

2.4 Applying Factor Models in Pairs Trading

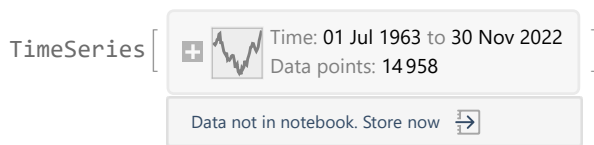
Pairs Trading With the Fama-French 5 Factor Model

In this chapter we will explore the use of factor models in pairs trading, beginning with the Fama-French Factor model.

We start by gathering the factor data:

```
In[ ]:= tsFFFactorsDaily =  
Fama-French 5-Factor Model FACTOR MODELS ["Data"] ["Daily"] ["PathComponent", Range[5]]
```

Out[]:=



The Fama-French model is fully described in the chapter on Factor Models. As a reminder, the five factors are as follows:

```
In[ ]:= tsFFFactorsDaily["MetaInformation"]  
Out[ ]:= {ComponentNames -> {Mkt-RF, SMB, HML, RMW, CMA}}
```

```
In[ ]:= Fama-French 5-Factor Model FACTOR MODELS ["Names"] // Dataset
```

Out[]:=

Mkt-RF	Excess Return on the Market
SMB	Small Minus Big
HML	High Minus Low
RMW	Robust Minus Weak
CMA	Conservative Minus Aggressive

The Rationale for a Factor Model

The main objective in pairs trading, and in statistical arbitrage in general, is to construct a portfolio that is market-neutral and has returns that follow a stable process. In the previous chapter, we used

the log-returns series of the PEP and KO stocks to create a combined portfolio, applying a Kalman filter to estimate the dynamic relationship between the two returns series. By taking weighted long and short positions in the two stocks, as determined by the β_t coefficient estimated in the Kalman model, we were able to eliminate market risk and achieve a returns process that is close to being stationary.

However, there is still the question of other risk factors such as size or value. While choosing a pair of closely-related stocks from the same industry may mitigate most of the factor exposure in the combined portfolio, it is also important to note that the two stocks will have different factor loadings and therefore some residual factor risk will remain in the combined portfolio.


The aim, here, therefore, is to eliminate as much factor exposure as possible by first fitting a factor model to each returns process and then using the residuals from those factor models to construct the trading signals.

Fitting the Fama - French 5 Factor Model


The FactorModel function applies the specified factor model to a specified time series of returns and produces an association containing the fitted model and the model residuals, as follows:


```
In[ ]:= FFmodels = <| # → FactorModel[tsReturnsSeries[#], tsFFFactorsDaily] |> & /@ symbols //
Association
```

Out[]:=


```
<| PEP → <| Time Series → TimeSeries [  Time: 05 Jul 2017 to 30 Nov 2022
Data points: 1363 ] ,

Model → FittedModel [ 0.627293 F1 - 0.299316 F2 - <<19>> F3 + 0.370647 F4 + 0.476138 F5 ] ,

Residuals → TimeSeries [  Time: 05 Jul 2017 to 30 Nov 2022
Data points: 1363 ] |> ,

KO → <| Time Series → TimeSeries [  Time: 05 Jul 2017 to 30 Nov 2022
Data points: 1363 ] ,

Model → FittedModel [ 0.573189 F1 - 0.247431 F2 + <<20>> F3 + 0.290038 F4 + 0.352566 F5 ] ,

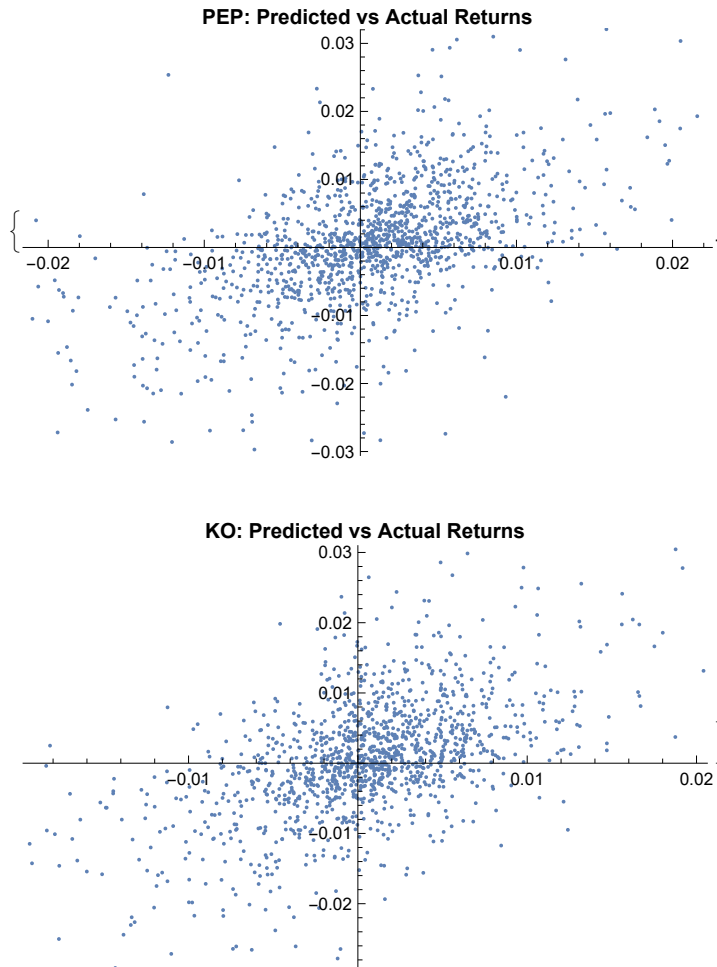
Residuals → TimeSeries [  Time: 05 Jul 2017 to 30 Nov 2022
Data points: 1363 ] |> |>
```

