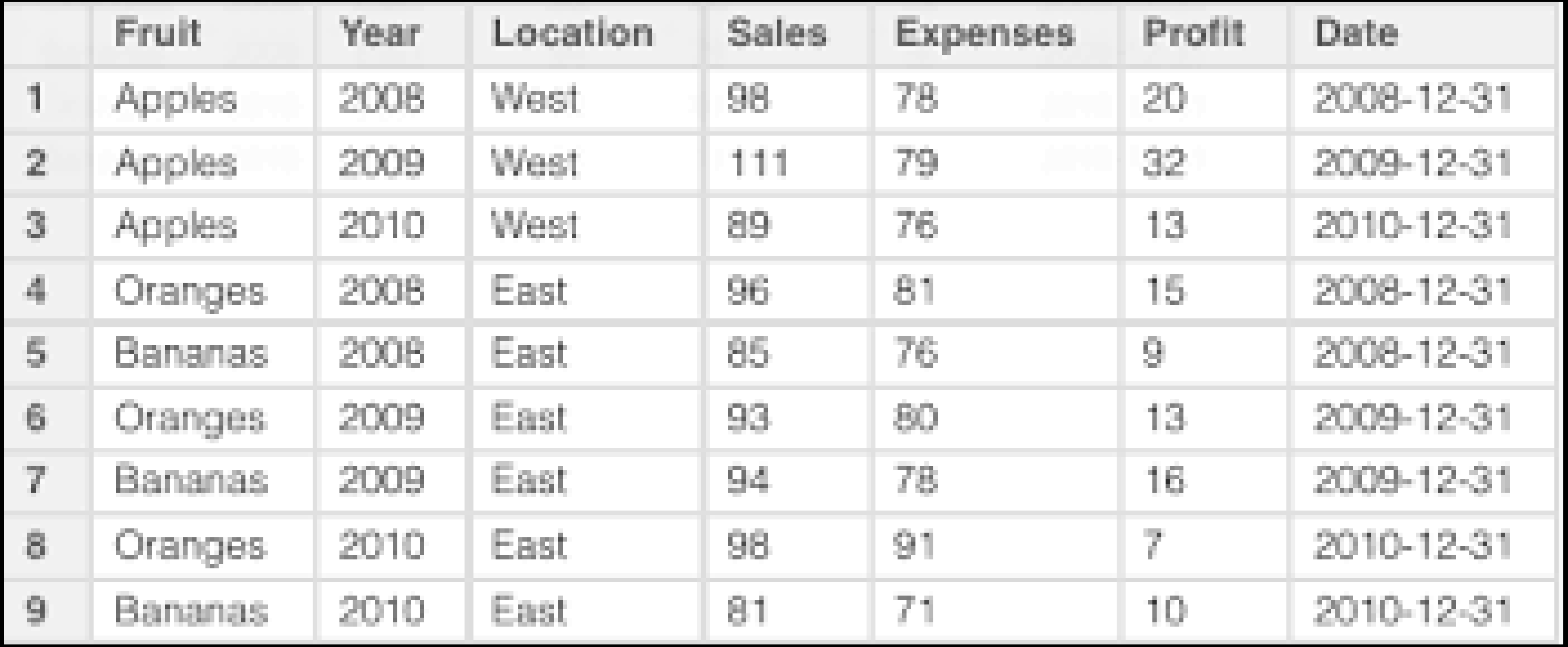
Types of Data Structures

1. Atomic Vectors
2. Data frames
3. Matrices
4. Lists
5. Arrays



	Fruit	Year	Location	Sales	Expenses	Profit	Date
1	Apples	2008	West	98	78	20	2008-12-31
2	Apples	2009	West	111	79	32	2009-12-31
3	Apples	2010	West	89	76	13	2010-12-31
4	Oranges	2008	East	96	81	15	2008-12-31
5	Bananas	2008	East	85	76	9	2008-12-31
6	Oranges	2009	East	93	80	13	2009-12-31
7	Bananas	2009	East	94	78	16	2009-12-31
8	Oranges	2010	East	98	91	7	2010-12-31
9	Bananas	2010	East	81	71	10	2010-12-31

Data frames

- Commonly used R object
- You will either:
  1. Get a data frame
  2. Make your own data frame
- Have columns and rows. Usually represent
  - Columns → variables
  - Rows → observations
- Each column is a vector!

