**DAY-4C**

1-CODE

# Load the iris dataset

data(iris)

# Set the seed for reproducibility

set.seed(123)

# Randomly sample the iris dataset

iris\_sample <- iris[sample(nrow(iris)),]

# Split the data into training and test sets

train <- iris\_sample[1:round(0.8\*nrow(iris\_sample)),]

test <- iris\_sample[(round(0.8\*nrow(iris\_sample))+1):nrow(iris\_sample),]

# Create a logistic regression model with the training data

model <- glm(Species ~ Petal.Length + Petal.Width, data = train, family = "binomial")

# Predict the probabilities of the model using the test set

probabilities <- predict(model, newdata = test, type = "response")

probabilities

# Convert probabilities to predicted species

predicted\_species <- ifelse(probabilities > 0.5, "versicolor", "setosa")

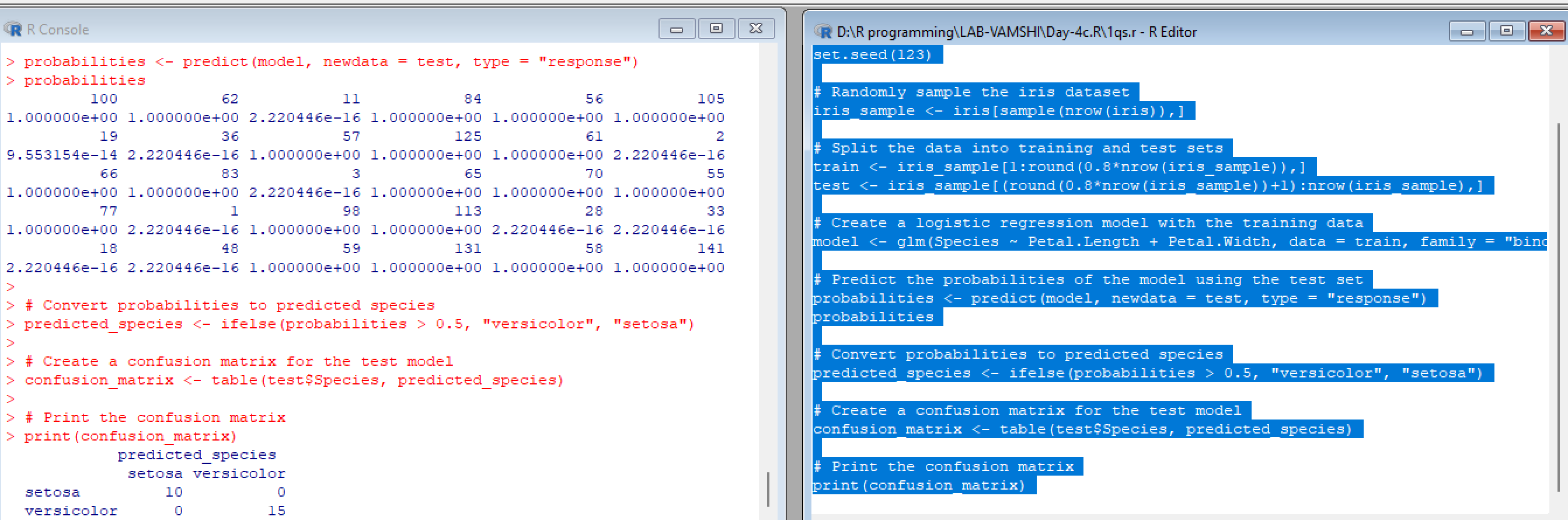
# Create a confusion matrix for the test model

confusion\_matrix <- table(test$Species, predicted\_species)

# Print the confusion matrix

print(confusion\_matrix)

OUTPUT



2-CODE

# Given values

x <- c(90, 50, 70, 80, 70, 60, 20, 30, 80, 90, 20)

# Compute the mean

mean\_x <- mean(x)

print(mean\_x)

# Compute the median

median\_x <- median(x)

print(median\_x)

