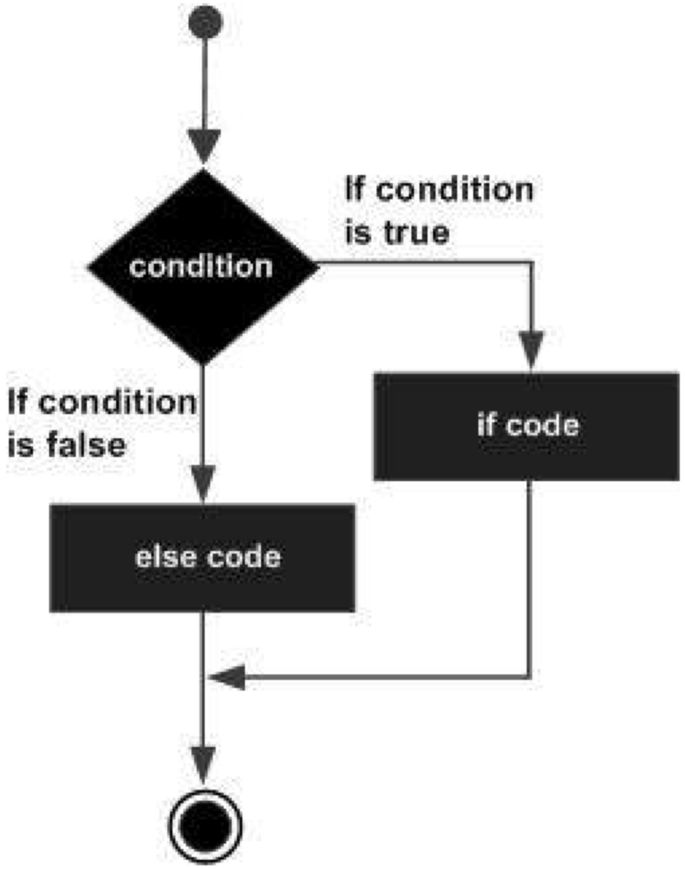
Session 3

(Customer Analytics)

Josep Curto
Professor, IE Business School | CEO | DS

Control statements – if/else/ifelse



Control statements – if/else/ifelse

```
if(boolean_expression 1) {
   // Executes when the boolean expression 1 is true.
} else if( boolean_expression 2) {
   // Executes when the boolean expression 2 is true.
} else if( boolean_expression 3) {
   // Executes when the boolean expression 3 is true.
} else {
   // executes when none of the above condition is true.
}
```

```
ifelse(Boolean_Vector,Outcome_If_True,Outcome_If_False)
```

Control statements – Logical Operators

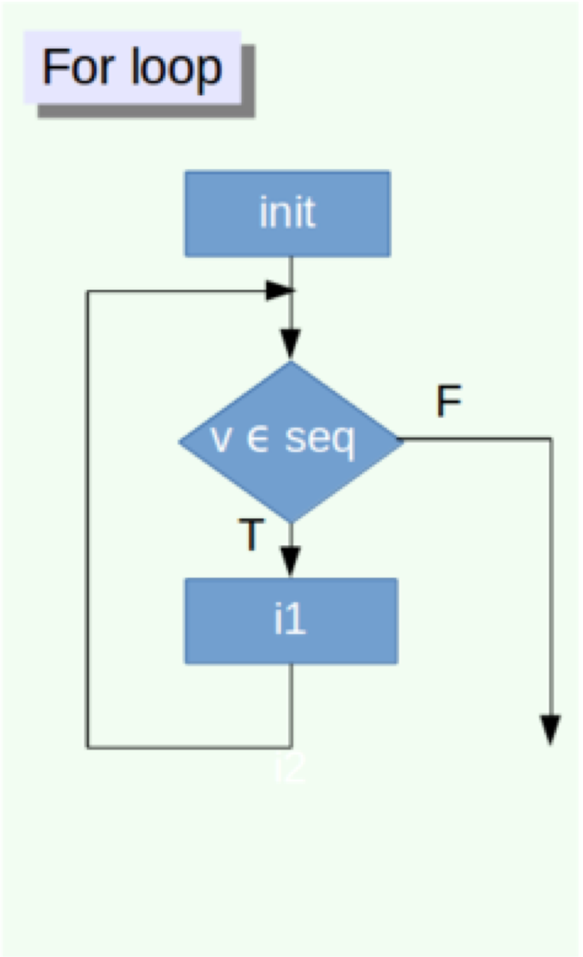
Operators	Meaning
<	Less than
<=	Less than or equal to
>	More than
>=	More than or equal to
==	Equal to
!=	Not equal to
!a	Not a
a|b	a or b
a&b	a and b
isTRUE(a)	Test if a is true

