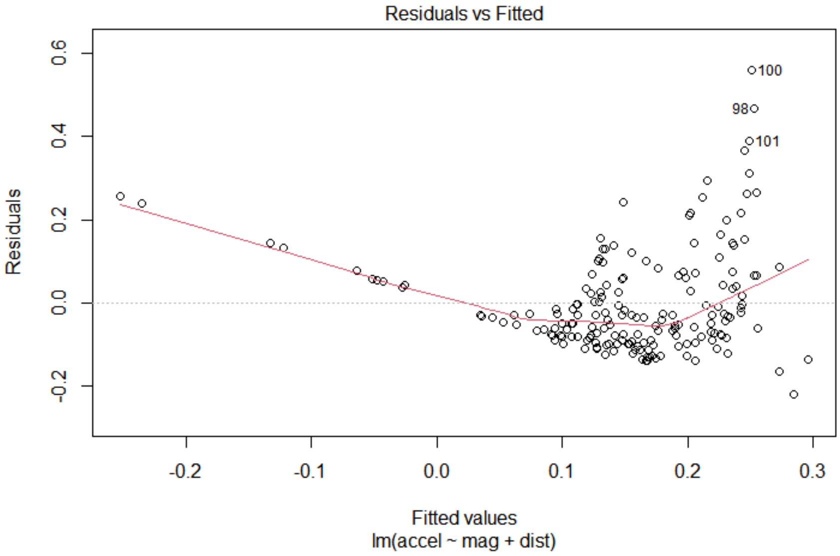
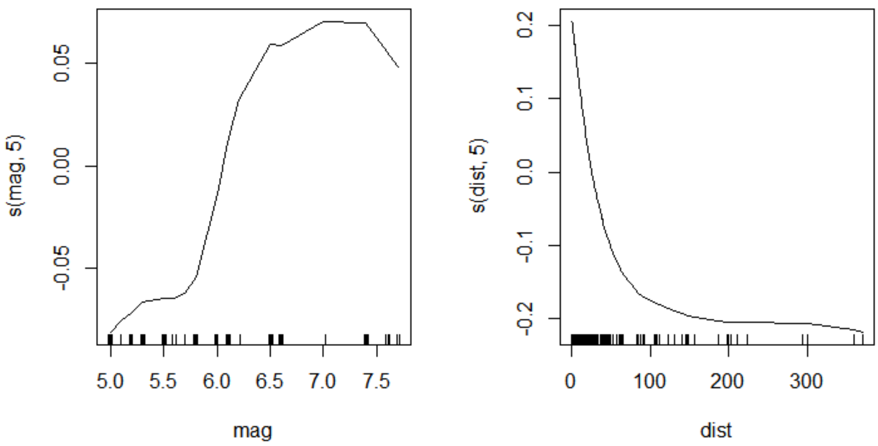
```
> ############################################
> # Statistical Modelling & Machine Learning #
> #             R Example1                   #
> ############################################
>
> ###################################################################
> # Modelling Example 1:  Nonlinear model with nonconstant variance #
> ###################################################################
>
> library(datasets)
> ?attenu
> dim(attenu)
[1] 182   5
>
> ##### Linear regression #####
> fit1 = lm(accel ~ mag + dist, data=attenu)
> plot(fit1)
```

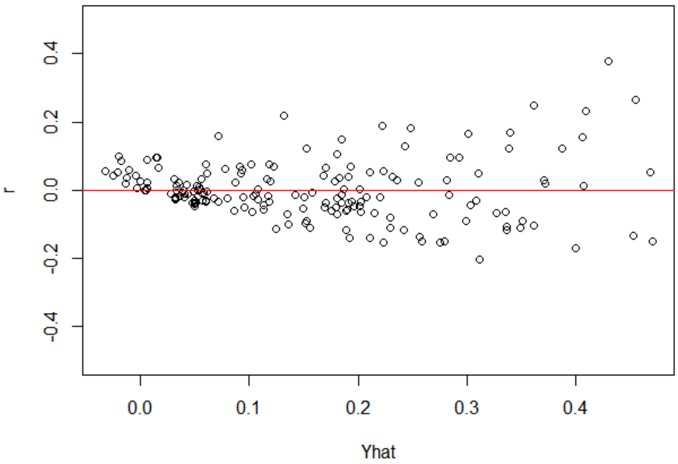



```
> ##### GAM #####
> library(gam)

> fit2 = gam(accel ~ s(mag,5) + s(dist,5),data=attenu)
> par(mfrow=c(1,2))
> plot(fit2)
```



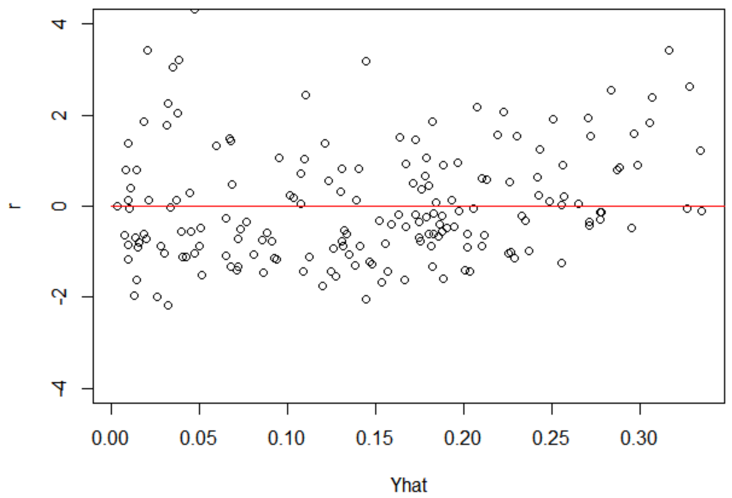
```
>
> fit2_1 = gam(accel ~ mag + s(dist,5),data=attenu)
> anova(fit2,fit2_1)
Analysis of Deviance Table

Model 1: accel ~ s(mag, 5) + s(dist, 5)
Model 2: accel ~ mag + s(dist, 5)
  Resid. Df Resid. Dev Df Deviance Pr(>Chi)
1       171     1.2960
2       175     1.4267 -4 -0.13065 0.001738 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
>
> # mag: linear function, dist: exponential function
>
> ########## Nonlinear Model with constant variance ##########
> # Y = accel, X1 = mag, X2 = dist.
