Linear model: Ficus plants

We have the height, H, of the ficus plants and the number of days since it has been planted, *Days*. This data set constitutes an example of data such that requires to transform the response variable by means of the logarithm, and to consider the regression $\log(H) \sim \alpha + \beta \cdot Days$. But as you will see, the model does not verify the *homoscedasticity* hypothesis. Moreover, as you will see, the non-linear regression model $H \sim e^{\alpha+\beta} \cdot Days$ does not verify the *homoscedasticity* property neither (at least it is not clear). Thus the analysis of this model points out the need of using generalized linear models and that is what we are going to do. The significance level is set to be equal to $alpha = 0.05$.

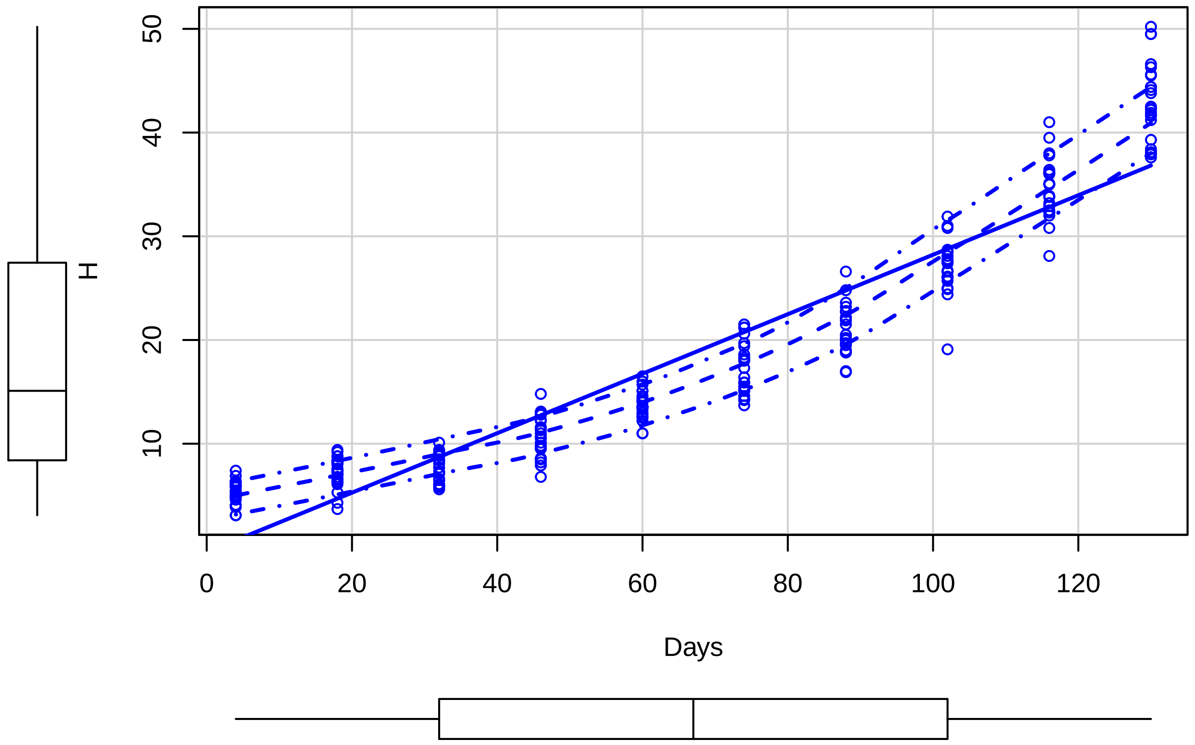
```
library(car)
```

```
## Loading required package: carData
```

```
library(emmeans)
dd<-read.csv2("Ficusdata.csv")
# head(dd)
```

Take a glimpse on the descriptive:

```
dd$FDays<-as.factor(dd$Days)
sp(H~Days, dd)
```



But before starting with GLM models, we shall we have a sneak peek on the usual linear models and see where it doesn't work.

First model: the regression line H with respect to the variable *Days*. Fit the data, compute the parameter estimations and interpret them.

$$H = \beta_0 + \beta_1(Days)$$

```
summary(model1<-lm(H~Days, dd))
```

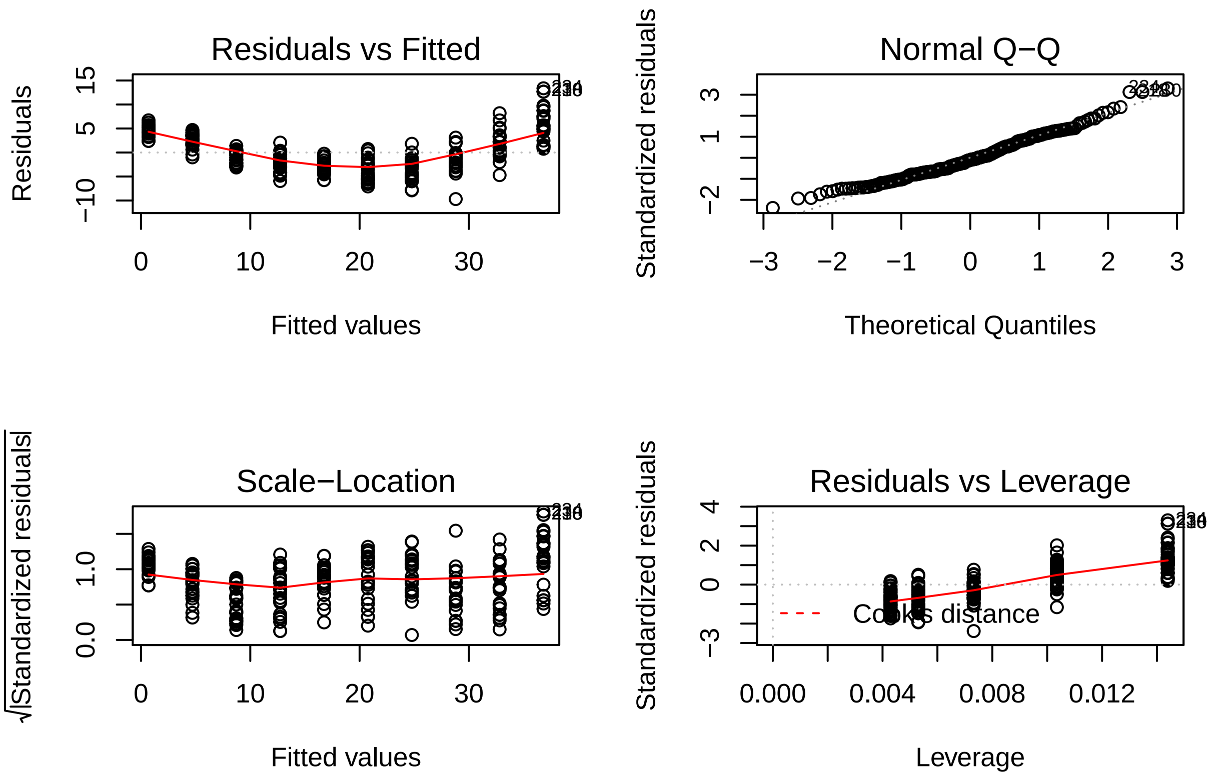
```
##
```

```
## Call:
## lm(formula = H ~ Days, data = dd)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.6952 -2.8803 -0.4206  2.9469 13.3749
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.456364   0.511898  -0.892    0.374
## Days         0.286780   0.006551  43.777   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.081 on 238 degrees of freedom
## Multiple R-squared:  0.8895, Adjusted R-squared:  0.8891
## F-statistic:  1916 on 1 and 238 DF,  p-value: < 2.2e-16
```

The Omnibus test is significative, so the model explains the variability in the data, and the Days parameter is not zero (it is significative). Also, note the R squared is quite high: 88.95% of the variability is captured by our model.

$$H = -0.456364 + 0.286780(Days)$$

```r
oldpar<-par(mfrow=c(2,2))
plot(model1, ask=F)
```



```r
par(oldpar)
```

1. Residuals vs Fitted

This plot shows if residuals have non-linear patterns. There could be a non-linear relationship between predictor variables and an outcome variable and the pattern could show up in this plot if the model doesn't capture the non-linear relationship. If you find equally spread residuals around a horizontal line without distinct patterns, that is a good indication you don't have non-linear relationships.

What do you think? I see a parabola, where the non-linear relationship was not explained by the model and was left out in the residuals.

2. Normal Q-Q

This plot shows if teh residuals are normally distributed. Do residuals follow a straight line well or do they deviate severely? It's good if residuals are lined well on the straight dashed line. I would not be concerned, but I see some observations that look a little off, observations numbered as 218, 230 and 234.

3. Scale-Location

It's also called *Spread-Location plot*. This plot shows if residuals are spread equally along the ranges of predictors. This is how you can check the assumption of equal variance (*homoscedasticity*). It's good if you see a horizontal line with equally (randomly) spread points.

What do you think? I observe that the residuals appear randomly spread.

