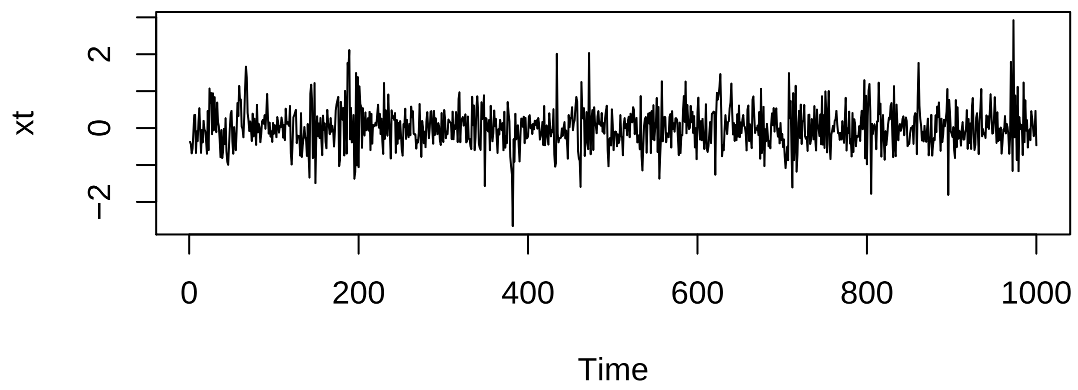
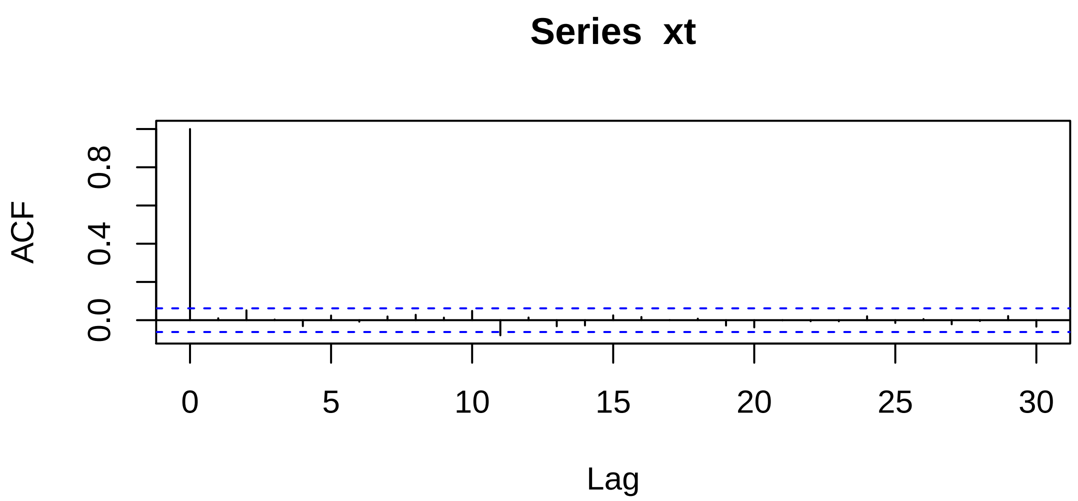
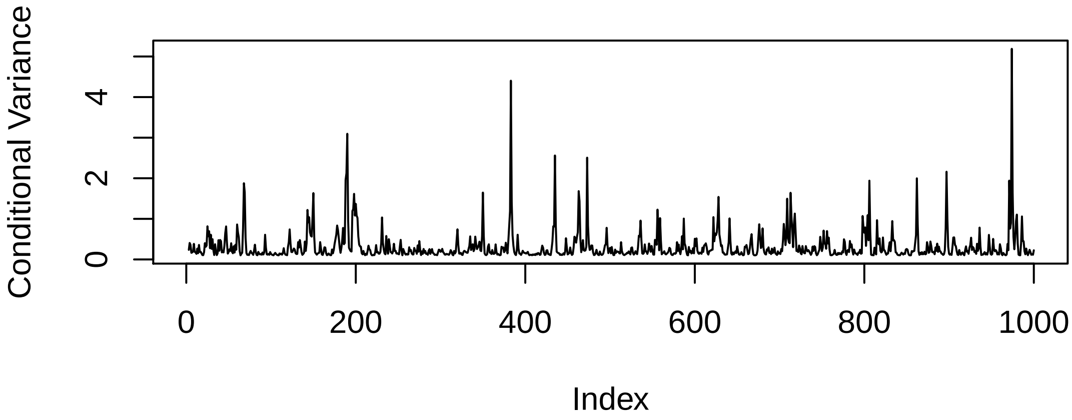
```r
# zad.1

library("tseries")

# a)

xt <- numeric(1100)
zt <- rnorm(1100)
alfa0 <- 0.1
alfa1 <- 0.5
alfa2 <- 0.2

xt[1:2] <- rnorm(2,0,alfa0/(1-alfa1-alfa2))

skwt <- numeric(1100)
for(i in 3:1100){
  skwt[i] <- alfa0 + alfa1*xt[i-1]^2 + alfa2*xt[i-2]^2
  xt[i] <- sqrt(skwt[i])*zt[i]
}

xt <- xt[101:1100]
ts.plot(xt)
```




```r
acf(xt)
```



```r
# b)

Box.test(xt,lag=20,type="Ljung")

##
##  Box-Ljung test
##
## data:  xt
## X-squared = 19.78, df = 20, p-value = 0.4717

# nie wykrywa zaleznosci ARCH!!!!!!!! bo on wykrywa tylko zaleznosc linowa
Box.test(xt^2,lag=20,type="Ljung")

##
##  Box-Ljung test
##
## data:  xt^2
## X-squared = 146.7, df = 20, p-value < 2.2e-16

# a tu jest zaleznosc nielinowa! i tu juz wykrywa, dlatego, gdy chcemy spr,
# czy jest efekt arch lub garch warto testowac kwadraty

# c)

mod <- arima(xt^2,c(2,0,0))
Box.test(mod$residuals,lag=20,type="Ljung")

##
##  Box-Ljung test
##
## data:  mod$residuals
## X-squared = 14.92, df = 20, p-value = 0.7808

# a teorerycznie powinien byc ar(2), wiec jest ok :D

# d)

arch <- garch(xt,order=c(0,2),trace=FALSE)    # order(garch, arch) stopien
