1 Getting started

(AST230) R for Data Science

Md Rasel Biswas

Getting Started with R

- You can write any code you want in the ***console***. For example:

```
print("Hello World!")
```

```
[1] "Hello World!"
```

Getting Started with R

- R can be used as a calculator. You can interact with R by typing something into the *console* at the command prompt and pressing enter

```
2+4
```

```
[1] 6
```

```
(59 + 73 + 2) / 3
```

```
[1] 44.66667
```

Note:

- You don't need to type an equals sign, just hit enter.
- Arithmetic operators are the same as they are in most other computer applications.

Arithmetic operators

Description	Operator
Addition	`+`
Subtraction	`-`
Multiplication	`*`
Division	`/`
Exponentiation	`^ or **`
integer division `10%/%3` is 3	`x %/% y`
modulus (x mod y) `10%%3` is 1	`x %% y`

Exercise 1.1

Use R to calculate the following:

- 5^2
- Add 8 to 22 and then multiply the answer by 3
- Divide 8 by 2.5 and then divide the answer by 3

Notice

The above calculations do not produce any kind of output that is remembered by R

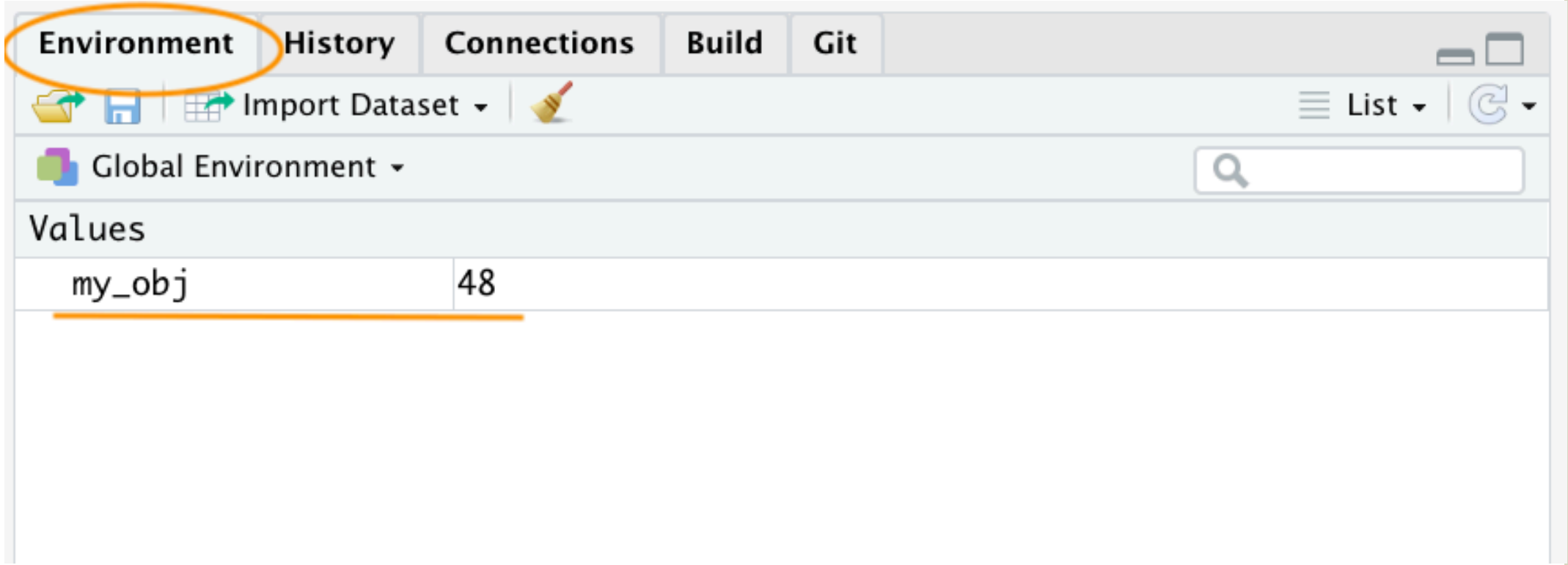
Creating objects

- To store our calculations in R we need to give it a name and tell R to store that as an object.
- *Assignment operators* `<-` or `=` can be used to assign a value to an R object, and it is preferable to use `<-` as `=` has other uses.