4. Residuals vs Leverage

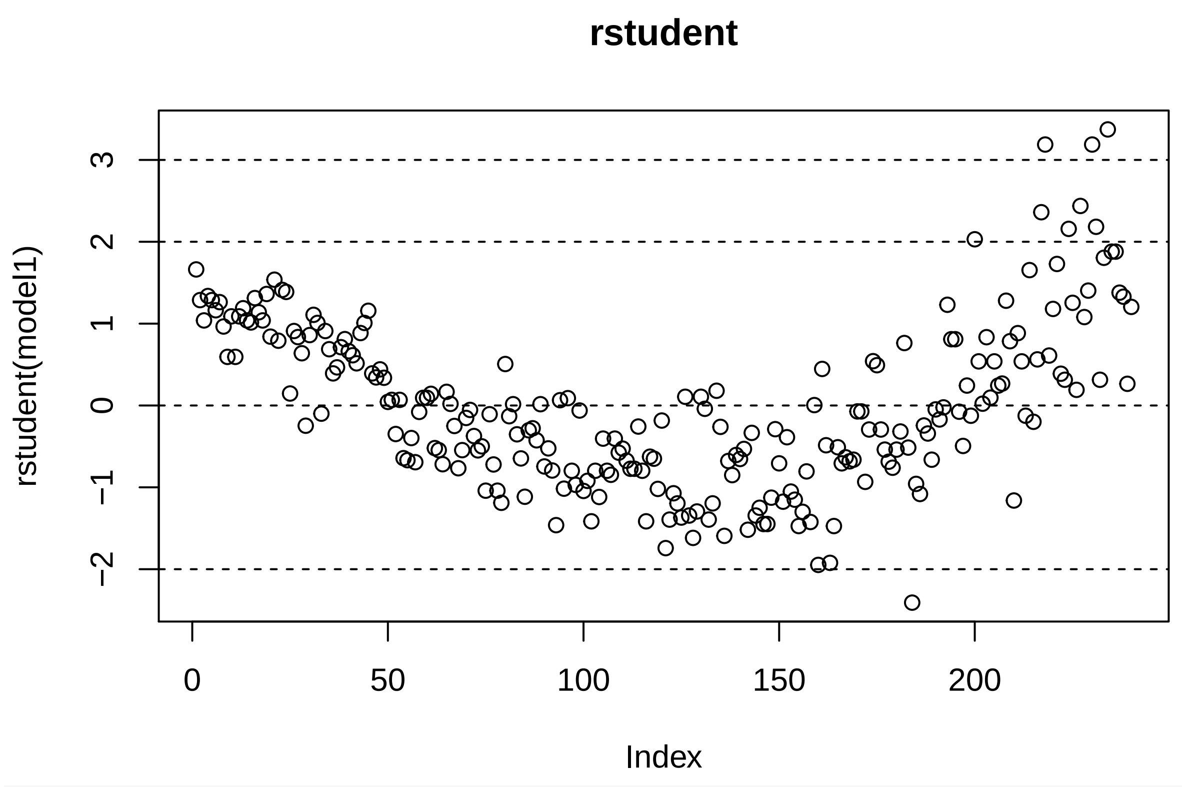
This plot helps us to find influential cases (i.e., subjects) if any. Not all outliers are influential in linear regression analysis (whatever outliers mean). Even though data have extreme values, they might not be influential to determine a regression line. That means, the results wouldn't be much different if we either include or exclude them from analysis. They follow the trend in the majority of cases and they don't really matter; they are not influential. On the other hand, some cases could be very influential even if they look to be within a reasonable range of the values. They could be extreme cases against a regression line and can alter the results if we exclude them from analysis. Another way to put it is that they don't get along with the trend in the majority of the cases.

Unlike the other plots, this time **patterns are not relevant**. We watch out for outlying values at the upper right corner or at the lower right corner. Those spots are the places where cases can be influential against a regression line. Look for cases outside of a dashed line, Cook's distance. When cases are outside of the Cook's distance (meaning they have high Cook's distance scores), the cases are influential to the regression results. The regression results will be altered if we exclude those cases.

What do you think? I see the typical look when there is no influential case, or cases. We do not even see the Cook's distance lines!

Also, let's check how many and what studentized residuals (if there are any) are greater than 3:

```r
plot(rstudent(model1),main="rstudent")
abline(h=c(-3,-2,0,2,3),lty=2)
```

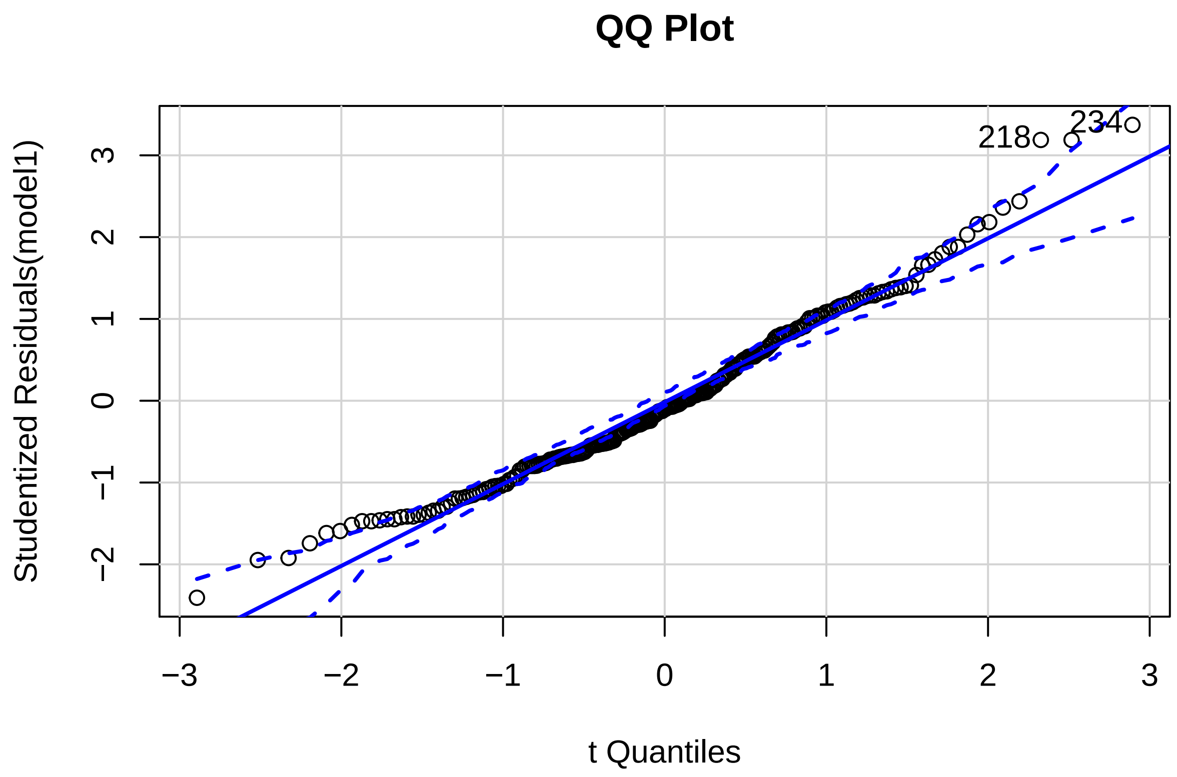


```
for (i in 1:length(rstudent(model1))) if (rstudent(model1)[i] > 3) print(rstudent(model1)[i])
```

```
##      218
## 3.188103
##      230
## 3.188103
##      234
## 3.372391
```

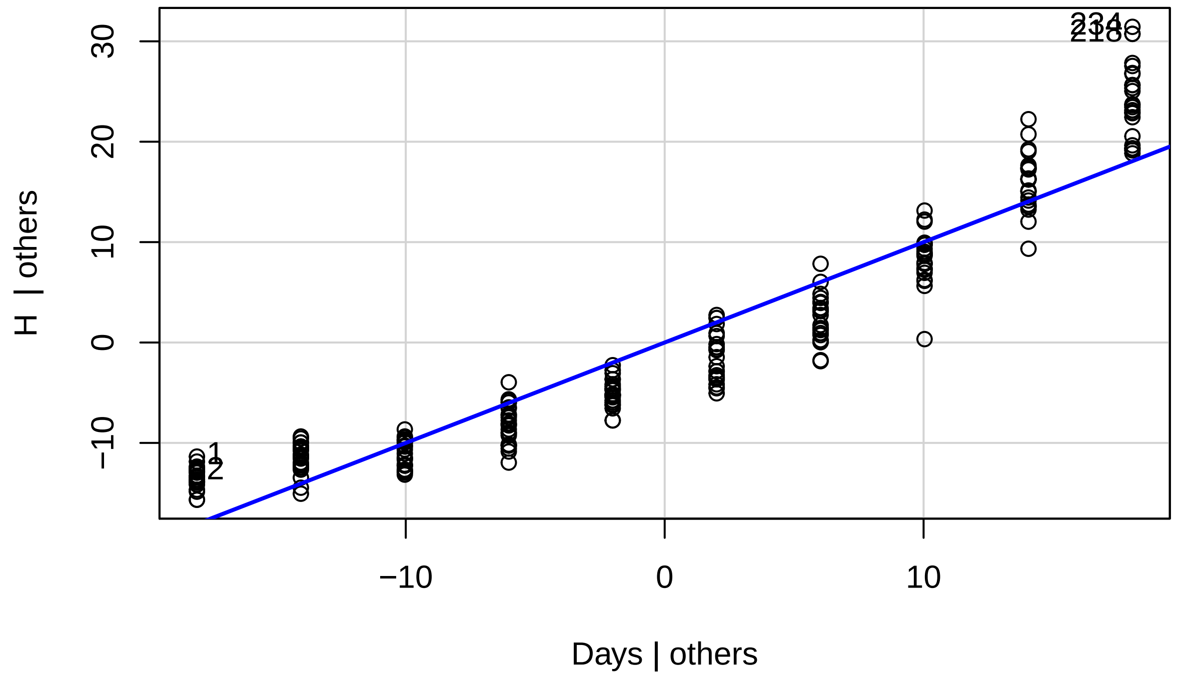
Exactly the ones we have spotted earlier in the QQ-plot!

```
# Assessing Outliers
qqPlot(model1, main="QQ Plot") #qq plot for studentized resid
```



```
## [1] 218 234
```

```
leveragePlots(model1) # leverage plots
```



