Mixed Effects Models Using Sloth Data

Joe Brehm, Mariel Boldis, Steven Bristow, and Janyne Little

For those wishing to follow along with the R-based demo in class, click here for the companion R script for this lecture.

Also, click here for the PPT slides!

Load the data!

Sloth data provided by Neam KD, Lacher Jr. TE (2018) Multi-scale effects of habitat structure and landscape context on a vertebrate with limited dispersal ability (the brown-throated sloth, *Bradypus variegatus*). Biotropica 50(4): 684-693. https://doi.org/10.1111/btp.12540

Dryad data package: Neam KD, Lacher Jr. TE (2018) Data from: Multi-scale effects of habitat structure and landscape context on a vertebrate with limited dispersal ability (the brown-throated sloth, Bradypus variegatus). Dryad Digital Repository. https://doi.org/10.5061/dryad.n8tt5

Download Zuur et al. data here
Download sloth data here.

Properties of mixed models

Remember these Assumptions.

The assumptions of generalised linear mixed models are a combination of the assumptions of GLMs and mixed models.

1. The observed y are independent, conditional on some predictors x
2. The response y come from a known distribution from the exponential family, with a known mean variance relationship
3. There is a uniform relationship between some known function of the mean of y and the predictors x and random effects z
4. Random effects z are independent of y
5. Random effects z are normally distributed

Install packages

```
library(Matrix)
library(lme4)
library(MASS)
library(arm)
library(sjstats)
library(ResourceSelection)
```

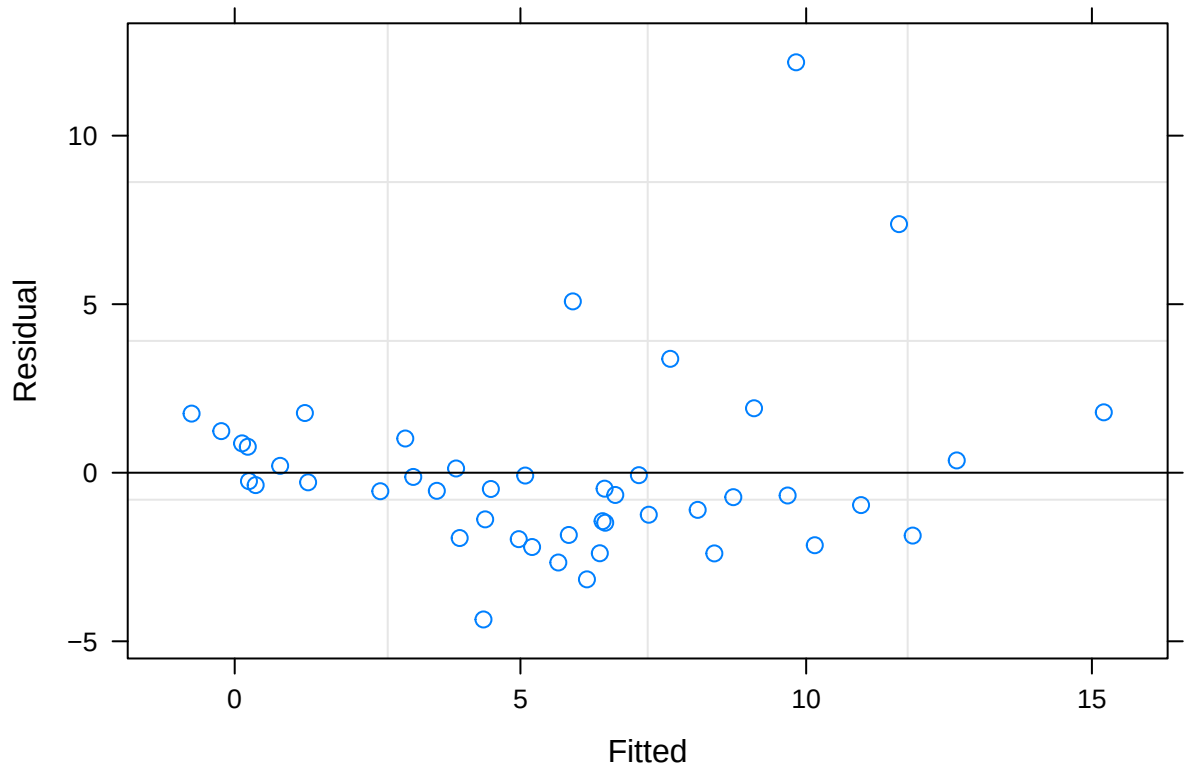
Zuur et al. Data

First, let's start with some nice neat data to show you what mixed effect models can do for you.

```
# load in
df.rikz <- read.table("RIKZ.txt",header=T)

# test for correlation
abs(cor(df.rikz[,c(2,5)]))
```

```
##           Richness     Beach
## Richness 1.0000000 0.4435666
## Beach    0.4435666 1.0000000
```

```
# define the model
lme.rikz <- lmer(Richness ~ NAP + (1 | Beach), data = df.rikz)

# residual vs fitted plot
plot(lme.rikz, xlab = "Fitted", ylab = "Residual")
```




```
summary(lme.rikz)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: Richness ~ NAP + (1 | Beach)
##    Data: df.rikz
##
## REML criterion at convergence: 239.5
##
```

```
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -1.4227 -0.4848 -0.1576  0.2519  3.9794
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  Beach    (Intercept) 8.668    2.944
##  Residual             9.362    3.060
## Number of obs: 45, groups:  Beach, 9
##
## Fixed effects:
##             Estimate Std. Error t value
## (Intercept)   6.5819     1.0958   6.007
## NAP          -2.5684     0.4947  -5.192
##
## Correlation of Fixed Effects:
##     (Intr)
## NAP -0.157
```

