Title

code - courseName

Christian Oppegård Moen

dd-mm-yyyy

Contents

GARCH(p,q)

Definitions for notational purposes:

- $\mathcal{R}_1 = (r_1, ..., r_{max(p,q)})$
- $\mathcal{R}_2 = (r_t, ..., r_{t+max(p,q)})$
- $\boldsymbol{\alpha} = (\alpha_0, ..., \alpha_p)$
- $\boldsymbol{\beta} = (\beta_1, ..., \beta_q)$
- $\boldsymbol{r}_{tp} = (1, r_{t-1}, ..., r_{t-p})^\top$
- $\boldsymbol{\sigma}^2_{tq} = (\sigma^2_{t-1}, ..., \sigma^p_{t-q})^\top$

GARCH(p, q) is given by

$$r_t = \sigma_t \epsilon_t, \tag{1}$$

$$\sigma_t^2 = \alpha_0 + \sum_{j=1}^{p} \alpha_j r_{t-j}^2 + \sum_{j=1}^{q} \beta_j \sigma_{t-j}^2. \tag{2}$$

$$\sigma_t^2 = \boldsymbol{\alpha} \boldsymbol{r}_{tp}^2 + \boldsymbol{\beta} \boldsymbol{\sigma}_{tq}^2. \tag{3}$$

We know that $r_t|\mathcal{R}_2 \sim \mathcal{N}(0, \sigma_t^2)$

The log likelihood $l(\boldsymbol{\alpha}, \boldsymbol{\beta}|\mathcal{R}_1) \propto -\ln L(\boldsymbol{\alpha}, \boldsymbol{\beta}|\mathcal{R}_1)$ is given by

$$l(\boldsymbol{\alpha}, \boldsymbol{\beta}|\mathcal{R}_1) = \sum_{t=m+1}^{n} \ln(\sigma_t^2)) + \frac{r_t^2}{\sigma_t^2} \tag{4}$$

In the following chunk we implement the so called critical function.

```
garchLL = function(params, r, p = 1, q = 0) {

    n = length(r)
    m = max(p, q)

    alpha = exp(params[1:(p + 1)])
    if (q == 0) {
        beta = 0
    } else {
        beta = exp(params[-(1:(p + 1))])
    }

    # initialize variance and set first m values
    sigma_sq = numeric(n)
    sigma_sq[1:m] = t(alpha) %*% c(1, r[p:1]^2)  # should they be zero?
    # sigma_sq[1:m] = 0 # says so above eq. (5.52)

    # Iteratively compute each variance
    for (t in (m + 1):n) {
        sigma_sq[t] = sum(alpha * c(1, r[(t - 1):(t - p)]^2)) + sum(beta * sigma_sq[(t -
            1):(t - q)])
```

```
    }
    ll = sum(log(sigma_sq) + r^2/sigma_sq)
    return(ll)
}
```

The variance σ_t^2 can be estimated by *one-step-ahead* forecasting given by

$$\hat{\sigma}_t^2 = \hat{\boldsymbol{\alpha}} \boldsymbol{r}_{tp}^2 + \hat{\boldsymbol{\beta}} \hat{\boldsymbol{\sigma}}_{tq}^2 \tag{5}$$

```
sigmaForecast = function(mod) {
    n = mod$n
    m = mod$m
    p = mod$p
    r = mod$r

    alpha = mod$estim[1:(p + 1)]
    if (mod$q == 0)
        {
            beta = 0
            q = 1
        }  # If ARCH(p) model
 else {
        beta = mod$estim[-(1:(p + 1))]
        q = mod$q
    }

    sf = numeric(n)  # sigma squared forecasts
    # sf[1:p] = sum(alpha*c(1,r[p:1]^2)) sf[1:q] = 0 sf[1:m] =
    # alpha[1]/(1-sum(c(alpha, beta))) # GPT
    sf[1:m] = sum(alpha * c(1, r[1:p]^2))  # used in sim test

    for (t in (m + 1):n) {
        sf[t] = sum(alpha * c(1, r[(t - 1):(t - p)]^2)) + sum(beta * sf[(t - 1):(t -
            q)])
        if (is.na(sf[t])) {
            browser()
        }
    }
    return(sf)
}

# TODO: make garch function
garch = function(r, p = 1, q = 0, init = c(0.1, 0.1)) {
    n = length(r)

    alpha = init[1:(p + 1)]
    if (q == 0)
        {
            beta = 0
            q = 1
        }  # If ARCH(p) model
 else {
        beta = init[-(1:(p + 1))]
```

```
        q = q
    }

    estim = exp(optim(par = log(init), fn = garchLL, r = r, p = p, q = q, method = "BFGS")$par)

    mod = list(p = p, q = q, n = n, m = max(p, q), init = init, estim = estim, r = r)

    mod$sigmaForecast = sigmaForecast(mod)
    return(mod)

}

modResults = function(mod, main = "", plot = T) {
    # print(rbind(estim = mod$estim, init = mod$init))
    if (plot) {
        plot(mod$r, type = "l", main = main)
        lines(mod$sigmaForecast, col = "cyan")
    }
}
```