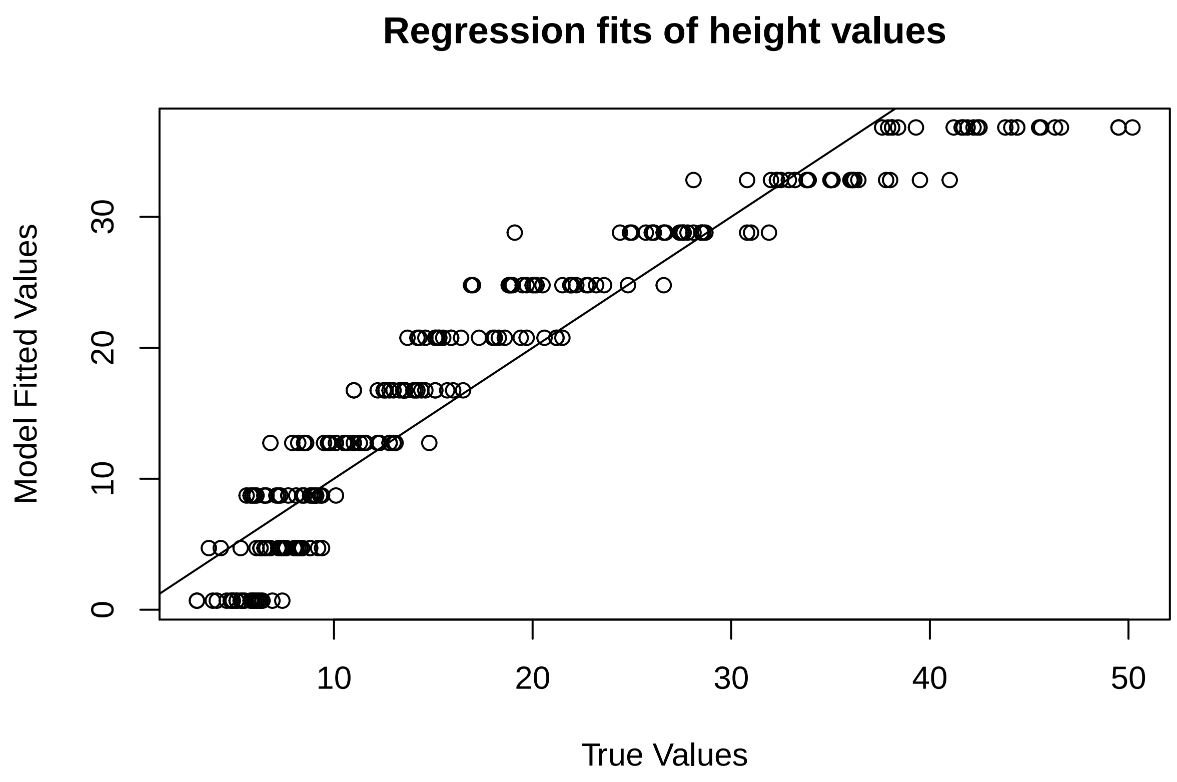
218 and 234 could be outliers.

To see the fitted values from a regression object (the values of the dependent variable predicted by the model), access the *fitted.values* attribute from a regression object with $fitted.values.

You can use the fitted values from a regression object to plot the relationship between the true values and the model fits. If the model does a good job in fitting the data, the data should fall on a diagonal line:

```
# Plot the relationship between true values and linear model fitted values.

plot(x = dd$H,                          # True values on x-axis
     y = model1$fitted.values,          # fitted values on y-axis
     xlab = "True Values",
     ylab = "Model Fitted Values",
     main = "Regression fits of height values")
abline(b = 1, a = 0)                    # Values should fall around this line!
```



Again I see a parabola-like tendency…

```
# Create a dataframe of new data
new_day <- data.frame(Days = c(10))
# Predict the value of the new data using
# the model1 regression model
predict(object = model1,      # The regression model
        newdata = new_day)    # dataframe of new data
```

```
##        1
## 2.411439
```

$$H = -0.456364 + 0.286780 \cdot (10) = 2.411436$$

This result tells us that the expected height of the ficus on day 10 is 2.411436, respectively according to our regression model.

Let us define $FDays$ as the variable $Days$ considered as a Factor. This can be done because there are sufficient number of observations for each value of the Days variable. Compare the fits with and without

FDays as an extra explanatory variable. Do it by means of the anova test anova(model1,model2). In particular, for the first model considered, compare model $H \sim Days$ with the model $H \sim Days + FDays$.

If you are choosing between a very simple model with 1 parameter, and a very complex model with, say, with 10 parameters, the very complex model needs to provide a much better fit to the data in order to justify its increased complexity. If it can't, then the more simpler model should be preferred.

To compare the fits of two models, you can use the anova() function with the regression objects as two separate arguments. The anova() function will take the model objects as arguments, and return an ANOVA testing whether the more complex model is significantly better at capturing the data than the simpler model. If the resulting p-value is sufficiently low (usually less than 0.05), we conclude that the more complex model is significantly better than the simpler model, and thus favor the more complex model. If the p-value is not sufficiently low (usually greater than 0.05), we should favor the simpler model.

```
FDays <- as.factor(dd$Days)
model1f<-lm(H~Days+FDays, dd)
anova(model1, model1f)
```

```
## Analysis of Variance Table
##
## Model 1: H ~ Days
## Model 2: H ~ Days + FDays
##   Res.Df    RSS Df Sum of Sq      F    Pr(>F)
## 1    238 3963.8
## 2    230 1184.2  8    2779.6 67.481 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

As you can see, the result shows a Df of 8 (indicating that the more complex model has 8 additional parameters), and a very small p-value ($< .001$). This means that adding the *FDays* to the model did lead to a significantly improved fit over the model1. Also, you can easily check in the summary of model1f that R squared has increased up to 96,7% and the residual standard error has decreased:

```
summary(model1f)
```

```
##
## Call:
## lm(formula = H ~ Days + FDays, data = dd)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.0333 -1.2708 -0.0333  1.3156  7.1667
##
## Coefficients: (1 not defined because of singularities)
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.119312   0.478106   8.616 1.13e-15 ***
## Days         0.299339   0.005199  57.580  < 2e-16 ***
## FDays18     -2.236574   0.621842  -3.597 0.000394 ***
## FDays32     -5.948148   0.595739  -9.984  < 2e-16 ***
## FDays46     -7.230556   0.577682 -12.516  < 2e-16 ***
## FDays60     -8.358796   0.568439 -14.705  < 2e-16 ***
## FDays74     -9.170370   0.568439 -16.133  < 2e-16 ***
## FDays88     -9.565278   0.577682 -16.558  < 2e-16 ***
## FDays102    -7.518519   0.595739 -12.620  < 2e-16 ***
## FDays116    -4.142593   0.621842  -6.662 1.97e-10 ***
## FDays130           NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.269 on 230 degrees of freedom
## Multiple R-squared:  0.967,  Adjusted R-squared:  0.9657
## F-statistic: 748.8 on 9 and 230 DF,  p-value: < 2.2e-16
```

```
anova(model1f)
```

```
## Analysis of Variance Table
##
## Response: H
##            Df Sum Sq Mean Sq  F value    Pr(>F)
## Days        1  31917   31917 6198.926 < 2.2e-16 ***
## FDays       8   2780     347   67.481 < 2.2e-16 ***
## Residuals 230   1184       5
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Given that we have a large number of observations for each value of the variable *Days*, perform the **Levene's test** test to compare the variances in the different groups and see if the homoscedasticity property may be assumed or not. Levene's test is about testing equality of variances for a given variable between *groups* split by a categorical variable (i.e. gender, geography), so you cannot use numerical variables (use the factored *Days* variable).

Levene's test can be used to answer the following question: **Is the assumption of equal variances valid?** Let's check that:

```
leveneTest(resid(model1)~FDays, dd) #FDays is a factor variable.
```

```
## Levene's Test for Homogeneity of Variance (center = median)
##        Df F value    Pr(>F)
## group   9   5.099 2.699e-06 ***
##       230
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Recall that Levene's test tests the null hypothesis that the homogeneity of variance across groups holds. In other words, the null hypothesis assumes that the variances between heights across ficus plants do not differ significantly. Look for the p-value and determine based on its value whether the null hypothesis might be rejected or not.

2.699e-06<0.001 so we reject null hypothesis. There is no homogeneity of variance in our data.

Estimate the mean and the standard deviation associated to the variable H when *Days* is equal to 0, 105 and 150. Remind that the variance of the response variable is constant and equal to the error variance. That is, we ask you to estimate: $\mathbb{E}(H|Days = a)$ and $Var(H|Days = a)$, for $a = 0, 105$ and 150.

```
# Create a dataframe of the days
new_days <- data.frame(Days = c(0,105,150))
predict(object = model1,     # The regression model
        newdata = new_days,  # dataframe of new data
        se.fit = TRUE)
```

```
## $fit
##          1          2          3
## -0.4563636 29.6555682 42.5606818
##
## $se.fit
##         1         2         3
## 0.5118977 0.3624403 0.6041816
##
## $df
## [1] 238
##
## $residual.scale
## [1] 4.080989
```

fit returns the mean estimations and $residual.scale is the residual standard deviation estimation for for Days=0,105 and 150. You can also compute it with:

```
print(cbind(mu=predict(model1,data.frame(Days=c(0,105,150))),sd=summary(model1)$sigma))
```

