T downscaling TMax month 01 model polynomic

Viacheslav Shalisko

16 de abril de 2018

Some important variables

```
# 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")'.
```

Read source raster data

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

Produce random points

```
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
```

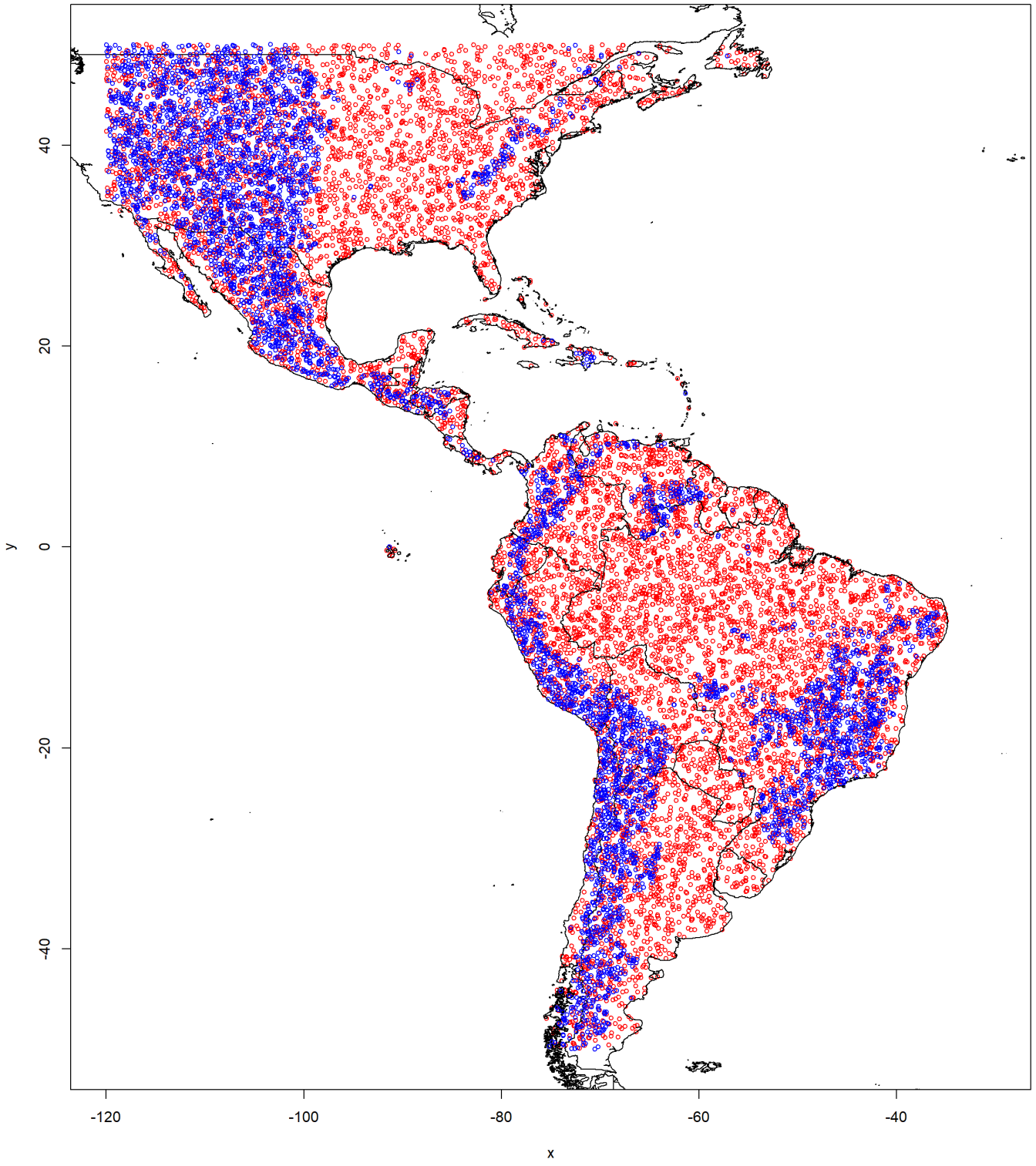
```
dim(random_points_mask2)
```

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

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

Render random points

```
# 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)
```

