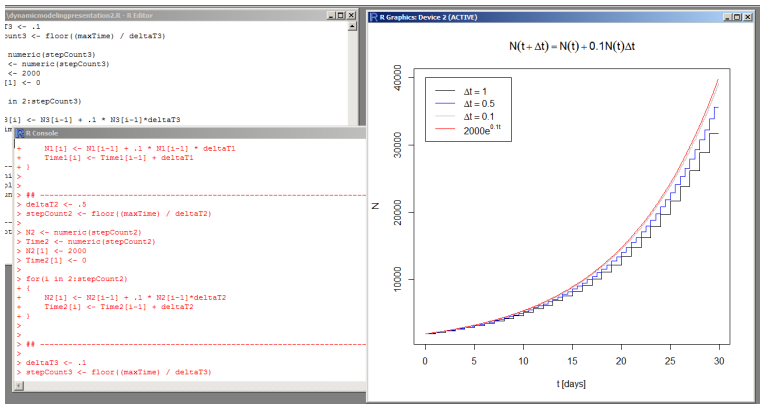
R - Repetition of some features

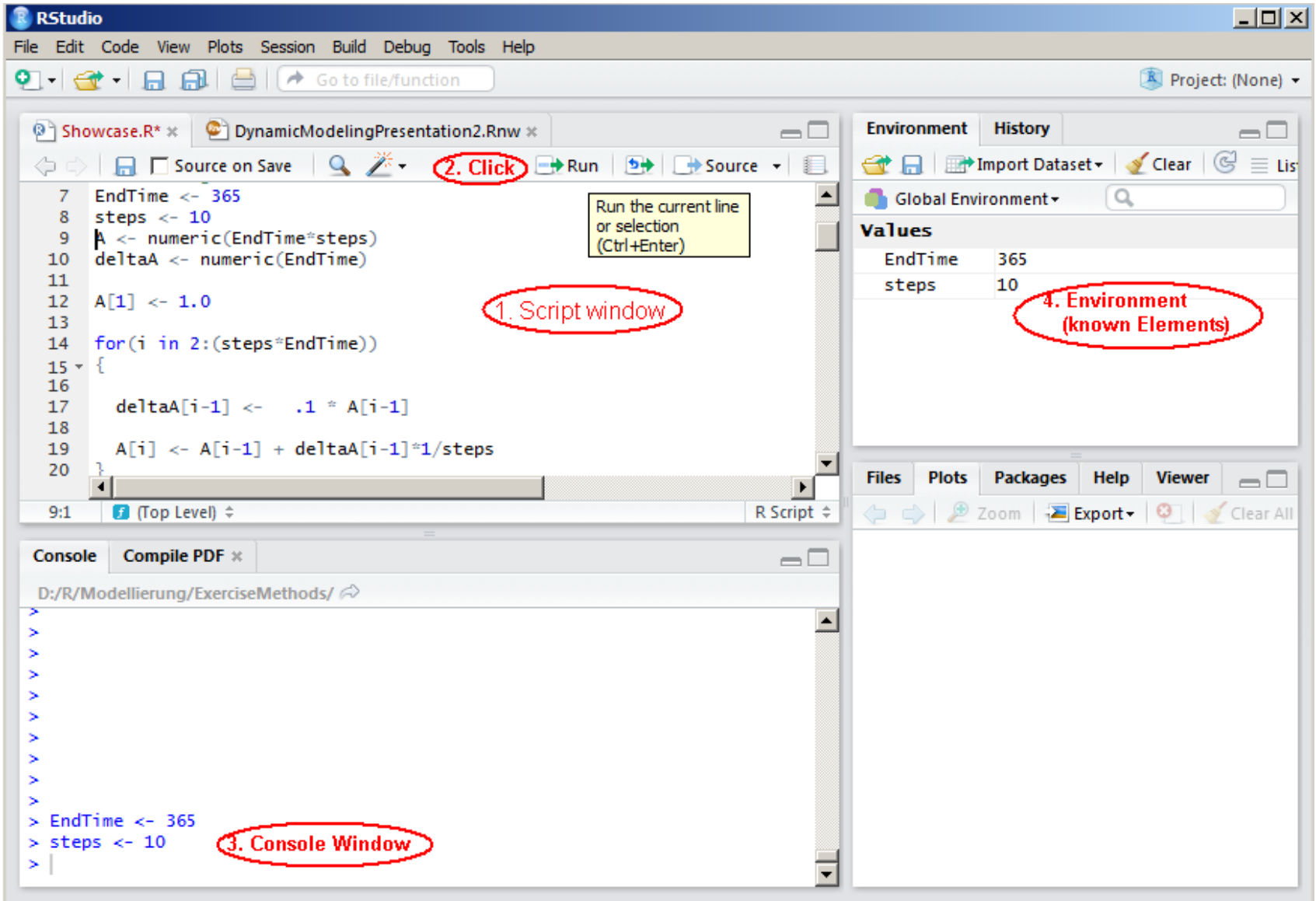
Gunther Krauss

Crop Science Research Group
Institute of Crop Science and Resource Conservation (INRES)

January 11, 2022



RStudio - usage



start:end

If you write `start:end` in R, it gives you a vector of numers from `start` to `end`

```
1:6
```

```
# [1] 1 2 3 4 5 6
```

```
4:9
```

```
# [1] 4 5 6 7 8 9
```

Subscripting vectors

If `V` is a vector, then `V[i]` gives the value of the ith element of the vector.

```
V <- c(3,5,9,6,12,1,7)
V[3] # should return 9
```

```
# [1] 9
```

```
i <- 3
V[i+2]
```

```
# [1] 12
```

```
# i is 3, i+2 is 5,
# V[5] is 12
```

Create empty vectors

To create an empty vector (i.e. all it's elements are 0), you can use `numeric(nrOfElements)`

```
size <- 7
EV <- numeric(size)
EV
```

```
# [1] 0 0 0 0 0 0 0
```

if/else statement

if/else statement is used to conditionally evaluate statements.

```
if (condition)
{
  yes_statement1
  yes_statement2
  ...
}
else
{
  no_statement1
  no_statement2
  ...
}
```

The `for`-loop in R

The `for`-loop is used to repeat tasks/commands in R. Here is the usage of the `for`-loop

```
for(var in sequence) expression
```

And here an example

```
for(var in 1:5) print(var)
```

```
# [1] 1
# [1] 2
# [1] 3
# [1] 4
# [1] 5
```

`for`-loop - looping over sequences

The for-loop takes a sequence (vector, list etc.) and for each element of the sequence it executes the expression. The value of var is substituted with the value of the actual element. In the example above the variable var has subsequently the values `1,2,3,4,5` and the expression `print(var)` is executed 5 times.

