01-09: Multiple Conditions

1 - Purpose

- introduce the logical operators and use them to combine conditions in a conditional statement
- create conditional statements that check for multiple values on a single variable
- create conditional statements that check the value of multiple variables

2 - Questions about the material...

If you have any questions about the material in this lesson *feel free to email them to the instructor here*.

3 - More Complex Decisions

Until now our conditional statements have only checked one condition (e.g., Is the animal a llama? Is the temperature greater than 50? Did the fish get caught from north port?). However, we often care about multiple conditions (e.g., Is the animal a llama ***or*** *an alpaca*? Is the *temperature between 50* ***and*** *80*?, Did the fish get caught from north port ***or*** south port?).

Multiple conditions in English are almost always denoted by the words ***and*** and ***or.*** In R, there are operators that represent ***and*** and ***or*** called *logical operators*. ***and*** is represent by **&&** while ***or*** is represented by **||**.

Operator Type	Purpose	R Symbols
Assignment	Assign a value to a variable	= or <-
Mathematical	Perform a mathematical operation on a numeric value	+, -, *, /, ^
Conditional	Compare two values	==, !=, >, <, >=, <=
Logical	Combine conditions	&&, ||, &, |

*Extension: The single character logical operators **&** and **|***

3.1 - Checking for multiple spellings of a word

A couple of lesson ago we asked the user for their favorite cheese and, of course, the answer is "Muenster". However, "Muenster" is not the easiest word to spell -- for example, it is often misspelled "Meunster". We can make our script more robust by using logical operators to check for alternative spellings (e.g., multiple conditions).

Instead of asking: Is ***favCheese*** equal to "Muenster"?
We want to ask: Is ***favCheese*** equal to "M*ue*nster" ***or*** "M*eu*nster"?

But in programming, we need to be more explicit and ask:
Is ***favCheese*** equal to "M*ue*nster" ***or*** is ***favCheese*** equal to "M*eu*nster"?

In R this is written as:

```
if( favCheese == "Muenster" || favCheese == "Meunster" )
if( favCheese == "Muenster" || favCheese == "Meunster" )
```

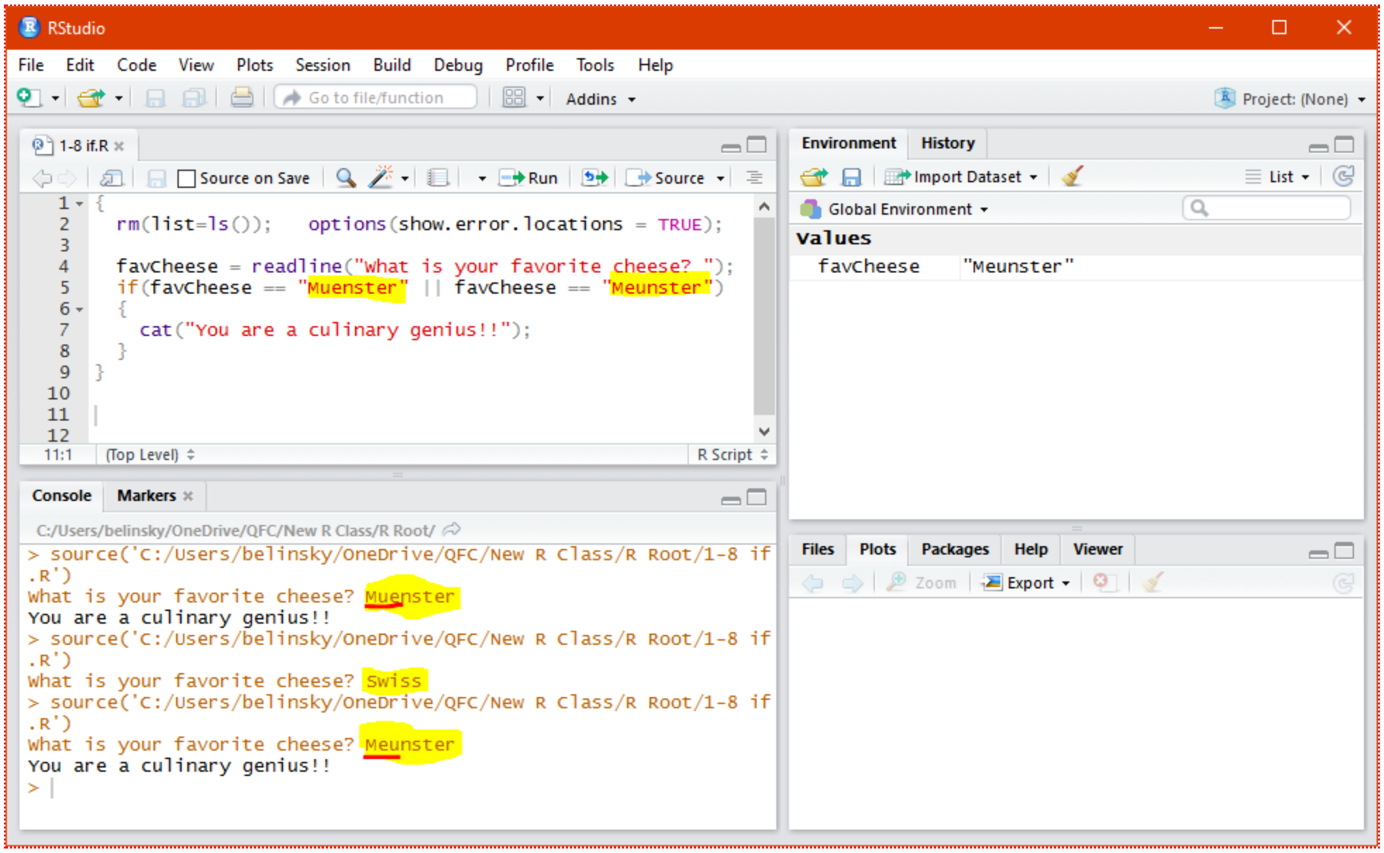
4 - The *or* operator (||) -- checking 2 conditions

