CT5102: Programming for Data Analytics

Week 5:
R Programming Structures

https://github.com/JimDuggan/CT5102

Dr. Jim Duggan,
Information Technology,
School of Engineering & Informatics

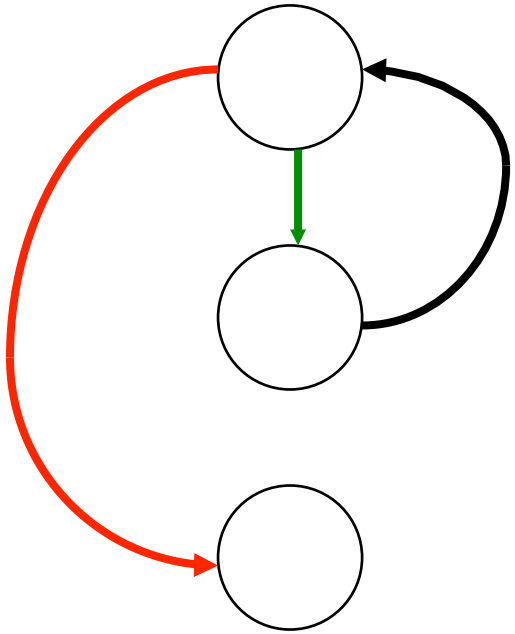
Overview

- R is a block-structured language, where blocks are delineated by {}
- Statements separated by newline characters, or with semicolon
- Variables are not declared (similar to JavaScript)

- Control Statements
- Arithmetic and Boolean Operators
- Writing “upstairs”
- Replacement functions

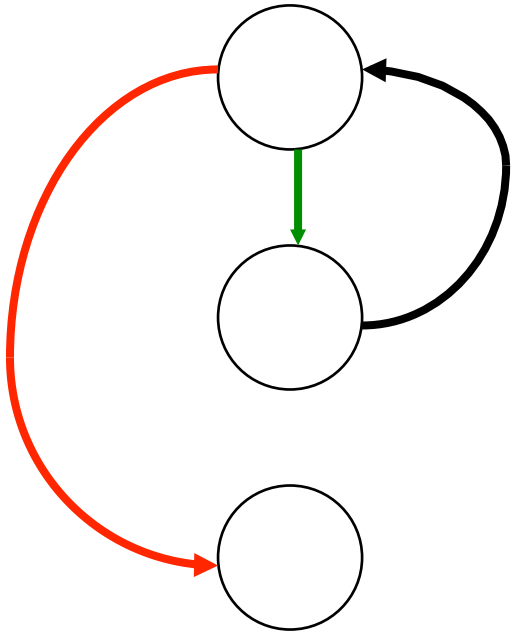
Loops - for

```
> x<-c(5,34,89)
>
> for(n in x){
+   print(n^2)
+ }
[1] 25
[1] 1156
[1] 7921
```



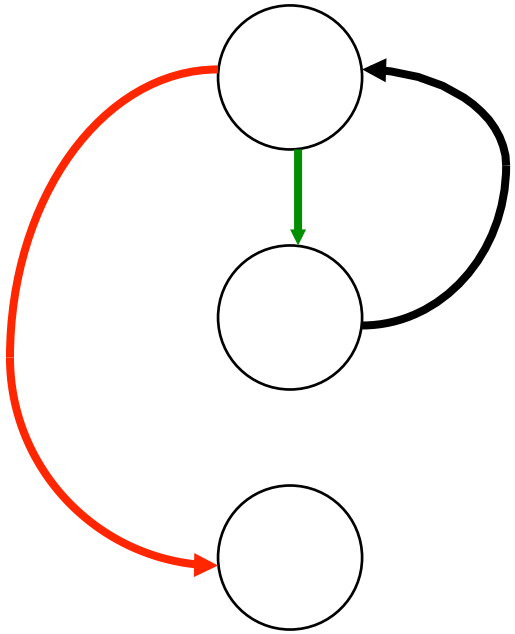
Loops - for

```
> x<-c(5,34,89)
>
> for(n in 1:length(x)){
+   print(x[n]^2)
+ }
[1] 25
[1] 1156
[1] 7921
```



