STSB6816 Test 1 of 2024 - Corrected memorandum

Mathematical Statistics and Actuarial Science; University of the Free State

2024/04/04

Time: 170 minutes; Marks: 45

MEMORANDUM

Instructions

- Answer all questions in a single R Markdown document. Please knit to PDF or Word at the end and submit both the PDF/Word document and the ".Rmd" file for assessment, in that order.
- Label questions clearly, as it is done on this question paper.
- All results accurate to about 3 decimal places.
- Show all derivations, formulas, code, sources, and reasoning.
- Intervals should cover 95% probability unless stated otherwise.
- No communication software, devices, or communication capable websites may be accessed prior to submission. You may not (nor even appear to) attempt to communicate or pass information to another student.
- Use of AI tools must be disclosed and summarised.

Introduction

The data is provided at https://ufs.blackboard.com. It is the monthly log-returns on a company share price over time (assume regular time intervals).

Senior analysts have suggested that you model the log-returns using a normal distribution as trading was stable during this period of interest. However, they point out that the company experienced gradually deteriorating sentiment during this period, depressing returns; along with gradually increasing volatility due to company directors behaving more and more erratically on social media.

Question 1

1.1) Write down the density function and log density function of Y_i where $Y_i \sim N(\mu_i, \sigma_i^2)$. **[2]**

```
cat("$$f(y_i)=(2\\pi)^{-0.5}\\sigma_i^{-1}\\exp \\{-0.5[(y_i -
\\mu_i)/\\sigma_i]^2\\}$$\n\n")
```

$$f(y_i) = (2\pi)^{-0.5}\sigma_i^{-1}\exp\{-0.5[(y_i - \mu_i)/\sigma_i]^2\}$$

```
cat("$$\\log f(y_i)=-0.5\\log(2\\pi) - \\log\\sigma_i - 0.5[(y_i -
\\mu_i)/\\sigma_i]^2$$\n\n")
```

$$\log f(y_i) = -0.5\log(2\pi) - \log\sigma_i - 0.5[(y_i - \mu_i)/\sigma_i]^2$$

Both typed out correctly to a multiplicative constant [2].

1.2) Now let $\mu_i = \beta_0 + \beta_1 x_i$ and $\sigma_i = \delta_0 + \delta_1 x_i$, where X is an explanatory variable (time in this case) and $\delta_0, \delta_1 > 0$. Derive the **log** likelihood of a sample $(y_1, y_2, \ldots, y_n)$ after plugging in these expressions. **[2]**

[Note, for interest, that this is *not* a standard model specification. *It will work* here though, if implemented correctly.]

```
cat("$$\\begin{aligned}
\\ell &= \\sum_{i=1}^n \\log f(y_i) \\\\
&=-0.5n\\log(2\\pi) - \\sum_{i=1}^n\\log(\\delta_0 + \\delta_1 x_i) -
0.5\\sum_{i=1}^n\\{[y_i - (\\beta_0 + \\beta_1 x_i)]/(\\delta_0 + \\delta_1
x_i)\\}^2
\\end{aligned}$$")
```

