HW3

Erik Andersen

2024-05-07

Question 1

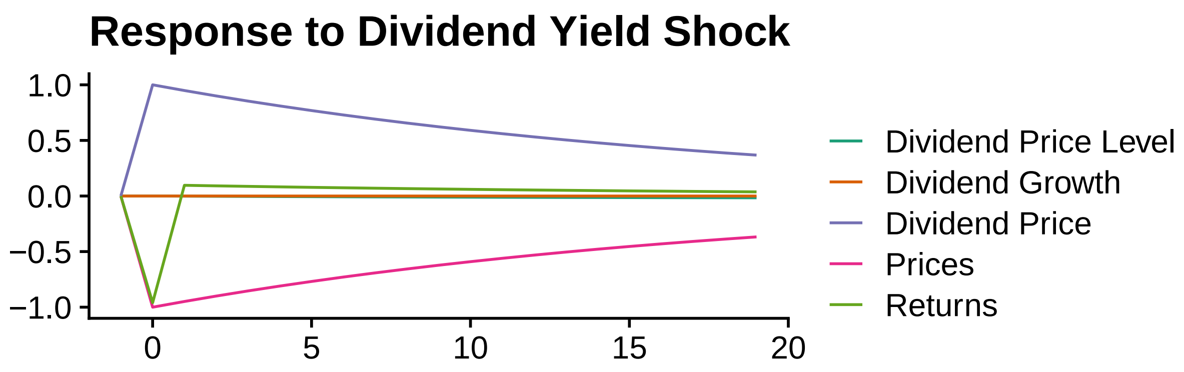
In this question, we will replicate the predictability VARs from class then replicate the impulse response functions. The equations and their estimates are given below. Besides the estimates for the standard deviations of the errors, the estimates are very close to those estimated in class. I don't know why the estimates are so much smaller for the standard deviations. Interestingly, they are numerically close, but off by two orders of magnitude.

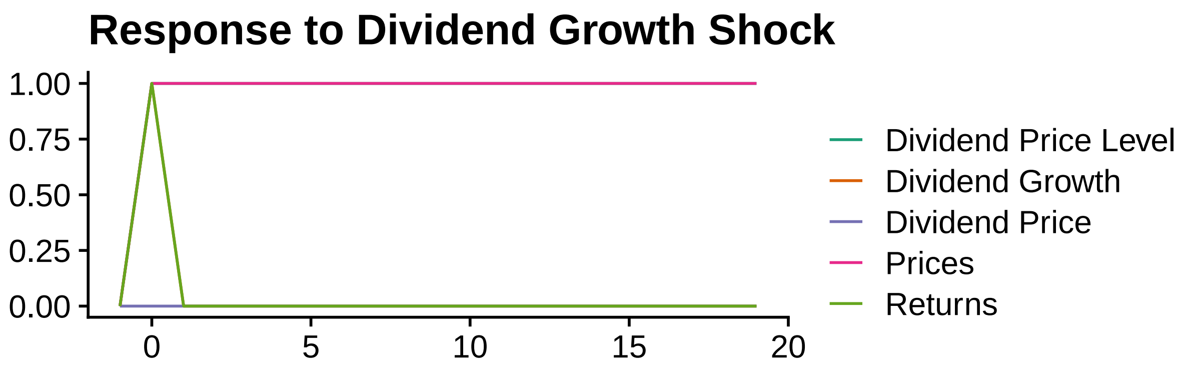
$$\begin{aligned} r_{t+1} &= b_r dp_t + e^r_{t+1} \\ \Delta d_{t+1} &= b_d dp_t + e^d + t + 1 \\ dp_{t+1} &= \Phi dp_t + e^{dp}_{t+1} \end{aligned}$$

Table 1:

	Estimates		Error Standard Deviation and Correlation		
	coefs	standard_errors	r	Div Growth	Div Price
Returns	0.0962	0.0617	0.2682	0.6422	-0.2860
Div Growth	-0.0013	0.0327	0.6422	0.1420	0.0876
Div Price	0.9487	0.0340	-0.2860	0.0876	0.1476

Now, I'll plot the impulse response functions following a one unit shock to dividend yield, and a one unit shock to dividends. The first graph below shows the response to a one unit shock to dividend yield. The second shows the same for a shock to dividend growth.





