Programming (ERIM)

Lecture 2: Control flow, branching, loop constructs

Tommi Tervonen

Econometric Institute, Erasmus School of Economics

ERASMUS UNIVERSITEIT ROTTERDAM

Course contents

L1	Introduction to programming paradigms and weakly typed languages
L2	Control flow, branching, loop constructs
L3	Subroutines and scoping
L4	Side effects, functions and procedures
L5	Programming by contract
L6	Test-driven development
L7	Vectorization
L8	Anonymous functions
L9	- Free topic -
L10	Parallel computing

Control flow

- Statements get evaluated one by one
- Control flow can branch depending on logical conditions

```
x = 2 + 2;
if x <= 4
  disp('x is max 4')
else
  disp('x is over 4')
  x
end % all blocks
% need an end in Matlab
```

```
x <- 2 + 2
if (x <= 4)
  message('x is max 4')
else {
  message('x is over 4')
  print(x)
} # a block statement
```

Logical operators

```
~x                                   !x
x  ==  y                             x  ==  y
x  ~=  y                             x  !=  y
x  <  y                              x  <  y
x  >  y                              x  >  y
x  <=  y                             x  <=  y
x  >=  y                             x  >=  y
x  >  y  &&  y  >  10                x  >  y  &&  y  >  10
x  <  y  ||  x  <  10                x  <  y  ||  x  <  10
```

- R boolean type values TRUE and FALSE (T and F)
- Matlab has no boolean (logical) type! i.e.:

```
~2  ==  0;  ~1  ==  0;  ~0  ==  1
```

Iteration: for-loop

- For-loop allows to iterate over a list of values

```
for x=1:10
  disp(x)
end

for x=[2, 4, 5]
 disp(x)
end
```

```
for (x in 1:10) {
  message(x)
}

for (x in c(2, 4, 5)) {
  message(x)
}
```

Iteration: while-loop

- Iterate while a condition holds
- Beware of infinite loops
- `break` allows to jump out from the closest loop, `next` / `continue` continues from the end of the current iteration

```
x = 1;
while (1) % dangerous
  disp(x)
  x = x + 1;
  if (x >= 10)
    break
  end
end
```

```
x <- 1
while (x <= 10) {
  x <- x + 1
  if (x %% 2 == 0) {
    next
  } else {
    message(x)
  }
}
```

Implicit looping in R

- Evaluate loop on a list of arguments, and return a new list with the evaluation results
- `apply, lapply, sapply`

```
x <- matrix(c(1, 2, 3, 4), byrow=TRUE, ncol=2)

# margin 1 is over first dim = rows
xrowsums <- apply(x, 1, function(x) sum(x))
xcolsums <- apply(x, 2, function(x) sum(x))
```

Plotting

Matlab

- Multiple method calls
- Need to `hold all` / `hold on` / `hold off`
- `hold all`, `plot`, `xlabel`, `ylabel`, ...
- No named arguments, plotting parameters with name-value pairs

R

- Single method call for single series into the plot
- `plot`, `lines`, `points`, ...
- Named arguments (e.g. `col='blue'`)