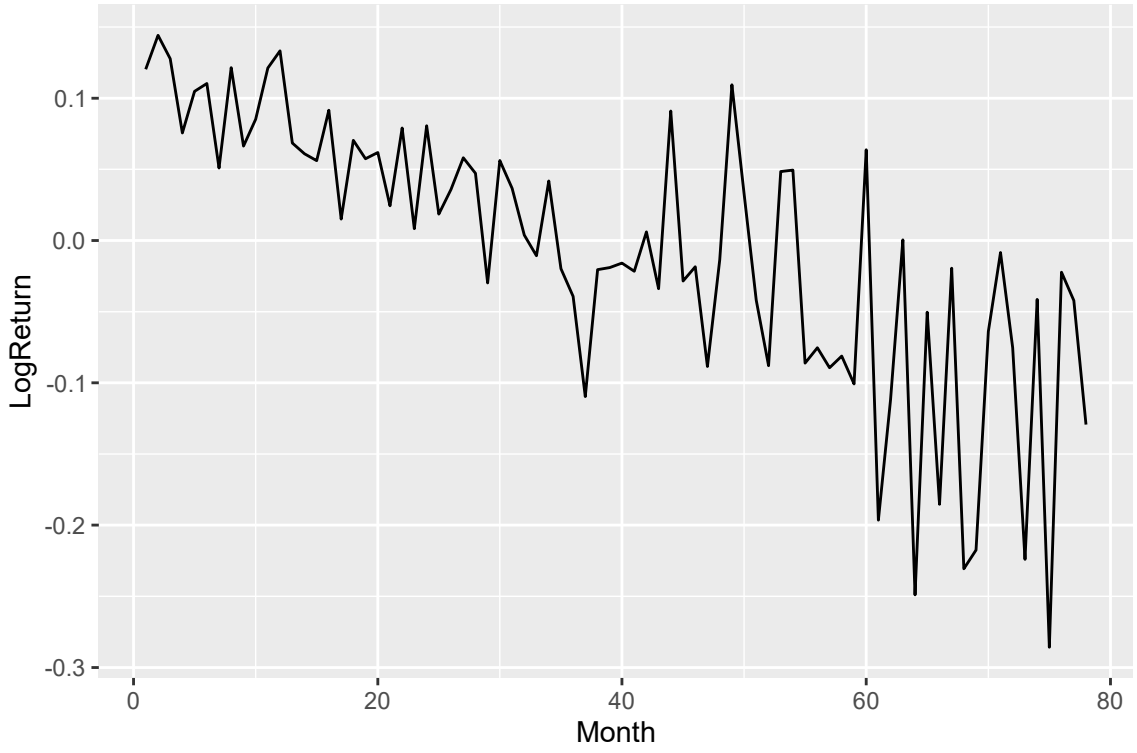
$$\begin{aligned}
\ell &= \sum_{i=1}^{n} \log f(y_i) \\
&= -0.5n\log(2\pi) - \sum_{i=1}^{n} \log(\delta_0 + \delta_1 x_i) - 0.5 \sum_{i=1}^{n} \{[y_i - (\beta_0 + \beta_1 x_i)]/(\delta_0 + \delta_1 x_i)\}^2
\end{aligned}$$

Typed out correctly to a multiplicative constant [2].

1.3) Import the data set into your statistical software and explore it visually. You could use a time series plot and discuss what you see. **[4]**

```
library(tidyverse)

"STSB6816Test1Data2024.xlsx" |> openxlsx::read.xlsx("TestData") -> d
d |> ggplot(aes(x = Month, y = LogReturn)) + geom_line()
```



Loading data [1], time series plot [1], and discussion saying something about downward slope and increasing variance [2].

1.4) Fit the model using uniform priors and find posterior mode estimates, or fit the model via the maximum likelihood approach and provide parameter estimates. **[10]**

[Any valid method will receive full credit (here only). You could use Stan's *optimizing*, or simulate the posterior (*sampling*) and then use kernel density estimates of the modes (via *density* perhaps), or try *mle* or *optim*.]

Hint: the parameter estimates you obtain should be (very roughly) about: $\delta_0 = 0.07$, $\delta_1 = 0.002$, $\beta_0 = 0.12$, $\beta_1 = -0.003$. *If you are unable to obtain reasonable estimates in good time then use these values in further questions that rely on them.*

Correction: the estimates should have been closer to 0.02 and 0.001 for the deltas.

```
library(rstan)
options(mc.cores = 3)

data {
  int<lower=0> n;         // number of observations
  real x[n];             // explanatory observations
  real y[n];           // dependent observations
}
parameters {
  real<lower=0> delta_0;
  real<lower=0> delta_1;
  real beta_0;
  real beta_1;
}
model {
  for (i in 1:n) {
    target += - log(delta_0 + delta_1*x[i]) - 0.5*((y[i] - (beta_0 +
beta_1*x[i]))/(delta_0 + delta_1*x[i]))^2;
  }
}

pars_of_interest <- c('delta_0', 'delta_1', 'beta_0', 'beta_1')
n <- nrow(d)
stan_data <- list(n = n, x = d$Month, y = d$LogReturn)

# First we use the optimisation via Stan approach
post_optim1 <- BayesTest1of2024model |>
  optimizing(stan_data)
post_optim1$par |> round(6)
```

