> start <- c(pars, shape_ = 1)
R> control_custom <- renewal.control(start = start, trace = 0)
```

For custom inter-arrival distributions, convolution based methods are the only option. If the user is willing to speed up the computation using a Richardson correction scheme (see Baker and Kharrat 2016, Section 3.5 for more details), the appropriate correction function that computes the correction coefficients must be passed. As argued by Baker and Kharrat (2016, Section 3.5), the appropriate values for the Weibull case are $(2, \alpha)$, where α is the shape parameter. This can be communicated to `renewal()` by defining a function whose argument is a named list of the distribution parameters, as in:

```
R> .getExtrapol <- function(distP) {
+      c(2, distP[["shape"]])
+  }
```

Once all the inputs are ready, `renewal()` can be called in the usual way with argument `dist` set to `"custom"`. The names of the parameters, the survival function and the extrapolation parameters are passed to `renewal()` through argument `customPars`. In our example these are `parNames`, `sWei` and `.getExtrapol`, respectively. This illustrates the syntax for preparing the list:

```
R> customPars <- list(parNames = parNames, survivalFct = sWei,
+    extrapolFct = .getExtrapol)
```

There is also an argument for the links. A model with our custom specified distribution can now be fitted with:

```
R> weiModelCust <- renewal(formula = regModel, data = fertility,
+    dist = "custom", link = link, control = control_custom,
+    customPars = customPars, computeHessian = FALSE)
```

Note that the computations in this example can be much slower than for the equivalent built-in distribution (that is why the hessian computation has been turned off), since the crucial parts of the latter are implemented in C++.

4.5. Working with the fitted models

The function `renewal()` produces an S3 object from class `"renewal"`. Methods for a number of R functions are provided, so that objects from class `"renewal"` can be manipulated and explored in a familiar way. Currently, methods for the following generics are available:

```
R> methods(class = "renewal")

 [1] coef         confint      df.residual  extractAIC   fitted
 [6] logLik       model.matrix nobs         predict      print
[11] residuals    se.coef      summary      vcov
see '?methods' for accessing help and source code
```

Model fit

The usual `summary()` method can be used to check the coefficients' estimates together with their standard errors (computed using numerical estimates of the gradient and Hessian) and Wald test statistics. Here is a summary of Winkelmann's model fitted above:

```
R> summary(gamModel)

Call:
renewal(formula = regModel, data = fertility, dist = "gamma",
    control = renewal.control(trace = 0))

Pearson residuals:
    Min      1Q  Median      3Q     Max
-2.6410 -0.7262 -0.0901  0.4890  6.7411
Inter-arrival dist.: gamma
              Links: rate: link log, shape: link log

Count model coefficients
               Estimate Std. Error z value Pr(>|z|)
rate_           1.55670    0.25234    6.17  6.9e-10 ***
rate_GERMAN    -0.18977    0.05904   -3.21  0.00131 **
rate_EDU        0.03169    0.02650    1.20  0.23185
rate_VOC       -0.14394    0.03584   -4.02  5.9e-05 ***
rate_UNI       -0.14605    0.12961   -1.13  0.25981
rate_CATH       0.20577    0.05780    3.56  0.00037 ***
rate_PROT       0.10714    0.06230    1.72  0.08547 .
rate_MUSL       0.52263    0.06984    7.48  7.3e-14 ***
rate_RURAL      0.05549    0.03119    1.78  0.07519 .
rate_YEAR_OF_B  0.00234    0.00195    1.20  0.22862
rate_AGEMARR   -0.02880    0.00533   -5.41  6.5e-08 ***
shape_          1.43932    0.07106   20.25  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Number of iterations in nlminb optimization: 54

Execution time 64.784
Log-likelihood: -2078.226 on 12 Df
```

The results are exactly the same as the ones in Winkelmann (1995, Table 1)[1]. Similarly, the results for `weiModel` below coincide with those given by McShane *et al.* (2008, Table 2)[2]:

```
R> summary(weiModel)

Call:
renewal(formula = regModel, data = fertility, dist = "weibull",
    control = renewal.control(trace = 0), seriesPars = list(terms = 80,
```

[1]Note that the regression model defined in Winkelmann (1995, Equation (17)) is slightly different and hence the constant value defined in Table 1 is related to our estimated `rate_` parameter by $exp(Constant) * \alpha = exp(scale\_)$.

[2]The value of λ in McShane *et al.* (2008, Table 2) is the exponential of the value of `scale_`. Also note that the standard errors in their table are obtained by the bootstrap procedure with 100 replicates.

