T downscaling TMax month 01 model

Viacheslav Shalisko

16 de abril de 2018

```
# random numbers seed
set.seed(1)

rpoints_mask1_number <- 10000
rpoints_mask2_number <- 10000

# month to be analysed (in current version the analysis is per month)
month <- 1

# dependent variable to be analysed
var_dependent <- "tmax"
var_dependent_name <- "Maximum temperature"

# geographic coordinate system string
mi_crs <- "+proj=longlat +ellps=WGS84 +datum=WGS84 +no_defs"

# map extent
# all data should be in geographic coodinates
ext_vector <- c(-120,-30,-50,50)

# output directory (relative to curretnt dir)
output_path <- 'C:/Users/vshal/Downloads/SDM_interpolations/models'

# name of low resolution source dependent variable
r_lores_name <- "C:/Users/vshal/Downloads/SDM_interpolations/wc2_30s_tmax_01_Amer_F1.tif"

# name of high resolution predictor variable
r_predictor_name <- "C:/Users/vshal/Downloads/SDM_recortes/gmted_med075_Amer.tif"

# names of first and second high resolution masks
r_mask1_name <- "C:/Users/vshal/Downloads/SDM_recortes/Continents_Amer_S1a.tif"
r_mask2_name <- "C:/Users/vshal/Downloads/SDM_recortes/Continents_Amer_S2a.tif"
```

```
knitr::opts_chunk$set(echo = TRUE)
knitr::opts_chunk$set(error = TRUE)

library(raster)       # raster processing
```

```
## Loading required package: sp
```

```
library(maptools)     # map processing
```

```
## Checking rgeos availability: FALSE
##      Note: when rgeos is not available, polygon geometry     computations in maptools depend on gpclib,
##      which has a restricted licence. It is disabled by default;
##      to enable gpclib, type gpclibPermit()
```

```
library(rworldmap)    # worldmap datasets
```

```
## Warning: package 'rworldmap' was built under R version 3.4.2
```

```
## ### Welcome to rworldmap ###
```

```
## For a short introduction type :   vignette('rworldmap')
```

```
library(rworldxtra)   # hires worldmap spatial dataframe
```

```
## Warning: package 'rworldxtra' was built under R version 3.4.2
```

```
library(dismo)       # SDM, here used to generate random points
library(mgcv)        # GAM models
```

```
## Loading required package: nlme
```

```
##
## Attaching package: 'nlme'
```

```
## The following object is masked from 'package:raster':
##
##     getData
```

```
## This is mgcv 1.8-17. For overview type 'help("mgcv-package")'.
```

```
r_lores <- raster(r_lores_name)
r_predictor <- raster(r_predictor_name)
r_mask1 <- raster(r_mask1_name)
r_mask2 <- raster(r_mask2_name)
```

```
random_points_mask1 <- randomPoints(r_mask1, n = rpoints_mask1_number, tryf = 3)
```

```
## Warning in randomPoints(r_mask1, n = rpoints_mask1_number, tryf = 3):
## generated random points = 0.8891 times requested number
```

```
random_points_mask2 <- randomPoints(r_mask2, n = rpoints_mask2_number, tryf = 5)
```

```
## Warning in randomPoints(r_mask2, n = rpoints_mask2_number, tryf = 5):
## generated random points = 0.5247 times requested number
```

```
dim(random_points_mask1)
```

```
## [1] 8891    2
```

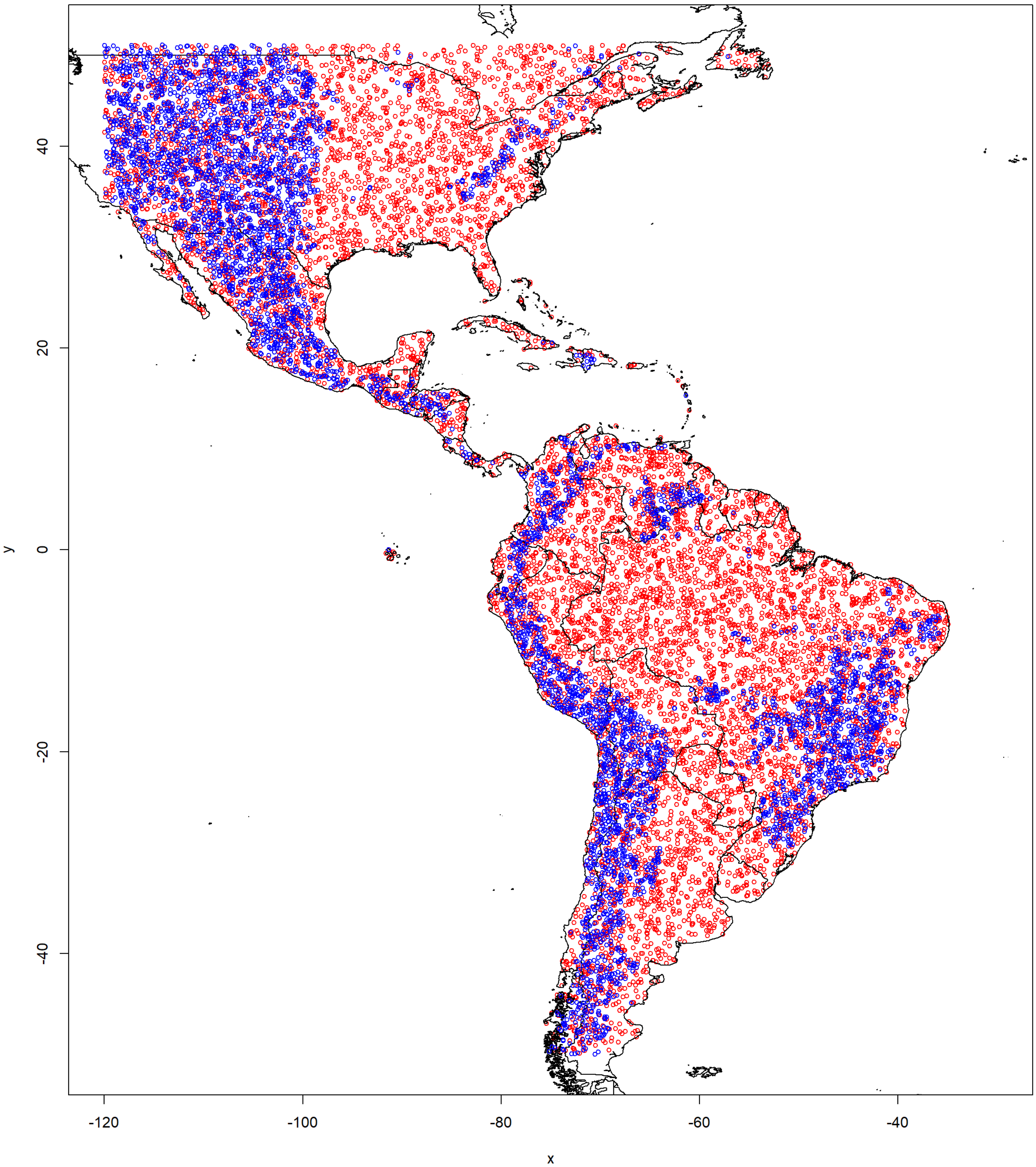
```
dim(random_points_mask2)
```

```
## [1] 5247    2
```

```
random_points_all <- rbind(random_points_mask1,random_points_mask2)
```

```
# background continents
plot(random_points_mask1, col='red',cex=0.7, xlim=ext_vector[1:2], ylim=ext_vector[3:4], axes=TRUE)
world_high <- getMap(resolution = "high")
plot(world_high, add=TRUE)

# render random points
points(random_points_mask2,col='blue',cex=0.7)
```

