FIAP

MBA+

ARTIFICIAL INTELLIGENCE & MACHINE LEARNING

FIAP

MBA+

PROGRAMANDO IA COM R

Prof. Elthon Manhas de Freitas
elthon@usp.br

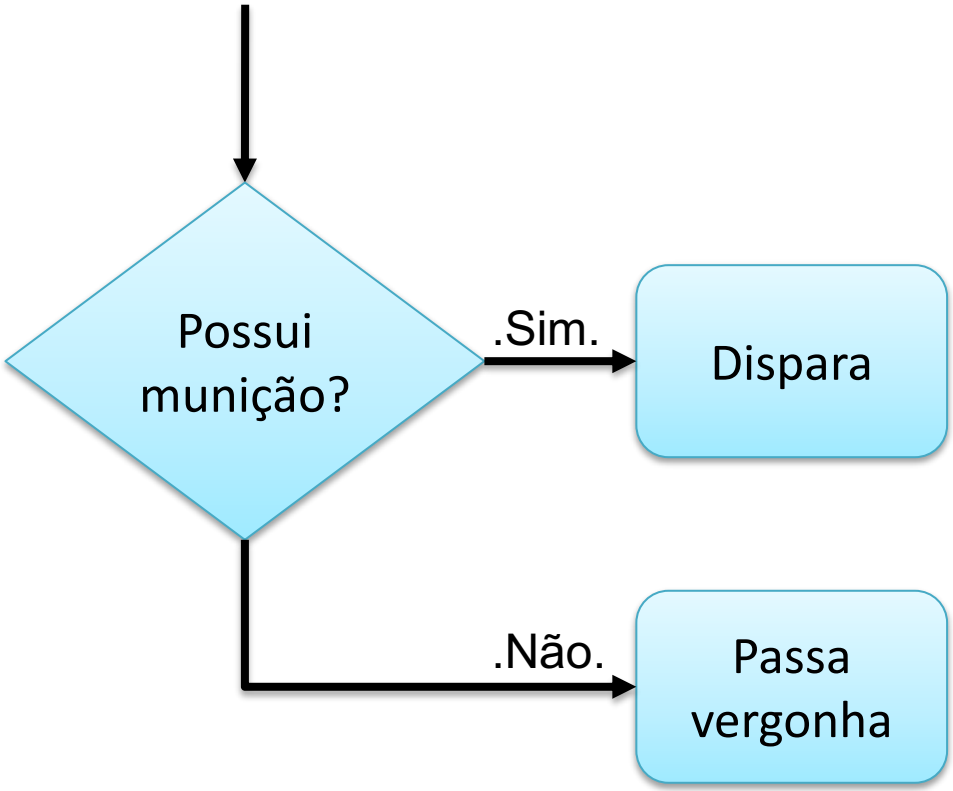
2018

Revisão da última aula

- O que vimos na aula passada?

Controle do Script – Basic Scripting

Condicionais



Condicionais

FIAP

- IF

```
if(condição){
    //Instruções
}
```

- IF / ELSE

```
if(condição) {
    //Instruções
} else {
    //Instruções
}
```

- IF / ELSE IF / ELSE

```
if (condição 1){
    //Comandos
} else if (condição 2){
    //Comandos
} else {
    //Comandos
}
```

