Lab Week 10 : Monte Carlo methods

STAT5003

Dr. Justin Wishart

Semester 2, 2022

Contents

Preparation and assumed knowledge

- Viewed the Monte Carlo methods content in Module 10.

Aims

- Implement Monte Carlo techniques to
  - Evaluate a mathematical integral
  - Conduct a simulation of pricing an exotic stock option.
  - Explore gambling strategies

This week, we will be exploring Monte Carlo method for integration and simulating games.

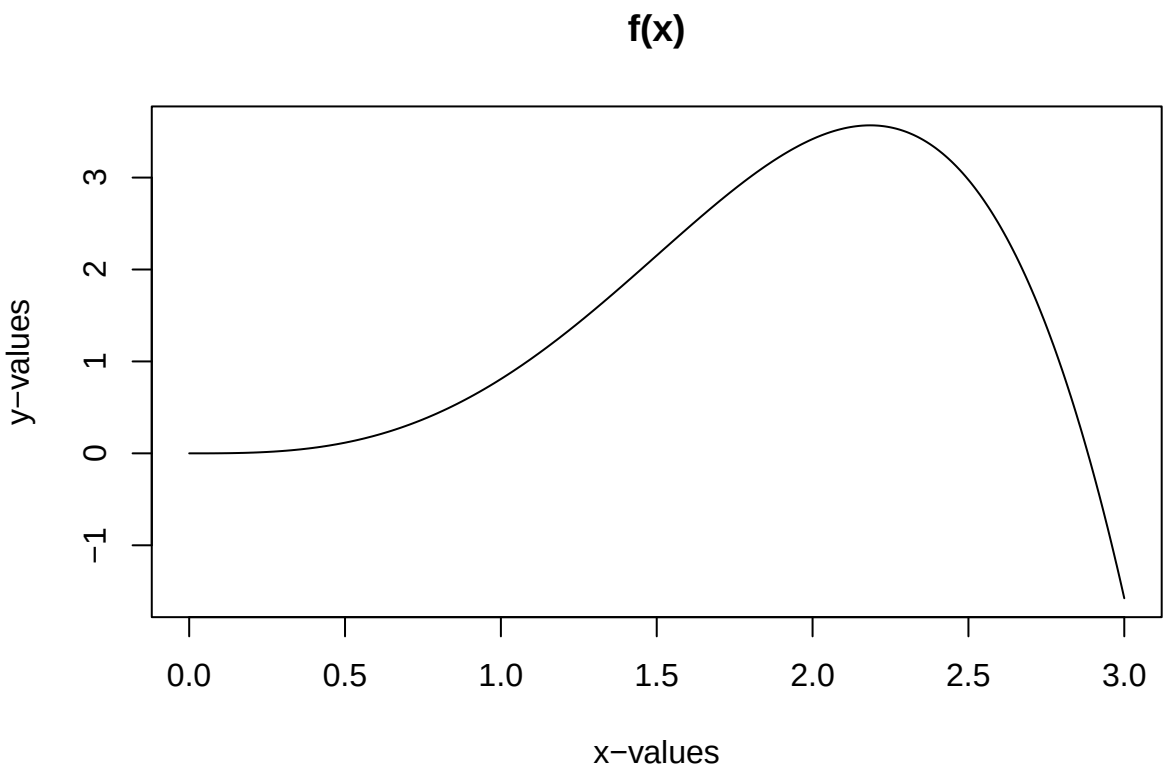
We won't be using any dataset in today's tutorial.

1 Evaluate an integral

Integrate the following function using the Monte Carlo integration method on the interval [0,3] on x. Compare the Monte Carlo estimate of the integral to the value calculated using the `integrate()` function in R.

$$f(x) = x^3 \sin\left(\frac{x + 3.4}{2}\right)$$

```
# plot data and create the true relationship line
s <- function(x) { x^3 * sin((x + 3.4) / 2) }
x.plot <- seq(0, 3, length.out = 1000)
y.plot <- s(x.plot)
plot(x.plot, y.plot, xlab = "x-values", ylab = "y-values", main = "f(x)", type = "l")
```




```
# Randomly sample from the interval we want to integrate
n <- 50000
x <- runif(n, min = 0, max = 3)
sx <- s(x)
# Remember to multiply by the length of the interval from 0 to 3.
ans.mc <- 3 * mean(sx)
ans.mc
```

```
## [1] 4.637169
```

```
# use the inbuilt integrate function
ans.quad <- integrate(s, 0, 3)
ans.quad
```

```
## 4.652802 with absolute error < 5.4e-14
```

2 Exotic Financial option simulation

Aside from the simple call and put options in finance, there are many more exotic variations of these options. One such exotic is called a **knock-out call option**. In a knock-out call option, you have a knock-out level and if the stock hits this level **at any time** before expiry, then the option expires worthless. To clarify, there are two standard types of options.