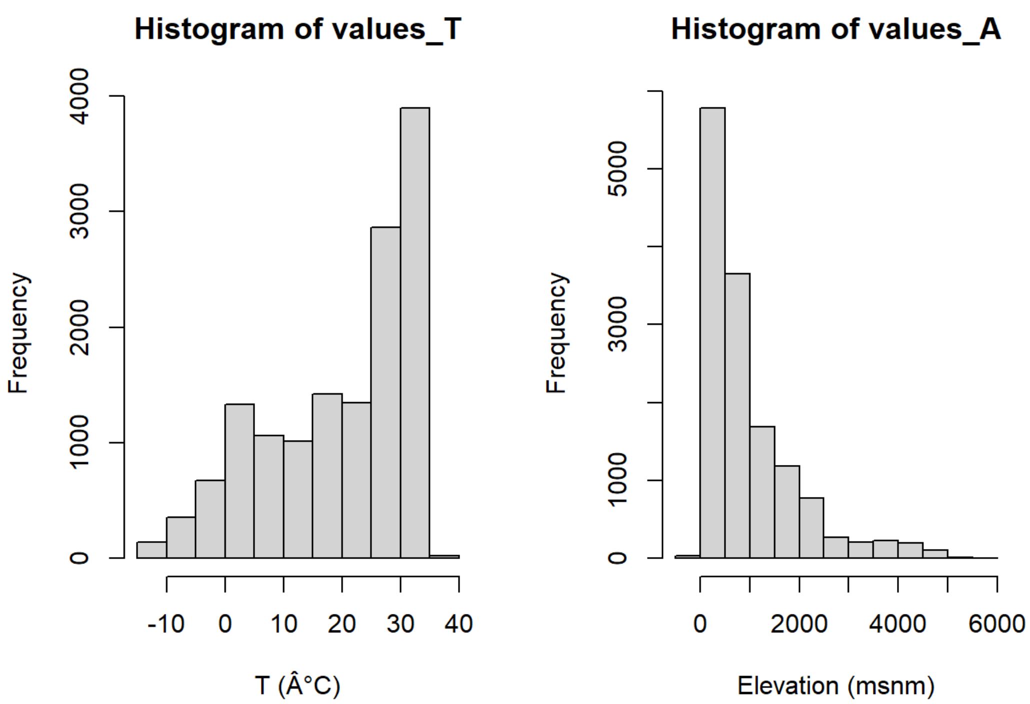



```
values_T <- extract(r_lores, random_points_all)
values_A <- extract(r_predictor, random_points_all)
values_Y <- random_points_all[,"y"]
values_X <- random_points_all[,"x"]

# remove NAs
intermediate_frame <- data.frame(values_X, values_Y, values_T, values_A)
intermediate_frame <- na.omit(intermediate_frame)

values_T <- as.vector(intermediate_frame[,3])
values_A <- as.vector(intermediate_frame[,4])
values_Y <- as.vector(intermediate_frame[,2])
values_X <- as.vector(intermediate_frame[,1])
```

```
par(mfrow=c(1, 2))
hist(values_T, col = "lightgray", xlab = "T (Â°C)")
hist(values_A, col = "lightgray", xlab = "Elevation (msnm)")
```



Simple models

```
mod_L1 <- lm(values_T ~ values_A)
summary(mod_L1)
```

```
##
## Call:
## lm(formula = values_T ~ values_A)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -35.109  -9.081   6.016   8.994  15.947
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 22.8615215  0.1382272  165.39   <2e-16 ***
## values_A    -0.0033763  0.0001017  -33.21   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11.83 on 14136 degrees of freedom
## Multiple R-squared:  0.07236,    Adjusted R-squared:  0.0723
## F-statistic:  1103 on 1 and 14136 DF,  p-value: < 2.2e-16
```

```
AIC(mod_L1)
```

```
## [1] 109976.9
```

```
mod_L2 <- lm(values_T ~ values_A + values_Y)
summary(mod_L2)
```

```
##
## Call:
## lm(formula = values_T ~ values_A + values_Y)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -32.471  -3.423   1.494   5.058  14.867
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.519e+01  8.605e-02  292.68   <2e-16 ***
## values_A    -3.306e-03  6.231e-05  -53.05   <2e-16 ***
## values_Y    -3.348e-01  2.184e-03 -153.32   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.247 on 14135 degrees of freedom
## Multiple R-squared:  0.6517, Adjusted R-squared:  0.6516
## F-statistic: 1.322e+04 on 2 and 14135 DF,  p-value: < 2.2e-16
```

```
AIC(mod_L2)
```

```
## [1] 96131.49
```

```
mod_L3 <- lm(values_T ~ values_A + values_Y + I(values_Y^2))
summary(mod_L3)
```

```
##
## Call:
## lm(formula = values_T ~ values_A + values_Y + I(values_Y^2))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.5600  -1.8789  -0.0641   2.1755   8.8632
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)    3.221e+01  5.005e-02   643.5   <2e-16 ***
## values_A      -2.947e-03  2.871e-05  -102.6   <2e-16 ***
## values_Y      -1.953e-01  1.175e-03  -166.3   <2e-16 ***
## I(values_Y^2) -1.007e-02  4.389e-05  -229.4   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.335 on 14134 degrees of freedom
## Multiple R-squared:  0.9262, Adjusted R-squared:  0.9262
## F-statistic: 5.917e+04 on 3 and 14134 DF,  p-value: < 2.2e-16
```

```
AIC(mod_L3)
```

```
## [1] 74184.97
```

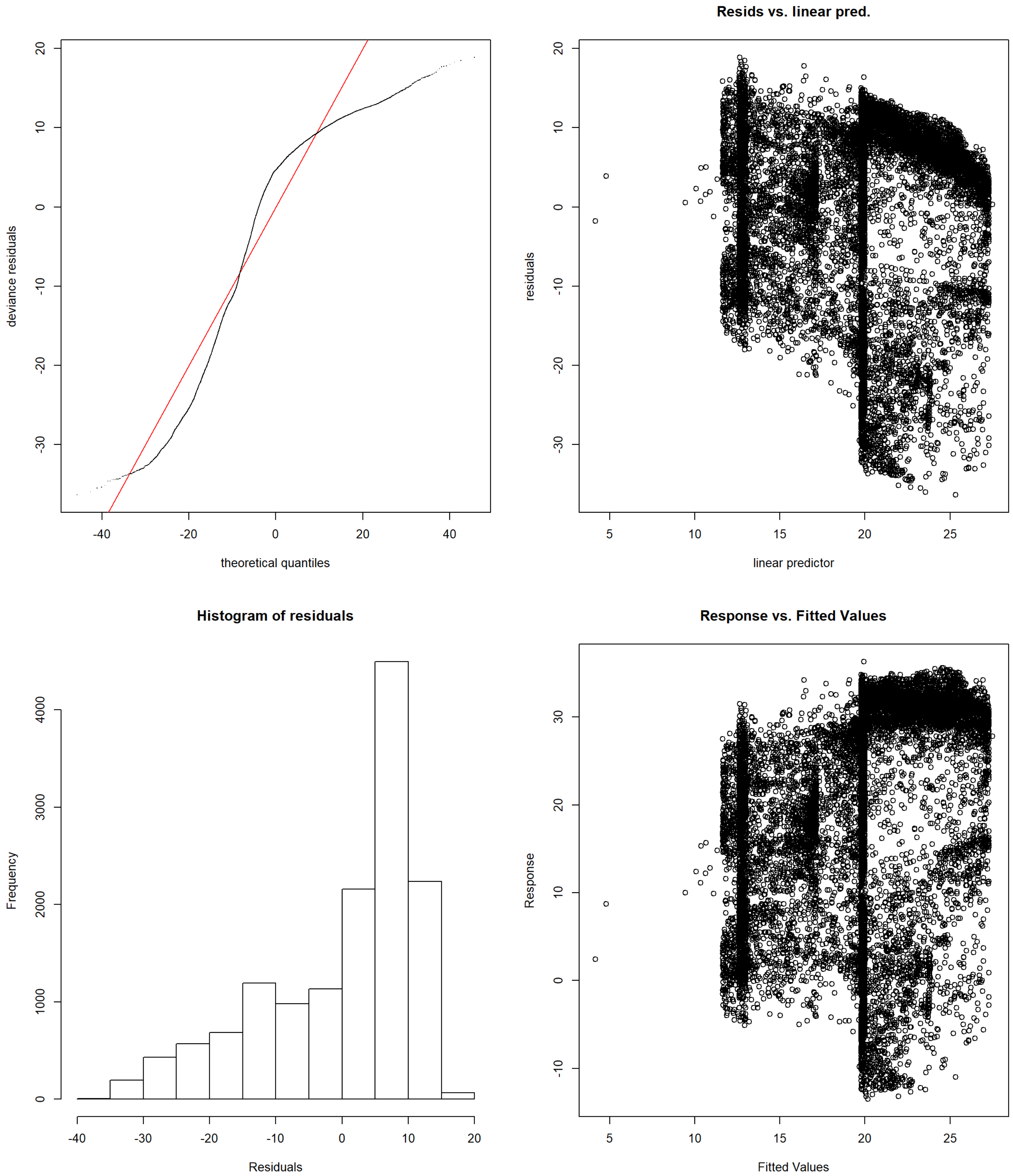
```
mod_A1 <- gam(values_T ~ s(values_A))
summary(mod_A1)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## values_T ~ s(values_A)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 19.67396    0.09681   203.2   <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(values_A) 8.703  8.973 217.7  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.121   Deviance explained = 12.2%
## GCV = 132.59  Scale est. = 132.5     n = 14138
```

```
AIC(mod_A1)
```

