BIOS 721|TOPIC 1: OBJECTS IN R – PART 2

Outline

- Last time we covered …
  - Mode of objects available in R
    - Numeric / Character / Logical
  - Class of objects available in R
    - Vector / Matrix / Array / Data Frame / List
  - Functions used to create objects in R

- This time we will cover …
  - Basic manipulations performed on objects in R
    - Indexing
    - Applying operators
    - Applying simple functions

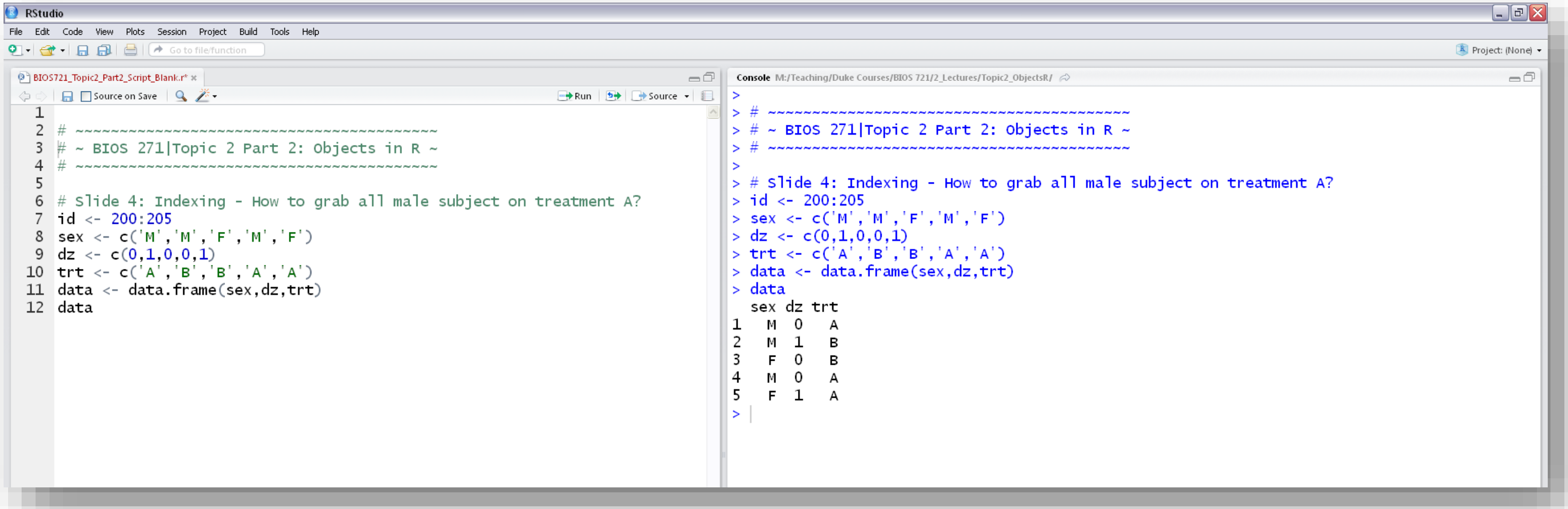
Indexing

- In Topic 1 Part 1, we discussed how to create several types/modes of objects in R.

- Once an object has been created, either by the user or as the output from an R function, the user will often need to select specific elements of that object for further manipulation.
    - This is known as ‘indexing’.

Indexing – Motivating Example

- For example, consider the data frame below. Suppose we wanted to create another data frame that subsets to all male subjects who were exposed to treatment A.
  - How could we get R to identify these subjects using code? → Indexing!



```r

# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# ~ BIOS 271|Topic 2 Part 2: Objects in R ~
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# Slide 4: Indexing - How to grab all male subject on treatment A?
id <- 200:205
sex <- c('M','M','F','M','F')
dz <- c(0,1,0,0,1)
trt <- c('A','B','B','A','A')
data <- data.frame(sex,dz,trt)
data
```

```
>
> # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
> # ~ BIOS 271|Topic 2 Part 2: Objects in R ~
> # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
>
> # Slide 4: Indexing - How to grab all male subject on treatment A?
> id <- 200:205
> sex <- c('M','M','F','M','F')
> dz <- c(0,1,0,0,1)
> trt <- c('A','B','B','A','A')
> data <- data.frame(sex,dz,trt)
> data
  sex dz trt
1   M  0   A
2   M  1   B
3   F  0   B
4   M  0   A
5   F  1   A
> 
```

Indexing

- In general, indexing allows users to modify objects by selecting or removing specified elements from the original object to created a new object.

- In R, users can index elements in object using their numerical indices (e.g. row and column NUMBER in a matrix) or by their dimension names if available (e.g. row and column NAME in a matrix).

- Let’s see examples of how indexing works for the following objects. Although the syntax is slightly different for each object, there is a pattern to the syntax.
  - Vectors
  - Matrices
  - Data Frames
  - Will discuss indexing other objects (Arrays and Lists) later in the course

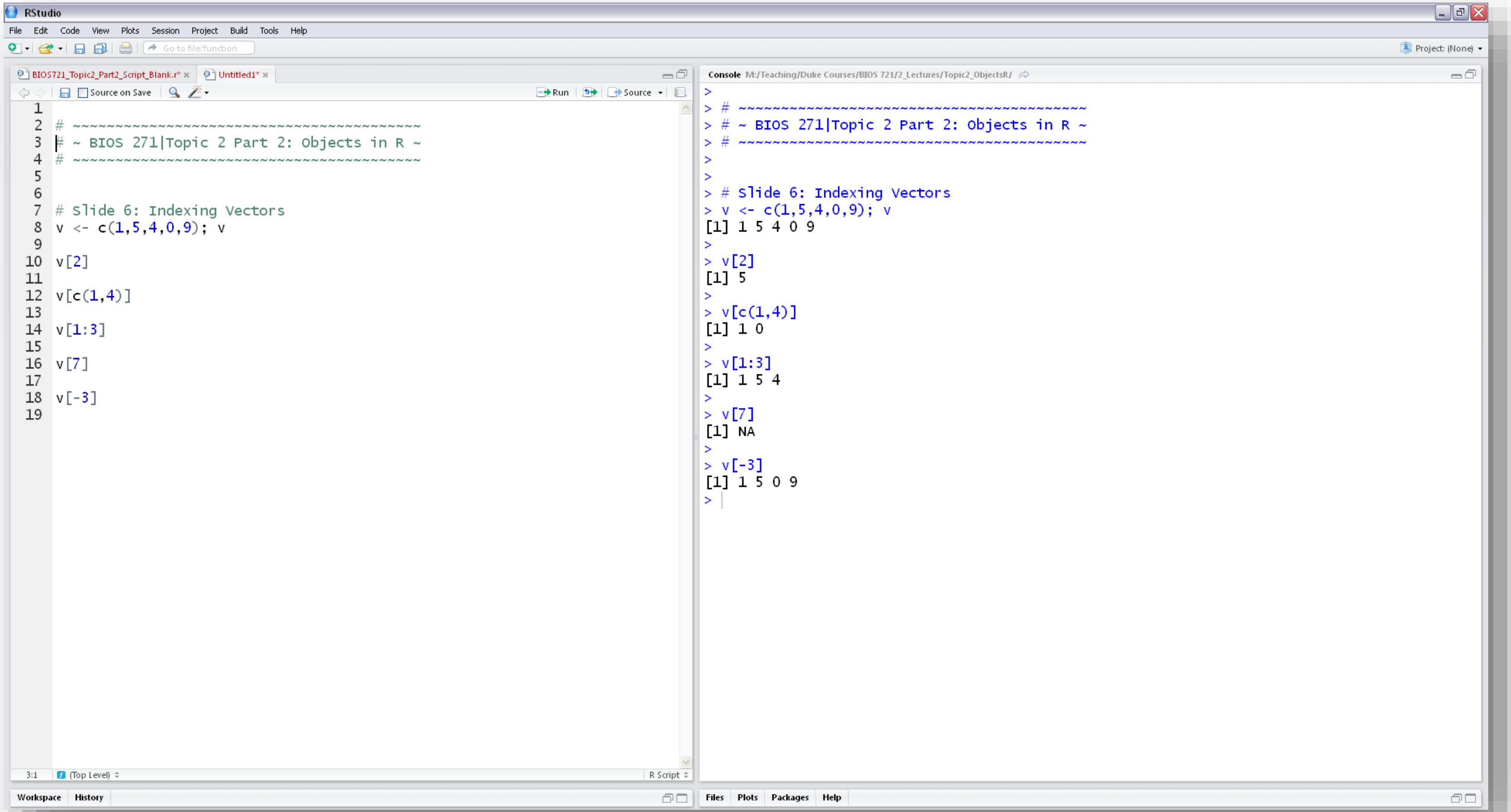
Indexing

- To select elements in a vector, indicate the position of the elements you want to retain or drop within square brackets next the object name.
  - Indexing Vectors: `object[element(s) position]`
  - **Example|** Consider the vector `v <- c(1,5,4,0,9)`

```
v[2]
v[c(1,4)]
v[1:3]
v[7]
v[-3]
```

Indexing

- Indexing Vectors



```r

# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# ~ BIOS 271|Topic 2 Part 2: Objects in R ~
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~


# Slide 6: Indexing Vectors
v <- c(1,5,4,0,9); v

v[2]

v[c(1,4)]

v[1:3]

v[7]

v[-3]

```

```
>
> # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
> # ~ BIOS 271|Topic 2 Part 2: Objects in R ~
> # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
>
>
> # Slide 6: Indexing Vectors
> v <- c(1,5,4,0,9); v
[1] 1 5 4 0 9
>
> v[2]
[1] 5
>
> v[c(1,4)]
[1] 1 0
>
> v[1:3]
[1] 1 5 4
>
> v[7]
[1] NA
>
> v[-3]
[1] 1 5 0 9
>
```

Indexing

- To select elements in a matrix, indicate the position of the elements you want to retain or drop within square brackets next the object name.
  - Indexing Matrices: `object[row(s),column(s)]`
  - Example| Consider the matrix `M <- matrix(1:6,3,2)`

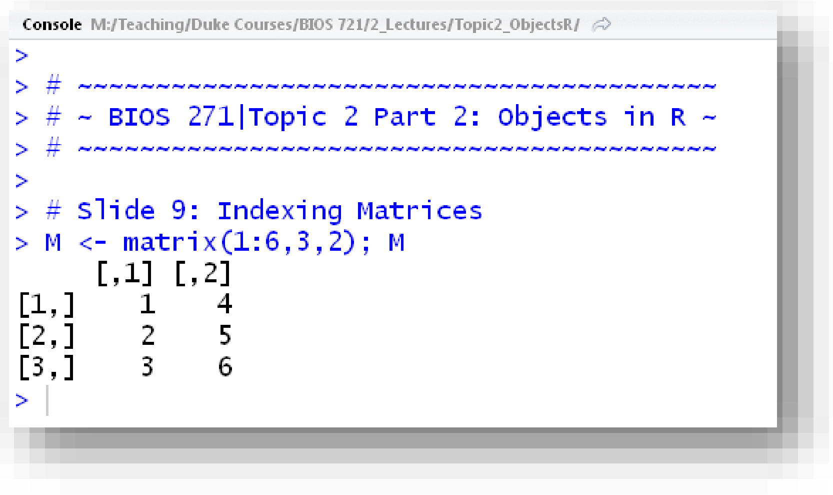
```
M[2,1]
M[1,2]
M[,1]
M[2:3,]
M[,-1]
```

Indexing

- To select elements in a matrix, indicate the position of the elements you want to retain or drop within square brackets next the object name.
  - Indexing Matrices: `object[row(s),column(s)]`

  - Example| Consider the matrix `M <- matrix(1:6,3,2)`

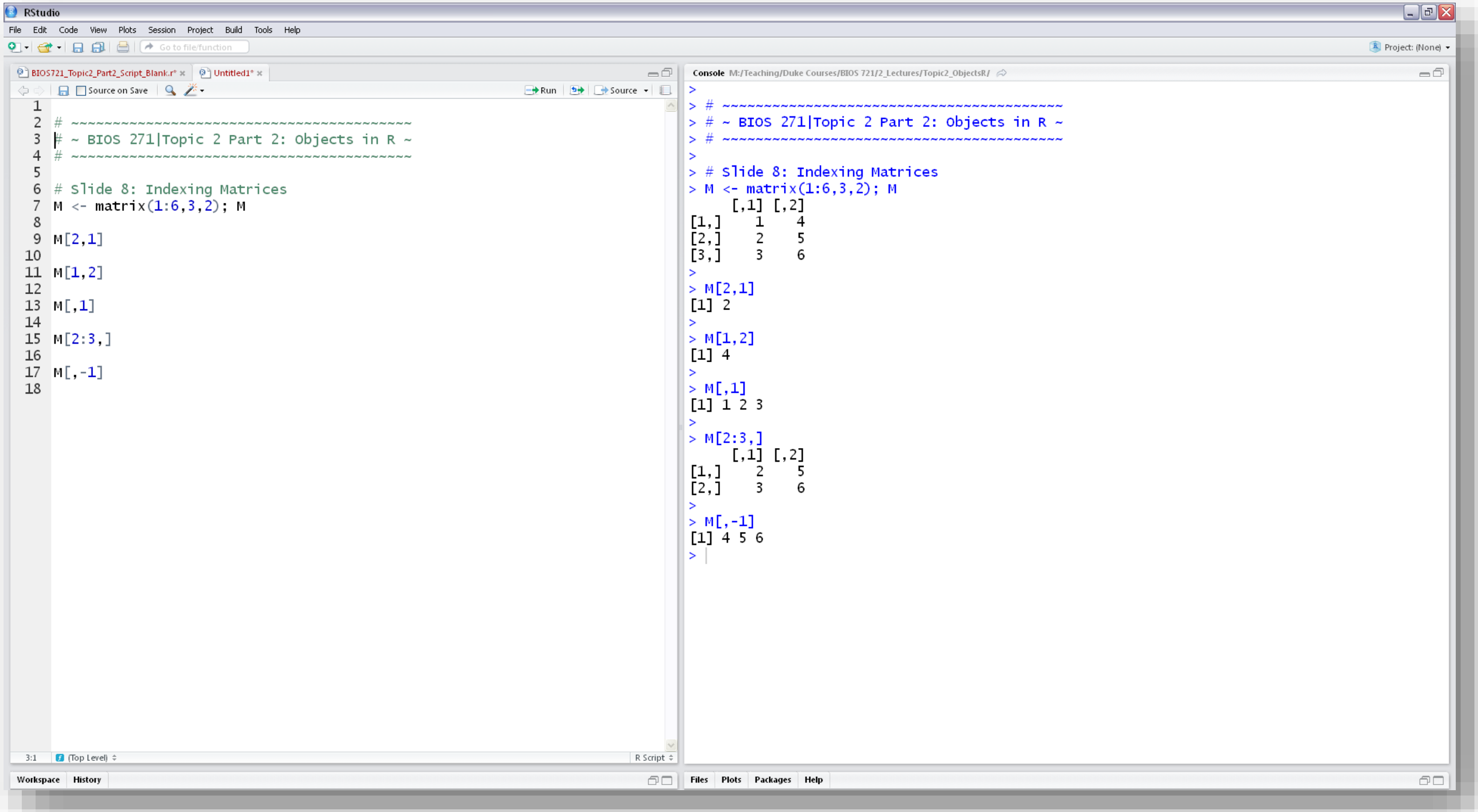
```
M[2,1]
M[1,2]
M[,1]
M[2:3,]
M[,-1]
```


Console M:/Teaching/Duke Courses/BIOS 721/2_Lectures/Topic2_ObjectsR/

```
>
> # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
> # ~ BIOS 271|Topic 2 Part 2: Objects in R ~
> # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
>
> # Slide 9: Indexing Matrices
> M <- matrix(1:6,3,2); M
     [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6
>
```

Indexing

- Indexing Matrices


RStudio
File Edit Code View Plots Session Project Build Tools Help
Go to file/function
Project: (None)
BIOS721_Topic2_Part2_Script_Blank.r*
Untitled1*
Source on Save
Run
Source
Console M:/Teaching/Duke Courses/BIOS 721/2_Lectures/Topic2_ObjectsR/
3:1
(Top Level)
R Script
Workspace
History
Files
Plots
Packages
Help

```
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# ~ BIOS 271|Topic 2 Part 2: Objects in R ~
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# Slide 8: Indexing Matrices
M <- matrix(1:6,3,2); M

M[2,1]

M[1,2]

M[,1]

M[2:3,]

M[,-1]
```

```
>
> # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
> # ~ BIOS 271|Topic 2 Part 2: Objects in R ~
> # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
>
> # Slide 8: Indexing Matrices
> M <- matrix(1:6,3,2); M
     [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6
>
> M[2,1]
[1] 2
>
> M[1,2]
[1] 4
>
> M[,1]
[1] 1 2 3
>
> M[2:3,]
     [,1] [,2]
[1,]    2    5
[2,]    3    6
>
> M[,-1]
[1] 4 5 6
>
```