Making a data frame

- A data frame can be built from scratch inside R using the function `data.frame()`
- `Arguments` are vectors with of same length

Importing data (simplified)

- Many datasets are imported as data frames
- A useful file format is .csv
- Key function is `read.table()` & its variants
- Data can also be imported from other formats e.g. .xls/.xlsx, .spv, .tab,…
- Easiest way to start –> save Excel files as CSV
- Many useful R packages for diff. data formats

Exploring data frames

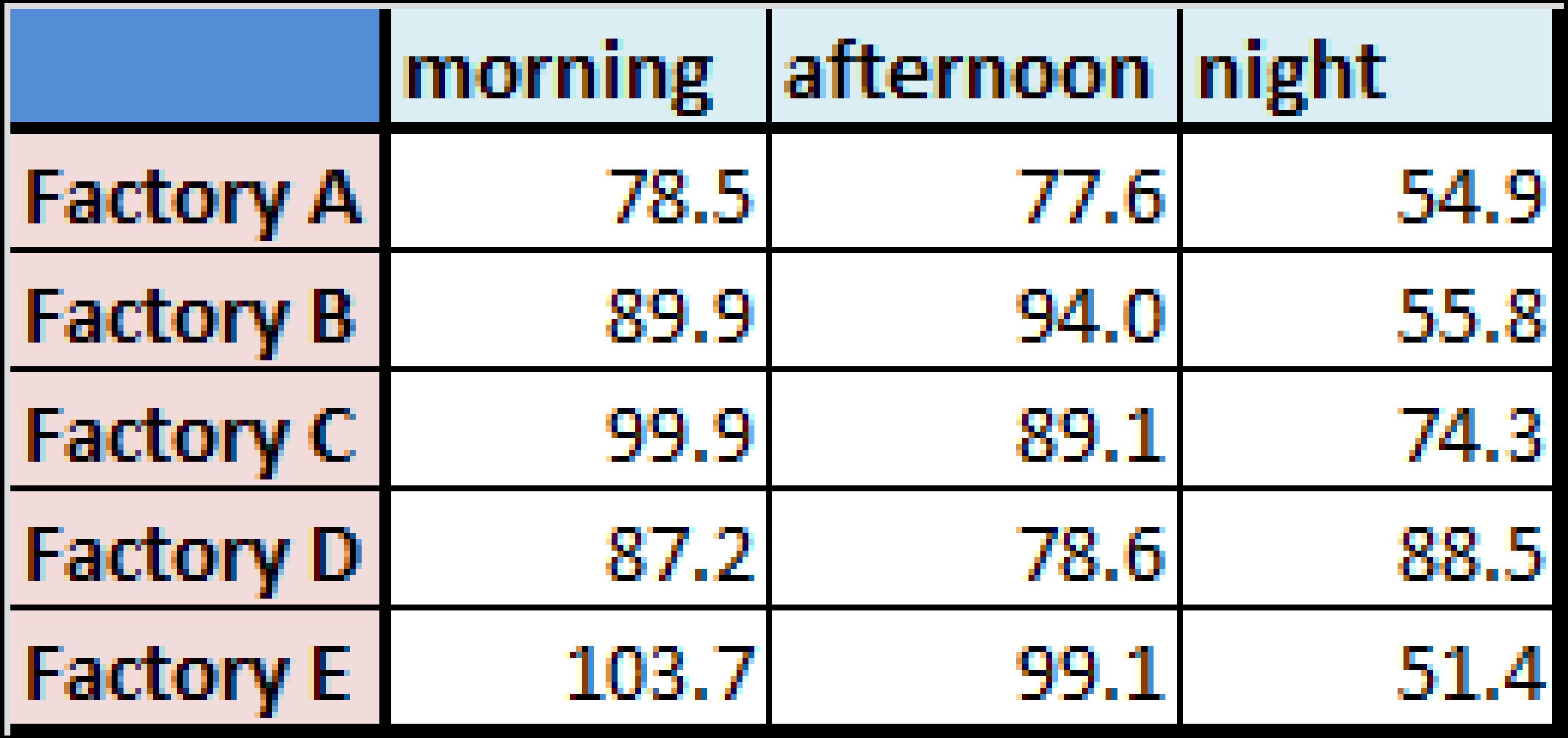
- A few functions
  - Check dimensions – `dim()`
  - View in spreadsheet format – `View()`
  - Examine structure – `str()`
  - See first/last records – `head()`/`tail()`
  - Summarise variables – `summary()`
  - See/set names of variable – `colnames()`
  - Check if – `is.data.frame()`
  - Change to – `as.data.frame()`
  - Make changes to data using `edit()`