- European option. Gives the option owner the right(but not the obligation) to purchase (or sell) a stock at the expiry date of the option.
- American option. Is a generalization of the European option whereby the option owner can have the right (but not obligation) to purchase (or sell) a stock at any time between the purchase of the option and its expiry date.

Assume a standard European option style trade. That is, the exotic knock out option in this lab will still use a European option style were the payoff (transaction) can only occur at the expiry date. The payoff function

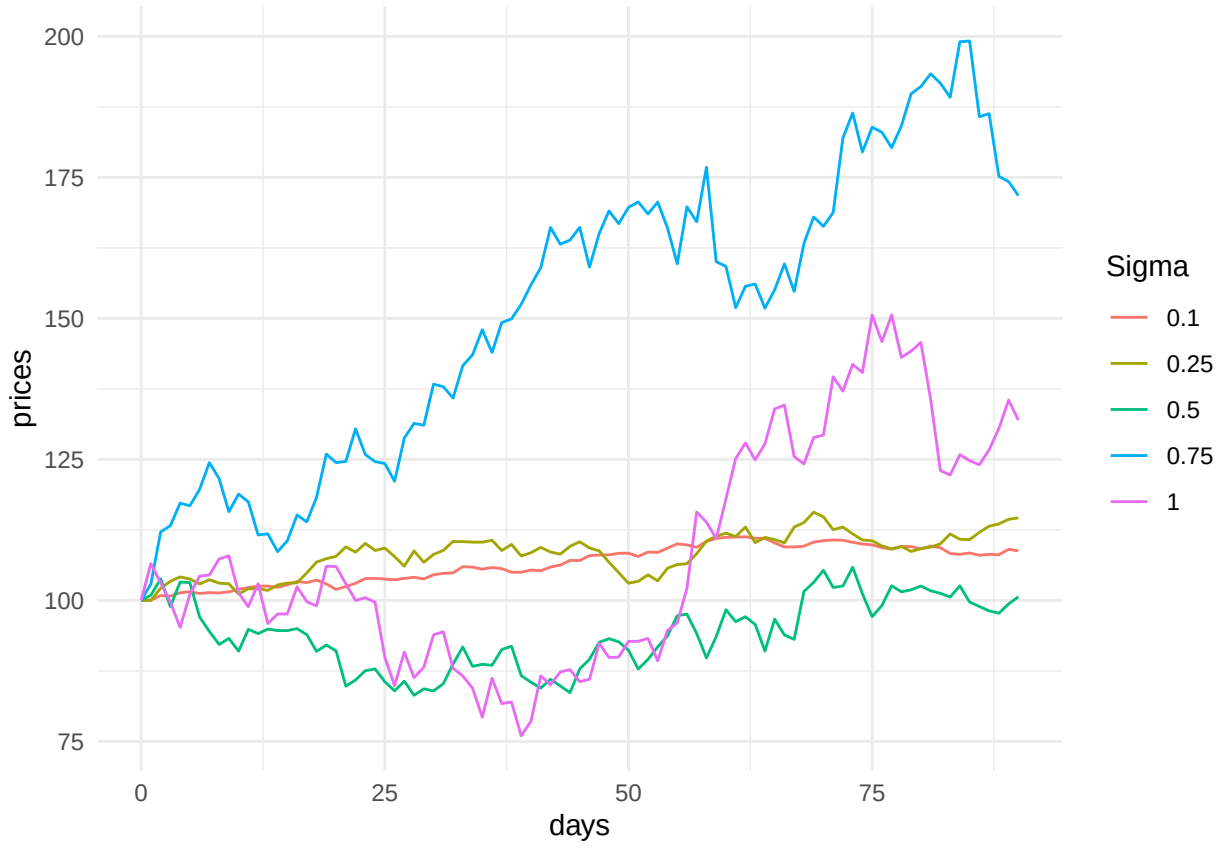
is computed at expiry but uses the entire history of the stock price between the time of the option being purchased and the expiry date. i.e. the payoff will be zero if the stock price hit or exceeds the knock out boundary at any time in its history. If the stock doesn't hit the boundary then the payoff is computed as the current stock price against the strike price, just as the regular vanilla option is priced.

