Non Linear regression explained

sameh magdeldin

2025-01-07

## Non linear regression analysis

Module contents

- Nonlinear regression
- Multiple non linear regression -Cross validation (simple example)

Packages needed in this tutorial

```r
library("ggplot2") # for plotting
library("plyr") #data manipulation
library("dplyr") #data manipulation "should be installed after plyr
library("RColorBrewer") #this is to give a nice coloring for ur figures
library("caret") # for cross validation
library("car") # vif testing
```

##Nonlinear regression [nls; nonliner least square]

In many cases in biology,processes are occuring in a nonlinear fashion. for example: population growth, enzymatic reaction, infection with a certain disease etc.. I will try to simplyfy how to make a prediction using a nonlinear model.

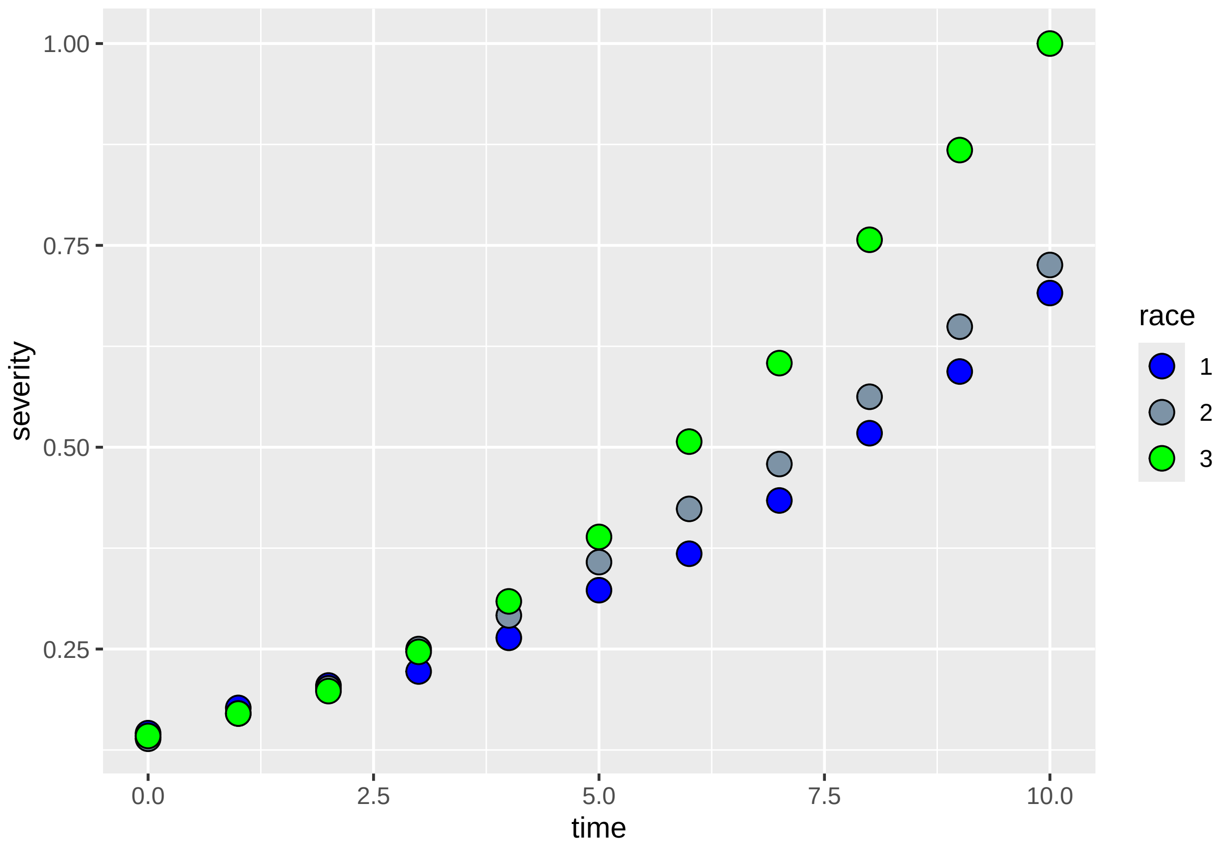
Lets assume a disease severity over a race and age for a different population I will construct the dataframe to work with

```r
time <- c(seq(0,10),seq(0,10),seq(0,10))
race <- c(rep(1,11),rep(2,11),rep(3,11))
severity <- c(42,51,59,64,76,93,106,125,149,171,199,40,49,58,72,84,103,
              122,138,162,187,209,41,49,57,71,89,112,146,174,218,250,288)/288
data <- data.frame(time=time,race=race,severity=severity)
```

