

# Bayesian Inference and Volatility Modeling Using Stan

R/FINANCE 2019

---

Luis Damiano (Department of Statistics, Iowa State University, Ames IA)  
Michael Weylandt (Department of Statistics, Rice University, Houston TX)

2019-05-17

# Introduction

---

# Introductions

Luis Damiano:

- Previously: FIRST Capital Markets (Argentina)
- Currently: Ph.D. Student at ISU Statistics



Michael Weylandt:

- Previously: Morgan Stanley (NYC)
- Currently: Ph.D. Student at Rice Statistics



## Credits

We (MW, LD) are not developers of **Stan**, only happy users.

Credit for **Stan** goes to the fantastic Stan community and the Stan Development Team

Several examples in this presentation are covered in the excellent **Stan** manual and papers referenced therein. Any mistakes in exposition are solely the responsibility of MW and LD.

The code samples in these slides are pedagogical in nature and, in some places, simplify the underlying models significantly

# Outline

- Stan Language Crash Course
- Coding Volatility Models in **Stan**
- The Bayesian Workflow
- Advanced GARCH Models [time allowing]

# Stan Language Crash Course

---

# Stan Language

**Stan**  $\approx$  C++ + BUGS

Statically typed probabilistic programming language divided into functional blocks: **data**, **parameters**, **model**, etc.

For this crowd, use through the **rstan** package

Write **Stan** code in a separate file → compile using  
`rstan::stan_model` in R → sample from the posterior using  
`rstan::sampling`

Uses semicolons;

For a more detailed intro, see the excellent Stan manual or MW's 2016 or 2017 R/Finance tutorials

## data block

The `data` block specifies the input to a Stan program

```
data {
    // No automatically sized arrays: must pass sizes as
    int<lower=1> T;
    vector[T]      r;
    // vector[T] is the mathematical idea of a vector
    // real r[T] is an array (container) of T reals
}
```

## parameters block

The `parameters` block specifies what you want to learn from your data

```
parameters {
    real mu;

    // Can add constraints to parameters
    real<lower=0> sigma;
    real<lower=0,upper=1> alpha1;

    // Constraints can depend on other parameters
    real<lower=0,upper=(1-alpha1)> alpha2;
}
```

## model block

The **model** block connects data and **parameters** by specifying a joint distribution (density)

Specify priors and likelihood here - can use sampling statements ( $\sim$ ) for brevity

```
model {  
    mu ~ normal(0, 1);  
    // Constraints on parameters are  
    // automatically respected  
    sigma ~ student_t(5, 0, 1);  
    r ~ normal(mu, sigma);  
}
```

## transformed parameters block

Often, it is useful to work with *transformations* of “raw” parameters - e.g., we may specify variance and correlation separately, but need to use a covariance matrix in our model.

The `transformed parameters` block fills this need. Essentially nothing in this block can’t be done in `model` instead, but it’s conceptually cleaner.

```
transformed parameters {
    // Need to "forward declare"
    // transformed parameters
    vector<lower=0>[T] sigma;
    sigma[1] = sigma1;

    for(t in 2:T){
        sigma[t] = sqrt(alpha0 +
                        beta1 * square(sigma[t - 1]) +
                        alpha1 * square(r[t-1]));
    }
}
```

## generated quantities block

We are typically more interested in functions of the parameters than the parameters themselves. These can be calculated in the `generated quantities` block if you want to have your entire analysis in Stan

```
generated quantities {  
    // Need to "forward declare"  
    // generated quantities  
    vector[T] std_residuals;  
    std_residuals = (r - mu) / sigma;  
}
```

## Coding Volatility Models in Stan

---

# Volatility Modeling: Perspectives

Traditional econometrician: volatility (heteroscedasticity) is a problem to be corrected

Financial econometrician: volatility is a quantity to be modeled

Bad trader: volatility is something to be survived

Good trader: volatility is something to make money from

⇒ For now, we adopt the second perspective

## Volatility Modeling: General approaches

A troubling fact: volatility moves as quickly (more quickly?) than prices

Basic estimator:  $r_t^2 \approx \sigma_t^2$  (quality of this approximation decreases with  $\sigma_t^2$ )

Standard approach: combine the basic estimator with parametric dynamics to pool information across time (*i.e.*, smooth our estimates)

Today we only discuss *second-moment* models - require a mean (expected return) model to be specified separately. We take constant mean for simplicity

# Stochastic Volatility

The “stochastic volatility” model (Kastner 2016; Kim et al. 1998):  
assume the volatility follows stochastic Ornstein-Uhlenbeck/AR(1)  
dynamics

$$r_t \sim \mathcal{N} \left( \mu, e^{h_t/2} \right)$$

$$= \mu + \epsilon_t e^{h_t/2}$$

$$h_t \sim \mathcal{N}(\omega + \phi(h_{t-1} - \omega), \sigma_\eta)$$

$$= \omega + \phi(h_{t-1} - \omega) + \eta_t$$

$$\begin{pmatrix} \epsilon_t \\ \eta_t \end{pmatrix} | \mathfrak{F}_{t-1} \sim \mathcal{N}(\mathbf{0}, \Sigma)$$

$$\Sigma = \begin{pmatrix} 1 & 0 \\ 0 & \sigma_\eta^2 \end{pmatrix}$$

# Stochastic Volatility

```
data {  
    int<lower=1> T;  
    vector[T]      r;  
}  
parameters {  
    real mu;  
    real omega;  
    // persistence of volatility  
    real<lower=-1,upper=1> phi;  
    real<lower=0> sigma_eta;  
    // Define raw log volatility parameter on  
    // 0(1) scale for better performance  
    vector[T] h_std;  
}
```

# Stochastic Volatility

```
transformed parameters {
    vector[T] h = h_std * sigma_eta;
    vector[T] sigma;
    h[1] /= sqrt(1 - phi * phi);
    h += omega;
    for (t in 2:T)
        h[t] += phi * (h[t-1] - omega);
    }
    sigma = exp(h/2);
}
model {
    // Priors
    h_std ~ normal(0, 1);
    phi      ~ uniform(-1, 1);
    sigma_eta ~ student_t(3, 0, 5);
    omega    ~ student_t(3, 0, 10);
    mu       ~ normal(0, 1);
    // Likelihood
    r ~ normal(mu, sigma);
}
```

# Stochastic Volatility

```
library(quantmod)
library(rstan)
library(xtsPlots) # github.com/michaelweylandt/xtsPlots

sv <- stan_model("sv.stan")
SPY <- getSymbols("SPY", from="2015-01-01", auto.assign=FALSE)
SPY.R <- na.omit(ROC(Ad(SPY)));

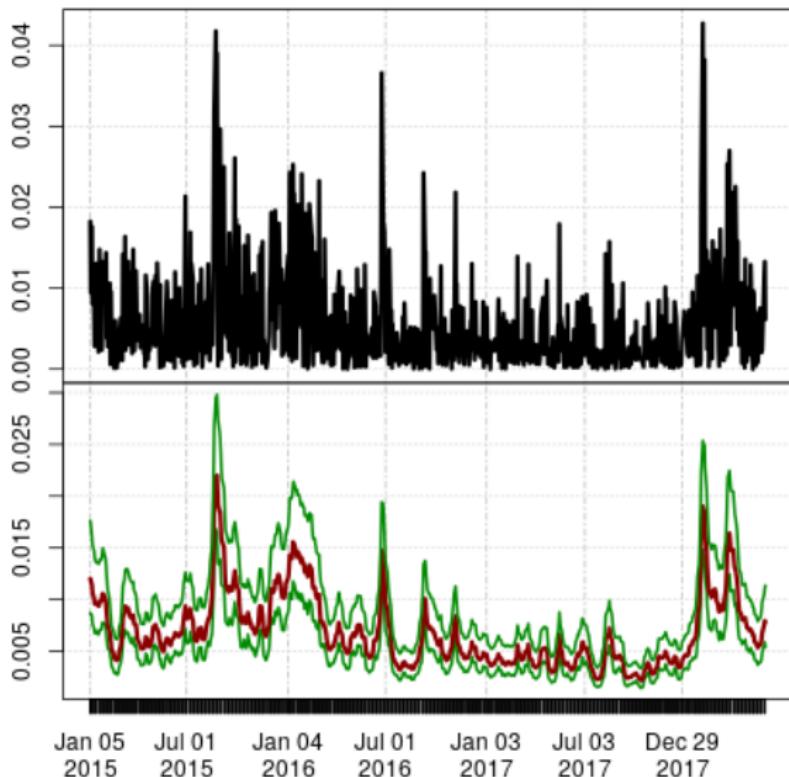
samples <- sampling(sv, data=list(T=length(SPY.R),
                                  # Stan is picky about types
                                  r=as.vector(SPY.R)))

sv_bands <- apply(as.matrix(samples, "sigma"), 2,
                  function(x) quantile(x, c(0.1, 0.5, 0.9)))

plot(merge(abs(SPY.R), t(sv_bands)),
     col=c("black", "green4", "red4", "green4"),
     lwd=c(3, 2, 3, 2),
     main=paste0("SPY Absolute Daily Returns ",
                "+ 90% Posterior Intervals for Volatility"),
     screens=c(1, 2, 2, 2))
```

