HW4

Erik Andersen

2024-05-15

Question 1

In this question, I'll calculate the unconditional CAPM using data from CRSP.

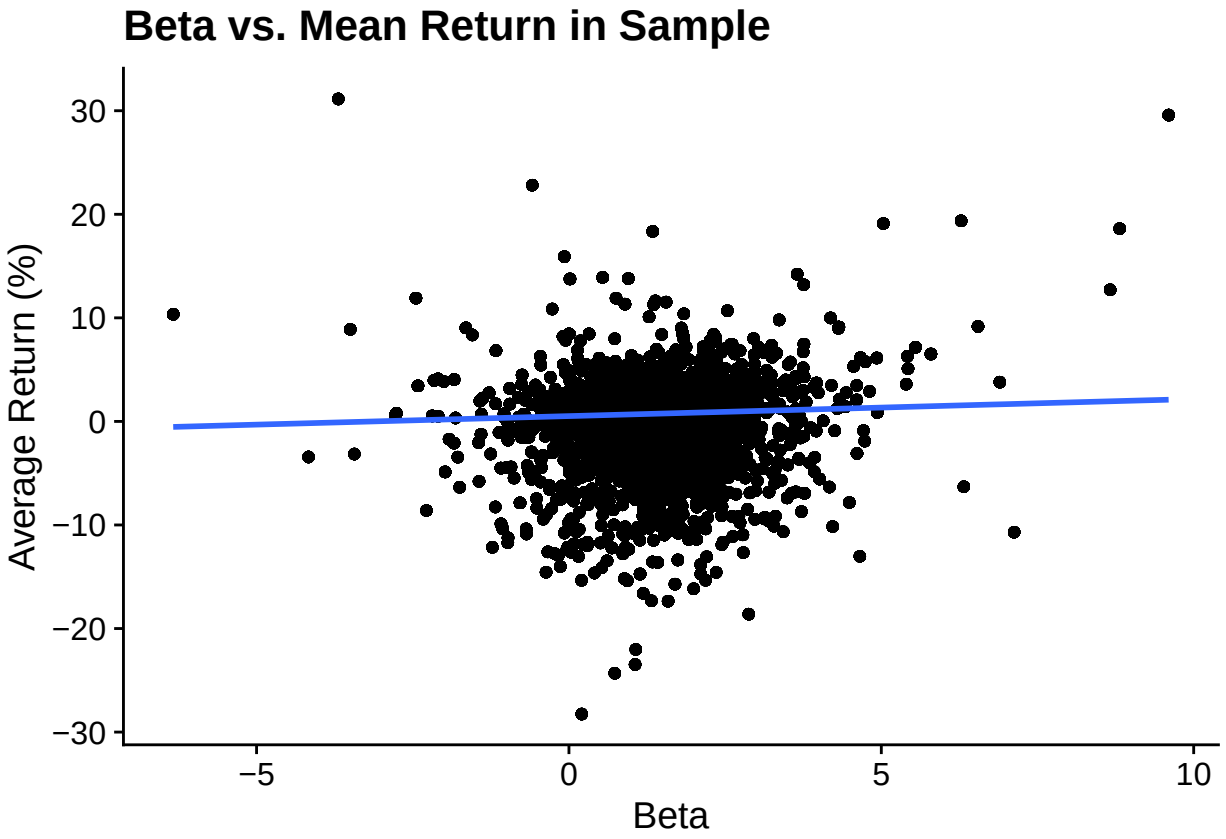
a) First I'll run the time series regression for each firm to get its market β. There's way to many to show so see the code appendix.

b) Now I'll calculate the cross sectional regression. The results are shown in table 1 below.

Table 1:

Term	Estimate	T-Stat	R-Squared
(Intercept)	0.5137	73.27	0.0025
beta	0.1649	32.68	0.0025

c) Below is the scatter-plot of the cross sectional regression for the unconditional CAPM. The market premium estimated from the cross sectional regression is 0.1649. The time series average of the RMRF factor is 0.97. These are not close in any way at all. We can probably explain this a bit by noticing the R^2 is almost 0. There's a lot of variation the cross sectional regression is not explaining about the return.



Question 2

a) These are the same values calculated above.

b) See the code for forming the portfolios.

c) In this question, I calculate the β's for each portfolio in a time series regression. Since there are only 20 portfolios I can actually show them this time. They are in table 2 below.

d) In this question, I calculate the market premium from the cross sectional regression. It is 0.0957 compared to 0.92 as the time average. Similar to question 1, they are very far apart. The market premium from the cross sectional regression is broadly similar to what we calculated in question 1. It is a bit smaller, but in the same ballpark.

