# Função de Irene

## Função ontogenizar ()

Quando podemos dizer que um comportamento é geneticamente determinado? Existem comportamentos puramente inatos ou puramente aprendidos? Biólogos, etólogos e estudantes do comportamento em geral, sabemos que não há resposta clara para essas perguntas. Contudo, determinar o momento da aparição de um comportamento qualquer (e.g.. brincadeira, alimentação independente da mãe, agressividade, migração de grupo, etc.) ao longo da ontogenia pode nos ajudar à obter respostas para estas questões. É por isso que, pese ao alto custo e esforço requerido, existem cada vez mais pesquisas longitudinais. Ainda, a democratização e barateamento das filmadoras digitais faz com que seja cada vez mais comum que os pesquisadores interessados no comportamento, filmem esse para uma análise posterior. Assim, estão sendo geradas enormes quantidades de dados em formato de vídeo digital que contém valiosa informação, e que poderiam, inclusive, ser reaproveitados para outras pesquisas desde que estejam corretamente classificados. Portanto, a função “ontogenizar” ira organizar os dados de vídeo introduzidos pelo usuário em pastas classificadas temporalmente na escala escolhida por ele (dias, semanas, meses, ou anos) e fornecer como resultado, ademais da pasta organizada, um data.frame contendo toda essa informação.

---

## Arquivos para download

Arquivos Exemplo (formato rar)

HELP de Ontogenizar (pdf)

HELP de Ontogenizar (txt)