Tests

```
testModel = function(mod) {
    n = mod$n
    m = mod$m
    p = mod$p

    alpha = mod$paramSim[1:(p + 1)]
    if (mod$q == 0) {
        beta = 0
        q = 1
    } else {
        beta = mod$paramSim[-(1:(p + 1))]
        q = mod$q
    }

    r = numeric(n)
    r[1:m] = rnorm(m)
    sigma_sq = numeric(n)
    sigma_sq[1:m] = sum(alpha * c(1, r[1:p]^2))
    # sigma_sq[1:m] = alpha[1]/(1-sum(c(alpha, beta))) # GPT

    # browser()
    for (t in (m + 1):n) {
        sigma_sq[t] = sum(alpha * c(1, r[(t - 1):(t - p)]^2)) + sum(beta * sigma_sq[(t -
            1):(t - q)])
        r[t] = rnorm(1, sd = sqrt(sigma_sq[t]))
    }
    # browser() estimation
    estim = exp(optim(par = log(mod$init), fn = garchLL, r = r, p = p, q = q, method = "BFGS")$par)
    comparison = (rbind(real = mod$paramSim, estim = estim, init = mod$init))
```

```
    return(list(r = r, estim = estim, comparison = comparison))
}
```

```
set.seed(420)
arch1 = list(p = 1, q = 0, m = 1, n = 10000, init = rep(0.1, 2), paramSim = c(0.01,
    0.2))


testResult = testModel(arch1)
testResult$comparison
```

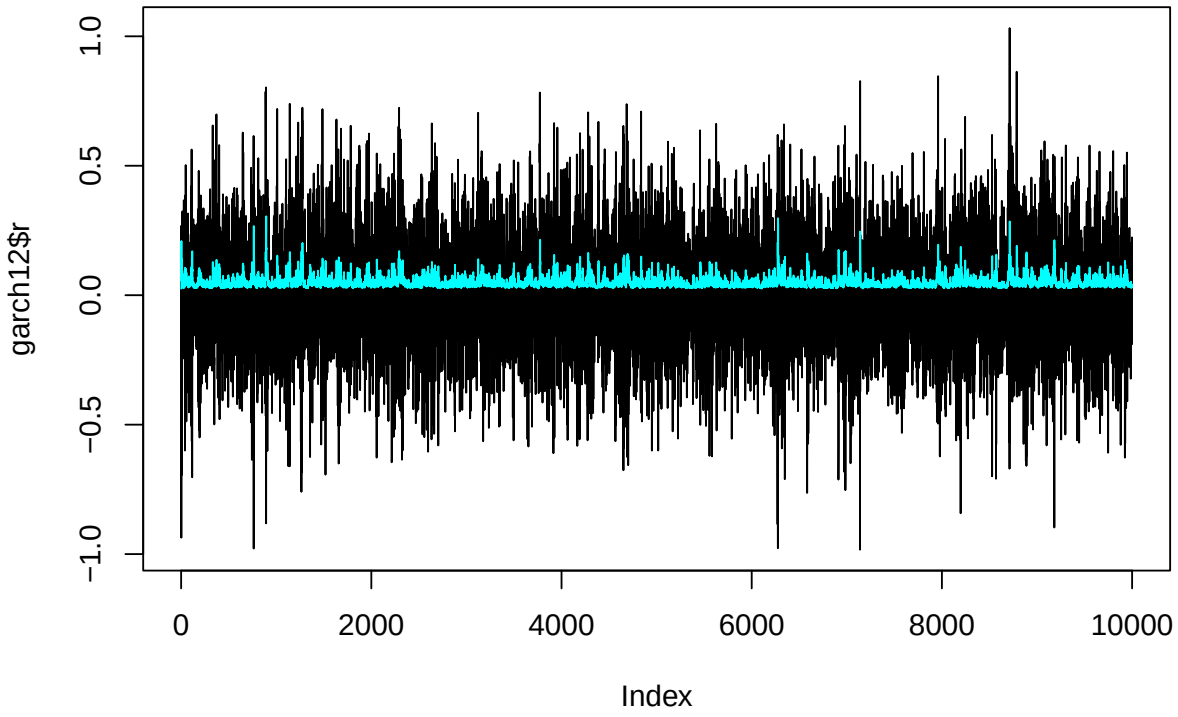
```
##              [,1]      [,2]
## real  0.010000000 0.2000000
## estim 0.009775681 0.1997736
## init  0.100000000 0.1000000
```

```
# qqnorm(testResult$r) plot(testResult$r)
```

```
set.seed(420)
garch12 = list(p = 1, q = 2, m = 2, n = 10000, init = rep(0.1, 4), paramSim = c(0.01,
    0.2, 0.1, 0.5))


testResult = testModel(garch12)
garch12$estim = testResult$estim
round(testResult$comparison, 3)
```

```
##       [,1] [,2]  [,3]  [,4]
## real  0.01 0.20 0.100 0.500
## estim 0.01 0.21 0.112 0.464
## init  0.10 0.10 0.100 0.100
```

