```
data {
  int<lower=0> T;
  array[T] real r;
  real h1;
  real lambda_0;
  real xi_0;
  real theta_0;
  real alpha_0;

}
parameters {
  real mu;

  real<lower=0> omega0;
  real alpha0;
  real alpha1;
  real<lower=0, upper=1> beta;

  real<lower=0> omega1;
  real<lower=0, upper=1> gamma;
  real<lower=gamma, upper=1> rho;

  real omega2;
  real tau;

  real<lower=0> sigma_2;

  // https://mc-stan.org/docs/2_19/reference-manual/initialization.html


}
transformed parameters {
  array[T] real<lower=0> h;
  h[1] = h1;

  array[T] real<lower=0> lambda;
  lambda[1] = lambda_0;

  array[T] real xi; // no need in any constrains
  xi[1] = xi_0;

  array[T] real theta; // no need in any constrains
  theta[1] = theta_0;

  array[T] real<lower=0> alpha;
  alpha[1] = alpha_0;

  for (t in 2:T) {

    alpha[t] = exp(alpha0 + alpha1 * (lambda[t-1] + xi[t-1])   );
```

```
    h[t] = sqrt(omega0 + alpha[t] * pow( (r[t - 1] - mu), 2) + beta * pow( h[t - 1], 2) ); // simple GARCH

    lambda[t] = omega1 + rho * lambda[t-1] + gamma * xi[t-1]; // constraints on parameters should ensure positivity

    theta[t] = omega2 + tau * (r[t - 1] - mu);

    vector[21] lp_jump_k;
    for(k in 0:20){

    lp_jump_k[k+1] = normal_lpdf(r[t] |   mu + theta[t] * (k - lambda[t]), sqrt( pow(h[t], 2) + k * sigma_2) ) + poisson_lpmf(k | lambda[t]);
    // mean definition according to 2004 paper
    // https://mc-stan.org/docs/stan-users-guide/summing-out-the-responsibility-parameter.html
    // recovering posterior mixture proportions

    }

    real lp_jump = log_sum_exp(lp_jump_k); // bottom part of equation (6) in 2002 paper

    vector[21] jump_k;
    for(j in 0:20) {

    jump_k[j+1] = j*exp( lp_jump_k[j+1] - lp_jump);
    // to get first part of (5) equation in 2002 paper

    }

    real sum_jump = sum(jump_k); // inference on average number of jumps at t: xi[t] + lambda[t]

    xi[t] = sum_jump - lambda[t];

    // Print statement to check the model
    // print("t = ", t, ", h[t] = ", h[t], ", lambda[t] = ", lambda[t], ", xi[t] = ", xi[t], ", gi[t] = ", gi[t]);

  }

}
model {

  mu ~ normal(0, 0.1);

  omega0 ~ normal(0, 0.5);
```

```
  alpha0 ~ normal(-2.5, 1);
  alpha1 ~ normal(0, 1);
  beta ~ beta(2.5, 1);

  omega1 ~ normal(0, 0.15);
  gamma ~ normal(0.3, 0.1);
  rho ~ normal(0.4, 0.1);

  omega2 ~ normal(0, 0.5);
  tau ~ normal(0, 0.5);

  sigma_2 ~ exponential(1);

  for (t in 2:T) {

  vector[21] lp_jump_k;
     for(k in 0:20) {

     lp_jump_k[k+1] = normal_lpdf(r[t] | mu + theta[t] * (k - lambda[t]), sqrt( pow(h[t],
2) + k * sigma_2) ) + poisson_lpmf(k | lambda[t]);
     // mean definition according to 2004 paper
     // it does ot see vector calculations from the transformed variables section, so
repeat

  }

  target += log_sum_exp( lp_jump_k );
  // += means adding to the likelihood function

}

}
generated quantities {
  vector[T] log_lik;
  vector[T] r_pred;

  for(t in 1:T) {
  log_lik[t] = 0;
  r_pred[t] = 0;
  }

  // log_lik
