3: Matrices

Introduction to R

This assignment consists of six parts:

- Intro to Basics
- Vectors
- Matrices (this document)
- Factors
- Data frames
- Lists

Create a script called `hw02-3.R` and save it in your `hw02` folder.

After you complete each exercise, commit and push your R script to your remote repo. See Part 0 for instructions. Do *not* push this document.

3.1 What's a matrix?

In R, a matrix is a collection of elements of the same data type (numeric, character, or logical) arranged into a fixed number of rows and columns. Since you are only working with rows and columns, a matrix is called two-dimensional.

You can construct a matrix in R with the `matrix()` function. Consider the following example:

```
matrix(1:9, byrow = TRUE, nrow = 3)
```

- Type `?matrix` in the console and press the enter key. Notice that the help tab in the lower left panel of RStudio describes the `matrix()` function and how to use it. The `?` followed by the function name is a short cut to get help for a function. You can also get help by putting your cursor inside the function name in a script and press the `F1` key (if your keyboard has one).

The information you provide in the parentheses of `matrix()` function are called arguments. For `matrix()`,

- The first argument is a vector of elements that `matrix()` will arrange into the rows and columns. Recall that `1:9` is a shortcut for `c(1, 2, 3, 4, 5, 6, 7, 8, 9)`.
- The argument `byrow = TRUE` tells the `matrix()` function to fill the matrix by rows. Change the argument to `byrow = FALSE` to fill the matrix by columns.
- The `nrow = 3` argument tells `matrix()` that the matrix should have three rows. You can use `nrow` or `ncol`.

You must supply some arguments to a function but other arguments usually have a default. If you look at the help for `matrix()`, the first argument is `data = NA`. That means you must provide the data. Argument `nrow = 1` defaults to creating a matrix with one row. By giving a different value to the argument, such as `nrow = 3`, you override the default.

Instructions

- Construct a matrix with 5 columns containing the numbers 10 up to 28, filled column-wise.

```
# Construct a matrix with 4 rows containing the numbers 11 up to 30, filled row-wise.




#
```

If your result looks like this, you wrote your code correctly. If it doesn't look like this, edit your code until you get the correct result.

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]   11   15   19   23   27
## [2,]   12   16   20   24   28
## [3,]   13   17   21   25   29
## [4,]   14   18   22   26   30
```

3.2 Construct a matrix

R contains many built-in data sets. One of them is called `chickwts` that contains the weights of six-week old chickens that were raised on different diets. Type `chickwts` in the console to see the entire data set. You'll use the first 10 observations from the first three diet types (horsebean, linseed, or soybean).

Instructions

Add code to your script to create the array.

- Enter `chick_weights <- chickwts$weight[c(1:20, 23:32)]` to create a vector with the weight data from the data set. You may copy and paste this line to be sure you get the data correctly.
- Create a matrix with three columns and 10 rows (`byrow = FALSE, ncol = 3`). Save the matrix in a variable called `weights`.

```
# Box office Star Wars (in millions!)
new_hope <- c(460.998, 314.4)
empire_strikes <- c(290.475, 247.900)
return_jedi <- c(309.306, 165.8)

# Create box_office


# Construct star_wars_matrix


#
```

3.3 Naming a matrix

To help you remember what is stored in `star_wars_matrix`, you would like to add the names of the movies for the rows. Not only does this help you to read the data, but it is also useful to select certain elements from the matrix.

Similar to vectors, you can add names for the rows and the columns of a matrix

```
rownames(my_matrix) <- row_names_vector
colnames(my_matrix) <- col_names_vector
```

I prepared two vectors for you: `region`, and `titles`. You will need these vectors to name the columns and rows of `star_wars_matrix`, respectively.

Instructions

- Use `colnames()` to name the columns of `star_wars_matrix` with the region vector.
- Use `rownames()` to name the rows of `star_wars_matrix` with the titles vector.
- Print out `star_wars_matrix` to see the result of your work.

