GAMLSS R Packages Reference Card

Contents

1 GAMLSS R Packages

The GAMLSS framework comprises elevent different packages written in R, i.e. the original **gamlss** package and ten add-on packages

1. the original **gamlss** package for fitting a GAMLSS model
2. the **gamlss.add** package for extra additive terms.
3. the **gamlss.cens** package for fitting censored (left, right or interval) response variables.
4. the **gamlss.data** package for data used
5. the **gamlss.demo** package useful for teaching purposes (i.e. plotting the distributions interactively)
6. the **gamlss.dist** package for `gamlss.family` distributions
7. the **gamlss.mx** package for fitting finite mixture distributions.
8. the **gamlss.nl** package for fitting non-linear models
9. the **gamlss.spatial** package for spatial models.
10. the **gamlss.tr** package for fitting truncated distributions.
11. the **gamlss.util** package for additional utilities

The GAMLSS framework packages can be downloaded and installed from CRAN, the R library at http://www.r-project.org/.

2 Fitting or Updating a Model (in gamlss package)

`gamlss()` for fitting and creating a `gamlss` object

`refit()` to refit a `gamlss` object (i.e. continue iterations) if it has not converged

`update()` to update a given `gamlss` model object

`histDist()` to fit a parametric distribution to a single (response) variable and plot

`gamlssML()` to fit a single parametric distribution to data (no explanatory variables)

`fitDist()` to fit several parametric distribution to a single set of data choosing the one with minimum GAIC

`histSmo()` to fit a non-parametric density function to a single set of data.

3 Extracting Information from the Fitted Model

`AIC()` to extract the generalized Akaike information criterion (GAIC) from a fitted `gamlss` model object

`GAIC()` identical to `AIC`

`coef()` to extract the linear coefficients from a fitted `gamlss` model object

`confint()` to get confidence intervals for the beta parameters

`deviance()` to extract the global deviance of the `gamlss` model object

`edf()` to extract the effective degrees of freedom from a fitted `gamlss` model object

`edfAll()` to extract the effective degrees of freedom for all the parameters from a fitted `gamlss` model object

`extractAIC()` to extract the generalized Akaike information criterion from a fitted `gamlss` model object

`fitted()` to extract the fitted values from a fitted `gamlss` model object

`formula()` to extract a model formula

`fv()` to extract the fitted values for a distribution parameter (see also `fitted()` and `lpred()`)

`gen.likelihood()` to generate the likelihood function of the fitted model (used by `vcoc()`)

`get.K()` to extract the K matrix (meat) for sandwich standard errors

`getSmo()` to extract information from fitted smoothing terms

`logLik()` to extract the log likelihood

`lp()` to extract the linear predictor for a distribution parameter (see also `lpred`)

`lpred()` to extract the fitted values, linear predictor or specified terms (with standard errors) for a distribution parameter.

`model.frame()` to extract the model frame of a specified distribution parameter

`model.matrix()` to extract the design matrix of a specified distribution parameter

`predict()` to predict from new data individual distribution parameter values (see also `lpred` below)

`predictAll()` to predict from new data all the distribution parameter values

`print()` : to print a `gamlss` object

`residuals()` to extract the normalized (randomized) quantile residuals from a fitted `gamlss` model object. See Dunn and Smyth (1996) for a definition of the normalized (randomized) quantile residuals.

`Rsq()` generalised R-squared

`rvcov()` to extract the robust (sandwich) variance-covariance matrix of the beta estimates (for all distribution parameter models). It can be done also with `vcov()`.

`summary()` to summarize the fit in a `gamlss` object

`terms()` to extract terms from a `gamlss` object

`vcov()` to extract the variance-covariance matrix of the beta estimates (for all distribution parameter models).

4 Selecting a Model

`add1()` to add a single term, from those supplied, to a fitted `gamlss` model object

`addterm()` to add a single term, from those supplied, to a fitted `gamlss` model object (used by `stepGAIC()` below).

`CV()` to compare cross validation global deviance of models after using `gamlssCV()` function

`drop1()` to drop one term at the time from the fitted `gamlss` model object.

`dropterm()` to fit all models that differ from the current fitted `gamlss` model object by dropping a single term (used by `stepGAIC()` below).