Aggregating with `for`

Let's have a look at another usecase of a for-loop:

```
numbers <- c(2,5,4,7,9)
fsum <- 0
for(v in numbers)
{
fsum <- fsum + v
}
fsum
```

```
# [1] 27
```

Here the values from the vector numbers are added consecutively to fsum, giving us the sum of the vector.

for-loop - iterate over the index

Instead of iterating over a vector, we can iterate over the index of a vector:

```
numbers <- c(2,5,4,7,9)
fsum <- 0
for(i in 1:length(numbers))
{
  fsum <- fsum + numbers[i]
  print(paste("Added element",i,"with the value",numbers[i]))
}
```

```
# [1] "Added element 1 with the value 2"
# [1] "Added element 2 with the value 5"
# [1] "Added element 3 with the value 4"
# [1] "Added element 4 with the value 7"
# [1] "Added element 5 with the value 9"
```

```
fsum
```

```
# [1] 27
```

Blocks of statements

If you want to execute more than one statement in a `for`-loop or in an `if`-block, then you have to include the statement in curly brackets {} and separate each statement by a semicolon or a new line

```
for(i in 1:3)
{
  print(paste("hey",i))
  print(paste("ho",3-i))
}
```

```
# [1] "hey 1"
# [1] "ho 2"
# [1] "hey 2"
# [1] "ho 1"
# [1] "hey 3"
# [1] "ho 0"
```

Control structures

`if` and `for` are not R objects or functions, they are reserved keywords. To get help you can't type `?if` or `?for`. You have to put the keywords in quotes

```
?"if"
?"for"
```

Plotting

To make the graphs look better, we passed some additional options to `plot`. To show two variables in one graph we used `matplot`. If you want to know more, you can explore the built-in help.

```
?plot
?matplot
```