Sloth Data

Now... let's get into a real ecological dataset.

```
rm(list=ls())

df.sloth <- read.csv("sloth_point5only_withPatchGroup.csv", header = TRUE, sep = ",")
str(df.sloth) # 25 variables with 25 observations
```

```
## 'data.frame':    25 obs. of  25 variables:
##  $ SITE_ID   : Factor w/ 25 levels "CH01_A","CH01_P",..: 2 3 5 7 10 12 13 14 15 16 ...
##  $ LONGITUDE : num  10.4 10.4 10.4 10.4 10.4 ...
##  $ LATITUDE  : num  -84.6 -84.6 -84.6 -84.6 -84.6 ...
##  $ LAND_USE  : Factor w/ 2 levels "mixed_use","plantation": 1 1 2 2 1 1 1 1 1 1 ...
##  $ patch_grou: Factor w/ 4 levels "A","B","C","D": 4 4 4 3 2 2 4 2 2 2 ...
##  $ OCCURRENCE: int  1 1 1 1 1 1 1 1 1 1 ...
##  $ PATCH_AREA: num  0.111 0.18 0.499 0.49 0.152 ...
##  $ PATCH_GYRA: num  20.5 16.6 26.6 26.4 17.4 ...
##  $ PATCH_SHAP: num  1.75 1.3 1.13 1.15 1.99 ...
##  $ PLAND_SF  : num  13.946 16.841 0.001 0.001 44.662 ...
##  $ DIST_SEC_F: num  29.46 36.05 140.53 69.32 4.71 ...
##  $ DIST_RIP_F: num  0.1 6.05 196.81 272.45 127.11 ...
##  $ DIST_ROAD : num  8.87 183.36 71.49 33.77 28.59 ...
##  $ PD        : num  1805 1404 201 602 802 ...
##  $ LPI       : num  27.9 36.1 100 98.2 40.5 ...
##  $ EDGE      : num  691.2 451.2 0 94.2 268.8 ...
##  $ AREA_WM   : num  0.0939 0.1133 0.4987 0.4807 0.1598 ...
##  $ GYRATE_WM : num  17.7 15.3 26.6 26 16.9 ...
##  $ SHAPE_WM  : num  1.92 1.6 1.13 1.16 1.55 ...
##  $ CWED      : num  970 614 0 189 345 ...
##  $ ECON_AM   : num  45.6 44.1 0 21.2 39.3 ...
##  $ CONTAG    : num  57 54.6 100 96.8 58.2 ...
##  $ PR        : int  5 4 1 2 3 3 5 4 2 4 ...
##  $ SIDI      : num  0.7009 0.6432 0 0.0152 0.6459 ...
##  $ SIEI      : num  0.8761 0.8576 0 0.0304 0.9688 ...
```

Before we can use this data, we need to subset the data to include numeric variables only. This subsetted data will be used to run a for loop with a Shapiro-Wilk test to test for normality.

```
# drop metadata, response, random effect, categorical data
df.sloth2 <- subset.data.frame(df.sloth, select = -c(SITE_ID, LONGITUDE, LATITUDE, LAND_USE, patch_grou
df.sloth2$PR <- as.numeric(df.sloth2$PR)
str(df.sloth2)
```

Testing Data for Normality

Shapiro-Wilk test to test for normality. Variables that don't pass will need to be transformed before we can use them.

```
IsItNormal <- function(df, save = F) { ### should be able to accept any dataframe of arbitrary size & na
  slothtest <- list()
  for(i in seq(dim(df)[2])) {
    slothtest <- append(slothtest, shapiro.test(df[[i]]))
  }
  mx.slothtest <- matrix(slothtest, byrow = T, ncol = 4)
  df.slothtest <- data.frame(var = as.character(colnames(df)),
                             W = as.numeric(mx.slothtest[,1]),
                             p = as.numeric(mx.slothtest[,2]),
                             row.names = NULL)
  if(save == T) {write.table(df.slothtest, file = paste0("IsItNormal", round(as.numeric(Sys.time())), "
  return(df.slothtest)
}

IsItNormal(df.sloth2)
```

We will also transform the data three ways and run the normality test. This method allows us to see if non-normal data prior to transformtion can be normalized through transforming it.

```
df.sloth.log <- as.data.frame(apply(df.sloth2, 2, log))
df.sloth.sqrt <- as.data.frame(apply(df.sloth2, 2, sqrt))
df.sloth.squared <- as.data.frame(apply(df.sloth2, 2, function(x) x^2))

df.slothtest.transformed <- cbind(IsItNormal(df.sloth2),
                                  IsItNormal(df.sloth.log)[,2:3],
                                  IsItNormal(df.sloth.sqrt)[,2:3],
                                  IsItNormal(df.sloth.squared)[,2:3])

colnames(df.slothtest.transformed)[4:5] <- paste0("log.", c("W", "p"))
colnames(df.slothtest.transformed)[6:7] <- paste0("sqrt.", c("W", "p"))
colnames(df.slothtest.transformed)[8:9] <- paste0("squared.", c("W", "p"))

# Table defining the normality at different transformations saved to your drive

write.table(df.slothtest.transformed, file = paste0("AreTransformedSlothsNormal_", round(as.numeric(Sys
```

If you open the table we wrote above, you will find that 5 variables are never normal, 5 variables that are normal if log transformed, 7 variables that are normal without transformation, 1 variable that's normal if square rooted, and 1 variable that is normal if squared.

Below we have transformed the necessary variables and saved them as their own object to be combined in the subsequent code.

```r
# data subsetted removing "never normal columms" based on normality tests
df.sloth3 <- subset.data.frame(df.sloth2, select = -c(CONTAG, DIST_SEC_F, PATCH_SHAP, PLAND_SF, SIEI))

# Log transform "PATCH_AREA", "DIST_ROAD","PD", "LPI","AREA_WM"
df.sloth4 <- subset.data.frame(df.sloth2, select = c(PATCH_AREA, DIST_ROAD,PD, LPI,AREA_WM))
df.sloth4.log <- log(df.sloth4)
colnames(df.sloth4.log) <- paste0("log", colnames(df.sloth4.log))

# data subset with only normal non-transformed variables
df.sloth.normal <- subset.data.frame(df.sloth3, select = c(DIST_RIP_F,ECON_AM,EDGE,GYRATE_WM,PATCH_GYRA

# sqrt CWED
df.sloth3.sqrt <- sqrt(df.sloth3["CWED"])
colnames(df.sloth3.sqrt) <- paste0("sqrt", colnames(df.sloth3.sqrt))

# squared SIDI
sq <- function(x){
  x^2
}

df.sloth3sq <- sq(df.sloth3["SIDI"])
colnames(df.sloth3sq) <- paste0("square", colnames(df.sloth3sq))
```

Combine all the transformed and normal data into one along with the categorical variables and metadata we removed originally in the beginning. The "allsloth" data will be the dataset we work from.

