B9DA100 Exam - First sitting (all students)

John O'Sullivan

The answers to the B9DA100 exam for all students are below.

Question 1:

The dataset channing is available from the boot library.

1.a:

Load the boot library and access the channing dataset contained in it. Load the help file and read about the dataset. Where is the retirement centre that the data was collected from located?

```r
library(boot)
data(channing)
```

`?channing` or `help(channing)` opens the help file. The retirement centre is located in Palo Alto in California.

1.b:

Look at the structure of the dataset and describe it briefly. What unit is the age of entry measured in?

```r
str(channing)
```

```
## 'data.frame':    462 obs. of  5 variables:
##  $ sex  : Factor w/ 2 levels "Female","Male": 2 2 2 2 2 2 2 2 2 2 ...
##  $ entry: num  782 1020 856 915 863 906 955 943 943 837 ...
##  $ exit : num  909 1128 969 957 983 ...
##  $ time : num  127 108 113 42 120 106 100 82 100 108 ...
##  $ cens : num  1 1 1 1 1 1 1 1 1 1 ...
```

The dataset contains records on 462 people, with 5 variables. They show the individual's sex, time of entry to the house, time of leaving (if they have left), the total time spent there, and a `cens` variable indicating if they are still in the centre, or have died.

Answer: The age of entry is measured in **months**.

1.c:

What is the mean age of entry to the centre? Convert your answer to years, rounded to 1 decimal place.

```r
# Find the mean:
mean(channing$entry)
```

```
## [1] 905.8788
```

```r
# Now find it in years, and round it:
round(mean(channing$entry) / 12, 1)
```

```
## [1] 75.5
```

Answer: The mean age at entry (in years) is 75.5 years.

1.d:

Which resident was the youngest on entry into the centre? Are they male or female?

```
channing[which.min(channing$entry), ]
```

```
##        sex entry exit time cens
## 319 Female   733  870  137    0
```

```
vec1 <- channing[which.min(channing$entry), ]
vec1
```

```
##        sex entry exit time cens
## 319 Female   733  870  137    0
```

```
vec1$sex
```

```
## [1] Female
## Levels: Female Male
```

Answer: She's female and in row 319. She was aged 733 months on entry to the centre.

1.e:

The cens variable should be recorded as a factor. Convert it to a factor using sensible labels (the help file will be useful for this). Attach it to the data frame, and call the new column status. Should status be an ordered factor?

```
channing$status <- factor(channing$cens,
                         levels = c(0, 1),
                         labels = c('Left.or.Alive', 'Died.There'))
```

Answer: No, `status` should not be ordered, as it is not an ordinal variable.

1.f:

Remove the cens column.

```
channing$cens <- NULL
```

1.g:

Use the status variable to find out how many people died in the centre.

```
table(channing$status)
```

```
##
## Left.or.Alive    Died.There
##           286           176
```

Answer: 176 people have died in the centre.

1.h:

Use the aggregate function to aggregate all 3 remaining numeric variables by the factors of sex and status.

Which sex/status combination had the lowest mean age when entering the centre?

```
aggregate(channing[ , c(2, 3, 4)],
          list(channing$sex, channing$status),
          mean)
```

```
##   Group.1       Group.2    entry     exit     time
## 1  Female Left.or.Alive 887.5362 976.2596 88.73191
## 2    Male Left.or.Alive 914.8431 997.9804 83.13725
## 3  Female    Died.There 930.0462 999.4231 70.14615
## 4    Male    Died.There 921.3478 984.4783 63.13043
```

```
df1 <- aggregate(channing[ , c(2, 3, 4)],
          list(channing$sex, channing$status),
          mean)
df1[which.min(df1$entry), ]
```

```
##   Group.1       Group.2    entry     exit     time
## 1  Female Left.or.Alive 887.5362 976.2596 88.73191
```

The result here should be a single dataframe from the single line of code, with 2 columns for the factors of `sex` and `status`, and 3 columns for the summary statistics of the numeric variables, grouped by the factors. Code should be used to find the minimum needed.

Answer: Females who have left the centre or remain alive there had the lowest mean age on entry of any sex/status combination.

Question 2:

The dataset Cars93 is available from the MASS library.

