Modeling and Analysis of Time Series Data

Chapter 16: A case study of financial volatility and a POMP model with observations driving latent dynamics

Edward L. Ionides

## Outline

Introduction

- Returns on investments in stock market indices or large companies are often found to be approximately uncorrelated.
- If investment returns are substantially correlated, investors can study their time series behavior and make money.
- If the investment is non-liquid (i.e., not reliably tradeable), or expensive to trade, then it might be hard to make money even if you can statistically predict a positive expected return.
- Otherwise, the market may notice a favorable investment opportunity. More buyers will lead to higher prices, and the opportunity will disappear.
- Consequently, most readily traded investments (e.g., stock market indices, or stock of large companies) have close to uncorrelated returns.
- The variability of the returns (called the volatility) can fluctuate considerably. Understanding this volatility is important for quantifying and managing the risk of investments.

- Recall the daily S&P 500 data that we saw earlier, in Chapter 3.

```r
dat <- read.table("sp500.csv",sep=",",header=TRUE)
plot(as.Date(dat$Date),dat$Close,
  xlab="date",ylab="S&P 500",type="l")
plot(as.Date(dat$Date),dat$Close, log="y",
  xlab="date",ylab="S&P 500",type="l")
```
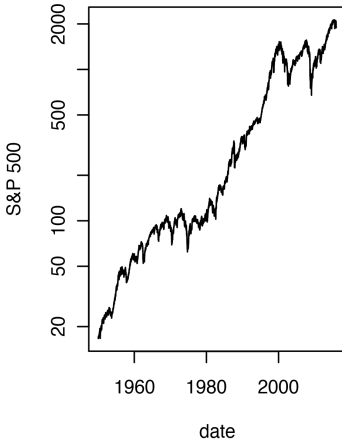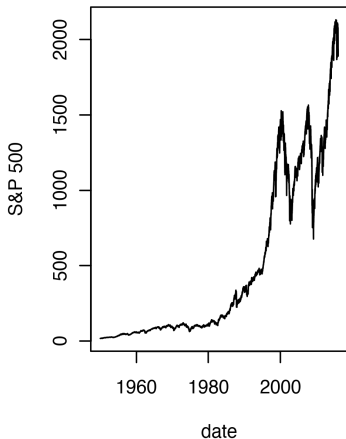
Returns, absolute returns, and autocorrelation

- We write $\{z_n, n = 1, \ldots, N\}$ for the S&P 500 index value.
- We write the return, i.e., the difference of the log of the index, as

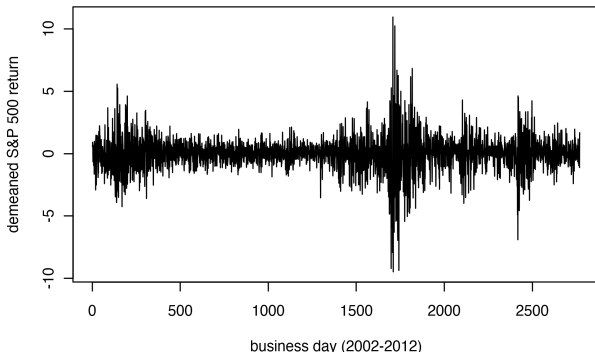$$y_n^* = \log(z_n) - \log(z_{n-1}).$$

* We saw in Chapter 3 that $y_{2:N}^*$ has negligible sample autocorrelation.

- However, the absolute deviations from average,

$$a_n^* = \left| y_n^* - \frac{1}{N-1} \sum_{k=2}^{N} y_k^* \right|$$

have considerable sample autocorrelation.

- We fit models to the demeaned daily returns for the S&P 500 index for 2002-2012, to compare with Bretó (2014).

business day (2002-2012)

Question 16.1. Is it appropriate to fit a stationary model to this series, or do we have evidence for violation of stationarity? Explain.

These are periods with higher & lower volatility. If these periods are modeled as stochastic & stationary episodes, we could use a stationary model. If we cut the data into intervals, we might treat each interval differently - not stationary model.

The ARCH model

- ARCH and GARCH models are widely used for financial time series modeling. We follow Cowpertwait and Metcalfe (2009) to introduce these models; see also Section 5.4 of (Shumway and Stoffer, 2017).
- An order p **autoregressive conditional heteroskedasticity** model, known as ARCH(p), has the form

$$Y_n = \epsilon_n \sqrt{V_n},$$

where $\epsilon_{1:N}$ is white noise and

$$V_n = \alpha_0 + \sum_{j=1}^{p} \alpha_j Y_{n-j}^2.$$

autoregression for the variance.