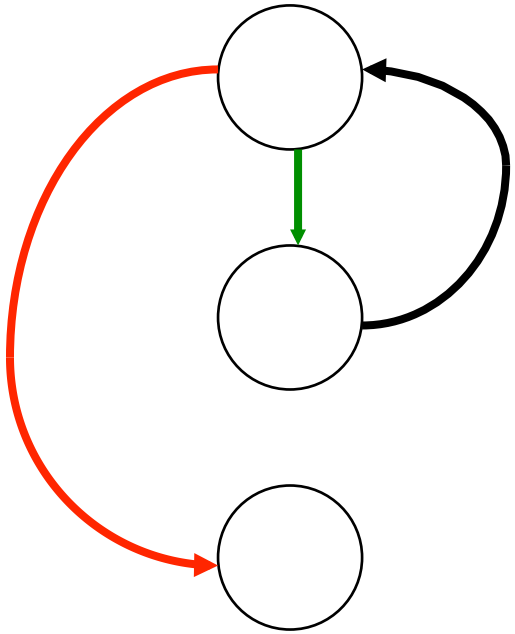
Loops - while

```
> i<-1
> while(i<=length(x)){
+   print(x[i]^2)
+   i<-i+1
+ }
[1] 25
[1] 1156
[1] 7921
```



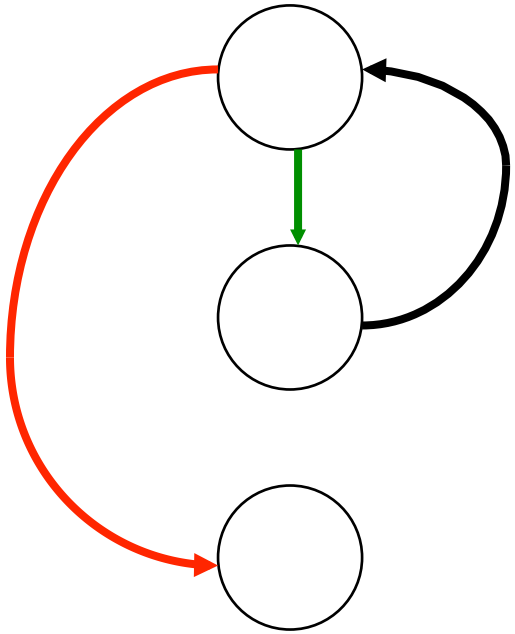
Loops - while

```
> i<-1
> while(TRUE){
+   print(x[i]^2)
+   i<-i+1
+   if(i>length(x)) break
+ }
[1] 25
[1] 1156
[1] 7921
```



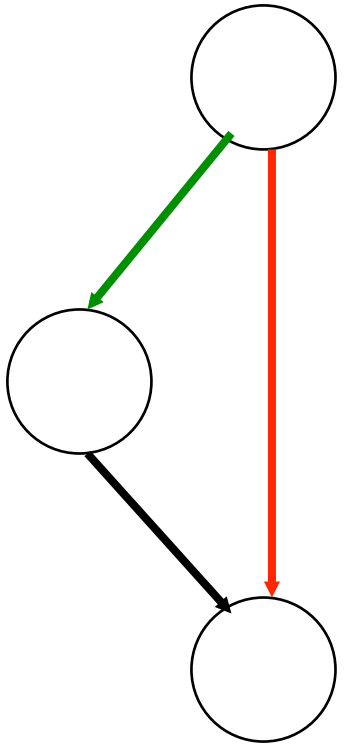
Loops – repeat (no condition)

```
> i<-1
> repeat {
+    print(x[i]^2)
+    i<-i+1
+    if(i>length(x)) break
+ }
[1] 25
[1] 1156
[1] 7921
```



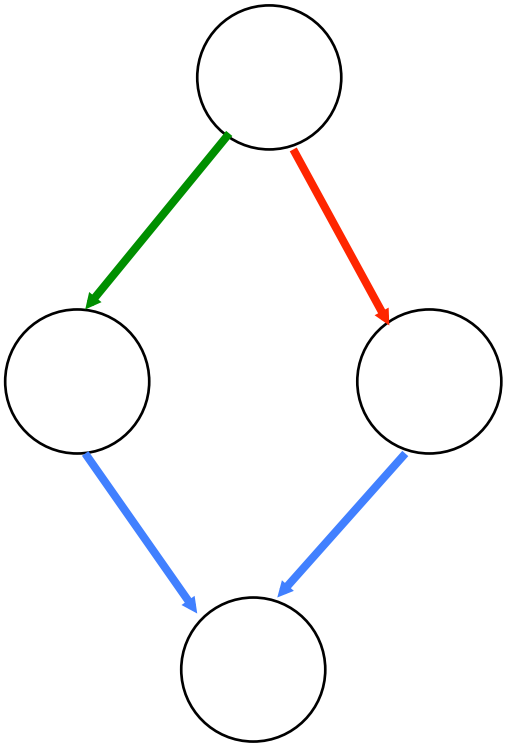
if

```
> x<-10
>
> if(x %% 2 == 0){
+    print("Even number...")
+ }
[1] "Even number..."
```



if else

```
> x<-11
> if(x %% 2 == 0){
+   print("Even number...")
+ } else
+   {
+   print("Odd number...")
+ }
[1] "Odd number..."
```