```
##           mu       sd
## 1 -0.4563636 4.080989
## 2 29.6555682 4.080989
## 3 42.5606818 4.080989
```

Second model: the regression parabola H with respect to the variable $Days$. Fit the data, compute the parameter estimations and interpret them.

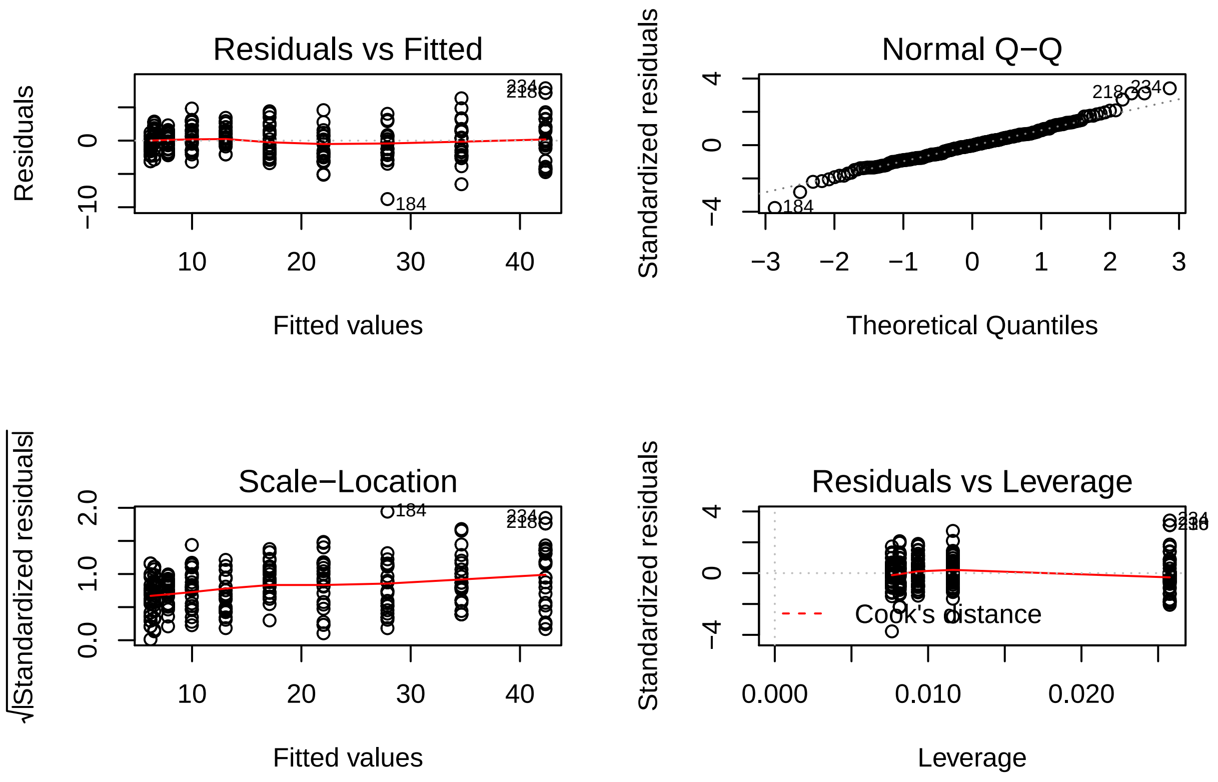
$$H = \beta_0 + \beta_1(Days) + \beta_2(Days^2)$$

```
Days2 <- dd$Days^2
summary(model2<-lm(H~Days+I(Days^2), dd))
```

```
##
## Call:
## lm(formula = H ~ Days + I(Days^2), data = dd)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.7770 -1.5422 -0.0596  1.3783  7.8653
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  6.2714174  0.4221346  14.856   <2e-16 ***
## Days        -0.0271203  0.0146695  -1.849   0.0657 .
## I(Days^2)    0.0023425  0.0001058  22.133   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.335 on 237 degrees of freedom
## Multiple R-squared:  0.964,  Adjusted R-squared:  0.9637
## F-statistic:  3171 on 2 and 237 DF,  p-value: < 2.2e-16
```

```
model2f<-lm(H~Days+I(Days^2)+FDays, dd)
```

```
oldpar<-par(mfrow=c(2,2))
plot(model2, ask=F)
```



```r
par(oldpar)
```

From model1, we see that this model captures the non-linear response in the residuals.

```r
leveneTest(resid(model2)~FDays)
```

```
## Levene's Test for Homogeneity of Variance (center = median)
##        Df F value    Pr(>F)    
## group   9   5.099 2.699e-06 ***
##       230                      
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

No homocedasticity and also there is something more about that model I don't like. The two variables seem to be highly correlated…

```r
vif(model2)
```

```
##      Days I(Days^2) 
##  15.31441  15.31441
```

Third model: the regression line $\log(H)$ with respect to the variable $Days$. This transformation is useful when one wants to stabilize the variances and the variances are approximately a quadratic function of the mean $Var(H_i) \simeq (\mu_i)^2$. Fit the data, compute the parameter estimations and interpret them.

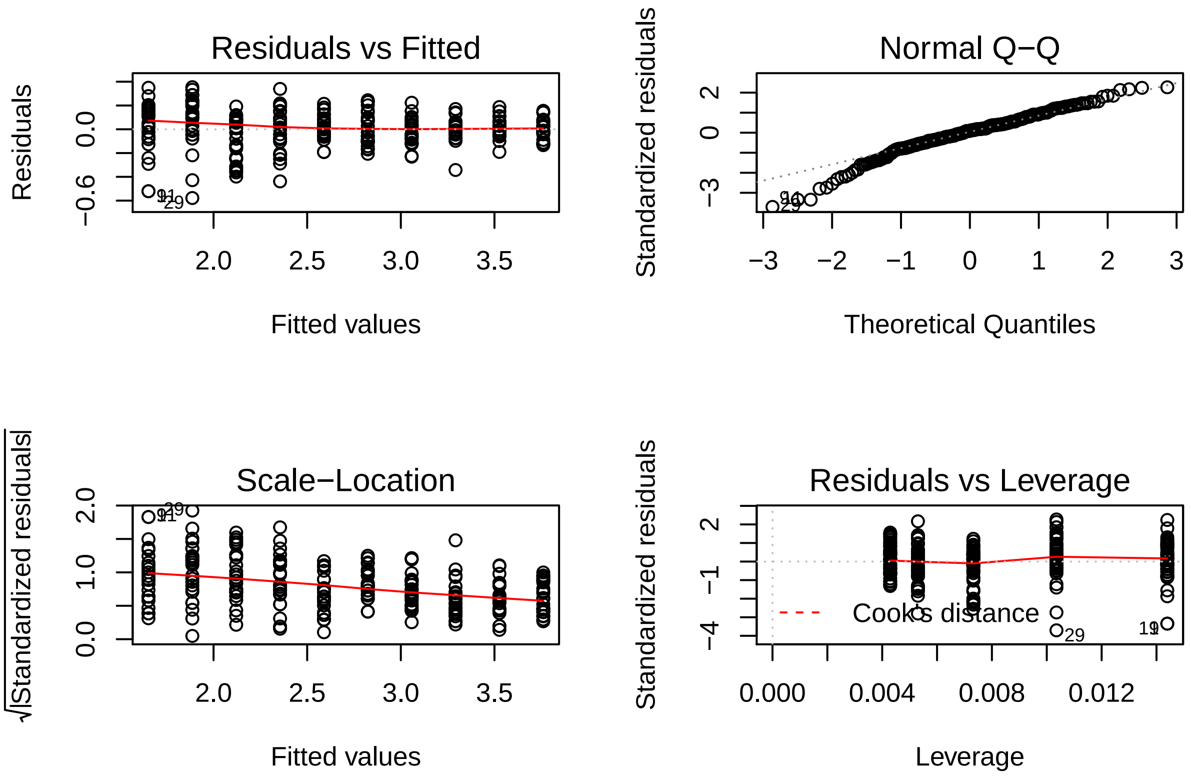
$$\log(H) = \beta_0 + \beta_1(Days)$$

```r
summary(model3<-lm(log(H)~Days, data = dd))
```

```
## 
## Call:
## lm(formula = log(H) ~ Days, data = dd)
```

```
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.57837 -0.07874  0.01515  0.09263  0.35401
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.585522   0.019688   80.53   <2e-16 ***
## Days        0.016732   0.000252   66.41   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.157 on 238 degrees of freedom
## Multiple R-squared:  0.9488, Adjusted R-squared:  0.9486
## F-statistic:  4410 on 1 and 238 DF,  p-value: < 2.2e-16
```

```
oldpar<-par(mfrow=c(2,2))
plot(model3, ask=F)
```



```
par(oldpar)
```

```
model3f <- lm(log(H)~Days+FDays, data=dd)
anova(model3, model3f)
```

```
## Analysis of Variance Table
##
## Model 1: log(H) ~ Days
## Model 2: log(H) ~ Days + FDays
##   Res.Df    RSS Df Sum of Sq      F  Pr(>F)
## 1    238 5.8632
## 2    230 5.5020  8    0.3612 1.8874 0.06288 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
leveneTest(resid(model3)~FDays)
```

