Day 2 - OLS Assumptions

David Carlson

September 23, 2020

Gauss-Markov assumptions

- The model is linear in the parameters
- No endogeneity in the model (independent variable X and ϵ are not correlated)
- Errors are normally distributed with constant variance
- No autocorrelation in the errors
- No multicollinearity between variable

Linearity

- The relationship between the predictor (x) and the outcome (y) is assumed to be linear
- Non-linearity of the outcome - predictor relationships
- Model plots: – Residuals vs Fitted. Used to check the linear relationship assumptions. A horizontal line, without distinct patterns is an indication for a linear relationship, what is good. – Normal Q-Q. Used to examine whether the residuals are normally distributed. It's good if residuals points follow the straight dashed line. – Scale-Location (or Spread-Location). Used to check the homogeneity of variance of the residuals (homoscedasticity). Horizontal line with equally spread points is a good indication of homoscedasticity. – Residuals vs Leverage. Used to identify influential cases, that is extreme values that might influence the regression results when included or excluded from the analysis.

```r
load('merged.Rdata') #load the data
head(merged)
```

```
##   CO2 year      pais    avgAA no.emig    exports    imports   pop remit
## 1 ARG 1995 argentina 3.399813   52440 1803371008 4206580992 24666    NA
## 2 ARG 1996 argentina 3.186754   92774 1974341632 4749123584    NA    NA
## 3 ARG 1997 argentina 3.394000  105734 2204026112 6085211136    NA    NA
## 4 ARG 1998 argentina 3.301217  120104 2211580160 6227363840    NA    NA
## 5 ARG 2000 argentina 2.688889  115978 3148713321 4784868410 26565    NA
## 6 ARG 2001 argentina 2.602674  139375 2900129494 3781205761    NA    NA
##        aid  inflation unempl        GDP  avgPop    propEmig
## 1        0  3.3761168  18.80 -2.8452096 27569.5 0.001902102
## 2        0  0.1556959  17.20  5.5266898 27569.5 0.003365095
## 3        0  0.5272583  14.90  8.1110468 27569.5 0.003835180
## 4   270000  0.9203365  12.80  3.8501789 27569.5 0.004356408
## 5 -2470000 -0.9359394  15.02 -0.7889989 27569.5 0.004206750
## 6  -470000 -1.0666355  17.40 -4.4088397 27569.5 0.005055405
```

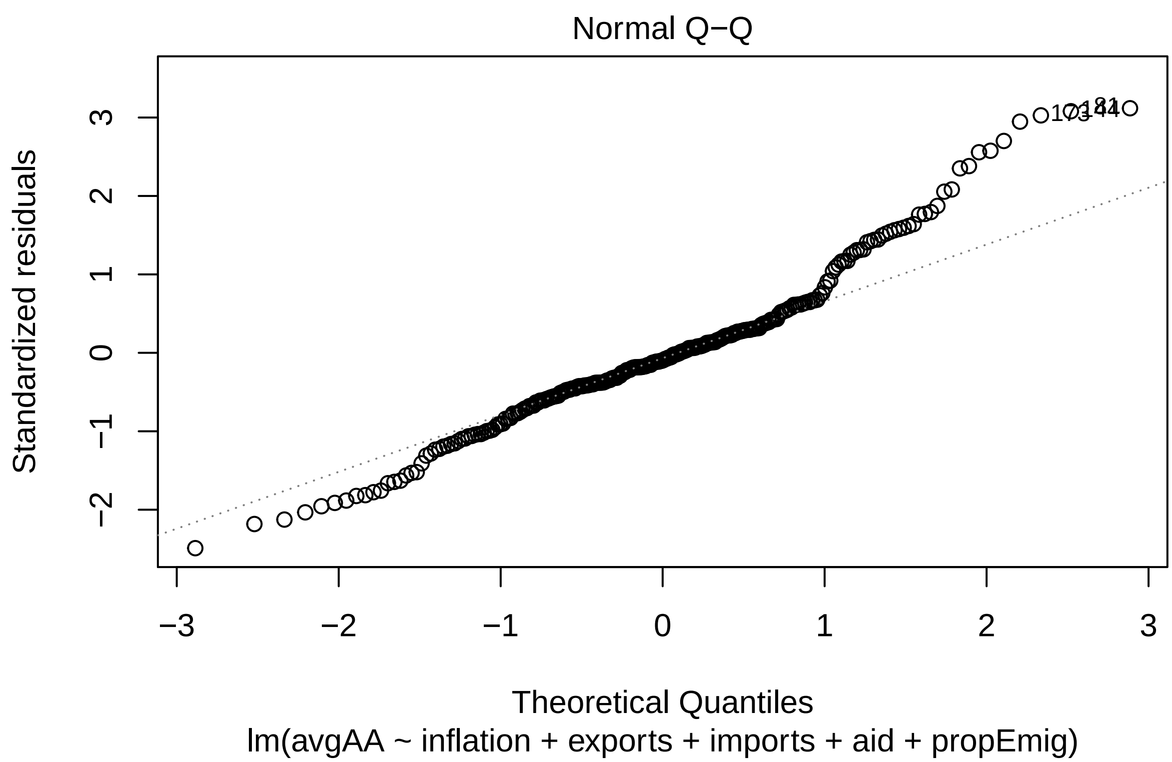
```r
mergedY = merged[!is.na(merged$avgAA),]
#model sentiment towards US as a function of inflation, with theoretical controls
mod = lm(avgAA ~ inflation + exports + imports + aid + propEmig, data = mergedY)
summary(mod)
```

```
##
## Call:
## lm(formula = avgAA ~ inflation + exports + imports + aid + propEmig,
##     data = mergedY)
```

```
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.9474 -0.2100 -0.0350  0.1534  1.1864
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.813e+00  4.177e-02  67.336  < 2e-16 ***
## inflation    9.121e-03  1.991e-03   4.582 7.29e-06 ***
## exports     -8.164e-12  3.396e-12  -2.404   0.0169 *
## imports      7.399e-12  5.024e-12   1.473   0.1421
## aid          1.147e-10  2.250e-10   0.510   0.6107
## propEmig     2.422e+00  4.490e-01   5.394 1.60e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3817 on 250 degrees of freedom
## Multiple R-squared:  0.1915, Adjusted R-squared:  0.1753
## F-statistic: 11.84 on 5 and 250 DF,  p-value: 2.726e-10
```

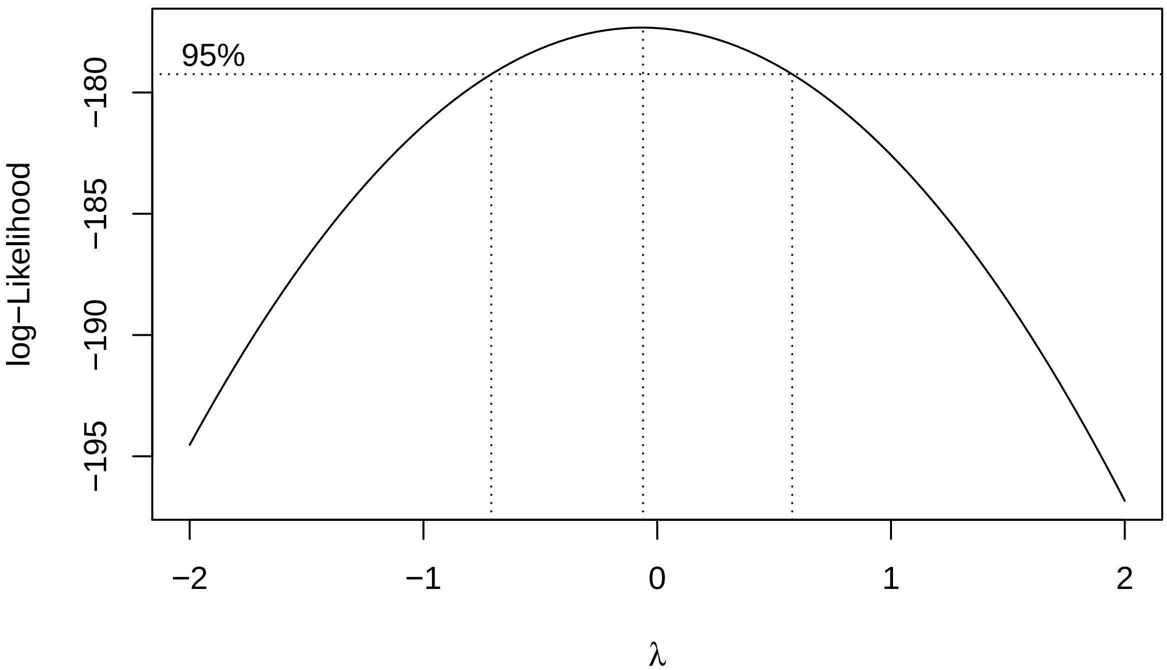
- To assess the assumption of linearity we want to ensure that the residuals are not too far away from 0 (standardized values less than -2 or greater than 2 are deemed problematic). To assess if the homoscedasticity assumption is met we look to make sure that there is no pattern in the residuals and that they are equally spread around the y = 0 line

```r
plot(mod, 2) #is this linear?
```



- Boxcox tranformation: Generic function used to compute the value(s) of an objective for one or more Box-Cox power transformations, or to compute an optimal power transformation based on a specified objective
- Data transformations are often used to induce normality, homoscedasticity, and/or linearity

```r
library(MASS)
boxcox(mod) #Box-Cox method only allows for strictly positive outcome
```



- Box-Cox: If lambda does not equal zero, transform outcome to $\frac{y^\lambda - 1}{\lambda}$, if zero, take the log

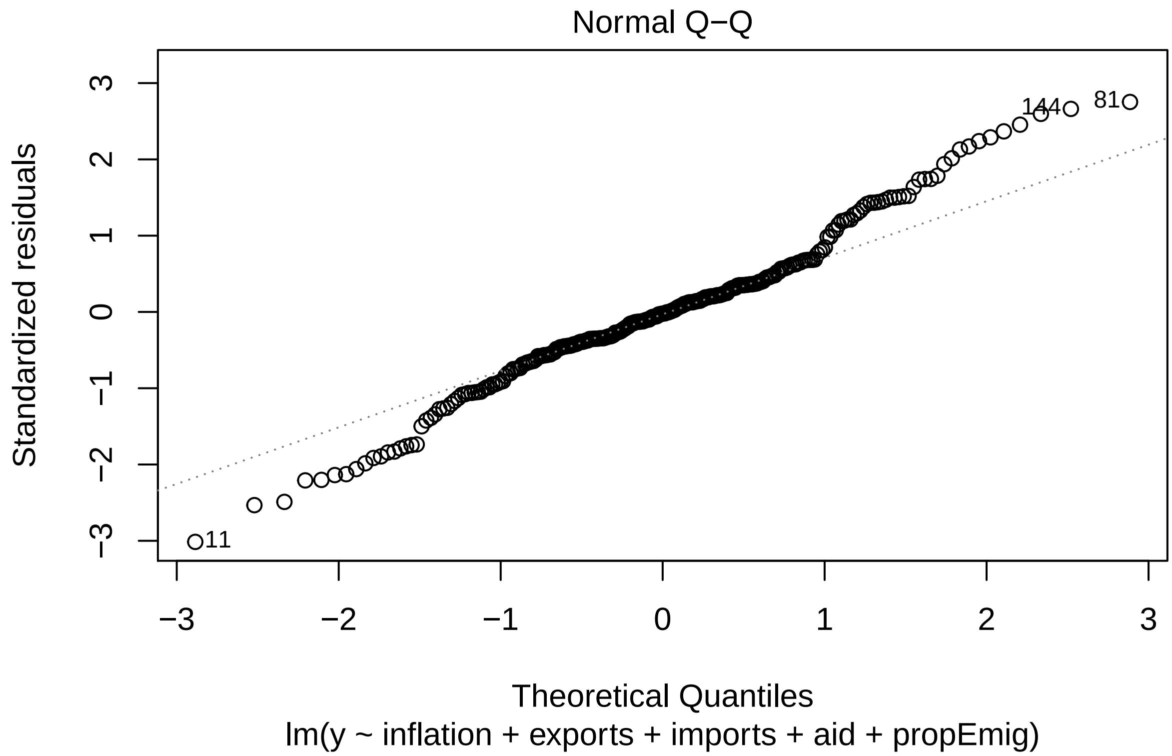
```r
mod2 = lm(I(log(avgAA)) ~ inflation + exports + imports + aid + propEmig, data = mergedY)
summary(mod)
```

```
##
## Call:
## lm(formula = avgAA ~ inflation + exports + imports + aid + propEmig,
##     data = mergedY)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.9474 -0.2100 -0.0350  0.1534  1.1864
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.813e+00  4.177e-02  67.336  < 2e-16 ***
## inflation    9.121e-03  1.991e-03   4.582 7.29e-06 ***
## exports     -8.164e-12  3.396e-12  -2.404   0.0169 *
## imports      7.399e-12  5.024e-12   1.473   0.1421
## aid          1.147e-10  2.250e-10   0.510   0.6107
## propEmig     2.422e+00  4.490e-01   5.394 1.60e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3817 on 250 degrees of freedom
## Multiple R-squared:  0.1915, Adjusted R-squared:  0.1753
## F-statistic: 11.84 on 5 and 250 DF,  p-value: 2.726e-10
```