> # Nonlinear model: Y = beta1 + beta2*X1 + beta3*X1^2 + beta4*exp(-beta5*X2).
>
> f = function(beta,X)
+ {
+   X1 = X[,1]; X2 = X[,2]
+   beta[1] + beta[2]*X1 + beta[3]*X1^2 + beta[4]*exp(-beta[5]*X2)
+ }
>
> # Objective function: RSS
> RSS = function(beta,Y,X) sum((Y-f(beta,X))^2)
>
> # Gradient vector of the objective function
> grv = function(beta,Y,X)
+ {
+   X1 = X[,1]; X2 = X[,2]
+   R = Y - f(beta,X)
+   c(-2*sum(R), -2*sum(R*X1), -2*sum(R*X1^2), -2*sum(R*exp(-beta[5]*X2)),
+     2*beta[4]*sum(R*X2*exp(-beta[5]*X2)))
+ }
>
> # Optimization
> X = cbind(attenu$mag,attenu$dist)
> colnames(X) = c('mag', 'dist')
> Y = attenu$accel
> ml1 = optim(rep(0.1,5), RSS, gr=grv, method='BFGS', X=X, Y=Y)
> ml1
$par
[1] -1.49610172  0.41779875 -0.02823150  0.45349508  0.04482179

$value
[1] 1.332736

$counts
function gradient
     121       37

$convergence
[1] 0

$message
NULL

>
> beta.hat = ml1$par
> beta.hat
[1] -1.49610172  0.41779875 -0.02823150  0.45349508  0.04482179
>
> # Fitted value
> Yhat = f(beta.hat,X)
>
```

```
> # Residual plot
> r = Y - Yhat
> par(mfrow=c(1,1))
> plot(Yhat,r,ylim=c(-0.5,0.5))
> lines(c(0,10),c(0,0),col='red')
> # Linearly increasing variance pattern.
```



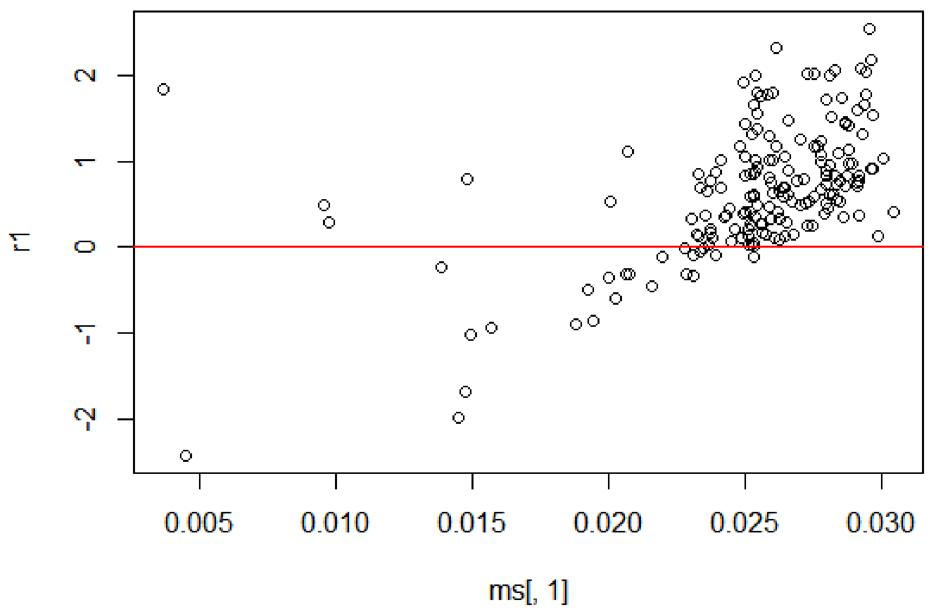

```
> ######### Nonlinear model with nonconstant variance ##########
>
> # To check whether a matrix is singular or not
> # install.packages('matrixcalc')
>
> # Objective function for mean function: Genearalized least square method.
> obj.mean = function(beta,Y,X,S) t(Y-f(beta,X)) %*% solve(S) %*% (Y-f(beta,X))
> # S: Covariance matrix
>
> # Gradient vector of the objective function
> gr.mean = function(beta,Y,X,S)
+ {
+   sigma2 = diag(S)
+   X1 = X[,1]; X2 = X[,2]
+   R = Y - f(beta,X)
+   c(-2*sum(R/sigma2), -2*sum(R*X1/sigma2), -2*sum(R*X1^2/sigma2),
+     -2*sum(R*exp(-beta[5]*X2)/sigma2),
+     2*beta[4]*sum(R*X2*exp(-beta[5]*X2)/sigma2))
+ }
>
> # Linear variance function: |r| = gam1 + gam2*Yhat.
