Scenario Analysis of Risk Parity using RcppParallel

Jason Foster
CRAN - Package ‘roll’

May 20, 2017
The **roll** package provides parallel functions for computing rolling statistics of time-series data; it uses Rcpp, RcppArmadillo, and RcppParallel:

```r
install.packages("roll")
```

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>roll_sum</td>
<td>Sums</td>
<td>roll_cor</td>
<td>Correlations</td>
</tr>
<tr>
<td>roll_prod</td>
<td>Products</td>
<td>roll_scale</td>
<td>Z-scores</td>
</tr>
<tr>
<td>roll_mean</td>
<td>Means</td>
<td>roll_lm</td>
<td>Linear models</td>
</tr>
<tr>
<td>roll_var</td>
<td>Variances</td>
<td>roll_eigen</td>
<td>Eigendecomps</td>
</tr>
<tr>
<td>roll_cov</td>
<td>Covariances</td>
<td>roll_pcr</td>
<td>PCA regressions</td>
</tr>
<tr>
<td>roll_sd</td>
<td>Standard deviations</td>
<td>roll_vif</td>
<td>Multicollinearity</td>
</tr>
</tbody>
</table>

Jason Foster

CRAN - Package 'roll'
Rolling beta for 2,000 portfolios!

Speed gains from parallel computing help decrease the trade-off between number of observations and window size:

Less than 1-minute!
Download the S&P 500 index using `quantmod` and calculate returns:

```r
library(quantmod)
getSymbols("^GSPC")
```

Then use `RcppArmadillo` and estimate a portfolio’s sensitivity to the S&P 500 index by rolling the `fastLmPure` function:

```r
library(RcppArmadillo)
fastLm_coef <- function(data) {
  return(coef(fastLmPure(cbind(1, data$x), data$y)))
}

arma_lm_coef <- rollapplyr(data = returns, width = 252,
FUN = fastLm_coef,
by.column = FALSE)
```
Repeat the exercise and take advantage of parallel processing in C++ by using the `roll_lm` function in the `roll` package:

```r
library(roll)
roll_lm_coef <- roll_lm(x = returns$x,
                          y = returns$y,
                          width = 252)$coefficients

And test that the coefficients from the `fastLmPure` and `roll_lm` functions are equal:

```r
all.equal(arma_lm_coef, roll_lm_coef)
```

```r
## [1] TRUE
```
RcppParallel, developed by the Rcpp Core team, makes it easy to create safe, portable, high-performance parallel algorithms using C++ and Rcpp:

```r
install.packages("RcppParallel")
```

In particular, RcppParallel provides two high-level parallel algorithms:

- `parallelFor()`: convert the work of a standard serial “for” loop into a parallel one
- `parallelReduce()`: compute and aggregate multiple values in parallel

For more information, visit the RcppParallel website: [http://rcppcore.github.io/RcppParallel/](http://rcppcore.github.io/RcppParallel/)
Style analysis of multi-asset portfolios

Download daily returns for 150 portfolios in the Tactical Allocation and World Allocation categories and apply a 15 asset class factor model on a daily rolling window:

<table>
<thead>
<tr>
<th>Stocks</th>
<th>Bonds &amp; other</th>
</tr>
</thead>
<tbody>
<tr>
<td>US Large Cap Value</td>
<td>US Gov 1-10Y</td>
</tr>
<tr>
<td>US Large Cap Growth</td>
<td>US Gov 10Y+</td>
</tr>
<tr>
<td>US Mid Cap</td>
<td>Non-US Gov</td>
</tr>
<tr>
<td>US Small Cap</td>
<td>IG Corp</td>
</tr>
<tr>
<td>Europe</td>
<td>HY Corp</td>
</tr>
<tr>
<td>Japan</td>
<td>MBS</td>
</tr>
<tr>
<td>Pacific ex Japan</td>
<td>Commodities</td>
</tr>
<tr>
<td>Emerging Markets</td>
<td></td>
</tr>
</tbody>
</table>

Asset allocation

Daily returns are readily available and help make inferences about style:

25x over RcppArmadillo version

Median portfolio

Speed test uses univariate betas

Jason Foster
CRAN - Package 'roll'
Concentrated in equity risk

A significant driver of return variability is the concentration of equity risk:

44x over R version!

Median portfolio

<table>
<thead>
<tr>
<th>Risk contribution (%)</th>
<th>2013</th>
<th>2014</th>
<th>2015</th>
<th>2016</th>
<th>2017</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stocks</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bonds &amp; other</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Median portfolio

<table>
<thead>
<tr>
<th>Risk contribution (%)</th>
<th>2013</th>
<th>2014</th>
<th>2015</th>
<th>2016</th>
<th>2017</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stocks</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bonds &amp; other</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Decline in realized volatility

From mixed economic data to rising geopolitical risk, volatility across markets and asset classes is near an all-time... low?

93x over R version!

Asset classes

Volatility (%)

Stocks

Bonds & other

Number of simulations

Seconds

Volatility (%)

2013 2014 2015 2016 2017

0 9 18 27 36

0 0.8 secs

0 250 500 750 1,000

0 78.8 secs

sd

roll_sd
Consider a portfolio of \( N \) assets and let \( \mathbf{w} = (w_1, \ldots, w_N) \) be a vector of asset weights. Also define \( \Sigma \) to be the covariance matrix of returns. Then the volatility of the portfolio is defined as:

\[
\sigma = \sqrt{\mathbf{w}^T \Sigma \mathbf{w}}
\]

and, by Euler’s theorem, the risk contribution of asset \( i \) is given by:

\[
\sigma_i = w_i \frac{\partial \sigma}{\partial w_i}
\]

**Goal**: solve for weights such that each asset contributes equal risk

*However, in practice, it is possible for managers to have discretion over the timing of investment decisions!*
Scenario analysis

Compute rolling statistics using the roll package and feed into a simple risk parity strategy:

- **Instruments**: 15 asset classes
- **Risk settings**: long-term; exponential decay
- **Approach**: passive; equal risk contributions
- **Volatility target**: 10% volatility level
- **Rebalancing**: monthly; systematic

Now, against the backdrop of the current environment of low realized volatility, consider a few generic scenarios:

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Volatility</th>
<th>Correlation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scenario 1</td>
<td>Average</td>
<td>Average</td>
</tr>
<tr>
<td>Scenario 2</td>
<td>Low</td>
<td>Low</td>
</tr>
<tr>
<td>Scenario 3</td>
<td>High</td>
<td>High</td>
</tr>
</tbody>
</table>

Jason Foster
A toy example: return to average

Simulate scenarios and mechanics of the simple risk parity strategy:

81x over R version!

Scenario 1

Volatility Correlation Total

2014 2015 2016 2017 2018

Leverage

2.6
2.4
2.2
2.0
1.8

Volatility Correlation Total

Number of simulations

Seconds

83.4 secs
1 secs

0 250 500 750 1,000

R
RcppParallel

0 24 48 71 95

Jason Foster

CRAN - Package 'roll'
Explore the scenarios

Parallel computing via RcppParallel helps to dig deeper and faster:

**Scenario 2**

<table>
<thead>
<tr>
<th>Year</th>
<th>Leverage</th>
<th>Volatility</th>
<th>Correlation</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>2016-07</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2017-01</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2017-07</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2018-01</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Scenario 3**

<table>
<thead>
<tr>
<th>Year</th>
<th>Leverage</th>
<th>Volatility</th>
<th>Correlation</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>2016-07</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2017-01</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2017-07</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2018-01</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Get the released version from CRAN:

```r
install.packages("roll")
```

Or the development version from GitHub:

```r
# install.packages("devtools")
devtools::install_github("jjf234/roll")
```