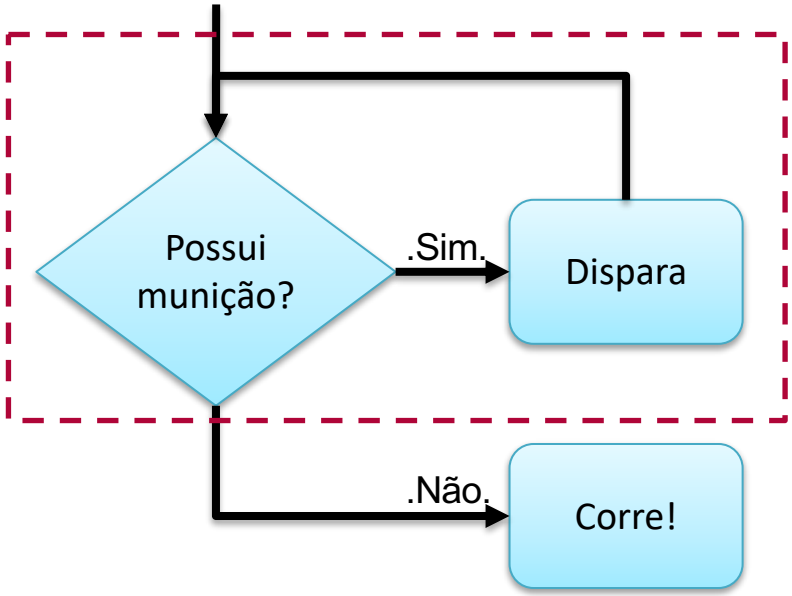
Condicionais – Ternários

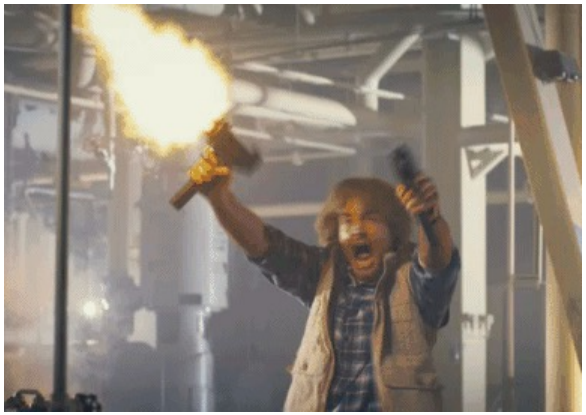
```
variável <- ifelse(condição, valor V, valor F);

if (condição){
    variável = (valor V);
} else {
    variável = (valor F);
}
```

Loop – Laços

- Laços são usados em programação quando precisamos que um mesmo trecho de código aconteça diversas vezes.





Loop - FOR

- Muito útil quando se sabe previamente a quantidade de “iterações” que o laço deve executar.

- Exemplo de `for`

```
for (i in vetor) {
    //Comandos
}
```

Tipos de vetores? E Matrizes? E tabelas?

Também podemos usar seqüencias

- seq
- seq_along
- etc.

Loop - WHILE

FIAP

- A instrução while indica que tudo o que estiver dentro do laço, será executado enquanto a condição do laço for verdadeira.

```
while(condição){
    //Comandos
}
```

- Perceba que não há declaração de variável no while, nem parte de atualização / incremento.

Loop – WHILE

- Mesmo código, escrito com while

```
strength = 1000.0;
for (i in 1:12){
    strength = strength * 1.01;
}
print( strength )
```

```
strength = 1000.0;
i = 1;
while(i <= 12){
    strength = strength * 1.01;
    i = i + 1
}
print( strength )
```