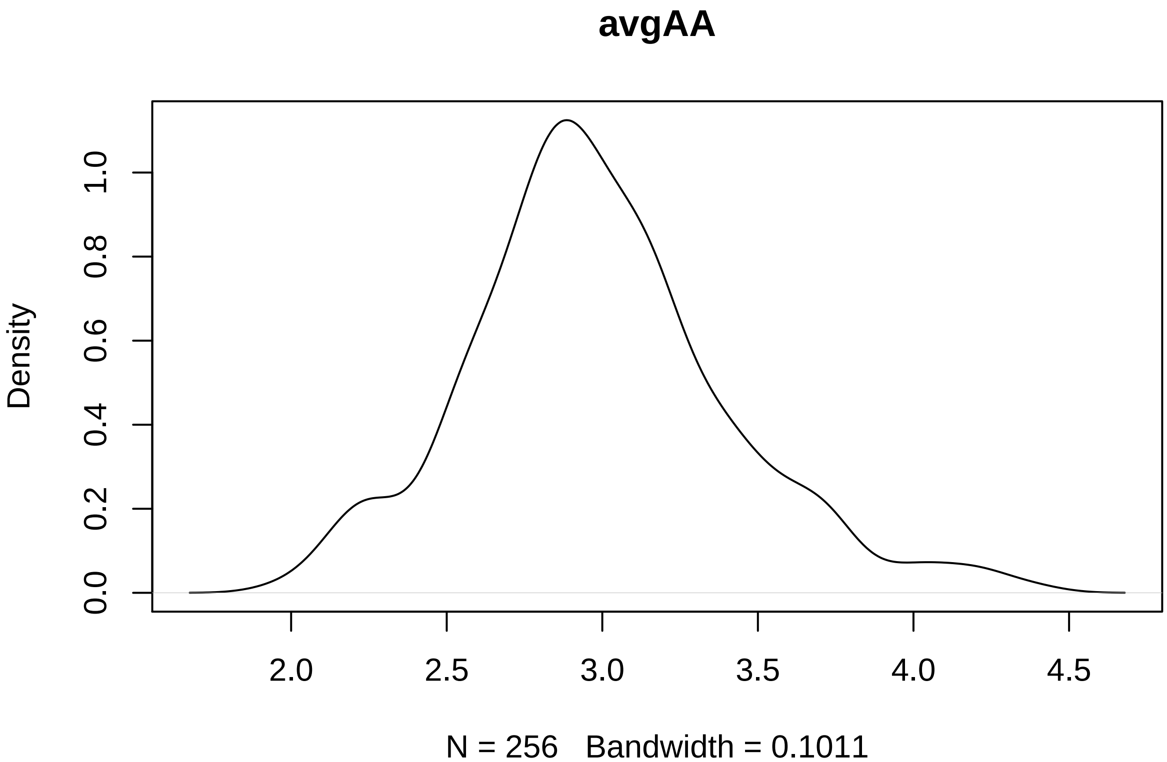
```r
summary(mod2)
```

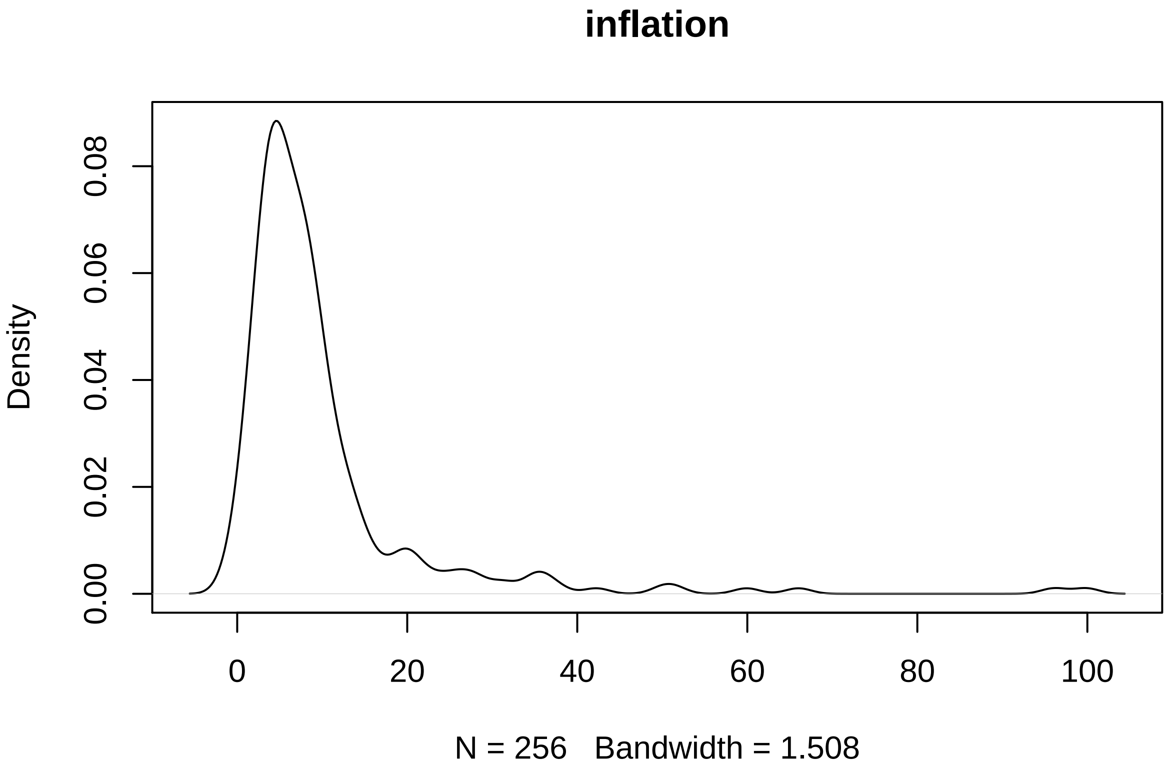
```
##
## Call:
## lm(formula = I(log(avgAA)) ~ inflation + exports + imports +
##     aid + propEmig, data = mergedY)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.37986 -0.06636 -0.00282  0.05911  0.34690
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.025e+00  1.384e-02  74.061  < 2e-16 ***
## inflation    2.943e-03  6.594e-04   4.463 1.22e-05 ***
## exports     -2.975e-12  1.125e-12  -2.645  0.00869 **
## imports      2.803e-12  1.664e-12   1.684  0.09343 .
## aid          6.401e-11  7.453e-11   0.859  0.39126
## propEmig     8.289e-01  1.487e-01   5.574 6.44e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1264 on 250 degrees of freedom
## Multiple R-squared:  0.1969, Adjusted R-squared:  0.1809
## F-statistic: 12.26 on 5 and 250 DF,  p-value: 1.216e-10
```

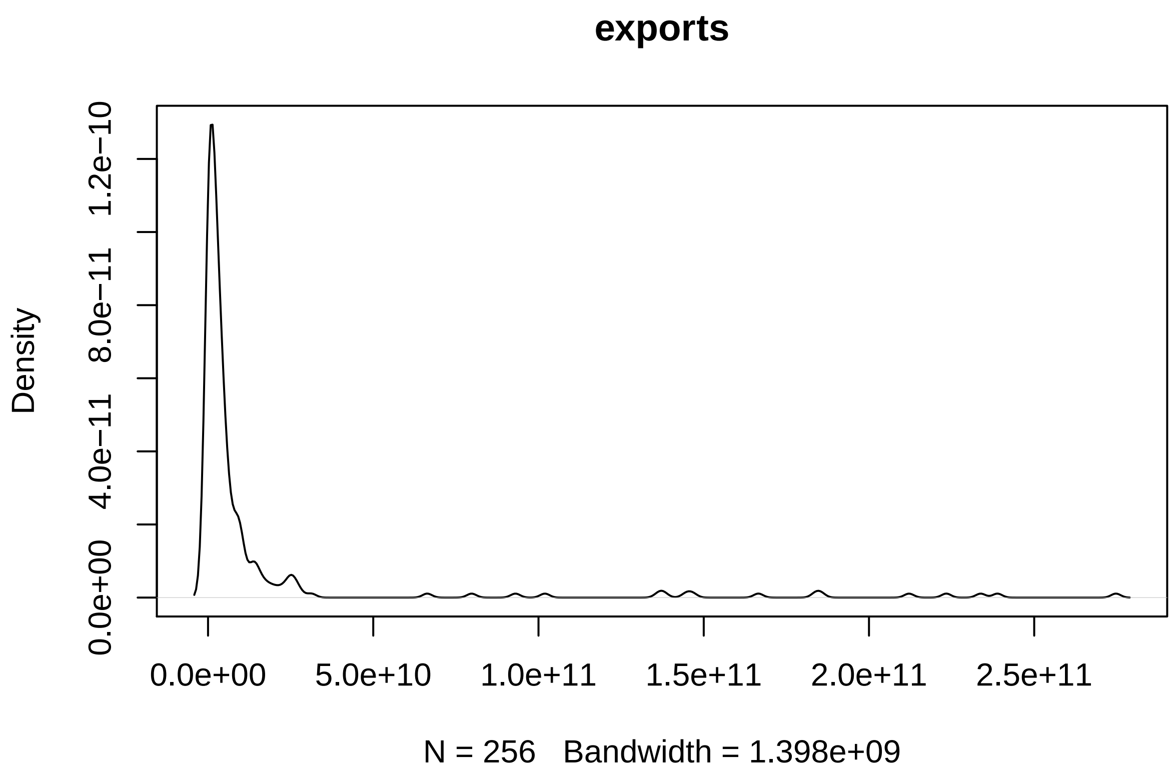
```r
#plot(mod2, 2) #need to create a new variable (I is not allowed in this function)
mergedY$y = log(mergedY$avgAA)
mod3 = lm(y ~ inflation + exports + imports + aid + propEmig, data = mergedY)
plot(mod3, 2) #still does not solve it - let's look at densities
```