alternatively load the data named nonlineardata.csv

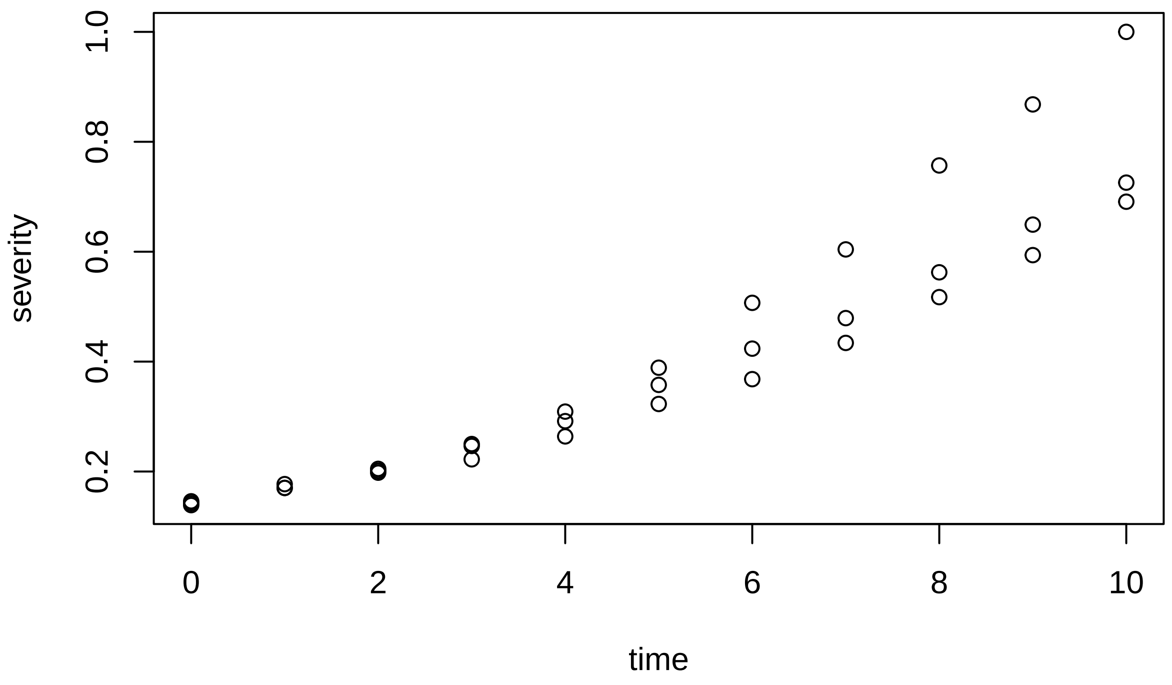
Now lets plot the dataframe, time versus severity for the 3 races

```r
ggplot(data,aes(time,severity,fill=race))+geom_point(shape=21,size=4)+
  scale_fill_gradient(low="blue", high="green",breaks=1:3, guide=guide_legend())
```



or i can use the base plot

```r
plot(severity~time)
```



It seems from the scatter plot that data can be fit nonlinear regression.

we have 2 options here, either to separate the 3 plots independantly and

fit 3 different models /race or to make an approximate Non linear model

for them. . . .ill do that

To do so , we need to know some **initial parameters**

These parameters guide the fitted model on how it should slop

to get these parameters, we will use **getInitial** and **SSlogis** function

```
getInitial(severity ~ SSlogis(time, alpha, xmid, scale),data = data)
```

```
##     alpha      xmid     scale 
##  2.212477 12.506994  4.572397
```

```
# my data is named "data" here
```

this is to extract **alpha**, **xmid**, and **scale**

From these 3 parameters, ill calculate the alpha, beta and gamma constatnts

setting the parameter as preparation for fitting our model

```
parameters <- c(alpha=2.212,beta=12.507/4.572,gamma=1/4.572)
# alpha:2.212
# beta:xmid/scale
# gamma (or r) is 1/scale
# these 3 parameters are called coefficients
```

```
parameters
```

```
##     alpha      beta     gamma 
## 2.2120000 2.7355643 0.2187227
```

Now i will fit my nonlinear model

```
fit<- nls(severity~alpha/(1+exp(beta-gamma*time)),data,start=parameters,trace=T)
```

```
## 0.1621433   (1.89e-03): par = (2.212 2.735564 0.2187227)
## 0.1621427   (4.99e-06): par = (2.212409 2.735298 0.2187056)
```

what is exponent (exp)??