Script da função ontogenizar

Script Exemplos

## Código da função ontogenizar

```
##############################          FUNÇÃO ONTOGENIZAR
#####################################
################################## Irene Delval/ irenedelval@usp.br
#####################################

ontogenizar= function (in_fol, out_fol, db, unit, period, subject)

{
```

```
#######################################
#     1. MENSAGENS DE ERRO            # (caso existam argumentos ausentes
ou incorretos)
#######################################

  if(missing(in_fol)) #se não há argumento para pasta de entrada, ou está
incorreto
  {
    stop("\n\n ATENÇÃO! Diretorio de origem (in_fol) faltante \n") #a função
para, e devolve esta mensagem
  }
  if(missing(out_fol)) #se não há argumento para pasta de saída, ou está
incorreto
  {
    stop("\n\n ATENÇÃO! Diretorio destino (out_fol) faltante \n")  #a função
para, e devolve esta mensagem
  }
  if(missing(db)) #se não há argumento para a data de origem ou nascimento
  {
    stop("\n\n ATENÇÃO! Argumento db faltante \n")  #a função para, e
devolve esta mensagem
  }

  if (missing(subject))
  {
    cat( paste ("\n\n ATENÇÃO! Argumento subject por default = Ind_X \n\n"))
    subject= "Ind_X"
  }
  if(missing(unit)) #se não há argumento para unidade temporal, ou está
incorreto
  {
    stop("\n\n ATENÇÃO! Argumento unit faltante\n") #a função para, e
devolve esta mensagem
  }
  if(is.element(unit, c("day","week", "month", "year"))== F)
    {
    stop("\n\n ATENÇÃO! Argumento unit tem que ser day, week, month ou
year\n\n") #a função para, e devolve esta mensagem
  }
  if(missing(period)) #se não há argumento para periodo, ou está incorreto
  {
    stop("\n\n ATENÇÃO! Argumento period faltante\n") #a função para, e
devolve esta mensagem
  }
  if(floor(period)!= period) #se o argumento period não é um número entero
  {
    stop("\n\n ATENÇÃO! Argumento period deve ser um valor entero\n") #a
função para, e devolve esta mensagem
  }
```

```
#####################################################
#     2. LEITURA DE ARQUIVOS E CLASSIFICAÇÃO       #
#####################################################

db= as.Date(db, format="%Y-%m-%d") #coerção ao formato Date, do argumento
data de nascimento do individuo - usuário coloca
arquivos= list.files (in_fol) #cria um objeto com o nome de todos arquivos
lista= file.mtime (list.files ( in_fol, full.names = TRUE, recursive =
TRUE)) #objeto com a data e horario de cada video
lista.data = as.Date(lista, format="%Y-%m-%d") #objeto somente com a data,
sem horario, pois não interessa

## Cálculo da IDADE (id) nas diferentes unidades:

####DIAS####
id.dia = difftime(lista.data, db, units = "days")
#calcula a idade em dias, para incluir no df de saída

####SEMANAS####
id.sem= ceiling (difftime (lista.data, db, units = "weeks") + rep
(0.000001,length(lista.data)))
#calculo da idade em semanas, arredondando para cima (ceiling) para unificar
o criterio (e.g. dia do nascimento = semana 1 e eliminar decimais). Antes,
sumamos 0.000001 a toda a lista para que a arredondagem (ceiling) funcione
também com valor zero (o dia do nascimento)

####MESES####
#mais complicado porque difftime não funciona (só serve em "days" e
"weeks"). Vamos ter que criar um "loop" para calcular repetidamente a
longitud da diferença na sequencia existente entre a data de nascimento e a
data do vídeo

id.mes= rep (NA, from=1, to= length(lista)) #criamos um vetor vazio onde
guardar as diferenças
for(j in 1:length(lista)) #criamos o loop para calcular as diferenças entre
as sequencias
{
  calc.mes= length ( seq (from= db, to= lista.data[j], by="months"))
#pedimos que meça o tamanho da seqüência desde a data de origem (db) para as
datas dos arquivos [j], em meses
  id.mes[j]= calc.mes
}

####ANOS####
id.ano= rep (NA, from=1, to= length(lista)) #criamos um vetor vazio onde
guardar as diferenças
for(k in 1:length(lista)) #criamos o loop para calcular as diferenças entre
as sequencias
{
  calc.ano= length ( seq (from= db, to= lista.data[k], by="year")) #pedimos
que meça o tamanho da seqüência desde a data de origem (db) para as datas
```

```
dos arquivos [k], em anos
  id.ano[k]= calc.ano
}

##########################################
#     3. CRIANDO DATAFRAME DE OUTPUT       #
##########################################

nome= rep (subject, length(lista)) #coluna com o nome de individuo
nascimento= rep (db, length(lista)) #coluna com a data de nascimento

setwd(out_fol) #para que o arquivo output apareça na pasta destino

output = data.frame (nome, nascimento, arquivos, lista.data, id.dia, id.sem,
id.mes, id.ano) #por fim criamos o data frame com toda a informação
