Tutorial 12: Model fit, Heteroskedasticity, and Multicollinearity

Anh Le

November 10, 2015

1. AIC and BIC (cover if have time)
2. Test of heteroskedasticity (graphical and White test)
3. Standard error robust to heteroskedasticity
4. FGLS approach to fixing heteroskedasticity
5. Generating VIF statistics related to multicollinearity
6. Generating lagged variables in R
7. Tips and tricks

- R project
- Animation in R
- Shiny webapp in R

AIC and BIC

The formula for AIC and BIC is:

$$AIC = 2k - 2ln(\hat{L}) \quad (1)$$
$$BIC = ln(n)k - 2ln(\hat{L}) \quad (2)$$

where, n is the number of data points, k is the number of parameters, L is the maximized value of the likelihood function of the model M, i.e. $L = p(x|\hat{\theta}, M)$, where $\hat{\theta}$ are the parameter values that maximize the likelihood function. (Source: Wikipedia)

In-class Quiz: While comparing two models, if one has lower AIC / BIC than the other, it is considered better. Why? (Hint: Think about how each of the two terms in AIC / BIC affects the value of AIC / BIC)

In-class Quiz: How are AIC / BIC different from R-squared?

```r
data(swiss) # Load the dataset
summary(swiss) # Get an overview of the dataset
```

```
##    Fertility      Agriculture     Examination      Education
##  Min.   :35.00   Min.   : 1.20   Min.   : 3.00   Min.   : 1.00
##  1st Qu.:64.70   1st Qu.:35.90   1st Qu.:12.00   1st Qu.: 6.00
##  Median :70.40   Median :54.10   Median :16.00   Median : 8.00
##  Mean   :70.14   Mean   :50.66   Mean   :16.49   Mean   :10.98
##  3rd Qu.:78.45   3rd Qu.:67.65   3rd Qu.:22.00   3rd Qu.:12.00
##  Max.   :92.50   Max.   :89.70   Max.   :37.00   Max.   :53.00
##     Catholic        Infant.Mortality
##  Min.   :  2.150   Min.   :10.80
##  1st Qu.:  5.195   1st Qu.:18.15
##  Median : 15.140   Median :20.00
##  Mean   : 41.144   Mean   :19.94
##  3rd Qu.: 93.125   3rd Qu.:21.70
##  Max.   :100.000   Max.   :26.60
```

```
lm1 <- lm(Fertility ~ . , data = swiss)
AIC(lm1)
```

```
## [1] 326.0716
```

```
BIC(lm1)
```

```
## [1] 339.0226
```

```
lm2 <- update(lm1, . ~ . -Examination)
AIC(lm1, lm2)
```

```
##     df      AIC
## lm1  7 326.0716
## lm2  6 325.2408
```

```
BIC(lm1, lm2)
```

```
##     df      BIC
## lm1  7 339.0226
## lm2  6 336.3417
```

Note how the AIC / BIC punishes models with more parameters. Model 1 is considered worse even though it has more parameters.

```
summary(lm1)$r.squared
```

```
## [1] 0.706735
```

```
summary(lm2)$r.squared
```

```
## [1] 0.6993476
```

On the other hand, R squared will always favor models with more parameters.

Heteroskedasticity test

Generate heteroskedastic data

For example, we have the following data generating process (DGP):

$$y = 2X + u \tag{3}$$
$$X = N(0,1) \tag{4}$$
$$u \sim N(0, |x|) \qquad \text{Note how } V(u) = |x|\text{, instead of being constant} \tag{5}$$