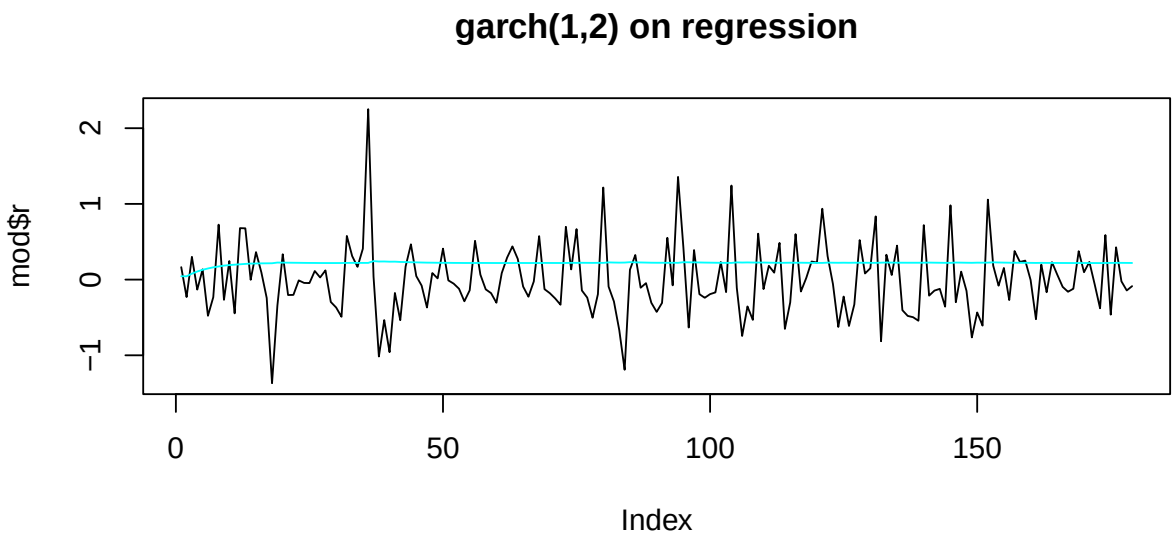
```
garch12$r = testResult$r
garch12$sigmaForecast = sigmaForecast(garch12)
plot(garch12$r, type = "l")
lines(garch12$sigmaForecast, col = "cyan")
```