We can put the above ***favCheese*** conditions into one conditional statement by using the ***or*** operator. The symbol for the ***or*** operator is (**||**). *Extension: || on the keyboard*. The ***or*** operator takes two conditions and returns ***TRUE*** *if either condition* is ***TRUE*** and returns ***FALSE*** only if both conditions are ***FALSE.***

```
{
  rm(list=ls());   options(show.error.locations = TRUE);

  favCheese = readline("What is your favorite cheese? ");
  if(favCheese == "Muenster" || favCheese == "Meunster")
  {
     cat("You are a culinary genius!!");
  }
}
```

The above code will execute the "culinary genius" codeblock if the user entered either spelling of "Muenster".



```
{
  rm(list=ls());   options(show.error.locations = TRUE);

  favCheese = readline("What is your favorite cheese? ");
  if(favCheese == "Muenster" || favCheese == "Meunster")
  {
    cat("You are a culinary genius!!");
  }
}
```

```
Values
favCheese      "Meunster"
```

```
C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/1-8 if
.R')
What is your favorite cheese? Muenster
You are a culinary genius!!
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/1-8 if
.R')
What is your favorite cheese? Swiss
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/1-8 if
.R')
What is your favorite cheese? Meunster
You are a culinary genius!!
>
```

Fig 1: Using or (||) to checking for two different spellings of our favorite cheese

Note: If there are multiple conditions then *each condition must be explicitly stated*. R will faithfully execute the following code but the result will be ***TRUE*** no matter what the user enters.

```
# error: need to be explicit about the conditions
```

```
if(favCheese == "Muenster" || "Meunster")
{
   cat("I will always be executed because there is
                   a logical error in the conditional statement");
}
```

We talk more about this issue here-- *Trap: All conditional statements must be explicitly stated*

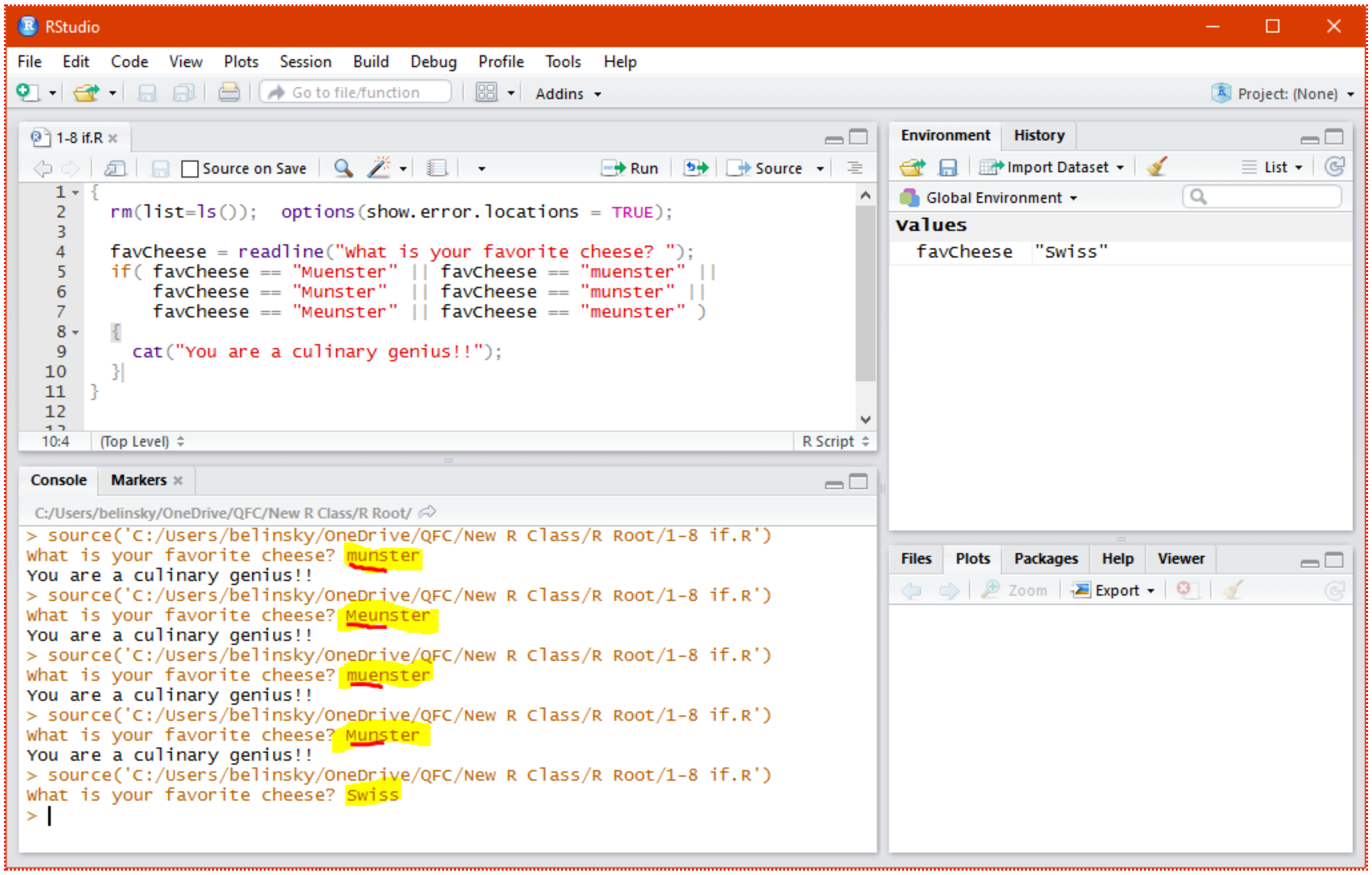
4.1 - The *or* operator (||) -- checking more than 2 conditions

We can use the (**||**) to check for more variations of "Muenster". The conditional statement in the script below checks six variations of "Muenster" and returns ***TRUE*** if ***favCheese*** matches *any of the six spellings.*

```
{
  rm(list=ls());  options(show.error.locations = TRUE);

  favCheese = readline("What is your favorite cheese? ");
  if( favCheese == "Muenster" || favCheese == "muenster" ||
      favCheese == "Munster"  || favCheese == "munster" ||
      favCheese == "Meunster" || favCheese == "meunster" )
  {
     cat("You are a culinary genius!!");
  }
}
```