Exponents are shorthand for repeated multiplication of the same thing by itself. For instance, the shorthand for multiplying three copies of the number 5 is shown on the right-hand side of the "equals" sign in (5)(5)(5) = 53. The "exponent", being 3 in this example, stands for however many times the value is being multiplied

```
fit
```

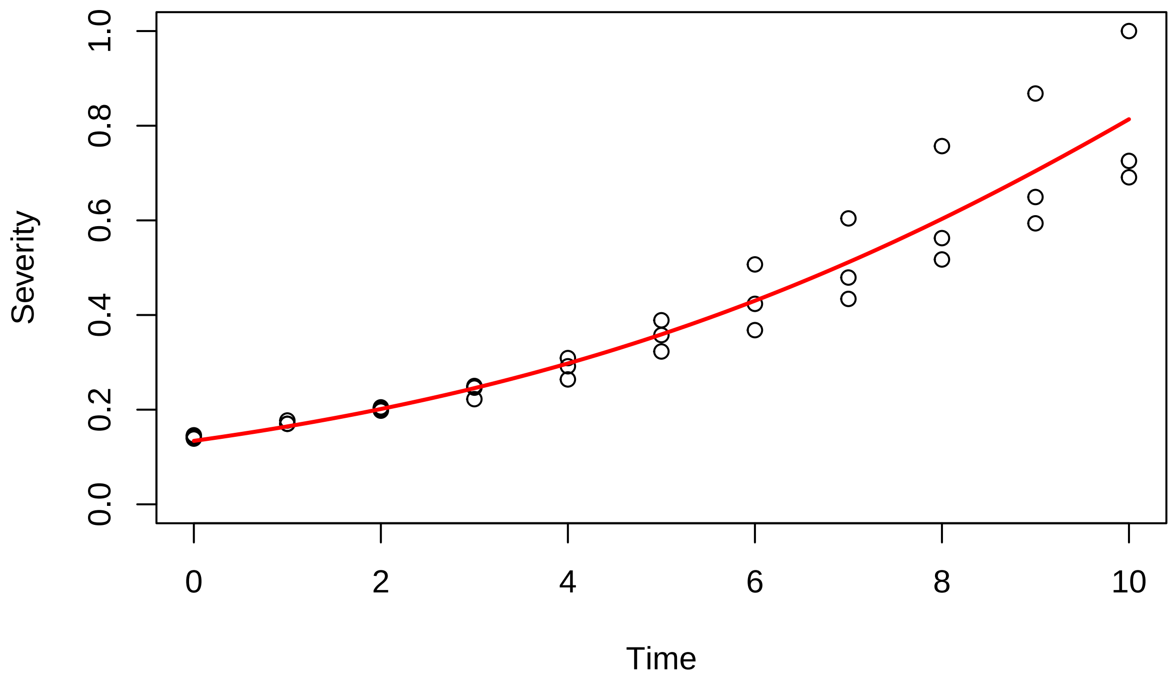
```
## Nonlinear regression model
##   model: severity ~ alpha/(1 + exp(beta - gamma * time))
##    data: data
##  alpha   beta  gamma
## 2.2124 2.7353 0.2187
##  residual sum-of-squares: 0.1621
##
## Number of iterations to convergence: 1
## Achieved convergence tolerance: 4.989e-06
```

lets see how our data fits the nonlinear model

2.21/(1+exp(2.74-0.22*x)

Now, lets draw our scatter plot and then plot the formula over the plot as a line to see if it fits it or not

```
plot(severity ~ time, xlab = "Time", ylab = "Severity",
     xlim = range(time), ylim = c(0, max(severity)))
curve(2.21 / (1 + exp(2.74 - 0.22 * x)),
      from = min(time), to = max(time),
      add = TRUE, col = "red", lwd = 2)
```



Ok , it seems that the model fits our data

note that in base plot u call the scatter, then you add the line to an

#existing plot

##Prediction

we have the equation as follows

2.21/(1+exp(2.74-0.22*x)

```r
#lets call the data again
head(data,ncol=5)
```

```
##   time race  severity
## 1    0    1 0.1458333
## 2    1    1 0.1770833
## 3    2    1 0.2048611
## 4    3    1 0.2222222
## 5    4    1 0.2638889
## 6    5    1 0.3229167
```

#to get y reads if x is 1 2.21/(1+exp(2.74-0.22*1)) # just replace x

#to get y reads if x is 4 2.21/(1+exp(2.74-0.22*4)) # read # 5 (0.2978957)

I will use the function ability of R for mass prediction (predict several

#values at once), do yo remember ?? (in intro to R module)

```r
prediction <- function(x) 2.21/(1+exp(2.74-0.22*x))
```

i will test it

```r
prediction(4) # u can set a vector with different entery also
```

```
## [1] 0.2976937
```

it works!

