universität innsbruck

gamlss2: The Next Generation of Distributional Regression

Nikolaus Umlauf, Mikis Stasinopoulos, Fernanda De Bastiani, Gillian Heller, Thomas Kneib, Andreas Mayr, Robert Rigby, Reto Stauffer, Achim Zeileis

http://nikum.org

The GAMLSS Journey

- Similar to GLMs, GAMLSS are strongly related to the IWSM.
- **2001**, first GAMLSS talk by Mikis and Bob at the IWSM 2001 in Odense, DK:

  "*The GAMLSS project: A Flexible Approach to Statistical Modelling*".
- **20** years ago, 2005-06-24, first CRAN *gamlss* package.
- **2005**, JRSS C., `doi:10.1111/j.1467-9876.2005.00510.x`

Volume 54, Issue 3
June 2005

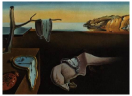
Generalized Additive Models for Location, Scale and Shape

R. A. Rigby ✉ , D. M. Stasinopoulos

Journal of the Royal Statistical Society Series C: Applied Statistics, Volume 54, Issue 3, June 2005, Pages 507–554, https://doi.org/10.1111/j.1467-9876.2005.00510.x

Published: 14 April 2005

- **2007**, JSS, `doi:10.18637/jss.v023.i07`

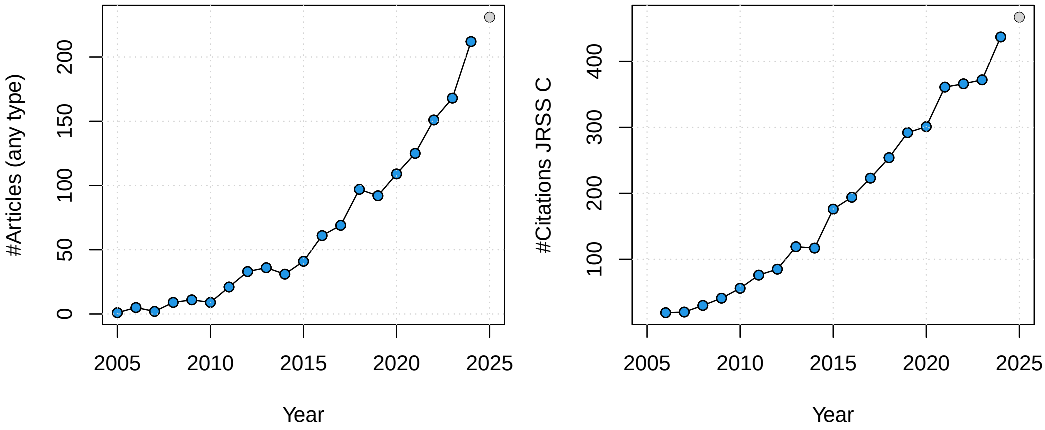


Generalized Additive Models for Location Scale and Shape (GAMLSS) in R

D. Mikis Stasinopoulos, Robert A. Rigby

The GAMLSS Journey

Google Scholar queries (`GAMLSS + distributional regression`):



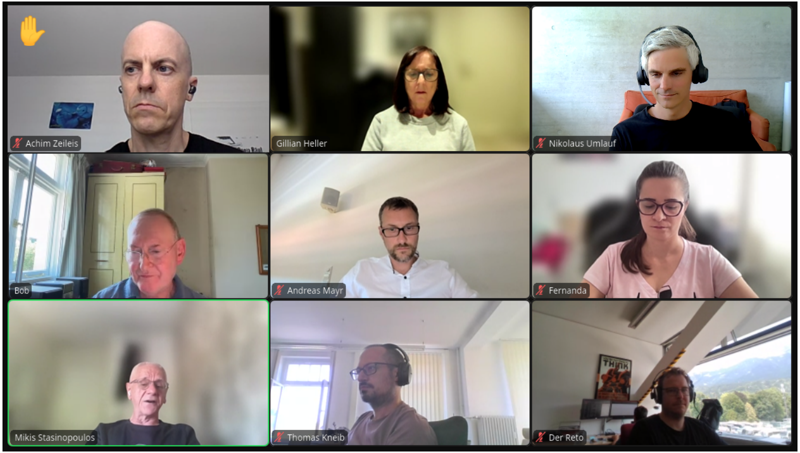
The GAMLSS Journey

- R package *gamlss*: flexible regression for location, scale, shape.
- Limitations in existing R packages (*gamlss*, *VGAM*, *mgcv*, *bamlss*)
- Need for extensibility, custom families, custom algorithms, efficiency, etc.

- **July 2023**,
  IWSM Dortmund, first discussions on how to improve the *gamlss* software.
- **September 2023**,
  started first function for processing arbitrary complex formulae.
- **December 2025**,
  first "proof of concept" of *gamlss2*.

The GAMLSS Journey

The **GAMLSS Working Party**:



What is a Regression Model?

- Typical answer is something like

$$\mathrm{E}(y_i|\boldsymbol{x}_i) = h(\eta(\boldsymbol{x}_i))$$

  and much of recent statistical research has focused on flexibility in specifying the regression predictor $\eta(\boldsymbol{x}_i)$, e.g.

  - generalized additive models,
  - random effects,
  - spatial effects,
  - etc.

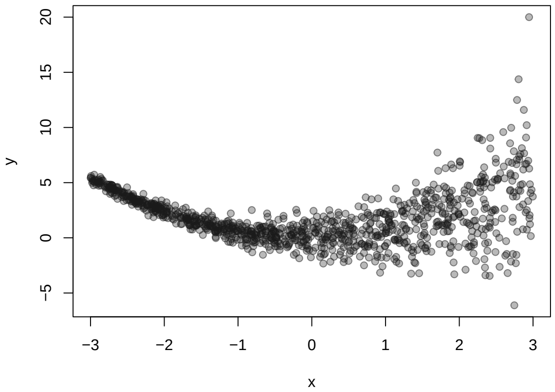
Regression Beyond the Mean

- Classical regression has focused on relating the conditional mean of a response y_i to covariate information x_i for observations $(x_1, y_1), \dots, (x_n, y_n)$.

