Financial Risk Forecasting
Seminar Week 6: Univariate Volatility Models

Jon Danielsson
London School of Economics

Version 4.0

6 Univariate volatility

6.1 Why volatility modelling matters for financial risk

Volatility modelling is fundamental to financial risk management. Unlike returns, volatility is not directly observable and changes over time in predictable patterns. Understanding and forecasting volatility is important for several reasons:

- Value-at-Risk calculations: VaR estimates depend directly on volatility forecasts
- Portfolio optimisation: Risk-adjusted returns require accurate volatility estimates
- Derivative pricing: Options and other derivatives are highly sensitive to volatility assumptions
- Regulatory capital: Basel requirements use volatility-based risk measures
- Risk budgeting: Allocating risk across strategies requires volatility forecasts

GARCH models capture the clustering of volatility periods observed in financial markets - periods of high volatility tend to be followed by more high volatility and calm periods by more calm periods.

For more detail, see Univariate volatility models in the R notebook.

In this seminar, we will use the `Returns.RData` file that we created in Seminar 2. We focus on coding the univariate GARCH model using `rugarch` package as well as `car` for QQ plots. You need to install both GARCH packages before proceeding. Please review the theory of GARCH that we discussed in the lecture.

6.2 The plan for this week

1. Load data and packages and do simple statistics
2. Build and run univariate GARCH models

3. Print and plot the outputs
4. Work with various GARCH specifications
5. Assess model quality using likelihood ratio tests and residual analysis
6. GARCH simulations

6.3 Loading data and libraries

```r
library(rugarch)
library(car)
```

```r
load('Returns.RData')
```

6.4 Preliminary analysis

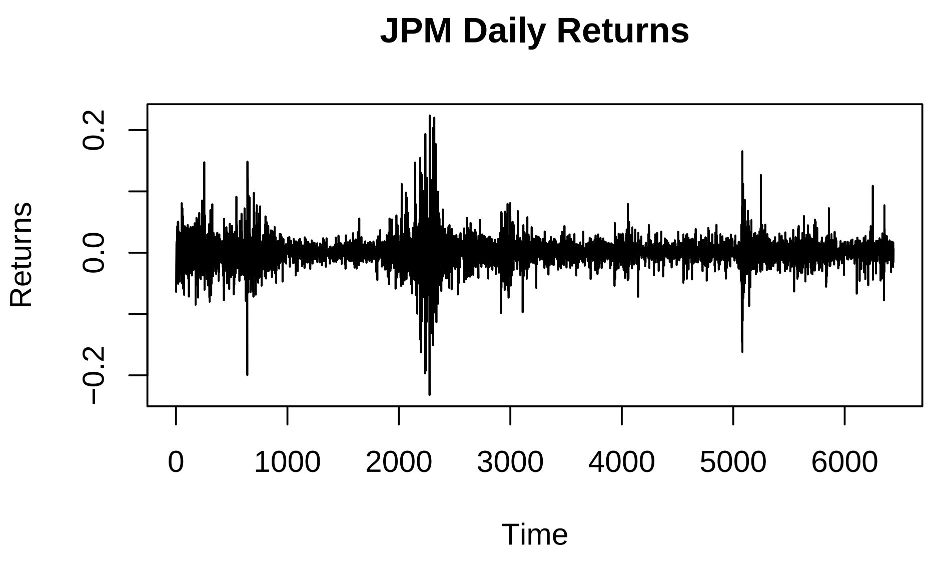
Before fitting GARCH models, examine the data for evidence of volatility clustering. We use JPMorgan Chase (JPM) as our example since financial stocks often exhibit clear volatility patterns.

```r
y=Returns$JPM
```

6.4.1 Examining the return series

First, plot the return series to visualize volatility clustering:

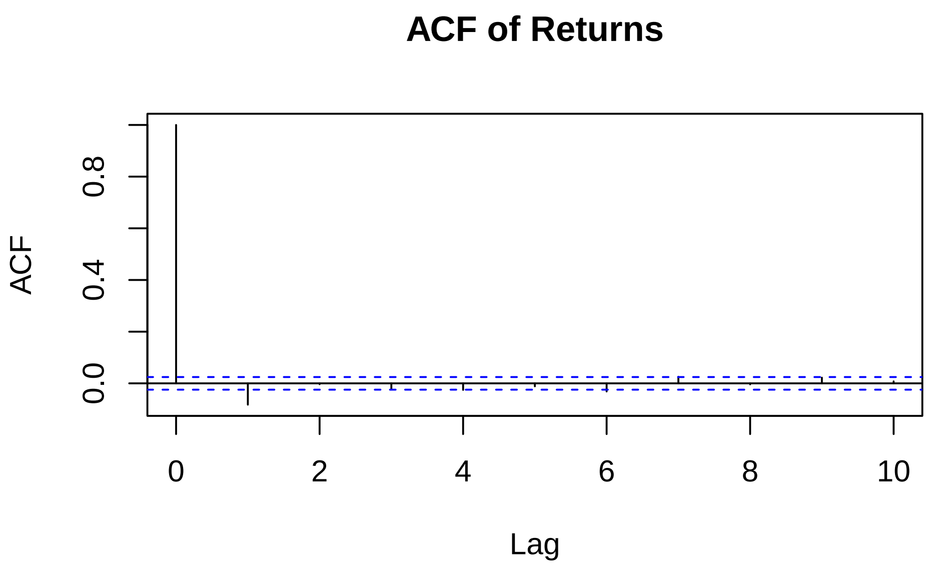
```r
plot(y,type='l', main="JPM Daily Returns", xlab="Time", ylab="Returns")
```



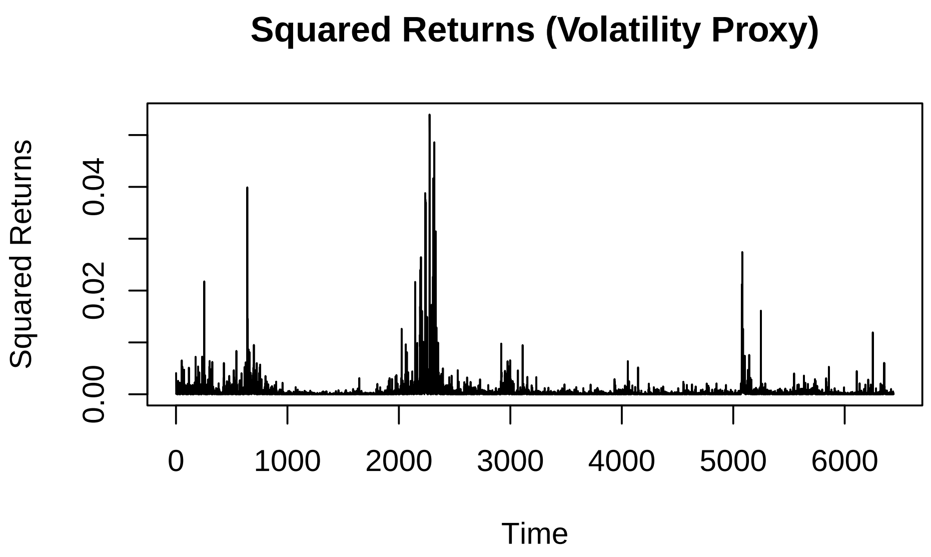
6.4.2 Autocorrelation analysis

The key insight for GARCH modelling comes from examining autocorrelations. Returns themselves typically show little autocorrelation, but squared returns (a proxy for volatility) often show strong autocorrelation:

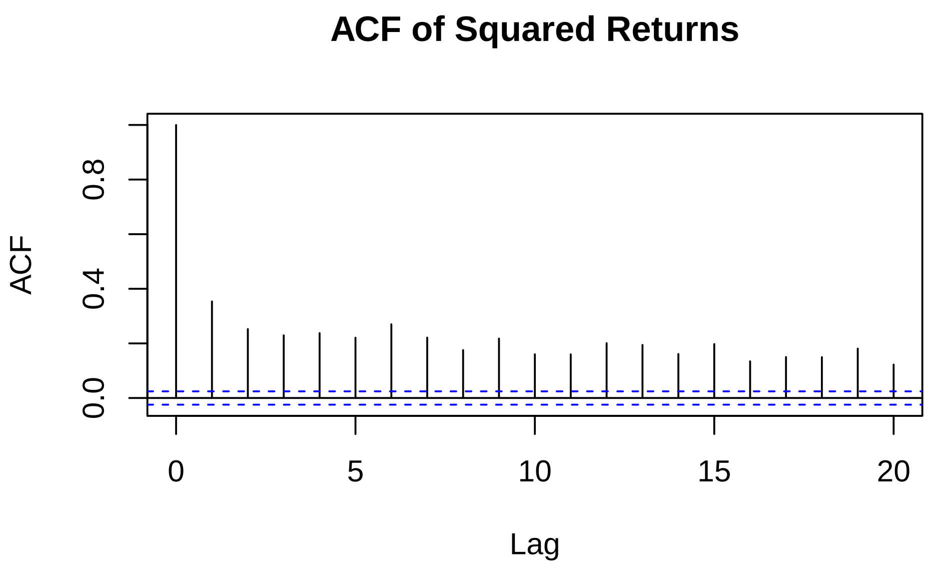
```r
# ACF of returns - should show little correlation
acf(y,lag=10, main="ACF of Returns")
```



```
# Plot squared returns
plot(y^2, type='l', main="Squared Returns (Volatility Proxy)", xlab="Time", ylab="Squared
```



```
# ACF of squared returns - often shows persistence
acf(y^2,lag=20, main="ACF of Squared Returns")
```