It is important to note that else must be in the same line as the closing braces of the if statements.

http://www.programiz.com/r-programming/if-else-statement

if else if

```
> x<-0
> if(x<0){
+   print("Negative number")
+ } else if (x > 0){
+   print("Positive number")
+ } else {
+   print("Zero!")
+ }
[1] "Zero!"
```

Challenge 5.1

- Implement the following decision table in an R function called getCost(age, card)

Conditions	Rules				
< 5 years	✓	×	×	×	×
>= 5 and < 18	×	✓	×	×	×
>= 18 and < 55 with concession card	×	×	✓	×	×
>= 18 and < 55 no concession card	×	×	×	✓	×
>= 55	×	×	×	×	✓
Actions					
Free Admission	✓	×	×	×	×
$8.00	×	✓	✓	×	×
$12.00	×	×	×	✓	×
$6.00	×	×	×	×	✓

http://hsc.csu.edu.au/ipt/project_work/3287/design_tools.htm

Solution

```
cost<-function(age, card=F){
  if(age < 5){
    return (0)
    } else if(age < 18){
       return (8)
    } else if(age < 55 && card){
      return (8)
    } else if (age < 55 && !card){
      return (12)
    } else{
      return (6)
    }
}
```

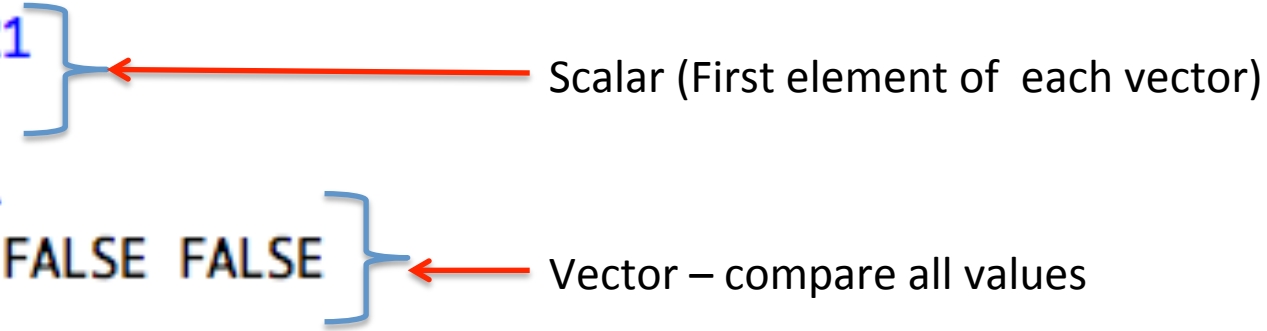
```
> cost(1)
[1] 0
> cost(6)
[1] 8
> cost(18)
[1] 12
> cost(1)
[1] 0
> cost(6)
[1] 8
> cost(17)
[1] 8
> cost(24)
[1] 12
> cost(24,T)
[1] 8
> cost(56)
[1] 6
```

List of Basic Operators

Operation	Description	Operation	Description
x+y	Addition	**x-y**	Subtraction
x * y	Multiplication	**x /y**	Division
x^y	Exponentiation	**x && y**	Modular arithmetic
x%/%y	Integer division	**x == y**	Test for equality
x <= y	Test of LT or equals	**x >= y**	Test of GT or equals
x && y	AND for scalars	**x || y**	OR for scalars
x & y	AND for vectors	**x | y**	OR for vectors
!x	Boolean negation		

Difference between && and &

```
> ages<-c(19,20,31,34)
> ages
[1] 19 20 31 34
> lt21<-ages<21
> lt21
[1]  TRUE  TRUE FALSE FALSE
> student<-c(T,F,T,F)
> student
[1]  TRUE FALSE  TRUE FALSE
> student && lt21
[1] TRUE
> student & lt21
[1]  TRUE FALSE FALSE FALSE
```