Note: the conditional statement have been broken up into multiple line to make it easier to read.



```
{
  rm(list=ls());  options(show.error.locations = TRUE);

  favCheese = readline("What is your favorite cheese? ");
  if( favCheese == "Muenster" || favCheese == "muenster" ||
      favCheese == "Munster"  || favCheese == "munster" ||
      favCheese == "Meunster" || favCheese == "meunster" )
  {
    cat("You are a culinary genius!!");
  }
}
```

```
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/1-8 if.R')
What is your favorite cheese? munster
You are a culinary genius!!
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/1-8 if.R')
What is your favorite cheese? Meunster
You are a culinary genius!!
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/1-8 if.R')
What is your favorite cheese? muenster
You are a culinary genius!!
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/1-8 if.R')
What is your favorite cheese? Munster
You are a culinary genius!!
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/1-8 if.R')
What is your favorite cheese? Swiss
> 
```

```
Values
  favCheese    "Swiss"
```

Fig 2: Multiple conditions allowing for many possible spelling of muenster (still have grammar error...)

5 - Getting a range of numbers: the and (&&) operator

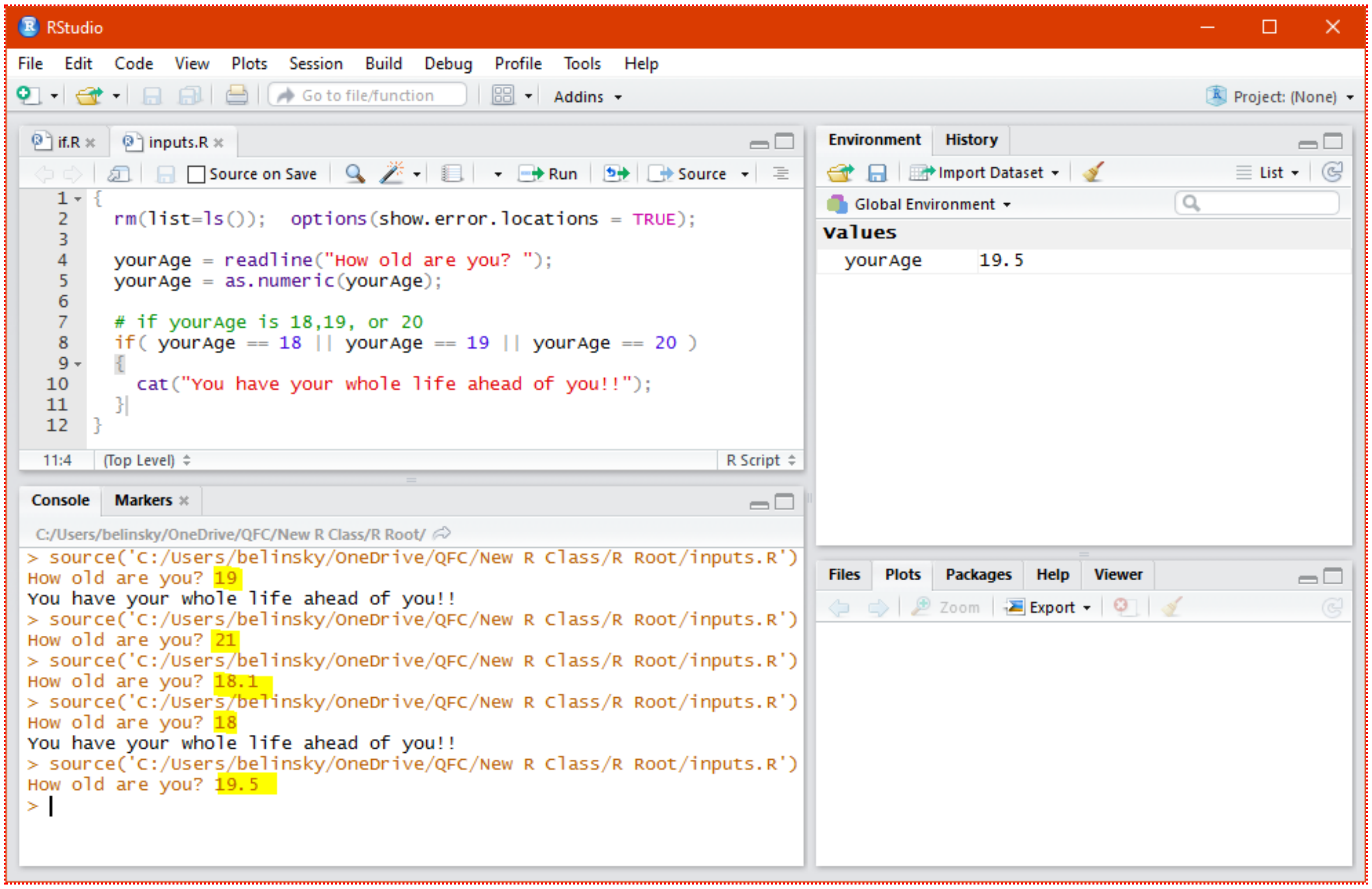
The (**||**) operator can also be used to check for different numeric value. For example if you want to output a message for anyone who is **18**, **19**, or **20** years old you can check the three conditions: ***(yourAge == 18)*** or (***yourAge == 19***) or (***yourAge == 20***).

```
{
  rm(list=ls());  options(show.error.locations = TRUE);

  yourAge = readline("How old are you? ");
  yourAge = as.numeric(yourAge);

  # if yourAge is 18,19, or 20
  if( yourAge == 18 || yourAge == 19 || yourAge == 20 )
  {
     cat("You have your whole life ahead of you!!");
  }
}
```

The conditional statement is only going to be ***TRUE*** for the integers **18**, **19**, and **20**. It will be ***FALSE*** for all other value, including decimal values like 18.5 or 20.1.



```r
{
  rm(list=ls());   options(show.error.locations = TRUE);

  yourAge = readline("How old are you? ");
  yourAge = as.numeric(yourAge);

  # if yourAge is 18,19, or 20
  if( yourAge == 18 || yourAge == 19 || yourAge == 20 )
  {
    cat("You have your whole life ahead of you!!");
  }
}
```

```
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/inputs.R')
How old are you? 19
You have your whole life ahead of you!!
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/inputs.R')
How old are you? 21
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/inputs.R')
How old are you? 18.1
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/inputs.R')
How old are you? 18
You have your whole life ahead of you!!
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/inputs.R')
How old are you? 19.5
>
```

```
Values
  yourAge     19.5
```

Fig 3: Using or (||) to check for three different ages

This coding is unwieldy if you have a larger range of numbers. For example "all ages between **20** and **40**" would require **21** conditions:

```r
if(yourAge == 20 || yourAge ==21 || yourAge == 22 ||
   yourAge == 23 || ... || yourAge == 40 )
```

And you would need an infinite number of conditions if you want to include all decimal numbers in between **20** and **40**.

