Microsoft

Intro to R

Mithun Prasad, PhD
miprasad@Microsoft.com

R's Data Structures

R's BASE DATA STRUCTURES

	Homogeneous	**Heterogeneous**
1d	Atomic vector	List
2d	Matrix	Data frame
nd	Array	

R's Basic Data Types and Data Structures

Numerics
Integers
Characters
Logical

Vector
Matrix
Array
List
Data Frame

How Many Dimensions Does your Data Have ?

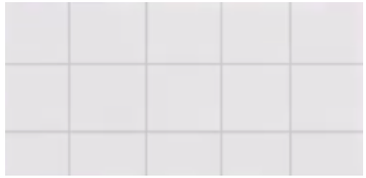
1?

Vector

>1?

How Many Data Types ?

1?



Matrix

Array

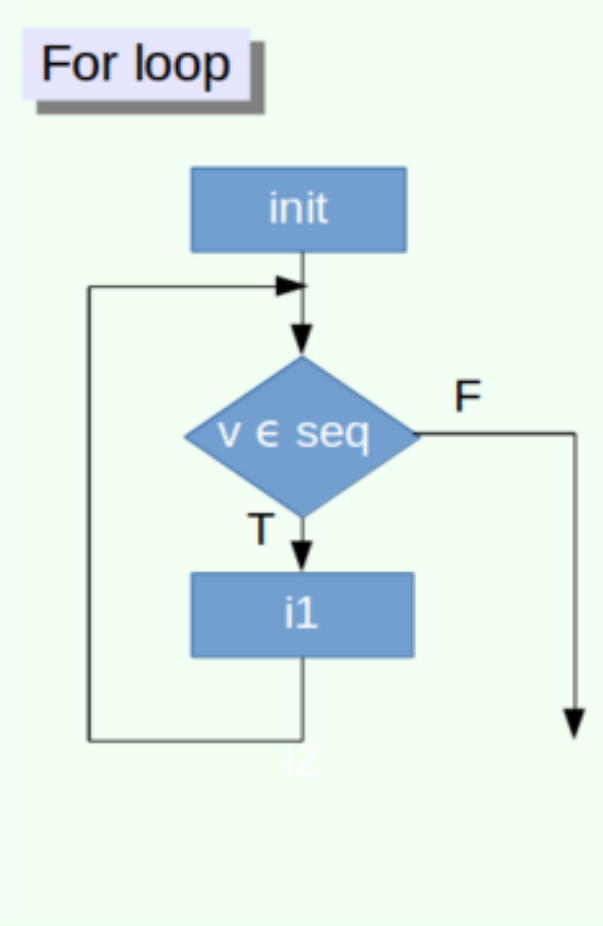
>1?

