Mandatory Assignment 2

Emil H. Erbas, Jeppe Vanderhaegen and Jacob G. Vestergaard

2024-04-19

Exercise 1

We read in the monthly CRSP data `crsp_monthly` and the monthly market excess returns `factors_ff3_monthly` from the `tidy_finance_*.sqlite` database. The code is self explanatory, and has been taken from Stefan's website.

Exercise 2

We generate the variable `mktcap_lag_12` which contains the firm's market capitalization 12 months before the measurement date.

We compute the momentum of stock i using the following formula, and assign the obtained values to the variable `Mom12`:

$$Mom_{i,t}^{12} = 100 \cdot \frac{mc_{i,t-1} - mc_{i,t-12}}{mc_{i,t-12}}$$

When comparing the differences between computing momentum based on relative changes in prices versus market capitalization, it is reasonable to expect different outcomes. Momentum calculated solely from price changes focuses only on fluctuations in stock prices. This approach primarily captures market trends and may be influenced by events like stock splits or dividends. However, when market capitalization is considered, the analysis also incorporates broader market dynamics. This includes corporate actions such as share issuance and buybacks, as well as overall changes in the company's equity structure. Therefore, using market capitalization offers a more comprehensive view, reflecting both price movements and changes in the firm's financial composition.

Please note that we have omitted all data observations prior to 1990-01-01 because beta values are only available from that point onward. This step is crucial for the upcoming exercise and prevents any potential merging issues arising from different dates among the variables. Additionally, it enables us to discard the initial 12 observations for each `permno`, ensuring there are no missing values for the lagged market capitalization.

Exercise 3

We calculate monthly portfolio breakpoints using `Mom12` as the sorting variable, organizing the data into deciles. Observations are assigned to deciles from 1 to 10, with higher `Mom12` values placed in higher deciles, relative to the corresponding month.

To calculate the equal-weighted average of `Mom12` and `mktcap` for each of the ten portfolios, we calculate the average `Mom12` and `mktcap` for each month, in each decile. These results are stored in the dataset `monthly_averages`, where one is able to observe the average values, grouped by `month` and `decile`. Using these values, we are able to average using `decile` as the grouping variable, yielding in the equal-weighted average values for `Mom12` and `mtkcap` for the ten portfolios. The results can be observed in the table below:

decile	Mom12_average	mktcap_average
1	-53.768575	834.7804
2	-29.048602	2625.5938
3	-16.268761	4017.5684
4	-6.654780	5206.4054
5	1.909558	5730.3414
6	10.624938	6322.0175
7	20.758165	6448.0642
8	34.576825	6510.0933
9	58.931935	5492.2351
10	210.460974	2768.6340

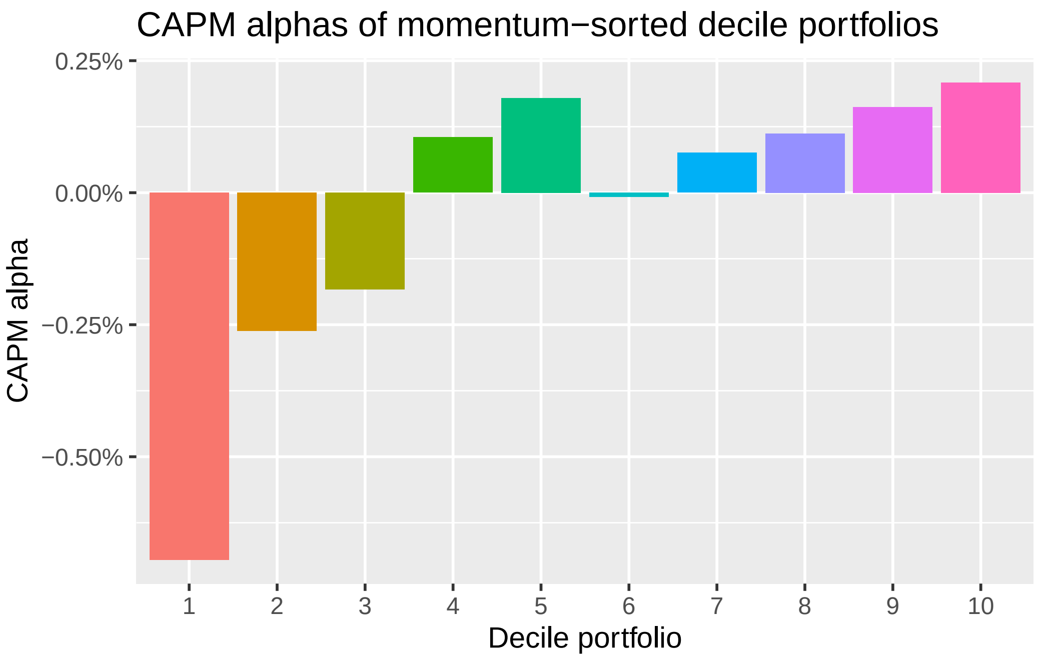
We proceed by calculating value-weighted monthly excess returns for the decile portfolios and storing the results in the dataset called `momenmtum_portfolios`. Using these results, we calculate the average excess return and CAPM alpha for the ten portfolios sorted by momentum. Results are showcased in the diagrams on the next page.

