Elements of Sample Projects

VERSION: 2018-03-11 · 23:25:13

Summary. The following are elements which you may combine and modify to create a project. **Note:** The calculations in this document and accompanying **R** files may contain typos – some perhaps intentional. Therefore, test even the most basic likelihood simplifications with built-in **R** functions (e.g., `dnorm`, `?Distributions`).

1. High-Dimensional Analysis of Financial Data

The file **`snp500-adj_close_2004-2018.csv`** contains daily asset values of 45 stocks from the S&P500 between August 19, 2004 and March 2, 2018. It is a data frame of 3408 rows and 48 columns consisting of the following elements:

- **Column 1:** The date.
- **Columns 2-8:** Various assets in the `Financial` sector.
- **Columns 9-13:** Various assets in the `Basic_Minerals` sector.
- **Columns 14-22:** Various assets in the `Tech` sector.
- **Columns 23-27:** Various assets in the `Industrial` sector.
- **Columns 28-34:** Various assets in the `Services` sector.
- **Column 35-39:** Various assets in the `Consumer_Goods` sector.
- **Column 40-46:** Various assets in the `Healthcare` sector.
- **Column 47:** The `GSPC` composite index of all stocks in the S&P500.
- **Column 48:** The `VIX` volatility index for the S&P500.

A simple but extremely powerful and effective framework for jointly analyzing and forecasting these assets is the so-called GARCH-Gaussian-Copula (GARCHGC) model. In a nutshell, the GARCHGC model consists of two parts: (i) a marginal GARCH model for each asset and (ii) a multivariate normal on the between-series residuals. The exact construction is detailed below.

1.1. GARCH(1, 1) Marginal Model

Let $s_{ti} > 0$ denote the value of asset i on day t, and $y_{ti} \in \mathbb{R}$ denote its return. The three most common return metrics are:

- *Absolute (i.e., regular) returns:* $y_{ti} = s_{ti} - s_{t-1,i}$.

- *Relative returns:* $y_{ti} = (s_{ti} - s_{t-1,i})/s_{t-1,i}$.
- *Log returns:* $y_{ti} = \log(s_{ti}) - \log(s_{t-1,i})$.

Then a marginal GARCH(1, 1) model for the return series of asset i is

$$\begin{aligned} y_{ti} &= \mu_i + \varepsilon_{ti} \\ \varepsilon_{ti} &= \sigma_{ti}\, x_{ti} \\ \sigma_{ti}^2 &= \omega_i + \alpha_i\, \varepsilon_{t-1,i}^2 + \beta_i\, \sigma_{t-1,i}^2 \\ x_{ti} &\overset{\text{iid}}{\sim} f(x \,|\, \boldsymbol{\eta}_i). \end{aligned} \tag{1.1}$$

In the simplest GARCH(1, 1) model we saw in class, the residual distribution $f(x \,|\, \boldsymbol{\eta})$ is just $\mathcal{N}(0, 1)$. However, considering e.g., a noncentral-t distribution allows for asymmetry and heavy-tailed returns as is often the case in financial data.

1.2. Gaussian Residual Dependence Model

The GARCH model (1.1) fully specifies the marginal time series of each asset i. However, it does not specify how the assets depend on each other. A very simple way of modeling dependence is through the model residuals alone. That is, let $\boldsymbol{x}_t = (x_{t1}, \ldots, x_{tD})$ denote the GARCH residuals for all D assets on day t. To be consistent with model (1.1), we wish to model $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots$ as iid random vectors, for which marginally $x_{ti} \overset{\text{iid}}{\sim} f(x \,|\, \boldsymbol{\eta}_i)$ for fixed i and different t.

In order to achieve this, recall the *Probability Integral Transform*. That is, if $F(x \,|\, \boldsymbol{\eta}_i)$ is the CDF of x_{ti}, then $u_{ti} = F(x_{ti} \,|\, \boldsymbol{\eta}) \overset{\text{iid}}{\sim} \text{Unif}(0, 1)$ for each asset i. Similarly, if $\Phi(z)$ is the CDF of $\mathcal{N}(0, 1)$, then

$$z_{ti} = \Phi^{-1}\Big(F(x_{ti} \,|\, \boldsymbol{\eta}_i)\Big) \overset{\text{iid}}{\sim} \mathcal{N}(0, 1).$$

So let $\boldsymbol{z}_t = (z_{t1}, \ldots, z_{tD})$ denote the transformed vector of residuals, which are now marginally standard normal. We thus complete the specification of the GarchGC model by letting the transformed residuals be multivariate normal:

$$\boldsymbol{z}_t \overset{\text{iid}}{\sim} \mathcal{N}(\mathbf{0}, \boldsymbol{\Omega}), \tag{1.2}$$

where $\boldsymbol{\Omega}_{D\times D}$ is a (possibly structured) correlation matrix (i.e., a variance matrix with diagonal elements $\Omega_{ii} \equiv 1$). The marginal GARCH parameters of asset i are $\boldsymbol{\theta}_i = (\alpha_i, \beta_i, \omega_i, \mu_i, \boldsymbol{\eta}_i)$, such that the full set of parameters of the GarchGC model defined by (1.1) and (1.2) are $\boldsymbol{\Theta} = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_D, \boldsymbol{\Omega})$.

