aneur: comparando stan y cgam

1. Aortic Aneurysm Progression Data

This dataset contains longitudinal measurements of grades of aortic aneurysms, measured by ultrasound examination of the diameter of the aorta.

A data frame containing 4337 rows, with each row corresponding to an ultrasound scan from one of 838 men over 65 years of age.

- ptnum (numeric) Patient identification number
- age (numeric) Recipient age at examination (years)
- diam (numeric) Aortic diameter
- state (numeric) State of aneurysm.

The states represent successive degrees of aneurysm severity, as indicated by the aortic diameter.

- State 1 Aneurysm-free < 30 cm
- State 2 Mild aneurysm 30-44 cm
- State 3 Moderate aneurysm 45-54 cm
- State 4 Severe aneurysm > 55 cm

683 of these men were aneurysm-free at age 65 and were re-screened every two years. The remaining men were aneurysmal at entry and had successive screens with frequency depending on the state of the aneurysm. Severe aneurysms are repaired by surgery.

```
data(aneur)
attach(aneur)
head(aneur)
```

```
##   ptnum      age diam state
## 1     1 60.00000   29     1
## 2     1 65.47671   29     1
## 3     1 67.50411   29     1
## 4     1 70.04384   29     1
## 5     1 72.07671   29     1
## 6     1 74.08767   29     1
```

```
tail(aneur)
```

```
##      ptnum      age diam state
## 4332   838 73.40822   43     2
## 4333   838 73.61644   43     2
## 4334   838 73.87671   42     2
## 4335   838 74.05753   43     2
## 4336   838 74.31507   41     2
## 4337   838 74.56712   40     2
```

```
#help(aneur)
dim(aneur)
```

```
## [1] 4337    4
```

```
(N = n_distinct(aneur$ptnum))   # subjects
```

```
## [1] 838
```

```
(K = max(table(aneur$ptnum)))   # times
```

```
## [1] 21
```

```
table(table(aneur$ptnum))
```

```
##
##   2   3   4   5   6   7   8   9  10  11  12  14  15  16  17  18  19  21
## 121 107  99  96 260  97  12  12   9   5   2   5   5   3   1   2   1   1
```

```
J = 4   # categories
Y_diam = array(NA,dim=c(N,K))
Y_state = array(NA,dim=c(N,K))
X_age = array(NA,dim=c(N,K))
Ki = table(aneur$ptnum)
Ni = c(0,cumsum(Ki))+1
for(i in 1:N){
    aneur_i = aneur[aneur$ptnum==i,]
    for(k in 1:Ki[i]){
        Y_diam[i,k] = aneur_i$diam[k]
        Y_state[i,k] = aneur_i$state[k]
        X_age[i,k] = aneur_i$age[k]
    }
}
```

```
(Y_diam[11:18,1:8])
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
## [1,]   29   29   29   29   29   29   29   NA
## [2,]   29   29   29   29   29   29   29   NA
## [3,]   29   29   29   29   29   29   29   NA
## [4,]   29   29   NA   NA   NA   NA   NA   NA
## [5,]   29   29   29   29   29   29   29   NA
## [6,]   29   29   29   29   29   29   29   NA
## [7,]   29   29   29   29   NA   NA   NA   NA
## [8,]   29   29   34   NA   NA   NA   NA   NA
```

```
(Y_state[11:18,1:8])
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
## [1,]    1    1    1    1    1    1    1   NA
## [2,]    1    1    1    1    1    1    1   NA
## [3,]    1    1    1    1    1    1    1   NA
## [4,]    1    1   NA   NA   NA   NA   NA   NA
## [5,]    1    1    1    1    1    1    1   NA
## [6,]    1    1    1    1    1    1    1   NA
## [7,]    1    1    1    1   NA   NA   NA   NA
## [8,]    1    1    2   NA   NA   NA   NA   NA
```

```
(X_age[11:18,1:8])
```

```
##      [,1]     [,2]     [,3]     [,4]     [,5]     [,6]     [,7] [,8]
## [1,]   60 65.45205 67.45205 69.92877 72.01096 74.01096 76.00000   NA
## [2,]   60 65.44932 67.46301 69.92603 71.96986 73.96986 75.92055   NA
## [3,]   60 65.45753 67.44658 69.92329 71.96712 73.96712 75.91781   NA
## [4,]   60 65.44384       NA       NA       NA       NA       NA   NA
## [5,]   60 65.43836 67.42192 69.93699 71.94247 73.94247 75.89315   NA
## [6,]   60 65.40822 67.40822 70.04932 72.07123 74.06575 76.09041   NA
## [7,]   60 65.38082 67.38082 70.02192       NA       NA       NA   NA
## [8,]   60 65.47123 67.47123       NA       NA       NA       NA   NA
```

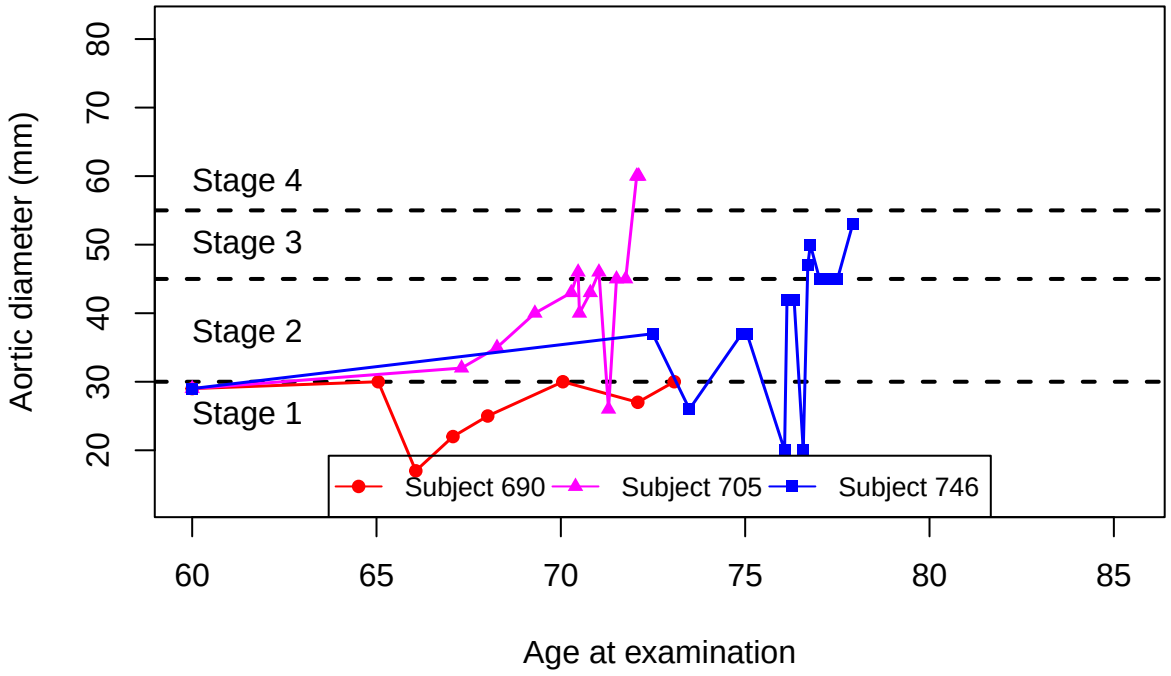
```
(Ki[11:18])
```

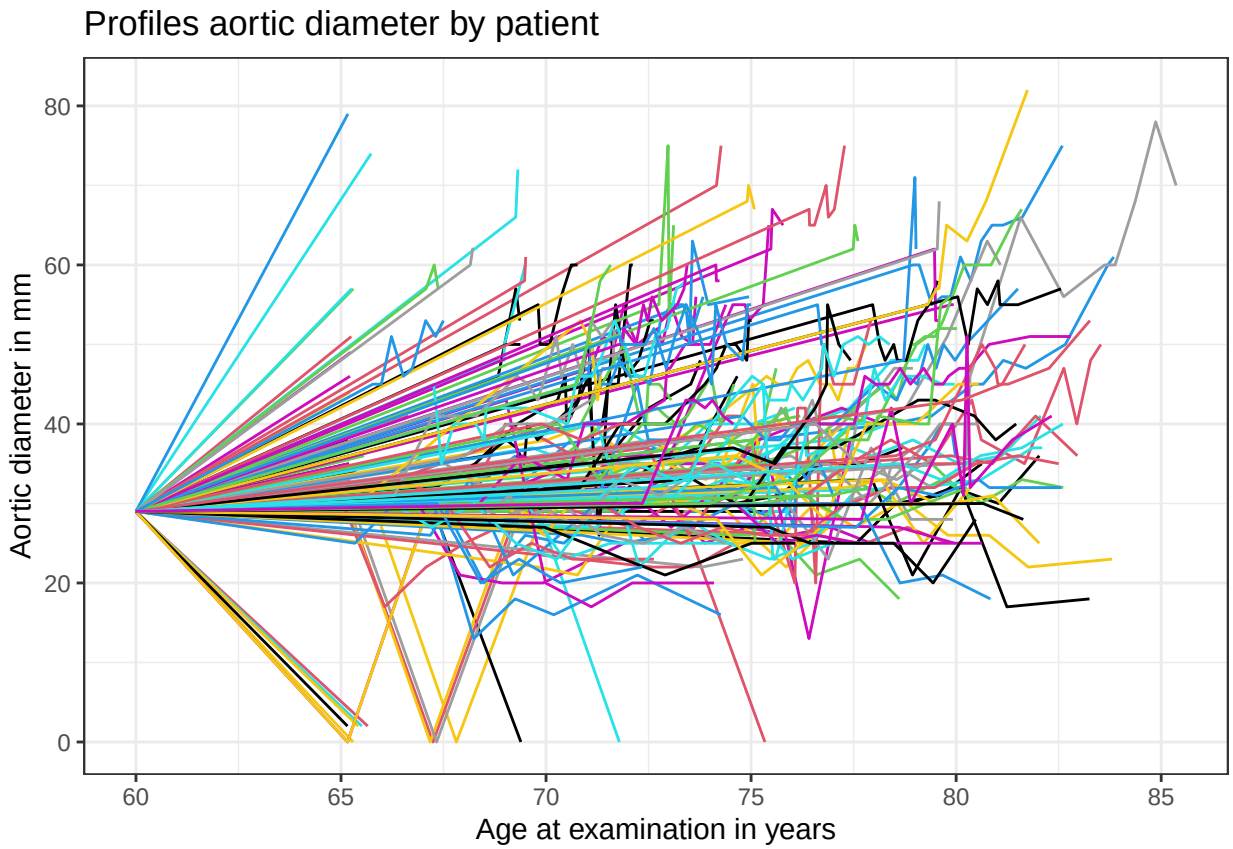
```
##
## 11 12 13 14 15 16 17 18
##  7  7  7  2  7  7  4  3
```

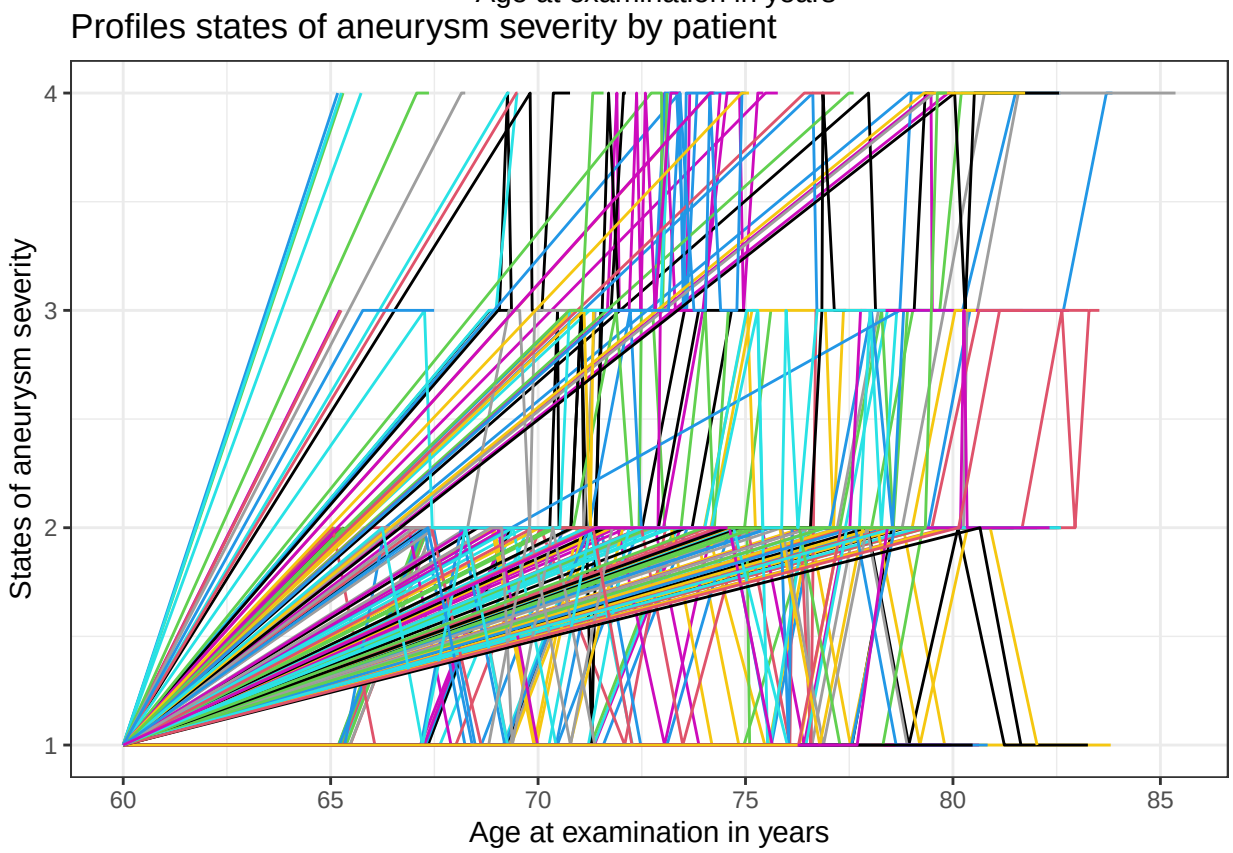
```
### Considering only data having more than one screen (state>1)
idx2 = c()
for(i in 1:N){
  if( sum(Y_state[i,1:Ki[i]])>Ki[i]){
    idx2 = c(idx2,i)
  }
}
Y2_diam = Y_diam[idx2,]
Y2_state = Y_state[idx2,]
X2_age = X_age[idx2,]
N2 = length(idx2)
Ki2 = Ki[idx2]

### Considering only data having more than one screen (diam!=29, or diam<29 & dim>29)
idx3 = c()
for(i in 1:N){
  if( min(Y_diam[i,1:Ki[i]])!=max(Y_diam[i,1:Ki[i]])){
    idx3 = c(idx3,i)
  }
}
Y3_diam = Y_diam[idx3,]
```

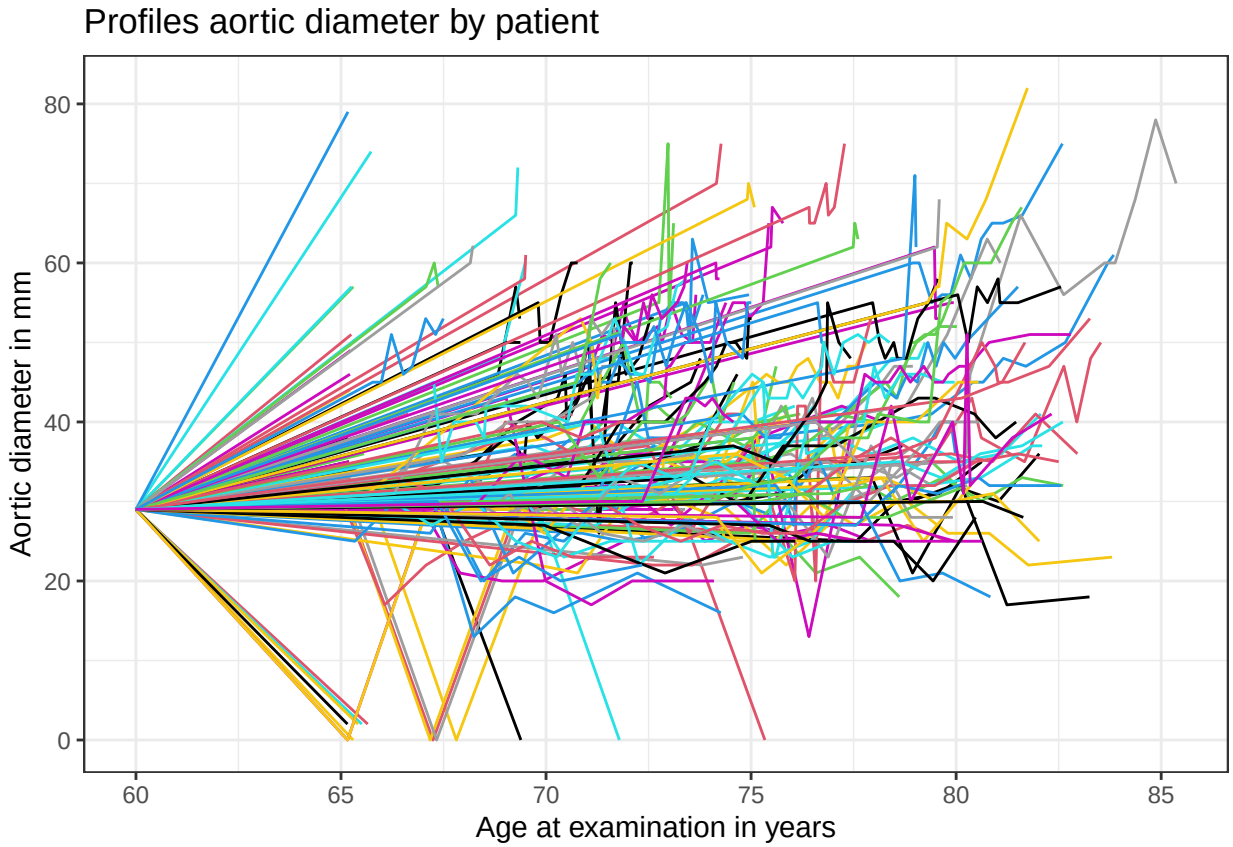
```
Y3_state = Y_state[idx3,]
X3_age = X_age[idx3,]
N3 = length(idx3)
Ki3 = Ki[idx3]

aneur2 = aneur%>%filter(aneur$ptnum%in%idx2)
aneur3 = aneur%>%filter(aneur$ptnum%in%idx3)
### Creo que es mejor trabajar con aneur3
```