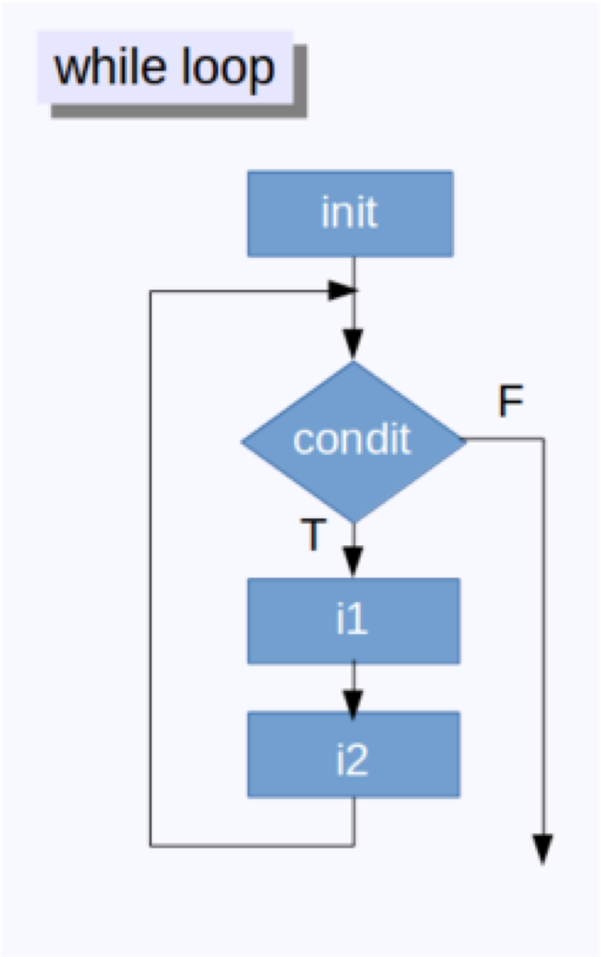
Control statements – if/else/ifelse

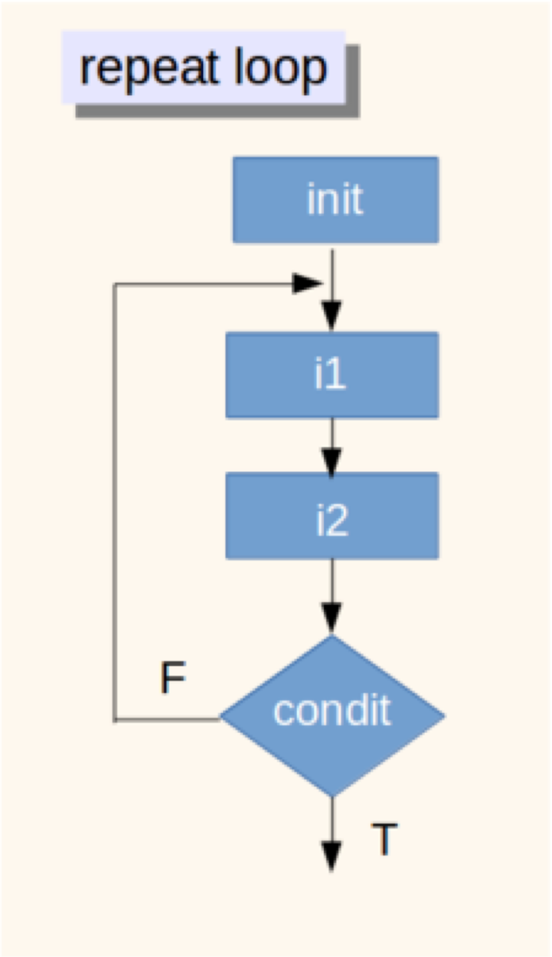
```r
# Our first control statement
x <- c("What","is","truth")

if("Truth" %in% x){
  print("Truth is found the first time")
}else if("truth" %in% x){
  print("truth is found the second time")
}else{
  print("No truth found")
}
```

```r
ifelse("Truth" %in% x,
       print("Truth is found the first time"),
       ifelse("truth" %in% x,
              print("truth is found the second time"),
              print("No truth found"))
)
```