Indexing

- Can also index matrices using row and column names to identify the elements that you would like to retain.
  - Indexing Matrices: `object["rownames","columnnames"]`
  - Example| Consider the matrix

```
M <- matrix(1:4,2,2,byrow=TRUE)
rownames(M) <- c('r1','r2')
colnames(M) <- c('C1','C2')

M['r2','C2']
M['r2','c2']
M['r1',]
M[,'C1']
```

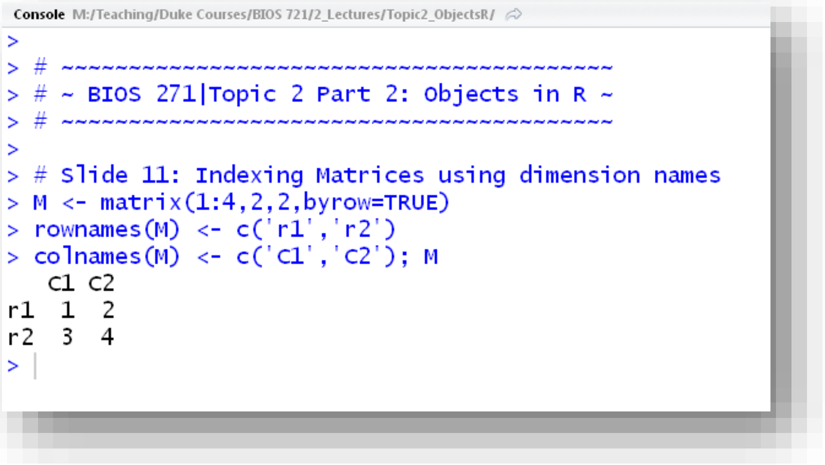
Indexing

- Can also index matrices using row and column names to identify the elements that you would like to retain.
  - Indexing Matrices: `object["rownames","columnnames"]`

  - **Example|** Consider the matrix

```
M <- matrix(1:4,2,2,byrow=TRUE)
rownames(M) <- c('r1','r2')
colnames(M) <- c('C1','C2')
```

```
M['r2','C2']
M['r2','c2']
M['r1',]
M[,'C1']
```

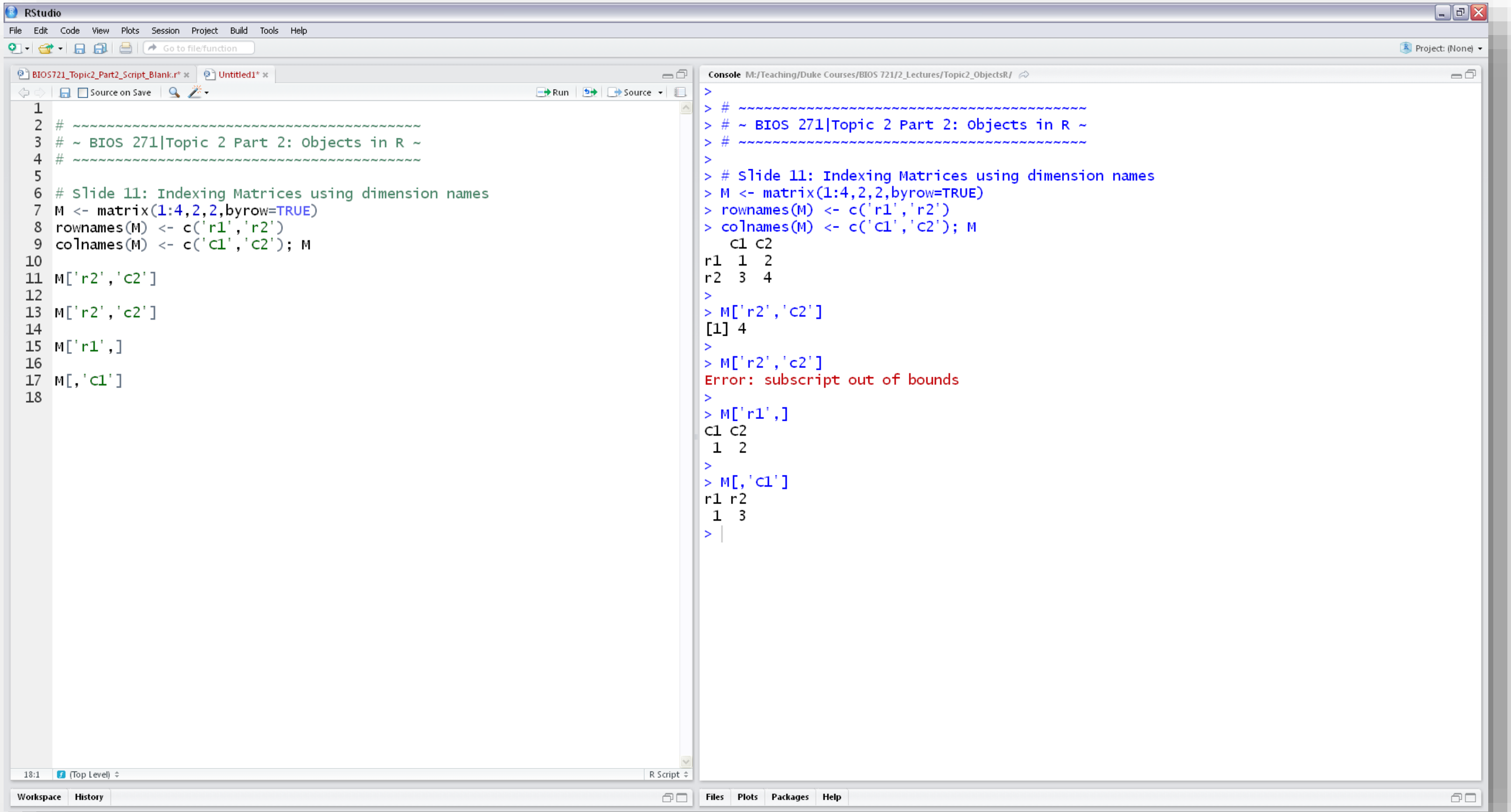


Console M:/Teaching/Duke Courses/BIOS 721/2_Lectures/Topic2_ObjectsR/

```
>
> # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
> # ~ BIOS 271|Topic 2 Part 2: Objects in R ~
> # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
>
> # Slide 11: Indexing Matrices using dimension names
> M <- matrix(1:4,2,2,byrow=TRUE)
> rownames(M) <- c('r1','r2')
> colnames(M) <- c('C1','C2'); M
   C1 C2
r1  1  2
r2  3  4
> 
```

Indexing

- Indexing Matrices


RStudio
File Edit Code View Plots Session Project Build Tools Help
Go to file/function
Project: (None)
BIOS721_Topic2_Part2_Script_Blank.r*
Untitled1*
Source on Save
Run
Source
Console M:/Teaching/Duke Courses/BIOS 721/2_Lectures/Topic2_ObjectsR/
18:1
(Top Level)
R Script
Workspace
History
Files
Plots
Packages
Help

```

# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# ~ BIOS 271|Topic 2 Part 2: Objects in R ~
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# Slide 11: Indexing Matrices using dimension names
M <- matrix(1:4,2,2,byrow=TRUE)
rownames(M) <- c('r1','r2')
colnames(M) <- c('C1','C2'); M

M['r2','C2']

M['r2','c2']

M['r1',]

M[,'C1']

```

```
>
> # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
> # ~ BIOS 271|Topic 2 Part 2: Objects in R ~
> # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
>
> # Slide 11: Indexing Matrices using dimension names
> M <- matrix(1:4,2,2,byrow=TRUE)
> rownames(M) <- c('r1','r2')
> colnames(M) <- c('C1','C2'); M
   C1 C2
r1  1  2
r2  3  4
>
> M['r2','C2']
[1] 4
>
> M['r2','c2']
Error: subscript out of bounds
>
> M['r1',]
C1 C2
 1  2
>
> M[,'C1']
r1 r2
 1  3
>
```

Indexing

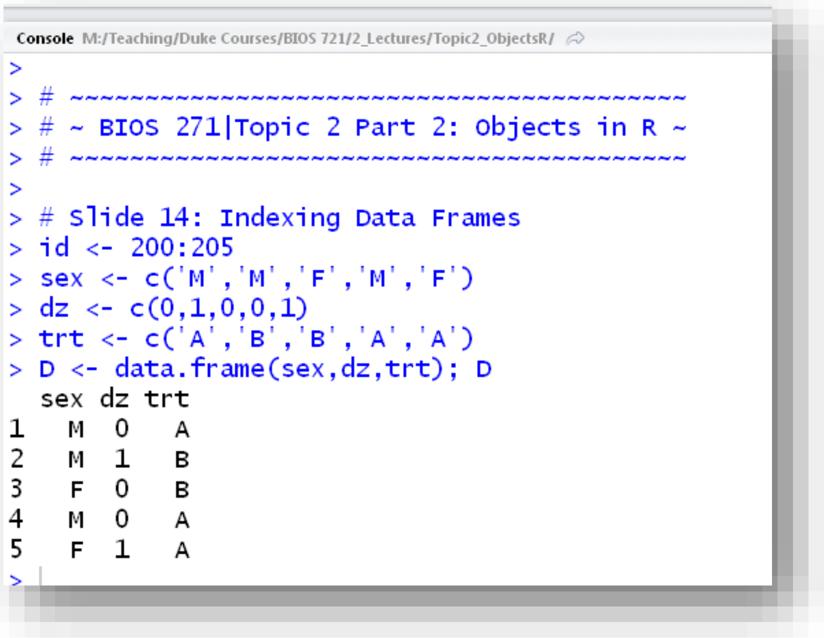
- To select elements in a data frame, indicate the position of the elements you want to retain or drop within square brackets next the object name.
  - Indexing Data Frames: `object[row(s),column(s)]`
  - Example| Consider the data frame

```
id <- 200:205
sex <- c('M','M','F','M','F')
dz <- c(0,1,0,0,1)
trt <- c('A','B','B','A','A')
D <- data.frame(sex,dz,trt)

D[1,3]
D[2:4,2]
D[,1]
```

Indexing

- To select elements in a data frame, indicate the position of the elements you want to retain or drop within square brackets next the object name.
  - Indexing Data Frame: `object[row(s),column(s)]`
  - Example| Consider the data frame

```
id <- 200:205
 sex <- c('M','M','F',
 dz <- c(0,1,0,0,1)
 trt <- c('A','B','B',
 D <- data.frame(sex,d

 D[1,3]
 D[2:4,2]
 D[,1]
```

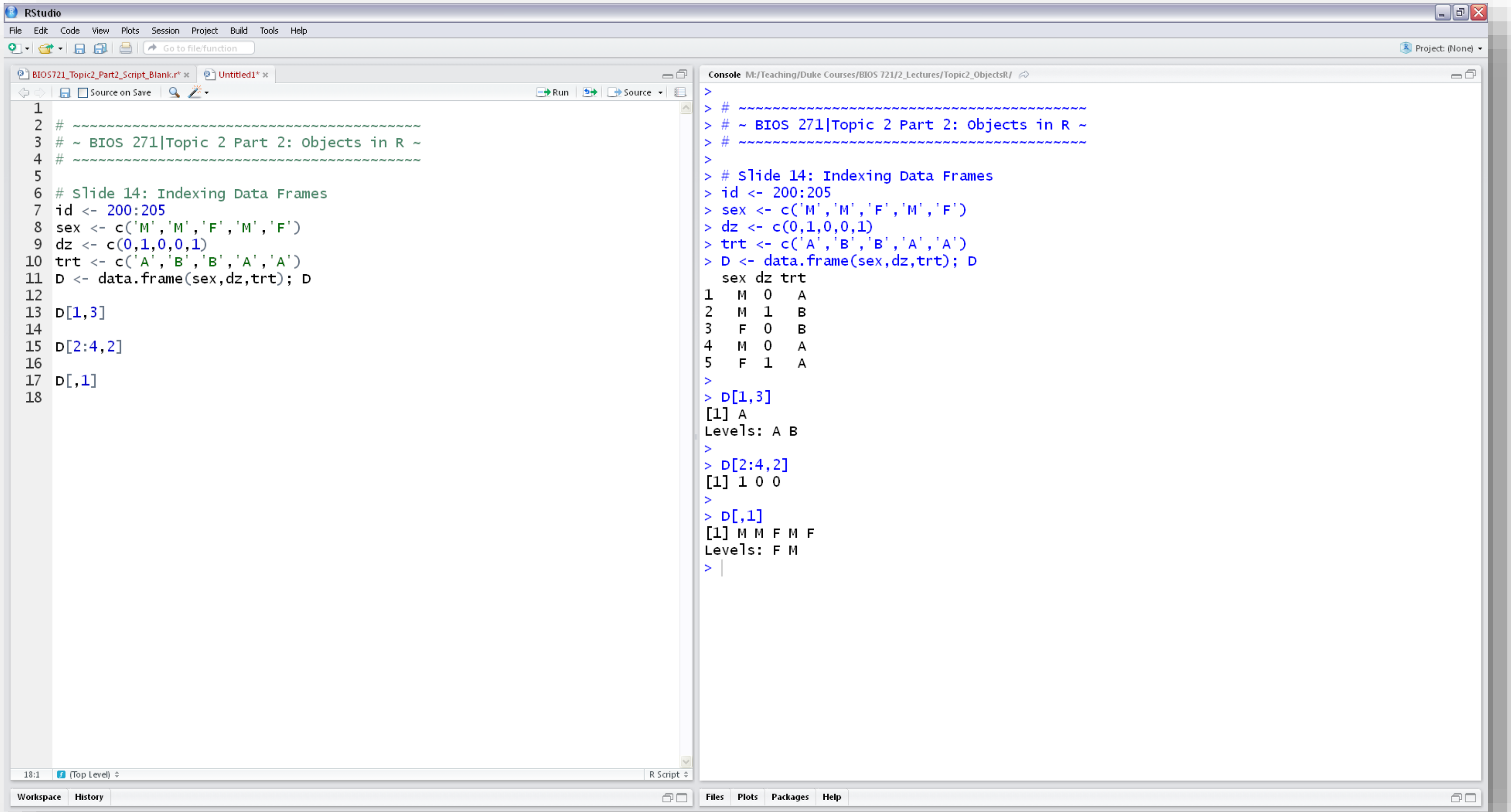


Console M:/Teaching/Duke Courses/BIOS 721/2_Lectures/Topic2_ObjectsR/

```
>
> # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
> # ~ BIOS 271|Topic 2 Part 2: Objects in R ~
> # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
>
> # Slide 14: Indexing Data Frames
> id <- 200:205
> sex <- c('M','M','F','M','F')
> dz <- c(0,1,0,0,1)
> trt <- c('A','B','B','A','A')
> D <- data.frame(sex,dz,trt); D
  sex dz trt
1   M  0   A
2   M  1   B
3   F  0   B
4   M  0   A
5   F  1   A
>
```

Indexing

- Indexing Data Frames


RStudio
File Edit Code View Plots Session Project Build Tools Help
Go to file/function
Project: (None)
BIOS721_Topic2_Part2_Script_Blank.r*
Untitled1*
Source on Save
Run
Source
Console M:/Teaching/Duke Courses/BIOS 721/2_Lectures/Topic2_ObjectsR/
18:1
(Top Level)
R Script
Workspace
History
Files
Plots
Packages
Help

```r

# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# ~ BIOS 271|Topic 2 Part 2: Objects in R ~
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# Slide 14: Indexing Data Frames
id <- 200:205
sex <- c('M','M','F','M','F')
dz <- c(0,1,0,0,1)
trt <- c('A','B','B','A','A')
D <- data.frame(sex,dz,trt); D

D[1,3]

D[2:4,2]

D[,1]

```

```
>
> # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
> # ~ BIOS 271|Topic 2 Part 2: Objects in R ~
> # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
>
> # Slide 14: Indexing Data Frames
> id <- 200:205
> sex <- c('M','M','F','M','F')
> dz <- c(0,1,0,0,1)
> trt <- c('A','B','B','A','A')
> D <- data.frame(sex,dz,trt); D
  sex dz trt
1   M  0   A
2   M  1   B
3   F  0   B
4   M  0   A
5   F  1   A
>
> D[1,3]
[1] A
Levels: A B
>
> D[2:4,2]
[1] 1 0 0
>
> D[,1]
[1] M M F M F
Levels: F M
> 
```