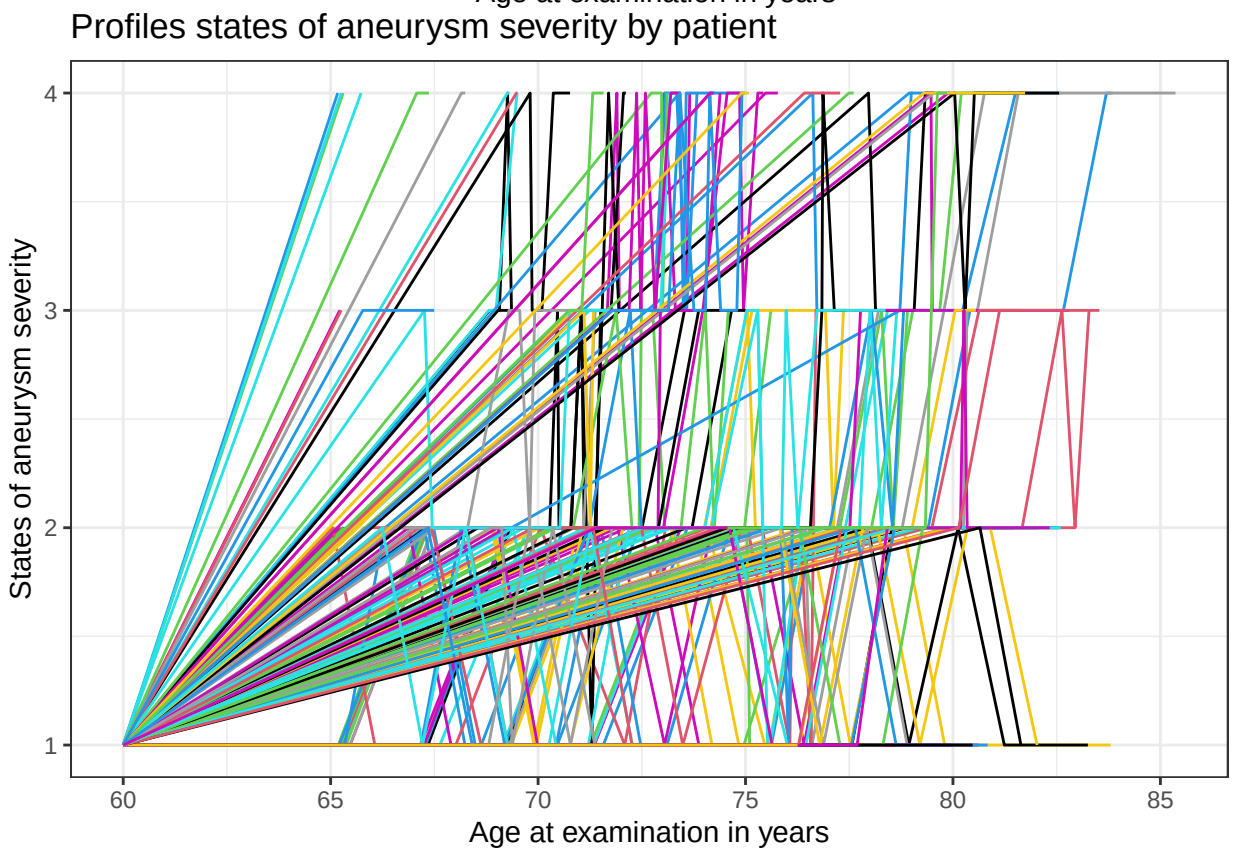
```
##
##  67  80 119
##   6   6   6
```




Profiles aortic diameter by patient
Aortic diameter in mm
0
20
40
60
80
60
65
70
75
80
85
Age at examination in years


Profiles states of aneurysm severity by patient
States of aneurysm severity
1
2
3
4
60
65
70
75
80
85
Age at examination in years





La variable respuesta puede ser continua (''diam'') u ordinal (''state''), y la unica covariable es la edad

("age") \

$$diam_{it} \quad = \quad \beta_0 + f_1(age_{it}) + b_{0i} + age_{it} \times b_{1i} + \varepsilon_{it}, \qquad b_i \sim \mathrm{N}(0, \psi), \quad \varepsilon_i \sim \mathrm{N}(0, \Lambda\sigma^2),$$

where f_1 is a non-decreasing smoothing function and $b_{1i} > 0$.

Quiza solo debemos considerar intercepto fijo, pero NO intercepto aleatorio, y SI pendiente aleatorio

$$diam_{it} \quad = \quad \beta_0 + f_1(age_{it}) + age_{it} \times b_{1i} + \varepsilon_{it}, \qquad b_{1i} \sim \mathrm{N}(0, \psi), \quad \varepsilon_i \sim \mathrm{N}(0, \Lambda\sigma^2),$$

The ordinal response $state_{it}$ is modelled in terms of the cumulative probabilities $P(state_{it} \leq j|b_i)$ by using the proportional odds model,

$$P(state_{it} \leq j|b_i) \quad = \quad \eta_{it,j},$$

subject to

$$\eta_{it,j} \quad = \quad \kappa_j + \beta_0 + f_1(age_{it}) + age_{it} \times b_{1i}, \qquad b_{1i} \sim \mathrm{N}(0, \psi),$$

where the constraints are such that f_1 is a non-decreasing smoothing function and $b_{1i} > 0$, and for the breakpoints $\kappa_j < \kappa_{j+1}$ with $j = 1, 2$.

```
y = aneur3$diam -29
x1 = aneur3$age -60
x2 = aneur3$age -60
id = as.numeric(as.factor(aneur3$ptnum))

(n = length(y))
```

```
## [1] 1387
```

```
(N = n_distinct(id))
```

```
## [1] 229
```

```
Ni = c(0,cumsum(table(id)))+1
k1 = 4 #knots
k2 = 4 #knots
knots1 = quantile(x1, c(0.33,0.67))
knots2 = quantile(x2, c(0.33,0.67))
```

2. Generar la matriz diseño X para los B-splines

Note que $f(x)$ se representa como:

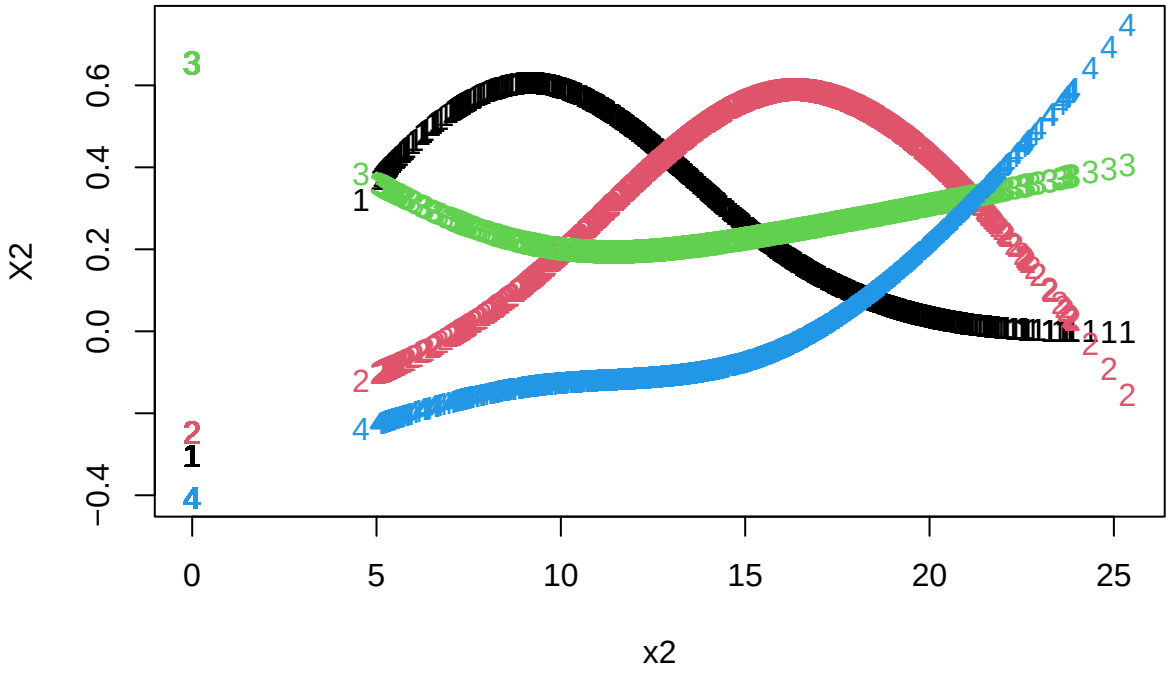
$$\begin{aligned} f(x) &= f_1(x_1) \\ &= \sum_{j=1}^{h_1} \beta_{1j} I_{1j}(x) \end{aligned}$$

para β_{1j} parámetros desconocidos, y para los $I_{1j}(x)$ se utilizar'an I-splines y B-splines.

El número de knots se elige lo suficientemente grande para evitar **over-smoothing**, pero lo suficientemente pequeño para evitar excesivo costo computacional.

El número de *knots* K es considerado a priori.

```
# Generate a basis matrix for Natural Cubic Splines
X2 <- ns(x = x2, knots = knots2, intercept = TRUE)
###X2 = (X2-mean(X2))/sd(X2)
matplot(x2, X2)
```

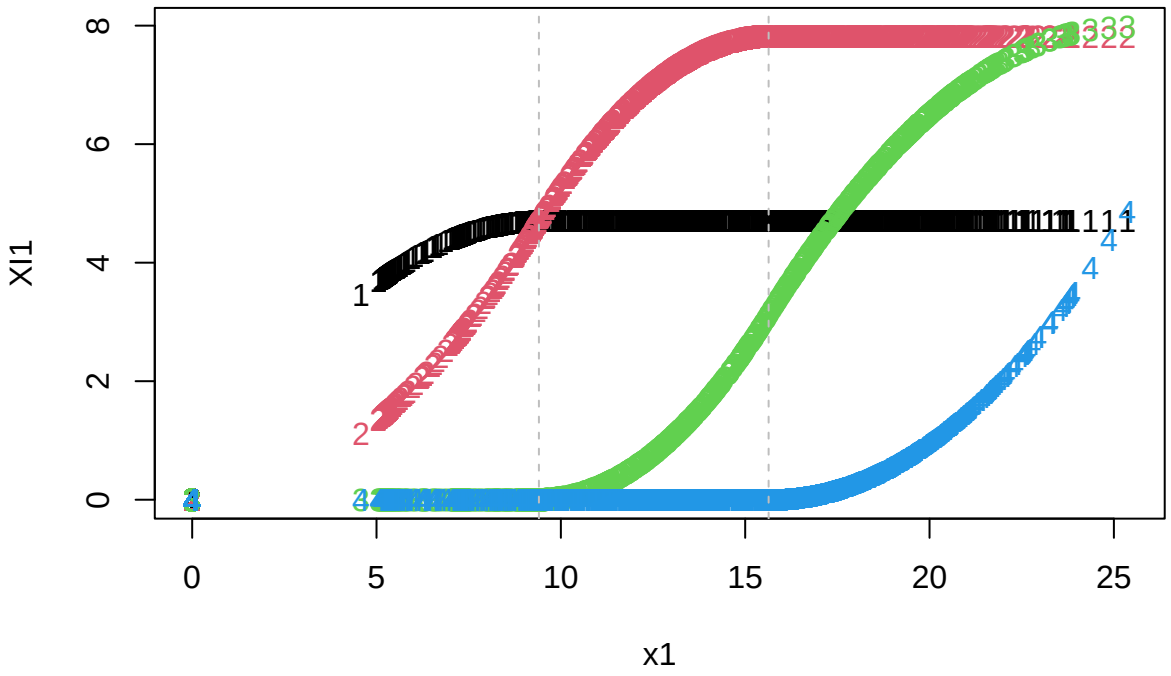


3. Generar la matriz diseño $XI1$ para los I-splines

$$f_1(x_1) \quad \sum_{j=1}^{h_1} \beta_{1j} I_{1j}(x_1)$$

$$I_{1j}(x_1) \quad = \quad \int_{x_0}^{x_1} B_{1j}(u) d_u$$

```
### ibs: integrated basis splines
### degree = 3 cubic splines
XI1 <- ibs(x1, knots = knots1, degree = 1, intercept = TRUE)
###XI1 = (XI1-mean(XI1))/sd(XI1)
matplot(x1, XI1)
abline(v = knots1, h = knots1, lty = 2, col = "gray")
```



4. Definir la penalización $S1$ y $S2$

La flexibilidad ajustada de f es controlada por K, a través de una penalización cuadrática de la forma:

$$\sum_j \lambda_j \beta^T S_j \beta$$

donde los S_j son matrices de coeficientes conocidos, y los λ_j son parámetros de suavizamiento estimados.

```
#Este es el código que produce la matriz de diferenciación.
#No es el óptimo, pero funciona.
#"k" es el número de b-splines y
#"d" el orden de la diferenciación.
#Adjunto el artículo donde discutimos esto (página 7).

diffMatrix = function(k, d = 2){
  if( (d<1) || (d %% 1 != 0) )stop("d must be a positive integer value");
  if( (k<1) || (k %% 1 != 0) )stop("k must be a positive integer value");
  if(d >= k)stop("d must be lower than k");
  out = diag(k);
  for(i in 1:d){
    out = diff(out);
  }
  return(out)
}
(D1 = diffMatrix(k=k1, d=2))
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1   -2    1    0
## [2,]    0    1   -2    1
```

```
(D2 = diffMatrix(k=k2, d=2))
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1   -2    1    0
## [2,]    0    1   -2    1
```

```
(S1 = t(D1)%*%D1 + diag(1,k1)*10e-4)
```

```
##        [,1]   [,2]   [,3]   [,4]
## [1,]  1.001 -2.000  1.000  0.000
## [2,] -2.000  5.001 -4.000  1.000
## [3,]  1.000 -4.000  5.001 -2.000
## [4,]  0.000  1.000 -2.000  1.001
```

```
(S2 = t(D2)%*%D2 + diag(1,k2)*10e-4)
```

```
##        [,1]   [,2]   [,3]   [,4]
## [1,]  1.001 -2.000  1.000  0.000
## [2,] -2.000  5.001 -4.000  1.000
## [3,]  1.000 -4.000  5.001 -2.000
## [4,]  0.000  1.000 -2.000  1.001
```

5. Lineal NO restricciones

5.1 Lineal fit without constraints:

5.2 LME: Lineal fit without constraints:

```
set.seed(123)
# mod.lme <- lme( y ~ x1 , random=~1|id )
mod.lme <- lme( y  ~ x1 , random=list(id=~1) )
summary(mod.lme)
```

```
## Linear mixed-effects model fit by REML
##   Data: NULL
##        AIC      BIC   logLik
##   9776.121 9797.055 -4884.06
##
## Random effects:
##  Formula: ~1 | id
##         (Intercept) Residual
## StdDev:    8.473532 6.824656
##
## Fixed effects:  y ~ x1
##                  Value Std.Error   DF   t-value p-value
## (Intercept) -0.4908194 0.6929494 1157 -0.708305  0.4789
## x1           0.6645373 0.0342989 1157 19.374878  0.0000
##  Correlation:
##    (Intr)
## x1 -0.502
##
## Standardized Within-Group Residuals:
##         Min          Q1         Med          Q3         Max
## -4.39202484 -0.46543276 -0.01792227  0.51801631  4.26459889
##
## Number of Observations: 1387
## Number of Groups: 229
```

```
sigma(mod.lme)
```

```
## [1] 6.824656
```

```
set.seed(123)
mod.lme1 <- lme( y ~ x1 , random=~0+x1|id )
mod.lme1 <- lme( y  ~ x1 , random=list(id=~0+x1) )
summary(mod.lme1)
```

```
## Linear mixed-effects model fit by REML
##   Data: NULL
##        AIC      BIC    logLik
##   9097.928 9118.862 -4544.964
##
## Random effects:
##  Formula: ~0 + x1 | id
##               x1 Residual
## StdDev: 1.273777 4.748012
##
```

```
## Fixed effects:  y ~ x1
##                 Value Std.Error   DF   t-value p-value
## (Intercept) -0.8446225 0.2968668 1157 -2.845123  0.0045
## x1           0.7353480 0.0918708 1157  8.004154  0.0000
##  Correlation:
##    (Intr)
## x1 -0.301
##
## Standardized Within-Group Residuals:
##        Min         Q1        Med         Q3        Max
## -5.6959071 -0.3607015  0.1205509  0.3598872  3.9137646
##
## Number of Observations: 1387
## Number of Groups: 229
```

```
sigma(mod.lme1)
```

```
## [1] 4.748012
```

```
# mod.lme3 <- lme( y ~ x1 , random=~x1|id )
mod.lme3 <- lme( y  ~ x1 , random=list(id=~x1) )
```

```
## Error in lme.formula(y ~ x1, random = list(id = ~x1)): nlminb problem, convergence error code = 1
##   message = iteration limit reached without convergence (10)
```

```
# Error in lme.formula(y ~ x1, random = ~x1 | id) :
# nlminb problem, convergence error code = 1 message = iteration
# limit reached without convergence (10)
```

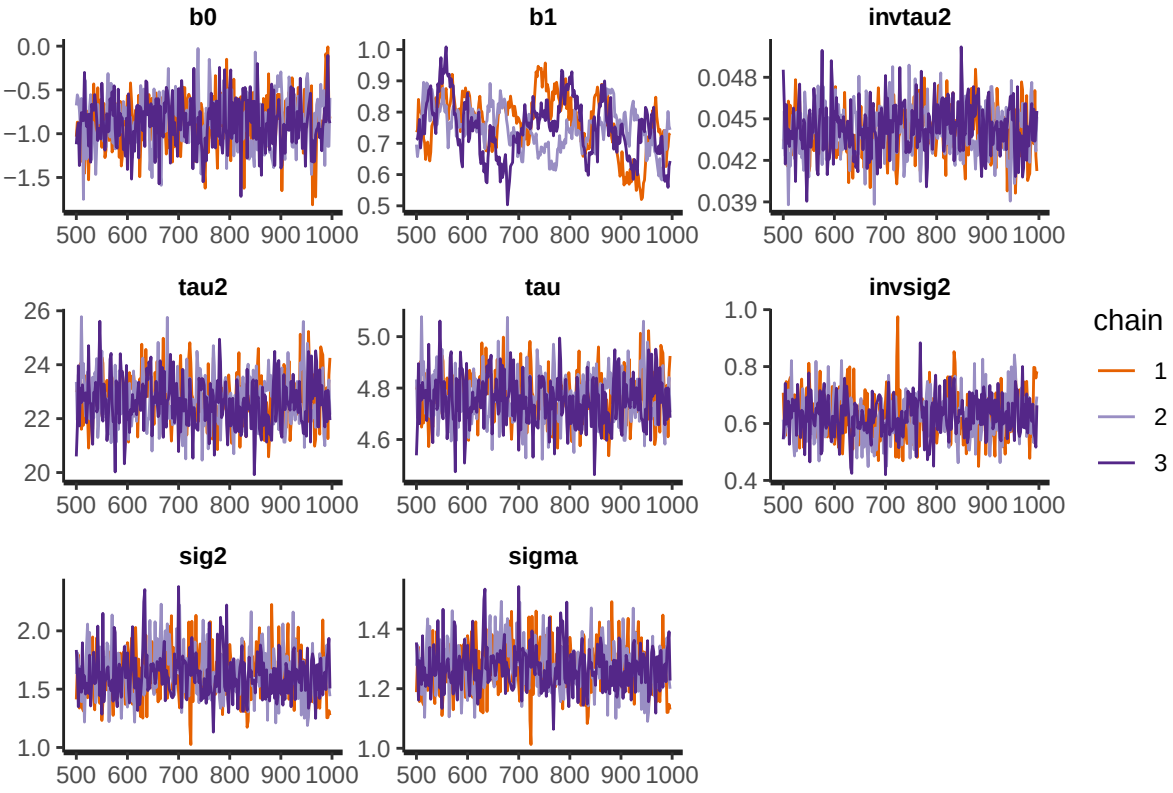
```
datos.lme <- list( y = y ,
                   n = length(y) , N = N , Ni = Ni,
                   x1 = x1 , id = id )
param.lme = c("b0","b1", "invtau2","tau2","tau", "invsig2","sig2","sigma")
```