Now, i can put any values, lets make a new time list as (x) and get

#prediction of them

```
time1 <-data$time+0.324 # just making another data from the original time
        #values
```

```
data$time1 <- time1 # inserting in the data frame # you can also omit it
```

```
prediction(data$time1) # check it on the graph
```

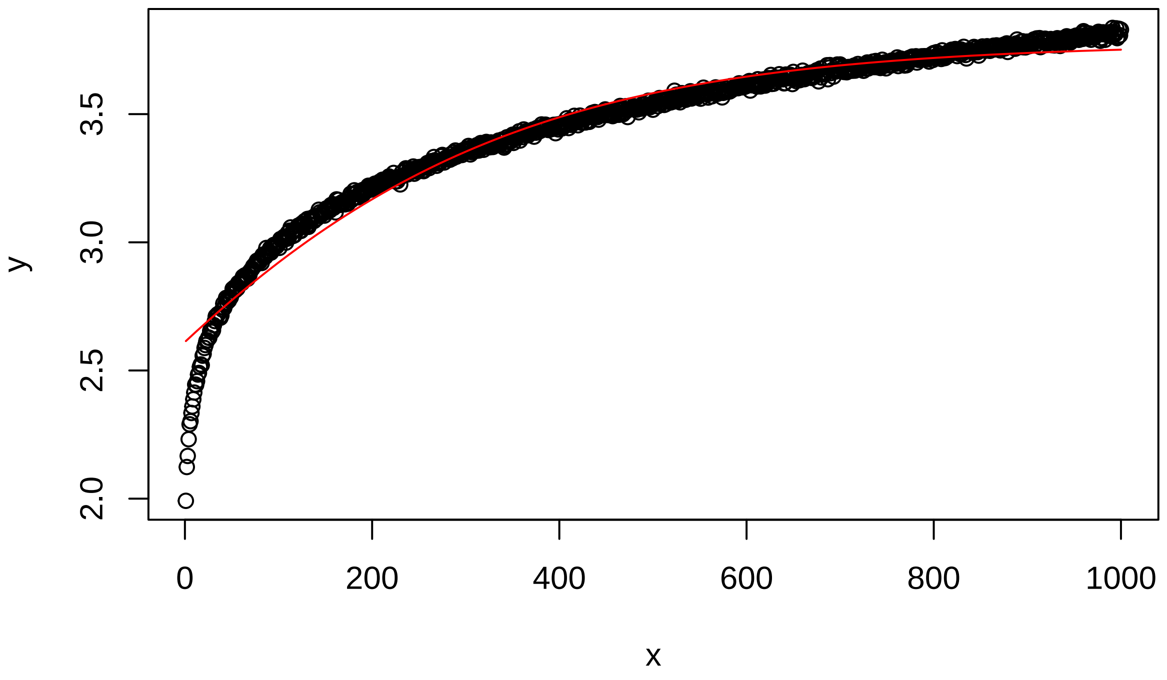
```
##  [1] 0.1433064 0.1757662 0.2148140 0.2614218 0.3165377 0.3810017 0.4554362
##  [8] 0.5401182 0.6348487 0.7388425 0.8506715 0.1433064 0.1757662 0.2148140
## [15] 0.2614218 0.3165377 0.3810017 0.4554362 0.5401182 0.6348487 0.7388425
## [22] 0.8506715 0.1433064 0.1757662 0.2148140 0.2614218 0.3165377 0.3810017
## [29] 0.4554362 0.5401182 0.6348487 0.7388425 0.8506715
```

Lets take another example model

```
set.seed(5)
x <- 1:1000
y <- 1 + x^0.15 + rnorm(1000, 0, 0.01)
datax <- data.frame(x=x,y=y)
plot(y~x)
getInitial(y ~ SSlogis(x, alpha, xmid, scale),data = datax)
```

```
##       alpha        xmid       scale
##    3.776111 -191.542957  237.274007
```

```
parameters1 <- c(alpha=3.776111,beta=-191.542957/237.274007,gamma=1/237.274007)
curve(3.7761110 /(1+exp(-0.807264813-0.004214537 *x)),from=x[1],to=x[1000],
                    add=TRUE,col = "red")
```



##Multiple nonlinear regression

Multiple nonlinear regression is a powerful statistical technique used to model complex relationships between a dependent (response) variable and multiple independent (predictor) variables when the relationship cannot be adequately described by a linear combination of parameters. Unlike linear regression, which assumes additive and proportional effects, nonlinear regression allows for curvilinear, multiplicative, or asymptotic relationships, making it indispensable in fields like biology, economics, engineering, and environmental science.