- If $\epsilon_{1:N}$ is Gaussian, then $Y_{1:N}$ is called a Gaussian ARCH(p). Note, however, that a Gaussian ARCH model is not a Gaussian process, just a process driven by Gaussian noise.
- If $Y_{1:N}$ is a Gaussian ARCH(p), then $Y_{1:N}^2$ is AR(p), but not Gaussian AR(p).

The GARCH model

- The **generalized ARCH** model, known as GARCH(p,q), has the form

$$Y_n = \epsilon_n \sqrt{V_n},$$

  where

$$V_n = \alpha_0 + \sum_{j=1}^{p} \alpha_j Y_{n-j}^2 + \sum_{k=1}^{q} \beta_k V_{n-k}$$

  and $\epsilon_{1:N}$ is white noise.

- The GARCH(1.1) model is a popular choice (Cowpertwait and Metcalfe, 2009) which can be fitted using `garch()` in the `tseries` R package.

Fitting a GARCH model

[handwritten:] e.g. suppose data $y^*_{1:N}$ are masses, measured in g. Now change the scale to kg. This multiplies the density by 1000 (since it contracts the scale by 1000), so adds $N \log(1000)$ to the log likelihood.

```
require(tseries)
fit.garch <- garch(sp500.ret.demeaned,grad = "numerical",
  trace = FALSE)
L.garch <- tseries:::logLik.garch(fit.garch)
```

- This 3-parameter model has a maximized log likelihood of -4019.7.
- It appears that a bug in this version of tseries means that simply logLik(fit.garch) does not work.
- From ?garch we learn this is actually a conditional log likelihood given the first $\max(p,q)$ values.

Question 16.2. It is usually inappropriate to present numerical results to five significant figures. Does that apply to the log likelihood reported here? Why?

[handwritten:] Only differences of log likelihood are relevant, and these are statistically relevant on the scale of 1 log unit. The actual value of the log likelihood is less meaningful & depends on scale.

- We are now in a position to employ the framework of likelihood-based inference for GARCH models. In particular, profile likelihood, likelihood ratio tests, and AIC are available.

- We can readily simulate from a fitted GARCH model, if we want to investigate properties of a fitted model that we don't know how to compute analytically.

- However, GARCH is a black-box model, in the sense that the parameters don't have clear interpretation. We can develop an appropriate GARCH(p,q) model, and that may be useful for forecasting, but it won't help us understand more about how financial markets work.

- We seek models that let us entertain different hypotheses about how volatility behaves.

Stochastic volatility models

- Volatility can be modeled as a latent stochastic process, partially observed via the returns.
- A Markovian assumption for volatility leads to a POMP model.
- As usual for POMP modeling, additional dependence (on previous lags or other variables) can be included.
- These are called **stochastic volatility models**.
- The basic stochastic volatility model (Kastner, 2016) is

this is similar to GARCH (similarly, basic structural POMP model is similar to ARMA).

$$Y_n = \epsilon_n \exp\{X_n/2\} \tag{1}$$

$$X_n = \mu + \phi(X_{n-1} - \mu) + \sigma\eta_n \tag{2}$$

$$X_0 = \mu + \frac{\sigma}{\sqrt{1-\phi^2}}\eta_0 \tag{3}$$

where ϵ_n and η_n are iid normal$[0,1]$. Here, X_n is the **log volatility**.

- We can use the flexibility of the POMP framework to see if we can do better.

Volatility leverage

- It is a fairly well established empirical observation that negative shocks to a stockmarket index are associated with a subsequent increase in volatility.

  this is not an observable.
  quantity since volatility is only
  indirectly observable.

- This phenomenon is called **leverage**.
- Here, we formally define leverage, R_n on day n as the correlation between index return on day $n-1$ and the increase in the log volatility from day $n-1$ to day n.
- Models have been proposed which incorporate leverage into the dynamics (Bretó, 2014).
- We present a pomp implementation of Bretó (2014), which models R_n as a random walk on a transformed scale,

$$R_n = \frac{\exp\{2G_n\} - 1}{\exp\{2G_n\} + 1},$$

where $\{G_n\}$ is the usual, Gaussian random walk.