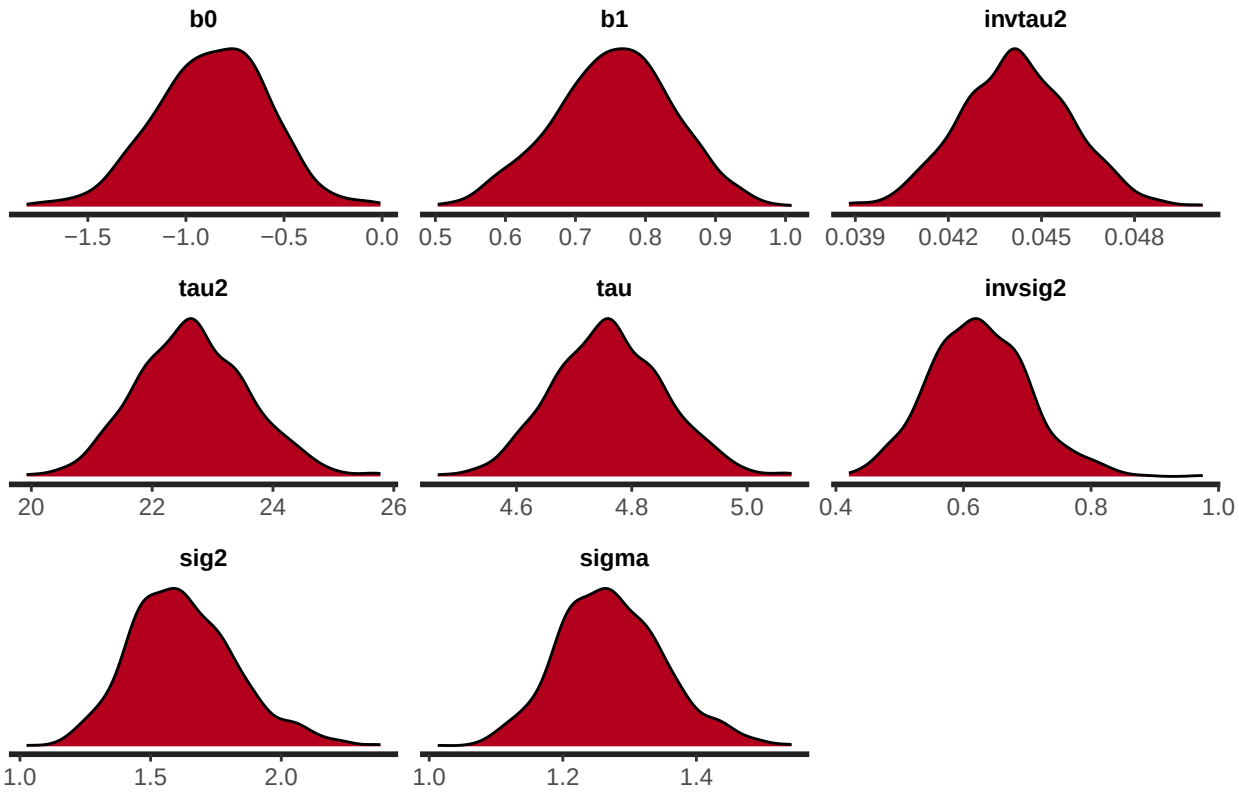
```
fit.lme.non.reslope <- stan("jagam_9_aneur_lme_non_reslope.stan",
            data=datos.lme,
            chains=3,warmup=500,iter=1000,thin=2,cores=6 )


print(fit.lme.non.reslope, pars=param.lme, digits=5)
```

```
## Inference for Stan model: jagam_9_aneur_lme_non_reslope.
## 3 chains, each with iter=1000; warmup=500; thin=2;
## post-warmup draws per chain=250, total post-warmup draws=750.
##
##             mean se_mean      sd     2.5%      25%      50%      75%    97.5%
## b0      -0.86352 0.01219 0.28820 -1.44039 -1.05267 -0.85779 -0.66135 -0.33062
## b1       0.75458 0.01395 0.08488  0.58258  0.69883  0.75710  0.81077  0.92101
## invtau2  0.04417 0.00007 0.00179  0.04082  0.04290  0.04418  0.04540  0.04759
## tau2    22.67621 0.03807 0.92264 21.01414 22.02444 22.63652 23.30877 24.49930
## tau      4.76097 0.00399 0.09673  4.58412  4.69302  4.75779  4.82792  4.94968
```

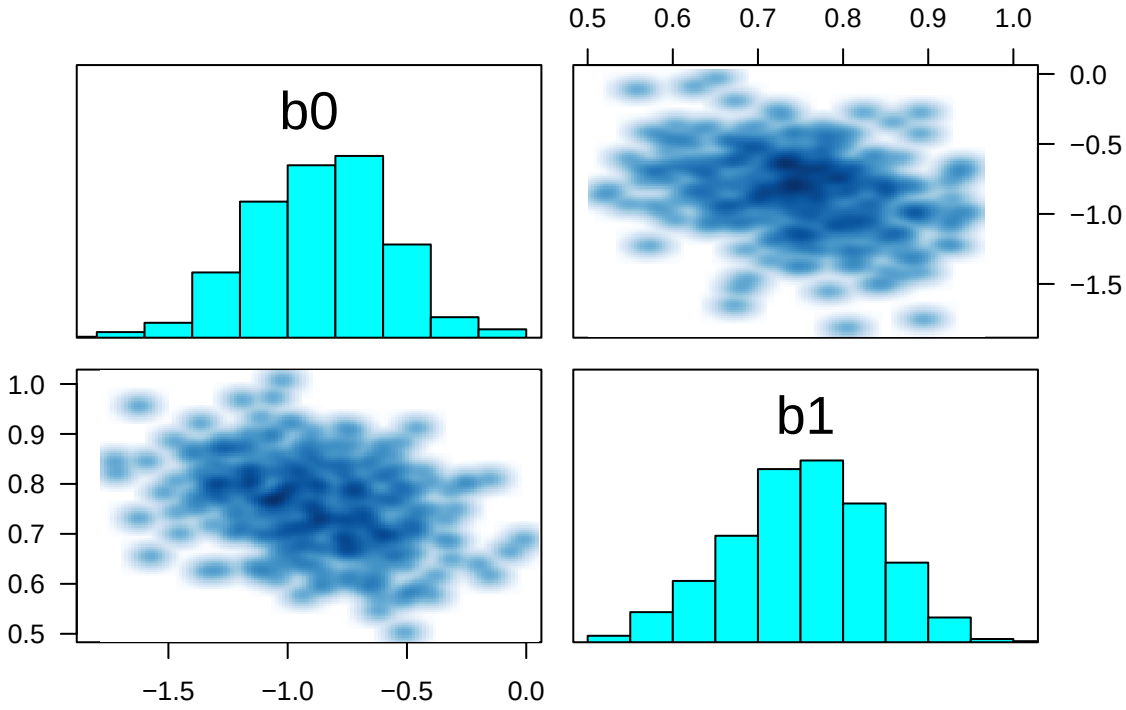
```
## invsig2  0.62446 0.00337 0.07751  0.48122  0.57091  0.62092  0.67642  0.79266
## sig2     1.62628 0.00881 0.20401  1.26157  1.47837  1.61052  1.75157  2.07805
## sigma    1.27279 0.00343 0.07937  1.12320  1.21588  1.26906  1.32347  1.44155
##          n_eff    Rhat
## b0         559 1.00039
## b1          37 1.03563
## invtau2    591 1.00259
## tau2       587 1.00226
## tau        588 1.00235
## invsig2    530 0.99946
## sig2       536 0.99961
## sigma      534 0.99956
##
## Samples were drawn using NUTS(diag_e) at Tue Jan  9 14:21:33 2024.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

```
stan_trace(fit.lme.non.reslope,pars=param.lme)
stan_dens(fit.lme.non.reslope,pars=param.lme)
```

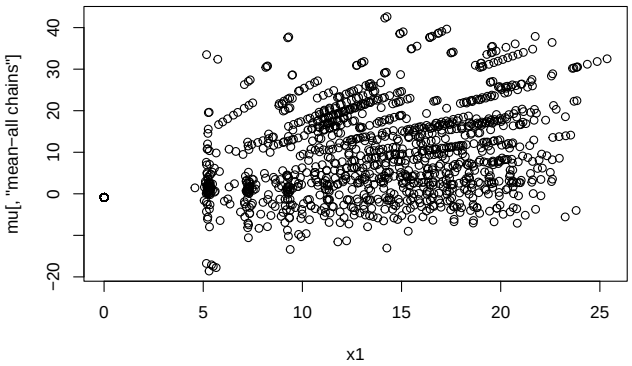





```
pairs(fit.lme.non.reslope, pars = c("b0","b1"), las = 1)
```




```
mu=get_posterior_mean(fit.lme.non.reslope,"mu")
plot(x1,mu[,"mean-all chains"])
```


mu[, "mean−all chains"]
−20
0
10
20
30
40
0
5
10
15
20
25
x1

6. Lineal creciente

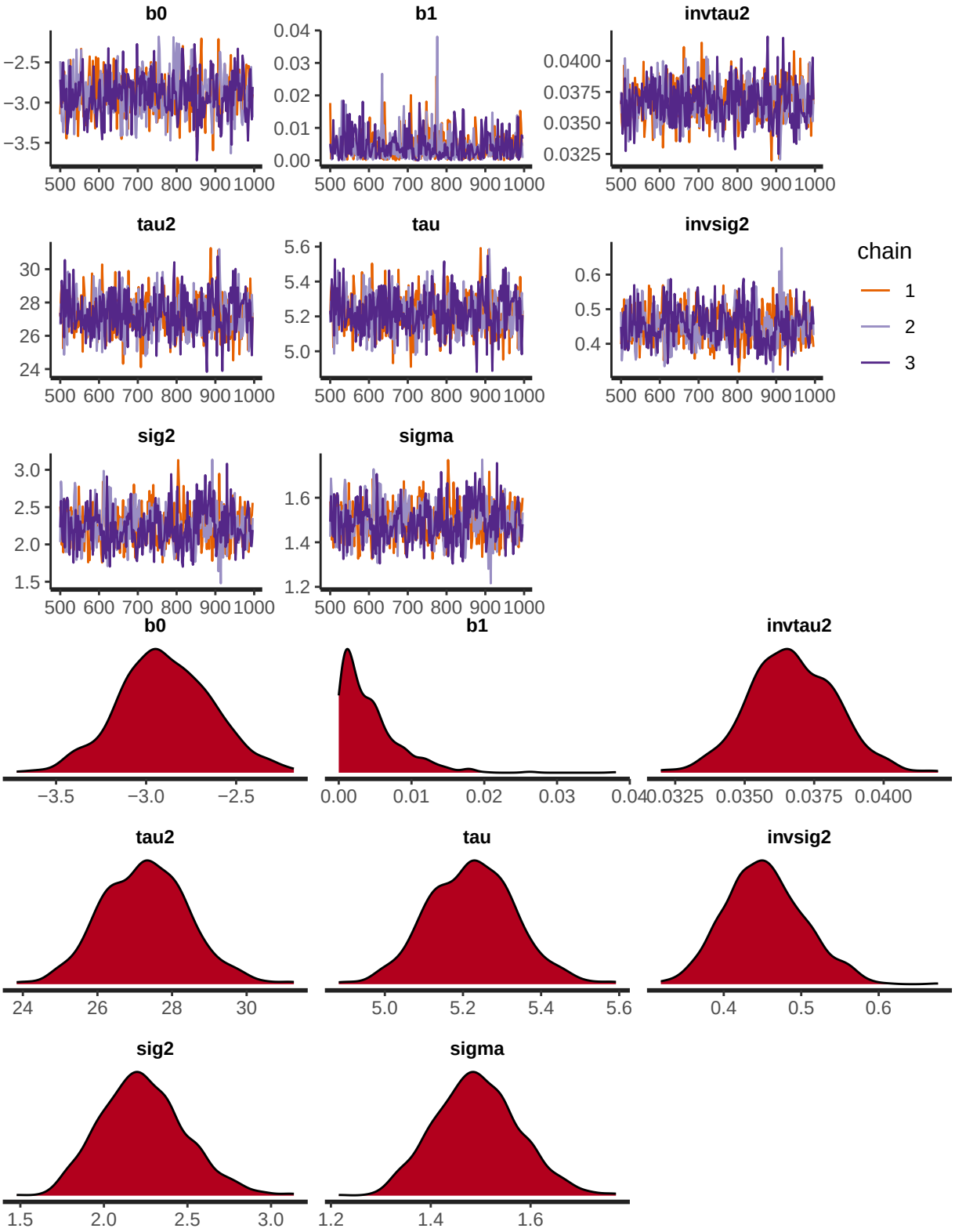
6.1. Lineal creciente

6.2. LME: Lineal creciente

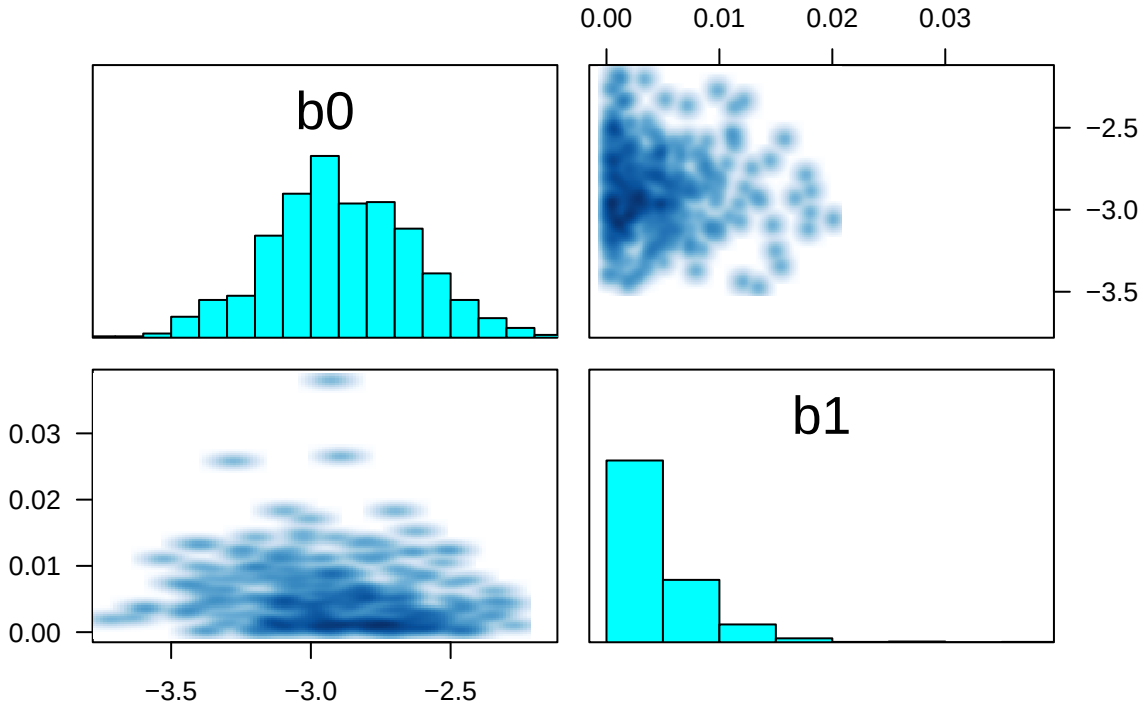
```
fit.lme.incr.reslope <- stan("jagam_9_aneur_lme_incr_reslope.stan",
            data=datos.lme,
            chains=3,warmup=500,iter=1000,thin=2,cores=6 )

print(fit.lme.incr.reslope, pars=param.lme, digits=5)
```

```
## Inference for Stan model: jagam_9_aneur_lme_incr_reslope.
## 3 chains, each with iter=1000; warmup=500; thin=2;
## post-warmup draws per chain=250, total post-warmup draws=750.
##
##             mean se_mean      sd     2.5%      25%      50%      75%    97.5%
## b0      -2.89304 0.01114 0.25649 -3.40123 -3.06703 -2.90904 -2.71514 -2.35359
## b1       0.00428 0.00016 0.00414  0.00017  0.00124  0.00315  0.00593  0.01437
## invtau2  0.03671 0.00006 0.00156  0.03365  0.03564  0.03663  0.03779  0.03989
## tau2    27.28711 0.04603 1.16124 25.06678 26.46436 27.29714 28.05491 29.71447
## tau      5.22253 0.00439 0.11104  5.00667  5.14435  5.22467  5.29669  5.45110
## invsig2  0.45313 0.00222 0.05075  0.36050  0.41899  0.45030  0.48579  0.55917
## sig2     2.23464 0.01118 0.25145  1.78837  2.05851  2.22076  2.38670  2.77393
## sigma    1.49253 0.00371 0.08368  1.33730  1.43475  1.49022  1.54490  1.66551
##         n_eff    Rhat
## b0        531 1.00189
## b1        661 0.99830
## invtau2   649 0.99775
## tau2      636 0.99787
## tau       639 0.99784
## invsig2   525 1.00138
## sig2      505 1.00136
## sigma     510 1.00136
##
## Samples were drawn using NUTS(diag_e) at Tue Jan  9 14:22:20 2024.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

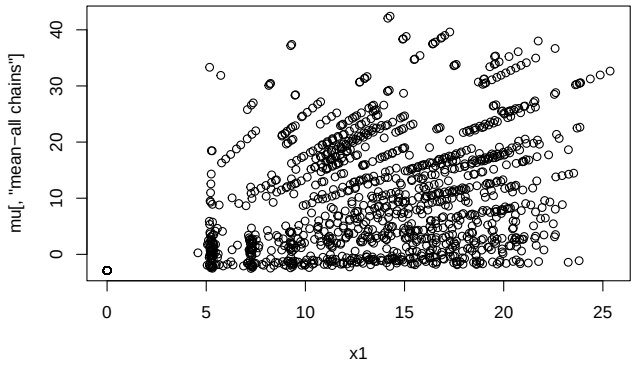
```
stan_trace(fit.lme.incr.reslope,pars=param.lme)
stan_dens(fit.lme.incr.reslope,pars=param.lme)
```




```
pairs(fit.lme.incr.reslope, pars = c("b0","b1"), las = 1)
```




```
mu=get_posterior_mean(fit.lme.incr.reslope,"mu")
plot(x1,mu[,"mean-all chains"])
```