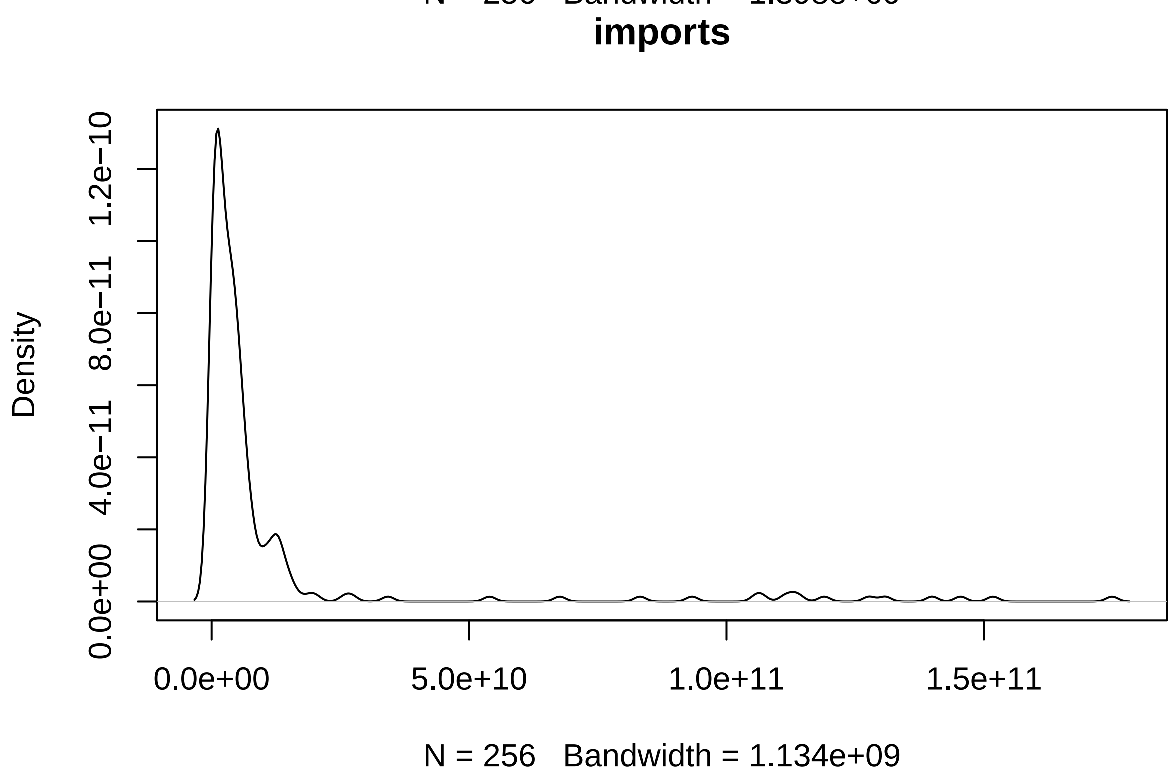


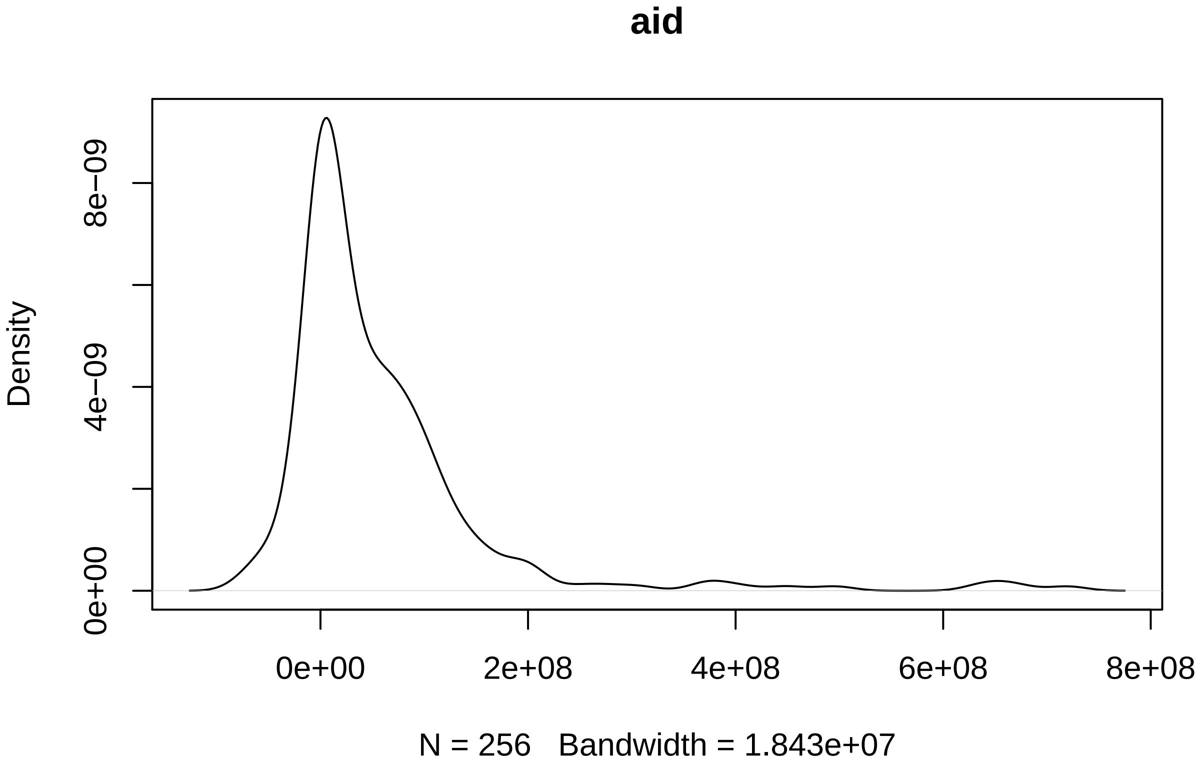
```r
vars = c('avgAA', 'inflation', 'exports', 'imports', 'aid', 'propEmig')
for(var in vars) plot(density(mergedY[,var]), main = var)
```

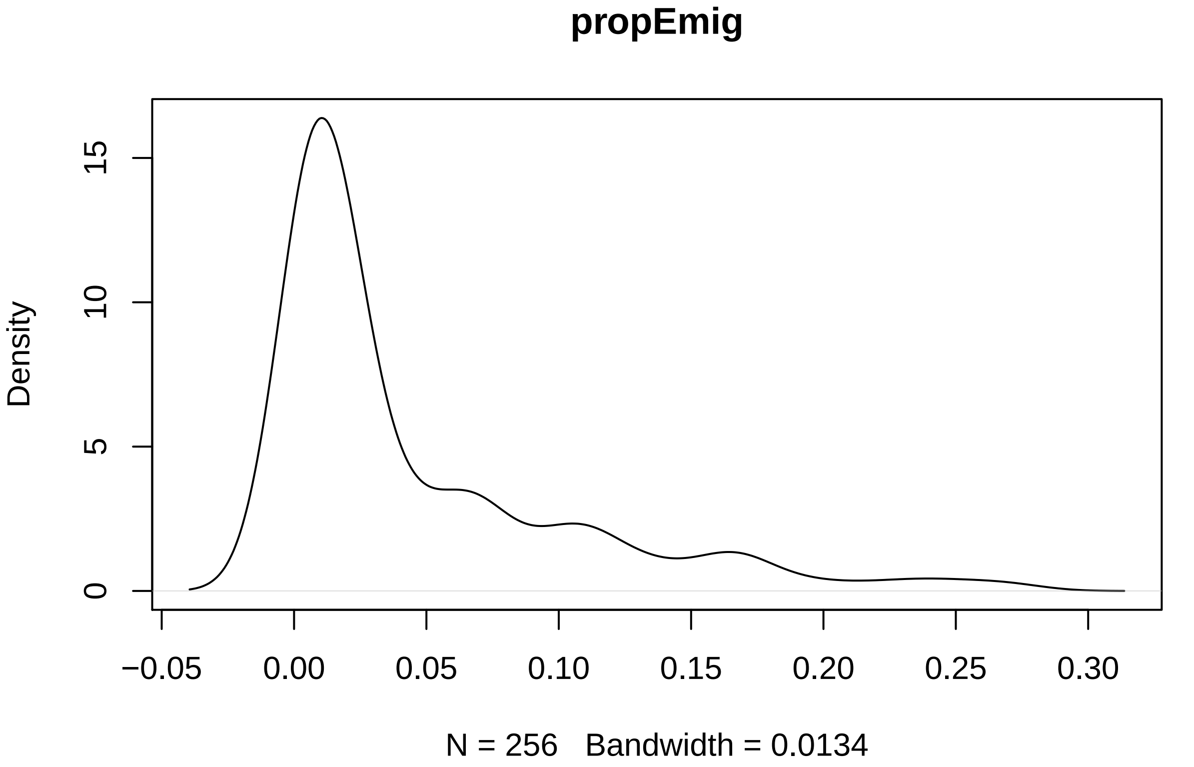





exports
Density
0.0e+00
4.0e-11
8.0e-11
1.2e-10
0.0e+00
5.0e+10
1.0e+11
1.5e+11
2.0e+11
2.5e+11
N = 256 Bandwidth = 1.398e+09


imports
Density
0.0e+00
4.0e-11
8.0e-11
1.2e-10
0.0e+00
5.0e+10
1.0e+11
1.5e+11
N = 256 Bandwidth = 1.134e+09





- All of the independent variables are problematic, with long tails

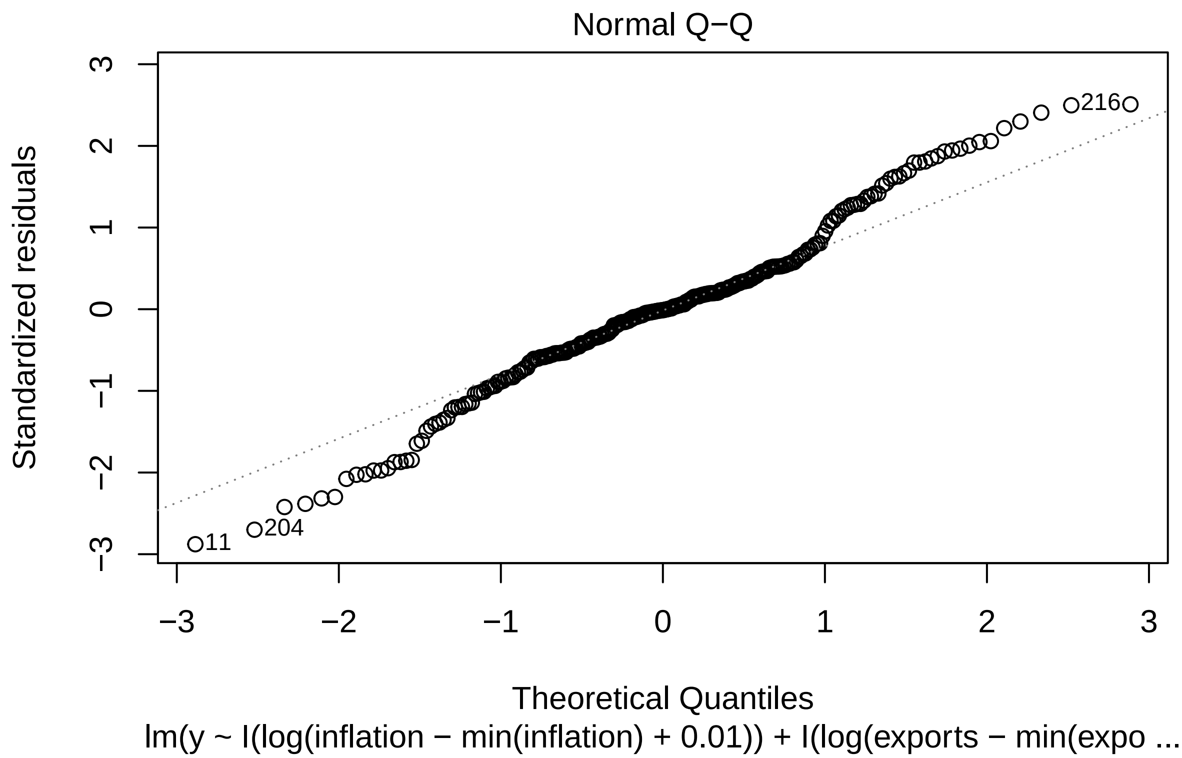
```
summary(mergedY[, vars])
```

```
##      avgAA          inflation          exports              imports
##  Min.   :1.977   Min.   :-1.067   Min.   :2.933e+07   Min.   :6.397e+07
```