Time-varying parameters

- A special case of this model, with the Gaussian random walk having standard deviation zero, is a fixed leverage model.

- The POMP framework provides a general approach to time-varying parameters. Considering a parameter as a latent, unobserved random process that can progressively change its value over time (following a random walk, or some other stochastic process) leads to a POMP model.

- The resulting POMP model is usually non-Gaussian, even when the original model is Gaussian and the perturbations are Gaussian, unless the time-varying parameter enters the model additively.

- Many real-world systems are non-stationary and could be investigated using models with time-varying parameters.

Modeling parameters as random may generate less understanding than finding a relationship with covariates. But, if the model fits well it should be taken seriously.

- Following the notation and model representation in equation (4) of Bretó (2014), we propose a model, *large −ve Y leads to increase in H when R_n is −ve.*

$$Y_n = \exp\{H_n/2\}\,\epsilon_n, \tag{4}$$

$$H_n = \mu_h(1 - \phi) + \phi H_{n-1} + \beta_{n-1} R_n \exp\{-H_{n-1}/2\} + \omega_n, \tag{5}$$

$$G_n = G_{n-1} + \nu_n, \quad \Leftarrow RW \ driving \ the \ leverage, \ R_n. \tag{6}$$

where $\beta_n = Y_n \sigma_\eta \sqrt{1 - \phi^2}$, $\{\epsilon_n\}$ is an iid $N(0, 1)$ sequence, $\{\nu_n\}$ is an iid $N(0, \sigma_\nu^2)$ sequence, and ω_n is $N(0, \sigma_{\omega,n}^2)$ with

$$\sigma_{\omega,n}^2 = \sigma_\eta^2 (1 - \phi^2)(1 - R_n^2).$$

- Here, H_n is the log volatility. The latent state is $X_n = (G_n, H_n)$, noting that R_n is a function of G_n.

measurement,
Y_n shows up in the process model: a violation of POMP assumptions? We could add Y_n to the latent state. This is a POMP, but has a perfectly observed latent state.

Building a POMP model

- A complication is that transition of the latent variables from (G_n, H_n) to (G_{n+1}, H_{n+1}) depends on the observable variable Y_n.

- This situation appears to be a violation of the POMP model structure.

- It is not so uncommon. For example, the same thing happens in a dynamic system subject to a control measure which is a function of the observed data.

- We can write out an extended model to fit this situation into the POMP structure, to provide access to methodology for POMP models.

- Formally, a POMP representation has state variable $X_n = (G_n, H_n, Y_n)$ and measurement variable Y_n being perfect observation of this component of X_n.

- When the latent state is continuous and there is no measurement error, the basic particle filter fails since all prediction particles are inconsistent with the data. We need a modification of sequential Monte Carlo (SMC).

- We write the filtered particle j at time $n-1$ as
$$X^F_{n-1,j} = (G^F_{n-1,j}, H^F_{n-1,j}, y^*_{n-1}).$$

- Now we can construct prediction particles at time n,
$$(G^P_{n,j}, H^P_{n,j}) \sim f_{G_n, H_n | G_{n-1}, H_{n-1}, Y_{n-1}}(g_n | G^F_{n-1,j}, H^F_{n-1,j}, y^*_{n-1})$$
with corresponding weight
$$w_{n,j} = f_{Y_n | G_n, H_n}(y^*_n | G^P_{n,j}, H^P_{n,j}).$$

- Resampling with probability proportional to these weights gives an SMC representation of the filtering distribution at time n.

- A derivation of this is given as an Appendix.

(handwritten annotation) The filtering & prediction revision calculations still hold when we condition everything on $y^*_{1:n-1}$, thus they should hold conditionally even when the model is not quite a POMP.

- We can coerce the basic sequential Monte Carlo algorithm, implemented as pfilter in **pomp**, into carrying out this calculation by building two different pomp objects, one to do filtering and another to do simulation.