7. Spline NO restricciones

7.1 For a spline-based fit without constraints:

7.2 LME: For a spline-based fit without constraints:

```
set.seed(123)
mod.lme.s <- lme( y ~  XI1 , random=list(id=~1) )
summary(mod.lme.s)
```

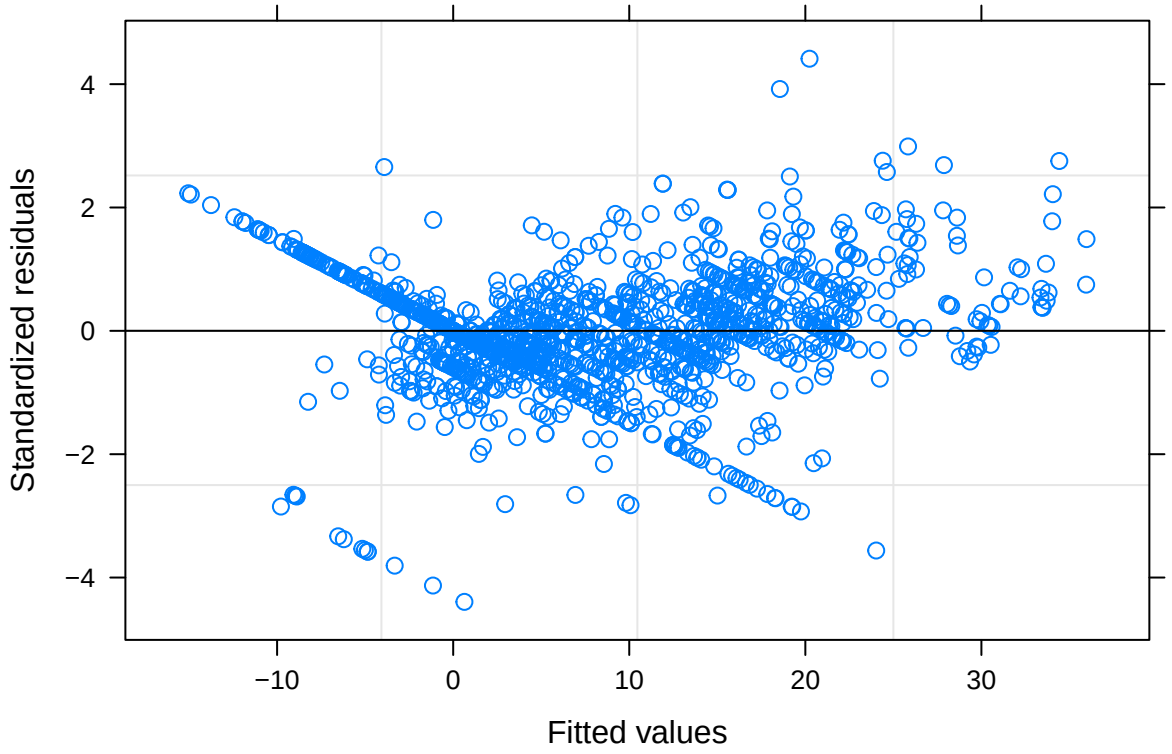
```
## Linear mixed-effects model fit by REML
##   Data: NULL
##        AIC      BIC    logLik
##   9753.189 9789.808 -4869.595
##
## Random effects:
##  Formula: ~1 | id
##         (Intercept) Residual
## StdDev:     8.51601 6.744787
##
## Fixed effects:  y ~ XI1
##                Value Std.Error   DF  t-value p-value
## (Intercept) 0.045604 0.7173634 1154 0.063571  0.9493
## XI11        0.147201 0.2661071 1154 0.553166  0.5803
## XI12        0.970901 0.2158582 1154 4.497864  0.0000
## XI13        0.253511 0.2281802 1154 1.111010  0.2668
## XI14        3.889209 0.6769512 1154 5.745184  0.0000
##  Correlation:
##      (Intr) XI11   XI12   XI13
## XI11 -0.264
## XI12  0.039 -0.860
## XI13 -0.013  0.300 -0.605
## XI14  0.007 -0.160  0.290 -0.662
##
## Standardized Within-Group Residuals:
##          Min           Q1          Med           Q3          Max
## -4.394201884 -0.453908702 -0.003561512  0.499733599  4.412553978
##
## Number of Observations: 1387
## Number of Groups: 229
```

```
sigma(mod.lme.s)
```

```
## [1] 6.744787
```

```
plot(mod.lme.s)
anova(mod.lme,mod.lme.s)
```

```
##           Model df      AIC      BIC    logLik   Test  L.Ratio p-value
## mod.lme       1  4 9776.121 9797.055 -4884.060
## mod.lme.s     2  7 9753.189 9789.808 -4869.595 1 vs 2 28.93154  <.0001
```




```
set.seed(123)
mod.lme.s1 <- lme( y ~  XI1 , random=list(id=~0+x1) )
summary(mod.lme.s1)
```

```
## Linear mixed-effects model fit by REML
##   Data: NULL
##        AIC      BIC    logLik
##   8950.966 8987.585 -4468.483
##
## Random effects:
##  Formula: ~0 + x1 | id
##               x1 Residual
## StdDev: 1.401535 4.375082
##
## Fixed effects:  y ~ XI1
##                 Value Std.Error   DF   t-value p-value
## (Intercept)  0.021674 0.2886553 1154  0.075084  0.9402
## XI11        -0.217886 0.2073866 1154 -1.050629  0.2936
## XI12         1.053298 0.1960705 1154  5.372035  0.0000
## XI13         1.768322 0.2216114 1154  7.979381  0.0000
## XI14         4.374687 0.5119063 1154  8.545874  0.0000
##  Correlation:
##      (Intr) XI11   XI12   XI13
## XI11 -0.353
## XI12  0.041 -0.466
## XI13 -0.015  0.297 -0.121
## XI14  0.007 -0.067  0.310 -0.346
##
## Standardized Within-Group Residuals:
```

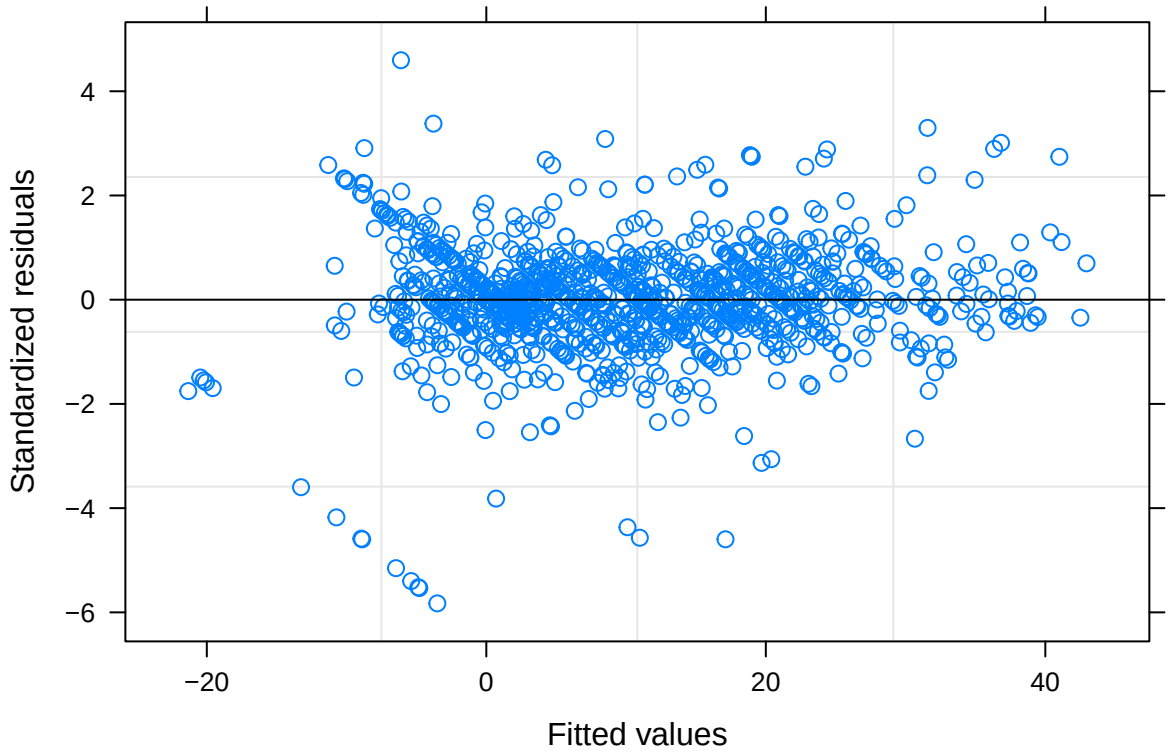
```
##          Min           Q1          Med           Q3          Max
## -5.827242762 -0.328706726 -0.004953857  0.373749636  4.595933776
##
## Number of Observations: 1387
## Number of Groups: 229
```

```
sigma(mod.lme.s1)
```

```
## [1] 4.375082
```

```
plot(mod.lme.s1)
anova(mod.lme,mod.lme.s)
```

```
##           Model df      AIC      BIC    logLik   Test  L.Ratio p-value
## mod.lme       1  4 9776.121 9797.055 -4884.060
## mod.lme.s     2  7 9753.189 9789.808 -4869.595 1 vs 2 28.93154  <.0001
```




```
mod.gamm <- gamm( y  ~  s(x1, k=k1)  , random=list(id=~1) )
summary(mod.gamm$gam)
```

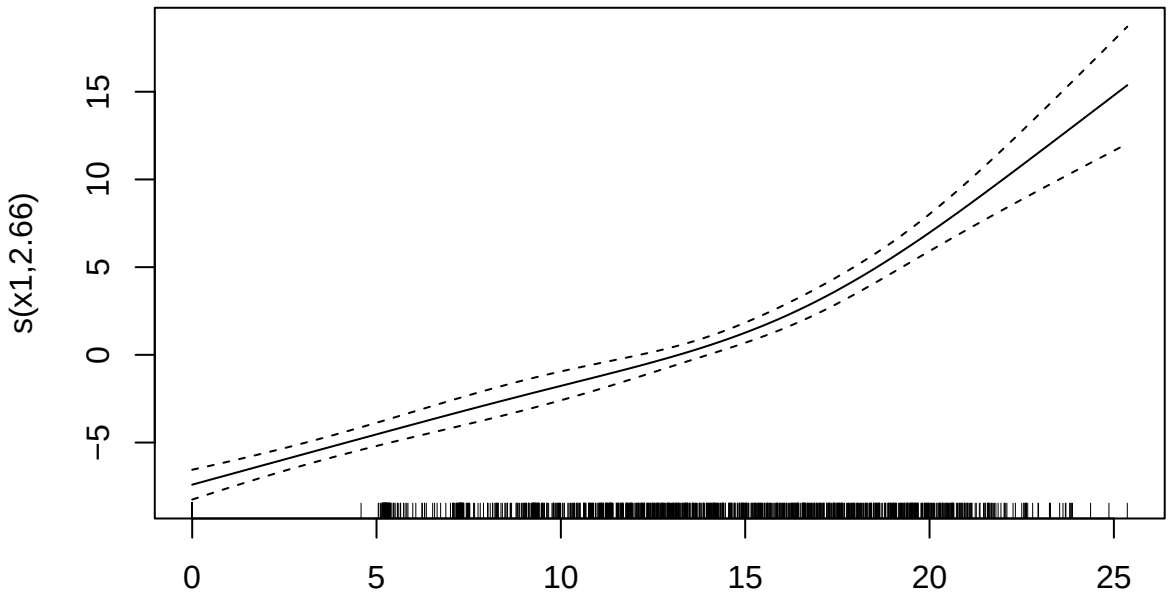
```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## y ~ s(x1, k = k1)
##
```

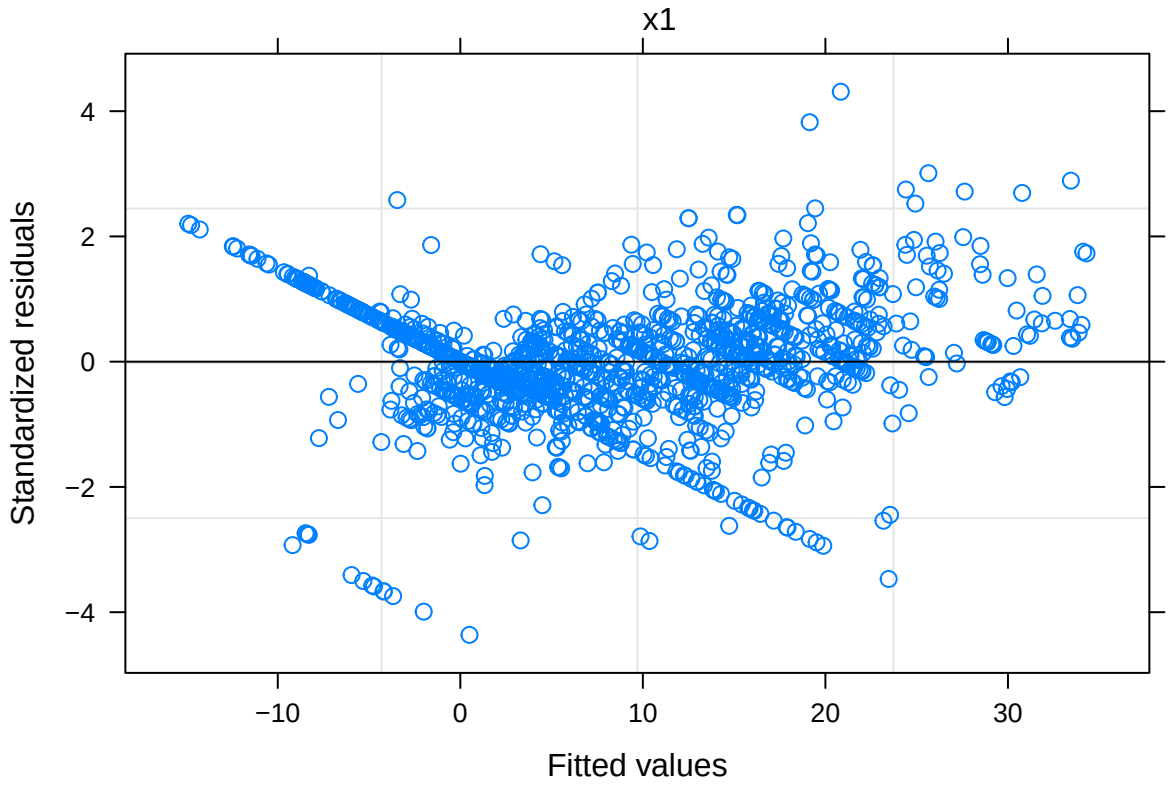
```
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   7.2836     0.6059   12.02   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##         edf Ref.df   F p-value
## s(x1) 2.659  2.659 147  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =   0.16
##   Scale est. = 45.747    n = 1387
```

```
summary(mod.gamm$lme)
```

```
## Linear mixed-effects model fit by maximum likelihood
##   Data: strip.offset(mf)
##        AIC      BIC    logLik
##   9765.232 9791.406 -4877.616
##
## Random effects:
##  Formula: ~Xr - 1 | g
##  Structure: pdIdnot
##             Xr1     Xr2
## StdDev: 11.6555 11.6555
##
##  Formula: ~1 | id %in% g
##         (Intercept) Residual
## StdDev:    8.542761 6.763645
##
## Fixed effects:  y.0 ~ X - 1
##                 Value Std.Error   DF   t-value p-value
## X(Intercept) 7.283611 0.6060915 1157 12.017345       0
## Xs(x1)Fx1    7.373808 1.0052097 1157  7.335591       0
##  Correlation:
##           X(Int)
## Xs(x1)Fx1 0
##
## Standardized Within-Group Residuals:
##          Min           Q1          Med           Q3          Max
## -4.361985592 -0.455510067 -0.004707745  0.493794967  4.310914825
##
## Number of Observations: 1387
## Number of Groups:
##         g id %in% g
##         1       229
```

```
plot(mod.gamm$gam)
plot(mod.gamm$lme)
```






```
mod.gamm1 <- gamm( y  ~  s(x1, k=k1)  , random=list(id=~0+x1) )
summary(mod.gamm1$gam)
```