# Compute the mode

mode\_x <- names(table(x))[table(x) == max(table(x))]

print(mode\_x)

# Given values

x <- c(90, 50, 70, 80, 70, 60, 20, 30, 80, 90, 20)

# Find the 2nd highest value

x\_sorted <- sort(unique(x), decreasing = TRUE)

second\_highest <- x\_sorted[2]

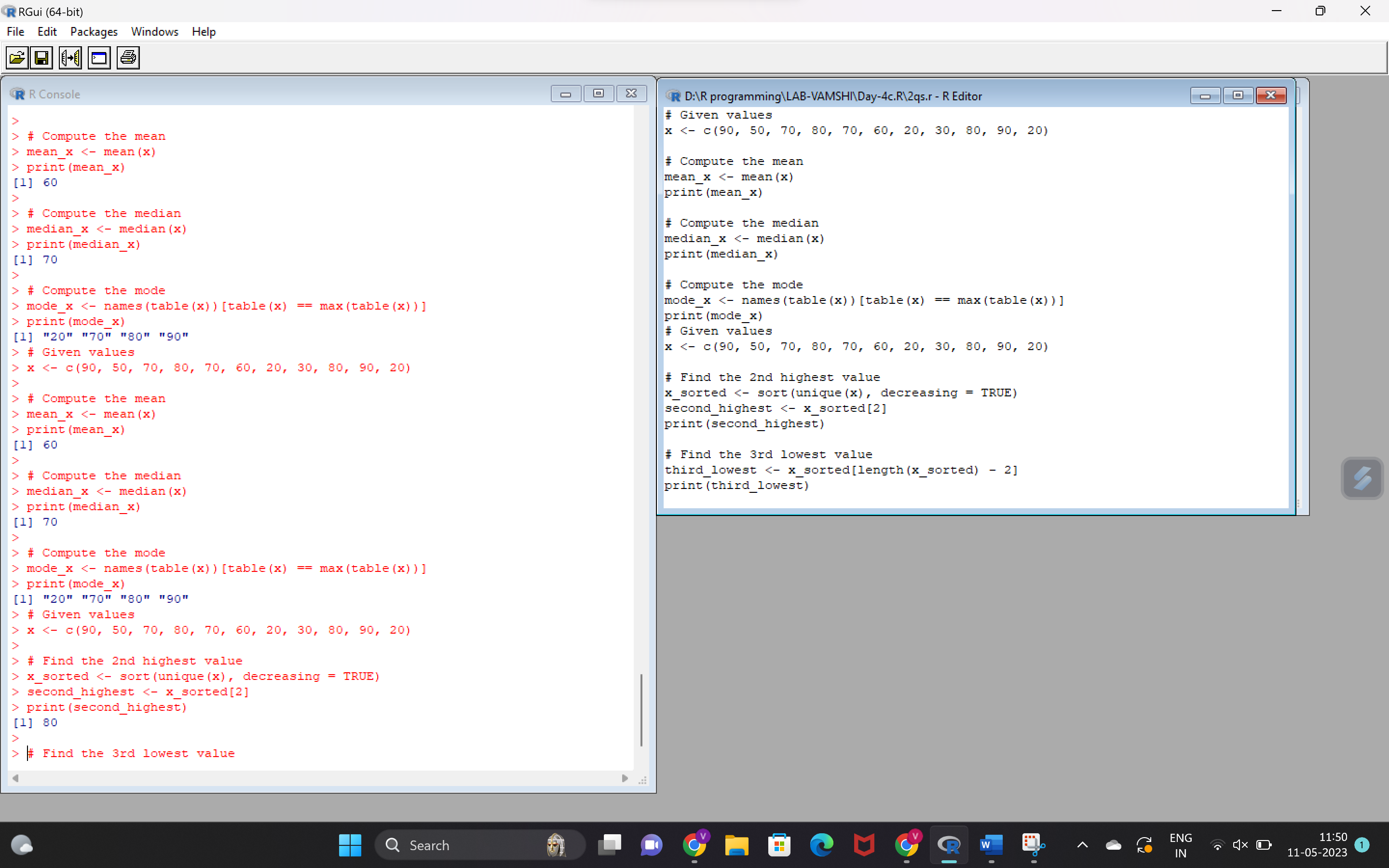
print(second\_highest)