6 - Multiple conditions using the and (&&) operator

We need to create a conditional statement that looks at a range of numbers (20 through 40) -- this is done with two conditions connected using the ***and*** operator (**&&**).

The picture (*Fig.4*) shows the overlap between two conditions that make up "all ages between **20** and **40**":

1. top blue arrow: (***yourAge > 20***)
2. bottom orange arrow: (***yourAge < 40***)

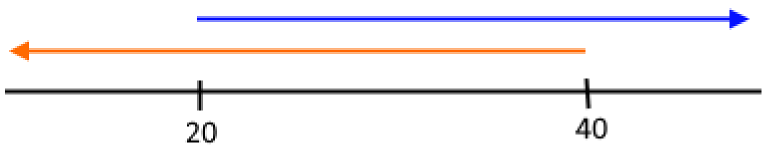


Fig 4: Creating a range using the **&&** operator

The overlap between the two arrows represents when the conditions (***yourAge*** > ***20***) and (***yourAge*** < ***40***) are both ***TRUE***.

In R the statement is:

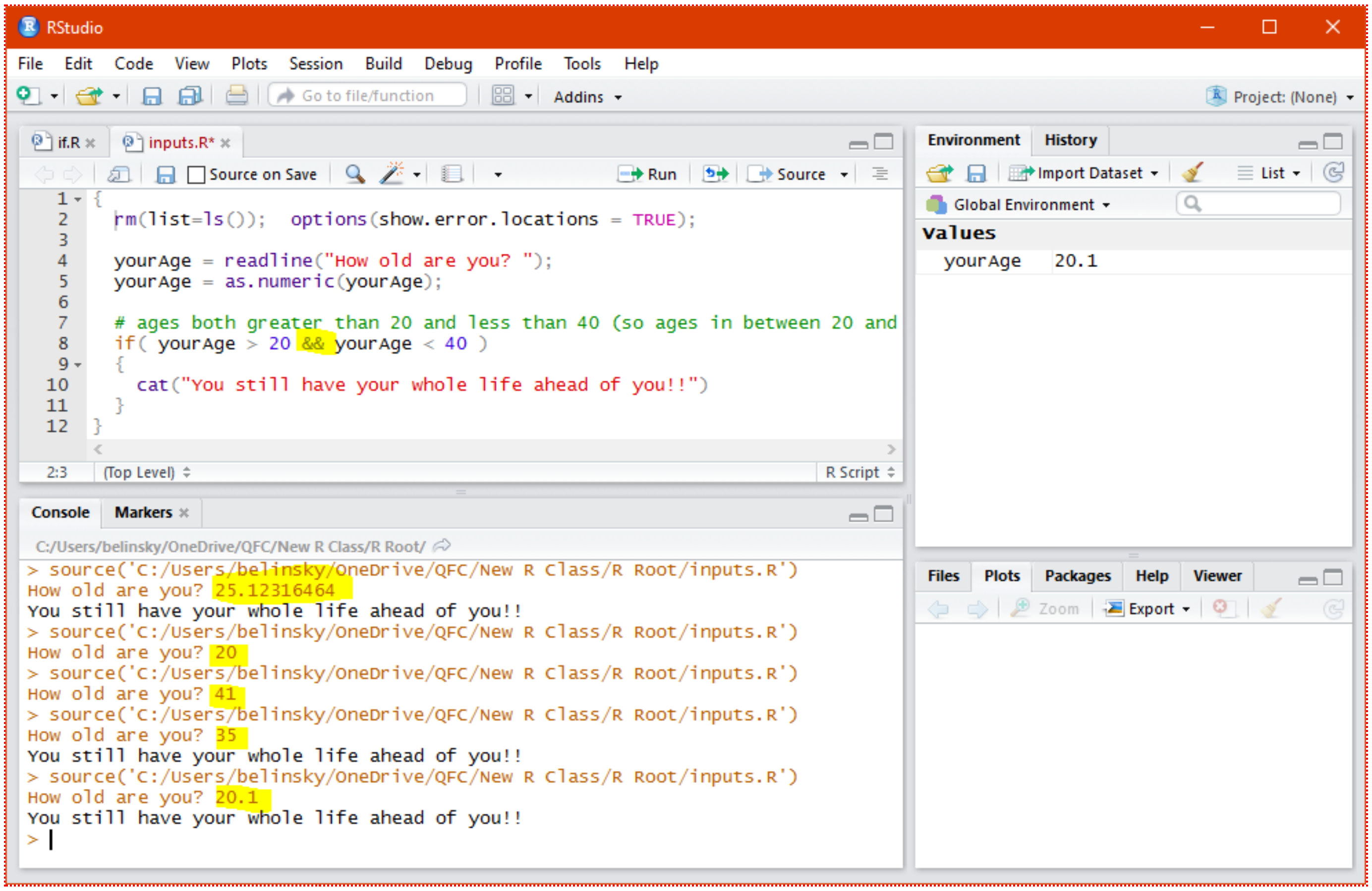
```r
if (yourAge > 20 && yourAge < 40 )
```

And putting the above conditional statement into a script:

```r
{
  rm(list=ls());  options(show.error.locations = TRUE);

  yourAge = readline("How old are you? ");
  yourAge = as.numeric(yourAge);

  # ages both greater than 20 and less than 40 (so ages in between 20 and 40)
  if( yourAge > 20 && yourAge < 40 )
  {
     cat("You still have your whole life ahead of you!!")
  }
}
```



```
Values
  yourAge    20.1
```

```
C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/inputs.R')
How old are you? 25.12316464
You still have your whole life ahead of you!!
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/inputs.R')
How old are you? 20
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/inputs.R')
How old are you? 41
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/inputs.R')
How old are you? 35
You still have your whole life ahead of you!!
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/inputs.R')
How old are you? 20.1
You still have your whole life ahead of you!!
> |
```

*Fig 5: Using the **&&** operator to check a range of numbers*

Where ***or*** *(* **||** *)* outputs ***TRUE*** if either of the conditions are ***TRUE***, ***and*** *(* **&&** *)* outputs ***TRUE*** only *if both the conditions* are ***TRUE***.