Question 2

In this question, we will look at an extension to the Mehra Prescott model of the equity premium puzzle. The transition probabilities matrix is given below.

$$\begin{bmatrix} \phi & 1-\phi-\eta & \eta \\ 1-\phi-\eta & \phi & \eta \\ \frac{1}{2}x & \frac{1}{2} & 0 \end{bmatrix}$$

a) In this question, I will compute the stationary probabilities, $\bar{q}$ of the transition matrix listed above. For stationarity, the following condition must hold: $\bar{q} = Q'\bar{q}$ where Q is the transition matrix. That corresponds to the following system of four equations. The final equation comes from the condition that the sum of the probabilities must equal 1.

$$\begin{aligned} q_1 &= \phi q_1 + (1-\phi-\eta)q_2 + \frac{1}{2}q_3 \\ q_2 &= (1-\phi-\eta)q_1 + \phi q_2 + \frac{1}{2}q_3 \\ q_3 &= \eta q_1 + \eta q_2 \\ 1 &= q_1 + q_2 + q_3 \end{aligned}$$

Solving this system of equations gives us the following values for $\bar{q}$.

$$
\begin{aligned}
q_1 &= \frac{1}{2(1+\eta)} \\
q_2 &= \frac{1}{2(1+\eta)} \\
q_3 &= \frac{\eta}{1+\eta}
\end{aligned}
$$

b) In this question, I will compute the first and second moments of x_t as well as $E[x_t x_{t-1}]$. X is defined as follows. $x \in \lambda_1, \lambda_2, \lambda_3$ where the $\lambda' s$ are defined below.

$$
\begin{aligned}
\lambda_1 &= 1+\mu+\delta \\
\lambda_2 &= 1+\mu-\delta \\
\lambda_3 &= \frac{1+\mu}{2}
\end{aligned}
$$

The first moment of x_t is given by $x_t \bar{q}$, or the expected value with the expectation relative to the stationary probabilities.

$$
\begin{aligned}
E[x_t] = &\begin{bmatrix} 1+\mu+\delta & 1+\mu-\delta & \frac{1+\mu}{2} \end{bmatrix} \begin{bmatrix} \frac{1}{2(1+\eta)} \\ \frac{1}{2(1+\eta)} \\ \frac{\eta}{1+\eta} \end{bmatrix} \\
=&\frac{1}{2(1+\eta)} 2(1+\mu) + \frac{\eta}{1+\eta}\frac{1+\mu}{2} \\
=&\frac{(2+\eta)(1+\mu)}{2(1+\eta)}
\end{aligned}
$$

The second moment is calculated as follows.

$$
\begin{aligned}
E[x_t^2] = &\begin{bmatrix} (1+\mu+\delta)^2 & (1+\mu-\delta)^2 & (\frac{1+\mu}{2})^2 \end{bmatrix} \begin{bmatrix} \frac{1}{2(1+\eta)} \\ \frac{1}{2(1+\eta)} \\ \frac{\eta}{1+\eta} \end{bmatrix} \\
=&\frac{1}{2(1+\eta)}(2(1+\mu)^2 + 2\delta^2) + \frac{\eta}{1+\eta}\frac{(1+\mu)^2}{4}
\end{aligned}
$$

0

Finally, we can calculate $E[x_t x_{t-1}]$ as follows. The inside terms are found by multiplying x_t and x_{t-1} and remembering that x_t is found by multiplying x_{t-1} by the transition matrix Q. I've skipped listing several steps of simplifying for brevity's sake.

