

Stats 20 Midterm Exam

Version A

J.Sparks

Introduction to Statistical Programming with R
UCLA

Instructions: Please turn your phone into airplane mode (or turn it off) and place it in your bag. You will have 65 minutes to complete this exam.

Please write your responses for the **free response questions** directly on the blank pages and mark each page with your name and student ID number.

Mark your answers to all **multiple choice** questions on the scantron provided and make sure you have as well as your name and student ID number on the file. Choose the one best option offered and mark your choice clearly on the provided scantron; ambiguous responses will be given 0 points.

There are 5 free response questions and a series of multiple choice questions worth 2 point each for a **total of 100 points**. (This practice exam has a few more for extra practice)

You hereby understand and agree to abide by the instructions above and UCLA's policy on academic integrity. All work you submit is a product of your own honest effort.

All questions on this exam should be done using only functions or syntax discussed in the lecture notes, homework, or required reading. No credit will be given for use of outside functions. At the end of this exam you will find a list of allowed functions.

Academic Misconduct: Any potential violation of UCLA's policy on academic integrity will be reported to the Office of the Dean of Students. All work on this exam must be your own. You may be recorded before, during, and after the exam.

Note: Any and all course material may not be posted online or shared with anyone at any time without explicit written permission by the instructor (Joshua Sparks), even after the quarter is over. Failure to comply is a breach of academic integrity.

By signing my name below, I hereby understand and agree to abide by the instructions above and UCLA's policy on academic integrity. All work I submit is a product of my own honest effort.

Name (PRINT CLEARLY):

UID:

Section (Circle One): Lecture 1 Lecture 2

Signature:

(Turn over when exam starts)

Free Response Questions

Problem 1

(16 points total) A small weather station tracks air quality data for 7 consecutive days. The data is stored in the following vectors:

```
date <- c("2025/4/18", "2025/4/19", "2025/4/20", "2025/4/21", "2025/4/22",
          "2025/4/23", "2025/4/24")
pm25 <- c(NA, 35, 60, 40, 20, NA, 30) # PM2.5 concentration in  $\mu\text{g}/\text{m}^3$ 
wind_speed <- c(5, 3, 4, NA, 6, 2, NA) # Wind speed in km/h
alert_day <- c(TRUE, FALSE, TRUE, FALSE, FALSE, TRUE, FALSE) # Was a national weather alert issued?
```

Each index corresponds to a single day. So `date[1]`, `pm25[1]`, `wind_speed[1]`, and `alert_day[1]` describe Day 1.

(a) (3 points) Write code that identifies the dates with missing PM2.5 measurements. Store these dates in a vector called `missing_pm25_dates`.

```
# SOLUTION
missing_pm25_dates <- date[is.na(pm25)]
missing_pm25_dates
```

```
[1] "2025/4/18" "2025/4/23"
```

(b) (3 points) Write R code to compute the mean PM2.5 level, ignoring missing values.

```
# SOLUTION
mean(pm25, na.rm = TRUE)
```

```
[1] 37
```

(c) (6 points) Are days with national weather alerts more likely to have missing PM2.5 measurements? Write R code to compute the proportions of missing PM2.5 measurements on alert and non-alert days and compare which proportion is higher.

```
# SOLUTION
prop_miss_alert <- mean(is.na(pm25[alert_day]))
prop_miss_alert
```

```
[1] 0.6666667
```

```
prop_miss_non_alert <- mean(is.na(pm25[!alert_day]))
prop_miss_non_alert
```

```
[1] 0
```

```
prop_miss_alert > prop_miss_non_alert
```

```
[1] TRUE
```

(d) (4 points) Write code to compute the number of days with complete data (no NAs in either `pm25` or `wind_speed`)?

```
# SOLUTION
sum(!(is.na(pm25) | is.na(wind_speed)))
```

```
[1] 3
```

```
sum(!is.na(pm25) & !is.na(wind_speed))
```

```
[1] 3
```

The famous (Fisher's or Anderson's) iris data set gives the measurements in centimeters of the sepal (green and leaflike leaves surrounding the colorful part of the flower) length and width and petal (colorful "leaves" of a flower) length and width in centimeters, respectively, for 50 flowers from each of 3 species of iris. The species are *Iris setosa*, *Iris versicolor*, and *Iris virginica*.