If squared returns show significant autocorrelation, this indicates volatility clustering and justifies using GARCH models. ## Build and run univariate GARCH models

We start by comparing two GARCH specifications:

```
# spec0 uses all default settings: GARCH(1,1) with constant mean
spec0 = ugarchspec()
spec0
```

```
*---------------------------------*
*       GARCH Model Spec          *
*---------------------------------*

Conditional Variance Dynamics
------------------------------------
GARCH Model      : sGARCH(1,1)
Variance Targeting  : FALSE

Conditional Mean Dynamics
------------------------------------
Mean Model       : ARFIMA(1,0,1)
Include Mean         : TRUE
GARCH-in-Mean        : FALSE

Conditional Distribution
------------------------------------
Distribution    :  norm
Includes Skew   :  FALSE
Includes Shape  :  FALSE
Includes Lambda :  FALSE
```

```r
# spec1 is modified to exclude the mean term
spec1=ugarchspec(mean.model =
  list(armaOrder=c(0,0),include.mean=FALSE))
spec1
```

```
*---------------------------------*
*       GARCH Model Spec          *
*---------------------------------*

Conditional Variance Dynamics
------------------------------------
GARCH Model     : sGARCH(1,1)
Variance Targeting  : FALSE

Conditional Mean Dynamics
------------------------------------
Mean Model      : ARFIMA(0,0,0)
Include Mean        : FALSE
GARCH-in-Mean       : FALSE

Conditional Distribution
------------------------------------
Distribution    :  norm
Includes Skew   :  FALSE
Includes Shape  :  FALSE
Includes Lambda :  FALSE
```

```r
output0=ugarchfit(spec0,y,solver = "hybrid")
output1=ugarchfit(spec=spec1,
  data=y,solver = "hybrid")
```

The default specification (`spec0`) uses an ARFIMA(1,0,1) mean model with a constant term. The modified specification (`spec1`) uses a simpler ARMA(0,0) mean model with no constant term (`include.mean=FALSE`).

About the solver argument: The `solver = "hybrid"` argument specifies the optimisation algorithm. It's optional for simple models but often necessary for complex specifications or difficult datasets. When omitted, `ugarchfit()` uses the default solver. Use `solver = "hybrid"` when you encounter convergence problems or for complex models like apGARCH. You can find more details in the notebook.

