Distributional Modelling in R

Smooth Additive Terms

Thomas Kneib, Nikolaus Umlauf

https://nikum.org/dmr.html

Additive Predictors

- Each of the regression predictors η_{ik} in a distributional regression model is additively composed of an intercept $\gamma_0^{\theta_k}$ and a sum of J_k functions $s_j^{\theta_k}(\boldsymbol{x}_i)$:

$$\eta_{ik} = \gamma_0^{\theta_k} + s_1^{\theta_k}(\boldsymbol{x}_i) + \ldots + s_j^{\theta_k}(\boldsymbol{x}_i) + \ldots + s_{J_k}^{\theta_k}(\boldsymbol{x}_i).$$

- More compact form:

$$\eta_{ik} = \gamma_{0k} + s_{1k}(\boldsymbol{x}_i) + \ldots + s_{jk}(\boldsymbol{x}_i) + \ldots + s_{J_k k}(\boldsymbol{x}_i).$$

- The functions $s_j^{\theta_k}(\boldsymbol{x}_i)$ represent a variety of different effects.

Basis Function Expansions

- Dropping the indices j and k, each function is expanded in a basis as

$$s(\boldsymbol{x}_i) = \sum_{l=1}^{L} \gamma_l B_l(\boldsymbol{x}_i)$$

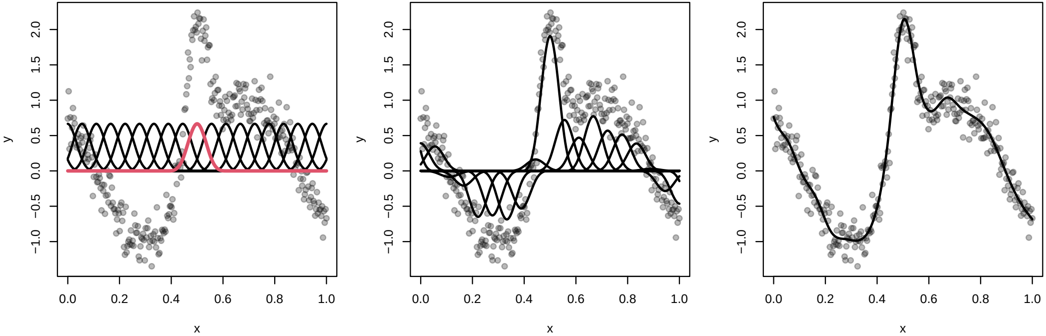
  where γ_l are the basis amplitudes and $B_l(\boldsymbol{x}_i)$ represent different types of basis functions.
- In matrix notation, each of the predictors can then be written for all observations as

$$\boldsymbol{\eta} = \gamma_0 \mathbf{1}_n + \boldsymbol{B}_1 \boldsymbol{\gamma}_1 + \ldots + \boldsymbol{B}_J \boldsymbol{\gamma}_J.$$

B-Splines

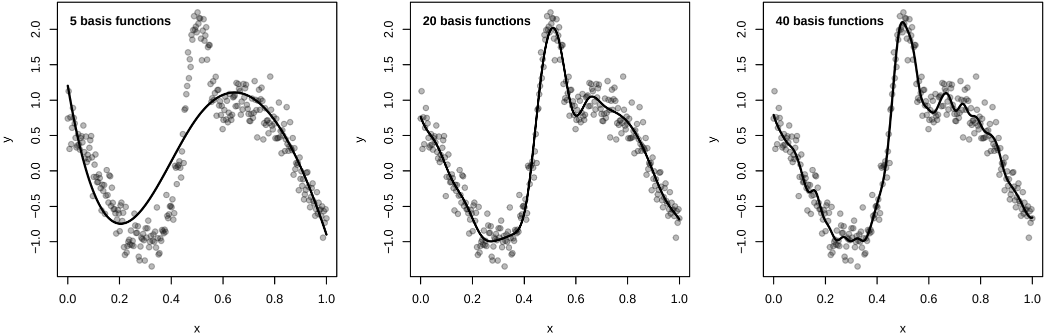
- Approximate functions $s(x)$ based on a linear combination of B-spline basis functions:

$$s(x) = \sum_{l=1}^{L} \gamma_l B_l(x)$$



Influence of the Number of Knots

- Estimated B-splines for a varying number of knots:



Regularization

- Unregularized estimates crucially depend on the number of basis functions.