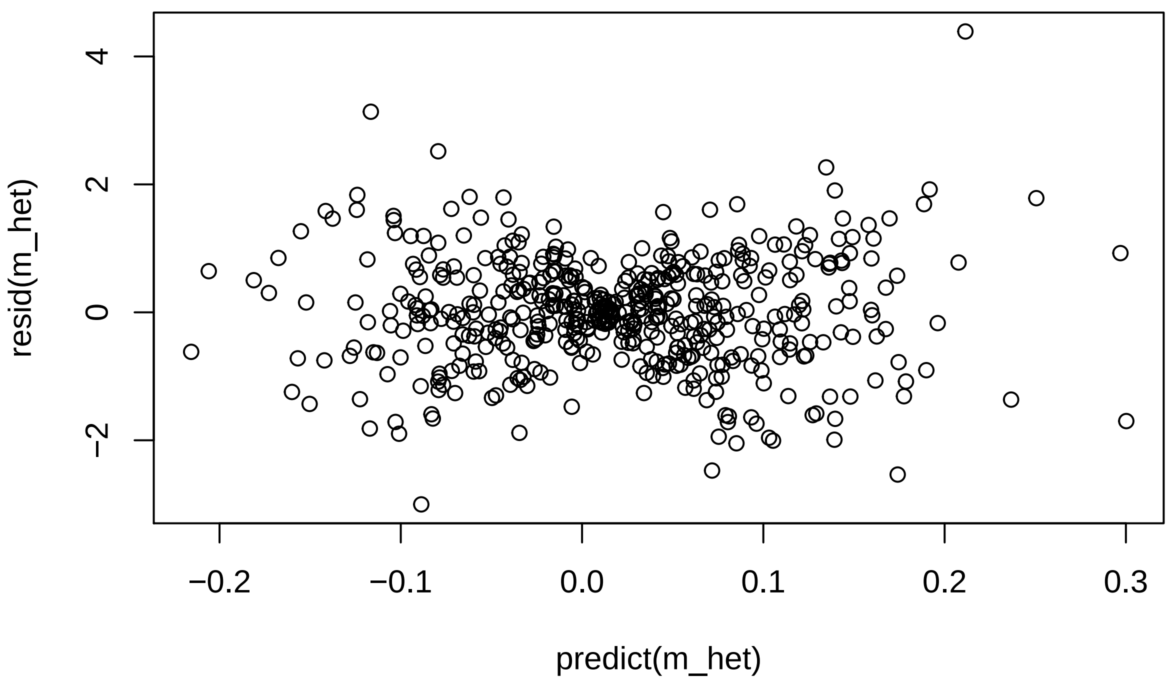
I implement that DGP in R as follows:

```
x <- rnorm(500)
u <- rnorm(500, sd = sqrt(abs(x)))
y <- 0.01 * x + u
```

Test for Heteroskedasticity:

Visual test

```
m_het <- lm(y ~ x)
plot(resid(m_het) ~ predict(m_het))
```



Breush-Pagan and White test

```
library(AER) # For the tests
```

```
## Breusch-Pagan test (p. 113)
bptest(m_het, varformula = ~ x)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  m_het
## BP = 5.3532, df = 1, p-value = 0.02068
```

```
## White test
bptest(m_het, ~ x + I(x^2))
```

```
##
##  studentized Breusch-Pagan test
##
## data:  m_het
## BP = 51.967, df = 2, p-value = 5.195e-12
```

Note how the BP test doesn't pick up the heteroskedasticity, but the White test does.

> The defaultBreusch-Pagan test specified by is a test for linear forms of heteroskedasticity, e.g. as $\hat{y}$ goes up, the error variancesgo up. In this default form, the test does not work well for non-linear forms of heteroskedasticity, such as the hourglass shape we saw before (where error variances got larger as X got more extreme in either direction). The default test also has problems when the errors are not normally distributed
>
> Part of the reason the test is more general is because it adds a lot of terms to test for more types of heteroskedasticity. For example, adding the squares of regressors helps to detect nonlinearities such as the hourglass shape. In a large data set with many explanatory variables, this may make the test difficult to calculate. Also, the addition of all these terms may make the test less powerful in those situations when a simpler test like the default Breusch-Pagan would be appropriate, i.e. adding a bunch of extraneous terms may make the test less likely to produce a significant result than a less general test would. (Source)

Get the correct standard error

White HC standard errors (sandwich standard error)

```
vcovHC(m_het, type = "HC") # from package sandwich, loaded by AER
```

```
##               (Intercept)             x
## (Intercept)  0.0014298558 -0.0001715223
## x           -0.0001715223  0.0027554777
```

Hypothesis test with White standard error (more precisely, with the heteroskedasticity-consistent estimation of the cov matrix, calculated above). Notice how the coefficients are basically the same as regular OLS. Only the standard error is different.

```
coeftest(m_het, vcov = vcovHC(m_het, type = "HC"))
```

```
##
## t test of coefficients:
##
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.015360   0.037813  0.4062   0.6848
## x           -0.082134   0.052493 -1.5647   0.1183
```

```r
summary(m_het)
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.0018 -0.4855  0.0039  0.5514  4.3899
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.01536    0.03819   0.402    0.688
## x           -0.08213    0.03842  -2.138    0.033 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8531 on 498 degrees of freedom
## Multiple R-squared:  0.009094,   Adjusted R-squared:  0.007104
## F-statistic:  4.57 on 1 and 498 DF,  p-value: 0.03302
```

Foonote: Etymology of the "sandwich" estimator of the standard error

Formula for standard error is:

$$V(\hat{\beta}_{OLS}|X) = (X'X)^{-1}X'V(\epsilon|X)X(X'X)^{-1}$$

, which reduces to $\sigma^2(X'X)^{-1}$ only in the case of homoskedasticity.