$$
\begin{aligned}
E[x_t x_{t-1}] &= \frac{1}{2(1+\eta)}(\lambda_1^2\phi + \lambda_1\lambda_2(1-\phi-\eta) + \lambda_1\lambda_3\eta + \lambda_2\lambda_1(1-\phi-\eta) + \lambda_2^2\phi + \lambda_2\lambda_3\eta) + \frac{\eta}{1+\eta}(\lambda_3\lambda_1\frac{1}{2} + \lambda_2\lambda_3\frac{1}{2} \\
&= \frac{1}{1+\eta}(1+\mu)^2 + \frac{2\phi+\eta-1}{1+\eta}\delta^2
\end{aligned}
$$

c) In this part, I will compute sample values for the moments I calculated above given the following parameter values: $\mu = 0.018$, $\delta = 0.036$, and $\phi = -0.14$. Plugging these into the above equations, I get the following sample moments.

$$\begin{aligned} E[x_t] &= 1.018 \\ E[x_t^2] &= 1.037i \\ E[x_t x_{t-1}] &= 1.036 \end{aligned}$$

d) See the code appendix below for the program. Table 2 shows the calibrated values for μ, δ, and ϕ. They match closely to Mehra and Prescott's values.

Table 2:

Parameter	Value
Mu	0.0181
Delta	0.0349
Phi	0.4257

e) See code for the function.

f) Table 3 shows the results of the simulations looping over 0.0001, 0.0002, and 0.0005 for η, 2 and 10 for α, and 0.97, 0.997, and 0.999 for β. These are reasonable values for each parameter and doing only these values saves the table from being immensely long. Note in particular the returns and risk premium for $\eta = 0.0001$, $\alpha = 10$, and $\beta = -.997$. The risk free rate is about 3%, the expected return is about 10%, so the risk premium is 7%. All of these are inline with stylized facts about returns we see.

g) I don't think it fully solves the equity premium puzzle. We can get numbers close to the ones we observe, but it still requires a very high risk aversion level of 10. The values are also very sensitive to η which is the probability of collapse. Taking the same α and β that generate the numbers we observe, if we double η to 0.0002, all of sudden we require a negative risk free rate. That is unrealistic and its not a good feature of a model to be that sensitive to parameter values.

Table 3:

Parameters					
Eta	Alpha	Beta	Risk-Free Rate	Expected Return	Risk Premium
1e-04	2	0.970	6.450	6.742	0.2920
2e-04	2	0.970	6.419	6.726	0.3078
5e-04	2	0.970	6.323	6.678	0.3550
1e-04	2	0.997	3.568	3.852	0.2848
2e-04	2	0.997	3.537	3.837	0.3002
5e-04	2	0.997	3.444	3.790	0.3461
1e-04	2	0.999	3.360	3.645	0.2843
2e-04	2	0.999	3.329	3.629	0.2996
5e-04	2	0.999	3.237	3.583	0.3455
1e-04	10	0.970	5.760	13.246	7.4863
2e-04	10	0.970	-2.784	8.725	11.5088
5e-04	10	0.970	-21.730	-1.431	20.2982
1e-04	10	0.997	2.896	10.215	7.3187
2e-04	10	0.997	-5.416	5.827	11.2435
5e-04	10	0.997	-23.849	-4.000	19.8494
1e-04	10	0.999	2.690	9.997	7.3066
2e-04	10	0.999	-5.606	5.619	11.2244
5e-04	10	0.999	-24.002	-4.185	19.8171