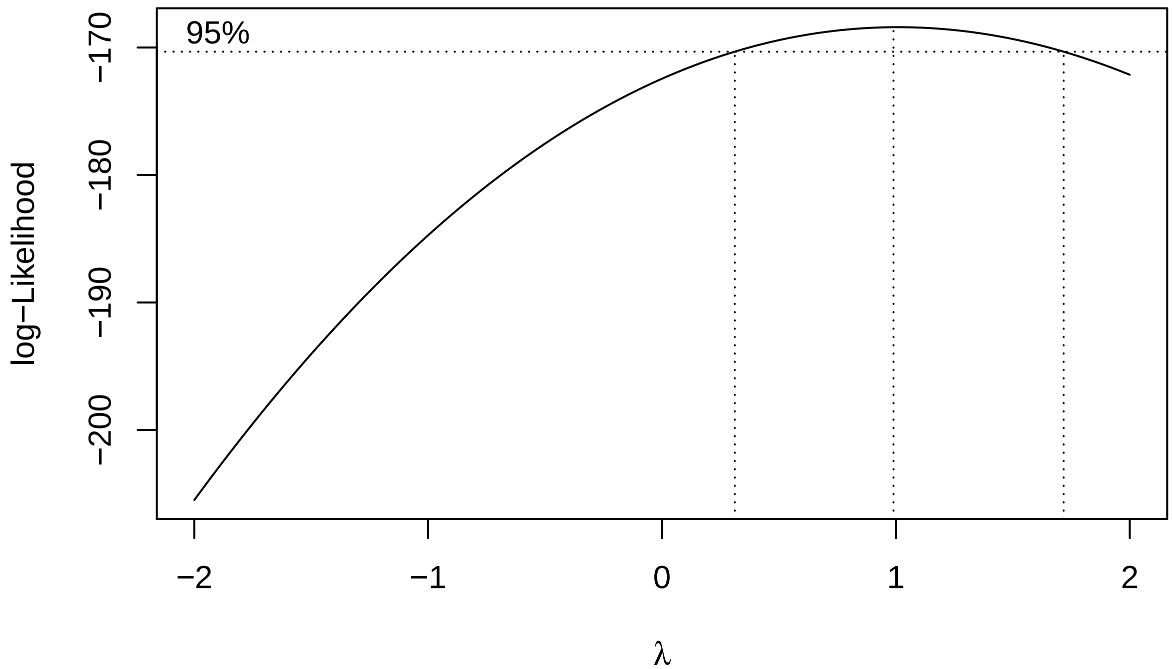
```
##  1st Qu.:2.736   1st Qu.: 3.983   1st Qu.:5.373e+08   1st Qu.:1.179e+09
##  Median :2.940   Median : 6.785   Median :2.714e+09   Median :3.564e+09
##  Mean   :2.987   Mean   : 9.953   Mean   :1.451e+10   Mean   :1.129e+10
##  3rd Qu.:3.192   3rd Qu.:10.788   3rd Qu.:6.849e+09   3rd Qu.:6.298e+09
##  Max.   :4.376   Max.   :99.877   Max.   :2.747e+11   Max.   :1.749e+11
##       aid                propEmig
##  Min.   :-70750000   Min.   :0.0007504
##  1st Qu.:   510000   1st Qu.:0.0068730
##  Median : 27175000   Median :0.0186564
##  Mean   : 60035156   Mean   :0.0460132
##  3rd Qu.: 83680000   3rd Qu.:0.0673746
##  Max.   :719750000   Max.   :0.2734466
```

- But, values are not strictly positive

```r
mod4 = lm(y ~ I(log(inflation - min(inflation) + .01)) + I(log(exports - min(exports) + .01)) + I(log(in
plot(mod4, 2)
```




```r
boxcox(mod4)
```




```r
summary(mod4)
```

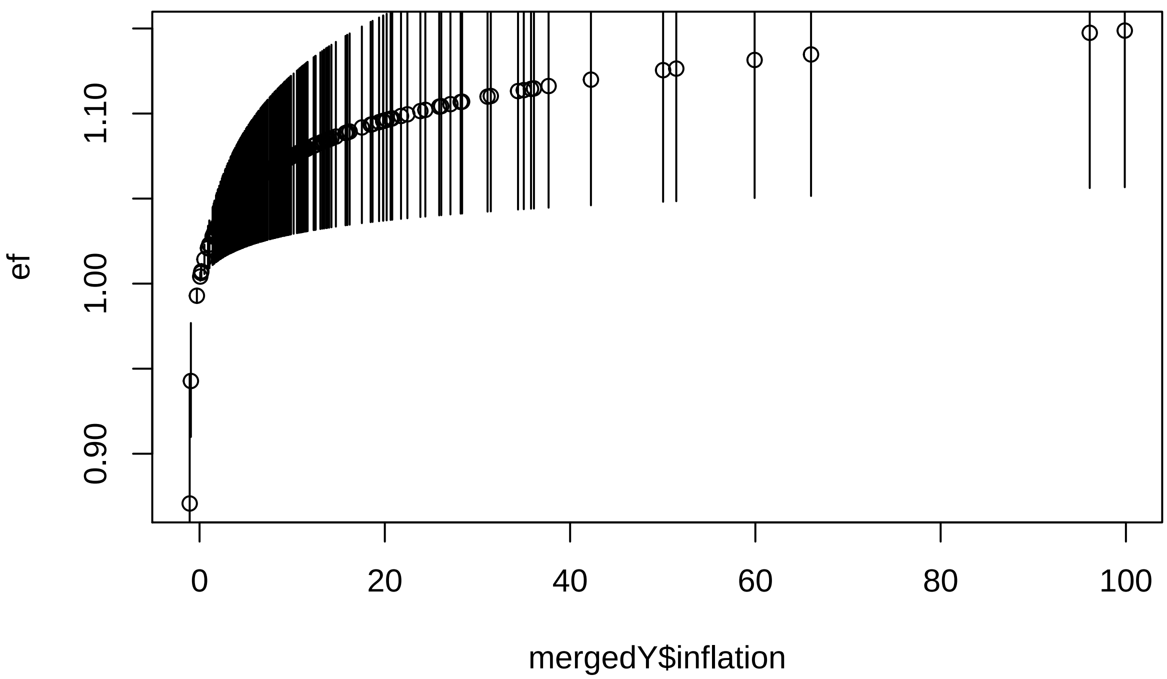
```
##
## Call:
## lm(formula = y ~ I(log(inflation - min(inflation) + 0.01)) +
##     I(log(exports - min(exports) + 0.01)) + I(log(imports - min(imports) +
##     0.01)) + I(log(aid - min(aid) + 0.01)) + I(log(propEmig -
##     min(propEmig) + 0.01)), data = mergedY)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.37616 -0.07056 -0.00148  0.06571  0.32687
##
## Coefficients:
##                                            Estimate Std. Error t value
## (Intercept)                                1.362352   0.148881   9.151
## I(log(inflation - min(inflation) + 0.01))  0.030053   0.009238   3.253
## I(log(exports - min(exports) + 0.01))     -0.012695   0.004135  -3.070
## I(log(imports - min(imports) + 0.01))      0.005977   0.004661   1.282
## I(log(aid - min(aid) + 0.01))             -0.003266   0.005362  -0.609
## I(log(propEmig - min(propEmig) + 0.01))    0.040945   0.009292   4.407
##                                           Pr(>|t|)
## (Intercept)                                < 2e-16 ***
## I(log(inflation - min(inflation) + 0.01))  0.00130 **
## I(log(exports - min(exports) + 0.01))      0.00238 **
## I(log(imports - min(imports) + 0.01))      0.20086
## I(log(aid - min(aid) + 0.01))              0.54300
## I(log(propEmig - min(propEmig) + 0.01))   1.56e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1313 on 250 degrees of freedom
## Multiple R-squared:  0.1341, Adjusted R-squared:  0.1168
```

```
## F-statistic: 7.744 on 5 and 250 DF,  p-value: 8.622e-07
```

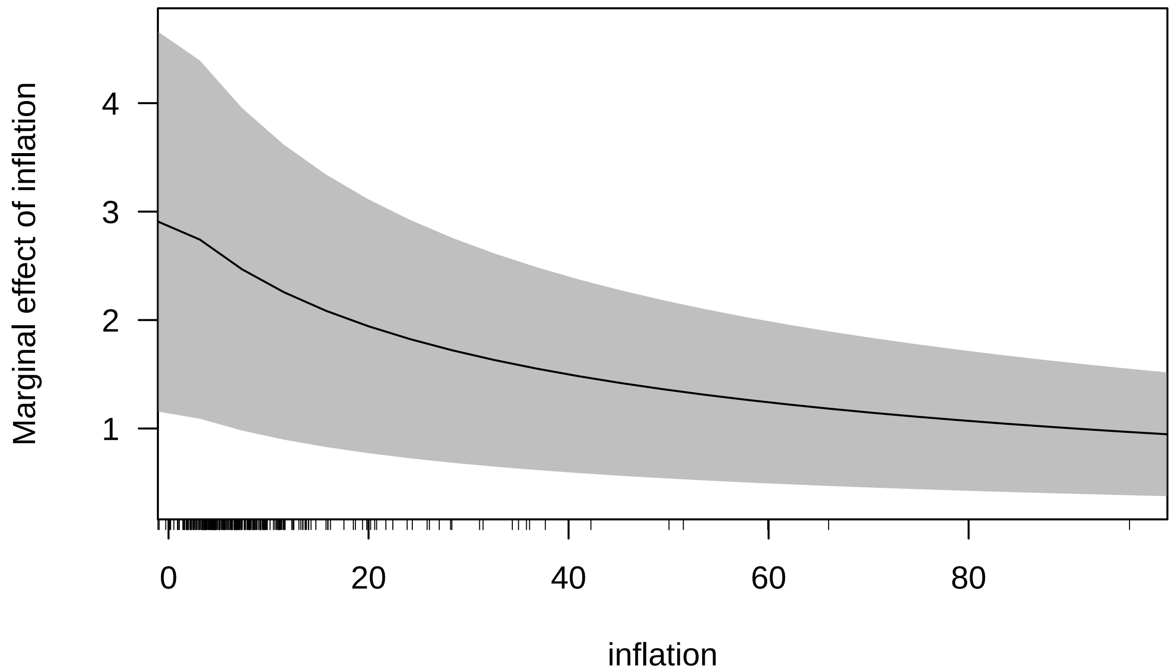
```
#get it back to interpretable
coef(mod)[2]
```

```
##   inflation
## 0.009120888
```

```
#log(y) ~ log(X - c)*b
#exponentiate both sides
#y ~ (X - c)^b
ef = (mergedY$inflation - min(mergedY$inflation) + .01)^coef(mod4)[2]
plot(mergedY$inflation, ef) #diminshing effect
#what about uncertainty?
ef.lower = (mergedY$inflation - min(mergedY$inflation) + .01)^(coef(mod4)[2] - 1.96*coef(summary(mod4))
ef.upper = (mergedY$inflation - min(mergedY$inflation) + .01)^(coef(mod4)[2] + 1.96*coef(summary(mod4))
plot(mergedY$inflation, ef) #diminshing effect
segments(x0 = mergedY$inflation, y0 = ef.lower, x1 = mergedY$inflation, y1 = ef.upper)
```




```
library(margins)
cplot(mod4, x = 'inflation', what = 'effect') #what is the difference?
```



- What if we suspect a non-linear relationship and want to test for it?
- We'll use the Boston data set [in MASS package], for predicting the median house value (mdev), in Boston Suburbs, based on the predictor variable lstat (percentage of lower status of the population)

```
#we'll use tidyverse this time
library(tidyverse)
```

```
## Registered S3 method overwritten by 'rvest':
##   method            from
##   read_xml.response xml2
## -- Attaching packages ---------------------------------------------------------------------------
## v ggplot2 2.2.1     v purrr   0.2.5
## v tibble  2.1.3     v dplyr   0.8.4
## v tidyr   1.0.2     v stringr 1.3.1
## v readr   1.3.1     v forcats 0.3.0
## -- Conflicts -------------------------------------------------------------------------------------
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## x dplyr::select() masks MASS::select()
```