summary(arch)$coef   # mniej wiecej wychodza wartosci teoretyczne

##     Estimate  Std. Error  t value Pr(>|t|)
## a0    0.1006     0.01021    9.851 0.00e+00
## a1    0.5675     0.06852    8.283 2.22e-16
## a2    0.1774     0.04238    4.187 2.83e-05

# e)

logLik(arch)  # logarytm wiarogodnosci

## 'log Lik.' -692.1 (df=3)

AIC(arch)

## [1] 1390

# f)

fit <- fitted(arch)
plot(fit[,1]^2,type="l",ylab="Conditional Variance")
```




```
# wykres warunkowej wariancji
# jaka mielismy wariancje w czasie t pod warunkiem wczesniejszego momentu

# g)

ga <- garch(xt,order=c(1,1),trace=FALSE)
summary(ga)

##
## Call:
## garch(x = xt, order = c(1, 1), trace = FALSE)
##
## Model:
## GARCH(1,1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -3.09777 -0.70481 -0.00749  0.67592  3.99037
##
## Coefficient(s):
##     Estimate  Std. Error  t value Pr(>|t|)
## a0    0.0808      0.0125     6.46  1.1e-10 ***
## a1    0.5891      0.0733     8.03  8.9e-16 ***
## b1    0.2038      0.0576     3.54  0.00041 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Diagnostic Tests:
##  Jarque Bera Test
##
## data:  Residuals
## X-squared = 4.895, df = 2, p-value = 0.08653
##
##
##  Box-Ljung test
##
## data:  Squared.Residuals
## X-squared = 2.423, df = 1, p-value = 0.1196

# jaque bera test -> test na noramlnosc reziduow
```