Subsetting

- The `$` operator
  - Look at the output of `str()` again - note the ‘`$`’
  - You can pick out individual columns with syntax `dataframe$columnname`
  - Same thing as a vector
  - Use to get or set variables

Types of Data Structures

1. Atomic Vectors
2. Data frames
3. Matrices
4. Lists



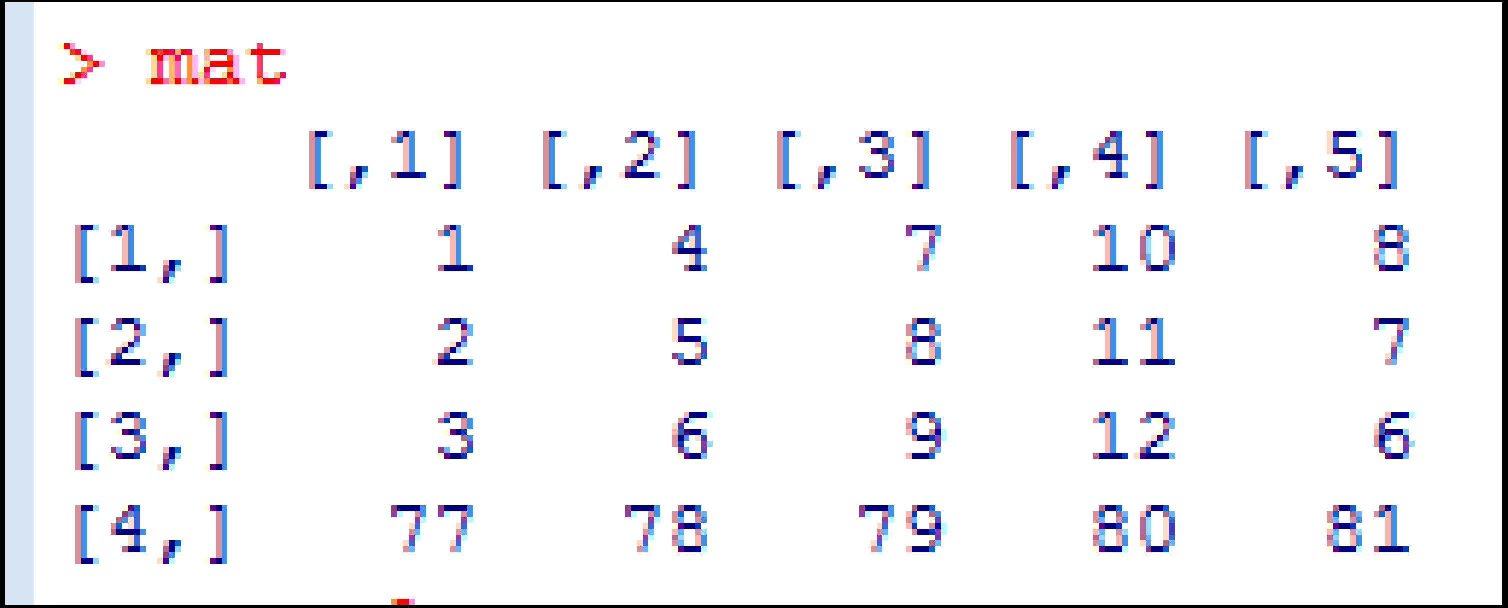
	morning	afternoon	night
Factory A	78.5	77.6	54.9
Factory B	89.9	94.0	55.8
Factory C	99.9	89.1	74.3
Factory D	87.2	78.6	88.5
Factory E	103.7	99.1	51.4

Matrices

- Like data frame are 2-dimensional
  - rows & columns
- Like vectors, only of particular type
  - integer, character, numeric, logical, etc.
- Can be built using `matrix()`
- Others include `rbind()`, `cbind()`
- Matrix arithmetic / linear algebra can also be done; useful in machine learning

- Matrix creation is also useful in other operations such as drawing multiple plots
- Exploring a matrix is somewhat similar to that of a data frame
  - `dim()`
  - `typeof()`
  - `class()`

- Run the script create-matrix.R to create a matrix, mat



```
> mat
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    4    7   10    8
[2,]    2    5    8   11    7
[3,]    3    6    9   12    6
[4,]   77   78   79   80   81
```

- bracket operator is also used for data frames

Indexing

- This is a crucial aspect of R programming
- Enable you to get or set elements of a data structure
- Bracket operator "[ ]" is used for indexing
  - For vectors: [ ]
  - For matrices & data frames: [,]
  - For lists: [ ] and [[ ]]

