ARCH Models in Julia

Simon A. Broda

University of Zurich and University of Amsterdam

simon.broda@uzh.ch

Introduction

- Daily financial returns data exhibit a number of *stylized facts*:

    - Volatility clustering

    - Non-Gaussianity, fat tails

    - Leverage effects: negative returns increase future volatility

- Other types of data (e.g., changes in interest rates) exhibit similar phenomena.

- These effects are important in many areas in finance, in particular in risk management.

- [G]ARCH ([**G**eneralized] **A**utoregressive **C**onditional **V**olatility) models are the most popular for modelling them.

Example: volatility clustering in AAPL returns

Example: fat tails in AAPL return density
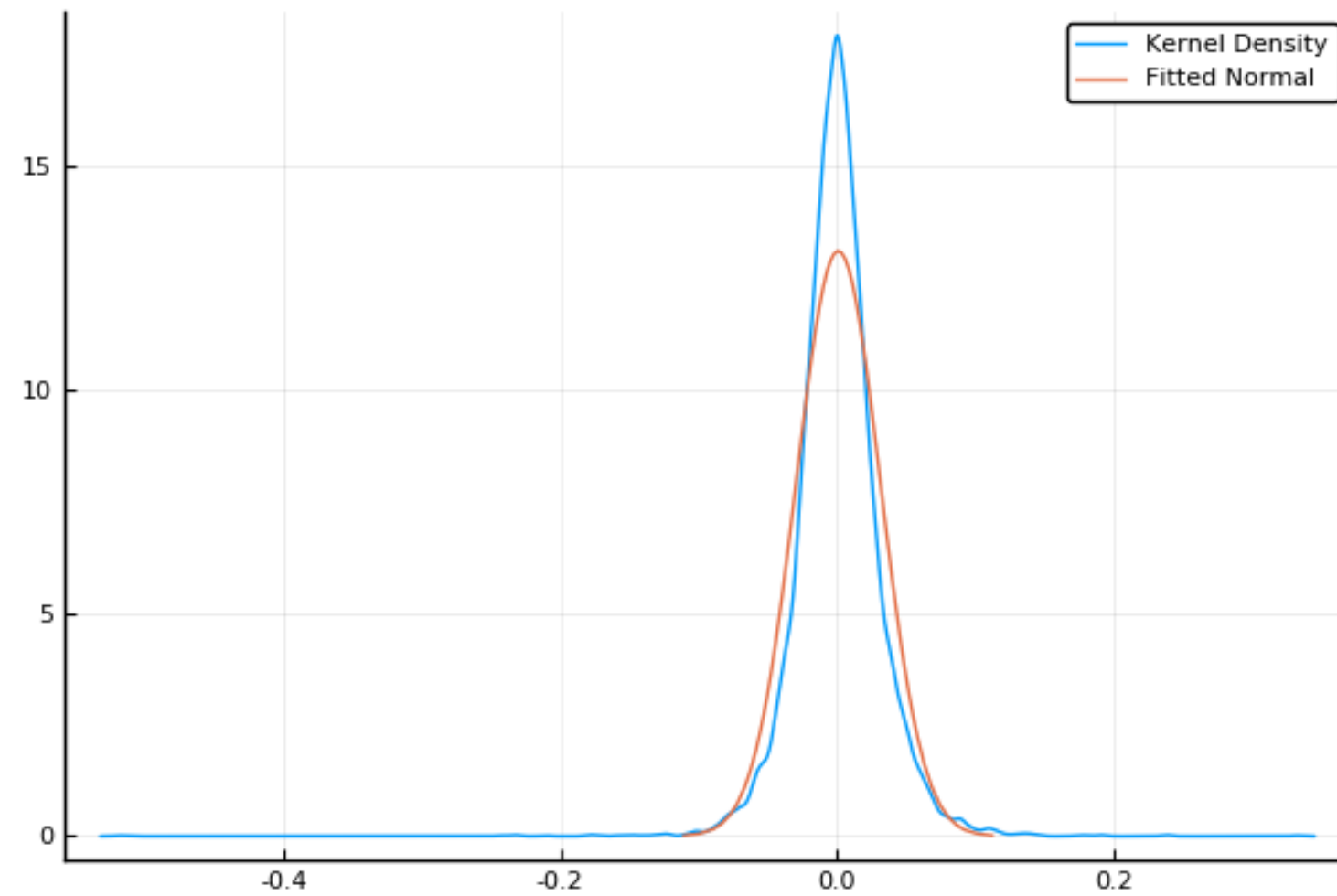
(G)ARCH Models

