Distributional Modelling in R

R - Packages

The following is an (incomplete) overview of R packages that can be used to estimate Generalized Additive Models for Location Scale and Shape (GAMLSS). Each package presented is accompanied by a short example using the dataset `rent` from the package **gamlss.data**. Comprehensive examples and further assistance can be found in the documentation of the respective packages and in the associated software papers.

1 gamlss

The **gamlss** package stands as one of the pioneering packages capable of estimating versatile distributional regression models incorporating diverse model terms. Its primary model fitting function, `gamlss()`, facilitates this process. Additionally, the accompanying **gamlss.dist** package offers a range of distributions to choose from. With **gamlss**, users can model distributions with up to four parameters, providing a comprehensive framework for statistical modeling and analysis.

Examples:

```
R> library("gamlss")
R> ## Load the Munich rent data.
R> data("rent", package = "gamlss.data")

R> ## Estimate model using the three parameter BCCGo distribution.
R> b <- gamlss(R ~ pb(Fl) + pb(A) + H + loc,
+    sigma.fo = ~ pb(Fl) + pb(A) + H + loc,
+    nu.fo = ~ pb(Fl) + pb(A) + H + loc,
+    family = BCCGo, data = rent)

R> ## Show model diagnostic plots.
R> plot(b)
R> ## Wormplot.
R> wp(b)
R> ## Model summary.
R> summary(b)
R> ## AIC()
R> GAIC(b)

R> ## Effect plots.
R> term.plot(b, pages = 1, what = "mu", ask = FALSE)
R> term.plot(b, pages = 1, what = "sigma", ask = FALSE)
R> term.plot(b, pages = 1, what = "nu", ask = FALSE)

R> ## Predict parameters.
R> p <- predictAll(b, newdata = rent[1:5, ])
R> ## Compute log-likelihood using the family.
R> ## Each family has a d*, p*, q* and r* function.
R> d <- with(p, dBCCGo(y, mu = mu, sigma = sigma, nu = nu, log = TRUE))
R> sum(d)
```

For more examples please see the corresponding paper

https://www.jstatsoft.org/article/view/v023i07

A list of books on GAMLSS can be found here

https://www.gamlss.com/

2 gamlss2

The primary purpose of this package is to facilitate the creation of advanced infrastructures designed to enhance the GAMLSS modeling framework. Notably, the **gamlss2** package represents a significant overhaul of its predecessor, **gamlss**, with a key emphasis on improving estimation speed and incorporating more flexible infrastructures. These enhancements enable the seamless integration of various algorithms into GAMLSS, including gradient boosting, Bayesian estimation, regression trees, and forests, fostering a more versatile and powerful modeling environment.
Moreover, the package expands its compatibility by supporting all model terms from the base R **mgcv** package. Additionally, the **gamlss2** package introduces the capability to accommodate more than four parameter families. Essentially, this means that users can now specify any type of model using these new infrastructures, making the package highly flexible and accommodating to a wide range of modeling requirements.
The development version of **gamlss2** is hosted on github at and can be installed via

```
R> devtools::install_github("gamlss-dev/gamlss2")
```

Examples:

```
R> library("gamlss2")
R> ## Load the Munich rent data.
R> data("rent", package = "gamlss.data")

R> ## Estimate model using the three parameter BCCGo distribution.
R> ## Model formula, uses R package Formula infrastructures (or list of formulas).
R> f <- R ~ s(Fl) + s(A) + H + loc |
+    s(Fl) + s(A) + H + loc |
+    s(Fl) + s(A) + H + loc
R> ## Estimation.
R> b <- gamlss2(f, family = BCCGo, data = rent)

R> ## Show model diagnostic plots.
R> plot(b, which = "resid")
R> ## Model summary.
R> summary(b)
R> ## AIC()
R> GAIC(b)

R> ## Effect plots.
R> plot(b)

R> ## Predict parameters.
R> p <- predict(b, newdata = rent[1:5, ], type = "parameter")
R> ## Compute log-likelihood using the family.
R> family(b)$loglik(rent$R[1:5], p)
R> ## Density.
R> family(b)$d(rent$R[1:5], p)
R> ## Quantiles.
R> family(b)$q(0.01, p)
R> family(b)$q(0.5, p)
R> family(b)$q(0.99, p)
R> ## Random numbers.
R> family(b)$r(1, p)
R> family(b)$r(10, p[1, ])
```

3 mgcv

The contributed R package **mgcv** stands out as perhaps the most extensively utilized package for flexible regression modeling. Its versatile infrastructures enable the creation of user-specific smooth terms within general models, offering unparalleled flexibility. Moreover, **mgcv** excels in estimating Generalized Additive Models (GAMs) even with very large datasets. However, despite its extensive capabilities, the range of implemented distributions for estimating Generalized Additive Models for Location Scale and Shape (GAMLSS) is not as

expansive. Additionally, implementing new distributions for GAMLSS is not straightforward within the **mgcv** framework.