`find.hyper()` to find the hyperparameters (e.g. degrees of freedom for smoothing terms and/or non-linear parameters) by minimizing the profile Generalized Akaike Information Criterion (GAIC) based on the global deviance

`getTGD)` to use a fitted model to evaluate the global deviance on validation/test data set

`gamlss.scope()` to define the scope for `stepGAIC()`

`gamlssVGD()` to fit a model to training data and evaluate the global deviance on validation/test data set

`gamlssCV()` to fit a k-fold gross validation model to a data set

`LR.test()` Likelihood ratio test for two different fitted models

`stepGAIC()` to select explanatory terms using GAIC

`stepGAIC.CH()` to select (additive) using the method of Chambers and Hastie (1992).

`stepGAIC.VR()` to select (parametric) terms using method of Venables and Ripley (2002).

`stepGAICAll.A()` to select explanatory terms for all the parameters using GAIC and strategy A

`stepGAICAll.B()` to select explanatory terms for all the parameters using GAIC and strategy B

`stepTGD()` (experimental) to select explanatory terms using the global deviance of a test data set.

`TGD()` to compare global deviance of models after using `getVGD()` function

`VGD()` to compare global deviance of model fitted using `gamlssVGD()`

5 Plotting and Diagnostics

`acfResid()` ACF plots for the residuals of a fitted model

`plot()` a plot of four graphs for the normalized (randomized) quantile residuals of a `gamlss` object. The residual plots are: (i) against an x-variable (ii) against the fitted values, (iii) a density plot and (iv) a QQ-plot. Note that residuals are randomized only for discrete response variables, see Dunn and Smyth (1996).

`par.plot()` for plotting parallel profile plots for individual participants in repeated measurement analysis

`pdf.plot()` for plotting the pdf functions for a given fitted `gamlss` object or a given `gamlss.family` distribution

`prof.dev()` for plotting the profile global deviance of one of the distribution parameters μ, σ, ν or τ.

`prof.term()` for plotting the profile global deviance of one of the model (beta) parameters. It can be also used to study the GAIC($\sharp$) information profile of a hyperparameter for a given penalty $\sharp$ for the GAIC.

`plot2way()` for plotting 2 way interactions in a model. Note that this only works for categorical variables (factors).

`Q.stats()` for printing the Q statistics of Royston and Wright (2000).

`rqres.plot()` for plotting QQ-plots of different realizations of normalized randomized quantile residuals for a model with a discrete `gamlss.family` distribution.

`show.link()` for showing available link functions for distribution parameters in any `gamlss.family` distribution

`term.plot()` for plotting additive (smoothing) terms in any distribution parameter model

`wp()` worm plot of the residuals from a fitted `gamlss` object. See vanBuuren and Fredriks (2001).

`dtop()` a detrended transform Own's plot of the residuals from a fitted `gamlss` object.

6 Checking the tail of a distribution

`loglogSurv1()` plots the (left or right) tails of the empirical log-log Survival function against `loglog(y)`, for detecting Type I tails.

`loglogSurv2()` plots the (left or right) tails of the empirical log-log Survival function against `log(y)` for detecting Type II tails.

`loglogSurv3()` plots the (left or right) tails of the empirical log-log Survival function against `y` for detecting Type III tails.

`loglogSurv()` This function combines the above three functions

`logSurv1()` For detecting extreme tails

The function `fitTail()` and `fitTailAll()` in the package **gamlss.tr** are also design for exploring the tails of distributions.

7 Centile Estimation

`lms()` to fit an LMS models for centile estimation

`calibration()` to plot calibrated centile curves (corrected for sample percetanges) against an x-variable.

`centiles()` to plot centile curves against an x-variable.

`centiles.com()` to compare centiles curves for more than one `object`.

`centiles.split()` as for `centiles()`, but splits the plot at specified values of x.

`centiles.pred()` to predict and plot centile curves for new x-values.

`centiles.fan()` to plot a fan-chart for the centiles.

`fittedPlot()` to plot fitted values for all the parameters against an x-variable

8 Additive Terms

Additive terms are in the main `gamlss` expept `nl` which is in package `gamlss.nl`

Additive terms	R Name
cubic splines	`cs(), scs()`
cycle P-spline	`cy()`
P-splines varying coefficient	`pvc()`
P-splines	`pb(), pbo(), ps()`
monotonic P-splines	`pbm()`
penalised lags	`la()`
`loess`	`lo()`
fractional polynomials	`fp()`
free knots	`fk()`
non-linear fit	`nl()`
random effects	`random(), ra()`
ridge regression	`ri()`
Simon Wood's gam	`ga()`
neutral networks	`nn()`
regression trees	`tr()`

Table 1: Additive terms implemented within the **gamlss** packages

9 `gamlss.family` Distributions

The package **gamlss.dist** contains all the continuous, discrete and mixed distributions for `gamlss.family`. The distributions are shown in tables 2, 3 and 4 respectively.