Problem 2

(14 points total) Consider the following vectors, which contain various measurements on these 150 iris flowers (only the first 8 elements of each vector are shown for brevity):

```
# show only first eight entries of each vector
```

```
sepal_length[1:8]
```

```
[1] 5.1 4.9 4.7 4.6 5.0 5.4 4.6 5.0
```

```
sepal_width[1:8]
```

```
[1] 3.5 3.0 3.2 3.1 3.6 3.9 3.4 3.4
```

```
petal_length[1:8]
```

```
[1] 1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5
```

```
petal_width[1:8]
```

```
[1] 0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2
```

```
# first 50 entries are setosa, entries 51:100 are versicolor, entries 101:150 are virginica
```

```
species[c(1, 50, 51, 100, 101, 150)]
```

```
[1] setosa      setosa      versicolor versicolor virginica  virginica
```

```
Levels: setosa versicolor virginica
```

```
length(species)
```

```
[1] 150
```

You can assume that all of the vectors have the same length and there are no missing values. The i th entry in each vector corresponds to the same flower, i.e., the first flower is of Species "*Iris setosa*", has a sepal length of 5.1cm, a sepal width of 3.5cm, a petal length of 1.4cm, and a petal width of 0.2cm.

(a) (6 points) Create two vectors called `setosa_sepal_length` and `versicolor_sepal_length` containing the sepal lengths of all iris setosa flowers and iris versicolor flowers in the data set respectively.

```
# SOLUTION
# using a numeric index
setosa_sepal_length <- sepal_length[1:50]
versicolor_sepal_length <- sepal_length[51:100]
# OR
# using a logical index
setosa_sepal_length <- sepal_length[species == "setosa"]
versicolor_sepal_length <- sepal_length[species == "versicolor"]
```

(b) (5 points) Compute the test statistic t for an unpaired samples t-test for two means in R to test whether the average sepal length differs between the two species iris setosa and iris versicolor? Hint: Here is the formula:

$$t = \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}},$$

where $\bar{x}_i$ is the sample mean of the i th sample and s_i is the sample standard deviation of the i th sample, $i = 1, 2$.

```
# SOLUTION
t <- (mean(setosa_sepal_length) - mean(versicolor_sepal_length)) /
  sqrt(sd(setosa_sepal_length)^2 / length(setosa_sepal_length) +
        sd(versicolor_sepal_length)^2 / length(versicolor_sepal_length))
t
```

```
[1] -10.52099
```

(c) (3 points) You compute your test statistic $t = -10.52$. Based on this value, does there appear to be evidence that a null hypothesis of “no difference in average sepal lengths between the two species” is false? Explain your reasoning.

<- SOLUTION: ->

<- This test statistic is more than 10 standard errors from the hypothesized mean of zero; this suggests that the null hypothesis of $\mu_1 = \mu_2$ is false ->

Problem 3

(12 points total) Counting missing values.

(a) (4 points) Write code that counts the number of NA values (do **NOT** count NaNs) in a vector `x` defined as

```
# Hint: there are three NAs in this vector
x <- c(NA, -1, 0, 1, NA, NaN, NA, Inf, NaN, NaN, 1, 10)
```

and stores it in a new object called `n_NAs`:

```
# SOLUTION
n_NAs <- sum(is.na(x)) - sum(is.nan(x))
```

(b) (3 points) Assuming an existing object `n_NAs` in your working directory that contains the correct number of missing values in `x`, write a command in R that will count the number of values that are **NOT** NAs in the same vector `x`.

```
# SOLUTION
length(x) - n_NAs
```

```
[1] 9
```

```
length(x) - sum(is.na(x)) + sum(is.nan(x))
```

```
[1] 9
```

```
length(x) - (sum(is.na(x)) - sum(is.nan(x)))
```

```
[1] 9
```

(c) (5 points) Assuming an existing object `n_NAs` in your working directory that contains the correct number of NAs in `x`, write code that computes the test statistic z_{obs} for a one-proportion test defined as

$$z_{obs} = \frac{p - p_0}{\sqrt{\frac{p_0(1-p_0)}{n}}}$$

testing the Null hypothesis that the **proportion** of NAs is equal to 0.5 ($H_0 : p = 0.5$) against the alternative hypothesis that it is not ($H_a : p \neq 0.5$).

Note: You do not actually have to compute the value of the test statistic, just write the R code that will compute it.