Set seed for reproducibility

set.seed(123)

Generate synthetic dummy data

```
X1 <- runif(100, 0, 10)
X2 <- runif(100, 0, 10)
Y <- 2 * exp(0.5 * X1) / (1 + 0.3 * X2) + rnorm(100, 0, 0.5)  # Add noise
```

Create data frame

```
data <- data.frame(Y, X1, X2)
```

Try different initial parameter estimates

```
start_vals <- list(a = 2, b = 0.5, c = 0.3)
```

Fit a nonlinear model: Y = (a * exp(b * X1)) / (1 + c * X2)

```
model <- nls(Y ~ (a * exp(b * X1)) / (1 + c * X2),
             data = data,
             start = start_vals,
             algorithm = "port",    # Use Levenberg-Marquardt algorithm
             trace = TRUE)          # Show iteration progress
```

```
##   0:     11.908476:  2.00000 0.500000 0.300000
##   1:     11.728858:  2.00937 0.499620 0.300990
##   2:     11.664020:  2.02417 0.498753 0.300822
##   3:     11.663951:  2.02429 0.498749 0.300822
##   4:     11.663951:  2.02429 0.498749 0.300822
```

Display summary of model

```r
summary(model)
```

```
##
## Formula: Y ~ (a * exp(b * X1))/(1 + c * X2)
##
## Parameters:
##   Estimate Std. Error t value Pr(>|t|)
## a 2.024294   0.020341   99.52   <2e-16 ***
## b 0.498749   0.001193  418.13   <2e-16 ***
## c 0.300822   0.001354  222.22   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4904 on 97 degrees of freedom
##
## Algorithm "port", convergence message: relative convergence (4)
```

Generate a sequence of X1 values for a smooth fit

```r
X1_seq <- seq(min(data$X1), max(data$X1), length.out = 100)
```

Create a new data frame for predictions

```r
new_data <- data.frame(X1 = X1_seq)
```

```r
new_data$X2 <- mean(data$X2)  # Keep X2 constant
```

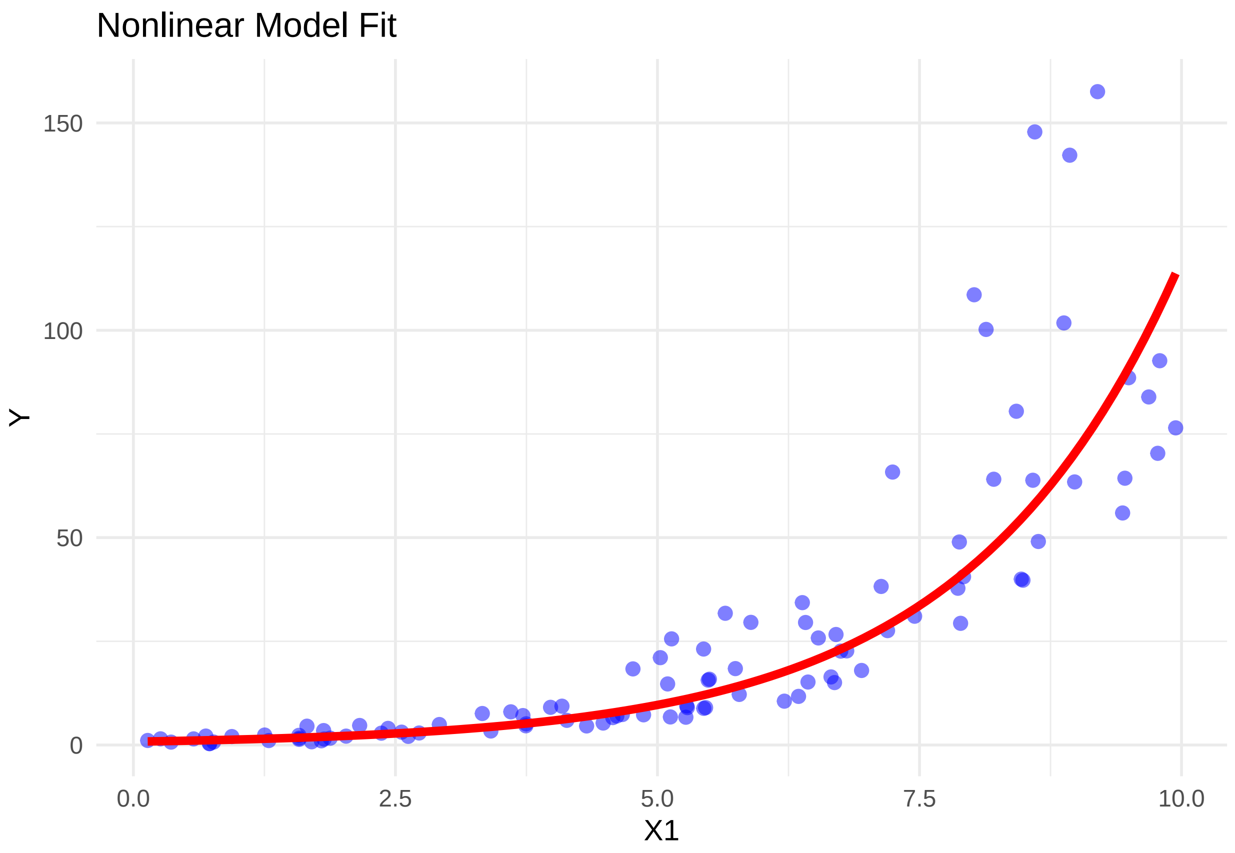
Manually compute predictions

```r
new_data$Predicted_Y <- predict(model, newdata = list(X1 = new_data$X1,
                      X2 = new_data$X2))
```