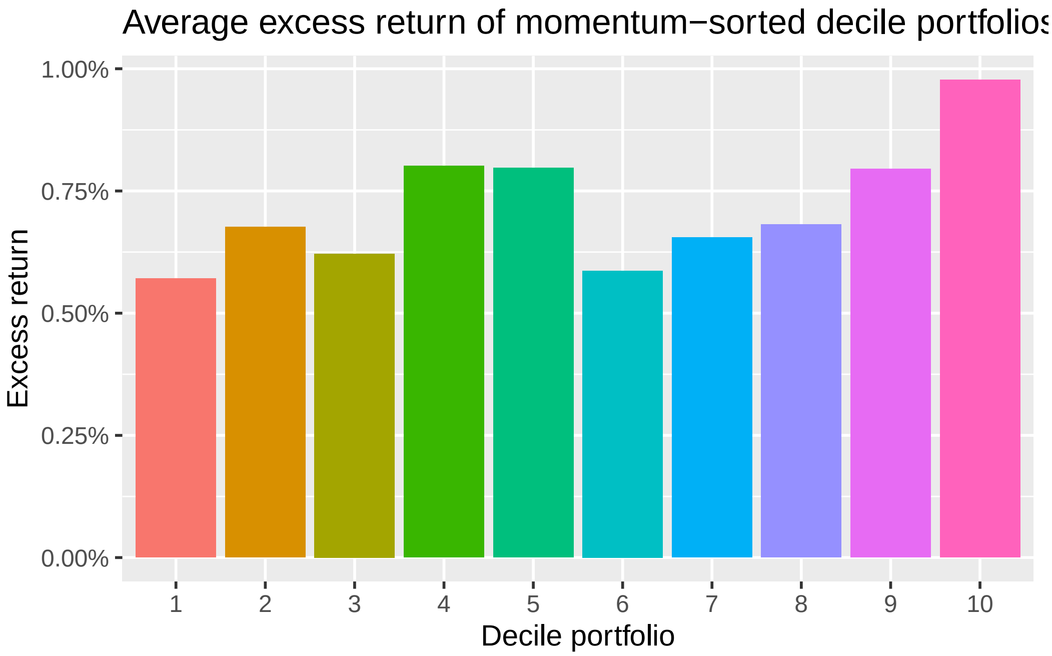
We now turn our analysis to the momentum strategy, which involves constructing a portfolio that goes long on past winners and short on past losers. We conduct a regression without a slope coefficient, e.g. $y_i = b + e_i$, on the long-short portfolio, where the intercept corresponds to the mean. Subsequently, we perform a hypothesis test on the intercept. Our results indicate, that the mean return of the long-short portfolio is not significantly different from zero. This implies that we cannot conclude that the performance of the strategy is significantly better. Results of the regression can be observed below. We then estimate $\hat{\alpha} = 0.009$ and $\hat{\beta} = -0.74$ for the long-short portfolio using a linear regression. We perform the following two hypothesis tests on these estimates:

$$H_0 : \beta = 0$$

$$H_0 : \alpha \leq 0$$

We are able to reject both null hypothesis. These results indicate, that β is different from zero, and alpha is significantly different from zero in the positive direction, which would imply that the portfolio has a performance that exceeds the expected return given its beta. Results of the regression can be observed below.