6.1 - Checking the values of multiple variable using &&

The ***and*** (***&&***) operator can also be used to make conditional operations on multiple variables. For instance you might want to look for people who like llamas (***favAnimal == "Llama"***) and like Muenster cheese (***favCheese == "Muenster"***):

```
if( favAnimal == "Llama" && favCheese == "Muenster" )  # TRUE for this author
```

or you might want all fish under the age of 5 (***fishAge < 5***) that were caught at night (***catchTime == "night"***):

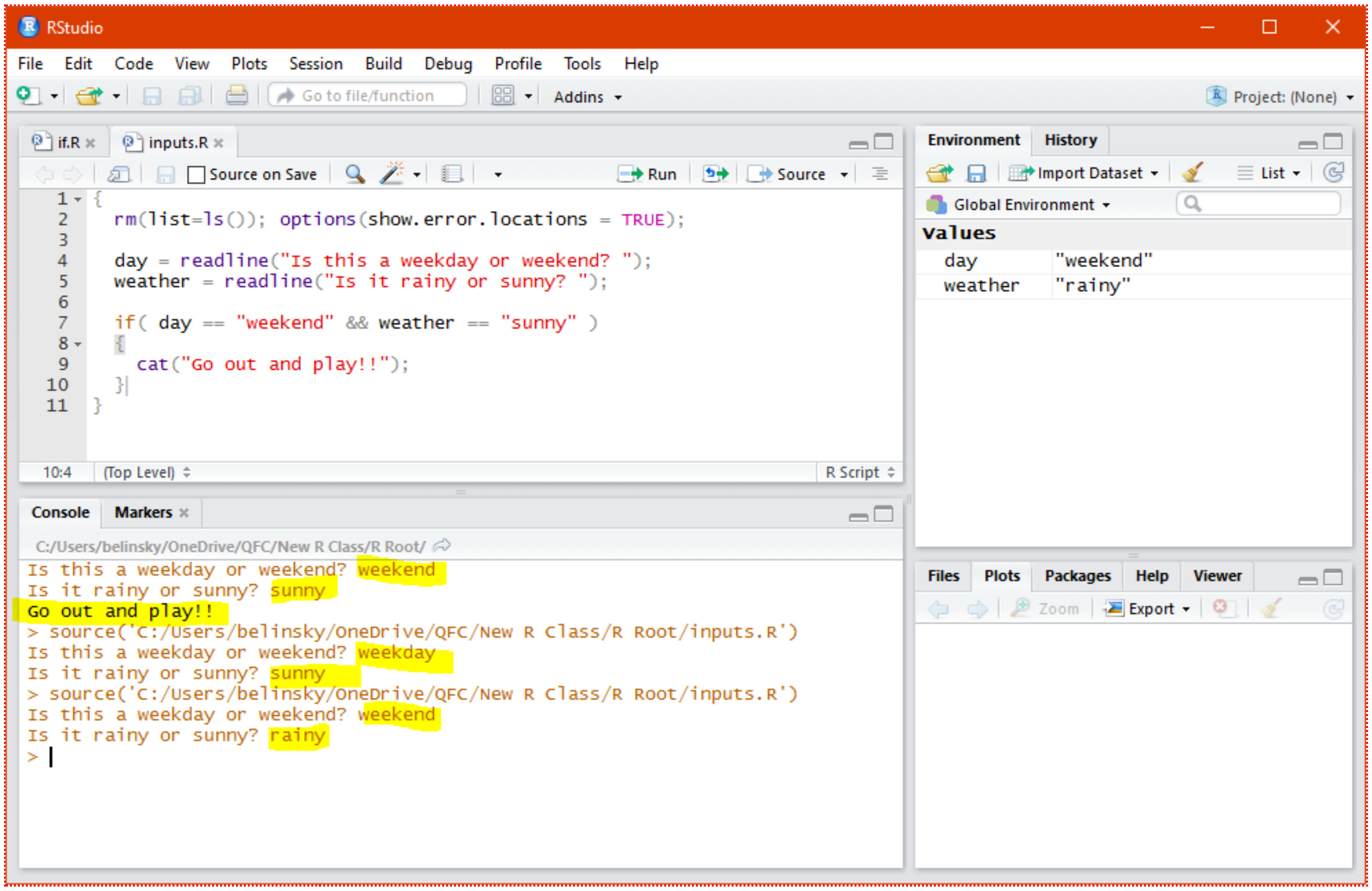
```
if( fishAge < 5 && catchTime == "night" )
```

Or, you might want a simple check of both day and weather:

```
{
  rm(list=ls()); options(show.error.locations = TRUE);

  day = readline("Is this a weekday or weekend? ");
  weather = readline("Is it rainy or sunny? ");

  if( day == "weekend" && weather == "sunny" )
  {
     cat("Go out and play!!");
  }
}
```



```
{
  rm(list=ls()); options(show.error.locations = TRUE);

  day = readline("Is this a weekday or weekend? ");
  weather = readline("Is it rainy or sunny? ");

  if( day == "weekend" && weather == "sunny" )
  {
    cat("Go out and play!!");
  }
}
```

```
Is this a weekday or weekend? weekend
Is it rainy or sunny? sunny
Go out and play!!
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/inputs.R')
Is this a weekday or weekend? weekday
Is it rainy or sunny? sunny
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/inputs.R')
Is this a weekday or weekend? weekend
Is it rainy or sunny? rainy
> 
```

```
Values
  day        "weekend"
  weather    "rainy"
```

*Fig 6: Using **&&** to check the value of two different variables*

7 - If-else-if with the and (&&) operator

The above script (*Fig.6*) has two variables (***day*** and ***weather***) and each variable has two possible values (***weekday***,***weekend*** **&** ***sunny***,***rainy***).
This means there are *four possible combinations of day and weather:*