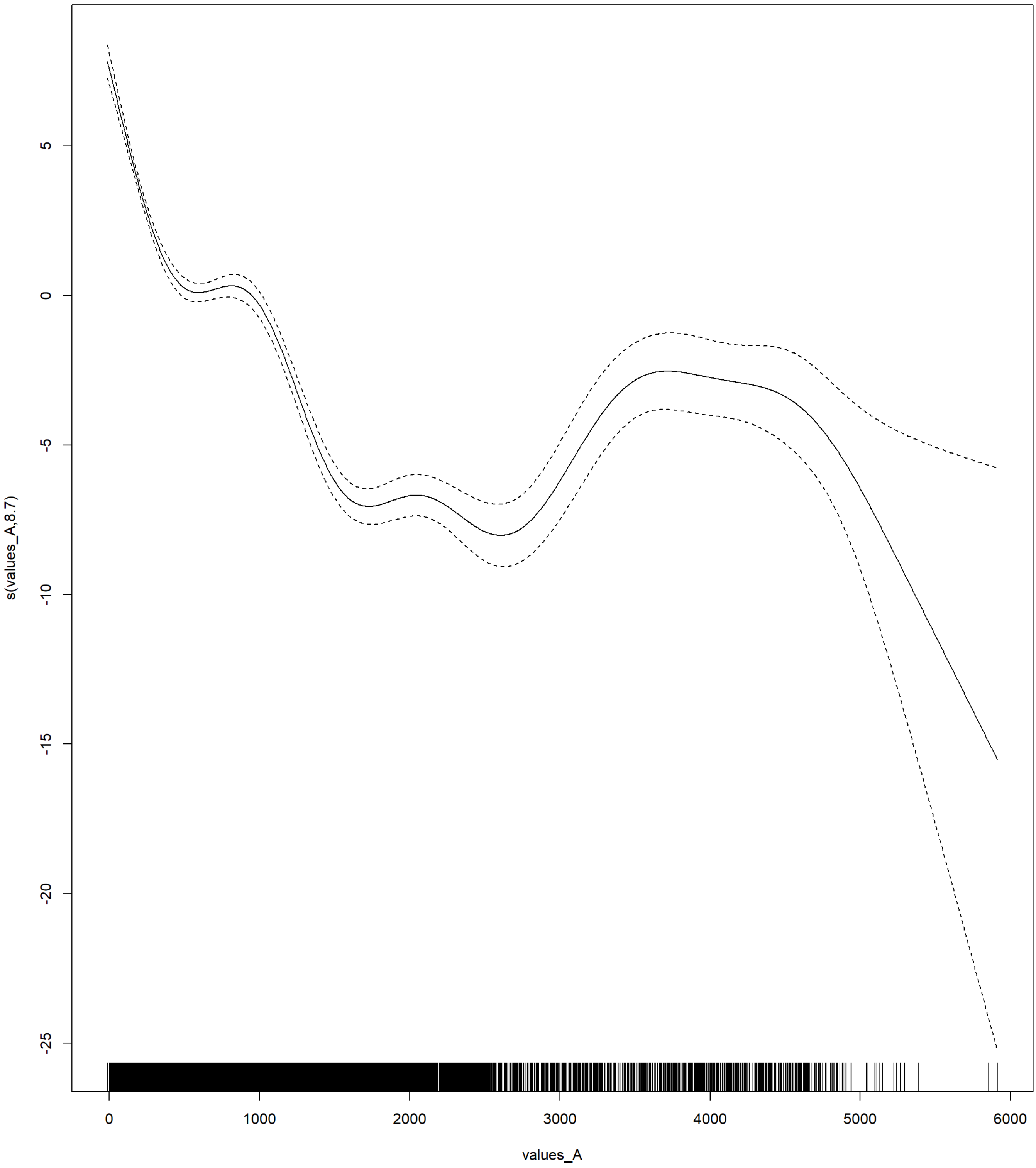
```
## [1] 109220.2
```

```
gam.check(mod_A1)
```




```
##
## Method: GCV   Optimizer: magic
## Smoothing parameter selection converged after 7 iterations.
## The RMS GCV score gradient at convergence was 9.729613e-05 .
## The Hessian was positive definite.
## Model rank =  10 / 10
##
## Basis dimension (k) checking results. Low p-value (k-index<1) may
## indicate that k is too low, especially if edf is close to k'.
##
##              k' edf k-index p-value
## s(values_A) 9.0 8.7    0.97   0.015 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
plot(mod_A1,n=1000)
```



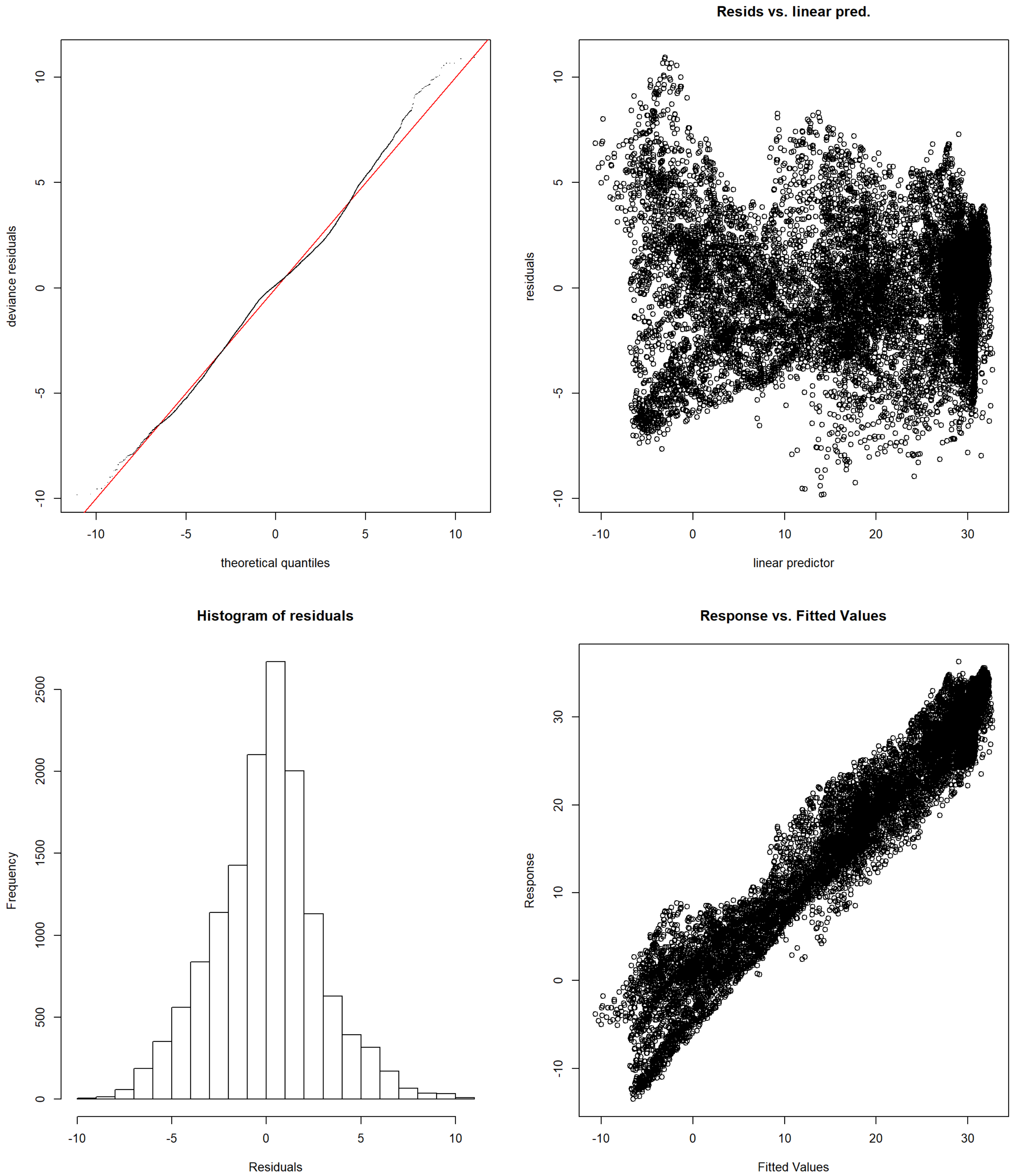

```
mod_A2 <- gam(values_T ~ s(values_A) + s(values_Y))
summary(mod_A2)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## values_T ~ s(values_A) + s(values_Y)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  19.6740     0.0234   840.6   <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(values_A) 8.163  8.802  2037  <2e-16 ***
## s(values_Y) 8.897  8.997 25281  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.949   Deviance explained = 94.9%
## GCV = 7.7536  Scale est. = 7.7437    n = 14138
```

```
AIC(mod_A2)
```