- For the implementation in **pomp**, we proceed to write Csnippet code for the two versions of rprocess.

```
sp500_statenames <- c("H","G","Y_state")
sp500_rp_names <- c("sigma_nu","mu_h","phi","sigma_eta")
sp500_ivp_names <- c("G_0","H_0")
sp500_paramnames <- c(sp500_rp_names,sp500_ivp_names)
```

```
rproc1 <- "
  double beta,omega,nu;
  omega = rnorm(0,sigma_eta * sqrt( 1- phi*phi ) *
    sqrt(1-tanh(G)*tanh(G)));
  nu = rnorm(0, sigma_nu);
  G += nu;
  beta = Y_state * sigma_eta * sqrt( 1- phi*phi );
  H = mu_h*(1 - phi) + phi*H + beta * tanh( G )
    * exp(-H/2) + omega;
"
rproc2.sim <- "
  Y_state = rnorm( 0,exp(H/2) );
 "

rproc2.filt <- "
  Y_state = covaryt;
 "
sp500_rproc.sim <- paste(rproc1,rproc2.sim)
sp500_rproc.filt <- paste(rproc1,rproc2.filt)
```

```
sp500_rinit <- "
  G = G_0;
  H = H_0;
  Y_state = rnorm( 0,exp(H/2) );
"
```

```
sp500_rmeasure <- "
  y=Y_state;
"
```

```
sp500_dmeasure <- "
  lik=dnorm(y,0,exp(H/2),give_log);
"
```

Parameter transformations

- For optimization procedures such as iterated filtering, it is convenient to transform parameters to be defined on the whole real line.
- We therefore write transformation functions for σ_η, σ_ν and ϕ,

```
library(pomp)
sp500_partrans <- parameter_trans(
  log=c("sigma_eta","sigma_nu"),
  logit="phi"
)
```

- We can now build a pomp object suitable for filtering, and parameter estimation by iterated filtering or particle MCMC.
- Note that the data are also placed in a covariate slot.
- This is a device to allow the state process evolution to depend on the data. In a POMP model, the latent process evolution depends only on the current latent state. In **pomp**, the consequence of this structure is that rprocess doesn't have access to the observation process.
- However, a POMP model does allow for the possibility for the basic elements to depend on arbitrary covariates. In **pomp**, this means rprocess has access to a covariate slot.
- The code below gives an example of how to fill the covariate slot and how to use it in rprocess.

```
sp500.filt <- pomp(data=data.frame(
    y=sp500.ret.demeaned,time=1:length(sp500.ret.demeaned)),
  statenames=sp500_statenames,
  paramnames=sp500_paramnames,
  times="time",
  t0=0,
  covar=covariate_table(
    time=0:length(sp500.ret.demeaned),
    covaryt=c(0,sp500.ret.demeaned),
    times="time"),
  rmeasure=Csnippet(sp500_rmeasure),
  dmeasure=Csnippet(sp500_dmeasure),
  rprocess=discrete_time(step.fun=Csnippet(sp500_rproc.filt),
    delta.t=1),
  rinit=Csnippet(sp500_rinit),
  partrans=sp500_partrans
)
```

- Simulating from the model is convenient for developing and testing the code, as well as to investigate a fitted model:

```
params_test <- c(
  sigma_nu = exp(-4.5),
  mu_h = -0.25,
  phi = expit(4),
  sigma_eta = exp(-0.07),
  G_0 = 0,
  H_0=0
)

sim1.sim <- pomp(sp500.filt,
  statenames=sp500_statenames,
  paramnames=sp500_paramnames,
  rprocess=discrete_time(
    step.fun=Csnippet(sp500_rproc.sim),delta.t=1)
)

sim1.sim <- simulate(sim1.sim,seed=1,params=params_test)
```

- Now. to build the filtering object from `sim1.sim`, we need to copy the new simulated data into the covariate slot, and put back the appropriate version of `rprocess`.

```
sim1.filt <- pomp(sim1.sim,
  covar=covariate_table(
    time=c(timezero(sim1.sim),time(sim1.sim)),
    covaryt=c(obs(sim1.sim),NA),
    times="time"),
  statenames=sp500_statenames,
  paramnames=sp500_paramnames,
  rprocess=discrete_time(
    step.fun=Csnippet(sp500_rproc.filt),delta.t=1)
)
```

Filtering on simulated data

- We check that we can indeed filter and re-estimate parameters successfully for this simulated data.
- As previously discussed, we set up code to switch between different levels of computational intensity:

```
run_level <- 3
sp500_Np <-            switch(run_level, 100, 1e3, 2e3)
sp500_Nmif <-          switch(run_level,  10, 100, 200)
sp500_Nreps_eval <-    switch(run_level,   4,  10,  20)
sp500_Nreps_local <-   switch(run_level,  10,  20,  20)
sp500_Nreps_global <-  switch(run_level,  10,  20, 100)
```