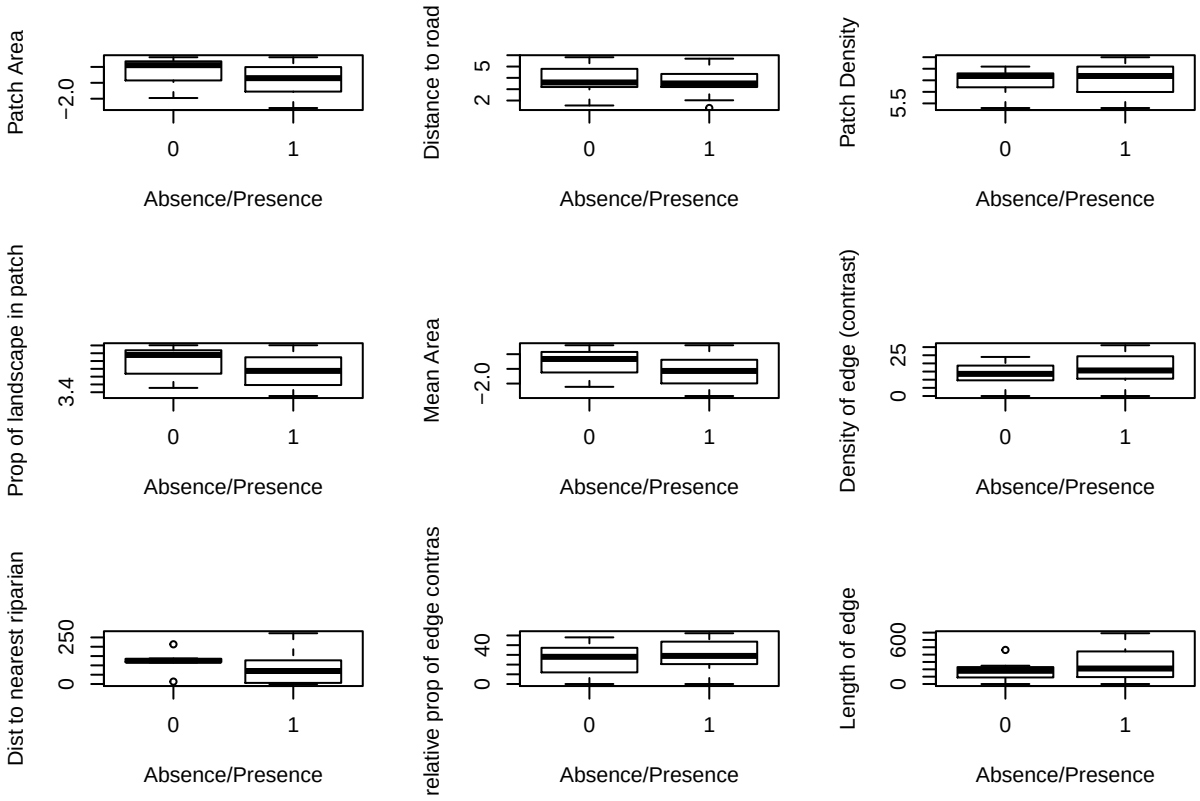
```r
# Working Dataset
allsloth <- cbind(df.sloth[,1:6], df.sloth4.log, df.sloth3.sqrt, df.sloth.normal)

str(allsloth)
```

```
## 'data.frame':    25 obs. of  19 variables:
##  $ SITE_ID      : Factor w/ 25 levels "CH01_A","CH01_P",..: 2 3 5 7 10 12 13 14 15 16 ...
##  $ LONGITUDE    : num  10.4 10.4 10.4 10.4 10.4 ...
##  $ LATITUDE     : num  -84.6 -84.6 -84.6 -84.6 -84.6 ...
##  $ LAND_USE     : Factor w/ 2 levels "mixed_use","plantation": 1 1 2 2 1 1 1 1 1 1 ...
##  $ patch_grou   : Factor w/ 4 levels "A","B","C","D": 4 4 4 3 2 2 4 2 2 2 ...
##  $ OCCURRENCE   : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ logPATCH_AREA: num  -2.198 -1.716 -0.696 -0.714 -1.885 ...
##  $ logDIST_ROAD : num  2.18 5.21 4.27 3.52 3.35 ...
##  $ logPD        : num  7.5 7.25 5.3 6.4 6.69 ...
##  $ logLPI       : num  3.33 3.59 4.61 4.59 3.7 ...
##  $ logAREA_WM   : num  -2.366 -2.178 -0.696 -0.733 -1.834 ...
##  $ sqrtCWED     : num  31.1 24.8 0 13.7 18.6 ...
##  $ DIST_RIP_F   : num  0.1 6.05 196.81 272.45 127.11 ...
##  $ ECON_AM      : num  45.6 44.1 0 21.2 39.3 ...
##  $ EDGE         : num  691.2 451.2 0 94.2 268.8 ...
##  $ GYRATE_WM    : num  17.7 15.3 26.6 26 16.9 ...
##  $ PATCH_GYRA   : num  20.5 16.6 26.6 26.4 17.4 ...
##  $ PR           : num  5 4 1 2 3 3 5 4 2 4 ...
##  $ SHAPE_WM     : num  1.92 1.6 1.13 1.16 1.55 ...
```