Loop – BREAK / CONTINUE

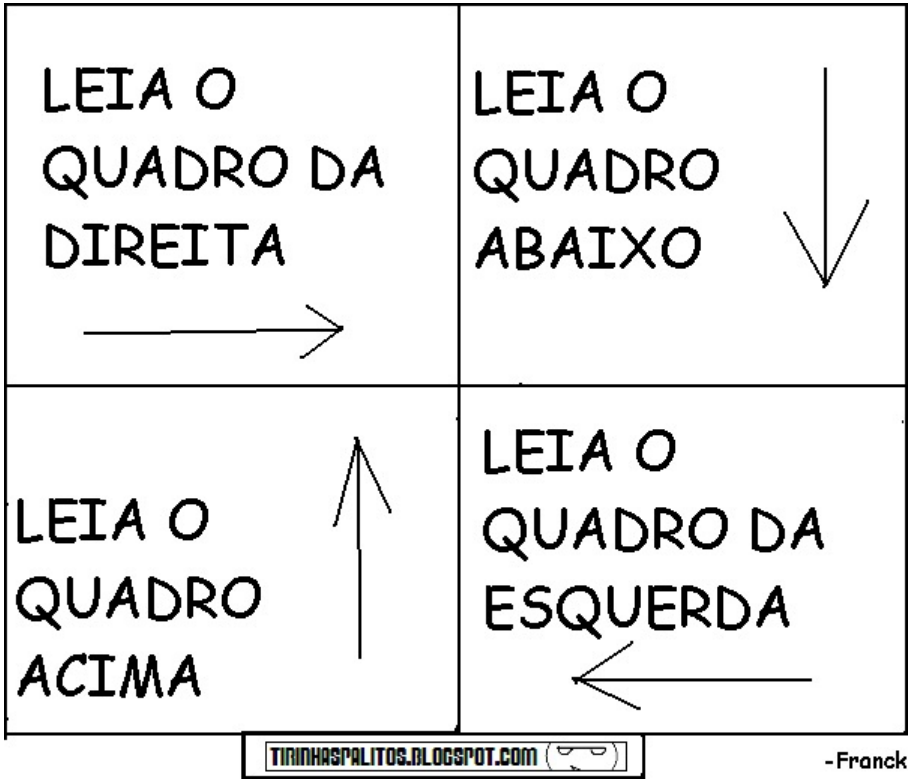
FIAP

- Utilizados para alterar o fluxo do loop

- `break`
  - Utilizado para “quebrar o laço”, ou seja, sair do loop a qualquer custo.

- `next`
  - Utilizado para ir para a próxima iteração do loop, sem executar os próximos comandos dentro do loop.

Loop infinito



Loop infinito

FIAP

Loop – REPEAT

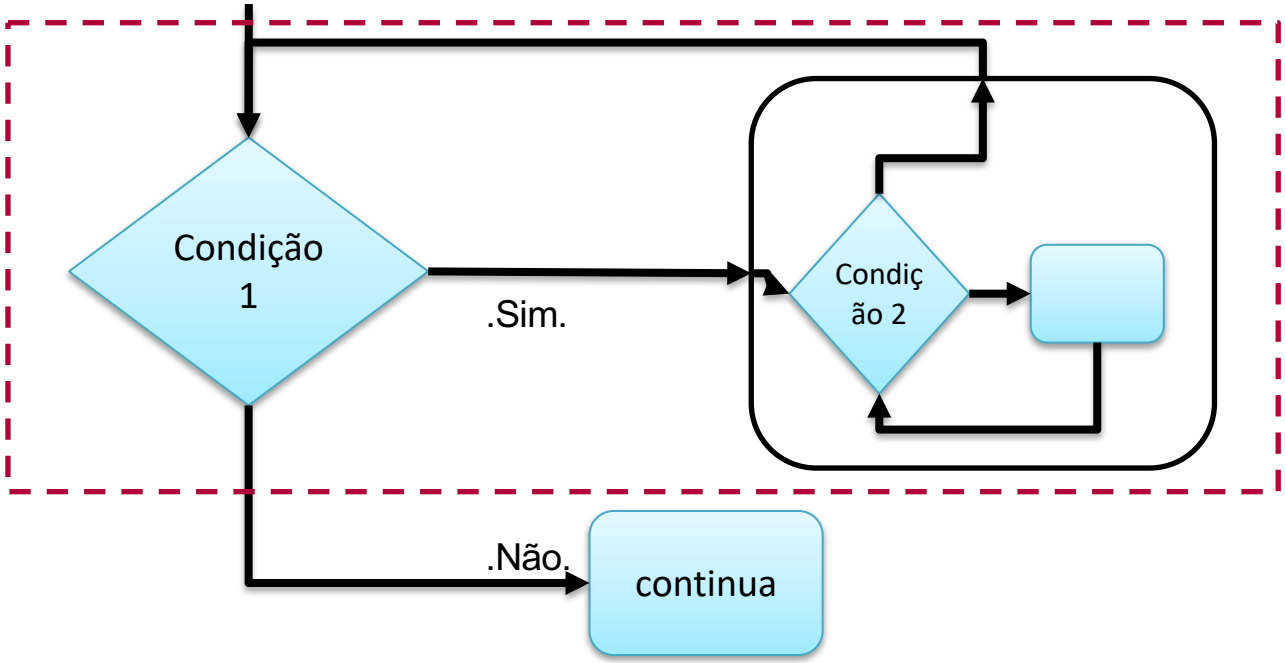
FIAP

- Comando usado para causar um loop infinito
  - O laço só para com um `break`
    - (ou com `return`, no caso de uma função, que veremos em seguida)

```
x0 <- 1
tolerancia <- 1e-8

repeat {
    x1 <- computeEstimate(x0)        #Função fictícia
    if(abs(x1 - x0) < tolerancia) {  ## Suficiente?
        break
    } else {
        x0 <- x1
    }
}
```

Loops encadeados – nested loops



Tarefa – Loop Acumulado

FIAP

- Criar script que varre o dataset `AirPassengers`
  - Observe que este dataset não é matrix, table ou data.frame, mas um TimeSeries

- Crie um vetor que contenha o valor acumulado de passageiros ao longo do tempo.