relevante
output = output [order(output$id.dia), ] #para obter o dataframe ordenado
cronologicamente


write.table (output, "output.txt", eol= "\n", sep="\t", col.names= T,
row.names=F, quote= F) #escreve o df na pasta de destino


##################################################
#     4.COPIANDO ARQUIVOS NA PASTA DESTINO       #
##################################################

# Os arquivos que irão ser copiados na pasta destino dependerão das escolha
do usuário nos argumentos unit e period

####DIAS####
#Não vejo muito interesse na classificação por dias, mas vou deixar como
opção porque talvez faça sentido em outras pesquisas longitudinais

if (unit== "day") #se o usuário escolhe organizar a pasta por dia
{
  f.interes.dia = output [output$id.dia==period,3] #cria as "files de
interesse" a serem copiadas, do dia preferido
  # terceira coluna[3] é a coluna com nome dos arquivos
  f.interes.dia = as.character (f.interes.dia)
  # coerção ao formato carater para evitar problemas de leitura
  paths.dia = file.path (in_fol, f.interes.dia)
  # cria caminho para cada arquivo... analogo a funçao "paste"
  destino.dia = file.path (out_fol, paste(nome[1], period, unit))
  # damos nome à pasta que será criada, pelo nome do individuo, periodo e
unidade
  class(destino.dia) # para conferir a classe do objeto
  dir.create(path = destino.dia) #cria a pasta de destino, por dias
  for(d in 1:length(paths.dia))
```

```
  {
    file.copy(from = paths.dia[d], to = destino.sem, recursive = F,
copy.mode = T, copy.date = T) #caminho dos arquivos a serem copiados,
mantendo data do arquivo e permisos originais
  }
}

####SEMANAS####

if (unit== "week") #se o usuário escolhe organizar a pasta por semana
{
  f.interes.sem = output [output$id.sem==period,3] #cria as "files de
interesse" a serem copiadas, da semana escolhida
 # terceira coluna[,3] é a coluna com nome dos arquivos

  f.interes.sem = as.character (f.interes.sem)
  # coerção ao formato carater por problemas de leitura

  paths.sem = file.path (in_fol, f.interes.sem)
  # cria caminho para cada arquivo... analogo a funçao "paste"

  destino.sem = file.path (out_fol, paste(nome[1], period, unit))
  # damos nome à pasta que será criada, pelo nome do individuo, periodo e
unidade
  class(destino.sem) # para conferir a classe do objeto
  dir.create(path = destino.sem) #cria a pasta de destino

  for(a in 1:length(paths.sem))
    {
    file.copy(from = paths.sem[a], to = destino.sem, recursive = F,
copy.mode = T, copy.date = T) # caminho dos arquivos a serem copiados,
mantendo data do arquivo e permisos originais
    }
}

####MESES####
if (unit== "month") #se o usuário escolhe organizar a pasta por mês
{
  f.interes.mes = output [output$id.mes==period,3] #cria as "files de
interesse" a serem copiadas, do mês escolhido
  # terceira coluna[,3] é a coluna com nome dos arquivos
  f.interes.mes = as.character (f.interes.mes)
  # coerção ao formato carater para evitar problemas de leitura
  paths.mes = file.path (in_fol, f.interes.mes)
  # cria caminho para cada arquivo... analogo a funçao "paste"
  destino.mes = file.path (out_fol, paste(nome[1], period, unit))
  # damos nome à pasta que será criada, pelo nome do individuo, periodo e
unidade
  class(destino.mes) # para conferir a classe do objeto
  dir.create(path = destino.mes) #cria a pasta de destino
  for(b in 1:length(paths.mes))
```

```
  {
    file.copy(from = paths.mes[b], to = destino.mes, recursive = F,
copy.mode = T, copy.date = T) #caminho dos arquivos a serem copiados,
mantendo data do arquivo e permisos originais
  }
}

####ANOS####
if (unit== "year") #se o usuário escolhe organizar a pasta por ano
{
  f.interes.ano = output [output$id.ano==period,3] #cria as "files de
interesse" a serem copiadas, do ano escolhido
  # terceira coluna[,3] é a coluna com nome dos arquivos
  f.interes.ano = as.character (f.interes.ano)
  # coerção ao formato carater para evitar problemas de leitura
  paths.ano = file.path (in_fol, f.interes.ano)
  # cria caminho para cada arquivo... analogo a funçao "paste"
  destino.ano = file.path (out_fol, paste(nome[1], period, unit))
  # damos nome à pasta que será criada, pelo nome do individuo, periodo e
unidade
  class(destino.ano) # para conferir a classe do objeto
  dir.create(path = destino.ano)  #cria a pasta de destino
  for(c in 1:length(paths.ano))
  {
    file.copy(from = paths.ano[c], to = destino.ano, recursive = F,
copy.mode = T, copy.date = T) #caminho dos arquivos a serem copiados,
mantendo data do arquivo e permisos originais
  }
}

}

################################### FIM
########################################
```