```r
# ljunga boxa dla kwadratow reziduow
summary(ga)$coef
```

```
##      Estimate  Std. Error  t value  Pr(>|t|)
## a0   0.08079     0.01251    6.459 1.053e-10
## a1   0.58911     0.07333    8.033 8.882e-16
## b1   0.20378     0.05762    3.536 4.055e-04
```

```r
# h)

AIC(ga)
```

```
## [1] 1396
```

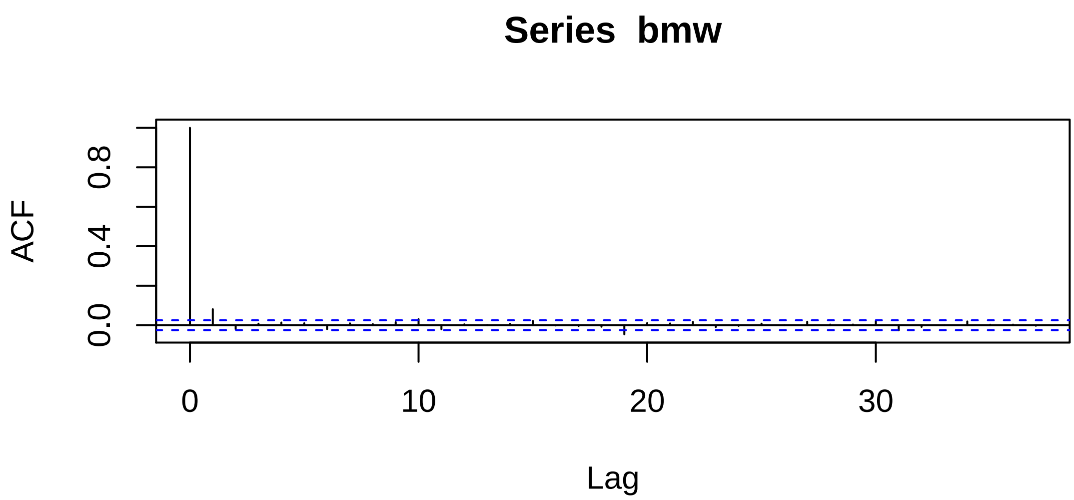
```r
AIC(arch)
```

```
## [1] 1390
```

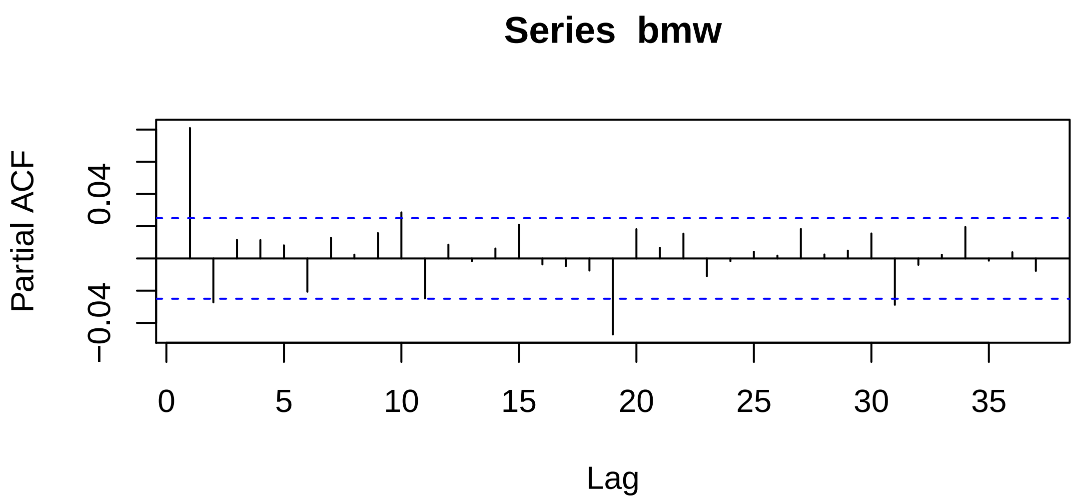
```r
# lepszy jest ten, ktory ma mniejsza wartosc AIC


# zad.3

# O nieadekwatnosci modelowania zwrotow na
# podstawie liniowych modeli autoregresyjnych

library("evir")

data(bmw,package="evir")
# bmw - wektor zwrotow logarytmicznych

head(bmw)
```

```
## [1]  0.047704  0.007127  0.008883 -0.012441 -0.003570  0.000000
```

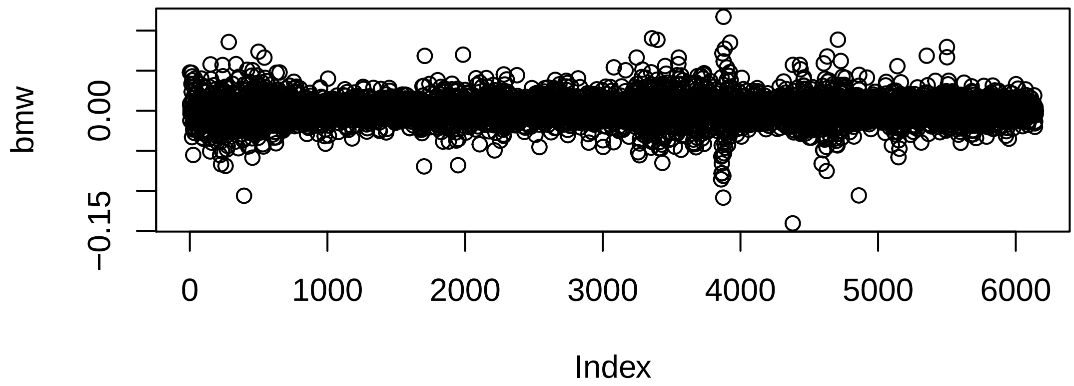
```r
bmw <- as.vector(bmw)
n <- length(bmw)

acf(bmw)
```



```r
pacf(bmw)
```



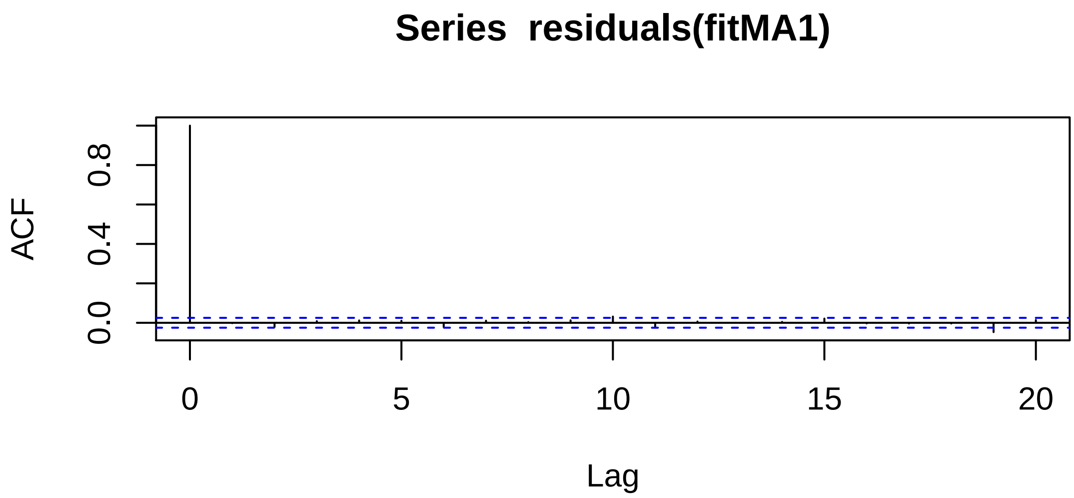

```r
plot(bmw)
```



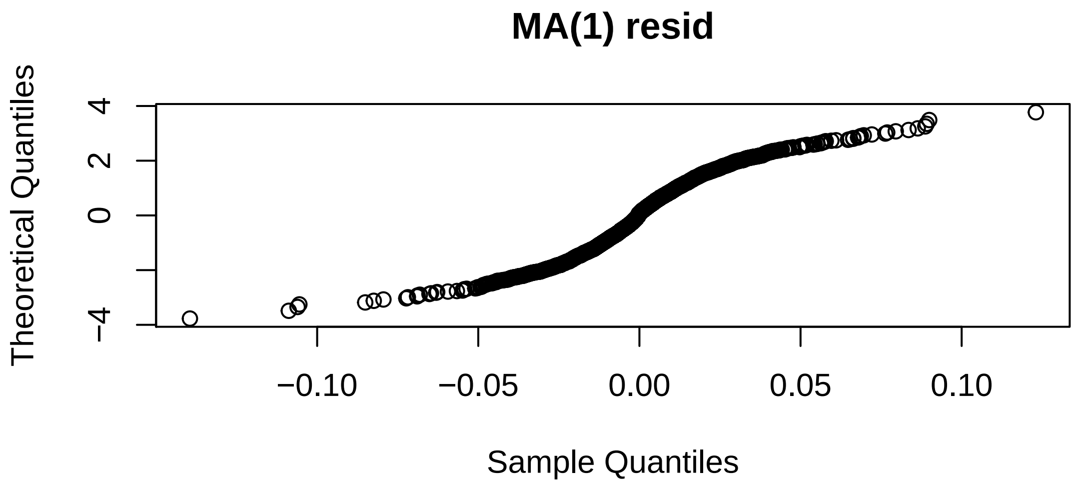

```r
for (p in 0:3) {
  for (q in 0:3) {
    a <- AIC(arima(bmw,c(p,0,q)),k=log(n))
    print(c(p,q,a))
  }
}
```

```