# Find the 3rd lowest value

third\_lowest <- x\_sorted[length(x\_sorted) - 2]

print(third\_lowest)

OUTPUT:



3-CODE

# Load the airquality dataset

data(airquality)

# i. Compute the mean temperature

mean\_temp <- sum(airquality$Temp)/length(airquality$Temp)

print(mean\_temp)

# Alternatively, you can use the built-in mean() function:

mean\_temp2 <- mean(airquality$Temp, na.rm = TRUE)

print(mean\_temp2)

# ii. Extract the first five rows from airquality

first\_five\_rows <- airquality[1:5, ]

print(first\_five\_rows)

# iii. Extract all columns from airquality except Temp and Wind

all\_cols\_except\_temp\_wind <- airquality[, !(names(airquality) %in% c("Temp", "Wind"))]

print(all\_cols\_except\_temp\_wind)

# iv. Which was the coldest day during the period?

coldest\_day <- airquality[which.min(airquality$Temp), ]

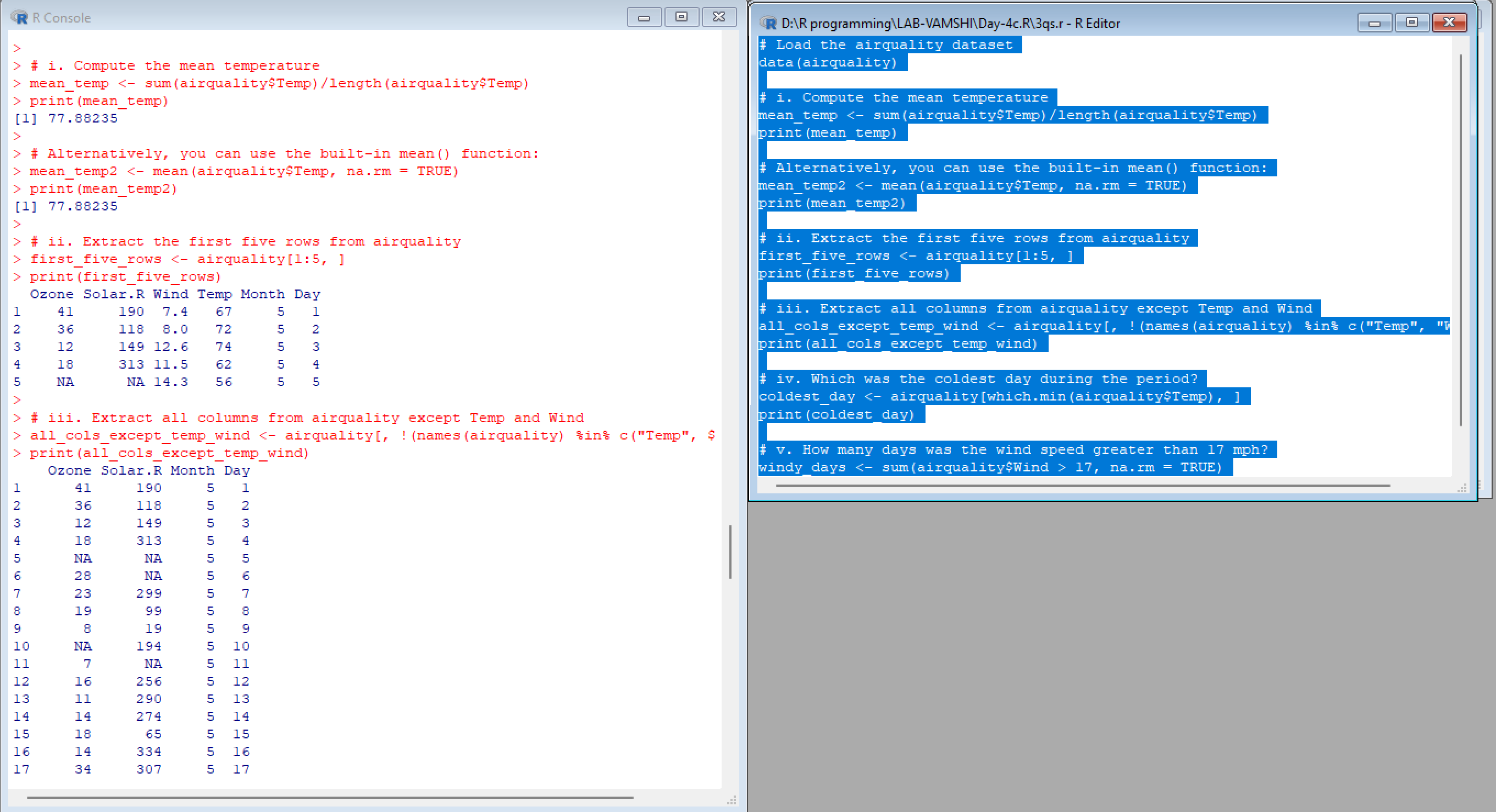
print(coldest\_day)