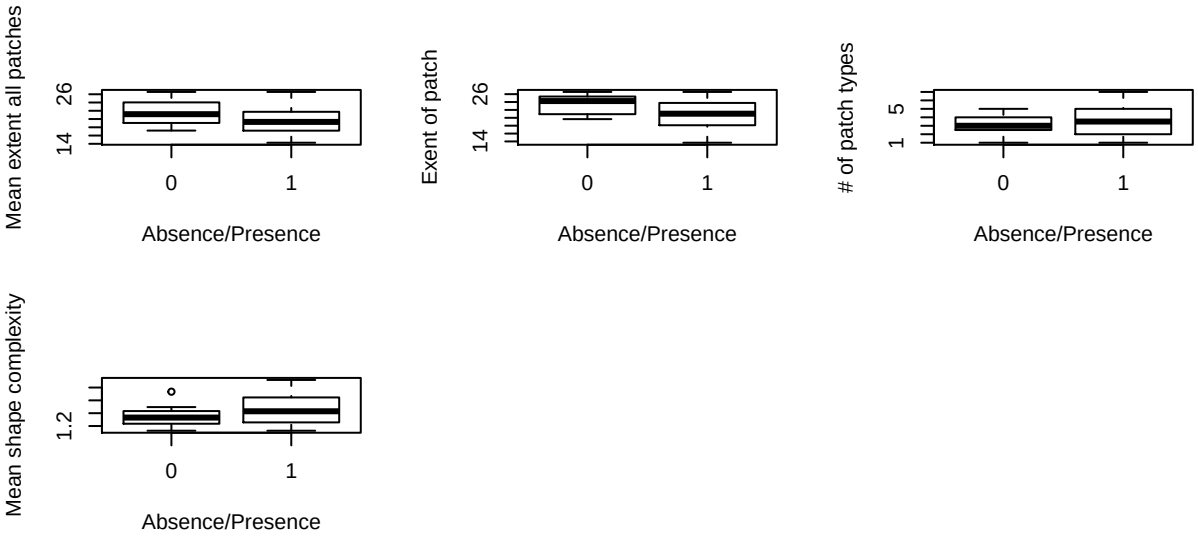
Let's visualize our data

Boxplots for all variables in relation to Sloth Presence/Absence

```
par(mfrow=c(3,3))
boxplot(logPATCH_AREA ~ OCCURRENCE, data = allsloth, xlab='Absence/Presence', ylab = 'Patch Area')
boxplot(logDIST_ROAD ~ OCCURRENCE, data = allsloth, xlab= "Absence/Presence", ylab = "Distance to road")
boxplot(logPD ~ OCCURRENCE, data = allsloth, xlab= "Absence/Presence", ylab = "Patch Density") #per 100
boxplot(logLPI ~ OCCURRENCE, data = allsloth, xlab= "Absence/Presence", ylab = "Prop of landscape in pa
boxplot(logAREA_WM ~ OCCURRENCE, data = allsloth, xlab= "Absence/Presence", ylab = "Mean Area") #Mean a
boxplot(sqrtCWED ~ OCCURRENCE, data = allsloth, xlab= "Absence/Presence", ylab = "Density of edge (cont
boxplot(DIST_RIP_F ~ OCCURRENCE, data = allsloth, xlab= "Absence/Presence", ylab = "Dist to nearest ripa
boxplot(ECON_AM ~ OCCURRENCE, data = allsloth, xlab= "Absence/Presence", ylab = "relative prop of edge
boxplot(EDGE ~ OCCURRENCE, data = allsloth, xlab= "Absence/Presence", ylab = "Length of edge") #Total l
```




```
par(mfrow=c(3,3))
boxplot(GYRATE_WM ~ OCCURRENCE, data = allsloth, xlab= "Absence/Presence", ylab = "Mean extent all patc
boxplot(PATCH_GYRA ~ OCCURRENCE, data = allsloth, xlab= "Absence/Presence", ylab = "Exent of patch") #E
boxplot(PR ~ OCCURRENCE, data = allsloth, xlab= "Absence/Presence", ylab = "# of patch types") #Number
boxplot(SHAPE_WM ~ OCCURRENCE, data = allsloth, xlab= "Absence/Presence", ylab = "Mean shape complexity
```



Using GLM (General Linear Model)

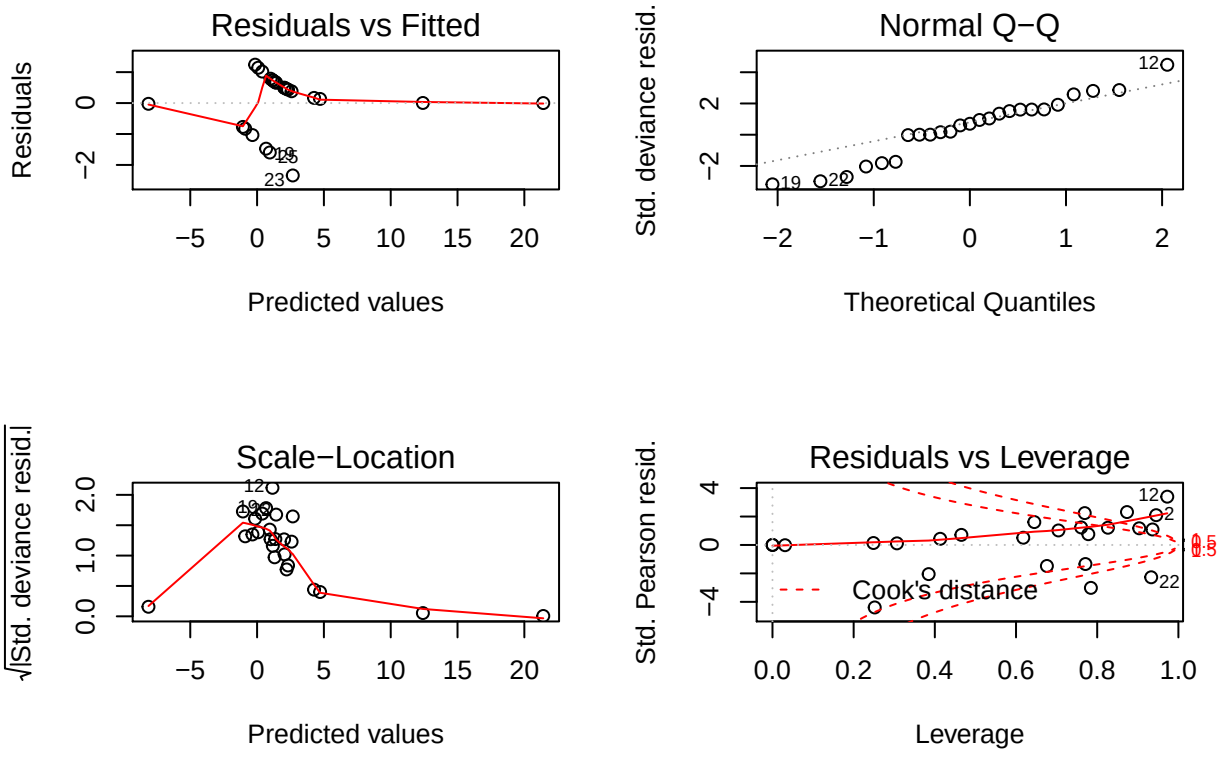
We want to know if any of these variables have an effect on presence/absence of sloths. Let's start with a general linear model to explore how sloth presence is a function of all our numeric variables.

```
# GLM
glm.sloth <- glm(OCCURRENCE ~ logPATCH_AREA + LAND_USE + logDIST_ROAD + logPD + logLPI + logAREA_WM + s

summary(glm.sloth)
```

```
##
## Call:
## glm(formula = OCCURRENCE ~ logPATCH_AREA + LAND_USE + logDIST_ROAD +
##     logPD + logLPI + logAREA_WM + sqrtCWED + DIST_RIP_F + ECON_AM +
##     EDGE + GYRATE_WM + PATCH_GYRA + PR + SHAPE_WM, family = binomial,
##     data = allsloth)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.3400  -0.0243   0.4336   0.6905   1.2407
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)         195.945784 267.964646   0.731    0.465
## logPATCH_AREA       -38.339788  34.484681  -1.112    0.266
## LAND_USEplantation -11.262983   9.554315  -1.179    0.238
```

```
## logDIST_ROAD          0.434644   0.897102   0.484    0.628
## logPD               -15.966172  19.725186  -0.809    0.418
## logLPI               14.946616  28.582318   0.523    0.601
## logAREA_WM           67.631360  65.333588   1.035    0.301
## sqrtCWED              0.252461   0.813674   0.310    0.756
## DIST_RIP_F           -0.007093   0.018879  -0.376    0.707
## ECON_AM               0.200762   0.353373   0.568    0.570
## EDGE                  0.104386   0.100029   1.044    0.297
## GYRATE_WM            -1.778528   2.863770  -0.621    0.535
## PATCH_GYRA           -2.267971   3.118349  -0.727    0.467
## PR                    3.319122   5.705280   0.582    0.561
## SHAPE_WM            -39.875707  47.309311  -0.843    0.399
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 29.648  on 24  degrees of freedom
## Residual deviance: 20.578  on 10  degrees of freedom
## AIC: 50.578
##
## Number of Fisher Scoring iterations: 8
```

```
par(mfrow=c(2,2))
plot(glm.sloth)
```