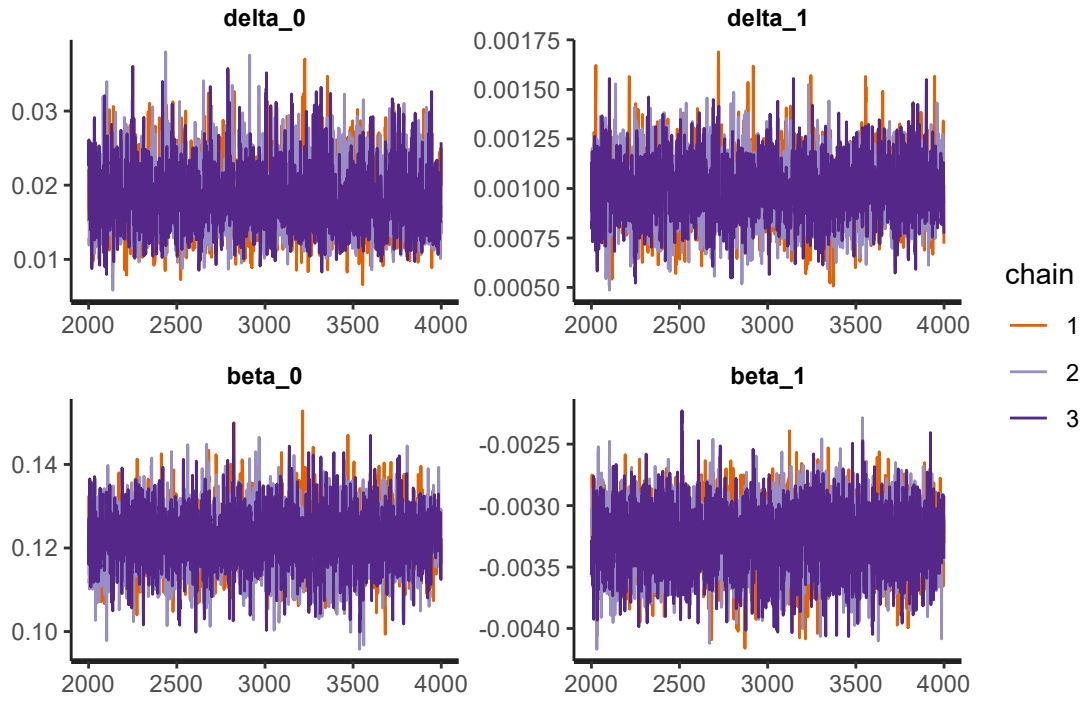
delta_0	delta_1	beta_0	beta_1
0.061521	0.000012	0.120837	-0.003237

```
# Now we use the simulation and kernel density approach:
post_fit1 <- BayesTest1of2024model |>
  sampling(data = stan_data,
           pars = pars_of_interest,
           chains = 3,
           iter = 4000
           )
```

```
# Optional evaluation of the posterior fit
(post_fit1 |> summary(pars = pars_of_interest))$summary |> kable(digits = 3)
```

	mean	se_mean	sd	2.5%	25%	50%	75%	97.5%	n_eff	Rhat
delta_0	0.018	0	0.005	0.011	0.015	0.018	0.021	0.028	1801.209	1.002
delta_1	0.001	0	0.000	0.001	0.001	0.001	0.001	0.001	1709.911	1.003
beta_0	0.122	0	0.007	0.108	0.118	0.122	0.127	0.136	2714.009	1.001
beta_1	-0.003	0	0.000	-0.004	-0.003	-0.003	-0.003	-0.003	3461.432	1.000

```
post_fit1 |> traceplot(pars = pars_of_interest)
```




```
# Posterior mode estimates
post_sims1 <- post_fit1 |> rstan::extract(pars = pars_of_interest)
post_sims1 |> sapply(\(sims) {
  dens <- density(sims)
  dens$x[which.max(dens$y)]
}) |> round(6)
```

delta_0	delta_1	beta_0	beta_1
0.016675	0.001015	0.122499	-0.003304

```
# Lastly we will use straight maximum likelihood
# This approach requires first coding a function that calculates the
# negative log likelihood as a function of the parameter vector
neglogl <- \(p, x, y) {
  # sum(log(p[1] + p[2]*x)) + 0.5*sum(((y - (p[3] + p[4]*x))/(p[1] +
p[2]*x))^2)
  # Or
  -sum(dnorm(y, (p[3] + p[4]*x), (p[1] + p[2]*x), log = TRUE))
}
# And then specifying initial values and constraints, plus data
mle_optim1 <- optim(
  par = c(0.02, 0.001, 0.1, -0.003),
```

```
  fn = neglogl,
  method = "L-BFGS-B",
  lower = c(0.00001, 0.00001, -Inf, -Inf),
  x = d$Month,
  y = d$LogReturn,
  hessian = TRUE
)
mle_optim1$par |> setNames(pars_of_interest) |> round(6)
# This method appears to fail to converge when provided with the correct
model and values
```

Obtained parameter estimates close to the posterior mode of the given model via any sensible approach [8]. Approach incorporated given constraints and not unnecessary constraints. [2]

1.5) Obtain a reasonable prediction and 95% prediction interval for the log-return of the next month $(n + 1)$. **[9]**

[Note that the interval will require simulating the posterior distribution of the parameters, or otherwise accounting for their uncertainty, if not already done.]

