R package VGAMextra

by Victor Miranda
PhD Candidate, University of Auckland
Supervisor: Thomas Yee.

January 19, 2018

This document provides some examples and guidance of my package VGAMextra, for R. It comprises additions and extensions of the package VGAM (Yee, 2015), with functions, as well as S3/S4 methods addressing three main topics:

- **Time series modelling**. A novel class of VGLMs to model univariate time series, called vector generalized linear time series models (VGLTSMs). It is characterized by incorporating *past information* into the VGLM/VGAM loglikelihood. (Miranda and Yee, Manuscript in preparation, 2018a) gives further details.

- **1–parameter distribution mean modelling**. We return full circle by developing new link functions for the mean of 1–parameter distributions. VGAMs, VGLMs and GAMLSSs are restricted to location, scale and shape. However, the VGLM/VGAM framework has infrastructure to accommodate new links functions as a function of the parameters. See Miranda and Yee (Manuscript in preparation, 2018b) for more information.

- **Quantile modelling of 1–parameter distributions**. Similarly, we have developed link functions to model the quantiles of several 1–parameter distributions. Collectively, they represent an alternative to quantile regression by directly modelling the quantile function for distributions beyond the exponential family This framework is under development (Miranda and Yee, Manuscript in preparation, 2018c).

At present, this document shows examples on time series modelling, although it is reviewed and updated very often. Usage details on *mean and quantile modelling* will be incorporated over time. Shortly, all this information will also be available through my website, which is under construction.

For bugs and fixes, please email me at v.miranda@auckland.ac.nz

Note, my package depends on VGAM so make sure to install this firstly!

1 Vector generalized linear time series models

This section shows some examples of modelling choices for VGLTSMs. This sub–class of VGLMs accommodates several family functions describing many time series models as special cases.

1.1 AR(1) with ARCH(1) errors.

Chan et al. (2013) proposed a long and technical methodology to estimate the tail index of an AR(1) with ARCH(1)–errors involving its estimation by QMLE, given by

$$Y_t = Y_t | \Phi_{t-1} = \alpha Y_{t-1} + \sqrt{\omega + \beta \; Y_{t-1}^2} \; \varepsilon_t, \tag{1}$$

with $\varepsilon_t \overset{\text{iid}}{\sim} N(0,1)$; and α, $\omega > 0$, $\beta > 0$ to be estimated.

A quick inspection reveals that the (conditional) variance equation is

$$\sigma^2_{t|\Phi_{t-1}} = \text{Var}(Y_t | \Phi_{t-1}) = \omega + \beta Y_{t-1}^2.$$

This allows to model (1) as a VGLM. with linear predictors

$$\begin{aligned}
\eta_1 &= \; g_1(\mu^\star) = \mu^\star, && \text{(intercept–only)} \\
\eta_2 &= \; g_2(\sigma^2_{t|\Phi_{t-1}}) = \sigma^2_{t|\Phi_{t-1}} = \omega + \beta Y_{t-1}^2, \\
\eta_3 &= \; g_3(\alpha) = \alpha, && \text{(intercept–only)}.
\end{aligned}$$

or, alternatively, $\eta_2 = g_2(\sigma^2_{t|\Phi_{t-1}}) = \log \sigma^2_{t|\Phi_{t-1}}$, where the *effect* of Y_{t-1}^2 is constrained to $\sigma^2_{t|\Phi_{t-1}}$ (This is central).

The following code generates random observations from (1) and Figure 1.1 shows the resulting time series.

```
> ## Vector generalized linear time series models.
> ## Package VGAMextra last update: 30/11/2017
>
> my.loc <-  "~/phdvgam/myRlibs"
> library("VGAM", lib.loc = my.loc)
> library("VGAMextra", lib.loc = my.loc)
>
>
> ## Chan et.al. (2013). An ARCH(1, 1,). The variance equation
> ## parametrized in terms of lagged observations.
>
> # Generate some data
```

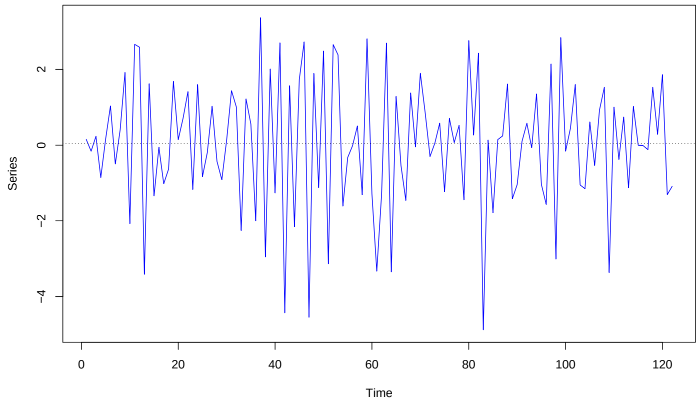


Figure 1.1. Simulated data from Model (1)

```
> set.seed(1)
> nn       <- ceiling(runif(1, 150, 160))
> my.rho   <- rhobit(-1.0, inverse = TRUE)  # -0.46212
> my.mu    <- 0.0
> my.omega <- 1
> my.b  <- 0.5
> covdata <- data.frame(x2 =  sort(runif(n = nn)))
> tsdata  <- transform(covdata, index = 1:nn, TS1   = runif(nn))
>
> for (ii in 2:nn)
    tsdata$TS1[ii] <- my.mu + my.rho * tsdata$TS1[ii-1] +
    sqrt(my.omega + my.b * (tsdata$TS1[ii-1])^2) * rnorm(1)
> # Remove the burn-in data:
> nnr <- ceiling(nn/5)
> tsdata <- tsdata[-(1:nnr), ]
> tsdata["index"] <- 1:(nn - nnr)
```

Model (1) can be seen as an AR(1) with ARCH(1) errors, such that either of the family functions `ARMA.GARCHff()` or `ARff()` from my package, along with the modelling function `vglm()` from VGAM, can be used to fit such structure straightforwardly.

Firstly, `ARMA.GARCHff()` imposes an ARMA(p, q) over the conditional mean, and assumes an GARCH(r, s) model for the variance equation, with i.i.d innovations from the standard Normal. The parameter vector comprises the coefficients involved with, both the conditional mean and variance equations, plus the drift parameter (denoted $\mu^\star$).

Thus, three linear predictors are involved with model (1): $\boldsymbol{\eta} = (\mu^\star, \sigma^2_{t|\Phi_{t-1}}, \alpha)^T$.

The statistical framework handled by `ARMA.GARCHff()` is

$$
\begin{aligned}
z_t &\overset{\text{iid}}{\sim} N(0,1) \\
\varepsilon_{t|\Phi_{t-1}} &= z_t \cdot \sigma^2_{\varepsilon_t|\Phi_{t-1}} \\
Y_t|\Phi_{t-1} &\sim N(\mu_{t|\Phi_{t-1}}, \sigma^2_{\varepsilon_t|\Phi_{t-1}}) \\
\mu_{t|\Phi_{t-1}} &= \mu^\star + \boldsymbol{\vartheta}^T \boldsymbol{y}_{t-u} + \boldsymbol{\phi}^T \boldsymbol{\varepsilon}_{t-v}
\end{aligned} \tag{2}
$$

with z_t independent of $\sigma^2_{\varepsilon_t|\Phi_{t-1}}$ with choices (for $\sigma^2_{\varepsilon_t|\Phi_{t-1}}$) shown in Table 1.1.