Create the plot

```r
ggplot(data, aes(x = X1, y = Y)) +
  geom_point(color = "blue", alpha = 0.5, size = 2) +
  geom_line(data = new_data, aes(x = X1, y = Predicted_Y),
            color = "red", size = 1.5) +
  labs(title = "Nonlinear Model Fit",
       x = "X1",
       y = "Y") +
  theme_minimal()
```

```
## Warning: Using ‘size‘ aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use ‘linewidth‘ instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```



```r
dev.off()# clean ur board
```

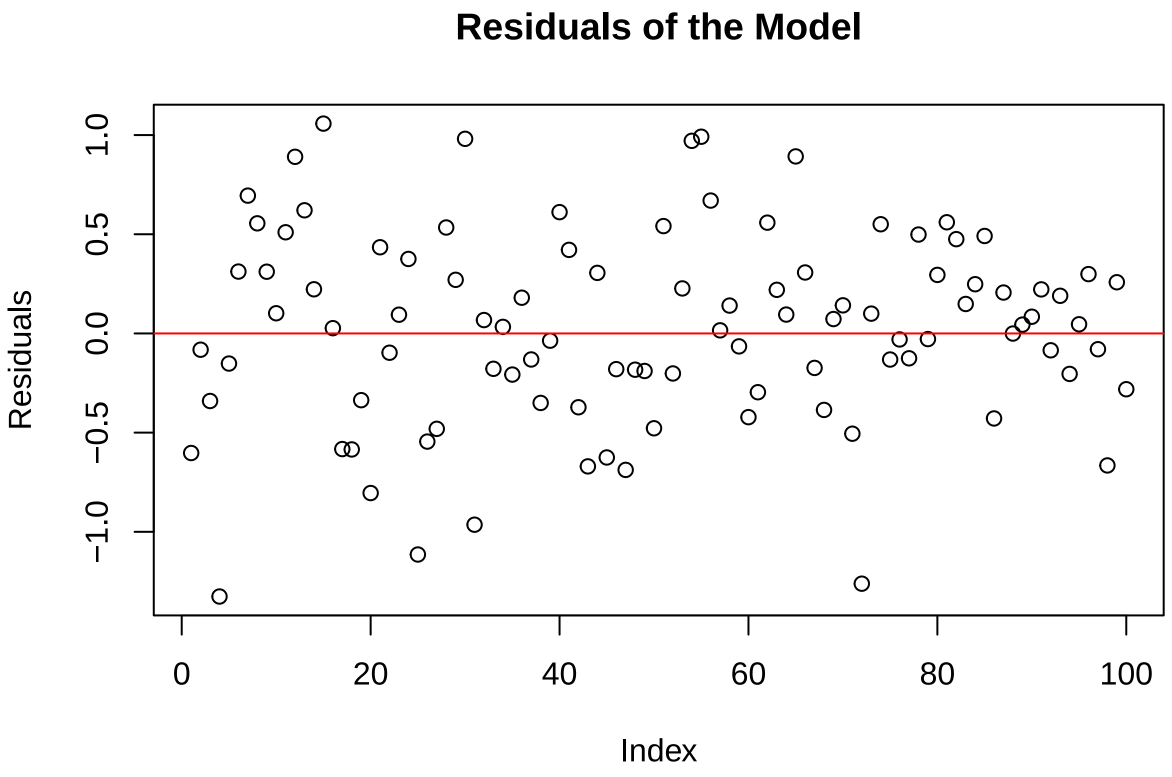
```