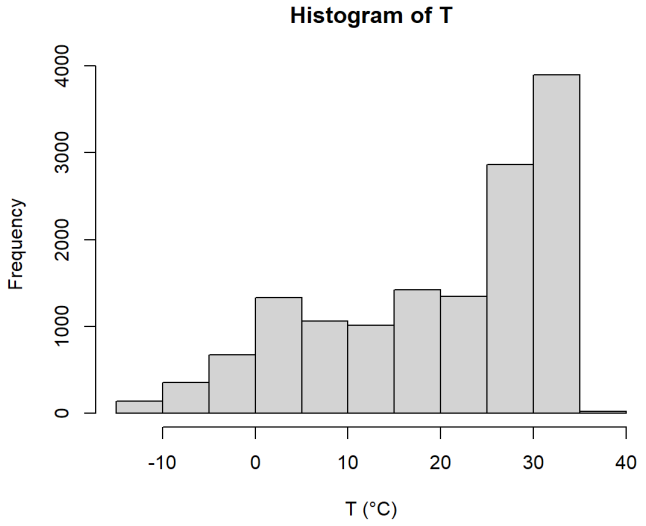


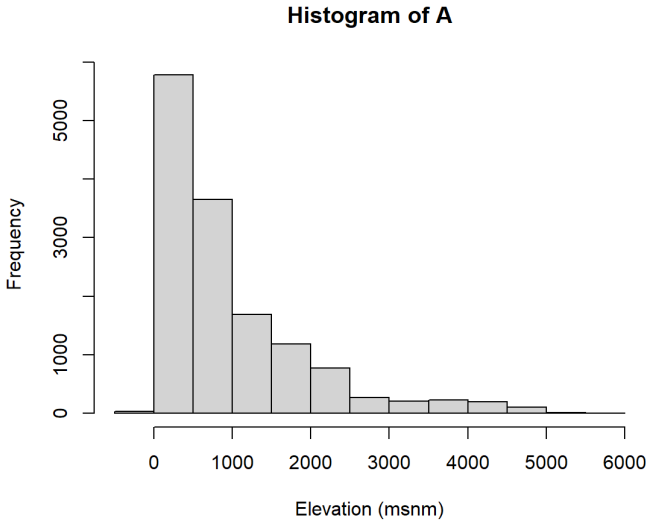
Sample raster data

```
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)

T <- as.vector(intermediate_frame[,3])
A <- as.vector(intermediate_frame[,4])
Y <- as.vector(intermediate_frame[,2])
X <- as.vector(intermediate_frame[,1])
```

Resumen of sampled values

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





Produce two-variable polynomic model

```
mod_P1 <- lm(T ~ A + Y + I(Y^2) + I(A*Y) + I(A*Y^2))
summary(mod_P1)
```

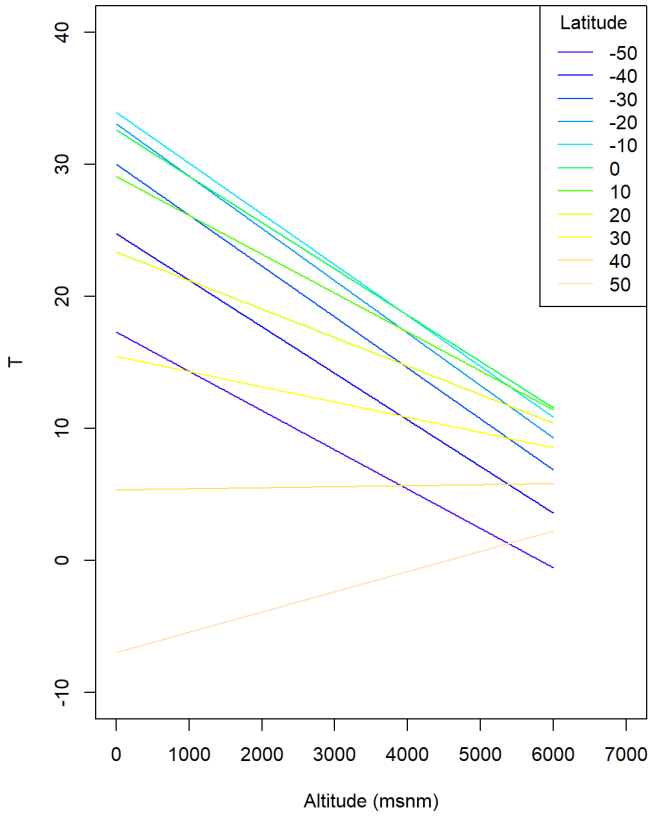
```
##
## Call:
## lm(formula = T ~ A + Y + I(Y^2) + I(A * Y) + I(A * Y^2))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.1321  -1.8741  -0.2029   1.6512  10.5283
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.260e+01  5.093e-02  640.19   <2e-16 ***
## A           -3.509e-03  4.398e-05  -79.80   <2e-16 ***
## Y           -2.428e-01  1.545e-03 -157.18   <2e-16 ***
## I(Y^2)      -1.098e-02  5.670e-05 -193.61   <2e-16 ***
## I(A * Y)     4.509e-05  1.253e-06   35.97   <2e-16 ***
## I(A * Y^2)   1.115e-06  5.399e-08   20.65   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.965 on 14132 degrees of freedom
## Multiple R-squared:  0.9417, Adjusted R-squared:  0.9417
## F-statistic: 4.566e+04 on 5 and 14132 DF,  p-value: < 2.2e-16
```