## [1]      0      0 -34367
## [1]      0      1 -34401
## [1]      0      2 -34395
## [1]      0      3 -34386
## [1]      1      0 -34399
## [1]      1      1 -34395
## [1]      1      2 -34386
## [1]      1      3 -34378
## [1]      2      0 -34394
## [1]      2      1 -34386
## [1]      2      2 -34377
## [1]      2      3 -34369
## [1]      3      0 -34386
```

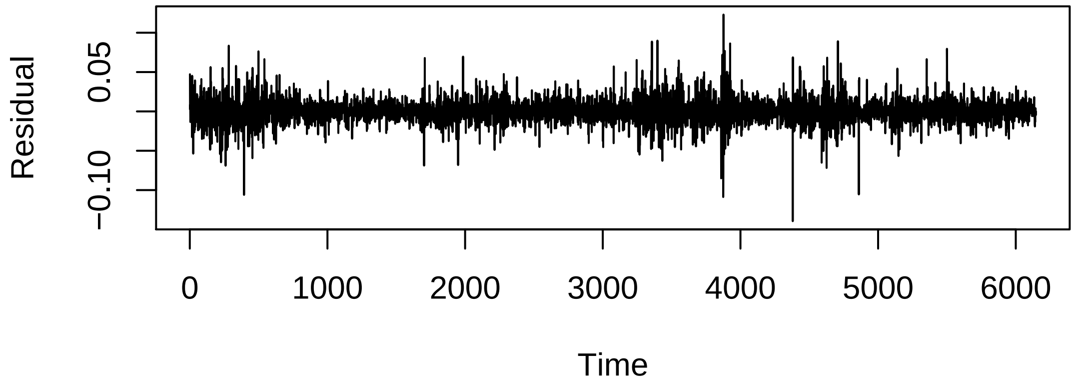
```
## [1]      3      1 -34378
## [1]      3      2 -34369
## [1]      3      3 -34361

# metoda na oko - minimum sugeruje model MA(1)

fitMA1 <- arima(bmw, order = c(0,0, 1))

Box.test(fitMA1$resid,lag=20,type="Ljung")

##
##  Box-Ljung test
##
## data:  fitMA1$resid
## X-squared = 37.31, df = 20, p-value = 0.01074

Box.test(fitMA1$resid^2,lag=20,type="Ljung")  # wskazuje na szereg ARCH

##
##  Box-Ljung test
##
## data:  fitMA1$resid^2
## X-squared = 1274, df = 20, p-value < 2.2e-16

acf( residuals(fitMA1),lag.max=20)
```




```
qqnorm(residuals(fitMA1),datax=T,main="MA(1) resid")
```

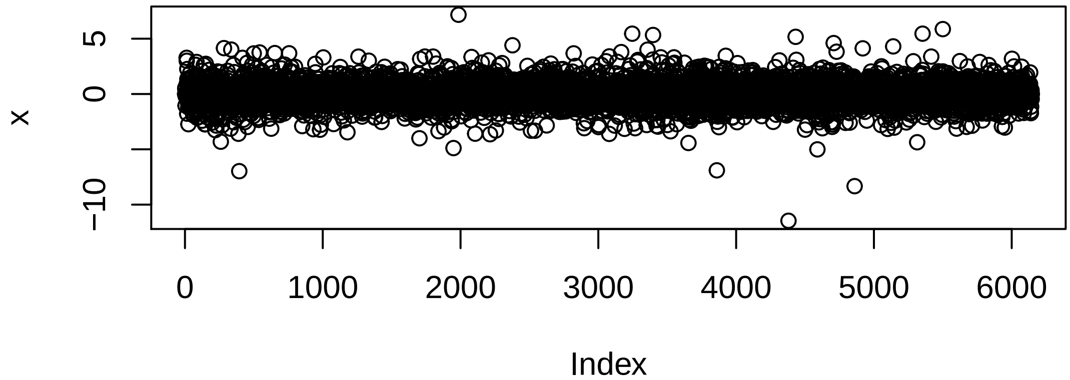



```r
plot(residuals(fitMA1),ylab="Residual")
```

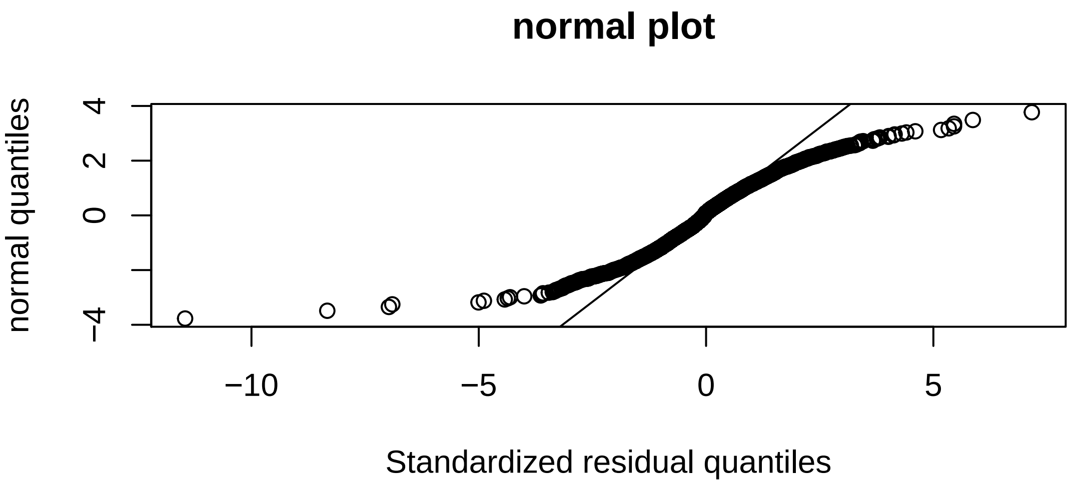



```r
# rozklad rezyduów nie jest normalny
# widoczne skupienia zmienności => rezydua są zależne


# dopasowujemy model MA(1) + rezydua GARCH(1,1), rozklad warunkowy normalny

library("fGarch")

bmw.garch_norm <- garchFit(~arma(0,1)+garch(1,1), data=bmw,
                           cond.dist="norm",trace=FALSE)
