CountTimeSeries.jl Documentation - Modelling Univariate Count Data Time Series in Julia

Manuel Stapper
Westfälische Wilhelms-Universität Münster,
Department of Economics (CQE), Germany

Version: May, 3rd 2021

Abstract: A Julia package is developed to deal with univariate count data time series. For a general class of integer ARMA and GARCH type models, the package allows for simulation, likelihood based estimation and inference. Nested are also regressions with integer response variable. Information criteria and the non-randomized PIT histogram may be used to assess the model choice. Further, h-step ahead predictions can be carried out including prediction intervals from a parametric bootstrap.

The package is currently available in the GitHub repository `https://github.com/ManuelStapper/CountTimeSeries.jl`.

Keywords: Julia, Count Data, Time Series Analysis, Count Regression

1 Introduction

The Julia package presented in the following was developed for the analysis of count time series. It covers a broad class of integer-valued GARCH and ARMA like models. Time series following these models can be generated and parameters are estimated using numerical maximization of the likelihood. Confidence intervals are computed via standard maximum likelihood procedures. Information criteria and the non-randomized PIT histogram provide model diagnostic devices. Forecasts can then be used including simulation based forecast intervals.

The structure of the CountTimeSeries package is kept modular to allow for easy extension of alternative estimation methods of model frameworks. In the following, the INGARCH and INARMA framework is introduced first together with the notation used throughout this paper and in the package. Then, the functions contained in the package are introduced. Their in- and outputs are summarized and details are provided. This manual concludes with an outlook of possible future extensions.

2 Framework

The two model frameworks incorporated in the CountTimeSeries package shall first be presented from a theoretical point of view in the following section.

2.1 INGARCH(p, q)

Developed by Ferland et al. (2006), the INGARCH framework builds a cornerstone in modeling count data time series. Starting with their basic framework, different generalizations have been developed thereafter. Some important extensions are included in the CountTimeSeries package. Models without regressors can be formalized as

$$\begin{aligned} Y_t|\mathcal{F}_{t-1} &\sim \begin{cases} \mathcal{D}(\lambda_t, \theta) & \text{with probability } (1-\omega) \\ \delta_0 & \text{with probability } \omega \end{cases} \\ f^{-1}(\lambda_t) &= \beta_0 + \sum_{i=1}^{p} \alpha_i Y_{t-i} + \sum_{i=1}^{q} \beta_i \lambda_{t-i} \ , \end{aligned} \tag{1}$$

where Y_t is observable process, $\mathcal{F}_t$ the information set at time t and $\mathcal{D}(\lambda_t, \theta)$ a distribution with support $\mathbb{N}_0$, mean parameter λ_t and possibly further parametrized by θ. Possible candidates for $\mathcal{D}$ are Poisson and Negative Binomial distribution. Zero inflation is incorporated whenever $\omega > 0$. Then $Y_t|\mathcal{F}_{t-1}$ follows a Dirac distribution with probability ω and distributed according to $\mathcal{D}(\lambda_t, \theta)$ with probability $1 - \omega$. The link function f is restricted to either the identity function or the exponential function during the following.

In many applications, the observed count process is influenced by regressors and not purely self driven. To include regressors in above models, the second row in (1) is altered to

$$\begin{aligned} \lambda_t &= \nu_t + \sum_{i=1}^{r_E} \eta_i X_{i,t} \\ \nu_t &= \beta_0 + \sum_{i=1}^{p} \alpha_i Y_{t-i} + \sum_{i=1}^{q} \beta_i \nu_{t-i} + \sum_{i=r_E+1}^{r} \eta_i X_{i,t} \end{aligned}$$

for an identity link and

$$\begin{aligned} \log(\lambda_t) &= \nu_t + \sum_{i=1}^{r_E} \eta_i X_{i,t} \\ \nu_t &= \beta_0 + \sum_{i=1}^{p} \alpha_i \log(Y_{t-i}+1) + \sum_{i=1}^{q} \beta_i \nu_{t-i} + \sum_{i=r_E+1}^{r} \eta_i X_{i,t} \end{aligned}$$

for a log-linear link. Thereby, r denotes the number of regressors, where the first r_E regressors enter the system externally and the remaining r_I regressors enter it internally, affecting not only λ_t but also ν_t. A Poisson or NB regression is nested in this framework and can be specified setting $p = q = 0$.