The following functions exist to generate a new log or or logit family from a existing `gamlss.family` defined on $(-\infty, \infty)$:

`gen.Family()` Generates a log or logit family from a `gamlss.family` defined on $(-\infty, \infty)$

`Family( )` fits a log or logit version of a `gamlss.family` defined on $(-\infty, \infty)$

`Family.d()` pdf of a log or logit version of a `gamlss.family` defined on $(-\infty, \infty)$

`Family.p()` cdf of a log or logit version of a `gamlss.family` defined on $(-\infty, \infty)$

`Family.p()` inverse cdf of a log or logit version of a `gamlss.family` defined on $(-\infty, \infty)$

`Family.p()` Generates random values from a log or logit version of a `gamlss.family` defined on $(-\infty, \infty)$

The following functions exist to generate a hazard function from a existing `gamlss.family`

`hazardFun()` takes as an argument a `gamlss.family` object and creates the hazard function for it

`gen.hazard( )` generates a hazard function called hNAME where NAME is a gamlss.family i.e. hGA().

10 Package gamlss.add Functions

blag(), llag(), wlag() basis, list and weights for lag values

fitFixedKnots() for fitting fixed break points

fitFreeKnots() for fitting free break points

fk() for fitting free break points within **gamlss** (using **fitFreeKnots()**)

fk.control() control function for **fk()**

ga() interface for using the smoothers of the **gam()** function of the Simon Wood's package **mgcv**

ga.control() control function for **ga()**

la() for fitting penalised lags within **gamlss** (using **penLags()**)

la.control() control function for **la()**

nn() interface for using Brian Ripley's neural network function **nnet()** within **gamlss**

nn.control() control function for **nn()**

penLags() for fitting penalised lags

tr() interface for using the function **rpart()** within **gamlss**

11 Package gamlss.cens Functions

gen.cens() Generates appropriate functions to be used to fit a censored response variable

cens() Fits a censored distribution from a **gamlss.family** distribution

cens.d() Generates a censored probability density function from a **gamlss.family** distribution

cens.p() Generates a censored cumulative density function of a **gamlss.family** distribution

cens.q() Generates a censored inverse cumulative density function of a **gamlss.family** distribution

Distributions	R Name
beta	BE()
Box-Cox Cole and Green	BCCG()
Box-Cox power exponential	BCPE()
Box-Cox t	BCT()
exponential	EXP()
exponential Gaussian	exGAUS()
exponential gen. beta type 2	EGB2()
gamma	GA()
generalised beta type 1	GB1()
generalised beta type 2	GB2()
generalised gamma	GG()
generalised inverse Gaussian	GIG()
generalised t	GT()
Gumbel	GU()
inverse Gamma	IGAMMA()
inverse Gaussian	IG()
Johnson's SU (μ the mean)	JSU()
Johnson's original SU	JSUo()
logistic	LO()
logistic normal	LOGITNO()
log normal	LOGNO()
log normal re-parametrisation of LOGNO()	LOGNO2()
log normal (Box-Cox)	LNO()
NET	NET()
normal	NO()
normal family	NOF()
power exponential	PE()
Pareto 2	PARETO2()
Pareto 2 original	PARETO2o()
Pareto re-parametrisation of PARETO2()	GP()
reverse Gumbel	RG()
skew normal type 1	SN1()
skew normal type 2	SN2()
skew power exponential type 1	SEP1()
skew power exponential type 2	SEP2()
skew power exponential type 3	SEP3()
skew power exponential type 4	SEP4()
sinh-arcsinh original	SHASHo()
sinh-arcsinh original 2	SHASHo2()
sinh-arcsinh	SHASH()
skew t type 1	ST1()
skew t type 2	ST2()
skew t type 3	ST3()
skew t re-parametrisation of ST3()	SST()
skew t type 4	ST4()
skew t type 5	ST5()
t Family	TF()
t Family re-parametrisation of TF()	TF2()
Weibull	WEI()
Weibull (PH)	WEI2()
Weibull (μ the mean)	WEI3()