Table 1.1. Conditional variance models ($\sigma^2_{\varepsilon_t|\Phi_{t-1}}$) handled by (2) as special cases.

Model † §	Conditional variance ‡
Linear ARCH (LARCH)	$\sigma^2_{\varepsilon_t|\Phi_{t-1}} = \omega + \boldsymbol{\alpha}^T \boldsymbol{\varepsilon}^2_{t-r}$
Generalized–ARCH (GARCH)	$\sigma^2_{\varepsilon_t|\Phi_{t-1}} = \omega + \boldsymbol{\alpha}^T \boldsymbol{\varepsilon}^2_{t-r} + \boldsymbol{\gamma}^T \boldsymbol{\sigma}^2_{\varepsilon_{t-s}}$
Integrated–ARCH (IGARCH)	$\sigma^2_{\varepsilon_t|\Phi_{t-1}} = \omega + \boldsymbol{\alpha}^T \boldsymbol{\varepsilon}^2_{t-r} + \boldsymbol{\gamma}^T \boldsymbol{\sigma}^2_{\varepsilon_{t-s}}$ Subject to $\sum_r \alpha_r + \sum_s \gamma_s = 1$.
Taylor–Schwert	$\sigma_{\varepsilon_t|\Phi_{t-1}} = \omega + \boldsymbol{\alpha}^T \, |\boldsymbol{\varepsilon}_{t-r}| + \boldsymbol{\gamma}^T \boldsymbol{\sigma}_{\varepsilon_{t-s}}$
Asymmetric–GARCH (AGARCH)	$\sigma^2_{\varepsilon_t|\Phi_{t-1}} = \omega + \boldsymbol{\alpha}^T \boldsymbol{\varepsilon}^2_{t-r} + \boldsymbol{\zeta}^T \boldsymbol{\varepsilon}_{t-r} + \boldsymbol{\gamma}^T \boldsymbol{\sigma}^2_{\varepsilon_{t-s}}$
LogSD–GARCH (Log–GARCH)	$\log \sigma_{\varepsilon_t|\Phi_{t-1}} = \omega + \boldsymbol{\alpha}^T \, |\boldsymbol{\varepsilon}_{t-r}| + \boldsymbol{\gamma}^T \log \boldsymbol{\sigma}_{\varepsilon_{t-s}}$
Multiplicative–GARCH (M-GARCH)	$\log \sigma^2_{\varepsilon_t|\Phi_{t-1}} = \omega + \boldsymbol{\alpha}^T \log \boldsymbol{\varepsilon}^2_{t-r} + \boldsymbol{\gamma}^T \log \boldsymbol{\sigma}^2_{\varepsilon_{t-s}}$

† For all models, $\beta_{(j)1} = 0, \ldots, \beta_{(j)K} = 0, j = 1, \ldots, M$, that is, no covariates $\boldsymbol{x}_{t,(1)}$ admitted.
§ The ARMA model on $\mu_{t|\Phi_{t-1}}$ is optional.
‡ $\boldsymbol{\varepsilon}_{t-r}$ denotes $\boldsymbol{\varepsilon}_{t-r|\Phi_{t-1}}$ and $\boldsymbol{\sigma}_{\varepsilon_{t-s}}$ denotes $\boldsymbol{\sigma}_{\varepsilon_{t-s}|\Phi_{t-1}}$ for simplicity.

The linear predictor handled by this family function is $\boldsymbol{\eta}^T = (\mu^\star, \sigma^2_{\varepsilon_t|\Phi_{t-1}}, \boldsymbol{\vartheta}^T, \boldsymbol{\phi}^T)^T$ (in this order), with parameter vector

$$\boldsymbol{\theta}^T = (\mu^\star, \boldsymbol{\alpha}^T, \boldsymbol{\zeta}^T, \boldsymbol{\gamma}^T, \boldsymbol{\vartheta}^T, \boldsymbol{\phi}^T)^T.$$

On the other hand, `ARff()` models the AR process with zero–mean Normal innovations, as follows:

$$
\begin{aligned}
\varepsilon_t|\Phi_{t-1} &\sim N(0, \sigma^2_{\varepsilon_t|\Phi_{t-1}}) \\
Y_t|\Phi_{t-1} &\sim N(\mu_{t|\Phi_{t-1}}, \sigma^2_{\varepsilon_t|\Phi_{t-1}}) \\
\mu_{t|\Phi_{t-1}} &= \mu^\star \ + \boldsymbol{\vartheta}^T \boldsymbol{y}_{t-u}
\end{aligned} \tag{3}
$$

The linear predictor associated with `ARff()` is $\boldsymbol{\eta} = (\mu^\star, \sigma^2_{\varepsilon_t|\Phi_{t-1}}, \boldsymbol{\vartheta}^T)^T$. Both family functions manage covariates and multiple responses See Miranda and Yee (Manuscript in preparation, 2018a) for further details.

Now, let's go back to Model (1). Here are some notes:

- Our parameter vector is

$$\boldsymbol{\theta} = (\mu^\star, \omega, \beta, \alpha)^T,$$

  including coefficients from the variance equation, $\sigma^2_{t|\Phi_{t-1}} = \omega + \beta Y^2_{t-1}$, which specifies one linear predictor.

- We have two linear predictors modelled as intercept–only: $\mu^\star$ and α, besides $\log \sigma^2_{t|\Phi_{t-1}} = \omega + \beta Y^2_{t-1}$.

- **Most importantly**, to fit model (1) constraining the effect of Y^2_{t-1} over $\sigma^2_{t|\Phi_{t-1}}$, a number of choices are available, outlined below:

  1. **Through my family function** `ARMA.GARCHff()`.
     This is the easiest option. We just need to specify the ARMA order, the GARCH order, and the model type of interest. `ARMA.GARCHff()` handles several choices for the variance equation through the argument `type.TS`, e.g., ARCH, GARCH, Taylor-Schwert, etc., as per Table 2. Thus, to fit Model (1) set `type.TS = "ARCH"`, in addition, yielding:

```
> # Estimate the parameters.
> fit1 <-
   vglm(TS1 ~ 1, ARMA.GARCHff(ARMAorder = c(1, 0),   # ARMA order
                              GARCHorder = c(1, 0),  # ARCH order
                              type.TS = "ARCH",      # 'ARCH' type
                              type.param = "observed"),
        crit = "loglikelihood", trace = TRUE, data = tsdata)

VGLM    linear loop  1 :  loglikelihood = -218.3519
VGLM    linear loop  2 :  loglikelihood = -218.08409
VGLM    linear loop  3 :  loglikelihood = -218.06373
VGLM    linear loop  4 :  loglikelihood = -218.06192
VGLM    linear loop  5 :  loglikelihood = -218.06175
VGLM    linear loop  6 :  loglikelihood = -218.06174
VGLM    linear loop  7 :  loglikelihood = -218.06173

Checks on stationarity / invertibility successfully performed.
No roots liying inside the unit circle.
Further details within the 'summary' output.
```

     Some notes:

     (a) `type.TS` specifies the model type. Choices are `"ARCH", "GARCH", "IGARCH", "Taylor-Schwert", "A-GARCH", "Log-GARCH", "M-GARCH"`.