CAPM alphas of momentum−sorted decile portfolios


Average excess return of momentum−sorted decile portfolios

std.error	statistic	p.value	mean_average_return	hypothesis
0.0048019	0.8461138	0.3980015	0.0040629	Not Rejected

std.error	statistic	p.value	alpha	beta	hypothesis
0.0045638	1.981016	0.0482843	0.0090409	NA	Rejected
0.1014262	-7.299441	0.0000000	NA	-0.7403548	Rejected

Exercise 4

Ridge regression is particularly suited for dealing with multi collinearity in data, a common issue in financial datas ets like stock data. This regression method incorporates a penalty term, represented by λ, which helps to moderate the size of coefficients. By doing so, it effectively shrinks these coefficients to address potential correlations between independent variables.

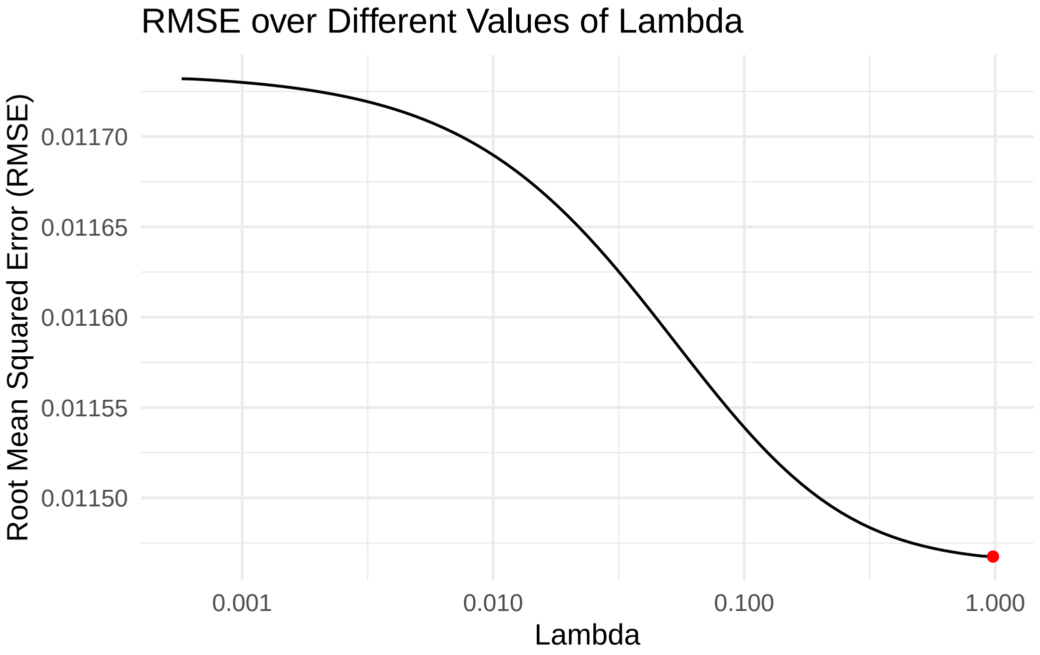
In our analysis, after selecting and preparing our data, we divided it into training and testing sets. The training set was further partitioned into 10 smaller subsets for the purposes of cross-validation, a process streamlined by the `cv.glmnet` function. This function performs ridge regression repeatedly, training the model 10 times, each time with one of the subsets withheld to serve as a validation set. This technique ensures that every subset is used as a validation set exactly once, providing a robust estimate of model performance.

