Programming for Data Analytics

3. Matrices and Data Frames

Dr. Jim Duggan,
School of Engineering & Informatics
National University of Ireland Galway.

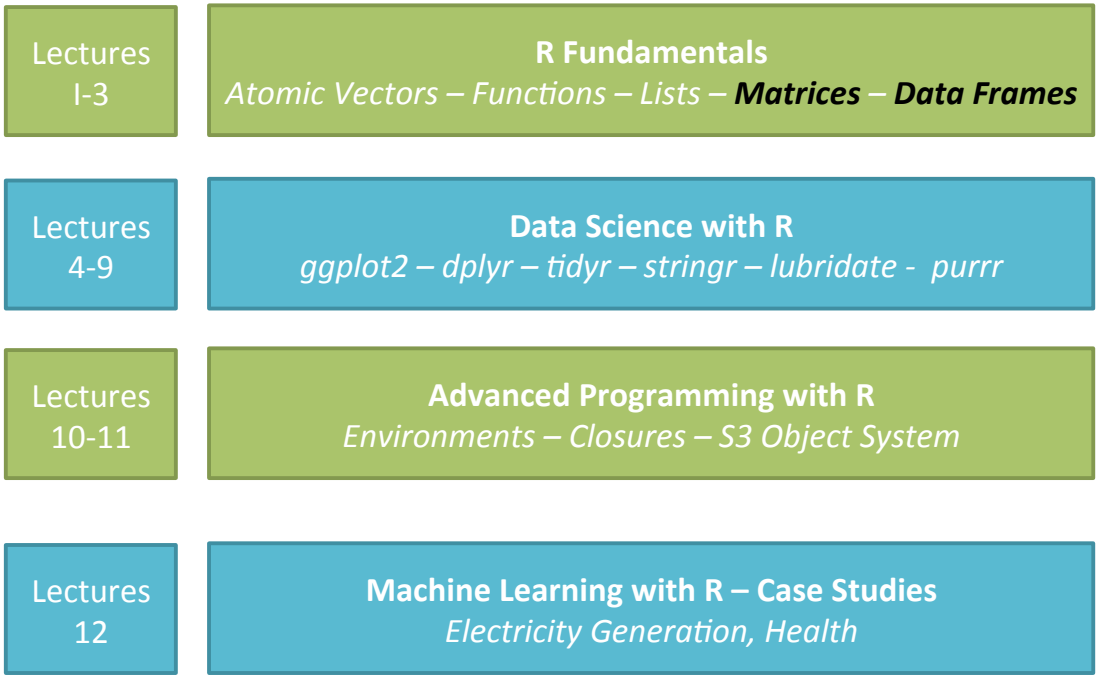
https://twitter.com/_jimduggan

Course Overview

Lectures I-3	**R Fundamentals** *Atomic Vectors – Functions – Lists –* ***Matrices – Data Frames***
Lectures 4-9	**Data Science with R** *ggplot2 – dplyr – tidyr – stringr – lubridate - purrr*
Lectures 10-11	**Advanced Programming with R** *Environments – Closures – S3 Object System*
Lectures 12	**Machine Learning with R – Case Studies** *Electricity Generation, Health*

Lecture Overview

- Matrices
  - Subsetting
  - apply function
- Data Frames
  - Filtering
  - Adding Columns
  - Cleaning Data



(1) Matrices

	Homogenous	Heterogenous
1d	Atomic Vector	List
2d	**Matrix**	Data Frame
nd	Array	

- A matrix can be initialized from a vector, where the numbers of rows and columns are specified.
- R stores matrices by column-major order, and by default matrices are filled in this manner.