     (b) `type.param` specifies the parametrization class for the variance equation. In this example, Y^2_{t-1} is "observed" data. Alternatively, `type.param =`

`residuals` ($\{\varepsilon_t\}$) is also available, employed for the usual parametrization.

(c) We maximize the log–likelihood (argument `crit`). See Yee (2015) for more choices.

Furthermore, my family function `ARMA.GARCH()` internally checks whether the fitted process, $\{\hat{Y}_t\}$, is *stationary* or *invertible*. A short output in this regard is shown along with Fisher scoring iterations. In our example, all roots lie outside the unit circle. Else, the fitted process may not comply with either stationarity or invertibility conditions. Further details about estimates, se's, etc., are given along with the summary:

```
> summary(fit1)


Call:
vglm(formula = TS1 ~ 1, family = ARMA.GARCHff(ARMAorder = c(1,
    0), GARCHorder = c(1, 0), type.TS = "ARCH", type.param = "observed"),
    data = tsdata, crit = "loglikelihood", trace = TRUE)


Pearson residuals:
             Min     1Q   Median    3Q  Max
drift1    -1.981 -0.353 -0.03165 0.358 1.52
noiseVar1 -0.707 -0.648 -0.37072 0.351 6.55
ARcoeff11 -1.965 -0.405 -0.00701 0.365 2.52

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept):1   0.1518     0.1287    1.18    0.238
(Intercept):2   1.6356     0.2916    5.61  2.0e-08 ***
(Intercept):3  -0.4799     0.0862   -5.57  2.6e-08 ***
ARCH(1)         0.2603     0.1439    1.81    0.071 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Number of linear predictors:  3

Names of linear predictors: drift1, noiseVar1, ARcoeff11

Log-likelihood: -218.06 on 362 degrees of freedom

Number of iterations: 7

-----

** Standard errors based on the asymptotic
 distribution of the MLE estimates:

     ARcoeff1 drift
       -0.480 0.152
s.e.    0.078 0.199
```

```
  Estimated linear predictor of sigma^2 (SD errors):
(Intercept)     ARCH(1)
     1.6356      0.2603


Loglikelihood: -218.062
AIC: 442.123, AICc: 442.327, BIC: 450.536
-----

** Summary of checks on stationarity / invertibility:

 Polynomial roots of the AR component computed from the estimated
 coefficients: (Examining stationarity/invertibility)

      Model1
Root1  2.084
```

Finally, the estimated coefficients:

```
> ## Estimated coefficients:
> ## True values 'drift = 0','(AR coeff) alpha = -0.46212',
> ##  Variance equation -> ' (Intercept) omega = 1',
> ## '(TS1l1sq) beta = 0.5'
> coef(fit1, matrix = TRUE)

               drift1 noiseVar1  ARcoeff11
(Intercept) 0.1518497 1.6356150 -0.4798513
ARCH(1)     0.0000000 0.2602659  0.0000000
```

2. **Using constraint matrices**

This option involves the use of *constraint matrices*, another functionality conferred by the VGLM/VGAM framework. Here we just need some linear algebra to end up with the two required matrices "modelling" $\mu^\star$ and α as intercept–only:

$$\boldsymbol{\eta} = \begin{pmatrix} \mu^\star \\ \sigma^2_{t-1} \\ \alpha \end{pmatrix} = \begin{pmatrix} \beta_{(1)1} \\ \beta_{(2)1} + \beta_{(2)2} Y^2_{t-1} \\ \beta_{(3)1} \end{pmatrix} = \begin{pmatrix} \beta_{(1)1} \\ \beta_{(2)1} \\ \beta_{(3)1} \end{pmatrix} + \begin{pmatrix} 0 \\ Y^2_{t-1} \\ 0 \end{pmatrix} \cdot \beta_{(2)1}$$

$$= \underbrace{\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}}_{\mathbf{H}_1} \cdot \begin{pmatrix} \beta_{(1)1} \\ \beta_{(2)1} \\ \beta_{(3)1} \end{pmatrix} + (Y^2_{t-1} \cdot \mathrm{I}_3) \cdot \underbrace{\begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}}_{\mathbf{H}_2} \cdot \beta_{(2)2}$$

(a) These matrices must be manually entered for several of the VGAM family functions. `ARMA.GARCH()`, however, computes these matrices internally on the basis of the desired model (GARCH, ARCH, etc.). Hence, to fit (1), one sets the same model as in option 1!

```
> # Estimate the parameters.
> fit1bis <-
    vglm(TS1 ~ 1, ARMA.GARCHff(ARMAorder = c(1, 0),   # ARMA order
                               GARCHorder = c(1, 0),  # ARCH order
                               type.TS = "ARCH",
                               type.param = "observed"),
         crit = "loglikelihood", trace = TRUE, data = tsdata)
VGLM    linear loop  1 :  loglikelihood = -218.3519
VGLM    linear loop  2 :  loglikelihood = -218.08409
VGLM    linear loop  3 :  loglikelihood = -218.06373
VGLM    linear loop  4 :  loglikelihood = -218.06192
VGLM    linear loop  5 :  loglikelihood = -218.06175
VGLM    linear loop  6 :  loglikelihood = -218.06174
VGLM    linear loop  7 :  loglikelihood = -218.06173

Checks on stationarity / invertibility successfully performed.
No roots liying inside the unit circle.
Further details within the 'summary' output.
> fts.1 <- ts(fitted.values(fit1bis))
```

The (internally computed) constraint matrices:

```
> constraints(fit1bis)
$`(Intercept)`
     [,1] [,2] [,3]
[1,]    1    0    0
[2,]    0    1    0
[3,]    0    0    1

$`ARCH(1)`
     [,1]
[1,]    0
[2,]    1
[3,]    0
```

(b) Another option is my family function `ARff()`. Here, the constraint matrices must be entered manually via an object of class list, with all elements named accordingly. The `R` code is shown below.