Indexing

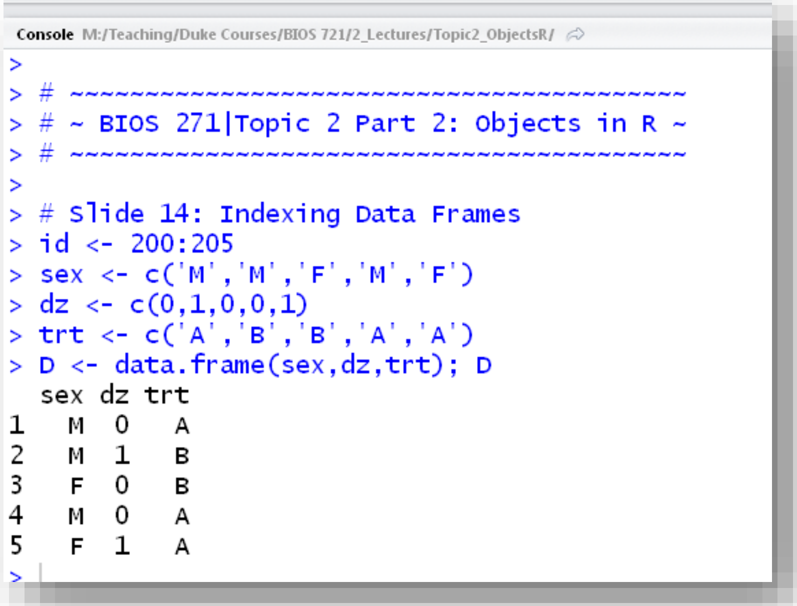
- Can also index data frames using the column names to identify the columns that you would like to retain.
  - Indexing Data Frames: `object[row(s),"columnnames"]`
    `object$columnname[row(s)]`
  - Example| Consider the data frame

```
id <- 200:205
 sex <- c('M','M','F','M','F')
 dz <- c(0,1,0,0,1)
 trt <- c('A','B','B','A','A')
 D <- data.frame(sex,dz,trt)

 D[1,'sex']
 D[,'trt']
 D$trt
 D$dz[c(1,4)]
```

Indexing

- Can also index data frames using the column names to identify the columns that you would like to retain.
  - Indexing Data Frames: `object[row(s),"columnnames"]`
    `object$columnname[row(s)]`
  - Example| Consider the data frame

```
id <- 200:205
sex <- c('M','M','F',
dz <- c(0,1,0,0,1)
trt <- c('A','B','B',
D <- data.frame(sex,d

D[1,'sex']
D[,'trt']
D$trt
D$dz[c(1,4)]
```

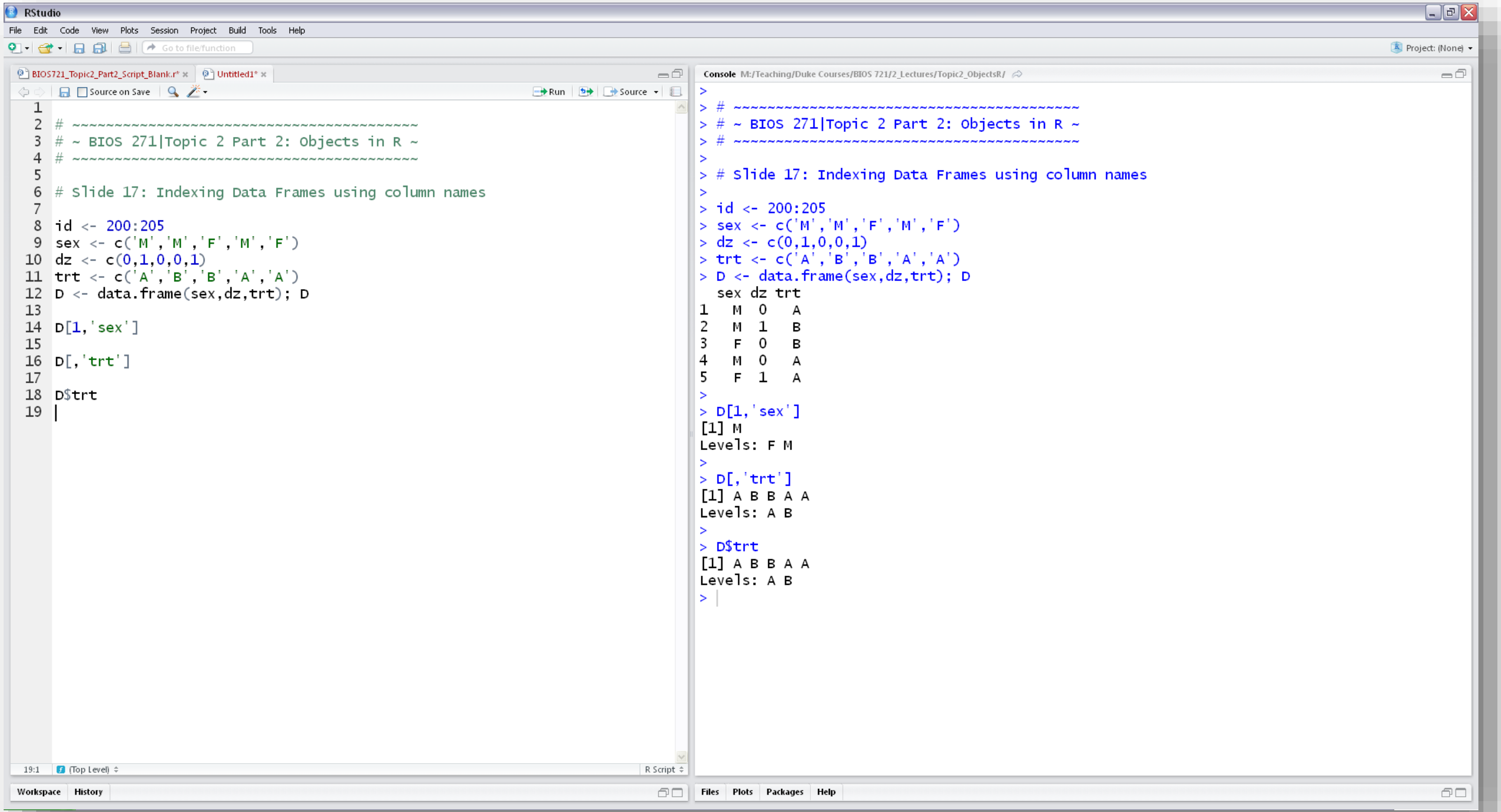


Console M:/Teaching/Duke Courses/BIOS 721/2_Lectures/Topic2_ObjectsR/

```
>
> # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
> # ~ BIOS 271|Topic 2 Part 2: Objects in R ~
> # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
>
> # Slide 14: Indexing Data Frames
> id <- 200:205
> sex <- c('M','M','F','M','F')
> dz <- c(0,1,0,0,1)
> trt <- c('A','B','B','A','A')
> D <- data.frame(sex,dz,trt); D
  sex dz trt
1   M  0   A
2   M  1   B
3   F  0   B
4   M  0   A
5   F  1   A
>
```

Indexing

□ Indexing Data Frames

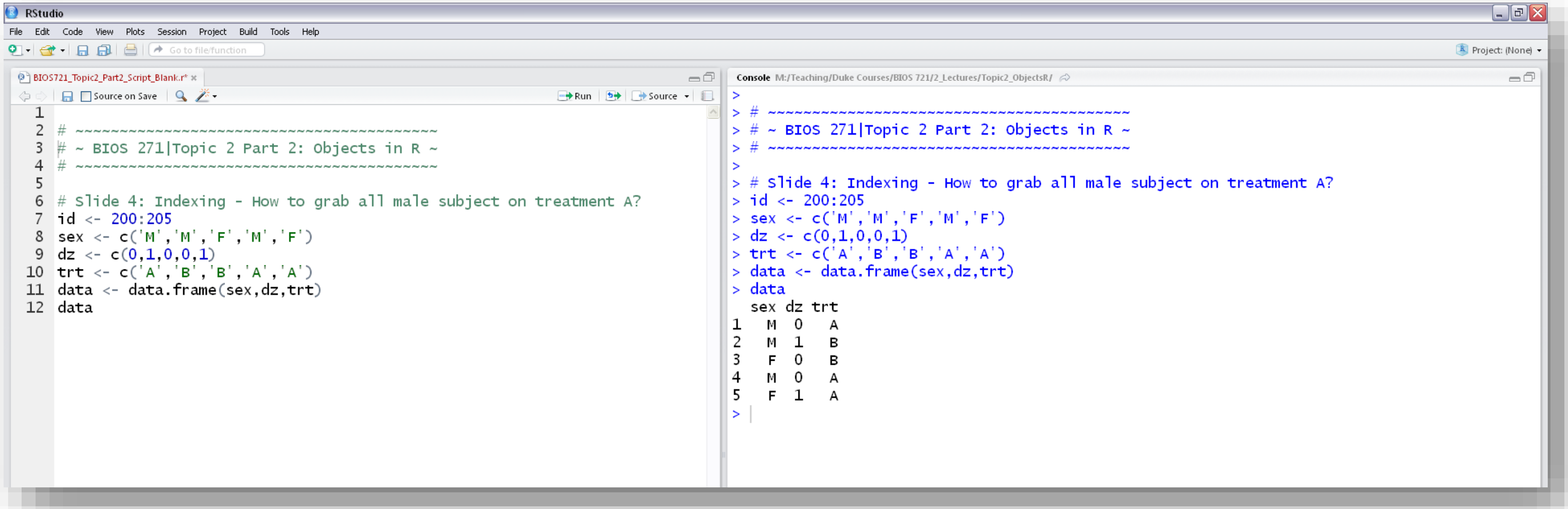

RStudio
File Edit Code View Plots Session Project Build Tools Help
Go to file/function
Project: (None)
BIOS721_Topic2_Part2_Script_Blank.r*
Untitled1*
Source on Save
Run
Source
Console M:/Teaching/Duke Courses/BIOS 721/2_Lectures/Topic2_ObjectsR/
19:1
(Top Level)
R Script
Workspace
History
Files
Plots
Packages
Help

```

# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# ~ BIOS 271|Topic 2 Part 2: Objects in R ~
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# Slide 17: Indexing Data Frames using column names

id <- 200:205
sex <- c('M','M','F','M','F')
dz <- c(0,1,0,0,1)
trt <- c('A','B','B','A','A')
D <- data.frame(sex,dz,trt); D

D[1,'sex']

D[,'trt']

D$trt

```

```
>
> # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
> # ~ BIOS 271|Topic 2 Part 2: Objects in R ~
> # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
>
> # Slide 17: Indexing Data Frames using column names
>
> id <- 200:205
> sex <- c('M','M','F','M','F')
> dz <- c(0,1,0,0,1)
> trt <- c('A','B','B','A','A')
> D <- data.frame(sex,dz,trt); D
  sex dz trt
1   M  0   A
2   M  1   B
3   F  0   B
4   M  0   A
5   F  1   A
>
> D[1,'sex']
[1] M
Levels: F M
>
> D[,'trt']
[1] A B B A A
Levels: A B
>
> D$trt
[1] A B B A A
Levels: A B
>
```

Indexing – Motivating Example

- Recall this example?
  - Consider the data frame below. Suppose we wanted to create another data frame that subsets to all male subjects who were exposed to treatment A.
  - How could we get R to identify these subjects using code?
    - Indexing!
    - Using indexing, could create a data frame called sub that contains this subset of the data.

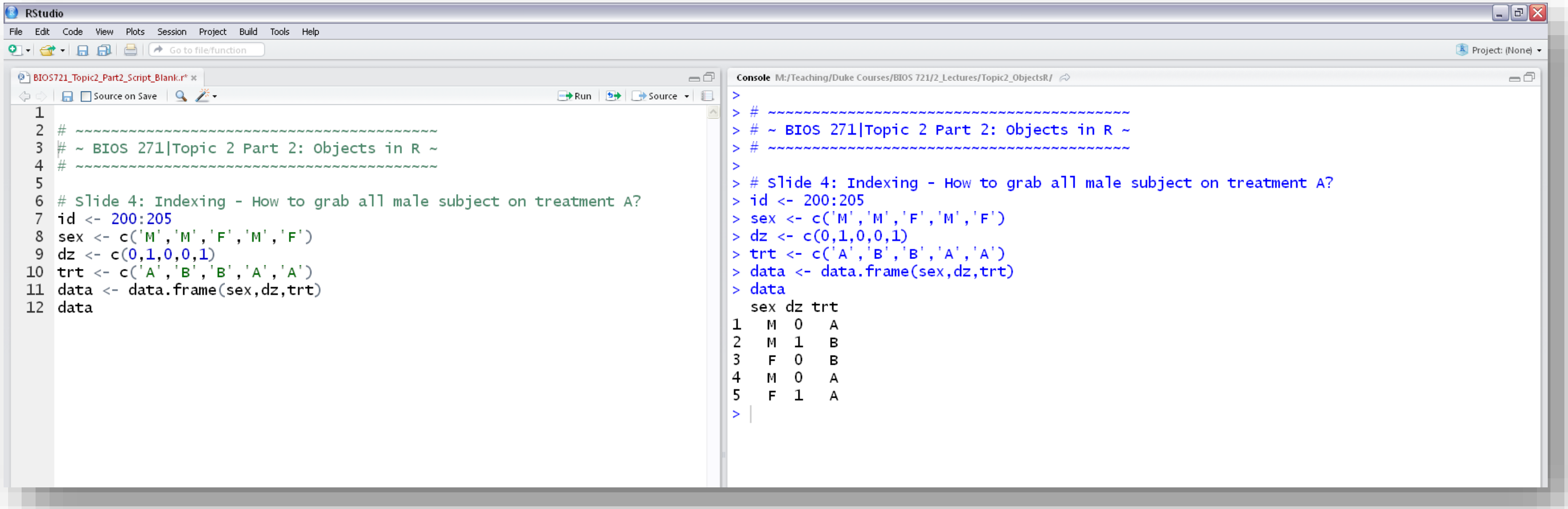

RStudio
File Edit Code View Plots Session Project Build Tools Help
Go to file/function
Project: (None)
BIOS721_Topic2_Part2_Script_Blank.r*
Source on Save
Run
Source
Console M:/Teaching/Duke Courses/BIOS 721/2_Lectures/Topic2_ObjectsR/

```r

# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# ~ BIOS 271|Topic 2 Part 2: Objects in R ~
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# Slide 4: Indexing - How to grab all male subject on treatment A?
id <- 200:205
sex <- c('M','M','F','M','F')
dz <- c(0,1,0,0,1)
trt <- c('A','B','B','A','A')
data <- data.frame(sex,dz,trt)
data
```

```
>
> # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
> # ~ BIOS 271|Topic 2 Part 2: Objects in R ~
> # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
>
> # Slide 4: Indexing - How to grab all male subject on treatment A?
> id <- 200:205
> sex <- c('M','M','F','M','F')
> dz <- c(0,1,0,0,1)
> trt <- c('A','B','B','A','A')
> data <- data.frame(sex,dz,trt)
> data
  sex dz trt
1   M  0   A
2   M  1   B
3   F  0   B
4   M  0   A
5   F  1   A
> 
```

Indexing – Motivating Example

- Recall this example?
  - For solution – come to class!

