Spatial correlation structure in linear regression

Roberto Molowny-Horas (CREAF-EMF)

March 16th-17th, 2022

We load the abundance data for the Eurasian jay, as before.

```
suppressPackageStartupMessages(library(terra))
suppressPackageStartupMessages(library(sf))
pxy <-  vect(st_read("Eurasian jay.gpkg", quiet = T)) # We will use a SpatVector below.
```

Then, we fetch climatic variables and DEM data from the worldclim.org web site.

```
w <- rast(raster::getData("worldclim", var="bio", res=10))    # Bio variables.
dem <- rast(raster::getData("worldclim", var="alt", res=10))  # DEM in meters.
slope <- terrain(dem,"slope")                                 # Slope in degrees
t_mean <- subset(w,"bio1")/10                                 # Annual temp.
p_total <- subset(w,"bio12")                                  # Annual precip.
names(t_mean) <- "t_mean"                                     # Naming.
names(p_total) <- "p_total"
names(dem) <- "dem"
names(slope) <- "slope"
```

Next, we crop those rasters for ease of use.

```
t_mean <- crop(t_mean,ext(pxy) + 1)             # Cropping rasters.
p_total <- crop(p_total,ext(pxy) + 1)
dem <- crop(dem,ext(pxy) + 1)
slope <- crop(slope,ext(pxy) + 1)
```

We will extract values of those rasters at the locations of the point pattern.

```
pxy$t_mean <- extract(t_mean, pxy, method="simple")$t_mean    # Add variables
pxy$p_total <- extract(p_total, pxy, method="simple")$p_total
pxy$dem <- extract(dem, pxy, method="simple")$dem
pxy$slope <- extract(slope, pxy, method="simple")$slope
pxy <- na.omit(pxy, field="")
```

Finally, we fit a 'gls' model from the 'nlme' R package. It extends the linear model 'lm' to the case when errors are correlated (spatial or temporal). Only fixed factors are modeled. First, we make a data.frame as input for the 'gls' function and sample it down to 100 points to have quick results.

```
suppressPackageStartupMessages(library(nlme))
df <- data.frame(pxy,crds(pxy))
df <- df[sample(nrow(df),200),]
df <- df[df$dem > 0,]                           # To eliminate infinite weights below.
```

Then, a model with the four variables is fitted without spatial structure to investigate the best 'varPower' function to account for heteroscedasticity (that we know there is).

Notice that, for the sake of the example, we are assuming normality of the residuals of a response variable that is rather Poisson-like (i.e. counts).

```
suppressPackageStartupMessages(library(nlme))
f <- formula(individualCount~t_mean+I(t_mean^2)+p_total+I(p_total^2)+dem+I(dem^2)+slope+I(slope^2))
re_weights <- list()
re_weights[[1]] <- gls(f,data=df)
re_weights[[2]] <- gls(f,data=df,weights=varPower(form=~dem))
re_weights[[3]] <- gls(f,data=df,weights=varPower(form=~slope))
re_comp <- setNames(sapply(re_weights,function(x) sqrt(mean(resid(x)^2,na.rm=T))),
                    c("None","DEM","Slope"))
print(re_comp)
```

```
##     None      DEM    Slope
## 1.110075 1.110577 1.110309
```

We try a different approach by splitting the data into 70%-30% training-testing subsets.

```
fperm <- function() {
  nr <- nrow(df)
  i <- sample(nr,round(nr*.7))
  dff <- df[i,]
  y <- list()
  y[[1]] <- gls(f,data=dff)
  y[[2]] <- gls(f,data=dff,weights=varPower(form=~dem))
  y[[3]] <- gls(f,data=dff,weights=varPower(form=~slope))
  dff <- df[-i,]
  sapply(y,function(x) sqrt(mean((predict(x,newdata=dff)-dff$individualCount)^2,na.rm=T)))
}
re_simu <- setNames(apply(replicate(100,fperm()),1,mean),c("None","DEM","Slope"))
print(re_simu)
```

```
##     None      DEM    Slope
## 1.207743 1.205781 1.206072
```

Other functional forms for the variance function structure may also be interesting to try out (see ?varClasses).