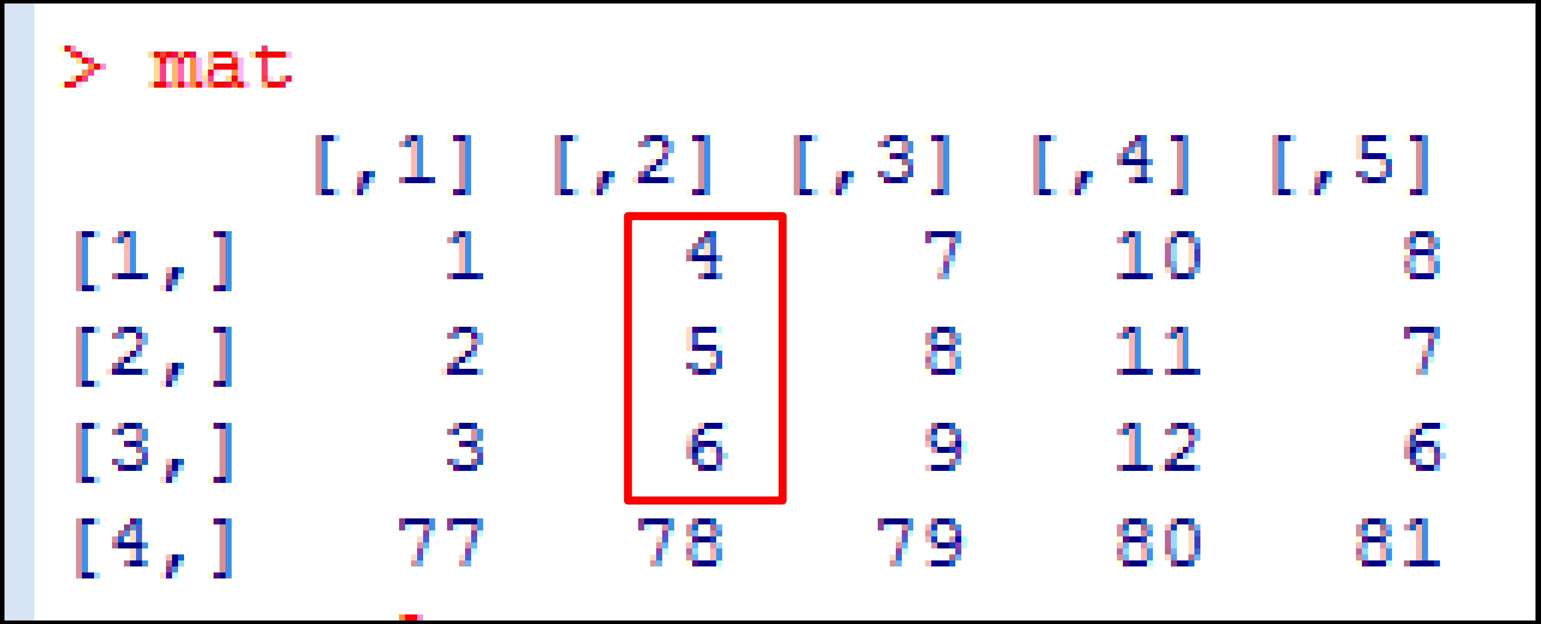
- The format for indexing is `[row, column]`
  - `[1, 2]` means “first row, second column”
- If you want to **get** a matrix value on the 4th row and the 3rd column, you run

```
matrix[4, 3]
```

- If you want to **change** a matrix value on the 4th row and the 3rd column, you run