# Stochastic Volatility

SPY Absolute Daily Returns + 90% Posterior Intervals for Volatility



SV models are nice, but they are “high-dimensional” models - more parameters than data points (even in the simplest case)  $\implies$  tricky computation

GARCH type models (Bollerslev 1986, 2010) address this by proposing *deterministic* updates to the volatility (conditional on parameters and previous observations)

$$r_t \sim \mathcal{N}(\mu, \sigma_t^2)$$
$$\sigma_t^2 = \alpha_0 + \alpha_1 r_t^2 + \beta_1 \sigma_{t-1}^2$$

Few “raw” parameters (+ data) determine a whole series of  $\sigma_t^2 \implies$  use **transformed parameters** block

```
data {  
    int<lower=1> T;  
    vector[T] r;  
}  
parameters {  
    real mu;  
    real<lower=0> alpha0;  
    real<lower=0,upper=1> alpha1;  
    real<lower=0,upper=(1-alpha1)> beta1;  
  
    real<lower=0> sigma1; // Starting volatility  
}
```

```
transformed parameters {
  vector<lower=0>[T] sigma;
  sigma_market[1] = sigma1;

  for(t in 2:T){
    sigma[t] = sqrt(alpha0 +
                      alpha1 * square(r[t - 1]) +
                      beta1 * square(sigma[t - 1]));
  }
}
```

# GARCH

```
model {
    // Priors
    sigma1 ~ normal(sd(r), 3 * sd(r));

    mu          ~ normal(0, 1);
    alpha0      ~ normal(0, 1);
    alpha1      ~ normal(0, 1);
    beta1       ~ normal(1, 1);

    // Likelihood
    r ~ normal(mu, sigma);
}
```

# GARCH - Initial Volatility

How to initialize the volatility process?

- Analytical long-run volatility
- Sample volatility
- Early part of sample volatility

I prefer the latter two approaches, typically making the initial volatility itself random

## Extensions

Extending to other sampling distributions (non-normal returns) is easy - just change the likelihood

Leverage effects can be added by changing the **transformed parameters** updates

**Tip:** easier to specify GARCH dynamics based on returns than standardized returns. This avoids a non-linear transform of parameters for which you need a *Jacobian correction* (though this is a modeling choice you can check - more below)

# The Bayesian Workflow

---

# Running Example

Daily Log-Returns of SPY from 1993-01-01 to 2019-05-18:

```
library(doParallel)
library(quantmod)
library(moments)
library(rstan)
library(xts)

rstan_options(auto_write = TRUE)
options(mc.cores = parallel::detectCores())

p <- getSymbols("SPY", src = "yahoo",
                 from = "1993-01-01", to = "2019-05-18",
                 auto.assign = FALSE)
y <- na.omit(ROC(log(Cl(p)), n = 1, type = "continuous"))
```

# Running Example

And fit using a GARCH(1, 1) model:

```
fit <- stan(file = "stan/garch11.stan",
             data = list(
               T      = length(y) # Length of the series,
               H      = 1 # Produce (H = 1)-step-ahead forecast
               y      = as.numeric(y) # observed series (daily returns)
               sigma1 = sd(y) # Initial volatility
             ))
## In-sample estimates
spred <- extract(fit, pars = "spred")[[1]]
ypred <- extract(fit, pars = "ypred")[[1]]
## Out-of-sample estimates
sfore <- extract(fit, pars = "sfore")[[1]]
yfore <- extract(fit, pars = "yfore")[[1]]
```

## Scientific Problem

---

We would like to measure and forecast daily volatility in stock prices.

## Substantive Knowledge

What do we know about returns and volatility? (Cont 2001)

- Unconditional heavy tails
- Gain/loss asymmetry (except for foreign exchange rates)
- Shape of distribution changes with time scale
- Volatility clusters
- Conditional heavy tails (after correcting for volatility clustering)
- Slow decay of autocorrelation in absolute/squared returns
- Leverage effect
- Others

*Use the model and/or priors to formally account for your substantive knowledge*

# Exploratory Data Analysis

*A method to build a network of increasingly complex models that capture features and heterogeneities present in the data (Gelman 2004).*

What are the possible sources of heterogeneity?

- Non-linearity
- Time-varying properties/relationships (coefficients)
- Hierarchies
- Clusters
- Latent variables (ex. latent states)

## GARCH(1,1) Model

Recall GARCH(1, 1) Model: (Bollerslev 1986; Engle 1982)

$$y_t \sim \mathcal{N}(0, \sigma_t^2)$$
$$\sigma_t^2 = \alpha_0 + \alpha_1 y_{t-1}^2 + \beta \sigma_{t-1}^2$$

where  $\sigma_t^2$  is the conditional variance at time  $t$  and  $y_t$  is the (non-filtered) log-return at time  $t$ .

Stationarity constraints on parameters:

- $\alpha_0 > 0$
- $\alpha_1, \beta \geq 0$
- $\alpha_1 + \beta < 1$

Ensure that unconditional variance is finite and positive and the conditional variance is positive

## GARCH(1, 1) Model - Interpretation of Parameters

Interpretation (Alexander 2008):

- **Error parameter:**  $\alpha_1$  measures the reaction of conditional volatility to market shocks. The larger the coefficient, the more sensitive to unexpected movements in prices.
- **Lag parameter:**  $\beta$  measures the persistence in conditional volatility. The larger the coefficient, the longer it takes for volatility to die out.
- **Rate of convergence:**  $\alpha_1 + \beta$  measures how quickly conditional volatility converges to long-term average. The closer to one, the slower the convergence.

## Generative (Forward) Model

The **predictive prior distribution** PrPD,  $p(y)$ , is the distribution of the unknown but observable  $y$  before considering our sample.

Depends only on the prior and structure of the likelihood - not the observed data!

$$p(y) = \int p(y, \theta) d\theta = \int p(\theta)p(y|\theta)d\theta$$

- **Generative models:** Bayesian models with proper priors.
- **Goal:** To understand the model structure before making the measurements.
- **Methodology:** Visualize simulations from the prior marginal distribution and assess consistency between chosen priors and domain knowledge.
- **Recommendations** (Gabry et al. 2019):
  - At least some mass around extreme but plausible data sets (ex. extreme log-returns).
  - No mass on completely implausible values (ex. negative prices or

## Generative (Forward) Model

Rather than evaluating the PrPD directly, we can let Stan handle it:

```
data {  
    int<lower=0> T;  
    real<lower=0> sigma1;  
}  
  
parameters {  
    real<lower=0> alpha0;  
    real<lower=0, upper=1> alpha1;  
    real<lower=0, upper=(1-alpha1)> beta1;  
}  
  
model {  
    // Priors  
    alpha0 ~ normal(0, 0.5) T[0, ]; // close to zero and small  
    alpha1 ~ beta(0.50, 1.50); // slightly more likely to be close to z  
    beta1 ~ beta(2.50, 0.80); // slightly more likely to be close to c  
}
```

Note: no observations  $y$  in the **data** block - purely prior-driven!

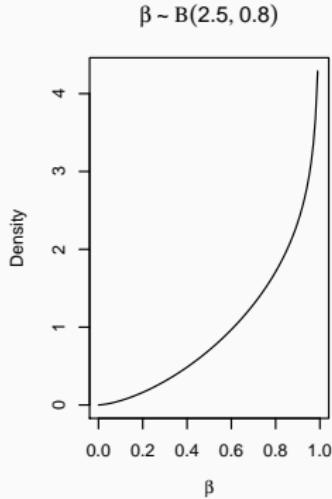
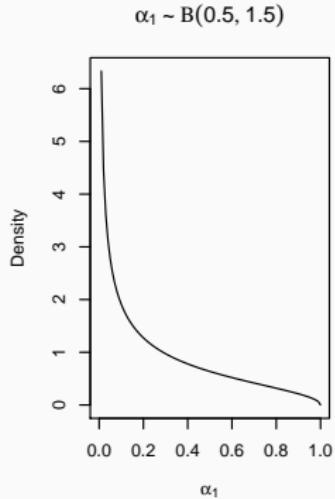
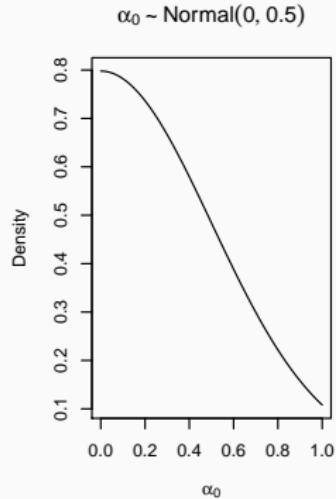
## Priors

From previous experience, we expect that:

- $\alpha_0 > 0$  is very close to zero.
- $\alpha_1 \geq 0$  is close to zero, with values above 0.1 being “relatively large”.
- $\beta \geq 0$  is close to one, with commonly-seen values in the range [0.70, 0.99].