> # For linear variance function, we can consider absolute residuals,
> # instead of squared residuals.
> # gam.hat = (Z^T W Z)^(-1) Z^T W |r|.
>
> beta.new = ml1$par     # initial parameter.
> W = diag(rep(1,length(Y)))
> mdif = 100000
>
> while(mdif > 0.000001)
+ {
+   Yhat = f(beta.new,X)
+   r = Y - Yhat
+   Z = cbind(1,Yhat)
+   gam.hat = solve(t(Z) %*% W %*% Z) %*% t(Z) %*% W %*% abs(r)
+   sigma = Z %*% gam.hat
+   S = diag(as.vector(sigma^2))
+
```

```
+   if (is.non.singular.matrix(S)) W = solve(S)
+   else W = solve(S + 0.000000001*diag(rep(1,nrow(S))))
+
+   ml2 = optim(beta.new, obj.mean, gr=gr.mean,method='BFGS', Y=Y, X=X, S=S)
+   beta.old = beta.new
+   beta.new = ml2$par
+   mdif = max(abs(beta.new - beta.old))
+ }
>
> beta.new
[1] -1.08473727  0.30740006 -0.02149087  0.33394053  0.02685877
>
> Yhat = f(beta.new,X)
> sigma = Z %*% gam.hat
> r = (Y - Yhat)/sigma
>
> # Residual plot
> plot(Yhat,r,ylim=c(-4,4))
> lines(c(0,10),c(0,0),col='red')
```



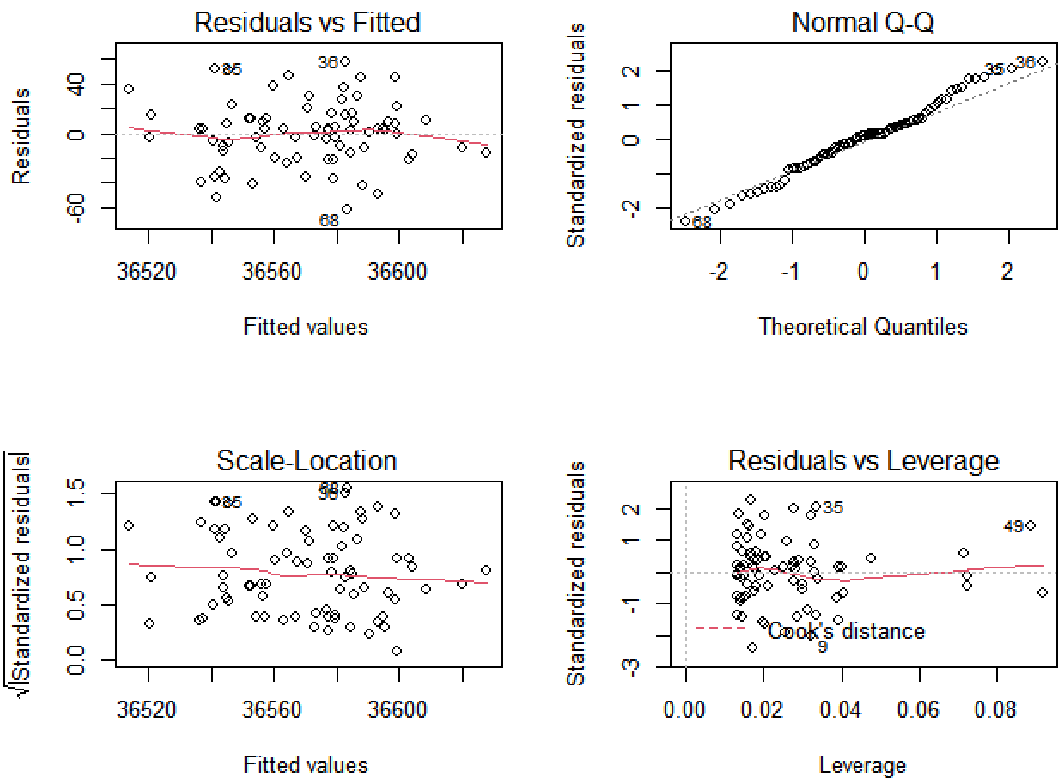

```
> ##### Linear regression with nonconstant variance #####
> # lmvar:
> # Linear mean function
> # Linear variance function: log(sigma) = X*beta
> # install.packages('lmvar')
> library(lmvar)
>
> X_s = cbind(attenu$mag, attenu$dist)
> colnames(X_s) = c('mag', 'dist')
> fit3 = lmvar(attenu$accel, X, X_s)
> summary(fit3)
Call:
 lmvar(y = attenu$accel, X_mu = X, X_sigma = X_s)

Number of observations:  182
Degrees of freedom    :  6

Z-scores:
    Min      1Q  Median      3Q     Max
-2.4255  0.2383  0.6385  1.0469  2.5417

Coefficients:
               Estimate  Std. Error  z value  Pr(>|z|)
(Intercept)   1.4106e-02 1.2999e-02   1.0852    0.2779
```

```
mag           2.4043e-03  1.7962e-03   1.3386    0.1807    
dist         -7.6248e-05  6.6991e-06 -11.3819 < 2.2e-16 ***
(Intercept_s) -4.1948e+00  4.9252e-01  -8.5171 < 2.2e-16 ***