Mês (unidade)	Passageiros	Acumulado
1	5	5
2	4	9
3	10	19
…	…	…
n	12	$\sum_{i=1}^{n} passageiros[i]$

Métodos: Funções e Sub-rotinas

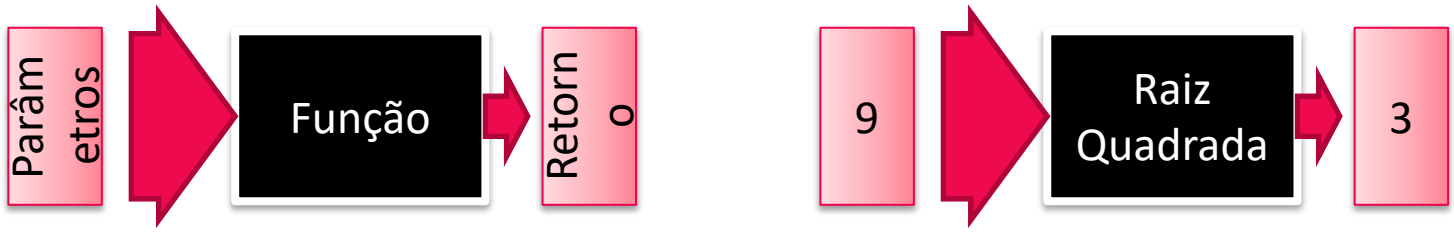
Métodos – Sub Rotina

FIAP

- Trata-se de um conjunto de instruções “empacotadas” para serem reutilizadas (sub-algoritmo);
- Podem ser vistas como um conjunto de procedimentos agrupados para um determinado fim;
- Podem receber parâmetros;
- Interessante enxergá-las sob o conceito “atômico”
  - Que não pode ser cortado, indivisível.

Métodos – Funções

- Similares às sub-rotinas, porém retornam valores.
  - Obs.: Em R, as sub-rotinas são funções que não tem retorno



O retorno de uma função pode ser dado:

- Pela instrução “return(valor)”
- Pelo último comando executado na função!

Sintaxe de uma função

FIAP

- Para declarar a função:

```
obj_name <- function(<parâmetros>){
    instruções
    return ( valor)
}
```

- Para executar a função:

```
obj_name(<parâmetros>)
```

Funções Recursivas (overview)

- São funções que dentro delas, chamam a si mesmas;
- Propensas a Loop infinito;

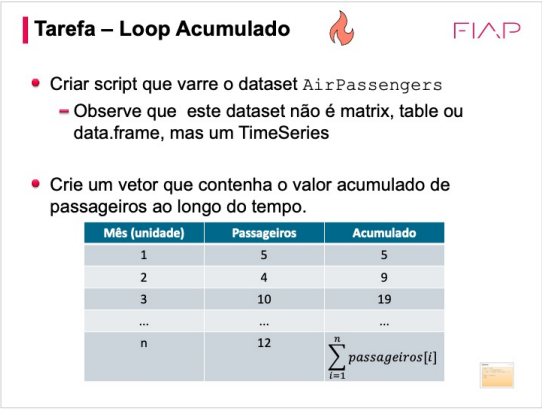
Exercício Tarefa

FIAP

- Criar função Fatorial que retorna o fatorial de um número inteiro.
  - O fatorial de um número n é n * n-1 * n-2 ... até n = 1
  - Exemplo 4! (lê-se 4 fatorial) é dado por:
    - 4 * 3 * 2 * 1
- Fazer uma função que recebe um vetor numérico e retorna seus valores invertidos.
  - Exemplo:
    - Entrada: 44, 67 , 5, 47, 8, 1, 79, 128
    - Saída: 128, 79, 1, 8, 47, 5, 67, 44

Exercícios

- Fazer uma função que dá a mesma resposta do Exercício “Loop Acumulado”



- Passar como parâmetro a Série `AirPassengers`

- Passar como parâmetro a coluna `mpg` da tabela `mtcars`
  - Perceberam que pode ser acessada por `mtcars$mpg` ?

