Predicting Returns using Bootstrapping and Multiple Linear Regression

Fall 2022

Andrew Finn

General Concept

The purpose of this model is to predict cumulative stock returns over the next month, but in theory this could be any period of time.

The processes of this model operate along the following general path. Given a certain security, historic returns are selected depending on how highly correlated they are to "recent"(target) returns. To be more specific, it checks if the two periods of returns being compared have a correlation coefficient higher than a given parameter, this parameter is denoted *cc*(up to this point that period has been three months). The data **following** returns that meet this criteria is selected to become a pool which N samples of size s are drawn from.

These samples are used to estimate the cumulative returns over the following month(denoted $\mathbb{E}[m_{(n-x)}]$) as well as the variance of the samples(denoted $\mathbb{V}[s_{(n-x)}]$). This is repeated numerous times, yielding a list of $\mathbb{E}[m_i]$, $\mathbb{V}[s_i]$, and actual returns for the estimated month.

A linear model is then created with the formula,

$lm(Actual\ Returns \sim \mathbb{E}[m_i] + \mathbb{V}[s_i])$

which will better understood later. The P-value and R-squared of the model are stored with the corresponding correlation coefficient(*cc*) that was used for data selection.

Everything listed up until this point is then repeated for a given list of correlation coefficients(generally .5, .6, .7, .8, .9), with the P-value and R-squared of each model saved with it's respective *cc*.

The best model is then selected, "best" being defined by the having the highest R-squared out of all models with a $p - value < .05$.

Data selection takes place again, using the *cc* that corresponds to the best model, with the target returns being the most recent monthly returns in real time. This leave us with individual values for $\mathbb{E}[m_{(n+1)}]$ and $\mathbb{V}[s_{(n+1)}]$, which are then imputed into the best model to make a prediction for $m_{(n+1)}$.

A Deeper Dive

The following is a detailed description of the actions of my program/model when given a single security.

GetData() Function

The model begins with the function GetData() which takes one argument "Ticker",

Monthly Log returns are pulled from Yahoo Finance via the *quantmod* package.

These monthly returns(*denoted* m_n) are assigned to a vector $returns = [m_1, m_2, ..., m_n]$, with length n. From the **beginning** of *returns*, monthly returns are removed so that n is divisible by 4.

The purpose of making n divisible by 4, is so that the contents of *returns* can be put into a matrix labeled *bstrap* with the following format.

$$bstrap = \begin{bmatrix} m_1 & m_2 & ... & m_{n-3} \\ m_2 & m_3 & ... & m_{n-2} \\ m_3 & m_4 & ... & m_{n-1} \\ m_4 & m_5 & ... & m_n \end{bmatrix}, \; ncol = number\ of\ columns\ in\ bstrap$$

This matrix provides the data our model will use to predict the returns of the future month, in the form of chronological "paths", which will be practical for comparison and estimation.

Select() Function

Next is the Select() function, which does the actual modeling/prediction.

Select() takes 4 arguments

$name = Ticker$

$s = sample\ size$

$N = number\ of\ samples$

$cc = lower\ limit\ of\ correlation\ coefficient$

The following details how Select(), and more specifically Slct() make a prediction, as it's rather logically complicated(though significantly slimmed down compared to earlier iterations).

When Select() is called, it forms the data frame *Expected* which has the following format

$\mathbb{E}[m_{(n-x)}]$	$\mathbb{V}[s_{(n-x)}]$	Actual Returns
$\mathbb{E}[m_n]$	$\mathbb{V}[s_n]$	m_n
$\mathbb{E}[m_{n-1}]$	$\mathbb{V}[s_{n-1}]$	m_{n-1}
...	...	...
$\mathbb{E}[m_{(s*2)}]$	$\mathbb{V}[s_{(s*2)}]$	$m_{(s*2)}$

The contents of this data frame will be explained more thoroughly as the operations of the function Select() are fleshed out, but it's important to know that it is created first, and is the only operation prior to calling the next function Slct().

Slct() Function

Slct() is a recursive function containing the argument x, and is nested inside of the function Select(). It does all the real work in the program. Slct() is called by Select() with the argument $x = 0$.

Looking at the matrix *bstrap*...

$$bstrap = \begin{bmatrix} m_1 & m_2 & ... & m_{n-3} \\ m_2 & m_3 & ... & m_{n-2} \\ m_3 & m_4 & ... & m_{n-1} \\ m_4 & m_5 & ... & m_n \end{bmatrix}$$

Estimations, and later predictions, are being made for each month in the fourth row i.e.

$$bstrap = \begin{bmatrix} m_1 & m_2 & ... & m_{n-3} \\ m_2 & m_3 & ... & m_{n-2} \\ m_3 & m_4 & ... & m_{n-1} \\ m_4 & m_5 & ... & m_n \end{bmatrix}$$

To do this, Slct() iterates through each column in *bstrap* using x, starting from the end, $(ncol - x)$ *where* $x = 0$, down until column $(2 * s)$, stopping there to preserve the size of the sample pool. The first iteration where $x = 0$ will be focused on in the following description for the sake of simplicity.

To be clear, the first iteration is looking to make an estimation and prediction for m_n, because $x = 0$.

The function creates a new data frame j with 2 columns. The first column contains correlation coefficients of the first three rows of each column, when compared to the first three rows of column $ncol$(the returns of the three months prior to m_n). The second column contains the corresponding column number.