- Linear model:

$$y_i = \beta_0 + \beta x_i + \varepsilon_i$$

with ε_i i.i.d. $\mathrm{N}(0, \sigma^2)$



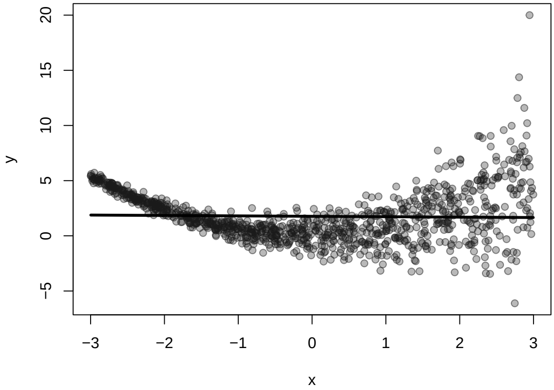
Regression Beyond the Mean

- Classical regression has focused on relating the conditional mean of a response y_i to covariate information x_i for observations $(x_1, y_1), \ldots, (x_n, y_n)$.
- Linear model:

$$y_i = \beta_0 + \beta x_i + \varepsilon_i$$

with ε_i i.i.d. $\mathrm{N}(0, \sigma^2)$

$\Rightarrow \mathrm{E}(y_i | x_i) = \mu_i(x_i) = \beta_0 + \beta x_i$



Regression Beyond the Mean

- Classical regression has focused on relating the conditional mean of a response y_i to covariate information x_i for observations $(x_1, y_1), \ldots, (x_n, y_n)$.

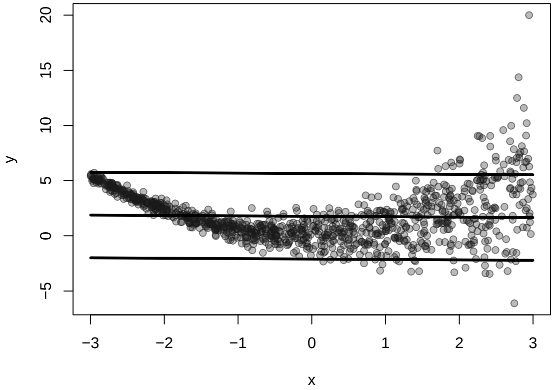
- Linear model:

$$y_i = \beta_0 + \beta x_i + \varepsilon_i$$

with ε_i i.i.d. $\mathrm{N}(0, \sigma^2)$

$\Rightarrow \; \mathrm{E}(y_i | x_i) = \mu_i(x_i) = \beta_0 + \beta x_i$

$\Rightarrow \; \mathrm{Var}(y_i | x_i) = \sigma^2$



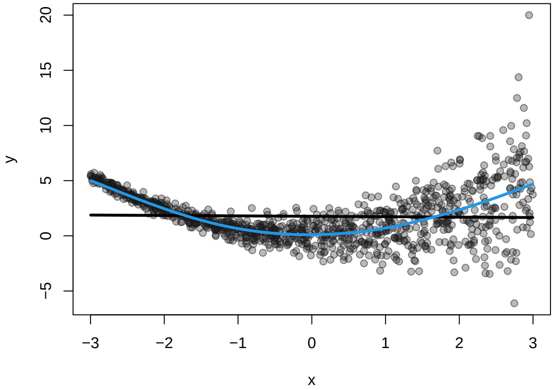
Regression Beyond the Mean

- Classical regression has focused on relating the conditional mean of a response y_i to covariate information x_i for observations $(x_1, y_1), \ldots, (x_n, y_n)$.

- Nonparametric model

$$\mathrm{E}(y_i|x_i) = \mu_i(x_i) = \beta_0 + f(x_i)$$

with $y_i|x_i \sim \mathrm{N}\left(\mu_i(x_i), \sigma^2\right)$



Regression Beyond the Mean

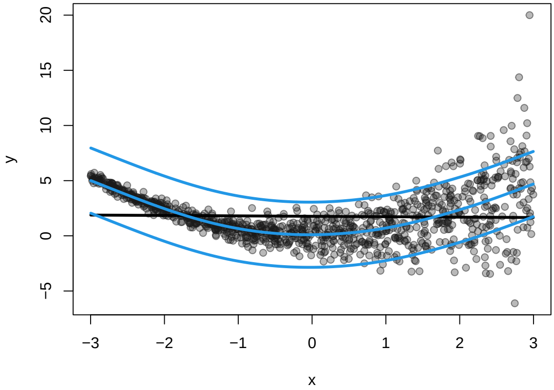
- Classical regression has focused on relating the conditional mean of a response y_i to covariate information x_i for observations $(x_1, y_1), \ldots, (x_n, y_n)$.

- Nonparametric model

$$\mathrm{E}(y_i|x_i) = \mu_i(x_i) = \beta_0 + f(x_i)$$

with $y_i|x_i \sim \mathrm{N}\left(\mu_i(x_i), \sigma^2\right)$

σ^2 fixed.



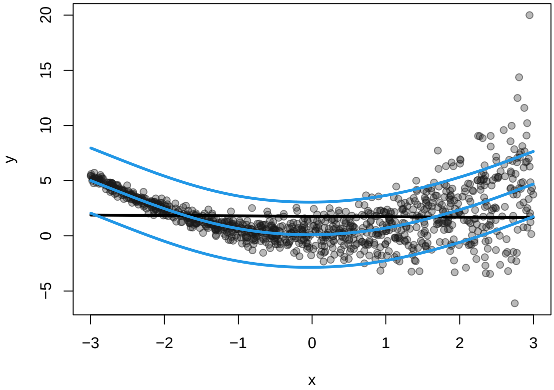
Regression Beyond the Mean

- Nonparametric model for location and scale.