Use Monte Carlo simulation to estimate the price of a knock-out call option.

2.1 Stock price simulation

Write code to simulate the prices of stocks using the Geometric Brownian motion type model discussed in the Module content. Write a function to easily simulate a new realization of a stock price assuming the Geometric Brownian motion model. The function should have parameters for the number of days in the simulation, the initial stock price (S_0), drift term (μ) and volatility (σ). Using your function, produce a plot of 5 different simulations of stock prices over a 90 day period assuming $S_0 = \$100$, $\mu = 0.05$ but the volatility changes from $\sigma = 0.1, 0.25, 0.5, 0.75, 1$. Assume the increment in time is daily over a year period ($dT = 1/365$)

```
stock.sim <- function(ndays = 90, S0 = 100, mu = 0.05, sigma = 0.5) {
    N <- rnorm(ndays)
    S <- S0
    dT <- 1 / 365
    Ssim <- S
    for (i in 1:ndays) {
        dS <- S * (mu * dT + sigma * sqrt(dT) * N[i])
        S <- S + dS
        Ssim <- c(Ssim, S)
    }
    Ssim
}
sigmas <- c(0.1, 0.25, 0.5, 0.75, 1)
simulated.prices <- unlist(lapply(sigmas, function(s) stock.sim(sigma = s)))
price.dat <- data.frame(days = rep(0:90, 5),
                        prices = simulated.prices,
                        Sigma = factor(rep(sigmas, each = 91)))
ggplot(price.dat) +
    theme_minimal() +
    geom_line(aes(x = days, y = prices, group = Sigma, colour = Sigma))
```



2.2 Compute the exotic option pay-offs

Write another function that takes a stock price vector (stock prices over a time period) and determines the pay-off of a this exotic knockout option. The function will need to have three arguments as input, the historical stock prices to be evaluated, the option strike price and the knockout level. E.g. suppose a knockout option had a strike of \$105, knock out level of \$130. Then this function would need to check that the entire stock price vector doesn't exceed \$130 (if it does the payoff is worthless, i.e. zero) and otherwise give the payoff as the maximum of the stock price at expiry minus the strike price and zero.

```
knockout.payoff <- function(stock.prices, strike.price, knockout)
    if (any(stock.prices > knockout)) 0 else max(tail(stock.prices, 1) - strike.price, 0)
```

2.3 Conduct Monte carlo simulation of pay-offs

Using the pricing strategy discussed in the Module, that is, assuming the fair price is the average expected payoff of the option, compute the appropriate price for a knockout exotic option that expires in 90 day with a strike price of \$105 and a knockout level of \$130. Assume that the associated stock prices have the calibration of $S_0 = \$100$, $\mu = 0.05$ and $\sigma = 0.5$. Determine how much cheaper is this knock-out call option compared to a vanilla call option. Create a plot to make that comparison.

```
n.iterations <- 5000
S0 <- 100
mu <- 0.05
sigma <- 0.5
ndays <- 90
vanilla.payoff <- function(stock.prices, strike.price) max(stock.prices - strike.price, 0)
simulated.prices <- lapply(rep(ndays, n.iterations), stock.sim,
                           S0 = S0, mu = mu, sigma = sigma)
```