Given a vector of parameters θ, the likelihood is easily computed as

$$\begin{aligned} L(\theta) &= \prod_{t=M+1}^{T} P(Y_t = y_t | \mathcal{F}_{t-1}) \\ &= \prod_{t=M+1}^{T} P_{\mathcal{D}}(Y_t = y_t | \mathcal{F}_{t-1})(1-\omega) + \mathbb{1}\{y_t = 0\}\omega \ . \end{aligned}$$

2.2 INARMA(p, q)

Besides the INGARCH framework, count data time series are often modelled in an ARMA like structure. Different approaches have been developed starting with McKenzie (1985) and Alzaid and Al-Osh (1988). In the scope of this package, we focus on the thinning based INARMA framework discussed by Weiß et al. (2019), who introduced an efficient Maximum Likelihood based estimation. This approach is generalized to a framework similar to the one presented in the previous subsection, . An INARMA(p, q) model with possible zero inflation and deterministic regressors can be formalized as

$$\begin{aligned} Y_t &= R_t + \sum_{i=1}^{q} \beta_i \circ R_{t-i} + \sum_{i=1}^{p} \alpha_i \circ Y_{t-i} + Z_t \\ R_t &\sim \begin{cases} \mathcal{D}_1(\lambda_t, \theta) & \text{with probability } (1-\omega) \\ \delta_0 & \text{with probability } \omega \end{cases} \\ f_1^{-1}(\lambda_t) &= \beta_0 + \sum_{i=1}^{r_I} \eta_i X_{i,t} \\ Z_t &\sim \mathcal{D}_2(\mu_t, \theta) \\ f_2^{-1}(\mu_t) &= \sum_{i=r_I+1}^{r} \eta_i X_{i,t} \end{aligned} \tag{2}$$

The thinning operator $\circ$ is defined for $p \in [0,1]$ and $X \in \mathbb{N}_0$ as $p \circ X = \sum_{i=1}^{X} B_i$, where $B_i \overset{\text{iid}}{\sim} \text{Bin}(1,p)$ for strictly positive X and zero if $X = 0$. Thinning operators in above framework are assumed mutually independent. Both distributions $\mathcal{D}_1$ and $\mathcal{D}_2$ have support $\mathbb{N}_0$, means λ_t and μ_t respectively and might be further parametrized by θ. Possible distributions are limited to Poisson and Negative Binomial. Both link functions f_1 and f_2 are either the identity or the exponential function. Zero inflation can be incorporated in the distribution of R_t and translates to an inflation of zeros in Y_t. Regressors can either enter the system externally by affecting the mean of Z_t or enter it internally affecting the mean of R_t.

The likelihood is not easily computed for the INARMA framework, since the process $\{R_t\}$ is unobservable. Every possible path of $\{R_t\}$ is considered during the evaluation of the likelihood. The computation time increases drastically with a higher MA order q. From the first line of (2) it is obvious that $R_t \leq Y_t$. Therefore, the possible values of R_t given $Y_t = y_t$ are limited. This property can be exploited when computing the likelihood.

For a general q, the likelihood can be computed using

$$b_t(r_t, r_{t-1}, ..., r_{t-q}) := \mathrm{P}(R_t = r_t, ..., R_{t-q} = r_{t-q}; Y_t = y_t, ..., Y_{M+1} = y_{M+1} | Y_M = y_M, ..., Y_1 = y_1) \ ,$$

where $M := \max\{p, q\}$, which gives

$$L(\theta) = \sum_{r_{T-q}=0}^{y_{T-q}} \cdots \sum_{r_T=0}^{y_T} b_T(r_T, ..., r_{T-q}) \ .$$

The arrays $b_t(r_t, ..., r_{t-q})$ are initialized by

$$\begin{aligned} b_{M+1}(r_{M+1}, ..., r_{M-q+1}) =& \mathrm{P}(R_{M+1} = r_{M+1}) \prod_{i=M-q+1}^{M} \mathrm{P}(R_i = r_i | R_i \leq y_i) \cdot \\ & \mathrm{P}\left(\sum_{i=1}^{p} \alpha_i \circ y_{M+1-i} + \sum_{i=1}^{q} \beta_i \circ r_{M+1-i} + Z_{M+1} = y_{M+1} - r_{M+1} \right) \ . \end{aligned}$$

Then for $t = M+2, ..., T$ the following recursion holds

$$\begin{aligned} b_t(r_t, ..., r_{t-q}) =& \mathrm{P}(R_t = r_t) \left(\sum_{r_{t-q-1}=0}^{y_{t-q-1}} b_{t-1}(r_{t-1}, ..., r_{t-q-1}) \right) \cdot \\ & \mathrm{P}\left(\sum_{i=1}^{p} \alpha_i \circ y_{t-i} + \sum_{i=1}^{q} \beta_i \circ r_{t-i} + Z_t = y_t - r_t \right) \ . \end{aligned}$$

Weiß et al. (2019) describe an efficient likelihood evaluation technique for $q = 1$, which translates above computation to a matrix product recursion. The idea is extended to $q = 2$, likelihood evaluation of models with higher order are computationally demanding and not feasible for practical purposes, see Dungey et al. (2019).

