Scenario Analysis of Risk Parity using RcppParallel

Jason Foster
CRAN - Package ‘roll’

May 20, 2017

roll: rolling statistics

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

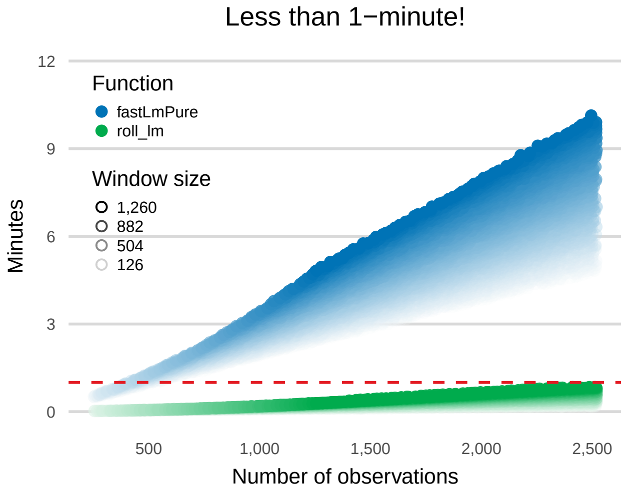
```r
install.packages("roll")
```

Function	Description
roll_sum	Sums
roll_prod	Products
roll_mean	Means
roll_var	Variances
roll_cov	Covariances
roll_sd	Standard deviations

Function	Description
roll_cor	Correlations
roll_scale	Z-scores
roll_lm	Linear models
roll_eigen	Eigendecomps
roll_pcr	PCA regressions
roll_vif	Multicollinearity

Rolling beta for 2,000 portfolios!

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



RcppArmadillo version

Download the S&P 500 index using **quantmod** and calculate returns:

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

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

```
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)
```

RcppParallel version

Repeat the exercise and take advantage of parallel processing in C++ by using the **roll_lm** function in the **roll** package:

```
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:

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