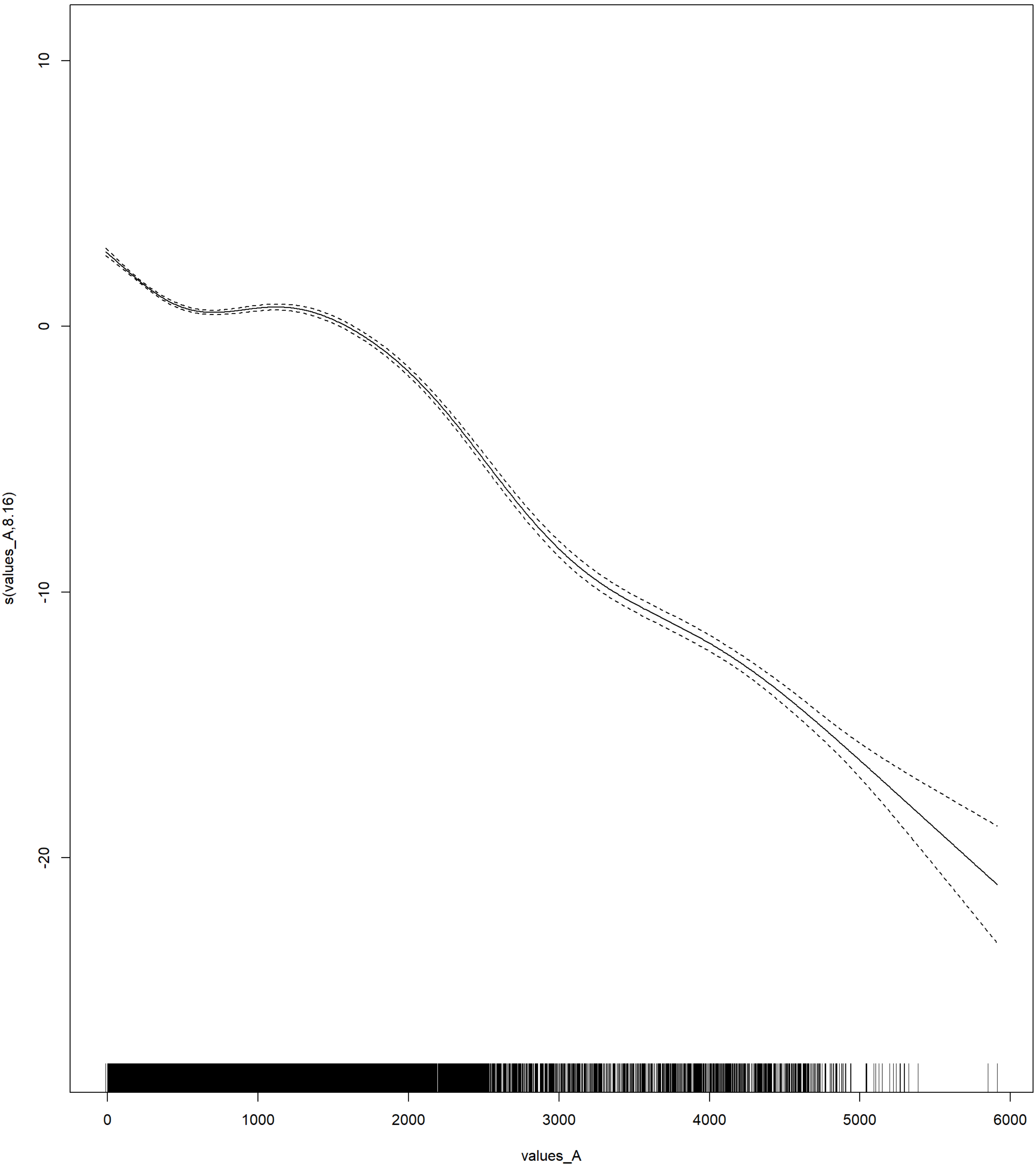
```
## [1] 69080.8
```

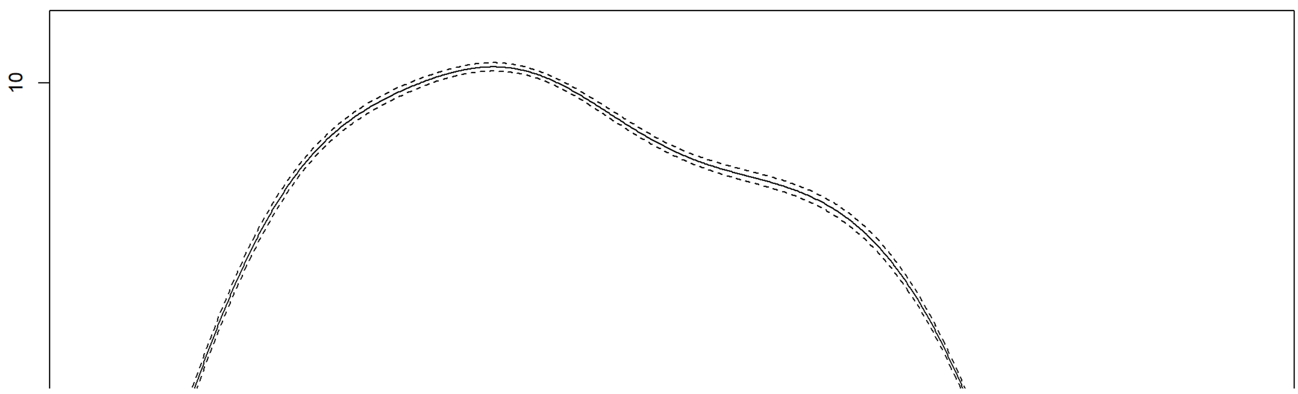
```
gam.check(mod_A2)
```

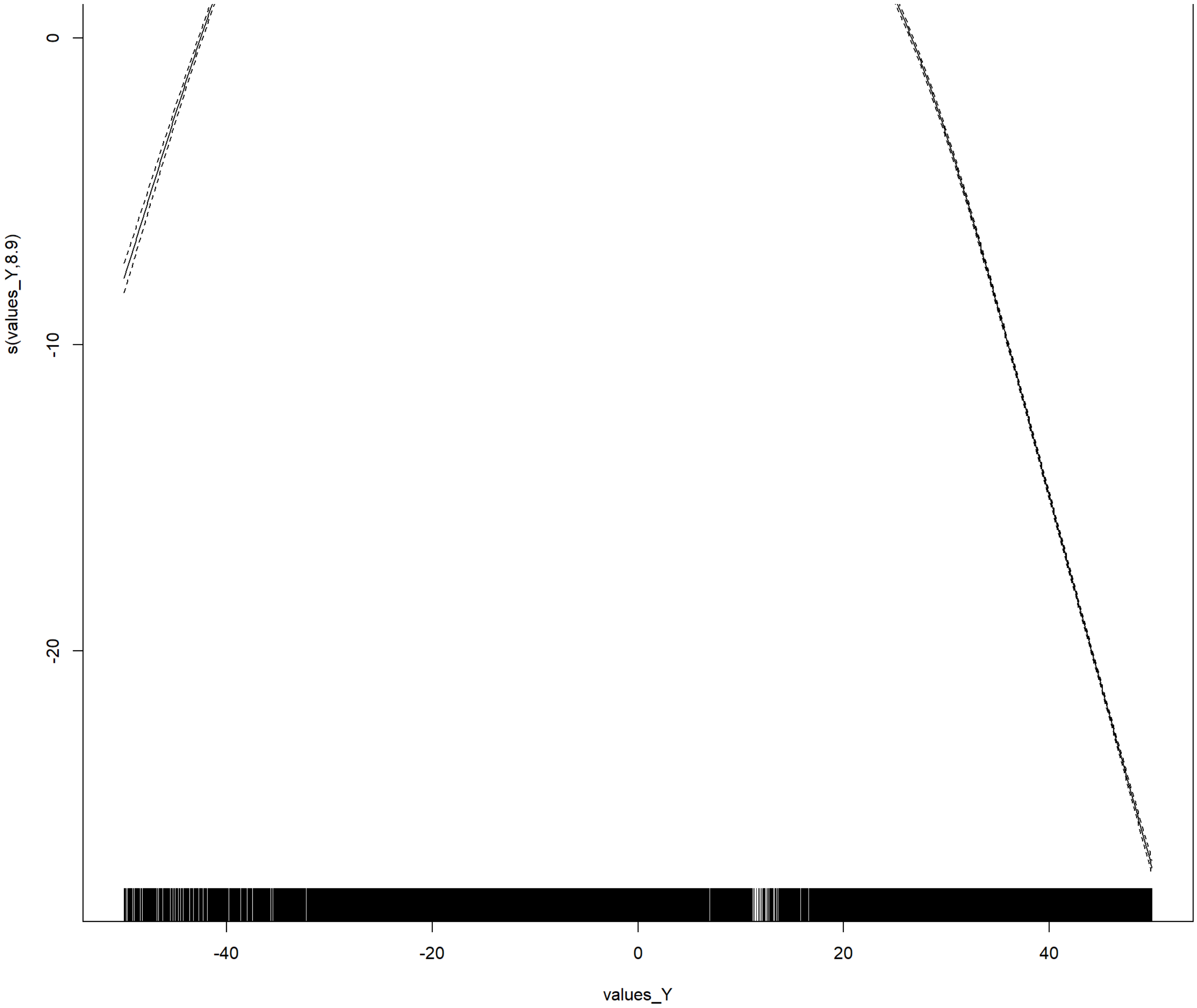

deviance residuals
theoretical quantiles
Resids vs. linear pred.
residuals
linear predictor
Histogram of residuals
Frequency
Residuals
Response vs. Fitted Values
Response
Fitted Values

```
##
## Method: GCV   Optimizer: magic
## Smoothing parameter selection converged after 12 iterations.
## The RMS GCV score gradient at convergence was 7.260532e-06 .
## The Hessian was positive definite.
## Model rank =  19 / 19
##
## Basis dimension (k) checking results. Low p-value (k-index<1) may
## indicate that k is too low, especially if edf is close to k'.
##
##               k'  edf k-index p-value
## s(values_A) 9.00 8.16    1.01    0.71
## s(values_Y) 9.00 8.90    0.94  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
plot(mod_A2,n=1000)
```


