Code Appendix

Erik Andersen

2024-05-27

```
here::i_am("HW5/code/HW5.R")

# Load packages
pacman::p_load(tidyverse, ggplot2, magrittr)


########### Load data ####################

portfolios_df = read_csv(here::here("HW5", "data", "25_Portfolios_5x5.CSV"),
                         skip = 15) |> # This is just trial and error to get rid of extra lines at the
  mutate(date = ym(`...1`)) |>
  # Rearrange and drop badly formatted date column
  select(date, everything(), -1) |>
  # The last row is just NAs so remove it
  filter(row_number() < 8723) |>
  # We only need after January 1963 and before December 2023
  filter(year(date) >= 1963 & year(date) <= 2023) |>
  # This data set has annualized returns and several other equal rated returns and several other sepeci
  filter(row_number() <= 732) |>
  # Clean the names
  janitor::clean_names()

ff_df = read_csv(here::here("HW5", "data", "F-F_Research_Data_Factors.CSV"),
                 skip = 3) |>
  # Same process
  mutate(date = ym(...1)) |>
  select(date, everything(), -1) |>
  # There are annual factors at the bottom we don't need
  filter(row_number() <= 1173) |>
  # Select same interval
  filter(year(date) >= 1963 & year(date) <= 2023) |>
  # Clean names
  janitor::clean_names()

# Merge data sets
returns_df = left_join(ff_df, portfolios_df)

# Create excess returns for each portfolio
returns_df %<>%
  # 6:length(returns_df) are the columns of interest
  # All the returns are characters for some reason so make them numeric
  mutate(across(6:length(returns_df), ~ as.numeric(.x)),
```

```r
         # Subtract the risk free rate from each of the return columns
         across(6:length(returns_df), ~ .x - rf))

# Extract the relevant names of the columns to loop over
name = names(returns_df)
# We only want the names of the different sortings, so select those
name = name[6:length(name)]

########## Question 1 ####################

# Calculate mean returns for each portfolio
mean_returns = map(name, function(x){

  # Calculate mean returns. Selecting the column this way gives a list data type so we have to unlist i
  return(mean(unlist(returns_df[,x])))
}) |>
  # Organize sensibly
  matrix(nrow = 5, byrow = T)

# Set the names
rownames(mean_returns) = c("Small","2","3","4","Big")
colnames(mean_returns) = c("Low B/M", "2", "3", "4", "High B/M")









########## Question 2 ####################

# Run the regression for each portfolio
coefficients_1f = map(name, function(x){

  # Run regression for each column
  estimates = lm(unlist(returns_df[,x]) ~ returns_df$mkt_rf) |>
    broom::tidy() |>
    select(estimate, statistic)


  return(estimates)
})

# Extract alphas
alpha = map_dbl(1:length(coefficients_1f), function(i){
  coefficients_1f[[i]] |>
    select(estimate) |>
    nth(1) |>
    unlist()
}) |>
  # Organize by portfolio
  matrix(nrow = 5, byrow = T)

# Set the names
```

```r
rownames(alpha) = c("Small","2","3","4","Big")
colnames(alpha) = c("Low B/M", "2", "3", "4", "High B/M")

# Extract alpha t-stats
alpha_t_stat = map_dbl(1:length(coefficients_1f), function(i){
  coefficients_1f[[i]] |>
    select(statistic) |>
    nth(1) |>
    unlist()
}) |>
  # Organize by portfolio
  matrix(nrow = 5, byrow = T)

# Set the names
rownames(alpha_t_stat) = c("Small","2","3","4","Big")
colnames(alpha_t_stat) = c("Low B/M", "2", "3", "4", "High B/M")