- We carry out replications in parallel, using all available cores on either a laptop or a single node of a SLURM cluster.

```
library(doParallel)
cores <-  as.numeric(Sys.getenv('SLURM_NTASKS_PER_NODE', unset=NA))
if(is.na(cores)) cores <- detectCores()
registerDoParallel(cores)
library(doRNG)
registerDoRNG(34118892)
```

```
stew(file=sprintf("pf1-%d.rda",run_level),{
  t.pf1 <- system.time(
    pf1 <- foreach(i=1:sp500_Nreps_eval,
      .packages='pomp') %dopar% pfilter(sim1.filt,Np=sp500_Np))
})
(L.pf1 <- logmeanexp(sapply(pf1,logLik),se=TRUE))

                     se
-3658.7879118      0.1459427
```

- In 4.3 seconds, we obtain a log likelihood estimate of -3658.79 with a Monte Carlo standard error of 0.15.

- Notice that the replications are averaged using the `logmeanexp` function, since the likelihood estimate is unbiased on the natural scale but not the log scale.

- We could test the numerical performance of an iterated filtering likelihood maximization algorithm on simulated data.

- We could also study the statistical performance of maximum likelihood estimators and profile likelihood confidence intervals on simulated data.

- However, here we are going to cut to the chase and start fitting models to data.

Fitting the stochastic leverage model to S&P500 data

- We are now ready to try out iterated filtering on the S&P500 data. We will use the IF2 algorithm of Ionides *et al.* (2015), implemented by `mif2`.

```
sp500_rw.sd_rp <- 0.02
sp500_rw.sd_ivp <- 0.1
sp500_cooling.fraction.50 <- 0.5
sp500_rw.sd <- rw.sd(
  sigma_nu  = sp500_rw.sd_rp,
  mu_h      = sp500_rw.sd_rp,
  phi       = sp500_rw.sd_rp,
  sigma_eta = sp500_rw.sd_rp,
  G_0       = ivp(sp500_rw.sd_ivp),
  H_0       = ivp(sp500_rw.sd_ivp)
)
```

```
stew(file=sprintf("mif1-%d.rda",run_level),{
  t.if1 <- system.time({
  if1 <- foreach(i=1:sp500_Nreps_local,
    .packages='pomp', .combine=c) %dopar% mif2(sp500.filt,
      params=params_test,
      Np=sp500_Np,
      Nmif=sp500_Nmif,
      cooling.fraction.50=sp500_cooling.fraction.50,
      rw.sd = sp500_rw.sd)
  L.if1 <- foreach(i=1:sp500_Nreps_local,
    .packages='pomp', .combine=rbind) %dopar% logmeanexp(
      replicate(sp500_Nreps_eval, logLik(pfilter(sp500.filt,
        params=coef(if1[[i]]),Np=sp500_Np))), se=TRUE)
  })
})
r.if1 <- data.frame(logLik=L.if1[,1],logLik_se=L.if1[,2],
  t(sapply(if1,coef)))
if (run_level>1) write.table(r.if1,file="sp500_params.csv",
  append=TRUE,col.names=FALSE,row.names=FALSE)
```

- This investigation took 20 minutes.
- The repeated stochastic maximizations can also show us the geometry of the likelihood surface in a neighborhood of this point estimate:

```
pairs(~logLik+sigma_nu+mu_h+phi+sigma_eta,
  data=subset(r.if1,logLik>max(logLik)-20))
```
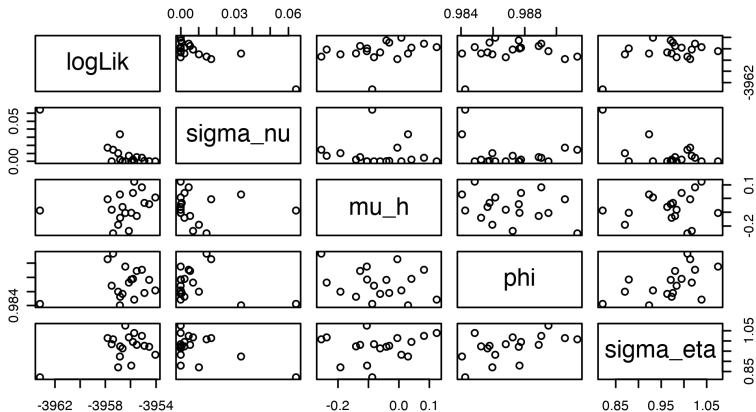
Likelihood maximization using randomized starting values

- As for our other case studies, carrying out searches starting randomly throughout a large box can lead to reasonble evidence for successful global maximization.