mag_s         4.7234e-01  8.3854e-02   5.6328 1.773e-08 ***
dist_s       -2.1555e-02  9.7305e-04 -22.1514 < 2.2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Standard deviations:
   Min     1Q Median     3Q    Max 
0.0002 0.0843 0.1310 0.2018 0.3213 

Comparison to model with constant variance (i.e. classical linear model)
Log likelihood-ratio: 37.74527
Additional degrees of freedom: 2
p-value for difference in deviance: 4.05e-17
> 
> ms = predict(fit3, X_mu=X, X_sigma=X_s)
> r1 = (Y - ms[,1])/ms[,2]
> plot(ms[,1],r1)
> lines(c(-10,10),c(0,0),col='red')
```



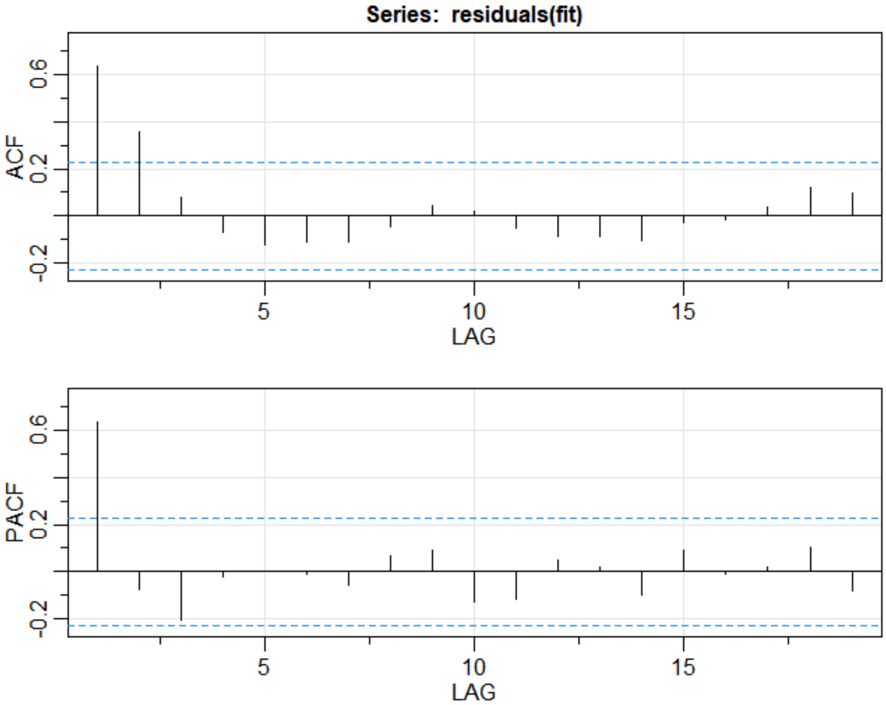

```
> ########################################################
> # Modelling Example 2:  Model with time correlations #
> ########################################################
> 
> tsdat = read.table('tsdat.txt',header=T)
> 
> fit = lm(y ~ x, data=tsdat)
> summary(fit)

Call:
lm(formula = y ~ x, data = tsdat)

Residuals:
    Min      1Q  Median      3Q     Max 
-61.998 -16.403   2.365  12.663  57.931 

Coefficients:
           Estimate Std. Error t value Pr(>|t|)    
```

```
(Intercept) 3.600e+04  7.152e+01 503.377  < 2e-16 ***
x           1.610e+00  2.023e-01   7.957 1.56e-11 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 25.83 on 74 degrees of freedom
Multiple R-squared:  0.4611,    Adjusted R-squared:  0.4538
F-statistic: 63.31 on 1 and 74 DF,  p-value: 1.565e-11


> par(mfrow=c(2,2))
> plot(fit)
```

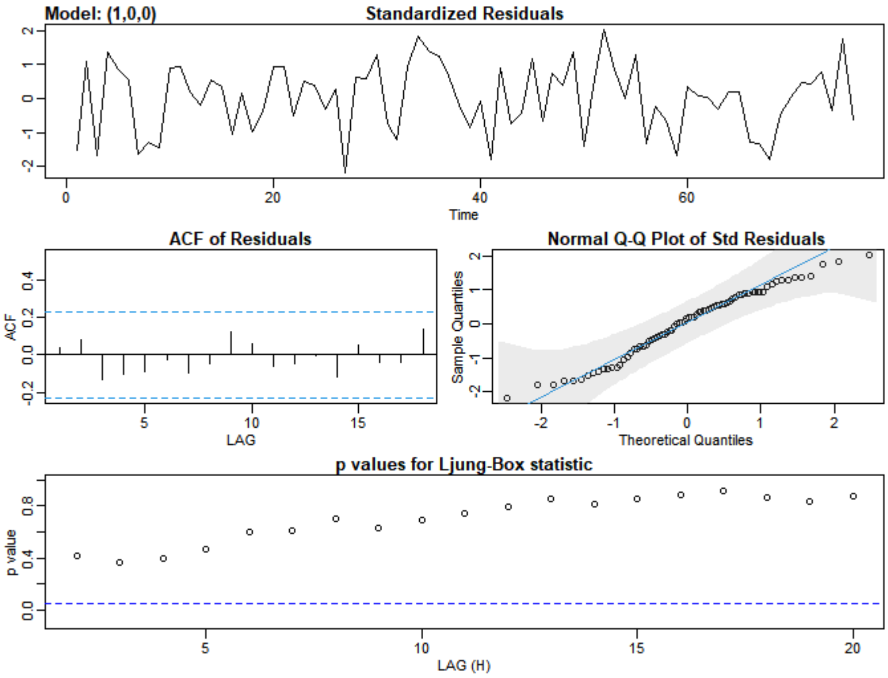



```
> # Durbin-Watson test
> # install.packages('lmtest')
> library(lmtest)
>
> dwtest(fit)

        Durbin-Watson test

data:  fit
DW = 0.69523, p-value = 4.72e-11
alternative hypothesis: true autocorrelation is greater than 0

>
> # Check ACF & PACF
> # install.packages('astsa')
> library(astsa)
>
> # AR(p): ACF: Exponentially decreasing; PACF: Non-zero values at first p lags.