10
0
-10
-20
s(values_A,8.16)
0
1000
2000
3000
4000
5000
6000
values_A


10



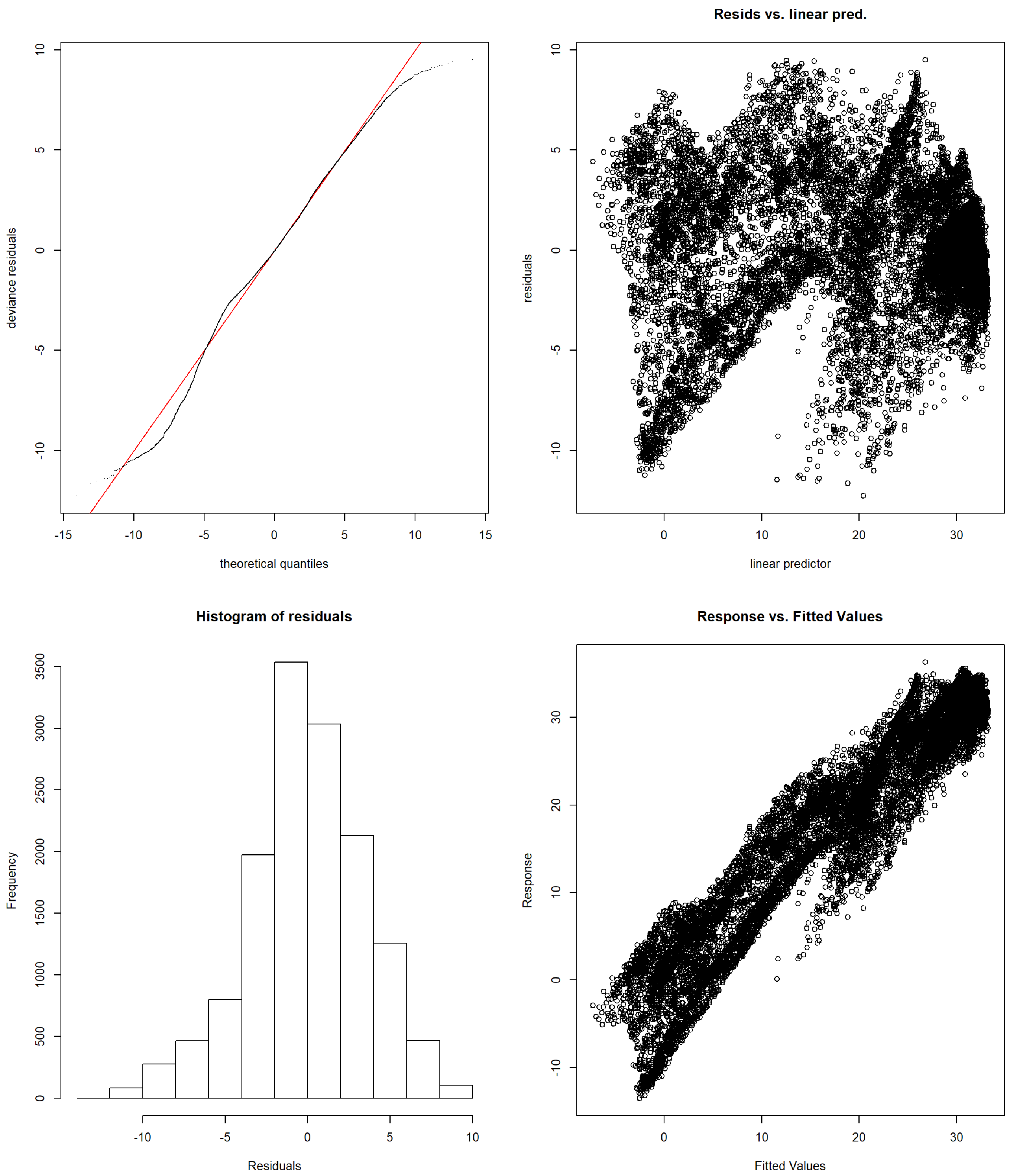

```
mod_A3 <- gam(values_T ~ values_A + s(values_Y,k=3))
summary(mod_A3)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## values_T ~ values_A + s(values_Y, k = 3)
##
## Parametric coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.235e+01  4.144e-02  539.40   <2e-16 ***
## values_A    -2.838e-03  3.052e-05  -92.98   <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(values_Y)   2      2 71767  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.917   Deviance explained = 91.7%
## GCV = 12.543  Scale est. = 12.54     n = 14138
```

```
AIC(mod_A3)
```

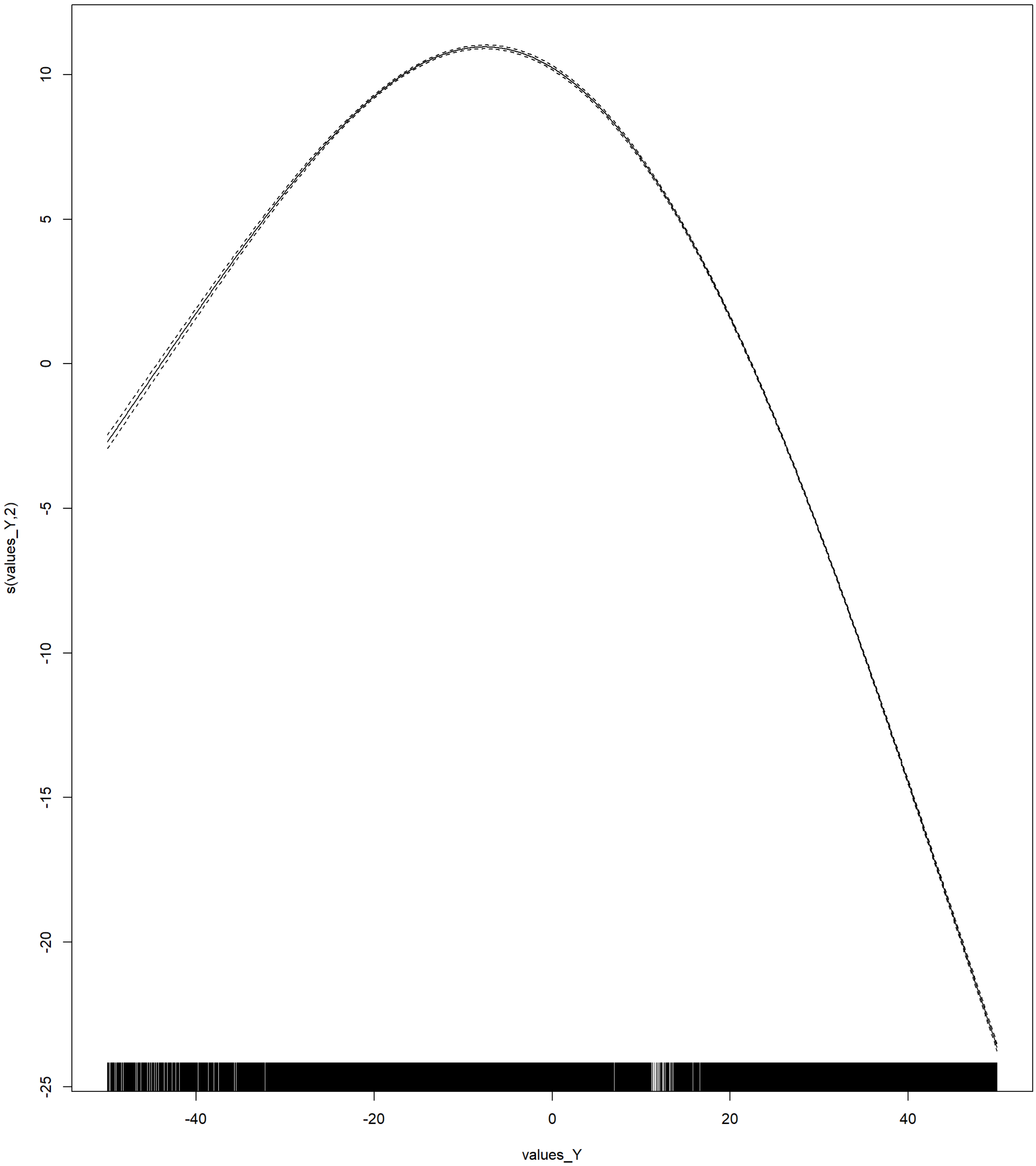
```
## [1] 75881.35
```

```
gam.check(mod_A3)
```



```
##
## Method: GCV   Optimizer: magic
## Smoothing parameter selection converged after 15 iterations.
## The RMS GCV score gradient at convergence was 2.17802e-06 .
## The Hessian was positive definite.
## Model rank =  4 / 4
##
## Basis dimension (k) checking results. Low p-value (k-index<1) may
## indicate that k is too low, especially if edf is close to k'.
##
##             k' edf k-index p-value
## s(values_Y)  2   2    0.68  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
plot(mod_A3,n=1000)
```



Spatial model

```
# spatial multivariate model
mod_B1 <- gam(values_T ~
              values_A +
                te(values_A,values_X,values_Y,k=c(2,10),d=c(1,2)) +
                s(values_Y,k=3) +
                s(values_X,values_Y,k=600),
              family=gaussian(link = "identity"),
              method="REML",
              na.action = "na.omit")
```

```
## Warning in te(values_A, values_X, values_Y, k = c(2, 10), d = c(1, 2)): one
## or more supplied k too small - reset to default
```

```
print(summary(mod_B1))
```