We can obtain the r-squared and scatterplots for each of the fitted factor models, which indicate that the Fama-French factors explain more than 40% of the variation in returns, in either stock.

```
In[ ]:= <| # → FFmodels [# , "Model"] ["RSquared"] |> & /@ symbols
Out[ ]:= { <| PEP → 0.453364 |> , <| KO → 0.420193 |> }
```

```
In[ ]:= ListPlot[Transpose@{FFmodels[#, "Model"] ["PredictedResponse"],
  FFmodels[#, "Time Series"] ["Values"]}, ImageSize → Medium,
  PlotLabel → Style[StringJoin[#, ": Predicted vs Actual Returns"], Bold]] & /@ symbols
```

```
Out[ ]:=
```



Comparing the two factors models, we can see that PEP has considerably greater exposure than KO to the market factor F_1 , the RMW factor F_4 , and also to the CMA factor F_5 . Also worth noting is that the loadings on the size factor F_3 are of opposite sign. These findings suggest that it will be difficult to neutralize every dimension of factor risk in the combined long-short pair portfolio simply by using a Kalman model, as we did in the previous chapter: the best we can hope to achieve with that straightforward approach is to eliminate most of the market exposure.

```
In[ ]:= <| # → Normal[FFmodels[#, "Model"]] |> & /@ symbols
```

```
Out[ ]:=
```

```
{ <| PEP → 0.627293 F1 - 0.299316 F2 - 0.127179 F3 + 0.370647 F4 + 0.476138 F5 |> ,
  <| KO → 0.573189 F1 - 0.247431 F2 + 0.0624475 F3 + 0.290038 F4 + 0.352566 F5 |> }
```



So, by first removing factor exposure from the two returns processes, the intention here is to mitigate as much as possible of the exposure of the combined pairs portfolio, not just to market risk, but to all five of the risk factors in the Fama-French model.

Generating Trading Signals

We pick up the residuals from the factor models and check them for stationarity:

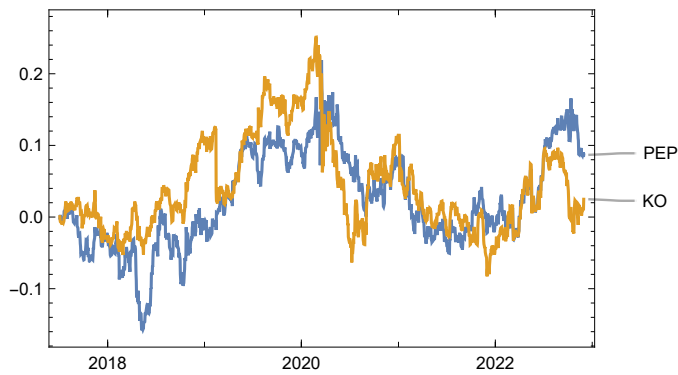
```
In[ ]:= tsResidualsSeries = <|# → FFmodels[#, "Residuals"] |> & /@ symbols // Association
```

```
Out[ ]:=
```

```
<| PEP → TimeSeries [  Time: 05 Jul 2017 to 30 Nov 2022  
Data points: 1363 ],  
KO → TimeSeries [  Time: 05 Jul 2017 to 30 Nov 2022  
Data points: 1363 ] |>
```

```
In[ ]:= DateListPlot[TimeSeries[Accumulate[tsResidualsSeries[#] ["Values"]],  
tsResidualsSeries[#] ["DateList"]] & /@ symbols, , PlotLabels → symbols]
```

```
Out[ ]:=
```