When there's heteroskedasticity, the estimate is (as done in Stata) (notice the "bread" ($(X'X)^{-1}$) and the "meat" ($X'V(\epsilon|X)X$) of the sandwich):

$$V(\hat{\beta}_{OLS}|X) = (X'X)^{-1}X'V(\epsilon|X)X(X'X)^{-1} \tag{6}$$

$$\hat{V}(\hat{\beta}_{OLS}|X) = (X'X)^{-1}X'\begin{pmatrix} \hat{\epsilon}_1^2 & 0 & \dots & 0 \\ 0 & \hat{\epsilon}_2^2 & \dots & 0 \\ \vdots & \vdots & \dots & \vdots \\ 0 & 0 & \dots & \hat{\epsilon}_n^2 \end{pmatrix} X(X'X)^{-1} \tag{7}$$

$$\hat{V}(\hat{\beta}_{OLS}|X) = \frac{N}{N-K}(X'X)^{-1}\sum_{i=1}^{N}\{X_iX_i'\hat{\epsilon}_i^2\}(X'X)^{-1} \qquad \text{Similar to slide 62 of your lecture notes} \tag{8}$$

The constant is added since we are estimating the sample variance of the error. In practice, $N >> K$, so it doesn't matter.

FGLS approach to fixing heteroskedasticity

Math footnote:

- Case 1 (GLS): $V(\epsilon|X) = \sigma^2\Sigma(X)$, where $\Sigma(X)$ is known. In this case, β_{OLS} is inefficient, but $\beta_{GLS} = (X'\Sigma(X)^{-1}X)^{-1}X'\Sigma(X)^{-1}y$ is BLUE again (intuition: we weigh the observation so that ones with small variance has higher weights, and pay less attention to the observation with high variance. Thus the WLS estimate is more efficient than OLS). To do this weighting, we run linear regression with weight $= \Sigma(X)^{-1}$. Equivalently, divide y and all x's by $\frac{1}{\sqrt{\hat{\sigma}}}$).

This is called *generalized* least square because OLS is a special case where $\Sigma(X) = I$

- Case 2 (FGLS): More realistically, we don't actually know $\Sigma(X)$ (and it has more parameters than our number of observations). So we must impose some structure so that it's *feasible* to estimate it. For example, in the lecture, we assume that $\Sigma(X)$ is some linear combinations of X's (but not of X^2, or some non-linear combinations)

Some Resources on FGLS

Following this assumption, we estimate $V(\epsilon|X)$ by regressing the residual squared, i.e. $\hat{\epsilon}^2$ on all X, then get the predicted value of this auxiliary regression.

```r
LinearModel.OLS <- lm(y ~ x)

# Auxiliary regression of residual squared against all X (there's only 1 x in this case)
auxiliary.FGLS <- lm(I(resid(LinearModel.OLS)^2) ~ x)

# The weight is the predicted value of the auxiliary regression
w <- predict(auxiliary.FGLS)

# Use the w in weighted least square regression
LinearModel.WLS <- lm(y ~ x, weights = 1/w)

summary(LinearModel.WLS)
```

```
##
## Call:
## lm(formula = y ~ x, weights = 1/w)
##
## Weighted Residuals:
##     Min      1Q  Median      3Q     Max
## -4.1256 -0.5679  0.0022  0.6561  4.4678
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.01589    0.03849   0.413   0.6799
## x           -0.06972    0.03788  -1.840   0.0663 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.019 on 498 degrees of freedom
## Multiple R-squared:  0.006756,   Adjusted R-squared:  0.004762
## F-statistic: 3.387 on 1 and 498 DF,  p-value: 0.06629
```

Generating VIF statistics related to multicollinearity

The formula for VIF:

$$VIF_k = \frac{1}{1 - R_k^2}$$

where R_k^2 is the R-squared of the regression of X_k against all the other X. If this R-squared is larged, it means that X_k is highly correlated with all the other X's.