Hosmer-Lemeshow Goodness of Fit Test

This test will allow us to evaluate our glm and defines how well our model fits depending on the difference between the model and the observed data.

This is one approach for binary data. Based on our p-value, this model appears to be "ok" and shows that some of our covariates could potentially be significant. But, we haven't accounted for random effects to explain any variance in the response, and the way we can do this with our binary sloth data is to implement a mixed effects model.

```
# Test Fit of Selected GLM
hoslem.test(allsloth$OCCURRENCE, fitted(glm.sloth))
```

```
##
##  Hosmer and Lemeshow goodness of fit (GOF) test
##
## data:  allsloth$OCCURRENCE, fitted(glm.sloth)
## X-squared = 14.472, df = 8, p-value = 0.07025
```

To fit a model for the presence or absence of sloths, we will use glmer with family=binomial from the "lmer" package. Using a lmer function would not allow us to define family as "binomial"– which is what we are looking at with sloth presence/absence. But first, we need to scale the data and identify any collinearity among the variables.

```
# Sloth Data is scaled to make sure models run

allslothnotscaled <- allsloth
str(allsloth)

sc <- scale(allsloth[7:19])
allsloth <- cbind(allsloth[1:6], sc)
```

Identifying collinearity for our numeric variables only

```
headers <- colnames(allsloth)[7:19]
cor.m <- abs(cor(allsloth[,headers]))
cor.m.above75 <- cor.m > 0.75
cor.m.above75
```

```
##               logPATCH_AREA logDIST_ROAD logPD logLPI logAREA_WM sqrtCWED
## logPATCH_AREA          TRUE        FALSE  TRUE   TRUE       TRUE     TRUE
## logDIST_ROAD          FALSE         TRUE FALSE  FALSE      FALSE    FALSE
## logPD                  TRUE        FALSE  TRUE   TRUE       TRUE     TRUE
## logLPI                 TRUE        FALSE  TRUE   TRUE       TRUE     TRUE
## logAREA_WM             TRUE        FALSE  TRUE   TRUE       TRUE     TRUE
## sqrtCWED               TRUE        FALSE  TRUE   TRUE       TRUE     TRUE
## DIST_RIP_F            FALSE        FALSE FALSE   TRUE       TRUE    FALSE
## ECON_AM               FALSE        FALSE  TRUE   TRUE       TRUE     TRUE
## EDGE                   TRUE        FALSE  TRUE   TRUE       TRUE     TRUE
## GYRATE_WM              TRUE        FALSE  TRUE   TRUE       TRUE     TRUE
## PATCH_GYRA             TRUE        FALSE  TRUE   TRUE       TRUE     TRUE
## PR                     TRUE        FALSE  TRUE   TRUE       TRUE     TRUE
## SHAPE_WM               TRUE        FALSE  TRUE   TRUE       TRUE     TRUE
##               DIST_RIP_F ECON_AM  EDGE GYRATE_WM PATCH_GYRA    PR SHAPE_WM
## logPATCH_AREA      FALSE   FALSE  TRUE      TRUE       TRUE  TRUE     TRUE
## logDIST_ROAD       FALSE   FALSE FALSE     FALSE      FALSE FALSE    FALSE
## logPD              FALSE    TRUE  TRUE      TRUE       TRUE  TRUE     TRUE
```

```
## logLPI              TRUE    TRUE  TRUE      TRUE       TRUE  TRUE     TRUE
## logAREA_WM          TRUE    TRUE  TRUE      TRUE       TRUE  TRUE     TRUE
## sqrtCWED           FALSE    TRUE  TRUE      TRUE       TRUE  TRUE     TRUE
## DIST_RIP_F          TRUE   FALSE FALSE     FALSE      FALSE FALSE    FALSE
## ECON_AM            FALSE    TRUE FALSE      TRUE      FALSE FALSE     TRUE
## EDGE               FALSE   FALSE  TRUE      TRUE       TRUE  TRUE     TRUE
## GYRATE_WM          FALSE    TRUE  TRUE      TRUE       TRUE  TRUE     TRUE
## PATCH_GYRA         FALSE   FALSE  TRUE      TRUE       TRUE  TRUE     TRUE
## PR                 FALSE   FALSE  TRUE      TRUE       TRUE  TRUE     TRUE
## SHAPE_WM           FALSE    TRUE  TRUE      TRUE       TRUE  TRUE     TRUE
```

Bootstrapping

Because we only have 25 observations per variable, we bootstrap the data to get more data to account for singularity.