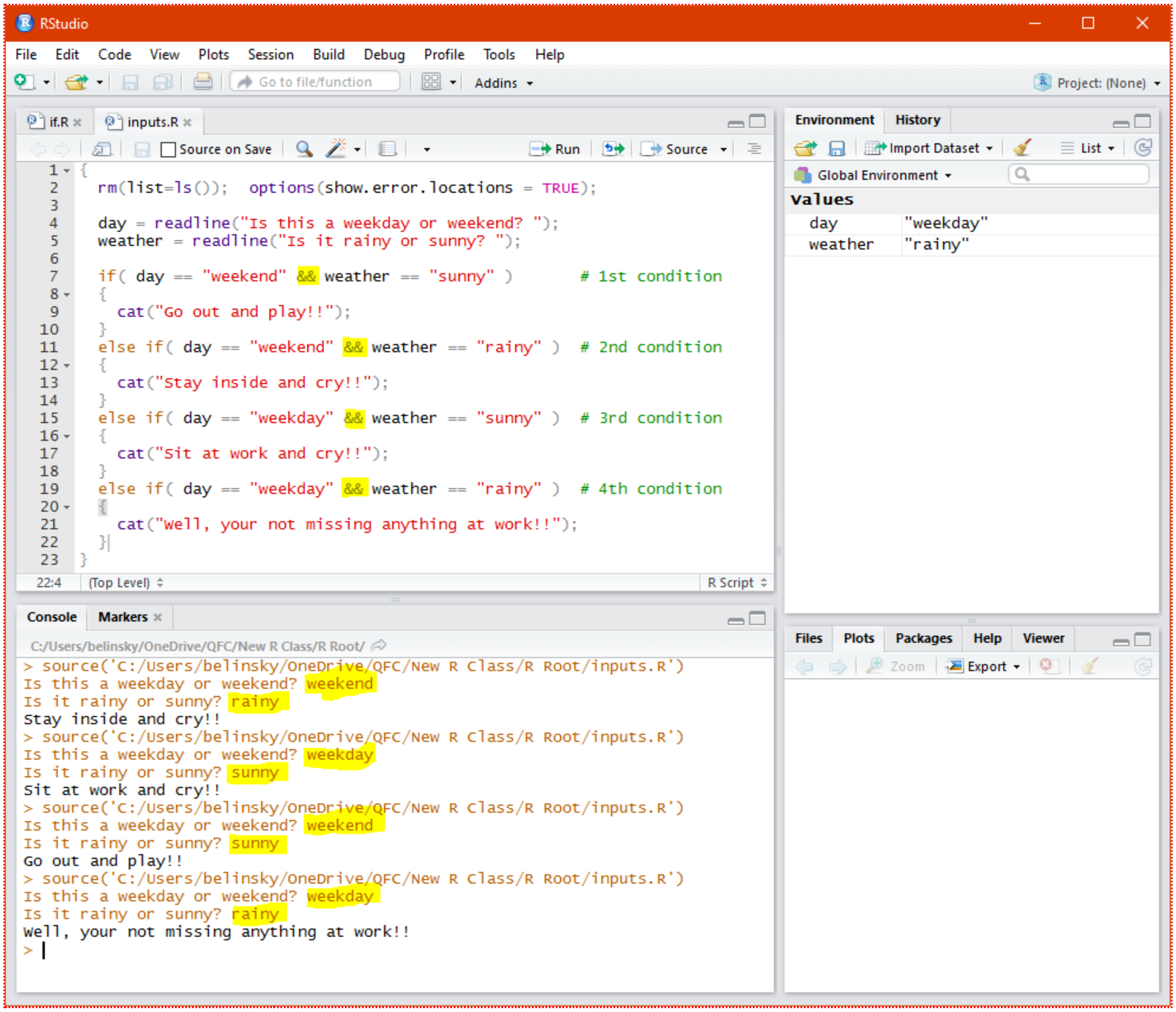
1. weekday/sunny
2. weekday/rainy
3. weekend/sunny
4. weekend/rainy

We can use an ***if-else-if*** structure to handle the four possible ***day/weather*** conditions and provide a different response for each of the four possibilities:

```
{
  rm(list=ls());  options(show.error.locations = TRUE);

  day = readline("Is this a weekday or weekend? ");
  weather = readline("Is it rainy or sunny? ");

  if( day == "weekend" && weather == "sunny" )       # 1st condition
  {
     cat("Go out and play!!");
  }
  else if( day == "weekend" && weather == "rainy" )  # 2nd condition
  {
     cat("Stay inside and cry!!");
  }
  else if( day == "weekday" && weather == "sunny" )  # 3rd condition
  {
     cat("Sit at work and cry!!");
  }
  else if( day == "weekday" && weather == "rainy" )  # 4th condition
  {
     cat("Well, your not missing anything at work!!");
  }
}
```



```
{
  rm(list=ls());  options(show.error.locations = TRUE);

  day = readline("Is this a weekday or weekend? ");
  weather = readline("Is it rainy or sunny? ");

  if( day == "weekend" && weather == "sunny" )       # 1st condition
  {
    cat("Go out and play!!");
  }
  else if( day == "weekend" && weather == "rainy" )  # 2nd condition
  {
    cat("Stay inside and cry!!");
  }
  else if( day == "weekday" && weather == "sunny" )  # 3rd condition
  {
    cat("Sit at work and cry!!");
  }
  else if( day == "weekday" && weather == "rainy" )  # 4th condition
  {
    cat("Well, your not missing anything at work!!");
  }
}
```

Values

day	"weekday"
weather	"rainy"

```
C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/inputs.R')
Is this a weekday or weekend? weekend
Is it rainy or sunny? rainy
Stay inside and cry!!
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/inputs.R')
Is this a weekday or weekend? weekday
Is it rainy or sunny? sunny
Sit at work and cry!!
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/inputs.R')
Is this a weekday or weekend? weekend
Is it rainy or sunny? sunny
Go out and play!!
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/inputs.R')
Is this a weekday or weekend? weekday
Is it rainy or sunny? rainy
Well, your not missing anything at work!!
> 
```

*Fig 7: Checking the four possible **day** and **weather** conditions*

7.1 - Using else as an error condition

In the previous script (*Fig.7*), the four possible combinations of day and weather are presented in the ***if-else-if*** structure and the script outputs a message for all four conditions. However, there is no message if all four conditions fails. In other words, we have no error condition.