- $\text{E}(y_i|x_i) = \mu_i(x_i) = \beta_{0,\mu} + f_\mu(x_i)$

  $$\begin{aligned}\text{Var}(y_i|x_i) &= \sigma_i^2(x_i) \\ &= \exp\left(\beta_{0,\sigma^2} + f_{\sigma^2}(x_i)\right)\end{aligned}$$

  with $y_i|x_i \sim \text{N}\left(\mu_i(x_i), \sigma_i^2(x_i)\right)$.



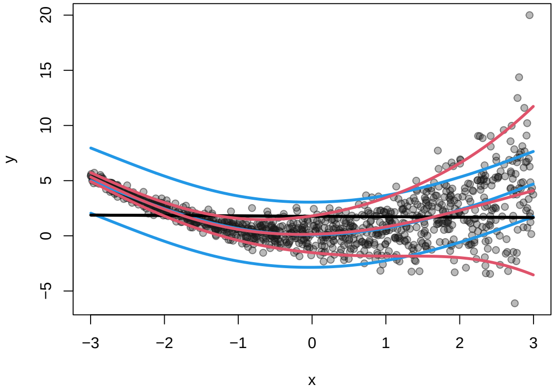
Regression Beyond the Mean

- Nonparametric model for location and scale.

- $\text{E}(y_i|x_i) = \mu_i(x_i) = \beta_{0,\mu} + f_\mu(x_i)$

  $$\begin{aligned}\text{Var}(y_i|x_i) &= \sigma_i^2(x_i) \\ &= \exp\left(\beta_{0,\sigma^2} + f_{\sigma^2}(x_i)\right)\end{aligned}$$

  with $y_i|x_i \sim \text{N}\left(\mu_i(x_i), \sigma_i^2(x_i)\right)$.



Model Specification

Any parameter of a population distribution $\mathcal{D}$ may be modeled by explanatory variables

$$y \sim \mathcal{D}\left(\theta_1(\mathbf{x}; \boldsymbol{\beta}_1), \ \ldots, \ \theta_K(\mathbf{x}; \boldsymbol{\beta}_K)\right),$$

with $\boldsymbol{\beta} = (\boldsymbol{\beta}_1^\top, \ldots, \boldsymbol{\beta}_K^\top)^\top$.

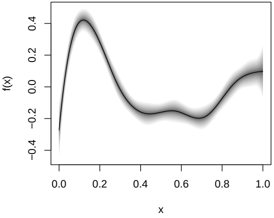
Each parameter is linked to a structured additive predictor

$$h_k(\theta_k(\mathbf{x}; \boldsymbol{\beta}_k)) = f_{1k}(\mathbf{x}; \boldsymbol{\beta}_{1k}) + \ldots + f_{J_k k}(\mathbf{x}; \boldsymbol{\beta}_{J_k k}),$$

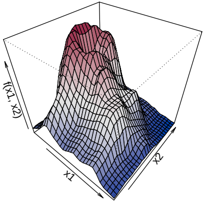
- $j = 1, \ldots, J_k$ and $k = 1, \ldots, K$.
- $h_k(\cdot)$: Link functions for each distribution parameter.
- $f_{jk}(\cdot)$: Model terms of one or more variables.

Model Terms $f_{jk}(\cdot)$

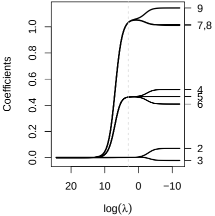
Nonlinear Effects



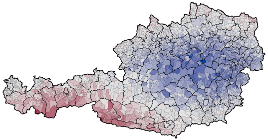
Two−Dimensional Surfaces



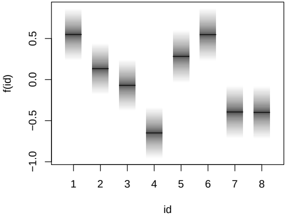
LASSO & Factor Clustering



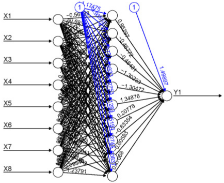
Spatially Correlated Effects f(x) = f(s)



Random Intercepts f(x) = f(id)



Neural Networks



Estimation

The main building block of regression model algorithms is the probability density function $d_y(\mathbf{y}|\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K)$.

Estimation typically requires to evaluate the log-likelihood

$$\ell(\boldsymbol{\beta}; \mathbf{y}, \mathbf{X}) \;=\; \sum_{i=1}^{n} \log\, d_y(y_i; \theta_1(\mathbf{x}_i; \boldsymbol{\beta}_1),\; \ldots\;,\; \theta_K(\mathbf{x}_i; \boldsymbol{\beta}_K)),$$

with $\mathbf{X} = (\mathbf{X}_1, \ldots, \mathbf{X}_K)$.

The log-posterior (frequentist penalized log-likelihood)

$$\log\, \pi(\boldsymbol{\beta}, \boldsymbol{\tau}; \mathbf{y}, \mathbf{X}, \boldsymbol{\alpha}) \propto \ell(\boldsymbol{\beta}; \mathbf{y}, \mathbf{X}) + \sum_{k=1}^{K} \sum_{j=1}^{J_k} \left[\log\, p_{jk}(\boldsymbol{\beta}_{jk}; \boldsymbol{\tau}_{jk}, \boldsymbol{\alpha}_{jk})\right],$$

where $p_{jk}(\cdot)$ are priors, $\boldsymbol{\tau}_{jk}$ (smoothing) variances and $\boldsymbol{\alpha}_{jk}$ fixed hyper parameters.