> Singularity is often a problem with mixed effect models with highly correlated data or not enough data.

```
# bootstrap an additional 100 samples

set.seed(537)
vec.boots <- sample(1:25, 100, replace = T)
allsloth <- allsloth[vec.boots,]
```

Now we run a loop for all of the following iterations

```
# create formulae to define models

ls.formulae.aov <- paste("OCCURRENCE ~", c( # prefix for all model formulae
                         "logDIST_ROAD * DIST_RIP_F",
                         "logDIST_ROAD + DIST_RIP_F",
                         "logDIST_ROAD * DIST_RIP_F + logAREA_WM",
                         "logDIST_ROAD * DIST_RIP_F * logAREA_WM",
                         "logDIST_ROAD * DIST_RIP_F + EDGE",
                         "logDIST_ROAD * DIST_RIP_F * GYRATE_WM",
                         "logDIST_ROAD * DIST_RIP_F + GYRATE_WM",
                         "logDIST_ROAD * DIST_RIP_F * PATCH_GYRA",
                         "logDIST_ROAD * DIST_RIP_F + PATCH_GYRA",
                         "logDIST_ROAD * DIST_RIP_F * PR",
                         "logDIST_ROAD * DIST_RIP_F + PR",
                         "logDIST_ROAD * DIST_RIP_F * SHAPE_WM",
                         "logDIST_ROAD * DIST_RIP_F + SHAPE_WM"
                       ))

ls.formulae <- paste(ls.formulae.aov, "+ (1 | LAND_USE)")
```

We create a function to run all of our interested formulas. Warning for running this formula– if data comes back with errors, it will crash the loop and need to address those formulas or write them out.

> Keep in mind, this is considered "data mining." Investigating this many models at once will usually return variables that are "significant." This is bad practice in science.

```
# make the glmer

ls.glmer <- lapply(ls.formulae, function(x) {print(x); glmer(x, data = allsloth, family = binomial)})
```

```
## Found more than one class "family" in cache; using the first, from namespace 'MatrixModels'
## Also defined by 'lme4'
## Found more than one class "family" in cache; using the first, from namespace 'MatrixModels'
## Also defined by 'lme4'
## Found more than one class "family" in cache; using the first, from namespace 'MatrixModels'
## Also defined by 'lme4'
## Found more than one class "family" in cache; using the first, from namespace 'MatrixModels'
## Also defined by 'lme4'
## Found more than one class "family" in cache; using the first, from namespace 'MatrixModels'
## Also defined by 'lme4'
## Found more than one class "family" in cache; using the first, from namespace 'MatrixModels'
## Also defined by 'lme4'
## Found more than one class "family" in cache; using the first, from namespace 'MatrixModels'
## Also defined by 'lme4'
## Found more than one class "family" in cache; using the first, from namespace 'MatrixModels'
## Also defined by 'lme4'
## Found more than one class "family" in cache; using the first, from namespace 'MatrixModels'
## Also defined by 'lme4'
## Found more than one class "family" in cache; using the first, from namespace 'MatrixModels'
## Also defined by 'lme4'
## Found more than one class "family" in cache; using the first, from namespace 'MatrixModels'
## Also defined by 'lme4'
## Found more than one class "family" in cache; using the first, from namespace 'MatrixModels'
## Also defined by 'lme4'
## Found more than one class "family" in cache; using the first, from namespace 'MatrixModels'
## Also defined by 'lme4'
## Found more than one class "family" in cache; using the first, from namespace 'MatrixModels'
## Also defined by 'lme4'
## Found more than one class "family" in cache; using the first, from namespace 'MatrixModels'
## Also defined by 'lme4'
## Found more than one class "family" in cache; using the first, from namespace 'MatrixModels'
## Also defined by 'lme4'
## Found more than one class "family" in cache; using the first, from namespace 'MatrixModels'
## Also defined by 'lme4'
## Found more than one class "family" in cache; using the first, from namespace 'MatrixModels'
## Also defined by 'lme4'
```

```
## Found more than one class "family" in cache; using the first, from namespace 'MatrixModels'
## Also defined by 'lme4'
## Found more than one class "family" in cache; using the first, from namespace 'MatrixModels'
## Also defined by 'lme4'
## Found more than one class "family" in cache; using the first, from namespace 'MatrixModels'
## Also defined by 'lme4'
## Found more than one class "family" in cache; using the first, from namespace 'MatrixModels'
## Also defined by 'lme4'
## Found more than one class "family" in cache; using the first, from namespace 'MatrixModels'
## Also defined by 'lme4'
## Found more than one class "family" in cache; using the first, from namespace 'MatrixModels'
## Also defined by 'lme4'
## Found more than one class "family" in cache; using the first, from namespace 'MatrixModels'
## Also defined by 'lme4'
## Found more than one class "family" in cache; using the first, from namespace 'MatrixModels'
## Also defined by 'lme4'
## Found more than one class "family" in cache; using the first, from namespace 'MatrixModels'
## Also defined by 'lme4'
## Found more than one class "family" in cache; using the first, from namespace 'MatrixModels'
## Also defined by 'lme4'
## Found more than one class "family" in cache; using the first, from namespace 'MatrixModels'
## Also defined by 'lme4'
## Found more than one class "family" in cache; using the first, from namespace 'MatrixModels'
## Also defined by 'lme4'
## Found more than one class "family" in cache; using the first, from namespace 'MatrixModels'
## Also defined by 'lme4'
## Found more than one class "family" in cache; using the first, from namespace 'MatrixModels'
## Also defined by 'lme4'
## Found more than one class "family" in cache; using the first, from namespace 'MatrixModels'
## Also defined by 'lme4'
## Found more than one class "family" in cache; using the first, from namespace 'MatrixModels'
## Also defined by 'lme4'
## Found more than one class "family" in cache; using the first, from namespace 'MatrixModels'
## Also defined by 'lme4'
## Found more than one class "family" in cache; using the first, from namespace 'MatrixModels'
## Also defined by 'lme4'
```

```
## Found more than one class "family" in cache; using the first, from namespace 'MatrixModels'
## Also defined by 'lme4'
## Found more than one class "family" in cache; using the first, from namespace 'MatrixModels'
## Also defined by 'lme4'
## Found more than one class "family" in cache; using the first, from namespace 'MatrixModels'
## Also defined by 'lme4'
```