  $\Rightarrow$ Add a regularization term that penalizes function estimates with high variability.

- Effectively leads to a trade-off between fidelity to the data and smoothness.
- Popular approach (e.g. for smoothing splines): Use a penalty based on the integrated squared second derivative:

$$\text{pen}(s) = \lambda \int (s''(x))^2 dx.$$

Difference Penalty

- Easy approximation for B-splines in terms of difference penalties, e.g. in case of first order differences

$$\text{pen}(\gamma) = \lambda \sum_{l=2}^{L} (\gamma_l - \gamma_{l-1})^2 = \lambda \boldsymbol{\gamma}' \boldsymbol{K} \boldsymbol{\gamma}$$

  with penalty matrix $\boldsymbol{K}$.
- The smoothing parameter λ determines the impact of the penalty and should be estimated jointly with the regression coefficients.

Spatial Effects

- Estimate a separate regression coefficient γ_l for each region $l \in \{1, \ldots, L\}$.
- Instable estimates result if the number of regions L is large compared to the sample size.

  $\Rightarrow$ Regularised estimation, to enforce spatial smoothness, i.e., effects of neighboring regions should be similar.

Spatial Penalty

- Penalty based on squared differences of neighboring regions:

$$\text{pen}(\gamma) = \lambda \sum_{l=1}^{L} \sum_{r \in N(l)} (\gamma_l - \gamma_r)^2$$

  where $N(l)$ denotes the regions of region l.
- In matrix form:

$$\text{pen}(\gamma) = \lambda \gamma' \boldsymbol{K} \gamma$$

  with penalty matrix

$$\boldsymbol{K}[l, r] = \begin{cases} -1 & l \neq r, l \sim r, \\ 0 & l \neq r, l \nsim r, \\ |N(l)| & l = r, \end{cases}$$

Generic Framework – Basis Functions

- Let $\boldsymbol{x}_i$ denote some generic type of covariate information and assume

$$s(\boldsymbol{x}_i) = \sum_{l=1}^{L} \gamma_l B_l(\boldsymbol{x}_i)$$

  with L basis functions $B_l(\boldsymbol{x}_i)$.

- The vector of function evaluations $\boldsymbol{s} = (s(\boldsymbol{x}_1), \ldots, s(\boldsymbol{x}_n))'$ at the observed covariate values is then given by

$$\boldsymbol{s} = \boldsymbol{B}\gamma,$$

  where $\boldsymbol{B}$ is the design matrix obtained from the basis function evaluations and γ is the corresponding vector of basis coefficients.

Generic Framework – Penalty

- To regularize estimation, consider quadratic penalties

$$\text{pen}(\boldsymbol{\gamma}) = \lambda \boldsymbol{\gamma}' \boldsymbol{K} \boldsymbol{\gamma}$$

  with positive semidefinite penalty matrix $\boldsymbol{K}$ and smoothing parameter $\lambda \geq 0$.
- Can also be interpreted as assuming the prior $\boldsymbol{\gamma} \sim \text{N}(\mathbf{0}, \tau^2 \boldsymbol{K}^-)$ as a prior distribution in a Bayesian framework.
- Note: Often $\boldsymbol{K}$ does not have full rank such that $\boldsymbol{K}^-$ is a generalized inverse.

Other Types of Effects

- The generic framework supports a number of further effect types such as

Nonlinear effects of continuous covariates	$s(\mathbf{x}) = s(x_1)$
Two-dimensional surfaces	$s(\mathbf{x}) = s(x_1, x_2)$
Spatially correlated effects	$s(\mathbf{x}) = s_{spat}(x_s)$
Varying coefficients	$s(\mathbf{x}) = x_1 s(x_2)$
Spatially varying effects	$s(\mathbf{x}) = x_1 s_{spat}(x_s)$ or $x_1 s(x_2, x_3)$
Random intercepts with cluster index c	$s(\mathbf{x}) = \beta_c$
Random slopes with cluster index c	$s(\mathbf{x}) = x_1 \beta_c$

Smooth Terms in `mgcv`

- In the *mgcv* package, smooth terms are used to model complex nonlinear relationships between predictors and responses.
- Behind the scene, smooth terms are constructed using the `smooth.construct()` function.
- The `smooth.construct()` function accepts smooth specification objects such as `s()`, `te()`, and/or `ti()`, and utilizes them to generate the associated design and penalty matrices.
- Smooth terms can be used in GAM, GAMM and GAMLSS.