> # MA(q): ACF: Non-zero values at first q lags; PACF: Exponentially decreasing.
> # ARMA(p,q): ACF: Similar to ACF of AR(p); PACF: Similar to PACF of MA(q).
>
> acf2(residuals(fit))
    [,1]  [,2]  [,3]  [,4]  [,5]  [,6]  [,7]  [,8] [,9] [,10] [,11] [,12] [,13] [,14] [,15]
ACF  0.64  0.36  0.08 -0.07 -0.12 -0.11 -0.11 -0.04 0.04  0.02 -0.05 -0.08 -0.09  -0.1 -0.03
```

```
PACF 0.64 -0.07 -0.20 -0.02  0.00 -0.01 -0.06  0.06 0.09 -0.13 -0.12  0.05  0.02  -0.1  0.09
     [,16] [,17] [,18] [,19]
ACF  -0.01  0.04  0.12  0.10
PACF -0.01  0.02  0.10 -0.08
```




```
> ar1 = sarima (residuals(fit), 1,0,0, no.constant=T)   #AR(1)
initial  value 3.229004
iter   2 value 2.958676
iter   3 value 2.958511
iter   4 value 2.958366
iter   4 value 2.958366
final  value 2.958366
converged
initial  value 2.971232
iter   2 value 2.971136
iter   3 value 2.971118
iter   3 value 2.971118
iter   3 value 2.971118
final  value 2.971118
converged
> ar1$fit

Call:
stats::arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D,
    Q), period = S), xreg = xmean, include.mean = FALSE, transform.pars = trans,
    fixed = fixed, optim.control = list(trace = trc, REPORT = 1, reltol = tol))

Coefficients:
         ar1
      0.6488
s.e.  0.0875

sigma^2 estimated as 378.1:  log likelihood = -333.64,  aic = 671.29
```

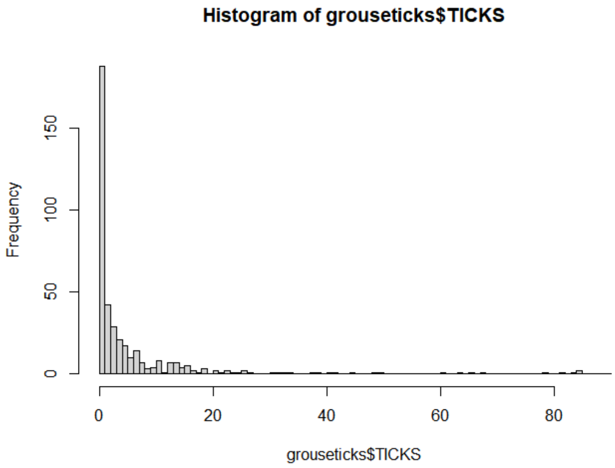



```
> # MLE: Multivariate normal distribution
> X = cbind(1,tsdat$x)
> Y = tsdat$y
> n = length(Y)
> S = diag(rep(1,n))    # initial covariance matrix
>
> mdif = 1000000
> beta.old = rep(100000,2)
> I = 0
>
> while(mdif > 0.0000001)
+ {
+   beta.new = as.vector(solve(t(X) %*% solve(S) %*% X) %*%t(X) %*% solve(S) %*% Y)
+   r = as.vector(Y - (X %*% beta.new))
+   ar1 = sarima (r, 1,0,0, no.constant=T, details=F)
+   alpha = ar1$fit$coef
+   sigma2 = ar1$fit$sigma2
+
+   mdif = max(abs(beta.new - beta.old))
+   beta.old = beta.new
+   I = I + 1
+   # Construct covariance matrix
+   S = matrix(nrow=n,ncol=n)
+   for (i in 1:n)
+   {
+     for (j in 1:n)
+     {
+       if (i == j) S[i,j] = 1
+       if (i != j) S[i,j] = alpha^(abs(i-j))
+     }
+   }
+   S = (sigma2 / (1-alpha^2)) * S
+ }
>
> round(beta.new,4)
[1] 35986.2865     1.6521
> # MLE: Product of conditional distribution (Approximation)
> # Y_t | Y_t-1 ~ N(X_t*beta + alpha*epsilon_t-1, sigma^2)
>
> fit = lm(y ~ x, data=tsdat)
>
> Yt = tsdat$y[2:n]
```

```
> Xt = tsdat$x[2:n]
> et = residuals(fit)[1:(n-1)]
> mdif = 10000
> b.old = rep(0,3)
>
> while(mdif > 0.0000001)
+ {
+   fit.temp = lm(Yt ~ Xt + et)
+   b.new = fit.temp$coefficient
+   mdif = max(abs(b.new[1:2] - b.old[1:2]))
+
+   et = (Y - X %*% b.new[1:2])[1:(n-1)]
+   b.old = b.new
+ }
>
> round(b.new,4)
(Intercept)         Xt         et
 35999.7941     1.6205     0.6379

> # Built-in function
> # cochrane.orcutt => f: linear model, error: AR(p) process.