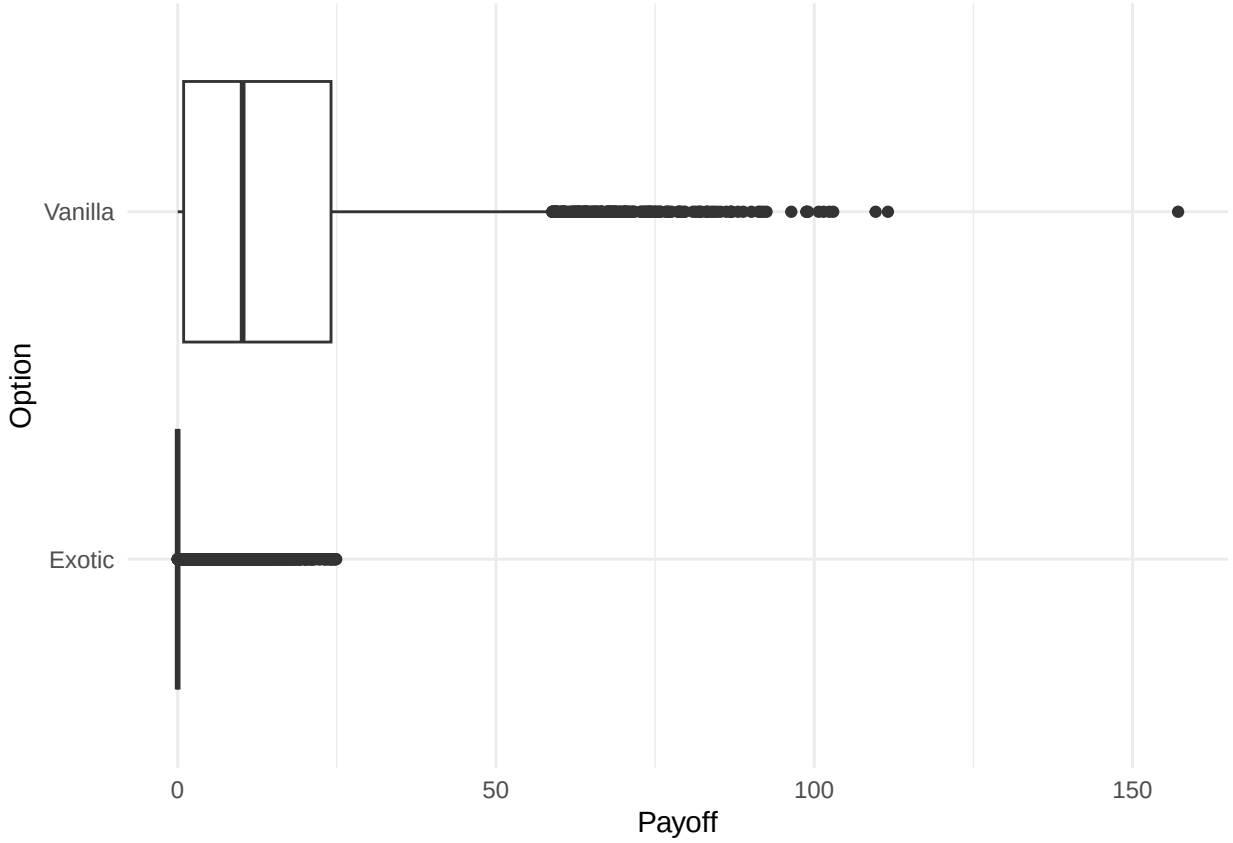
```
exotic.pay.offs <- vapply(simulated.prices, knockout.payoff, numeric(1), strike.price = 105, knockout =
vanilla.pay.offs <- vapply(simulated.prices, vanilla.payoff, numeric(1), strike.price = 105)
mean(exotic.pay.offs)
```

```
## [1] 1.543946
```

```
mean(vanilla.pay.offs)
```

```
## [1] 15.8219
```

```
dat <- data.frame(Option = rep(c("Vanilla", "Exotic"), each = n.iterations),
                  Payoff = c(vanilla.pay.offs, exotic.pay.offs))
ggplot(data = dat) + theme_minimal() +
  geom_boxplot(aes(x = Payoff, y = Option))
```



The exotic option is much cheaper since near half the time the option expires worthless since the stock hit the knockout level while the vanilla option might have a very large payoff in those situations.

3 Optional: Roulette

For this question we will perform Monte Carlo simulation on the roulette casino game. A roulette wheel has numbers from 1 to 36, alternating between red and black, and an extra green pocket labelled as 0. There are various bets you can make, but the simplest is to bet on red or black, in which case if the ball lands on your colour, you win double your bet.