```
        iter = 400, eps = 1e-10), weiMethod = "series_acc")

Pearson residuals:
    Min      1Q  Median      3Q     Max
-2.6622 -0.7299 -0.0936  0.4978  6.7416
Inter-arrival dist.: weibull
              Links: scale: link log, shape: link log

Count model coefficients
                Estimate Std. Error z value Pr(>|z|)
scale_           1.39722    0.31484    4.44  9.1e-06 ***
scale_GERMAN    -0.22255    0.07179   -3.10  0.00194 **
scale_EDU        0.03853    0.03268    1.18  0.23841
scale_VOC       -0.17335    0.04395   -3.94  8.0e-05 ***
scale_UNI       -0.18146    0.16029   -1.13  0.25762
scale_CATH       0.24200    0.07016    3.45  0.00056 ***
scale_PROT       0.12314    0.07553    1.63  0.10302
scale_MUSL       0.63876    0.08663    7.37  1.7e-13 ***
scale_RURAL      0.06806    0.03812    1.79  0.07418 .
scale_YEAR_OF_B  0.00230    0.00230    1.00  0.31630
scale_AGEMARR   -0.03403    0.00635   -5.36  8.6e-08 ***
shape_           1.23615    0.03371   36.67  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Number of iterations in nlminb optimization: 48

Execution time 29.284
Log-likelihood: -2077.022 on 12 Df
```

Boostrap standard errors

Bootstrap standard erros and confidence intervals can be computed by setting `type = "boot"` and specifying the desired number of boostrap samples qith argument `R`. Here is an example with $R = 5$ replicates:

```
R> se_boot <- se.coef(object = weiModel, type =  "boot", R = 5)
R> confint_boot <- confint(object = weiModel, type = "boot", R = 5)
```

Comparing fitted models

In the previous section, we fitted three models to the fertility data. One way to select the "best" model is to use information criteria such as AIC or BIC. For completeness, we also include the Poisson model (previously fitted for use as initial values) and add it to the list of models to be compared:

```
R> poissModel <- IV
```

```
R> mat <- cbind(
+    logLik = c(logLik(poissModel),
+               logLik(gamModel),
+               logLik(weiModel),
+               logLik(gengamModel_ext)),
+    nPars = c(length(coef(poissModel)),
+              length(coef(gamModel)),
+              length(coef(weiModel)),
+              length(coef(gengamModel_ext))),
+    AIC = c(AIC(poissModel),
+            AIC(gamModel),
+            AIC(weiModel),
+            AIC(gengamModel_ext)),
+    BIC = c(BIC(poissModel),
+            BIC(gamModel),
+            BIC(weiModel),
+            BIC(gengamModel_ext))
+    )
R> rownames(mat) <- c("Pois", "gam", "wei", "gengam_ext")
R> print(mat)

           logLik nPars  AIC  BIC
Pois        -2102    11 4226 4282
gam         -2078    12 4180 4242
wei         -2077    12 4178 4240
gengam_ext  -2040    33 4145 4314
```

The generalized gamma model has the largest log-likelihood and is favoured by AIC. On the other hand, BIC, which penalizes the number of parameters more strongly than AIC, favors the less parsimonious Weibull model. Note also that for this dataset, the three renewal regression models clearly outperform the Poisson one.

Prediction

Predictions from the fitted model are obtained by calling `predict()`. Two types of prediction are available: predicting a given count (response) value (`type = "prob"`), i.e., the probability of observing a specific value of the count variable (the value of `Y` in our `data.frame`) given the (individual) covariates or predicting the expected value (`type = "response"`).

The procedure is illustrated for the first few individuals in the `fertility` data:

```
R> newData <- head(fertility)

R> predNew.response <- predict(weiModel, newdata = newData, type = "response",
+    se.fit = TRUE)
R> predNew.prob <- predict(weiModel, newdata = newData, type = "prob",
+    se.fit = TRUE)
```

```
R> predtable <- data.frame(newData$Y, predNew.prob$values,
+    predNew.response$values)
R> names(predtable) <- c("Y", "P(Y=y|x)", "E(Y|x)")
R> predtable
```

```
  Y P(Y=y|x) E(Y|x)
1 2 0.284675 2.6349
2 3 0.253154 2.6257
3 2 0.303079 2.3851
4 4 0.095872 2.1319
5 2 0.295155 2.5046
6 2 0.285129 2.6296
```

The covariates are not printed here since they were shown in Section 4.

To conclude this section, one can verify that the results produced by the built-in model and the user defined Weibull model are identical (up to rounding errors):

```
R> cbind(builtIn = coef(weiModel), user = coef(weiModelCust))
```

```
                builtIn    user
scale_           1.3972  1.3973
scale_GERMAN    -0.2226 -0.2226
scale_EDU        0.0385  0.0385
scale_VOC       -0.1733 -0.1734
scale_UNI       -0.1815 -0.1815
scale_CATH       0.2420  0.2420
scale_PROT       0.1231  0.1231
scale_MUSL       0.6388  0.6388
scale_RURAL      0.0681  0.0681
scale_YEAR_OF_B  0.0023  0.0023
scale_AGEMARR   -0.0340 -0.0340
shape_           1.2362  1.2362
```

5. Additional details

5.1. Count probability computation methods

In order to compute the different count probabilities defined in equation (2), two families of algorithms have been implemented in **Countr**:

Taylor series expansion: this is specific to the Weibull renewal process. Following the method of McShane *et al.* (2008), the exponential in the Weibull density can be expanded out and series transformation techniques can be used to speed up convergence. Two algorithms have been implemented: a matrix approach (`series_mat`) using `series_terms` terms and a

series accelerated method based on the Euler and van-Wijngaarden transformations (Press 2007, Chapter 5) controlled by `series_acc_niter` number of iterations and a convergence tolerance `series_acc_eps`.

Convolution methods: as developed and described in Baker and Kharrat (2016), three algorithms are available: the naive convolution that computes all the probabilities up to the desired one (`conv_naive`), the direct convolution that computes a reduced number of convolutions to produce the result (`conv_direct`) and the dePril's convolution method (`conv_dePril`). The number of discretization steps can be controlled by setting `conv_steps`. By default, the built-in distributions apply Richardson correction (`conv_extrap = TRUE`).

5.2. Naming conventions

The names of the more technical functions in the package are somewhat verbose. We use the following conventions. The names of the renewal count models are formed by concatenating the name of the inter-arrival distribution and the word 'Count'. Functions that accept the inter-arrival distribution as a parameter simply contain the word 'Count'. Following the conventions from base R, names of functions that compute densities (actually probabilities, since the distributions are discrete) start with 'd'. Functions with suffix `_bi` (short for 'built-in') do computations for any of the built-in models, the particular one being chosen by argument `dist`. Functions with suffix `_user` accept a user specified distribution for the inter-arrival times. See the documentation of the individual functions for further details.

6. Conclusion and Future Work

Count regression models derived from renewal processes are a flexible class of models that extends the Poisson model and allows the use of inter-arrival times distributions that are more flexible than the exponential. The **Countr** package implements this class of models using standard R framework (very similar to `glm()`) and hence allows users familiar with the generalized linear models to experience a more flexible class of models with minimum additional effort. Usual methods for model fitting, goodness of fit diagnosis and prediction are also provided.