# dopasuje arma do danych a garch do residuow
summary(bmw.garch_norm)  # ljung box -> to Q() oznacza jaki lag

##
## Title:
##  GARCH Modelling
##
## Call:
##  garchFit(formula = ~arma(0, 1) + garch(1, 1), data = bmw, cond.dist = "norm",
##     trace = FALSE)
##
## Mean and Variance Equation:
```

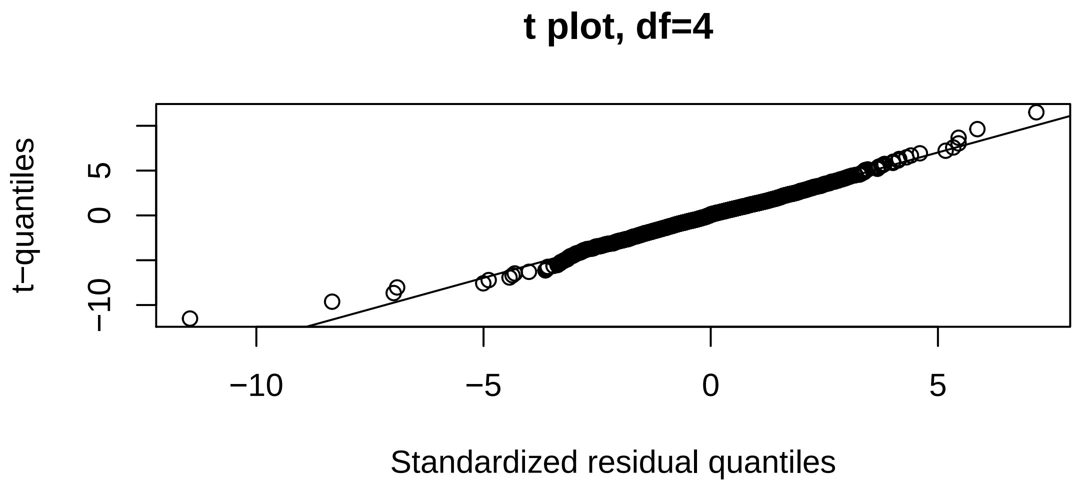
```
##  data ~ arma(0, 1) + garch(1, 1)
## <environment: 0x00000000091da668>
##  [data = bmw]
##
## Conditional Distribution:
##  norm
##
## Coefficient(s):
##         mu         ma1       omega      alpha1       beta1
## 4.4430e-04  1.0023e-01  8.9488e-06  1.0251e-01  8.5886e-01
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu     4.443e-04   1.738e-04    2.556   0.0106 *
## ma1    1.002e-01   1.443e-02    6.946 3.76e-12 ***
## omega  8.949e-06   1.453e-06    6.160 7.28e-10 ***
## alpha1 1.025e-01   1.139e-02    9.003  < 2e-16 ***
## beta1  8.589e-01   1.585e-02   54.193  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  17757    normalized:  2.889
##
## Description:
##  Wed Jun 11 13:07:39 2014 by user: Marta
##
##
## Standardised Residuals Tests:
##                                 Statistic p-Value
##  Jarque-Bera Test   R    Chi^2  11330     0
##  Shapiro-Wilk Test  R    W      NA        NA
##  Ljung-Box Test     