```
# Box office Star Wars (in millions!)
new_hope <- c(460.998, 314.4)
empire_strikes <- c(290.475, 247.900)
return_jedi <- c(309.306, 165.8)

# Construct matrix
star_wars_matrix <- matrix(c(new_hope, empire_strikes, return_jedi), nrow = 3, byrow = TRUE)

# Vectors region and titles, used for naming
region <- c("US", "non-US")
titles <- c("A New Hope", "The Empire Strikes Back", "Return of the Jedi")

# Name the columns with region


# Name the rows with titles


# Print out star_wars_matrix


#
```

3.4 Calculating the worldwide box office

The single most important thing for a movie to become an instant legend in Tinseltown is its worldwide box office figures.

To calculate the total box office revenue for the three Star Wars movies, you have to take the sum of the US revenue column and the non-US revenue column.

In R, the function `rowSums()` conveniently calculates the totals for each row of a matrix. This function creates a new vector:

```
rowSums(my_matrix)
```

Instructions

- Calculate the worldwide box office figures for the three movies and put these in the vector named `worldwide_vector`.
- Print the value of `worldwide_vector`.

```
# Construct star_wars_matrix
box_office <- c(460.998, 314.4, 290.475, 247.900, 309.306, 165.8)
star_wars_matrix <- matrix(box_office, nrow = 3, byrow = TRUE,
                           dimnames = list(c("A New Hope", "The Empire Strikes Back", "Return of the Je
                                           c("US", "non-US")))

# Calculate worldwide box office figures


# Print out worldwide_vector


#
```

3.5 Adding a column for the Worldwide box office

In the previous exercise you calculated the vector that contained the worldwide box office receipt for each of the three Star Wars movies. However, this vector is not yet part of `star_wars_matrix`.

You can add a column or multiple columns to a matrix with the `cbind()` function, which merges matrices and/or vectors together by column. For example:

```
big_matrix <- cbind(matrix1, matrix2, vector1 ...)
```

Instructions

- Add `worldwide_vector` as a new column to the `star_wars_matrix` and assign the result to `all_wars_matrix`. Use the `cbind()` function.

```
# Construct star_wars_matrix
box_office <- c(460.998, 314.4, 290.475, 247.900, 309.306, 165.8)
star_wars_matrix <- matrix(box_office, nrow = 3, byrow = TRUE,
                           dimnames = list(c("A New Hope",
                                             "The Empire Strikes Back",
                                             "Return of the Jedi"),
                                           c("US", "non-US")))

# The worldwide box office figures
worldwide_vector <- rowSums(star_wars_matrix)

# Bind the new variable worldwide_vector as a column to star_wars_matrix


#
```

3.6 Adding a row

Just like every action has a reaction, every `cbind()` has an `rbind()`. (Yeah, pretty bad metaphor.)

Your R workspace, where all variables you defined 'live' (check out what a workspace is), has already been initialized and contains two matrices:

- `star_wars_matrix` that we have used all along, with data on the original trilogy,
- `star_wars_matrix2`, with similar data for the prequels trilogy.

Type the name of these matrices in the console and hit Enter if you want to have a closer look. If you want to check out the contents of the workspace, you can type `ls()` in the console.

Instructions

- Use `rbind()` to paste together `star_wars_matrix` and `star_wars_matrix2`, in this order. Assign the resulting matrix to `all_wars_matrix`.

```
# star_wars_matrix and star_wars_matrix2 are available in your workspace
box_office2 <- c(474.5, 552.5, 310.7, 338.7, 380.3, 468.5)
star_wars_matrix2 <- matrix(box_office2, nrow = 3, byrow = TRUE,
                            dimnames = list(c("The Phantom Menace",
                                              "Attack of the Clones",
                                              "Revenge of the Sith"),
                                            c("US", "non-US")))

# Combine both Star Wars trilogies in one matrix


#
```

3.7 The total box office revenue for the entire saga

Just like `cbind()` has `rbind()`, `colSums()` has `rowSums()`. The code chunk already contains the `all_wars_matrix` that you constructed in the previous exercise; type `all_wars_matrix` to have another look. Let's now calculate the total box office revenue for the entire saga.

Instructions

- Calculate the total revenue for the US and the non-US region and assign `total_revenue_vector`. You can use the `colSums()` function.
- Print out `total_revenue_vector` to have a look at the results.