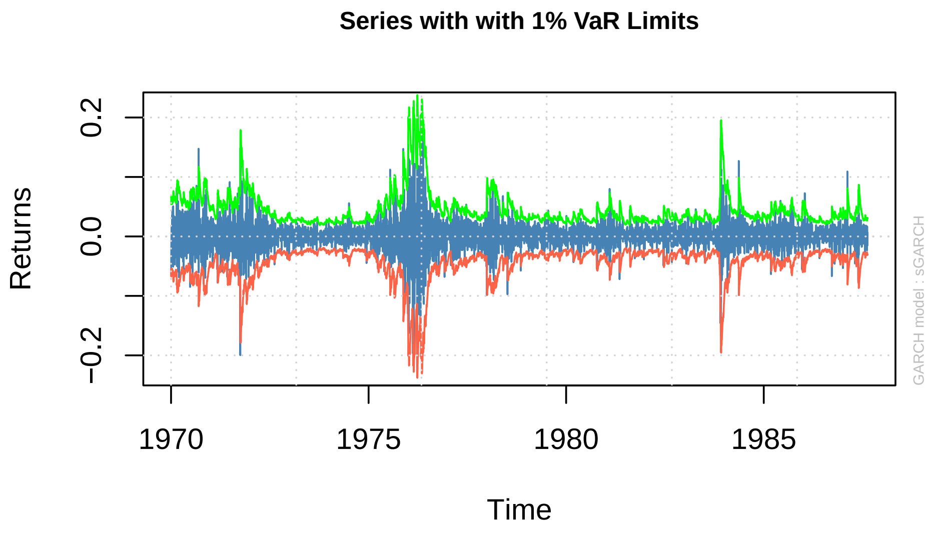
6.5 Print and plot the outputs

When you run `plot(output1)`, it displays a menu of options for making various plots. In an interactive session, this would show a menu. Here we'll specify which plots to create:

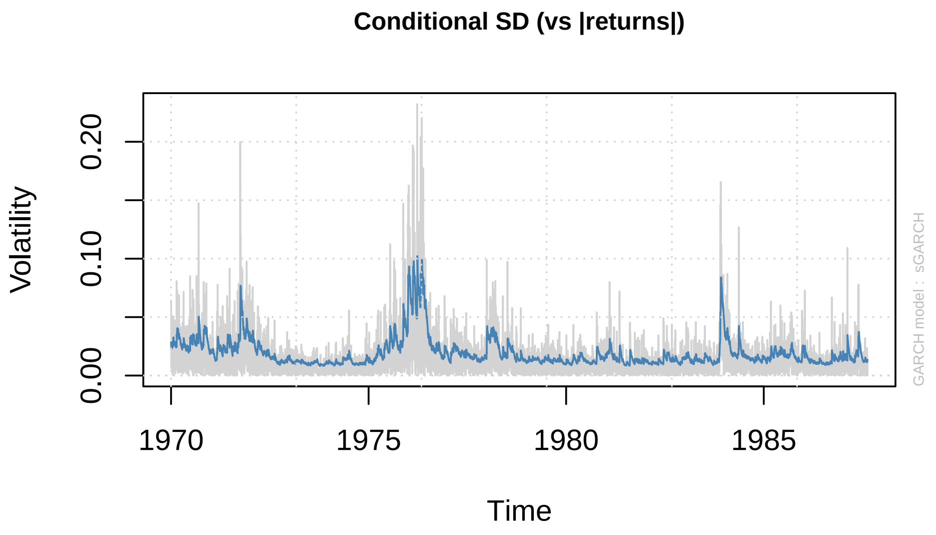
```r
# This would show an interactive menu
plot(output1)
```

```
# Instead, let's create specific plots
plot(output1, which = 2)  # Conditional SD (volatility)
```

```
please wait...calculating quantiles...
```



```
plot(output1, which = 3)  # Series with 2 Conditional SD Superimposed
```



```
names(output1@fit)
```

```
 [1] "hessian"        "cvar"           "var"            "sigma"
 [5] "condH"          "z"              "LLH"            "log.likelihoods"
 [9] "residuals"      "coef"           "robust.cvar"    "A"
[13] "B"              "scores"         "se.coef"        "tval"
[17] "matcoef"        "robust.se.coef" "robust.tval"    "robust.matcoef"
[21] "fitted.values"  "convergence"    "kappa"          "persistence"
[25] "timer"          "ipars"          "solver"
```

```r
output1@fit$coef
```

```
       omega       alpha1        beta1 
3.734768e-06 8.349090e-02 9.092409e-01 
```

```r
l1=likelihood(output1)
coef(output1)
```

```
       omega       alpha1        beta1 
3.734768e-06 8.349090e-02 9.092409e-01 
```

6.6 Work with various GARCH specifications

There are many GARCH variants available. We explore them systematically, starting with basic specifications, then adding distributional assumptions and finally examining advanced models.