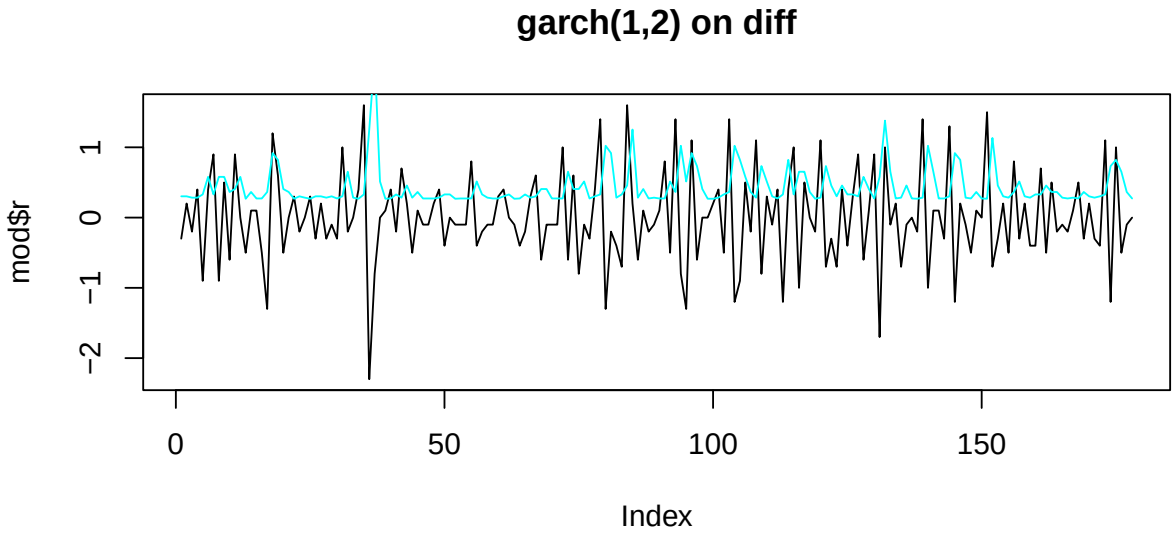


Real data (HOW YOU FIT IT AND VIEW IT)

```
df <- read.csv("projectdata.csv", header = T, sep = ";", dec = ",", stringsAsFactors = FALSE)


# qqnorm(r) plot(r, type = 'l')
```

```
# make data to fit
fit = lm(Inflation ~ Unemployed + Consumption + InterestRate, data = df)
r = df$Inflation - fit$fitted.values
d = diff(df$Inflation)
par(mfrow = c(2, 1))
# inflation - regression
garch12regr = garch(r, p = 1, q = 2, init = rep(0.1, 1 + 2 + 1))
modResults(garch12regr, main = "garch(1,2) on regression")

# diff inflation
garch12 = garch(d, p = 1, q = 2, init = rep(0.1, 1 + 2 + 1))
modResults(garch12, "garch(1,2) on diff")
```