Examples:

```
R> library("mgcv")
R> ## Load the Munich rent data.
R> data("rent", package = "gamlss.data")

R> ## Estimate model using the two parameter normal distribution.
R> ## Model formula.
R> f <- list(
+    R ~ s(Fl) + s(A) + H + loc,
+      ~ s(Fl) + s(A) + H + loc
+  )
R> ## Estimation.
R> b <- gam(f, family = gaulss, data = rent)

R> ## Show model diagnostic plots.
R> gam.check(b)
R> ## Model summary.
R> summary(b)
R> ## AIC()
R> AIC(b)

R> ## Effect plots.
R> plot(b, pages = 1)
R> plot(b, pages = 1, scale = 0)
R> plot(b, pages = 1, scale = 0, scheme = 1)

R> ## Predict parameters.
R> p <- predict(b, newdata = rent[1:5, ], type = "response")
R> ## Compute log-likelihood.
R> sum(dnorm(rent$R[1:5], mean = p[, 1], sd = 1/p[, 2], log = TRUE))
R> ## Quantiles.
R> qnorm(0.01, mean = p[, 1], sd = 1/p[, 2])
R> qnorm(0.5, mean = p[, 1], sd = 1/p[, 2])
R> qnorm(0.99, mean = p[, 1], sd = 1/p[, 2])
```

4 bamlss

The R package **bamlss** offers robust infrastructures tailored for estimating Bayesian additive models for location scale and shape. Users can define smooth terms utilizing the flexible functionalities provided by the **mgcv** package. Additionally, the package allows users to implement custom optimizer and sampler functions, enhancing flexibility and adaptability. Key features include a backfitting algorithm for posterior mode estimation, an efficient Markov Chain Monte Carlo (MCMC) algorithm for comprehensive Bayesian inference, a gradient boosting algorithm for variable selection, and an interface to TensorFlow through the **keras** package for estimating deep distributional neural network models, among others.

Examples:

```
R> library("bamlss")
R> library("gamlss.dist")
R> ## Load the Munich rent data.
R> data("rent", package = "gamlss.data")

R> ## Estimate model using the three parameter BCCGo distribution.
R> ## Model formula, uses R package Formula infrastructures (or list of formulas).
R> f <- R ~ s(Fl) + s(A) + H + loc |
+    s(Fl) + s(A) + H + loc |
+    s(Fl) + s(A) + H + loc
R> ## Estimation.
R> b <- bamlss(f, family = BCCGo, data = rent)
```

```
R> ## Show model diagnostic plots.
R> plot(b, which = 3:5)
R> ## Model summary.
R> summary(b)
R> ## DIC()
R> DIC(b)

R> ## Effect plots.
R> plot(b, pages = 1)
R> ## Traceplots.
R> plot(b, which = "samples")

R> ## Predict parameters for all samples.
R> p <- predict(b, newdata = rent[1:5, ], type = "parameter", FUN = identity)
R> ## Compute the mean over samples instead (default).
R> p <- predict(b, newdata = rent[1:5, ], type = "parameter", FUN = mean)
R> ## Predictions are lists, transform to data frame.
R> p <- as.data.frame(p)
R> ## Compute log-likelihood using the family.
R> family(b)$loglik(rent$R[1:5], p)
R> ## Density.
R> family(b)$d(rent$R[1:5], p)
R> ## Quantiles.
R> family(b)$q(0.01, p)
R> family(b)$q(0.5, p)
R> family(b)$q(0.99, p)
R> ## Random numbers.
R> family(b)$r(1, p)
R> family(b)$r(10, p[1, ])
```

For more examples please see the corresponding paper

https://www.jstatsoft.org/article/view/v100i04

5 gamboostLSS

The core functionality of the R package **gamboostLSS** revolves around a gradient boosting algorithm, specifically designed to facilitate automatic variable selection within the framework of Generalized Additive Models for Location Scale and Shape (GAMLSS). This package seamlessly integrates with all families available in the **gamlss.dist** package, ensuring a comprehensive range of distribution options. Additionally, **gamboostLSS** offers built-in features for cross-validation and provides a selection of base learners tailored for estimating flexible smooth effect models.