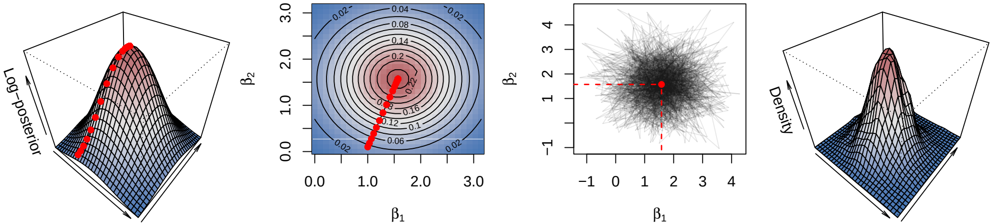
Estimation

Bayesian point estimates of parameters are obtained by:

1. Maximization of the log-posterior for posterior mode estimation.
2. Solving high dimensional integrals, e.g., for posterior mean or median estimation.

Problems 1 and 2 are commonly solved by computer intensive iterative algorithms of the following type:

$$(\boldsymbol{\beta}^{[t+1]}, \boldsymbol{\tau}^{[t+1]}) = \mathbb{U}(\boldsymbol{\beta}^{[t]}, \boldsymbol{\tau}^{[t]}; \mathbf{y}, \mathbf{X}, \boldsymbol{\alpha}).$$



Estimation

Consider the following updating scheme

$$\beta_k^{[t+1]} = \mathbb{U}_k(\beta_k^{[t]}; \cdot) = \beta_k^{[t]} - \mathbf{H}_{kk}\left(\beta_k^{[t]}\right)^{-1} \mathbf{s}\left(\beta_k^{[t]}\right).$$

Assuming model terms that can be written as a matrix product of a design matrix and coefficients we obtain an iteratively weighted least squares scheme given by

$$\beta_{jk}^{[t+1]} = \mathbb{U}_{jk}(\beta_{jk}^{[t]}; \cdot) = (\mathbf{X}_{jk}^\top \mathbf{W}_{kk} \mathbf{X}_{jk} + \mathbf{G}_{jk}(\boldsymbol{\tau}_{jk}))^{-1} \mathbf{X}_{jk}^\top \mathbf{W}_{kk}(\mathbf{z}_k - \boldsymbol{\eta}_{k,-j}^{[t+1]}),$$

with working observations $\mathbf{z}_k = \boldsymbol{\eta}_k^{[t]} + \mathbf{W}_{kk}^{-1\,[t]} \mathbf{u}_k^{[t]}$, working weights $\mathbf{W}_{kk}^{-1\,[t]}$ and score vector $\mathbf{u}_k^{[t]}$.

This is essentially the RS backfitting algorithm.

Installation

The *gamlss2* package is provided at

```
https://github.com/gamlss-dev/gamlss2
```

Install with:

```r
install.packages("gamlss2",
  repos = c("https://gamlss-dev.R-universe.dev",
            "https://cloud.R-project.org"))
```

CRAN version coming soon!

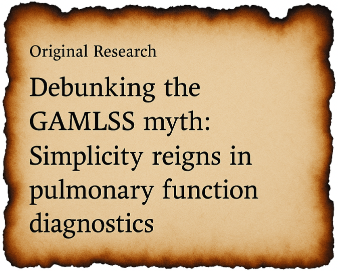
Main Function

```
gamlss2(formula, data, family = NO,
          subset, na.action, weights, offset, start = NULL,
          control = gamlss2_control(...), ...)
```

- **Modular design** facilitating easy development of new features.
- Use of ***Formula*** package for **arbitrary complex formulae**.
- Use of ***mgcv*** `smooth.construct()`.
- Easy development of **new families**, `?gamlss2.family`.
- **New special model term** design, `?special_terms`.
- Model fitting function is **exchangeable**, `?gamlss2_control`, `?RS`.
- Support for **large data** sets, argument `binning = TRUE`.
- **Improved** `predict()` method.
- Use of ***topmodels*** package for post-estimation results.

Spirometry Data

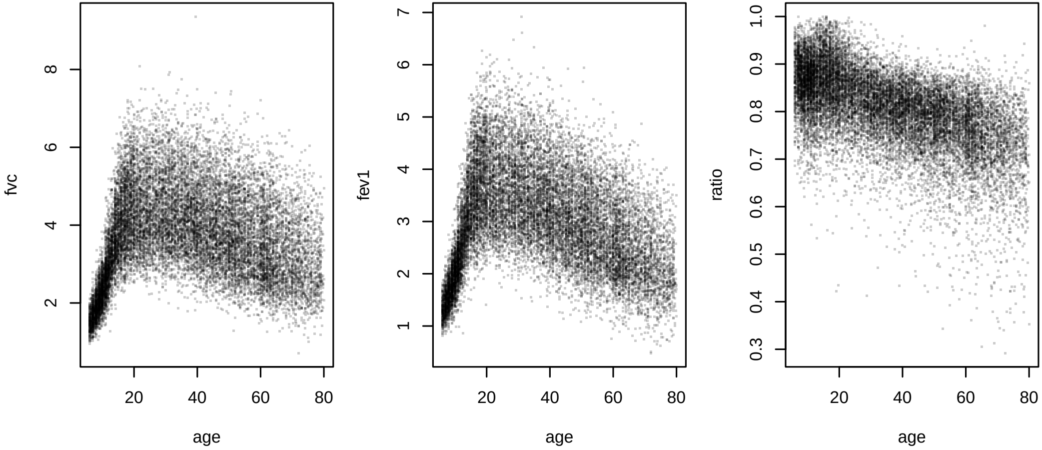
- Paper published in 2025.
- "Simple models match GAMLSS in accuracy but are easier to use".
- Spirometry measurements from NHANES 2007-2012, $n = 16596$.




```
R> data("SpirometryUS", package = "gamlss2")
R> head(SpirometryUS, 3)
        fvc  fev1     ratio   pef   fef volume  fet gender   age weight height
41475 2.473 2.025 0.8188435 5.755 2.119     94  8.7 female 62.50  138.9  154.7
41476 1.742 1.551 0.8903559 3.635 2.122     50  5.4 female  6.75   22.0  120.4
41479 3.545 2.957 0.8341326 8.932 4.081    100 10.5   male 52.50   65.7  154.4
        bmi ethnicity
41475 58.04     other
41476 15.18     other
41479 27.56   mexican
```