> # install.packages("orcutt")
> library(orcutt)
>
> fit = lm(y ~ x, data=tsdat)
> cochrane.orcutt(fit)
Cochrane-orcutt estimation for first order autocorrelation

Call:
lm(formula = y ~ x, data = tsdat)

 number of interaction: 4
 rho 0.63843

Durbin-Watson statistic
(original):    0.69523 , p-value: 4.72e-11
(transformed): 1.83115 , p-value: 2.515e-01

 coefficients:
 (Intercept)          x
35994.746022    1.634731


> ##########################################################
> # Modelling Example 3:  Model with spatial correlations #
> ##########################################################
>
> # install.packages('spdep')
> library(spdep)
>
> data(oldcol)
>
> ?COL.OLD
> # 'COL.nb' has the neighbors list.
>
> # 2-D Coordinates of observations
> crds = cbind(COL.OLD$X, COL.OLD$Y)
>
> # Compute the maximum distance
> mdist = sqrt(sum(diff(apply(crds, 2, range))^2))
>
> # All obs. between 0 and mdist are identified as neighborhoods.
> dnb = dnearneigh(crds, 0, mdist)
>
> # Compute Euclidean distance between obs.
> dists = nbdists(dnb, crds)
>
> # Compute Power distance weight d^(-2)
> glst = lapply(dists, function(d) d^(-2))
```

```
>
> # Construct weight matrix with normalization
> # style='C': global normalization; 'W': row normalization
> lw = nb2listw(dnb, glist=glst, style='C')
>
> # Spatial Autoregressive Model
> fit = lagsarlm(CRIME ~ HOVAL + INC, data=COL.OLD, listw=lw)
> summary(fit)

Call:lagsarlm(formula = CRIME ~ HOVAL + INC, data = COL.OLD, listw = lw)

Residuals:
      Min        1Q    Median        3Q       Max
-25.93357  -4.98320  -0.50733   4.96160  25.17053

Type: lag
Coefficients: (asymptotic standard errors)
             Estimate Std. Error z value  Pr(>|z|)
(Intercept) 46.04321    5.01268  9.1853 < 2.2e-16
HOVAL       -0.23848    0.07684 -3.1036  0.001912
INC         -0.85776    0.27588 -3.1092  0.001876

Rho: 0.24802, LR test value: 25.318, p-value: 4.8614e-07
Asymptotic standard error: 0.038854
    z-value: 6.3834, p-value: 1.7316e-10
Wald statistic: 40.748, p-value: 1.7316e-10

Log likelihood: -174.7182 for lag model
ML residual variance (sigma squared): 72.254, (sigma: 8.5002)
Number of observations: 49
Number of parameters estimated: 5
AIC: 359.44, (AIC for lm: 382.75)
LM test for residual autocorrelation
test value: 3.0704, p-value: 0.079732

> # install.packages('spatialreg')
> library(spatialreg)
>
> # Fitted values
> predict(fit)
This method assumes the response is known - see manual page
           fit     trend    signal
1001 21.139363 17.202962  3.936402
1002 40.669968 34.285530  6.384438
1003 36.930992 27.465263  9.465730
1004 26.111840 20.919545  5.192295
1005 12.853113 10.100641  2.752472
1006 30.589971 26.072868  4.517103
1007 38.576177 30.853021  7.723156
1008 29.365348 25.437905  3.927443
1009 47.328746 33.281923 14.046823
1010 15.294460 11.390062  3.904399
1011 31.907170 27.682394  4.224776
1012 20.555831 16.513986  4.041845
----<omitted>--------------------

> ####################################################
> # Modelling Example 4:  Generalized Linear Models #
> ####################################################
>
> ########## Cumulative logit model ##########
> # install.packages('ordinal')
> library(ordinal)
>
> ?wine
>
> ?clm
>
> fit = clm(rating ~ temp + contact, data=wine, link = 'logit')
```

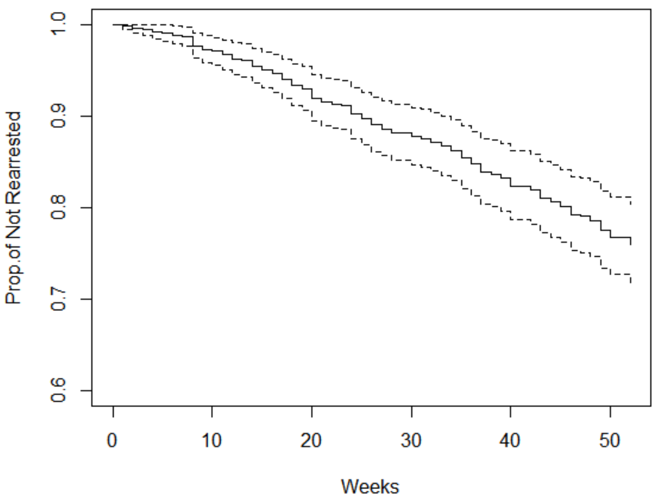
```
> summary(fit)
formula: rating ~ temp + contact
data:    wine

 link  threshold nobs logLik AIC    niter max.grad cond.H 
 logit flexible  72   -86.49 184.98 6(0)  4.02e-12 2.7e+01

Coefficients:
           Estimate Std. Error z value Pr(>|z|)    
tempwarm     2.5031     0.5287   4.735 2.19e-06 ***