1.3. GarchGC Model Calibration

Let $\boldsymbol{Y}_i = (y_{1i}, \ldots, y_{Ti})$ denote the returns for asset i on days $t = 1, \ldots, T$, and $\boldsymbol{Y} = (\boldsymbol{Y}_1, \ldots, \boldsymbol{Y}_D)$ denote the data for all D assets. A simple multi-stage estimator for the GarchGC parameters $\boldsymbol{\Theta}$ is as follows:

1. For each asset i, fit the GARCH(1, 1) model (1.1) to $\boldsymbol{Y}_i$ to get the MLE $\hat{\boldsymbol{\theta}}_i$. One way to do this is to use the **R** package `rugarch`, which provides several options for the residual distribution $f(x \mid \boldsymbol{\eta})$. For more information see `?rgarchdist`.
2. For each asset i, calculate the model residuals $\hat{x}_{ti} = (y_{ti} - \hat{\mu}_i)/\hat{\sigma}_{ti}$ as we saw in class[1].
3. Transform the residuals to the normal scale by setting $\hat{z}_{ti} = \Phi^{-1}\Big(F(\hat{x}_{ti} \mid \hat{\boldsymbol{\eta}}_i)\Big)$.
4. Fit a correlation matrix $\hat{\boldsymbol{\Omega}}$ by pretending that $\hat{\boldsymbol{z}}_t \overset{\text{iid}}{\sim} \mathcal{N}(\mathbf{0}, \boldsymbol{\Omega})$. The two simplest estimators are $\hat{\boldsymbol{\Omega}} = \mathbf{I}$ and $\hat{\boldsymbol{\Omega}} = \widehat{\text{cor}}(\hat{\boldsymbol{Z}})$ (i.e., the sample correlation), respectively corresponding to independent-GARCH and unstructured variance assumptions.

The resulting estimator $\hat{\boldsymbol{\Theta}}$ is not the MLE. However, it is likely to be quite close, and thus is a good starting value for using the **R** function `optim` to maximize the loglikelihood of the GarchGC model, which is

$$\ell(\boldsymbol{\Theta} \mid \boldsymbol{Y}) = \sum_{t=1}^{T} \left(\log \varphi_D(\boldsymbol{z}_t^{(\boldsymbol{\Theta})} \mid \boldsymbol{\Omega}) + \sum_{i=1}^{D} \Big[\log f(x_{ti}^{(\boldsymbol{\theta}_i)} \mid \boldsymbol{\eta}_i) - \log \sigma_{ti}^{(\boldsymbol{\theta}_i)} - \log \varphi(z_{ti}^{(\boldsymbol{\theta}_i)}) \Big] \right),$$

where the superscripts emphasize the fact that these quantities depend on both $\boldsymbol{Y}$ and model parameters. However, with $D \approx 50$ assets this is an enormous optimization problem which may easily overwhelm `optim` even with good starting values[2]. Most of the time people don't bother with this and stick with the multi-stage estimator $\hat{\boldsymbol{\Theta}} \neq \hat{\boldsymbol{\Theta}}_{\text{ML}}$.

1.4. Sector-Level Hierarchical Modeling

Consider the following model for grouping the correlations between the GarchGC models by the market sector to which they belong. Suppose that each of the D assets fall into one of K sectors (here we have $D = 45$ and $K = 7$, excluding the two indices). Let S_k denote the mean of sector k, and $\boldsymbol{S} = (S_1, \ldots, S_K)$. Let $\boldsymbol{z}_t^{(k)} = (z_{t1}^{(k)}, \ldots, z_{tN_K}^{(k)})$ denote the transformed residuals on day t of the N_k assets in sector k, such that $D = \sum_{k=1}^{K} N_k$. Then a hierarchical

[1] You can do this directly with `rugarch`, though it may have a different way of setting the initial volatility $\hat{\sigma}_1$ than what we saw in class.

[2] Though perhaps not? Or perhaps one could cycle through parameter blocks a few times, i.e., blockwise coordinate ascent? Would be nice if this worked!

model for the transformed residuals is

$$\begin{aligned} \boldsymbol{S} &\sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}) \\ \boldsymbol{z}_t^{(k)} \,|\, \boldsymbol{S} &\overset{\text{ind}}{\sim} \mathcal{N}(S_k, \boldsymbol{\Psi}_k). \end{aligned}$$

Write an EM algorithm to find the MLE of $\boldsymbol{\Omega} = (\boldsymbol{\Sigma}, \boldsymbol{\Psi}_1, \ldots, \boldsymbol{\Psi}_K)$ using the transformed residuals $\boldsymbol{Z} = (z_1, \ldots, z_T)$[3]. The various steps are outlined below.

- In the EM setup, the observed data is $\boldsymbol{y}_{\text{obs}} = \boldsymbol{Z}$ and the missing data is $\boldsymbol{y}_{\text{miss}} = \boldsymbol{S}$. Thus, the complete data likelihood is

$$\begin{aligned} \ell(\boldsymbol{\Omega} \,|\, \boldsymbol{S}, \boldsymbol{Z}) &= -\frac{1}{2}\Big[\boldsymbol{S}'\boldsymbol{\Sigma}^{-1}\boldsymbol{S} + \log|\boldsymbol{\Sigma}|\Big] - \frac{1}{2}\sum_{k=1}^{K}\sum_{t=1}^{T}\Big[(\boldsymbol{z}_t^{(k)} - S_k)'\boldsymbol{\Psi}_k^{-1}(\boldsymbol{z}_t^{(k)} - S_k) + \log|\boldsymbol{\Psi}_k|\Big] \\ &= -\frac{1}{2}\Big[\operatorname{tr}(\boldsymbol{\Sigma}^{-1}\boldsymbol{S}\boldsymbol{S}') + \log|\boldsymbol{\Sigma}|\Big] \\ &\quad - \frac{T}{2}\sum_{k=1}^{K}\Big[\operatorname{tr}(\boldsymbol{\Psi}_k^{-1}\boldsymbol{T}_k) - 2S_k \operatorname{tr}(\boldsymbol{\Psi}_k^{-1}\bar{\boldsymbol{z}}_k\mathbf{J}_{N_k}') + S_k^2 \operatorname{tr}(\boldsymbol{\Psi}_k^{-1}\mathbf{J}_{N_k}\mathbf{J}_{N_k}') + \log|\boldsymbol{\Psi}_k|\Big], \end{aligned}$$

  where $\bar{\boldsymbol{z}}_k = \frac{1}{T}\sum_{t=1}^{T}\boldsymbol{z}_t^{(k)}$, $\boldsymbol{T}_k = \frac{1}{T}\sum_{t=1}^{T}\boldsymbol{z}_t^{(k)}\boldsymbol{z}_t^{(k)\prime}$, and $\mathbf{J}_n$ is a column vector of n ones.