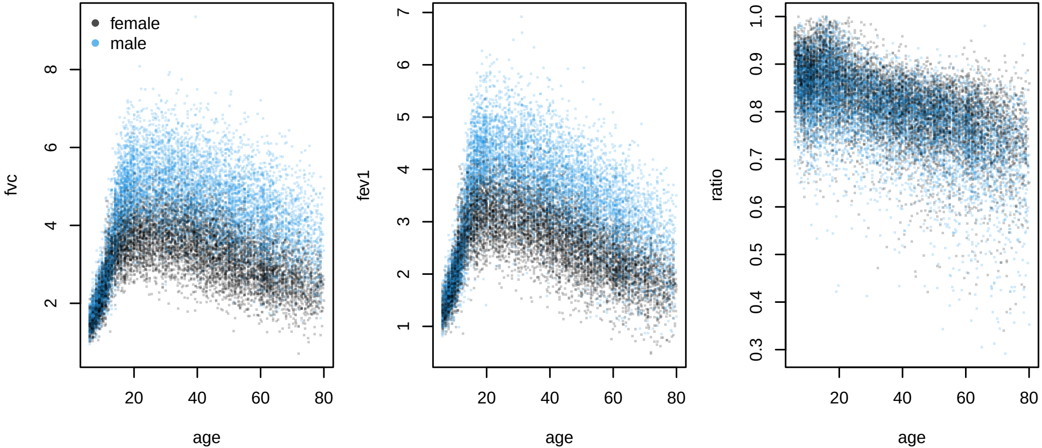
Spirometry Data

Spirometry Measurements from NHANES 2007-2012.



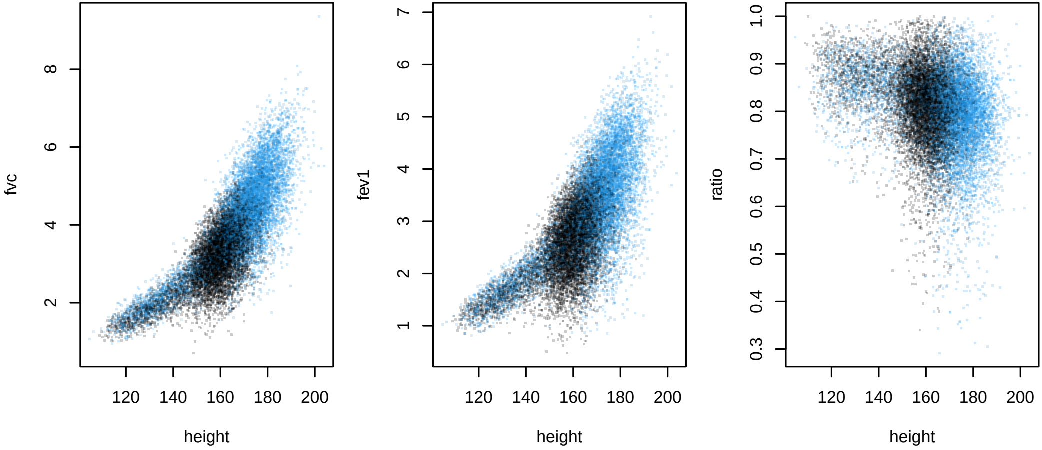
Spirometry Data

Spirometry Measurements from NHANES 2007-2012.



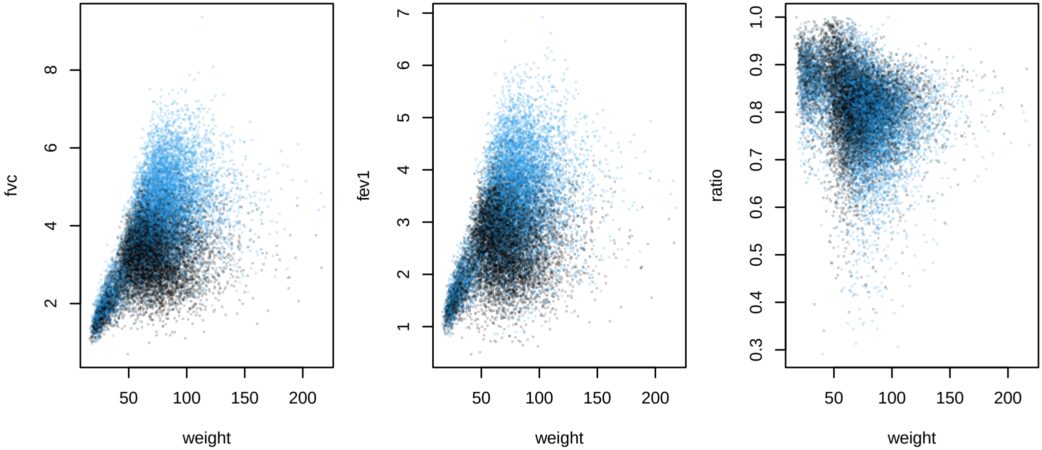
Spirometry Data

Spirometry Measurements from NHANES 2007-2012.



Spirometry Data

```
d <- subset(SpirometryUS, gender == "female")
```



Families

Simplified "`gamlss2.family`" class, inspired by *bamlss* and *distributions3*.

```
R> ND <- function(...) {
+    rval <- list(
+      "family" = "Normal Distribution",
+      "names" = c("mu", "sigma"),
+      "links" = c(mu = "identity", sigma = "log"),
+      "pdf" = function(y, par, log = FALSE) {
+        dnorm(y, mean = par$mu, sd = par$sigma, log = log) },
+      "cdf" = function(y, par, ...) {
+        pnorm(y, mean = par$mu, sd = par$sigma, ...) },
+      "quantile" = function(p, par) {
+        qnorm(p, mean = par$mu, sd = par$sigma) }
+    )
+    class(rval) <- "gamlss2.family"
+    rval
+  }
```

Additionally: `random()`, `mean()`, `variance()`, `z_weights(y, eta, peta, j)`, `...`