Para raciocinar ...

- O que é a variável i? Uma função ou um número?
- O que é esperado da “Execução 1”?
- E da “Execução 2”?

```
h <- function() {
  x <- 10
  function() {
    x
  }
}
i <- h()

#Execução 1:
i()
x <- 20
#Execução 2:
i()
```

Tópicos avançados de programação em R

Aviso!
A seção a seguir trata de tópicos avançados (como o nome diz).

É aconselhável o uso de equipamentos de Proteção Individual para seguir nesta seção.

Funções Built-in

- Trata-se do pacote nativo de funções do R
  - é a base de tudo, o core
  - Funções de
    - conversão, de verificação, testes de valores, print, trigonometria, matemática básica, sequenciais

- Pode ser encontrado nos pacotes básicos do R:
  - "`base::`", "`stats::`", "`utils::`", "`graphics::`", etc.
  - São compilados em C++ ou Fortran
  - Alta performance

Entendendo o escopo

- Os escopos, no R são similares ao das demais linguagens, entretanto são organizados em “Environments”

- Sua estrutura é praticamente a estrutura de uma lista.

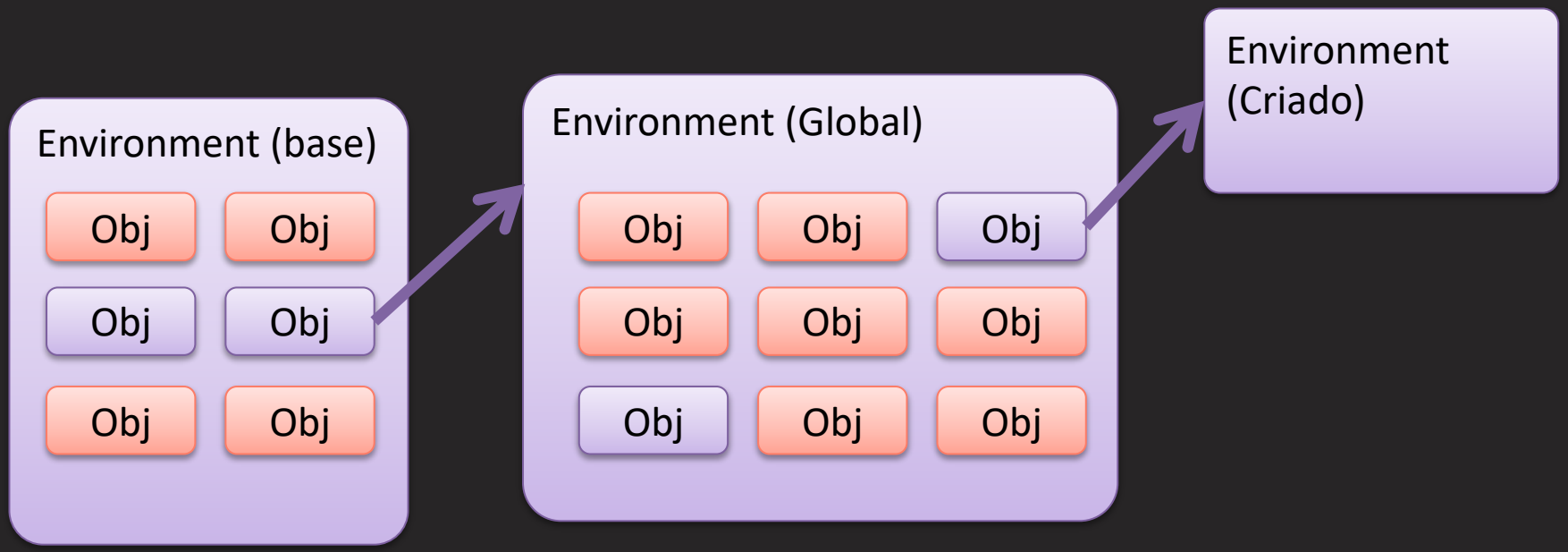
- Todas as as variáveis são armazenadas em algum “Environment”
  - a raiz de tudo é o “Base Environment”, pois o cada environment também é uma variável. Não é aconselhável alterar nada no “Base Environment”,
  - O “Global Environment” é o nosso ponto de partida, seria nossa área de trabalho principal