This approach, however, implies a slight change wrt option 1. We must incorporate Y_{t-1}^2 as an *explanatory* in the `formula`, which must be computed firstly, and then *constrain* its effect over $\sigma^2_{\varepsilon_t|\Phi_{t-1}}$ via the object `const.mat`. Here, `zero = NULL` produces all linear predictors to be modelled in terms of Y_{t-1}^2 (not intercept–only), but `const.mat` inhibits the effect of Y_{t-1}^2 as desired. The code is:

```
> ## Constraint matrices
> (const.mat <- list('(Intercept)' = diag(3), 'TS1l1sq' = cbind(c(0, 1, 0))))
$`(Intercept)`
```

```
     [,1] [,2] [,3]
[1,]    1    0    0
[2,]    0    1    0
[3,]    0    0    1

$TS1l1sq
     [,1]
[1,]    0
[2,]    1
[3,]    0
```

```
> ## Set up the data using function WN.lags() from VGAMextra.
> tsdata2 <- transform(tsdata, TS1l1sq =
              WN.lags(y = cbind(tsdata[, "TS1"])^2, lags = 1))
>
> ## Fitting the model
> fit2 <- vglm(TS1 ~ TS1l1sq, ARff(order = 1,   # AR order
                                  zero = NULL, noChecks = FALSE,
                                  var.arg = TRUE, lvar = "identitylink"),
              crit = "loglikelihood", trace = TRUE,
              # Constraints...
              constraints = const.mat, data = tsdata2)
```

```
VGLM    linear loop  1 :  loglikelihood = -217.71018
VGLM    linear loop  2 :  loglikelihood = -216.35111
VGLM    linear loop  3 :  loglikelihood = -216.10699
VGLM    linear loop  4 :  loglikelihood = -216.07562
VGLM    linear loop  5 :  loglikelihood = -216.07187
VGLM    linear loop  6 :  loglikelihood = -216.07142
VGLM    linear loop  7 :  loglikelihood = -216.07137
VGLM    linear loop  8 :  loglikelihood = -216.07137
VGLM    linear loop  9 :  loglikelihood = -216.07137

Checks on stationarity / invertibility successfully performed.
No roots lying inside the unit circle.
Further details within the 'summary' output.
```

Finally, let's check the constraint matrices and the estimated coefficients (Similar results!):

```
> ## Estimated coefficients
> ## True values 'drift = 0','(AR coeff) alpha = -0.46212',
> # Variance equation -> ' (Intercept) omega = 1', '(TS1l1sq) beta = 0.5'
> coef(fit2, matrix = TRUE)
```

```
              ARdrift1 noiseVar1   ARcoeff11
(Intercept) 0.1644261  1.243996 -0.5297543
TS1l1sq     0.0000000  0.372093  0.0000000
```

```
> constraints(fit2)
```

```
$`(Intercept)`
     [,1] [,2] [,3]
[1,]    1    0    0
[2,]    0    1    0
[3,]    0    0    1

$TS1l1sq
     [,1]
[1,]    0
[2,]    1
[3,]    0
```

3. **Using the argument `zero`.**
A third option comes up by using the argument `zero`, from the modelling function `vglm()`. Argument `zero` is an object of class *vector* specifying the names (then a vector of character strings) or positions (then an integer vector) of those **linear predictors** to be modelled as **intercept–only**. For further details, see the help documentation of `CommonVGAMffArguments` from VGAM.

Here, again, `TS1l1sq` must be entered as a covariate along with the formula. Recall, the linear predictor is $\boldsymbol{\eta} = (\mu^\star, \sigma^2_{\varepsilon_t|\Phi_{t-1}}, \alpha)^T$. Then, the code to fit the model under this approach is:

```
> # Fit the model
> fit2bis <- vglm(TS1 ~ TS1l1sq,
                 ARff(order = 1, lvar = "identitylink",
                      var.arg = TRUE, zero = c("drift", "coeff")),
                 crit = "loglikelihood", trace = TRUE, data = tsdata2)

VGLM    linear loop  1 :  loglikelihood = -217.71018
VGLM    linear loop  2 :  loglikelihood = -216.35111
VGLM    linear loop  3 :  loglikelihood = -216.10699
VGLM    linear loop  4 :  loglikelihood = -216.07562
VGLM    linear loop  5 :  loglikelihood = -216.07187
VGLM    linear loop  6 :  loglikelihood = -216.07142
VGLM    linear loop  7 :  loglikelihood = -216.07137
VGLM    linear loop  8 :  loglikelihood = -216.07137
VGLM    linear loop  9 :  loglikelihood = -216.07137

Checks on stationarity / invertibility successfully performed.
No roots lying inside the unit circle.
Further details within the 'summary' output.
```

Note,

(a) `zero = c("drift", "coeff")` indicates that the *drift* and the AR coefficient are modelled as intercept–only. Hence, Y^2_{t-1} affects $\sigma^2_{\varepsilon_t|\Phi_{t-1}}$ exclusively.

(b) `lvar = "identitylink"` enables the identity link to model $\sigma^2_{\varepsilon_t|\Phi_{t-1}}$

(c) `var.arg = TRUE` allows to model the variance, $\sigma^2_{\varepsilon_t|\Phi_{t-1}}$, directly. If `FALSE`, then $\sigma_{\varepsilon_t|\Phi_{t-1}}$ is modelled instead.

Finally, the summary and estimated coefficients:

```
> summary(fit2bis)


Call:
vglm(formula = TS1 ~ TS1l1sq, family = ARff(order = 1, lvar = "identitylink",
    var.arg = TRUE, zero = c("drift", "coeff")), data = tsdata2,
    crit = "loglikelihood", trace = TRUE)
```

```
Pearson residuals:
              Min      1Q    Median     3Q   Max
ARdrift1  -1.8937 -0.3511 -0.047568 0.3914 2.051
noiseVar1 -0.7067 -0.6498 -0.410287 0.4273 5.927
ARcoeff11 -1.8603 -0.3509  0.003344 0.3787 2.407

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept):1  0.16443    0.12275   1.340  0.18040
(Intercept):2  1.24400    0.25167   4.943 7.70e-07 ***
(Intercept):3 -0.52975    0.09787  -5.413 6.21e-08 ***
TS1l1sq        0.37209    0.13290   2.800  0.00511 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Number of linear predictors:  3

Names of linear predictors: ARdrift1, noiseVar1, ARcoeff11

Log-likelihood: -216.0714 on 362 degrees of freedom

Number of iterations: 9

-----

** Standard errors based on the asymptotic
 distribution of the MLE estimates:

     ARcoeff1 drift
       -0.530 0.164
s.e.    0.081 0.212

  Estimated linear predictor of sigma^2 (SD errors):
(Intercept)     TS1l1sq
     1.2440      0.3721


Loglikelihood: -216.071
AIC: 438.143, AICc: 438.346, BIC: 446.555
-----

** Summary of checks on stationarity / invertibility:

 Polynomial roots of the AR component computed from the estimated
 coefficients: (Examining stationarity/invertibility)

      Model1
Root1  1.888


> ############# Estimated coefficients #################
> ## True values 'drift = 0','(AR coeff) alpha = -0.46212',
> # Variance equation -> ' (Intercept) omega = 1', '(TS1l1sq) beta = 0.5'
```

```
> coef(fit2bis, matrix=TRUE)

              ARdrift1 noiseVar1  ARcoeff11
(Intercept) 0.1644261  1.243996 -0.5297543
TS1l1sq     0.0000000  0.372093  0.0000000
```

1.2 Time series of counts

This section shows (briefly) the performance of another family function in my package: `VGLM.INGARCHff()`, for time series of counts.