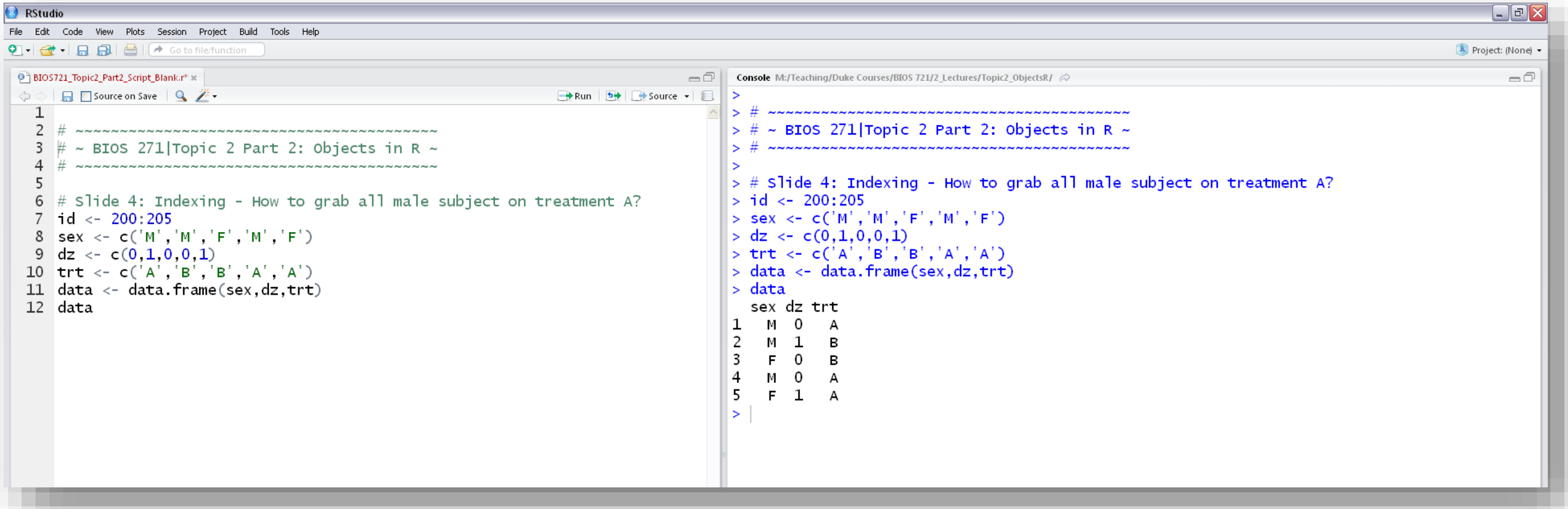

RStudio
File Edit Code View Plots Session Project Build Tools Help
Go to file/function
Project: (None)
BIOS721_Topic2_Part2_Script_Blank.r*
Source on Save
Run
Source
Console M:/Teaching/Duke Courses/BIOS 721/2_Lectures/Topic2_ObjectsR/

```
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# ~ BIOS 271|Topic 2 Part 2: Objects in R ~
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# Slide 4: Indexing - How to grab all male subject on treatment A?
id <- 200:205
sex <- c('M','M','F','M','F')
dz <- c(0,1,0,0,1)
trt <- c('A','B','B','A','A')
data <- data.frame(sex,dz,trt)
data
```

```
>
> # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
> # ~ BIOS 271|Topic 2 Part 2: Objects in R ~
> # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
>
> # Slide 4: Indexing - How to grab all male subject on treatment A?
> id <- 200:205
> sex <- c('M','M','F','M','F')
> dz <- c(0,1,0,0,1)
> trt <- c('A','B','B','A','A')
> data <- data.frame(sex,dz,trt)
> data
  sex dz trt
1   M  0   A
2   M  1   B
3   F  0   B
4   M  0   A
5   F  1   A
>
```

Indexing – Motivating Example

- Recall this example?
  - To create the data frame sub on the previous slide, we had to "hard code" indexing – ok for this example because the data set is small, but what if there were thousands on observations?
  - We would like to use code to automate identifying these subjects
    - Need to use indexing in conjunction with R operators and functions!



```

# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# ~ BIOS 271|Topic 2 Part 2: Objects in R ~
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# Slide 4: Indexing - How to grab all male subject on treatment A?
id <- 200:205
sex <- c('M','M','F','M','F')
dz <- c(0,1,0,0,1)
trt <- c('A','B','B','A','A')
data <- data.frame(sex,dz,trt)
data
```

```
>
> # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
> # ~ BIOS 271|Topic 2 Part 2: Objects in R ~
> # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
>
> # Slide 4: Indexing - How to grab all male subject on treatment A?
> id <- 200:205
> sex <- c('M','M','F','M','F')
> dz <- c(0,1,0,0,1)
> trt <- c('A','B','B','A','A')
> data <- data.frame(sex,dz,trt)
> data
  sex dz trt
1   M  0   A
2   M  1   B
3   F  0   B
4   M  0   A
5   F  1   A
> 
```

R Operators vs. R Functions

- R Operators
  - Set of supported language constructs (i.e. symbols) that perform operations on programming objects.
  - For example, + for addition

- R Functions
  - Packaged set of commands designed to perform a specific task.
  - Can be simple like `mean(x, …)` to calculate the mean of a data object x.
  - Can also be complex like `lm(y~x, …)` to estimate the linear relationship between the variables y and x.
    - For now, we will focus on simple R functions that everyone should have in their toolbox … actually not sure how you would program without knowing these functions.

R Operators

- + → addition
- − → subtraction
- * → multiplication
- / → division
- ^ → raise to a power
- %% → remainder
- %*% → matrix mult

- < → less than?
- > → greater than?
- <= → less than or equal?
- >= → greater than or equal to?
- ! → not
- == → equal to?
- != → not equal to?
- & → and
- | → or

R Operators

- + → addition
- − → subtraction
- * → multiplication
- / → division
- ^ → raise to a power
- %% → remainder
- %*% → matrix mult

Mathematical Operators (return #s)
vs.
Logical Operators (return TRUE/FALSE)

- < → less than?
- > → greater than?
- <= → less than or equal?
- >= → greater than or equal to?
- ! → not
- == → equal to?
- != → not equal to?
- & → and
- | → or

R Operators

- □ + → addition
- □ – → subtraction
- □ * → multiplication
- □ / → division
- □ ^ → raise to a power
- □ %% → remainder
- □ %*% → matrix mult

Mathematical Operators (return #s)
vs.
Logical Operators (return TRUE/FALSE)

- □ < → less than?
- □ > → greater than?
- □ <= → less than or equal?
- □ >= → greater than or equal to?
- □ ! → not
- □ == → equal to?
- □ != → not equal to?
- □ & → and
- □ | → or

R Operators

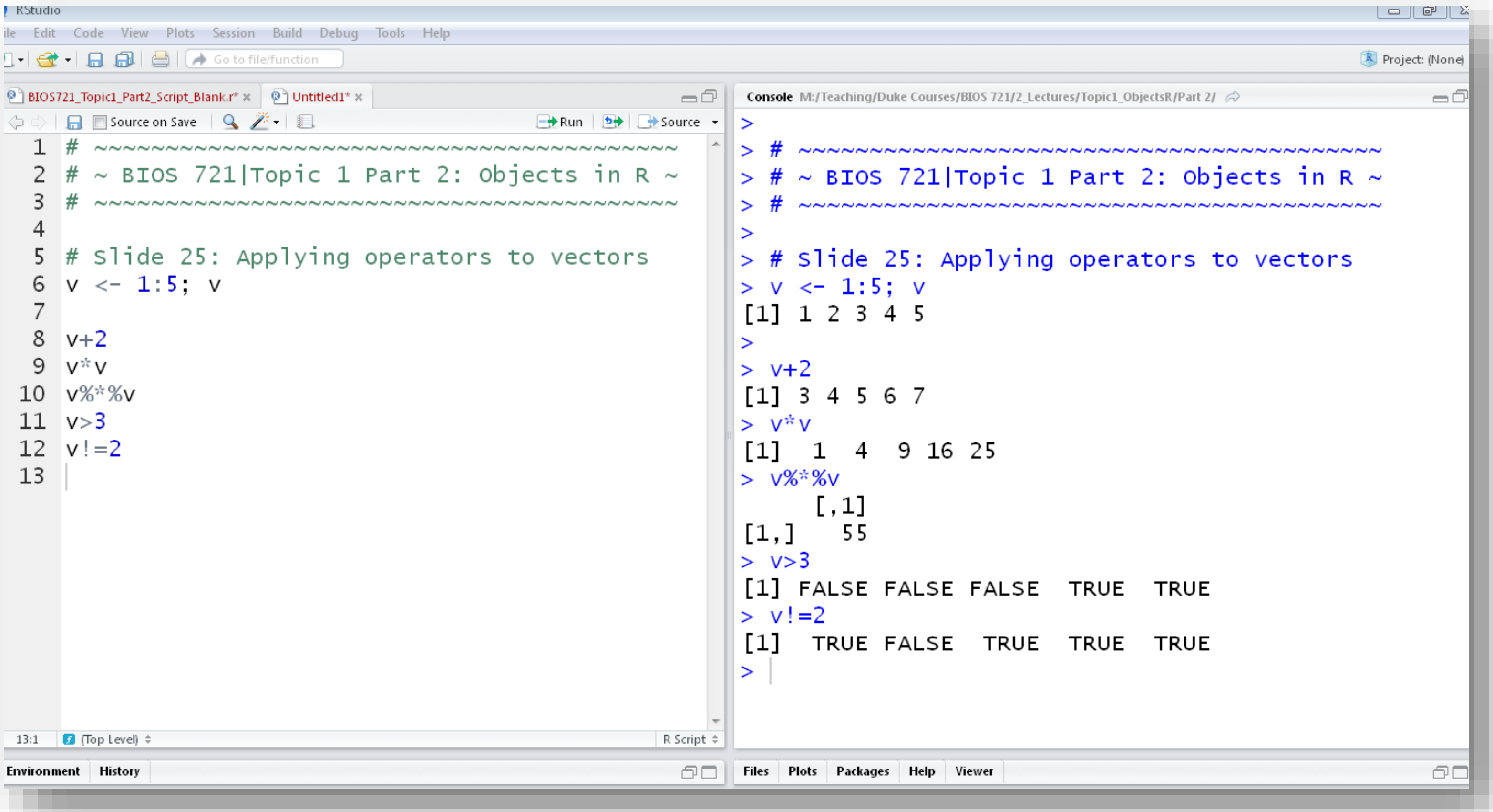
- These operators can be applied to any* object!
  - Will execute them element-wise (except for `%*%`)
    - That is, if applied to vector, will return a vector.
    - I.e. R is a 'vectorized' language – really useful trick
  - * Any compatible object (i.e. can't add two character values)

  - Example| Consider the vector `v <- 1:5`

```
v+2
v*v
v%*%v
v>3
v!=2
```

R Operators

- R applies operators element-wise

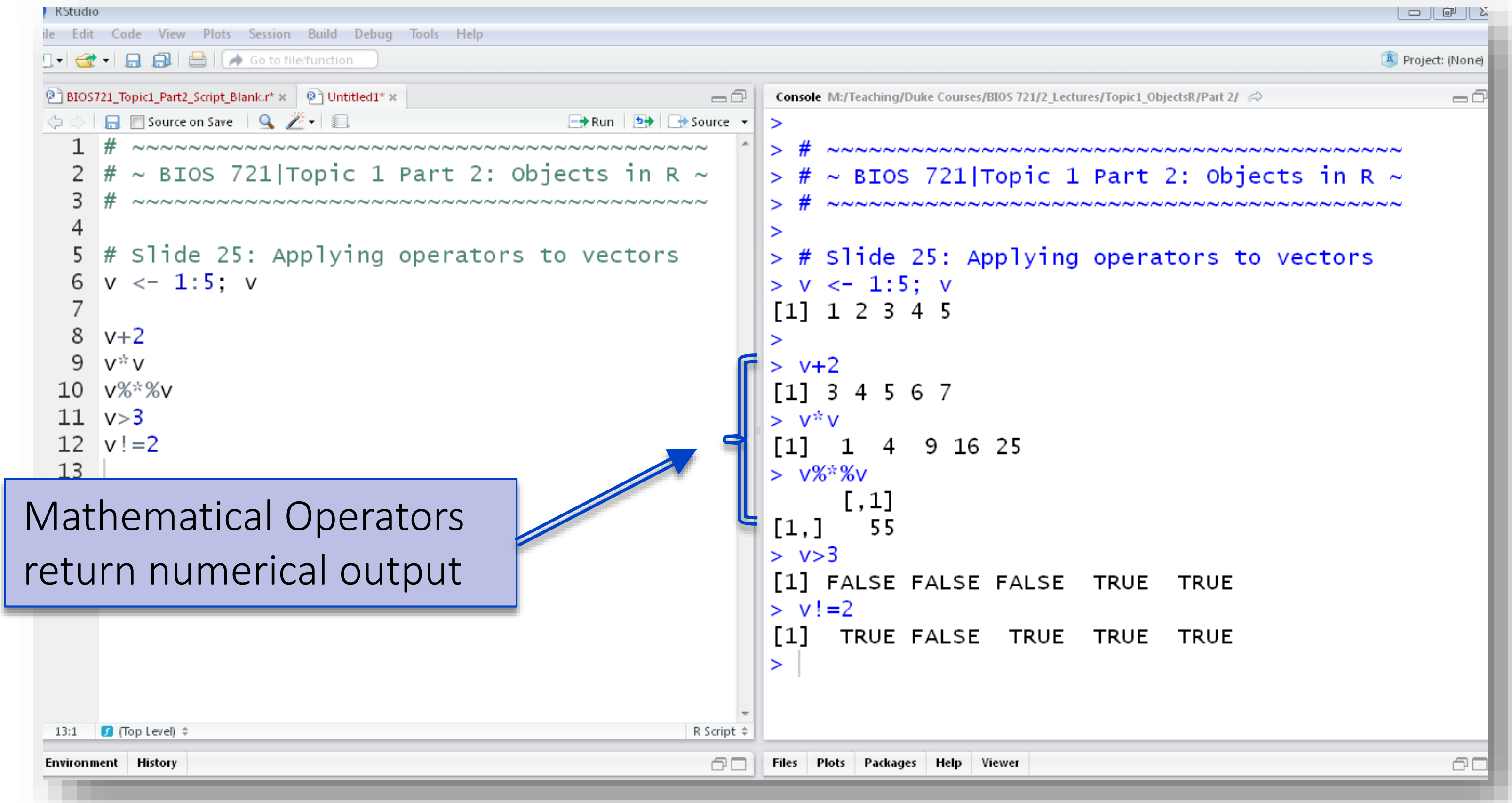

RStudio
ile Edit Code View Plots Session Build Debug Tools Help
Go to file/function
Project: (None)
BIOS721_Topic1_Part2_Script_Blank.r*
Untitled1*
Source on Save
Run
Source
Console M:/Teaching/Duke Courses/BIOS 721/2_Lectures/Topic1_ObjectsR/Part 2/
13:1
(Top Level)
R Script
Environment
History
Files
Plots
Packages
Help
Viewer

```r
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# ~ BIOS 721|Topic 1 Part 2: Objects in R ~
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# Slide 25: Applying operators to vectors
v <- 1:5; v

v+2
v*v
v%*%v
v>3
v!=2

```

```
>
> # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
> # ~ BIOS 721|Topic 1 Part 2: Objects in R ~
> # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
>
> # Slide 25: Applying operators to vectors
> v <- 1:5; v
[1] 1 2 3 4 5
>
> v+2
[1] 3 4 5 6 7
> v*v
[1]  1  4  9 16 25
> v%*%v
     [,1]
[1,]   55
> v>3
[1] FALSE FALSE FALSE  TRUE  TRUE
> v!=2
[1]  TRUE FALSE  TRUE  TRUE  TRUE
>
```

R Operators

- R applies operators element-wise


RStudio
ile Edit Code View Plots Session Build Debug Tools Help
Go to file/function
Project: (None)
BIOS721_Topic1_Part2_Script_Blank.r*
Untitled1*
Source on Save
Run
Source
Console M:/Teaching/Duke Courses/BIOS 721/2_Lectures/Topic1_ObjectsR/Part 2/
13:1
(Top Level)
R Script
Environment
History
Files
Plots
Packages
Help
Viewer