For both stocks, all variants of the unit root tests strongly reject the null hypothesis of a unit root in the residuals process:

```
In[ ]:= <|# → UnitRootTest[tsResidualsSeries[#] ["Values"], Automatic, All] |> & /@ symbols
```




```
Out[ ]:=
```

```
{ <| PEP → { 1.92219 × 10-18, 5.03542 × 10-21, 4.22255 × 10-18, 4.21158 × 10-21 } |> ,  
  <| KO → { 3.94559 × 10-18, 2.11893 × 10-20, 7.12979 × 10-18, 2.12818 × 10-20 } |> }
```

We next proceed to fit a Kalman Filter model, this time using the factor model residuals, rather than the raw returns. Note that in this case there is less time-variation in the β_t coefficient than in the previous Kalman model estimated on raw returns. This likely reflects the fact that we have successfully eliminated residual factor risk in the returns processes.

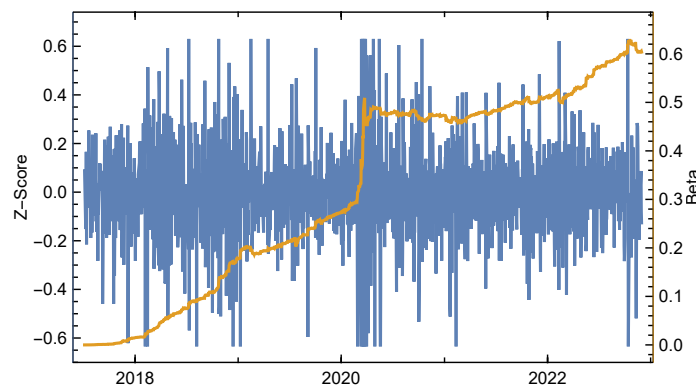
```
In[ ]:= {tsBeta, tsZscores, tsQ} = ReturnsTimeSeriesKalman[tsResidualsSeries]
```

```
Out[ ]:=
```

```
{TimeSeries [  Time: 06 Jul 2017 to 30 Nov 2022  
Data points: 1362 ],  
TimeSeries [  Time: 06 Jul 2017 to 30 Nov 2022  
Data points: 1362 ], TimeSeries [  Time: 06 Jul 2017 to 30 Nov 2022  
Data points: 1362 ] }
```

```
In[ ]:= ListLinePlot[{tsZscores, tsBeta["PathComponent", "Beta"]},
  MultiaxisArrangement -> All, PlotLabels -> {"\nZ-Score", "Beta"}]
```

Out[]:=

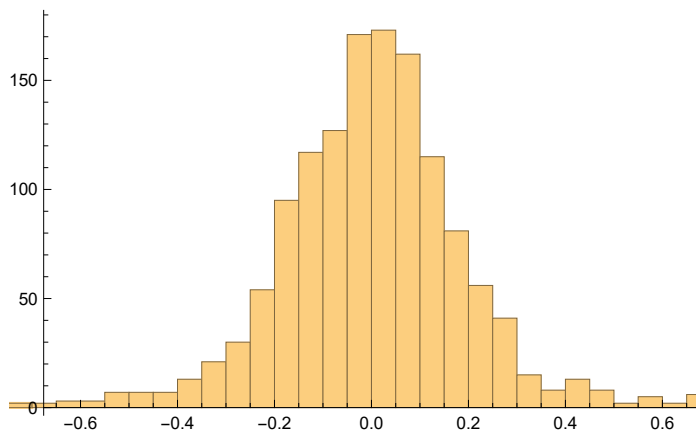


Distribution of the Kalman Model Zscores

Looking at the distribution of the Zscores, we once again are led to reject the assumption of Normality, as before. In fact, we determine that the Zscore distribution is quite well described by a Student-t distribution:

```
In[ ]:= Histogram[tsZscores]
```

Out[]:=



```
In[ ]:= H = DistributionFitTest[tsZscores, Automatic, "HypothesisTestData"];  
H["TestDataTable", All]
```

```
Out[ ]:=
```