Declaring a matrix

```
>
> a <- matrix(1:6, ncol=3, nrow=2)
>
> a
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
>
> dim(a)
[1] 2 3
```

Adding rows and columns

```
>
> cbind(a,c(7,8))
     [,1] [,2] [,3] [,4]
[1,]    1    3    5    7
[2,]    2    4    6    8
>
>
> rbind(a,c(7,8,9))
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
[3,]    7    8    9
```

Naming rows and columns

```
> rownames(a) <- c("A","B")
>
> a
  [,1] [,2] [,3]
A    1    3    5
B    2    4    6
>
> colnames(a) <- c("a","b","c")
>
> a
  a b c
A 1 3 5
B 2 4 6
```

Subsetting Matrices

- The most common way of subsetting 2d matrix is a simple generalisation of 1d subsetting
- Supply a 1d index for each dimension, separated by a comma
- Blank subsetting is useful, as it lets you keep all rows or all columns

Using row index…

```
> b <- matrix(1:9, nrow=3)
>
> colnames(b) <- c("A","B","C")
>
> b
     A B C
[1,] 1 4 7
[2,] 2 5 8
[3,] 3 6 9
>
> b[1:2,]
     A B C
[1,] 1 4 7
[2,] 2 5 8
```

```
> b[c(T,F),]
     A B C
[1,] 1 4 7
[2,] 3 6 9
>
>
> b[-3,]
     A B C
[1,] 1 4 7
[2,] 2 5 8
```

Using column index…

```
> b
     A B C
[1,] 1 4 7
[2,] 2 5 8
[3,] 3 6 9
>
> b[,1:2]
     A B
[1,] 1 4
[2,] 2 5
[3,] 3 6
```

```
> b[,c(T,F)]
     A C
[1,] 1 7
[2,] 2 8
[3,] 3 9
>
> b[,c("A","C")]
     A C
[1,] 1 7
[2,] 2 8
[3,] 3 9
```

Sample Matrix Operations

Operator or Function	Description
A * B	Element-wise multiplication
A / B	Element-wise division
A %*% B	Matrix multiplication
t(A)	Transpose of A
e<-eigen(A)	List of eigenvalues and eigenvectors for matrix A

apply() function

- The **apply()** function can be used to process rows and columns for a matrix, and the general form of this function (Matloff 2009) is **apply(m, dimcode, f, fargs)**, where:
  - **m** is the target matrix
  - **dimcode** identifies whether it's a row or column target. The number 1 applies to rows, whereas 2 applies to columns
  - **f** is the function to be called
  - **fargs** are the optional set of arguments that can be applied to the function **f**.

Examples…

```
> b
     A B C
[1,] 1 4 7
[2,] 2 5 8
[3,] 3 6 9
>
> apply(b,1,sum)
[1] 12 15 18
```

```
> b
     A B C
[1,] 1 4 7
[2,] 2 5 8
[3,] 3 6 9
>
> apply(b,2,sum)
 A  B  C
 6 15 24
```

(2) Data Frames

- The most common way of storing data in R
- Under the hood, a data frame is a list of equal-length vectors
- A two-dimensional structure, it shares properties of both a list and a matrix

	Homogenous	Heterogenous
1d	Atomic Vector	List
2d	Matrix	**Data Frame**
nd	Array	

Creating a data frame…

```
> df <- data.frame(x=1:5,y=LETTERS[1:5],stringsAsFactors=F)
>
> str(df)
'data.frame':   5 obs. of  2 variables:
 $ x: int  1 2 3 4 5
 $ y: chr  "A" "B" "C" "D" ...
```

```
> df
  x y
1 1 A
2 2 B
3 3 C
4 4 D
5 5 E
```

Sample Data Set (mtcars)

A data frame with 32 observations on 11 variables.