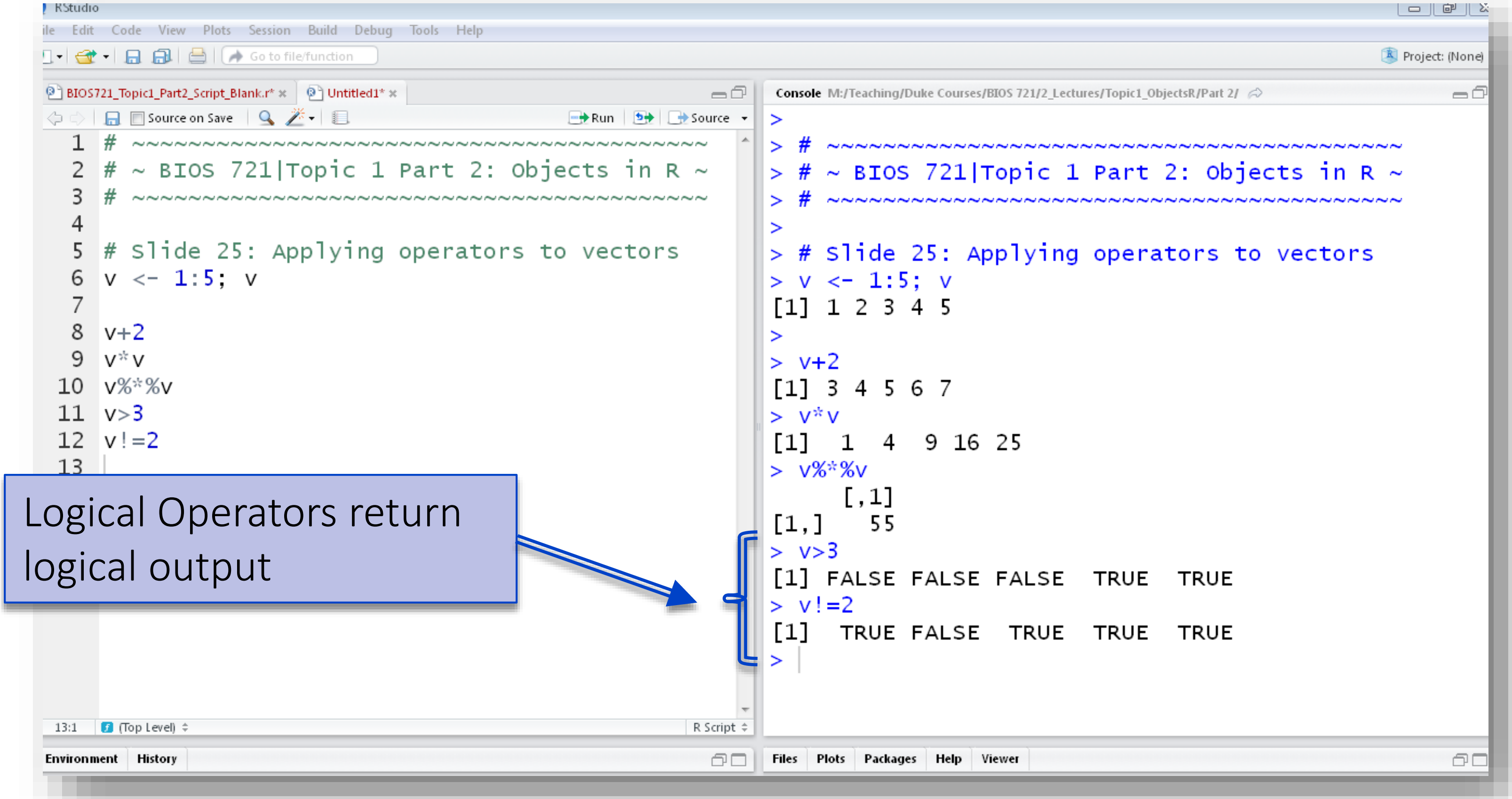
```
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# ~ BIOS 721|Topic 1 Part 2: Objects in R ~
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# Slide 25: Applying operators to vectors
v <- 1:5; v

v+2
v*v
v%*%v
v>3
v!=2
```

```
>
> # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
> # ~ BIOS 721|Topic 1 Part 2: Objects in R ~
> # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
>
> # Slide 25: Applying operators to vectors
> v <- 1:5; v
[1] 1 2 3 4 5
>
> v+2
[1] 3 4 5 6 7
> v*v
[1]  1  4  9 16 25
> v%*%v
     [,1]
[1,]   55
> v>3
[1] FALSE FALSE FALSE  TRUE  TRUE
> v!=2
[1]  TRUE FALSE  TRUE  TRUE  TRUE
>
```

Mathematical Operators return numerical output

R Operators

- R applies operators element-wise



RStudio
ile Edit Code View Plots Session Build Debug Tools Help
Go to file/function
Project: (None)
BIOS721_Topic1_Part2_Script_Blank.r*
Untitled1*
Source on Save
Run
Source
Console M:/Teaching/Duke Courses/BIOS 721/2_Lectures/Topic1_ObjectsR/Part 2/
13:1
(Top Level)
R Script
Environment
History
Files
Plots
Packages
Help
Viewer

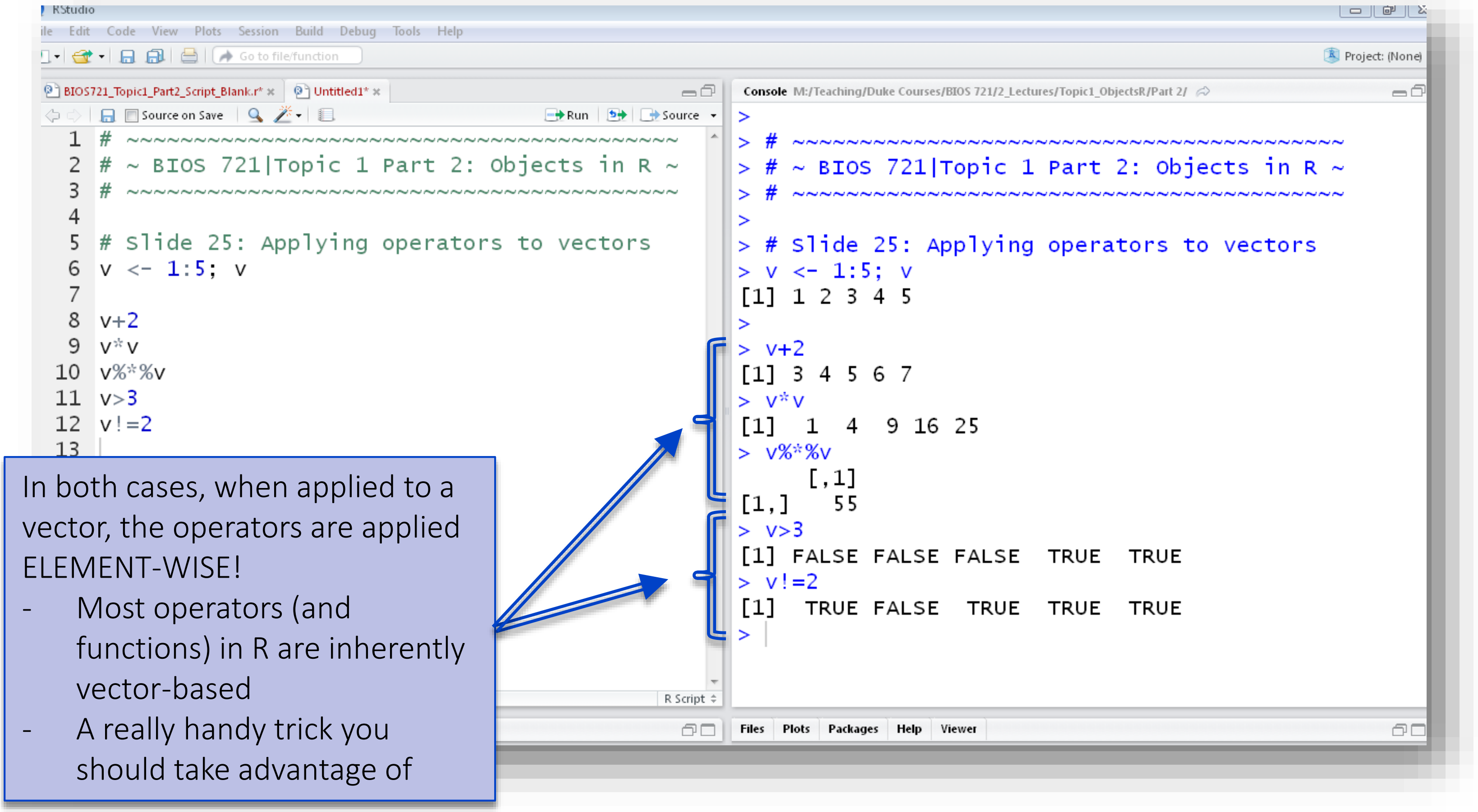
```
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# ~ BIOS 721|Topic 1 Part 2: Objects in R ~
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# Slide 25: Applying operators to vectors
v <- 1:5; v

v+2
v*v
v%*%v
v>3
v!=2

```

```
>
> # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
> # ~ BIOS 721|Topic 1 Part 2: Objects in R ~
> # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
>
> # Slide 25: Applying operators to vectors
> v <- 1:5; v
[1] 1 2 3 4 5
>
> v+2
[1] 3 4 5 6 7
> v*v
[1]  1  4  9 16 25
> v%*%v
     [,1]
[1,]   55
> v>3
[1] FALSE FALSE FALSE  TRUE  TRUE
> v!=2
[1]  TRUE FALSE  TRUE  TRUE  TRUE
>
```

Logical Operators return logical output

R Operators

- R applies operators element-wise


RStudio
ile Edit Code View Plots Session Build Debug Tools Help
Go to file/function
Project: (None)
BIOS721_Topic1_Part2_Script_Blank.r*
Untitled1*
Source on Save
Run
Source
Console M:/Teaching/Duke Courses/BIOS 721/2_Lectures/Topic1_ObjectsR/Part 2/
R Script
Files Plots Packages Help Viewer

```
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# ~ BIOS 721|Topic 1 Part 2: Objects in R ~
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# Slide 25: Applying operators to vectors
v <- 1:5; v

v+2
v*v
v%*%v
v>3
v!=2

```

```
>
> # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
> # ~ BIOS 721|Topic 1 Part 2: Objects in R ~
> # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
>
> # Slide 25: Applying operators to vectors
> v <- 1:5; v
[1] 1 2 3 4 5
>
> v+2
[1] 3 4 5 6 7
> v*v
[1]  1  4  9 16 25
> v%*%v
     [,1]
[1,]   55
> v>3
[1] FALSE FALSE FALSE  TRUE  TRUE
> v!=2
[1]  TRUE FALSE  TRUE  TRUE  TRUE
>
```

In both cases, when applied to a vector, the operators are applied ELEMENT-WISE!

- Most operators (and functions) in R are inherently vector-based
- A really handy trick you should take advantage of

R Operators

- The results are the same for matrices.
  - That is, will execute them element-wise (except for `%*%`)
    - I.e. R is a ‘vectorized’ language – really useful trick
    - Makes sense – a matrix is just an organized vector …
  - Example| Consider the matrices

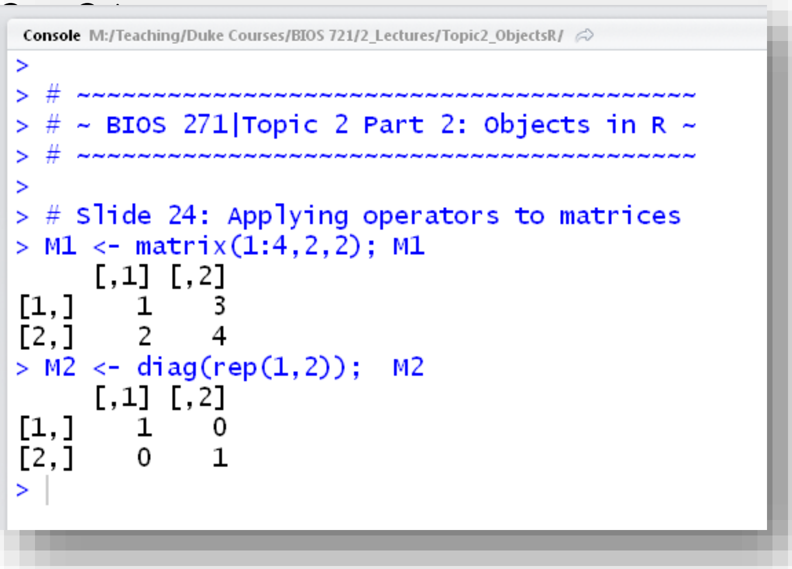
```
M1 <- matrix(1:4,2,2)
M2 <- diag(rep(1,2))

M1/2
M1-M2
M1==3
```

R Operators

- The results are the same for matrices.
  - That is, will execute them element-wise (except for `%*%`)
    - I.e. R is a ‘vectorized’ language – really useful trick
    - Makes sense – a matrix is just an organized vector …

  - Example| Consider the matrices

```
M1 <- matrix(1:4,
M2 <- diag(rep(1,

M1/2
M1-M2
M1==3
```

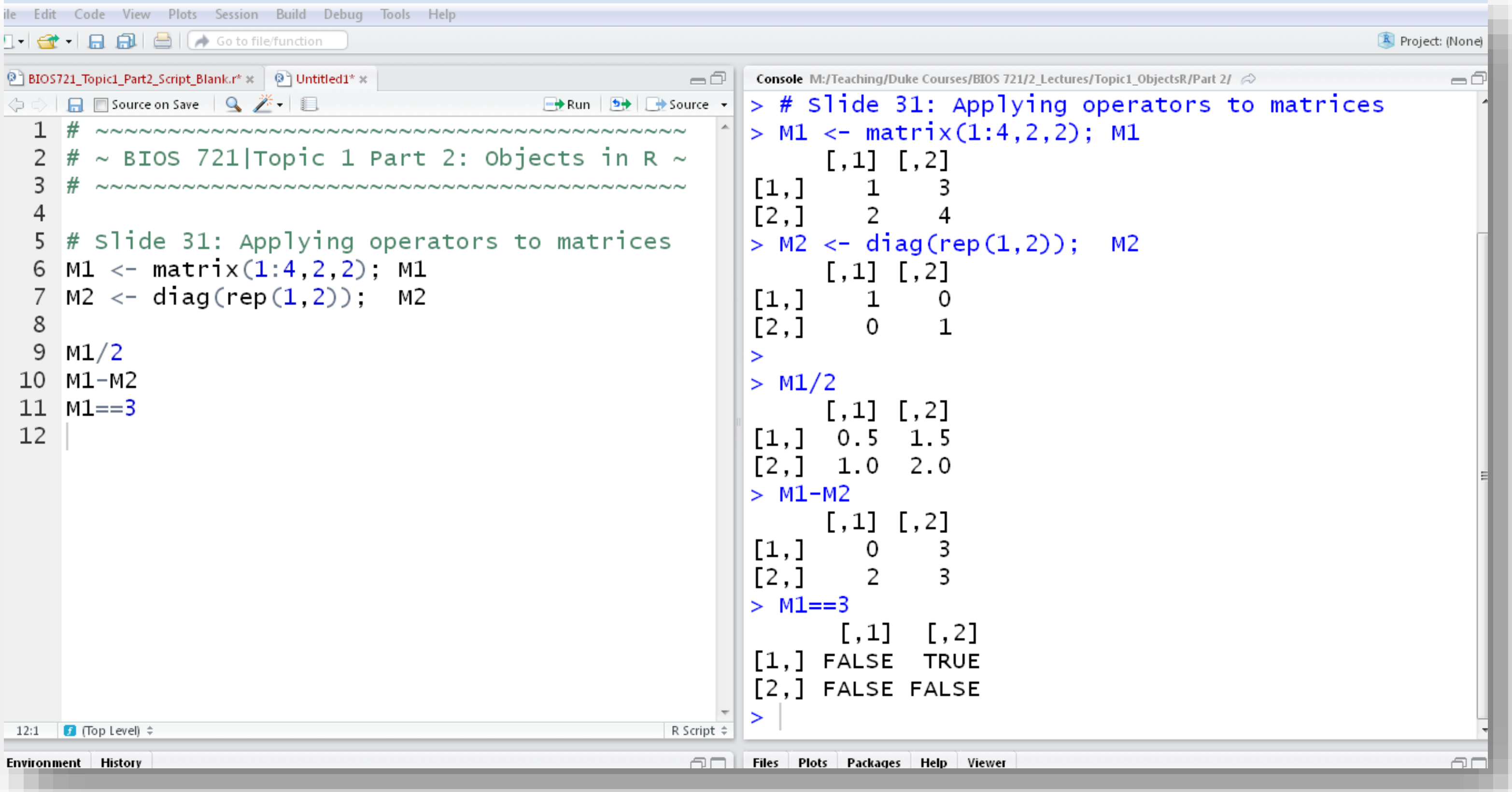


Console M:/Teaching/Duke Courses/BIOS 721/2_Lectures/Topic2_ObjectsR/

```
>
> # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
> # ~ BIOS 271|Topic 2 Part 2: Objects in R ~
> # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
>
> # Slide 24: Applying operators to matrices
> M1 <- matrix(1:4,2,2); M1
     [,1] [,2]
[1,]    1    3
[2,]    2    4
> M2 <- diag(rep(1,2));  M2
     [,1] [,2]
[1,]    1    0
[2,]    0    1
> |
```