Hint: I recommend defining the variables p , p_0 , and n as a first step. Then implement the formula for z_{obs} using these variables.

```
# SOLUTION
n <- length(x)
p <- n_NAs / n
p0 <- 0.5
z_obs <- (p - p0) / sqrt(p0 * (1 - p0) / n)
z_obs
```

```
[1] -1.732051
```

Problem 4

(14 points total) Consider the following vector of measurements, in millimeters:

```
x_mm <- c(74, 86, 91, 90, 67)
```

- (a) (3 Points) Suppose I want to transform this vector from millimeters to meters (so, divide by 1000). Write the code for the new vector (`x_m`) using R's syntax for scientific notation

```
#solution
x_m <- x_mm/1e3
```

- (b) (3 Points) In R, if we were print element `x_m[2]` out to twenty digits to the right of the decimal, would this value read 0.08600000000000000000 (or just 0.086)? Explain why or why not, directly mentioning how our personal number system differs from the number system R uses.

<- SOLUTION: ->

<- Likely not; by dividing my 1000, the floating point arithmetic (of using base 2 instead of base 10) we are creating a small rounding error, and so the printed part is slightly different->

```
#proof
print(x_m[2], digits=20)
```

```
[1] 0.085999999999999993117
```

- (c) (4 Points) Examine the R output below.

```
x_mm*(34/1000)/34==x_m
```

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

Explain briefly why this output occurred. Use precise language used in class.

<- SOLUTION: ->

<- Similar to above; by dividing by 34 (and multiplying by 34 to cancel it out), the rounding error causes the values to be slightly different->

```
#solution -- see elements 2 and 3
print(x_mm*(34/1000)/34, digits=20)
```

```
[1] 0.073999999999999996336 0.0860000000000000006994 0.091000000000000011435
```

```
[4] 0.089999999999999996669 0.0670000000000000003997
```

```
print(x_m, digits=20)
```

```
[1] 0.073999999999999996336 0.085999999999999993117 0.09099999999999997558
```

```
[4] 0.089999999999999996669 0.0670000000000000003997
```

- (d) (4 Points)

Suppose we wish to take the fifth element of `x_mm` and raise it to the 10000th power as such:

```
x_mm[5]^10000
```

What should the output be and explain your reasoning, including if it doesn't align "mathematically".

<- SOLUTION: ->

<- While this value is finite, it is extremely large, so large that it is beyond the computing power of R, and so it reads as Inf->

Problem 5

(24 points total) Your friend is about to hand in their homework where they implemented the following variance. It works, but it is a bit of a mess and does not make much sense as to what is happening.

```
numeric <- c(1,3,2,5,6)
MyValues=numeric
a=(length(MyValues)-1)
b=1/a; b2=mean(numeric)
c=(b2 - MyValues)
  cp2=c^2
cp2s=sum(cp2)
  Fin_res= cp2s*1 / a
# believe it or not, this actually works
Fin_res
```

[1] 4.3

- (a) **(10 Points)** To begin, translate the steps made in the above code into a pseudo-code that explains the process fully.

<- SOLUTION: ->

<-

- (1) Create the vector numeric, based on the values above.
- (2) Set MyValues=numeric
- (3) Define the object a to be one less than the length of the vector
- (4) Take the reciprocal of a and name it b
- (5) Make b2 to be the mean of the numeric vector
- (6) Define c as the difference of b2 and MyValues
- (7) Call cp2 to be the square of object c
- (8) Take the sum of cp2 and name is cp2s
- (9) Compute Fin_res as cp2s times 1 divided by a
- (10) Print out Fin_res

->

- (b) **(12 Points)** Please help them rewrite this code that provides clear names of the objects defined and includes commenting that guides the process. Fix at least 6 mistakes and comment them (there are way more than that) for full credit.

```
# SOLUTION
num <- c(1, 3, 2, 5, 6)
n_minus_1 <- (length(num)-1)
xbar <- mean(num)
deviations <- (xbar - num)
deviations_squared <- deviations^2
sum_deviations_squared <- sum(deviations_squared)
variance <- sum_deviations_squared * 1 / n_minus_1
variance
```

[1] 4.3

- (c) **(2 Points)** Explain briefly (1-2 short sentences) why this two-pass formula provided is preferred over $s^2 = \frac{1}{n-1}(\sum_{i=1}^n (x_i^2) - n\bar{x}^2)$.

<- SOLUTION: ->

<- With the one-pass formula, we are more likely to get wrong answers when our observations are really large or really small->

Multiple Choice Questions

Mark your answers to all multiple choice questions on the exam provided. Each multiple choice question is worth 1 point.