Code Appendix

Erik Andersen

2024-05-07

```r
# HW 3 Code
# Load packages
pacman::p_load(tidyverse, magrittr, kableExtra, nleqslv)

# Load data
# I saved the date in HW2 where I already calculated the div-price ratio and dividend growth rate, so I
returns_df = read_csv(here::here("HW3", "data", "returns.csv"))
```

```
## Rows: 98 Columns: 6
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (6): date, vwretd, vwretx, t90ret, div_price, div_growth
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
# Rename the returns sequences to things that make sense
returns_df %<>%
  rename("dividends" = vwretd,
         "no_dividends" = vwretx)

#### Question 1 ######
# First we will replicate the VARs
# We can just run them equation by equation
reg_returns = returns_df %>% lm(log((dividends+1) /lag((dividends+1))) ~ lag(log(div_price)) ,.)
reg_div_growth = returns_df %>% lm(log(div_growth) ~ lag(log(div_price)),.)
reg_div_price = returns_df %>% lm(log(div_price) ~ lag(log(div_price)),.)

# Calculate correlations between all the various variables
return_div_growth = cor(reg_returns$residuals, reg_div_growth$residuals)
return_div_price = cor(reg_returns$residuals, reg_div_price$residuals)
growth_price = cor(reg_div_growth$residuals, reg_div_price$residuals)

# Calculate the standard deviations
returns_sd = sd(reg_returns$residuals)
div_growth_sd = sd(reg_div_growth$residuals)
div_price_sd = sd(reg_div_price$residuals)

# Make table object
coefs = c(coef(reg_returns)[2], coef(reg_div_growth)[2], coef(reg_div_price)[2])
names(coefs) = c("Returns", "Div Growth", "Div Price")

standard_errors = c(summary(reg_returns)$coefficients[2,2], summary(reg_div_growth)$coefficients[2,2],
var_cov = matrix(c(returns_sd, return_div_growth, return_div_price, return_div_growth,
                   div_growth_sd, growth_price, return_div_price, growth_price,
                   div_price_sd), byrow = T, nrow = 3)
colnames(var_cov) = c("r", "Div Growth", "Div Price")

var_table = cbind(coefs, standard_errors, var_cov)


###############
# Impulse response function for dividend growth

# Define shocks
epsilon_r = 1
epsilon_d = 1
epsilon_dp = 0

# Initialize series
ds = c(0, coefs[2]*0 + epsilon_d)
dgs = c(0, coefs[2]*0 + epsilon_d)
dps = c(0, coefs[3]*0 + epsilon_dp)
ps = c(0, (1 - coefs[3])*0 + (epsilon_d - epsilon_dp))
rs = c(0, coefs[1] *0 + epsilon_r)

# Simulate VAR forward
for(i in 2:20){
  ds = c(ds,ds[i] + coefs[2] * dps[i])
  dgs = c(dgs, coefs[2] *dps[i])
  dps = c(dps, coefs[3] *dps[i])
  ps = c(ps,ps[i] + (1 - coefs[3]) *dps[i])
  rs = c(rs, coefs[1] *dps[i])
}

impulse_response_growth = data.frame(cbind(c(-1,0:19),dgs,dps,ps,rs,ds))
colnames(impulse_response_growth) = c('t','dg','dp','p','r','d')

# Plot
plot_div_growth =
  impulse_response_growth |>
  as_tibble() |>
  pivot_longer(cols = -t) |>
  ggplot(aes(x = t, y = value, color = name)) +
  geom_line() +
  labs(x = "", y = "", color = "") +
  scale_color_brewer(palette = "Dark2", labels = c("Dividend Price Level", "Dividend Growth",
                                                   "Dividend Price", "Prices", "Returns")) +
  ggtitle("Response to Dividend Growth Shock") +
  cowplot::theme_cowplot()

ggsave(here::here("HW3", "plots", "div_growth_irf.pdf"))
```

```
## Saving 6.5 x 4.5 in image
```

```r
# Same thing for dividend yield
# Define shocks
er2 = -0.96 # rho
ed2 = 0
edp2 = 1

# Initialize series
ds2 = c(0, coefs[2]*0 + ed2)
dgs2 = c(0, coefs[2]*0 + ed2)
dps2 = c(0, coefs[3]*0 + edp2)
