Matricies and Dataframes in R

Sean Hellingman ©

Data Visualization and Manipulation through Scripting (ADSC1010)

shellingman@tru.ca

Fall 2024

THOMPSON RIVERS UNIVERSITY

Topics

2. Introduction
3. Matrices and Arrays
4. Matrix Creation
5. Matrix Operations
6. Data Frames
7. Data Frame Creation
8. Data Frame Operations
9. Exercises and References

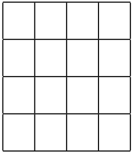
Introduction

- Important to understand some of the data structures in R.
- We have already spoken about vectors and classes of data in R.
- We are going to talk about two very important data structures in R:
  1. Matrices
  2. Data frames

Matrices and Arrays in R

Matrix in R

- In R, a **matrix** is simply a vector that has additional attributed called dimensions.
- Instead of only having one-dimensional indexes, there are now two.
- Matrix:



Array in R

- In R, an **array** is simply a multidimensional matrix.
- *We won't speak very much about arrays right now*.
- There are many online resources about arrays if you are interested.

`matrix()` Function

- The `matrix()` function may be used to create a matrix.
- Arguments:
  - `nrow = 4` sets the number of rows (four in this case).
  - `byrow = FALSE` fills the matrix by columns.

Example 1

- Create a matrix from a sequence 1 to 16 in four rows called `Matrix.1`.
- What happens when you change `byrow` to `TRUE` (call it `Matrix.2`)?

Row and Column Names

- Sometimes it is useful to define row and column names for a matrix.
- Functions:
  - `rownames(Matrix.1) <- c("A","B","C")` sets the row names of `Matrix.1` to A, B, and C.
  - `colnames(Matrix.1) <- c("a","b","c")` sets the row names of `Matrix.1` to a, b, and c.

Matrix Operators

Operation/Function	Description
+	Matrix addition
-	Matrix subtraction
*	Element by element multiplication
%*%	Matrix multiplication
diag()	Returns diagonal elements of a matrix
t()	Returns the transpose of a matrix
solve()	Returns the inverse of a matrix

Positional Indexing

- To extract elements based on their position we write the position within `[row, column]`

- Use `Matrix.3[2,3]` to extract the second element from the 3^{rd} column of Matrix.3.

- Negative indexing allows us to extract everything *except* the negatively indexed rows.
  - Use `Matrix.3[-2,-3]` to extract every element of Matrix.3 apart from the elements in row 2 and column 3.

- `Matrix.3[,3]` would extract the 3^{rd} column of `Matrix.3`

- `Matrix.3[3,c(1,5,7)]` would extract the 1st, 5th, 7th elements from 3^{rd} row.

Example 2

- Create a 5 x 2 matrix called `Matrix.3` from a sequence 1 to 19 increasing by 2.
- Return the diagonal elements of `Matrix.3`.
- Multiply `Matrix.1` by `Matrix.3`. What happens?
- Remove the 3^{rd} row of `Matrix.3`.

Diagonal Matrices

- `diag(x,nrow,ncol)` will create a `nrow`×`ncol` matrix whose diagonal entries are given by the elements of `x`.
    - `diag(c(1,-1,0),nrow=3,ncol=3)`
    - `diag(c(1,-1,0),nrow=4,ncol=3)`
    - `diag(c(1,-1,0),nrow=3,ncol=4)`
- If dimensions don't line up, will repeat or ignore elements as needed.
    - `diag(c(1,-1,0),nrow=7,ncol=7)`
    - `diag(c(1,-1,0),nrow=2,ncol=2)`
- $n \times n$ identity matrix can be created using `diag(1,n,n)`

`rbind()`

- The `rbind()` function combines (binds) multiple groups of rows together.
- If `x` and `y` are vectors of the same length, `rbind(x,y)` will create a matrix whose first and second rows are `x` and `y`, respectively.
  - Will label the rows `x` and `y`.
  - Can shut the labelling off with `rbind(x,y,deparse.level=0)`
- `rbind(x,y,z)` would create a matrix with three rows, etc.

cbind()

- The `cbind()` function combines (binds) multiple columns together.
- If `x` and `y` are vectors of the same length, `cbind(x,y)` will create a matrix whose first and second columns are `x` and `y`, respectively.
  - Labelling (and its shut-off) are the same.
  - `cbind(x,y,z)` would create a matrix with three columns, etc.

Example 3

- Run the following code in R:
  - `x <- c(1,2,3)`
  - `y <- c(4,7,6)`
  - `A <- rbind(x,y)`
  - `B <- cbind(x,y)`
  - `C <- rbind(A,B)`
  - `D <- cbind(A,B)`
- Comment on your results.