[, 1] mpg Miles/(US) gallon
[, 2] cyl Number of cylinders
[, 3] disp Displacement (cu.in.)
[, 4] hp Gross horsepower
[, 5] drat Rear axle ratio
[, 6] wt Weight (1000 lbs)
[, 7] qsec 1/4 mile time
[, 8] vs V/S
[, 9] am Transmission (0 = automatic, 1 = manual)
[,10] gear Number of forward gears
[,11] carb Number of carburetors

Using head()

```
> head(mtcars)
                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

Subsetting rows (note list operator $ can be used to identify column)

```
> mtcars[mtcars$gear == 5,]
                mpg cyl  disp  hp drat    wt qsec vs am gear carb
Porsche 914-2  26.0   4 120.3  91 4.43 2.140 16.7  0  1    5    2
Lotus Europa   30.4   4  95.1 113 3.77 1.513 16.9  1  1    5    2
Ford Pantera L 15.8   8 351.0 264 4.22 3.170 14.5  0  1    5    4
Ferrari Dino   19.7   6 145.0 175 3.62 2.770 15.5  0  1    5    6
Maserati Bora  15.0   8 301.0 335 3.54 3.570 14.6  0  1    5    8
>
> mtcars[mtcars$gear == 5 & mtcars$cyl == 8,]
                mpg cyl disp  hp drat   wt qsec vs am gear carb
Ford Pantera L 15.8   8  351 264 4.22 3.17 14.5  0  1    5    4
Maserati Bora  15.0   8  301 335 3.54 3.57 14.6  0  1    5    8
```

Accessing rows via indices

```
> mtcars[1:8,]
                   mpg cyl  disp  hp drat    wt  qsec vs am gear carb
Mazda RX4         21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag     21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
Datsun 710        22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive    21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
Valiant           18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
Duster 360        14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4
Merc 240D         24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
```

Sampling from a data frame…

- Selecting n random observations from a data frame

```
> mtcars[sample(nrow(mtcars),2),]
                  mpg cyl  disp  hp drat    wt  qsec vs am gear carb
Chrysler Imperial 14.7   8 440.0 230 3.23 5.345 17.42  0  0    3    4
Merc 280          19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
>
> mtcars[sample(nrow(mtcars),2),]
                mpg cyl  disp hp drat    wt qsec vs am gear carb
Merc 240D      24.4   4 146.7 62 3.69 3.190 20.0  1  0    4    2
Toyota Corolla 33.9   4  71.1 65 4.22 1.835 19.9  1  1    4    1
```

Filtering rows and columns

```
> mtcars[mtcars$cyl==6,c("mpg","cyl")]
                  mpg cyl
Mazda RX4        21.0   6
Mazda RX4 Wag    21.0   6
Hornet 4 Drive   21.4   6
Valiant          18.1   6
Merc 280         19.2   6
Merc 280C        17.8   6
Ferrari Dino     19.7   6
```

Filtering behaviour (> 1 column)

```
> mtcars[1:4,1:2]
                  mpg cyl
Mazda RX4        21.0   6
Mazda RX4 Wag    21.0   6
Datsun 710       22.8   4
Hornet 4 Drive   21.4   6
> str(mtcars[1:4,1:2])
'data.frame':   4 obs. of  2 variables:
 $ mpg: num  21 21 22.8 21.4
 $ cyl: num  6 6 4 6
```

Filtering behaviour (1 column)

- This shows the difference between **simplifying** and **preserving**
- This also is a common source of programming errors, as R automatically converts a 1 column data frame result to a vector
- Use drop=FALSE to prevent
- tibbles (later) are designed to avoid this problem.