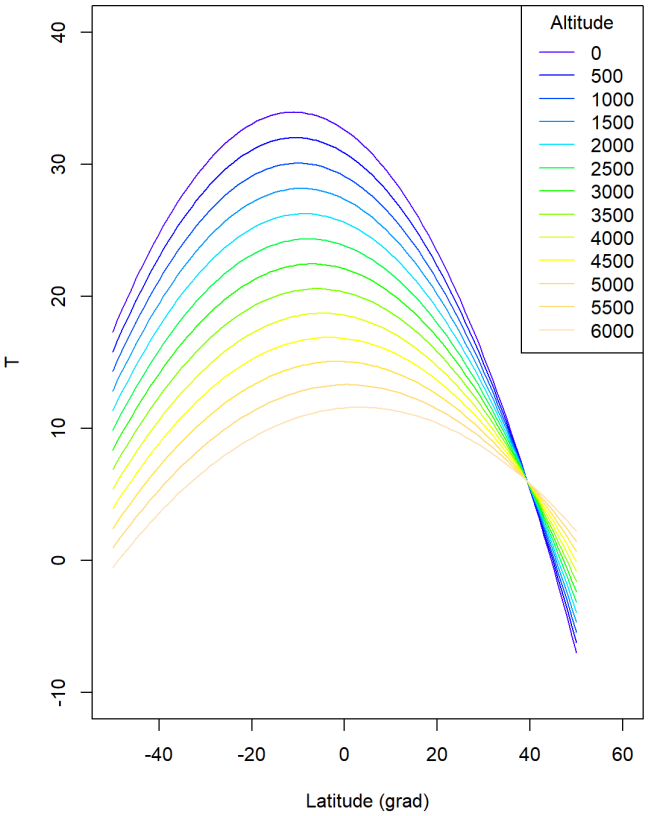
```
AIC(mod_P1)
```

```
## [1] 70862.36
```