# Extract betas
beta = map_dbl(1:length(coefficients_1f), function(i){
  coefficients_1f[[i]] |>
    select(estimate) |>
    nth(2) |>
    unlist()
}) |>
  # Organize by portfolio
  matrix(nrow = 5, byrow = T)

# Set the names
rownames(beta) = c("Small","2","3","4","Big")
colnames(beta) = c("Low B/M", "2", "3", "4", "High B/M")

# Extract beta t-stats
beta_t_stat = map_dbl(1:length(coefficients_1f), function(i){
  coefficients_1f[[i]] |>
    select(statistic) |>
    nth(2) |>
    unlist()
}) |>
  # Organize by portfolio
  matrix(nrow = 5, byrow = T)

# Set the names
rownames(beta_t_stat) = c("Small","2","3","4","Big")
colnames(beta_t_stat) = c("Low B/M", "2", "3", "4", "High B/M")


# Rerun regression to get residuals
residuals_1f = sapply(name, function(x){

  # Run regression for each column
  res = lm(unlist(returns_df[,x]) ~ returns_df$mkt_rf)$residuals
```

```
  return(res)
}) |>
  # Transpose to get into right form
  t()

# Calculate GRS test
# Parameters
alphas = c(t(alpha))
N = nrow(residuals_1f)
t = ncol(residuals_1f)

# Covariance
cov_p = residuals_1f %*% t(residuals_1f)/ t

# Two bits of formula
first = (1 + mean(returns_df$mkt_rf)^2/var(returns_df$mkt_rf))^(-1)
second = as.numeric(t(alphas) %*% solve(cov_p) %*% alphas)

# GRS statistic
GRS_1f = (t - N - 1)/N * first * second

# P-value
GRS_1f_p_value = 1 - pf(GRS_1f, N, t-N-1)

# Critical values
f_crits = cbind(round(qf(1-0.1, N, t-N-1), 4),
                round(qf(1-0.05, N, t-N-1), 4),
                round(qf(1-0.01, N, t-N-1), 4))

colnames(f_crits) = c("10%", "5%", "1%")

# Chi-squared critical values
chi_crits = cbind(round(qchisq(1-0.1, N), 4),
                  round(qchisq(1-0.05, N), 4),
                  round(qchisq(1-0.01, N), 4))

colnames(chi_crits) = c("10%", "5%", "1%")

### Calculate mean absolute alphas
mean_abs_alpha = mean(abs(alpha))

### Plot expected vs actual returns

# Calculate the predicted returns
predicted_returns = map(name, function(x){
  reg = lm(unlist(returns_df[,x]) ~ returns_df$mkt_rf)

  predicted = predict(reg, returns_df[,x])

  return(predicted)
})

# Plot
```

```r
plot_1f = returns_df |>
  # Select only the portfolios
  select(name) |>
  # Pivot longer for plotting
  pivot_longer(cols = everything()) |>
  # The previous function doesn't group the portfolios together. This does. The weird factor thing lets
  arrange(factor(name, levels = names(returns_df)[6:length(returns_df)]), name) |>
  cbind(unlist(predicted_returns)) |>
  # Cbind gives stupid names so fix that
  rename("actual" = value,
         "predicted" = `unlist(predicted_returns)`) |>
  ggplot(aes(x = actual, y = predicted)) +
  geom_point(alpha = 0.3) +
  # 45 Degree line
  geom_abline(intercept = 0, slope = 1, size = 3, color = 'blue') +
  # Market excess returns
  geom_point(data = returns_df, aes(x = mkt_rf, y = mkt_rf), col = 'red', alpha = 0.5) +
  # Risk free rate
  annotate("point", x = 0, y = 0, col = 'green') +
  cowplot::theme_cowplot() +
  labs(x = "Actual Return", y = "Predicted Returns", title = "Expected vs Actual Returns for FF 25")

ggsave(here::here("HW5", "plots", "one_factor.pdf"))



# Cross sectional regression on only ff25
reg_cross = lm(as.numeric(mean_returns) ~ as.numeric(beta))

# Cross sectional regression on that plus the whole market and the risk free asset
reg_cross_rf = lm(c(as.numeric(mean_returns), mean(returns_df$mkt_rf), 0) ~ c(as.numeric(beta), 1, 0))


