Programming for E&BI 2025 Exam Solutions

Short-Answer Questions (5 points)

Question 1 (2 points)

Write an R command that calculates the following:

$$\frac{|-3| + \log_3(9)}{2}$$

where $|x|$ is the absolute value of x.

Provide both the numerical answer and the R command.

Answer:

```
(abs(-3) + log(9, base = 3))/2
```

```
[1] 2.5
```

Question 2 (1 point)

Write an R command in the box below that repeats the sequence $(3, 2, 1)$ one hundred times.

The resulting vector should be of the form $(3, 2, 1, 3, 2, 1, \ldots, 3, 2, 1, 3, 2, 1)$ and have 300 elements.

Answer:

```
rep(seq(from = 3, to = 1, by = -1), times = 100)
```

```
  [1] 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3
 [38] 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2
 [75] 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1
[112] 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3
[149] 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2
[186] 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1
[223] 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3
[260] 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2
```

```
[297] 1 3 2 1
```

Another valid approach would be:

```
rep(3:1, times = 100)
```

```
  [1] 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3
 [38] 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2
 [75] 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1
[112] 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3
[149] 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2
[186] 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1
[223] 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3
[260] 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1 3 2
[297] 1 3 2 1
```

Question 3 (1 point)

The vector `x` is the sequence $(1, 2, \ldots, 19, 20)$. This can be created in R with the command:

```
x <- 1:20
```

Write an R command in the box below *using indexing* that returns all the even numbers from `x`. That is, your command should be of the form `x[???]` where `???` indexes the even elements of `x`. The output should have 10 elements.

Answer: There are many ways to answer this question. One way is to create the sequence $(2, 4, \ldots, 18, 20)$ and use that to index x:

```
x <- 1:20
x[seq(2, 20, by = 2)]
```

```
 [1]  2  4  6  8 10 12 14 16 18 20
```

Another way to create the same sequence is to use `1:10 * 2`:

```
x[1:10 * 2]
```

```
 [1]  2  4  6  8 10 12 14 16 18 20
```

Another way is to alternate `FALSE`, `TRUE` 10 times. You can do this with `rep(c(FALSE, TRUE), 10)`.

```
x[rep(c(FALSE, TRUE), 10)]
```

```
 [1]  2  4  6  8 10 12 14 16 18 20
```

One option making use of the modulo operator is as follows:

```
x[x %% 2 == 0]
```

```
 [1]  2  4  6  8 10 12 14 16 18 20
```

Here, `x %% 2` returns the remainder when dividing an element of x by 2. If the remainder is 0, then it is exactly divisible by 2 and thus even. So `x %% 2 == 0` returns `TRUE` when `x` is even and `FALSE` when odd.

Question 4 (1 point)

The variable `x` is a single number. Write an R command in that box below that returns `TRUE` if `x` is in the interval $(2, 3)$ and `FALSE` otherwise. The interval $(2, 3)$ includes all numbers between 2 and 3 but not including 2.0 and 3.0. That is, it returns `TRUE` if $x > 2$ AND $x < 3$ and `FALSE` otherwise.

Answer: The answer is:

```
(x > 2) & (x < 3)
```

To see this, generate some values of x both inside and outside the interval:

```
x <- c(0, 1, 2, 2.2, 2.8, 3, 4)
(x > 2) & (x < 3)
```

```
[1] FALSE FALSE FALSE  TRUE  TRUE FALSE FALSE
```

Data Analysis (8 points)

Download the dataset ranking-clicks.csv. The dataset contains information on the weekly number of clicks products receive in an online web store. The variable descriptions are:

- `product_id`: The product ID (1 to 25).
- `week`: The week number (1 to 6).
- `ranking`: The position ranking of the product on the web store (1 is at the top of the webpage, 25 is at the bottom of the webpage).
- `price`: The price of the product.
- `clicks`: The total number of clicks the product received in the week.

When reading the dataset into R, assign it to `df`.

Question 5 (2 points)

What is the median of the variable `price`?