```
## [1] TRUE
```

RcppParallel: parallel programming tools for ‘Rcpp’

`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/

Style analysis of multi-asset portfolios

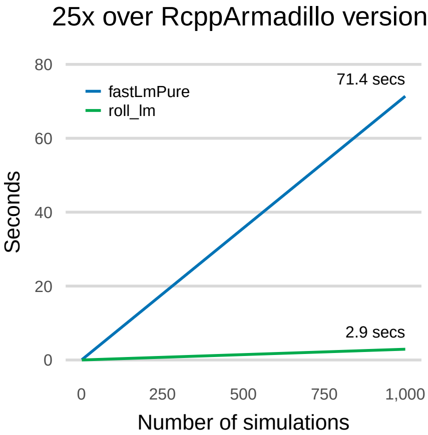
Download daily returns for 150 portfolios in the Tactical Allocation and World Allocation categories and apply a 15 asset class factor model on a daily rolling window:

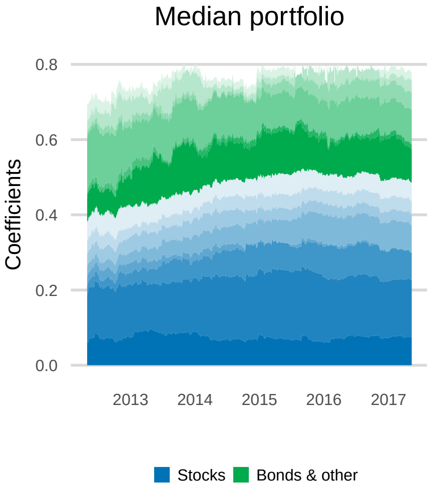
Stocks	Bonds & other
US Large Cap Value	US Gov 1-10Y
US Large Cap Growth	US Gov 10Y+
US Mid Cap	Non-US Gov
US Small Cap	IG Corp
Europe	HY Corp
Japan	MBS
Pacific ex Japan	Commodities
Emerging Markets	

Sharpe, William F. 1988. "Determining a Fund's Effective Asset Mix." *Investment Management Review*.

Asset allocation

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

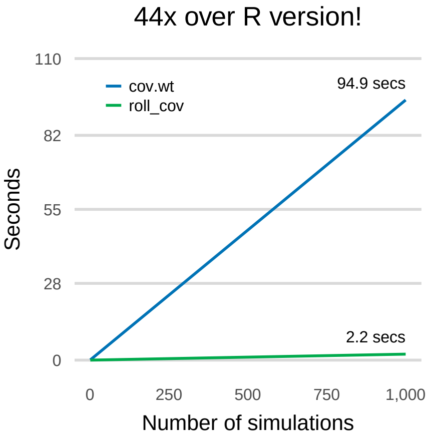


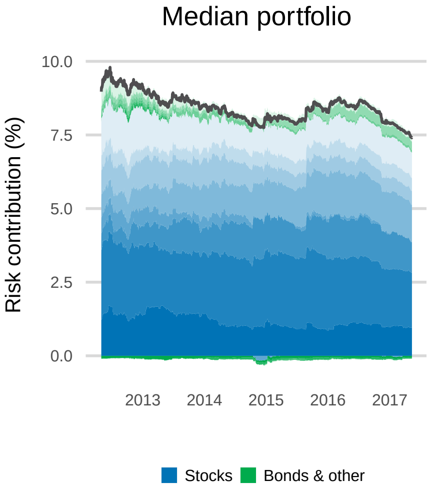


Speed test uses univariate betas

Concentrated in equity risk

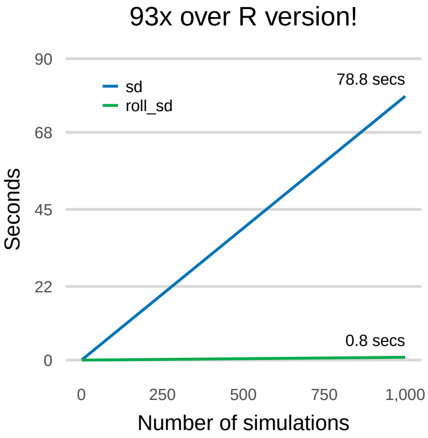
A significant driver of return variability is the concentration of equity risk:

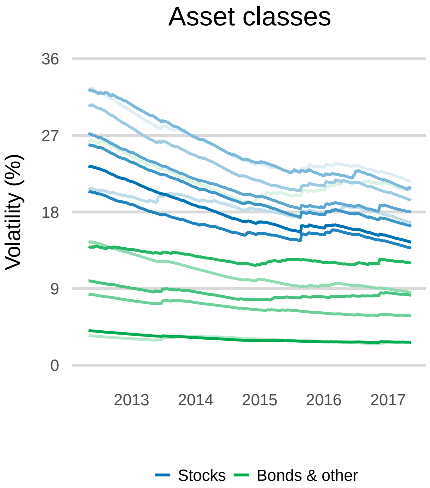




Decline in realized volatility

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





Simple risk parity strategy

Consider a portfolio of N assets and let $\mathbf{w} = (w_1, \ldots, w_N)$ be a vector of asset weights. Also define Σ 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:

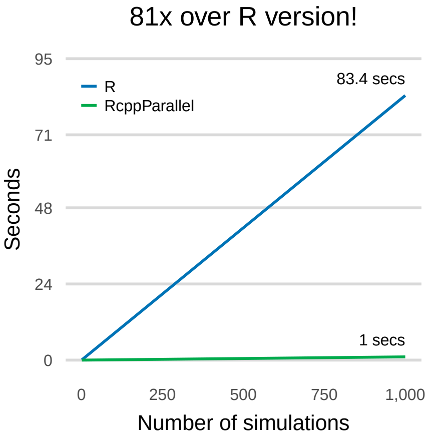
- **Instruments**: 15 asset classes
- **Risk settings**: long-term; exponential decay
- **Approach**: passive; equal risk contributions
- **Volatility target**: 10% volatility level
- **Rebalancing**: monthly; systematic

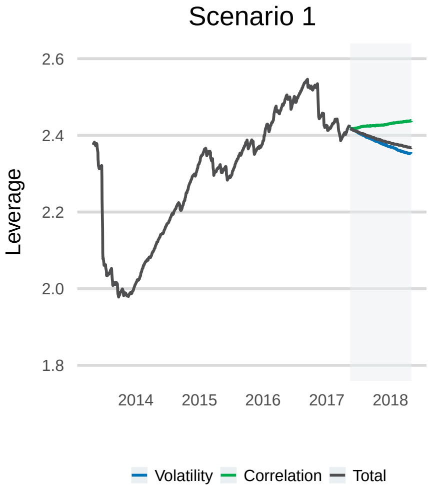
Now, against the backdrop of the current environment of low realized volatility, consider a few generic scenarios:

Scenario	Volatility	Correlation
Scenario 1	Average	Average
Scenario 2	Low	Low
Scenario 3	High	High

A toy example: return to average

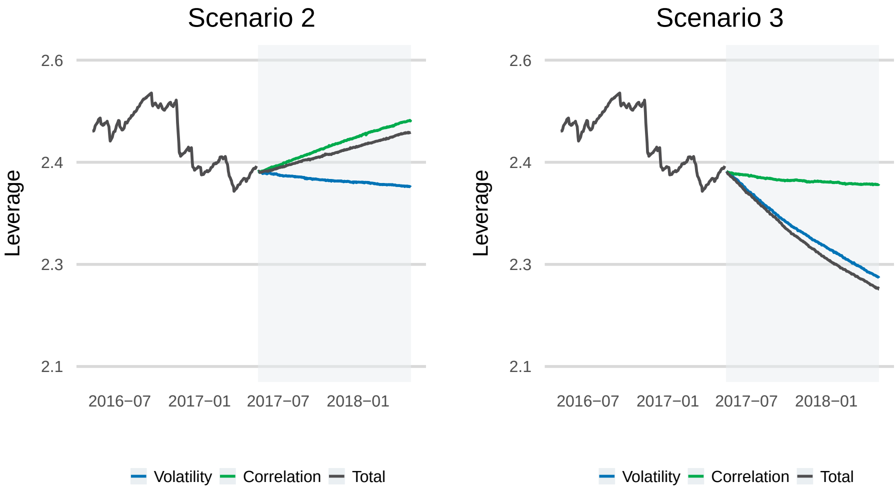
Simulate scenarios and mechanics of the simple risk parity strategy:





Explore the scenarios

Parallel computing via **`RcppParallel`** helps to dig deeper and faster:



roll: rolling statistics

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")
```