6.6.1 Basic model specifications

6.6.1.1 ARCH model

The ARCH(1) model has no GARCH term, only the ARCH term. Use ARCH when volatility clustering is relatively short-lived and doesn't persist over long periods:

6.6.2 Distribution variations

6.6.2.1 GARCH with Student-t distribution

Adding fat tails to capture extreme events. Use Student-t distribution when your data shows more frequent extreme returns than predicted by the normal distribution:

```r
spec3=ugarchspec(
  distribution.model='std',
  mean.model = list(
    armaOrder=c(0,0),
    include.mean=FALSE)
  )
output3=ugarchfit(spec=spec3,data=y)
```

6.6.3 Advanced model specifications

6.6.3.1 Asymmetric Power GARCH (apGARCH)

This model captures both asymmetry and power transformation. Use apGARCH when negative shocks (bad news) have a larger impact on volatility than positive shocks of the same magnitude:

6.6.3.2 apGARCH with Student-t distribution

Combining asymmetric volatility responses with fat-tailed distributions. Use for stocks showing both leverage effects and extreme return events:

```
spec5=ugarchspec(
  variance.model=list(model="apARCH", garchOrder=c(1,1)),
  mean.model = list(armaOrder=c(0,0),include.mean=FALSE),
  distribution.model='std')
output5=ugarchfit(spec=spec5,data=y,solver = "hybrid")
```

6.6.3.3 apGARCH with Skewed Student-t distribution

Adding skewness to capture asymmetric return distributions. Use this when returns are not only fat-tailed but also systematically skewed (e.g., more frequent small positive returns and occasional large negative returns):

```
spec6=ugarchspec(
  variance.model=list(model="apARCH", garchOrder=c(1,1)),
  mean.model = list(armaOrder=c(0,0),include.mean=FALSE),
  distribution.model='sstd')
output6=ugarchfit(spec=spec6,data=y,solver = "hybrid")
```

6.6.4 Systematic model comparison

Now we compare all models systematically:

```
# Extract log-likelihoods
cat("Log-likelihoods:\n")
```

```
Log-likelihoods:
```

```
cat("GARCH(1,1):", likelihood(output1), "\n")
```

```
GARCH(1,1): 17040.47
```

```
cat("ARCH(1):", likelihood(output2), "\n")
```

```
ARCH(1): 15939.9
```

```
cat("GARCH-t:", likelihood(output3), "\n")
```

```
GARCH-t: 17286.57
```

```
cat("apGARCH:", likelihood(output4), "\n")
```

```
apGARCH: 17141.54
```

```
cat("apGARCH-t:", likelihood(output5), "\n")
```

```
apGARCH-t: 17361.4
```

```
cat("apGARCH-sstd:", likelihood(output6), "\n")
```

```
apGARCH-sstd: 17363.28
```

```
# Extract volatility persistence
cat("\nVolatility persistence:\n")
```

```

Volatility persistence:
```

```r
cat("GARCH(1,1):", sum(coef(output1)[2:3]), "\n")
```

```
GARCH(1,1): 0.9927318
```

```r
cat("ARCH(1):", coef(output2)[2], "\n")
```

```
ARCH(1): 0.5511824
```

```r
# Compare shape parameters
cat("\nDistribution parameters:\n")
```

```

Distribution parameters:
```

```r
cat("GARCH-t df:", coef(output3)["shape"], "\n")
```

```
GARCH-t df: 5.40673
```

```r
cat("apGARCH-t df:", coef(output5)["shape"], "\n")
```

```
apGARCH-t df: 5.762461
```

```r
cat("apGARCH-sstd df:", coef(output6)["shape"], "\n")
```

```
apGARCH-sstd df: 5.749031
```

```r
cat("apGARCH-sstd skew:", coef(output6)["skew"], "\n")
```

```
apGARCH-sstd skew: 0.9692253
```

Interpreting these results for risk assessment:

- Volatility persistence: Values close to 1 indicate long-lasting volatility shocks, important for multi-day VaR forecasts
- Shape parameters (df): Lower values indicate fatter tails and higher tail risk; values below 4 suggest extreme events are more likely than normal distribution predicts
- Skewness: Negative values indicate more frequent large negative returns than positive ones, affecting downside risk estimates
- Log-likelihood: Higher values indicate better model fit; use for model selection in risk system implementation