Visualize model

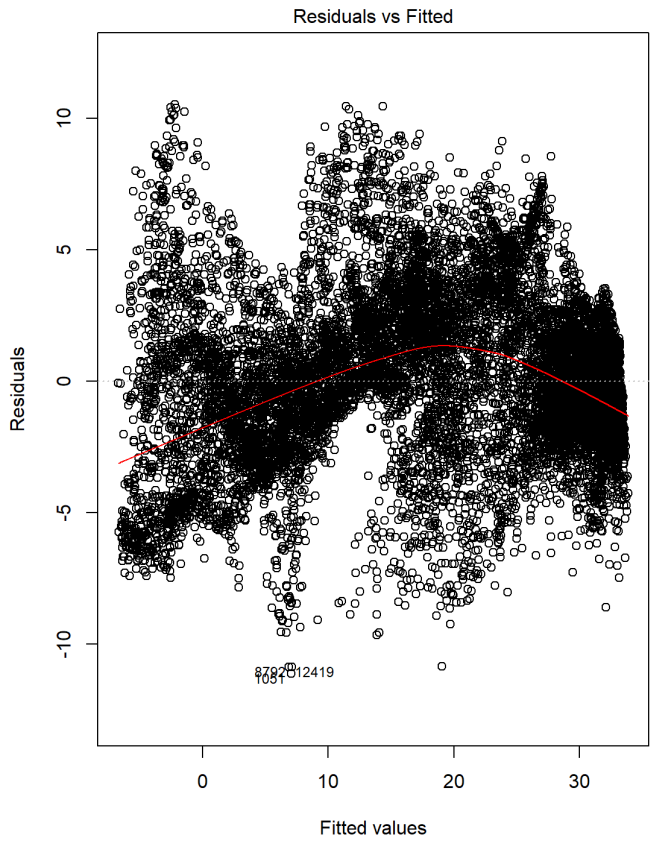
```
par(mfrow=c(1, 2))

Aseq <- seq(from=0,to=6000,by=20)
Yseq <- seq(from=-50,to=50,by=1)
AseqD <- seq(from=0,to=6000,by=500)
YseqD <- seq(from=-50,to=50,by=10)

# empty plot for T from altitude
TA <- predict(mod_P1, newdata = data.frame(A=Aseq,Y=rep(0,length(Aseq))))
plot(Aseq, TA, type = "n", ylab = "T", xlab = "Altitude (msnm)",
     xlim = c(0,7000), ylim = c(-10,40))
YseqD_col <- topo.colors(length(YseqD))

# draw lines for T from altitude by latitude class
for (i in 1:length(YseqD)) {
  Ti <- predict(mod_P1, newdata = data.frame(A=Aseq,Y=rep(YseqD[i],length(Aseq))))
  lines(Aseq, Ti, col = YseqD_col[i])
}

legend("topright", legend = YseqD, title = "Latitude",
  lty = rep(1,length(YseqD)), col = YseqD_col)


# empty plot for T from latitude
TY <- predict(mod_P1, newdata = data.frame(Y=Yseq,A=rep(0,length(Yseq))))
plot(Yseq, TY, type = "n", ylab = "T", xlab = "Latitude (grad)",
     xlim = c(-50,60), ylim = c(-10,40))
AseqD_col <- topo.colors(length(AseqD))

# draw lines for T from latitude by altitude class
for (i in 1:length(AseqD)) {
  Ti <- predict(mod_P1, newdata = data.frame(Y=Yseq,A=rep(AseqD[i],length(Yseq))))
  lines(Yseq, Ti, col = AseqD_col[i])
}

legend("topright", legend = AseqD, title = "Altitude",
  lty = rep(1,length(AseqD)), col = AseqD_col)
```