ps2 = c(0, (1 - coefs[3])*0 + (ed2-edp2)) # See homework for why shocks work like this
rs2 = c(0, coefs[1]*0 + er2)

# Simulate forward
for(i in 2:20){
  ds2 = c(ds2, ds2[i] + coefs[2]*dps2[i])
  dgs2 = c(dgs2, coefs[2]*dps[i])
  dps2 = c(dps2, coefs[3]*dps2[i])
  ps2 = c(ps2, ps2[i] + (1 - coefs[3])*dps2[i])
  rs2 = c(rs2, coefs[1]*dps2[i])
}

impulse_response_yield = data.frame(cbind(c(-1,0:19),dgs2,dps2,ps2,rs2,ds2))
colnames(impulse_response_yield) = c('t','dg','dp','p','r','d')

# Plot
plot_div_yield =
  impulse_response_yield |>
  as_tibble() |>
  pivot_longer(cols = -t) |>
  ggplot(aes(x = t, y = value, color = name)) +
  geom_line() +
  labs(x = "", y = "", color = "") +
  scale_color_brewer(palette = "Dark2", labels = c("Dividend Price Level", "Dividend Growth",
                                                   "Dividend Price", "Prices", "Returns")) +
  ggtitle("Response to Dividend Yield Shock") +
  cowplot::theme_cowplot()

ggsave(here::here("HW3", "plots", "div_yield_irf.pdf"))
```

```
## Saving 6.5 x 4.5 in image
```

```r
############ Question 2 ###################



### Part d
# Function to match the moments from parts b and c
# Define eta
eta = 0.0001

mpmoment = function(initial) {
  # Unpack argument. We have to do it this clunky way because the solver takes a vector
  mu = initial[1]
  delta = initial[2]
  phi = initial[3]

  # Calculate moments
  m1 = (1 + mu) * (2 + eta) / (2 + 2 * eta)
  m2 = (4 + 4 * mu^2 + 8 * mu + 4 * delta^2 + eta + 2 * eta + eta * mu^2) /
    (4 * (1 + eta))
  m3 = (delta^2 * (2 * phi + eta - 1) + (1 + mu)^2) / (1 + eta)
  return(c(m1 - 1.018, m2 - 1.03762, m3 - 1.03614256)) # These values from the calculated moments given
}

# Define parameters as
mu = 0.018
delta = 0.036
phi = -0.14

# Find values
moments = nleqslv(c(mu, delta, phi),mpmoment)$x |> as.data.frame()

colnames(moments) = c("")
rownames(moments) = c("Mu", "Delta", "Phi")


### Part e
# Function to calculate risk premium and risk free rate given different values of free parameters
risk_premia = function(eta, beta, alpha) {
  # Calculate stationary probabilities and transition matrix
  p = c(1 / (2 + 2 * eta), 1 / (2 + 2 * eta), eta / (1 + eta))
  Q = matrix(c(as.numeric(moments[3,]), 1 - as.numeric(moments[3,]) - eta, 0.5, 1 - as.numeric(moments[
             nrow = 3)

  # Calculate endowment levels
  x = c(1 + as.numeric(moments[1,]) + as.numeric(moments[2,]), 1 + as.numeric(moments[1,]) - as.numeric

  # Calculate other parameters
  xf = x^-alpha
  rf = p %*% (1 / (beta * Q %*% as.matrix(xf)) - 1)
  xd = diag(x^(1 - alpha))
  one = matrix(c(1, 1, 1), nrow = 3)
  a = diag(3) - beta * Q %*% xd
  b = beta * Q %*% xd %*% one
  w = solve(a, b)

  # Initialize returns vectors
  r = matrix(c(0, 0, 0, 0, 0, 0, 0, 0, 0), nrow = 3)
  R = matrix(c(0, 0, 0), ncol = 3)

  # Calculate return rates
  for (i in 1:3) {
    for (j in 1:3) {
      r[i, j] = x[j] * (w[j] + 1) / w[i] - 1
    }
    R[, i] = Q[i, ] %*% r[i, ]
  }
  re = p %*% t(R)
  return(c(rf, re, re - rf))
}

### PArt f


# Initialize vector for simulations
sim = c()

# Loop over parameter space
for (alp in c(2, 10)) {
  for (bet in c(0.97, 0.997, 0.999)) {
    for (et in c(0.0001, 0.0002, 0.0005)) {
      epp = round(risk_premia(et, bet, alp) * 100, 4)
      epp = c(et, alp, bet, epp)
      sim = c(sim, epp)
    }
  }
}

sim = data.frame(t(matrix(sim, nrow = 6)))
colnames(sim) = c('Eta',
                  'Alpha',
                  'Beta',
                  'Risk-Free Rate',
                  'Expected Return',
                  'Risk Premium')
```