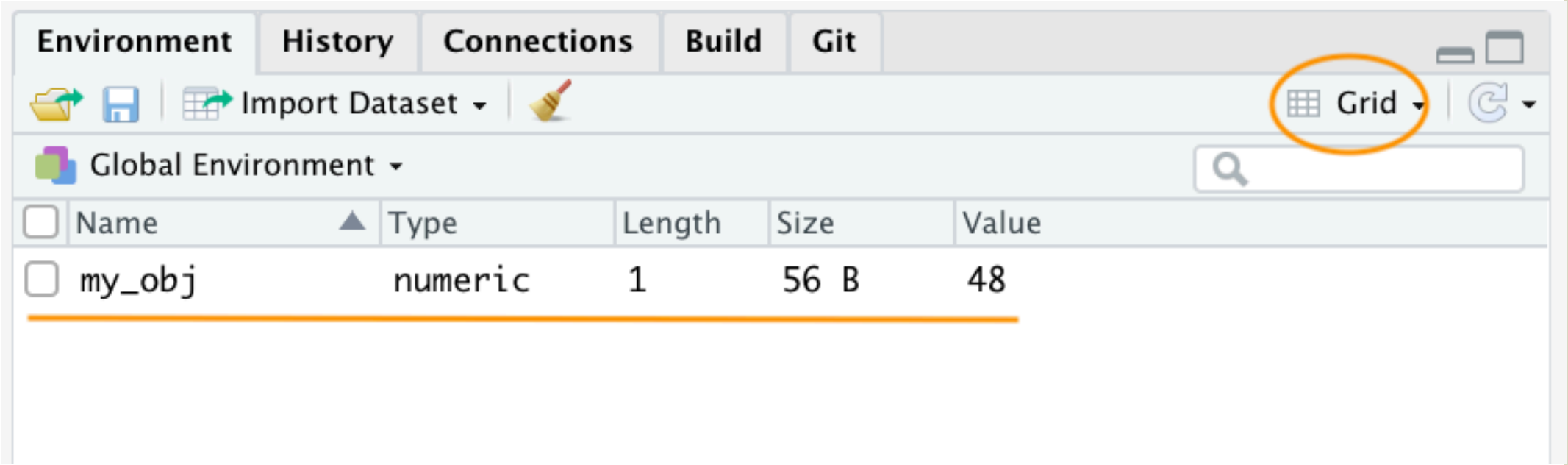
```
my_obj <- 48
```

- Now that we've created this object, R knows all about it and will keep track of it during this current R session
- All of the objects you create will be stored in the current workspace and you can view all the objects in your workspace in RStudio by clicking on the `Environment` tab in the top right-top pane

Creating objects



Creating objects



Creating objects

- If you click on the down arrow on the 'List' icon in the same pane and change to 'Grid' view, RStudio will show you a summary of the objects including
  - the type (numeric - it's a number)
  - the length (only one value in this object)
  - its 'physical' size, and
  - its value (48 in this case)

Creating objects

- There are many different types of values that you can assign to an object. For example

```
my_obj2 <- "R is cool"
```

- Here we have created an object called `my_obj2` and assigned it a value of `R is cool` which is a character string.

Note

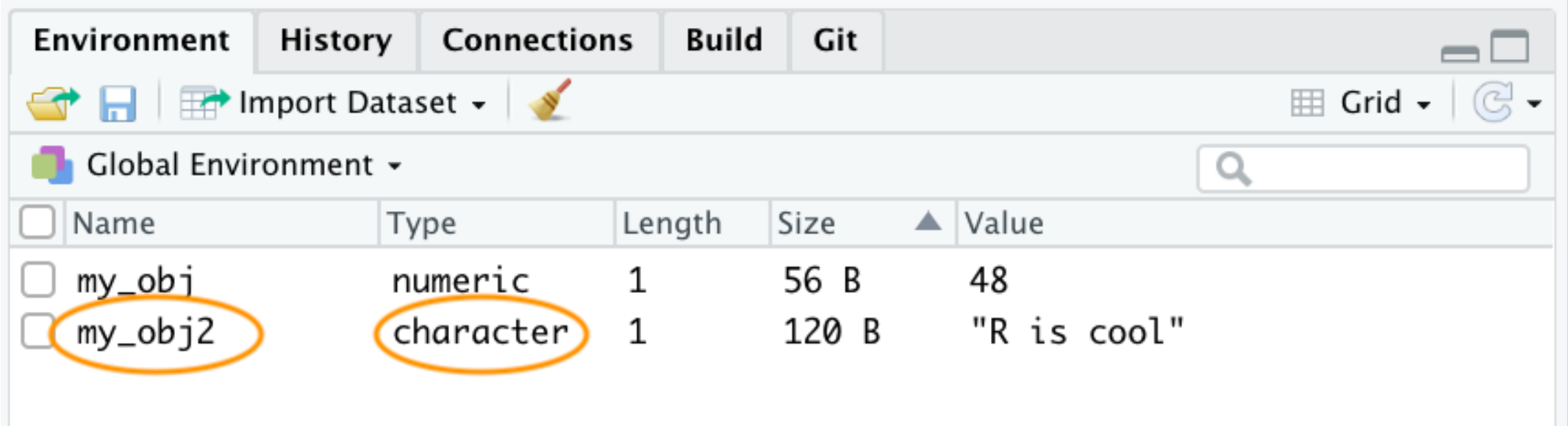
- We have enclosed the string in quotes.
- If you forget to use the quotes you will receive an error message.

```
my_obj2 <- R is cool
```

```
Error: <text>:1:14: unexpected symbol
1: my_obj2 <- R is
                 ^
```

Creating objects

- Our workspace now contains both objects we've created so far with `my_obj2` listed as type character.