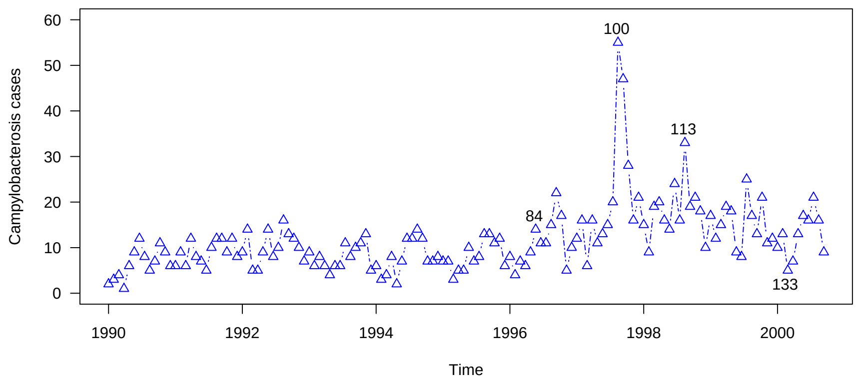


Figure 1.2. Number of campylobacterosis infections in Northern Québec reported every 28 days.

Here, we analyze the number of campylobacterosis infections in Northern Québec, Canada, reported every 28 days between January 1990 and October 2010. Compared to this, results from packages tscount, gamlss, and glarma, are also presented. The series is displayed in Figure 1.2, and can be retrieved from package tscount.

In particular, Fokianos and Fried (2010), Fokianos and Fried (2012), Liboschik et al. (2016a), and Liboschik et al. (2016b) present a compendium of intervention analysis techniques using this set of data, but restricted to the Poisson and negative binomial distributions (with mean response $\lambda_{t|\Phi_{t-1}}$), and investigate INGARCH models with single intervention effects only. For this dataset, they propose a conservative approach to predict the average change in rate of campylobacter infections over (yearly) *seasonal* effects (i.e., regressing on λ_{t-13}), plus short–term distributed impacts from the last reported period (accounting for serial dependence), and no intervention effects. The linear predictor has the following form:

$$\log \lambda_{t|\Phi_{t-1}} = \omega + \vartheta_1 y_{t-1} + \phi_1 \lambda_{t-13|\Phi_{t-14}}. \tag{4}$$

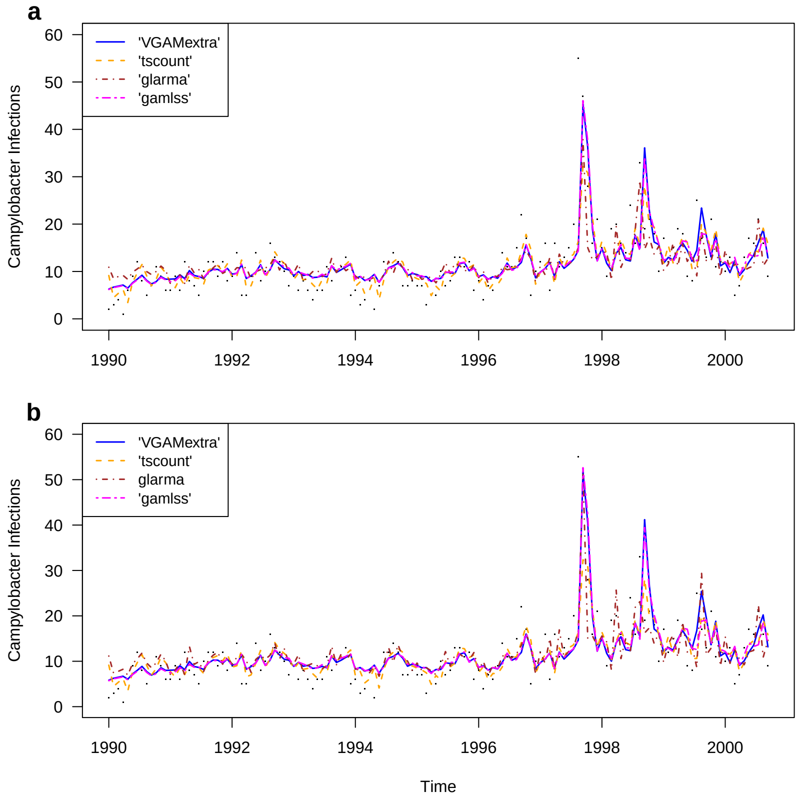


Figure 1.3. Campylobacterosis infections: Fitted values based on Model (4) assuming (**a**) Poisson and (**b**) negative binomial response. No intervention effects.

Imposing Poisson and negative binomial distributions on the response, Figure 1.3 shows the predicted values after fitting model (4) using packages VGAMextra, tscount, glarma, and gamlss. Figure 1.4 presents the PIT histrograms computed with `PIT()` from my package, VGAMextra. PITs show evidence that the negative binomial produces more accurate predictions.

Later, the authors incorporated intervention analysis, identifying intervention–influence at times t_{84} and t_{100}. They utilized the order–$(p,\ q)$ INGARCH–class to explore this series, but including *single–intervention* only (See Liboschik et al. (2016a), Section 6). The INGARCH linear predictor with 's' types of interventions with decay rates $\delta_1, \ldots, \delta_s$ (known and fixed) occurring at points $\tau_1, \ldots, \tau_s$ is

$$g(\lambda_t) = \beta_0 + \sum_{k=1}^{p} \beta_k \tilde{g}(Y_{t-i_k}) + \sum_{\lessdot=1}^{q} \alpha_{\lessdot} g(\lambda_{t-j_{\lessdot}}) + \sum_{i=1}^{s} \omega_m \delta^{t-\tau_m} \mathbb{1}(t \geq \tau_m),$$

where ω_m, $m = 1, \ldots, s$ are the intervention sizes.

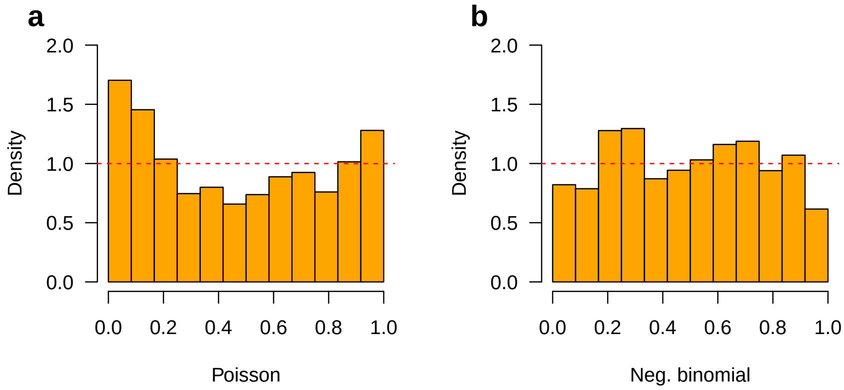


Figure 1.4. PIT histograms build on Model (4), computed with `PIT()` from VGAMextra, assuming (**a**) Poisson and (**b**) negative binomial distributions.