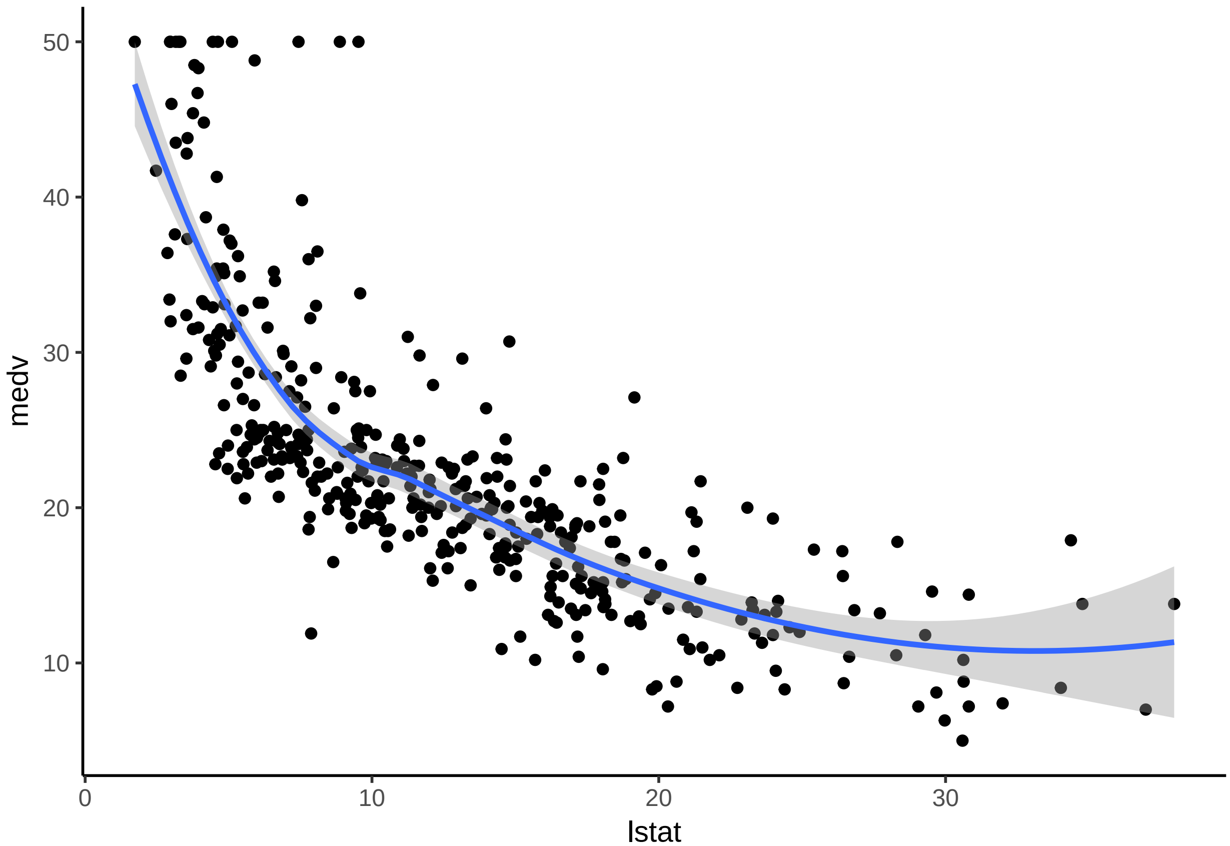
```
library(caret)
```

```
## Loading required package: lattice
##
## Attaching package: 'caret'
## The following object is masked from 'package:purrr':
##
##     lift
```

```
theme_set(theme_classic())
# Load the data
data("Boston", package = "MASS")
```

```
# Split the data into training and test set
set.seed(123)
training.samples <- Boston$medv %>%
  createDataPartition(p = 0.8, list = FALSE)
train.data  <- Boston[training.samples, ]
test.data <- Boston[-training.samples, ]
ggplot(train.data, aes(lstat, medv) ) +
  geom_point() +
  stat_smooth()
```

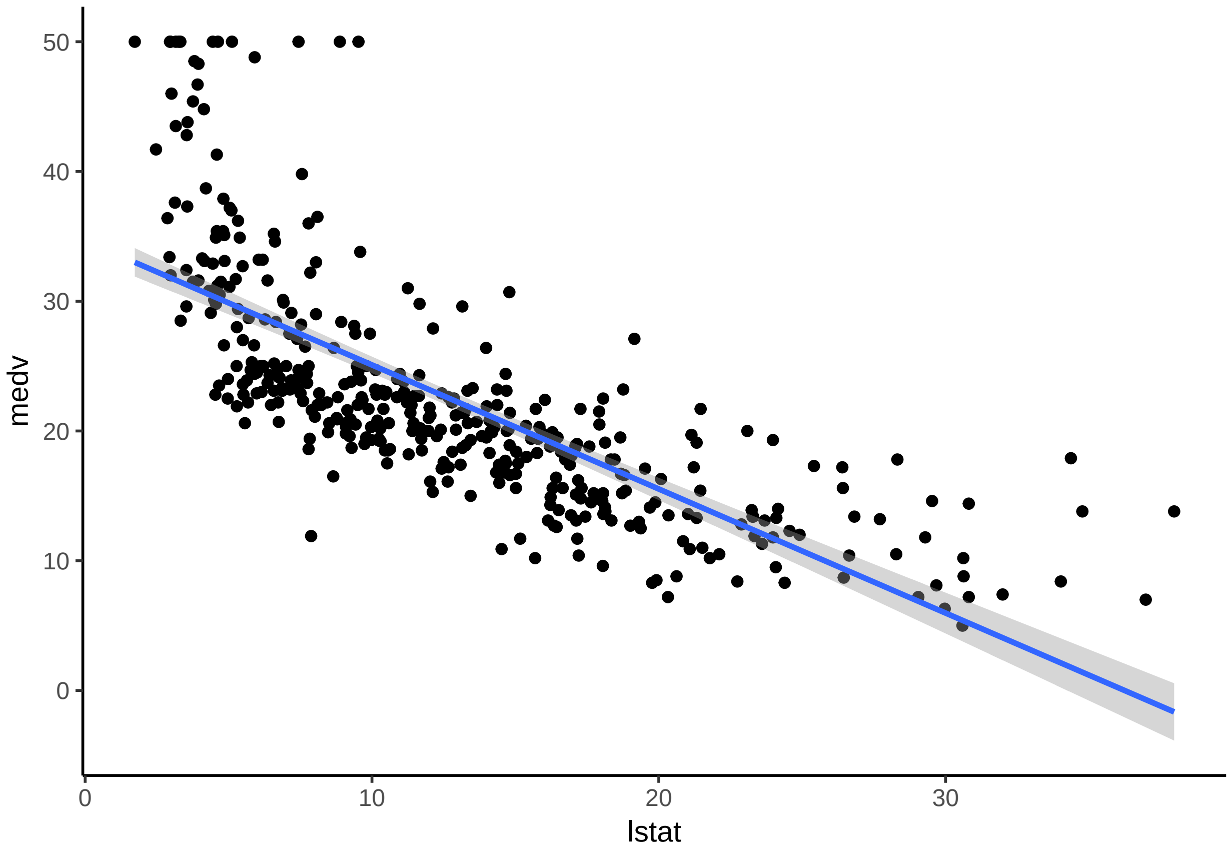
```
## `geom_smooth()` using method = 'loess'
```




```
# Build the model
model <- lm(medv ~ lstat, data = train.data)
# Make predictions
predictions <- model %>% predict(test.data)
# Model performance
data.frame(
  RMSE = RMSE(predictions, test.data$medv),
  R2 = R2(predictions, test.data$medv)
)
```

```
##       RMSE       R2
## 1 6.503817 0.513163
```

```
ggplot(train.data, aes(lstat, medv) ) +
  geom_point() +
  stat_smooth(method = lm, formula = y ~ x)
```




```r
# Squaring
model2 = lm(medv ~ poly(lstat, 2, raw = TRUE), data = train.data)
# 6 degree polynomial
lm(medv ~ poly(lstat, 6, raw = TRUE), data = train.data) %>%
  summary()
```

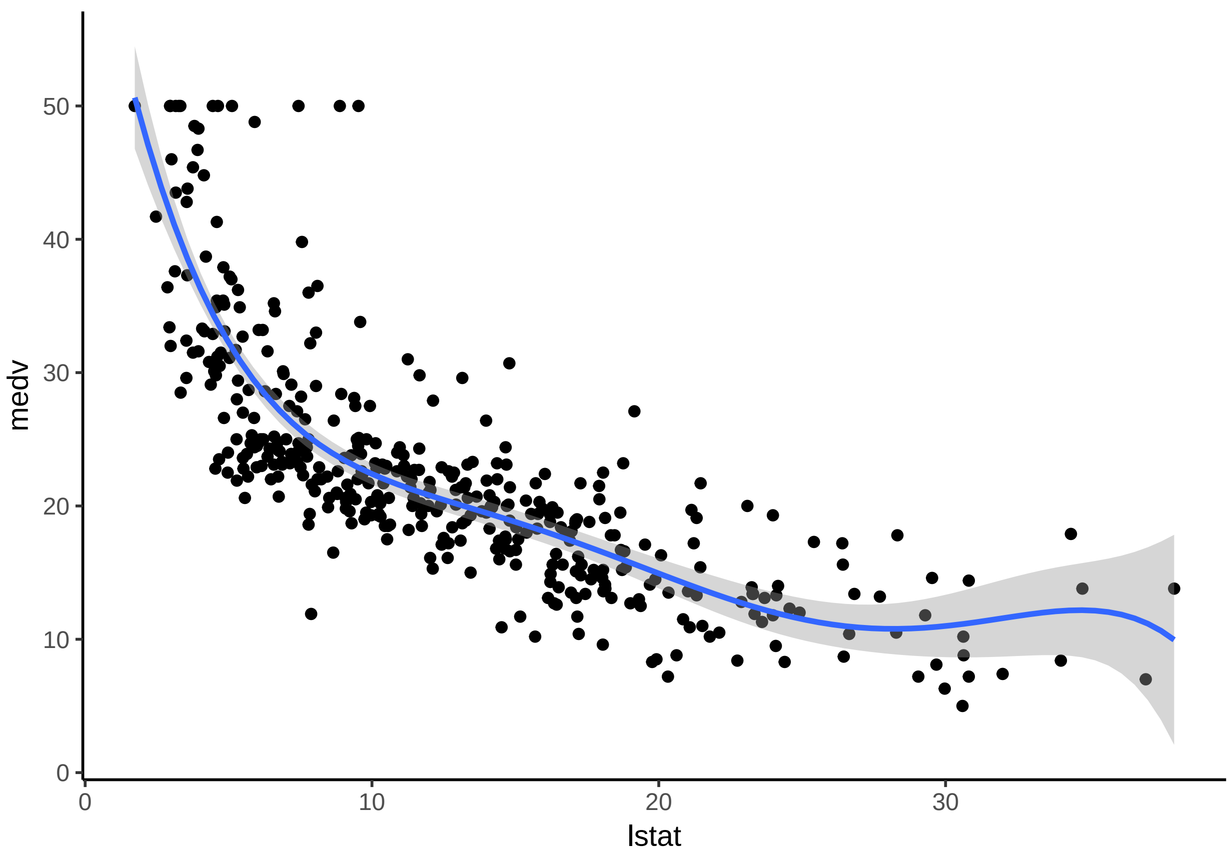
```
##
## Call:
## lm(formula = medv ~ poly(lstat, 6, raw = TRUE), data = train.data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -13.1962  -3.1527  -0.7655   2.0404  26.7661
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                  7.788e+01  6.844e+00  11.379  < 2e-16 ***
## poly(lstat, 6, raw = TRUE)1 -1.767e+01  3.569e+00  -4.952 1.08e-06 ***
## poly(lstat, 6, raw = TRUE)2  2.417e+00  6.779e-01   3.566 0.000407 ***
## poly(lstat, 6, raw = TRUE)3 -1.761e-01  6.105e-02  -2.885 0.004121 **
## poly(lstat, 6, raw = TRUE)4  6.845e-03  2.799e-03   2.446 0.014883 *
## poly(lstat, 6, raw = TRUE)5 -1.343e-04  6.290e-05  -2.136 0.033323 *
## poly(lstat, 6, raw = TRUE)6  1.047e-06  5.481e-07   1.910 0.056910 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.188 on 400 degrees of freedom
```

```
## Multiple R-squared:  0.6845, Adjusted R-squared:  0.6798
## F-statistic: 144.6 on 6 and 400 DF,  p-value: < 2.2e-16
```

```
# Drop the sixth
# Build the model
model3 <- lm(medv ~ poly(lstat, 5, raw = TRUE), data = train.data)
# Make predictions
predictions <- model3 %>% predict(test.data)
# Model performance
data.frame(
  RMSE = RMSE(predictions, test.data$medv),
  R2 = R2(predictions, test.data$medv)
)
```

```
##       RMSE        R2
## 1 5.270374 0.6829474
```

```
ggplot(train.data, aes(lstat, medv) ) +
  geom_point() +
  stat_smooth(method = lm, formula = y ~ poly(x, 5, raw = TRUE))
```