Non-Numeric Values

- Like vectors, R allows for matrices with non-numeric entries.
- Some Examples:
  - `matrix(c("This","Topic","Is","Complete"),2,2,byrow=TRUE)`
  - `matrix(c(TRUE, TRUE, FALSE, TRUE),2,2,byrow=FALSE)`
  - `matrix(c("This","Topic","Is","Complete"),4,1)`
  - `matrix(c(TRUE, TRUE, FALSE, TRUE),1,4)`

Data Frames in R

Data Frames

- Most commonly used data structure to store data.
- May contain a mixture of different data types.
- Excel worksheet-like, but more powerful.
    - Under the hood, a data frame is a list of equal-length vectors.
- All columns are of equal length.

Creating Data Frames

- Imported data is commonly formatted as a data frame.
- May use `data.frame()` to construct a data frame.
- Can convert a matrix to a data frame using `as.data.frame()`.

Example 4

- Create a data frame from the following vectors:
  - `Player <- c("Anchor", "Adekugbe", "Gauld", "Ahmed")`
  - `Height <- c(191, 178, 168, 180)`
  - `International <- C(FALSE, TRUE, TRUE, TRUE)`
- Examine the classes of the columns.
- Use the `dim()` function to check the dimensions.
- Use the `str()` function to get more details about the data frame.

`rbind()` & `cbind()`

- Can use `rbind()` to add rows to an existing data frame.
  - Must contain the same number of columns.
  - Recommended to add data frames to existing data frames with `rbind()`.
- Can use `cbind()` to add columns to an existing data frame.
  - Must contain the same number of rows.
  - Columns must have the same data type.

Attributes

- May change row and column names:
  - `rownames(x) <- c("row.1","row.2","row.3")`
  - `colnames(x) <- c("col.1","col.2","col.3")`
  - `names(x) <- c("col_1","col_2","col_3")`
- *Note:* `colnames()` and `names()` work the same for data frames.
- We can also add a comment to the data frame, which does not affect how the data frame operates.
  - `comment(x) <- "This is a comment"`
  - The comment will show in the attributes (`attributes()`).

Positional Indexing

- Positional indexing in data frames is very similar to matrices.
- Use `DataFrame.1[1,]` to return the first row of `DataFrame.1`.
- Use `DataFrame.1[,4]` to return the 4^{th} column of `DataFrame.1`.
- Use `DataFrame.1[1:4,5]` to return rows 1 to 4 of column 5 of `DataFrame.1`.

Selecting Observations I

- Use `subset()` function to select columns by name from a data frame or matrix:
  - `subset(df, select=colname)`
  - `subset(df, select=c(colname.1,...,colname.N))`
- Delete `col2` and `col3` to create a new data frame with without `col2` and `col3`:
  - `df2 <- subset(df, select=-c(col2,col3))`

Selecting Observations II

- Can also use the `subset()` function to logically select a subset of data from a data frame.
    - `subset(df, subset=(col1 > 0))` gives a data frame with all columns of `df`, but only the rows with `col1` element greater than 0.
    - `subset(df, select=c(col1, col2), subset=(col1 > 0))` selects `col1` and `col2` of `df`, but only the rows with `col1` element greater than 0.

Example 5

- Using the *mtcars* dataset in R:
  - Use the `str()` function to get more details about the data frame.
  - Change the name of the first column to 'MPG'.
  - Create a data frame from the first 3 columns of *mtcars*.
  - Create a data frame of cars that have a MPG greater than 20.

Editing a Data Frame

- Can use the `edit()` function to create an editor window.
- *I strongly suggest saving a temporary object to avoid making permanent mistakes.*
- Further information may be found online.

Merging Data Frames

- Command: `merge(df1, df2, by="col.name")`
- Does not require the rows to be sorted or even to occur in the same order.
- Discards rows that appear in only one data frame or the other.

Example 6

- Create a data frame from the following vectors:
  - `Player <- c("Hasal", "Adekugbe", "Gauld", "Ahmed")`
  - `Position <- c("GK","DF","MF","DF")`

- Use the `merge()` function to add the playing position to the data frame from *Example 4*.

Dealing with `NA` values

- *Recall*: `NA` values are logical values indicating missing data.
- `na.omit()` removes *any* row that contains `NA` values.
- `df1 <- df[!is.na(df[,3]),]` only removes the rows with `NA` values in column 3.
- `df2 <- df[!(is.na(df[,3]) & is.na(df[,5])),]` removes rows with `NA` values in column 3 AND column 5.