R    Q(10)  14.79     0.1398
##  Ljung-Box Test     R    Q(15)  19.78     0.1806
##  Ljung-Box Test     R    Q(20)  30.22     0.06643
##  Ljung-Box Test     R^2  Q(10)  5.054     0.8875
##  Ljung-Box Test     R^2  Q(15)  7.528     0.9413
##  Ljung-Box Test     R^2  Q(20)  9.264     0.9796
##  LM Arch Test       R    TR^2   6.053     0.9134
##
## Information Criterion Statistics:
##    AIC    BIC    SIC   HQIC
## -5.777 -5.771 -5.777 -5.775

# wykres kwantylowy dla rezyduów

x <- bmw.garch_norm@residuals / bmw.garch_norm@sigma.t
plot(x)
```




```
qqnorm(x,datax=T,ylab= "Standardized residual quantiles",
       main=" normal plot", xlab="normal quantiles")
qqline(x,datax=T)
```




```
# duze odstepstwa od normalnosci - grubsze ogony.
# Dopasowujemy rozklad t do rezyduow.
# Wykonujemy qqplot  dla rozkladu t o 4 st. sw.

grid = (1:n)/(n+1)
qqplot(sort(x), qt(grid,df=4),
       main= " t plot, df=4",xlab= "Standardized residual quantiles",
       ylab="t-quantiles")
abline(   lm(   qt(c(.25,.75),df=4)~quantile(x,c(.25,.75))    )    )
```




```
# zmieniamy rozklad warunkowy na t

bmw.garch_t <- garchFit(~arma(0,1)+garch(1,1),cond.dist="std",
                        data=bmw,trace=FALSE)

options(digits=4)
summary(bmw.garch_t)  # parametr shape-> stopnie swobody.