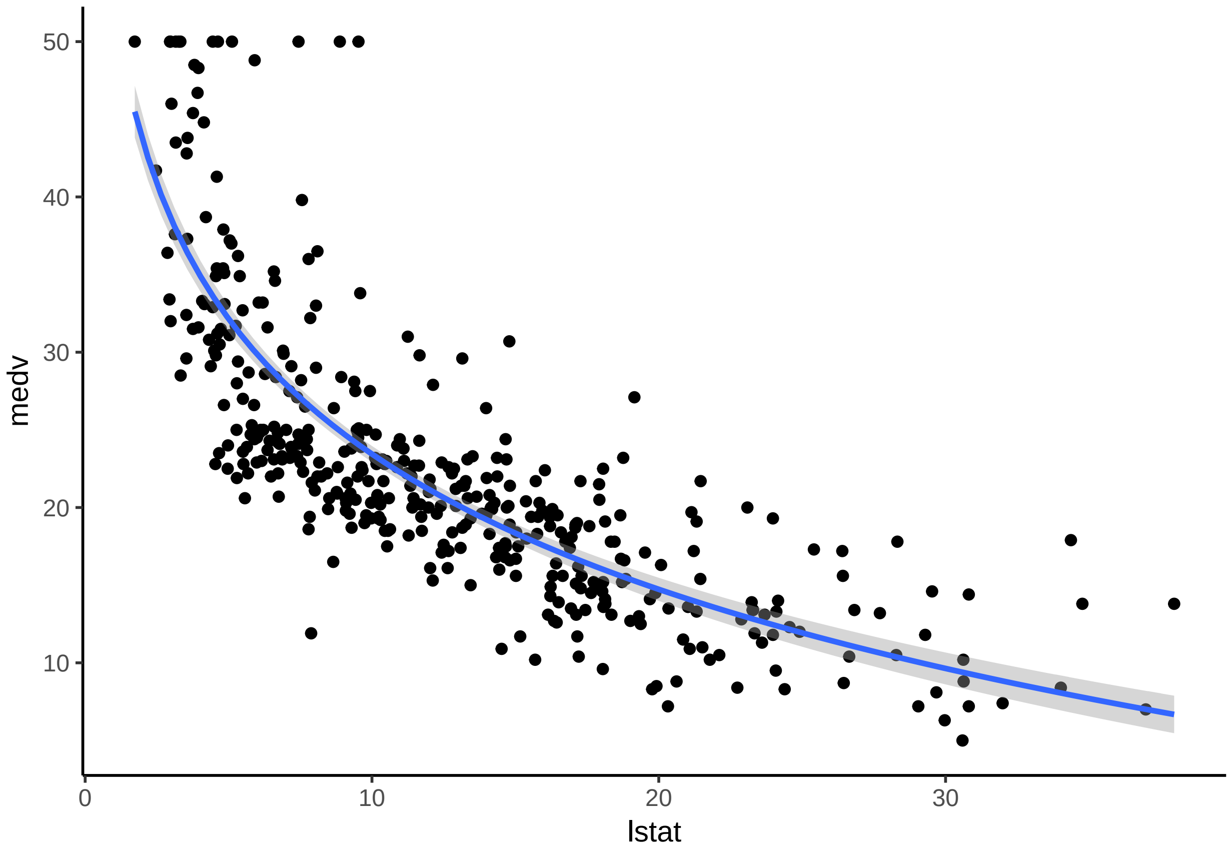



```
# Log transformation
# Build the model
model4 <- lm(medv ~ log(lstat), data = train.data)
# Make predictions
predictions <- model4 %>% predict(test.data)
# Model performance
data.frame(
  RMSE = RMSE(predictions, test.data$medv),
```

```
  R2 = R2(predictions, test.data$medv)
)
```

```
##       RMSE        R2
## 1 5.467124 0.6570091
```

```
ggplot(train.data, aes(lstat, medv) ) +
  geom_point() +
  stat_smooth(method = lm, formula = y ~ log(x))
```



- Polynomial regression only captures a certain amount of curvature in a nonlinear relationship. An alternative, and often superior, approach to modeling nonlinear relationships is to use splines
- Splines provide a way to smoothly interpolate between fixed points, called knots. Polynomial regression is computed between knots. In other words, splines are series of polynomial segments strung together, joining at knots
- You need to specify two parameters: the degree of the polynomial and the location of the knots. In our example, we'll place the knots at the lower quartile, the median quartile, and the upper quartile:

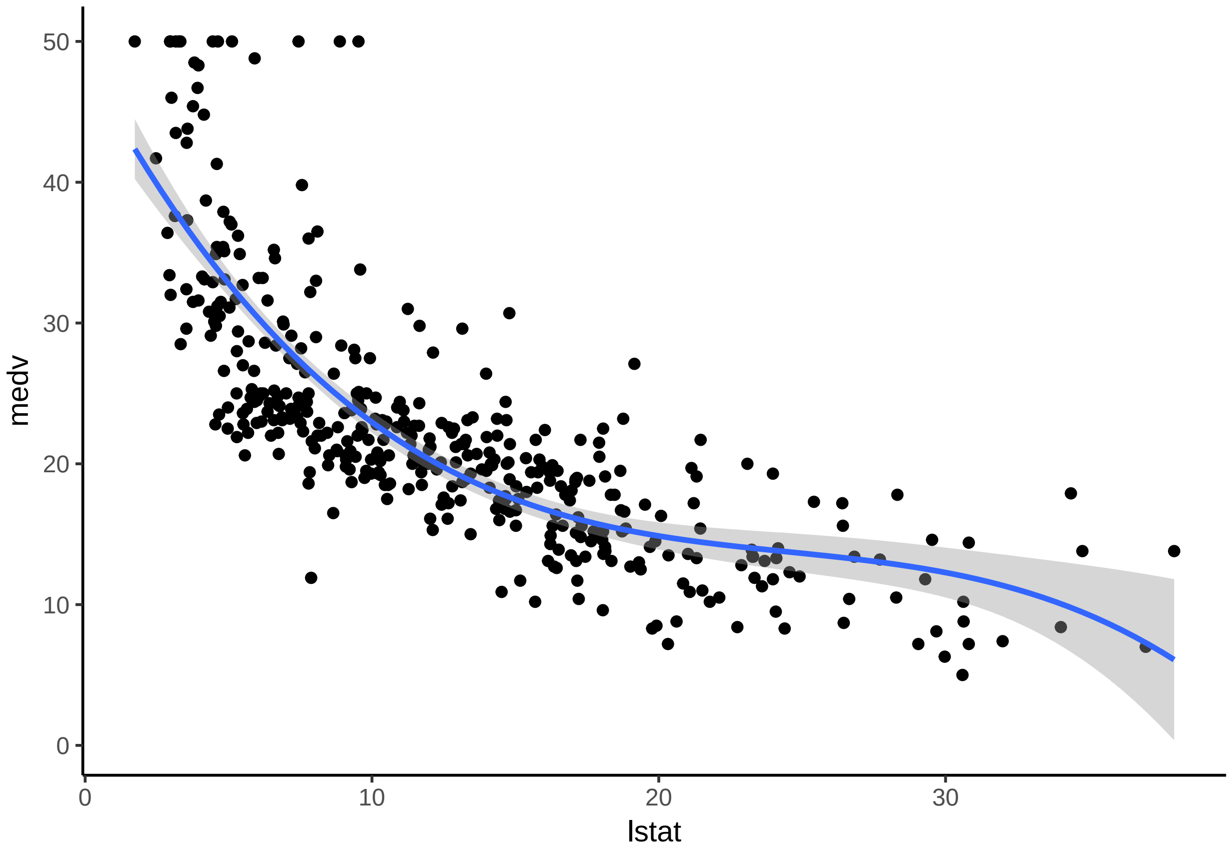
```
knots <- quantile(train.data$lstat, p = c(0.25, 0.5, 0.75))
library(splines)
# Build the model
knots <- quantile(train.data$lstat, p = c(0.25, 0.5, 0.75))
model <- lm (medv ~ bs(lstat, knots = knots), data = train.data)
# Make predictions
predictions <- model %>% predict(test.data)
# Model performance
data.frame(
  RMSE = RMSE(predictions, test.data$medv),
```

```r
  R2 = R2(predictions, test.data$medv)
)
```

```
##       RMSE        R2
## 1 5.317372 0.6786367
```

- Note that, the coefficients for a spline term are not interpretable

```r
ggplot(train.data, aes(lstat, medv) ) +
  geom_point() +
  stat_smooth(method = lm, formula = y ~ splines::bs(x, df = 3))
```



- Once you have detected a non-linear relationship in your data, the polynomial terms may not be flexible enough to capture the relationship, and spline terms require specifying the knots.
- Generalized additive models, or GAM, are a technique to automatically fit a spline regression. This can be done using the `mgvc` package:

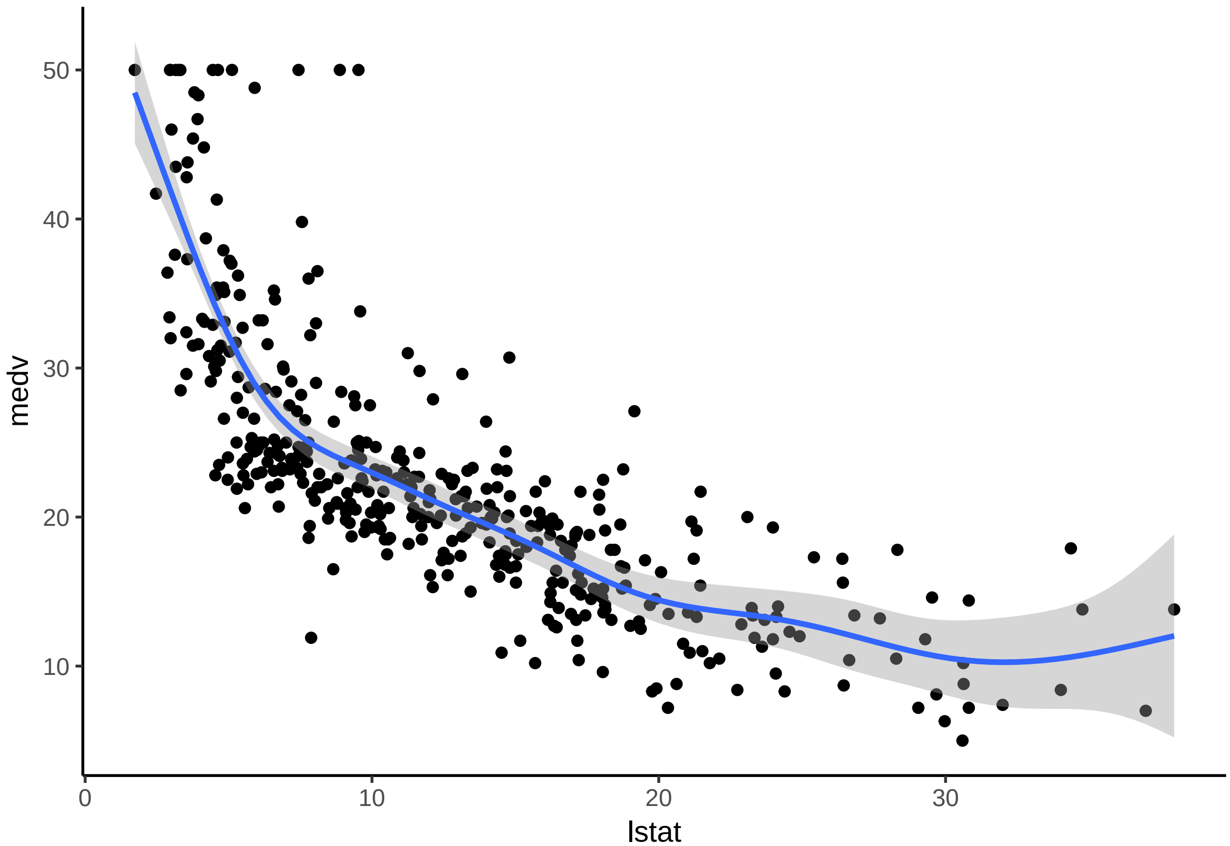
```r
library(mgcv)
```

```
## Loading required package: nlme

##
## Attaching package: 'nlme'

## The following object is masked from 'package:dplyr':
##
##     collapse

## This is mgcv 1.8-33. For overview type 'help("mgcv-package")'.
```

```r
# Build the model
model <- gam(medv ~ s(lstat), data = train.data)
# Make predictions
predictions <- model %>% predict(test.data)
# Model performance
data.frame(
  RMSE = RMSE(predictions, test.data$medv),
  R2 = R2(predictions, test.data$medv)
)
```

```
##       RMSE        R2
## 1 5.318856 0.6760512
```

```r
ggplot(train.data, aes(lstat, medv) ) +
  geom_point() +
  stat_smooth(method = gam, formula = y ~ s(x))
```