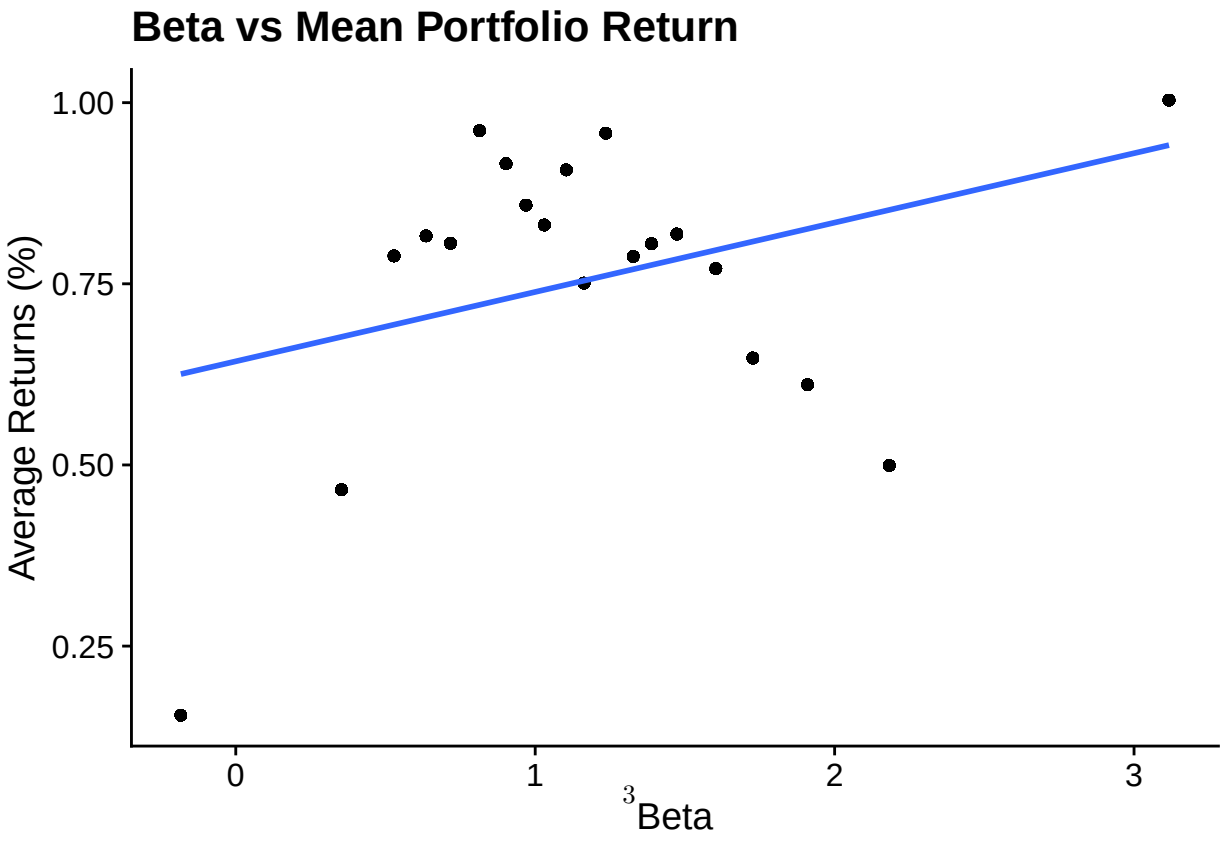
e) Here is the scatter plot of the cross sectional regression using portfolios rather than individual stocks. Table 3 shows the relevant statistics.

Table 2:

Portfolio	Beta
1	-0.1834
2	0.3531
3	0.5286
4	0.6357
5	0.7172
6	0.8143
7	0.9026
8	0.9692
9	1.0307
10	1.1041
11	1.1628
12	1.2357
13	1.3271
14	1.3886
15	1.4728
16	1.6034
17	1.7267
18	1.9097
19	2.1823
20	3.1168

Table 3:

Term	Estimate	T-Stat	R-Squared
(Intercept)	0.6430	85.66	0.0025
beta	0.0957	17.69	0.0025



f) In this question I will run the GRS test on the 20 portfolios. The GRS test produces a test statistic of 14.4574 which has associated p-value of 0. We strongly reject the null hypothesis that the α's are jointly 0.