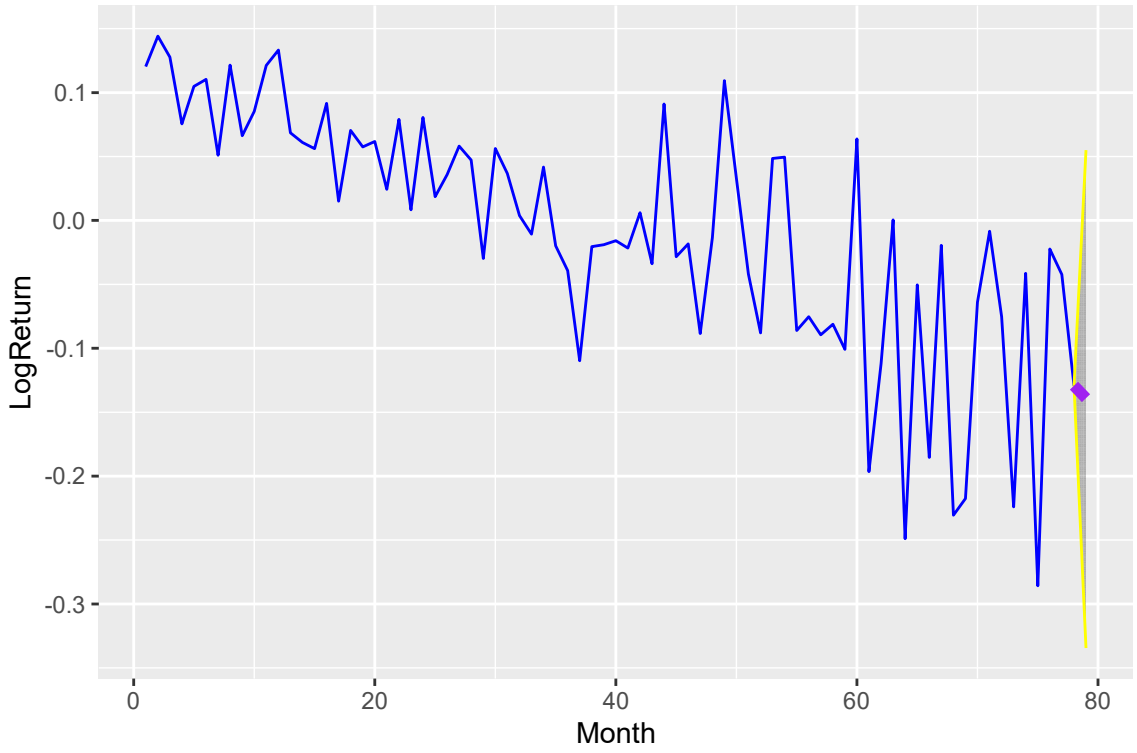
```
n_sims <- length(post_sims1$delta_0)
mu <- post_sims1$beta_1*(n+1) + post_sims1$beta_0
s <- post_sims1$delta_1*(n+1) + post_sims1$delta_0
pred_sims <- rnorm(n_sims, mu, s)
pred_ests <- pred_sims |> quantile(c(0.025, 0.5, 0.975))
cat('\n The log-return of next month is equally likely to be more or less than ',
    pred_ests[2] |> round(3),
    ' with 95% prediction interval (',
    pred_ests[1] |> round(3), '; ', pred_ests[3] |> round(3), ').\n', sep = '')
```

```
|
|  The log-return of next month is equally likely to be more or less than -0.139 with 95% prediction interval
(-0.334; 0.055).
```

Using the parameter values in the correct linear forms [1]. Using the full uncertainty of the parameter values [4]. Incorporating residual variation correctly [2]. Calculating and presenting an estimate and interval [2].