For example

$j =$

ρ_i	i
ρ_1	1
ρ_2	2
...	...
$\rho_{(ncol-1)}$	$(ncol - 1)$

For the case where $x = 0$ & we're trying to predict m_n, each entry in the first column of j can be defined as.

$$\rho_i = \rho(\begin{bmatrix} m_i \\ m_{(i+1)} \\ m_{(i+2)} \end{bmatrix}, \begin{bmatrix} m_{(n-3)} \\ m_{(n-2)} \\ m_{(n-1)} \end{bmatrix}) \; where \; i = index \; of \; any \; column \; < ncol$$

Now the argument cc is used. Any row in j where $\rho_i < cc$, is removed from j. Now the second column of j contains the indexes of all columns in *bstrap* that met this criteria. This index will be denoted I_{cc}.

I_{cc} is used to take the returns in the **fourth** row of each column of *bstrap* that corresponds to I_{cc}, which are put into the vector $HcorR$. This could be notated as...

$HcorR = bstrap[4, I_{CC}]$

$HcorR$ is the pool then used to take N samples of size s(two arguments from Select()). The mean and variance of each sample are saved to respective vectors, of which the means are taken in order to calculate $\mathbb{E}[m_n]$ and $\mathbb{V}[s_n]$.

The entries for the first row of *Expected*, shown below, have now been completed and the first iteration of Slct() is finished

$\mathbb{E}[m_i]$	$\mathbb{V}[s_i]$	Actual Returns
$\mathbb{E}[m_n]$	$\mathbb{V}[s_n]$	m_n
...	...	...

This process would then repeat for $x = 1, 2, 3...$ up until $x = ncol - (s * 2)$, with each row of *Expected* being filled out with the parameters estimated for the $(ncol - x)^{th}$ column, as well as the actual return of the estimated month.

Multiple Linear Model

Once *Expected* has been completed with estimations and actual returns for relevant columns in bstrap, the following linear model is created.

$lm(Actual \; Returns \sim \mathbb{E}[m_i] + \mathbb{V}[s_i])$

The R^2 and $p-value$ of the model are saved with respect to the argument *cc*.

The entire process as described up to this point is repeated for

$cc \in seq(.45, .9, .05)$

For each *cc*, the R^2 and $p-value$ from the model created with regards to that *cc* is saved. Once the program has iterated through all $cc \in seq(.45, .9, .05)$, the model with the highest R^2 whose $p-value < .05$ is saved along with it's corresponding *cc*. If all models are insignificant, the program would stop here.

Final Prediction - Slct2() Function

In the Slct2() function, assuming that there was a model deemed valid in Select(), the previous bootstrapping process is repeated for *only* the most recent 3 months using the *cc* from model returned by Slct(), the most recent 3 months being...

$$\begin{bmatrix} m_{(n-2)} \\ m_{(n-1)} \\ m_{(n)} \end{bmatrix}$$

Which is different from the first iteration of Slct(), which was comparing

$$\begin{bmatrix} m_{(n-3)} \\ m_{(n-2)} \\ m_{(n-1)} \end{bmatrix}$$

to all columns prior as a form of model training.

After this is done, whats left is a single $\mathbb{E}[m_{(n+1)}]$ and $\mathbb{V}[s_{(n+1)}]$, with no $m_{(n+1)}$, as $\mathbb{E}[m_{(n+1)}]$ and $\mathbb{V}[s_{(n+1)}]$ are bootstrapped expectations for the coming(future) month. These two values are then put into the "Best Model" that was saved earlier, which yields our prediction for $m_{(n+1)}$.

Three different outputs are given when this is finished, but it only gives these outputs if the best model had a $p-value < .05$ with and $R^2 > .25$, to filter through mediocre ones. The first output is *Prediction* which looks like this...

Number of Residuals	R-squared	P-value	MSE	Prediction
value	R^2	P-val.	value	$m_{(n+1)}$
...	...	...	...	...

The second output is *Estimate* which looks like this...

$\mathbb{E}[m_{(n+1)}]$	$\mathbb{V}[s_{(n+1)}]$	CC	length(HcorR)
$\mathbb{E}[m_{(n+1)}]$	$\mathbb{V}[s_{(n+1)}]$	cc	value
...	...	...	...

The third output is the actual model that was used.

Prediction, *Estimate*, and the model are put into a list whose name corresponds to the Ticker.

Prediction() Function

This is the function that calls all of the functions listed prior, as well as defaults for potential errors. Prediction() takes two arguments,a vector of tickers to run through the model, and a list of $cc's$ to be used by Select().

Nested inside Prediction() is the recursive function cycl().Assuming there is more than one ticker in the vector given to Prediction(), cycle will recursively run all of the tickers through the model, returning the corresponding list with *Prediction*, *Estimate*, and the best model. If no valid models were found or there was some other error, an error is printed and it continues through the rest of the tickers.

Temporary Conclusion

After running the model a fair bit, it seems like it creates "no significant models" a large majority of the securities that it's given, and if they are significant their R^2 is negligible.While this is unfortunate, it has come up with models for securities that have $R^2 > .7$ and are significant at $\alpha = .05$. These promising outcomes are likely just luck after testing so many securities with numerous $cc's$, but they are rather intriguing.

I am now working on storing and analyzing the actual statistics from this model, to see if it does do anything consistently or if there are any trends in the parameters being used such as cc, s, N, terms of the model, or the period time being compared when selecting data.

I chose to end this write-up here as the project is ongoing and this write-up must conclude. While this isn't a formal conclusion, as I don't have one yet, this serves as an explanation of the processes I have created so far.

I plan to update this.