Creating objects

- To change the value of an existing object we simply reassign a new value to it.
- For example, to change the value of `my_obj2` from `"R is cool"` to the number `1024`

```
my_obj2 <- 1024
```

Creating objects

- Once we have created a few objects, we can do stuff with our objects.
- For example, the following code creates a new object `my_obj3` and assigns it the value of `my_obj` added to `my_obj2` which is $1072 (48 + 1024 = 1072)$

```
my_obj3 <- my_obj + my_obj2
my_obj3
```

```
[1] 1072
```

Creating objects

```
char_obj <- "hello"
char_obj2 <- "world!"
char_obj3 <- char_obj + char_obj2
```

```
Error in char_obj + char_obj2: non-numeric argument to binary operator
```

- Reading the error message is important in R
- This error message is essentially telling you that either one or both of the objects `char_obj` and `char_obj2` is not a number and therefore cannot be added together

Creating objects

- Another error message that you'll get quite a lot when you first start using R is `Error: object '**' not found`. As an example, take a look at the code below.

```
my_obj <- 48
my_obj4 <- my_obj + no_obj
```

```
Error in eval(expr, envir, enclos): object 'no_obj' not found
```

- R returns an error message because we haven't created (defined) the object `no_obj` yet.
- If you check your environment, you'll see that object `my_obj4` has not been created

Naming objects

- In R, object names are case sensitive, and a valid object name (syntactic name) consists of a combination of
  - (`a-z`, `A-Z`), (`0-9`), (`.`), (`-`), and (`_`)
  - An object name cannot start with a number or a hyphen or an underscore and can start with a dot, but it must be followed by a letter
- A good programming practice is to use meaningful object names in the codes, and self-explanatory object names increase the readability of the codes
- Existing R functions, names, or words (e.g., `mean`, `log`, `exp`, `TRUE`, `c`, etc.) should not be used as object names

Exercise 1.2

1. Create an object called `x1` which is the number 73
2. Create another object called `x2` which is the answer to the sum $101 + 36$
3. Multiply `x1` and `x2` together and store the object as another object called `x3`
4. Subtract 1 from `x3` and calculate the 4th root.
5. The answer should be 10

Use of brackets

- **Parentheses** `(` & `)` are used to define arithmetic expressions, and they must be matched; unmatched parentheses will result in errors

```
((3 + 12)/3 + 8)
```

```
[1] 13
```

- Using **curly brackets** `{` & `}` do not result in any error but should not be used as they have some specific uses in R, e.g. defining a function

```
{10 + 2}+ 5
```

```
[1] 17
```

- The **square brackets** `[` & `]` **cannot** be used in arithmetic expressions

```
[2 + 7]/3
```

```
Error: <text>:1:1: unexpected '['
1: [
    ^
```

R functions

- Functions are ready-made pieces of codes.
- Some of the mathematical functions in R:

Description	R symbol	Example
square root	sqrt	`sqrt(225)`
natural logarithm	log	`log(50)`
exponential	exp	`exp(3)`
absolute	abs	`abs(-10)`
factorial	factorial	`factorial(6)`
sine function	sin	`sin(25)`
inverse cosine	acos	`acos(-8.67)`

R functions

- R has a large collection of built-in functions that are called like this:

```
function_name(arg1 = val1, arg2 = val2, ...)
```

- Let's try using `sqrt()`
  - `sqrt(25)` or `sqrt(x = 25)` will return a value `5`

Note

- inputs (called *arguments* in R) should be within the parentheses `( )` even if there's no input.
- Multiple inputs, if needed, are separated by commas, e.g. `fun_name(`*input1*, *input2*, *input3*`)`

Some useful built-in R functions

```r
round(3.567, digits=2)
```

```
[1] 3.57
```

```r
floor(3.567)
```

```
[1] 3
```

```r
ceiling(3.567)
```

```
[1] 4
```

```r
pi   # Not a function, but useful
```

```
[1] 3.141593
```

Exercise 1.3

1. Why does this code not work?

```
my_variable <- 10
my_varıable
```

```
Error in eval(expr, envir, enclos): object 'my_varıable' not found
```

Exercise 1.4

1. Create an object called `myObject` and assign it a value between 1 and 100
2. Add 13 to `myObject`, making sure the object itself stores the updated value
3. Is `myObject` divisible by 2? by 3? by 13? by 21? Use the R code to get the answer.
4. How many times can 5 fit in `myObject`?