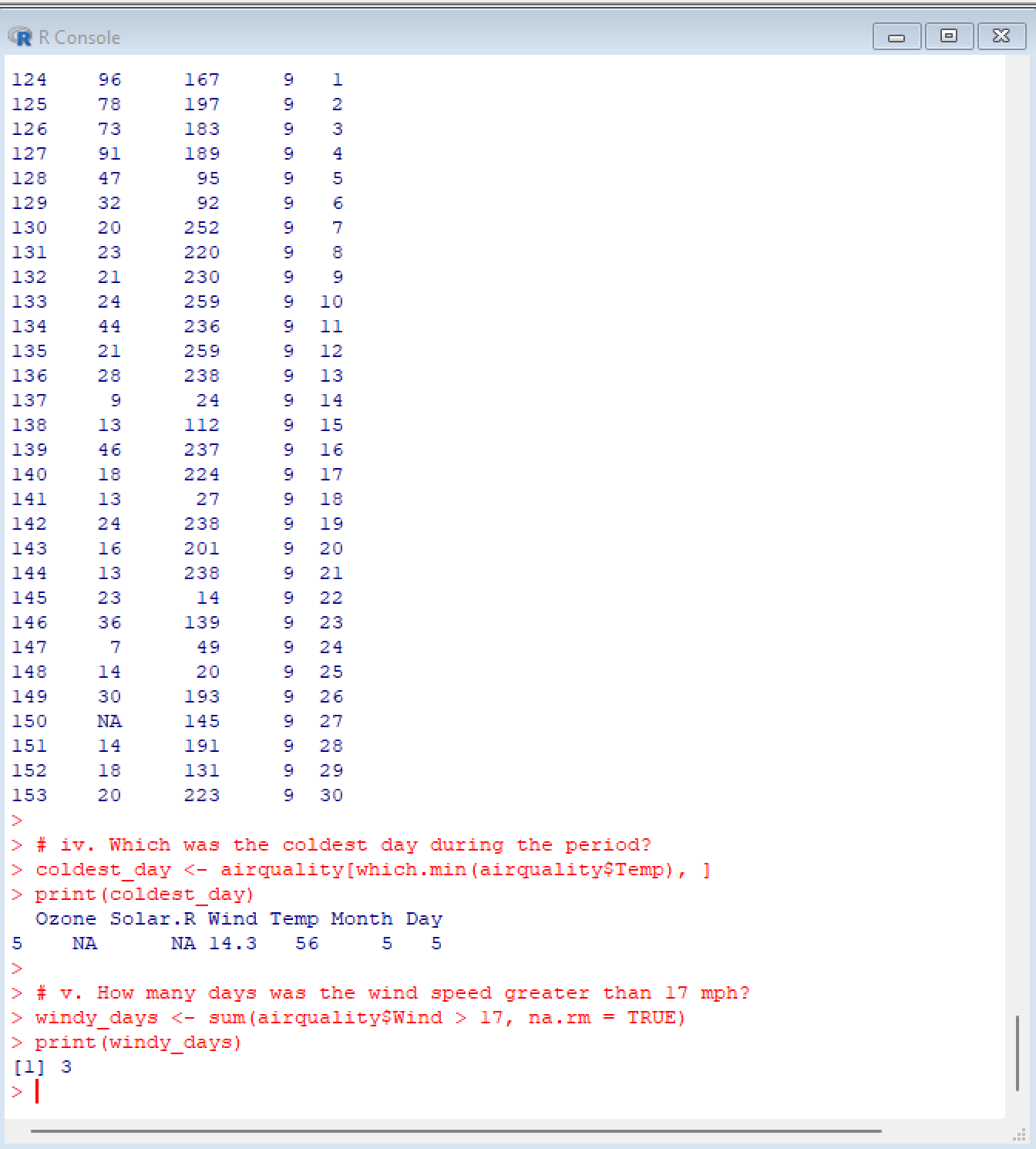
# v. How many days was the wind speed greater than 17 mph?