	Statistic	P-Value
Anderson-Darling	25.2045	0.
Baringhaus-Henze	28.5568	1.98841×10^{-13}
Cramér-von Mises	3.87689	0.
Jarque-Bera ALM	12 562.2	0.
Kolmogorov-Smirnov	0.0829255	0.
Kuiper	0.16144	0.
Mardia Combined	12 562.2	0.
Mardia Kurtosis	111.433	0.
Mardia Skewness	0.0398311	0.841811
Pearson χ^2	204.097	1.4891×10^{-26}
Shapiro-Wilk	0.875728	1.2778×10^{-31}
Watson U ²	3.87441	0.

```
In[ ]:= d = FindDistribution[tsZscores["Values"]]
```

```
Out[ ]:=
```

```
StudentTDistribution[0.00748215, 0.1273, 2.55469]
```

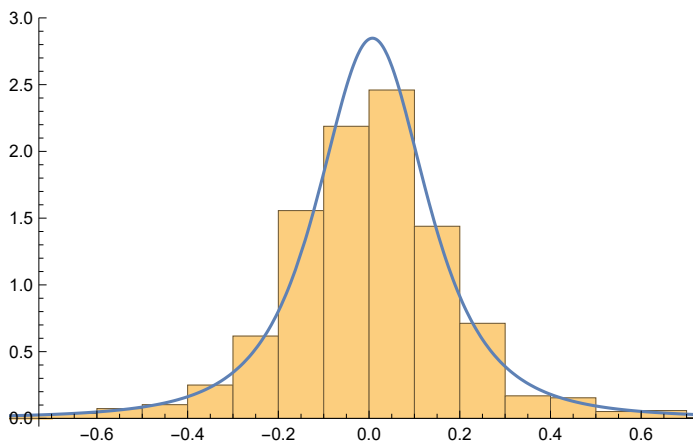
```
In[ ]:= H = DistributionFitTest[tsZscores, d, "HypothesisTestData"];  
H["TestDataTable", All]
```

```
Out[ ]:=
```

	Statistic	P-Value
Anderson-Darling	2.09413	0.081584
Cramér-von Mises	0.35563	0.0949307
Kolmogorov-Smirnov	0.0432583	0.0118627
Kuiper	0.0483575	0.0323241
Pearson χ^2	46.3877	0.0943853
Watson U ²	0.200091	0.0384934

```
In[ ]:= Show[{Histogram[tsZscores, 40, "PDF"], Plot[PDF[d, x], {x, -1, 1}]}]
```

```
Out[ ]:=
```




Signal Generation

One common method for generating trading signals from Zscores is to use an ARMA time series model or a continuous model such as an Ornstein-Uhlenbeck model. However, a significant challenge is that the Zscores do not conform to a Gaussian distribution. This means that the parameters estimated from

a Gaussian likelihood will be imprecise and any forecasts produced by these models will be unreliable. While there are ways to estimate ARMA models with a non-Gaussian residual process, this would require a deep dive into econometric modeling, which is beyond the scope of this discussion. Instead, I will take a more practical approach, using the empirical distribution of the cumulative Zscores.

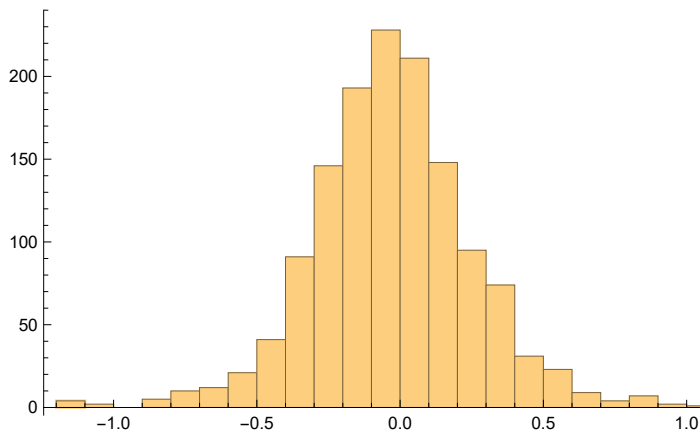
```
In[ ]:= tsCumulativeZscores = TimeSeries[Accumulate[tsZscores["Values"]], tsZscores["DateList"]]
```

```
Out[ ]:=
```

```
TimeSeries [  Time: 06 Jul 2017 to 30 Nov 2022  
Data points: 1362 ]
```

```
In[ ]:= Histogram[tsCumulativeZscores["Values"]]
```

```
Out[ ]:=
```