---

# Página de Ajuda da Função ontogenizar

```
_____________________________________________________________________________

ontogenizar              package: nenhum              R Documentation
_____________________________________________________________________________

**Página de ajuda da função ontogenizar**

Função para a organização e classificação de informação coletada e
armazenada em diversos formatos de vídeo digital.
```

```
**Description:**
A função “ontogenizar” é uma função simples para a organizar informação de vídeo. Produz um dataframe (guardado em um output.txt) com os arquivos organizados cronologicamente, em dias, semanas, meses e anos. Também cria uma pasta, no diretório indicado pelo usuário, com os arquivos organizados conforme o recorte temporal pré-definido por ele nos argumentos.

**Usage:**

ontogenizar (in_fol="C:/Users/Fulano/Downloads/Macaco_fulano", out_fol= "C:/Users/Fulano/Downloads/Output", db= "2013-04-17", unit= "month", period= 3, subject= “Ind_X")

**Arguments:**

in_fol       endereço da pasta contendo os arquivos a ser classificados

out_fol      endereço onde o output.txt e a pasta resultante serão criados

db       data de nascimento (date_of_birth); vetor data. Data pela qual queira se-estabelecer à ordem  dos arquivos, em formato YYYY-MM-DD.

unit        recorte temporal desejado na “ontogenização” dos dados (“day”, “week”, “month”, “year”)

period      fracção de “unit” pela que os arquivos serão classificados; vetor numérico inteiro.

subject     nome do sujeito; vetor caráter. Se “subject” é NULL, utiliza-se por default subject= "Ind_X"



**Details:**

A função foi criada para ser usada em sistema operacional Windows.
As pastas (in_fol e out_fol) devem ser caminhos válidos, com o formato de barra de Windows (/).
Se o usuário quer classificar seus dados por vários períodos, basta rodar a função várias vezes modificando os argumentos “unit” e “period”.
Se o usuário especifica valores de “unit” e “period” inexistentes na pasta de origem (in_fol) a função criará uma pasta nomeada pelo recorte temporal, porém vazia.


**Value:**

comp1 : retorna um data.frame com as seguintes informações: nome, nascimento, arquivos, lista.data, id.dia, id.sem, id.mes, id.ano. Esse df é copiado na pasta de saída (out_fol), determinada pelo usuário.
```

```
comp2 : gera uma pasta, classificada conforme a unidade (“day”, “week”, “month” ou “year”)escolhida pelo usuário.

**Warning:**

Se a pasta criada por “unit” e “period” (e.g. 8 month) escolhidos pelo usuário já existe, o sistema devolve uma mensagem automática (e.g. ‘Sofia 8 month’ already exists).

Aviso de argumento faltante para in_fol, out_fol, db, period e unit. Todos eles são argumentos obrigatórios.

Caso argumento “unit” esteja mal escrito, o usuário será avisado.

Caso o argumento “period” não seja um número entero, o usuário será avisado.

Caso o argumento “subject” esteja faltando a função usará por default Ind_X. Uma mensagem de aviso informará ao usuário.


**Author(s):**

Irene Delval
irenedelval@usp.br


**See Also:**

difftime, Date, POSIXlt, POSIXct, seq


**Examples:**

#1# Baixar todos os arquivos de exemplo embaixo. Salve todos eles em um mesmo diretório. Utilize o caminho para este diretório no argumento “in_fol” dos exemplos
e.g. ontogenizar(in_fol="C:/Users/Fulano/Downloads/Macaco_fulano"...)

#2# Escolha o local onde os arquivos serão "ontogenizados" e coloque o caminho a esse diretório no argumento "out_fol".

#3# Escolha os valores dos argumentos obrigatórios db, unit, period. Por se tratar de um exemplo, também pode escolher o argumento "subject" que da sua preferência, ou deixar o default. (Normalmente o usuario saberá a identidade do indivíduo a “ontogenizar”).

#4# Execute a função e veja a pasta de saída e o data.frame criado (output.txt).

#5# Repita os passos 1-4 as vezes que achar necessário, mudando os valores
```

```
em unit e period.

##exemplo em meses:
ontogenizar (in_fol="C:/Users/Fulano/Downloads/Macaco_fulano", out_fol=
"C:/Users/Fulano/Downloads/Output", db= "2013-04-17", unit= "month", period=
3, subject= "Irene")

##exemplo em semanas:
ontogenizar (in_fol="C:/Users/Fulano/Downloads/Macaco_fulano", out_fol=
"C:/Users/Fulano/Downloads/Output", db= "2013-04-17", unit= "week", period=
1, subject= "Irene")

##exemplo em ano:
ontogenizar (in_fol="C:/Users/Fulano/Downloads/Macaco_fulano", out_fol=
"C:/Users/Fulano/Downloads/Output", db= "2013-04-17", unit= "ano", period=
1, subject= "Irene")

## exemplo Ind default:
ontogenizar (in_fol="C:/Users/Fulano/Downloads/Macaco_fulano", out_fol=
"C:/Users/Fulano/Downloads/Output", db= "2013-04-17", unit= "ano", period=
2)

################################## FIM DA DOCUMENTAÇÃO DE ontogenizar()
##################################
```