Table 2: Continuous distributions implemented within the **gamlss** packages

Distributions	R Name
beta binomial	`BB()`
binomial	`BI()`
geometric	`GEOM()`
Delaporte	`DEL()`
logarithmic	`LG()`
negative binomial type I	`NBI()`
negative binomial type II	`NBII()`
Poisson	`PO()`
Poisson inverse Gaussian	`PIG()`
Sichel	`SI()`
Sichel (μ the mean)	`SICHEL()`
Waring	`WARING()`
Yule	`YULE()`
zero altered beta binomial	`ZABB()`
zero altered binomial	`ZABI()`
zero altered logarithmic	`ZALG()`
zero altered neg. binomial	`ZANBI()`
zero altered poisson	`ZAP()`
zero inflated beta binomial	`ZIBB()`
zero inflated binomial	`ZIBI()`
zero inflated neg. binomial	`ZINBI()`
zero inflated poisson	`ZIP()`
zero inflated poisson (μ the mean)	ZIP2()
zero inflated poisson inv. Gaussian	`ZIPIG()`

Table 3: Discrete distributions implemented within the **gamlss** packages

beta inflated (at 0)	`BEOI()`
beta inflated (at 0)	`BEINF0()`
beta inflated (at 1)	`BEZI()`
beta inflated (at 1)	`BEINF1()`
beta inflated (at 0 and 1)	`BEINF()`
zero adjusted GA	`ZAGA()`
zero adjusted IG	`ZAIG()`

Table 4: Mixed distributions implemented within the **gamlss** packages

12 Package `gamlss.data`

Data sets in package Ôgamlss.dataÕ are :

`abdom` Abdominal Circumference Data

`acidity` The Acidity Data files for GAMLSS

`aep` The Hospital Stay Data

`aids` Aids Cases in England and Wales

`alveolar` The Alveolar Data files for GAMLSS

`CD4` The CD4 Count Data

`computer` The Computer Failure Data files for GAMLSS

`db` Head Circumference of Dutch Boys

`dbbmi` BMI of Dutch Boys

`fabric` The Fabric Data

`film30` Film revenue data for the 1930's

`film90` Film revenue data for the 1990's

`glass` The Glass Data files for GAMLSS

`hodges` Hodges data

`hodges1` Hodges data

`LGAclaims` The LGA Claims Data files for GAMLSS

`lice` Data files for GAMLSS

`margolin` The Margolin Data files for GAMLSS

`Mums` Mothers encouragement data

`mvi` The motor vehicle insurance data: a sample of 2000

`mvi` The motor vehicle insurance data

`parzen` The Parzen Data File for GAMLSS

`polio` Poliomyelitis cases in US

`rent` Rent data

`rent99` Munich rent data of 1999

`rent99.polys` The boundaries file for Munich rent data from the 1999 survey.

`species` The Fish Species Data files for GAMLSS

`stylo` The Stylometric Data files for GAMLSS

`tensile` The Tensile Data files for GAMLSS

`tse` The Turkish stock exchange index

`usair` US air pollution data set

`vas5` Visual analog scale (VAS) data

`VictimsOfCrime` Reported victims of crime data

13 Package gamlss.demo Functions

There are three types of demos:

`demoDist()`: for `gamlss.family` distributions. Every `gamlss.family` distribution has its own demo function for example, `demo.NO()` is a demo for the normal distribution.

`demoPsplines()`: for penalised smoothing techniques. There are five demos:

- `demo.BSplines()`: Demo for B-splines.
- `demo.histSmo()`: Demo for histogram smoothing method based on fitting a Poisson distribution.
- `demo.interpolateSmo()`: Demo for showing the interpolation and extrapolation on the Whittaker (random walk) smoothing method.
- `demo.PSplines()`: Demo for P-splines.
- `demo.RandomWalk()`: Demo for random walk (Whittaker smoothing method).

`demoLpolyS()`: for local polynomial smoothing. There are five demos:

- `demo.LocalRegression()` Demo for demonstrating some characteristics of local regression smoothing.
- `demo.Locmean()` Demo for local mean smoothing.
- `demo.Locpoly()` Demo for local polynomial smoothing.
- `demo.WLocmean()` A demo for weighted local mean (kernel) smoothing
- `demo.WLocpoly()` A demo for weighted local polynomial smoothing

The function `gamlss.demo()` invokes all of them.