R Operators

- R applies operators element-wise



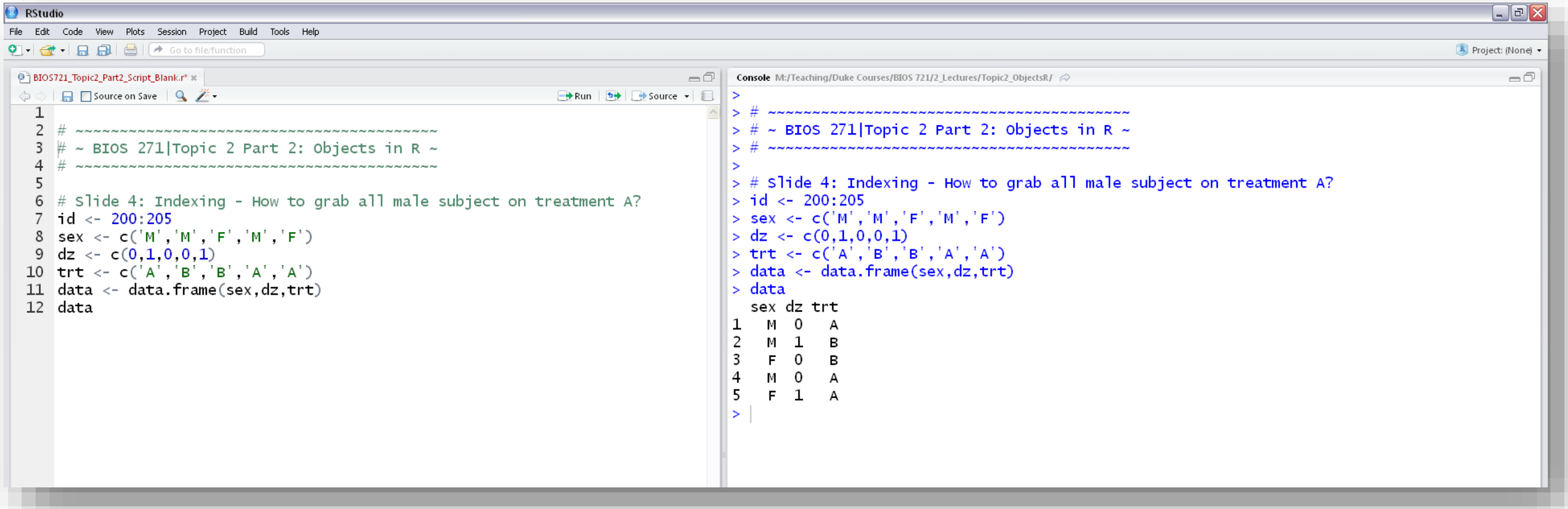
ile Edit Code View Plots Session Build Debug Tools Help
Go to file/function
Project: (None)
BIOS721_Topic1_Part2_Script_Blank.r*
Untitled1*
Source on Save
Run
Source
12:1
(Top Level)
R Script
Console M:/Teaching/Duke Courses/BIOS 721/2_Lectures/Topic1_ObjectsR/Part 2/
Environment
History
Files
Plots
Packages
Help
Viewer

```
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# ~ BIOS 721|Topic 1 Part 2: Objects in R ~
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# Slide 31: Applying operators to matrices
M1 <- matrix(1:4,2,2); M1
M2 <- diag(rep(1,2));  M2

M1/2
M1-M2
M1==3

```

```
> # Slide 31: Applying operators to matrices
> M1 <- matrix(1:4,2,2); M1
     [,1] [,2]
[1,]    1    3
[2,]    2    4
> M2 <- diag(rep(1,2));  M2
     [,1] [,2]
[1,]    1    0
[2,]    0    1
>
> M1/2
     [,1] [,2]
[1,]  0.5  1.5
[2,]  1.0  2.0
> M1-M2
     [,1] [,2]
[1,]    0    3
[2,]    2    3
> M1==3
      [,1]  [,2]
[1,] FALSE  TRUE
[2,] FALSE FALSE
>
```

Indexing – Motivating Example

- Recall this example?
  - Consider the data frame below. Suppose we wanted to create another data frame that subsets to all male subjects who were exposed to treatment A.
  - How can operators be used to automate the indexing needed to create a data frame called sub that contains this subset of the data?



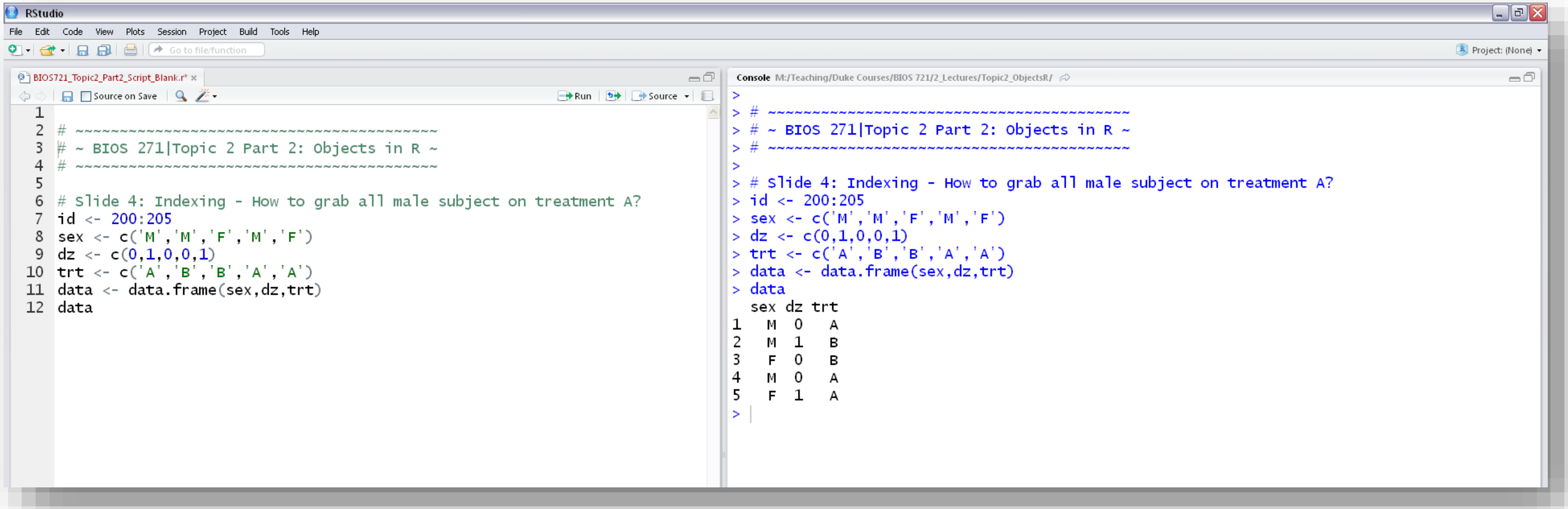
RStudio
File Edit Code View Plots Session Project Build Tools Help
Go to file/function
Project: (None)
BIOS721_Topic2_Part2_Script_Blank.r*
Source on Save
Run
Source
Console M:/Teaching/Duke Courses/BIOS 721/2_Lectures/Topic2_ObjectsR/

```

# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# ~ BIOS 271|Topic 2 Part 2: Objects in R ~
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# Slide 4: Indexing - How to grab all male subject on treatment A?
id <- 200:205
sex <- c('M','M','F','M','F')
dz <- c(0,1,0,0,1)
trt <- c('A','B','B','A','A')
data <- data.frame(sex,dz,trt)
data
```

```
>
> # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
> # ~ BIOS 271|Topic 2 Part 2: Objects in R ~
> # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
>
> # Slide 4: Indexing - How to grab all male subject on treatment A?
> id <- 200:205
> sex <- c('M','M','F','M','F')
> dz <- c(0,1,0,0,1)
> trt <- c('A','B','B','A','A')
> data <- data.frame(sex,dz,trt)
> data
  sex dz trt
1   M  0   A
2   M  1   B
3   F  0   B
4   M  0   A
5   F  1   A
>
```

Indexing – Motivating Example

- Recall this example?
  - For solution – come to class!

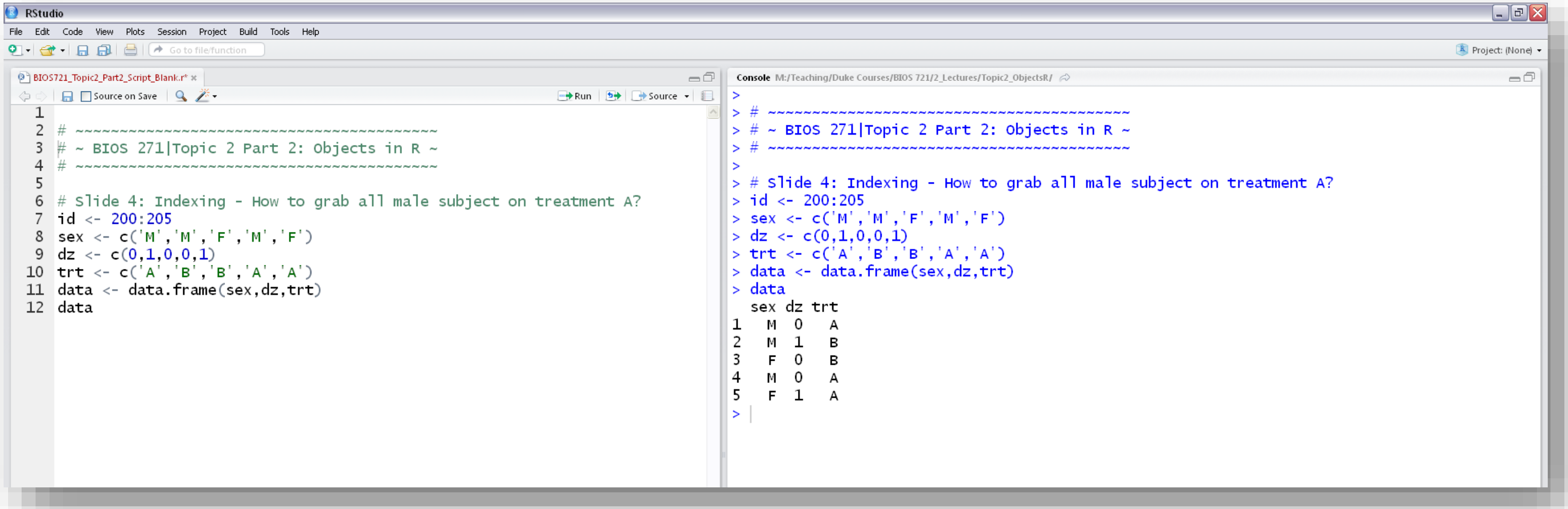

RStudio
File Edit Code View Plots Session Project Build Tools Help
Go to file/function
Project: (None)
BIOS721_Topic2_Part2_Script_Blank.r*
Source on Save
Run
Source
Console M:/Teaching/Duke Courses/BIOS 721/2_Lectures/Topic2_ObjectsR/

```

# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# ~ BIOS 271|Topic 2 Part 2: Objects in R ~
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# Slide 4: Indexing - How to grab all male subject on treatment A?
id <- 200:205
sex <- c('M','M','F','M','F')
dz <- c(0,1,0,0,1)
trt <- c('A','B','B','A','A')
data <- data.frame(sex,dz,trt)
data
```

```
>
> # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
> # ~ BIOS 271|Topic 2 Part 2: Objects in R ~
> # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
>
> # Slide 4: Indexing - How to grab all male subject on treatment A?
> id <- 200:205
> sex <- c('M','M','F','M','F')
> dz <- c(0,1,0,0,1)
> trt <- c('A','B','B','A','A')
> data <- data.frame(sex,dz,trt)
> data
  sex dz trt
1   M  0   A
2   M  1   B
3   F  0   B
4   M  0   A
5   F  1   A
>
```

Indexing – Motivating Example

- Recall this example?
  - The code on the previous slide uses R operators to determine whether each observation satisfies a particular condition.
  - But this alone doesn't completely automate the indexing needed to create the data frame sub – need an R function that … does what?


RStudio
File Edit Code View Plots Session Project Build Tools Help
Go to file/function
Project: (None)
BIOS721_Topic2_Part2_Script_Blank.r*
Source on Save
Run
Source
Console M:/Teaching/Duke Courses/BIOS 721/2_Lectures/Topic2_ObjectsR/

```

# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# ~ BIOS 271|Topic 2 Part 2: Objects in R ~
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# Slide 4: Indexing - How to grab all male subject on treatment A?
id <- 200:205
sex <- c('M','M','F','M','F')
dz <- c(0,1,0,0,1)
trt <- c('A','B','B','A','A')
data <- data.frame(sex,dz,trt)
data
```

```
>
> # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
> # ~ BIOS 271|Topic 2 Part 2: Objects in R ~
> # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
>
> # Slide 4: Indexing - How to grab all male subject on treatment A?
> id <- 200:205
> sex <- c('M','M','F','M','F')
> dz <- c(0,1,0,0,1)
> trt <- c('A','B','B','A','A')
> data <- data.frame(sex,dz,trt)
> data
  sex dz trt
1   M  0   A
2   M  1   B
3   F  0   B
4   M  0   A
5   F  1   A
> 
```

R Functions

- Where to begin?

- There a TON of R Functions.
  - Many are included in the base R installation
  - Many, many more that can be download in Packages developed by other R users
  - Only base R functions are “vetted”

- Today we will focus on simple R functions …
  - Ones that every statistician should have in their toolbox

R Functions

- The basic syntax associated with an R Function:
  - `function_name(inputs, …)`
  - The () tells R that the keyword being referenced is a function and not a data object (remember the [] tells R that an object is being indexed)

- To learn about functions, bring up the R help page by typing `?function_name` into the R console.
  - Example| `?mean`

- All functions belong to a package (look for `function_name{package_name}` in the upper left hand side of the R help page).
  - For non-base installation functions, you can download the package by using the `install.packages()` function and entering the name of the package in quotations as input.
  - We will discuss this more later in the course.

Useful R Functions

- Functions that will come in handy …
  - `sqrt(x)` → returns square root of x
  - `log(x)` → returns log base_e of x
  - `exp(x)` → returns e raised to the power x
  - `mean(x)` → returns mean/average of x
  - `median(x)` → returns median of x
  - `sd(x)` → returns standard deviation of x
  - `min(x)` → returns minimum of x
  - `max(x)` → returns maximum of x
  - `length(x)` → returns the number of elements in x
  - `dim(x)` → returns the no. of rows and cols of a matrix x
  - `solve(x)` → returns inverse of a matrix x
  - `t(x)` → returns transpose of a matrix x

Useful R Functions

- Functions that will come in handy …
  - `sqrt(x)` → returns
  - `log(x)` → returns
  - `exp(x)` → returns
  - `mean(x)` → returns
  - `median(x)` → returns median of x
  - `sd(x)` → returns standard deviation of x
  - `min(x)` → returns minimum of x
  - `max(x)` → returns maximum of x
  - `length(x)` → returns the number of elements in x
  - `dim(x)` → returns the no. of rows and cols of a matrix x
  - `solve(x)` → returns inverse of a matrix x
  - `t(x)` → returns transpose of a matrix x

When encountering new R functions, you should always ask …

- What type of object does it apply to?
- What type of object does it return?
- And how are these two objects related?