Provide both the numerical answer and the R command required to obtain the answer (if the dataframe is assigned to `df`).

Answer:

```
df <- read.csv("ranking-clicks.csv")
median(df$price)
```

```
[1] 79.49
```

Question 6 (2 points)

Calculate the total number of clicks product 1 received over the entire 6 weeks.

Provide both the numerical answer and the R command required to obtain the answer (if the dataframe is assigned to `df`).

Answer:

```
sum(df$clicks[df$product_id == 1])
```

```
[1] 29004
```

Question 7 (2 points)

Create a scatter plot with `ranking` on the horizontal axis and `clicks` on the vertical axis. Make the color of the points vary with the product's price.

Based on your plot, answer the following 2 questions.

Part (a): Choose the correction option from the following:

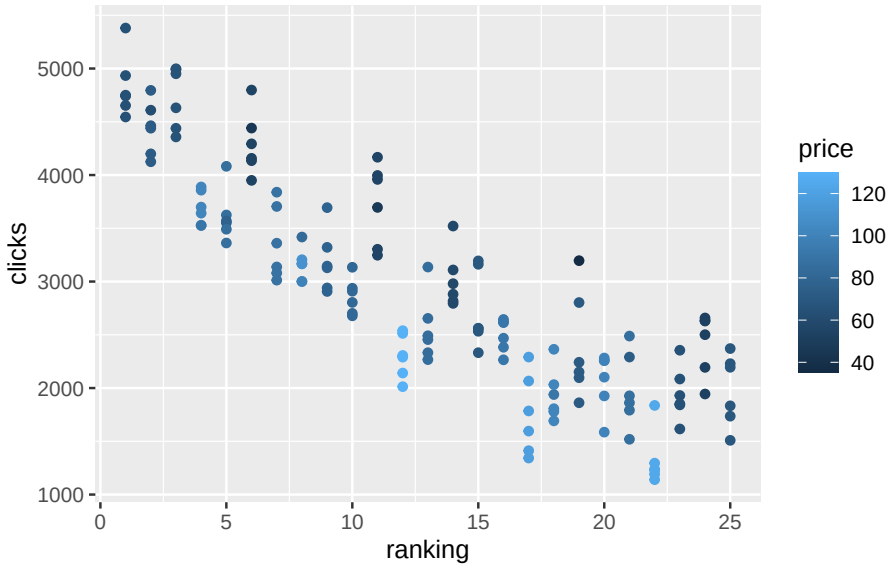
- Products higher on the webpage on average receive *more* clicks than products further down the webpage.
- Products higher on the webpage on average receive *fewer* clicks than products further down the webpage.
- Products higher on the webpage on average receive *the same number* clicks compared products further down the webpage.
- The optimal position on the webpage to receive the most clicks is roughly *half-way* down the webpage.

Part (b): Choose the correction option from the following:

- Products with a higher price on average receive *more* clicks.
- Products with a higher price on average receive *fewer* clicks.

Answer: We first make the plot:

```
library(ggplot2)
ggplot(df, aes(ranking, clicks, color = price)) + geom_point()
```



We can see that when ranking is low (higher on the webpage), clicks is high and when ranking is high (lower on the webpage), clicks is low. Therefore the correct option for the first question is: "Products higher on the webpage on average receive *more* clicks than products further down the webpage."

Note that the first question could be answered without making the color vary by price.

Looking at the colors for price, we can see that the brightest colors (the products with the highest prices) are below the general trend between ranking and clicks (look at ranking 12, 17 and 22), whereas the darkest colors (the products with the lowest prices) are above the general trend between ranking and clicks (look at ranking 6, 11 and 14). This tells us that products with higher prices on average receive *fewer* clicks, given the ranking.

Question 8 (1 point)

Write an R command in the box below using the `aggregate()` function that returns the total number of clicks for each product.