Converting One Structured Data Type into Another I

Conversion	**How**
Vector→List	`as.list(vec)`
Vector→Matrix	1-column matrix: `cbind(vec)` or `as.matrix(vec)` 1-row matrix: `rbind(vec)` $n \times m$ matrix: `matrix(vec,n,m)`
Vector→Data Frame	1-column data frame: `as.data.frame(vec)` 1-row data frame: `as.data.frame(rbind(vec))`
List→Vector	`unlist(lst)`
List→Matrix	1-column matrix: `as.matrix(lst)` 1-row matrix: `as.matrix(rbind(lst))` $n \times m$ matrix: `matrix(lst,n,m)`
List→Data Frame	List elements are columns: `as.data.frame(lst)`

Converting One Structured Data Type into Another II

Conversion	**How**
Matrix→Vector	`as.vector(mat)`
Matrix→List	`as.list(mat)`
Matrix→Data Frame	To convert a 1-row data frame: `dfrm[1,]`
Data Frame→Vector	To convert a 1-column data frame: `dfrm[,1]` or `dfrm[[1]]`
Data Frame→List	`as.list(dfrm)`
Data Frame→Matrix	`as.matrix(dfrm)`

Exercise 1

- Create the following vector `x <- seq(1,10)`.
- What do you expect each of the following commands to produce?
  - `matrix(x,nrow=5,ncol=2)`
  - `matrix(x,nrow=10,ncol=1)`
  - `matrix(x,nrow=1,ncol=10)`
  - `matrix(x,nrow=4,ncol=3)`
- Execute and confirm.

Exercise 2

- What do you expect the output of each of the following to be:
  - `rbind(c(1,1),c(-1,1))`
  - `cbind(c(1,1),c(-1,1))`
  - `rbind(seq(1,2,by=0.25),c(c(1,2),c(3,2,1)))`
  - `rbind(c(1,0,0),c(0,1,0))`
  - `cbind(c(1,0,0),c(0,1,0))`
  - `rbind(c(1,0,0),c(0,1,0),c(0,0,1))`
  - `cbind(c(1,0,0),c(0,1,0),c(0,0,1))`
- How does `t(rbind(x,y,z))` compare to `cbind(x,y,z)`?

Exercise 3

- Use `matrix` and `cbind` to create the matrix:

$$A = \begin{bmatrix} 7 & 7 & 7 & 0 & 0 & 0 & 0 \\ 7 & 7 & 7 & 0 & 0 & 0 & 0 \\ 7 & 7 & 7 & 0 & 0 & 0 & 0 \\ 7 & 7 & 7 & 0 & 0 & 0 & 0 \end{bmatrix}$$

- Use `matrix` and `rbind` to create the matrix:

$$B = \begin{bmatrix} -2 & -2 & 1 & 1 & 1 & 1 & 1 \\ -2 & -2 & 1 & 1 & 1 & 1 & 1 \\ -2 & -2 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Exercise 4

- Create the matrix:

$$\begin{bmatrix} 7 & 7 & 7 & 0 & 0 & 0 & 0 \\ 7 & 7 & 7 & 0 & 0 & 0 & 0 \\ 7 & 7 & 7 & 0 & 0 & 0 & 0 \\ 7 & 7 & 7 & 0 & 0 & 0 & 0 \\ -2 & -2 & 1 & 1 & 1 & 1 & 1 \\ -2 & -2 & 1 & 1 & 1 & 1 & 1 \\ -2 & -2 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Exercise 5

- Practice creating the following data frames:

```
df1 <- data.frame(col1 = 1:3,
                  col2 = c("this", "is", "ADSC1010"),
                  col3 = c(TRUE, FALSE, TRUE))
```

- `v1 <- 1:3`
- `v2 <-c("this", "is", "ADSC1010")`
- `df2 <- data.frame(col1 = v1, col2 = v2)`

Exercise 6

- Change the *row* and *column* names of `df1` and `df2` from Exercise 5.

Exercise 6

- Import the *starwars* dataset from the *dplyr* package
  - Use the `str()` function to get more details about the data frame.
  - Change the column names as appropriate.
  - Create a new data frame from the first 15 observations and the first four columns of the *starwars* data.
  - Add a column to your new data frame ranking the characters from your most favourite to least favourite.
  - Merge your new data frame with the original *starwars* data. What happens?

Exercise 7

- Play around with removing `NA` values from the *starwars* data frame.

References & Resources

1. Douglas, A., Roos, D., Mancini, F., Couto, A., & Lusseau, D. (2023). *An introduction to R*. Retrieved from https://intro2r.com/

- https://www.r-bloggers.com/2021/09/r-data-types/
- https://cran.r-project.org/web/packages/MASS/MASS.pdf