Examples:

```
R> library("gamboostLSS")
R> ## Load the Munich rent data.
R> data("rent", package = "gamlss.data")

R> ## Model formula with base learners.
R> f <- R ~ bbs(Fl) + bbs(A) + bols(H) + bols(loc)
R> ## Control arguments.
R> ctrl <- boost_control(trace = TRUE, mstop = c(mu = 1100, sigma = 1000, nu = 1000))
R> ## Estimation.
R> b <- gamboostLSS(f, families = as.families(BCCGo, stabilization = "MAD"),
+    data = rent, control = ctrl)
R> ## Plot effects.
R> par(mfrow = c(3, 4))
R> plot(b, type = "l")
R> ## Show stopping iteration.
R> mstop(b)
```

For more examples please see the corresponding paper

https://www.jstatsoft.org/article/view/v074i01

6 quantreg

The R package **quantreg** facilitates quantile regression analysis, offering efficient computation, robust inference, and diagnostic tools for assessing model fit. It allows users to specify models using a formula interface, compute standard errors and confidence intervals, and customize functionality as needed. **quantreg** is widely used in various fields for analyzing conditional quantiles of the response variable.

Examples:

```
R> library("quantreg")
R> ## Load the Munich rent data.
R> data("rent", package = "gamlss.data")

R> ## Model formula.
R> f <- R ~ ns(Fl, df = 4) + ns(A, df = 4) + H + loc
R> ## Specify quantiles for estimation.
R> tau <- seq(0.01, 0.99, by = 0.01)
R> ## Estimation.
R> b <- rq(f, data = rent, tau = tau)
R> ## Plot estimated coefficients.
R> plot(b)
R> ## Predict for each quantile.
R> p <- predict(b, newdata = rent[1:5, ])
```

7 qgam

The **qgam** package specializes in implementing quantile generalized additive models (QGAM). Leveraging the infrastructures of the **mgcv** package, it efficiently establishes the additive predictors required for modeling. Despite its notable flexibility, it's worth noting that computation times may extend considerably when handling very large datasets.

Examples:

```
R> library("qgam")
R> ## Load the Munich rent data.
R> data("rent", package = "gamlss.data")

R> ## Model formula.
R> f <- R ~ s(Fl) + s(A) + H + loc
R> ## Estimation.
R> b <- qgam(f, data = rent, qu = 0.5)
R> ## Plot effects.
R> plot(b, pages = 1, scheme = 1)
R> ## Predict.
R> p <- predict(b, newdata = rent[1:5, ])
R> ## Multiple quantiles.
R> b <- mqgam(f, data = rent, qu = c(0.01, 0.5, 0.99))
R> ## Predict.
R> p <- qdo(b, c(0.01, 0.5, 0.99), predict, newdata = rent[1:5, ])
R> ## Pinball loss.
R> pinLoss(rent$R[1:5], p[[1]], 0.01)
R> pinLoss(rent$R[1:5], p[[2]], 0.5)
R> pinLoss(rent$R[1:5], p[[3]], 0.99)
```

For more examples please see the corresponding paper

https://www.jstatsoft.org/article/view/v100i09

8 disttree

The **disttree** package is useful for estimating full probabilistic distributional trees and forests. Trees are grown based on instability tests of the score vectors and split variables, resulting in each leaf node containing a complete distributional model. The package is work in progress and only available from R-Forge at the moment.

Installation:

```
R> if(!require("disttree")) {
+     install.packages("disttree", repos = "http://R-Forge.R-project.org",
+       dependencies = TRUE)
+  }
```

Examples:

```
R> library("disttree")
R> ## Load the Munich rent data.
R> data("rent", package = "gamlss.data")

R> ## Estimate model using the two parameter GA distribution.
R> f <- R ~ Fl + A + H + loc
R> ## Estimation.
R> b <- disttree(f, family = GA, data = rent)

R> ## Plot the tree.
R> plot(b)

R> ## Estimate a forest.
R> b <- distforest(f, family = GA, data = rent, ntree = 100)

R> ## Predict parameters.
R> p <- predict(b, newdata = rent[1:5, ])

R> ## Compute log-likelihood using the GA distribution.
R> sum(dGA(rent$R[1:5], mu = p$mu, sigma = p$sigma, log = TRUE))

R> ## Quantiles.
R> qGA(0.01, mu = p$mu, sigma = p$sigma)
R> qGA(0.5, mu = p$mu, sigma = p$sigma)
R> qGA(0.99, mu = p$mu, sigma = p$sigma)

R> ## Random numbers.
R> rGA(1, mu = p$mu, sigma = p$sigma)
```