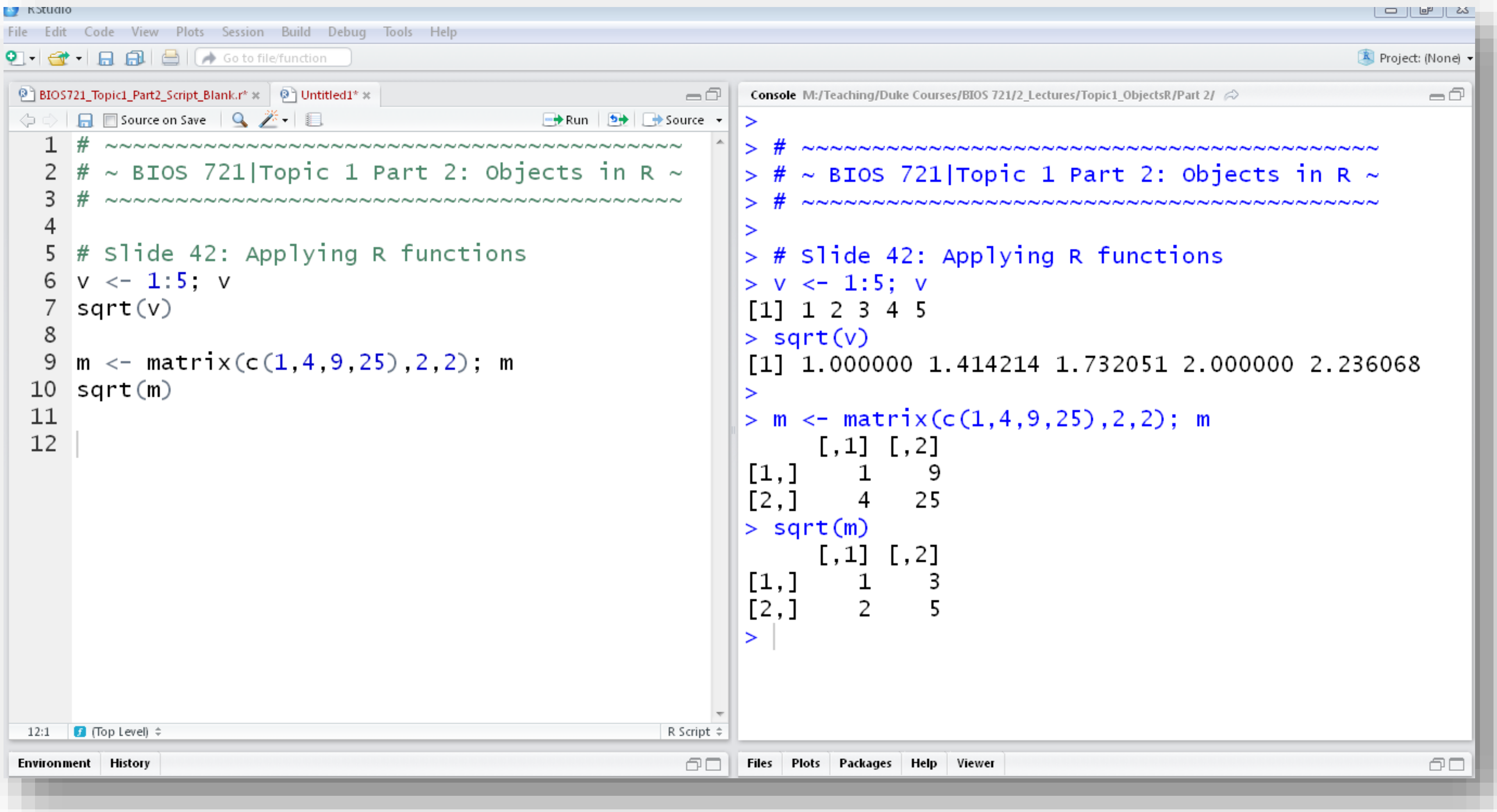
Useful R Functions

- Let’s consider the function sqrt()
  - Example| `v <- 1:5; sqrt(v)`

    ```
    m <- matrix(c(1,4,9,25),2,2); sqrt(m)
    ```

- Now let’s answer the 3 ‘function’ questions:
  - What type of object does it apply to?
  - What type of object will it return?
  - How are these two objects related to each other?

Useful R Functions


RStudio
File Edit Code View Plots Session Build Debug Tools Help
Go to file/function
Project: (None)
BIOS721_Topic1_Part2_Script_Blank.r*
Untitled1*
Source on Save
Run
Source
12:1
(Top Level)
R Script
Environment History
Console M:/Teaching/Duke Courses/BIOS 721/2_Lectures/Topic1_ObjectsR/Part 2/
Files Plots Packages Help Viewer

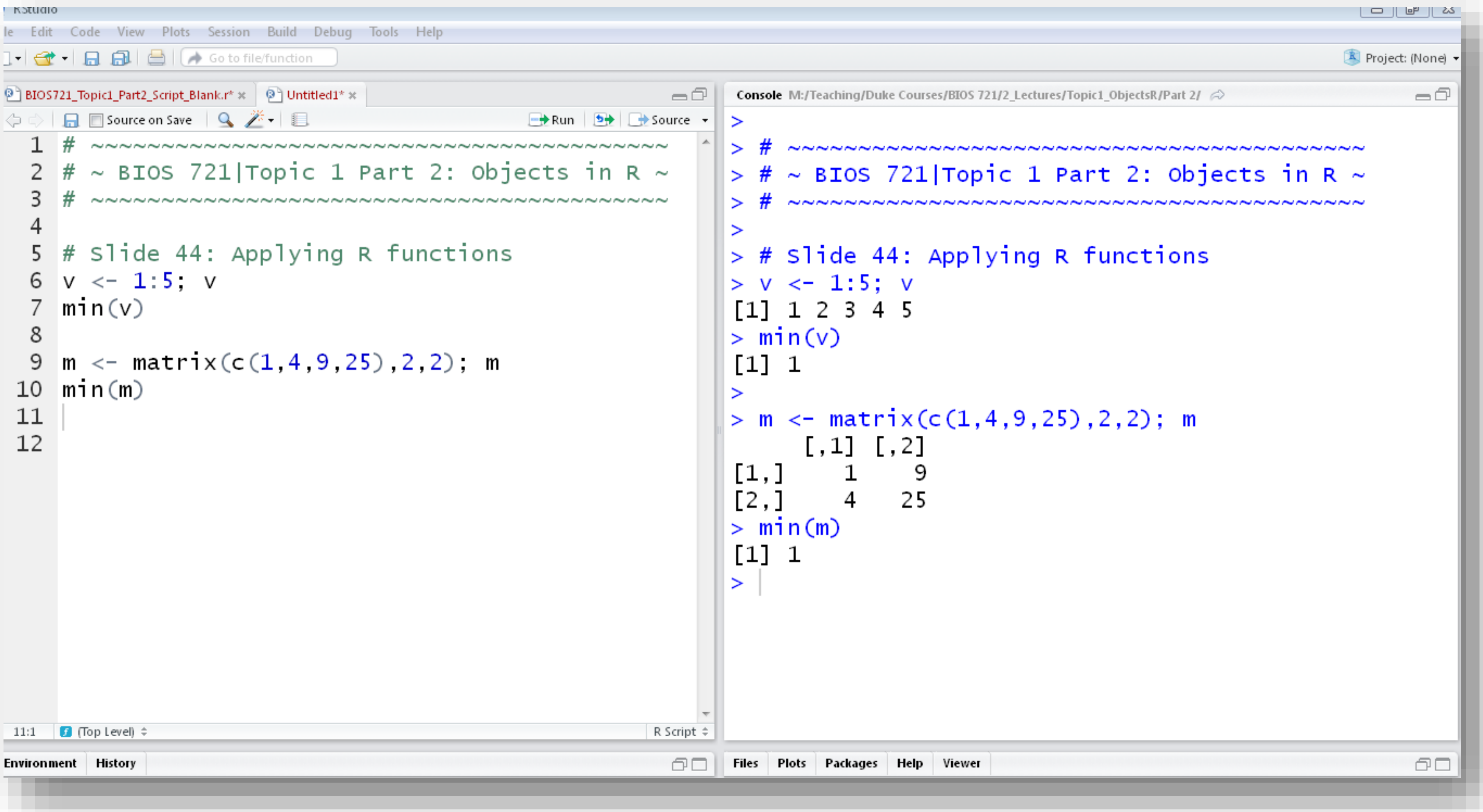
```
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# ~ BIOS 721|Topic 1 Part 2: Objects in R ~
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# Slide 42: Applying R functions
v <- 1:5; v
sqrt(v)

m <- matrix(c(1,4,9,25),2,2); m
sqrt(m)

```

```
>
> # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
> # ~ BIOS 721|Topic 1 Part 2: Objects in R ~
> # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
>
> # Slide 42: Applying R functions
> v <- 1:5; v
[1] 1 2 3 4 5
> sqrt(v)
[1] 1.000000 1.414214 1.732051 2.000000 2.236068
>
> m <- matrix(c(1,4,9,25),2,2); m
     [,1] [,2]
[1,]    1    9
[2,]    4   25
> sqrt(m)
     [,1] [,2]
[1,]    1    3
[2,]    2    5
>
```

Useful R Functions

- Let’s consider the function min()
  - Example|
    ```
    v <- 1:5; min(v)
    m <- matrix(c(1,4,9,25),2,2); min(m)
    ```
- Now let’s answer the 3 ‘function’ questions:
  - What type of object does it apply to?
  - What type of object will it return?
  - How are these two objects related to each other?

Useful R Functions


RStudio
le Edit Code View Plots Session Build Debug Tools Help
Go to file/function
Project: (None)
BIOS721_Topic1_Part2_Script_Blank.r*
Untitled1*
Source on Save
Run
Source
11:1
(Top Level)
R Script
Console M:/Teaching/Duke Courses/BIOS 721/2_Lectures/Topic1_ObjectsR/Part 2/
Environment
History
Files
Plots
Packages
Help
Viewer

```
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# ~ BIOS 721|Topic 1 Part 2: Objects in R ~
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# Slide 44: Applying R functions
v <- 1:5; v
min(v)

m <- matrix(c(1,4,9,25),2,2); m
min(m)

```

```
>
> # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
> # ~ BIOS 721|Topic 1 Part 2: Objects in R ~
> # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
>
> # Slide 44: Applying R functions
> v <- 1:5; v
[1] 1 2 3 4 5
> min(v)
[1] 1
>
> m <- matrix(c(1,4,9,25),2,2); m
     [,1] [,2]
[1,]    1    9
[2,]    4   25
> min(m)
[1] 1
>
```

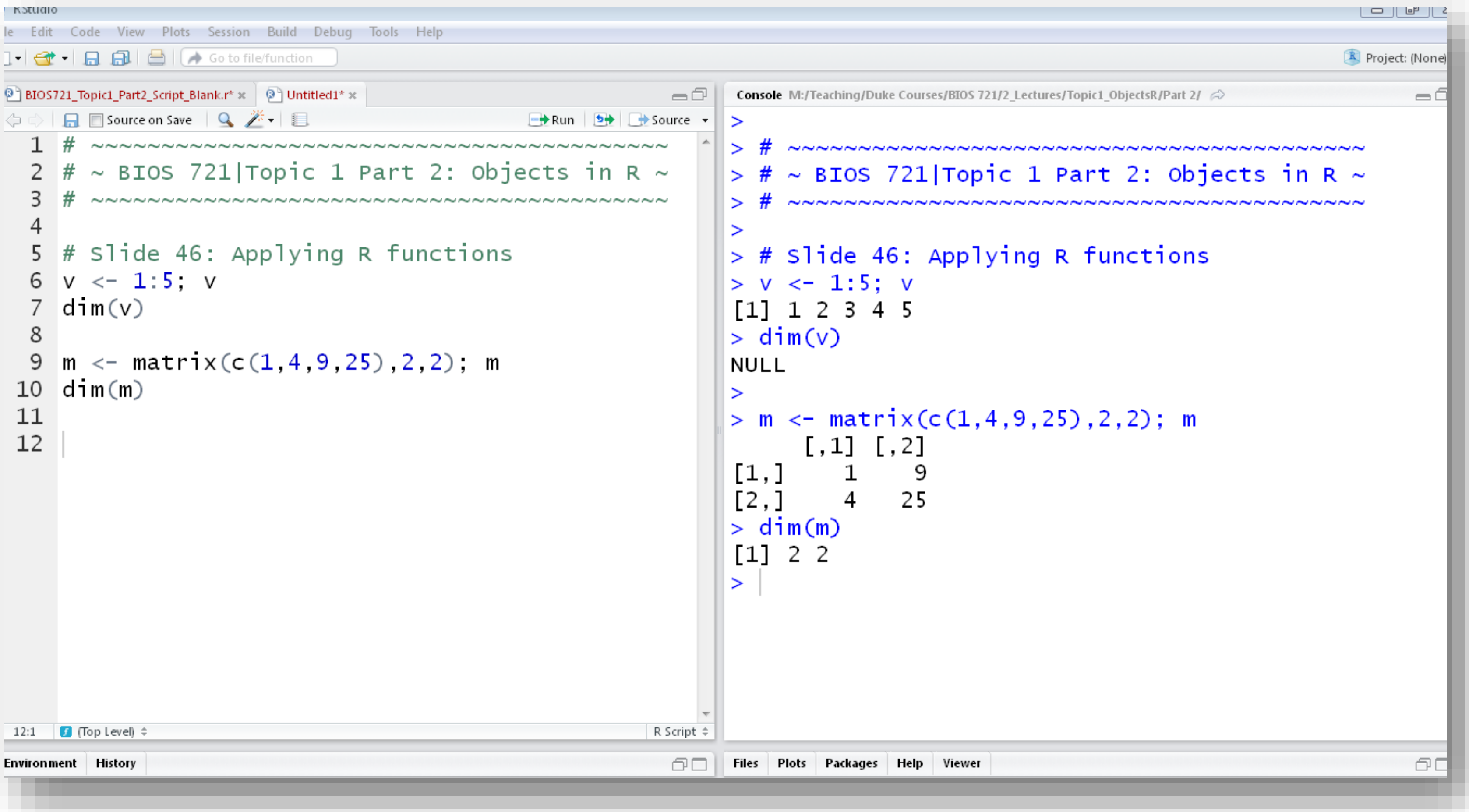
Useful R Functions

- Let's consider the function dim()
  - Example|

```
v <- 1:5; dim(v)
m <- matrix(c(1,4,9,25),2,2); dim(m)
```

- Now let's answer the 3 'function' questions:
  - What type of object does it apply to?
  - What type of object will it return?
  - How are these two objects related to each other?

Useful R Functions


RStudio
le Edit Code View Plots Session Build Debug Tools Help
Go to file/function
Project: (None)
BIOS721_Topic1_Part2_Script_Blank.r*
Untitled1*
Source on Save
Run
Source
Console M:/Teaching/Duke Courses/BIOS 721/2_Lectures/Topic1_ObjectsR/Part 2/
12:1
(Top Level)
R Script
Environment
History
Files
Plots
Packages
Help
Viewer

```r
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# ~ BIOS 721|Topic 1 Part 2: Objects in R ~
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# Slide 46: Applying R functions
v <- 1:5; v
dim(v)

m <- matrix(c(1,4,9,25),2,2); m
dim(m)

```

```
>
> # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
> # ~ BIOS 721|Topic 1 Part 2: Objects in R ~
> # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
>
> # Slide 46: Applying R functions
> v <- 1:5; v
[1] 1 2 3 4 5
> dim(v)
NULL
>
> m <- matrix(c(1,4,9,25),2,2); m
     [,1] [,2]
[1,]    1    9
[2,]    4   25
> dim(m)
[1] 2 2
>
```

Useful R Functions

- Functions that will come in handy …
  - `sqrt(x)` → returns
  - `log(x)` → returns
  - `exp(x)` → returns
  - `mean(x)` → returns
  - `median(x)` → retu
  - `sd(x)` → returns sta
  - `min(x)` → returns minimum of x
  - `max(x)` → returns maximum of x
  - `length(x)` → returns the number of elements in x
  - `dim(x)` → returns the no. of rows and cols of a matrix x
  - `solve(x)` → returns inverse of a matrix x
  - `t(x)` → returns transpose of a matrix x

When encountering new R functions, you should always ask …

- What type of object does it apply to?
- What type of object does it return?
- And how are these two objects related?