Challenge 5.2

- Write a function that takes a list of vectors and ANDs all the values
- For example:
  - v1<- c(TRUE, TRUE, FALSE)
  - v2<-c(FALSE, TRUE, TRUE)
  - v3<-c(TRUE,TRUE, FALSE)
- Should return
  - (FALSE, TRUE, FALSE
- Hints:
  - the function prod() gets the product of a vector, and TRUE & FALSE are also represented by 1 and 0
  - Also, a list can be converted to a data frame via the data.frame(l) function
  - The function t() will transpose a data frame

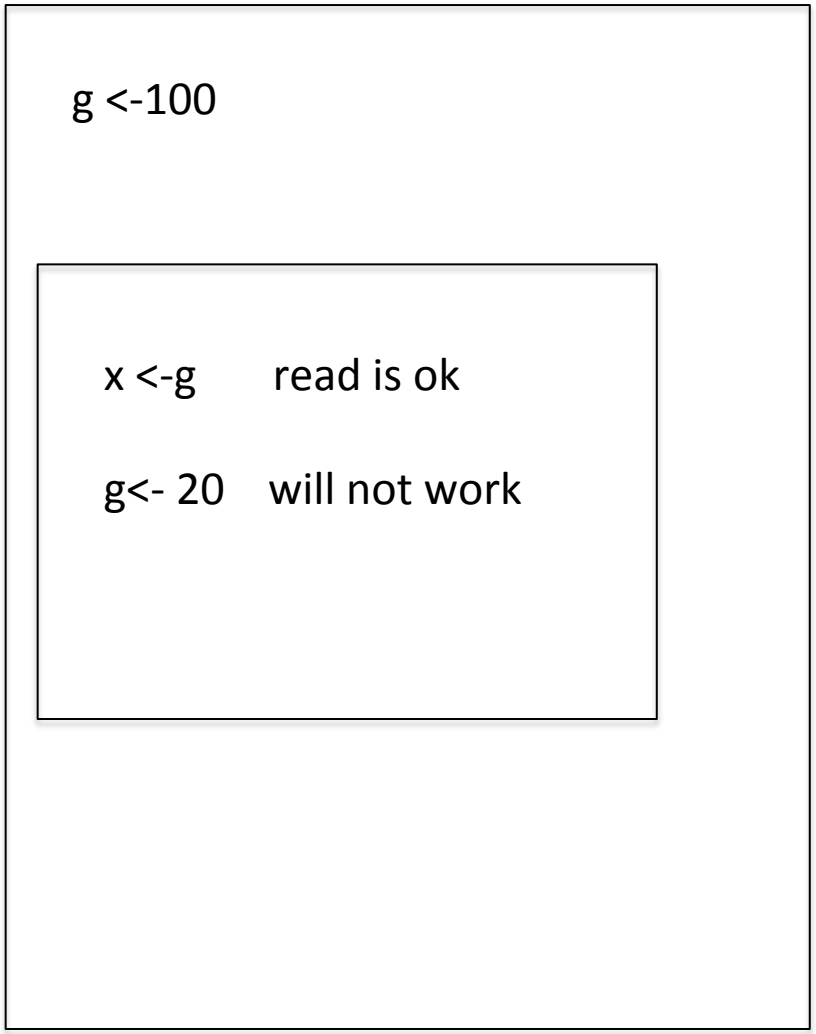
Writing Upstairs

- Code that exists at a certain level of the environment has at least read access to all the variables the level above it
- However, direct write access to variables at higher levels via the standard <- operator is not possible

```
g1<-100

f1<-function(){
  print(g1)
  g1<-20
}

f1()
print(g1)
```

```
> f1()
[1] 100
> print(g1)
[1] 100
```