Smooth Terms in `mgcv`

Examples:

```
R> d <- data.frame("x" = seq(0, 1, length = 300))
```

Create smooth term.

```
R> sc <- smooth.construct(s(x, bs = "ps", k = 10), d, knots = NULL)
R> print(names(sc))
 [1] "term"           "bs.dim"         "fixed"          "dim"
 [5] "p.order"        "by"             "label"          "xt"
 [9] "id"             "sp"             "deriv"          "X"
[13] "mono"           "D"              "S"              "rank"
[17] "null.space.dim" "knots"          "m"
```

Smooth Terms in mgcv

Design matrix.

```
R> print(head(sc$X))
          [,1]      [,2]      [,3]         [,4] [,5] [,6] [,7] [,8] [,9] [,10]
[1,] 0.1631980 0.6666180 0.1701839 5.682503e-08    0    0    0    0    0     0
[2,] 0.1519473 0.6657595 0.1822886 4.659653e-06    0    0    0    0    0     0
[3,] 0.1412258 0.6638588 0.1948895 2.583111e-05    0    0    0    0    0     0
[4,] 0.1310210 0.6609543 0.2079483 7.632610e-05    0    0    0    0    0     0
[5,] 0.1213200 0.6570842 0.2214268 1.688995e-04    0    0    0    0    0     0
[6,] 0.1121101 0.6522869 0.2352867 3.163063e-04    0    0    0    0    0     0
```

Smooth Terms in `mgcv`

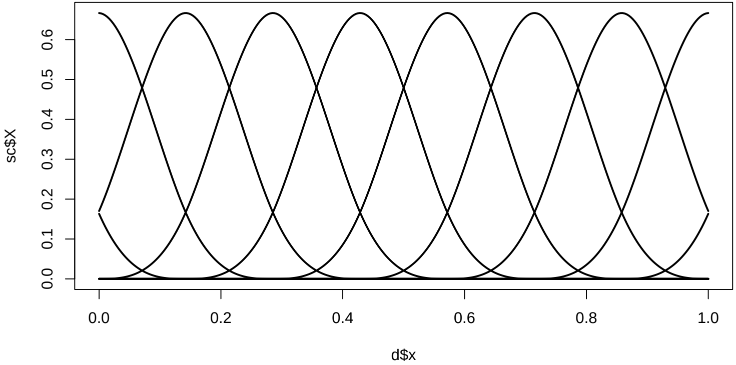
Penalty matrix.

```
R> print(head(sc$S[[1]]))
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
[1,]    1   -2    1    0    0    0    0    0    0     0
[2,]   -2    5   -4    1    0    0    0    0    0     0
[3,]    1   -4    6   -4    1    0    0    0    0     0
[4,]    0    1   -4    6   -4    1    0    0    0     0
[5,]    0    0    1   -4    6   -4    1    0    0     0
[6,]    0    0    0    1   -4    6   -4    1    0     0
```

Smooth Terms in `mgcv`

Plot the design matrix.

```
R> par(mar = c(4, 4, 0.5, 0.5))
R> matplot(d$x, sc$X, type = "l", lty = 1, col = 1, lwd = 2)
```



Smooth Terms in `mgcv`

- Formula syntax:

Linear effects $\mathbf{X}\beta$	`x1 + x2 + x3`
Nonlinear effects $s(\mathbf{x}) = s(x_1)$	`s(x1)`
Interaction surfaces $s(\mathbf{x}) = s(x_1, x_2)$	`s(x1,x2)`, `te(x1,x2)` or `ti(x1,x2)`
Discrete Spatial $s(\mathbf{x}) = s_{\text{spat}}(x_s)$	`s(xs,bs="mrf",xt=list(penalty=K))`
Varying coefficients $s(\mathbf{x}) = x_1 s(x_2)$	`s(x2,by=x1)`
Spatially varying effects $s(\mathbf{x}) = x_1 s_{\text{spat}}(x_s)$,	`s(xs,bs="mrf",xt=list(penalty=K),by=x1)`,
or $s(\mathbf{x}) = x_1 s(x_2, x_3)$	`s(x2,x3,by=x1)` or `te(x2,x3,by=x1)`
Random intercepts with clusters c: $s(\mathbf{x}) = \beta_c$	`s(id,bs="re")`
Random slopes with clusters c: $s(\mathbf{x}) = x_1 \beta_c$	`s(id,x1,bs="re")`

