COMP 226: Slides 23

Exam preparation

Rahul Savani

rahul.savani@liverpool.ac.uk

1. Consider the following R code.

```
mat <- matrix(0,nrow=3,ncol=3)
```

Which of the following will remove the third row from `mat`?

☐ **A.** `mat <- mat[,1:2]`

☐ **B.** `mat <- mat[-3,]`

☐ **C.** `mat <- mat[3,]`

☐ **D.** `mat <- mat[-c(1,2),]`

☐ **E.** `mat <- mat[-3]`

2. Consider the following R code.

```
x <- 1; y <- 2
func <- function() {x <- 2; y <<- 1}
func()
k <- x+y
```

Which of the following corresponds to the resulting element of `k`?

☐ **A.** 1

☐ **B.** 2

☐ **C.** 3

☐ **D.** 4

☐ **E.** 5

3. Consider the following R code.

```r
available <- c(10,4,7,10,12)
desired <- c(12,5,2,6,14)
traded <- sum(mapply(function(x,y) min(x,y), available, desired))
```

Which of the following corresponds to the resulting element of `traded`?

☐ **A.** 2

☐ **B.** 4

☐ **C.** 6

☐ **D.** 34

☐ **E.** 30

4. Which of the following types of price are best for backtesting equities using daily data?

- ☐ **A.** Open
- ☐ **B.** High
- ☐ **C.** Low
- ☐ **D.** Close
- ☐ **E.** Adjusted

5. Consider a 2-for-1 stock split. Which of the following describes the new additional step in the stock price adjustment process (for prices before the dividend date)?

☐ **A.** Add 2

☐ **B.** Subtract 2

☐ **C.** Subtract 3

☐ **D.** Divide by 2

☐ **E.** Multiply by 3

6. Suppose that the state of the limit order book is as follows.

Bids	Price	Offers
	110	6
	100	4
5	90	
5	80	

Suppose that a limit order arrives to buy 4 units at prices no worse that 105. What will be the best offer price *after* the order is processed?

☐ **A.** 80

☐ **B.** 90

☐ **C.** 100

☐ **D.** 105

☐ **E.** 110

7. Suppose that the state of the limit order book is as follows.

Bids	Price	Offers
	11	6
	10	4
5	9	
5	8	

Suppose that a limit order arrives to sell 6 units at prices no worse than 8. What will be the value of the spread *after* the order is processed?

☐ **A.** -1

☐ **B.** 0

☐ **C.** 1

☐ **D.** 2

☐ **E.** 3

8. Consider the following incomplete R code, which takes as its first argument a variable `askBook` which is a matrix with two columns with names `"price"` and `"volume"`. The function `checkAvailability` should return the total number of units available for sale in `askBook` at prices no worse than `priceThresh`, its second argument, which represents a price level.

```
1 checkAvailability <- function(askBook, priceThresh) {
2         available <-
3         return(available)
4 }
```

Which of the following correctly completes line 2 of the code?

☐ **A.** `max(askBook[askBook[,"price"]>= priceThresh),"volume"])`

☐ **B.** `min(askBook[askBook[,"price"]<= priceThresh),"volume"])`

☐ **C.** `max(askBook[askBook[,"price"]<= priceThresh),"volume"])`

☐ **D.** `sum(askBook[askBook[,"price"]<= priceThresh),"volume"])`

☐ **E.** `sum(askBook[askBook[,"price"]>= priceThresh),"volume"])`

9. Let the price of an asset in periods $t-1$ and t be P_{t-1} and P_t, respectively. Which of the following is the correct formula for the simple return of taking a short position from $t-1$ to t:

☐ **A.** $\frac{P_t - P_{t-1}}{P_t}$

☐ **B.** $\frac{P_t - P_{t-1}}{P_{t-1}}$

☐ **C.** $\log(P_t/P_{t-1})$

☐ **D.** $\log(P_{t-1}/P_t)$

☐ **E.** $\frac{P_{t-1} - P_t}{P_t}$

10. Which of the following expressions describes the simple return for a short position in terms of the corresponding simple return for a long position R_t?

☐ **A.** $(R_t - 1)/R_t$.

☐ **B.** $1 - R_t$

☐ **C.** $-R_t$

☐ **D.** $-R_t/(R_t + 1)$.

☐ **E.** $-(R_t + 1)/R_t$.

11. Consider the following R code. The vector `log_ret` represents the log returns of an asset. The vector `pos` represents unit positions in this asset taken by a trading strategy, where 1, 0, -1, represents long, flat, and short, respectively. The function `cumulativeLogReturns` should compute the cumulative log returns corresponding to `log_ret` and `pos`.

```
1 log_ret <- c(0.2,-0.3,0,0.4,0.1,0,-0.7)
2 pos <- c(0,-1,0,0,1,0,1)
3 cumulativeLogReturns <- function(log_ret,pos) {
4   cumLogRets <-
5 }
```

Which of the following correctly completes line 4?