1.6) Add your prediction and interval to the plot and state whether it seems like a reasonable extension of the observed patterns. **[3]**

```
d_new <- data.frame(Month = n:(n+1),
                    LogReturn = c(d$LogReturn[n], pred_ests[2]),
                    Lower = c(d$LogReturn[n], pred_ests[1]),
                    Upper = c(d$LogReturn[n], pred_ests[3]))
d |> ggplot(aes(x = Month, y = LogReturn)) + geom_line(colour = 'blue') +
  geom_ribbon(aes(ymin = Lower, ymax = Upper), data = d_new, alpha = 0.3, colour =
'yellow') +
  geom_line(data = d_new, colour = 'purple', linewidth = 2)
```



Plot looks about the same as before but with a slight extention showing the calculated values [2]. Statement about the width of the interval being wider than the data pattern or something similarly observant [1].

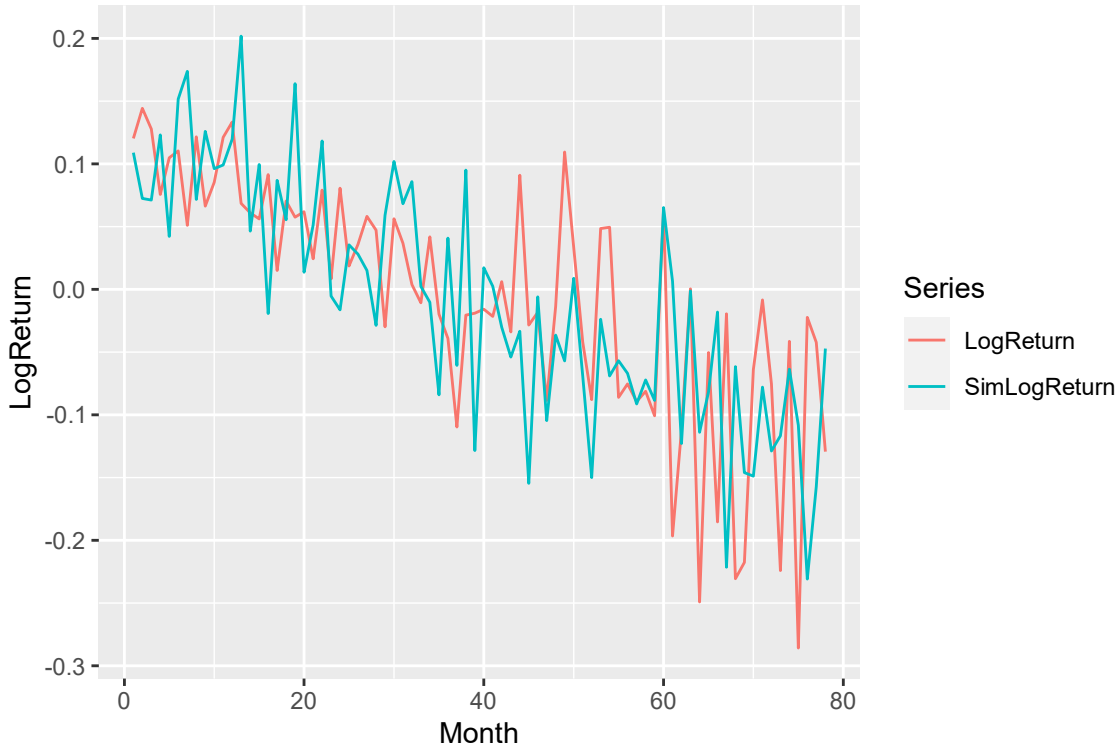
1.7) If you were formulating the model yourself, what would you change? Why and how? Would you change the how the variance is modelled or the constraints implemented? Would you change the prior distributions? Would you fit the model differently? **[6]**

Comments should include statements like: A more natural way to model the variance would be via an exponential transformation, as that allows the coefficients to be free and allows for the variance to decrease, not just increase. [2] Priors that pull the estimates towards zero might result in greater stability and accuracy. [2] Priors might be informed by expert knowledge of the company and market, if non-uniform priors are used. [2] It may be possible to transform the original series to be homoscedastic, and then use a simpler model, before transforming back. [2] Other well reasoned argument about the model or implementation. [2] **Any 3 of these get marks, to a maximum of 6 marks.**