```
# all_wars_matrix is available in your Notebook
all_wars_matrix

# Total revenue for US and non-US


# Print out total_revenue_vector


#
```

3.8 Selection of matrix elements

Similar to vectors, you can use the square brackets `[ ]` to select one or multiple elements from a matrix. Whereas vectors have one dimension, matrices have two dimensions. You should therefore use a comma to separate the rows you want to select from the columns. For example:

- `my_matrix[1,2]` selects the element at the first row and second column.
- `my_matrix[1:3,2:4]` results in a matrix with the data on the rows 1, 2, 3 and columns 2, 3, 4.

If you want to select all elements of a row or a column, no number is needed before or after the comma, respectively:

- `my_matrix[,1]` selects all elements of the first column.
- `my_matrix[1,]` selects all elements of the first row.

Back to Star Wars with this newly acquired knowledge! As in the previous exercise, `all_wars_matrix` is already available in your workspace.

Instructions

- Select the non-US revenue for all movies (the entire second column of `all_wars_matrix`), store the result as `non_us_all`.
- Use `mean()` on `non_us_all` to calculate the average non-US revenue for all movies. Simply print out the result.
- This time, select the non-US revenue for the first two movies in `all_wars_matrix`. Store the result as `non_us_some`.
- Use `mean()` again to print out the average of the values in `non_us_some`.

```
# all_wars_matrix is available in your workspace
all_wars_matrix

# Select the non-US revenue for all movies


# Average non-US revenue


# Select the non-US revenue for first two movies


# Average non-US revenue for first two movies


#
```

3.9 A little arithmetic with matrices

Similar to what you have learned with vectors, the standard operators like `+`, `-`, `/`, `*`, etc. work in an element-wise way on matrices in R.

For example, `2 * my_matrix` multiplies each element of `my_matrix` by two.

As a newly-hired data analyst for Lucasfilm, it is your job to find out how many visitors went to each movie for each geographical area. You already have the total revenue figures in `all_wars_matrix`. Assume that the price of a ticket was 5 dollars. Simply dividing the box office numbers by this ticket price gives you the number of visitors.

Instructions

- Divide `all_wars_matrix` by 5, giving you the number of visitors in millions. Assign the resulting matrix to `visitors`.
- Print out `visitors` so you can have a look.

```
# all_wars_matrix is available in your workspace
all_wars_matrix

# Estimate the visitors


# Print the estimate to the console


#
```

3.10 A little arithmetic with matrices (2)

Just like `2 * my_matrix` multiplied every element of `my_matrix` by two, `my_matrix1 * my_matrix2` creates a matrix where each element is the product of the corresponding elements in `my_matrix1` and `my_matrix2`.

After looking at the result of the previous exercise, big boss Lucas points out that the ticket prices went up over time. He asks to redo the analysis based on the prices you can find in `ticket_prices_matrix` (source: imagination).

Those who are familiar with matrices should note that this is not the standard matrix multiplication for which you should use `%*%` in R.

Instructions

- Divide `all_wars_matrix` by `ticket_prices_matrix` to get the estimated number of US and non-US visitors for the six movies. Assign the result to `visitors`.
- From the `visitors` matrix, select the entire first column, representing the number of visitors in the US. Store this selection as `us_visitors`.
- Calculate the average number of US visitors; print out the result.

```
# all_wars_matrix and ticket_prices_matrix are available for you
all_wars_matrix
ticket_prices <- c(5.0, 5.0, 6.0, 6.0, 7.0, 7.0, 4.0, 4.0, 4.5, 4.5, 4.9, 4.9)
ticket_prices_matrix <- matrix(ticket_prices, ncol = 2, byrow = TRUE,
                               dimnames = list(c("A New Hope",
                                                 "The Empire Strikes Back",
                                                 "Return of the Jedi",
                                                 "The Phantom Menace",
                                                 "Attack of the Clones",
                                                 "Revenge of the Sith"),
                                               c("US", "non-US")))


# Estimated number of visitors


# US visitors


# Average number of US visitors


#
```