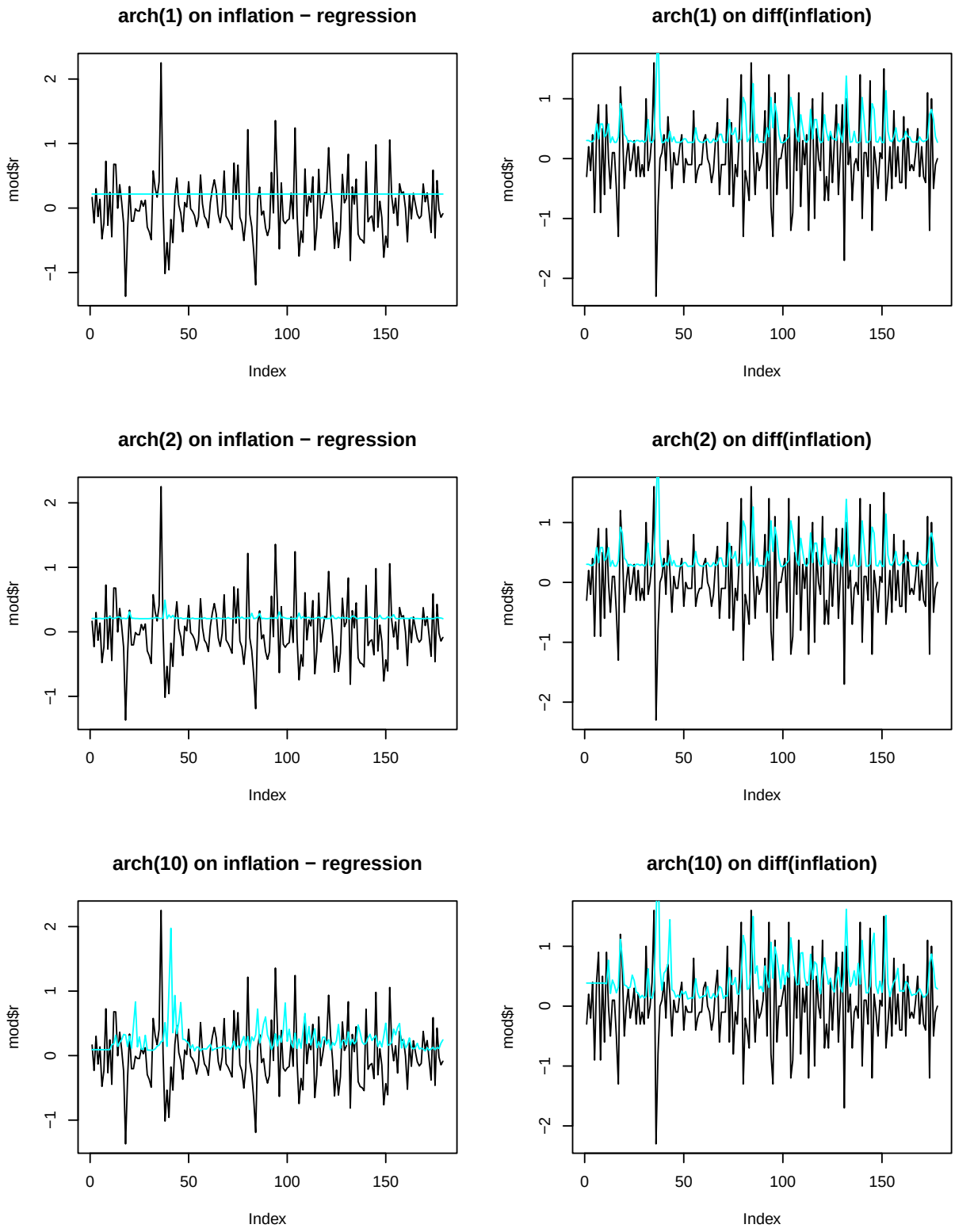





```
par(mfrow = c(1, 1))
```

```
par(mfrow = c(3, 2))
arch1regr = garch(r, p = 1, q = 0, init = rep(0.1, 2))
modResults(arch1regr, main = "arch(1) on inflation - regression")

arch1d = garch(d, p = 1, q = 0, init = rep(0.1, 2))
modResults(arch1d, main = "arch(1) on diff(inflation)")

arch2regr = garch(r, p = 2, q = 0, init = rep(0.1, 3))
modResults(arch2regr, main = "arch(2) on inflation - regression")

arch2d = garch(d, p = 2, q = 0, init = rep(0.1, 3))
```

```r
modResults(arch2d, "arch(2) on diff(inflation)")

arch10regr = garch(r, p = 10, q = 0, init = rep(0.1, 11))
modResults(arch10regr, main = "arch(10) on inflation - regression")

arch10d = garch(d, p = 10, q = 0, init = rep(0.1, 11))
modResults(arch10d, "arch(10) on diff(inflation)")
```




```
par(mfrow = c(1, 1))
```

```r
list(arch1regr = arch1regr$estim, arch1d = arch1d$estim, arch2regr = arch2regr$estim,
    arch2d = arch2d$estim, arch10regr = arch10regr$estim, arch10d = arch10d$estim)
```

```
## $arch1regr
## [1] 2.172051e-01 2.631814e-05
##
## $arch1d
## [1] 0.2681048 0.3845943
##
## $arch2regr
## [1] 2.040528e-01 2.842043e-05 5.669071e-02
##
## $arch2d
## [1] 2.673872e-01 3.877460e-01 9.556793e-05
##
## $arch10regr
##  [1] 6.607910e-02 1.507028e-06 7.553767e-02 4.280618e-05 1.562249e-01
##  [6] 3.652726e-01 2.839493e-07 8.483807e-02 2.002727e-06 6.598773e-07
## [11] 1.301979e-01
##
## $arch10d
##  [1] 1.143832e-01 4.845580e-01 5.315562e-08 2.358752e-08 5.056129e-02
##  [6] 6.419528e-02 5.766002e-08 2.064997e-01 3.922694e-09 4.332273e-13
## [11] 4.898945e-08
```