GAMLSS in *mgcv*

Load *mgcv* package.

```
R> library("mgcv")
```

Columbus Ohio crime data.

```
R> data(columb)       ## data frame
R> data(columb.polys) ## district shapes list
```

Neighborhood structure.

```
R> xt <- list(polys = columb.polys)
```

Model formula.

```
R> f <- list(
+    crime ~ s(income) + s(district, bs = "mrf", k = 20, xt = xt),
+          ~ s(income) + s(district, bs = "mrf", k = 20, xt = xt)
+  )
```

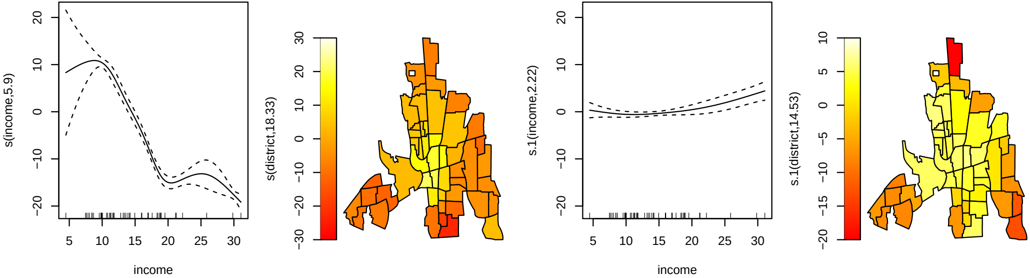
GAMLSS in *mgcv*

Estimate model.

```
R> b <- gam(f, data = columb, family = gaulss)
```

Visualize estimated effects.

```
R> par(mfrow = c(1, 4), mar = c(4, 4, 0.1, 1))
R> plot(b, pages = 0)
```



Spatial *mgcv*

Example:

London fire data.

```
R> data("LondonFire", package = "bamlss")
R> head(LondonFire)
                  coordinates arrivaltime   daytime  fsintens
12279 (-0.09832153, 51.65413)    6.033333 0.1263889  248.6067
12280 (-0.04467665, 51.48959)    3.400000 0.3266667 1646.1975
12281  (-0.1101969, 51.47268)    4.383333 0.4641667 1333.3776
12282  (-0.2448571, 51.45319)    5.800000 1.9297222  300.6900
12283  (-0.1874054, 51.48648)    5.133333 1.9308333 1195.7156
12284  (-0.2893525, 51.61031)    4.966667 3.5480556  319.6787
```

Spatial *mgcv*

Transform to data frame.

```
R> library("sp")
R> d <- as.data.frame(LondonFire)
R> head(d)
```

```
      arrivaltime   daytime  fsintens         lon      lat
12279    6.033333 0.1263889  248.6067 -0.09832153 51.65413
12280    3.400000 0.3266667 1646.1975 -0.04467665 51.48959
12281    4.383333 0.4641667 1333.3776 -0.11019694 51.47268
12282    5.800000 1.9297222  300.6900 -0.24485711 51.45319
12283    5.133333 1.9308333 1195.7156 -0.18740538 51.48648
12284    4.966667 3.5480556  319.6787 -0.28935255 51.61031
```

Spatial *mgcv*

Estimate spatial GAMLSS.

```
R> f <- list(
+    arrivaltime ~ s(daytime, bs = "cc") + s(fsintens) + s(lon, lat, k = 30),
+                ~ s(daytime, bs = "cc") + s(fsintens) + s(lon, lat, k = 30)
+  )

R> b <- gam(f, data = d, family = gaulss)
```