## null device
##           1
```

#Evaluation of the model

1. Fitting the model (earlier)

2. Plot residuals

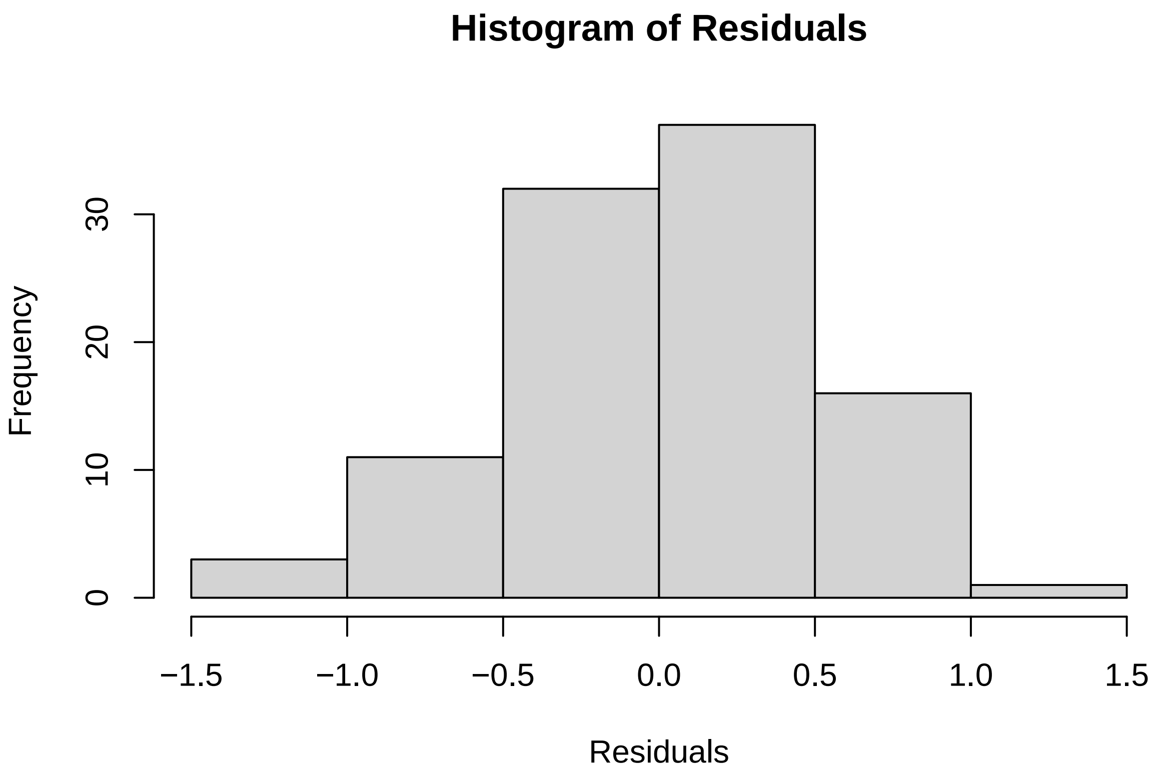
```r
plot(residuals(model), main = "Residuals of the Model", ylab = "Residuals",
xlab = "Index")
abline(h = 0, col = "red")
```



Tutors note Ideally, the residuals should be randomly distributed without any visible pattern (indicating the model is appropriate). If there's a pattern (e.g., a funnel shape), the model may need improvement.

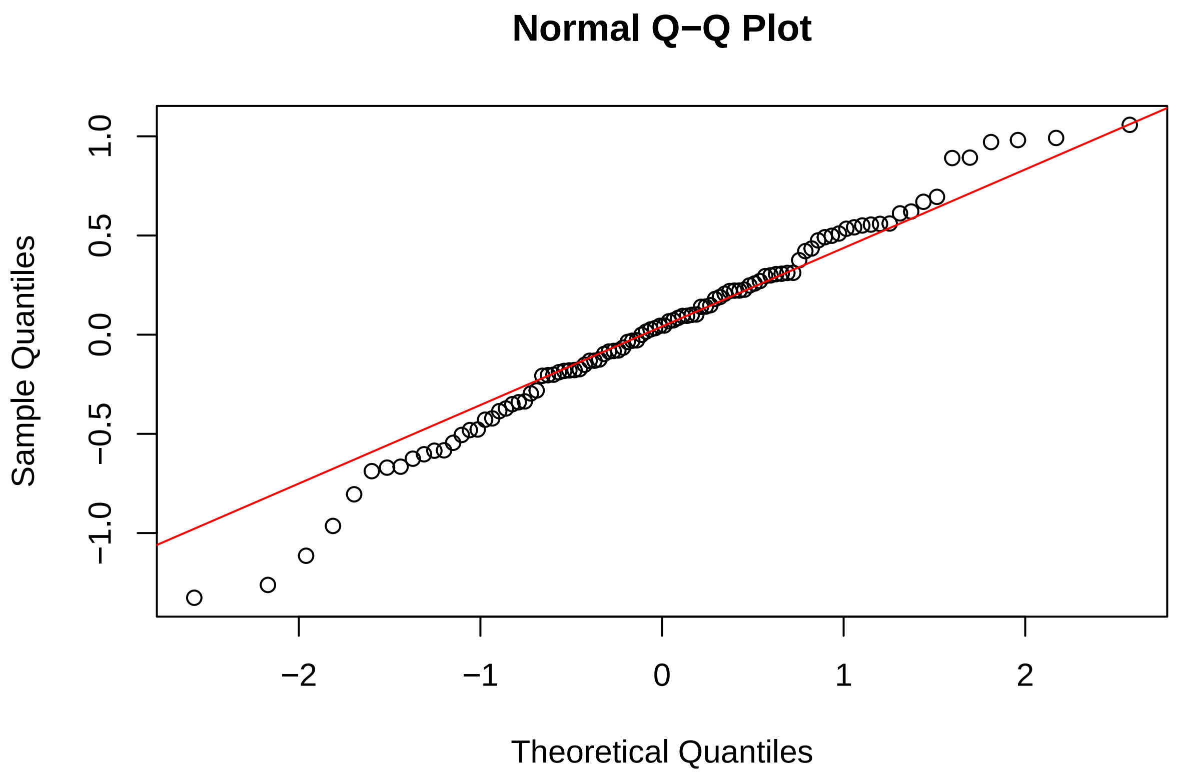
Histogram of residuals