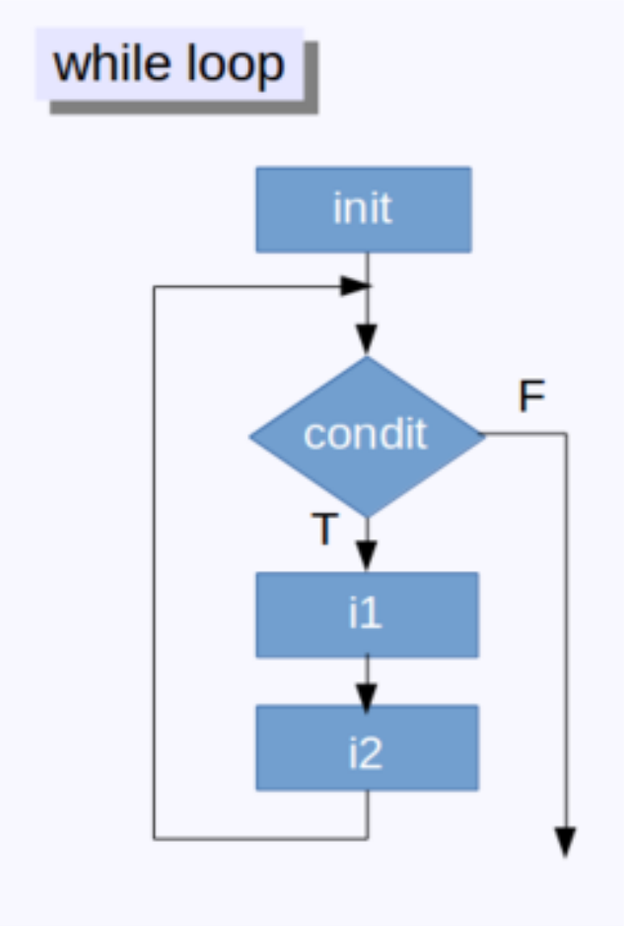
Data Frame

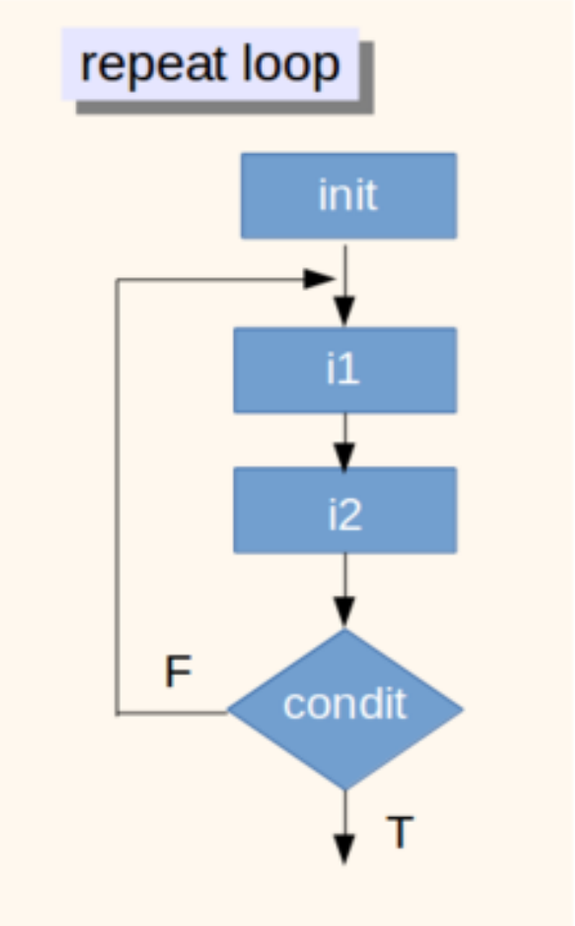
Microsoft

Control Flow

Loops







For Loop Example

```
u1 <- rnorm(30)
print("This loop calculates the square of the first 10 elements of vector u1")

# Initialize `usq`
usq <- 0

for(i in 1:10) {
  # i-th element of `u1` squared into `i`-th position of `usq`
  usq[i] <- u1[i]*u1[i]
  print(usq[i])
}
```

While Loop Example

```
readinteger <- function(){
  n <- readline(prompt="Please, enter your ANSWER: ")
}

response <- as.integer(readinteger())

while (response!=42) {
  print("Sorry, the answer to whatever the question MUST be 42");
  response <- as.integer(readinteger());
}
```

Repeat Loop Example

```
readinteger <- function(){
  n <- readline(prompt="Please, enter your ANSWER: ")
}

repeat {
  response <- as.integer(readinteger());
  if (response == 42) {
    print("Well done!");
    break
  } else print("Sorry, the answer to whatever the question MUST be 42");
}
```

Functions

```
function.name <- function(arguments)
{
  computations on the arguments
  some other code
}

check <- function(x) {
   if (x > 0) {
       result <- "Positive"
   }
   else if (x < 0) {
       result <- "Negative"
   }
   else {
       result <- "Zero"
   }
   return(result)
}
```

Microsoft

Conditions

```
if (test_expression) {
    statement1
} else {
    statement2
}
```

```
ifelse(test_expression,x,y)
```

Error Handling

Error Handling Functions

- warning(…) — outputs a message after a function finishes
- stop(…) — stops the execution of the function and outputs an error message
- suppressWarnings(expr) — evaluates expression and ignores any warnings
- tryCatch(…) — evaluates code and assigns exception handlers

```
result = tryCatch({
    expr
}, warning = function(w) {
    warning-handler-code
}, error = function(e) {
    error-handler-code
}, finally = {
    cleanup-code
}
```

Microsoft

Error Handling

- What happens when something goes wrong with your R code? What do you do?
- What tools do you have to address the problem?

Crash on an Unanticipated Error

```
inputs = list(1, 2, 4, -5, "oops", 0, 10)

for (input in inputs)
{
    print(paste("log of", input, "=", log(input)))
}
```

```
[1] "log of 1 = 0"
[1] "log of 2 = 0.693147180559945"
[1] "log of 4 = 1.38629436111989"
[1] "log of -5 = NaN"
Error in log(input) : Non-numeric argument to mathematical function
In addition: Warning message:
In log(input) : NaNs produced
```

Microsoft

Try Block

```
for (input in inputs)
{
    try(print(paste("log of", input, "=", log(input))))
}
```

```
[[1] "log of 1 = 0"
[1] "log of 2 = 0.693147180559945"
[1] "log of 4 = 1.38629436111989"
[1] "log of -5 = NaN"
Error in log(input) : Non-numeric argument to mathematical function
In addition: Warning message:
In log(input) : NaNs produced
[1] "log of 0 = -Inf"
[1] "log of 10 = 2.30258509299405"
```

Try Catch

```
for(input in inputs) {
    tryCatch(print(paste("log of", input, "=", log(input))),
             warning = function(w) {print(paste("negative argument", input));
                                    log(-input)},
             error = function(e) {print(paste("non-numeric argument", input));
                                  NaN})
 }
```

```
[1] "log of 1 = 0"
[1] "log of 2 = 0.693147180559945"
[1] "log of 4 = 1.38629436111989"
[1] "negative argument -5"
[1] "non-numeric argument oops"
[1] "log of 0 = -Inf"
[1] "log of 10 = 2.30258509299405"
```

Microsoft