Spatial *mgcv*

Model summary

```
R> summary(b)
```

```
Family: gaulss
Link function: identity logb

Formula:
arrivaltime ~ s(daytime, bs = "cc") + s(fsintens) + s(lon, lat,
    k = 30)
~s(daytime, bs = "cc") + s(fsintens) + s(lon, lat, k = 30)

Parametric coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept)   5.326916   0.024369  218.59   <2e-16 ***
(Intercept).1 0.593937   0.009336   63.62   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Spatial *mgcv*

```
Approximate significance of smooth terms:
                   edf Ref.df  Chi.sq p-value
s(daytime)       5.847  8.000  78.892  <2e-16 ***
s(fsintens)      7.274  8.278 237.433  <2e-16 ***
s(lon,lat)      24.459 27.651 119.965  <2e-16 ***
s.1(daytime)     4.563  8.000  90.006  <2e-16 ***
s.1(fsintens)    2.043  2.601   2.952    0.26
s.1(lon,lat)    24.581 27.742 163.778  <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Deviance explained = 11.3%
-REML =  11933  Scale est. = 1         n = 5838
```

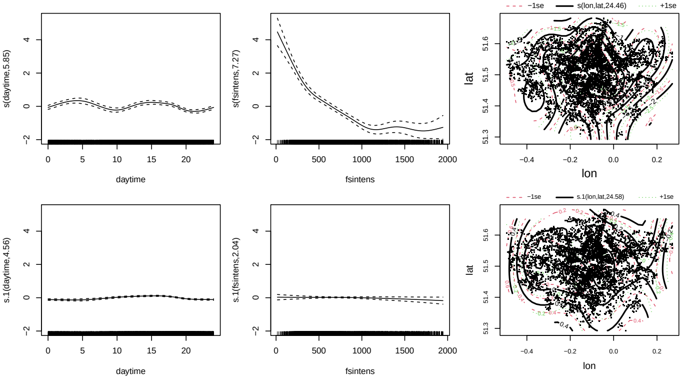
```
R> AIC(b)
```

```
[1] 23718.83
```

Spatial *mgcv*

Visualize effects.

```
R> par(mfrow = c(2, 3), mar = c(4, 4, 2, 1))
R> plot(b, pages = 0)
```



Spatial *gamlss2*

Package *gamlss2* (and *bamlss*) can use the same formulas and smooth terms as *mgcv*.

Estimate model using a gamma distribution.

```
R> library("gamlss2")
R> m <- gamlss2(f, data = d, family = GA)
```

```
GAMLSS-RS iteration  1: Global Deviance = 23060.6225 eps = 0.261198
GAMLSS-RS iteration  2: Global Deviance = 23005.1491 eps = 0.002405
GAMLSS-RS iteration  3: Global Deviance = 22882.9337 eps = 0.005312
GAMLSS-RS iteration  4: Global Deviance = 22868.1671 eps = 0.000645
GAMLSS-RS iteration  5: Global Deviance = 22865.4097 eps = 0.000120
GAMLSS-RS iteration  6: Global Deviance = 22864.6703 eps = 0.000032
GAMLSS-RS iteration  7: Global Deviance = 22864.0841 eps = 0.000025
GAMLSS-RS iteration  8: Global Deviance = 22863.736 eps = 0.000015
GAMLSS-RS iteration  9: Global Deviance = 22863.4106 eps = 0.000014
GAMLSS-RS iteration 10: Global Deviance = 22863.1017 eps = 0.000013
GAMLSS-RS iteration 11: Global Deviance = 22862.8094 eps = 0.000012
```

Spatial *gamlss2*

```
GAMLSS-RS iteration 12: Global Deviance = 22862.5299 eps = 0.000012
GAMLSS-RS iteration 13: Global Deviance = 22862.2649 eps = 0.000011
GAMLSS-RS iteration 14: Global Deviance = 22862.011 eps = 0.000011
GAMLSS-RS iteration 15: Global Deviance = 22861.7701 eps = 0.000010
GAMLSS-RS iteration 16: Global Deviance = 22861.539 eps = 0.000010
GAMLSS-RS iteration 17: Global Deviance = 22861.3194 eps = 0.000009
```

Spatial *gamlss2*

Model summary.