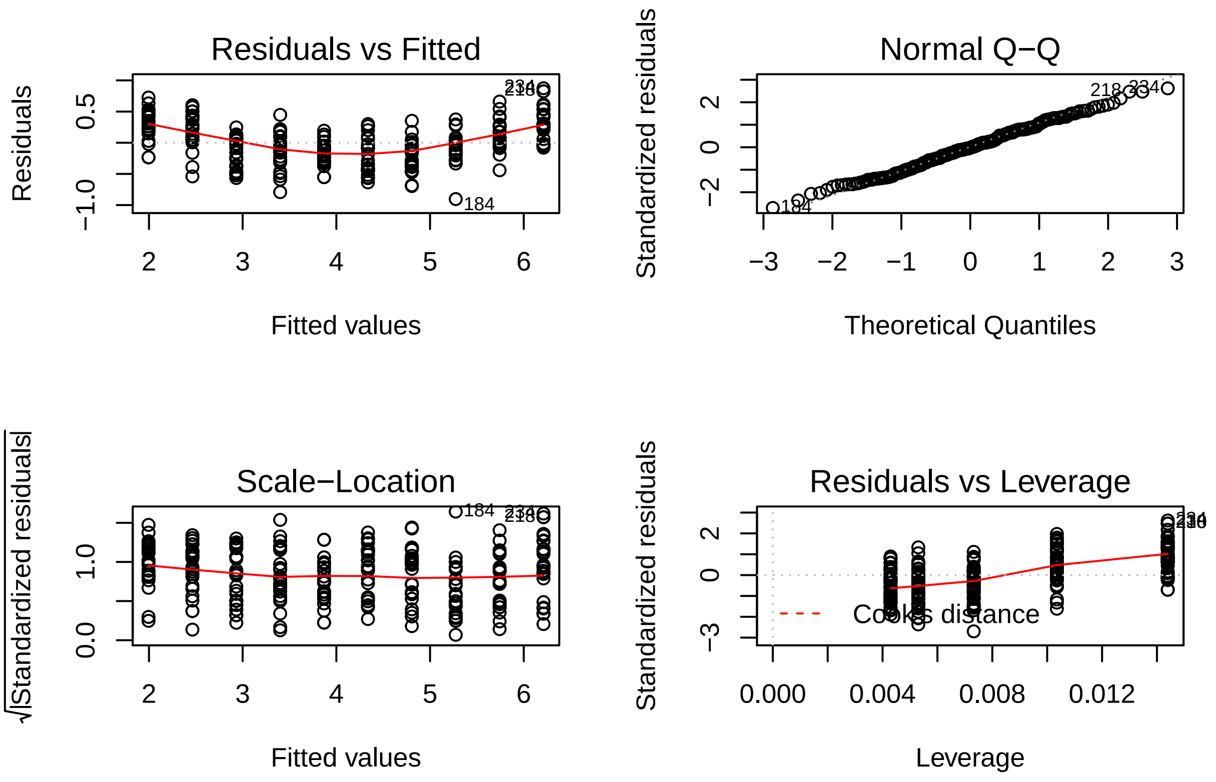
```
## Levene's Test for Homogeneity of Variance (center = median)
##        Df F value    Pr(>F)
## group   9  4.9414 4.476e-06 ***
##       230
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Fourth model: the regression line $\sqrt{H}$ with respect to the variable $Days$. This transformation is useful when one wants to stabilize the variances and the variances are approximately a linear function of the mean $Var(H_i) \simeq \mu_i$. Fit the data, compute the parameter estimations and interpret them.

```
summary(model4<-lm(sqrt(H)~Days, data = dd))
```

```
##
## Call:
## lm(formula = sqrt(H) ~ Days, data = dd)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.90381 -0.23459 -0.00804  0.24907  0.87420
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.8614427  0.0421684   44.14   <2e-16 ***
## Days        0.0334581  0.0005396   62.00   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3362 on 238 degrees of freedom
## Multiple R-squared:  0.9417, Adjusted R-squared:  0.9415
## F-statistic:  3844 on 1 and 238 DF,  p-value: < 2.2e-16
```

```
oldpar<-par(mfrow=c(2,2))
plot(model4, ask=F)
```



```r
par(oldpar)
```

```r
model4f<-lm(sqrt(H)~Days+FDays, data=dd)
anova(model4, model4f)
```

```
## Analysis of Variance Table
##
## Model 1: sqrt(H) ~ Days
## Model 2: sqrt(H) ~ Days + FDays
##   Res.Df    RSS Df Sum of Sq      F    Pr(>F)
## 1    238 26.898
## 2    230 15.866  8    11.031 19.989 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
leveneTest(resid(model4)~FDays)
```

```
## Levene's Test for Homogeneity of Variance (center = median)
##        Df F value Pr(>F)
## group   9  1.1647 0.3187
##       230
```

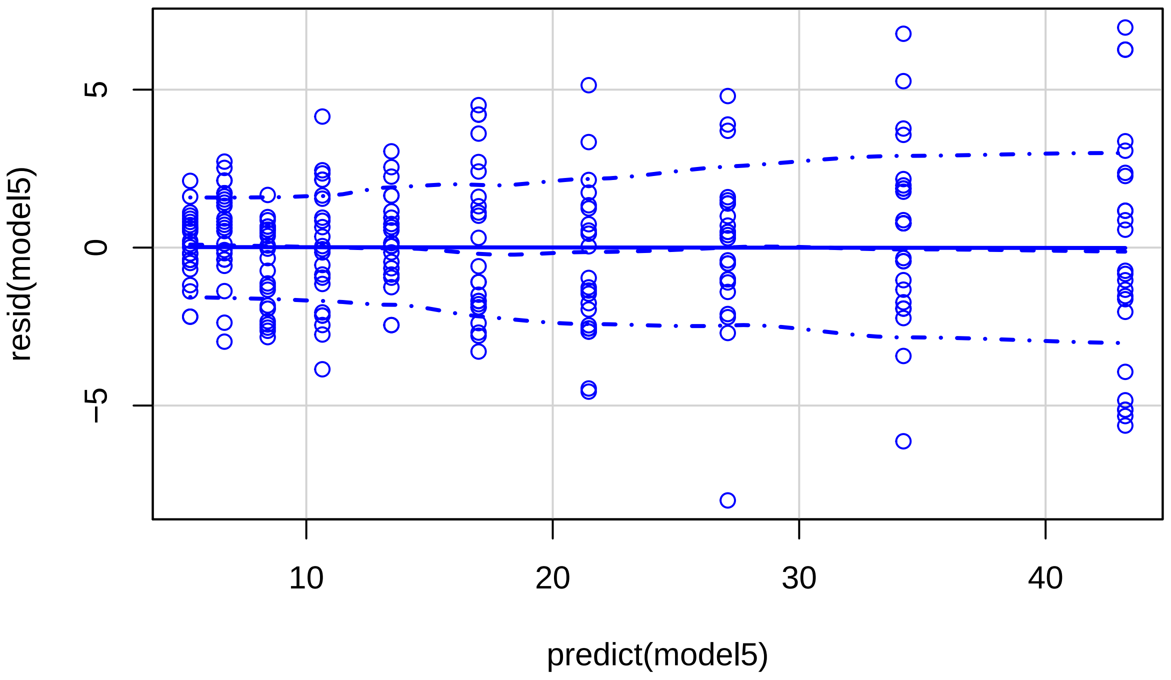
Look for the p-value and determine based on its value whether the null hypothesis might be rejected or not. 0.3187>0.001 so we accept null hypothesis! We are not sure about it but we accept that there is homogeneity of variance in our data. So this transformation has actually stabilized the variances of the data.

Let's try also the non-linear regression model defined as: $H_i = e^{\alpha+\beta \cdot Days_i} + e_i$, were e_i are iid r.v's with distribution $N(0, \sigma^2)$. We will take the coefficients from the logarithmic model:

```r
params<-coef(model3)
names(params)<-c("a","b")
summary(model5<-nls(H~exp(a+b*Days),start=params, data=dd))
```

```
##
## Formula: H ~ exp(a + b * Days)
##
## Parameters:
##    Estimate Std. Error t value Pr(>|t|)
## a 1.5985554  0.0276060   57.91   <2e-16 ***
## b 0.0166774  0.0002487   67.05   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.264 on 238 degrees of freedom
##
## Number of iterations to convergence: 2
## Achieved convergence tolerance: 1.011e-07
```

```
sp(resid(model5)~predict(model5),boxplot=F)
```



```
#pp<-predict(model5,data.frame(Days=c(0,105,150)))
pp<-exp(coef(model5)[1]+coef(model5)[2]*c(0,105,150))
print(cbind(pp,summary(model5)$sigma))
```

```
##              pp
## [1,]  4.945883 2.263892
## [2,] 28.493662 2.263892
## [3,] 60.350206 2.263892
```

```
leveneTest(resid(model5)~dd$FDays)
```

```
## Levene's Test for Homogeneity of Variance (center = median)
##        Df F value    Pr(>F)
## group   9   5.099 2.699e-06 ***
##       230
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

GLM models: Ficus plants

We define several models, from different families and links:

```
gmodelA <- glm(H~Days, family=gaussian(link="sqrt"), data=dd)
gmodelB <- glm(H~Days, family=Gamma(link="log"), data=dd)
gmodelC <- glm(H~Days, quasi(link="log", variance="mu"), data=dd)
```

```
print(gmodelA)
```

```
##
## Call:  glm(formula = H ~ Days, family = gaussian(link = "sqrt"), data = dd)
##
## Coefficients:
## (Intercept)         Days
##     1.55992      0.03706
##
## Degrees of Freedom: 239 Total (i.e. Null);  238 Residual
## Null Deviance:       35880
## Residual Deviance: 1823  AIC: 1174
```

```
summary(gmodelA)
```

```
##
## Call:
## glm(formula = H ~ Days, family = gaussian(link = "sqrt"), data = dd)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -9.4170  -1.7410   0.1247   2.0765   9.5232
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.559916   0.062123   25.11   <2e-16 ***
## Days        0.037061   0.000629   58.92   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 7.659018)
##
##     Null deviance: 35880.6  on 239  degrees of freedom
## Residual deviance:  1822.8  on 238  degrees of freedom
## AIC: 1173.7
##
## Number of Fisher Scoring iterations: 5
```

The estimation of the dispersion parameter obtained with the Pearson statistic is:

```
summary(gmodelA)$dispersion
```

```
## [1] 7.659018
```

The dispersion parameter, as you know, is computed as the quotient of the Pearson Statistic and the degrees of freedom:

```
(PS<-sum(residuals(gmodelA, type="pearson")^2))
```

```
## [1] 1822.815
```

```r
PS/gmodelA$df.res
```

```
## [1] 7.658889
```