- Suppose that at step m of the algorithm we have the parameter estimate $\hat{\boldsymbol{\Omega}}^{(m)}$. In order to calculate $Q_m(\boldsymbol{\Omega}) = E[\ell(\boldsymbol{\Omega} \,|\, \boldsymbol{S}, \boldsymbol{Z}) \,|\, \boldsymbol{Z}, \hat{\boldsymbol{\Omega}}^{(m)}]$, we need $E[\boldsymbol{S} \,|\, \boldsymbol{Z}]$ and $E[\boldsymbol{S}\boldsymbol{S}' \,|\, \boldsymbol{Z}] = \operatorname{var}(\boldsymbol{S} \,|\, \boldsymbol{Z}) + E[\boldsymbol{S} \,|\, \boldsymbol{Z}]\, E[\boldsymbol{S} \,|\, \boldsymbol{Z}]'$ (where, to simplify notation, the dependence on the parameter value $\boldsymbol{\Omega} = \hat{\boldsymbol{\Omega}}^{(m)}$ has been suppressed). To obtain these, note that $p(\boldsymbol{S} \,|\, \boldsymbol{Z}) = p(\boldsymbol{S} \,|\, \bar{\boldsymbol{Z}})$, where $\bar{\boldsymbol{Z}} = (\bar{\boldsymbol{z}}_1, \ldots, \bar{\boldsymbol{z}}_K)$, and that since

$$\begin{aligned} \boldsymbol{S} &\sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}) \\ \bar{\boldsymbol{z}}_k \,|\, \boldsymbol{S} &\overset{\text{ind}}{\sim} \mathcal{N}(S_k, \tfrac{1}{T}\boldsymbol{\Psi}_k), \end{aligned}$$

  the joint distribution of $(\boldsymbol{S}, \bar{\boldsymbol{Z}})$ is multivariate normal with $E[(\boldsymbol{S}, \bar{\boldsymbol{Z}})] = \mathbf{0}$ and

$$\operatorname{var}\begin{pmatrix} \boldsymbol{S} \\ \bar{\boldsymbol{z}}_1 \\ \bar{\boldsymbol{z}}_2 \\ \vdots \\ \bar{\boldsymbol{z}}_K \end{pmatrix} = \begin{pmatrix} \boldsymbol{\Sigma} & \Sigma_{11}\,\mathbf{J}_{K\times N_1} & \Sigma_{12}\,\mathbf{J}_{K\times N_2} & \cdots & \Sigma_{1K}\,\mathbf{J}_{K\times N_K} \\ \Sigma_{11}\,\mathbf{J}_{N_1\times K} & \frac{1}{T}\boldsymbol{\Psi}_1 + \Sigma_{11}\,\mathbf{J}_{N_1\times N_1} & \Sigma_{12}\,\mathbf{J}_{N_1\times N_2} & \cdots & \Sigma_{1K}\,\mathbf{J}_{N_1\times N_K} \\ \Sigma_{21}\,\mathbf{J}_{N_2\times K} & \Sigma_{21}\,\mathbf{J}_{N_1\times N_1} & \frac{1}{T}\boldsymbol{\Psi}_2 + \Sigma_{22}\,\mathbf{J}_{N_2\times N_2} & \cdots & \Sigma_{2K}\,\mathbf{J}_{N_2\times N_K} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \Sigma_{K1}\,\mathbf{J}_{N_K\times K} & \Sigma_{K1}\,\mathbf{J}_{N_K\times N_1} & \Sigma_{K2}\,\mathbf{J}_{N_K\times N_2} & \cdots & \frac{1}{T}\boldsymbol{\Psi}_K + \Sigma_{KK}\,\mathbf{J}_{N_K\times N_K} \end{pmatrix},$$

[3] In fact you will be using the estimated residuals $\hat{\boldsymbol{Z}}$ from the multi-stage estimator, but let's drop the "hat" to simplify notation.

where $\mathbf{J}_{n\times m}$ is an $n \times m$ matrix consisting of all ones. Thus we have $\boldsymbol{S} \,|\, \bar{\boldsymbol{Z}}, \hat{\boldsymbol{\Omega}}^{(m)} \sim \mathcal{N}(\hat{\boldsymbol{\mu}}^{(m)}, \hat{\boldsymbol{\Sigma}}^{(m)})$, with formulas for $\hat{\boldsymbol{\mu}}^{(m)}$ and $\hat{\boldsymbol{\Sigma}}^{(m)}$ given in terms of $\text{var}(\boldsymbol{S}, \bar{\boldsymbol{Z}} \,|\, \boldsymbol{\Omega} = \hat{\boldsymbol{\Omega}}^{(m)})$.

- After completing the E-step, we obtain

$$Q_m(\boldsymbol{\Omega}) = -\frac{1}{2}\Big[\,\text{tr}(\boldsymbol{\Sigma}^{-1}\hat{\boldsymbol{A}}^{(m)}) + \log|\boldsymbol{\Sigma}|\Big] - \frac{T}{2}\sum_{k=1}^{K}\Big[\,\text{tr}(\boldsymbol{\Psi}_k^{-1}\hat{\boldsymbol{B}}_k^{(m)}) + \log|\boldsymbol{\Psi}_k|\Big],$$

for which the M-step updates are $\hat{\boldsymbol{\Sigma}}^{(m+1)} = \hat{\boldsymbol{A}}^{(m)}$ and $\hat{\boldsymbol{\Psi}}_k^{(m+1)} = T^{-1}\hat{\boldsymbol{B}}_k^{(m)}$.

1.5. Evaluation of Financial Forecasting Models

A fundamental purpose of a GarchGC model is to make out-of-sample forecasts of a given investment portfolio. That is:

- If today is day t, let $\boldsymbol{S}_t^{(W)} = (\boldsymbol{s}_{t-W+1}, \ldots, \boldsymbol{s}_t)$ denote the last W days of asset values up to and including today.