contactyes   1.5278     0.4766   3.205  0.00135 ** 
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Threshold coefficients:
    Estimate Std. Error z value
1|2  -1.3444     0.5171  -2.600
2|3   1.2508     0.4379   2.857
3|4   3.4669     0.5978   5.800
4|5   5.0064     0.7309   6.850
> 
> 
> ########## Poisson regression model ##########
> 
> # For data
> # install.packages('lme4')
> library(lme4)

> 
> data(grouseticks)
> 
> ?grouseticks
> 
> head(grouseticks)
  INDEX TICKS BROOD HEIGHT YEAR LOCATION   cHEIGHT
1     1     0   501    465   95       32  2.759305
2     2     0   501    465   95       32  2.759305
3     3     0   502    472   95       36  9.759305
4     4     0   503    475   95       37 12.759305
5     5     0   503    475   95       37 12.759305
6     6     3   503    475   95       37 12.759305
> 
> hist(grouseticks$TICKS,breaks=0:90)
```




```
> 
> fit = glm(TICKS ~ HEIGHT*YEAR, data = grouseticks, family=poisson)
> summary(fit)

Call:
glm(formula = TICKS ~ HEIGHT * YEAR, family = poisson, data = grouseticks)
```

```
Deviance Residuals:
    Min       1Q   Median       3Q      Max
-6.0993  -1.7956  -0.8414   0.6453  14.1356

Coefficients:
                Estimate Std. Error z value Pr(>|z|)
(Intercept)    27.454732   1.084156   25.32   <2e-16 ***
HEIGHT         -0.058198   0.002539  -22.92   <2e-16 ***
YEAR96        -18.994362   1.140285  -16.66   <2e-16 ***
YEAR97        -19.247450   1.565774  -12.29   <2e-16 ***
HEIGHT:YEAR96   0.044693   0.002662   16.79   <2e-16 ***
HEIGHT:YEAR97   0.040453   0.003590   11.27   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 5847.5  on 402  degrees of freedom
Residual deviance: 3009.0  on 397  degrees of freedom
AIC: 3952

Number of Fisher Scoring iterations: 6

>
>
> ########## Negative binomial regression model ##########
> library(MASS)
>
> fit1 = glm.nb(TICKS ~ HEIGHT*YEAR, data = grouseticks, link=log)
> summary(fit1)

Call:
glm.nb(formula = TICKS ~ HEIGHT * YEAR, data = grouseticks, link = log,
    init.theta = 0.9000852793)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.3765  -1.0281  -0.5052   0.2408   3.2440

Coefficients:
                Estimate Std. Error z value Pr(>|z|)
(Intercept)    20.030124   1.827525  10.960  < 2e-16 ***
HEIGHT         -0.041308   0.004033 -10.242  < 2e-16 ***
YEAR96        -10.820259   2.188634  -4.944 7.66e-07 ***
YEAR97        -10.599427   2.527652  -4.193 2.75e-05 ***
HEIGHT:YEAR96   0.026132   0.004824   5.418 6.04e-08 ***
HEIGHT:YEAR97   0.020861   0.005571   3.745 0.000181 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for Negative Binomial(0.9001) family taken to be 1)

    Null deviance: 840.71  on 402  degrees of freedom
Residual deviance: 418.82  on 397  degrees of freedom
AIC: 1912.6

Number of Fisher Scoring iterations: 1


              Theta:  0.9001
          Std. Err.:  0.0867

 2 x log-likelihood:  -1898.5880

> ########## Proportional hazard model ##########
> library(survival)
>
> # For data
```

```
> # install.packages('carData')
> library(carData)
>
> ?Rossi
>
> Surv(Rossi$week, Rossi$arrest)
  [1] 20  17  25  52+ 52+ 52+ 23  52+ 52+ 52+ 52+ 52+ 37  52+ 25  46  28  52+ 52+ 52+ 52+
 [22] 52+ 24  52+ 52+ 52+ 52+ 52+ 52+ 52+ 52+ 52+ 52+ 52+ 52+ 50  52+ 52+ 52+ 52+ 52+ 52+
 [43] 10  52+ 52+ 52+ 52+ 20  52+ 52+ 52+ 52+ 52+ 50  52+ 52+ 52+ 52+ 52+ 52+  6  52+ 52+
 [64] 52+ 52  52+ 52+ 52+ 49  52+ 52+ 52+ 52+ 52+ 52+ 52+ 43  52+ 52+  5  27  52+ 52+ 52+
 [85] 22  52+ 52+ 18  52+ 52+ 52+ 52+ 52+ 52+ 52+ 24  52+ 52+ 52+ 52+  2  26  52+ 49  52+
[106] 21  48  52+ 52+ 52+ 52+ 52+ 52+ 52+ 52+ 52+ 52+ 52+  8  52+ 52+ 49  52+ 52+ 52+ 52+