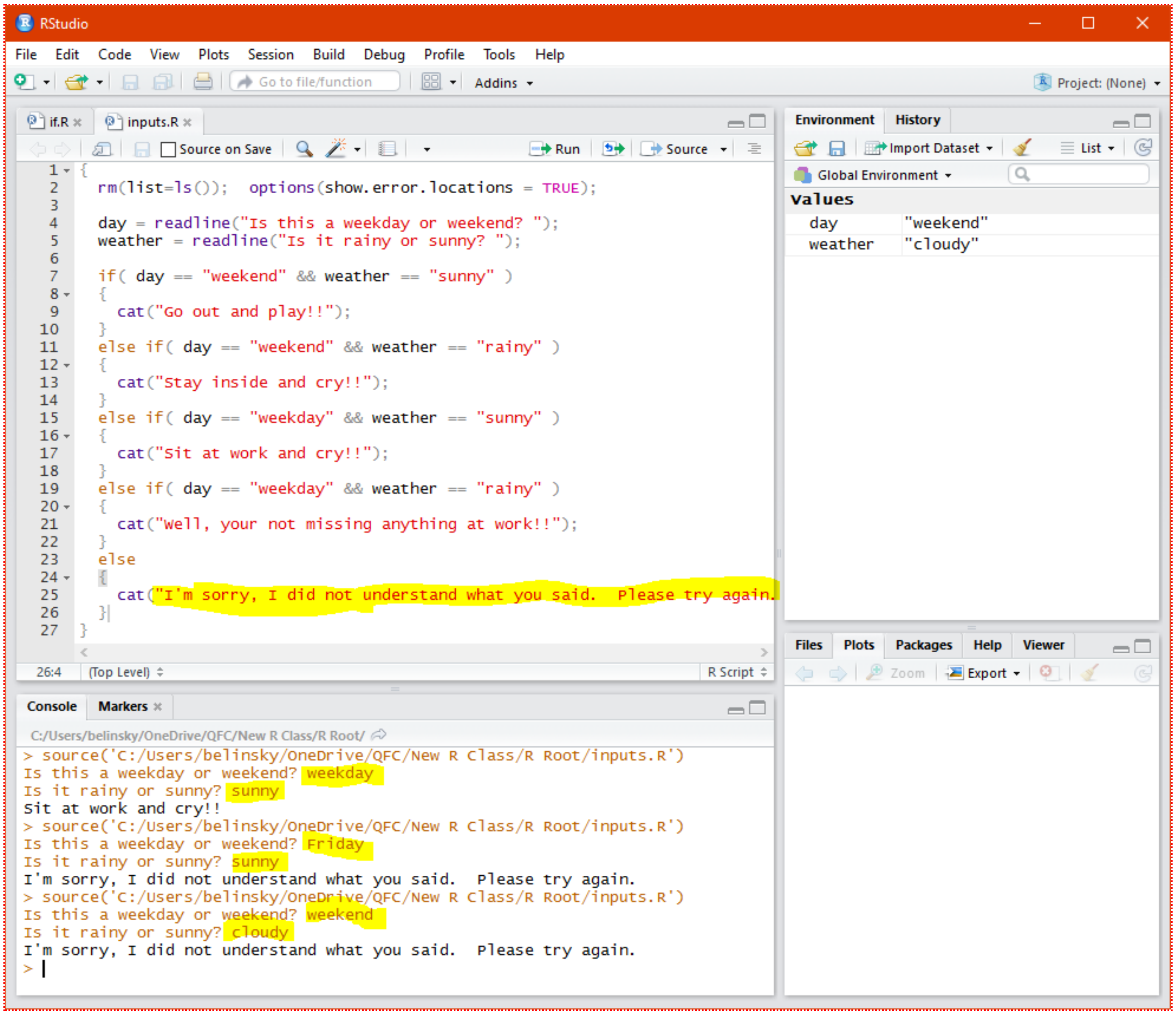
To add an *error condition*, we attach an ***else*** statement at the end of the ***if-else-if*** structure. The ***else*** statement is a waste-basket condition that captures anything the ***if-else-if*** structure missed.

So, we can take the above day/weather example and add an ***else*** as an error condition to capture every other possible input from the user:

```
{
  rm(list=ls());  options(show.error.locations = TRUE);

  day = readline("Is this a weekday or weekend? ");
  weather = readline("Is it rainy or sunny? ");

  if( day == "weekend" && weather == "sunny" )
  {
     cat("Go out and play!!");
  }
  else if( day == "weekend" && weather == "rainy" )
  {
     cat("Stay inside and cry!!");
  }
  else if( day == "weekday" && weather == "sunny" )
  {
     cat("Sit at work and cry!!");
  }
  else if( day == "weekday" && weather == "rainy" )
  {
     cat("Well, your not missing anything at work!!");
  }
  else
  {
    cat("I'm sorry, I did not understand what you said.  Please try again.");
  }
}
```



```
{
  rm(list=ls());  options(show.error.locations = TRUE);

  day = readline("Is this a weekday or weekend? ");
  weather = readline("Is it rainy or sunny? ");

  if( day == "weekend" && weather == "sunny" )
  {
    cat("Go out and play!!");
  }
  else if( day == "weekend" && weather == "rainy" )
  {
    cat("Stay inside and cry!!");
  }
  else if( day == "weekday" && weather == "sunny" )
  {
    cat("Sit at work and cry!!");
  }
  else if( day == "weekday" && weather == "rainy" )
  {
    cat("Well, your not missing anything at work!!");
  }
  else
  {
    cat("I'm sorry, I did not understand what you said.  Please try again.
  }
}
```

```
Values
  day       "weekend"
  weather   "cloudy"
```

```
C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/inputs.R')
Is this a weekday or weekend? weekday
Is it rainy or sunny? sunny
Sit at work and cry!!
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/inputs.R')
Is this a weekday or weekend? Friday
Is it rainy or sunny? sunny
I'm sorry, I did not understand what you said.  Please try again.
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/inputs.R')
Is this a weekday or weekend? weekend
Is it rainy or sunny? cloudy
I'm sorry, I did not understand what you said.  Please try again.
> |
```

*Fig 8: Adding an error condition (**else**) to an **if-else-if** structure*

It is good programming practice in an ***if-else-if*** structure to create an error statement (*Fig.8*) that executes when all other conditions are checked and returns ***FALSE***.