- For our volatility model, a box containing plausible parameter values might be

```
sp500_box <- rbind(
  sigma_nu=c(0.005,0.05),
  mu_h    =c(-1,0),
  phi = c(0.95,0.99),
  sigma_eta = c(0.5,1),
  G_0 = c(-2,2),
  H_0 = c(-1,1)
)
```

```
stew(file=sprintf("box_eval-%d.rda",run_level),{
  t.box <- system.time({
    if.box <- foreach(i=1:sp500_Nreps_global,
      .packages='pomp',.combine=c) %dopar% mif2(if1[[1]],
        params=apply(sp500_box,1,function(x)runif(1,x)))
    L.box <- foreach(i=1:sp500_Nreps_global,
      .packages='pomp',.combine=rbind) %dopar% {
        logmeanexp(replicate(sp500_Nreps_eval, logLik(pfilter(
          sp500.filt,params=coef(if.box[[i]]),Np=sp500_Np))),
          se=TRUE)}
  })
})
r.box <- data.frame(logLik=L.box[,1],logLik_se=L.box[,2],
  t(sapply(if.box,coef)))
if(run_level>1) write.table(r.box,file="sp500_params.csv",
  append=TRUE,col.names=FALSE,row.names=FALSE)
summary(r.box$logLik,digits=5)

  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 -3996   -3986   -3957   -3966   -3956   -3954
```

(handwritten annotations:) should bear in mind that errors may occur on a scale of 1 unit of log lik.

here, we are getting within 2 log units of our maximum value on 25% of searches. This is a reasonable level of success.

- This search took 64.1 minutes.
- The best likelihood found was -3953.8 with a standard error of 0.2.
- We see that optimization attempts from diverse remote starting points can approach our MLE, but do not exceed it. This gives us some reasonable confidence in our MLE.
- Plotting these diverse parameter estimates can help to give a feel for the global geometry of the likelihood surface

```
pairs(~logLik+log(sigma_nu)+mu_h+phi+sigma_eta+H_0,
    data=subset(r.box,logLik>max(logLik)-10))
```
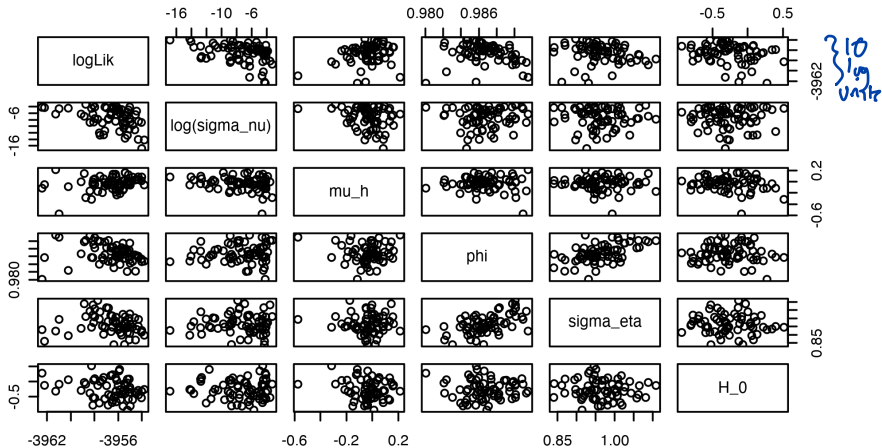
- This preliminary analysis does not show clear evidence for the hypothesis that $\sigma_\nu > 0$, also μ_h looks like it could plausibly be 0.

- That is likely because we are studying only a subset of the 1988 to 2012 dataset analyzed by Bretó (2014).

- Also, it might help to refine our inference be computing a likelihood profile over σ_ν, and μ_h.