Note: While there are more than 20 questions here (adding up to more than 100 points), it is only to give you more practice. Overall, you should expect only 100 points on the exam.

Problem 1

What is the output of the following code:

```
x <- -4:4
x[x < 0 | x > 0]
```

- A. [1] -4 -3 -2 -1 1 2 3 4
- B. [1] -4 -3 -2 -1 0 1 2 3 4
- C. [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
- D. integer(0)
- E. An error message.

<- SOLUTION: A->

<- All cases except for zero work here.->

Problem 2

What is the output of the following code:

```
x <- -4:4
x[x < 0 & x > 0]
```

- A. [1] -4 -3 -2 -1 1 2 3 4
- B. [1] -4 -3 -2 -1 0 1 2 3 4
- C. [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
- D. integer(0)
- E. An error message.

<- SOLUTION: D ->

<- there is no number between -4 and 4, inclusively, which satisfies those conditions ->

Problem 3

What is the output of

```
x <- c(-1, 0, 1, 2, NA)
y <- c(-1, 0, NA, 2, 3)
x | y
```

- A. [1] TRUE
- B. [1] TRUE FALSE TRUE TRUE TRUE
- C. [1] TRUE TRUE TRUE TRUE TRUE
- D. [1] TRUE FALSE NA TRUE NA

E. [1] TRUE FALSE TRUE TRUE NA

<- SOLUTION: B ->

<- When using “or”, non-zero values become TRUE; as long as one element is TRUE in the or test, the outcome is TRUE, even with NA’s ->

Problem 4

What is the output of

```
x <- c(-1, 0, 1, 2, NA)
y <- c(-1, 0, NA, 2, 3)
x & y
```

A. [1] TRUE

B. [1] TRUE FALSE TRUE TRUE TRUE

C. [1] TRUE TRUE TRUE TRUE TRUE

D. [1] TRUE FALSE NA TRUE NA

E. [1] TRUE FALSE TRUE TRUE NA

<- SOLUTION: D ->

<- Similar to above, but if one element is TRUE and another is NA, our outcome is NA because we don’t know whether or not the missing value is TRUE; if one is FALSE, however, the entire test result is FALSE for that element. ->

Problem 5

What is the output of

```
x <- c(-1, 0, 1, 2, NA)
y <- c(-1, 0, NA, 2, 3)
x[!is.na(y) & y > 0]
```

A. [1] 1 2

B. [1] 2 NA

C. [1] NA 2

D. [1] FALSE FALSE TRUE TRUE FALSE

E. [1] FALSE FALSE FALSE TRUE TRUE

<- SOLUTION: B ->

<- The fourth and fifth elements of y fit the conditions of !is.na(y) & y > 0 and the fourth and fifth elements of x are found in solution B ->

The following information is used in Problems Problem 6 and Problem 8.

Consider the following command:

```
x <- c(-3, 4 / 9, 2.5, 0, NA)
```

Problem 6

Which of the following commands will extract the number 0 from the vector $\mathbf{x}$?

- A. `x[4]`
- B. `x(4)`
- C. `x{4}`
- D. `x[-4]`
- E. `x(-4)`

<- SOLUTION: A->

<- Fairly straightforward. ->

Problem 7

Which of the following commands will extract everything but the number -3 from the vector $\mathbf{x}$?

- A. `x[-1]`
- B. `x[-c(-1)]`
- C. `x[-3]`
- D. `x[-c(-3)]`
- E. `x[4:length(x)]`

<- SOLUTION: A ->

<- Element 1 is (-3); so we want everything but element 1 ->

Problem 8

What is the output of the following command

```
x <- c(-3, 4 / 9, 2.5, 0, -1, NA)
length(x)
```

- A. [1] 0
- B. [1] 1
- C. [1] 5
- D. [1] 6
- E. [1] 7

<- SOLUTION: D ->

<- Even if NA, it counts toward an element in the vector. Only NULL would not ->

Problem 9

What is the output of the following command

```
x <- c(-3, 1, 2.2, 0, -1)
x[x]
```

- A. [1] 1.0 0.0
- B. [1] 1.0 0.0 -1.0

C. [1] -3.0 1.0

D. [1] -3.0 1.0 0.0 -1.0

E. An error message.

<- SOLUTION: E ->

<- The main issue: You can't have negative and positive indices within the same subsetting line ->