Let's generate some highly correlated data to check. The classic case of high multicollnearity is age and experience.

```
age <- rnorm(100, mean = 32, sd = 10)
# Experience is age - 16 years of high school / college + error term
# i.e. some people go for grad school, some drop out early
experience <- age - 16 + rnorm(100)
salary <- 2 * age + 3 * experience + rnorm(100)
```

Test for VIF

```
m_vif <- lm(salary ~ age + experience)
library(car)
vif(m_vif)
```

```
##        age experience
##   83.37706   83.37706
```

Interpret VIF: The variance of β_{age} is inflated by 83.377059 times due to the inclusion of experience in the regression.

Question: How do we deal with multicollinearity?

Generating lagged variables in R

```
df <- data.frame(year = 2000:2005, gdp = rnorm(6))
df$gdp_lag1 <- c(NA, df$gdp[1:(length(df$gdp) - 1)])
df$gdp_lag2 <- c(rep(NA, 2), df$gdp[1:(length(df$gdp) - 2)])
df
```

```
##   year         gdp   gdp_lag1   gdp_lag2
## 1 2000  0.44745937         NA         NA
## 2 2001 -0.38471307  0.4474594         NA
## 3 2002  0.86346071 -0.3847131  0.4474594
## 4 2003  0.79693477  0.8634607 -0.3847131
## 5 2004 -2.32568115  0.7969348  0.8634607
## 6 2005  0.09578791 -2.3256812  0.7969348
```

That looks like an awful lot of typing for basically the same pattern, so let's make a function. (Rule of thumb in programming: If you are typing something more than 3 times, you should write a function)

In-class Quiz: The first step of writing a function is to think about: what are the inputs and outputs of this function?

```
lag <- function(x, lag_period) {
  return(c(rep(NA, lag_period), x[1:(length(x) - lag_period)]))
}

# Try it
df <- data.frame(year = 2000:2005, gdp = rnorm(6))
df$gdp_lag1 <- lag(df$gdp, 1)
df$gdp_lag2 <- lag(df$gdp, 2)
df
```

```
##   year         gdp    gdp_lag1    gdp_lag2
## 1 2000 -0.87519246          NA          NA
## 2 2001 -0.01975433 -0.87519246          NA
## 3 2002 -1.27236784 -0.01975433 -0.87519246
## 4 2003  0.26114835 -1.27236784 -0.01975433
## 5 2004  0.66965689  0.26114835 -1.27236784
## 6 2005 -0.71141413  0.66965689  0.26114835
```

Testing for autocorrelation

Following is the example of autocorrelation. Note how the $e(t)$ and $e(t-1)$ are correlated. (This is an example of AR(1) process, as mentioned on slide 13 of lecture notes.)

$$e(t) = ae(t-1) + v(t) \tag{9}$$
$$Y(t) = X(t) + e(t) \tag{10}$$

We now model this DGP in R.

```
T <- 100 # Num of time periods

# Generate autocorrelated e
e <- vector(mode = 'numeric', length = T)
e[1] <- rnorm(1)
for (t in 2:T) {
  e[t] <- 0.8 * e[t - 1] + rnorm(1)
}

X <- rnorm(T)
Y <- X + e
```