```r
hist(residuals(model), main = "Histogram of Residuals", xlab = "Residuals")
```



Q-Q plot of residuals

```r
qqnorm(residuals(model))
qqline(residuals(model), col = "red")
```



```r
dev.off()
```

```
## null device
##           1
```

3. Cross-Validation (k-fold Cross-Validation)

Cross-validation helps assess how well the model generalizes to new data.

Set up k-fold cross-validation: Use the caret package for k-fold cross-validation. In this example, we will use 5-fold cross-validation.

In this context, k-fold cross-validation is used to assess how well the model

generalizes to unseen data. In k-fold cross-validation, the data is split into k equal subsets (or "folds"). The model is trained on k-1 folds and tested on the remaining fold. This process is repeated k times, each time using a different fold for testing, and the results are averaged to get an overall performance estimate.

#For instance, in 5-fold cross-validation:

The data is divided into 5 subsets. The model is trained on 4 of the subsets and tested on the remaining one. This process is repeated 5 times, each time using a different subset for testing. The performance metrics (like RMSE or R-squared) are averaged to provide a more reliable estimate of the model's generalizability.

Set up cross-validation with 5 folds

```r
train_control <- trainControl(method = "cv", number = 5)
```

Perform cross-validation using linear regression model

```r
cv_model <- train(Y ~ X1 + X2, data = data, method = "lm",
                  trControl = train_control)
```

View cross-validation results

```r
cv_model
```

```
## Linear Regression
##
## 100 samples
##   2 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 80, 80, 80, 80, 80
## Resampling results:
##
##   RMSE     Rsquared   MAE
##   19.5622  0.6911966  15.24633
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

3. Interpreting Cross-Validation Results

** RMSE (Root Mean Squared Error): Lower RMSE values indicate a better mode l because it means the predictions are closer to the actual values.

** b. R-squared (R^2)

$R^2 = 1$ means that the model perfectly explains the variance in Y. $R^2 = 0$ means that the model explains none of the variance in Y.

Summary of Steps:

Fit the model using the nls() function. Perform residual analysis (check for randomness and normality). Use k-fold cross-validation to assess generalizability. Calculate performance metrics (MSE, R-squared) to evaluate the model. Perform diagnostics (VIF for multicollinearity, overfitting checks). Refine the model if necessary based on performance.

Good luck