3 Functions and Structures

The following section describes functions and structures included in the CountTimeSeries package. Functions are summarized in a table, in- and output are presented together with the data type accepted and a short description. Structures are self defined data types. They can be used to standardize in- or output of functions. They are described here by a table with entries of the structure, their data type and a short description.

Further it should be noted that different packages are used inside the CountTimeSeries package, i.e. Optim, Distributions, LinearAlgebra, Random, Plots, Calculus and Roots.

3.1 Model Specification and Parameters

As a first step, different types of count data models are defined. The type `CountModel` defines the broad class containing all models described in Sections 2.1 and 2.2. Subtypes of this class are defined for the two frameworks, called `INGARCH` and `INARMA`. These three types are defined as abstract types, actual structs give information about model specifications and are defined as subtypes of `INGARCH` and `INARMA`. Namely, there are three subtypes of `INGARCH`, `INGARCHModel`, `INARCHModel` and `IIDModel`. The former contains the following specifications

Table 1: `INGARCHModel` - Model Specification

Variable	Type	Description
`distr`	`String`	Conditional distribution
`link`	`String`	Link function
`pastObs`	`Array{Integer, 1}`	Past observations included
`pastMean`	`Array{Integer, 1}`	Past means included
`X`	`Array{Abstract Float, 2}`	Regressor matrix
`external`	`Array{Bool, 1}`	Are regressors external?
`zi`	`Bool`	Zero inflation

The distribution `distr` can either be `"Poisson"` or `"NegativeBinomial"`. The link function `link` can either be `"Linear"` or `"Log"`. The entries `pastObs` and `pastMean` specify which lags are included in the definition of the conditional mean. Thereby it is possible to include no lags (`[]`) or non-consecutive lags (`[1, 12, 24]`). In case of regressors, the vector `external` must clarify which of those enter the system externally, having the same length as there are regressors. The regressor matrix `X` shall include the regressors row-wise. Finally, the boolean variable `zi` indicates if zero inflation is incorporated.

In case of no recursion in the conditional mean ($q = 0$), an INGARCH model reduces to an INARCH model. A specification object for INARCH(p) models is defined as `INARCHModel`, which includes all entries as in 1 besides `pastMean`. Further a model with no serial correlation is defined as `IIDModel`, with no entries `pastObs` and `pastMean`. Although this class is called IIDModel, observations might not be identically distributed in case of regressors.

In the same fashion as for INGARCH models, different thinning based models can be specified by `INARMAModel`, `INARModel` and `INMAModel`. The former contains the following entries:

Table 2: `INARMAModel` - Model Specification

Variable	Type	Description
`distr`	`Array{String, 1}`	Conditional distributions
`link`	`Array{String, 1}`	Link functions
`pastObs`	`Array{Integer, 1}`	Past observations included
`pastMean`	`Array{Integer, 1}`	Past means included
`X`	`Array{Abstract Float, 2}`	Regressor matrix
`external`	`Array{Bool, 1}`	Are regressors external?
`zi`	`Bool`	Zero inflation

In contrast to the INGARCH models, two distributions are specified for $\mathcal{D}_1$ and $\mathcal{D}_2$ as well as two link functions f_1 and f_2 in definition 2. The specification `INARModel` has no entry `pastMean` and `INMAModel` has no entry `pastObs`.

Implementing model types and subtypes like that allows to define functions based on the model and use multiple dispatch. For example, likelihood evaluation of an INGARCH(1, 1) model requires a loop, while it does not for INARCH models. A likelihood function can be implemented for the general class of INGARCH models and a more specific version, potentially more efficient, for INARCH models.