##
## Title:
##  GARCH Modelling
##
## Call:
##  garchFit(formula = ~arma(0, 1) + garch(1, 1), data = bmw, cond.dist = "std",
##     trace = FALSE)
##
## Mean and Variance Equation:
##  data ~ arma(0, 1) + garch(1, 1)
## <environment: 0x000000000a1aae90>
##  [data = bmw]
##
## Conditional Distribution:
##  std
##
## Coefficient(s):
##          mu          ma1        omega       alpha1        beta1        shape
## 1.3083e-04  6.8514e-02  6.0813e-06  9.3850e-02  8.8599e-01  4.0557e+00
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##          Estimate  Std. Error  t value Pr(>|t|)
## mu      1.308e-04   1.439e-04    0.909    0.363
## ma1     6.851e-02   1.293e-02    5.300 1.16e-07 ***
## omega   6.081e-06   1.349e-06    4.508 6.56e-06 ***
## alpha1  9.385e-02   1.322e-02    7.101 1.24e-12 ***
## beta1   8.860e-01   1.548e-02   57.223  < 2e-16 ***
## shape   4.056e+00   2.327e-01   17.428  < 2e-16 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  18158    normalized:  2.954
##
## Description:
##  Wed Jun 11 13:07:41 2014 by user: Marta
##
##
## Standardised Residuals Tests:
##                                 Statistic p-Value
##  Jarque-Bera Test   R    Chi^2  13383     0
##  Shapiro-Wilk Test  R    W      NA        NA
##  Ljung-Box Test     R    Q(10)  21.38     0.01859
##  Ljung-Box Test     R    Q(15)  25.92     0.03889
##  Ljung-Box Test     R    Q(20)  36.18     0.01465
##  Ljung-Box Test     R^2  Q(10)  5.801     0.8317
##  Ljung-Box Test     R^2  Q(15)  8.157     0.9173
##  Ljung-Box Test     R^2  Q(20)  10.77     0.9519
##  LM Arch Test       R    TR^2   7.008     0.8571
##
## Information Criterion Statistics:
##    AIC    BIC    SIC   HQIC
## -5.907 -5.900 -5.907 -5.905
```