Visual

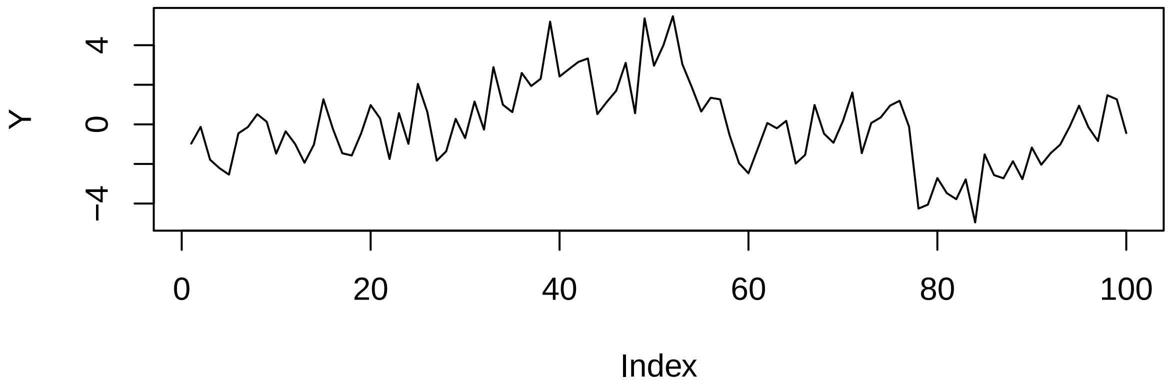
Let's take a look at Y

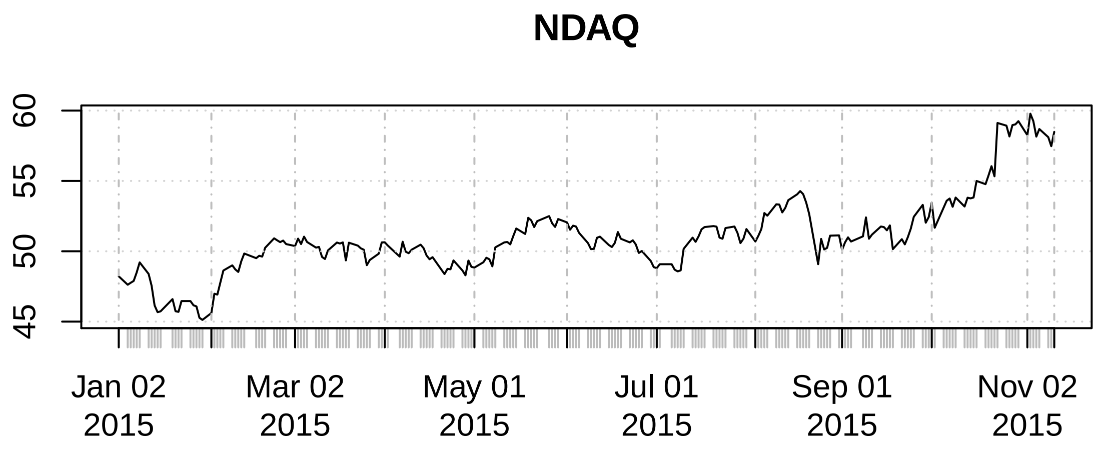
```
par(mfrow = c(2, 1))
plot(Y, type = 'l')
library(quantmod)
getSymbols(Symbols = 'NDAQ', src = 'yahoo', from = '2015-01-01')
```

```
## [1] "NDAQ"
```

```r
plot(NDAQ)
```

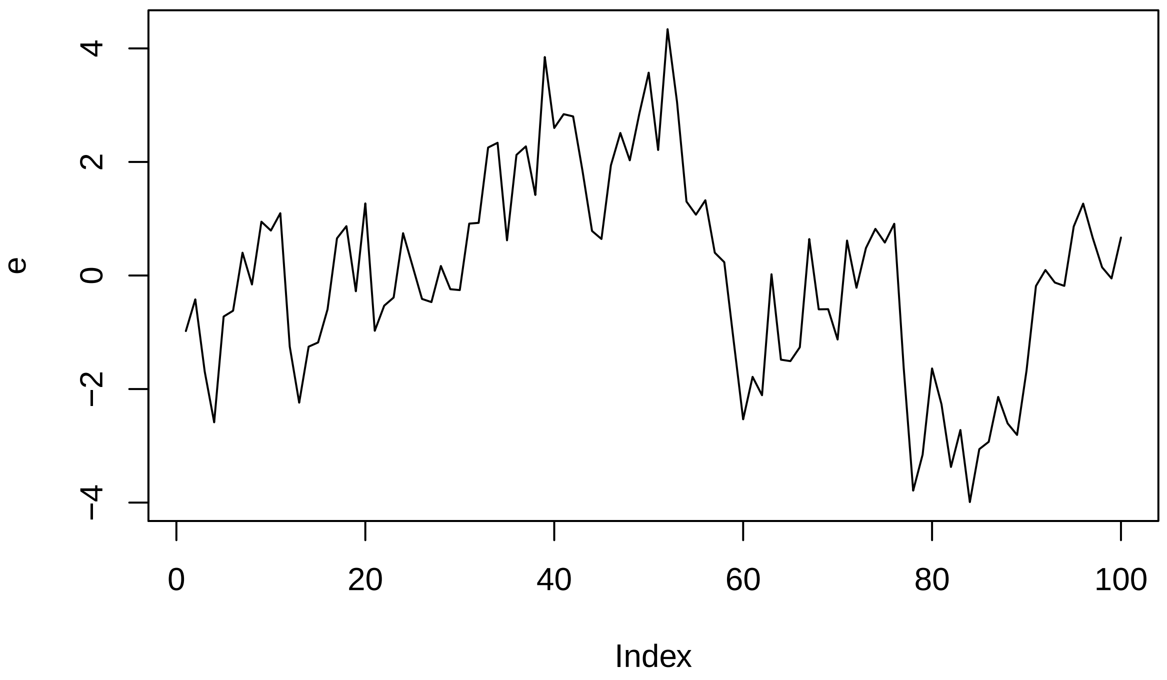
```
## Warning in plot.xts(NDAQ): only the univariate series will be plotted
```