```
> mtcars[1:4,1]
[1] 21.0 21.0 22.8 21.4
>
> mtcars[1:4,1,drop=F]
                  mpg
Mazda RX4        21.0
Mazda RX4 Wag    21.0
Datsun 710       22.8
Hornet 4 Drive   21.4
```

Challenge 3.1

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2
Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3	1
Duster 360	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3	4
Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4	2
Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2
Merc 280	19.2	6	167.6	123	3.92	3.440	18.30	1	0	4	4

- List all the cars that have an mpg greater than the average
- List the car(s) with the greatest displacement

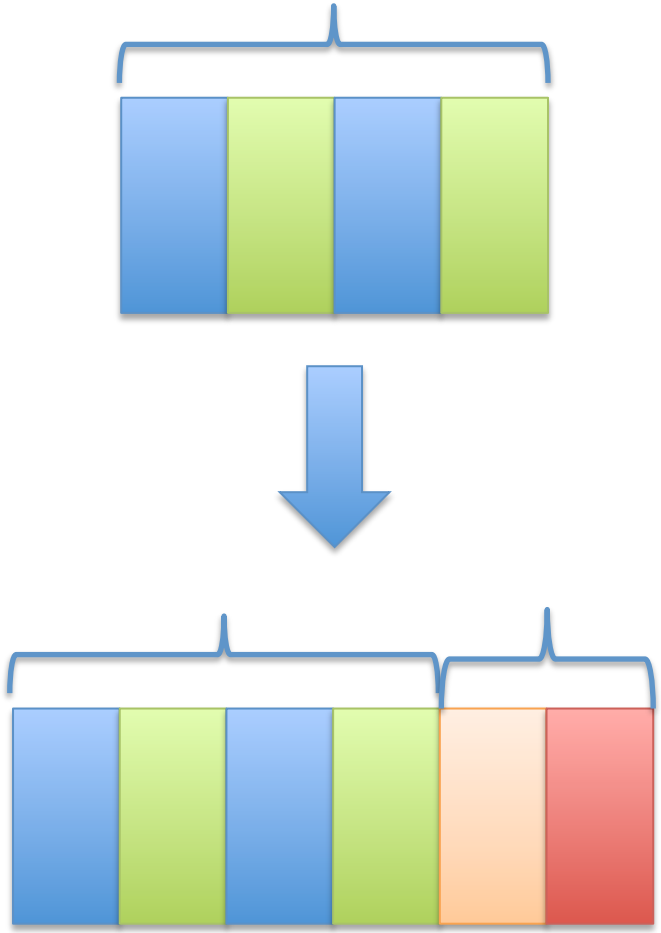
Sample Data Set (mtcars)

A data frame with 32 observations on 11 variables.

[, 1] mpg Miles/(US) gallon
[, 2] cyl Number of cylinders
[, 3] disp Displacement (cu.in.)
[, 4] hp Gross horsepower
[, 5] drat Rear axle ratio
[, 6] wt Weight (1000 lbs)
[, 7] qsec 1/4 mile time
[, 8] vs V/S
[, 9] am Transmission (0 = automatic, 1 = manual)
[,10] gear Number of forward gears
[,11] carb Number of carburetors

Adding Columns

- Often the initial data set may not contain sufficient information for analysis
- Adding new variables (columns) is an important feature to have
- Data frames support this: columns can be combined or new information used



Adding new columns

- Create a new column that contains kilometers per gallon

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2
Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3	1
Duster 360	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3	4
Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4	2
Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2
Merc 280	19.2	6	167.6	123	3.92	3.440	18.30	1	0	4	4

Solution

```
> mtcars$kpg <- mtcars$mpg * 8/5
>
> mtcars[1:8,]
                   mpg cyl  disp  hp drat    wt  qsec vs am gear carb   kpg
Mazda RX4         21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4 33.60
Mazda RX4 Wag     21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4 33.60
Datsun 710        22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1 36.48
Hornet 4 Drive    21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1 34.24
Hornet Sportabout 18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2 29.92
Valiant           18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1 28.96
Duster 360        14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4 22.88
Merc 240D         24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2 39.04
```

Challenge 3.2

- Create a new column called “Model”, which contains the type of car. How is the model information currently stored?

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2
Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3	1
Duster 360	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3	4
Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4	2
Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2
Merc 280	19.2	6	167.6	123	3.92	3.440	18.30	1	0	4	4

What does this code do?