6.7 Assess model quality using likelihood ratio tests and residual analysis

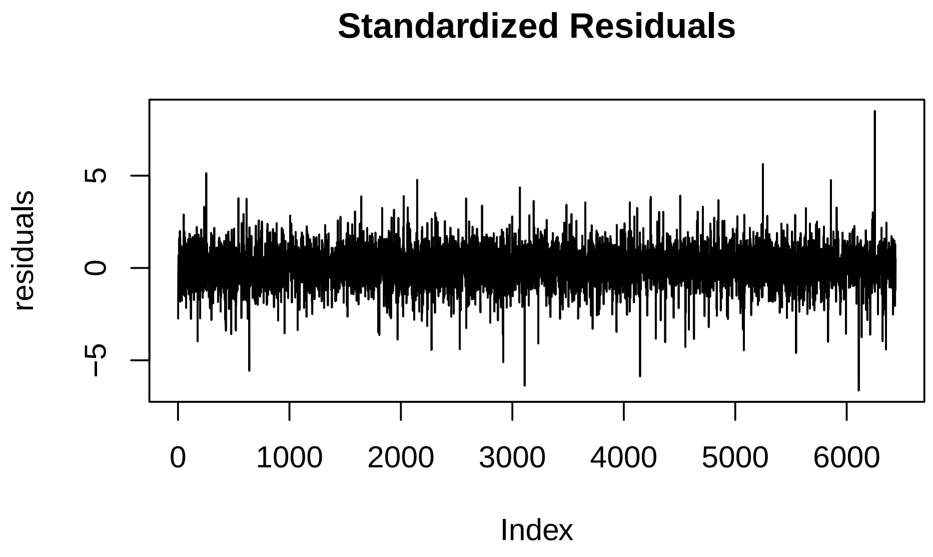
We can use the likelihood ratio test to compare the effect in nested specifications.

```r
LR_statistic=2*(likelihood(output3)-likelihood(output1))
p_value  = 1 - pchisq(LR_statistic, df = 1)
```

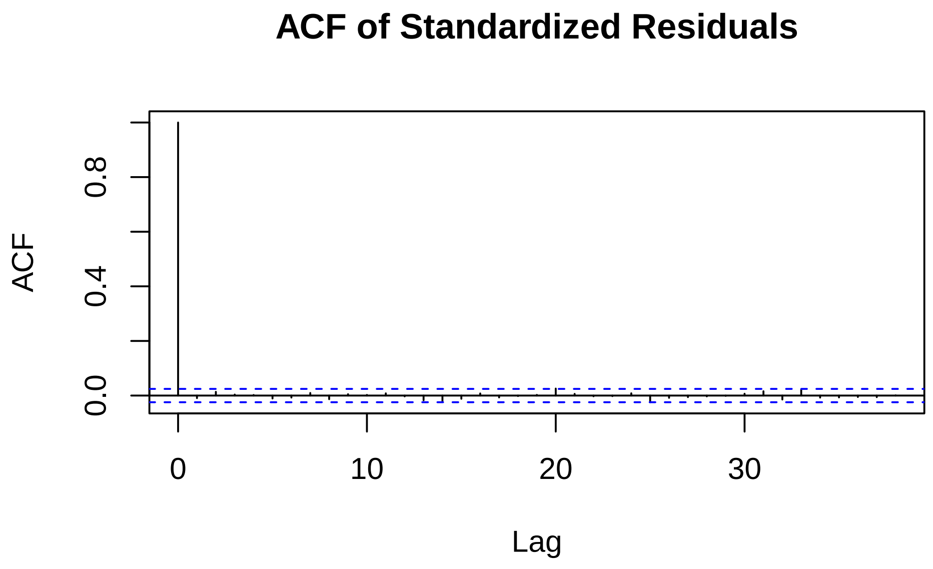
6.7.1 Residual analysis for model adequacy

The residuals provide information on whether the model is correctly specified. We examine standardized residuals, which should behave like white noise if the model captures all patterns in the data.