- From analyzing the RMSE and the R2 metrics of the different models, it can be seen that the polynomial regression, the spline regression and the generalized additive models outperform the linear regression model and the log transformation approaches

Endogeneity

```r
cor(mergedY[,'inflation'], summary(mod)$residuals) #we will deal with better tests later
```

```
## [1] 3.927932e-17
```

No autocorrelation in the errors

```r
library(dynlm)
```

```
## Loading required package: zoo

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```r
library(AER)
```

```
## Loading required package: car

## Loading required package: carData

##
## Attaching package: 'car'

## The following object is masked from 'package:dplyr':
##
##     recode

## The following object is masked from 'package:purrr':
##
##     some

## Loading required package: lmtest

## Loading required package: sandwich

## Loading required package: survival

##
## Attaching package: 'survival'

## The following object is masked from 'package:caret':
##
##     cluster
```

```r
data("USMacroG")
mod.dyn = dynlm(consumption ~ dpi + L(dpi), data = USMacroG)
summary(mod.dyn)
```

```
##
## Time series regression with "ts" data:
## Start = 1950(2), End = 2000(4)
##
## Call:
## dynlm(formula = consumption ~ dpi + L(dpi), data = USMacroG)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -190.02  -56.68    1.58   49.91  323.94
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -81.07959   14.50814  -5.589 7.43e-08 ***
```

```
## dpi           0.89117    0.20625   4.321 2.45e-05 ***
## L(dpi)        0.03091    0.20754   0.149    0.882
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 87.58 on 200 degrees of freedom
## Multiple R-squared:  0.9964, Adjusted R-squared:  0.9964
## F-statistic: 2.785e+04 on 2 and 200 DF,  p-value: < 2.2e-16
```

```r
durbinWatsonTest(mod.dyn)
```

```
##  lag Autocorrelation D-W Statistic p-value
##    1       0.9244708     0.0866355       0
##  Alternative hypothesis: rho != 0
```

```r
durbinWatsonTest(mod.dyn, max.lag = 4)
```

```
##  lag Autocorrelation D-W Statistic p-value
##    1       0.9244708     0.0866355       0
##    2       0.8634632     0.1342431       0
##    3       0.7947730     0.2123351       0
##    4       0.7183643     0.2914617       0
##  Alternative hypothesis: rho[lag] != 0
```

```r
library(itsadug)
```

```
## Loading required package: plotfunctions
```

```
##
## Attaching package: 'plotfunctions'
```

```
## The following object is masked from 'package:ggplot2':
##
##     alpha
```

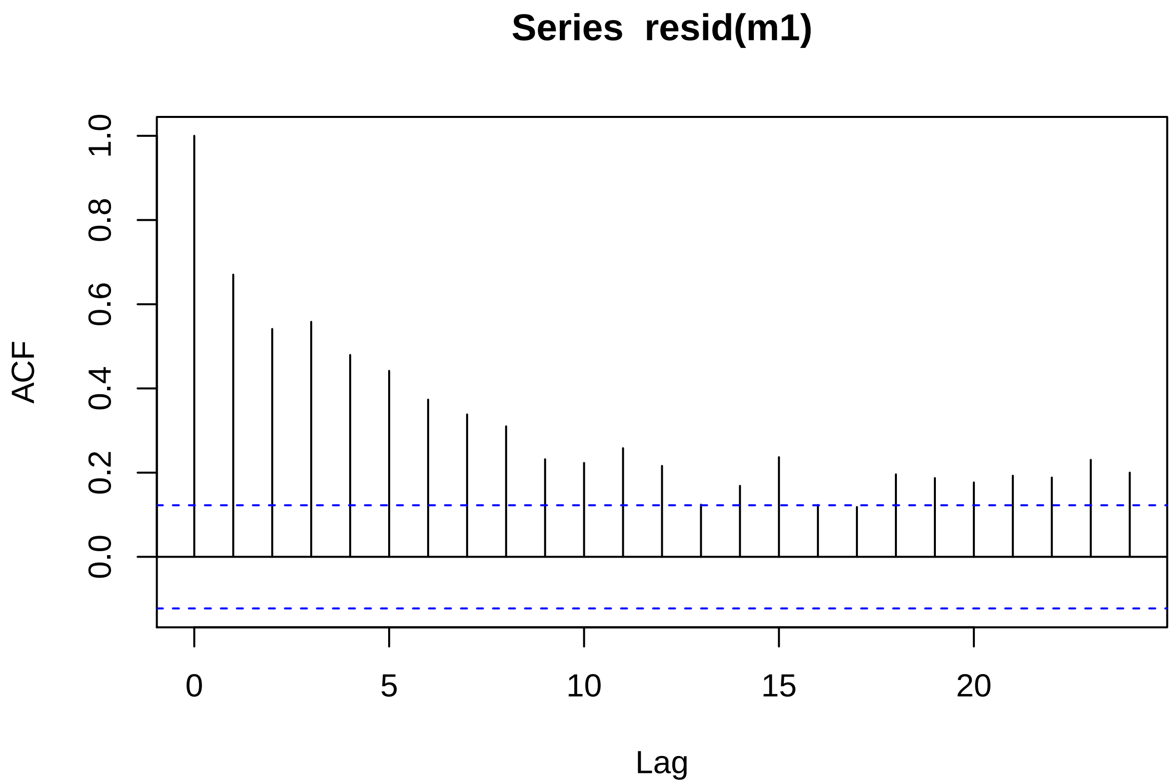
```
## Loaded package itsadug 2.3 (see 'help("itsadug")' ).
```

```r
mergedY = start_event(mergedY, column="year", event='pais', label.event="Event")
m1 <- bam(avgAA ~ te(year)+s(inflation), data = mergedY)
summary(m1)
```

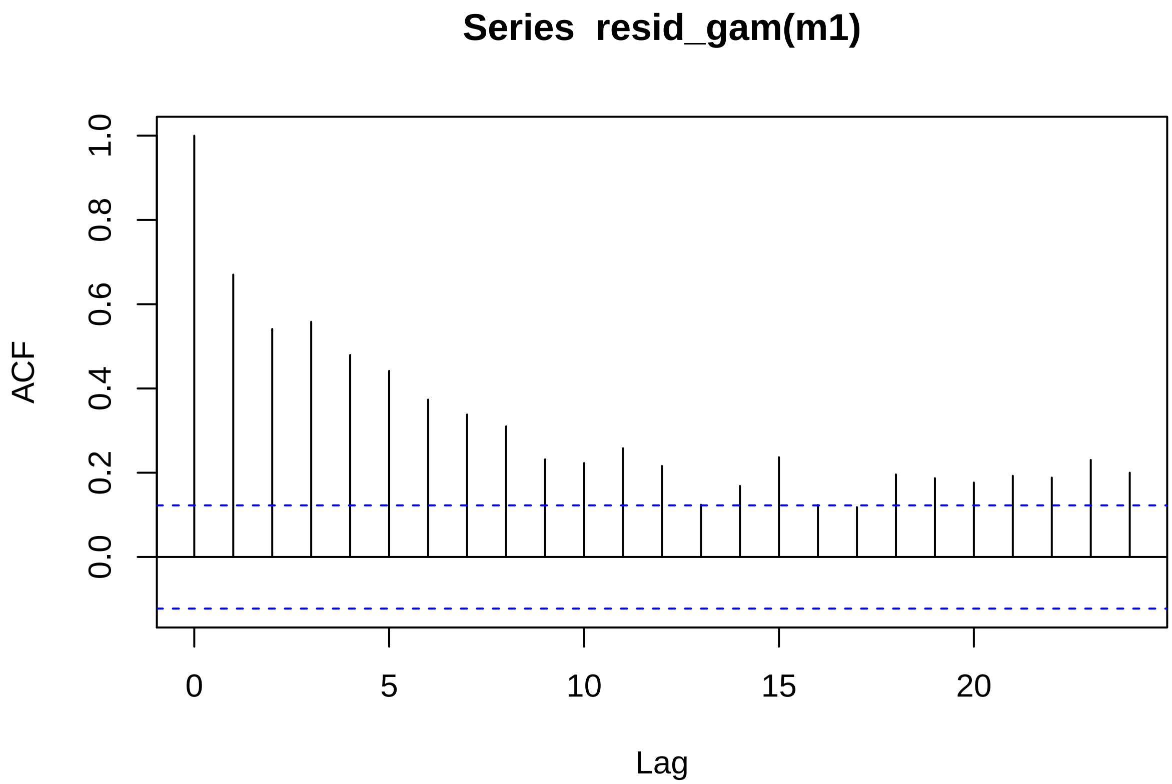
```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## avgAA ~ te(year) + s(inflation)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.98676    0.01899   157.3   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                edf Ref.df      F p-value
## te(year)     3.723  3.953 54.670  <2e-16 ***
## s(inflation) 1.000  1.000  0.059   0.808
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.478   Deviance explained = 48.7%
## fREML = 68.905  Scale est. = 0.092301  n = 256
```