To create an object that specifies a model, the user can either use constructors, for example calling `INGARCHModel` as function and giving the entries as input or use the wrapper function `Model`. The advantage of a wrapper function is that it can be implemented for any type of input and default values can be given. Table 3 summarizes the input of `Model()` and its default values. If no input is given, the function returns a model specification for IID Poisson distributed variables, no regressors and no zero inflation. The argument `model` gives the general framework, either `"INGARCH"` or `"INARMA"`, `distr` gives the conditional distribution(s), either `"Poisson"` or `"NegativeBinomial"`. Thereby, it can be either the string itself of a vector of strings for example in case of an INARMA model with regressors. The same holds for the link function(s), which can be either `"Linear"` or `"Log"`. As input for this wrapper function, the regressors can be a vector if there is only one regressor, or - in case of multiple regressors, as matrix where it does not matter if regressors are collected column-wise or row-wise. The function aims at correcting the input as good as possible, for example specifying which past observations are used in the conditional mean of an INGARCH needs to be given in the INGARCHModel struct as vector of integers. The wrapper function also accepts scalars or ranges given by `1:3` or such. The output of `Model()` is an object of type `INGARCHModel`, `INARCHModel`, `IIDModel`, `INARMAModel`, `INARModel` or `INMAModel`. Thereby, `Model(pastObs = 1)` returns an object of type `INARCHModel` rather than of type `INGARCHModel` with $q = 0$.

Table 3: `Model` - Wrapper function to create count model specification

Input	Type	Default	Description
`model`	undefined	`"INGARCH"`	Framework
`distr`	undefined	`"Poisson"`	Conditional distribution(s).
`link`	undefined	`"Linear"`	Link function
`pastObs`	undefined	`[]`	Past observations included
`pastMean`	undefined	`[]`	Past means included
`X`	undefined	`[]`	Regressor matrix
`external`	undefined	`[]`	Are regressors external?
`zi`	undefined	`false`	Zero inflation
Output			
unnamed	`<:CountModel`		Model specification

To parametrize a model, another structure is defined, which is summarized in Table 4. This structure is an alternative to merging all parameters to one vector and run into danger of loosing track of the mapping. However, it might be handy to work with parameters as a vector, for example during maximization of the likelihood. For that reason, two functions are introduced that switch between the two ways of parametrization. These functions are described in Tables 5 and 6.

Table 4: `parameter` - Parameters of a Count Model

Variable	Type	Description
β_0	`AbstractFloat`	Intercept
α	`Array{AbstractFloat, 1}`	Coefficients of past observations
β	`Array{AbstractFloat, 1}`	Coefficients of past means
η	`Array{AbstractFloat, 1}`	Coefficients of regressors
ϕ	`Array{AbstractFloat, 1}`	Overdispersion parameters
ω	`AbstractFloat`	Zero inflation probability

Table 5: `θ2par` - Mapping between parameter vector and struct

Input	Type	Description
`θ`	`Array{AbstactFloat, 1}`	Parameter vector
`model`	`CountModel`	Model specifications
Output		
pars	`parameter`	Parameters (struct)

Table 6: `par2θ` - Mapping between parameter struct and vector

Input	Type	Description
pars	`parameter`	Parameters (struct)
`model`	`CountModel`	Model specifications
Output		
`θ`	`Array{AbstractFloat, 1}`	Parameter vector

When working with certain models, there might be parameter restrictions to ensure stationarity and positivity of conditional means as well as overdispersion parameters. In any case, $\phi > 0$ and $\omega \in [0,1]$ must hold. For an INGARCH with linear link, the restrictions are $\beta_0 > 0$, $\alpha_i \geq 0$, $\beta_i \geq 0$, $\eta_i \geq 0$, and $\sum_{i=1}^{p} \alpha_i + \sum_{i=1}^{q} \beta_i < 1$. For an INGARCH with log-linear link, the restrictions are $|\alpha_i| < 1$, $|\beta_i| < 1$ and $|\sum_{i=1}^{p} \alpha_i + \sum_{i=1}^{q} \beta_i| < 1$.
For the INARMA model, $\alpha_i \geq 0$ and $\beta_i \geq 0$ must always hold and additionally, $\sum_{i=1}^{p} \alpha_i \leq 1$. The coefficients of regressors must be non-negative if the corresponding link function is the identity link.

To check if parameters are valid for a given model, a function `parametercheck` is introduced, see Table 7. Different methods are implemented for the function depending on the model and the type of parameter input (parameter struct or vector). Multiple dispatch is utilized, so a wrapper function is not needed.

Table 7: `parametercheck` - Check for validity of parameters

Input	Type	Description
`θ`	`parameter` or `Array{AbstractFloat, 1}`	Parameters
`model`	`CountModel`	Model specifications
Output		
unnamed	`Bool`	Are parameters valid?