8 - Application

If you have any questions regarding this application, feel free to email them to the instructor here.
You can attach files to the email above or send the whole Root Folder as a zipped file. Instructions for zipping the Root Folder are here.

A) Have a user enter values for:
1) The age of a fish
2) The weight of the fish
3) The location that the fish was caught (north or south)
4) The gender of the fish

B) Give a message if the fish is between 5 and 8 years old.

C) Give a message if the fish weighs between 50 and 150 grams.

D) Give a message for each of the four possible gender/port conditions (male & female, north and south) and add an error case for values that don't match any of the conditions.

E) Challenge: Give a message if the fish weighs between 20 and 100 grams and comes from either the north or south port.

Save you script file as ***app1-9.r*** *in the* ***scripts*** *folder of your RStudio Project for the class.*

9 - Extension: The or (||) operator on the keyboard

The ***or*** symbol (**||**) is made up of two "pipe characters" (**|**). On most keyboards, the pipe character (**|**) is on the same key as the backslash (\) and right above the ***Enter*** (*Fig 9*. Sometimes the pipe symbol will be broken like this: **¦**

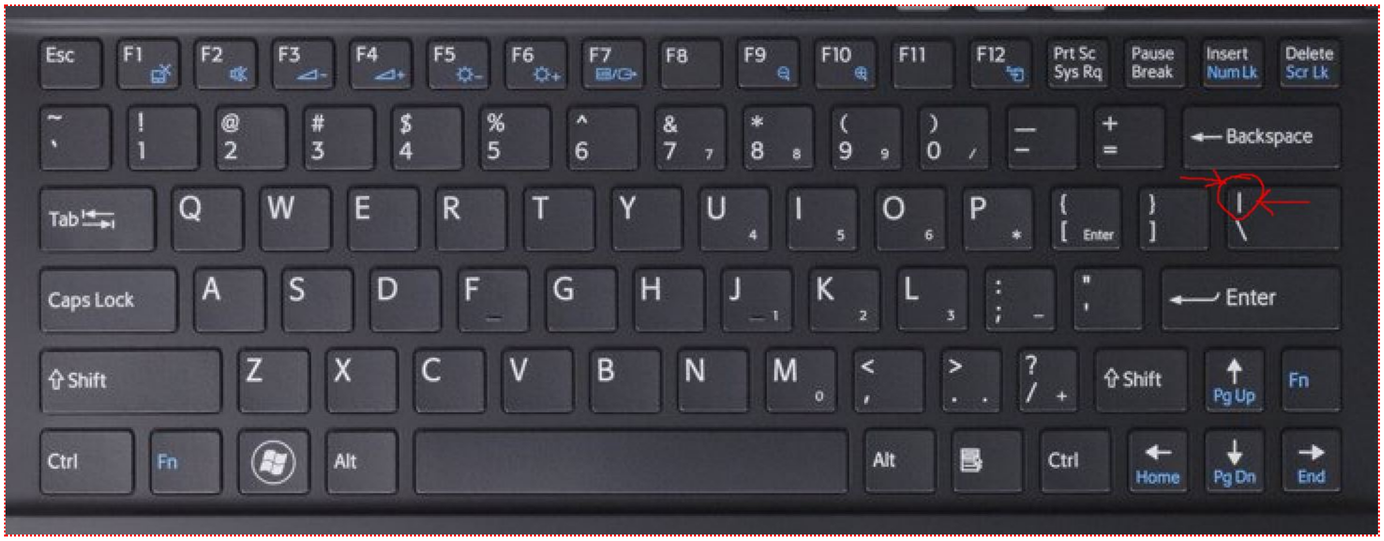


Fig 9: Keyboard - placement of pipe character

10 - Trap: All conditional statements must be explicit

When we are verbalizing conditional statements we often skip variable names if they have already been used for instance:

- If the fish's age is 2 or 3
- If your favorite animal is a llama or an alpaca

Because of this, it is intuitive to make the corresponding conditional statements:

- if (fishAge == 2 || 3)
- if (favAnimal == "llama" || "alpaca")

But in scripting every conditional statement must be explicit -- in other words all conditions need a variable and a value. In English this would be:

- If the fish's age is 2 or the fish's age is 3
- If the favorite animal is a llama or the favorite animal is an alpaca

And in script these conditional statements are:

- if (fishAge == 2 || fishAge == 3)
- if (favAnimal == "llama" || favAnimal == "alpaca")

10.1 - Why the conditions must be explicit

The following statements:

- if (fishAge == 2 || 3)
- if (favAnimal == "llama" || "alpaca")

Are effectively making the statements:

- if (fishAge == 2) ***or*** if(3)
- if (favAnimal == "llama") ***or*** if("alpaca")

if("alpaca"***)*** will cause an error because string values cannot be translated by R into a logical value (i.e., ***TRUE*** or ***FALSE***)

However, *R can translate all numeric values into a logical value*. In fact, all numeric values except **0**, get translated to ***TRUE*** and **0** gets translated as ***FALSE***.

So, ***3***, when used in a conditional statement is ***TRUE***, meaning **if(**fishAge == 2 || 3**)** is ***TRUE*** no matter what the value of ***fishAge*** is.

11 - Extension: The single character logical operators: and (&) or (|)

You will often see the operators **&** and **|** used in place of ***&&*** and **||**. For all the examples we have done so far, the single logical operators (***&,*** **|**) are functionally equivalent to the double logical operators (***&&,*** **||**). This is because we have only looked at variables with one value. The functionality between the single and double logical operator change when we start dealing with *variables that have multiple values* (called *vectors*). We will talk more about this when we introduce vectors. Essentially, single logical operators look at individual values in vectors whereas double operators look at the whole vector.