These priors (on their own) do not yield stationarity: stationarity is imposed through the constraints in the **parameters** block.

# Generative (Forward) Model: Priors



**Weakly informative priors:** although we use information from previous empirical studies to **regularize and stabilize** the density, we are conservative.

## Generative (Forward) Model: Generated Quantities

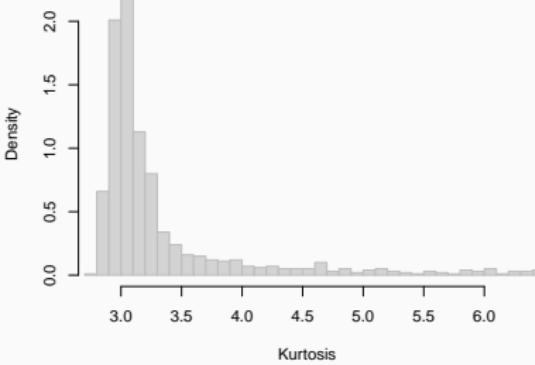
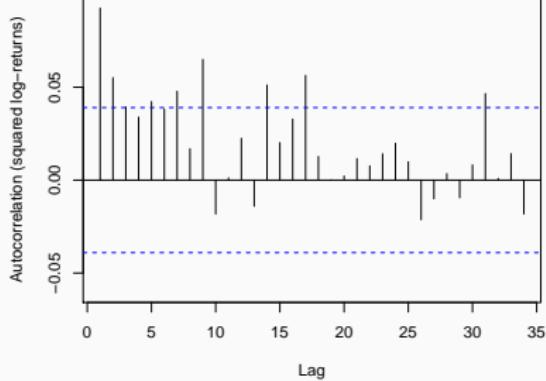
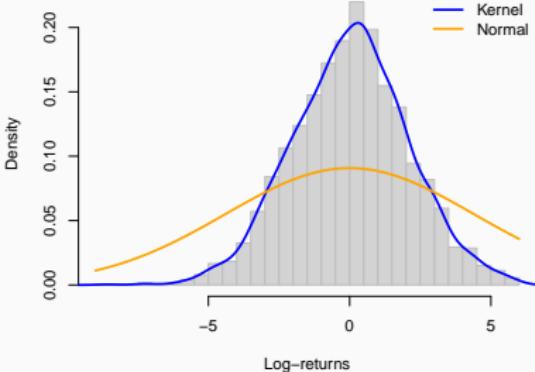
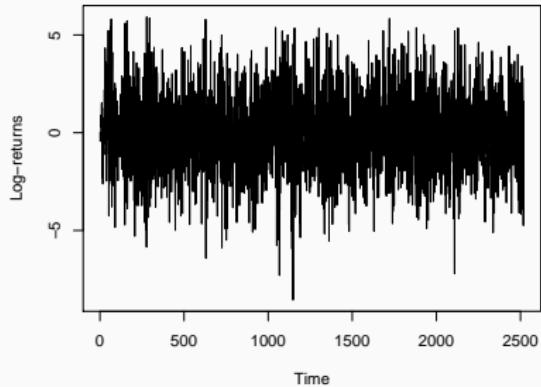
Generate samples from PrPD in the `generated quantities` block:

```
generated quantities {
    real<lower=0> spred[T];
    real ypred[T];

    spred[1] = sigma1;
    ypred[1] = normal_rng(0, sigma1);
    for (t in 2:T) {
        spred[t] = sqrt(
            alpha0
            + alpha1 * pow(ypred[t-1], 2)
            + beta1 * pow(spred[t-1], 2)
        );
        ypred[t] = normal_rng(0, spred[t]);
    }
}
```

If we had data in our model block, this would yield samples from the *posterior predictive density* instead

# Generative (Forward) Model: PrPD



# Software Validation

Before we analyze any estimates, we need to confirm that our software works as intended (Cook et al. 2006; Talts et al. 2018). For  $i \in 1, \dots, N$  replications:

- i. Draw one sample of the parameter vector  $\theta^{(i)}$  from the prior distributions  $p(\theta)$ .
- ii. Draw one sample of the observation vector  $y^{(i)}$  from the sampling distribution  $p(y|\theta^{(i)})$ .
- iii. Use Stan to estimate the model parameters.
- iv. Can the software recover the true parameters systematically?
  - Are the estimates reasonable (for example, do they take impossible values)?
  - Is there any bias (systematic error in estimation)?
  - Is the true value adequately covered by the posterior intervals?

## Software Validation - Step I

$$\theta^{(i)} \sim p(\theta)$$

```
simParams <- function() {  
  # Half-normal  
  alpha0 = abs(rnorm(n = 1, mean = 0, sd = 0.5))  
  alpha1 = rbeta(n = 1, shape1 = 0.5, shape2 = 1.5)  
  beta1 = rbeta(n = 1, shape1 = 2.5, shape2 = 0.8)  
  sigma1 = rexp(n = 1) / 100  
  if (alpha1 + beta1 >= 1) {  
    simParams() ## Accept-reject sampling  
  } else {  
    c(sigma1, alpha0, alpha1, beta1) }  
}
```

## Software Validation - Step II

$$y^{(i)} \sim p(y|\theta^{(i)})$$

```
simGARCH11 <- function(T, sigma1, alpha0, alpha1, beta1) {  
  if (sigma1 <= 0) {stop("Initial sigma must be positive.")}  
  if (alpha0 <= 0) {stop("Alpha0 must be greater than zero.")}  
  if (min(alpha1, beta1) < 0) {stop("Alpha1 and Beta1 can't be negative")}  
  if (alpha1 + beta1 >= 1) {stop("Alpha1 + Beta1 must be < 1.")}  
  
  y <- s <- numeric(T) # Pre-allocate  
  
  s[1] <- sigma1  
  y[1] <- rnorm(n = 1, mean = 0, sd = s[1])  
  
  for (t in 2:T) {  
    s[t] <- sqrt(alpha0 + alpha1 * y[t - 1]^2 + beta1 * s[t - 1]^2)  
    y[t] <- rnorm(n = 1, mean = 0, sd = s[t])  
  }  
  y  
}
```