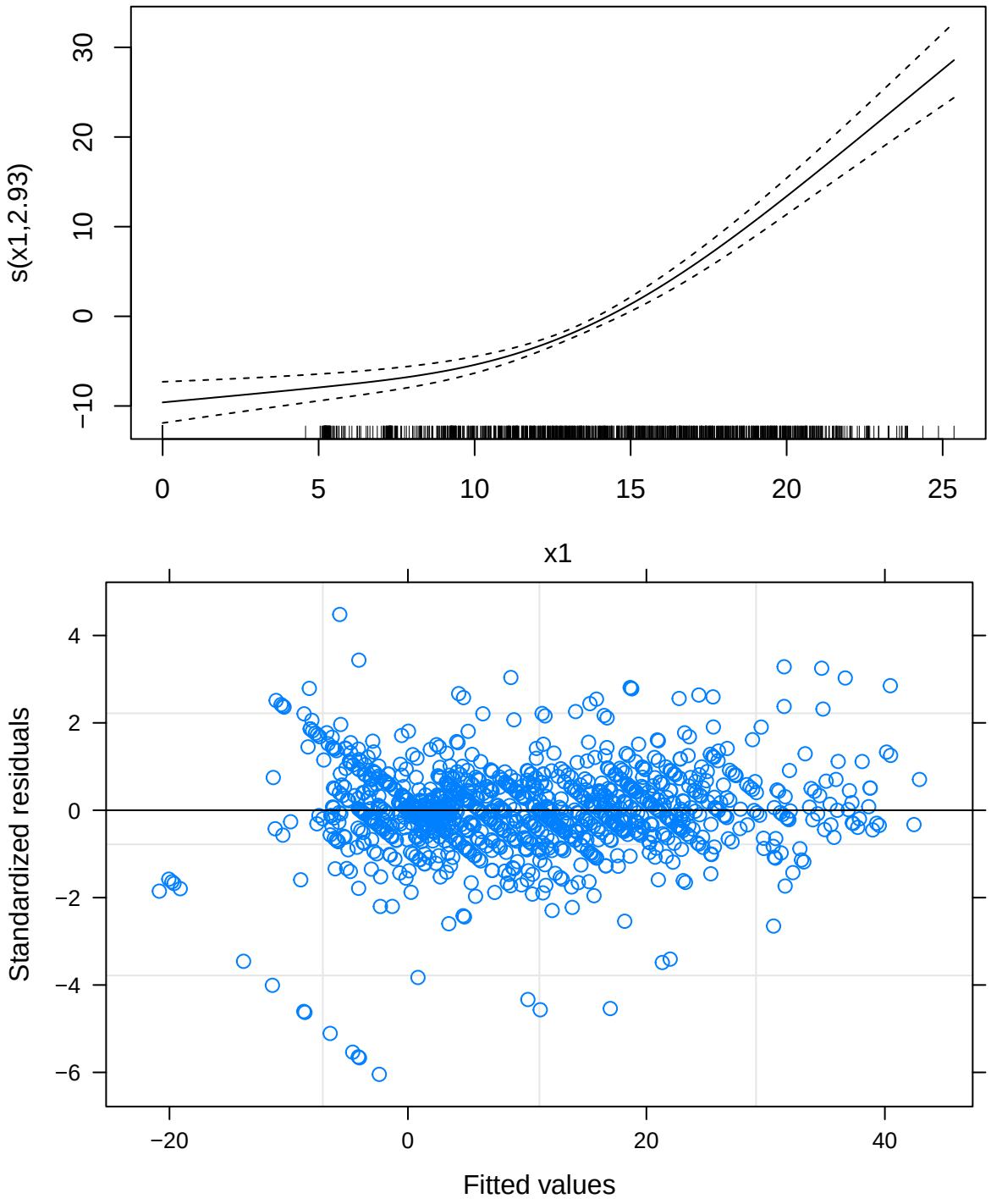
```
##
## Family: gaussian
## Link function: identity
##
## Formula:
```

```
## y ~ s(x1, k = k1)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    9.485      1.118   8.485   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##         edf Ref.df     F p-value
## s(x1) 2.928  2.928 69.47  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.0128
##   Scale est. = 19.36     n = 1387
```

```
summary(mod.gamm1$lme)
```

```
## Linear mixed-effects model fit by maximum likelihood
##   Data: strip.offset(mf)
##        AIC      BIC    logLik
##   8964.745 8990.919 -4477.372
##
## Random effects:
##  Formula: ~Xr - 1 | g
##  Structure: pdIdnot
##              Xr1      Xr2
## StdDev: 22.62108 22.62108
##
##  Formula: ~0 + x1 | id %in% g
##               x1 Residual
## StdDev: 1.380861  4.40003
##
## Fixed effects:  y.0 ~ X - 1
##                  Value Std.Error   DF   t-value p-value
## X(Intercept)  9.485195  1.118241 1157  8.482244       0
## Xs(x1)Fx1    10.777078  1.074914 1157 10.025993       0
##  Correlation:
##           X(Int)
## Xs(x1)Fx1 0.573
##
## Standardized Within-Group Residuals:
##         Min          Q1         Med          Q3         Max
## -6.04472640 -0.32800007  0.02577703  0.36868738  4.48138337
##
## Number of Observations: 1387
## Number of Groups:
##         g id %in% g
##         1       229
```

```
plot(mod.gamm1$gam)
plot(mod.gamm1$lme)
```




```
mod.gamm.ar <- gamm( y  ~  s(x1, k=k1)  ,  random=list(id=~1) ,
                     correlation=corAR1() )
summary(mod.gamm.ar$gam)
```

```
##
## Family: gaussian
## Link function: identity
##
```

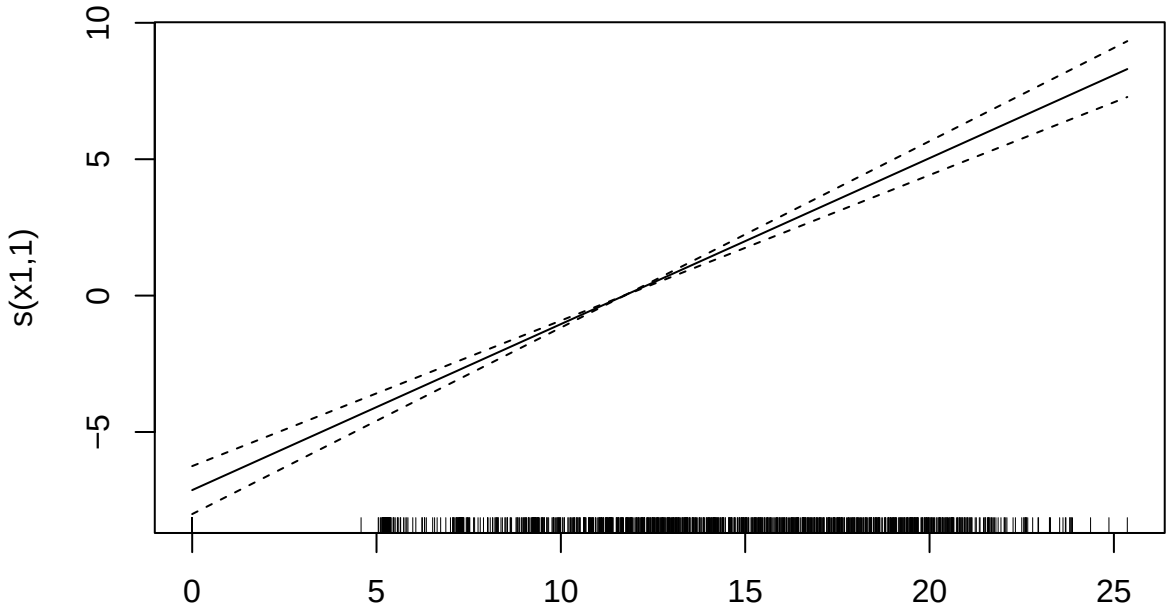
```
## Formula:
## y ~ s(x1, k = k1)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   7.7723     0.5601   13.88   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##       edf Ref.df     F p-value
## s(x1)   1      1 263.2  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =   0.16
##   Scale est. = 90.909    n = 1387
```

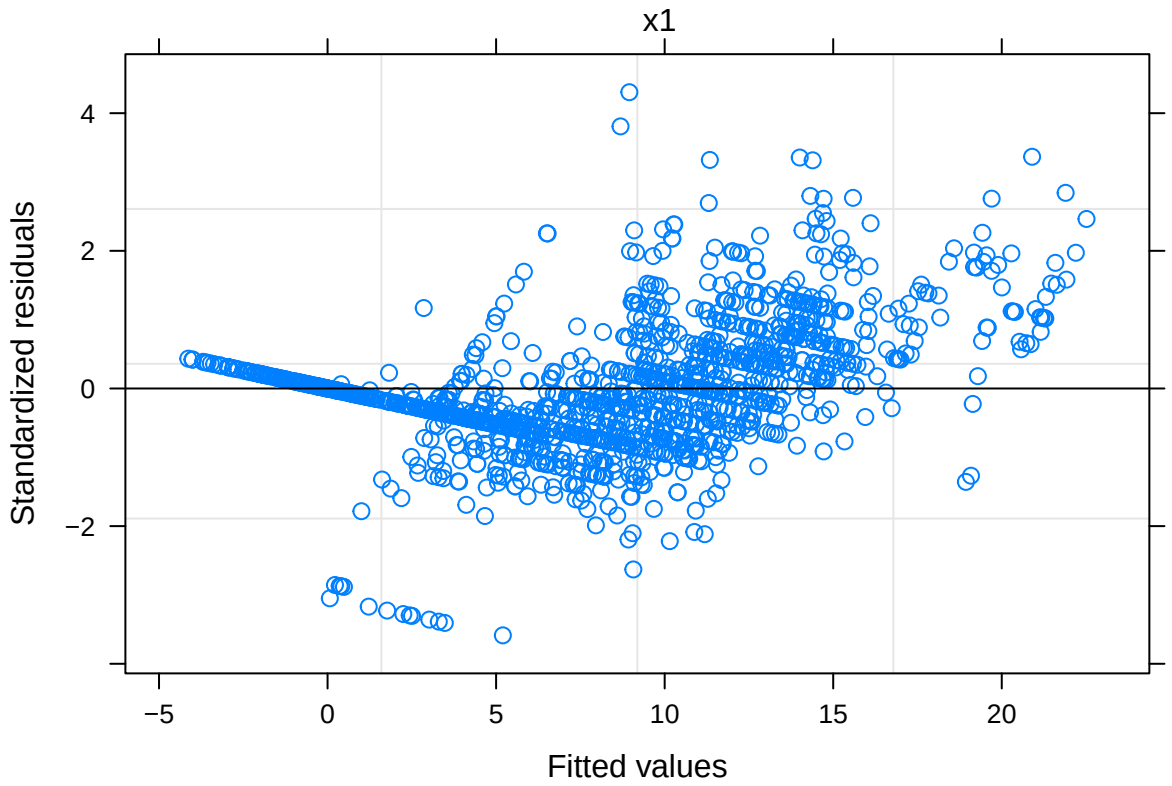
```
summary(mod.gamm.ar$lme)
```

```
## Linear mixed-effects model fit by maximum likelihood
##   Data: strip.offset(mf)
##        AIC      BIC    logLik
##   9587.898 9619.308 -4787.949
##
## Random effects:
##  Formula: ~Xr - 1 | g
##  Structure: pdIdnot
##                 Xr1         Xr2
## StdDev: 0.001703091 0.001703091
##
##  Formula: ~1 | id %in% g
##         (Intercept) Residual
## StdDev:    4.724431 9.534615
##
## Correlation Structure: AR(1)
##  Formula: ~1 | g/id
##  Parameter estimate(s):
##       Phi
## 0.6751387
## Fixed effects:  y.0 ~ X - 1
##                 Value Std.Error   DF  t-value p-value
## X(Intercept) 7.772282 0.5603182 1157 13.87119       0
## Xs(x1)Fx1    4.124876 0.2543414 1157 16.21787       0
##  Correlation:
##           X(Int)
## Xs(x1)Fx1 0.153
##
## Standardized Within-Group Residuals:
##         Min          Q1         Med          Q3         Max
## -3.58692289 -0.53267218 -0.08303584  0.40754736  4.30503270
##
## Number of Observations: 1387
## Number of Groups:
```

```
##          g id %in% g
##          1       229
```

```
plot(mod.gamm.ar$gam)
plot(mod.gamm.ar$lme)
```





```
mod.gamm.ar1 <- gamm( y  ~ s(x1, k=k1)  ,  random=list(id=~0+x1) ,
                     correlation=corAR1() )
summary(mod.gamm.ar1$gam)
```

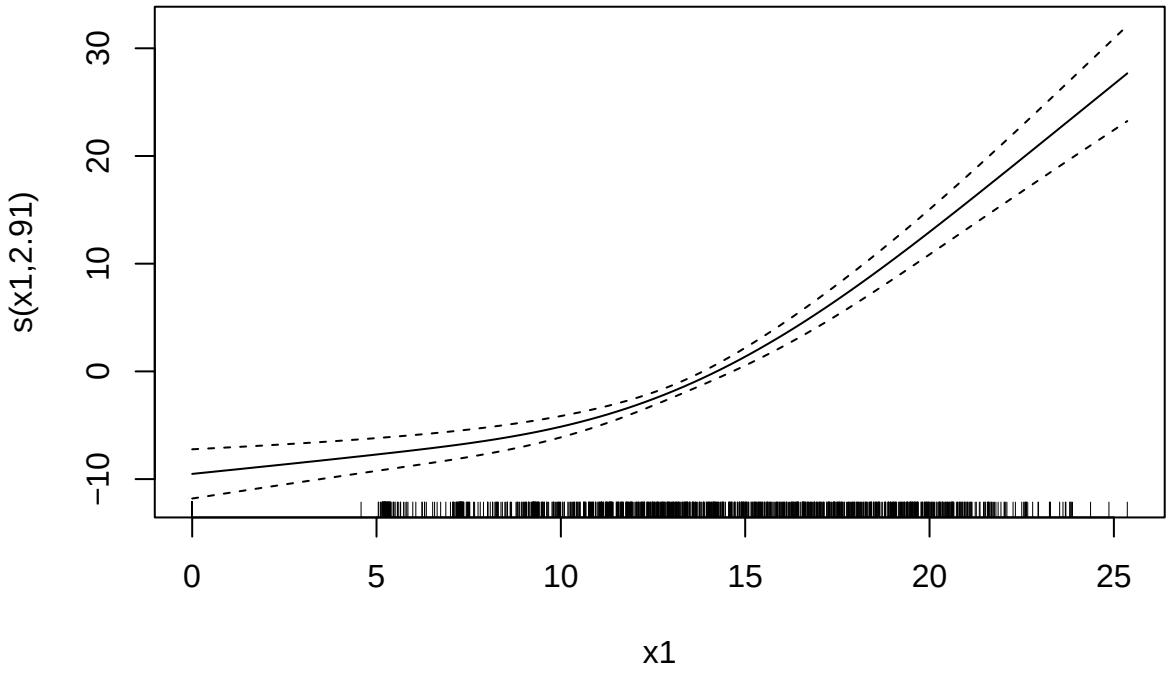
```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## y ~ s(x1, k = k1)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    9.379      1.112   8.432   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##         edf Ref.df     F p-value
## s(x1) 2.907  2.907 56.46  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.0301
##   Scale est. = 20.205    n = 1387
```

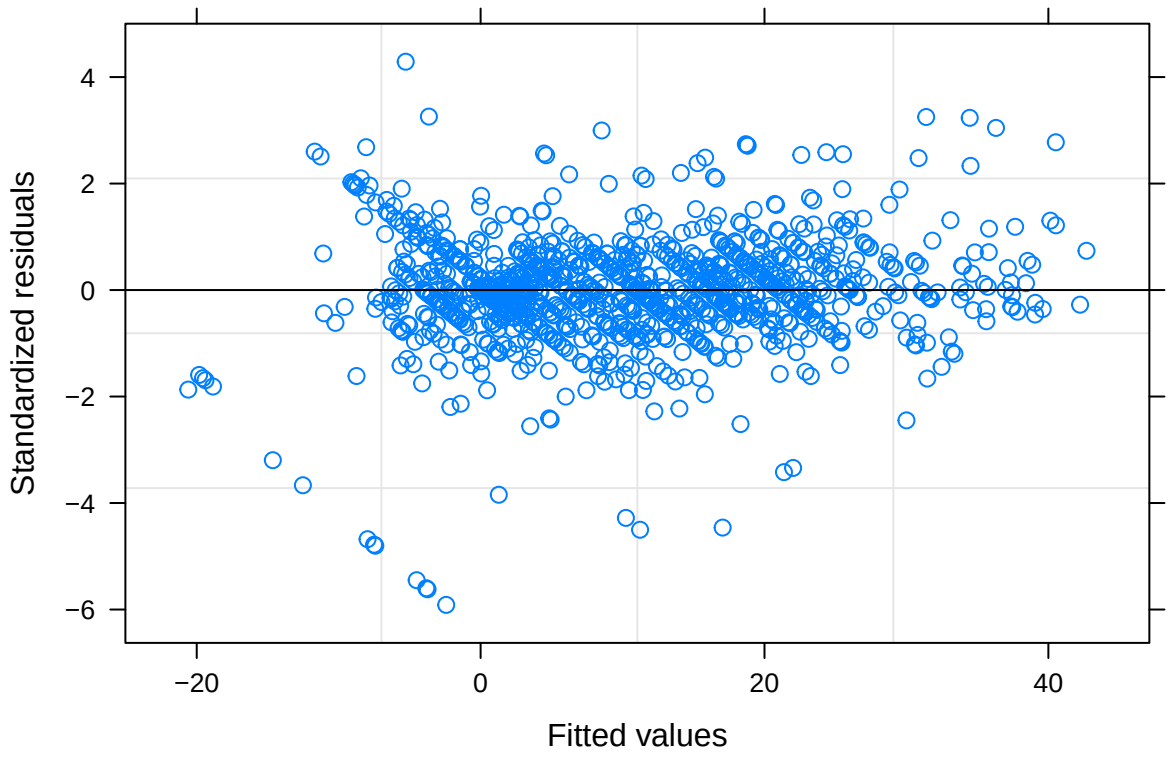
```
summary(mod.gamm.ar1$lme)
```

```
## Linear mixed-effects model fit by maximum likelihood
##   Data: strip.offset(mf)
##        AIC      BIC    logLik
##   8944.268 8975.678 -4466.134
##
## Random effects:
##  Formula: ~Xr - 1 | g
##  Structure: pdIdnot
##              Xr1      Xr2
## StdDev: 21.58851 21.58851
##
##  Formula: ~0 + x1 | id %in% g
##               x1 Residual
## StdDev: 1.366815 4.495042
##
## Correlation Structure: AR(1)
##  Formula: ~1 | g/id
##  Parameter estimate(s):
##       Phi
## 0.1527438
## Fixed effects:  y.0 ~ X - 1
##                  Value Std.Error   DF  t-value p-value
## X(Intercept)  9.379081  1.112715 1157 8.429006       0
## Xs(x1)Fx1    10.584722  1.134490 1157 9.329937       0
```

```
##  Correlation:
##           X(Int)
## Xs(x1)Fx1 0.535
##
## Standardized Within-Group Residuals:
##         Min          Q1         Med          Q3         Max
## -5.91253795 -0.33640243  0.03088513  0.35053651  4.28869217
##
## Number of Observations: 1387
## Number of Groups:
##         g id %in% g
##         1       229
```

```
plot(mod.gamm.ar1$gam)
plot(mod.gamm.ar1$lme)
```




Standardized residuals
4
2
0
−2
−4
−6
−20
0
20
40
Fitted values

```
datos.lme.add.non <- list( y = y ,
                           id = id ,
               n = length(y) ,
               N = N , Ni = Ni,
               k1=k1,
              XI1 = XI1,
              x1 = x1,
              zero = rep(0,1+k1),
              S1=S1  )
inits.lme.add.non <- function(){    list(
  "b0" = rnorm(1,0,0.1) ,
  "b1" = rnorm(k1,0,0.1) ,
  "invtau2" = rgamma(1,1,1) ,
  "lambda" = rgamma(1,1,1) ,
    "invsig2" = rgamma(1,1,1) ,
  "bre0" = rnorm(N,0,0.1)
  ) }
param.lme.add = c("b0","b1", "invtau2","tau2","tau", "lambda","rho", "invsig2","sig2","sigma")
```