```
R> summary(m)
Call:
gamlss2(formula = arrivaltime ~ s(daytime, bs = "cc") + s(fsintens) +
    s(lon, lat, k = 30) | s(daytime, bs = "cc") + s(fsintens) +
    s(lon, lat, k = 30), data = d, family = GA, ... = pairlist(x = FALSE))
---
Family: GA
Link function: mu = log, sigma = log
*--------
Parameter: mu
---
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 1.667231   0.004408   378.2   <2e-16 ***
---
Smooth terms:
```

Spatial *gamlss2*

```
    s(daytime) s(fsintens) s(lon,lat)
edf     6.5181      7.7094     25.113
*--------
Parameter: sigma
---
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -1.085590   0.008395  -129.3   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
---
Smooth terms:
    s(daytime) s(fsintens) s(lon,lat)
edf     6.8813      7.0257     24.539
*--------
n = 5838 df =  79.79 res.df =  5758.21
Deviance = 22861.3194 Null Dev. Red. = 3.99%
AIC = 23020.8914 elapsed =  1.81sec
```

Spatial *gamlss2*

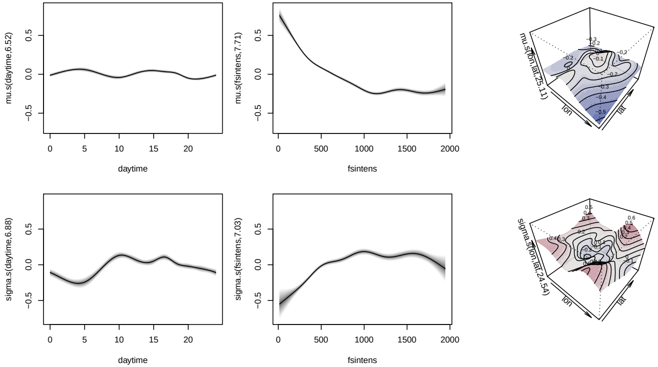
```
R> AIC(m)
[1] 23020.89
```

The gamma model using *gamlss2* performs better than the normal model estimated with *mgcv*.

Spatial *gamlss2*

Visualize effects.

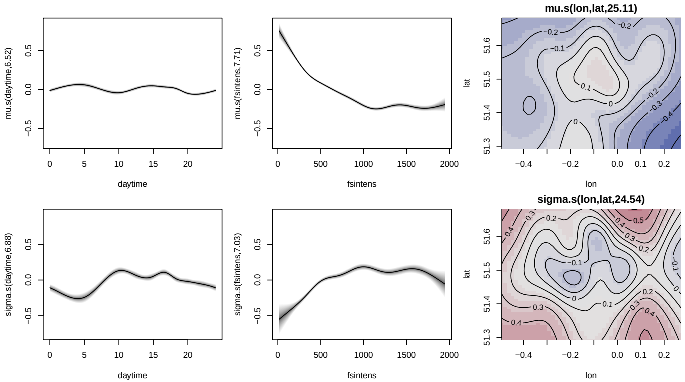
```
R> par(mfrow = c(2, 3), mar = c(4, 4, 2, 1))
R> plot(m, pages = 1, spar = FALSE)
```



Spatial *gamlss2*

Visualize effects.

```
R> par(mfrow = c(2, 3), mar = c(4, 4, 2, 1))
R> plot(m, pages = 1, spar = FALSE, image = TRUE)
```



Spatial *gamlss2*

Mean prediction.

```
R> nd <- data.frame("daytime" = 15, fsintens = 500, lon = 0, lat = 51.5)
R> par <- predict(m, newdata = nd, type = "parameter")
R> print(par)
```

```
       mu     sigma
1 6.77918 0.3033908
```

```
R> mfit <- family(m)$mean(par)
R> print(mfit)
```

```
[1] 6.77918
```

Spatial *gamlss2*

Probabilities & quantiles.