2.a:

Load the MASS library and access the Cars93 dataset contained in it. Load the help file and read about the dataset. How many variables does the dataset contain?

```
library(MASS)
data(Cars93)
```

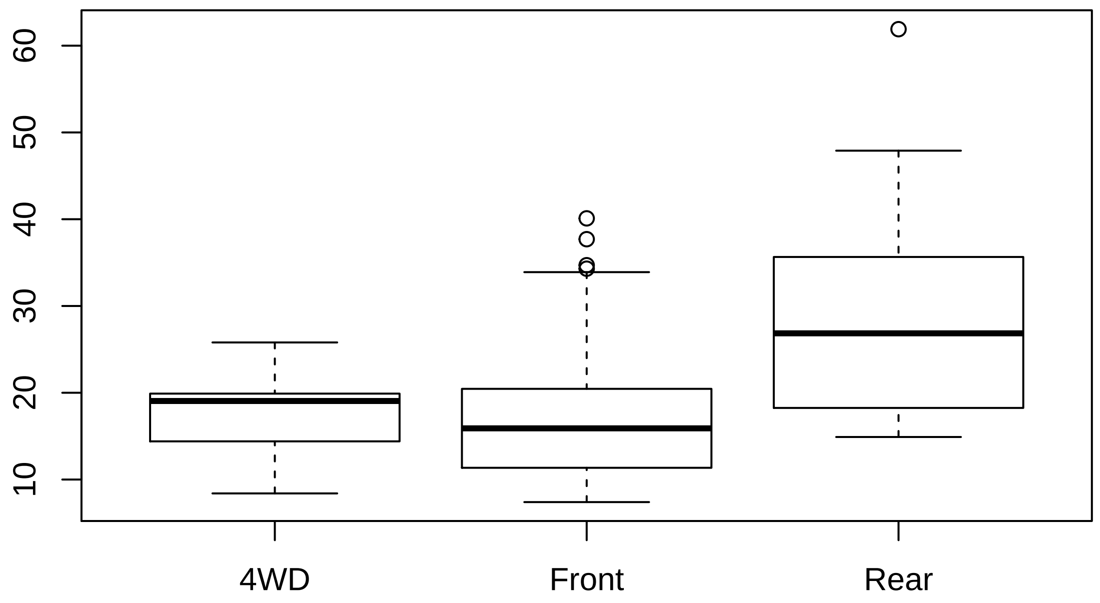
Answer: `?Cars93` loads the help file, and we can see that there are 27 variables (columns) in the dataset.

2.b:

Produce a boxplot of the Price variable grouped by the `DriveTrain` variable.

Comment on the resulting graph.

```
boxplot(Cars93$Price ~ Cars93$DriveTrain)
```



Answer: From the boxplot, we can see that cars with a rear wheel drive train are more expensive generally, and with a more variable price too. 4WD and front wheel drive train cars are generally cheaper, though there are a few expensive outliers with front wheel drive trains.

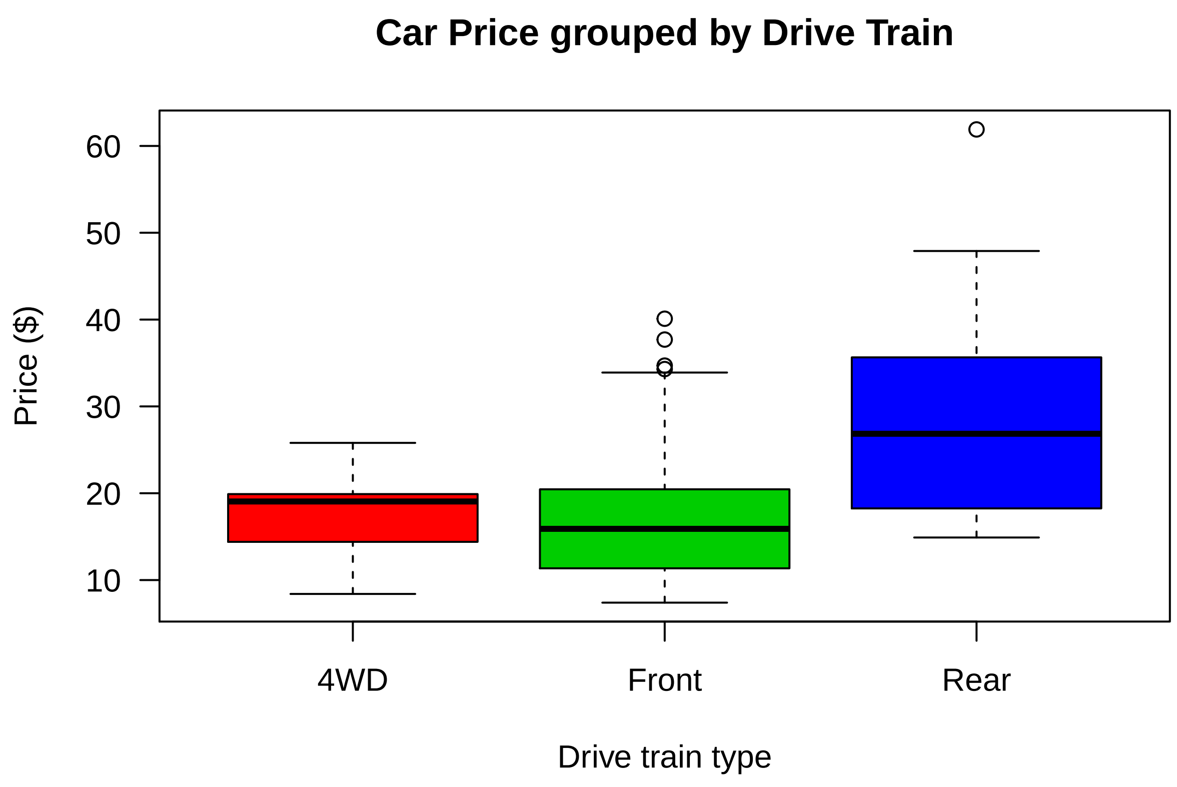
2.c:

Now make the plot look nice.

You should:

- Include different colours for the three boxplots
- Include a main title, and relevant x- and y-axis labels
- Rotate the numbers on the left axis so that they appear horizontal

```
boxplot(Cars93$Price ~ Cars93$DriveTrain,
        col = 2:4,
        las = 1,
        main = "Car Price grouped by Drive Train",
        ylab = "Price ($)",
        xlab = "Drive train type")
```



Question 3:

Let X be a normally-distributed random variable. The probability density function (pdf) of X is given below, where mu is the mean parameter and sigma is the standard deviation (sd) parameter. R has a built-in function dnorm() to evaluate this formula.

Run this code:

```
dnorm(0:2, mean=0, sd=1)
```

```
## [1] 0.39894228 0.24197072 0.05399097
```

3.a:

Write your own function called my.dnorm() which takes a vector x, a mean parameter, and a standard deviation parameter, and evaluates the pdf of the normal distribution. The mean parameter should default to 0 and the standard deviation parameter should default to 1 if no values are supplied for these arguments.

Test your function using my.dnorm(0:2, mean=0, sd=1) to confirm that it gives the same answer as R's function.

```
my.dnorm <- function(x, mean=0, sd=1) { # The function needs an input x, and
                                        # has default values of a mean of 0
                                        # and a standard deviation of 1


  a <- 1/sqrt(2 * pi * sd^2)            # Calculate the first part of the product
  b <- (x - mean)^2 / (2 * sd^2)        # Calculate the part in the exponential
  c <- a * exp(-b)                      # Evaluate the final expression needed

  return(c)                             # Return this
```

```r
}

# Alternative:
my.dnorm2 <- function(x, mean=0, sd=1) {  # The function needs an input x, and
                                          # has default values of a mean of 0
                                          # and a standard deviation of 1

  # Evaluate all parts needed in one single line, following the formula given:
  a <- 1 / sqrt(2 * pi * sd^2) * exp(-(x-mean)^2 / (2 * sd^2))

  # Return a:
  return(a)
}

# Testing to confirm they produce the same answer:
dnorm(0:2, mean=0, sd=1)
```

```
## [1] 0.39894228 0.24197072 0.05399097
```

```r
my.dnorm(0:2, mean=0, sd=1)
```

```
## [1] 0.39894228 0.24197072 0.05399097
```

```r
my.dnorm2(0:2, mean=0, sd=1)
```

```
## [1] 0.39894228 0.24197072 0.05399097
```

```r
all.equal(dnorm(0:2, mean=0, sd=1), my.dnorm(0:2, mean=0, sd=1), my.dnorm2(0:2, mean=0, sd=1))
```

```
## [1] TRUE
```

Answer: They produce the same answer!

3.b:

Benchmark dnorm() and my.dnorm() in order to compare their performance, and comment on the results.

```r
library(rbenchmark)
x <- rnorm(10000)

# Results will differ on different runs
df2 <- benchmark(dnorm(x), my.dnorm(x), my.dnorm2(x),
          replications=1000)

# Check the results:
df2
```

```
##           test replications elapsed relative user.self sys.self user.child
## 1     dnorm(x)         1000   0.505    2.577     0.453    0.044          0
## 2  my.dnorm(x)         1000   0.263    1.342     0.209    0.052          0
## 3 my.dnorm2(x)         1000   0.196    1.000     0.174    0.022          0
##   sys.child
## 1         0
## 2         0
## 3         0
```

Ideally include a copy-and-paste of the output of your system here, to support your interpretation.

Answer: On my laptop, my.dnorm2(x) is the fastest (`relative = 1`) and dnorm(x) is 2.577 times slower on my system.