Estimation

Model formula.

```
R> f <- fev1 ~ s(age) | s(age)
R> f <- fev1 ~ s(age) + s(height) + s(weight) | s(age)
R> f <- fev1 ~ s(age) + s(height) + s(weight) | . | 1
R> f <- fev1 ~ s(age) + s(height) + s(weight) + te(height, weight) | .
```

Estimation using the new family.

```
m1 <- gamlss2(f, family = ND, data = d)
GAMLSS-RS iteration 15: Global Deviance = 7101.8205 eps = 0.000008
```

Faster with *gamlss.dist* family `NO()`.

```
m2 <- gamlss2(f, family = NO, data = d)
GAMLSS-RS iteration 12: Global Deviance = 7083.5112 eps = 0.000008
```

Finding a Model

Several options for finding a model:

- Set `ridge = TRUE` in `gamlss2()`, for ridge penalty for linear effects.
- Special model term `la()` for Lasso penalized terms.
- Stepwise model term selection `step_gamlss2()`, `stepwise()`.
- Double shrinkage penalty `select_gamlss2()`, `sRS()`.
- Find distribution and model terms with `find_gamlss2()`.

Search using selected distributions implemented in *gamlss.dist*.

```
R> f <- fev1 ~ s(age) + s(height) + s(weight) + te(age, weight) | . | . | .
R> m <- find_gamlss2(f, data = d, select = TRUE, k = log(nrow(d)),
+    families = c("NO", "JSU", "BCPE"), mu.link = "log")
```

Finding a Model

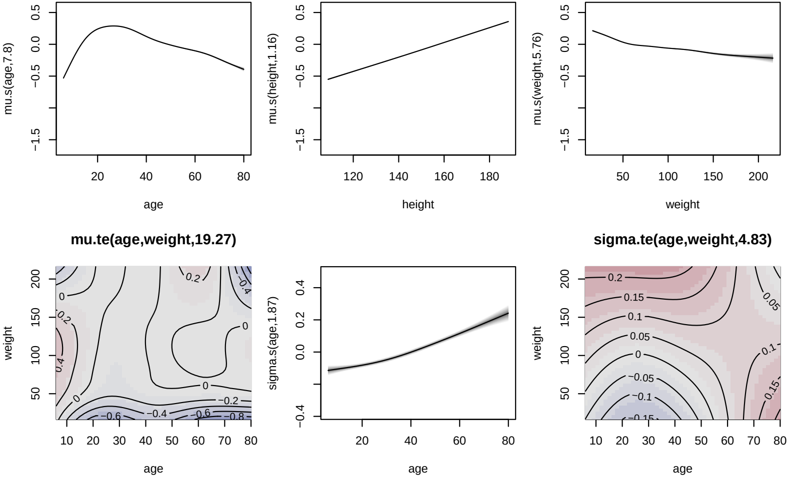
```
summary(m)
Call:
gamlss2(formula = formula, data = ..1, family = families[[j]],
    ... = pairlist(trace = FALSE, mu.link = "log", criterion = "BIC",
    criterion_refit = "BIC", select = TRUE, thres = c(0.9, 0.2), optimizer = sRS))
---
Family: BCPE
Link function: mu = log, sigma = log, nu = identity, tau = log
*--------
Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
mu.(Intercept)     0.914677   0.001854  493.32   <2e-16 ***
sigma.(Intercept) -1.928581   0.005623 -343.00   <2e-16 ***
nu.(Intercept)     1.470509   0.033103   44.42   <2e-16 ***
tau.(Intercept)    0.649488   0.013509   48.08   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Finding a Model

```
---
Smooth terms:
    mu.s(age) mu.s(height) mu.s(weight) mu.te(age,weight) sigma.s(age)
edf    7.7968       1.1599       5.7635           19.2724       1.8732
    sigma.te(age,weight)
edf                4.826
*--------
n = 8303 df =  44.69 res.df =  8258.31
Deviance = 6861.7347 Null Dev. Red. = 60.39%
AIC = 6951.1185 elapsed = 34.94sec
```

Estimated Effects

```
R> plot(m)
```



Diagnostics

Support of the *topmodels* package for probabilistic model assessment.

```
R> library("topmodels")
```

Q-Q plots for quantile residuals.

```
R> qqrplot(m)
```

Probability integral transform (PIT) histograms.

```
R> pithist(m)
```

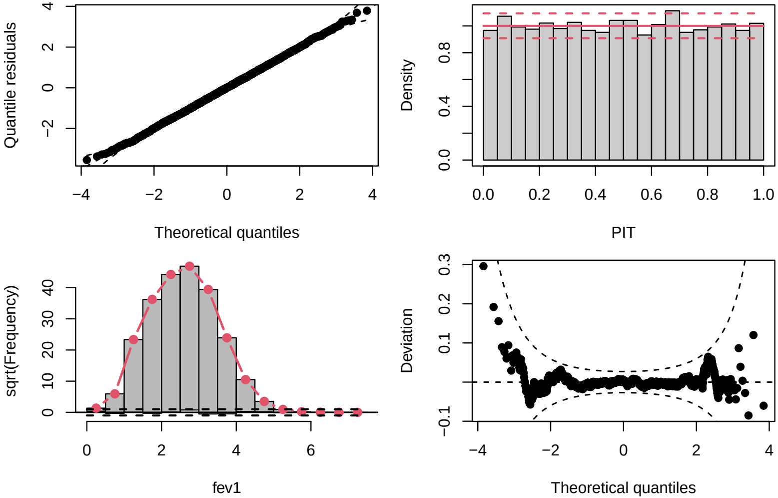
Rootogram, compare (square roots) of empirical frequencies with expected (fitted) frequencies.

```
R> rootogram(m)
```

Worm plots using quantile residuals.