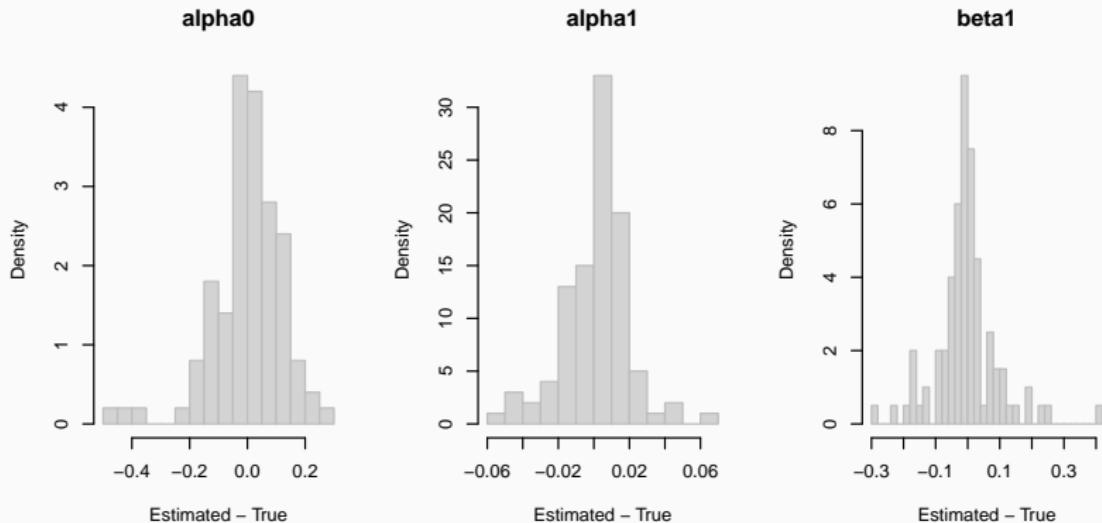
# Software Validation - Step III

```
N <- 100

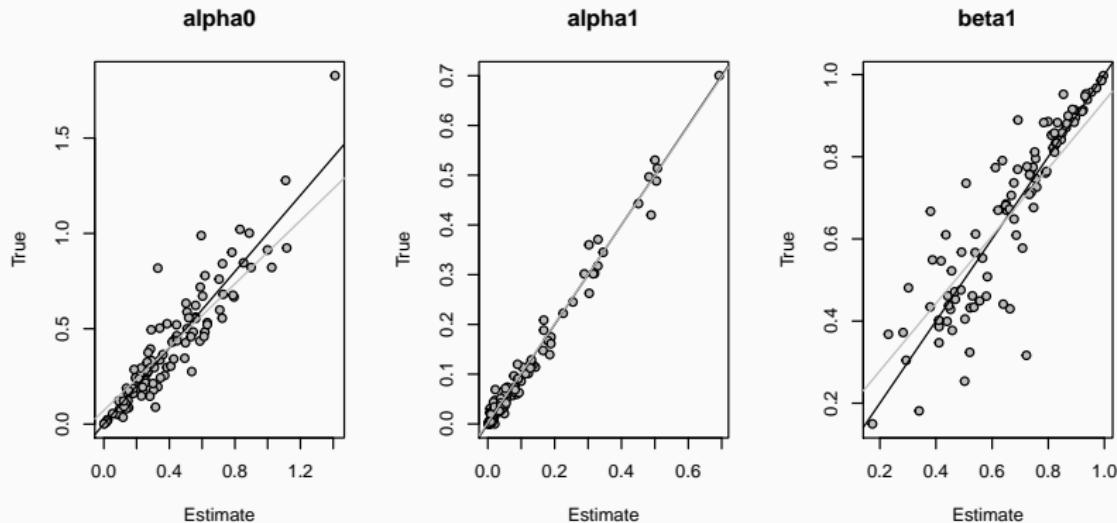
replicate(100, {
  params <- simParams()
  yfake <- simGARCH11(T = 252 * 10, sigma1 = params[1],
                        alpha0 = params[2], alpha1 = params[3],
                        beta1 = params[4])

  fit <- stan(
    file = "stan/garch11.stan",
    data = list(
      T      = length(yfake),
      H      = 1,
      y      = as.numeric(yfake),
      sigma1 = params[1] # Starting volatility
    ))
  list(
    true = params,
    estimates = extract(fit, pars = c("alpha0", "alpha1", "beta1"))
  )
})
```

# Software Validation - Step IV



# Software Validation - Step IV



Other diagnostics:

- The 95% posterior intervals for the parameters  $\alpha_0$ ,  $\alpha_1$ , and  $\beta$  contain the true value in the 89, 97, 93 percent of samples respectively.
- All values for the estimated parameters are within the restricted parameter space.
- MCMC chains mix well. (See, for example **Bayesplot**)

# Model Diagnostics

After constructing a probability model and computing the posterior distribution, we assess the fit of the model to **the data and our substantive knowledge**.

- Have we included **all** our knowledge about the problem?
- What aspects of reality are not captured by the model?
- Suspects: priors, likelihood, model structure, explanatory variables.

# Model diagnostics

- **Sensitivity analysis:** how much does posterior inference change when other reasonable priors and/or models are used?
- Judge by **practical implication:** there is **no true model**.
- Does inference **make sense?**
  - Not all knowledge is included formally in the model.
  - Use the substantive leftovers to analyze the results.
- **External validation:** predict **future** data and compare with future observations.

## Posterior Predictive Checks

---

*If the model fits, replicated data generated under the model should look similar to observed data. To put it another way, the observed data should look plausible under the posterior predictive distribution (Gelman et al. 2013).*

- A self-consistency check.
- A data-informed data generating model.

# Posterior Predictive Checks

The posterior predictive distribution (PPD):

$$p(y^{rep}|y) = \int p(y^{rep}|\theta)p(\theta|y)d\theta$$

- Fit a model  $p(\theta|y)$  to an observed sample  $y$ .
- Draw  $S$  simulated values  $y^{rep}$  (“replications”) from the joint posterior predictive distribution  $p(y^{rep}|y)$ .
- Define a statistic  $T(y)$  that measures the discrepancy between model and data.
- Compare the generated samples to the observed data.

Any systematic differences between the simulations and the data indicate potential failings of the model.

## How to Choose the Quantity $T$ ?

*A model can fail to reflect the data generating process in any number of ways. Compare a variety of statistics to evaluate more than one possible model failure (Gelman et al. 2013).*

- Choose a quantity that reflects aspects relevant to the **scientific purpose**.
- Especially useful to measure features of data **not directly addressed by the model** (ex. ranks, correlations, relationships with explanatory variables).
- **Discard sufficient statistics** because we look for features not explicitly included in the model. Choose statistics that are orthogonal to model parameters instead.

# How to Compare the Quantity $T$ ?

- Numerically:
  - Compare the magnitude  $T(y, \theta^s) - T(y^{rep\ s}, \theta^s)$ .
  - Compute the probability of the replicated data being more extreme than the observed data  
 $\Pr(T(y, \theta^s) \leq T(y^{rep\ s}, \theta^s)) \quad \forall s = 1, \dots, S.$
- Graphically:
  - Histogram of  $T(y, \theta^s) - T(y^{rep\ s}, \theta^s)$  should include zero.
  - Scatterplot  $T(y, \theta^s) \sim T(y^{rep\ s}, \theta^s)$  should be symmetric about the 45° line.

## Posterior Predictive Checks: GARCH(1,1)

```
transformed parameters {
  real<lower=0> sigma[T];
  sigma[1] = sigma1;
  for (t in 2:T) {
    sigma[t] = sqrt(
      alpha0
      + alpha1 * pow(y[t-1], 2)
      + beta1 * pow(sigma[t-1], 2)
    );
  }
}

model {
  // Priors
  alpha0 ~ normal(0, 0.5) T[0, ]; // close to zero and small
  alpha1 ~ beta(0.50, 1.50); // slightly more likely to be close to zero
  beta1 ~ beta(2.50, 0.80); // slightly more likely to be close to one

  // Sampling (likelihood)
  y ~ normal(0, sigma);
}
```