3.2 Simulate Time Series

Before introducing a function to create artificial time series from above frameworks, the thinning operator $\circ$ is implemented in Julia in different ways. First, the binomial thinning is defined as described before. In Julia the expression `p∘X` for a given $\texttt{p} \in [0, 1]$ and integer `X` performs the binomial thinning. An extension using a common random number `u` is implemented as `∘(p, X, u)`. Alternatively, thinning may be defined for a distribution `d` and an integer `X` as

$$\texttt{d} \circ \texttt{X} = \sum_{i=1}^{\texttt{X}} Z_i \qquad Z_i \overset{\text{iid}}{\sim} \texttt{d} .$$

Again, a version using a common random number `u` is added as `∘(d, X, u)`. A last generalization for any distribution `dI` and an integer distribution `dO` is introduced such that `dI ∘ dO` computes $\sum_{i=1}^{X} Z_i$ with X distributed according to `dO` and Z_i are iid distributed according to `dI`.

Next, the function `simulate` is described, summarized in Table 8. It contains different methods separated by model type and type of parameters (given as vector or struct). Only valid parameters are accepted. The conditional mean process of an INGARCH process is initialized by its marginal mean, the first observations of an INARMA process are simulated without autocorrelation. Then `T + burnin` observations are generated and the first `burnin` observations `discarded`. Alternatively, the user can set the first observations with the argument `pinfirst`. This can for example be helpful when simulating time series with regressors. Ignoring regressors during the burnin phase may lead to a time series that levelled off incorrectly. If `pinfirst` is not specified and a time series with regressors shall be created, the default setting is to use the regressors for $t = 1$ during the burnin.

Table 8: `simulate` - Simulate Time Series

Input	Type	Description
`T`	`Integer`	Length of time series
`model`	`CountModel`	Model specification
θ	`parameter` or `Array{AbstractFloat, 1}`	Parameters
`burnin`	`Integer`	Burnin phase (default 500)
`pinfirst`	`Array{Integer, 1}`	Pin first values of time series
Output		
`y`	`Array{Integer, 1}`	Simulated time series
λ	`Array{AbstractFloat, 1}`	Conditional means (INGARCH)
ν	`Array{AbstractFloat, 1}`	See model definition (INGARCH)
$\lambda_{\texttt{zi}}$	`Array{AbstractFloat, 1}}`	Conditional means with ZI (INGARCH)
`R`	`Array{Int64, 1}`	Innovation process (INARMA)

3.3 Likelihood Computation

Estimation and inference in the scope are - up to now - likelihood based exclusively. To prepare a likelihood function, some functions are defined first. For the INGARCH framework, one internal function called `LinPred` is introduced. Its use is to compute the linear predictors λ_t and ν_t, given a model and parameters. Different methods are implemented depending on the model. As it is only used internally, parameters are only accepted as structure. Due to repeated calls during likelihood maximization, the focus for this function is on its efficiency. An argument `initiate` specifies how the first linear predictors are computed. `"first"` sets it to the first observation of the time series, `"intercept"` sets the first values to the expectation if $\alpha_i = 0$ and $\beta_i = 0$, whereas `"marginal"` sets the first values to the marginal mean.

The thinning based INARMA process defined the observable process Y_t as a sum of different components. Thus, for likelihood computation given the distributions of the single components, a convolution function is needed. For two discrete random variables, say X_1 and X_2, the function `convolution` computes the distribution of $X_1 + X_2$. As input it needs a vector of probabilities $(P(X_1 = 0), P(X_1 = 1), ..., P(X_1 = k))'$ and $(P(X_2 = 0), P(X_2 = 1), ..., P(X_2 = k))'$ and gives out $P(X_1 + X_2 = i)$ for $i = 1, ..., k$. Another internally used function called `GetProbFromP` is defined. It takes a matrix $\texttt{P} \in \mathbb{R}^{M+1,K}$ of probabilities $\texttt{P}_{ij} = P(X_j = i)$ as input and returns the probability vector $P(X_1 + ... + X_K = i)$ with $i = 0, ..., M$. Both convolution functions are implemented with focus on efficiency.

The log-likelihood function `ll`, see Table 9, is then defined. For any model besides INARMA(p, q) where $q \geq 3$, it computes the log-likelihood as well as the contributions of single observations to the log-likelihood. This enables to compute for example information criteria on the basis of the same number of observations in case of different model orders, dropping the same first observations.