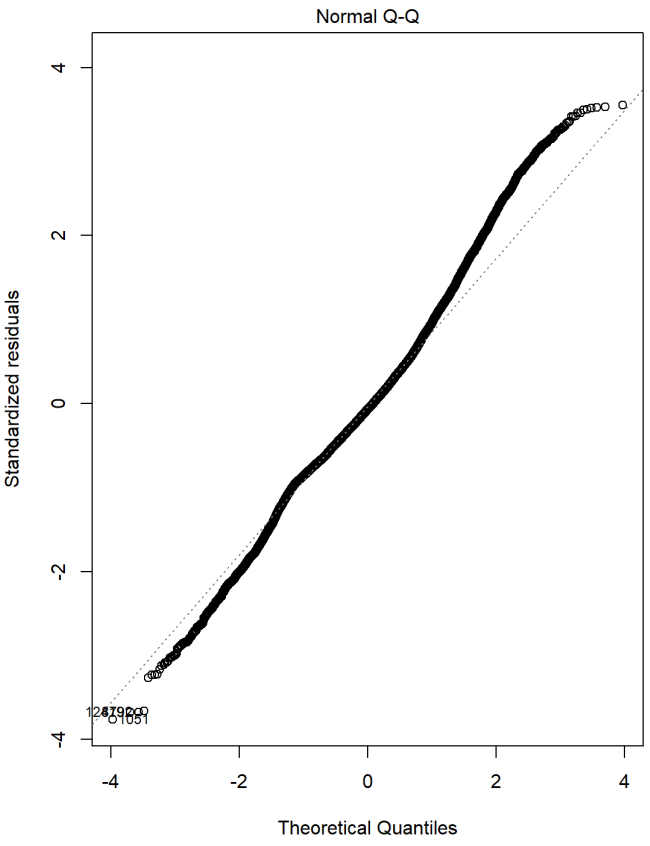


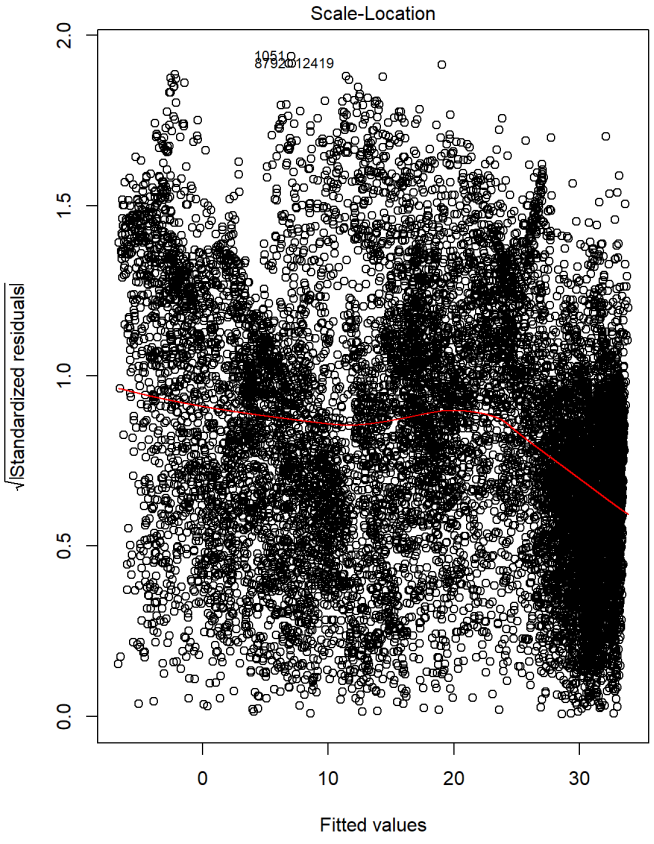


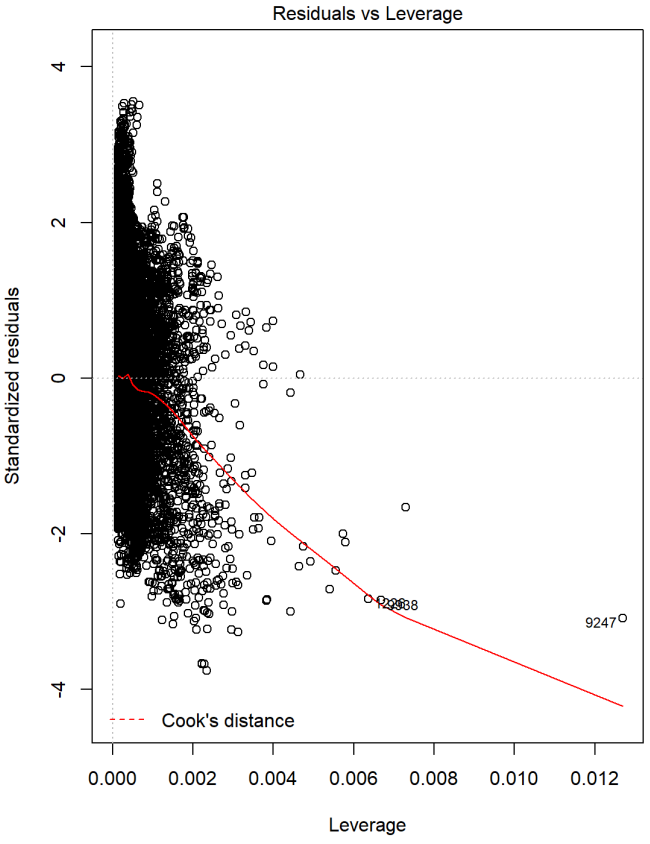
Residuals

```
par(mfcol = c(1, 2))
plot(mod_P1)
```

