Backtesting Trading Strategies in R

Martin Geissmann

02 Jun 2022

1 Introduction

Backtesting is simulating a systematic trading strategy on past data.

For example:

- Always invest in equal amounts in SPY, QQQ, and TLT. Rebalance annually.
- Invest in IWM if above 200-day moving average, otherwise AGG.
- Invest in the tangency portfolio (Markowitz' model) using all S&P 500 stocks.
- Invest in the 10 stocks with the highest retail trading volume (alternative data)

In general, a backtest is the result of two steps:

1. Signal →Theoretical portfolio weights
2. Theoretical portfolio weights →Actual weights and portfolio return

The second step is subject to assumptions, e.g. the rebalance frequency, fees, slippage etc.

What are our options for backtesting trading strategies in R?

- "Basic" R (xts or tibble/dplyr)
- Easy backtesting functions/packages
- Accounting backtesting packages

1.1 How to find R packages

Packages by name: https://cran.r-project.org/web/packages/available_packages_by_name.html

Search for packages using `packagefinder::findPackage`

```
library(packagefinder)

findPackage("backtest")
```

1.2 Packages

2 dplyr

2.1 SMA Rotation Strategy (Risk On/Off)

Rule: If the S&P 500 is trading above its 200-days SMA, then invest in stocks. If it's below, then rotate into gold.

```
library(tidyverse)
library(lubridate)
library(tidyquant)
```

Table 1: Packages

package	description	url
backtest	Old (2015). Same developer as 'strand'. Creates quantiles and calculates the average return within each quantile bucket.	link
strand	Somewhat current (end of 2020), vectorized computation, a bit tricky to set constraints etc. Shiny demo included.	link
QuantTools	Recently removed from CRAN. Last updated 2020. Uses C++ for strategy definitions. Includes a nice and simple function back_test() that allows for seeing the trades based on a signal.	link
rsims	Not on CRAN, recently updated on Github (Feb 2022). Only rebalances if actual weights deviate by more than the set buffer (trade_buffer) from the desired weights (theo_weights). Main function is cash_backtest(). Quite simple for what it aims to do.	link
SIT	Not on CRAN (anymore?). Last updated on Github Jan 2021. Very extensive.	link
quantstrat	Not on CRAN (anymore?). Besides SIT one of the most extensive backtesting packages. Several contributors on Github, last updated March 2021. Stores objects in separate environment, which is a bit untransparent (for a tidyverse accustomed user).	link
PMwR	Portfolio Management with R. Sandbox that allows for all kind of custom strategies. As a result, it's a bit tricky.	link
portfolioBacktest	New 2022. Quite extensive. Once a strategy is defined, it can be run on different datasets.	link
AssetAllocation	New 2022. Only constant weights for now, but should be updated soon. Easy and interesting approach.	link
PortfolioAnalytics	Portfolio optimization, updated 2018.	link
Strategy	Updated 2017.	link

```
library(RcppRoll)
library(scales)
```

Get data.

```
spy <- tq_get("SPY", from = as.Date("2005-01-01"))
gld <- tq_get("GLD", from = as.Date("2005-01-01"))
```

```
spy_gld <- full_join(spy %>%
                       select(date, spy.cl = close, spy.adj = adjusted),
                     gld %>%
                       select(date, gld.adj = adjusted),
                     by = "date") %>%
  arrange(date) %>%
  mutate(spy.r = spy.adj/lag(spy.adj)-1,
         gld.r = gld.adj/lag(gld.adj)-1)
```

```
strategy <- spy_gld %>%
  mutate(SMA = roll_mean(spy.cl, n = 200, align = "right", fill = NA_real_),
         SMA_signal = lag(spy.cl >= SMA)) %>%
  filter(!is.na(SMA_signal)) %>%
  mutate(strategy.r = ifelse(SMA_signal, spy.r, gld.r))
```

```
strategy_long <- strategy %>%
  select(date, spy.r, gld.r, strategy.r) %>%
  pivot_longer(-date, values_to = "r") %>%
  arrange(name, date) %>%
  group_by(name) %>%
  # cumulative performance
  mutate(p = cumprod(1+r)-1) %>%
  group_by(series = str_to_upper(str_remove(name, ".r$")))
```

```
strategy_long %>%
  ggplot(aes(x = date, y = p, color = series)) +
  geom_line() +
  scale_color_brewer(type = "qual", palette = 3) +
  scale_y_continuous(labels = percent) +
  scale_x_date(date_labels = "%b %y") +
  labs(x = "", y = "", color = "") +
  theme_minimal() +
  theme(legend.position = "top")
```

```
strategy_long %>%
  # annualized performance (PerformanceAnalytics, tidyquant)
  tq_performance(Ra = r, performance_fun = table.AnnualizedReturns, scale = 252)
```

```
strategy_long %>%
  # calculate drawdowns
  mutate(dd = (1+p)/cummax(1+p)-1) %>%
  ggplot(aes(x = date, y = dd, color = series)) +
  geom_line() +
  scale_color_brewer(type = "qual", palette = 3) +
  scale_y_continuous(labels = percent) +
  scale_x_date(date_labels = "%b %y") +
  labs(x = "", y = "", color = "") +
  theme_minimal() +
  theme(legend.position = "top")
```

3 portfolioBacktest

- requires xts data
- ideal of testing on several subsamples of data at once

```
library(portfolioBacktest)

data("dataset10")

dataset10$`dataset 1`$adjusted

# defining the portfolio (equal weighting)
my_portfolio <- function(dataset, ...) {
  prices <- dataset$adjusted
  print(nrow(prices))
  N <- ncol(prices)
  return(rep(1/N, N))
}

bt <- portfolioBacktest(my_portfolio, list(dataset10$`dataset 1`))

backtestSummary(bt)$performance

my_portfolio_sma <- function(dataset, ...) {
  prices <- dataset$adjusted
  print(prices)
```

```
    N <- ncol(prices)
  return(rep(1/N, N))
}

bt <- portfolioBacktest(portfolio_funs = my_portfolio_sma(), dataset_list = dataset10)
```

4 AssetAllocation

```
library(AssetAllocation)
library(PerformanceAnalytics)
names(basic_asset_alloc)
```

```
basic_asset_alloc$all_weather

ivy_port_weights()
```

```
## Example 1: backtesting one of the asset allocations in the package
strat_permanent <- basic_asset_alloc$permanent
strat_fancy <- list(name = "Fancy",
                    tickers = c("SPY", "TLT"),
                    default_weights = c(1, 0),
                    rebalance_frequency = "month",
                    portfolio_rule_fn)

# test using the data set provided in the package
bt_strat_permanent <- backtest_allocation(strat_permanent, ETFs_daily)

# plot cumulative returns
chart.CumReturns(bt_strat_permanent$returns,
                 main = paste0("Cumulative returns of the ",
                               bt_strat_permanent$strat$name,
                               " portfolio"),
                 ylab = "Cumulative returns")
```

Only supports constant weights for now.