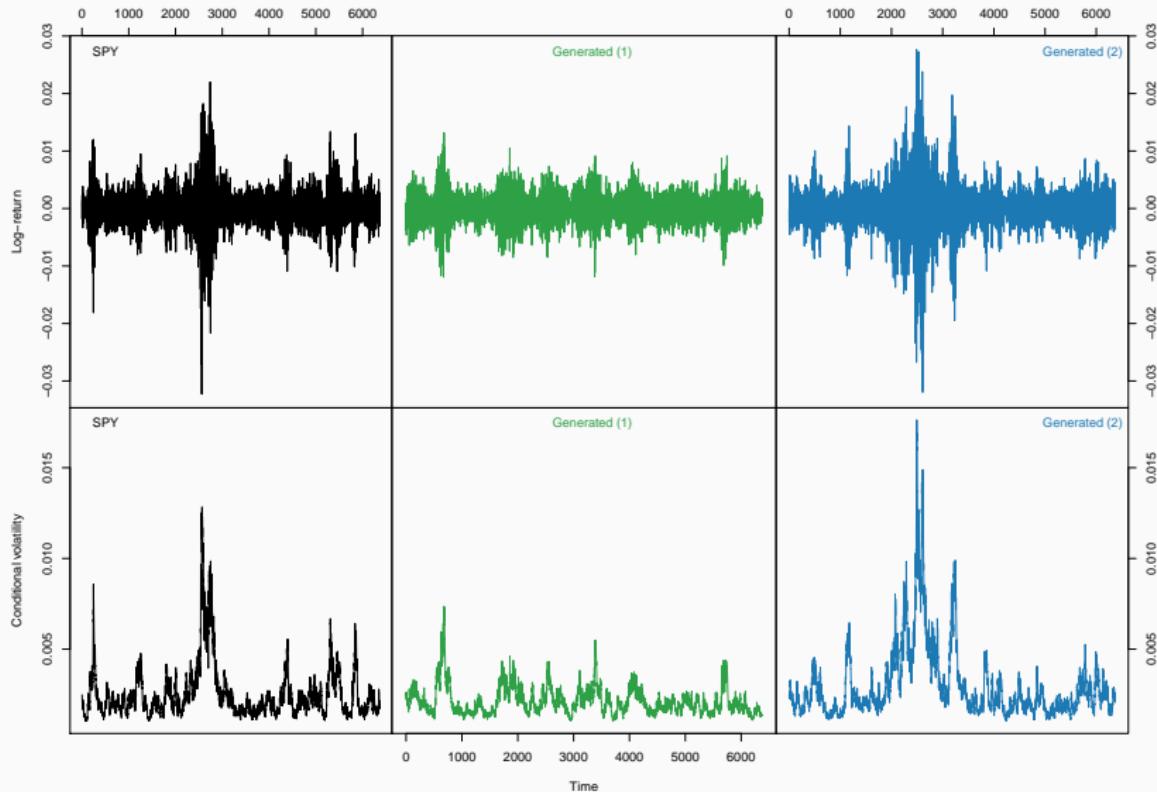
## Posterior Predictive Checks: GARCH(1,1)

```
generated quantities {
    real<lower=0> sfore[H]; real<lower=0> spred[T];
    real yfore[H]; real ypred[T];

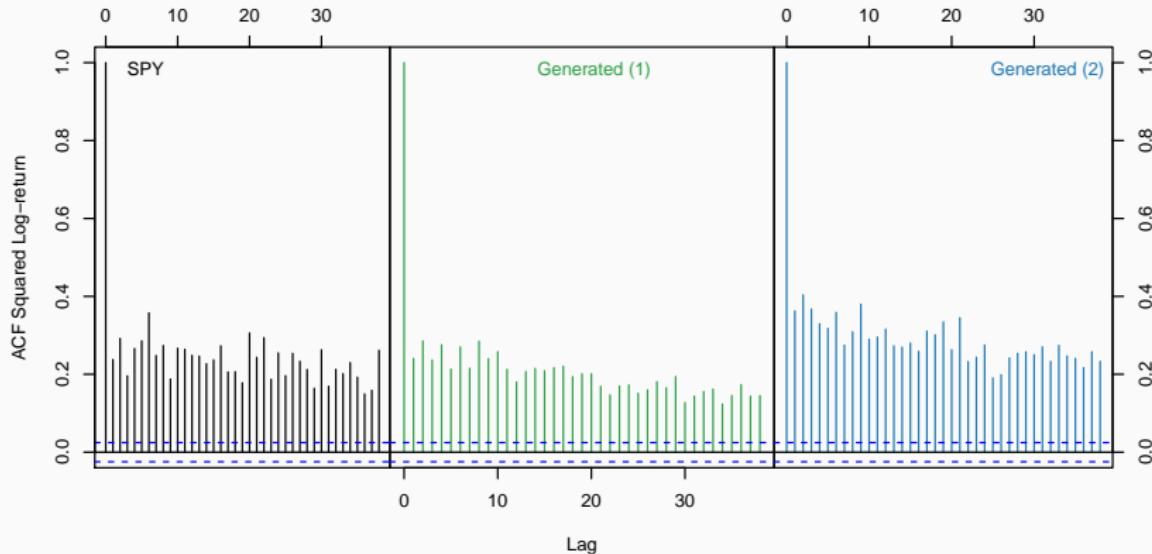
    for (h in 1:H) {
        sfore[h] = sqrt(
            alpha0
            + alpha1 * pow(y[T + h - 1], 2)
            + beta1 * pow(sigma[T + h - 1], 2)
        );
        yfore[h] = normal_rng(0, sfore[h]);
    }

    spred[1] = sigma1; ypred[1] = normal_rng(0, sigma1);
    for (t in 2:T) {
        spred[t] = sqrt(alpha0
            + alpha1 * pow(ypred[t-1], 2)
            + beta1 * pow(spred[t-1], 2)
        );
        ypred[t] = normal_rng(0, spred[t]);
    }
}
```

# PPC: Observed vs. Posterior Predictive Returns

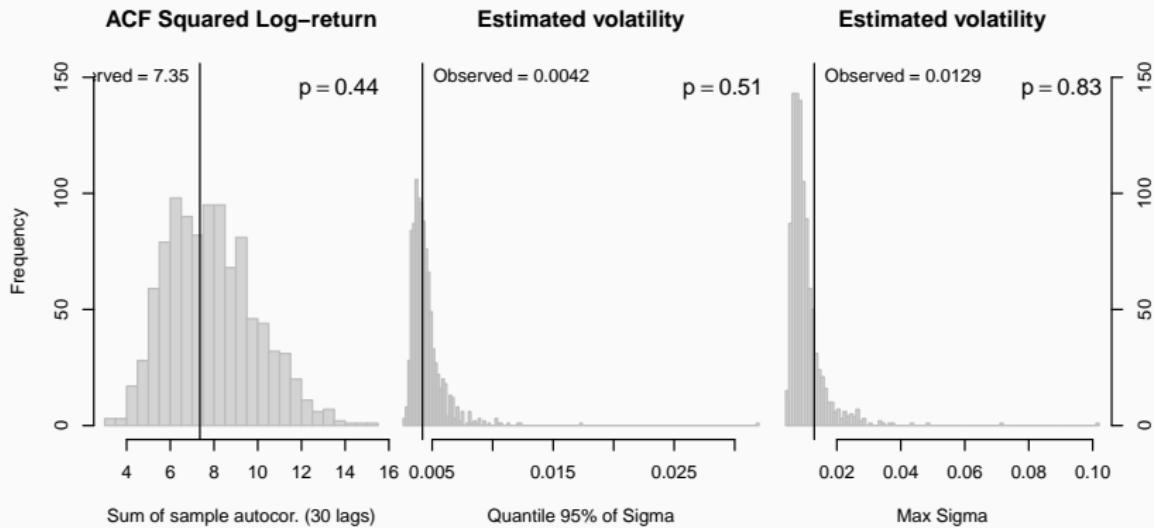


# PPC: Observed vs. Posterior Predictive Returns



Does our model replicate data features that are explicitly modeled?

# PPC: Observed vs. Posterior Predictive Returns



Does our model replicate data features that are **not** explicitly modeled?

# Out-of-Sample Forecast Evaluation

Methodology for **walk-forward validation**:

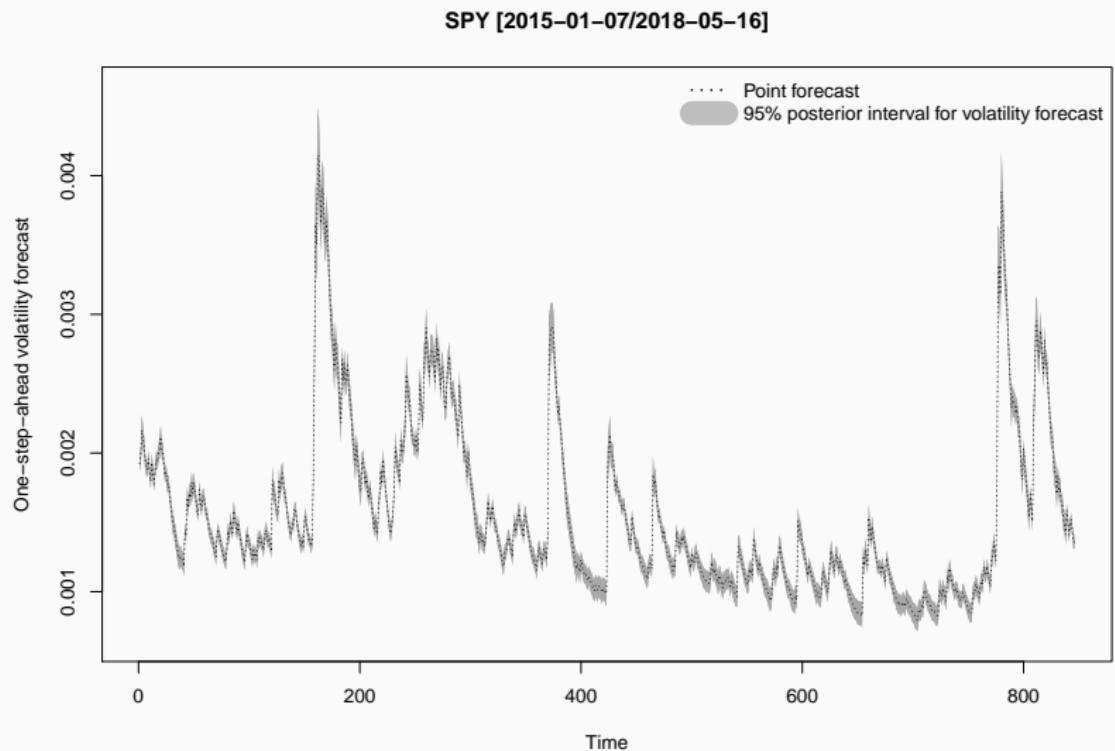
- Extract **subsamples** using a rolling window with 2520 observations each (approximately ten trading years).
- Re-estimate the parameters  $\theta$  every  $k$  days.
- Compute the  $h$ -step ahead **forecast for volatility**  $\hat{\sigma}_{t+h|t}$ .
- Draw a sample from the **expected distribution of log-returns**  $\hat{y}_{t+h|t} \sim \mathcal{N}(0, \hat{\sigma}_{t+h|t}^2)$ .