Table 9: `ll` - Computation of Log-Likelihood

Input	Type	Description
`y`	`Array{AbstractFloat, 1}`	Time series
`model`	`CountModel`	Model specification
`θ`	`parameter`	Parameters
`initiate`	`String`	Initialization method (INGARCH)
Output		
LL	`Float64`	Log-Likelihood
LLs	`Array{Float64, 1}`	Contributions to log-likelihood

3.4 Estimation Settings

To prepare the estimation function, two structures are defined, which unify the estimation settings and the reporting of results. Estimation settings are defined as structure `MLEControl`, see Table 10. It collects information about initial values for the maximization, the maximization procedure and an the choice whether confidence intervals shall be computed or not. Optimizing routines are - up to now - limited to `"NelderMean"`, `"BFGS"` and `"LBFGS"`. Additionally a wrapper function `MLESettings`, see Table 11, makes specification easier. The initial values can either be given by the user in form of a vector or parameter struct. As a default if not put in, the initial values are chosen in a way that matches theoretical marginal mean and sample mean of the time series. If the given or default initial values result in an infinite likelihood, a random search for starting values is carried out.

Table 10: `MLEControl` - Estimation Settings

Variable	Type	Description
`init`	`parameter`	Initial values
`optimizer`	`String`	Maximization method
`ci`	`Bool`	Shall confidence intervals be computed
`maxEval`	`Integer`	Maximum number of likelihood evaluations

Table 11: `MLESettings` - Create Object for Estimation Settings

Input	Type	Description
`y`	`Array{AbstractFloat, 1}`	Time series
`model`	`CountModel`	Model specification
`init`	`parameter` or `Array{AbstractFloat, 1}`	Initial parameters
`optimizer`	`String`	Optimizing routine
`ci`	`Bool`	Compute confidence intervals?
Output		
unnamed	`MLEControl`	Estimation Settings

If the distribution or link function is misspelled or simply unknown, it gives out a warning that the default is used. If the string specifying an optimization function is misspelled or unknown, the BFGS method is used as default and a warning is printed again. It returns an error if one of the following occurs: The size of `external` does not match the number of rows in `X`. The length of `y` does not match the number of columns in `X`. The parameters given in `init` does not match the number of parameters for the specified model or the initial parameters `init` are not valid.

Two unified structures of estimation results is defined next. They shall contain every information needed to assess model choice and perform predictions afterwards. The two structures `INGARCHResults` and `INARMAResults` hold the entries given in Tables 12 and 13

Table 12: `INGARCHResults` - Result Specification

Variable	Type	Description
`y`	`Array{Integer, 1}`	Time series
θ	`Array{AbstractFloat, 1}`	Estimated parameters (vector)
`pars`	`parameter`	Estimated parameters (struct)
λ	`Array{AbstractFloat, 1}`	Conditional means
`residuals`	`Array{AbstractFloat, 1}`	Residuals
`LL`	`AbstractFloat`	Maximum log-likelihood
`LLs`	`Array{AbstractFloat, 1}`	Log-likelihood contributions
`nPar`	`Integer`	Number of parameters
`nObs`	`Integer`	Number of observations
`se`	`Array{AbstractFloat, 1}`	Standard errors
`CI`	`Array{AbstractFloat, 2}`	Confidence intervals
`model`	`INGARCH`	Model specifications
`converged`	`Bool`	Has maximization converged?
`MLEControl`	`MLEControl`	Estimation settings

Table 13: `INARMAResults` - Result Specification

Variable	Type	Description
`y`	`Array{Integer, 1}`	Time series
θ	`Array{AbstractFloat, 1}`	Estimated parameters (vector)
`pars`	`parameter`	Estimated parameters (struct)
`LL`	`AbstractFloat`	Maximum log-likelihood
`LLs`	`Array{AbstractFloat, 1}`	Log-likelihood contributions
`nPar`	`Integer`	Number of parameters
`nObs`	`Integer`	Number of observations
`se`	`Array{AbstractFloat, 1}`	Standard errors
`CI`	`Array{AbstractFloat, 2}`	Confidence intervals
`model`	`INGARCH`	Model specifications
`converged`	`Bool`	Has maximization converged?
`MLEControl`	`MLEControl`	Estimation settings