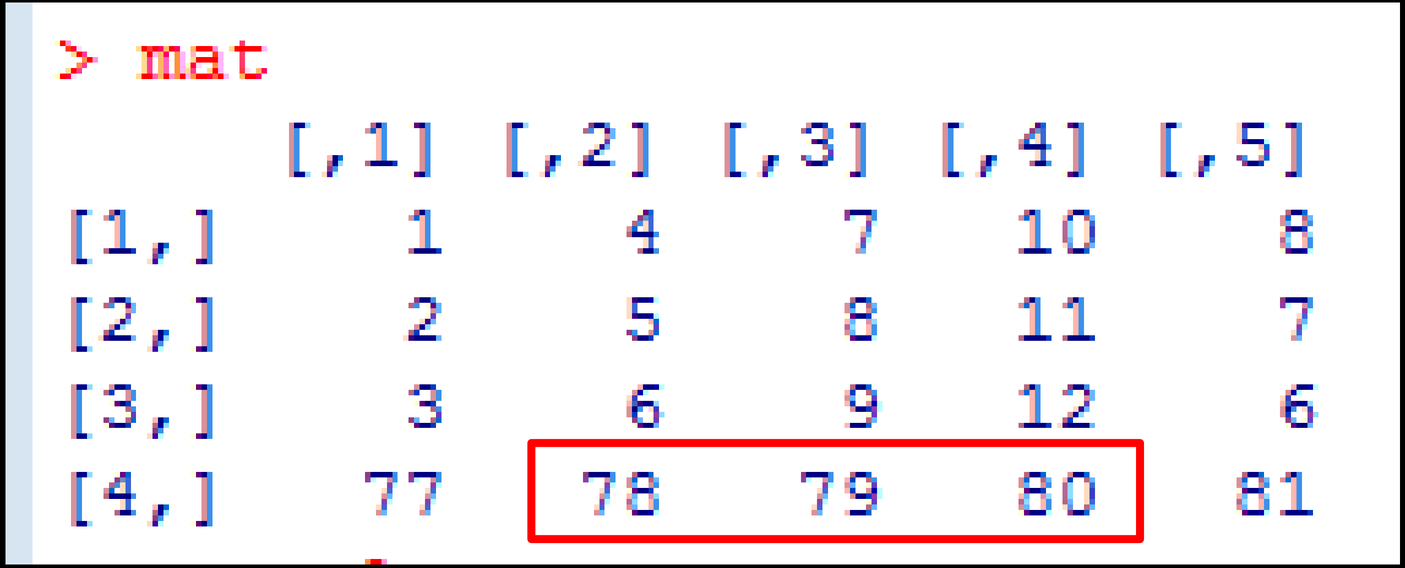
```
matrix[4, 3] <- <new value>
```

- Ranges also work well with indexing
  - First 3 rows of column 2 is written as `[1:3, 2]`



```
> mat
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    4    7   10    8
[2,]    2    5    8   11    7
[3,]    3    6    9   12    6
[4,]   77   78   79   80   81
```

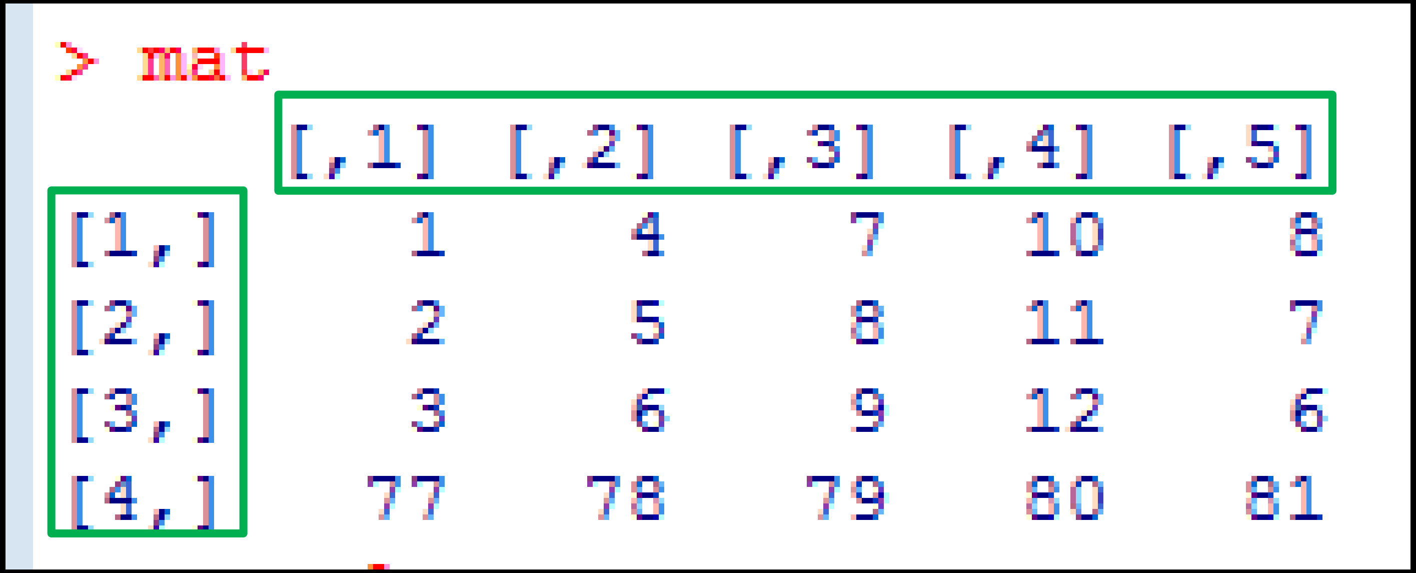
– Row 4 of columns 2 to 4 is coded as `[4, 2:4]`



```
> mat
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    4    7   10    8
[2,]    2    5    8   11    7
[3,]    3    6    9   12    6
[4,]   77   78   79   80   81
```