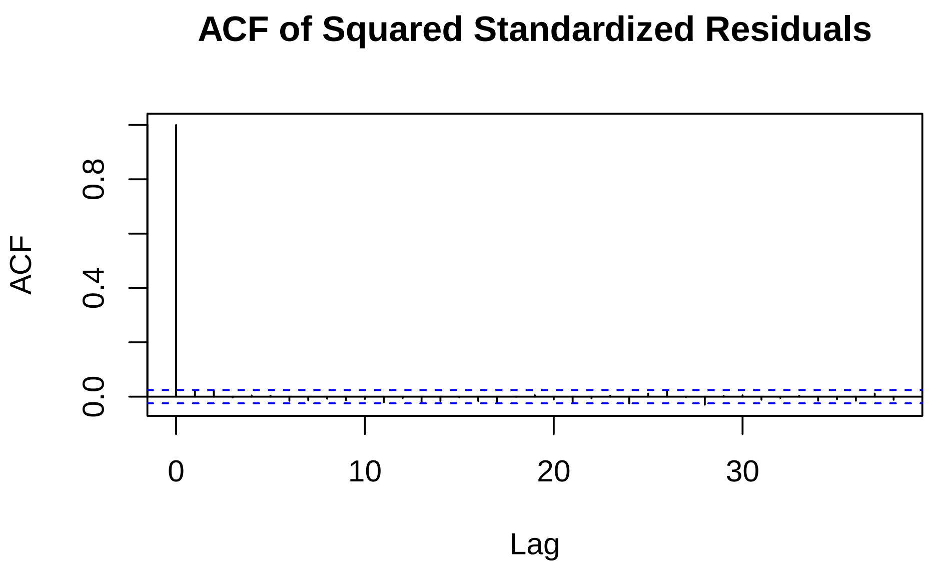
```
residuals=y/output3@fit$sigma
plot(residuals, main="Standardized Residuals", type='l')
```



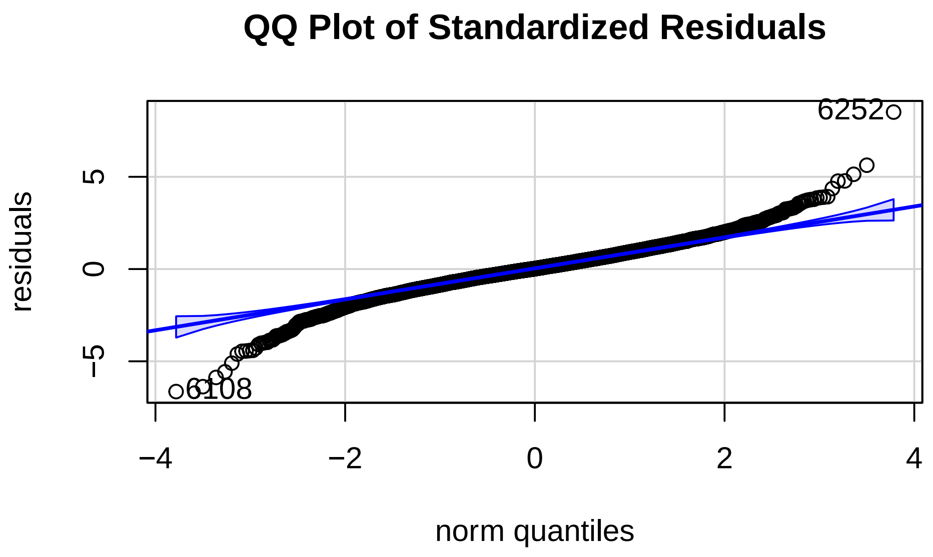
```
acf(residuals, main="ACF of Standardized Residuals")
```



```
acf(residuals^2, main="ACF of Squared Standardized Residuals")
```



```r
qqPlot(residuals, main="QQ Plot of Standardized Residuals")
```



```
[1] 6252 6108
```

How to interpret these diagnostics:

- Residual plot: Should show no patterns over time; systematic patterns indicate model misspecification
- ACF of residuals: Should show no significant autocorrelation; violations suggest the mean model needs improvement
- ACF of squared residuals: Should show no significant autocorrelation; violations indicate remaining volatility clustering that the GARCH model failed to capture
- QQ plot: Points should follow the diagonal line; deviations in the tails suggest the distributional assumption (normal, Student-t, etc.) may be incorrect

6.8 Handling convergence issues

GARCH models can sometimes fail to converge, especially with difficult datasets or complex specifications. Here are common troubleshooting approaches:

Common convergence problems: - Starting values too far from optimal - Poorly conditioned data (extreme outliers) - Over-parameterized models - Numerical precision issues

Solutions to try:

```r
# Try different solvers
output_gosolnp = ugarchfit(spec1, y, solver = "gosolnp")
output_nlminb = ugarchfit(spec1, y, solver = "nlminb")
output_lbfgs = ugarchfit(spec1, y, solver = "lbfgs")

# Check convergence status
output1@fit$convergence  # Should be 0 for successful convergence

# If convergence fails, try simpler specifications first
spec_simple = ugarchspec(
  variance.model = list(garchOrder = c(1,1)),
  mean.model = list(armaOrder = c(0,0), include.mean = FALSE),
  distribution.model = "norm"
)
```