3.5 Estimation

The function to estimate parameters of a given model and conduct inference, `fit`, is summarized in Table 14. The maximization of the log-likelihood is done numerically using the `Optim` package. Restriction of parameter space are taken into account by returning negative infinity if parameter input is invalid. In case of an INGARCH process, the user might use the Quasi-Poisson estimation introduced by Christou and Fokianos (2014). Therefore, a Poisson estimation must be computed first and the results forwarded to the function `QPois`, which takes only the estimation results as input and changes them. The overdispersion parameter is then estimated by solving

$$\sum_{t=\max\{P,Q\}+1}^{T} \frac{(y_t - \hat{\lambda}_t)^2}{\hat{\lambda}_t - \frac{\hat{\lambda}_t^2}{\phi}} = T - P - Q - r - 1$$

for ϕ. Thereby $\hat{\lambda}_t$ is the conditional mean at time t using the estimated parameters, P and Q are the highest lags considered for past observations and past means respectively. In case of non-consecutive lags, these differ from p and q.

In case confidence intervals shall be computed, the `Calculus` package provides methods to approximate the Hessian numerically giving standard errors. For user convenience, a summary of estimation results is optionally and by default printed to the console. In similar fashion as in R asterisks printed represent a level of significance from zero, one asterisk for a p-value below 5%, two for a p-value below 1% and three if the p-value is below 0.1%. For a linear link, these indicators are hardly meaningful, because testing for example $\alpha_1 > 0$ implies that the parameter is on the border of admissible parameter space under the null hypothesis, and the method of inference is invalid.

Table 14: `fit` - Estimation Function

Input	Type	Description
`y`	`Array{Integer, 1}`	Time series
`model`	`CountModel`	Model specification
`MLEControl`	`MLEControl`	Estimation settings (optional)
`printResults`	`Bool`	Print results? (optional)
`initiate`	`String`	Initialization method (optional)
Output		
`results`	`INGARCHResults` or `INARMRAResults`	Results

3.6 Model Diagnostics

To check for model accuracy, two different tools were implemented in CountTimeSeries. First, different information criteria are provided, namely AIC, BIC and HQIC, which all need only the results from estimation as necessary input and optional the integer argument `dropfirst`, see Table 15. This allows to compare information criteria for models of different order.

Table 15: `AIC`, `BIC`, `HQIC` - Information Criteria

Input	Type	Description
`results`	`INGARCHResults` or `INARMAResults`	Results of estimation
`dropfirst`	`Integer`	Number of observations to be ignored for log-likelihood
Output		
`ic`	`AbstractFloat`	AIC, BIC or HQIC

Another way to asses model choice is the non-randomized PIT histogram introduced by Czado et al. (2009). Conditional distributions and the observed outcomes are compared and transformed giving a histogram which is uniform distributed if the model choice is correct.

Let $P_t(y) = \mathrm{P}(Y_t = y|\mathcal{F}_{t-1})$ denote the conditional distribution of Y_t given the processes past. The PIT function $F_t : [0,1] \to \mathbb{R}^+$ is defined as

$$F_t(u|y) = \begin{cases} 0 & u \leq P_t(y-1) \\ \frac{u-P_t(y-1)}{P_t(y)-P_t(y-1)} & \text{if} \quad P_t(y-1) < u < P_t(y) \\ 1 & u \geq P_t(u) \end{cases}$$

and the mean PIT function as

$$\bar{F}(u) = \frac{1}{T}\sum_{t=1}^{T} F_t(u|y_t)$$

for $u \in [0,1]$. In a last step, heights of bins f_h with $h = 1, ..., H$ in the PIT histogram is calculated as

$$f_h = \bar{F}\left(\frac{h}{H}\right) - \bar{F}\left(\frac{h-1}{H}\right)$$

Because of the cumbersome computation of the distribution of Y_t given the processes past for an INARMA with $q > 0$, the pit histogram only supports INAR models up to now. In the CountTimeSeries package, a function `pit` is provided, where the users can input the results of an estimation conveniently, choose the number of bins to be plotted (with ten as default) and a level of significance to test the uniform distribution. For `level = 0`, no confidence region of bins is plotted, to obtain the 95% confidence region, `level` needs to be set to 0.95.