- If today is day t, let $\mathcal{P}_{t+k}$ denote the value of a given portfolio k days in the future.

Generally speaking, the future value of the portfolio is given by $\mathcal{P}_{t+k} = \mathcal{P}(\boldsymbol{S}_t^{(W)}, \boldsymbol{s}_{t+k})$, where $\mathcal{P} : \mathbb{R}^d \times \mathbb{R}^{Wd} \to \mathbb{R}$ is a known function. That is, the value of the portfolio is determined by (i) an investment strategy based on $\boldsymbol{S}_t^{(W)}$, the last W days of information and (ii) the values $\boldsymbol{s}_{t+k}$ of all assets k days in the future. Specific examples of portfolios will be given momentarily.

- Given $\boldsymbol{S}_t^{(W)} = (\boldsymbol{s}_{t-W+1}, \ldots, \boldsymbol{s}_t)$, the forecast distribution for the portfolio value $\mathcal{P}_{t+k}$ is defined as follows:

  1. Use $\boldsymbol{S}_t^{(W)}$ to compute $\hat{\boldsymbol{\Theta}}_t^{(W)} = \hat{\boldsymbol{\Theta}}(\boldsymbol{S}_t^{(W)})$, an estimate of the GarchGC parameters $\boldsymbol{\Theta} = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_D, \boldsymbol{\Omega})$. Now use $\boldsymbol{S}_t^{(W)}$ and $\hat{\boldsymbol{\Theta}}_t^{(W)}$ to compute $\hat{\boldsymbol{\sigma}}_t^{(W)}$, an estimate of the volatility vector $\boldsymbol{\sigma}_t = (\sigma_{t1}, \ldots, \sigma_{tD})$ on day t.

  2. Use $\hat{\boldsymbol{\sigma}}_t^{(W)}$ to calculate $\hat{\boldsymbol{\varepsilon}}_t^{(W)} = (\boldsymbol{y}_t - \hat{\boldsymbol{\mu}}_t^{(W)})/\hat{\boldsymbol{\sigma}}_t^{(W)}$, and use the GarchGC model defined by (1.1) and (1.2) to generate M iid k-length trajectories from

$$p(\boldsymbol{y}_{t+1}, \ldots, \boldsymbol{y}_{t+k} \,|\, \boldsymbol{\eta}_t = \hat{\boldsymbol{\eta}}_t^{(W)}, \boldsymbol{\sigma}_t = \hat{\boldsymbol{\sigma}}_t^{(W)}, \boldsymbol{\Theta} = \hat{\boldsymbol{\Theta}}_t^{(W)}).$$

  3. Convert each trajectory $\boldsymbol{y}_{t+1}, \ldots, \boldsymbol{y}_{t+k}$ into the corresponding asset value $\boldsymbol{s}_{t+k}$, and then into portfolio value $\mathcal{P}_{t+k} = \mathcal{P}(\boldsymbol{S}_t^{(W)}, \boldsymbol{s}_{t+K})$. The resulting histogram of $\mathcal{P}_{t+k}$ con-

verges to the k-step forecast distribution of $\mathcal{P}_{t+k}$:

$$p_{\text{FC}}(\mathcal{P}_{t+k} \mid S_t^{(W)}) := p(\mathcal{P}(\boldsymbol{s}_{t+k}) \mid S_t^{(W)}, \boldsymbol{\eta}_t = \hat{\boldsymbol{\eta}}_t^{(W)}, \boldsymbol{\sigma}_t = \hat{\boldsymbol{\sigma}}_t^{(W)}, \boldsymbol{\Theta} = \hat{\boldsymbol{\Theta}}_t^{(W)}).$$

From this forecast distribution we can obtain a point estimate $\hat{\mathcal{P}}_{t+k} = E_{\text{FC}}[\mathcal{P}_{t+k} \mid S_t^{(W)}]$, a prediction interval (consisting of the 2.5% and 97.5% quantiles of the forecast distribution), the probability that the value of the portfolio drops below zero, etc.

Given this forecasting framework, we can evaluate the performance of a GarchGC model as follows:

1. Pick an initial calibration period consisting of W days, i.e., days $t = 1, 2, \ldots, W$.
2. Pick a reasonable number of testing days N.
3. For each day $t = W, W+1, \ldots, W+N$, generate M iid draws from the porfolio forecast distribution $p_{\text{FC}}(\mathcal{P}_{t+k} \mid S_t^{(W)})$.
4. Let $\mathcal{P}_{t+k}^{(\text{obs})} = \mathcal{P}(S_t^{(W)}, s_{t+k}^{(\text{obs})})$ denote the actual value of the porfolio on day $t+k$ (i.e., using the true value of the assets on day $t+k$). Use the M iid draws from the forecast distribution to estimate $\mathrm{p}_t = \Pr_{\text{FC}}(\mathcal{P}_{t+k} \le \mathcal{P}_{t+k}^{(\text{obs})} \mid S_t^{(W)})$.

At the end of this process, you have N "rolling window p-values" $\mathrm{p}_W, \ldots, \mathrm{p}_{W+N}$.

Now consider the null hypothesis H_0 that the GarchGC model is correct. Then under H_0, $\mathrm{p}_t \sim \text{Unif}(0,1)$ for $t = W, \ldots, W+N$. Similarly, using the Probability Integral Transform we have $\mathrm{z}_t = \Phi^{-1}(\mathrm{p}_t) \sim \mathcal{N}(0,1)$, where $\Phi(z)$ is the CDF of the standard normal. Thus we can graphically evaluate the GarchGC model using QQ-plots. Moreover, if $k = 1$, then in fact $\mathrm{z}_t \overset{\text{iid}}{\sim} \mathcal{N}(0,1)$, which allows for the construction of various quantitative "goodness-of-fit" normality tests giving p-values against H_0. The most common ones are the Kolmogorov-Smirnov, Anderson-Darling, and Shapiro-Wilk tests, listed here in increasing order of statistical power.