Control statements – Loops







Control statements – Loops

```r
for(i in 1:10){
  if(i==6){
    next
  }
  if(i==8){
    break
  }
  print(i)
}
```

```r
i <- 10
while(i>0){
  print(i)
  i <- i-1
  if(i==8){
    next
  }else{
    print("ok")
  }
}
```

```r
v <- c("Hello IE", "looping")
counter <- 2

repeat{
  if(counter>5){
    break
  }
  print(c(v,counter))
  counter <- counter + 1
}
```

Control statements – Exercises

- Write a repeat{} loop that prints all the even numbers from 2 – 10
  - Start with i<-0

- Write a while() loop that increments the variable, “i“, 6 times, and prints “msg” at every iteration using the following variables:
  - msg <- c("Hello IE")
  - i <- 1

- Write a for() loop that prints the first four numbers of this sequence: x <- c(7, 4, 3, 8, 9, 25)

Writing functions

```r
say.hello <- function(name){
  print(sprintf("Hello %s",name))
}

hello.person <- function(first, last){
  print(sprintf("Hello %s %s",first, last))
}

say.hello("Josep")

hello.person("Josep","Curto")
hello.person(last="Jobs","Steve")
```

```r
hello.person(last="Gates")
```

Functions - Exercises

- Create a function that will return the sum of 2 numbers

- Create a function what will return TRUE if a given integer is inside a vector
  - With while
  - With for
  - With %in%

- Create a function that given a data frame will print by screen the name of the column and the class of data it contains (e.g. Variable1 is Numeric).

- Create a function that given an integer will calculate how many divisors it has (other than 1 and itself). Make the divisors appear by screen

Reading files

`read.csv`	for comma separated values with period as decimal separator.
`read.csv2`	for semicolon separated values with comma as decimal separator.
`read.delim`	tab-delimited files with period as decimal separator.
`read.delim2`	tab-delimited files with comma as decimal separator.
`read.fwf`	data with a predetermined number of bytes per column.

Argument	description
`header`	Does the first line contain column names?
`col.names`	`character` vector with column names.
`na.string`	Which strings should be considered `NA`?
`colClasses`	`character` vector with the types of columns. Will coerce the columns to the specified types.
`stringsAsFactors`	If `TRUE`, converts all `character` vectors into `factor` vectors.
`sep`†	Field separator.

†Used only internally by `read.fwf`

Reading files

`read.csv`	for comma separated values with period as decimal separator.
`read.csv2`	for semicolon separated values with comma as decimal separator.
`read.delim`	tab-delimited files with period as decimal separator.
`read.delim2`	tab-delimited files with comma as decimal separator.
`read.fwf`	data with a predetermined number of bytes per column.

Argument	description
`header`	Does the first line contain column names?
`col.names`	`character` vector with column names.
`na.string`	Which strings should be considered `NA`?
`colClasses`	`character` vector with the types of columns. Will coerce the columns to the specified types.
`stringsAsFactors`	If `TRUE`, converts all `character` vectors into `factor` vectors.
`sep`†	Field separator.

†Used only internally by `read.fwf`

Reading CSV files

```r
#choosing the file
customerData <- read.csv(file.choose())

# Load a csv file
customerData <- read.csv("s3.csv")
```

```r
#Validate that two datasets are the same
data2<-read.csv(file="exampledata.dat",header=TRUE,sep=" ")
all.equal(data,data2)

#Load data from a website
theurl<-"http://jaredlander.com/data/Tomato%20First.csv"
tomato<-read.table(file=theurl,header=TRUE,sep=",")

#choosing the file
save(tomato,file="tomato.data")
rm(tomato)
load(file="./data/tomato.data")
```

```r
#reading from a url
fileURL <- "http://www.stats.ox.ac.uk/pub/datasets/csb/ch11b.dat"
download.file(fileURL,destfi
data <- read.table("exampledata.dat",header=TRUE)
head(data)
```

Open Link (Command+Click)

Now we jump to GGPLOT2! Let's make some graphs!

Session Wrap-up

Q&A


Thank you!
Josep Curto | @josepcurto | jcurto@faculty.ie.edu