- Basic setup: given a sample of financial returns $\{r_t\}_{t \in \{1,\dots,T\}}$, decompose r_t as

$$r_t = \mu_t + \sigma_t z_t, \qquad z_t \overset{i.i.d.}{\sim} (0, 1),$$

  where $\mu_t \equiv \mathbb{E}[r_t \mid \mathcal{F}_{t-1}]$ and $\sigma_t^2 \equiv \mathbb{E}[(r_t - \mu_t)^2 \mid \mathcal{F}_{t-1}]$.

- Assume $\mu_t = 0$ for simplicity. Focus is on the *volatility* σ_t. G(ARCH) models make σ_t a function of *past* returns and variances. Examples:

Examples

- ARCH(q) (Engle, Ecta 1982):

$$\sigma_t = \omega + \sum_{i=1}^{q} \alpha_i r_{t-i}^2$$

- GARCH(p, q) (Bollerslev, JoE 1986)

$$\sigma_t = \omega + \sum_{i=1}^{p} \beta_i \sigma_{t-i}^2 + \sum_{i=1}^{q} \alpha_i r_{t-i}^2$$

- EGARCH(o, p, q) (Nelson, Ecta 1991)

$$\log(\sigma_t^2) = \omega + \sum_{i=1}^{o} \gamma_i z_{t-i} + \sum_{i=1}^{p} \beta_i \log(\sigma_{t-i}^2) + \sum_{i=1}^{q} \alpha_i (|z_t| - \mathbb{E}|z_t|)$$

Estimation

- G(ARCH) models are usually estimated by maximum likelihood: with f_z denoting the density of z_t,

$$\max \prod_t f(r_t \mid \mathcal{F}_{t-1}) = \max \prod_t \frac{1}{\sigma_t} f_z(r_t/\sigma_t).$$

- Recursive nature of σ_t means the computation cannot be "vectorized" $\Rightarrow$ loops.
- Julia is very well suited for this. Matlab (and the `rugarch` package for Python) have to implement the likelihood in C.

The ARCH Package

- ARCH.jl is not registered yet; available at https://github.com/s-broda/ARCH.jl
- 0.6 only so far; 0.7 support coming soon.
- Currently supported: simulation and estimation for ARCH, GARCH, and EGARCH models of arbitrary orders, with Gaussian and Student's t errors.
- Designed to be easily extensible with new models, distributions.
- Volatility specifications subtype `VolatilitySpec`. Parametrized on (o, p, q) to facilitate loop unrolling.
- Simulation and estimation return instances of `ARCHModel`, which implements `StatisticalModel` from `StatsBase`.
- Standard errors obtained by AD via `ForwardDiff.jl`.

Usage

In [3]:
```julia
using ARCH
srand(1); T = 10^4   # sample size
volaspec = GARCH{1, 1}([1., .9, .05])   # [omega, beta, alpha]
am = simulate(volaspec, T; dist=StdTDist(3.))   # returns ARCHModel
fit(GARCH{1, 1}, am.data; dist=StdTDist)   # returns ARCHModel
```

Out[3]:

GARCH{1,1} model with Student's t errors, T=10000.