1.6. Modern Portfolio Theory

The simplest portfolio consists of the value of a single asset, $\mathcal{P}_{t+k} = s_{t+k,i}$. However, GarchGC model forecasts for single assets are *completely independent* of the residual correlation matrix $\boldsymbol{\Omega} = \text{cor}(\boldsymbol{z}_t)$. In other words, fitting a univariate GARCH model (1.1) to asset i gives exactly the same forecasts as the multivariate GarchGC model with any correlation matrix $\boldsymbol{\Omega}$. In order to evaluate and/or compare various GarchGC models, consider the following investment strategy:

- Suppose that today is day t and let $\Delta^{(k)}\boldsymbol{s}_t = \boldsymbol{s}_{t+k} - \boldsymbol{s}_t$ denote the vector of asset returns

for k days into the future.

- Suppose that on day t, for any asset i you are allowed to do one of two things:

  1. Buy $q_i > 0$ shares of asset i and wait for k days. On day $t+k$ this investment in asset i will have generated $q_i \cdot \Delta^{(k)} s_{ti}$ dollars (which could be positive or negative, depending on the sign of $\Delta^{(k)} s_{ti}$).

  2. Short-sell $-q_i > 0$ shares of asset i. That is, assume that you can find someone in the market who's willing to buy $-q_i$ shares of asset i at today's price of s_{ti}, but only receive the shares in k days. For simplicity, assume that you will only purchase those shares on day $t+k$ and then immediately deliver them to your buyer. Thus, this short-sell of asset i will also generate $q_i \cdot \Delta^{(k)} s_{ti}$ dollars.

- Let

$$\boldsymbol{\mu} = E_{\mathrm{FC}}[\Delta^{(k)} \boldsymbol{s}_t \,|\, \boldsymbol{S}_t^{(W)}], \qquad \boldsymbol{\Sigma} = \mathrm{var}_{\mathrm{FC}}(\Delta^{(k)} \boldsymbol{s}_t \,|\, \boldsymbol{S}_t^{(W)})$$

denote the forecast mean and variance of the k-day return vector on day t (which is usually estimated by generating M iid length-k trajectories $\boldsymbol{y}_{t+1}, \ldots, \boldsymbol{y}_{t+k}$ as discussed above). Let $\boldsymbol{q} = (q_1, \ldots, q_D) \in \mathbb{R}^D$ denote the amount to buy or short-sell of each asset i today, on day t. Then the value on day $t+k$ of the portfolio is $\mathcal{P}_{t+k} = \boldsymbol{q}' \Delta^{(k)} \boldsymbol{s}_k$, which has forecasted mean and variance

$$E_{\mathrm{FC}}[\mathcal{P}_{t+k} \,|\, \boldsymbol{S}_t^{(W)}] = \boldsymbol{q}' \boldsymbol{\mu}, \qquad \mathrm{var}_{\mathrm{FC}}(\mathcal{P}_{t+k} \,|\, \boldsymbol{S}_t^{(W)}) = \boldsymbol{q}' \boldsymbol{\Sigma} \boldsymbol{q}.$$

The principle of Modern portfolio theory (MPT) is to pick the weights $\boldsymbol{q}$ of the portfolio to minimize the variance of a desired expected payoff, $\boldsymbol{q}' \boldsymbol{\mu} = C$. In other words, the MPT "optimal" portfolio allocation solves the constrained minimization problem

$$\min \boldsymbol{q}' \boldsymbol{\Sigma} \boldsymbol{q} \qquad | \qquad \boldsymbol{q}' \boldsymbol{\mu} = C, \tag{1.3}$$

which by Lagrange multiplies, is obtained by setting $\hat{\boldsymbol{q}} = \lambda \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}$, where $\lambda = C / \boldsymbol{\mu}' \boldsymbol{\Sigma} \boldsymbol{\mu}$.

Note that the optimal solution to the MPT allocation problem (1.3) is $\hat{\boldsymbol{q}} \propto \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}$, for any expected payoff value C. Thus for the calculation of rolling-window p-values, the value of C is immaterial (i.e., any C gives exactly the same result). What does matter is the value of k, i.e., how often you will change your investment strategy.

Also, note that even if you set $k = 30$, i.e., only change your strategy every month, you can make daily forecasts of $\mathcal{P}_{t+1} = \hat{q} q' \Delta^{(1)} \boldsymbol{s}_t$ which gives far more p-values to evaluate your model than if you only calculate one p-value for every $k = 30$ days.

2. Generalized Hyperbolic Regression Modeling for Survival Data

Survival Analysis is concerned with the estimation of the time-to-occurence of an event of interest, e.g., death of a patient from a given disease or failure of an electronic component. A number of computational methods and datasets for survival analysis are available in the **R** package `survival`.

Let $y \in \mathbb{R}$ denote the log-survival time response variable for a given subject, and $(\boldsymbol{x}, \boldsymbol{w})$ a pair of (possibly overlapping) covariate vectors having respective dimensions p and q. In class, we considered a Heteroscedastic Linear regression Model (HLM) of the form

$$y = \boldsymbol{x}'\boldsymbol{\beta} + \exp(\boldsymbol{w}'\boldsymbol{\gamma})^{1/2} \cdot \varepsilon, \qquad \varepsilon \sim \mathcal{N}(0, 1). \tag{2.1}$$

where $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ are unknown parameters to be estimated from data. The HLM model (2.1) has good computational properties, and allows us to capture covariate-dependent location and scaling of the response. However, it would be nice if we could also capture skewness and kurtosis. To this end, consider a normal variance-mean mixture model, defined as

$$y = \boldsymbol{x}'\boldsymbol{\beta} + \alpha V + [\exp(\boldsymbol{w}'\boldsymbol{\gamma})V]^{1/2} \cdot Z, \tag{2.2}$$