Compared to this, we illustrate the impact of *joint–intervention effects* on the series by incorporating interaction terms. Specifically, two influential observations at the bottom of the series have been selected, t_{113}, and t_{133}, around years 1998–2000 (See Figure 1.2). Also, the series may be negatively affected for both "shocks", with possibly singular effects at t_{133} (hence $\delta_4 = 0$). However, the shock t_{113} lies further in the vicinity of t_{100} and reasonably around t_{84}, and may collectively influence the series with exponential consequences. As a result, we set a conservative $\delta_3 = 0.5$. In addition, we slightly relax the permanent effect at t_{84}, identified by Fokianos and Fried (2010); Liboschik et al. (2016a) (the singular effect at t_{100} remains), resulting in $\boldsymbol{\delta} = (0.99, 0, 0.5, 0)^T$ and $\boldsymbol{\tau} = (84, 100, 113, 133)^T$, and hence the following linear predictor

$$\log \lambda_{t|\Phi_{t-1}} = \omega + \vartheta_1 y_{t-1} + \phi_1 \lambda_{t-13|\Phi_{t-14}} + \sum_{h=1}^{4} \omega_h \cdot \delta_h^{t-\tau_h} \mathbb{1}(t \geq \tau_h) + \omega_5 \cdot \delta_1^{t-\tau_1} \mathbb{1}(t \geq \tau_1) \cdot \delta_3^{t-\tau_3} \mathbb{1}(t \geq \tau_3). \tag{5}$$

This structure can only be handled by the modelling function `vglm()` using my family function `VGLM.INGARCHff()`. Figure 1.5 shows the fitted values *with and without* interactions terms, involving intervention analysis.

Finally, in addition to Poisson and negative binomial, the logarithmic and the Yule–Simon distributions are also handled by my family function `VGLM.INGARCHff()`.

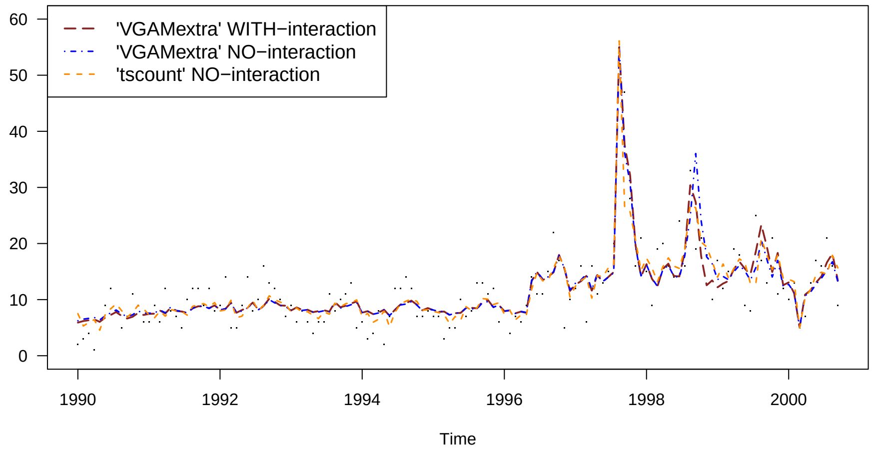


Figure 1.5. Fitted values from intervention effects model (5), compared with tscount (This package does not handle no joint–interventions).

The VGLM/VGAM framework is also able to handle (further examples soon).

- Cointegrated (bi–dimensional) time series
- Multivariate time series.
- Later, I will upgrade my framework on time series to handle VGAMs.
- Forecasting S4 methods are still under development.

2 On modelling the mean of 1–parameter distributions

We also have developed new links for the mean–function of several 1–parameter discrete and continuous distributions. These are presented in Tables 2.1 and 2.2.

Distribution	θ	Range of θ	Support	Mean μ	Link function $[\eta(\theta)]$
Borel–Tanner	a	$(0,1)$	$Q(1)\infty$	$Q/(1-\theta)$	`loge` $\left(Q^{-1}-\theta Q^{-1}\right)$
Geometric	p	$(0,1)$	$0(1)\infty$	$(1-\theta)/\theta$	$-$`logit`(θ)
Logarithmic	s ‡	$(0,1)$	$1(1)\infty$	$\dfrac{\theta}{(1-\theta)\,[-\log(1-\theta)]}$	`logit`(θ) $-$ `cloglog`(θ)
Positive Poisson	λ	$(0,\infty)$	$1(1)\infty$	$\dfrac{\theta}{1-e^{-\theta}}$	$-$`loge` $\left(\theta^{-1}-\theta^{-1}e^{-\theta}\right)$
Yule–Simon	ρ ‡	$(0,\infty)$	$1(1)\infty$	$\dfrac{\theta}{\theta-1},\ \theta>1$	$-$`loge` $\left(1-\theta^{-1}\right)$
zeta (Zipf)	s ‡	$(0,\infty)$†	$1(1)\infty$	$\zeta(\theta)/\zeta(\theta+1),\ \theta>1$	`loge` $[\zeta(\theta)/\zeta(\theta+1)]$

† The density and the moments of the Zipf distribution here conforms with the family function `zetaff` at package VGAM. Particularly, ζ is the Riemman Zeta function.
‡ These are 'shape' parameters.

Table 2.1. New links for the mean–function of some discrete distributions

Distribution	θ	Range of θ	Support	Mean μ	Link function $[\eta(\theta)]$
Exponential †	λ	$(0,\infty)$	(A,∞)	$A+\theta^{-1}$	`loge` $\left(A+\theta^{-1}\right)$
Gamma ‡	s	$(0,\infty)$	$(0,\infty)$	θ	`loge`(θ)
Inverse–χ^2	df	$[0,\infty)$	$(0,\infty)$	$\dfrac{1}{\theta-2},\ \theta>2$	$-$`loge`$(\theta-2)$
Maxwell §	a	$(0,\infty)$	$(0,\infty)$	$a^{-1/2}\sqrt{\dfrac{8}{\pi}}$	$\kappa_1-\dfrac{1}{2}$`loge`(θ)
Rayleigh ‖‡	b	$(0,\infty)$	$(0,\infty)$	$b\dfrac{\Gamma(0.5)}{\sqrt{2}}$	`loge`$(\theta)+\kappa_2$
Topp–Leone ♯	s	$(0,1)$	$(0,1)$	$1-\dfrac{4^{\theta}\Gamma(1+\theta)^2}{\Gamma(2+2\theta)}$	`logit` $(\mu(\theta)/\kappa_3)$

† A is a *location* parameter (fixed) and λ is a rate.
‡ No link functions required. The default link in VGAM accommodates this.
§ $\kappa_1=\frac{3}{2}\log 2-\log\Gamma(0.5)$, where Γ denotes the gamma function.
‖ $\kappa_2=\log\Gamma(0.5)-\frac{1}{2}\log 2$.
♯ $\kappa_3=\sup_{0<\theta<1}\left\{1-\frac{4^{\theta}\Gamma(1+\theta)^2}{\Gamma(2+2\theta)}\right\}$

Table 2.2. New links for the mean–function of some continuous distributions

2.1 An example

To illustrate how such mean–links operate, I will focus on the logarithmic distribution with unique parameter $\theta \in (0, 1)$. The mean, as a function of θ, is

$$-\frac{\theta}{(1-\theta)\log(1-\theta)}. \tag{6}$$

As a general approach in this subject, we take the logarithm of (6), producing the following interesting difference, and hence, specifying the new link function (called `logffMeanlink( · )`):