Mean equation parameters:

	Estimate	Std.Error	z value	Pr(>|z|)
μ	0.0031089	0.028261	0.110007	0.9124

Volatility parameters:

	Estimate	Std.Error	z value	Pr(>|z|)
ω	1.01996	0.16134	6.3218	<1e-9
β_1	0.898131	0.0121042	74.1999	<1e-99
α_1	0.0551944	0.0076214	7.24203	<1e-12

Distribution parameters:

	Estimate	Std.Error	z value	Pr(>|z|)
ν	2.92974	0.096228	30.4458	<1e-99

```
# select an EGARCH model without intercept by minimizing AIC; o, p, q < 3
# Uses multiple threads to estimate several (here 2*2*2=8) models
am2 = selectmodel(EGARCH, am.data; meanspec=NoIntercept, criterion=aic, maxlags=2, dist=StdTDist)
```

Out[4]:

EGARCH{1,1,1} model with Student's t errors, T=10000.

Volatility parameters:

	Estimate	Std.Error	z value	Pr(>|z|)
ω	0.153944	0.0235374	6.54041	<1e-10
γ_1	0.00552637	0.00946358	0.583962	0.5592
β_1	0.955436	0.00737491	129.552	<1e-99
α_1	0.145674	0.0150556	9.67573	<1e-21

Distribution parameters:

	Estimate	Std.Error	z value	Pr(>|z|)
ν	2.91693	0.0953735	30.5842	<1e-99

In [5]:
```
# Most of the interface of StatisticalModel is implemented:
# loglikelihood, nobs, fit, fit!, confint, aic, bic, aicc, dof, coef,
# coefnames, coeftable, CoefTable, informationmatrix, islinear, score, vcov:
confint(am2)'
```

Out[5]: 2×5 Array{Float64,2}:
```
 0.107812  -0.0130219  0.940981  0.116165  2.73
 0.200077   0.0240746  0.96989   0.175182  3.10385
```

Benchmarks

- Bollerslev and Ghysels (JBES 1996) data is de facto standard in comparing implementations of GARCH models.
- Data consist of daily German mark/British pound exchange rates (1974 observations).

```
In [6]:   @static if !isfile("DMGBP.txt")
              using HTTP
              open("DMGBP.txt", "w") do io
                  HTTP.get("http://people.stern.nyu.edu/wgreene/Text/Edition7/TableF20-1.txt", response_stream=io)
              end
          end
          r = convert.(Float64, readcsv("DMGBP.txt")[2:end]);
          @static if !isfile("DMGBP.png")
              using Plots
              plot(r)
              savefig("DMGBP")
          end
```

Bollerslev and Ghysels data

GARCH

- Fitting in Julia:

```
In [7]:  using BenchmarkTools
         @btime fit(GARCH{1, 1}, $r, meanspec=NoIntercept)   # Matlab doesn't use an intercept
```

Out[7]:

```
GARCH{1,1} model with Gaussian errors, T=1974.


Volatility parameters:

          Estimate   Std.Error  z value  Pr(>|z|)
ω        0.0108661  0.00657449  1.65277    0.0984
β₁        0.804431   0.0730395  11.0136    <1e-27
α₁        0.154597   0.0539319  2.86651    0.0042


  15.165 ms (2617 allocations: 125.69 KiB)
```

- Now Matlab:

```
In [8]:  using MATLAB
         mat"version"
```

Out[8]: "9.4.0.813654 (R2018a)"

```
In [13]:  # run this cell a few times to give Matlab a fair chance
          mat"tic; estimate(garch(1, 1), $r); toc; 0";
```

```
GARCH(1,1) Conditional Variance Model (Gaussian Distribution):

                    Value        StandardError      TStatistic         PValue
                  _____      _____      _____        _____

      Constant     0.010868        0.0012972          8.3779         5.3896e-17
      GARCH{1}      0.80452         0.016038           50.162                  0
      ARCH{1}       0.15433         0.013852           11.141         7.9448e-29

Elapsed time is 0.102115 seconds.
```

- ARCH.jl is faster by a factor of about 5-10, depending on the machine.