windy\_days <- sum(airquality$Wind > 17, na.rm = TRUE)

print(windy\_days)

OUTPUT:





4-CODE

data(airquality)

summary(airquality)

melted\_airquality <- melt(airquality, id.vars = c("Month", "Day"))

head(melted\_airquality)

melted\_airquality <- melt(airquality, id.vars = c("Month", "Day"))

head(melted\_airquality)

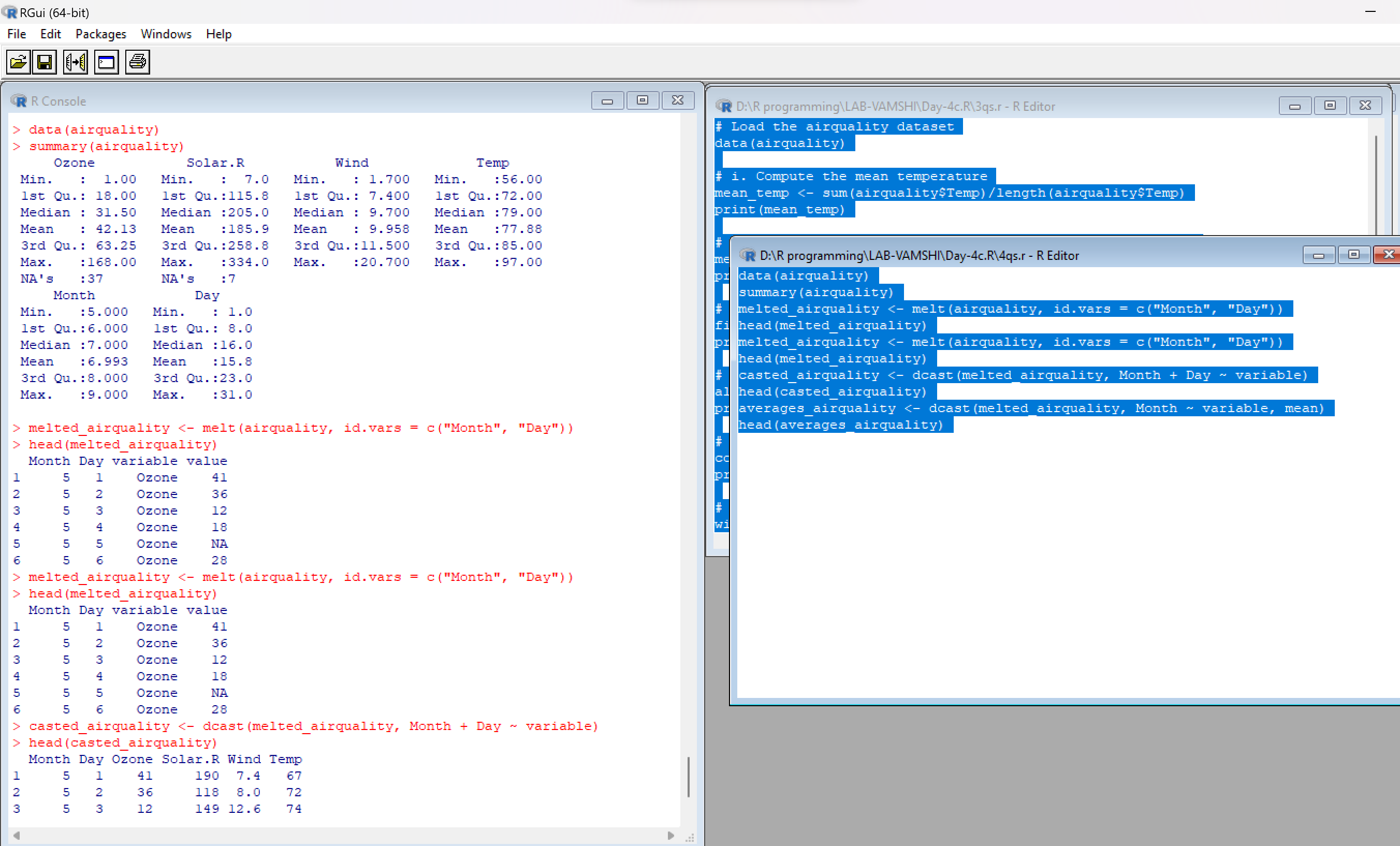
casted\_airquality <- dcast(melted\_airquality, Month + Day ~ variable)