```
R> wormplot(m)
```

Diagnostics



Special Model Terms

Several special model terms implemented:

- Smooth terms from *mgcv*: `s()`, `te()`, `ti()`.
- Smooth terms from *gamlss*: `pb()`, etc.
- Lasso penalized terms: `la()`, `plot_lasso()`.
- Neural networks from *nnet*: `n()`.
- Trees and forests from *partykit*: `tree()`, `cf()`.
- Loess smoothing: `lo()`.

New specials can be implemented with (see `special_terms`):

- Generic fitting method: `special_fit(x, ...)`.
- Generic predict method: `special_predict(x, ...)`.

Special Model Terms

Example: Segmented regression (linear B-spline with free knots).

```
R> fk <- function(x, ...) {
+    sx <- list()
+    ## ...
+    class(sx) <- c("special", "fk")
+    return(sx)
+  }
R> special_fit.fk <- function(x, ...) {
+    sx <- list()
+    ## ...
+    class(sx) <- "fk.fitted"
+    return(sx)
+  }
R> special_predict.fk.fitted <- function(x, ...) {
+    ## ...
+    return(p)
+  }
```

Special Model Terms

In action:

```
f <- fev1 ~ I(age^2) + I(height^2) + weight + I(age * height^2) + fk(age, k = 3)
b <- gamlss2(f, data = d, family = NO)
```

```
GAMLSS-RS iteration  3: Global Deviance = 8016.1594 eps = 0.000001
```

Compare:

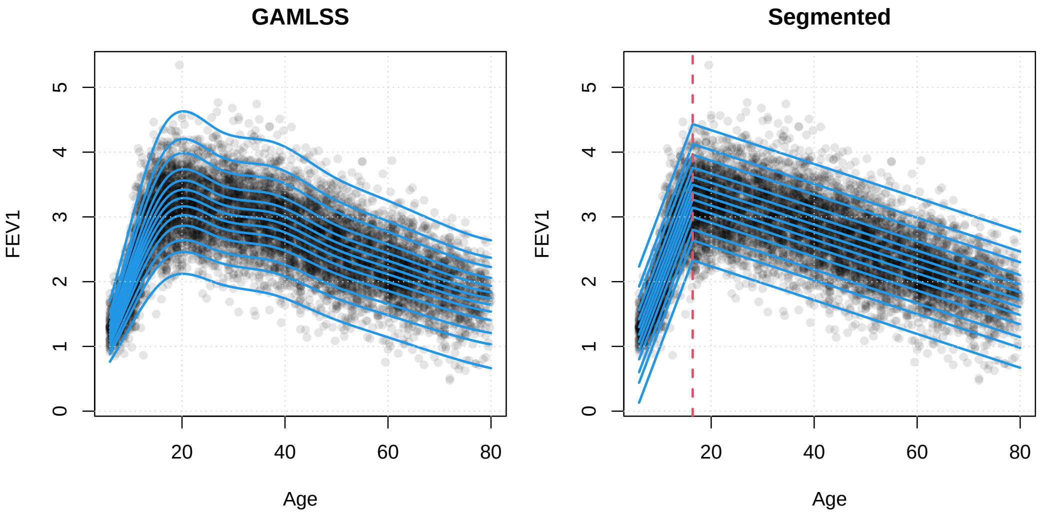
```
R> AIC(m1, m2, m, b)
```

```
        AIC       df
m  6951.119 44.69194
m2 7175.344 45.91658
m1 7207.094 52.63670
b  8034.159  9.00000
```

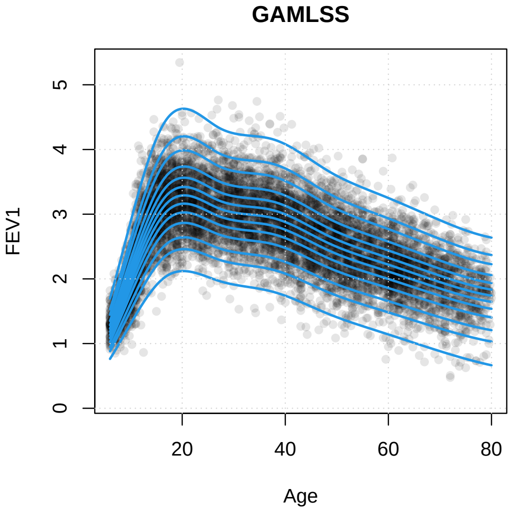
Predictions

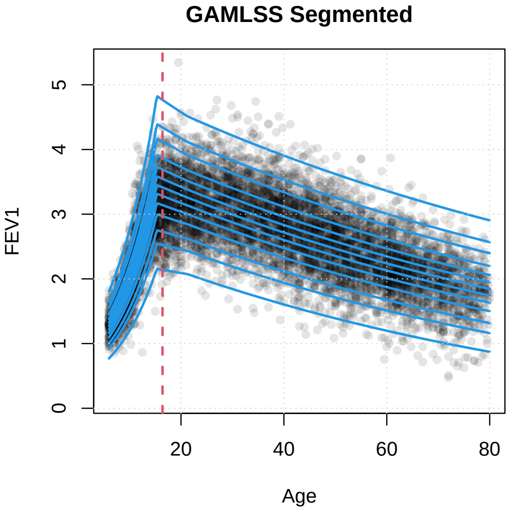
```
R> nd <- data.frame(
+    "age" = seq(6, 80, by = 1),
+    "height" = 140,
+    "weight" = 40
+  )
R> par <- predict(m, newdata = nd)
R> head(par, 4)
        mu     sigma       nu     tau
1 1.666610 0.1272847 1.470509 1.91456
2 1.734938 0.1266004 1.470509 1.91456
3 1.805685 0.1259245 1.470509 1.91456
4 1.877806 0.1252619 1.470509 1.91456
R> q5 <- family(m)$quantile(0.05, par)
R> head(q5)
[1] 1.297166 1.352547 1.409959 1.468576 1.527030 1.583589
R> qu <- quantile(m, newdata = nd, probs = seq(0.01, 0.99, by = 0.01))
```

Predictions



Predictions





Lasso

Lasso model terms are implemented via `la()`:

`la(x, type = 1, const = 1e-05, ...)`

- `type = 1`: simple lasso penalty.
- `type = 2`: group lasso penalty.
- `type = 3`: ordinal fusion penalty.
- `type = 4`: nominal fusion penalty.

Example: Fused LASSO, create truncated polynom design matrix.

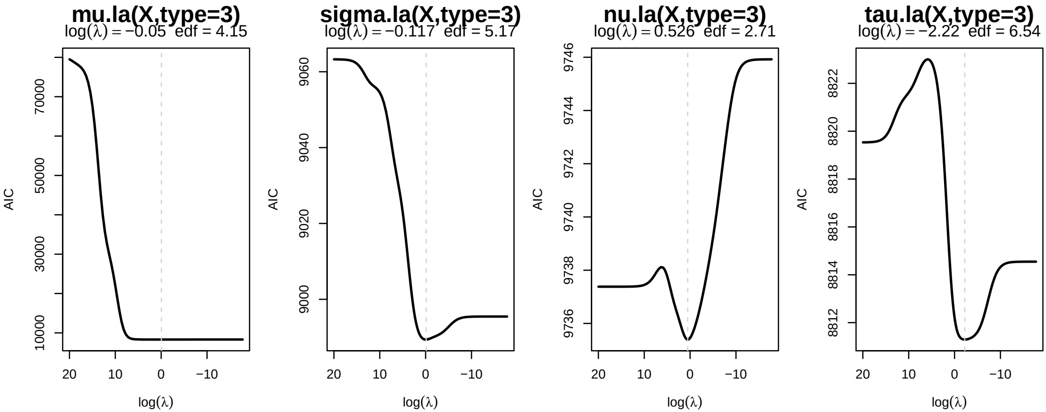
```
R> l <- 1
R> d$sage <- as.numeric(scale(d$age))
R> K <- seq(min(d$sage), max(d$sage), length = 10)
R> d$X <- pmax(outer(d$sage, K, "-"), 0)^l
```

Estimate GAMLSS Lasso model.