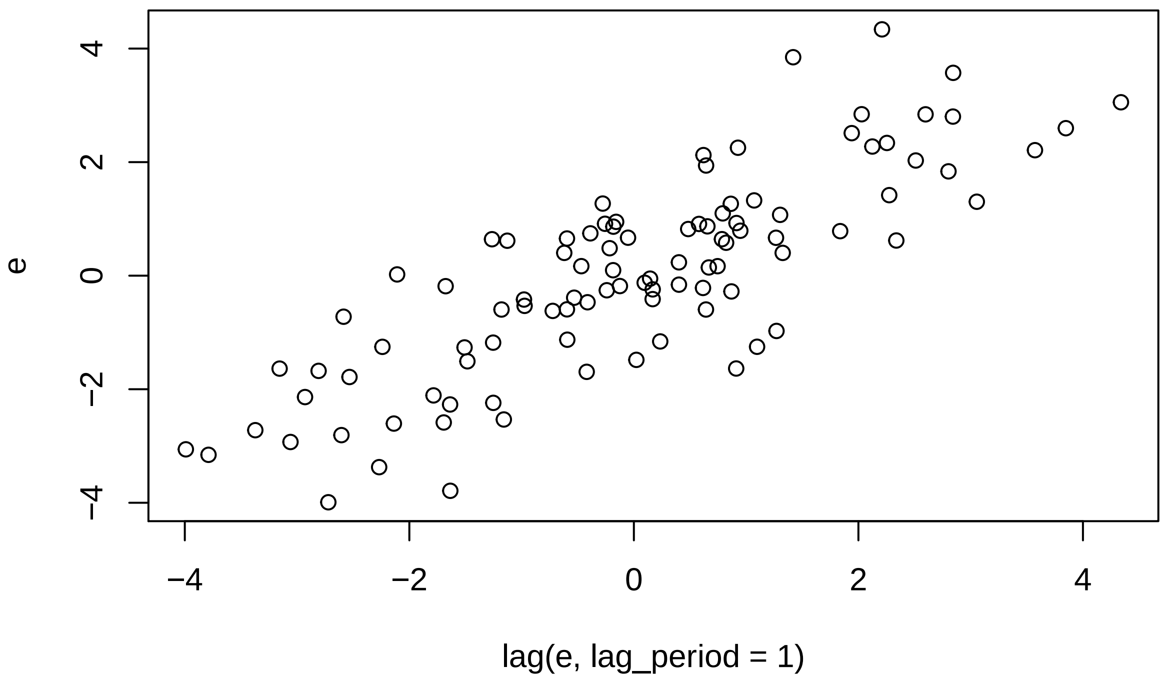


Plot residuals against time and against lagged residuals

```r
m_auto <- lm(Y ~ X)
e <- resid(m_auto)
plot(e, type = 'l')
```



```r
plot(lag(e, lag_period = 1), e)
```



Hypothesis testing

Regress residuals against X and lagged residuals, and then doing an F test for joint significance in the lagged residuals.

```r
lag_e <- lag(e, lag_period = 1)

# Reg residual against X and lagged residuals
m_autotest <- lm(e ~ X + lag_e)

# Doing an F test
library(car) # to run F-test
linearHypothesis(m_autotest, c("lag_e"))
```

```
## Linear hypothesis test
##
## Hypothesis:
## lag_e = 0
##
## Model 1: restricted model
## Model 2: e ~ X + lag_e
##
##   Res.Df     RSS Df Sum of Sq      F    Pr(>F)
## 1     97 307.166
## 2     96  98.408  1    208.76 203.65 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```