head(casted\_airquality)

averages\_airquality <- dcast(melted\_airquality, Month ~ variable, mean)

head(averages\_airquality)

output



5-code

# Find missing values

missing\_values <- apply(is.na(airquality), 2, sum)

print(missing\_values)

# Drop missing values if it's less than 10%

n\_obs <- nrow(airquality)

max\_missing <- 0.1 \* n\_obs

for (col in colnames(airquality)) {

n\_missing <- sum(is.na(airquality[, col]))

if (n\_missing <= max\_missing) {

airquality <- na.omit(airquality)

} else {

airquality[is.na(airquality[, col]), col] <- mean(airquality[, col], na.rm = TRUE)

}

}

# Linear regression of Ozone and Solar.R

fit <- lm(Ozone ~ Solar.R, data = airquality)

summary(fit)

library(ggplot2)

# Scatter plot of Ozone and Solar.R

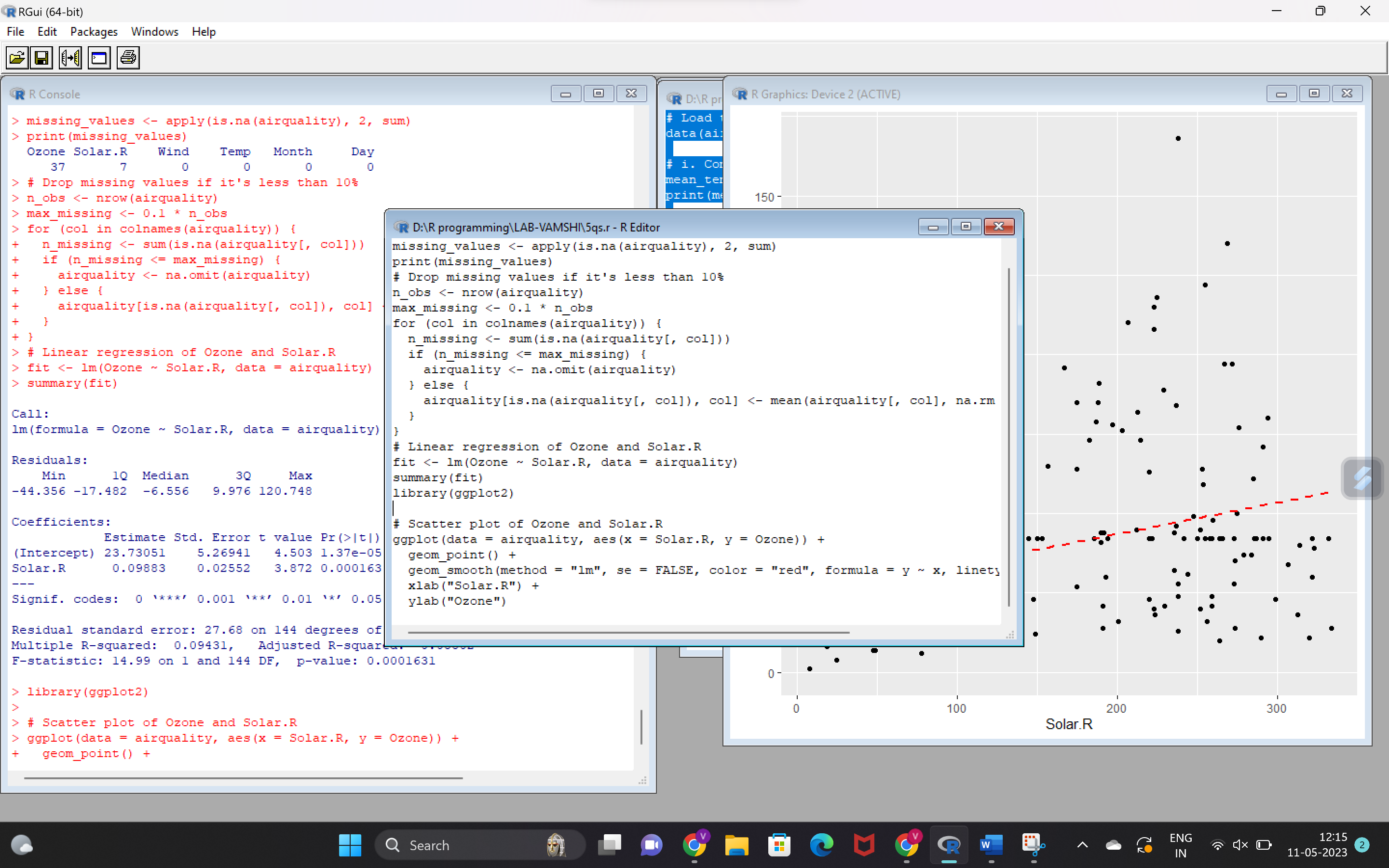
ggplot(data = airquality, aes(x = Solar.R, y = Ozone)) +

geom\_point() +

geom\_smooth(method = "lm", se = FALSE, color = "red", formula = y ~ x, linetype = "dashed", size = 1) +

xlab("Solar.R") +

ylab("Ozone")



6-code

# Load the dataset

data(ChickWeight)

# (i)

# Order the data frame in ascending order by feature name "weight" grouped by feature "diet"

ordered\_ChickWeight <- ChickWeight[order(ChickWeight$diet, ChickWeight$weight),]

# Extract the last 6 records from the ordered data frame

tail(ordered\_ChickWeight, 6)

# (ii)

library(reshape2)

# (ii) a

# Melting function based on "Chick", "Time", "Diet" features as ID variables

melted\_ChickWeight <- melt(ChickWeight, id.vars = c("Chick", "Time", "Diet"))