```
R> m4 <- gamlss2(fev1 ~ sage + la(X, type = 3) | . | . | ., data = d,
+    family = BCPE(mu.link = "log"), criterion = "AIC", add_ridge = TRUE)
```

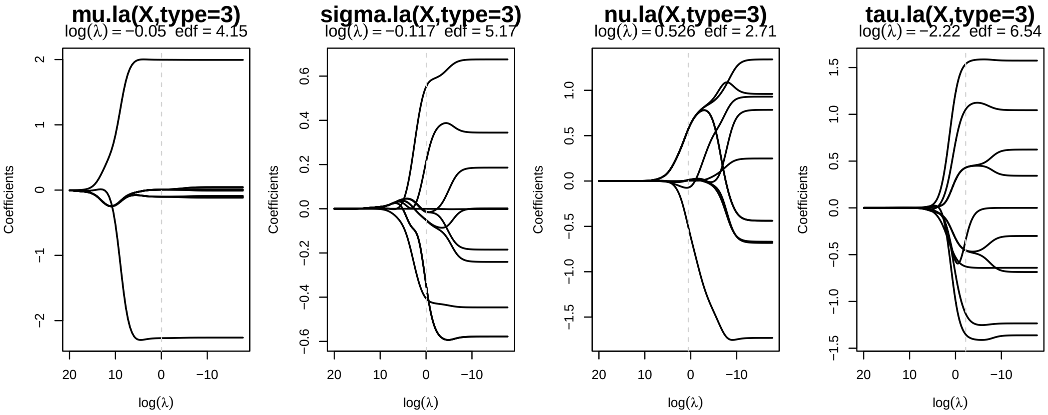
Lasso

Visualize using `plot_lasso(..., which = "criterion")`.

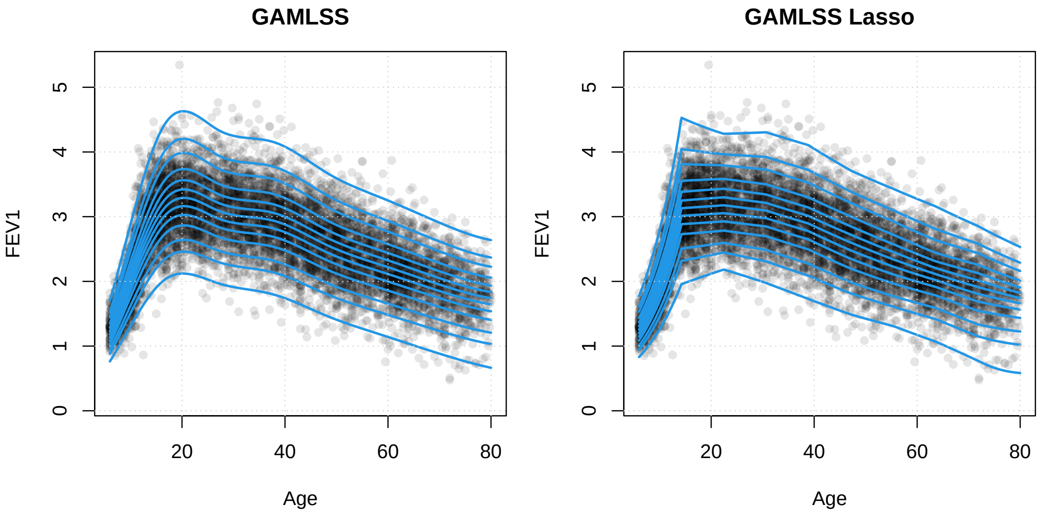


Lasso

Visualize using `plot_lasso(..., which = "coefficients")`.



Lasso



Summary

- *gamlss2*: modular framework for flexible distributional regression.
- Supports smooth terms, lasso, custom families, large datasets, a.o.
- Example: Spirometry data - model selection, estimation, diagnostics, prediction.
- Exchangeable estimation engines and integration with *topmodels* for post-estimation tools.

Thank you! Questions welcome.