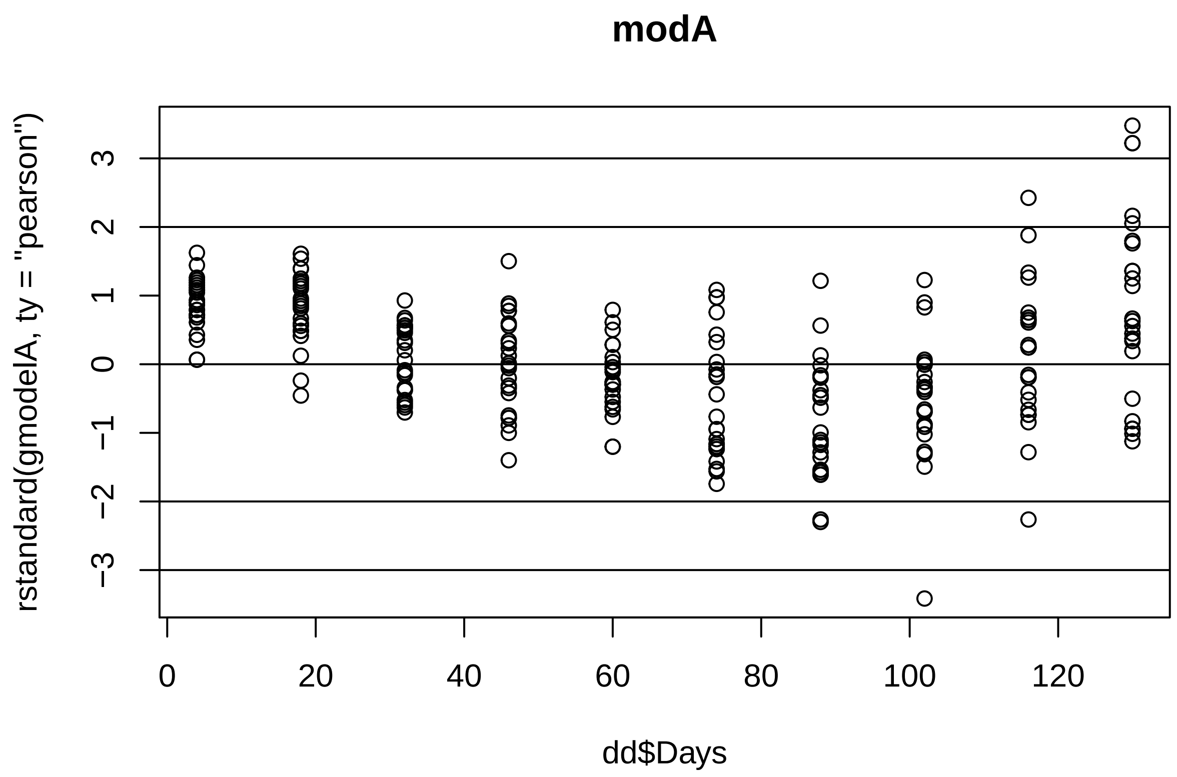
```r
#P valor
2*min(pchisq(PS, gmodelA$df.res), 1-pchisq(PS, gmodelA$df.res))
```

```
## [1] 0
```

```r
#IC
c(qchisq(0.025, gmodelA$df.res), qchisq(0.975, gmodelA$df.res))/gmodelA$df.res
```

```
## [1] 0.8284171 1.1874918
```

```r
plot(dd$Days,rstandard(gmodelA,ty="pearson"),main="modA")
abline(h=c(-3,-2,0,2,3))
```



```r
for (i in 1:length(rstandard(gmodelA))) if (abs(rstandard(gmodelA)[i]) > 2) {
  cat("value is ", rstandard(gmodelA)[i], " on day ", dd$Days[i], "\n")
}
```

```
## value is  -2.298732  on day  88
## value is  -2.262501  on day  88
## value is  -3.414457  on day  102
## value is  2.424853  on day  116
## value is  -2.263261  on day  116
## value is  2.053111  on day  130
## value is  3.221474  on day  130
## value is  2.162645  on day  130
## value is  3.221474  on day  130
## value is  3.477054  on day  130
```

```
gmodelAf <- glm(H~Days+FDays, family=gaussian(link="sqrt"), data=dd)
gmodelBf <- glm(H~Days+FDays, family=Gamma(link="log"), data=dd)
gmodelCf <- glm(H~Days+FDays, quasi(link="log", variance="mu"), data=dd)
```

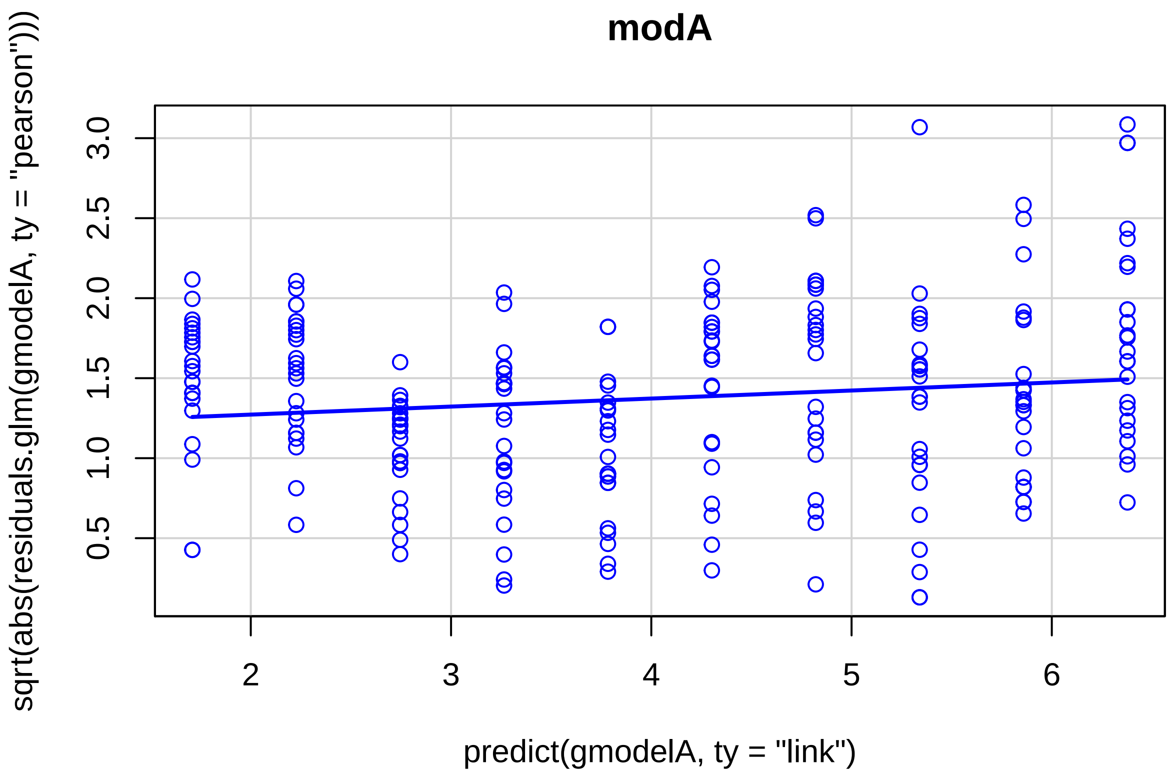
```
anova(gmodelA, gmodelAf, test="F")
```

```
## Analysis of Deviance Table
##
## Model 1: H ~ Days
## Model 2: H ~ Days + FDays
##   Resid. Df Resid. Dev Df Deviance      F    Pr(>F)
## 1       238     1822.8
## 2       230     1184.2  8    638.6 15.504 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
leveneTest(resid(gmodelA)~FDays)
```

```
## Levene's Test for Homogeneity of Variance (center = median)
##        Df F value    Pr(>F)
## group   9   5.099 2.699e-06 ***
##       230
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
sp(sqrt(abs(residuals.glm(gmodelA,ty="pearson")))~predict(gmodelA,ty="link"),boxplot=F,smooth=F,main="mo
```



```
mm<-predict(gmodelA,data.frame(Days=c(0,105,150)),ty="response")
print(mmA<-cbind(mu=mm,sd=sqrt(gmodelA$deviance/gmodelA$df.residual)))
```

```
##          mu       sd
## 1  2.433337 2.76747
## 2 29.716809 2.76747
## 3 50.680931 2.76747
```

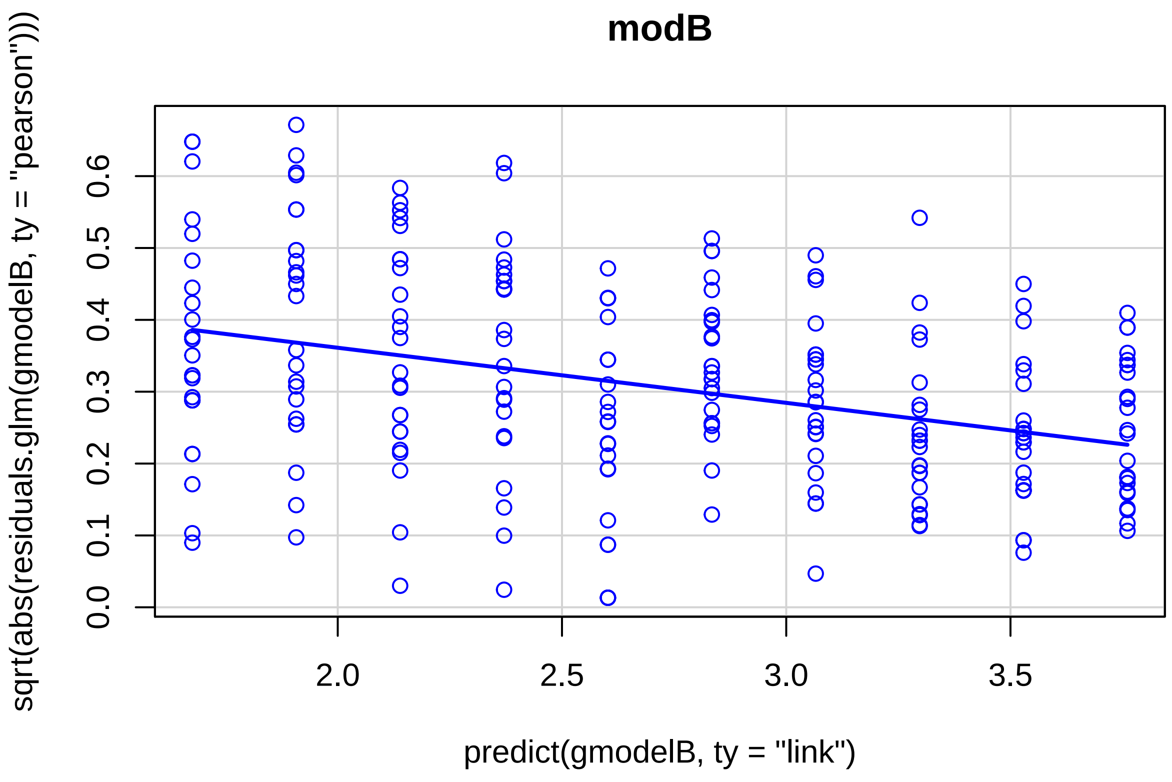
```
summary(gmodelB)
```

```
##
## Call:
## glm(formula = H ~ Days, family = Gamma(link = "log"), data = dd)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -0.54486  -0.08923   0.00054   0.08131   0.35279
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.6096300  0.0187494   85.85   <2e-16 ***
## Days        0.0165481  0.0002399   68.97   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for Gamma family taken to be 0.02234278)
##
##     Null deviance: 108.0180  on 239  degrees of freedom
## Residual deviance:   5.6503  on 238  degrees of freedom
## AIC: 1087.4
##
## Number of Fisher Scoring iterations: 4
```

```
leveneTest(resid(gmodelB, type="pearson")~FDays)
```

```
## Levene's Test for Homogeneity of Variance (center = median)
##        Df F value    Pr(>F)
## group   9  5.3625 1.158e-06 ***
##       230
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
sp(sqrt(abs(residuals.glm(gmodelB,ty="pearson")))~predict(gmodelB,ty="link"),
   boxplot=F,smooth=F,main="modB")
```



```
predict(gmodelB, new=data.frame(Days=c(150)), se.fit=TRUE, type = "response")
```

```
## $fit
##        1
## 59.84983
##
## $se.fit
##        1
## 1.324444
##
## $residual.scale
## [1] 0.149475
```

```
mm<-predict(gmodelB,data.frame(Days=c(0,105,150)),ty="response")
print(mmB<-cbind(mu=mm,sd=mm*sqrt(gmodelB$deviance/gmodelB$df.residual)))
```