☐ **A.** `cummax(log_ret*pos)`

☐ **B.** `cumsum(log_ret*pos)`

☐ **C.** `cumprod(log_ret*pos + 1) - 1`

☐ **D.** `cumprod(log_ret*pos + 1)`

☐ **E.** `cummax(log_ret*pos) - 1`

12. Consider the following R code.

```r
cm <- cummax(c(-1,1,-2,2,-3,3))
```

Which of the following corresponds to resulting elements of `cm`?

- ☐ **A.** `1 1 2 2 3 3`
- ☐ **B.** `-1 1 2 2 3 3`
- ☐ **C.** `-1 1 1 2 2 3`
- ☐ **D.** `-1 1 -2 2 -3 3`
- ☐ **E.** `3 3 3 3 3 3`

13. Suppose that `wealthIndex` is a vector in R that represents the Wealth Index of a trading strategy. Which of the following pieces of R code computes the corresponding maximum drawdown?

☐ **A.** `max(cummax(wealthIndex)/wealthIndex-1)`

☐ **B.** `max(wealthIndex/cummax(wealthIndex)-1)`

☐ **C.** `max(1-wealthIndex/cummax(wealthIndex))`

☐ **D.** `max(cummax(wealthIndex)-wealthIndex)`

☐ **E.** `max(cummin(wealthIndex)-wealthIndex)`

14. Consider the following R code for computing a simple moving average

```
prices  <- c(100,102,103,104)
s <- vector(mode='numeric',length=length(prices))
n <- 2
s[1] <- prices[1]
for (i in n:length(prices))
  s[i] <- sum(prices[i:(i-n+1)])/n
```

Which of the following correspond to the entries of the vector `s` after this code has been run?

☐ **A.** 100 102.5 103.5 104.5

☐ **B.** 102 103 104 105

☐ **C.** 100 102 103 104

☐ **D.** 100.0 101.0 102.5 103.5

☐ **E.** 101 103 103.5 104

15. Which of the following implement a high-pass filter using an input signal and the output of a simple moving average applied to the input?

☐ **A.** Add the original signal to the output of the simple moving average

☐ **B.** Subtract the original signal from the output of the moving average

☐ **C.** Multiply the original signal and the output of the simple moving average

☐ **D.** Subtract the output of the simple moving average from the signal

☐ **E.** Divide the output of the simple moving average by the original signal

16. If there is a universe of 5 stocks, how many different pairs of stocks are there?

☐ **A.** 10

☐ **B.** 21

☐ **C.** 28

☐ **D.** 30

☐ **E.** 32

17. Consider the following R code.

```r
prices <- c(100,101,99,98,99,101,97,103)
n <- 2
long <- short <- rep(0,length(prices))
for (i in (n+1):(length(prices))) {
    long[i] <- ifelse(all(prices[i] > prices[(i-1):(i-n)]), 1,0)
    short[i] <- ifelse(all(prices[i] < prices[(i-1):(i-n)]),-1,0)
}
signal <- long + short
```

Which of the following corresponds to the resulting elements of `signal`?

☐ **A.** 1 -1 1 1 0 0 -1 1

☐ **B.** 0 -1 0 0 1 1 1 0

☐ **C.** 0 0 -1 -1 0 1 -1 1

☐ **D.** 0 0 1 1 0 -1 1 -1

☐ **E.** 1 1 0 0 1 -1 0 -1

18. Suppose a trading strategy has k parameters labelled $1, 2, \ldots, k$, and that the number of parameter values that each parameter can take on is p_k for each parameter $1, 2, \ldots, k$. How many parameter combinations are there in total?

☐ **A.** $\sum_{i=1,\ldots,k} p_i$

☐ **B.** $\prod_{i=1,\ldots,k} p_i$

☐ **C.** $(\max_{i=1,\ldots,k} p_i)^k$

☐ **D.** $(\min_{i=1,\ldots,k} p_i)^k$

☐ **E.** $\exp(\sum_{i=1,\ldots,k} p_i)$

19. Consider the following R code that defines the parameters of a trading strategy and computes the number of parameter combinations.

```
n <- seq(from=10,to=100,by=10)
hold <- seq(from=10 ,to=20 ,by=1)
flag <- 0:1
params <- list(n,hold,flag)
nParams <- prod(sapply(params,length))
```

Which of the following corresponds to the resulting element of `nParams`?

☐ **A.** 100

☐ **B.** 200

☐ **C.** 220

☐ **D.** 242

☐ **E.** 244

20. Suppose that vector `pos` in R encodes the position sizes of a strategy as follows: if `pos[i]` is equal to k then the strategy was k units long if $k > 0$ and $-k$ units short if $k < 0$, and flat otherwise, if $k = 0$. The vector `pos` is of length `n`. Assume that `pos[1]` is 0. Which of the following is correct R code for the number of units of slippage that would be incurred assuming that every time the position changes by $x > 0$ units in absolute value then x units of slippage are incurred?

☐**A.** `sum(abs(pos[1:n] - pos[2:(n-1)]))`

☐**B.** `abs(sum(pos[1:n] - pos[2:(n-1)]))`

☐**C.** `abs(sum(pos[2:n] - pos[1:(n-1)]))`

☐**D.** `sum(abs(pos[2:n] - pos[1:(n-1)]))`

☐**E.** `abs(sum(pos[1:n] + pos[2:(n-1)]))`