```
> my_mtcars <- mtcars
>
> set.seed(0)
>
> rows <- sample(nrow(my_mtcars),3)
> col  <- sample(ncol(my_mtcars),1)
>
> my_mtcars[rows,col] <- NA
>
> rows
[1] 29  9 12
>
> col
[1] 7
```

Resulting data frame (3 rows)

```
> my_mtcars[rows,]
                 mpg cyl  disp  hp drat   wt qsec vs am gear carb
Ford Pantera L 15.8   8 351.0 264 4.22 3.17   NA  0  1    5    4
Merc 230       22.8   4 140.8  95 3.92 3.15   NA  1  0    4    2
Merc 450SE     16.4   8 275.8 180 3.07 4.07   NA  0  0    3    3
```

Find rows that have incomplete cases in the data set

```
> !complete.cases(my_mtcars)
 [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE
[12]  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[23] FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE
>
> my_mtcars[!complete.cases(my_mtcars),]
                mpg cyl  disp  hp drat   wt qsec vs am gear carb
Merc 230       22.8   4 140.8  95 3.92 3.15   NA  1  0    4    2
Merc 450SE     16.4   8 275.8 180 3.07 4.07   NA  0  0    3    3
Ford Pantera L 15.8   8 351.0 264 4.22 3.17   NA  0  1    5    4
>
> rows
[1] 29  9 12
```

Approximating missing values.

```
> mean(my_mtcars$qsec)
[1] NA
>
> mean(my_mtcars$qsec,na.rm = T)
[1] 17.80552
```

- Replace any missing values in the qsec column with the overall average value

Changing the values…

```
> my_mtcars[rows,"qsec"] <- mean(my_mtcars$qsec,na.rm = T)
>
> my_mtcars[rows,]
                 mpg cyl  disp  hp drat   wt     qsec vs am gear carb
Ford Pantera L 15.8   8 351.0 264 4.22 3.17 17.80552  0  1    5    4
Merc 230       22.8   4 140.8  95 3.92 3.15 17.80552  1  0    4    2
Merc 450SE     16.4   8 275.8 180 3.07 4.07 17.80552  0  0    3    3
```

Setting invalid values to NA

```
> my_mtcars <- mtcars
>
> rows <- sample(nrow(my_mtcars),3)
> col  <- sample(ncol(my_mtcars),1)
>
> my_mtcars[rows,col] <- -9000
>
> col  <- sample(ncol(my_mtcars),1)
>
> my_mtcars[rows,col] <- -1000
> my_mtcars[rows,]
                 mpg cyl  disp  hp drat   wt  qsec vs am  gear carb
Ford Pantera L 15.8   8 351.0 264 4.22 3.17 -9000  0  1 -1000    4
Merc 230       22.8   4 140.8  95 3.92 3.15 -9000  1  0 -1000    2
Merc 450SE     16.4   8 275.8 180 3.07 4.07 -9000  0  0 -1000    3
```

Removing invalid values…

```
> my_mtcars[rows,]
                 mpg cyl  disp  hp drat   wt  qsec vs am  gear carb
Ford Pantera L 15.8   8 351.0 264 4.22 3.17 -9000  0  1 -1000    4
Merc 230       22.8   4 140.8  95 3.92 3.15 -9000  1  0 -1000    2
Merc 450SE     16.4   8 275.8 180 3.07 4.07 -9000  0  0 -1000    3
>
> clean <- data.frame(apply(my_mtcars,2,
+                           function(x)ifelse(x<0,NA,x)))
>
> clean[rows,]
                 mpg cyl  disp  hp drat   wt qsec vs am gear carb
Ford Pantera L 15.8   8 351.0 264 4.22 3.17   NA  0  1   NA    4
Merc 230       22.8   4 140.8  95 3.92 3.15   NA  1  0   NA    2
Merc 450SE     16.4   8 275.8 180 3.07 4.07   NA  0  0   NA    3
```

Challenge 3.3

- Extend the previous example so that the apply function returns the average of all valid values instead of NA.
- *Hint: Apply functions can have more than one statement, once the {} brackets are used.*

```
> my_mtcars[rows,]
                 mpg cyl  disp  hp drat   wt  qsec vs am  gear carb
Ford Pantera L 15.8   8 351.0 264 4.22 3.17 -9000  0  1 -1000    4
Merc 230       22.8   4 140.8  95 3.92 3.15 -9000  1  0 -1000    2
Merc 450SE     16.4   8 275.8 180 3.07 4.07 -9000  0  0 -1000    3
```