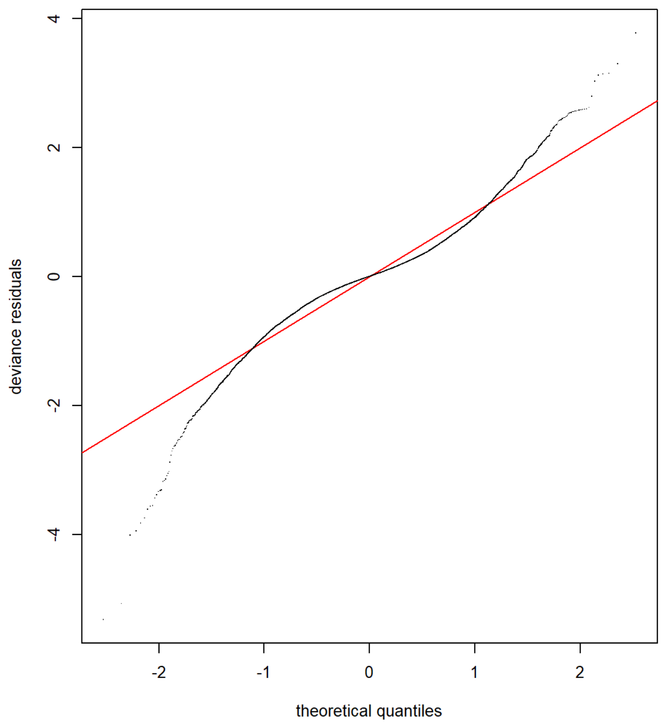
```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## values_T ~ values_A + te(values_A, values_X, values_Y, k = c(2,
##     10), d = c(1, 2)) + s(values_Y, k = 3) + s(values_X, values_Y,
##     k = 600)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.10674   11.98389   0.343    0.732
## values_A     0.01649    0.01269   1.299    0.194
##
## Approximate significance of smooth terms:
##                                   edf Ref.df       F p-value
## te(values_A,values_X,values_Y)  84.85  87.88  79.652  <2e-16 ***
## s(values_Y)                      1.00   1.00   0.228   0.633
## s(values_X,values_Y)           557.21 590.03 112.167  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.997   Deviance explained = 99.7%
## -REML =  15067  Scale est. = 0.40424   n = 14138
```

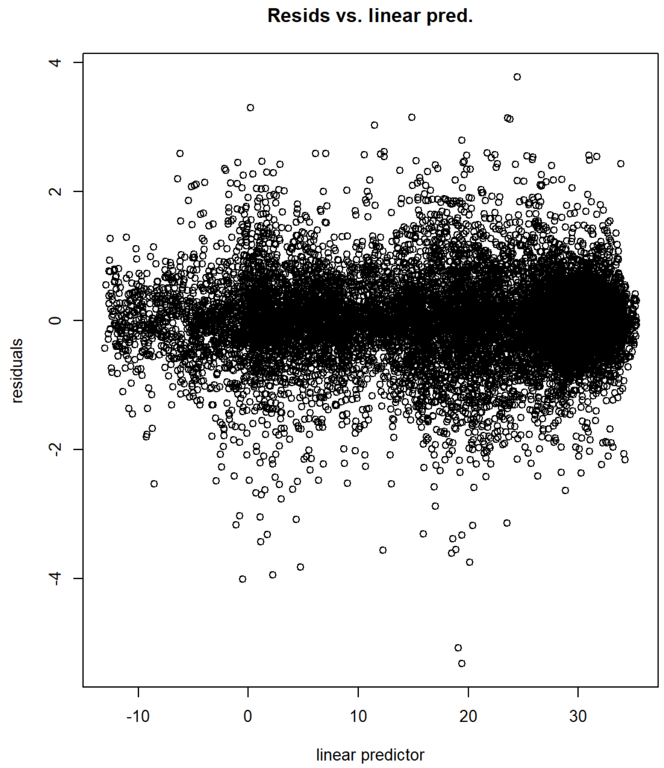
```
print(AIC(mod_B1))
```

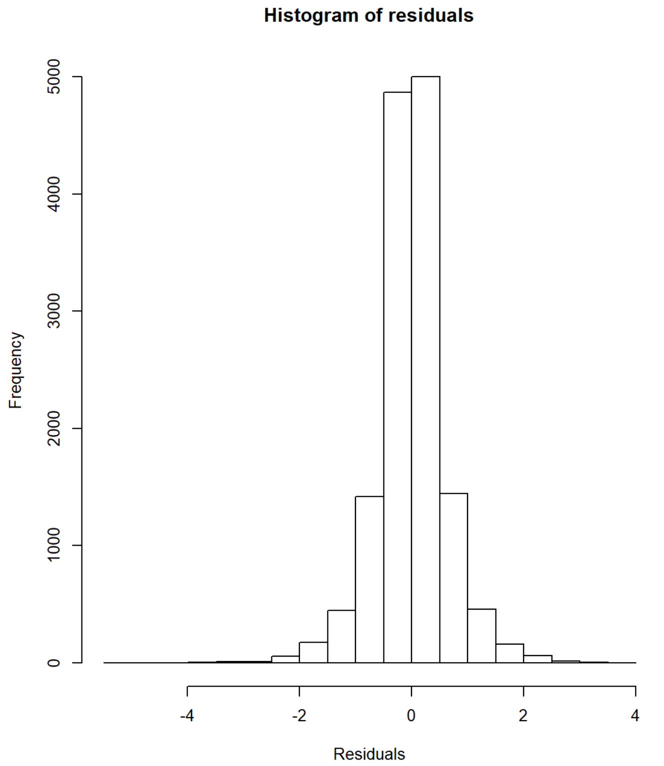
```
## [1] 27950.8
```

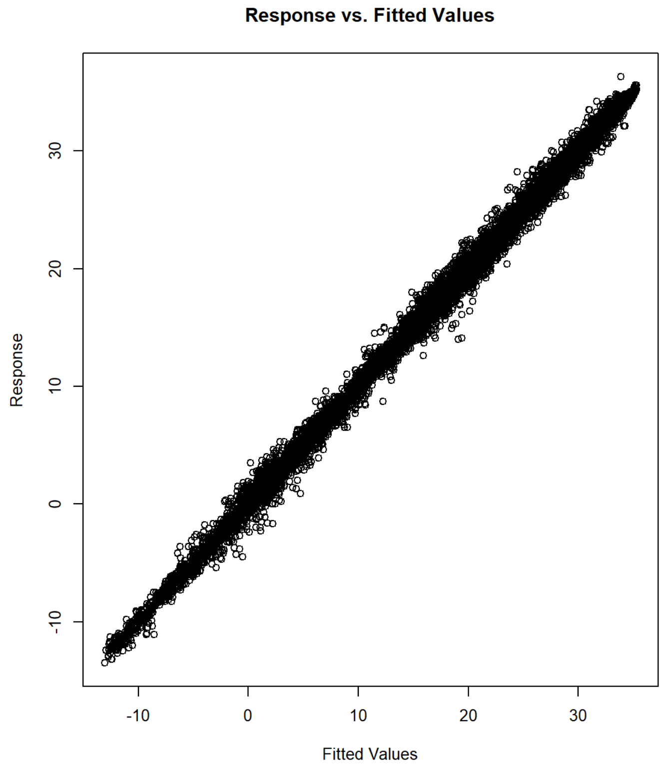
```
saveRDS(mod_B1, paste(output_path,"/","model_GAM_B1_",var_dependent,"_m",month,".rds",sep=""))
```

```
gam.check(mod_B1)
```