Environments

- Environments notáveis

- `.GlobalEnv`
- `globalenv()`

- `emptyenv()`
- `baseenv()`



Environments

- Atribuição de variáveis entre Environments
  - Façam a seguinte execução e avaliem o resultado

```
e1 = new.env()
assign("var1", 1, envir = e1 )
get("var1", envir = e1)
exists("var1", envir = e1)
rm("var1", envir = e1)
exists("var1", envir = e1)
```

  - Onde está o environment e1?

Environment em funções

FIAP

- As funções em R sempre passam parâmetro por valor, nunca por referência.

- Cada execução de uma função cria um novo environment ligado à execução.

- Para compartilhar resultados entre funções é necessário compartilhar o environment

Exercício opcional

- Criar uma função que:
  - Obtém o environment atual
  - Obtém o Global Environment
  - Imprima o environment atual
  - Imprima o Global Environment

- Executar esta função 5 vezes
  - O endereço do Environment foi o mesmo em todas as execuções?

Environment em funções

FIAP

- Como falamos, o GlobalEnvironment é usado como área de trabalho dos programas R.

- O que fazem os seguintes operadores de atribuição?

  - <<-
  - ->>

(explicação em lousa)

Lock de Variáveis – Atividade

FIAP

- O R não possui constantes, mas é possível "travar" variáveis, impedindo que seus valores sejam alterados.

```
a = 3
lockBinding('a', env = globalenv())
a = 4
rm(a, envir = globalenv())

a = 5
lockBinding('a', env = globalenv())
bindingIsLocked('a', env = globalenv())

a = 6
unlockBinding('a', env = globalenv())
bindingIsLocked('a', env = globalenv())
a = 7
```

Lock de Environments – Atividade

- Similar ao LOCK de variáveis é o LOCK de ambientes (envs). Este lock impede que variáveis sejam inseridas ou removidas, mas NÃO impede que os valores destas variáveis sejam alterados

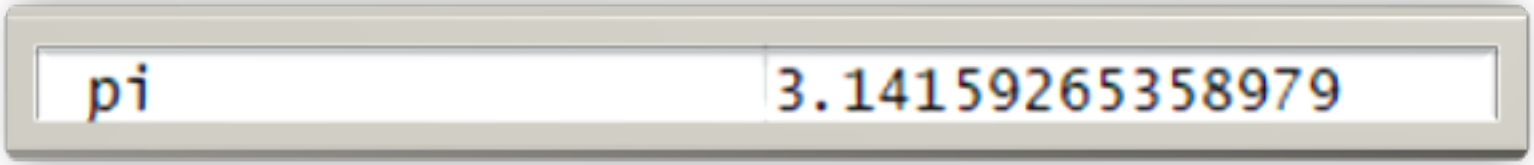
Perigo: Não há unlock de environment!!!

- `lockEnvironment`
- `environmentIsLocked`

Lock de Environments – Desafio

- Alterar o valor da variável pi

- Esta variável está localizada no “base” environment.



Perigo: Não salvar nada ao realizar este desafio!!!

(A não ser que seja no computador da FIAP)

Debugando o seu código

- A forma mais simples de debugar o seu código é através da instrução debug
  - debug(função) – a função passa a ser monitorada
  - undebug(função) – a função para de ser monitorada
  - debugonce() – a função será monitorada apenas uma vez

```
f <- function(){
  for (i in 1:10) {
    print(i)
  }
}

debugonce(f)
```

Exercício opcional

Praticar o Debug

- Debugar as funções criadas na aula:
  - Loop acumulado
  - Avaliação do Environment
- O que acontece ao se tentar debugar uma função Built-In?

Exercícios individuais

- Aprenda R no R
- Portfólio individual

FIAP

MBA+