It appears that a suitable value for the levelSpacing parameter, the level at which entry signals are generated, would be around ± 0.7 , which in the 1.5%-tile tails of the empirical distribution:

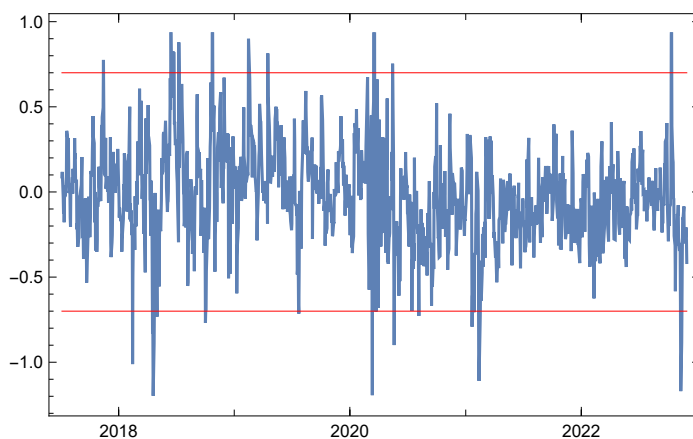
```
In[ ]:= DateListPlot[tsCumulativeZscores,
```

```
Epilog -> {Red, Line[{{tsCumulativeZscores["FirstDate"], 0.7},
```


```
{tsCumulativeZscores["LastDate"], 0.7}}], Line[{{tsCumulativeZscores["FirstDate"],
```

```
-0.7}, {tsCumulativeZscores["LastDate"], -0.7}}]}]
```

```
Out[ ]:=
```



```

In[ ]:= D = EmpiricalDistribution[tsCumulativeZscores]
Out[ ]:=
DataDistribution [  Type: Empirical
Data points: 1362 ]


In[ ]:= CDF[D, {-0.7, 0.7}]
Out[ ]:=
{0.0154185, 0.986784}

In[ ]:= levelSpacing = 0.7;

In[ ]:= tsSignals = ZscoreSignals[tsZscores, levelSpacing, maxTrades]
Out[ ]:=

```

```

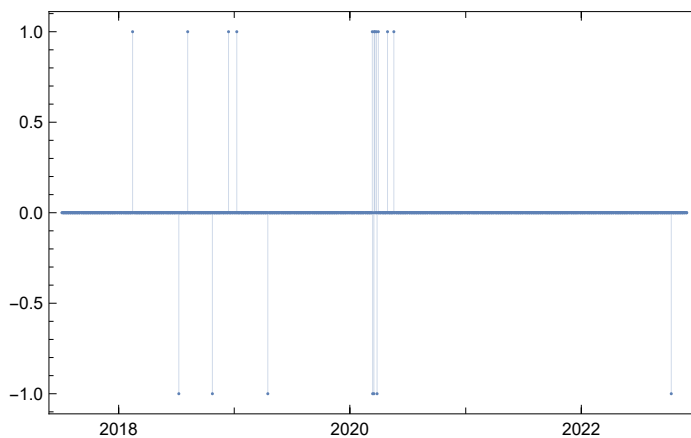
EventSeries [  Time: 06 Jul 2017 to 30 Nov 2022
Data points: 1362 ]

```

```

In[ ]:= DateListPlot[tsSignals, Joined → False, Filling → Axis]
Out[ ]:=

```



BackTesting the Kalman Factor Model

Now that we have generated a time series of entry signals, we can proceed to evaluate a pairs trading strategy based on those signals using the backtest system built into the Equities Entity Store.