AIC Model Comparison

Extract AIC values to identify best fit models from our iteration.

```
# extract AIC
ls.aic <- sapply(ls.glmer, function(x) summary(x)$AICtab[1])
```

Convert the list to a dataframe for ease of use and identification of model.

```
# convert to df, pull out the good models

df.modeleval <- data.frame(model = ls.formulae.aov, aic = ls.aic, index = 1:length(ls.formulae.aov), ra
df.modeleval$model <- as.character(df.modeleval$model)

df.modeleval$daic <- df.modeleval$aic - min(df.modeleval$aic)
df.modeleval <- df.modeleval[order(df.modeleval$daic),]
head(df.modeleval)
```

So what does the best model look like?

```
topmodel <- ls.glmer[[10]]
nexttopmodel <- ls.glmer[[6]]

summary(topmodel)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: binomial  ( logit )
## Formula: OCCURRENCE ~ logDIST_ROAD * DIST_RIP_F * PR + (1 | LAND_USE)
##    Data: allsloth
##
##      AIC      BIC   logLik deviance df.resid
##     48.3     71.7    -15.1     30.3       91
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -1.83046  0.00000  0.00479  0.10363  0.89291
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  LAND_USE (Intercept) 60.83    7.799
## Number of obs: 100, groups:  LAND_USE, 2
##
## Fixed effects:
##                             Estimate Std. Error z value Pr(>|z|)
## (Intercept)                  -4.8435     6.3521  -0.762  0.44576
```

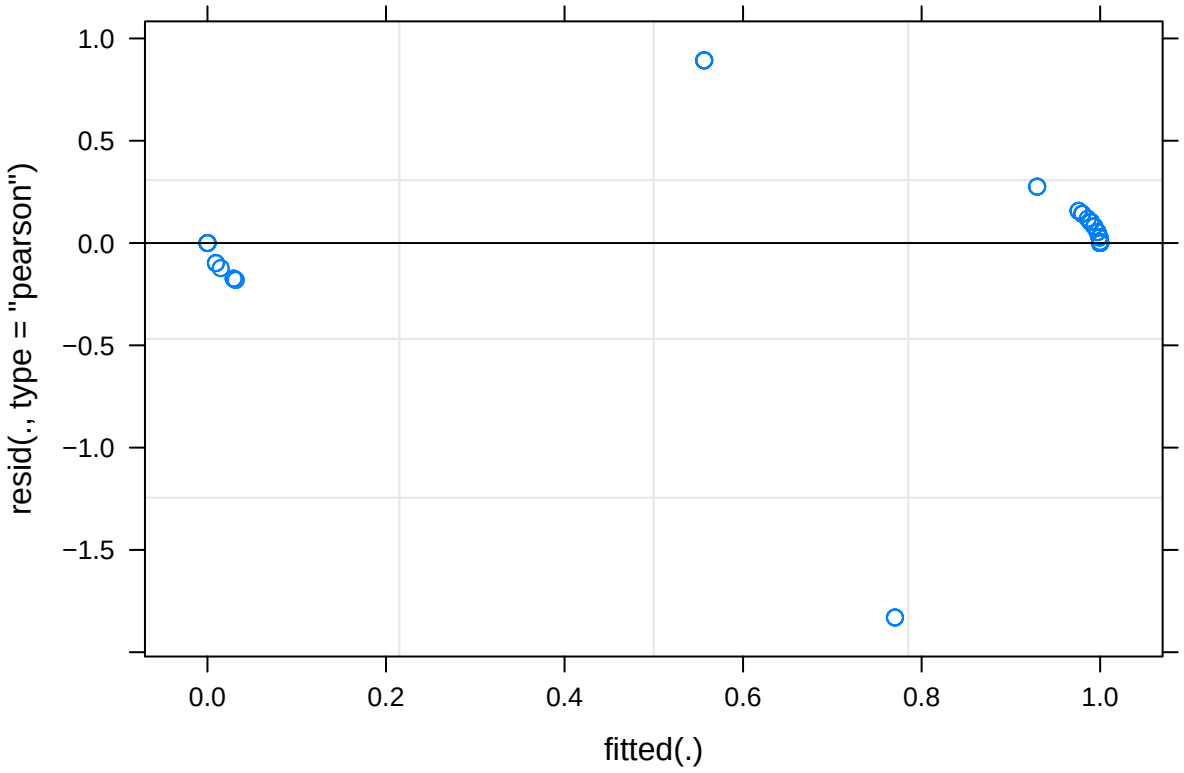
```
## logDIST_ROAD                 1.0124     1.9692   0.514  0.60717
## DIST_RIP_F                  -9.7892     4.6718  -2.095  0.03614 *
## PR                          -7.0951     2.5757  -2.755  0.00588 **
## logDIST_ROAD:DIST_RIP_F    -13.7165     4.6043  -2.979  0.00289 **
## logDIST_ROAD:PR             -0.7647     1.8579  -0.412  0.68063
## DIST_RIP_F:PR              -14.6826     5.8270  -2.520  0.01174 *
## logDIST_ROAD:DIST_RIP_F:PR   0.1940     2.0854   0.093  0.92588
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##                        (Intr) lgDIST_ROAD DIST_RIP_F PR
## lgDIST_ROAD            -0.063
## DIST_RIP_F              0.158 -0.711
## PR                      0.393 -0.102       0.445
## lgDIST_ROAD:DIST_RIP_F  0.324 -0.308       0.673      0.521
## lDIST_ROAD:P            0.194  0.683      -0.413      0.064
## DIST_RIP_F:             0.277 -0.635       0.918      0.531
## lDIST_ROAD:DIST_RIP_F: -0.198  0.134       0.095     -0.328
##                        lgDIST_ROAD:DIST_RIP_F lDIST_ROAD:P DIST_RIP_F:
## lgDIST_ROAD
## DIST_RIP_F
## PR
## lgDIST_ROAD:DIST_RIP_F
## lDIST_ROAD:P            0.193
## DIST_RIP_F:             0.813                 -0.220
## lDIST_ROAD:DIST_RIP_F:  0.081                  0.037       -0.013
## convergence code: 0
## Model failed to converge with max|grad| = 0.0610467 (tol = 0.001, component 1)
```

```
summary(nexttopmodel)
```