When models fail to converge: - Start with simpler specifications and build complexity gradually - Check data for extreme outliers that might cause numerical issues - Try different optimisation algorithms using the `solver` argument - Consider alternative model families (eGARCH, gjrGARCH) if apGARCH fails

6.9 GARCH simulations

It can be useful to simulate from a volatility model. Perhaps to identify whether a model describes the data properly, to run a Monte Carlo test with known stochastic process as input or in derivative pricing.

```r
ugarchsim(output1,10)
```

```
*------------------------------------*
*       GARCH Model Simulation       *
*------------------------------------*
Model : sGARCH
Horizon:  10
Simulations: 1
              Seed Sigma2.Mean Sigma2.Min Sigma2.Max Series.Mean Series.Min
sim1            NA    0.000427   0.000363   0.000477    0.010837    -0.0184
Mean(All)        0    0.000427   0.000363   0.000477    0.010837    -0.0184
Actual           0    0.000542   0.000067   0.010451    0.000382    -0.2323
Unconditional    0    0.000514         NA         NA    0.000000         NA
              Series.Max
sim1              0.0367
```

```
Mean(All)          0.0367
Actual             0.2239
Unconditional          NA
```

6.10 Recap

6.10.1 In this seminar, we have covered:

- Understanding and implementing univariate GARCH models
- Working with the `rugarch` package for volatility modeling
- Preliminary analysis using ACF plots to detect volatility clustering
- Specifying different GARCH models:
  - Standard GARCH(1,1) vs ARFIMA mean models
  - ARCH(1) for short-lived volatility clustering
  - GARCH with Student-t distribution for fat tails
  - Asymmetric Power GARCH (apGARCH) for leverage effects
  - Combined models with skewed Student-t distributions
- Extracting and interpreting model outputs:
  - Coefficients and likelihood values
  - Conditional volatility estimates
  - Volatility persistence and shape parameters
  - Standardized residuals
- Model comparison using likelihood ratio tests
- Diagnostic checking through residual analysis:
  - ACF plots of residuals and squared residuals
  - QQ plots for distributional assumptions
  - Interpreting diagnostic results for model adequacy
- Handling convergence issues and troubleshooting
- Simulating from fitted GARCH models

6.10.2 Some new functions used:

- `ugarchspec()` — specify a univariate GARCH model
- `ugarchfit()` — fit the specified GARCH model to data
- `ugarchsim()` — simulate from a fitted GARCH model
- `likelihood()` — extract log-likelihood from fitted model
- `coef()` — extract model coefficients
- `pchisq()` — chi-squared distribution function for LR test
- `@` — access S4 object slots in R

6.11 Optional exercises

1. Compare basic GARCH specifications:
   - Fit GARCH(1,1), GARCH(2,1) and GARCH(1,2) to the same stock
   - Extract one-day ahead volatility forecasts using `sigma(output)[length(y)]`
   - Plot all forecasts on the same graph with different colors
   - Which model gives the highest/lowest forecast?
2. Create a model comparison function:
   - Write a function `compare_garch(data, specs)` where specs is a list of model names

- For each spec, fit the model and extract: log-likelihood, AIC, BIC and volatility forecast
- Return results in a data frame sorted by AIC
- Test with specs = c("sGARCH", "eGARCH", "gjrGARCH")

3. Distribution comparison:
   - Fit GARCH(1,1) with normal, Student-t and skewed Student-t distributions
   - For each model, calculate standardized residuals
   - Create QQ plots for all three models in a 1x3 grid
   - Which distribution best captures the tail behaviour?
4. Sample size analysis:
   - Using the last 2000 observations of a stock
   - Fit GARCH(1,1) using samples of size 500, 1000, 1500, 2000
   - For each sample size, record the estimated omega, alpha and beta parameters
   - Plot how parameter estimates stabilize as sample size increases
5. Volatility forecasting evaluation:
   - Split your data into training (first 80%) and testing (last 20%)
   - Fit GARCH, EGARCH and GJR-GARCH models on training data
   - For each day in the test set, forecast one-day ahead volatility
   - Compare forecast accuracy using squared returns as a proxy for realized volatility
   - Calculate MSE for each model - which performs best?