As previously explained, the backtest system neither knows nor cares how the signal time series was generated: the fact that in this case it results from a factor modelling process makes no difference. We simply plug the signals into the backtest system and run it exactly as we did before. As this simply repeats the procedure that I described in depth in the earlier chapter, I am going to omit all the intermediate steps and associated explanations previously given, and instead combine the entire sequence, which takes less than 2.5 seconds to run:


```

In[ ]:= AbsoluteTiming[
  Multiplier = Apply[Divide, TimeSeriesWindow[tsPriceSeries[#] ["PathComponent", 4],
    {tsBeta["FirstDate"], tsBeta["LastDate"]}][["Values"] & /@ symbols];
  tsPortfolioUnits =
    TimeSeries[portfolioUnits * Transpose[{ConstantArray[1, tsBeta["PathLength"]],
      -Multiplier * tsBeta["PathComponent", 2][["Values"]]}, tsBeta["DateList"]];
  orderSpec = Association[{"Time Step" → timeStep,
    "Symbols" → symbols, "Order Type" → "MOO", "Prices" → tsPriceSeries,
    "Signals" → TimeSeriesWindow[tsSignals, {startDate, tsSignals["LastDate"]}],
    "Portfolio Units" → tsPortfolioUnits, "Max Trades" → maxTrades}];
  tsOrderList = GenerateOrderSpecifications[orderSpec];
  executionSpec = Association["Initial Investment" → tsInitialInvestment,
    "Orders" → tsOrderList, "Symbols" → symbols, "Point Value" → spreadPointValues];
  orderExecutions = GenerateExecutions[executionSpec];
  tradeSpec = Association["Executions" → orderExecutions["Executions"],
    "PositionQty" → orderExecutions["PositionQuantity"],
    "AveragePrice" → orderExecutions["AveragePrice"], "Symbols" → symbols,
    "Point Value" → spreadPointValues, "Slippage" → slippage, "Commission" → commission];
  tradePerformance = TradePerformance[tradeSpec];
  tsOpenPrices =
    Association[ Association[# → TimeSeriesWindow[tsPriceSeries[#] ["PathComponent", 1],
      {startDate, Today}]] & /@ symbols];
  positionSpec = Association["Initial Investment" → tsInitialInvestment, "Cash" →
    orderExecutions["Cash"], "PositionQty" → orderExecutions["PositionQuantity"],
    "ValuationPrices" → tsOpenPrices, "Slippage" → slippage, "Commission" → commission];
  portfolioUpdate = UpdatePortfolio[positionSpec];]

Out[ ]:=
{2.39345, Null}

```

The strategy based on factor modelling represents a clear improvement over the prior model in almost every performance category.

- The net profit has increased from \$39,356 to \$64,261
- The total return has increased from 38% to 64%
- CAGR has improved from 6.86% to 10.23%
- Share ratio has improved from 0.39 to 0.6
- Sortino ratio has improved from 0.59 to 0.86
- Drawdown has reduced from -30% to -20%
- Profit Factor has increased from 1.7 to 7
- Win/Loss ratio has risen from 1 to 7

Performance Summary

In[]:= **PortfolioSummaryReport[portfolioUpdate]**
 Out[]:=

Start Date	Fri 29 Dec 2017
End Date	Fri 30 Dec 2022
Investment Period	5. yr
Total Realized Profit	\$64 261.20
Total Unrealized Profit	\$-1 543.00
Total Net Profit	\$62 718.20
Transaction Costs	\$478.50
Total Return	62.72 %
CAGR	10.23 %
Annual Volatility	17.13 %
Annual Semi-Deviation	11.85 %
Sharpe Ratio	0.6
Sortino Ratio	0.86
Calmar Ratio	0.5
Max Drawdown Date	Wed 19 Feb 2020
Max Drawdown	\$-19 942.85
Max Drawdown Pct	-20.39 %

```
In[ ]:= Dataset@tradePerformance ["Trade Stats"]
Out[ ]:=
```

Realized Gross Profit	74886.5
Realized Gross Loss	-10625.3
Realized Net Profit	\$64261.20
Profit Factor	7.05
No. Trades	12
Avg. Trade	\$5355.10
Avg. Trade Duration	154.6 days
No. Winners	6
No. Losers	6
Win Rate	50. %
Av. Winning Trade	\$12481.08
Av. Losing Trade	\$-1770.88
Win/Loss Ratio	7.05
Slippage	\$319.00
Commission	\$159.50

```
In[ ]:= DateListPlot[portfolioUpdate["DailyPortfolioEquity"], PlotLabel -> Style["PEP-KO", Bold],
  Filling -> Axis, ImageSize -> Large, GridLines -> Automatic]
```

```
Out[ ]:=
```



Conclusion

Our research has revealed that over 40% of the fluctuations in the stock returns for the PEP-KO pair can be explained by risk factors outlined in the Fama-French 5 factor model. By eliminating these factors from the returns process, we are able to construct a more stable pairs portfolio and clearer trading signals. This leads to a significant enhancement in strategy performance, with an increase of around 50% in total profits, accompanied by a 1/3 reduction in strategy drawdown.