# make the table
ff25_cross = c(reg_cross$coefficients, broom::tidy(reg_cross)$std.error[2], mean(abs(reg_cross$residuals
ff25_extra_cross = c(reg_cross_rf$coefficients, broom::tidy(reg_cross_rf)$std.error[2], mean(abs(reg_cr

cross_table = rbind(ff25_cross, ff25_extra_cross)
colnames(cross_table) = c("Gamma", "Lambda", "SE Lambda", "Mean Absolute Alpha")
rownames(cross_table) = c("FF 25", "FF25 + RMRF + RF")

# Chi square test for both regressions
ff25_chi = chisq.test(abs(reg_cross$residuals))
ff25_extra_chi = chisq.test(abs(reg_cross_rf$residuals))


# Plot expected excess returns versus betas
plot_ff25 = tibble(excess_return = as.numeric(mean_returns),
       beta = as.numeric(beta)) |>
  ggplot(aes(x = beta, y = excess_return)) +
  geom_point() +
  geom_smooth(method = 'lm', se = F) +
  geom_abline(intercept = reg_cross_rf$coefficients[1], slope = reg_cross_rf$coefficients[2], col = 're
```

```
  cowplot::theme_cowplot() +
  labs(x = "Beta", y = "Expected Excess Returns", title = "FF 25 Beta vs Expected Excess Returns")

ggsave(here::here("HW5", "plots", "ff25.pdf"))



# Predicted versus actual returns for only ff25
plot_ff_predicted = tibble(actual = as.numeric(mean_returns),
       predicted = actual - reg_cross$residuals) |>
  ggplot(aes(x = actual, y = predicted)) +
  geom_point() +
  geom_abline(intercept = 0, slope = 1) +
  cowplot::theme_cowplot() +
  ylim(0,1) +
  xlim(0,1) +
  labs(x = "Actual Returns", y = "Predicted Returns", title =
         "FF25 Predicted vs Acutal Returns")
ggsave(here::here("HW5", "plots", "ff.pdf"))

# Predicted versus actual returns for whole sample
plot_all_predicted = tibble(actual = c(as.numeric(mean_returns), mean(returns_df$mkt_rf), 0),
       predicted = actual - reg_cross_rf$residuals) |>
  ggplot(aes(x = actual, y = predicted)) +
  geom_point() +
  geom_abline(intercept = 0, slope = 1) +
  cowplot::theme_cowplot() +
  ylim(0,1) +
  xlim(0,1) +
  labs(x = "Actual Returns", y = "Predicted Returns", title =
         "FF25 Predicted vs Acutal Returns With Market and RF Assets")
ggsave(here::here("HW5", "plots", "full_sample.pdf"))



########## Question 3 ####################




# Run the regression for each of the relevant columns
coefficients_3f = map(name, function(x){

  # Run regression for each column
  estimates = lm(unlist(returns_df[,x]) ~ returns_df$mkt_rf + returns_df$smb + returns_df$hml) |>
            broom::tidy() |>
            select(estimate)


  return(estimates)
})

# Get a vector for each different coefficient
```

```r
# Constant
a = map_dbl(1:length(coefficients_3f), function(i){
  coefficients_3f[[i]] |> nth(1) |> unlist()
}) |>
  # Organize
  matrix(nrow = 5, byrow = T)

# Set the names
rownames(a) = c("Small","2","3","4","Big")
colnames(a) = c("Low B/M", "2", "3", "4", "High B/M")


# Excess market risk
b = map_dbl(1:length(coefficients_3f), function(i){
  coefficients_3f[[i]] |> nth(2) |> unlist()
}) |>
  # Organize
  matrix(nrow = 5, byrow = T)

# Set the names
rownames(b) = c("Small","2","3","4","Big")
colnames(b) = c("Low B/M", "2", "3", "4", "High B/M")


# Small minus big
s = map_dbl(1:length(coefficients_3f), function(i){
  coefficients_3f[[i]] |> nth(3) |> unlist()
}) |>
  # Organize
  matrix(nrow = 5, byrow = T)

# Set the names
rownames(s) = c("Small","2","3","4","Big")