  for (t in 2:T) {
     vector[21] lp_jump_k;
     for(k in 0:20) {
        lp_jump_k[k+1] = normal_lpdf(r[t] | mu + theta[t] * (k - lambda[t]), sqrt(pow(h[t],
2) + k * sigma_2)) + poisson_lpmf(k | lambda[t]);
     }

     real lp_jump = log_sum_exp(lp_jump_k);
```

```
      log_lik[t] = lp_jump;
   }

   // r_pred
   for (t in 2:T) {
      vector[21] p_jump_k;
      for(k in 0:20) {
         p_jump_k[k+1] = normal_rng(mu + theta[t] * (k - lambda[t]), sqrt(pow(h[t], 2) +
k * sigma_2)) * exp( poisson_lpmf(k | lambda[t]) );
      }

      r_pred[t] = sum(p_jump_k);

   }

}
```

```
###############

data {
   int<lower=0> T;
   array[T] real r;

   array[T] real lambda;
   array[T] real theta;


   real sigma1;
}
parameters {
   real mu;
   real<lower=0> omega;
   real<lower=0,upper=1> alpha;
   real<lower=0,upper=(1-alpha)> beta;

   real rho1;
   real rho2;
   real rho3;
   real rho4;


}
transformed parameters {

   array[T] real<lower=0> sigma;
   sigma[1] = sigma1;
```

```
  sigma[2] = sigma1;

  array[T] real mu_trans;
  mu_trans[1] = mu;
  mu_trans[2] = mu;


  for (t in 3:T) {

    mu_trans[t] = mu +
                  rho1 * theta[t-1] + rho2 * theta[t-2] +
                  rho3 * lambda[t-1] + rho4 * lambda[t-2];

    sigma[t] = sqrt(omega
                  + alpha * pow(r[t-1] - mu, 2)
                  + beta * pow(sigma[t-1], 2));
  }

}
model {

  rho1 ~ normal(0,0.5);
  rho2 ~ normal(0,0.5);
  rho3 ~ normal(0,0.5);
  rho4 ~ normal(0,0.5);

  omega ~ normal(0, 1);
  alpha ~ beta(1, 2.5);
  beta ~ beta(2.5, 1);



  for (t in 3:T) {

   target+= normal_lpdf(r[t] | mu_trans[t], sigma[t]);

  }

}
generated quantities {
  vector[T] log_lik;

  for(t in 1:T) {
  log_lik[t] = 0;
  }
  for(t in 3:T) {
    // log_lik

    log_lik[t] = normal_lpdf(r[t] | mu_trans[t], sigma[t]);
```

```
  }


}

##############

library(rstan) # obsxerve startup messages
options(mc.cores = parallel::detectCores())
rstan_options(auto_write = TRUE)

setwd("~/Desktop/Thesis/BASICS")


# MODEL GARCH-ARJI

fit <- stan(
  file = "garch_arji_2004_assym_with_priors_DONE2.stan",
  data = list(
    T = length(lit_returns), # Length of the series
    r = as.numeric(lit_returns), # The observed series (daily log-returns)
    h1 = sd(lit_returns),
    lambda_0 = 0,
    xi_0 = 0,
    theta_0 = 0,
    alpha_0 = 0
  ), chains=1, iter = 4000)



traceplot(fit)
print(fit)
print(fit, pars = "h")
print(fit, pars = "lambda")
print(fit, pars = "xi")

#####
h <- extract(fit, pars = "h")
h <- as.data.frame(h)

h_mean <- sapply(10:ncol(h), function(i) mean(h[,i]))
plot(h_mean, type = "l")

lambda <- extract(fit, pars = "lambda")
lambda <- as.data.frame(lambda)

lambda_mean <- sapply(10:ncol(lambda), function(i) mean(lambda[,i]))
plot(lambda_mean, type = "l")

xi <- extract(fit, pars = "xi")
```

```
xi <- as.data.frame(xi)

xi_mean <- sapply(10:ncol(xi), function(i) mean(xi[,i]))
plot(xi_mean, type = "l")
mean(xi_mean)
sd(xi_mean)

omega2 <- extract(fit, pars = "omega2")