Problem 10

What is the output of the command

```
c(FALSE, 3, 1, TRUE)
```

A. [1] "FALSE" "3" "1" "TRUE"

B. [1] 0 3 1 1

C. [1] 1 3 1 0

D. [1] FALSE 3 1 TRUE

E. [1] FALSE TRUE TRUE TRUE

<- SOLUTION: B ->

<- You'll coerce the logical elements into 1's (TRUE) and 0's (FALSE) when combining with numerical elements (and character elements are missing) ->

Problem 11

What is the output of the command

```
sum(is.na(c(TRUE, FALSE, NA, NA, NaN)))
```

A. [1] 1

B. [1] 2

C. [1] 3

D. [1] 4

E. [1] 5

<- SOLUTION: C ->

<- Remember: NaN is a type of NA, so it gets counted.->

Problem 12

Which of the following would **NOT** be a valid name for an object in R?

A. ILoveStats20

B. i.love.stats.20

C. 1L0v3Stats20

D. i_love_stats_20

E. i10v3stats20

<- SOLUTION: C ->

<- You can't start with a number ->

Problem 13

What is the output of the following code:

```
obj <- c("this", "is", "content")
obj[length(obj) + 1] <- c("this is more content")
obj
```

- A. [1] "this" "is" "content"
- B. [1] "this" "is" "content" "this is more content"
- C. [1] "this" "is" "more" "content" "this" "is" "more" "content"
- D. [1] "this" "is" "more" "content"
- E. An error message.

<- SOLUTION: B ->

<- "this is more content" gets added to the next element ->

Problem 14

What is the output of the following code

```
x1 <- c(0, 1, 2)
x2 <- c(0, -1, 2)
x1 * x2
```

- A. [1] -5
- B. [1] 0 1 -0.5
- C. [1] 0 -1 4
- D. [1] 0 0 0 0 -1 -2 0 2 4
- E. [1] 0 0 0 0 -1 2 0 -2 4

<- SOLUTION: C ->

<- Keep in mind how vector arithmetic works in R ->

Problem 15

What is the output of the following code

```
x1 <- c(0, 1, 2, 3)
x2 <- c(0, -1, 2)
x1 * x2
```

- A. [1] 0 -1 4 with a warning message
- B. [1] 0 -1 4 without a warning message
- C. [1] 0 -1 4 0 with a warning message
- D. [1] 0 -1 4 0 without a warning message
- E. An error message

<- SOLUTION: C ->

<- 4 (elements) is not a multiple of 3 (elements), so we'll have a warning; note that we'll have recycling going on. ->

Problem 16

What is the output of the command

```
rep(c(3, 1, 4), times = 2)
```

- A. [1] 3 1 4 2
- B. [1] 3 3 1 1 4 4
- C. [1] 3 1 4 3 1 4
- D. [1] 3 1 4 2 3 1 4 2
- E. An error message.

<- SOLUTION: C ->

<- Recall that the times argument will replicate the vector multiple times ->

Problem 17

What is the output of the command

```
rep(c(3, 1, 4), each = 2)
```

- A. [1] 3 1 4 2
- B. [1] 3 3 1 1 4 4
- C. [1] 3 1 4 3 1 4
- D. [1] 3 1 4 2 3 1 4 2
- E. An error message.

<- SOLUTION: B ->

<- The each argument replicates the element that many times, where the replications occur element-by-element within the process ->

Problem 18

You want to randomly assign **exactly** half of your 50 subjects to the treatment group and the other half to the control group. How can you create a character vector that contains each subject's randomly assigned group?

- A. `sample(x = c("treatment", "control"), size = 50, replace = FALSE)`
- B. `sample(x = c("treatment", "control"), size = 50, replace = TRUE)`
- C. `sample(x = rep(c("treatment", "control"), each = 25), size = 50, replace = FALSE)`
- D. `sample(x = rep(c("treatment", "control"), each = 25), size = 50, replace = TRUE)`
- E. `rep(c("treatment", "control"), each = 25)`

<- SOLUTION: C ->

<- This is one of the trickier questions, and one that is pretty applicable to what they might do in actual coding for randomization, and we'll break this down. (1) We need to randomly assign the values, so we need to use the `sample()` function (2) We need to randomly assign fifty groups, 25 of control and 25 of treatment. (3) Our sample needs to be of size 50 (4) To make sure we get exactly 25 of each, from our 50 copies, we should sample without replacement.