Outcome:

- Volatility forecasts
- A full distribution of forecasted log-returns

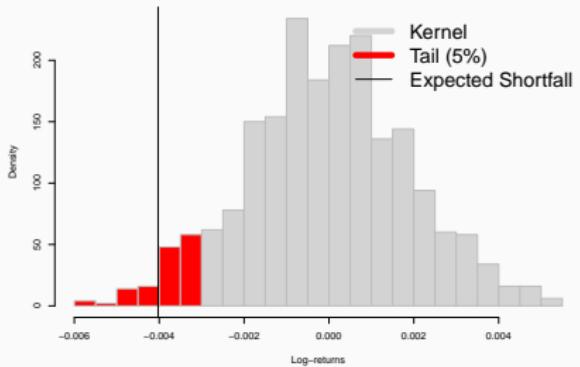
Unlike standard approaches, we account for parameter certainty “for free” (Ardia et al. 2017)

# Parameter uncertainty



# Distribution of expected return

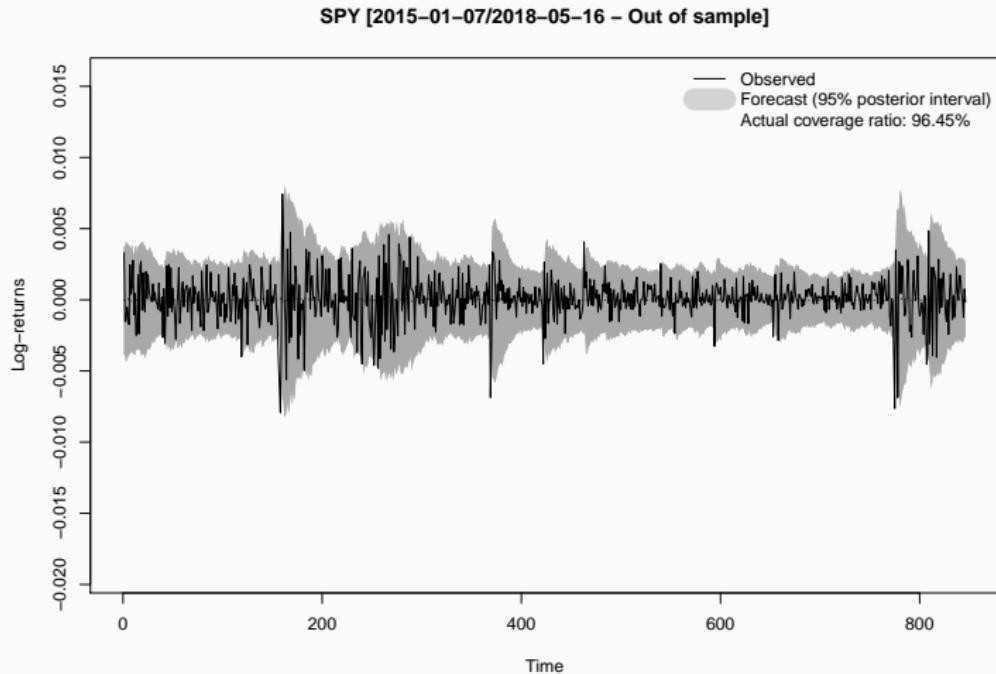
Draw a large sample of expected log-returns  $\hat{y}_{t+1|t}$  and use this empirical distribution to analyze risk:



- Mean/median log-returns.
- Standard deviation, interquartile range.
- Kurtosis.
- Quantiles (ex. VaR).
- Tail characteristics (ex. conditional mean for ES).

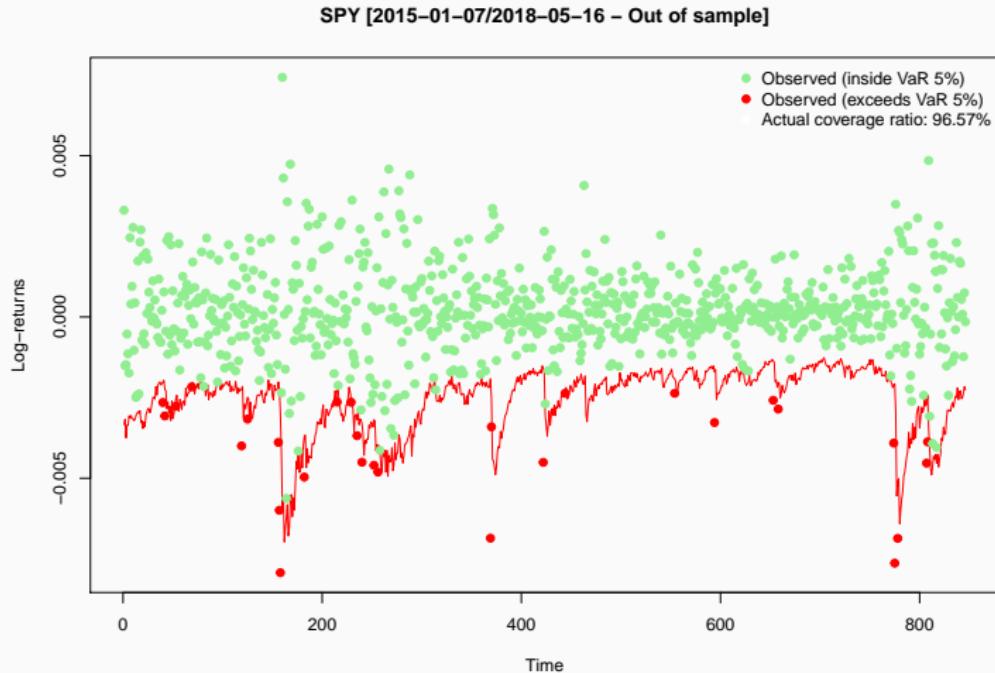
# Coverage ratio

How many times does the (ex-post) observed log-return exceed our  $1 - \alpha$  forecast interval?



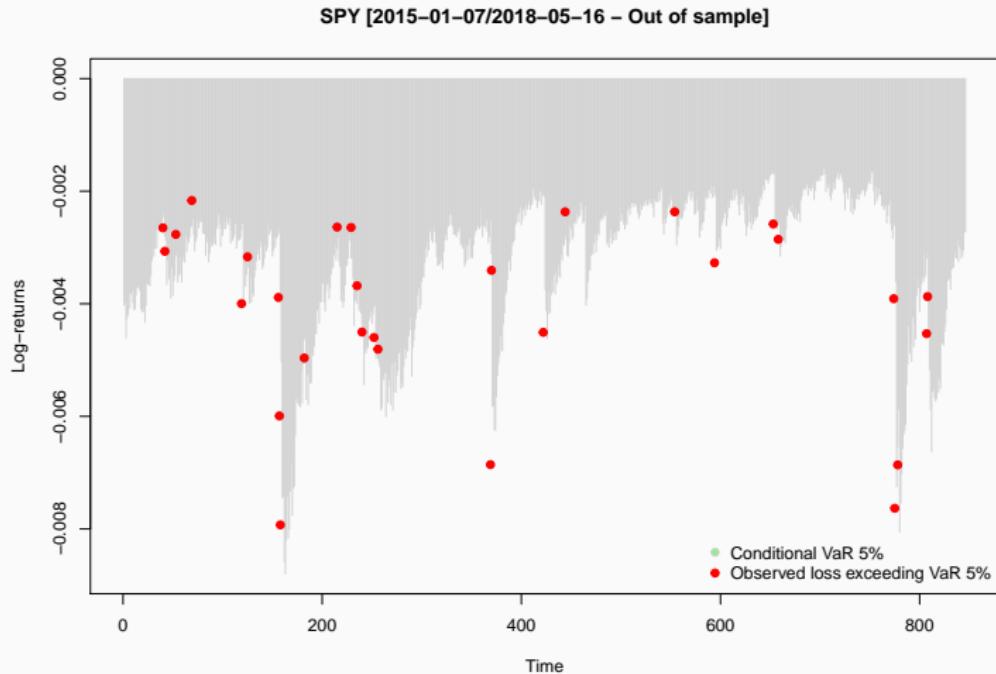
# Value at Risk

How many times does the (ex-post) observed log-return exceed our  $\alpha$  VaR? (Christoffersen 1998; Kupiec 1995)



# Expected Shortfall

How does the (ex-post) observed log-return compare with our  $\alpha$  ES?