where $Z \sim \mathcal{N}(0, 1)$, and $V > 0$ is some continuous random variable independent of Z. It is not too difficult to show that

$$\begin{aligned} E[y \,|\, \boldsymbol{x}, \boldsymbol{w}] &= \boldsymbol{x}'\boldsymbol{\beta} + \alpha\, E[V], \\ \operatorname{var}(y \,|\, \boldsymbol{x}, \boldsymbol{w}) &= \alpha^2 \operatorname{var}(V) + \exp(\boldsymbol{w}'\boldsymbol{\gamma})\, E[V], \end{aligned}$$

indicating that, like HLM, model (2.2) has a simple location-scale interpretation for its parameters $\boldsymbol{\beta}$, $\boldsymbol{\gamma}$, and α. Now we wish to find a flexible mixing variable V with good computational properties as well. A well-suited candidate for this is the Generalized Inverse-Gaussian (GIG) distribution:

$$V \sim \text{GIG}(\psi, \eta, \lambda) \qquad \Longleftrightarrow \qquad f_V(v \,|\, \psi, \eta, \lambda) = \frac{(v/\eta)^{\lambda-1}}{2\eta K_\lambda(\psi)} \exp\left\{-\frac{\psi}{2}\left(\frac{v}{\eta} + \frac{\eta}{v}\right)\right\}, \tag{2.3}$$

where $\psi, \eta > 0$ and $K_a(x)$ is the modified Bessel function of the second kind (given by the `besselK` function in **R**). To avoid parameter identifiability issues we set $\eta = 1$, such that y given $(\boldsymbol{x}, \boldsymbol{w})$ and the model parameters $\boldsymbol{\theta} = (\boldsymbol{\beta}, a, \boldsymbol{\gamma}, \psi, \lambda)$ follows a Generalized Hyperbolic

Distribution,

$$\begin{aligned} y \,|\, \boldsymbol{x}, \boldsymbol{w} &\sim \text{GH}(\mu, \alpha, \sigma, \psi, \lambda), \qquad \mu = \boldsymbol{x}'\boldsymbol{\beta}, \quad \sigma^2 = \exp(\boldsymbol{w}'\boldsymbol{\gamma}) \\ \iff \quad f(y \,|\, \boldsymbol{x}, \boldsymbol{w}, \boldsymbol{\theta}) &= \left(\frac{Q_{\boldsymbol{\theta}}(y)}{R_{\boldsymbol{\theta}}}\right)^{\frac{\lambda}{2}-\frac{1}{4}} \frac{K_{\lambda-1/2}\left(\sqrt{Q_{\boldsymbol{\theta}}(y)\cdot R_{\boldsymbol{\theta}}}\right)}{\sigma\sqrt{2\pi}\,K_\lambda(\psi)} \exp\left\{\frac{y-\mu}{\sigma^2/\alpha}\right\}, \end{aligned} \tag{2.4}$$

where $Q_{\boldsymbol{\theta}}(y) = \psi + (y-\mu)^2/\sigma^2$ and $R_{\boldsymbol{\theta}} = \psi + (\alpha/\sigma)^2$. As pointed out in the link above, the family of Generalized Hyperbolic distributions contains many other important distributions such as the Normal, Student-t, and Laplace distributions. Moreover, the Generalized Hyperbolic Regression (GHR) model (2.4) has excellent computational properties, as we'll see momentarily.

2.1. Bayesian Inference for the GHR Model

Suppose that we have collected n observations $(y_1, \boldsymbol{x}_1, \boldsymbol{w}_1), \ldots, (y_n, \boldsymbol{x}_n, \boldsymbol{w}_n)$, and define the notation $\boldsymbol{y} = (y_1, \ldots, y_n)$, $\boldsymbol{X}_{n\times p} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$, $\boldsymbol{W}_{n\times q} = (\boldsymbol{w}_1, \ldots, \boldsymbol{w}_n)$, and $\boldsymbol{D} = (\boldsymbol{y}, \boldsymbol{X}, \boldsymbol{W})$. Now suppose that

$$y_i \,|\, \boldsymbol{x}_i, \boldsymbol{w}_i \overset{\text{ind}}{\sim} \text{GH}(\boldsymbol{x}_i'\boldsymbol{\beta}, \alpha, e^{\boldsymbol{w}_i'\boldsymbol{\gamma}/2}, \psi, \lambda),$$

are drawn from the GHR model (2.4). Then for a flat prior $\pi(\boldsymbol{\theta}) \propto 1$, Bayesian inference can be conducted by sampling from the posterior distribution $p(\boldsymbol{\theta} \,|\, \boldsymbol{D})$ via an effective MCMC algorithm described below. The key to this algorithm is the concept of *data augmentation*. That is, suppose that for each observation i we additionally have the mixing variable V_i. Then the complete data model can be written as

$$\begin{aligned} V_i \,|\, \boldsymbol{x}_i, \boldsymbol{w}_i &\overset{\text{iid}}{\sim} \text{GIG}(\psi, 1, \lambda) \\ y_i \,|\, V_i, \boldsymbol{x}_i, \boldsymbol{w}_i &\overset{\text{ind}}{\sim} \mathcal{N}\Big(\boldsymbol{x}_i'\boldsymbol{\beta} + \alpha V_i, \exp(\boldsymbol{w}_i'\boldsymbol{\gamma}) V_i\Big). \end{aligned} \tag{2.5}$$

The idea of data augmentation is that, while MCMC sampling from $p(\boldsymbol{\theta} \,|\, \boldsymbol{D})$ is hard, sampling from the augmented posterior distribution $p(\boldsymbol{\theta}, \boldsymbol{V} \,|\, \boldsymbol{D})$ is comparatively straightforward. Moreover, if $(\boldsymbol{\theta}^{(1)}, \boldsymbol{V}^{(1)}), \ldots, (\boldsymbol{\theta}^{(M)}, \boldsymbol{V}^{(M)})$ are MCMC samples with stationary distribution $p(\boldsymbol{\theta}, \boldsymbol{V} \,|\, \boldsymbol{D})$, then *marginally* $\boldsymbol{\theta}^{(1)}, \ldots, \boldsymbol{\theta}^{(M)}$ have stationary distribution $p(\boldsymbol{\theta} \,|\, \boldsymbol{D}) = \int p(\boldsymbol{\theta}, \boldsymbol{V} \,|\, \boldsymbol{D})\, \mathrm{d}\boldsymbol{V}$[4].