[127] 52+ 52+ 52+ 52+  8  52+ 52+ 13  52+ 52+ 52+ 52+  8  52+ 33  52+ 52+ 11  52+ 52+ 52+
[148] 52+ 52+ 37  52+ 52+ 52+ 44  52+ 52  52+ 52+ 52+ 52+ 52+ 52+ 52+ 52+ 52+ 52+ 52+ 52+
[169] 52+ 52+ 52+ 52+  9  17  52+ 52+ 52+ 52+ 52+ 52+ 16  52+ 52+  3  52+ 52+ 52+ 52+ 52+
[190] 52+ 52+ 52+ 52+ 52+ 45  52+ 52+ 52+ 52+ 52+ 52+ 28  52+ 16  15  52+ 52+ 52+ 52+ 14
[211] 52+ 52+ 52+ 52+ 52+ 52+ 52+ 52+ 52+ 52+ 52+ 52+ 52+ 52+ 52+  7  52+ 52+ 43  46  40
[232] 52+ 14  52+ 52+  8  52+ 52+ 52+ 52+ 52+ 25  52+ 52+ 17  37  52+ 52+ 52+ 32  52+ 52+
[253] 52+ 52+ 52+ 52+ 52+ 52+ 12  52+ 18  52+ 52+ 14  52+ 52+ 52+ 52+ 38  52+ 24  20  32
[274] 52+ 52+ 52+ 52+ 52+ 52+ 52+ 52+ 52+ 52+ 52+ 31  20  40  52+ 52+ 52+ 52+ 42  52+ 26
[295] 52+ 52+ 52+ 52+ 52+ 47  52+ 52+ 40  52+ 52+ 52+ 52+ 21  52+ 52+ 52+ 52+ 52+  1  43
[316] 24  11  52+ 52+ 52+ 52+ 33  52+ 46  36  52+ 52+ 18  52+ 52+ 50  52+ 34  52+ 35  52+
[337] 52+ 39   9  52+ 52+ 52+ 34  52+ 52+ 52+ 44  52+ 52+ 35  30  39  52+ 52+ 52+ 52+ 19
[358] 52+ 43  52+ 48  37  20  52+ 52+ 36  52+ 52+ 52+ 52+ 52+ 52+ 52+ 30  52+ 52+ 52+ 52+
[379] 52+ 52+ 52+ 52+ 42  52+ 52+ 52+ 52+ 26  40  52+ 52+ 52+ 35  52+ 46  52+ 49  52+ 52+
[400] 49  52+ 52+ 52+ 52+ 52+ 35  52+ 52+ 52+ 52+ 27  52+ 52+ 52+ 52  45   4  52  36  52+
[421] 52+  8  15  52+ 19  52+ 12  52+ 52+ 52+ 52+ 52+
>
> fit = coxph(Surv(week,arrest) ~ fin + age+ race + wexp + mar + prio,
+            data=Rossi)
> summary(fit)
Call:
coxph(formula = Surv(week, arrest) ~ fin + age + race + wexp +
    mar + prio, data = Rossi)

  n= 432, number of events= 114

                   coef exp(coef) se(coef)      z Pr(>|z|)
finyes         -0.37352   0.68831  0.19082 -1.957 0.050295 .
age            -0.05640   0.94516  0.02184 -2.583 0.009796 **
raceother      -0.30983   0.73357  0.30780 -1.007 0.314133
wexpyes        -0.15331   0.85786  0.21218 -0.723 0.469957
marnot married  0.44339   1.55799  0.38136  1.163 0.244958
prio            0.09336   1.09785  0.02832  3.296 0.000981 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

               exp(coef) exp(-coef) lower .95 upper .95
finyes            0.6883     1.4528    0.4735    1.0005
age               0.9452     1.0580    0.9056    0.9865
raceother         0.7336     1.3632    0.4013    1.3410
wexpyes           0.8579     1.1657    0.5660    1.3003
marnot married    1.5580     0.6419    0.7378    3.2898
prio              1.0979     0.9109    1.0386    1.1605

Concordance= 0.642  (se = 0.027 )
Likelihood ratio test= 33.08  on 6 df,   p=1e-05
Wald test            = 32.01  on 6 df,   p=2e-05
Score (logrank) test = 33.43  on 6 df,   p=9e-06

>

> # Estimated survival function
> plot(survfit(fit),ylim=c(0.6,1),xlab="Weeks", ylab="Prop.of Not Rearrested")
```




```
> # Estimated survival function
> plot(survfit(fit),ylim=c(0.6,1),xlab="Weeks", ylab="Prop.of Not Rearrested")
> hist(grouseticks$TICKS,breaks=0:90)
> # Estimated survival function
> plot(survfit(fit),ylim=c(0.6,1),xlab="Weeks", ylab="Prop.of Not Rearrested")
> # Estimated survival functions for financial aid
> Rossi.fin = with(Rossi, data.frame(fin=c(0, 1), age=rep(mean(age), 2),
+                              race=rep(mean(race=='other'),2),
+                              wexp=rep(mean(wexp=="yes"),2),
+                              mar=rep(mean(mar=="not married"),2),
+                              prio=rep(mean(prio),2)))
>
> plot(survfit(fit,newdata=Rossi.fin), conf.int=TRUE,
+     lty=c(1, 2), ylim=c(0.6, 1), col=c('red','blue'),
+     xlab="Weeks", ylab="Prop. of Not Rearrested")
>
> legend("bottomleft", legend=c("fin = no","fin = yes"),
+       lty=c(1 ,2),col=c('red','blue'))
```