deviance residuals
theoretical quantiles
4
2
0
-2
-4
-2
-1
0
1
2

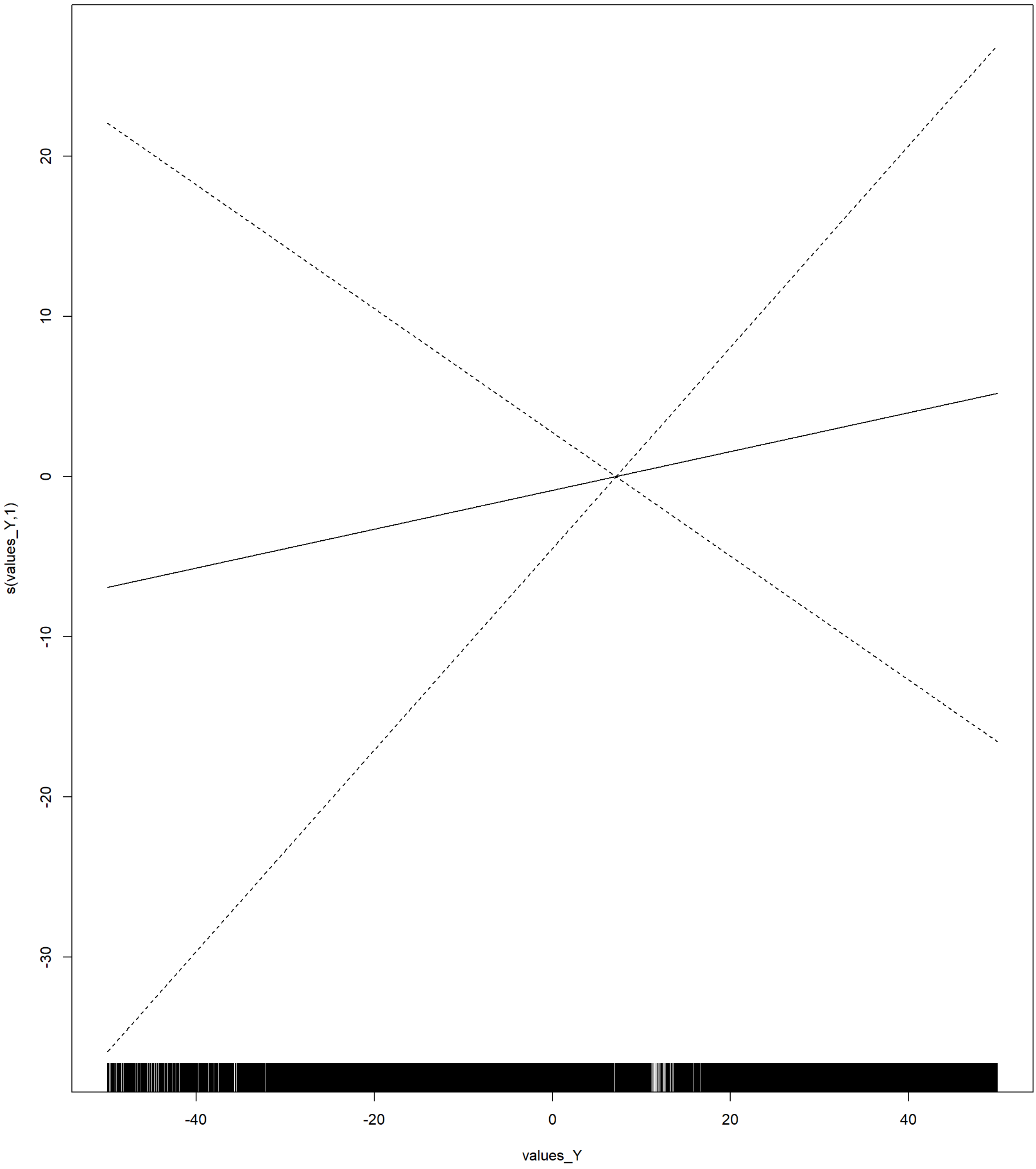

Resids vs. linear pred.
residuals
linear predictor
4
2
0
-2
-4
-10
0
10
20
30


Histogram of residuals
Frequency
Residuals
5000
4000
3000
2000
1000
0
-4
-2
0
2
4