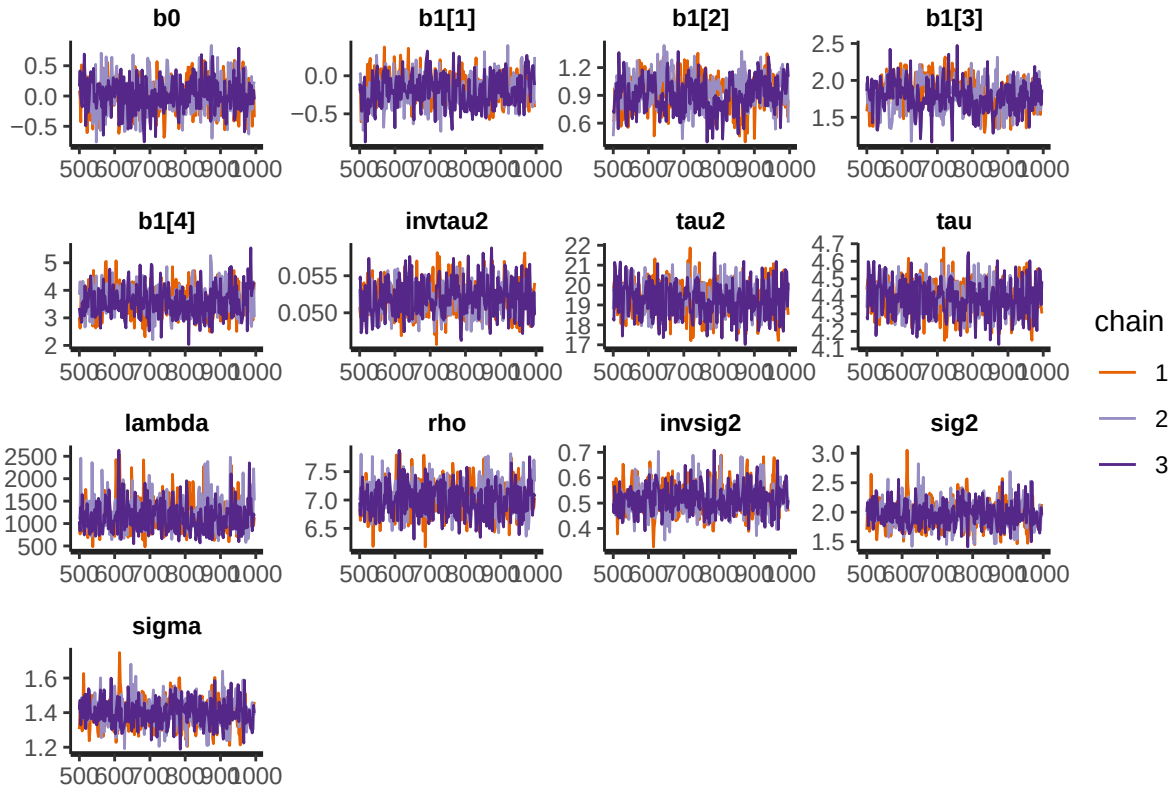
```
fit.lme.add.non.reslope <- stan("jagam_9_aneur_lme_add_non_reslope.stan",
            data=datos.lme.add.non,
            chains=3,warmup=500,iter=1000,thin=2,cores=6,
            init= inits.lme.add.non)
```

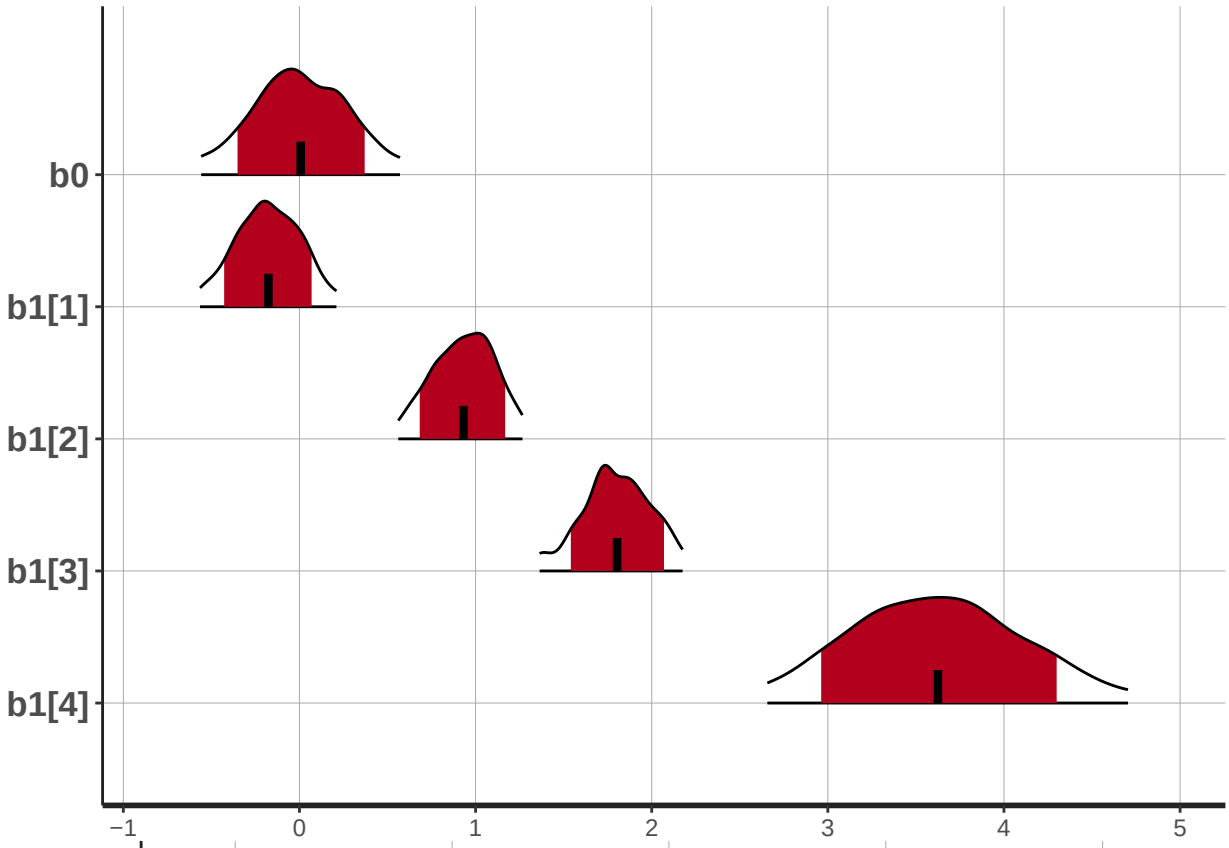
```
print(fit.lme.add.non.reslope, pars=param.lme.add, digits=5)
```

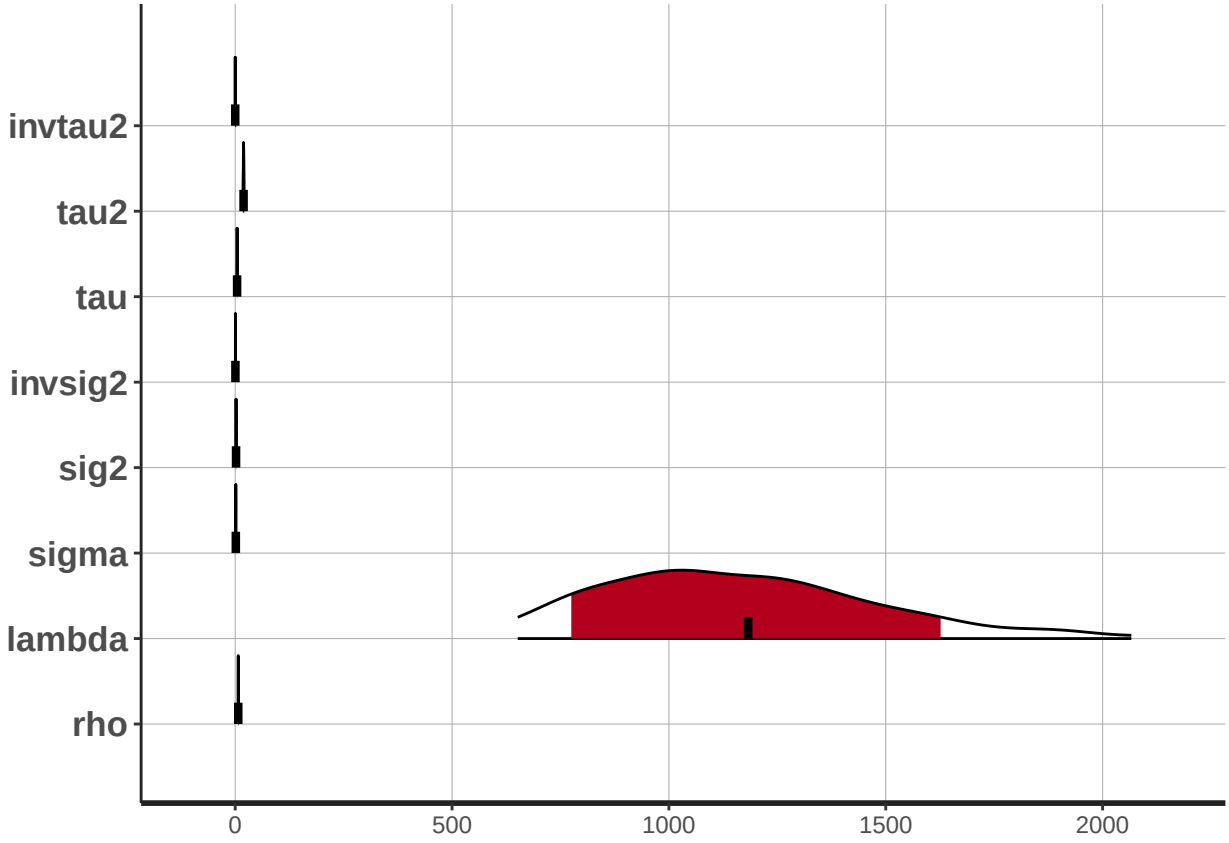
```
## Inference for Stan model: jagam_9_aneur_lme_add_non_reslope.
## 3 chains, each with iter=1000; warmup=500; thin=2;
## post-warmup draws per chain=250, total post-warmup draws=750.
##
##               mean  se_mean        sd      2.5%       25%        50%        75%
## b0         0.00645  0.01116   0.28193  -0.55818  -0.18017   -0.00824    0.20660
## b1[1]     -0.17681  0.01082   0.19754  -0.57032  -0.31189   -0.17659   -0.03990
## b1[2]      0.93244  0.01723   0.18581   0.56522   0.79819    0.94173    1.06738
## b1[3]      1.80348  0.01905   0.20723   1.36371   1.67998    1.80463    1.94244
## b1[4]      3.62571  0.02522   0.52394   2.63619   3.25662    3.61182    3.96627
## invtau2    0.05191  0.00010   0.00224   0.04776   0.05035    0.05184    0.05328
## tau2      19.30049  0.03565   0.82697  17.70262  18.76798   19.28854   19.86125
## tau        4.39222  0.00406   0.09420   4.20745   4.33220    4.39187    4.45660
## lambda  1182.71253 16.54124 353.24060 646.72960 931.38378 1137.21453 1368.79723
## rho        7.03351  0.01338   0.28898   6.47193   6.83667    7.03633    7.22169
## invsig2    0.51525  0.00229   0.05767   0.41268   0.47521    0.51246    0.54996
## sig2       1.96515  0.00885   0.22103   1.55847   1.81831    1.95136    2.10433
## sigma      1.39965  0.00313   0.07835   1.24839   1.34845    1.39691    1.45063
##              97.5% n_eff    Rhat
## b0         0.57039   638 0.99819
## b1[1]      0.20983   333 1.00530
## b1[2]      1.26549   116 1.01307
## b1[3]      2.18503   118 1.02319
## b1[4]      4.70298   432 0.99964
## invtau2    0.05649   537 1.00186
## tau2      20.93845   538 1.00134
## tau        4.57585   538 1.00147
```

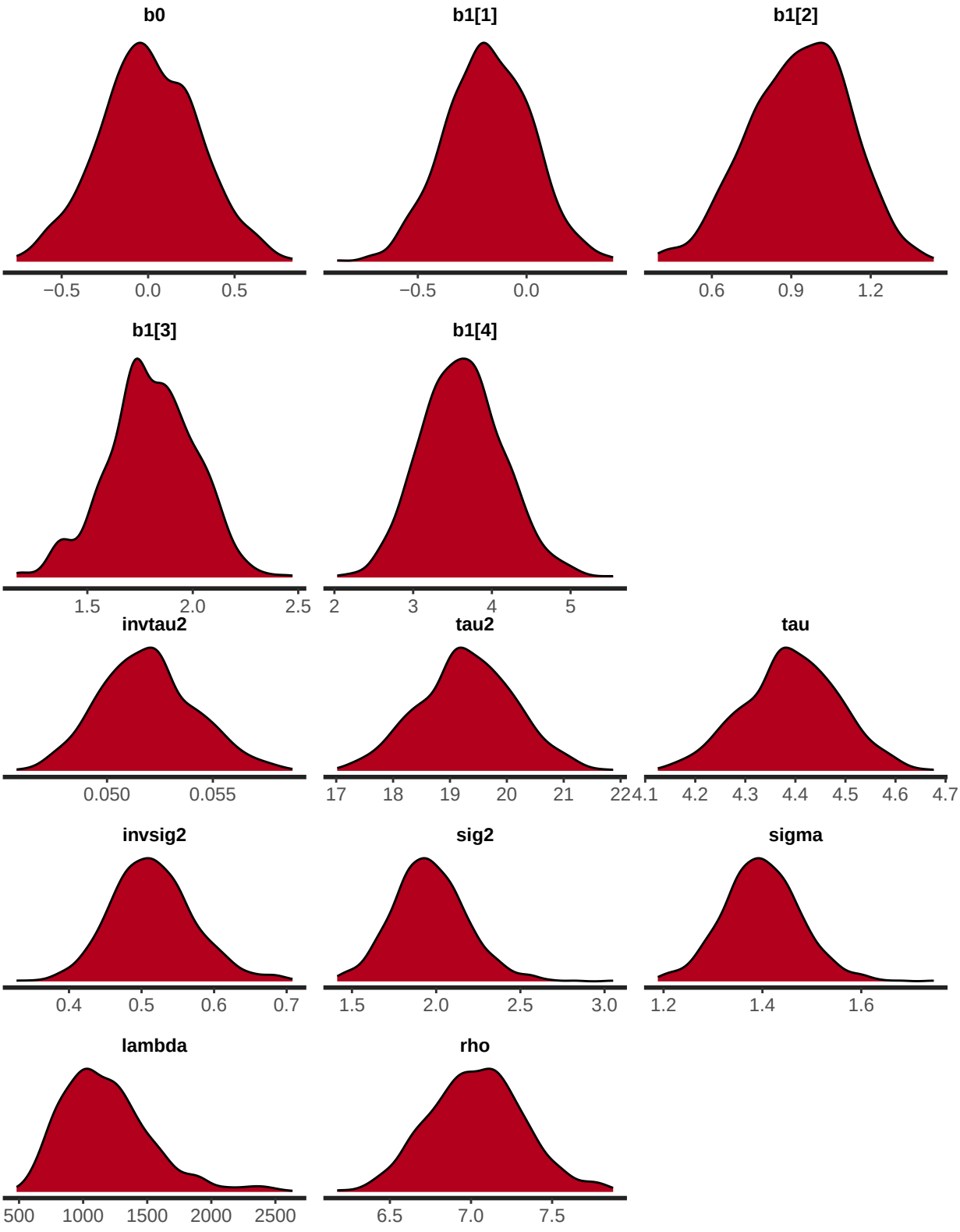
```
## lambda  2064.98819   456 1.00403
## rho        7.63287   466 1.00500
## invsig2    0.64166   634 0.99920
## sig2       2.42320   623 0.99921
## sigma      1.55666   627 0.99923
##
## Samples were drawn using NUTS(diag_e) at Tue Jan  9 14:24:04 2024.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