Based on this criteria, only (C) will work ->

Problem 19

Which of the following will produce the same output as

```
-5 + 3:5 * 2
```

- A. $((-5 + 3):5) * 2$
- B. $(-5 + 3):(5 * 2)$
- C. $(-5 + (3:5)) * 2$
- D. $-5 + (3:(5 * 2))$
- E. $-5 + ((3:5) * 2)$

<- SOLUTION: E ->

<- Parentheses are key, and help visualize what is going on. Option E maintains the order of operation. ->

Problem 20

Which of the following is meant by

```
3.2 * exp(4)
```

- A. 3.2^4
- B. $3.2 \times e^4$
- C. $3.2 \times e^{-4}$
- D. 3.2×2^4
- E. 3.2×10^4

<- SOLUTION: B ->

<- Note that exp() gives us the mathematical e. ->

Problem 21

What is the output of

```
c(1, 3, 5) + c(1, 2)
```

- A. [1] 2 5 **without** a warning message
- B. [1] 2 5 with a warning message
- C. [1] 2 5 6 **without** a warning message
- D. [1] 2 5 6 with a warning message
- E. None of the above: R will throw an error message

<- SOLUTION: D ->

<- Recycling will happen, and the length of the longer vector is not a multiple of the length of the shorter vector, so we'll get a warning message ->

Problem 22

Which of the following does **NOT** produce the same output as

```
seq(3, 10, by = 2)
```

- A. `c(3L, 5L, 7L, 9L)`
- B. `(3:10)[c(TRUE, FALSE)]`
- C. `seq_along((3:10)[c(TRUE, FALSE)])`
- D. `(3:10)[(3:10) %% 2 == 1]`

<- SOLUTION: C ->

<- Remember that `seq_along()`, no matter what the original sequence looks like, starts at 1 and follows along the elements (think of it as labelling ID values for observations). So, `seq_along((3:10)[c(TRUE, FALSE)])=seq_along(c(3,5,7,9))= c(1,2,3,4) ->`

Problem 23

Which code generates the following matrix?

```
      [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
[2,]    1    2    3    4
[3,]    1    2    3    4
```

- A. `matrix(rep(1:4, each = 3), nrow = 3, ncol = 4)`
- B. `matrix(rep(1:4, each = 3), nrow = 4, ncol = 3)`
- C. `matrix(rep(1:4, times = 3), nrow = 3, ncol = 4)`
- D. `matrix(rep(1:4, times = 3), nrow = 4, ncol = 3)`
- E. `matrix(rep(1:4, times = 4), nrow = 4, ncol = 3)`

<- SOLUTION: A ->

<- Note that the exam only shows the matrix. It will read the values as columns unless you use `byrow=T`, and since we see three 1's along the column, three 2's, and so on, it means we want to use the sequence 1:4, but `rep()` with 3 each. Assure students that this question is a bit tricky and will likely use the `byrow=T` command in the future, if needed. ->

Problem 24

Consider the following matrix A

```
A
      [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
[2,]    5    6    7    8
[3,]    9   10   11   12
```

How would you extract the vector

```
[1] 11  7
```

from the matrix A?

- A. `A[2:3, 2]`
- B. `A[2:3, 3]`
- C. `A[3:2, 3]`
- D. `A[3, 2:3]`
- E. `A[3, 3:2]`

<- SOLUTION: C ->

<- Note that value 7 comes from position (2,3) and value 11 comes from position (3,3). To get 11 first, and then 7, we'll use only column 3, but in the order of row 3 and then row 2.

So, we use A[3:2, 3] ->

Problem 25

What is the output of the following lines of code:

```
x <- 1:6
y <- x > 3
y & c(TRUE, FALSE, NA)
```

[1] FALSE FALSE FALSE TRUE FALSE NA

A. [1] FALSE FALSE FALSE TRUE FALSE FALSE

B. [1] FALSE FALSE FALSE TRUE FALSE NA

C. [1] FALSE FALSE FALSE TRUE FALSE TRUE

D. [1] TRUE FALSE NA TRUE TRUE TRUE

E. [1] TRUE FALSE FALSE TRUE TRUE TRUE

<- SOLUTION: B ->

<- Many steps! Let's process what the vector y looks like first: y=c(FALSE, FALSE, FALSE, TRUE, TRUE, TRUE)

Note for an & to be TRUE, both elements must be TRUE. If you have an NA & TRUE, you get NA. If you have a FALSE anywhere, the output will be FALSE.

With recycling vector elements, our output will be B. ->

Congratulations, you have reached the end of the exam!