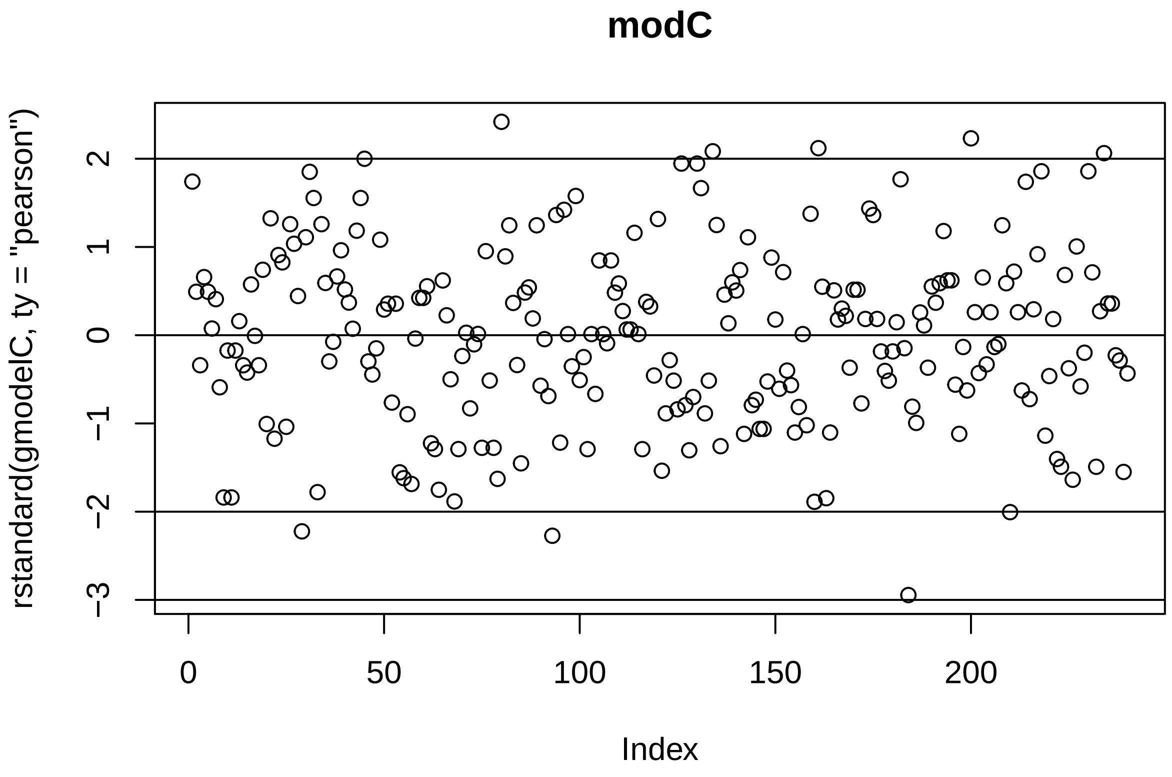
```
##          mu        sd
## 1  5.000961 0.7705511
## 2 28.422355 4.3793340
## 3 59.849827 9.2216982
```

```
summary(gmodelC)
```

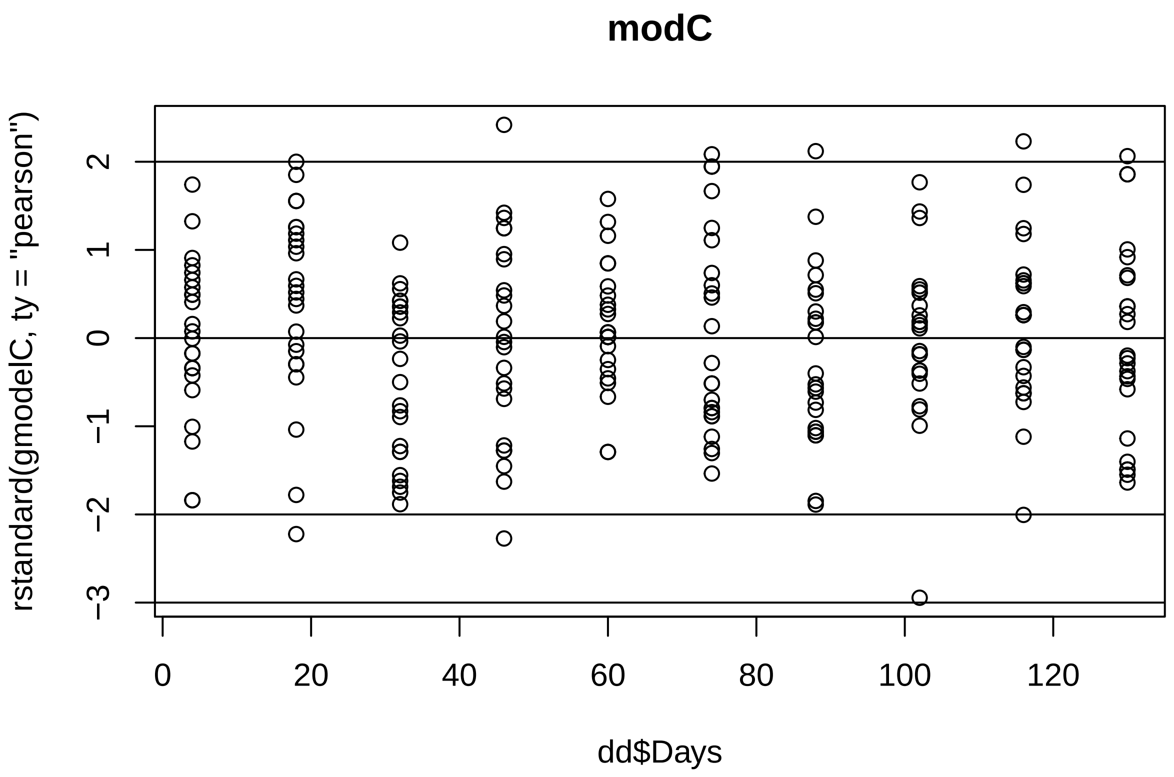
```
##
## Call:
## glm(formula = H ~ Days, family = quasi(link = "log", variance = "mu"),
##     data = dd)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.62347  -0.33503   0.00701   0.31139   1.19175
```

```
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.6028119  0.0216644   73.98   <2e-16 ***
## Days        0.0166344  0.0002203   75.49   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for quasi family taken to be 0.2741126)
##
##     Null deviance: 1839.175  on 239  degrees of freedom
## Residual deviance:   66.093  on 238  degrees of freedom
## AIC: NA
##
## Number of Fisher Scoring iterations: 4
```

```
plot(rstandard(gmodelC,ty="pearson"),main="modC")
abline(h=c(-3,-2,0,2,3))
```