1.8) Simulate a new time series, over the same period as your current time series, by plugging the parameter estimates you obtained into the model. Does your series look like the original? What does that tell you about your model fit? **[4]**

```
mu_hat <- post_optim1$par['beta_1']*d$Month + post_optim1$par['beta_0']
s_hat <- post_optim1$par['delta_1']*d$Month + post_optim1$par['delta_0']
d$SimLogReturn <- rnorm(n, mu_hat, s_hat)

plot_d <- d |> pivot_longer(-Month, names_to = "Series", values_to = "LogReturn")
plot_d |> ggplot(aes(x = Month, y = LogReturn, colour = Series)) + geom_line()
```

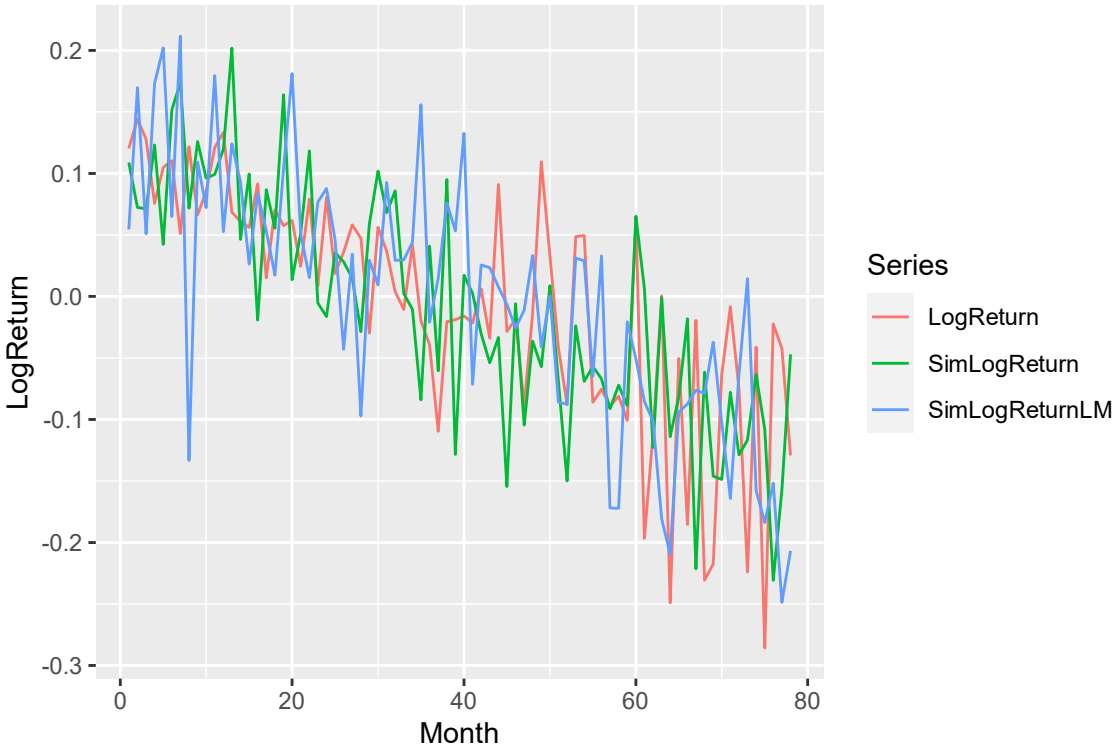


New random values simulated from plugged in expected value and standard deviation patterns [2]. Comment describing any apparent systematic differences, such as overestimated base or final variance. [2]

1.9) Fit an ordinary linear regression model (ignoring heteroscedasticity). Simulate a new time series, over the same period as your current time series, by plugging the parameter estimates you obtained from the ordinary linear model into the ordinary linear model. Does your series look like the original? Would you say the fit is better or worse? **[5]**

```
lm1 <- lm(LogReturn ~ Month, data = d)
lm1_coef <- lm1 |> coef()
mu_hat <- lm1_coef[2]*d$Month + lm1_coef[1]
s_hat <- summary(lm1)$sigma
d$SimLogReturnLM <- rnorm(n, mu_hat, s_hat)

plot_d <- d |> pivot_longer(-Month, names_to = "Series", values_to = "LogReturn")
plot_d |> ggplot(aes(x = Month, y = LogReturn, colour = Series)) + geom_line()
```



Fitting model [1], using fit and standard deviation to simulate [1], and plotting [1]. Comment about overestimating the variance early and underestimating it late. [2]

Points total

The points on the test add up to **45**