Countr currently contains four built-in distributions for which the computations are optimised by programming crucial parts in C++ and choosing taylored parameters for optimisation functions. Although users can define their own inter-arrival times distribution, this may result in long computation times as demonstrated in Section 4.4. In future versions of the package, a larger choice of survival distributions will be available.

Renewal regression models can be extended in many directions. One of them is to allow the time to the first event to have a different distribution from the inter-arrival times for later events. This gives rise to a type of hurdle model that we call "modified renewal processes". This family of models is being studied by the authors and an experimental version is shipped with **Countr**. Another direction in which the **Countr** package can be extended is by allowing multivariate (and especially bivariate) models to be fitted. A Copula (Cameron and Trivedi 2013, Section 8.5) can be used the take into account dependence between the count marginals. Such models will also be included in future versions of **Countr**.

References

Baker R, Kharrat T (2016). "Event Count Distributions from Renewal Processes: Fast Computation of Probabilities." *Scandinavian Journal of Statistics*, **Under review**.

Bittner E, Nussbaumer A, Janke W, Weigel M (2007). "Self-affirmation model for football goal distributions." *EPL (Europhysics Letters)*, **78**(5), 58002.

Cameron AC, Trivedi PK (2013). *Regression analysis of count data*, volume 53. Cambridge university press.

Chambers J, Hastie T, Pregibon D (1990). "Statistical Models in S." In *Compstat*, pp. 317–321. Springer.

Eddelbuettel D, Sanderson C (2014). "RcppArmadillo: Accelerating R with high-performance C++ linear algebra." *Computational Statistics and Data Analysis*, **71**, 1054–1063. URL http://dx.doi.org/10.1016/j.csda.2013.02.005.

Feller W (1970). *An introduction to probability theory and its applications*, volume 2. John Wiley & Sons.

Jackson C (2016). "flexsurv: A Platform for Parametric Survival Modeling in R." *Journal of Statistical Software*, **70**(8), 1–33. doi:10.18637/jss.v070.i08.

Jose K, Abraham B (2013). "A Counting Process with Gumbel Inter-Arrival Times for Modeling Climate Data." *Journal of Environmental Statistics*, **4**(5). ISSN 1945-1296. URL http://jes.stat.ucla.edu/v04/i05.

Jose KK, Abraham B (2011). "A Count Model Based on Mittag-Leffler Inter-Arrival Times." *Statistica*, **71**(4), 501–514.

Kharrat T (2016). *A Journey Across Football Modelling with Application to Algorithmic Trading (PhD thesis)*. University of Manchester.

Kharrat T, Boshnakov GN (2016). *Countr: Flexible Univariate and Bivariate Count Process Probability*. R package version 3.2.4.

McCullagh P, Nelder JA (1989). *Generalized linear models*, volume 37. CRC press.

McShane B, Adrian M, Bradlow ET, Fader PS (2008). "Count Models Based on Weibull Inter-Arrival Times." *Journal of Business & Economic Statistics*, **26**(3).

Nash JC, Varadhan R (2011). "Unifying Optimization Algorithms to Aid Software System Users: optimx for R." *Journal of Statistical Software*, **43**(9), 1–14. URL http://www.jstatsoft.org/v43/i09/.

Prentice RL (1974). "A log gamma model and its maximum likelihood estimation." *Biometrika*, **61**(3), 539–544.

Press WH (2007). *Numerical recipes 3rd edition: The art of scientific computing*. Cambridge university press.

Winkelmann R (1995). "Duration Dependence and Dispersion in Count-Data Models." *Journal of Business & Economic Statistics*, **13**(4), 467–474.

Winkelmann R (2013). *Econometric analysis of count data*. Springer Science & Business Media.

Zeileis A, Kleiber C, Jackman S (2008). "Regression Models for Count Data in R." *Journal of statistical software*, **27**(8), 1–25.

Affiliation:

Tarak Kharrat
Salford Business School
Maxwell Building, Salford
M5 4WT Greater Manchester, United Kingdom E-mail: T.Kharrat@salford.ac.uk

Georgi N. Boshnakov
School of Mathematics
The University of Manchester
Oxford Road, Manchester M13 9PL, UK
URL: http://www.maths.manchester.ac.uk/~gb/

Journal of Statistical Software http://www.jstatsoft.org/
published by the American Statistical Association http://www.amstat.org/
Volume VV, Issue II *Submitted:* yyyy-mm-dd
MMMMMM YYYY *Accepted:* yyyy-mm-dd