colnames(s) = c("Low B/M", "2", "3", "4", "High B/M")


# Value minus growth
h = map_dbl(1:length(coefficients_3f), function(i){
  coefficients_3f[[i]] |> nth(4) |> unlist()
}) |>
  # Organize
  matrix(nrow = 5, byrow = T)

# Set the names
rownames(h) = c("Small","2","3","4","Big")
colnames(h) = c("Low B/M", "2", "3", "4", "High B/M")

# Standard errors of alphas
a_se = map(name, function(x){

  # Run regression for each column
  estimates = lm(unlist(returns_df[,x]) ~ returns_df$mkt_rf + returns_df$smb + returns_df$hml) |>
    broom::tidy() |>
```

```
    select(statistic) |>
    nth(1) |> unlist()


  return(estimates)
}) |>
  matrix(nrow = 5, byrow = T)
rownames(a_se) = c("Small","2","3","4","Big")
colnames(a_se) = c("Low B/M", "2", "3", "4", "High B/M")

# Get R^2
R2 = map(name, function(x){

  # Run regression for each column
  estimates = lm(unlist(returns_df[,x]) ~ returns_df$mkt_rf + returns_df$smb + returns_df$hml) |>
    summary()


  return(estimates$r.squared)
}) |>
  matrix(nrow = 5, byrow = T)
rownames(R2) = c("Small","2","3","4","Big")
colnames(R2) = c("Low B/M", "2", "3", "4", "High B/M")



# Rerun regression to get residuals
residuals_3f = sapply(name, function(x){

  # Run regression for each column
  res = lm(unlist(returns_df[,x]) ~ returns_df$mkt_rf + returns_df$smb + returns_df$hml)$residuals


  return(res)
}) |>
  # Transpose to get into right form
  t()

# Calculate GRS test
# Parameters
alphas = c(t(a))
N = nrow(residuals_3f)
t = ncol(residuals_3f)

# Covariance
cov_p = residuals_3f %*% t(residuals_3f)/ t

# Risk factors
mean_rf = c(mean(returns_df$mkt_rf), mean(returns_df$smb), mean(returns_df$hml))
var_rf = var(returns_df[,2:4])

# Two bits of formula
first = (1 + t(mean_rf) %*% solve(var_rf) %*% mean_rf)^(-1)
```

```r
second = as.numeric(t(alphas) %*% solve(cov_p) %*% alphas)

# GRS statistic
GRS_3f = (t - N - 1)/N * first * second

# P-value
GRS_3f_p_value = 1 - pf(GRS_3f, N, t-N-1)


# Chi-square test
chi_3f = t*first*second
chi_3f_p = 1 - pchisq(t*first*second, N)

# MEan absolute alpha
mean_abs_alpha_3f = mean(abs(alphas))

# Calculate the predicted returns
predicted_returns_3f = map(name, function(x){
  reg = lm(unlist(returns_df[,x]) ~ returns_df$mkt_rf + returns_df$smb + returns_df$hml)

  predicted = predict(reg, returns_df[,x])

  return(predicted)
})

# Plot
plot_predicted_3f = returns_df |>
  # Select only the portfolios
  select(name) |>
  # Pivot longer for plotting
  pivot_longer(cols = everything()) |>
  # The previous function doesn't group the portfolios together. This does. The weird factor thing lets
  arrange(factor(name, levels = names(returns_df)[6:length(returns_df)]), name) |>
  cbind(unlist(predicted_returns_3f)) |>
  # Cbind gives stupid names so fix that
  rename("actual" = value,
         "predicted" = `unlist(predicted_returns_3f)`) |>
  ggplot(aes(x = actual, y = predicted)) +
  geom_point(alpha = 0.3) +
  # 45 Degree line
  geom_abline(intercept = 0, slope = 1, size = 3, color = 'blue') +
  # Market excess returns
  geom_point(data = returns_df, aes(x = mkt_rf, y = mkt_rf), col = 'red', alpha = 0.5) +
  # Risk free rate
  annotate("point", x = 0, y = 0, col = 'green') +
  cowplot::theme_cowplot() +
  labs(x = "Actual Return", y = "Predicted Returns", title = "Expected vs Actual Returns Using 3 Factors

ggsave(here::here("HW5", "plots", "expected_3f.pdf"))
```