Answer:

```
aggregate(clicks ~ product_id, data = df, FUN = sum)
```

```
  product_id clicks
1          1  29004
2          2  26631
```

```
3            3  28369
4            4  22142
5            5  21681
6            6  25774
7            7  20134
8            8  18957
9            9  19136
10          10  17163
11          11  22368
12          12  13802
13          13  15335
14          14  18103
15          15  16342
16          16  14998
17          17  10493
18          18  11612
19          19  14351
20          20  12412
21          21  11879
22          22   7931
23          23  11680
24          24  14560
25          25  11873
```

Question 9 (1 point)

Write an R command in the box below to reshape the data such that there are 25 rows, one for each product, and the columns are:

- The product ID.
- The number of clicks for the product in week 1.
- The number of clicks for the product in week 2.
- The number of clicks for the product in week 3.
- The number of clicks for the product in week 4.
- The number of clicks for the product in week 5.
- The number of clicks for the product in week 6.

The output should be the following:

```
   product_id    1    2    3    4    5    6
1           1 4934 4545 4742 5380 4653 4750
2           2 4440 4463 4609 4125 4795 4199
3           3 4439 4951 4358 4998 4631 4992
4           4 3858 3699 3529 3642 3888 3526
5           5 3552 3625 3569 3491 4082 3362
6           6 4157 4441 4293 4135 3950 4798
7           7 3080 3137 3839 3013 3705 3360
```

```
8           8 3205 3167 3000 3001 3418 3166
9           9 3145 2940 3321 2907 3129 3694
10         10 2701 2937 2909 3134 2804 2678
11         11 3996 3247 3696 4167 3303 3959
12         12 2141 2539 2513 2306 2013 2290
13         13 3136 2455 2267 2654 2332 2491
14         14 3521 2795 3109 2881 2980 2817
15         15 2332 3162 2562 2533 3194 2559
16         16 2614 2383 2644 2624 2468 2265
17         17 2066 2291 1596 1343 1785 1412
18         18 1779 2364 1692 1806 1939 2032
19         19 2097 1862 3196 2151 2242 2803
20         20 1586 2258 1926 2258 2282 2102
21         21 1927 2291 2488 1792 1519 1862
22         22 1140 1295 1837 1191 1237 1231
23         23 2085 1852 1616 2355 1842 1930
24         24 2658 1944 2501 2194 2634 2629
25         25 2195 1833 2229 2371 1736 1509
```

Hint: Load the `reshape2` package using the command `library(reshape2)`. You do not need to include loading this package in your answer.

Answer:

```r
library(reshape2)
dcast(df, product_id ~ week, value.var = "clicks")
```

```
   product_id    1    2    3    4    5    6
1           1 4934 4545 4742 5380 4653 4750
2           2 4440 4463 4609 4125 4795 4199
3           3 4439 4951 4358 4998 4631 4992
4           4 3858 3699 3529 3642 3888 3526
5           5 3552 3625 3569 3491 4082 3362
6           6 4157 4441 4293 4135 3950 4798
7           7 3080 3137 3839 3013 3705 3360
8           8 3205 3167 3000 3001 3418 3166
9           9 3145 2940 3321 2907 3129 3694
10         10 2701 2937 2909 3134 2804 2678
11         11 3996 3247 3696 4167 3303 3959
12         12 2141 2539 2513 2306 2013 2290
13         13 3136 2455 2267 2654 2332 2491
14         14 3521 2795 3109 2881 2980 2817
15         15 2332 3162 2562 2533 3194 2559
16         16 2614 2383 2644 2624 2468 2265
17         17 2066 2291 1596 1343 1785 1412
18         18 1779 2364 1692 1806 1939 2032
```

```
19          19 2097 1862 3196 2151 2242 2803
20          20 1586 2258 1926 2258 2282 2102
21          21 1927 2291 2488 1792 1519 1862
22          22 1140 1295 1837 1191 1237 1231
23          23 2085 1852 1616 2355 1842 1930
24          24 2658 1944 2501 2194 2634 2629
25          25 2195 1833 2229 2371 1736 1509
```

Data Cleaning (4 points)

Download the dataset sales-oct-2025.csv. The dataset contains information on total number of units sold for a store throughout October 2025. The variables are:

- `date`: The date in US format (month-day-year).
- `day_of_week`: The (abbreviated) day of the week.
- `sales`: The total number of units sold on that day.