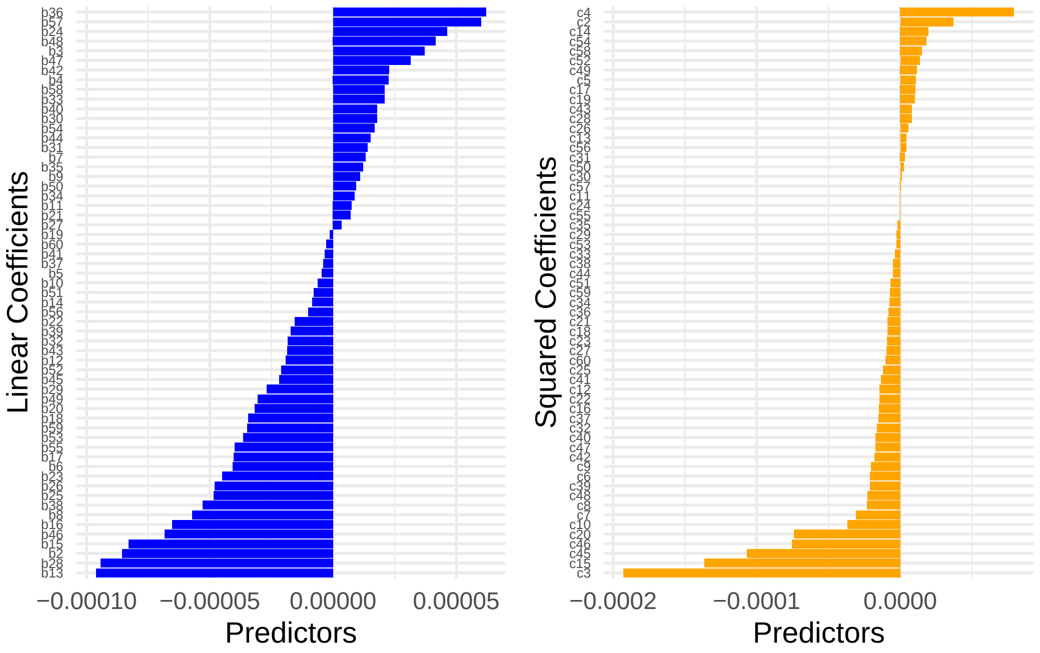
The value of λ that minimizes the average cross-validation error is identified as the optimal one, accessed via `min.lambda`. We then retrain the model with this optimal λ using the entire training set. To evaluate the performance of our model on unseen data, we use it to make predictions on the test set and compute the Root Mean Squared Error (RMSE). This metric offers a clear indication of the model's accuracy, helping us to gauge how effectively it predicts new data.

```
Call:  cv.glmnet(x = train_predictors_matrix, y = train_response_matrix,      alpha = 0, i

Measure: Mean-Squared Error

    Lambda Index Measure        SE Nonzero
min 0.9799     1 0.01147 0.0007544     118
1se 0.9799     1 0.01147 0.0007544     118
```





The optimal value for the lambda hyperparameter is found at the point on the RMSE graph where the error is minimized, indicated by a distinct red dot. This value is chosen as it yields the lowest RMSE, tailored to avoid overfitting while still capturing significant patterns in the data.

Furthermore, the bar plots exhibit the regression coefficients for our linear and squared predictors, with the b series representing linear and the c series the squared terms. The length and direction of each bar reflect the magnitude and the sign of the coefficient, respectively. By examining the coefficients for evidence of momentum or other predictability patterns, larger coefficients (positive or negative) signify a stronger relationship with future returns. For instance, positive coefficients associated with recent returns might could suggest a possible momentum effect, implying that stocks with robust returns in the recent past are likely to perform well shortly. Conversely, negative coefficients might hint at a mean-reversion tendency, indicating potential reversals in stock price movements. In practice should the b2-b12 coefficients be positive in order to determine a possible momentum effect, but since the coefficients are both positive and negative, we can not see a clear effect.

Should the short term coefficients be largely negative, one could bet against the momentum effect, this is called a mean-reversion strategy. For each individual stock, one could search for either momentum effects, thereby choosing either to bet against it or follow the possible momentum effect.