With a flat prior $\pi(\boldsymbol{\theta}) \propto 1$, the augmented posterior distribution for the complete data

[4]This is exactly the same principle by which the histogram of any single θ_j from an MCMC sampler on $p(\boldsymbol{\theta} \,|\, \boldsymbol{D})$ converges to the marginal posterior $p(\theta_j \,|\, \boldsymbol{D})$.

model (2.5) can be written as

$$\begin{aligned}\log p(\boldsymbol{\theta}, \boldsymbol{V} \mid \boldsymbol{D}) &= \log p(\boldsymbol{y}, \boldsymbol{V} \mid \boldsymbol{\theta}, \boldsymbol{X}, \boldsymbol{W}) \\ &= -\sum_{i=1}^{n} \left[\frac{(y_i - \boldsymbol{x}_i'\boldsymbol{\beta} - \alpha V_i)^2}{2 \cdot \exp(\boldsymbol{w}_i'\boldsymbol{\gamma}) V_i} + \frac{\boldsymbol{w}_i'\boldsymbol{\gamma}}{2} + (\tfrac{3}{2} - \lambda) \log V_i + \frac{\psi}{2}\left(V_i + V_i^{-1}\right) + \log K_\lambda(\psi) \right].\end{aligned} \tag{2.6}$$

A Gibbs sampler for this posterior distribution can be constructed as follows.

• *Conditional update of $\boldsymbol{\beta}$.* By discarding all terms in (2.6) which don't involve $\boldsymbol{\beta}$, we arrive at

$$\log p(\boldsymbol{\beta} \mid \mathcal{A}_{-\beta}) = -\frac{1}{2} \sum_{i=1}^{n} \frac{(y_i - \boldsymbol{x}_i'\boldsymbol{\beta} - \alpha V_i)^2}{\sigma_i^2}, \qquad \sigma_i^2 = \exp(\boldsymbol{w}_i'\boldsymbol{\gamma}) V_i,$$

where $\mathcal{A}_{-\beta}$ denote all variables in the model except $\boldsymbol{\beta}$. This is a quadratic function in $\boldsymbol{\beta}$, which means that the conditional posterior of $\boldsymbol{\beta}$ must be normal. With some algebra we find that

$$\boldsymbol{\beta} \mid \mathcal{A}_{-\beta} \sim \mathcal{N}(\hat{\boldsymbol{\beta}}, \hat{\boldsymbol{\Sigma}}),$$

where $\hat{\boldsymbol{\beta}} = (\boldsymbol{X}'\boldsymbol{\Delta}\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{\Delta}(\boldsymbol{y} - \alpha\boldsymbol{V})$, $\hat{\boldsymbol{\Sigma}} = (\boldsymbol{X}'\boldsymbol{\Delta}\boldsymbol{X})^{-1}$, and $\boldsymbol{\Delta} = \text{diag}(\sigma_1^{-2}, \ldots, \sigma_n^{-2})$.

Note: Let $\boldsymbol{X}$ and $\boldsymbol{\delta}$ such that $\boldsymbol{\Delta} = \text{diag}(\boldsymbol{\delta})$ be given. Contrast the following methods to compute $\boldsymbol{X}'\boldsymbol{\Delta}\boldsymbol{X}$ in **R**:

```
# Assume expressions for X and del are given
XDX <- crossprod(X, diag(del) %*% X) # BAD: matrix multiplication in 2nd term is O(n^2*p)
XDX <- crossprod(X, X * del) # GOOD: mat mult in 2nd term is O(n*p)
```

Alternatively you can extract the relevant computations from **`bayeslm-functions.R`** posted on LEARN.

• *Conditional update of $\boldsymbol{\gamma}$.* Discarding all terms in (2.6) which don't depend on $\boldsymbol{\gamma}$, we obtain

$$\log p(\boldsymbol{\gamma} \mid \mathcal{A}_{-\gamma}) = -\frac{1}{2} \sum_{i=1}^{n} \left[\frac{R_i}{\exp(\boldsymbol{w}_i'\boldsymbol{\gamma})} + \boldsymbol{w}_i'\boldsymbol{\gamma} \right], \qquad R_i = \frac{(y_i - \boldsymbol{x}_i'\boldsymbol{\beta} - \alpha V_i)^2}{V_i},$$

where $\mathcal{A}_{-\gamma}$ denotes all variables in the model except $\boldsymbol{\gamma}$. This does not correspond to a distribution from which we can sample from analytically. However, $\log p(\boldsymbol{\gamma} \mid \mathcal{A}_{-\gamma})$ is convex, and thus its mode can be easily obtained by the Newton-Raphson algorithm. In fact, we saw in class that $\log p(\boldsymbol{\gamma} \mid \mathcal{A}_{-\gamma})$ has exactly the same form as the Gamma

regression

$$R_i \,|\, \boldsymbol{w}_i \overset{\text{ind}}{\sim} \text{Gamma}\Big(\tfrac{1}{2}, 2\exp(\boldsymbol{w}_i'\boldsymbol{\gamma})\Big),$$

the mode of which can be calculated with **R**'s built-in function `glm`[5]. Moreover, the (inverse) quadrature of `glm` objects can be extracted with `vcov`, from which we obtain a Metropolized-Independence multivariate normal proposal matching the mode and quadrature of $\log p(\boldsymbol{\gamma} \,|\, \boldsymbol{\mathcal{A}}_{-\boldsymbol{\gamma}})$.