## Advanced GARCH Models

---

# Realized GARCH

The “base estimator”  $\sigma_t^2 \approx r_t^2$  is not the only estimator of volatility

Many other “realized volatility” measures exist – see

**TTR::volatility()**

(Garman and Klass 1980; Rogers and Satchell 1991; Yang and Zhang 2000)

Hansen and Huang (2016) and Hansen et al. (2012) augment GARCH dynamics with realized measures

$$r_t \sim \mathcal{N}(\mu, \sigma_t^2)$$

$$\sigma_t^2 = \alpha_0 + \alpha_1 r_{t-1}^2 + \beta_1 \sigma_{t-1}^2$$

$$\varsigma_t \sim \mathcal{N}(\xi + \phi \sigma_t + \delta_1 |r_t| + \delta_2 r_t^2, \nu^2)$$

# Realized GARCH

```
data {
    int<lower=1> T;
    vector[T] return_market;
    vector[T] realized_vol;
}
parameters {
    real mu_market;
    real<lower=0> alpha0;
    real<lower=0,upper=1> alpha1;
    real<lower=0,upper=(1-alpha1)> beta1;

    real<lower=0> sigma_market1; // Starting volatility

    real<lower=0> xi;
    real<lower=0> phi;
    real<lower=0> delta1;
    real<lower=0> delta2;
    real<lower=0> nu;
}
```

# Realized GARCH

```
transformed parameters {
  vector<lower=0>[T] sigma_market;
  sigma_market[1] = sigma_market1;

  for(t in 2:T){
    sigma_market[t] = sqrt(alpha0 +
                           alpha1 * square(return_market[t-1]) +
                           beta1 * square(sigma_market[t-1]));
  }

  rv_market_mean = xi + phi * sigma_market +
    delta_1_rv * fabs(return_market) +
    delta_2_rv * square(return_market);
}

}
```

# Realized GARCH

```
model {
    // Priors
    sigma_market1 ~ normal(sd(return_market),
                           3 * sd(return_market));

    mu_market ~ normal(0, 1);
    alpha0    ~ normal(0, 1);
    alpha1    ~ normal(0, 1);
    beta1     ~ normal(1, 1);
    xi        ~ normal(0, 1);
    phi       ~ normal(1, 1);
    delta1   ~ normal(0, 1);
    delta2   ~ normal(0, 1);
    nu        ~ normal(1, 1);

    // Likelihood
    return_market ~ normal(mu_market, sigma_market);
    realized_vol ~ normal(rv_market_mean, nu);
}
```

## Going Multivariate: Realized Beta GARCH

Hansen et al. (2014) propose *realized beta GARCH*:

- multivariate single-factor GARCH
- each asset has a combination of market volatility and idiosyncratic volatility
- naturally extended to multi-factor and multi-asset models

$$r_t \sim \mathcal{N}(\mu, \sigma_t^2)$$

$$\sigma_t^2 = \alpha_0 + \alpha_1 r_{t-1}^2 + \beta_1 \sigma_{t-1}^2$$

$$s_t \sim \mathcal{N}(\xi + \phi \sigma_t + \delta_1 |r_t| + \delta_2 r_t^2, \nu^2)$$

$$r_t^A \sim \mathcal{N}(\mu^A, \sigma_{A,t}^2)$$

$$\sigma_{A,t}^2 = \alpha_{A,0} + \alpha_{A,1} r_{A,t}^2 + \beta_{A,1} \sigma_{A,t-1}^2 + \gamma \sigma_t^2$$

# Multivariate Skew Normal

Multivariate GARCH requires a multivariate observation distribution:  
Stan (currently) only has the multivariate normal.

We can add the *multivariate skew normal* (Azzalini and Capitanio 1999) as a user-defined function and use it like a built-in distribution:

```
functions{
    real multi_skew_normal_lpdf(vector y, vector mu,
                                vector sigma, vector alpha,
                                matrix omega){
        real retval = 0; int K = rows(y);

        retval += multi_normal_cholesky_lpdf(y | mu,
                                              diag_pre_multiply(sigma, omega));

        for (i in 1:K){
            retval += normal_lcdf(dot_product(alpha, (y - mu) ./ sigma));
        }
        return retval;
    }
}
```

# Realized Beta GARCH

Realized Beta GARCH Stan Code:

```
data {
    int<lower=1> T;
    vector[T] return_market;
    vector[T] return_asset;
    vector<lower=0>[T] realized_vol_market;
}

transformed data{
    // Put both series a single object
    vector[2] returns[T];

    for(t in 1:T){
        returns[t][1] = return_market[t];
        returns[t][2] = return_asset[t];
    }
}
```

# Realized Beta GARCH

```
transformed parameters {
    vector<lower=0>[2] sigma[T];
    vector[T] rv_market_mean;

{
    vector[T] sigma_market;
    vector[T] sigma_asset;

    sigma_market[1] = sigma_market1;
    sigma_asset[1] = sigma_asset1;

    for(t in 2:T){
        // First GARCH dynamics
        sigma_market[t] = sqrt(omega_market +
                               gamma_market * square(sigma_market[t - 1])
                               tau_1_market * fabs(return_market[t-1])
                               tau_2_market * square(return_market[t-1])
                               zeta_market * realized_vol_market[t-1])
    }
}
```

# Realized Beta GARCH

(Continued)

```
sigma_asset[t] = sqrt(omega_asset +
                      gamma_asset * square(sigma_asset[t - 1])
                      beta_asset * square(sigma_market[t - 1])
                      tau_1_asset * fabs(return_asset[t-1]) +
                      tau_2_asset * square(return_asset[t-1]));
}

for(t in 1:T){
  sigma[t][1] = sigma_market[t];
  sigma[t][2] = sigma_asset[t];
}

// Expected realized vol
rv_market_mean = xi + phi * sigma_market +
                  delta_1_rv * fabs(return_market) +
                  delta_2_rv * square(return_market);
}
```

# Realized Beta GARCH

---

Let Stan automatically enforce complex stationarity constraints for us:

```
parameters {
    vector[2] mu;
    vector[2] alpha;
    cholesky_factor_corr[2] L;

    real<lower=0> omega_market;
    real<lower=0,upper=1> zeta_market;
    real<lower=0,upper=1> gamma_market;
    real<lower=0,upper=(1-gamma_market)> tau_1_market;
    real<lower=0,upper=(1-gamma_market-tau_1_market)> tau_2_market;

    real<lower=0> sigma_market1;
```

# Realized Beta GARCH

---

(Continued)

```
real<lower=0> omega_asset;
real<lower=-1,upper=1> beta_asset;
real<lower=0,upper=(1-beta_asset)> gamma_asset;
real<lower=0,upper=(1-beta_asset-gamma_asset)> tau_1_asset;
real<lower=0,upper=(1-beta_asset-gamma_asset-tau_1_asset)> tau_2_asset;

real<lower=0> sigma_asset1;

real xi;
real phi;
real<lower=0> delta_1_rv;
real<lower=0> delta_2_rv;
real<lower=0> rv_sd;
}
```

# Realized Beta GARCH

Priors here are robust MLE + 10 standard errors for SPY + 1-month e-mini:

```
model {
    mu          ~ normal(0, 1);
    alpha       ~ normal(0, 1);
    L           ~ lkj_corr_cholesky(1);
    beta_asset  ~ normal(0, 1);
    zeta_market ~ normal(0, 1);

    omega_market ~ normal(0.002, 0.025);
    gamma_market ~ normal(0.8, 0.6);
    tau_1_market ~ normal(0, 0.1);
    tau_2_market ~ normal(0.1, 0.7);
```

# Realized Beta GARCH

(Continued)

```
omega_asset ~ normal(0.002, 0.025);
gamma_asset ~ normal(0.8, 0.6);
tau_1_asset ~ normal(0, 0.1);
tau_2_asset ~ normal(0.1, 0.7);

xi ~ normal(0.02, 0.6);
phi ~ normal(15, 60);
delta_1_rv ~ normal(1.15, 8);
delta_2_rv ~ normal(1.15, 14);
rv_sd ~ normal(0.05, 0.25);
```

# Realized Beta GARCH

---

(Continued)

```
// Initialize initial vol with weak prior
sigma_market1 ~ normal(sd(return_market), 5 * sd(return_market));
sigma_asset1 ~ normal(sd(return_asset), 5 * sd(return_asset));

// Likelihood
realized_vol_market ~ normal(rv_market_mean, rv_sd);
for(t in 1:T){
    returns[t] ~ multi_skew_normal_chol(mu, sigma[t], alpha, L);
}
}
```

# Markov-Switching GARCH

“Regime Switching” GARCH: multiple GARCH processes with different parameters – return for each day is drawn from a GARCH processes selected by an underlying Hidden Markov Model (Haas et al. 2004a,b)

$$r_t | x_t = k \sim \text{GARCH}(\alpha_{0,k}, \alpha_{1,k}, \beta_{1,k}) \quad (\text{GARCH})$$

$$x_t | x_{t-1} = k \sim \text{Categorical}(p_k) \quad (\text{HMM})$$

See Damiano et al. (2018) for background on HMMs in Stan and Ardia (R/Fin 2017) for more details on Markov-switching GARCH models generally