- Note the indices (or is it indexes?) of the matrix along the margins of the output

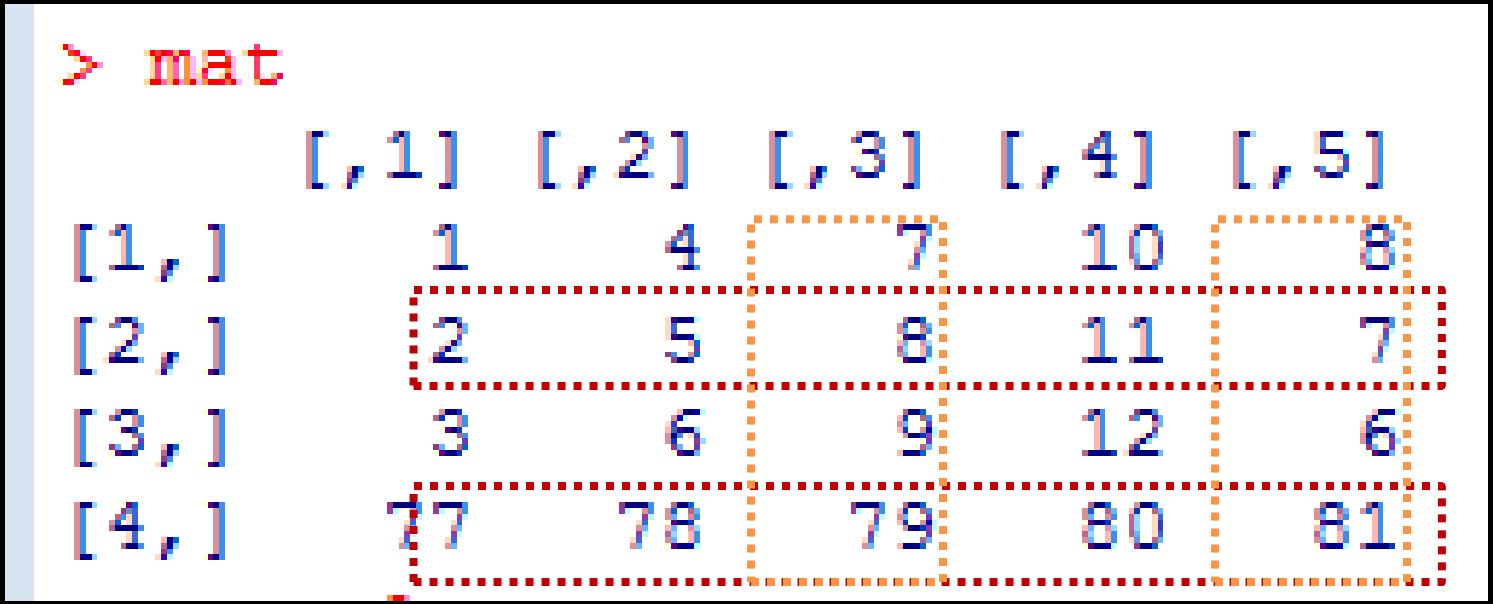
(also, learn that there is no trivial or frivolous output in R. Studying it can teach you a lot!)



```
> mat
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    4    7   10    8
[2,]    2    5    8   11    7
[3,]    3    6    9   12    6
[4,]   77   78   79   80   81
```