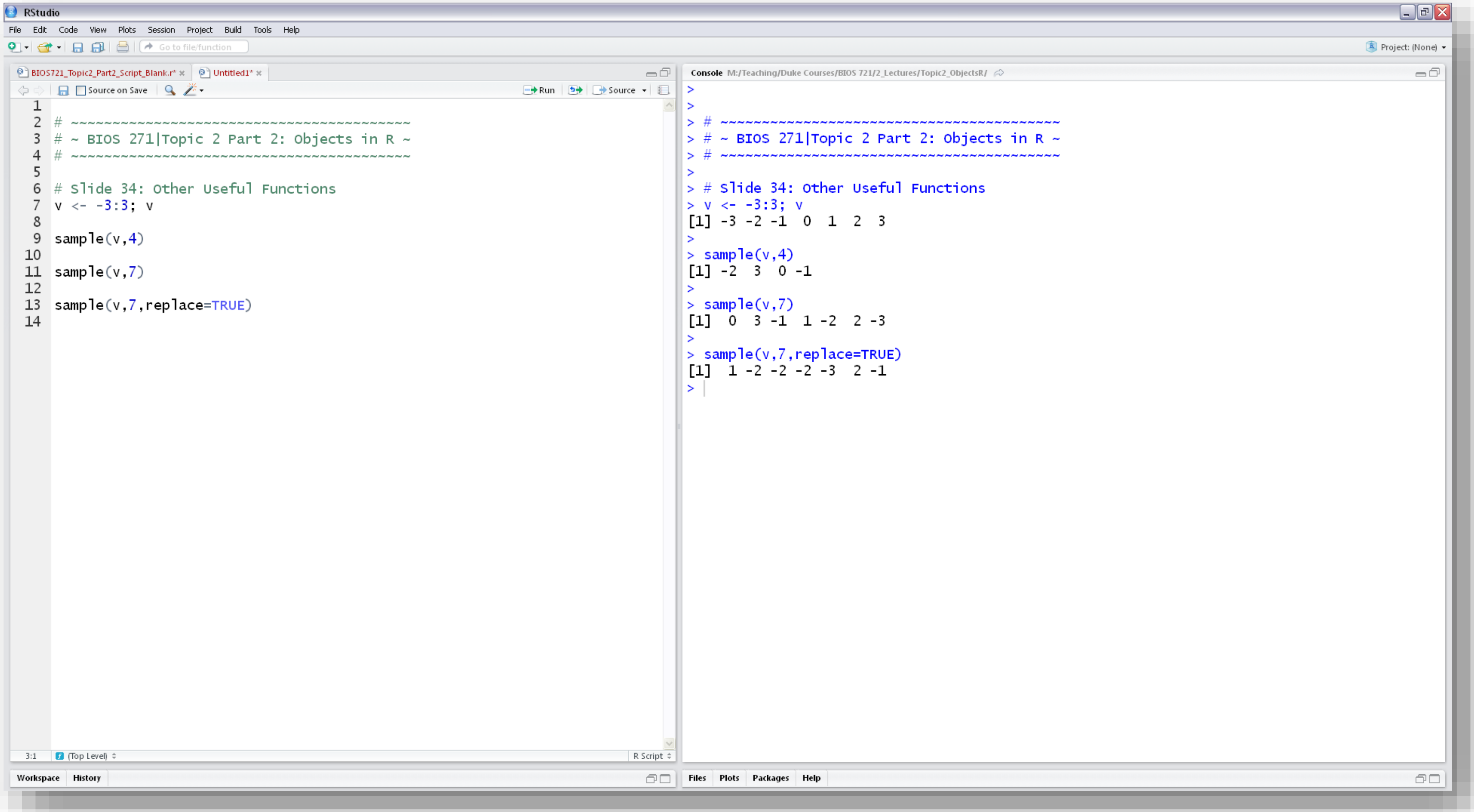
This is just a starter list … simple functions that all statisticians should have in their toolbox!

Other Useful R Functions

- To randomly shuffle/select elements in a vector use
  - `sample(x,size,replace=FALSE…)`
    - Returns `size` randomly selected elements from the vector `x`.
    - Can selected with (`replace=TRUE`) or without (`replace=FALSE`) replacement.

-
```
v <- -3:3
sample(v,4)
sample(v,7)
sample(v,7,replace=TRUE)
```

Other Useful R Functions


RStudio
File Edit Code View Plots Session Project Build Tools Help
Go to file/function
Project: (None)
BIOS721_Topic2_Part2_Script_Blank.r*
Untitled1*
Source on Save
Run
Source
Console M:/Teaching/Duke Courses/BIOS 721/2_Lectures/Topic2_ObjectsR/
3:1
(Top Level)
R Script
Workspace
History
Files
Plots
Packages
Help

```

# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# ~ BIOS 271|Topic 2 Part 2: Objects in R ~
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# Slide 34: Other Useful Functions
v <- -3:3; v

sample(v,4)

sample(v,7)

sample(v,7,replace=TRUE)

```

```
>
>
> # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
> # ~ BIOS 271|Topic 2 Part 2: Objects in R ~
> # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
>
> # Slide 34: Other Useful Functions
> v <- -3:3; v
[1] -3 -2 -1  0  1  2  3
>
> sample(v,4)
[1] -2  3  0 -1
>
> sample(v,7)
[1]  0  3 -1  1 -2  2 -3
>
> sample(v,7,replace=TRUE)
[1]  1 -2 -2 -2 -3  2 -1
> 
```

Other Useful R Functions

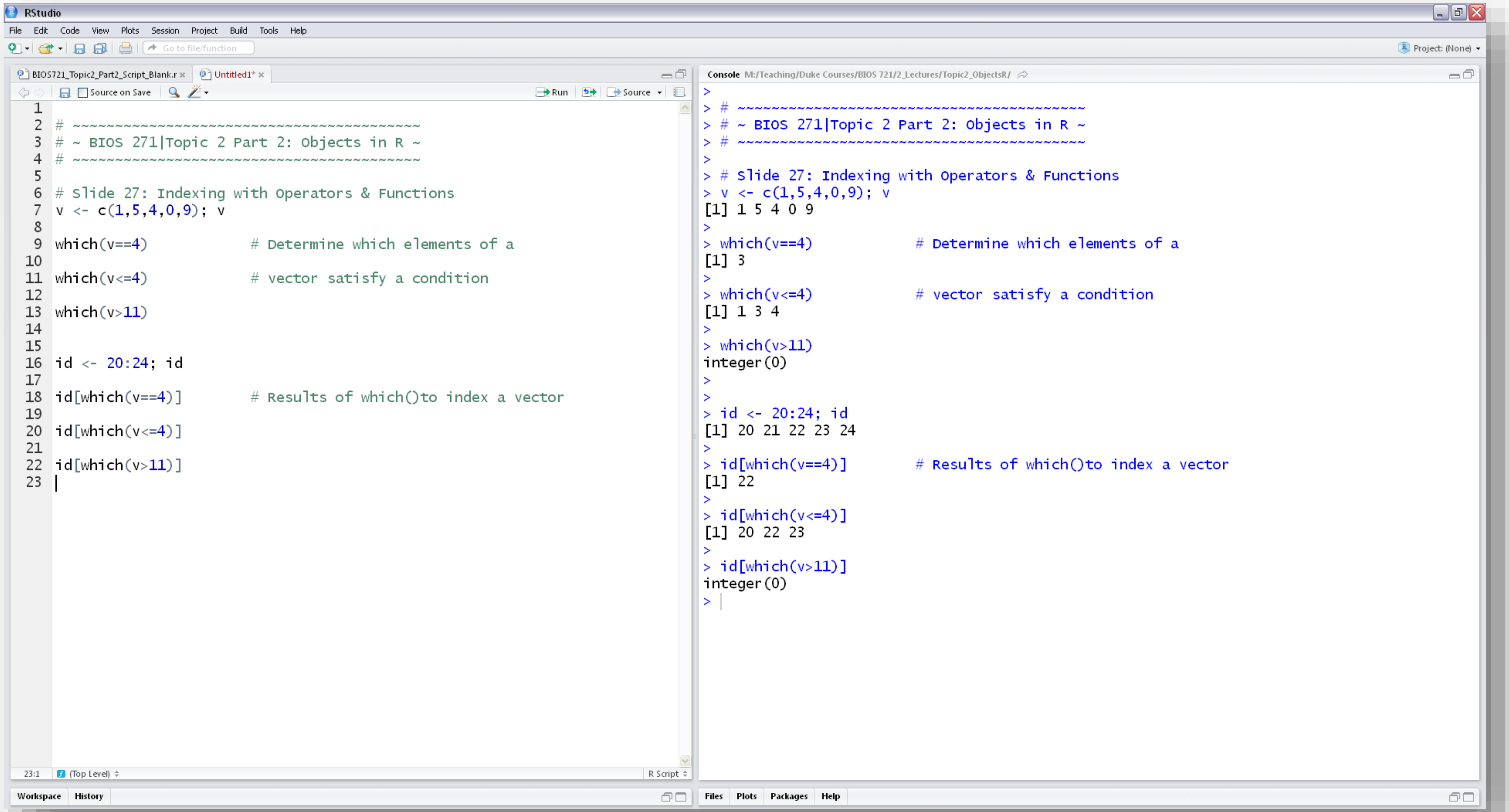
- Can use `which()` to determine which elements of a vector satisfy a condition

  ```
  v <- c(1,5,4,0,9)
   which(v==4)
   which(v<=4)
   which(v>11)
  ```

- Can then use the results of `which()` to index a vector

  ```
  id <- 20:24
   id[which(v==4)]
   id[which(v<=4)]
   id[which(v>11)]
  ```

Other Useful R Functions

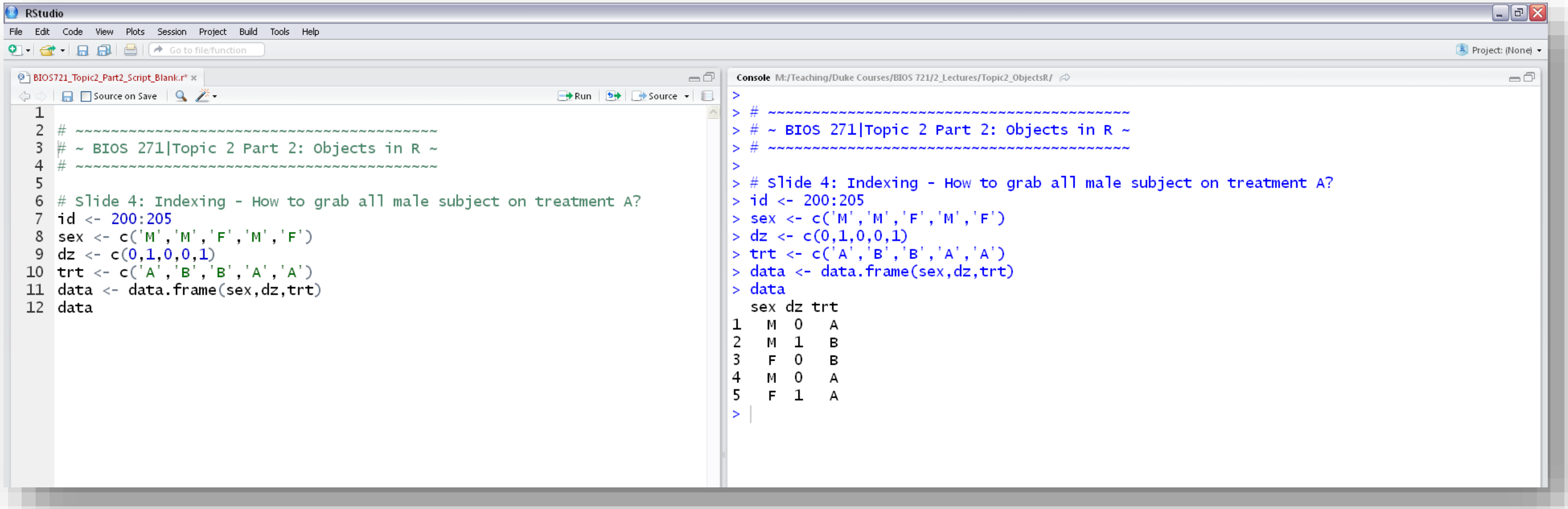


```r

# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# ~ BIOS 271|Topic 2 Part 2: Objects in R ~
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# Slide 27: Indexing with Operators & Functions
v <- c(1,5,4,0,9); v

which(v==4)           # Determine which elements of a

which(v<=4)           # vector satisfy a condition

which(v>11)


id <- 20:24; id

id[which(v==4)]       # Results of which()to index a vector

id[which(v<=4)]

id[which(v>11)]

```

```
>
> # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
> # ~ BIOS 271|Topic 2 Part 2: Objects in R ~
> # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
>
> # Slide 27: Indexing with Operators & Functions
> v <- c(1,5,4,0,9); v
[1] 1 5 4 0 9
>
> which(v==4)           # Determine which elements of a
[1] 3
>
> which(v<=4)           # vector satisfy a condition
[1] 1 3 4
>
> which(v>11)
integer(0)
>
>
> id <- 20:24; id
[1] 20 21 22 23 24
>
> id[which(v==4)]       # Results of which()to index a vector
[1] 22
>
> id[which(v<=4)]
[1] 20 22 23
>
> id[which(v>11)]
integer(0)
>
```

Putting it Together – Automated Indexing

- Back to this example …
  - Consider the data frame below. Suppose we wanted to create another data frame that subsets to all male subjects who were exposed to treatment A.
  - How can we achieve this using R in an automated fashion (i.e. for any size data frame)?

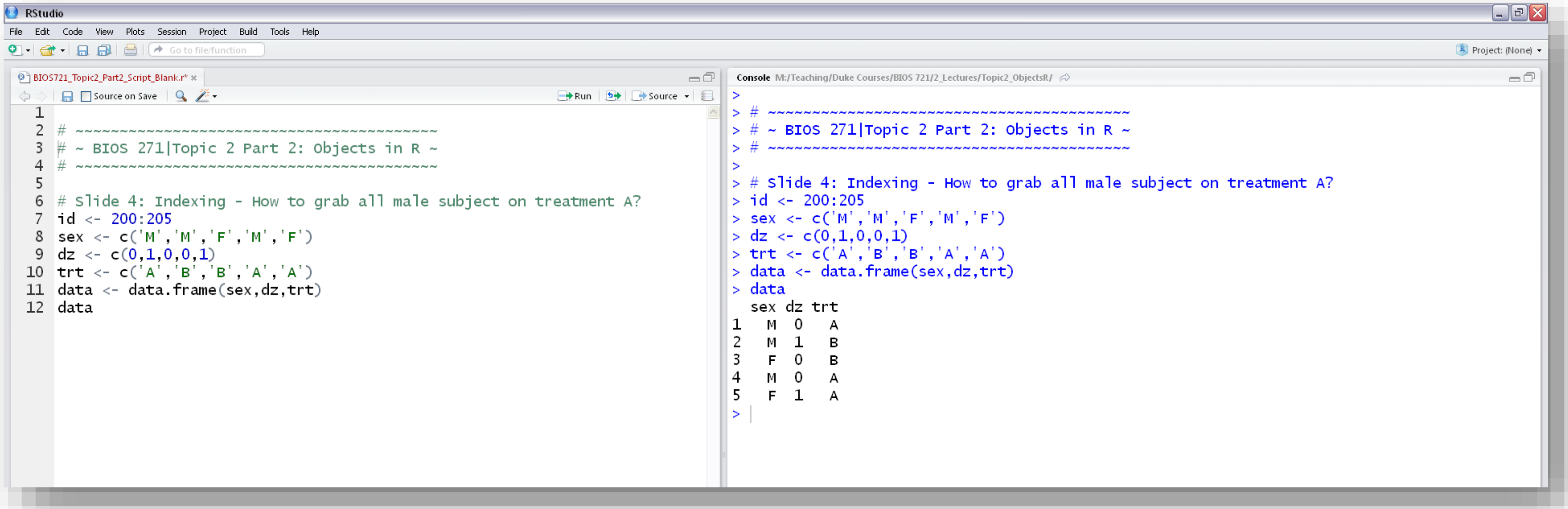


```r

# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# ~ BIOS 271|Topic 2 Part 2: Objects in R ~
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# Slide 4: Indexing - How to grab all male subject on treatment A?
id <- 200:205
sex <- c('M','M','F','M','F')
dz <- c(0,1,0,0,1)
trt <- c('A','B','B','A','A')
data <- data.frame(sex,dz,trt)
data
```

```
>
> # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
> # ~ BIOS 271|Topic 2 Part 2: Objects in R ~
> # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
>
> # Slide 4: Indexing - How to grab all male subject on treatment A?
> id <- 200:205
> sex <- c('M','M','F','M','F')
> dz <- c(0,1,0,0,1)
> trt <- c('A','B','B','A','A')
> data <- data.frame(sex,dz,trt)
> data
  sex dz trt
1   M  0   A
2   M  1   B
3   F  0   B
4   M  0   A
5   F  1   A
>
```

Indexing – Motivating Example

- Recall this example?
  - For solution – come to class!



```

# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# ~ BIOS 271|Topic 2 Part 2: Objects in R ~
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# Slide 4: Indexing - How to grab all male subject on treatment A?
id <- 200:205
sex <- c('M','M','F','M','F')
dz <- c(0,1,0,0,1)
trt <- c('A','B','B','A','A')
data <- data.frame(sex,dz,trt)
data
```

```
>
> # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
> # ~ BIOS 271|Topic 2 Part 2: Objects in R ~
> # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
>
> # Slide 4: Indexing - How to grab all male subject on treatment A?
> id <- 200:205
> sex <- c('M','M','F','M','F')
> dz <- c(0,1,0,0,1)
> trt <- c('A','B','B','A','A')
> data <- data.frame(sex,dz,trt)
> data
  sex dz trt
1   M  0   A
2   M  1   B
3   F  0   B
4   M  0   A
5   F  1   A
> 
```

Next Time

- Next Time begin Topic 2
  - Data Management in R