Question 3

In this question, I will show that industry portfolio β's vary widely as Fama French showed.

a) See the code appendix for my calculation of the monthly β for each industry by month.

b) Below is a table of the standard errors on the changes in the monthly β's for each industry. The value is calculated as below. The standard errors are given in table 4.

$$\sigma^2_{TimeSeries}(\hat{\beta}) = \sigma^2_{True}(\beta) + \sigma^2_{MeanStandardError}(\hat{\beta})$$

Table 4:

Industry	Time Series SD
Autos	0.3827
Beer	0.4332
Books	0.3889
BusEq	0.3459
Carry	0.4125
Chems	0.2663
Clths	0.3473
Cnstr	0.3224
Coal	0.6889
ElcEq	0.3483
FabPr	0.2959
Fin	0.2771
Food	0.2107
Games	0.4291
Hlth	0.2801
Hshld	0.2795
Meals	0.4055
Mines	0.6048
Oil	0.4541
Other	0.3231
Paper	0.2721
Rtail	0.2388
Servs	0.5207
Smoke	0.4075
Steel	0.4566
Telcm	0.2787
Trans	0.3742
Txtls	0.3748
Util	0.3709
Whlsl	0.5044

Question 4

In this question, I will compare the conditional and unconditional CAPM on the industry portfolios.

a) Table 5 shows the α's and β's for each industry across the entire time series estimated both unconditionally and unconditionally. The unconditional mean absolute value of the α's is 0.0355, while the conditional mean is 0.0386. They are basically identical.

Table 5:

Industry	Alpha	Beta
Autos	0.0262	1.1988
Beer	0.0365	0.8473
Books	0.0300	0.8290
BusEq	0.0369	1.0448
Carry	0.0339	1.0800
Chems	0.0303	0.9980
Clths	0.0391	0.7562
Cnstr	0.0319	1.0032
Coal	0.0456	1.0885
ElcEq	0.0311	1.1793
FabPr	0.0307	1.0847
Fin	0.0348	0.9483
Food	0.0368	0.7040
Games	0.0312	1.1324
Hlth	0.0374	0.8089
Hshld	0.0313	0.8353
Meals	0.0389	0.8063
Mines	0.0471	0.7990
Oil	0.0372	0.9781
Other	0.0336	0.7974
Paper	0.0332	0.8798
Rtail	0.0347	0.8784
Servs	0.0632	0.7954
Smoke	0.0446	0.6245
Steel	0.0223	1.2190
Telcm	0.0256	0.7998
Trans	0.0366	1.0456
Txtls	0.0292	0.8492
Util	0.0250	0.7973
Whlsl	0.0489	0.8502

c) Table 6 shows the conditional and unconditional alphas for each industry along with their standard errors. The magnitudes of the conditional and unconditional α's are similar but the conditional ones have much bigger standard errors.

Table 6:

Industry	Alpha	SE	Conditional Alpha	Conditional SE
Autos	0.0262	0.0040	0.0194	0.1890
Beer	0.0365	0.0051	0.0455	0.1886
Books	0.0300	0.0046	0.0333	0.1906
BusEq	0.0369	0.0034	0.0409	0.1605
Carry	0.0339	0.0040	0.0390	0.1915
Chems	0.0303	0.0030	0.0306	0.1228
Clths	0.0391	0.0042	0.0432	0.1918
Cnstr	0.0319	0.0032	0.0325	0.1520
Coal	0.0456	0.0088	0.0490	0.3980
ElcEq	0.0311	0.0037	0.0315	0.1497
FabPr	0.0307	0.0030	0.0332	0.1423
Fin	0.0348	0.0030	0.0397	0.1331
Food	0.0368	0.0025	0.0428	0.1052
Games	0.0312	0.0049	0.0432	0.2171
Hlth	0.0374	0.0033	0.0438	0.1453
Hshld	0.0313	0.0035	0.0355	0.1300
Meals	0.0389	0.0045	0.0493	0.1783
Mines	0.0471	0.0060	0.0475	0.2735
Oil	0.0372	0.0046	0.0374	0.2328
Other	0.0336	0.0044	0.0416	0.1706
Paper	0.0332	0.0034	0.0328	0.1378
Rtail	0.0347	0.0029	0.0400	0.1448
Servs	0.0632	0.0095	0.0690	0.2669
Smoke	0.0446	0.0048	0.0450	0.2131
Steel	0.0223	0.0042	0.0158	0.2022
Telcm	0.0256	0.0032	0.0275	0.1329
Trans	0.0366	0.0035	0.0363	0.1753
Txtls	0.0292	0.0045	0.0308	0.2169
Util	0.0250	0.0034	0.0315	0.1568
Whlsl	0.0489	0.0065	0.0495	0.2190

Code Appendix

Erik Andersen

2024-05-07

```
# 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.
```

```
# 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], s
```

```r
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[3
             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')
```