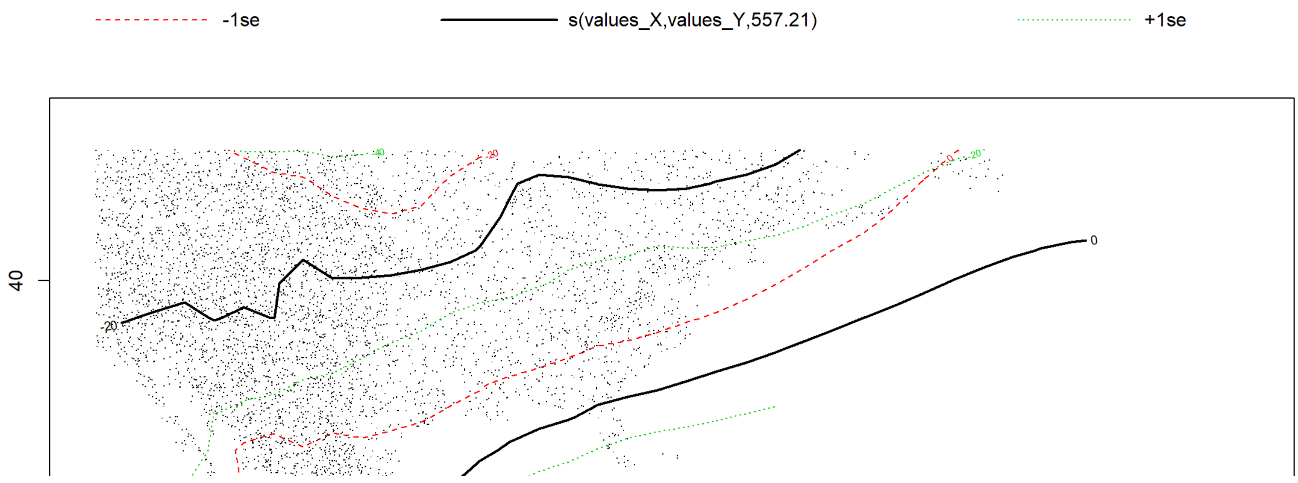

Response vs. Fitted Values
Response
Fitted Values
30
20
10
0
-10
-10
0
10
20
30

```
##
## Method: REML   Optimizer: outer newton
## full convergence after 7 iterations.
## Gradient range [-7.665426e-05,2.523688e-05]
## (score 15066.96 & scale 0.4042368).
## eigenvalue range [-6.629211e-07,7077.446].
## Model rank =  702 / 702
##
## Basis dimension (k) checking results. Low p-value (k-index<1) may
## indicate that k is too low, especially if edf is close to k'.
##
##                                   k'   edf k-index p-value
## te(values_A,values_X,values_Y) 100.0  84.8    0.72  <2e-16 ***
## s(values_Y)                      2.0   1.0    0.99    0.14
## s(values_X,values_Y)           598.0 557.2    0.76  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

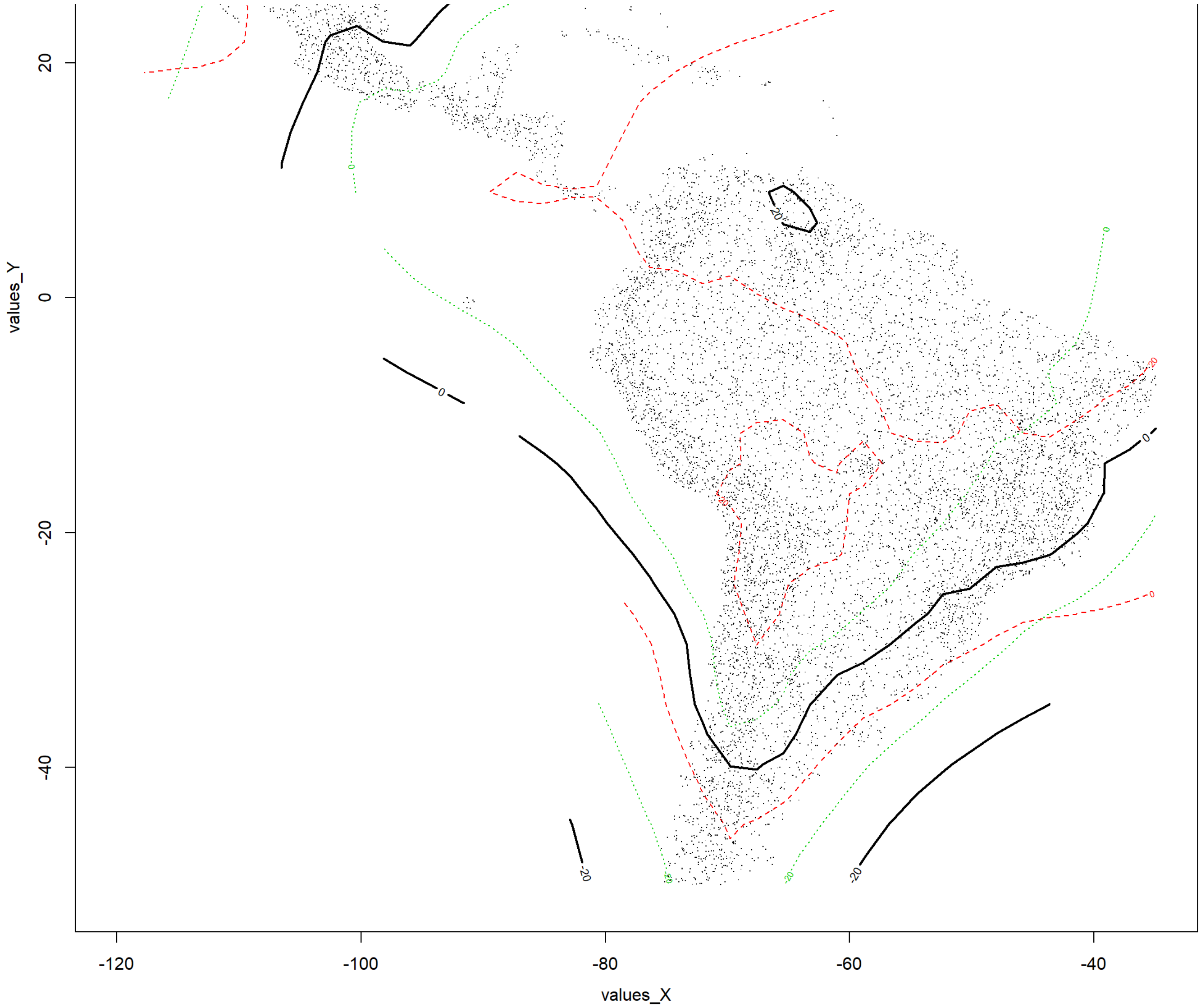
```
plot(mod_B1,n=1000)
```



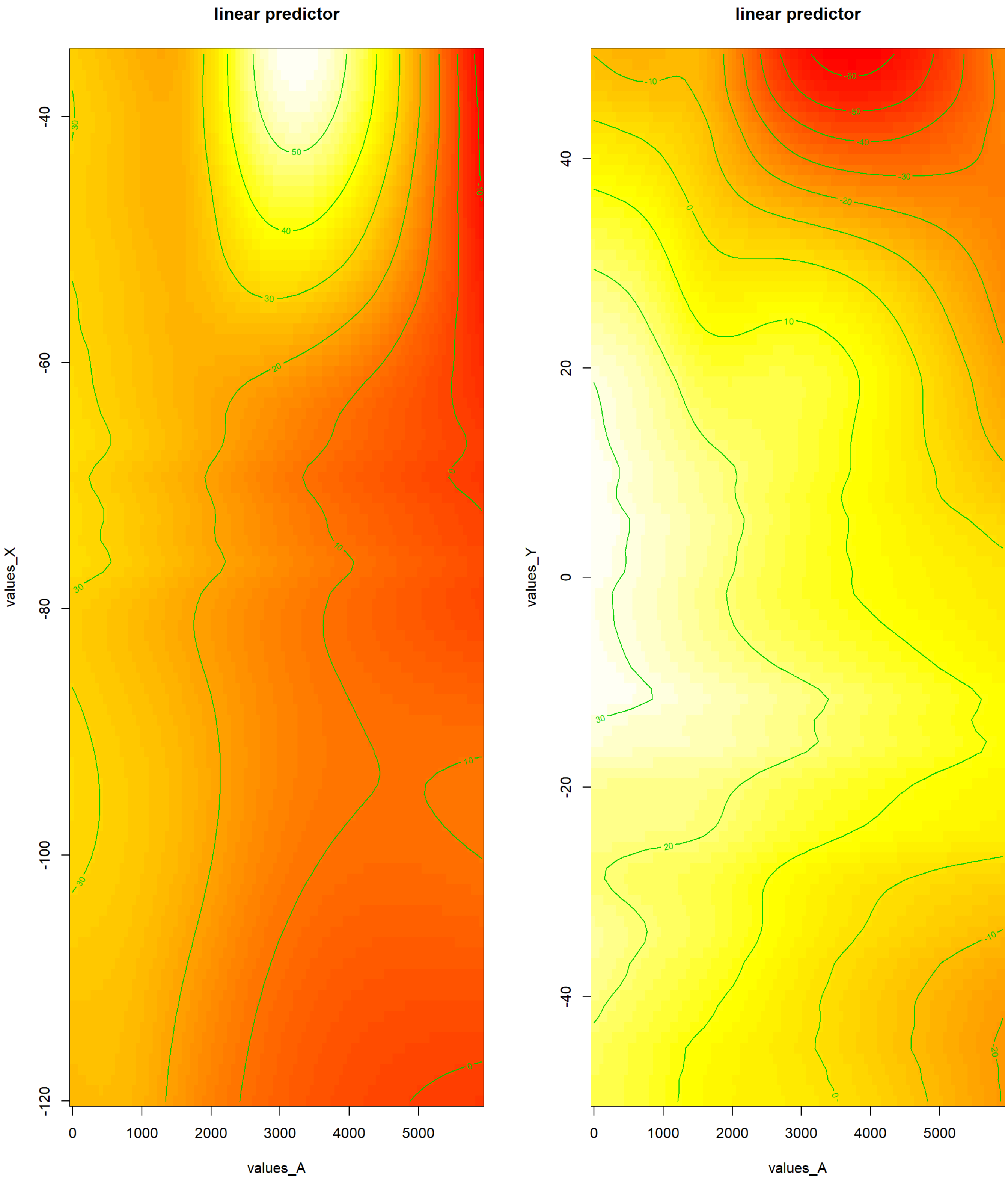
s(values_Y,1)
values_Y
-40
-20
0
20
40
-30
-20
-10
0
10
20



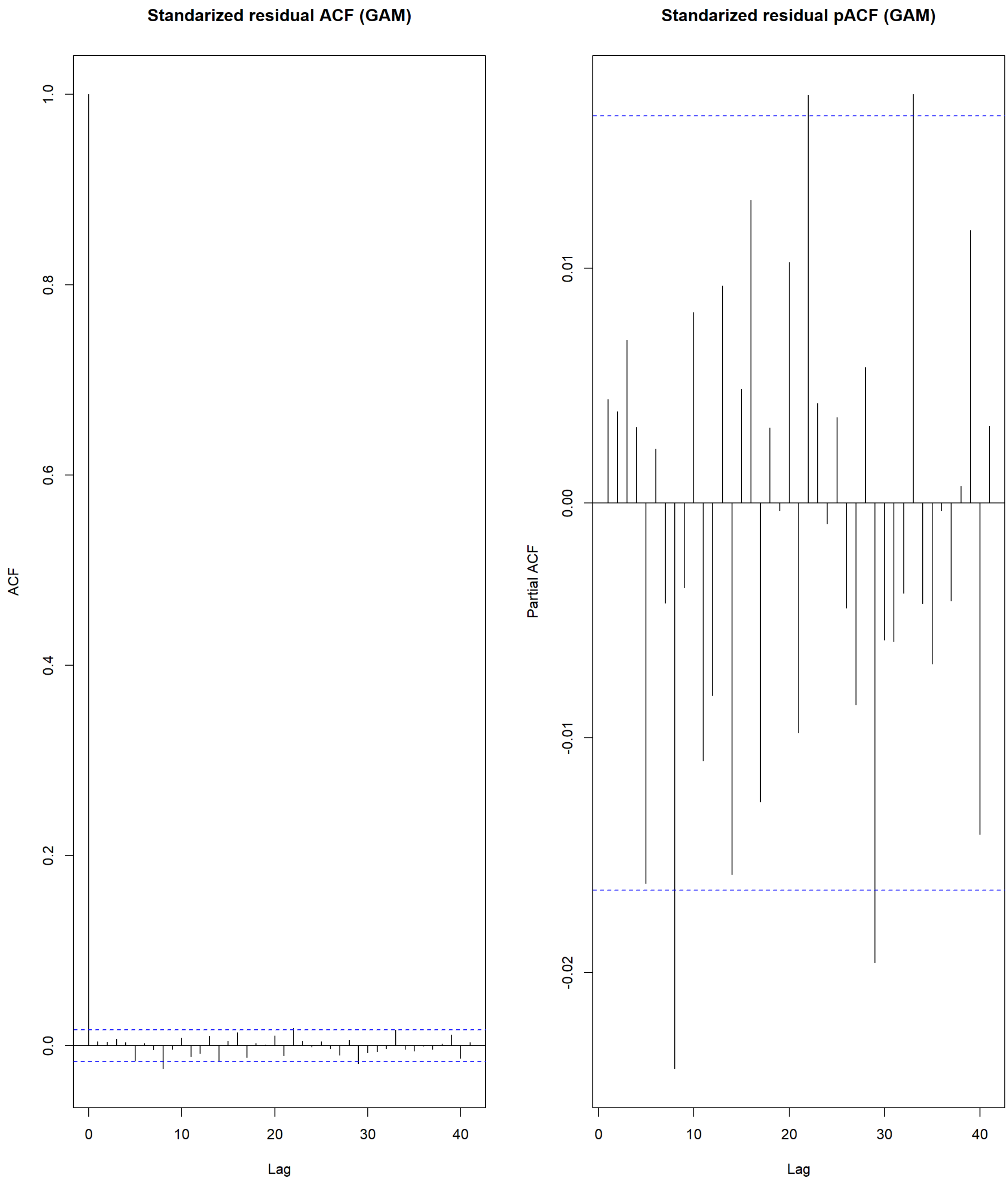
-1se
s(values_X,values_Y,557.21)
+1se
40
-20
0




```
par(mfcol = c(1, 2))
vis.gam(mod_B1,view=c("values_A","values_X"),n.grid=100,plot.type = "contour")
vis.gam(mod_B1,view=c("values_A","values_Y"),n.grid=100,plot.type = "contour")
```




```
residuals_mod_B1 <- resid(mod_B1, type = "pearson")
par(mfcol = c(1, 2))
acf(residuals_mod_B1, main="Standarized residual ACF (GAM)")
pacf(residuals_mod_B1, main="Standarized residual pACF (GAM)")
```




```
# add residuals to spatial points dataframe
residuals_mod_B1_sp <- SpatialPointsDataFrame(
                data.frame(values_X, values_Y), data.frame(resids = residuals_mod_B1),
                proj4string = CRS(mi_crs))

# plot residuals
bubble(residuals_mod_B1_sp, zcol='resids', scales = list(draw = TRUE),
      col = c("coral", "lightblue"), main = "Distribution of residuals (GAM)",
      key.space = list("bottom"))
```

Distribution of residuals (GAM)