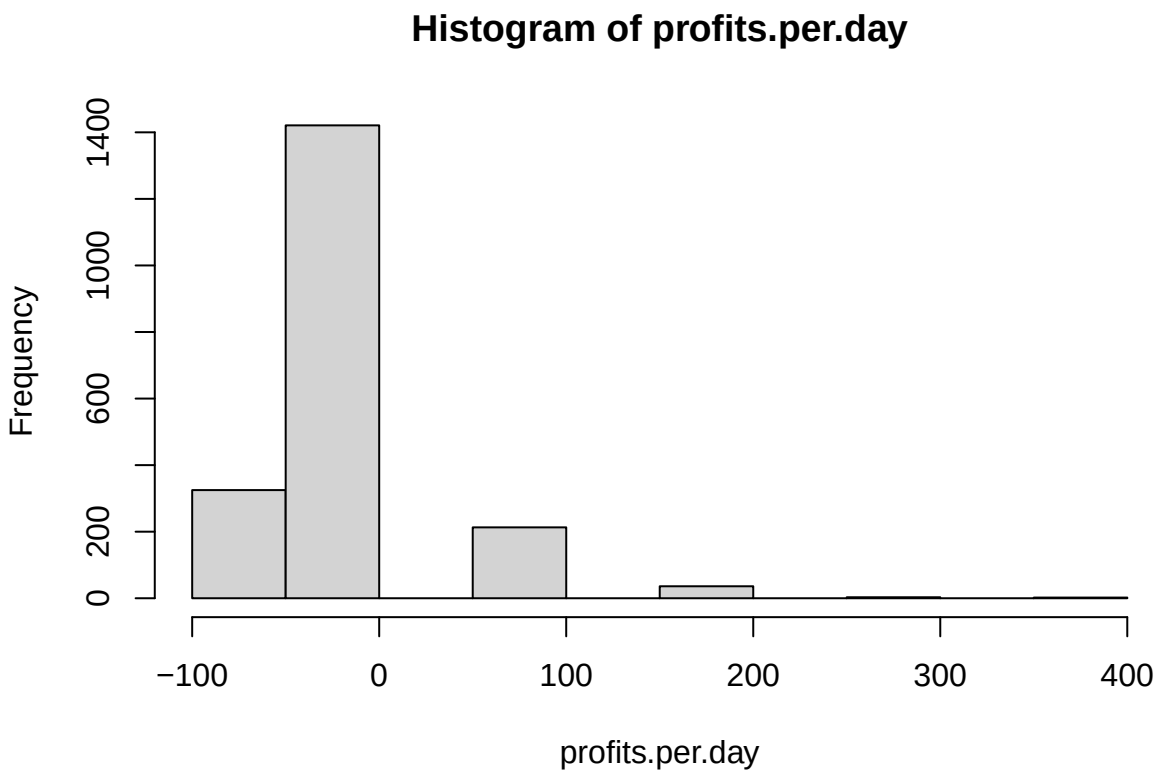
```
![](roulette.jpg){width=240px}
```

```
You decide that you want to become a professional gambler and come up with two strategies for beating t
```

3.1 First strategy

In the first strategy, you believe that if a roulette wheel has had a run of blacks, then it is more likely to spin a red in the next spin. So you wait for a run of five blacks, then bet all in ($100) on red in the next spin. If you lose all $100, then you go home early that night. Simulate 1000 nights of games and estimate the expected win rate per night. **Note** This question asks you to explore Gambler's fallacy, which is the belief that if something has happened more frequently in the past, it will happen less frequently in the future which is not true if the roulette wheel is balanced.

```
onenightA <- function() {
    n.spins <- 60
    spins <- sample.int(37, size = n.spins, replace = TRUE)

    # Let's say odd numbers are red and even numbers are black
    reds <- sapply(spins, function(x) ifelse((x %% 2) == 0, 1, 0))

    red.runs <- 0
    curr.bankroll <- 100

    for (i in 1:n.spins) {
        if (curr.bankroll <= 0) break
        # run of 5 reds, then this run is black
        if (red.runs >= 5 && spins[i] == 37) {
            curr.bankroll <- curr.bankroll - 100
        } else if (red.runs >= 5 && reds[i] == 0) {
            curr.bankroll <- curr.bankroll + 100
        } else if (red.runs >= 5 && reds[i] == 1) {
            curr.bankroll <- curr.bankroll - 100
        }

        red.runs <- if (reds[i] == 1) red.runs + 1 else 0
    }
    curr.bankroll
}


n <- 1000
profits.per.day <- vector("numeric", n)
for (i in 1:n) {
   profit <- onenightA() - 100
   profits.per.day <- c(profits.per.day, profit)
}

hist(profits.per.day)
```