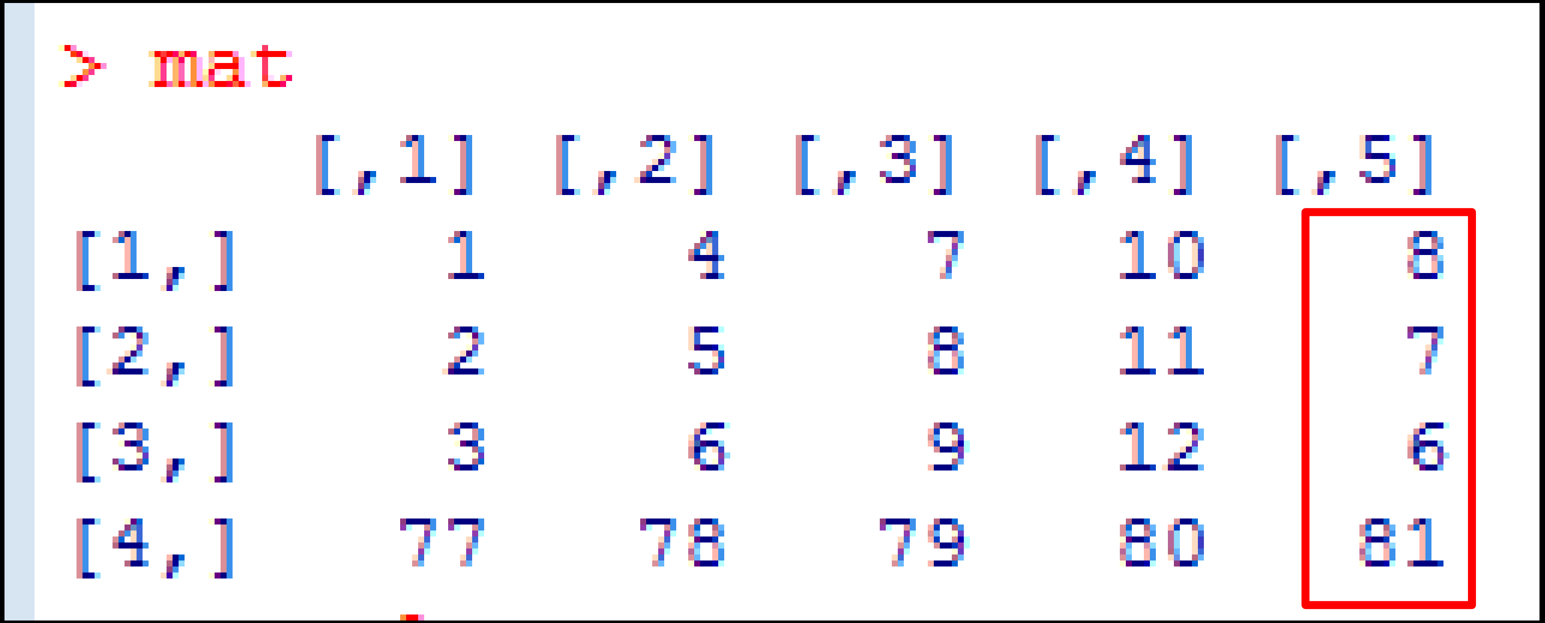
- Nit-picking is also possible
  - To select rows 2 & 4 of columns 3 & 5 write

```
[c(2, 4), c(3, 5)] # concatenate function
```