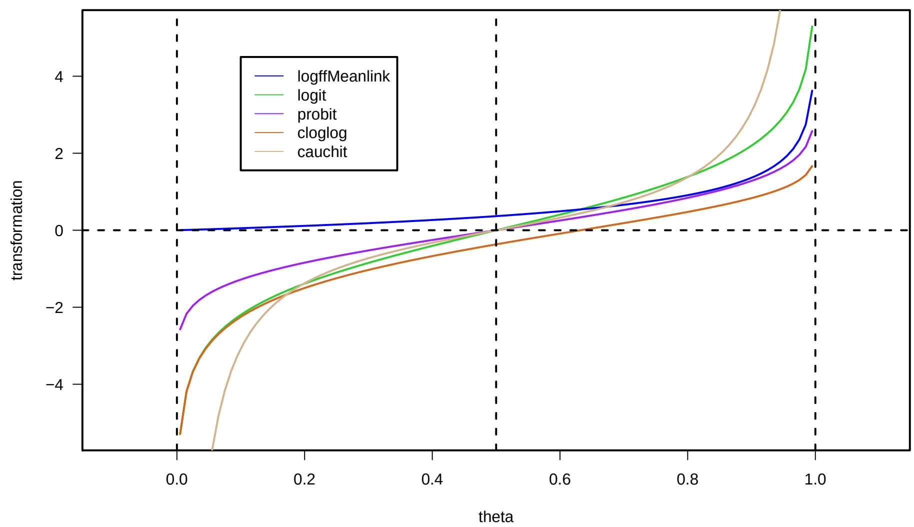
$$\texttt{logffMeanlink}(\theta) = \texttt{logit}(\theta) - \texttt{cloglog}(\theta). \tag{7}$$

The mean–link, or more precisely, our linear predictor (as a function of θ) for a set of covariates $\boldsymbol{x}$, is then given by

$$\eta(\theta) = \texttt{logffMeanlink}(\theta) = \boldsymbol{\beta}^T\boldsymbol{x}.$$

To accommodate new links, the VGLM/VGAM framework requires, among others, smootheness (we must be able to compute the inverse, at least numerically), and the derivatives $\partial\eta/\partial\theta$ and $\partial\theta/\partial\eta$ as a function of θ (See Yee (2015) for further details). Particularly, I have implemented (7) in my package, via the function `logffMeanlink()`. Figure 2.1 shows this link plotted over $(0, 1)$, and is compared to other popular probability links.

Figure 2.1. Some probability link functions



The following code shows `logffMeanlink()` in action while fitting a VGLM, with one covariate, $X2$, yielding

$$\eta(\theta) = \texttt{logit}(\theta) - \texttt{cloglog}(\theta) = \beta_1 + \beta_2 X2. \tag{8}$$

```
> nn <- 120
> # Reference: logffMeanlink(theta = 1, inverse = TRUE) ~ 0.8263
> set.seed(2)
> log.data <- data.frame(X2 = runif(nn, 0, 1))
> log.data <- transform(log.data,
                        y = rlog(nn, shape = logffMeanlink(theta = 1 + 1 * X2,
                                                            inverse = TRUE)))
> head(log.data)

         X2  y
1 0.1848823  1
2 0.7023740  1
3 0.5733263 25
4 0.1680519  7
5 0.9438393  1
6 0.9434750  8

> ## logffMeanlink(theta) = logit(theta) - cloglog(theta)
>
> ## fit the vglm
> fit4 <- vglm(y ~ X2, family = logff(lshape = logffMeanlink, zero = NULL),
               data = log.data, trace = TRUE)

VGLM    linear loop  1 :  loglikelihood = -270.77373
VGLM    linear loop  2 :  loglikelihood = -270.72577
VGLM    linear loop  3 :  loglikelihood = -270.72575
VGLM    linear loop  4 :  loglikelihood = -270.72575

> ## Estimated coefficients. True is beta0 = beta1 = 1.
> coef(fit4, matrix = TRUE)

            logfflink(shape)
(Intercept)        1.0728599
X2                 0.9138094
```

If we have no covariates involved (intercept–only model, hence $\eta(\theta) = \beta_1$), our estimate would be obtained by applying the *inverse* of `logffMeanlink( )`, i.e.,

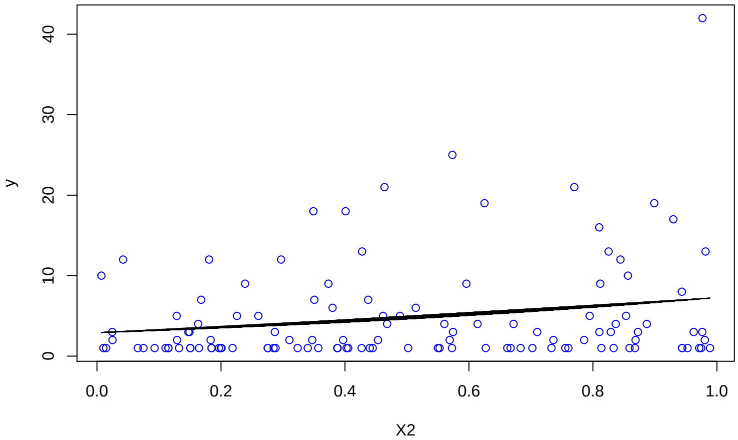
$$\widehat{\alpha} = \texttt{logffMeanlink}^{-1}(\widehat{\beta}_1.)$$

The `R` code would be:

```
> # An intercept only model.
> fit4.bis <- vglm(y ~ 1, family = logff(lshape = logffMeanlink, zero = NULL),
                 data = log.data, trace = TRUE)

VGLM    linear loop  1 :  loglikelihood = -273.18342
VGLM    linear loop  2 :  loglikelihood = -273.18342

> # Our estimate: (True is 1.0)
> logffMeanlink(coef(fit4.bis), inverse = TRUE)

[1] 0.9254939
```

Finally, Figure 2.2 shows the fitted values.

Figure 2.2. Fitted values from (8).



3 On quantile modelling

Quantile regression has been exceedingly addressed in the literature with methodologies relying on optimizing variants of many target functions (e.g., minimizing the absolute error functions) with no overriding framework.

I propose an alternative to quantile regresssion by directly modelling any set of $100p\%$ quantiles via *VGLM/VGAM quantile–links*, currently encompassing several 1–parameter distributions. At a later stage, we will extend this work to distributions with P parameters.

In this document I will restrict myself to one example: The Maxwell distribution (for simplicity) with rate parameter θ. Hopefully, this example will provide suffice grounds for users to address their own models. Further details/information will be incorporated over time.