Table 16: `pit` - Non-Randomized PIT histogram

Input	Type	Description
`results`	`INGARCHResults` or `INARMAResults`	Results of estimation
`nbins`	`Integer`	Number of bins
`level`	`AbstractFloat` $\in [0,1)$	Level of significance
Output		
unnamed	`Array{AbstractFloat, 1}`	height of bins

3.7 Prediction

After choosing a suitable model and fitting it to an observed time series, the user might be interested in predicting the future course of the time series. Forecasting can be performed deterministically in case of an INGARCH framework. For a one-step ahead forecast given the time series up to $t = T$, $\hat{Y}_{T+1|T}$, the conditional mean of Y_{T+1} is used. To forecast further, unobserved $Y_{T+1}, Y_{T+2}, ...$ are replaced by their corresponding prediction. For INARMA models and if prediction intervals shall be computed, predictions can be found by a parametric bootstrap. The user may specify the number of time series being simulated by the argument `nChains`. The mean across all chains is used as point prediction, quantiles approximate 95% prediction intervals. When regressors are included in the model specification, their values for observations to be predicted must be provided. The gray rows in Table 17 indicate which arguments only need to be given for a simulation based prediction and also what output is given in that case.

Table 17: `predict` - Prediction Function

Input	Type	Description
`results`	`Rspec`	Results from estimation
`h`	`Mspec`	Model specification
`nChain`	`Int64`	Prediction horizon
`Xnew`	`Array{Float64, 2}`	New regressors
Output		
pred	`Array{Float64, 1}`	Predicted values
`Qmat`	`Array{Float64, 2}`	95% Prediction intervals
`predMat`	`Array{Float64, 2}`	Matrix of prediction chains

4 Outlook

The Julia package presented here is designed to cover a broad collection of count data time series models that are frequently used in practice. Fast computation time and a suitable infrastructure make simulation studies possible. The modular skeleton on the package allows for easy extension. Planned, but not yet realized are the following:
Theoretical moments for different model specifications serve two purposes. Firstly, they can be used to compare theoretical moments for a fitted model with their sample counterparts. Secondly, once such functions are implemented, a GMM estimation can be added as an alternative to likelihood based methods.
Conceivable is also to add the option of Bayesian estimation to the package. To incorporate the infrastructure of a Metropolis-Hastings or Gibbs-Sampling, only one wrapper function would need to be extended.
In a similar way to existing R packages, an outlier detection function could further be added. Closely related to it and a possible extension are robust estimation techniques. Making multivariate count data time series modeling possible would also be an interesting add-on to the package.

References

[Alzaid and Al-Osh 1988] Alzaid, A.A. ; Al-Osh, M.: First-Order Integer-Valued Autoregressive (INAR(1)) Process: Distributional and Regression Properties. In: *Statistica Neerlandica* 41 (1988), Nr. 1, 53–60. http://www.jstor.org/stable/3214650. – ISSN 00219002

[Christou and Fokianos 2014] Christou, V. ; Fokianos, K.: Quasi-Likelihood Inference for Negative Binomial Time Series Models. In: *Journal of Time Series Analysis* 35 (2014), Nr. 1, 55-78. http://dx.doi.org/10.1111/jtsa.12050. – DOI 10.1111/jtsa.12050

[Czado et al. 2009] Czado, C. ; Gneiting, T. ; Held, L.: Predictive Model Assessment for Count Data. In: *Biometrics* 65 (2009), Dezember, Nr. 4, 1254-1261. http://dx.doi.org/10.1111/j.1541-0420.2009.01191.x. – DOI 10.1111/j.1541–0420.2009.01191.x

[Dungey et al. 2019] Dungey, M. ; Martin, V.L. ; Tang, C. ; Tremayne, A.: A threshold mixed count time series model: estimation and application. In: *Studies in Nonlinear Dynamics & Econometrics* 24 (2019), Nr. 2. http://dx.doi.org/10.1515/snde-2018-0029. – DOI 10.1515/snde–2018–0029

[Ferland et al. 2006] Ferland, R. ; Latour, A. ; Oraichi, D.: Integer-Valued GARCH Process. In: *Journal of Time Series Analysis* 27 (2006), nov, Nr. 6, S. 923–942. http://dx.doi.org/10.1111/j.1467-9892.2006.00496.x. – DOI 10.1111/j.1467–9892.2006.00496.x

[McKenzie 1985] McKenzie, E.: Some Simple Models for Discrete Variate Time Series. In: *Journal of the American Water Resources Association* 21 (1985), Nr. 4, 645–650. http://www.jstor.org/stable/3214650. – ISSN 00219002

[Weiß et al. 2019] Weiss, C. ; Feld, M. ; Khan, N. ; Sunecher, Y.: INARMA Modeling of Count Time Series. In: *Stats* 2 (2019), Juni, Nr. 2, 284-320. http://dx.doi.org/10.3390/stats2020022. – DOI 10.3390/stats2020022