hist(omega2[[1]])

theta <- extract(fit, pars = "theta")
theta <- as.data.frame(theta)

theta_mean <- sapply(10:ncol(theta), function(i) mean(theta[,i]))
plot(theta_mean, type = "l")


length(index(lit[11:nrow(lit),]))
index_xts <- index(lit[11:nrow(lit),])

h_mean_xts <- as.xts(h_mean, order.by = index(lit[11:nrow(lit),]))
plot.xts(h_mean_xts)

lambda_mean_xts <- as.xts(lambda_mean, order.by = index(lit[11:nrow(lit),]))
plot(lambda_mean_xts)

xi_mean_xts <- as.xts(xi_mean, order.by = index(lit[11:nrow(lit),]))
plot(xi_mean_xts)

theta_mean_xts <- as.xts(theta_mean, order.by = index(lit[11:nrow(lit),]))
plot(theta_mean_xts)

########

library(loo)
ll <- extract_log_lik(fit)
waic(ll)
loo(fit)

########
r_pred <- extract(fit, pars = "r_pred")
r_pred <- as.data.frame(r_pred)

#mu <- extract(fit, pars = "mu")
#mu <- as.data.frame(mu)
#mu_mean <- sapply(1:ncol(mu), function(i) mean(mu[,i]))

library(rethinking)
?PI
r_pred_ci <- apply(r_pred, 2, PI, prob=0.99)
```

```
r_pred_ci <- t(r_pred_ci)

r_pred_ci_xts <- as.xts(r_pred_ci, order.by = index(lit[2:nrow(lit),]))
r_pred_ci_xts <- cbind(r_pred_ci_xts, lit_returns)
plot.xts(r_pred_ci_xts)

# MODEL GARCH
fit_garch <- stan(
  file = "GARCH for LIT.stan",
  data = list(
    T = length(lit_returns), # Length of the series
    r = as.numeric(lit_returns), # The observed series (daily log-returns)
    sigma1 = sd(lit_returns)
  ), chains=1, iter = 4000)

compare(fit, fit_garch)
PSIS(fit)
PSIS(fit_garch)



r_pred_garch <- extract(fit_garch, pars = "r_pred_garch")
r_pred_garch <- as.data.frame(r_pred_garch)

r_pred_garch_ci <- apply(r_pred_garch, 2, PI, prob=0.99)
r_pred_garch_ci <- t(r_pred_garch_ci)

r_pred_garch_ci_xts <- as.xts(r_pred_garch_ci, order.by = index(lit[2:nrow(lit),]))
r_pred_garch_ci_xts <- cbind(r_pred_garch_ci_xts, lit_returns)
plot.xts(r_pred_garch_ci_xts)


#####
lit_returns_short <- lit_returns[10:length(lit_returns)]
lit_returns_short_xts <- as.xts(lit_returns_short, order.by = index(lit[11:nrow(lit),]))


###############
library(rstan) # observe startup messages
options(mc.cores = parallel::detectCores())
rstan_options(auto_write = TRUE)

setwd("~/Desktop/Thesis/BASICS")

fit_mean <- stan(
  file = "GARCH DOES include JUMPS in mean.stan",
  data = list(
    T = nrow(df), # Length of the series
    r = as.vector(df$TSLA.Adjusted), # The observed series (daily log-returns)
    theta = as.vector(df$theta),
```

```
    lambda = as.vector(df$lambda),
    sigma1 = sd(df$TSLA.Adjusted)
  ), chains=1, iter = 3000)

print(fit_mean)
library(loo)
ll_fit <- extract_log_lik(fit_mean)
waic(ll_fit)

fit2 <- stan(
  file = "GARCH DOES NOT include JUMPS in mean.stan",
  data = list(
    T = nrow(df), # Length of the series
    r = as.vector(df$TSLA.Adjusted), # The observed series (daily log-returns)
    sigma1 = sd(df$TSLA.Adjusted)
  ), chains=1, iter = 3000)
print(fit2)

ll_fit2 <- extract_log_lik(fit2)
waic(ll_fit2)

library(rethinking)
compare(fit_mean, fit2)
```