General Idea



- Solutions
  - Superassignment operator <<-
  - assign() function

Example 1
Change a global value

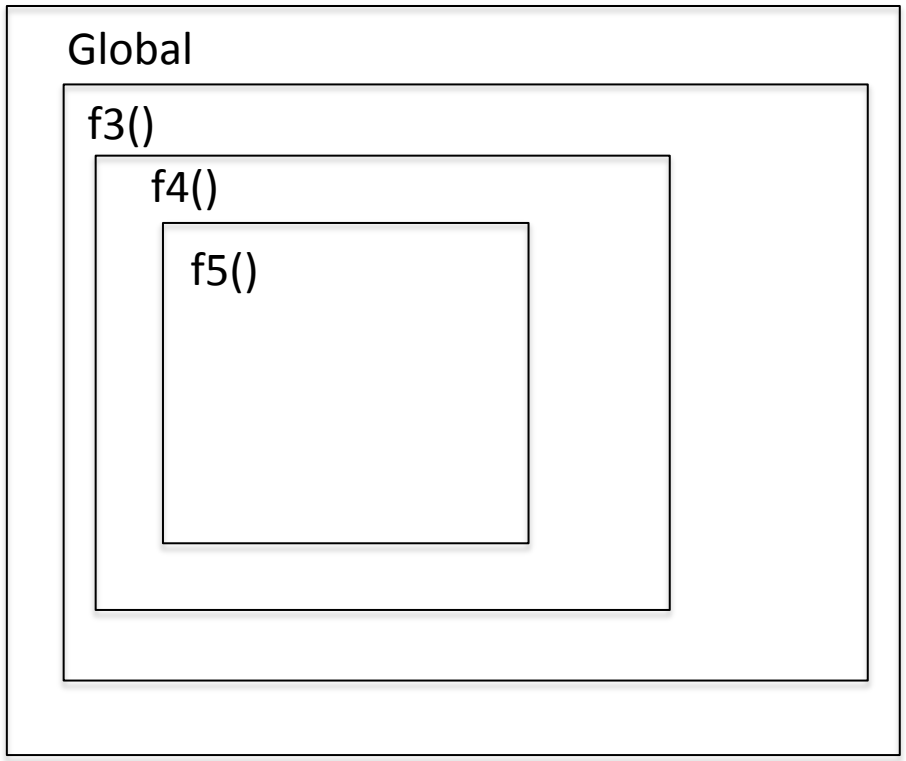
- Superassignment operator changes the global value
- Typically used to write top level variable
- However:
  - The operator will search up the environment hierarchy, stopping at the first level the name is encountered
  - If no name is found, the variable is assigned at the global level.

```
g2<-100

f2<-function(){
  print(g2)
  g2<<-20
}

f2()
print(g2)
```

```
> f2()
[1] 100
> print(g2)
[1] 20
```

Challenge 5.3

- Which variables are visible at the global level?




```
f3<-function(){
  f4<-function(){
    g3<<-40
    g5<-10
    f5<-function(){
      g4<<-20
      g5<<-45
    }
    f5()
    print(g5)
  }

  f4()
  print(g3)
}
```

assign() function

Assign a Value to a Name

Description

Assign a value to a name in an environment.

Usage

```
assign(x, value, pos = -1, envir = as.environment(pos),
       inherits = FALSE, immediate = TRUE)
```

Arguments

`x`	a variable name, given as a character string. No coercion is done, and the first element of a character vector of length greater than one will be used, with a warning.
`value`	a value to be assigned to `x`.

```
> f7<-function(){
+   assign("g9",10,pos=.GlobalEnv)
+ }
> g9
Error: object 'g9' not found
> f7()
> g9
[1] 10
```

Use of global variables

- In R, a global variable includes any variable located higher in the environment hierarchy than the level of the given code of interest
- Can be useful to change data in functions rather than returning lists etc
- Commonly used in R
  - R Library functions
  - Threaded code and GPU code (provide main avenue of communication between parallel actors)

Replacement Functions

- Consider the following example
- A function call with a parameter changes the parameter
- How does it work?

```
> v1<-c(1,2,3)
> names(v1)<-c("One","Two","Three")
> v1
  One   Two Three
    1     2     3
```

What is a replacement function?

- Any assignment statement in which the left hand side is not just an identifier (meaning a variable name) is considered a replacement function (Matloff 2009)
- The replacement function has one more argument than the original function

g(u) <- v

R will be try to execute as
u<-“g<-”(u,value=v)

g<- needs to have been defined.

Example

Replacement function definition

```
'mypaste<-' <- function(x,value){
  x<-paste(x,value,sep='+')
}

> v1<-c("Str1","Str2")
> mypaste(v1)<-c("X","Y")
> v1
[1] "Str1-X" "Str2-Y"
```

Challenge 5.4

- Write a replacement function that sets the second value in a vector to a target value

```
> x<-1:10
> x
 [1]  1  2  3  4  5  6  7  8  9 10
> second(x)<-100
> x
 [1]   1 100   3   4   5   6   7   8   9  10
```