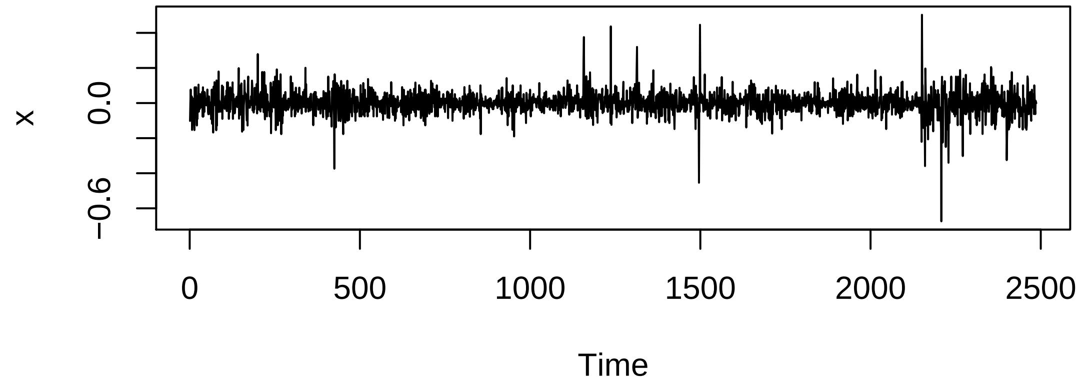
```
loglik_bmw <- bmw.garch_t@fit$llh # -loglik dla modelu bmw.garch_t

BIC_bmw_t <- 2*loglik_bmw+log(n)*6
as.numeric(BIC_bmw_t) # wartość kryterium BIC dla tego modelu
```

```
## [1] -36263
```

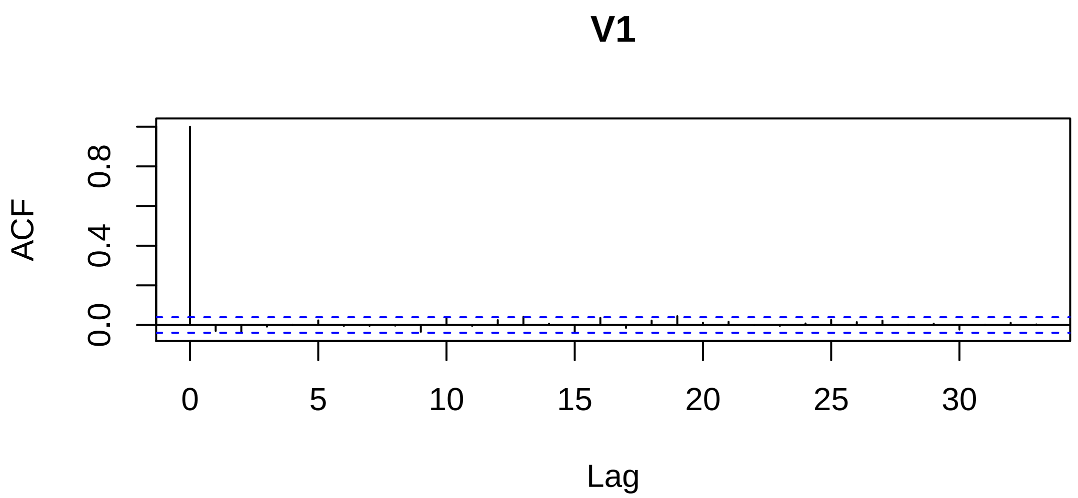
```
# lepiej (mniej) niz w modelu ma cos dopasowanym na poczatku tego zadania

# zad.2

# 1)

x <- read.table("http://gamma.mini.pw.edu.pl/~szymanowskih/lab6/exch1.txt")
head(x,2)
```

```
##       V1
## 1 -0.102
## 2  0.000
```

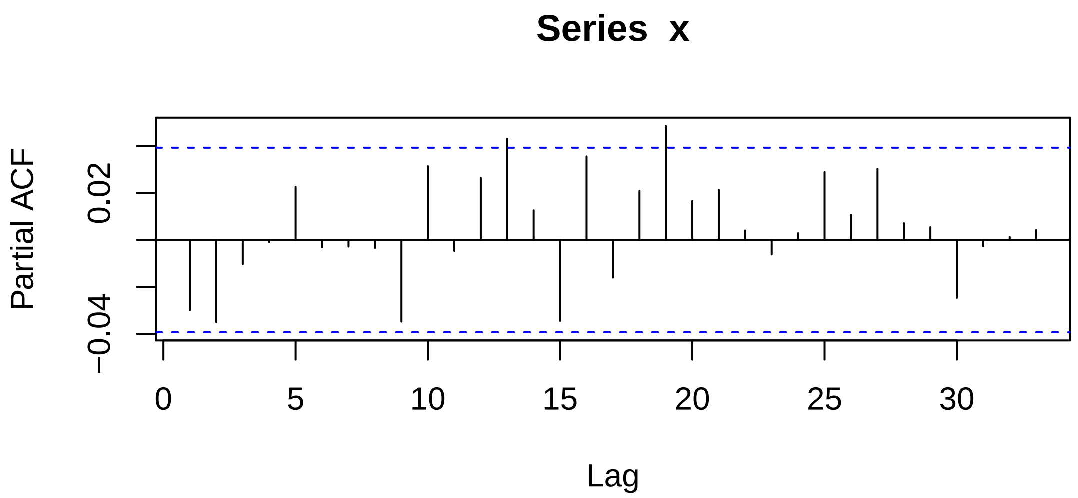
```
ts.plot(x)
```




```
acf(x)
```

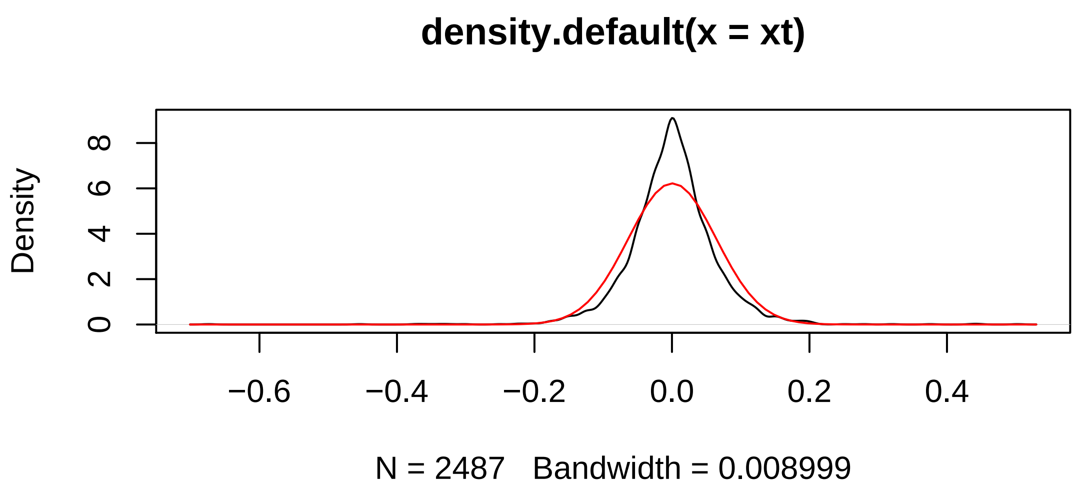



```
pacf(x)
```




```
Box.test(x,lag=20,type="Ljung")

##
##  Box-Ljung test
```

```
##
## data:  x
## X-squared = 32.24, df = 20, p-value = 0.04079

Box.test(x^2,lag=20,type="Ljung")

##
##  Box-Ljung test
##
## data:  x^2
## X-squared = 94.2, df = 20, p-value = 1.356e-11

# te powyzsze testy to bardzo charakterystyczna rzecza dla efektu arch
# dla zwyklych danych nic sie nie dzieje, a dla kwadratow jest juz problem

xt <- as.numeric(as.matrix(x))

# 2)

plot(density(xt))
curve(dnorm(x,mean(xt),sd(xt)),add=T,col="red")
```




```
# 4)

for(i in 0:3){
  for(j in 1:3){
    a <- AIC(garch(xt,order=c(i,j),trace=FALSE),k=log(length(xt)))
    print(c(i,j,a))
  }
}

## [1]     0     1 -6845
## [1]     0     2 -6851
## [1]     0     3 -6901
## [1]     1     1 -6959
## [1]     1     2 -6947
## [1]     1     3 -6863
## [1]     2     1 -6977
## [1]     2     2 -6907
## [1]     2     3 -6927
```

```
## [1]      3     1 -6971
## [1]      3     2 -6956
## [1]      3     3 -6894

# szukam minimum
```