Benchmark likelihoods for alternative models

- To assess the overall success of the model, it is helpful to put the log likelihoods in the context of simpler models, called **benchmarks**.
- Benchmarks provide a complementary approach to residual analysis and the investigation of simulations from the fitted model.
- The GARCH(1,1) model for this dataset has a maximized log likelihood of -4019.7 with 3 fitted parameters.
- Our stochastic volatility model, with time-varying leverage, model has a maximized log likelihood of -3953.8 with 6 fitted parameters. AIC favors the stochastic volatility model.
- A model which both fits better and has meaningful interpretation has clear advantages over a simple statistical model.
- The disadvantage of the sophisticated modeling and inference is the extra effort required.

Can a mechanistic model be helpful if it loses to a non-mechanistic alternative?

- Sometimes, the mechanistic model does not beat simple benchmark models. That does not necessarily mean the mechanistic model is entirely useless.

- We may be able to learn about the system under investigation from what a scientifically interpretable model fails to explain.

- We may be able to use preliminary results to improve the model, and subsequently beat the benchmarks.

- If the mechanistic model fits disastrously compared to the benchmark, our model is probably missing something important. We must reconsider the model, based on clues we might obtain by carrying out residual analysis and looking at simulations from the fitted model.

Appendix: Proper weighting for a partially plug-and-play algorithm with a perfectly observed state space component

- Suppose a POMP model with $X_n = (U_n, V_n)$ and measurement model $f_{Y_n|X_n}(y_n \mid u_n, v_n) = f_{Y_n|V_n}(y_n|v_n)$, depending only on v_n.
- The proper weight for an SMC proposal density $q_n(x_n|x_{n-1})$ is

$$w_n(x_n|x_{n-1}) = \frac{f_{Y_n|X_n}(y_n^*|x_n)f_{X_n|X_{n-1}}(x_n|x_{n-1})}{q_n(x_n|x_{n-1})}.$$

- Consider the proposal $q_n(u_n, v_n|x_{n-1}) = f_{U_n|X_{n-1}}(u_n|x_{n-1})g_n(v_n)$. This is partially plug-and-play, in the sense that the U_n part of the proposal is drawn from a simulator of the dynamic system.
- Computing the weights, we see that the transition density for the U_n component cancels out and does not have to be computed, i.e.,

$$w_n(x_n|x_{n-1}) = \frac{f_{Y_n|V_n}(y_n^*|v_n)f_{U_n|X_{n-1}}(u_n|x_{n-1})f_{V_n|U_n,X_{n-1}}(v_n|u_n, x_{n-1})}{f_{U_n|X_{n-1}}(u_n|x_{n-1})g_n(v_n)}$$

$$= \frac{f_{Y_n|V_n}(y_n^*|v_n)f_{V_n|U_n,X_{n-1}}(v_n|u_n, x_{n-1})}{g_n(v_n)}.$$

- Now consider the case where the V_n component of the state space is perfectly observed, i.e., $Y_n = V_n$. In this case,

$$f_{Y_n|V_n}(y_n|v_n) = \delta(y_n - v_n),$$

interpreted as a point mass at v_n in the discrete case and a singular density at v_n in the continuous case.

- We can choose $g_n(v_n)$ to depend on the data, and a natural choice is

$$g_n(v_n) = \delta(y_n^* - v_n),$$

for which the proper weight is

$$w_n(x_n|x_{n-1}) = f_{Y_n|U_n,X_{n-1}}(y_n^*|u_n, x_{n-1}).$$

- This is the situation in the context of our case study, with $U_n = (G_n, H_n)$ and $V_n = Y_n$.

References and Acknowledgements

Bretó C (2014). "On idiosyncratic stochasticity of financial leverage effects." *Statistics & Probability Letters*, **91**, 20–26. doi: 10.1016/j.spl.2014.04.003.

Cowpertwait PS, Metcalfe AV (2009). *Introductory time series with R*. Springer Science & Business Media.

Ionides EL, Nguyen D, Atchadé Y, Stoev S, King AA (2015). "Inference for dynamic and latent variable models via iterated, perturbed Bayes maps." *Proceedings of the National Academy of Sciences of the U.S.A.*, **112**(3), 719–724. doi: 10.1073/pnas.1410597112.

References and Acknowledgements II

Kastner G (2016). "Dealing with stochastic volatility in time series using the R package stochvol." *Journal of Statistical Software*, **69**. doi: 10.18637/jss.v069.i05.

Shumway RH, Stoffer DS (2017). *Time Series Analysis and its Applications: With R Examples*. Springer.

- Compiled on March 23, 2022 using R version 4.1.2.
-
- We acknowledge previous versions of this course.