```
R> p3 <- family(m)$p(3, par)
R> print(p3)
```

```
[1] 0.01192371
```

```
R> p6 <- family(m)$p(6, par)
R> print(p6)
```

```
[1] 0.3851141
```

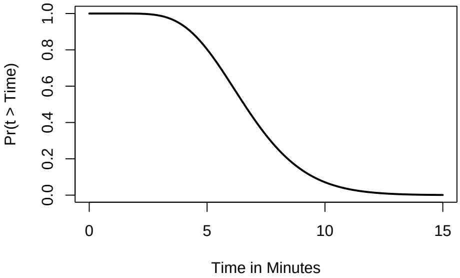
```
R> q5 <- family(m)$q(0.5, par)
R> print(q5)
```

```
[1] 6.572354
```

Spatial *gamlss2*

Full risk plot.

```
R> prob <- rep(NA, 100); atime <- seq(0, 15, length = 100)
R> for(i in 1:100)
+    prob[i] <- 1 - family(m)$p(atime[i], par)
R> par(mar = c(4, 4, 0.5, 0.5))
R> plot(prob ~ atime, type = "l", lwd = 2,
+    xlab = "Time in Minutes", ylab = "Pr(t > Time)")
```



Spatial *gamlss2*

Model using discrete spatial information of London boroughs.

Load *sf* package.

```
R> library("sf")
```

Transform to *sf* map.

```
R> map <- st_as_sf(LondonBoroughs)
R> dim(map)
```

```
[1] 33  1
```

Extract unique coordinates.

```
R> co <- d[, c("lon", "lat")]
R> co <- st_as_sf(co, coords = c("lon", "lat"),
+    crs = st_crs(map))
```

Create polygon indices of London boroughs.

Spatial *gamlss2*

```
R> map$ID <- 1:nrow(map)
R> i <- st_intersects(co, map)
R> d$ID <- map$ID[as.integer(i)]
R> d2 <- na.omit(d)
R> map2 <- subset(map, map$ID %in% d2$ID)
```

Compute penalty matrix of neighboring regions.

```
R> nb <- st_intersects(map2, map2)
R> nbm <- as.matrix(nb) * -1
R> dim(nbm)
```

```
[1] 33 33
```

```
R> diag(nbm) <- apply(nbm, 1, function(x) sum(abs(x)) - 1)
R> print(nbm[4, ])
```

```
 [1]  0  0  0  4 -1  0 -1  0  0  0  0  0  0  0  0  0  0  0  0 -1  0  0 -1  0  0
[26]  0  0  0  0  0  0  0  0
```

```
R> rownames(nbm) <- colnames(nbm) <- map2$ID
```

Spatial *gamlss2*

Model formula.

```
R> d2$ID <- as.factor(d2$ID)
R> f <- ~ s(daytime, bs = "cc") + s(fsintens) +
+    s(ID, bs = "mrf", xt = list("penalty" = nbm))
R> f <- rep(list(f), 2)
R> f[[1]] <- update(f[[1]], arrivaltime ~ .)
R> print(f)
```

```
[[1]]
arrivaltime ~ s(daytime, bs = "cc") + s(fsintens) + s(ID, bs = "mrf",
    xt = list(penalty = nbm))

[[2]]
~s(daytime, bs = "cc") + s(fsintens) + s(ID, bs = "mrf", xt = list(penalty = nbm))
```

Spatial *gamlss2*

Estimate model using MRF.

```
R> b <- gamlss2(f, data = d2, family = GA)
```

```
GAMLSS-RS iteration  1: Global Deviance = 23009.3995 eps = 0.261719
GAMLSS-RS iteration  2: Global Deviance = 22827.8016 eps = 0.007892
GAMLSS-RS iteration  3: Global Deviance = 22825.7045 eps = 0.000091
GAMLSS-RS iteration  4: Global Deviance = 22825.5981 eps = 0.000004
```

```
R> AIC(b)
```

```
[1] 22992.52
```

Predict spatial effect for mu parameter.

```
R> nd <- unique(d2[, "ID", drop = FALSE])
R> nd$daytime <- 15
R> nd$fsintens <- 500
R> nd$fit <- predict(b, newdata = nd, model = "mu",
+    type = "terms", terms = "s(ID)")
R> nd <- nd[order(nd$ID), ]
R> map2$fit <- nd$fit
```

Spatial *gamlss2*

Visualize.

```
R> library("ggplot2")
R> library("viridis")
R> ggplot(map2) + geom_sf(aes(fill = fit)) +
+     scale_fill_viridis(option = "plasma") + theme_bw()
```