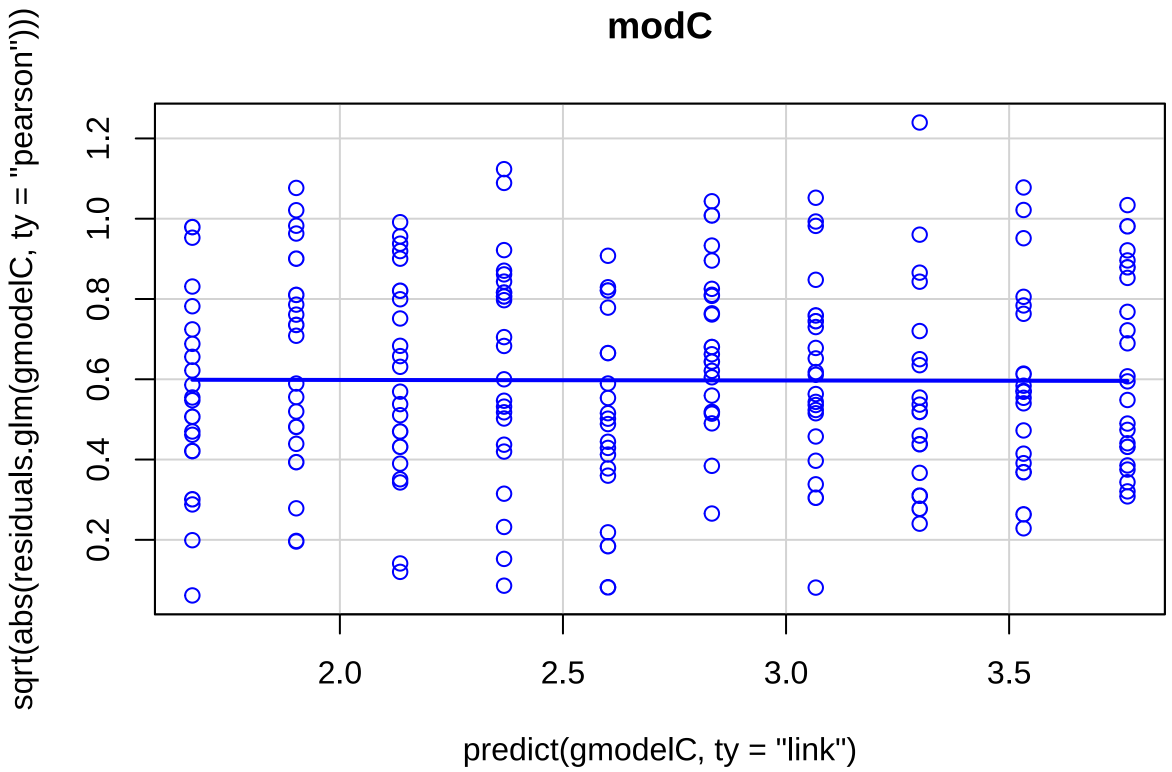
```
plot(dd$Days,rstandard(gmodelC,ty="pearson"),main="modC")
abline(h=c(-3,-2,0,2,3))
```



```
leveneTest(resid(gmodelC, type="pearson")~FDays)
```

```
## Levene's Test for Homogeneity of Variance (center = median)
##        Df F value Pr(>F)
## group   9  1.1901 0.3022
##       230
```

```
sp(sqrt(abs(residuals.glm(gmodelC,ty="pearson")))~predict(gmodelC,ty="link"),
   boxplot=F,smooth=F,main="modC")
```



```
mm<-predict(gmodelC,data.frame(Days=c(0,105,150)),ty="response")
print(mmC<-cbind(mu=mm,sd=sqrt(mm)*sqrt(gmodelC$deviance/gmodelC$df.residual)))
```

```
##         mu       sd
## 1  4.96698 1.174450
## 2 28.48625 2.812587
## 3 60.21783 4.089318
```

```
1-gmodelA$deviance/gmodelA$null.deviance
```

```
## [1] 0.9491977
```

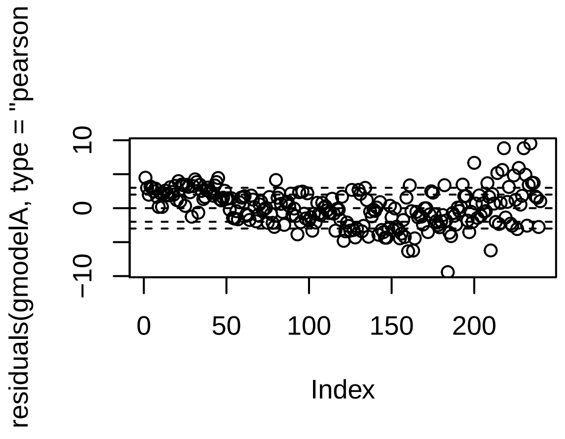
```
1-gmodelB$deviance/gmodelB$null.deviance
```

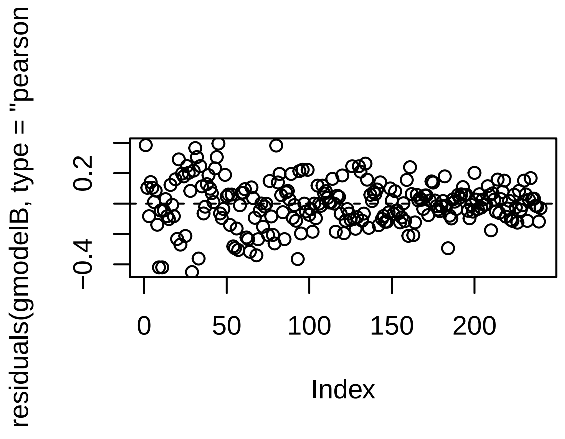
```
## [1] 0.947691
```

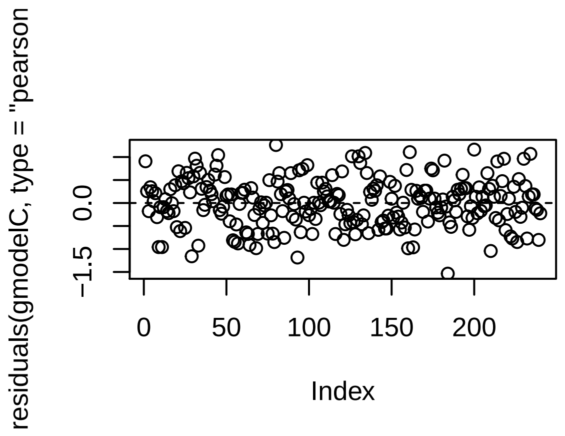
```
1-gmodelC$deviance/gmodelC$null.deviance
```

```
## [1] 0.9640639
```

```
oldpar<-par(mfrow=c(2,2))
plot(residuals(gmodelA,type="pearson"))
abline(h=c(-3,-2,0,2,3), lty=2)
plot(residuals(gmodelB,type="pearson"))
abline(h=c(-3,-2,0,2,3), lty=2)
plot(residuals(gmodelC,type="pearson"))
abline(h=c(-3,-2,0,2,3), lty=2)
par(oldpar)
```







```
print(rbind(logLik=c(modA=logLik(gmodelA),modB=logLik(gmodelB),modC=logLik(gmodelC)),
              AIC=c(AIC(gmodelA),AIC(gmodelB),AIC(gmodelC)),
              "R2"=c(1-gmodelA$deviance/gmodelA$null.deviance,1-gmodelB$deviance/gmodelB$null.deviance,
```

```
##                 modA        modB      modC
## logLik -583.8450840 -540.723672        NA
## AIC    1173.6901680 1087.447345        NA
## R2        0.9491977    0.947691 0.9640639
```

Linear Regression

The dataset includes data on 150 diamonds sold at an auction. Here are the first few rows of the dataset:

```
head(diamonds)
```

```
##   weight clarity color value
## 1   9.35    0.88     4 182.5
## 2  11.10    1.05     5 191.2
## 3   8.65    0.85     6 175.7
## 4  10.43    1.15     5 195.2
## 5  10.62    0.92     5 181.6
## 6  12.35    0.44     4 182.9
```

Our goal is to come up with a linear model we can use to estimate the value of each diamond as a linear combination of three independent variables: its *weight* and *clarity*. The linear model will estimate each diamond's value using the following equation:

$$Value = \beta_0 + Weight \cdot \beta_1 + Clarity \cdot \beta_2$$

where β_1 is the increase in value for each increase of 1 in weight, β_2 is the increase in value for each increase of 1 in clarity (etc.). Finally, β_0 is the baseline value of a diamond with a value of 0 in all independent variables (the *intercept*). Because $Value$ is the dependent variable we will specify the formula as:

```
diamonds.lm <- lm(formula = value ~ weight + clarity, data = diamonds)
diamonds.lm
```

```
##
## Call:
## lm(formula = value ~ weight + clarity, data = diamonds)
##
## Coefficients:
## (Intercept)       weight      clarity
##     145.446        2.219       22.036
```

To see the results of the regression analysis, including estimates for each of the beta values, we'll use the $summary()$ function:

```
summary(diamonds.lm)
```

```
##
## Call:
## lm(formula = value ~ weight + clarity, data = diamonds)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -10.034  -3.802  -0.196   3.207  11.166
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  145.446      2.795   52.04   <2e-16 ***
## weight         2.219      0.199   11.15   <2e-16 ***
## clarity       22.036      2.129   10.35   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.681 on 147 degrees of freedom
## Multiple R-squared:  0.6334, Adjusted R-squared:  0.6284
## F-statistic:   127 on 2 and 147 DF,  p-value: < 2.2e-16
```

```
# Which components are in the regression object? access names(diamonds.lm).
```

The resulting regression model is:

$$Value = 145.446 + Weight \cdot 2.219 + Clarity \cdot 22.036$$

Once you have created a regression model with $lm()$, you can use it to easily predict results from new datasets using the $predict()$ function.For example, let's say I discovered 3 new diamonds with the following characteristics:

- Diamond 1: weight = 20, clarity = 1.5.
- Diamond 2: weight = 10, clarity = 0.2.
- Diamond 3: weight = 15, clarity = 5.0.

Which of these has more value respectively according to our regression model?

I'll use the $predict()$ function to predict the value of each of these diamonds using the regression model *diamond.lm* that I created before. The two main arguments to $predict()$ are *object* – the regression object we've already defined), and *newdata* – the dataframe of new data. Warning! The dataframe that you use in the *newdata* argument to *predict*() must have column names equal to the names of the coefficients in the model. If the names are different, the *predict*() function won't know which column of data applies to which coefficient and will return an error:

```r
# Create a dataframe of new diamond data
diamonds.new <- data.frame(weight = c(20, 10, 15), clarity = c(1.5, 0.2, 5.0))

# Predict the value of the new diamonds using
#  the diamonds.lm regression model
predict(object = diamonds.lm,     # The regression model
        newdata = diamonds.new)   # dataframe of new data
```

```
##        1        2        3
## 222.8722 172.0393 288.9061
```

To include interaction terms in a regression model, just put an asterix (*) between the independent variables.