# (ii) b

# Cast function to display the mean value of weight grouped by Diet

casted\_ChickWeight\_mean <- dcast(melted\_ChickWeight, Diet ~ variable, mean)

casted\_ChickWeight\_mean

# (ii) c

# Cast function to display the mode of weight grouped by Diet

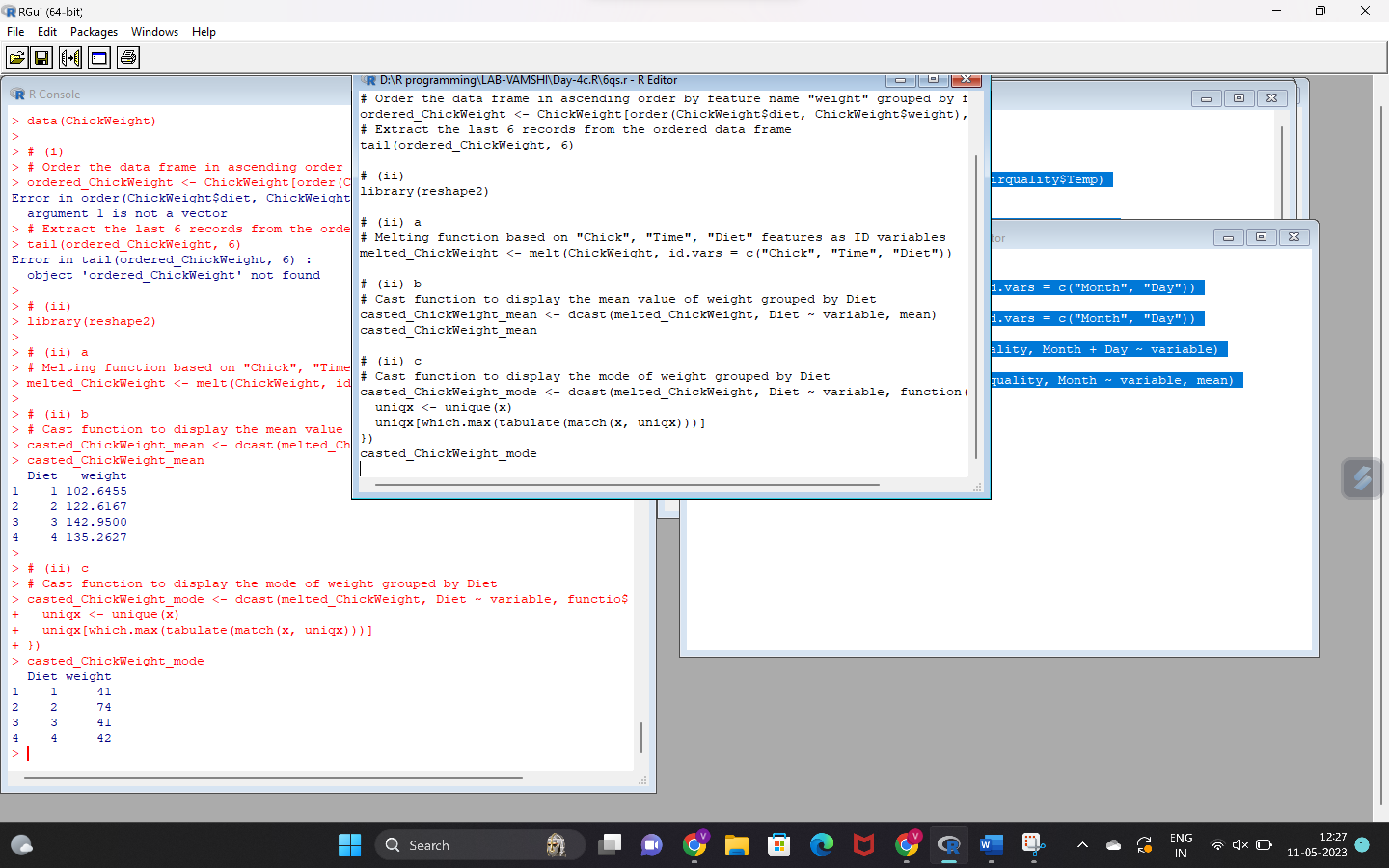
casted\_ChickWeight\_mode <- dcast(melted\_ChickWeight, Diet ~ variable, function(x) {

uniqx <- unique(x)

uniqx[which.max(tabulate(match(x, uniqx)))]

})

casted\_ChickWeight\_mode



Code-7