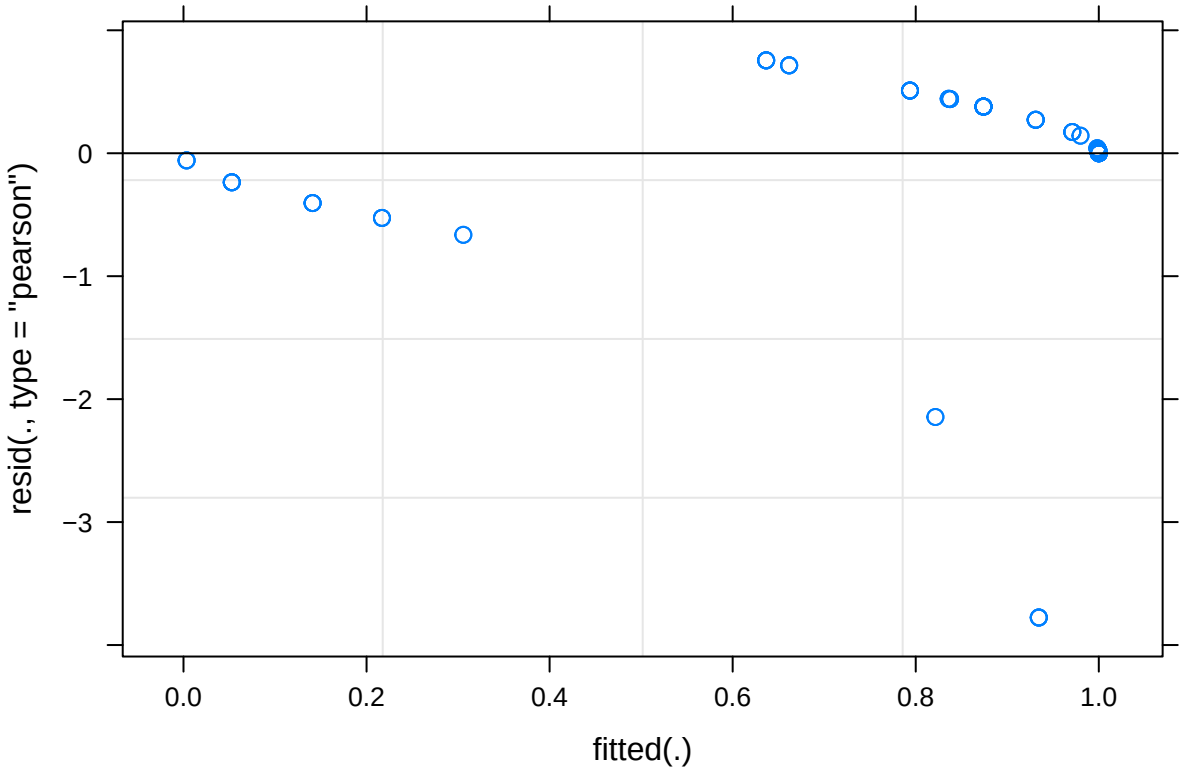
```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: binomial  ( logit )
## Formula:
## OCCURRENCE ~ logDIST_ROAD * DIST_RIP_F * GYRATE_WM + (1 | LAND_USE)
##    Data: allsloth
##
##      AIC      BIC   logLik deviance df.resid
##     80.3    103.7    -31.1     62.3       91
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -3.7763  0.0000  0.0279  0.3796  0.7556
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  LAND_USE (Intercept) 20.49    4.527
## Number of obs: 100, groups:  LAND_USE, 2
##
## Fixed effects:
##                                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)                     -0.75168    3.31600  -0.227  0.82067
## logDIST_ROAD                     1.36639    0.63636   2.147  0.03178 *
```

```
## DIST_RIP_F                        -2.00942    0.74332  -2.703  0.00687 **
## GYRATE_WM                          0.09988    0.60320   0.166  0.86848
## logDIST_ROAD:DIST_RIP_F           -5.68677    1.90678  -2.982  0.00286 **
## logDIST_ROAD:GYRATE_WM             1.18666    0.80166   1.480  0.13880
## DIST_RIP_F:GYRATE_WM               5.99374    2.07665   2.886  0.00390 **
## logDIST_ROAD:DIST_RIP_F:GYRATE_WM -0.91318    0.92154  -0.991  0.32172
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##                       (Intr) lgDIST_ROAD DIST_RIP_F GYRATE
## lgDIST_ROAD           -0.051
## DIST_RIP_F            -0.006 -0.409
## GYRATE_WM             -0.008 -0.192      -0.195
## lgDIST_ROAD:DIST_RIP_F  0.174 -0.256       0.297      0.062
## lDIST_ROAD:G          -0.200 -0.037       0.067     -0.194
## DIST_RIP_F:           -0.203  0.387      -0.287     -0.109
## lDIST_ROAD:DIST_RIP_F:  0.142 -0.336      -0.220     -0.012
##                       lgDIST_ROAD:DIST_RIP_F lDIST_ROAD:G DIST_RIP_F:
## lgDIST_ROAD
## DIST_RIP_F
## GYRATE_WM
## lgDIST_ROAD:DIST_RIP_F
## lDIST_ROAD:G          -0.769
## DIST_RIP_F:           -0.920                  0.734
## lDIST_ROAD:DIST_RIP_F:  0.407                -0.442       -0.525
```

```
plot(topmodel)
```




```
plot(nexttopmodel)
```


resid(., type = "pearson")
0
−1
−2
−3
0.0
0.2
0.4
0.6
0.8
1.0
fitted(.)