My proposed quantile–link for the $p\%$ quantile of the Maxwell distribution is specified by the corresponding linear predictor:

$$\eta(\theta; p) = \frac{1}{2}\log 2 + \log \texttt{qgamma(p, 1.5)} - \frac{1}{2}\log\theta,$$

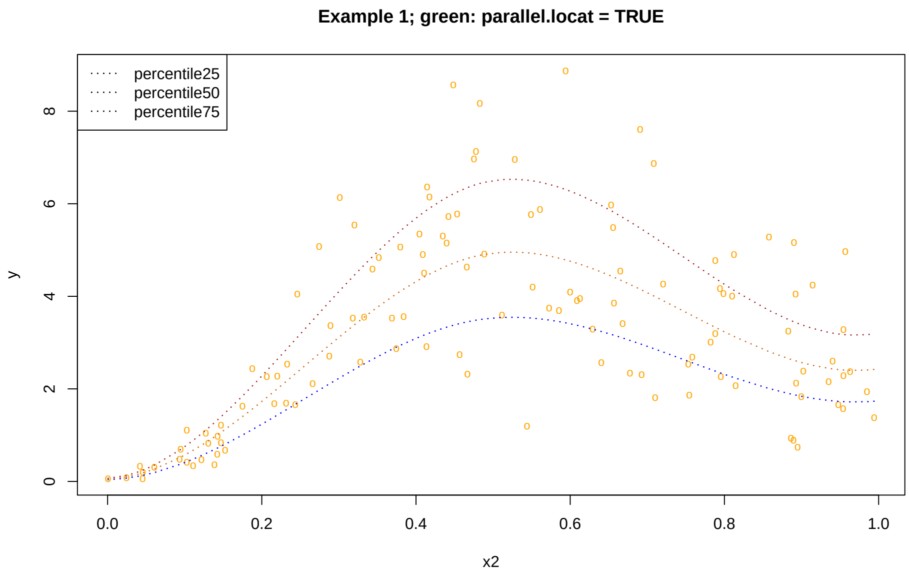
called `maxwellQlink`. This and other quantile–links have been implemented in `R`, and are available in my package. In particular, $\eta(\theta; p)$ above is available through the function `maxwellQlink()`.

The following code generates random data distributed as Maxwell, where its rate is modelled as

$$\text{rate} = \exp\left[2 - 6 * \sin(2x_2 - 0.2)/(x_2 + 0.5)^2\right],$$

for a random covariate x_2. Next, we use splines to fit a VGAM, incorporating our quantile modelling function. Here, we are interested on modelling the 25%, 50% and 75%.

```
> set.seed(123)
>
> # An artificial covariate.
> maxdata <- data.frame(x2 = sort(runif(n <- nn)))
> # The 'rate' function.
> mymu <- function(x) exp(2 -  6 * sin(2 * x - 0.2) / (x + 0.5)^2)
> # Set up the data.
> maxdata <- transform(maxdata, y = rmaxwell(n, rate = mymu(x2)))
>
> # 25%, 50% amd 75% quantiles are to be modelled.
> mytau <- c(0.25, 0.50, 0.75)
> mydof <- 4
>
> # Use VGAM to fit the data.
> fit <- vgam(Q.reg(y, pvector = mytau) ~ bs(x2, df = mydof),
              family = maxwell(link = maxwellQlink(p = mytau), zero = NULL),
```




```
                data = maxdata, trace = TRUE, eps = 1e-4)

VGAM  vlm.wfit  loop  1 :  loglikelihood = -833.99
VGAM  vlm.wfit  loop  2 :  loglikelihood = -528.28
VGAM  vlm.wfit  loop  3 :  loglikelihood = -514.6
VGAM  vlm.wfit  loop  4 :  loglikelihood = -514.53
VGAM  vlm.wfit  loop  5 :  loglikelihood = -514.53
```

Some notes:

- `Q.reg()`, from my package, must be included in the formula. Here, `pvector` is a numeric vector containing the quantiles to be modelled (entries between 0 and 1).
- `bs()` are usual B–spline basis, from package splines.
- `maxwell()` is the VGAM family function that estimates the parameter of the Maxwell distribution (by MLE using Fisher scoring).

Finally, let's check the percentage of data below the 25%, 50% and 75% curves.

```
> # Below the 25% quantile
> round(length(predict(fit)[, 1][(maxdata$y
                            <= predict(fit)[, 1]  )]) /nn, 3) * 100

[1] 24.2

> # Below the 50% quantile
> round(length(predict(fit)[, 2][(maxdata$y
                                  <= predict(fit)[, 2] )]) /nn, 3) * 100
```

```
[1] 50.8

> # Below the 75% quantile.
> round(length(predict(fit)[, 3][(maxdata$y
                                  <= predict(fit)[, 3] )]) /nn, 3) * 100

[1] 77.5
```

Further options, choices and details are to be incorporated over time, or via a couple of papers in preparation. See the references.

Package VGAMextra tested okay on R version 3.4.3.
Document continuously updated...

Victor Miranda
Last update: January 19, 2018.

References

N. Chan, D. Li, L. Peng, and R. Zhang. Tail index of an AR(1) model with ARCH(1) errors. *Econometric Theory*, 29(5):920–940, 2013.

W. Diethelm, many others, and see the SOURCE file. *fArma: ARMA time series modelling*, 2013a. R package version 3010.79.

W. Diethelm, C. Yohan, M. Michal, B. Chris, C. Pierre, et al. *fGarch: Rmetrics – Autoregressive Conditional Heteroskedastic Modelling*, 2013b. R package version 3010.82.

K. Fokianos. Some recent progress in count time series. *Statistics*, 45:49–58, 2011.

K. Fokianos and R. Fried. Interventions in INGARCH process. *Journal of Time Series Analysis*, 31(3):210–225, 2010.

K. Fokianos and R. Fried. Interventions in log–linear Poisson autoregression. *Statistical Modelling*, 12(4):299–322, 2012.

K. Fokianos and D. Tjostheim. Log–linear Poisson autoregression. *Journal of Multivariate Analysis*, 102(3):563–578, 2011.

T. Liboschik, K. Fokianos, and R. Fried. tscount: An R package for analysis of count time series following generalized linear models. *To appear in Journal of Statistical Software*, 2016a.

T. Liboschik, P. Keischke, K. Fokianos, and R. Fried. Modelling interventions in INGARCH processes. *International Journal of Computer Mathematics*, 93(4):640–657, 2016b.

V. Miranda and T. Yee. Vector generalized linear time series models. Manuscript in preparation, 2018a.

V. Miranda and T. Yee. On mean modelling of 1–parameter distributions using VGLMs. Manuscript in preparation, 2018b.

V. Miranda and T. Yee. Vector generalized linear and additive models towards quantile modelling: A general framework for quantile regression. Manuscript in preparation, 2018c.

B. Pfaff. *Analysis of integrated and cointegrated time series with R*. Springer, Seattle, Washington, USA, 2011.

R Core Team. *R: A language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Austria, 2017. URL `https://www.R-project.org/`.

R. Rigby and D. Stasinopouls. Generalized additive models for location, scale and shape, (with discussion). *Applied Statistics*, 54(3):507–554, 2005.

G. Sucarrat. *lgarch: simulation and estimation of Log-GARCH models*, 2015. R package version 0.6-2.

T. Yee. *Vector generalized linear and additive models with an implementation in R*. Springer, New York, USA, 2015.