```
stan_trace(fit.lme.add.non.reslope, pars=param.lme.add)
stan_plot(fit.lme.add.non.reslope, pars=c("b0","b1"), point_est = "mean", show_density = TRUE)
stan_plot(fit.lme.add.non.reslope, pars=c("invtau2","tau2","tau", "invsig2","sig2","sigma",  "lambda","
stan_dens(fit.lme.add.non.reslope, pars=c("b0","b1"))
stan_dens(fit.lme.add.non.reslope, pars=c("invtau2","tau2","tau", "invsig2","sig2","sigma", "lambda","rh
```

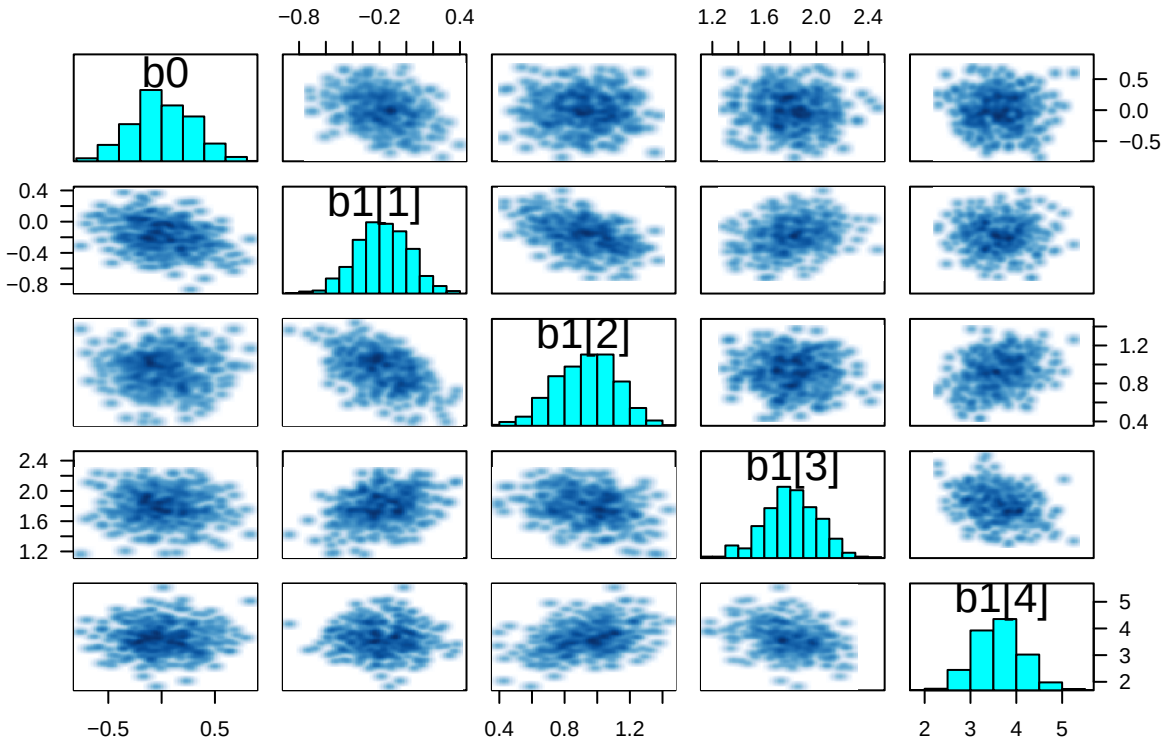



b0
b1[1]
b1[2]
b1[3]
b1[4]
−1
0
1
2
3
4
5

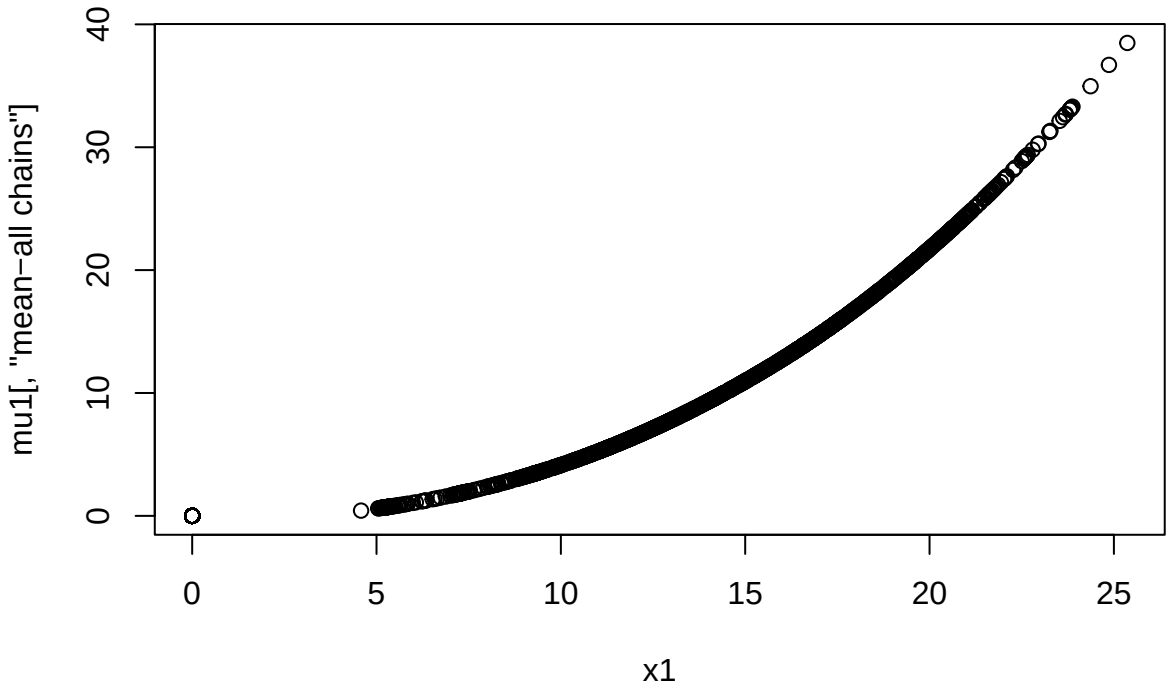

invtau2
tau2
tau
invsig2
sig2
sigma
lambda
rho
0
500
1000
1500
2000



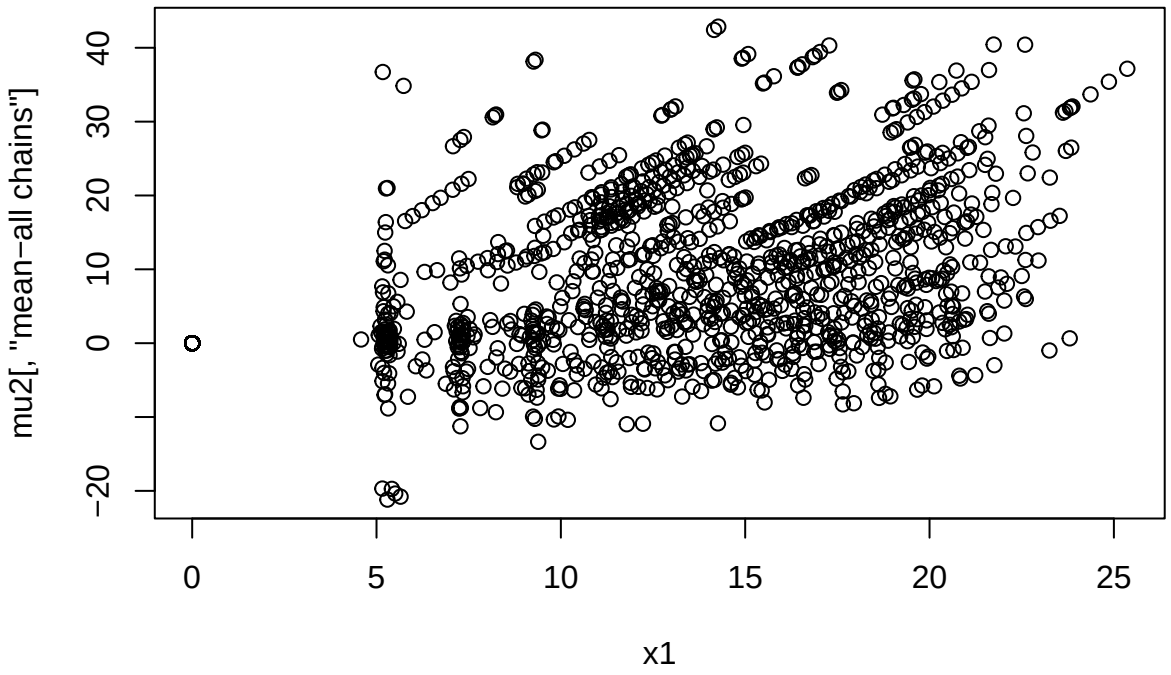

```
pairs(fit.lme.add.non.reslope, pars = c("b0","b1"), las = 1)
```




```
mu1 = get_posterior_mean(fit.lme.add.non.reslope,"mu1")
plot(x1,mu1[,"mean-all chains"])
```




```
mu2 = get_posterior_mean(fit.lme.add.non.reslope,"mu2")
plot(x1,mu2[,"mean-all chains"])
```


mu2[, "mean−all chains"]
−20
0
10
20
30
40
0
5
10
15
20
25
x1

8. Spline con restricciones creciente

8.1. LIN: Spline con restricciones creciente

8.2. LME: Spline con restricciones creciente

```
datos.lme.add.incr <- list( y = y ,
                            id = id ,
                n = length(y) ,
                N = N , Ni = Ni,
                k1=k1,
               XI1 = XI1,
               x1 = x1,
               zero = rep(0,1+k1),
               S1=S1  )
inits.lme.add.incr <- function(){   list(
  "b0" = rnorm(1,0,0.1) ,
  "b1" = abs(rnorm(k1,0,0.1)),
  "invtau2" = rgamma(1,1,1) ,
  "lambda" = rgamma(1,1,1) ,
  "invsig2" = rgamma(1,1,1)
  ) }
```

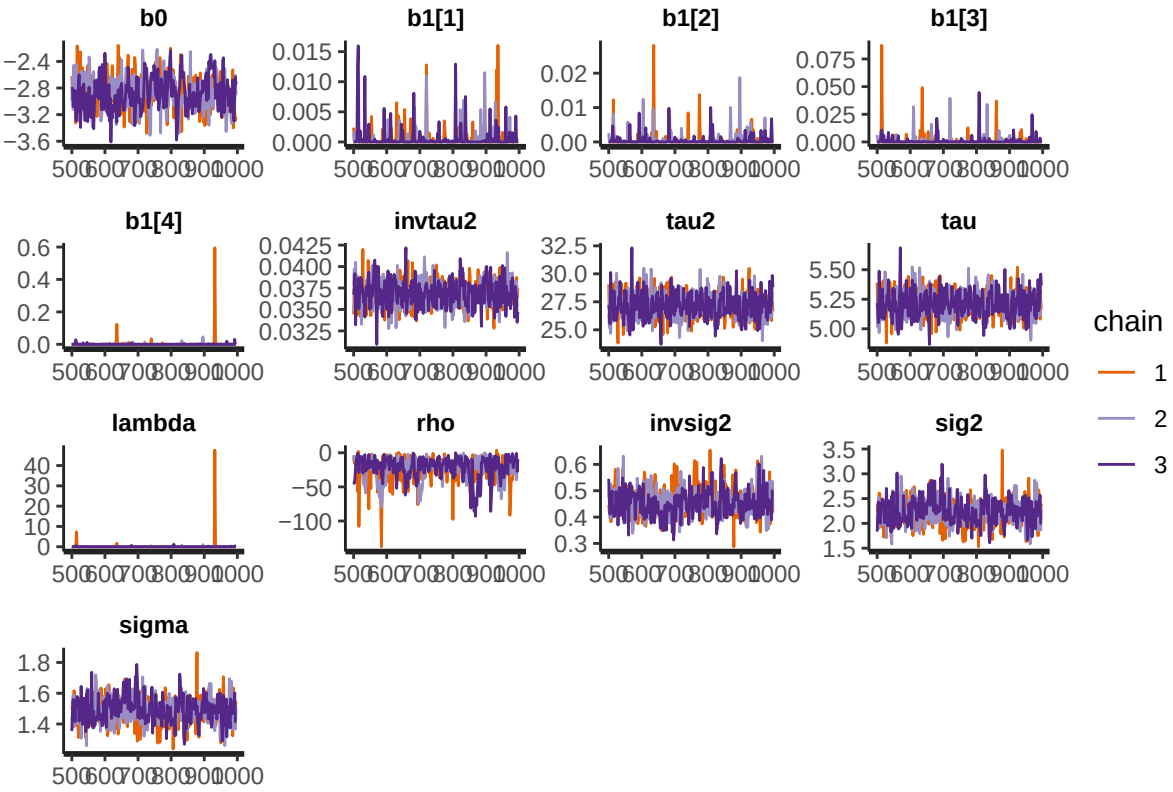
```
fit.lme.add.incr.reslope <- stan("jagam_9_aneur_lme_add_incr_reslope.stan",
            data=datos.lme.add.incr,
            chains=3,warmup=500,iter=1000,thin=2,cores=6,
            init= inits.lme.add.incr)
```

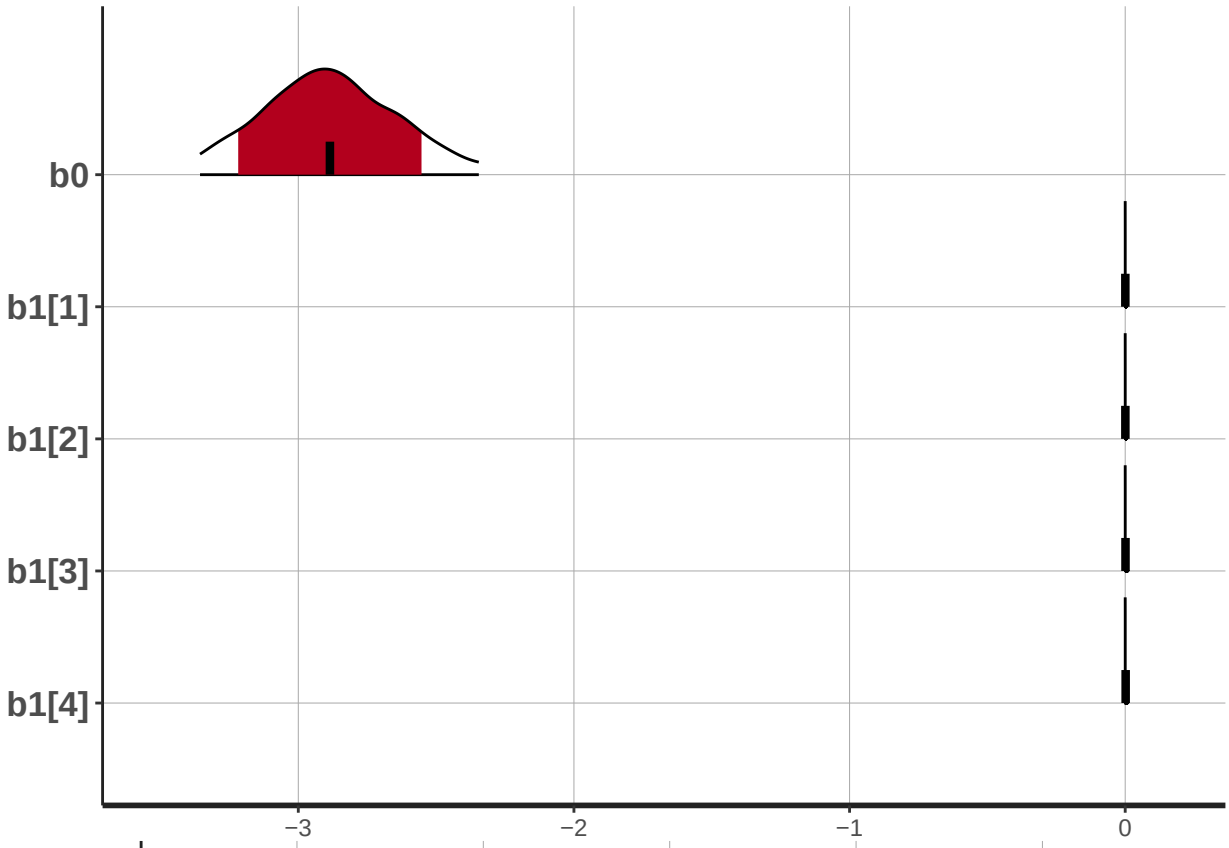
```
print(fit.lme.add.incr.reslope, pars=param.lme.add, digits=5)
```

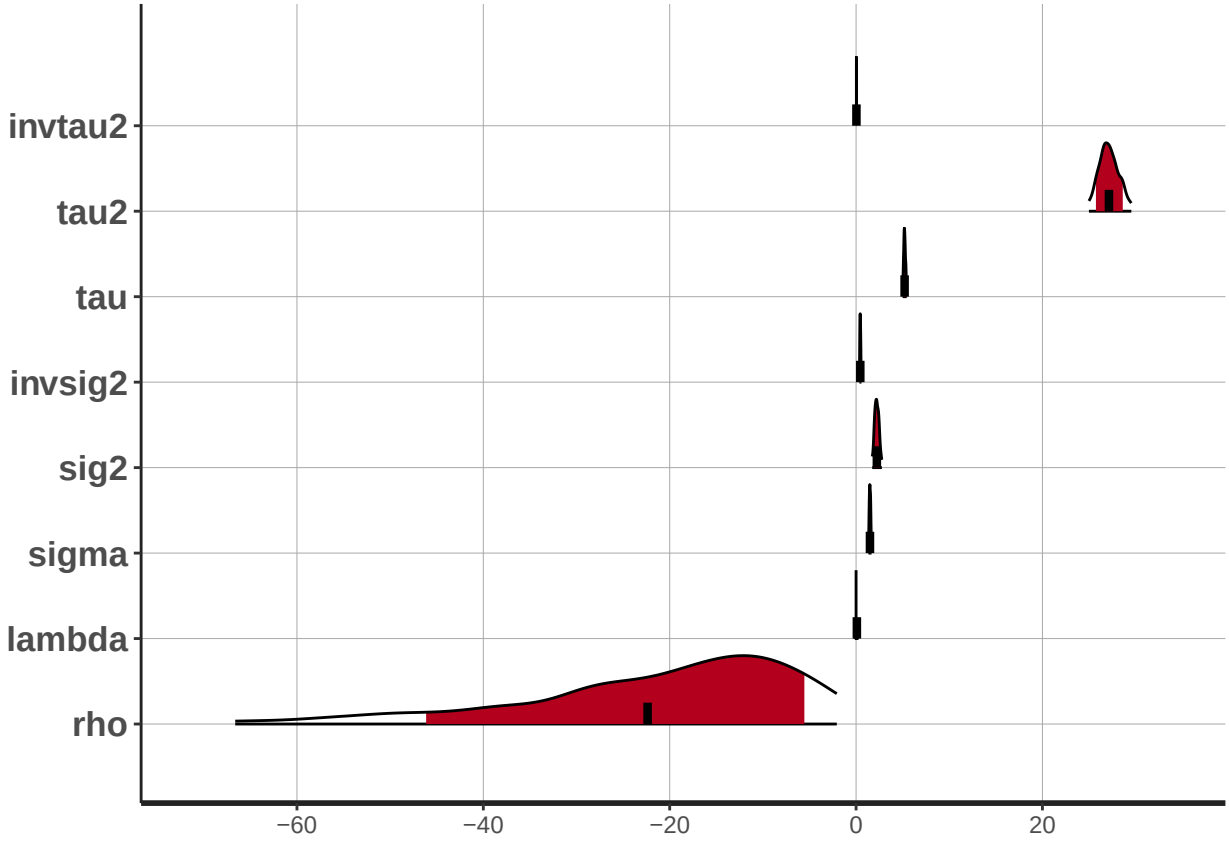
```
## Inference for Stan model: jagam_9_aneur_lme_add_incr_reslope.
## 3 chains, each with iter=1000; warmup=500; thin=2;
## post-warmup draws per chain=250, total post-warmup draws=750.
##
##              mean se_mean       sd      2.5%       25%       50%       75%
## b0       -2.88538 0.01432  0.25326  -3.35610  -3.06045  -2.89640  -2.71761
## b1[1]     0.00050 0.00007  0.00174   0.00000   0.00000   0.00000   0.00011
## b1[2]     0.00050 0.00008  0.00192   0.00000   0.00000   0.00000   0.00010
## b1[3]     0.00104 0.00019  0.00517   0.00000   0.00000   0.00000   0.00011
## b1[4]     0.00165 0.00082  0.02240   0.00000   0.00000   0.00000   0.00010
## invtau2   0.03691 0.00006  0.00154   0.03386   0.03590   0.03694   0.03787
## tau2     27.14214 0.04397  1.14260  24.98860  26.40683  27.07197  27.85748
## tau       5.20866 0.00421  0.10943   4.99886   5.13876   5.20307   5.27802
## lambda    0.08767 0.06396  1.75998   0.00000   0.00000   0.00000   0.00002
## rho     -22.35671 1.22923 17.30482 -66.51326 -29.33107 -18.17919 -10.62172
## invsig2   0.45444 0.00242  0.05333   0.36123   0.41633   0.45104   0.48756
## sig2      2.23053 0.01180  0.26012   1.73718   2.05104   2.21712   2.40197
## sigma     1.49098 0.00395  0.08678   1.31802   1.43214   1.48900   1.54983
##            97.5% n_eff    Rhat
## b0      -2.34516   313 1.00821
## b1[1]    0.00551   630 0.99792
## b1[2]    0.00529   637 0.99812
## b1[3]    0.01003   741 0.99887
## b1[4]    0.00803   755 0.99907
## invtau2  0.04002   673 1.00064
## tau2    29.53686   675 1.00071
```

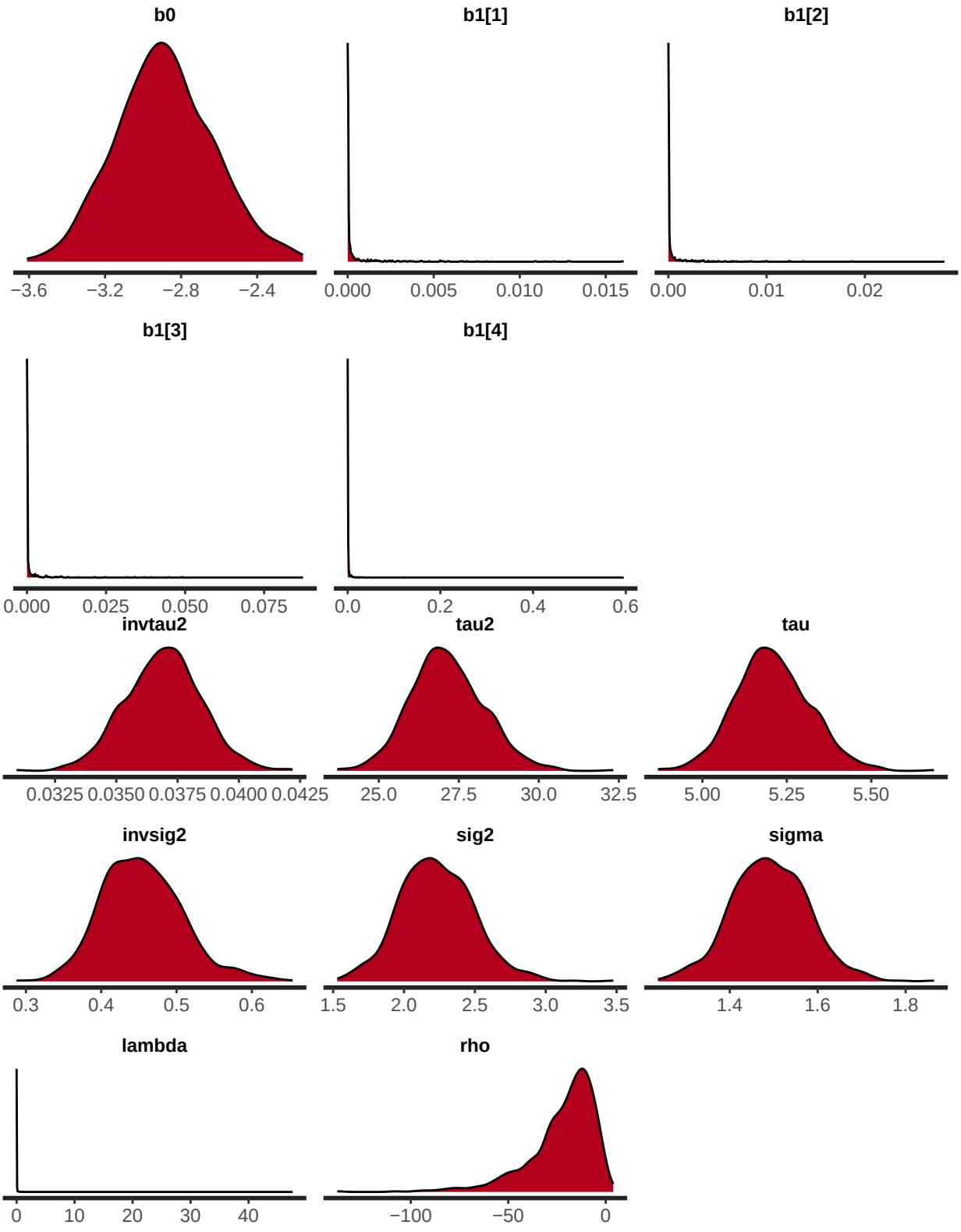
```
## tau       5.43478   675 1.00070
## lambda    0.12528   757 0.99956
## rho      -2.07947   198 1.00598
## invsig2   0.57565   485 1.00374
## sig2      2.76834   486 1.00353
## sig       1.66383   482 1.00358
##
## Samples were drawn using NUTS(diag_e) at Tue Jan  9 14:28:41 2024.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

```
stan_trace(fit.lme.add.incr.reslope, pars=param.lme.add)
stan_plot(fit.lme.add.incr.reslope, pars=c("b0","b1"), point_est = "mean", show_density = TRUE)
stan_plot(fit.lme.add.incr.reslope, pars=c("invtau2","tau2","tau", "invsig2","sig2","sigma",  "lambda","
stan_dens(fit.lme.add.incr.reslope, pars=c("b0","b1"))
stan_dens(fit.lme.add.incr.reslope, pars=c("invtau2","tau2","tau", "invsig2","sig2","sigma", "lambda","
```