14 Package gamlss.mx Functions

`dMX()` for evaluating a pdf function

`gamlss.MX()` for fitting a finite mixtures with no parameters in common (in **gamlss.mx**) package

`gamlss.MXfits()` for fitting several `gamlss.MX()` models

`gamlss.NP()` for fitting a finite mixtures with parameters in common (in **gamlss.mx**) package

`pMX()` for evaluating a cdf function

15 Package gamlss.nl Functions

`nlgamlss()` Fitting a non-linear GAMLSS, (note that here the first argument is `y`, not `formula`)

`nl())` for fitting a non-linear terms as additive term within (gamlss)

16 Package gamlss.spatial Functions

`draw.polys` plots the fitted values of fitted `MRF` object.

`MRF()` Fitting Markov Random Fields Functions (using the Q-function)

`mrf()` Markov Random Field fitting within GAMLSS (using the Q-function)

`MRFA()` Fitting Markov Random Fields Functions (using the alternating method)

`mrfa()` Markov Random Field fitting within GAMLSS (using the alternating method)

`nb2nb()` transforms from a shape file neighbour (S4 object) to the neighbour required form for functions MRF()

`nb2prec()` creates the precision matrix from the neighbour information.

`polys2nb()` gets the neighbour information from the polygons.

`polys2polys()` transforms a shape file polygons (S4 object) to the polygons required form for the functions MRF() and MRFA().

17 Package gamlss.tr Functions

`gen.trun()` Generates a truncate distribution from a `gamlss.family`

`trun()` fits a truncate Distribution from a `gamlss.family`

`trun.d()` Truncated probability density function of a `gamlss.family` distribution

`trun.p()` Truncated cumulative density function of a `gamlss.family` distribution

`trun.q()` Truncated inverse cumulative density function of a `gamlss.family` distribution

`trun.r()` Generates random values from a truncated density function of a `gamlss.family` distribution

`fitTail()` Fits a truncated distribution to the tail of the data

`fitTailAll()` A sequence of truncated fits for plotting the results in a Hill type plot.

18 Package `gamlss.util` Functions

`centiles.ts()` Plots the centile curves for a time series GAMLSS object

`garmaFit()` A function to fit a GARMA model

`lagPlot()` Lag plot for time series data

`plotSimpleGamlss()` A plot to demonstrate how the distribution of the response changes according to values of one explanatory variable

`scattersmooth()` Two dimensional Smooth scatter plots

19 The `gamlss` Function Arguments

`formula` a model formula (including the response variable y) for the `mu` parameter (compulsory), e.g. `y` $\sim$ `x`.

`sigma.formula` a model formula for `sigma`, e.g. $\sim$`x`

`nu.formula` a model formula for `nu`, e.g. $\sim$`x`

`tau.formula` a model formula formula for `tau`, e.g. $\sim$`x`

`family` a `gamlss.family` distribution family

`data` a data frame containing the variables occurring in the formulae

`weights` a vector of weights. `weights` can be used i) to weight out observations (with weights equal to 1 or 0) ii) for a weighted likelihood analysis (typically appropriate if weights represent frequencies). Any other use of the `weights` could have side effects.

`contrasts` list of contrasts to be used for some or all of the factors appearing as variables in the parameter(s) model formula.

`method` the algorithms for GAMLSS, i.e. RS(), CG() or mixed().

`start.from` a fitted GAMLSS model from which to take the starting values for the current model

`mu.start` vector or scalar for initial values for the location parameter `mu`.

`sigma.start` vector or scalar for initial values for the scale parameter `sigma`.

`nu.start` vector or scalar of initial values for the shape parameter `nu`.

`tau.start` vector or scalar of initial values for `tau`.

`mu.fix` fixing the `mu` parameter

`sigma.fix` fixing the `sigma` parameter

`nu.fix` fixing the `nu` parameter

`tau.fix` fixing the `tau` parameter

`control` control parameters of the outer iterations algorithm (see`gamlss.control`) function

`i.control` control parameters of the inner iterations of the RS algorithm (see `glim.control`)

Mikis Stasinopoulos, Bob Rigby
London Metropolitan University
`d.stasinopoulos@londonmet.ac.uk`
`r.rigby@londonmet.ac.uk`
Vlasios Voudouris
ABM analytics
`vlasios.voudouris@abm-analytics.com`
Gillian Heller
Macquarie Univarsity
`fgillian.heller@mq.edu.au`
Fernanda De Bastiani
Federal University of Pernambuco
`fernandadebastiani@gmail.com`