# Stan Implementation

luisdamiano/stancon2018/stan/hmm\_garch.stan

```
transformed parameters {
...
// GARCH Component
// -----
// Initialize at unconditional variances
sigma_t[1, 1] = alpha0[1] / (1 - alpha1[1] - beta1[1]); // Low-vol
sigma_t[1, 2] = alpha0[2] / (1 - alpha1[2] - beta1[2]); // High-vol

// GARCH dynamics rolling forward
for(t in 2:T){
  for(i in 1:2){
    sigma_t[t, i] = sqrt(alpha0[i] +
                          alpha1[i] * pow(y[t-1], 2) +
                          beta1[i] * pow(sigma_t[t-1, i], 2));
  }
}
}
```

# Stan Implementation

luisdamiano/stancon2018/stan/hmm\_garch.stan

```
transformed parameters {
...
// HMM Component
// -----
// Calculate log p(state at t = j | history up to t) recursively
// Markov property allows us to do one-step updates
real accumulator[2];

// Assume initial equal distribution among two states
// Better model would be to weight by HMM stationary distribution
log_alpha[1, 1] = log(0.5) + normal_lpdf(y[1] || 0, sigma_t[1, 1]);
log_alpha[1, 2] = log(0.5) + normal_lpdf(y[1] || 0, sigma_t[1, 2]);

for(t in 2:T){
    for(j in 1:2) { // Current state
        for(i in 1:2) { // Previous state
            accumulator[i] = log_alpha[t-1, i] + // Probability from previous obs
                log(A[i, j]) + // Transition probability
                // (Local) likelihood / evidence for given state
                normal_lpdf(y[t] || 0, sigma_t[t-1, i]);
        }
        log_alpha[t, j] = log_sum_exp(accumulator);
    }
}
```

## References

---

## References 1

- Alexander, Carol (2008). *Practical Financial Econometrics*. Market Risk Analysis 2. Wiley. ISBN: 978-0470998014.
- Ardia, David, Jeremy Kolly, and Denis-Alexandre Trottier (2017). "The Impact of Parameter and Model Uncertainty on Market Risk Predictions from GARCH-Type Models". *Journal of Forecasting* 36.7, pp. 808–823. DOI: [10.1002/for.2472](https://doi.org/10.1002/for.2472).
- Azzalini, A. and A. Capitanio (1999). "Statistical Applications of the Multivariate Skew Normal Distribution". *Journal of the Royal Statistical Society, Series B: Statistical Methodology* 61.3, pp. 579–602. DOI: [10.1111/1467-9868.00194](https://doi.org/10.1111/1467-9868.00194).
- Bollerslev, Tim (1986). "Generalized autoregressive conditional heteroskedasticity". *Journal of Econometrics* 31.3, pp. 307–327. DOI: [10.1016/0304-4076\(86\)90063-1](https://doi.org/10.1016/0304-4076(86)90063-1).

## References 2

---

- Bollerslev, Tim (2010). "Glossary to ARCH". *Volatility and Time Series Econometrics: Essays in Honor of Robert Engle*. Ed. by Tim Bollerslev, Jeffrey Russell, and Mark Watson. Advanced Texts in Econometrics. Oxford University Press. DOI: [10.1093/acprof:oso/9780199549498.003.0008](https://doi.org/10.1093/acprof:oso/9780199549498.003.0008).
- Christoffersen, Peter F. (1998). "Evaluating Interval Forecasts". *International Economic Review* 39.4, pp. 841–862. DOI: [10.2307/2527341](https://doi.org/10.2307/2527341).
- Cont, R. (2001). "Empirical properties of asset returns: stylized facts and statistical issues". *Quantitative Finance* 1.2, pp. 223–236. DOI: [10.1080/713665670](https://doi.org/10.1080/713665670).
- Cook, Samantha R., Andrew Gelman, and Donald B. Rubin (2006). "Validation of Software for Bayesian Models Using Posterior Quantiles". *Journal of Computational and Graphical Statistics* 15.3, pp. 675–692. DOI: [10.1198/106186006X136976](https://doi.org/10.1198/106186006X136976).

## References 3

---

- Damiano, Luis, Brian Peterson, and Michael Weylandt (2018). "A Tutorial on Hidden Markov Models using Stan". *StanCon 2018: Proceedings of the Stan User's Conference 2018*. DOI: [10.5281/zenodo.1284341](https://doi.org/10.5281/zenodo.1284341).
- Engle, Robert F. (1982). "Autoregressive Conditional Heteroscedasticity with Estimates of the Variance of United Kingdom Inflation". *Econometrica* 50.4, pp. 987–1007. DOI: [10.2307/1912773](https://doi.org/10.2307/1912773).
- Gabry, Jonah, Daniel Simpson, Aki Vehtari, Michael Betancourt, and Andrew Gelman (2019). "Visualization in Bayesian workflow". *Journal of the Royal Statistical Society, Series A: Statistics in Society* 182.2, pp. 389–402.
- Garman, Mark B. and Michael J. Klass (1980). "On the Estimation of Security Price Volatilities from Historical Data". *Journal of Business* 53.1, pp. 67–78. DOI: [10.1086/296072](https://doi.org/10.1086/296072).

## References 4

---

- Gelman, Andrew (2004). "Exploratory Data Analysis for Complex Models". *Journal of Computational and Graphical Statistics* 13.4, pp. 755–779. DOI: [10.1198/106186004X11435](https://doi.org/10.1198/106186004X11435).
- Gelman, Andrew, John B. Carlin, Hal S. Stern, David B. Dunson, Aki Vehtari, and Donald B. Rubin (2013). *Bayesian Data Analysis*. 3rd. Texts in Statistical Science. CRC Press.
- Haas, Markus, Stefan Mittnik, and Marc S. Paoletta (2004a). "A New Approach to Markov-Switching GARCH Models". *Journal of Financial Econometrics* 2.1, pp. 493–530. DOI: [10.1093/jjfinec/nbh020](https://doi.org/10.1093/jjfinec/nbh020).
- (2004b). "Mixed Normal Conditional Heteroskedasticity". *Journal of Financial Econometrics* 2.1, pp. 211–250. DOI: [10.1093/jjfinec/nbh009](https://doi.org/10.1093/jjfinec/nbh009).

## References 5

- Hansen, Peter Reinhard and Zhuo Huang (2016). "Exponential GARCH Modeling with Realized Measures of Volatility". *Journal of Business & Economic Statistics* 34.2, pp. 269–287. DOI: [10.1080/07350015.2015.1038543](https://doi.org/10.1080/07350015.2015.1038543).
- Hansen, Peter Reinhard, Zhuo Huang, and Howard Howan Shek (2012). "Realized GARCH: A Joint Model for Returns and Realized Measures of Volatility". *Journal of Applied Econometrics* 27, pp. 877–906. DOI: [10.1002/jae.1234](https://doi.org/10.1002/jae.1234).
- Hansen, Peter Reinhard, Asger Lunde, and Valeri Voev (2014). "Realized Beta GARCH: A Multivariate GARCH Model with Realized Measures of Volatility". *Journal of Applied Econometrics* 29, pp. 774–799. DOI: [10.1002/jae.2389](https://doi.org/10.1002/jae.2389).
- Kastner, Gregor (2016). "Dealing with Stochastic Volatility in Time Series using the R Package stochvol". *Journal of Statistical Software* 69.5. DOI: [10.18637/jss.v069.i05](https://doi.org/10.18637/jss.v069.i05).

## References 6

- Kim, Sangjoon, Neil Shephard, and Siddhartha Chib (1998). “Stochastic Volatility: Likelihood Inference and Comparison with ARCH Models”. *The Review of Economic Studies* 65.3, pp. 361–393.
- Kupiec, Paul H. (1995). “Techniques for Verifying the Accuracy of Risk Measurement Models”. *Journal of Derivatives* 3.2, pp. 73–84. DOI: [0.3905/jod.1995.407942](https://doi.org/10.3905/jod.1995.407942).
- Rogers, L.C.G. and S.E. Satchell (1991). “Estimating Variance from High, Low, and Closing Prices”. *Annals of Applied Probability* 1.4, pp. 504–512. DOI: [10.1214/aoap/1177005835](https://doi.org/10.1214/aoap/1177005835).
- Talts, Sean, Michael Betancourt, Daniel Simpson, Aki Vehtari, and Andrew Gelman (2018). “Validating Bayesian Inference Algorithms with Simulation-Based Calibration”. *ArXiv* 1804.06788. URL: <https://arxiv.org/abs/1804.06788>.

## References 7

Yang, Dennis and Qiang Zhang (2000). "Drift-Independent Volatility Estimation Based on High, Low, Open, and Close Prices". *Journal of Business* 73.3, pp. 477–492. doi: 10.1086/209650.