When reading the dataset into R, assign it to `df`.

Question 10 (1 point)

Write an R command in the box below using the `as.Date()` function that will correctly format the `date` variable to an R date.

```r
df <- read.csv("sales-oct-2025.csv")
df$date <- as.Date(df$date, format = "%m/%d/%y")
```

Question 11 (3 points)

Perform the following cleaning steps:

- Part (a): Write an R command in the box below that will replace the values of `sales` on Saturdays and Sundays with the value `NA` (i.e. replace `"Closed"` with `NA`).
- Part (b): Write an R command in the box below that will convert the variable `sales` from character to numeric.
- Part (c): Write an R command in the box below that will drop all rows in `df` will missing observations. The resulting dataframe should have 23 rows.

Answer:

```r
df$sales[df$day_of_week %in% c("Sat", "Sun")] <- NA
df$sales <- as.numeric(df$sales)
df <- na.omit(df)
```

Optimization (3 points)

The following 3 questions will involve working with the following mathematical function defined over all real numbers x:

$$f(x) = 10 + 4x - 2x^2$$

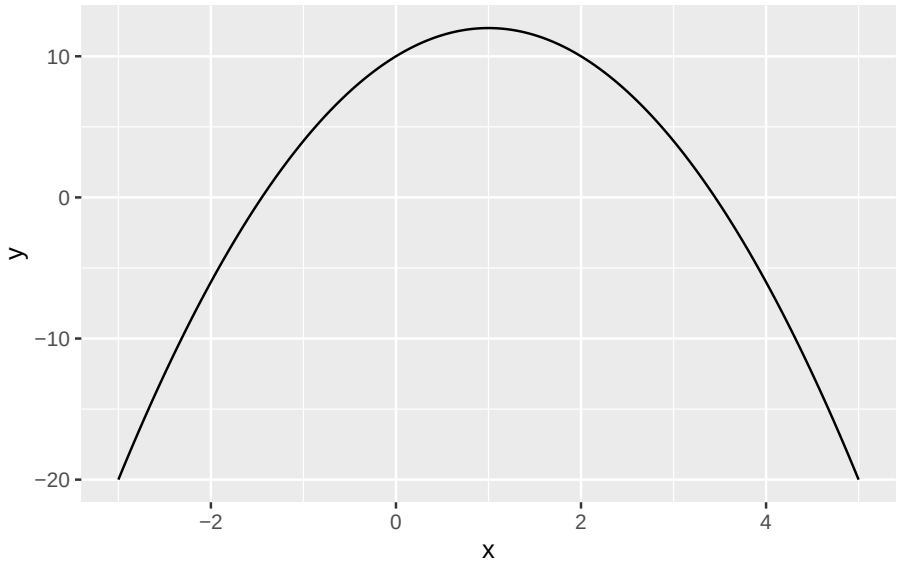
Question 12 (1 point)

Plot the function between the x values -3 and $+5$. Choose the answer below which best describes the shape of this function:

- Straight line
- Flat
- U shape
- Inverted U shape (upside-down U)

Answer:

```
f <- function(x) {
  y <- 10 + 4*x - 2 * x^2
  return(y)
}
library(ggplot2)
x <- seq(-3, 5, length.out = 2000)
y <- f(x)
df <- data.frame(x, y)
ggplot(df, aes(x, y)) + geom_line()
```




```
# We can see that it has an inverted U shape.
```

Question 13 (1 point)

Use R to find the value of x at an extreme point of this function.

Type this value of x in the box below.

Answer:

```
f_max <- optimize(f, c(-100, 100), maximum = TRUE)
f_max$maximum
```

```
[1] 1
```

Question 14 (1 point)

What value does the function take at the extreme point?

Answer:

```
f_max$objective
```

```
[1] 12
```

```r
# or alternatively:
f(f_max$maximum)
```

```
[1] 12
```