- Estimates are quite similar, but standard errors and t-statistics differ.

- So which standard errors are correct? Let's compare with the results from Brooks et. al. (Int. J. Fcst. 2001).

- Brooks et. al. compare implementations of the GARCH(1, 1) model. They use a model with intercept, so let's re-estimate in Julia (Matlab doesn't seem to allow this):

```
In [14]:  fit(GARCH{1, 1}, r)
```

Out[14]:

```
GARCH{1,1} model with Gaussian errors, T=1974.


Mean equation parameters:

        Estimate   Std.Error    z value Pr(>|z|)
μ    −0.00616637 0.00920163 −0.670139   0.5028


Volatility parameters:

        Estimate   Std.Error z value Pr(>|z|)
ω     0.0107606 0.00649493 1.65677   0.0976
β₁    0.805875   0.0725003 11.1155   <1e-27
α₁    0.153411   0.0536586 2.85903   0.0042
```

- Brooks et. al. give the estimates (t-**stats**) $\mu = -0.00619(\mathbf{-0.67})$, $\omega = 0.0108(\mathbf{1.66})$, $\beta_1 = 0.806(\mathbf{11.11})$, $\alpha_1 = 0.153(\mathbf{2.86})$. Pretty close!

EGARCH

- Julia:

```
In [15]: @btime fit(EGARCH{1, 1, 1}, $r, meanspec=NoIntercept)
```

Out[15]:

```
EGARCH{1,1,1} model with Gaussian errors, T=1974.


Volatility parameters:

          Estimate  Std.Error   z value  Pr(>|z|)
ω        -0.128026  0.0518431  -2.46948    0.0135
γ₁       -0.032216  0.0255372  -1.26153    0.2071
β₁        0.911947  0.0331381   27.5196   <1e-99
α₁        0.333243   0.070109   4.75321    <1e-5


  28.546 ms (3365 allocations: 172.89 KiB)
```

- Matlab:

```
In [16]:  mat"tic; estimate(egarch(1, 1), $r); toc; 0"   # Matlab sets o=q
```

Out[16]: 0.0

```
EGARCH(1,1) Conditional Variance Model (Gaussian Distribution):

                     Value        StandardError      TStatistic         PValue
                   _____     _____     _____       _____

    Constant        -0.1283         0.015788          -8.1267         4.4118e-16
    GARCH{1}         0.91186        0.0084535          107.87                  0
    ARCH{1}          0.33317        0.021769           15.305         7.1324e-53
    Leverage{1}     -0.032252       0.012564           -2.567           0.010258

Elapsed time is 0.147375 seconds.
```

- Brooks et. al. give no benchmark results. But again, Julia is faster by a factor of about 5-
  10.

TODO

- 0.7 compatibility

- docs

- forecasting

- more models, distributions

- Value at Risk

- backtesting

- MGARCH

References

- Bollerslev, T (1986). Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics* **31**, 307–327.
- Bollerslev, T. & Ghysels, E. (1996). Periodic Autoregressive Conditional Heteroscedasticity. *Journal of Business & Economic Statistics* **14**, 139-151. https://doi.org/10.1080/07350015.1996.10524640.
- Brooks, C., Burke, S. P., & Persand, G. (2001). Benchmarks and the accuracy of GARCH model estimation. *International Journal of Forecasting* **17**, 45-56. https://doi.org/10.1016/S0169-2070(00)00070-4.
- Engle, R. F. (1982). Autoregressive Conditional Heteroscedasticity with Estimates of the Variance of United Kingdom Inflation. *Econometrica* **50**, 987-1007. https://doi.org/10.2307/1912773.
- Nelson, D.B. (1991). Conditional Heteroskedasticity in Asset Returns: A New Approach. *Econometrica* **59**, 347--370. https://doi.org/10.2307/2938260.