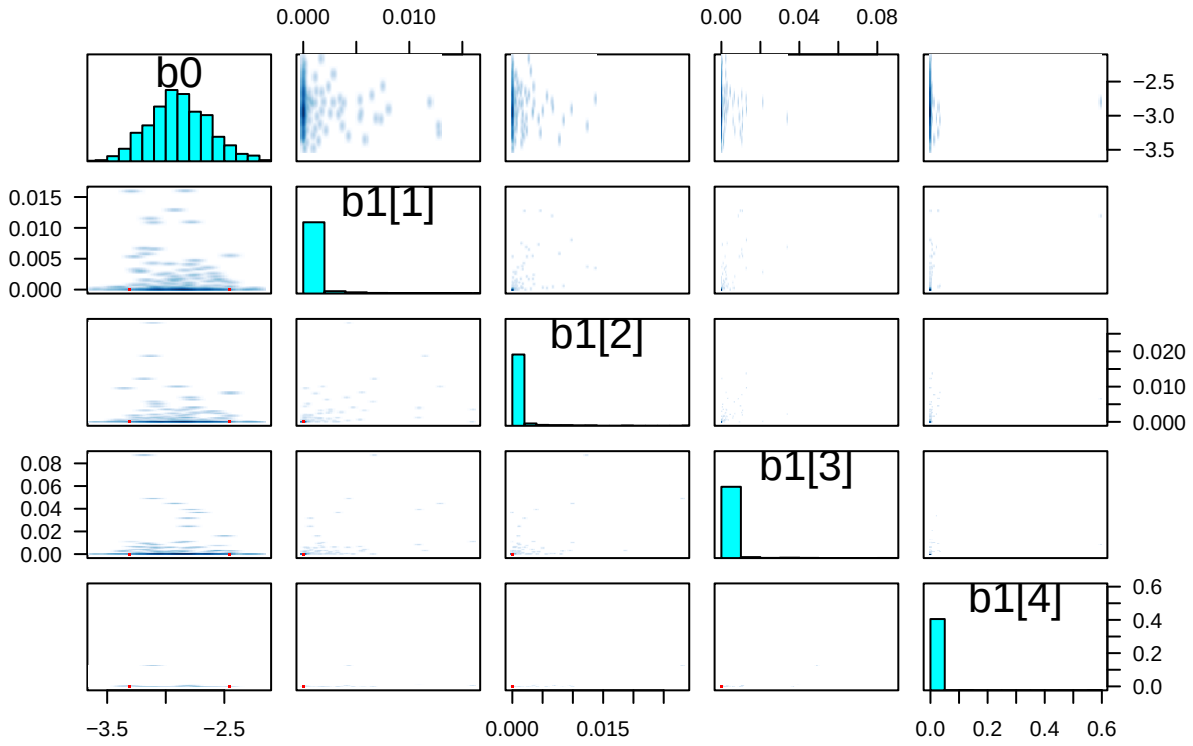



b0
b1[1]
b1[2]
b1[3]
b1[4]
−3
−2
−1
0

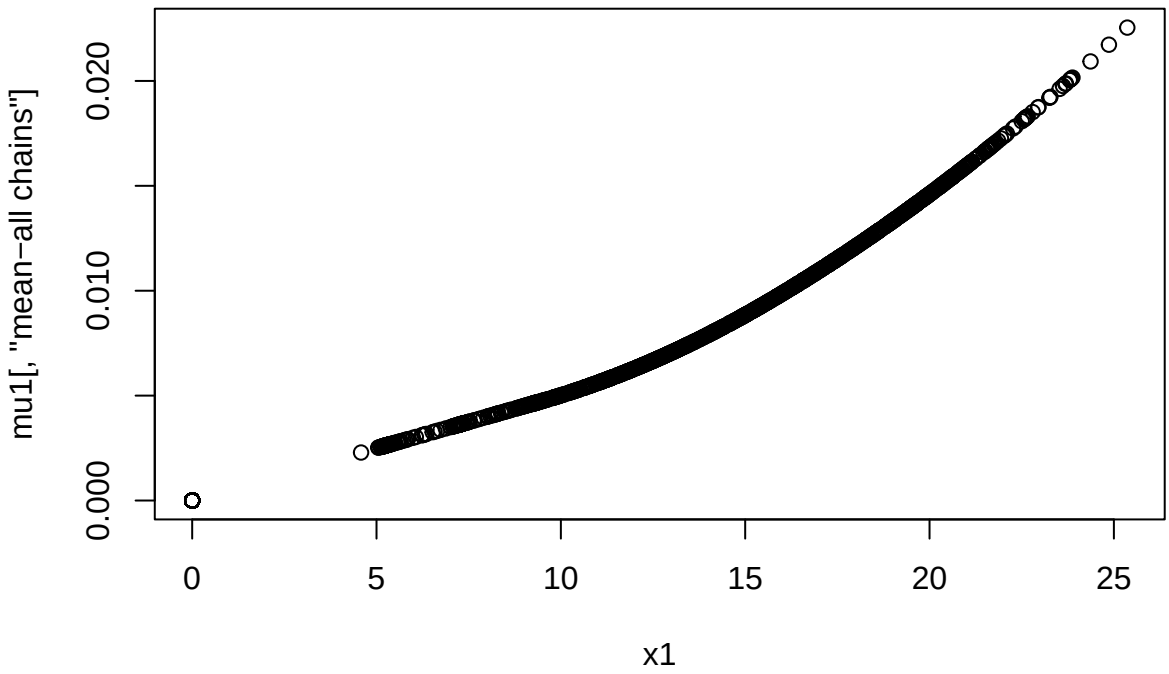

invtau2
tau2
tau
invsig2
sig2
sigma
lambda
rho
−60
−40
−20
0
20



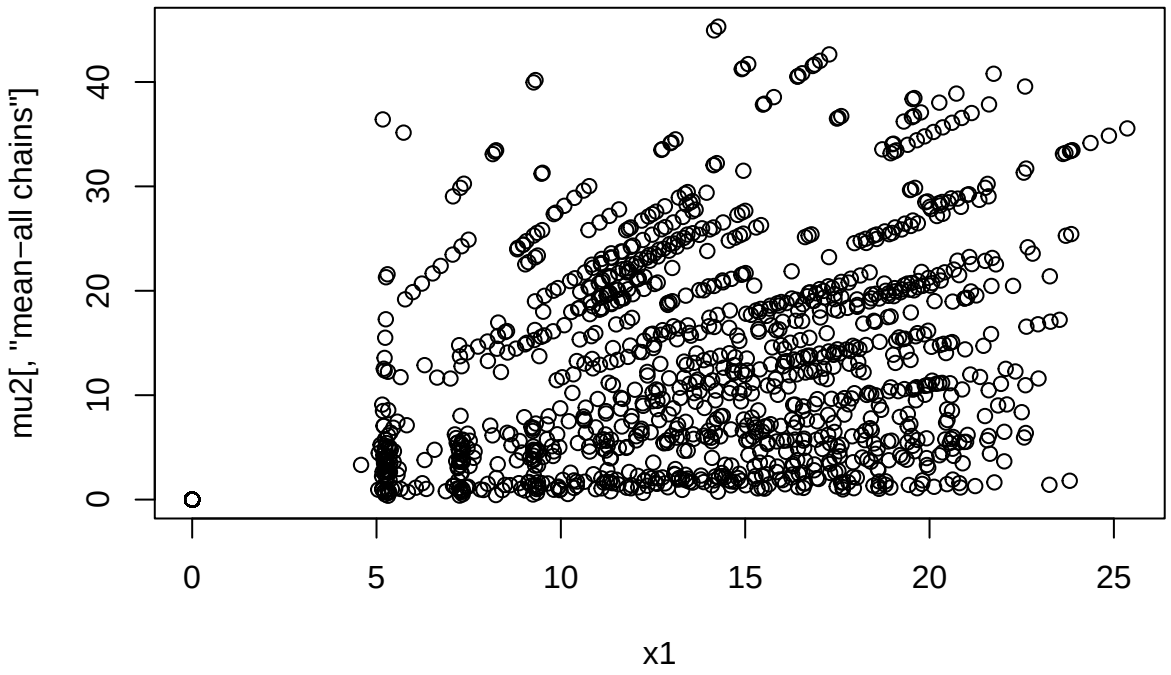

```
pairs(fit.lme.add.incr.reslope, pars = c("b0","b1"), las = 1)
```




```
mu1 = get_posterior_mean(fit.lme.add.incr.reslope,"mu1")
plot(x1,mu1[,"mean-all chains"])
```




```
mu2 = get_posterior_mean(fit.lme.add.incr.reslope,"mu2")
plot(x1,mu2[,"mean-all chains"])
```


mu2[, "mean−all chains"]
0
10
20
30
40
0
5
10
15
20
25
x1

Comparar

```
### http://ritsokiguess.site/docs/2019/06/25/going-to-the-loo-using-stan-for-model-comparison/
library(loo)
```

```
## This is loo version 2.4.1

## - Online documentation and vignettes at mc-stan.org/loo

## - As of v2.0.0 loo defaults to 1 core but we recommend using as many as possible. Use the 'cores' ar

##
## Attaching package: 'loo'

## The following object is masked from 'package:rstan':
##
##     loo
```

```
loo3_sample = fit.lme.non.reslope
loo6_sample = fit.lme.incr.reslope
loo9_sample = fit.lme.add.non.reslope
loo12_sample = fit.lme.add.incr.reslope

### we have to extract those log-likelihood terms that we so carefully had Stan calculate for us:
log_lik_3 =extract_log_lik(loo3_sample, merge_chains = F)
log_lik_6 =extract_log_lik(loo6_sample, merge_chains = F)
log_lik_9 =extract_log_lik(loo9_sample, merge_chains = F)
log_lik_12 =extract_log_lik(loo12_sample, merge_chains = F)

r_eff_3 =relative_eff(log_lik_3)
r_eff_6 =relative_eff(log_lik_6)
r_eff_9 =relative_eff(log_lik_9)
r_eff_12 =relative_eff(log_lik_12)
```

```
###  look at the results for each model, first the one with mu estimated:
(loo_3 <- loo(log_lik_3, r_eff=r_eff_3))
```

```
## Warning: Some Pareto k diagnostic values are too high. See help('pareto-k-diagnostic') for details.

##
## Computed from 750 by 1387 log-likelihood matrix
##
##          Estimate    SE
## elpd_loo  -4290.7  57.7
## p_loo       253.3  23.0
## looic      8581.4 115.4
## ------
## Monte Carlo SE of elpd_loo is NA.
##
## Pareto k diagnostic values:
```

```
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     1256  90.6%   78
##  (0.5, 0.7]   (ok)         86   6.2%   36
##    (0.7, 1]   (bad)        29   2.1%   7
##    (1, Inf)   (very bad)   16   1.2%   1
## See help('pareto-k-diagnostic') for details.
```

```
(loo_6 <- loo(log_lik_6, r_eff=r_eff_6))
```

```
## Warning: Some Pareto k diagnostic values are too high. See help('pareto-k-diagnostic') for details.
```

```
##
## Computed from 750 by 1387 log-likelihood matrix
##
##          Estimate    SE
## elpd_loo  -4359.2  59.9
## p_loo       156.5  14.6
## looic      8718.4 119.7
## ------
## Monte Carlo SE of elpd_loo is NA.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     1283  92.5%   60
##  (0.5, 0.7]   (ok)         71   5.1%   51
##    (0.7, 1]   (bad)        23   1.7%   21
##    (1, Inf)   (very bad)   10   0.7%   1
## See help('pareto-k-diagnostic') for details.
```

```
(loo_9 <- loo(log_lik_9, r_eff=r_eff_9))
```

```
## Warning: Some Pareto k diagnostic values are too high. See help('pareto-k-diagnostic') for details.
```

```
##
## Computed from 750 by 1387 log-likelihood matrix
##
##          Estimate    SE
## elpd_loo  -4183.5  61.5
## p_loo       263.1  22.9
## looic      8367.0 122.9
## ------
## Monte Carlo SE of elpd_loo is NA.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     1246  89.8%   70
##  (0.5, 0.7]   (ok)         87   6.3%   20
##    (0.7, 1]   (bad)        39   2.8%   9
##    (1, Inf)   (very bad)   15   1.1%   1
## See help('pareto-k-diagnostic') for details.
```

```
(loo_12 <- loo(log_lik_12, r_eff=r_eff_12))
```

```
## Warning: Some Pareto k diagnostic values are too high. See help('pareto-k-diagnostic') for details.
```

```
##
## Computed from 750 by 1387 log-likelihood matrix
##
##          Estimate    SE
## elpd_loo  -4352.3  59.5
## p_loo       151.6  13.8
## looic      8704.5 119.1
## ------
## Monte Carlo SE of elpd_loo is NA.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     1293  93.2%   114
##  (0.5, 0.7]   (ok)         62   4.5%   47
##    (0.7, 1]   (bad)        26   1.9%   10
##    (1, Inf)   (very bad)    6   0.4%   2
## See help('pareto-k-diagnostic') for details.
```

```
#compare(loo_1, loo_2)
### The second model fits better than the first one, since its looic is smaller.
```

```
###  look at the results for each model, first the one with mu estimated:
(waic_3 <- waic(log_lik_3, r_eff=r_eff_3))
```

```
## Warning:
## 102 (7.4%) p_waic estimates greater than 0.4. We recommend trying loo instead.
```

```
##
## Computed from 750 by 1387 log-likelihood matrix
##
##           Estimate    SE
## elpd_waic  -4276.0  57.0
## p_waic       238.6  21.5
## waic        8552.0 114.0
##
## 102 (7.4%) p_waic estimates greater than 0.4. We recommend trying loo instead.
```

```
(waic_6 <- waic(log_lik_6, r_eff=r_eff_6))
```

```
## Warning:
## 75 (5.4%) p_waic estimates greater than 0.4. We recommend trying loo instead.
```

```
##
## Computed from 750 by 1387 log-likelihood matrix
##
##           Estimate    SE
```

```
## elpd_waic  -4348.3  58.8
## p_waic       145.6  13.0
## waic        8696.5 117.7
##
## 75 (5.4%) p_waic estimates greater than 0.4. We recommend trying loo instead.
```

```
(waic_9 <- waic(log_lik_9, r_eff=r_eff_9))
```

```
## Warning:
## 114 (8.2%) p_waic estimates greater than 0.4. We recommend trying loo instead.

##
## Computed from 750 by 1387 log-likelihood matrix
##
##           Estimate    SE
## elpd_waic  -4172.0  61.8
## p_waic       251.6  23.0
## waic        8344.0 123.5
##
## 114 (8.2%) p_waic estimates greater than 0.4. We recommend trying loo instead.
```

```
(waic_12 <- waic(log_lik_12, r_eff=r_eff_12))
```

```
## Warning:
## 70 (5.0%) p_waic estimates greater than 0.4. We recommend trying loo instead.

##
## Computed from 750 by 1387 log-likelihood matrix
##
##           Estimate    SE
## elpd_waic  -4342.1  58.6
## p_waic       141.4  12.1
## waic        8684.2 117.2
##
## 70 (5.0%) p_waic estimates greater than 0.4. We recommend trying loo instead.
```

```
#compare(waic_1, waic_2)
### The second model fits better than the first one, since its looic is smaller.
```