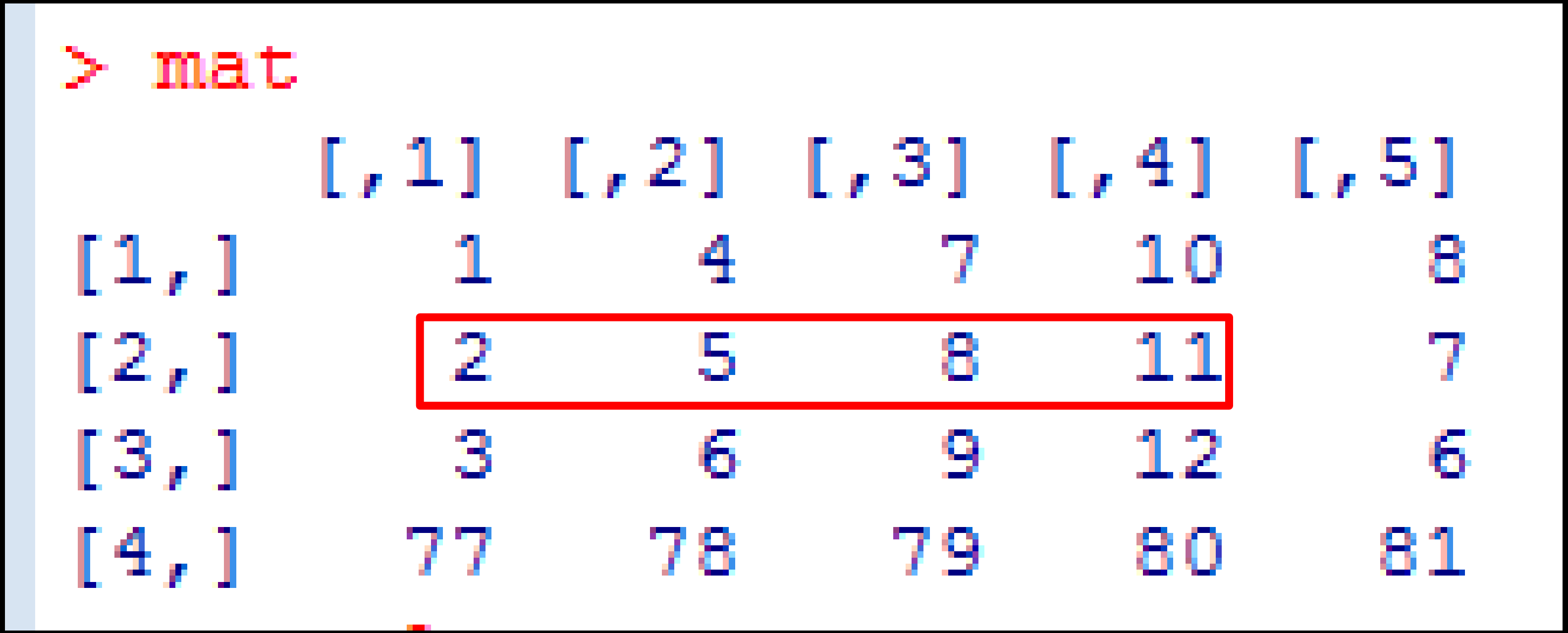
```
> mat
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    4    7   10    8
[2,]    2    5    8   11    7
[3,]    3    6    9   12    6
[4,]   77   78   79   80   81
```

- An empty value means ‘ALL’
  - E.g. `[, 5]` (all rows in column 5)



```
> mat
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    4    7   10    8
[2,]    2    5    8   11    7
[3,]    3    6    9   12    6
[4,]   77   78   79   80   81
```

- Negative indexing is there too
  - Row 2 except column 5 is written as `[2, -5]`



```
> mat
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    4    7   10    8
[2,]    2    5    8   11    7
[3,]    3    6    9   12    6
[4,]   77   78   79   80   81
```

Indexing works for vectors, too!

- Bracket notation applies to vectors
- Because they have 1 dimension, there will be no comma inside the brackets

    i.e. `[ ]` and not `[ , ]`

- Try it out!

Lists

- R lists are unique data structures – very versatile
- Heterogenous – can hold any kind of R object in memory, singly or in combination
- A list can also contain lists

Making a list

- The function `list()` is used to form a list
- Download the sample script create-list.R to see how lists can be formed
- Go run each line of code and carefully study the output.
- Observe that the code constructs list(s) of all the R objects studied thus far (and more!)

Quiz

1. How many list elements are there in `mumbo_jumbo`?
2. List the different types of R objects you can find among the elements.

Indexing lists

- The bracket operator `[ ]` is also used to extract elements of a list
  - Single `[ ]`
    - Will give you a list
  - Double `[[ ]]`
    - Will give you the element
  - Looking at `str()`, you may notice that `$` also identifies each element of the list.
    - To access them the elements must be named – easily done with the function names()

Exercises

- Run `mumbo_jumbo[1]`, and then `mumbo_jumbo[[1]]`
- Now call `typeof()` on each of them e.g. `typeof(mumbo_jumbo[1])`
- A well-known expert recently shared a photo on Twitter to help understanding of R list indexing. Click here to view it.

Data frames are lists!

- Recall that `$` is used to mark **named** columns of a data frame
- Also recall that each column in a data frame can stand alone as a vector
- A data frame is essentially a list with these characteristics
  - Made up of vectors
  - All the vectors are of equal length
  - Run `typeof()` on any data frame to check.

Homework

- Name the elements of `mumbo_jumbo` with the `names()` function as follows (Hint: First, visit the help file with `?names`)
  1. Calcium
  2. EAR_done
  3. Bin.code
  4. Bin.code_expt
  5. filesOnFlash
  6. Circle
  7. table_def
  8. MS_simul
  9. my_hex
  10. my_bits
- Use the name and `$` to extract any element from the list.
- What is the type of the extracted element? What does that tell you about the nature of indexing done with `$`.
- Run and review file [create-vector.R]()

Questions?