Next, we try fitting the dataset with different spatial correlation structures to see which one explains best the datset.

```
w <- varPower(form=~dem)
re_corr <- list()
re_corr[[1]] <- gls(f,data=df,correlation=corExp(form=~x+y),weights=w)
re_corr[[2]] <- gls(f,data=df,correlation=corExp(form=~x+y,nugget=T),weights=w)
re_corr[[3]] <- gls(f,data=df,correlation=corSpher(form=~x+y),weights=w)
re_corr[[4]] <- gls(f,data=df,correlation=corSpher(form=~x+y,nugget=T),weights=w)
re_corr[[5]] <- gls(f,data=df,correlation=corGaus(form=~x+y),weights=w)
re_corr[[6]] <- gls(f,data=df,correlation=corGaus(form=~x+y,nugget=T),weights=w)
re_corr[[7]] <- gls(f,data=df,correlation=corRatio(form=~x+y),weights=w)
re_corr[[8]] <- gls(f,data=df,correlation=corRatio(form=~x+y,nugget=T),weights=w)
print(data.frame(SD=sapply(re_corr,function(x) sd(resid(x))),AIC=sapply(re_corr,AIC)))
```

```
##         SD     AIC
## 1 1.113352 720.762
## 2 1.113352 722.762
## 3 1.113352 720.762
## 4 1.113352 722.762
## 5 1.113352 720.762
## 6 1.113352 722.762
## 7 1.113352 720.762
## 8 1.113352 722.762
```

Minimal differences between models. We choose a model with one of the lowest AIC.

```
print(summary(re_corr[[3]]))
```

```
## Generalized least squares fit by REML
##   Model: f
##   Data: df
##       AIC      BIC   logLik
##   720.762 759.5993 -348.381
##
## Correlation Structure: Spherical spatial correlation
##  Formula: ~x + y
##  Parameter estimate(s):
##       range
## 0.003878936
## Variance function:
##  Structure: Power of variance covariate
##  Formula: ~dem
##  Parameter estimates:
##       power
## -0.05370577
##
## Coefficients:
##                   Value Std.Error    t-value p-value
## (Intercept)   0.4120786 1.5555259  0.2649127  0.7914
## t_mean        0.4502108 0.2411058  1.8672747  0.0634
## I(t_mean^2)  -0.0193094 0.0105172 -1.8359758  0.0679
## p_total      -0.0034094 0.0026223 -1.3001888  0.1951
## I(p_total^2)  0.0000021 0.0000014  1.5072674  0.1334
## dem           0.0006439 0.0007556  0.8522194  0.3952
## I(dem^2)     -0.0000001 0.0000005 -0.2503970  0.8026
## slope        -0.3748141 0.5642350 -0.6642873  0.5073
```

```
## I(slope^2)    0.0263597 0.1864203  0.1413993  0.8877
##
##  Correlation:
##              (Intr) t_mean I(t_^2) p_totl I(p_^2) dem    I(d^2) slope
## t_mean       -0.671
## I(t_mean^2)   0.623 -0.988
## p_total      -0.526 -0.261  0.290
## I(p_total^2)  0.490  0.282 -0.310  -0.989
## dem           0.300 -0.402  0.400   0.005  0.008
## I(dem^2)     -0.508  0.538 -0.511   0.095 -0.107  -0.916
## slope        -0.114  0.216 -0.255  -0.063  0.017  -0.534  0.380
## I(slope^2)    0.141 -0.188  0.220   0.007  0.031   0.381 -0.286 -0.892
##
## Standardized residuals:
##        Min         Q1        Med         Q3        Max
## -1.1767363 -0.5937673 -0.4430611  0.3627444  5.5784705
##
## Residual standard error: 1.451358
## Degrees of freedom: 197 total; 188 residual
```

Finally, we can use function 'dredge' from R package 'MuMIn' to try to select the best model.

```
library(MuMIn)
re_dred <- dredge(re_corr[[3]])
```

```
## Warning in dredge(re_corr[[3]]): comparing models fitted by REML
```

```
## Fixed term is "(Intercept)"
```

```
print(head(re_dred))
```

```
## Global model call: gls(model = f, data = df, correlation = corSpher(form = ~x +
##     y), weights = w)
## ---
## Model selection table
##    (Int)      slp    slp^2    t_men df   logLik  AICc delta weight
## 1  1.676                             4 -304.358 616.9  0.00  0.753
## 17 1.692 -0.05511                    5 -305.005 620.3  3.40  0.138
## 33 1.686          -0.04255           5 -305.818 622.0  5.03  0.061
## 49 1.655  0.14720 -0.09366           6 -305.740 623.9  7.00  0.023
## 65 1.651                    0.002409  5 -306.857 624.0  7.10  0.022
## 81 1.665 -0.05596           0.002708  6 -307.499 627.4 10.52  0.004
## Models ranked by AICc(x)
```