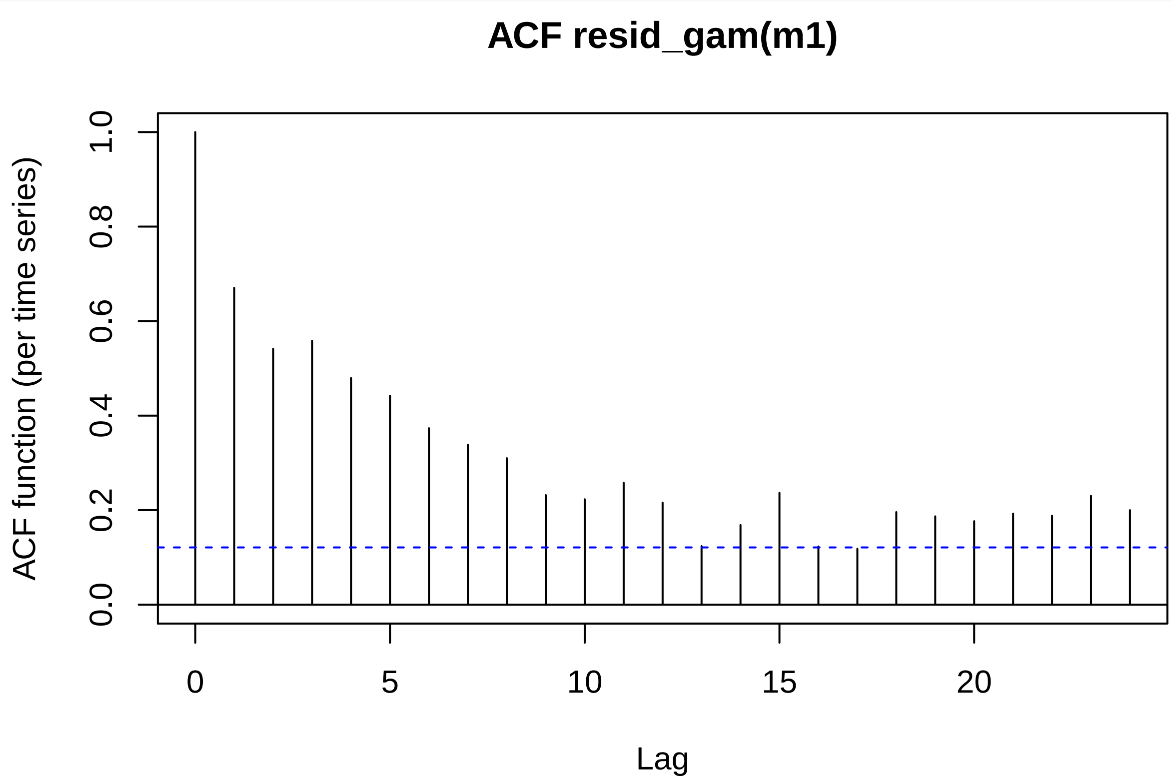
```
acf(resid(m1))
```



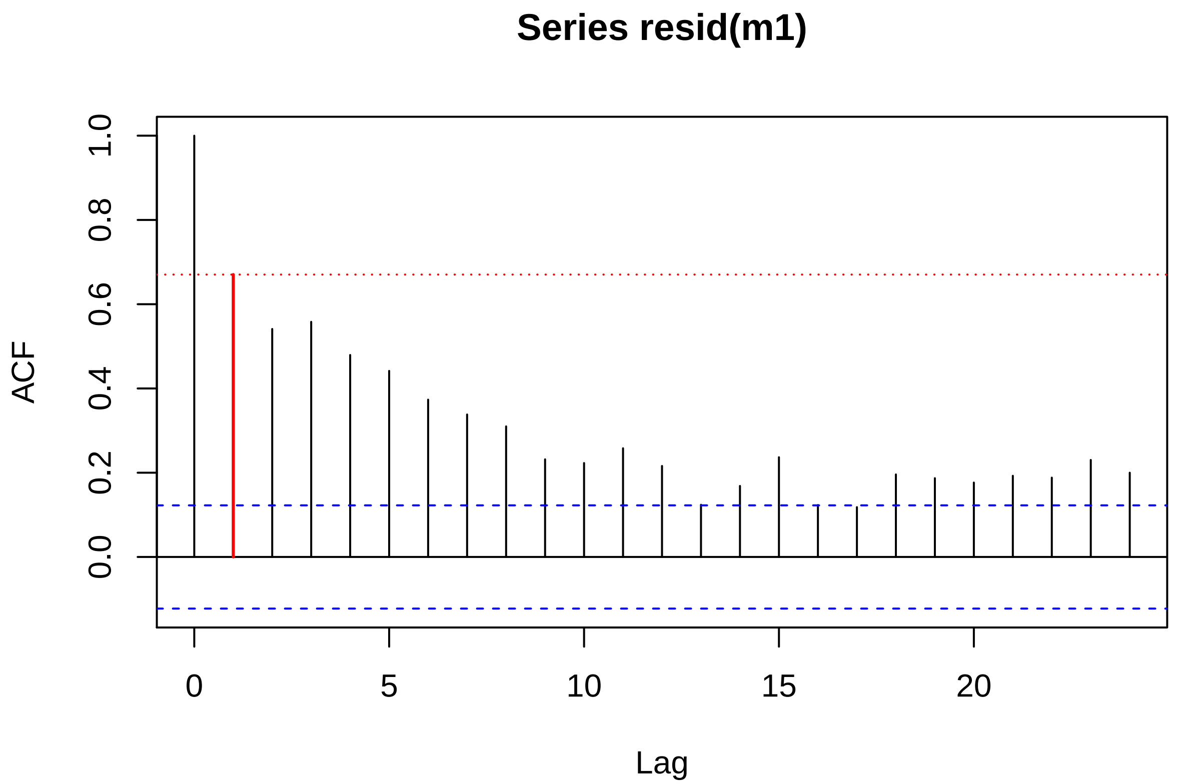

```
acf(resid_gam(m1))
```




```
acf_resid(m1)
```



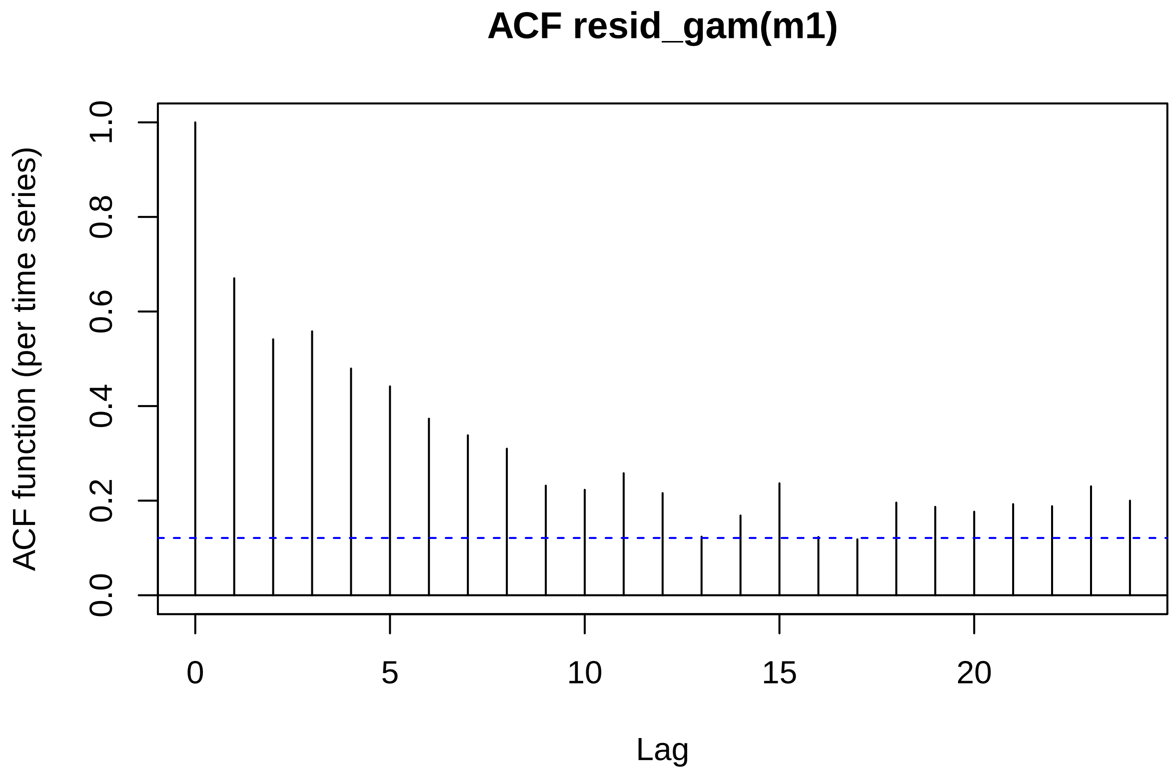

```
r1 <- start_value_rho(m1, plot=TRUE)
```



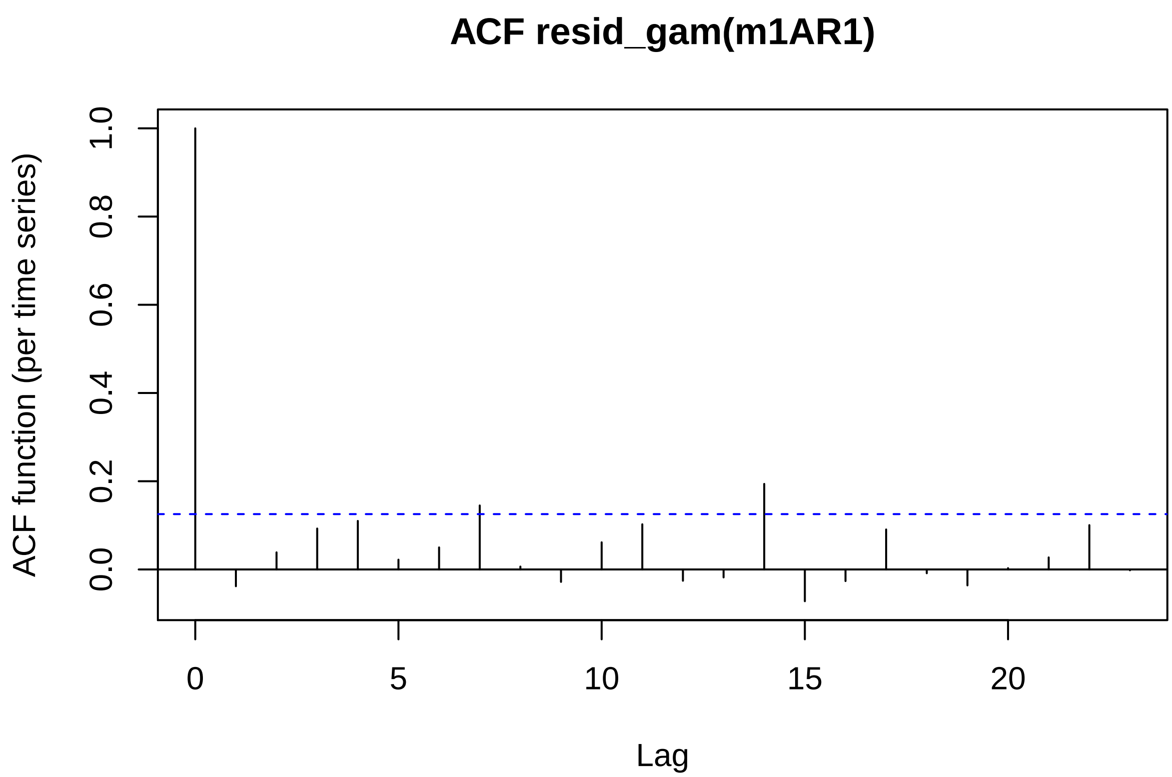

```
m1AR1 <- bam(avgAA ~ te(year)+s(inflation), data=mergedY, rho=r1, AR.start=mergedY$start.event)
summary(m1AR1)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## avgAA ~ te(year) + s(inflation)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.97442    0.03627   82.02   <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
## te(year)     3.802  3.977 34.521  <2e-16 ***
## s(inflation) 2.754  3.392  2.587  0.0481 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.466   Deviance explained = 47.9%
## fREML = -17.785  Scale est. = 0.08098   n = 256
```

```
acf_resid(m1)
```




```r
acf_resid(m1AR1)
```



No multicollinearity between variable

- For a given predictor (p), multicollinearity can assessed by computing a score called the variance inflation factor (or VIF), which measures how much the variance of a regression coefficient is inflated due to multicollinearity in the model
- The smallest possible value of VIF is one (absence of multicollinearity). As a rule of thumb, a VIF value that exceeds 5 or 10 indicates a problematic amount of collinearity
- When faced to multicollinearity, the concerned variables should be removed, since the presence of multicollinearity implies that the information that this variable provides about the response is redundant in the presence of the other variables

```
vif(mod)
```

```
## inflation   exports   imports       aid  propEmig
##  1.049810 35.429523 35.240148  1.081114  1.203017
```

```
#exports and imports are very high
mod.vif = lm(avgAA ~ inflation + aid + propEmig,
    data = mergedY)
summary(mod.vif)
```

```
##
## Call:
## lm(formula = avgAA ~ inflation + aid + propEmig, data = mergedY)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.95854 -0.23616 -0.01669  0.16685  1.25157
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.833e+00  4.245e-02  66.736  < 2e-16 ***
## inflation    8.595e-03  2.091e-03   4.110 5.36e-05 ***
## aid         -6.187e-11  2.322e-10  -0.266 0.790162
## propEmig     1.564e+00  4.371e-01   3.578 0.000415 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4019 on 252 degrees of freedom
## Multiple R-squared:  0.09633,    Adjusted R-squared:  0.08557
## F-statistic: 8.954 on 3 and 252 DF,  p-value: 1.171e-05
```