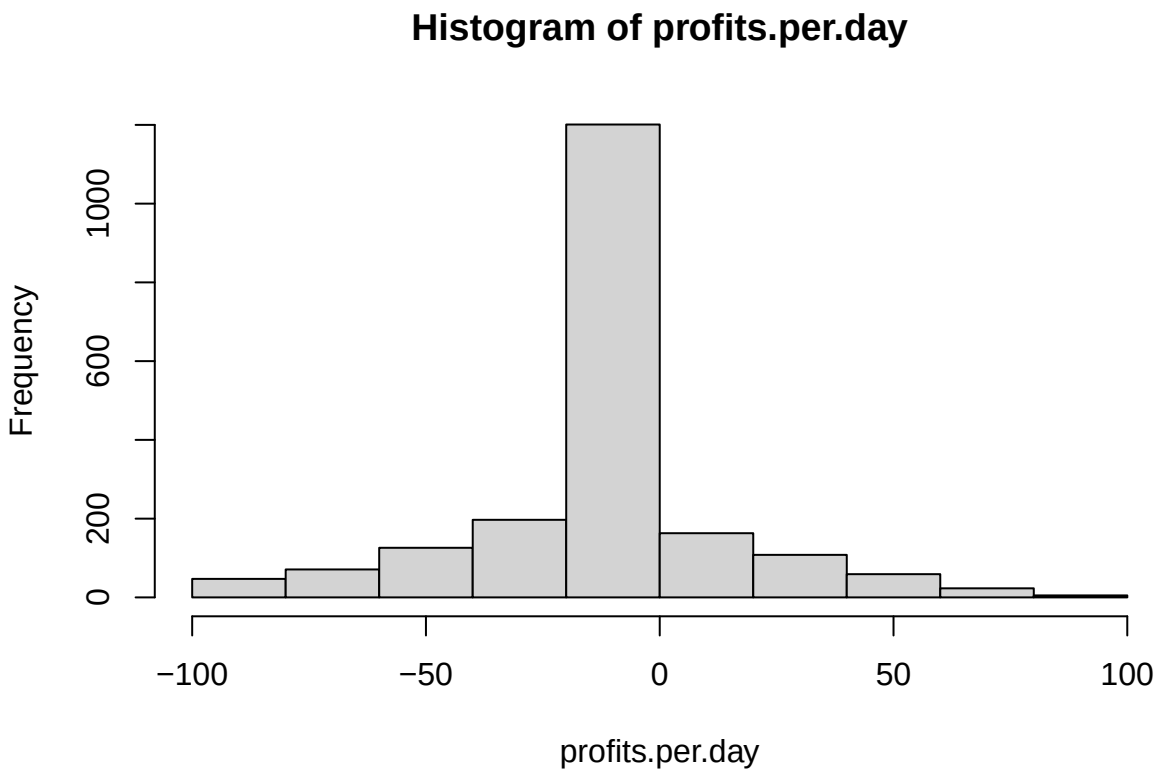


3.2 Second strategy

In the second strategy, you take a more cautious approach. You bet $5 on red each time, and go home early if you lose all your money ($100), i.e. you bust. Simulate 1000 nights of games and estimate the probability of busting each night.

```
onenightB <- function() {
    n.spins <- 60
    spins <- sample.int(37, size = n.spins, replace = TRUE)

    # Let's say odd numbers are red and even numbers are black
    reds <- sapply(spins, function(x) ifelse((x %% 2) == 0, 1, 0))

    curr.bankroll <- 100

    for (i in 1:n.spins) {
        if (curr.bankroll <= 0)
            break

        if (spins[i] == 37) {
            curr.bankroll <- curr.bankroll - 5
        } else if (reds[i] == 1) {
            curr.bankroll <- curr.bankroll + 5
        } else if (reds[i] == 0) {
            curr.bankroll <- curr.bankroll - 5
        }
    }
    curr.bankroll
}
```

```r
n <- 1e3
profits.per.day <- vector("numeric", n)
for (i in 1:n) {
   profit <- onenightB() - 100
   profits.per.day <- c(profits.per.day, profit)
}

hist(profits.per.day)
```




```r
# Average loss per day
mean(profits.per.day)
```

```
## [1] -4.44
```