- *Conditional update of* $\boldsymbol{V}$. Upon simplifying (2.6), we find that

$$\log p(\boldsymbol{V} \,|\, \boldsymbol{\mathcal{A}}_{-\boldsymbol{V}}) = -\sum_{i=1}^{n} \left[A_i \log V_i + B_i V_i + C_i / V_i\right],$$

of which each term in the summation is precisely the log-PDF of a GIG distribution (2.3) up to its normalizing constant. Therefore,

$$V_i \,|\, \boldsymbol{\mathcal{A}}_{-\boldsymbol{V}} \overset{\text{ind}}{\sim} \text{GIG}(\hat{\psi}_i, \hat{\eta}_i, \hat{\lambda}_i),$$

where $\hat{\psi}_i$, $\hat{\eta}_i$, and $\hat{\lambda}_i$ are functions of A_i, B_i, and C_i. To sample from a GIG distribution, you may use the **R** function `rgig` in **`ghyp-functions.R`**. However, this requires you to compile the underlying **C++** code, for which instructions can be found in Appendix A.

- *Conditional updates of* ψ *and* λ. Simplifying (2.6) gives

$$p(\lambda, \psi \,|\, \boldsymbol{\mathcal{A}}_{-(\lambda,\psi)}) = \lambda S - \psi T - n \log K_\lambda(\psi), \qquad \begin{aligned} S &= \textstyle\sum_{i=1}^{n} \log V_i, \\ T &= \textstyle\sum_{i=1}^{n} \left(V_i + V_i^{-1}\right). \end{aligned}$$

While this log-PDF is convex, calculating the mode via Newton-Raphson requires the gradient and Hessian of $K_\lambda(\psi)$, for which numerically stable algorithms do not seem to be readily available. Therefore, use MWG updates for each component of (λ, ψ).

2.2. The Effect of Right-Censoring

A common setting in survival analysis is that time-to-event responses are subject to right-censoring. That is, instead of always observing the true log-survival time y_i, we sometimes observe $Y_i = \min(y_i, C_i)$ and $\delta_i = \mathbb{1}\{Y_i = y_i\}$, where C_i is the censoring time of subject i. Censoring typically occurs because the event happens after the end of the study, patient moves to a different city or becomes too sick to continue participating in the study, etc.

Assume that the censoring time C is fixed or random, but conditionally independent of y

[5] See class notes and **`hlm-functions.R`**.

given the covariates $(\boldsymbol{x}, \boldsymbol{w})$. A data augmentation framework puts

$$\begin{aligned} \boldsymbol{y}_{\text{obs}} &= (\boldsymbol{Y}, \boldsymbol{\delta}, \boldsymbol{X}, \boldsymbol{W}), \\ \boldsymbol{y}_{\text{miss}} &= (\boldsymbol{V}, \{y_i : \delta_i = 1\}), \qquad \boldsymbol{y}_{\text{comp}} = (\boldsymbol{V}, \boldsymbol{D}, \boldsymbol{\delta}). \end{aligned}$$

Since y and C are conditionally independent, the censoring mechanism is said to be ignorable, such that $p(\boldsymbol{\theta}, \boldsymbol{V} \mid \boldsymbol{D}, \boldsymbol{\delta}) = (\boldsymbol{\theta}, \boldsymbol{V} \mid \boldsymbol{D})$ in (2.6). Thus, the MCMC in the presence of censoring proceeds exactly like the uncensored case, with the added step of Gibbs sampling from the unobserved event times from a truncated normal distribution,

$$y_i \mid Y_i, V_i, \boldsymbol{x}_i, \boldsymbol{w}_i, \delta_i = 0 \overset{\text{ind}}{\sim} \mathcal{N}(\mu_i + \alpha V_i, \sigma_i^2 V_i) \times \mathbb{1}\{y > Y_i\}.$$

To sample from this, note that if $U \sim \text{Unif}(0, 1)$ and $\Phi(z)$ is the CDF of $\mathcal{N}(0, 1)$, then

$$\begin{aligned} y &= \Phi^{-1}(1 - aU) \\ &\sim \mathcal{N}(0, 1) \times \mathbb{1}\{y > a\}. \end{aligned}$$

A nice dataset for assessing the effect of censoring is the `colon` dataset in `survival`.

A. Interfacing C++ code from R

The package to do this is called `Rcpp`, which allows you both to compile **C++** code on-the-fly, or inside an **R** package. For a simple example of how to use the first approach, see the documentation for `Rcpp::sourceCpp`. The second approach is highly recommended for this project. Details instructions can be found in the `Rcpp` vignettes or in this chapter of Hadley Wickham's book.

The main challenge of using `Rcpp` is installing the package itself. While it's completely straightforward on Linux, for OSX or Windows you need to install an **R**-compliant **C++** compiler. Here's how to do this:

- **On Windows.** Install the latest version of `Rtools`, which can be downloaded here. Let the installer modify your system variable `Path`.
- **On OSX.** Install the `Xcode` command line tools. Then open `Terminal` and enter the command `xcode-select --install`. You can simply press "install" without also obtaining the entire `Xcode` suite.

To check that the **C++** compiler is set up correctly, install `Rcpp` and run the following **R** code:

```
Rcpp::cppFunction("double AddTest(double x, double y) {return x + y;}")
AddTest(5.2, 3.4)
```

If the output is 8.6 then you are good to go.

Note: Several **R** packages contain random number generators for the GIG distribution which don't need you to compile **C++** code, such as `rmutil`, `GeneralizedHyperbolic`, and `Runuran`. Not only are these algorithms about 50-100x slower than that provided by **`randGIG.cpp`**, but none of them allow for vectorized sampling. In other words, if you want to draw $n = 500$ GIGs each with its own parameter set $(\psi_1, \eta_1, \lambda_1), \ldots, (\psi_n, \eta_n, \lambda_n)$, then the other packages require you to use a for-loop, thus resulting in more like 1000x slower computations than **`randGIG.cpp`**. Therefore, you will be rewarded for the effort to integrate this **C++** code into your MCMC algorithm.