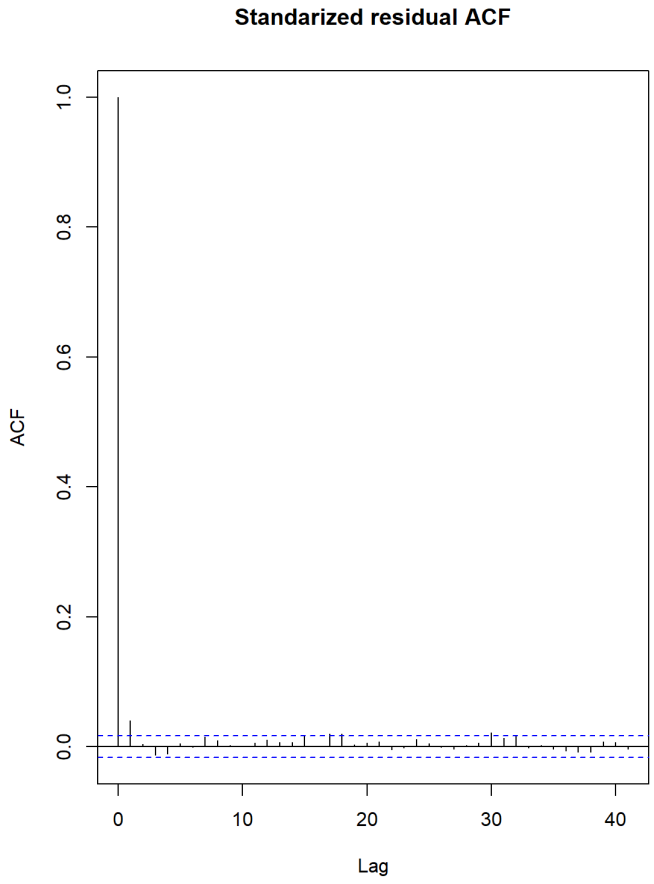


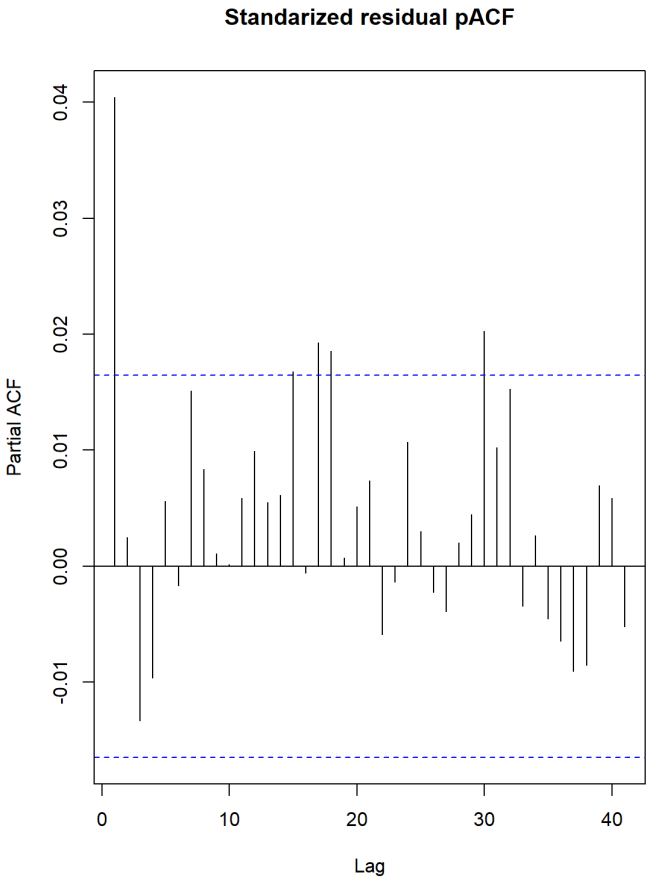







```
residuals_mod_P1 <- resid(mod_P1, type = "pearson")
acf(residuals_mod_P1, main="Standarized residual ACF")
pacf(residuals_mod_P1, main="Standarized residual pACF")
```





Spatial distribution of residuals

```
# add residuals to spatial points dataframe
residuals_mod_P1_sp <- SpatialPointsDataFrame(
                  data.frame(X, Y), data.frame(resids = residuals_mod_P1),
                  proj4string = CRS(mi_crs))

# plot residuals
bubble(residuals_mod_P1_sp, zcol='resids', scales = list(draw = TRUE),
       col = c("coral", "lightblue"), main = "Distribution of residuals",
       key.space = list("bottom"))
```

Distribution of residuals