

# Funções

Arquivos para teste: [all\\_genes.csv](#) [genes\\_of\\_interest.csv](#)

## match\_genes

### Código

```
#match_genes finds the "control" genes (gene with different biological
function than the gene of interest) closest located to an optimal distance
for a given list of genes of interest.
match_genes = function(all, interest, opt = 10^4, graph = F, list=F) {
  #Initializes the data.frame that is going to be returned.
  match = data.frame(matrix(ncol=3, nrow=0))
  #The function is gonna use the median to calculate gene distances.
  all$md = apply(all[,c(3,4)], 1, function (x) median(x))
  #Same as above, but now for interest data.frame.
  interest$md = apply(interest[,c(3,4)], 1, function (x) median(x))
  #Here, we start the first loop; we are going to define the control genes
  one by one.
  for (i in 1:nrow(interest)) {
    #Subsets the all data.frame containing only genes in the same
  chromosome as the interest gene.
    data = subset(all, all[,2] == interest[i, 2])
    #Removes genes that were a match already.
    data = subset(data, (!(data[,1] %in% match[,2])))
    #Gets only the genes that have different function from the gene of
  interest. You can use this column as a filter and the algorithm will work
  the same.
    data = subset(data, (data[,5] != interest[i,5]))
    #Just in case the data.frame isn't ordered by position in the
  chromosome. This isn't necessary, but makes it easier to interpret the code
  when you manually inspect it.
    data = data[order(data$md),]
    #Calculates the relative distance between the gene of interest median
  position and every gene median position available.
    data$dists = abs(interest[i,6] - data[,6])
    #Finds the index of the gene that is the closest located to the
  optimal distance.
    j = which.min(abs(data$dists-opt))
    #Tests if no gene is found and creates a data.frame with the gene of
  interest ID, the ID of the gene closest to the optimal distance and the
  distance itself (these last two may be NAs if no gene is found).
    if (!length(j)) {
      m = data.frame(interest[i,1], cbind(NA, NA))
    } else {
      m = data.frame(interest[i,1], data[j,][c(1,7)])
    }
  }
```

```
#Binds the row created before to the existing match data.frame.
match = rbind(match, m)
}
#Assign names to the columns of match.
colnames(match) = c("interest_ID", "match_ID", "distance")
#Checks if the parameter graph is True.
if (graph) {
  #Starts a graphics device X.
  x11()
  #Plot the distance between genes in a boxplot.
  boxplot(match$distance)
  #Puts a title to the plot.
  title("Distances between each pair gene of interest/control")
  #Traces a line in the optimal distance.
  abline(h=opt, col="red")
  #Closes the graphic device.
  dev.off()
}
#Checks what is going to be returned depending on the parameter list
if(list) {
  #Returns a list with the match data.frame and the median match
distance. It is used in the shuffle_match function.
  return (list(match, median(match$distance)))
  #If the argument list is false, then only the data.frame match is
going to be returned.
} else {
  #Returns the match data.frame.
  return(match)
}
}
```

[match\\_genes.r](#)

## Help

match_genes	package:misc	R Documentation
-------------	--------------	-----------------

### Description:

The function finds one possible set of control genes, defined as genes with different biological function (you can code this however you like), to a given set of genes of interest. The control genes will be defined based on an optimal distance to the genes of interest.

**Usage:**

```
match_genes(all, interest, ...)
```

**Arguments:**

`all`: data.frame containing all genes that could possibly be a control gene.  
`interest`: data.frame containing the genes of interest.  
`opt`: the optimal distance (i.e., the distance from the genes of interest to which the control genes must be close to). Default: `opt = 10^4`  
`graph`: boolean indicating whether a graph with the distances must be produced or not. Default: `graph = F`  
`list`: boolean indicating which kind of outcome should be returned: just the data.frame with the matches (`list=F`) or a list with the data.frame and the median match distance (`list=T`). Default: `list = F`

**Details:**

Both the `all` data.frame and the `interest` data.frame must contain five columns (gene ID, chromosome, start position, end position and the function). Note that the last column may be treated as a filter, which can be biological function, gene ontology, etc.

**Value:**

If `list=T`:

A list of two elements is returned. The first element is the match data.frame with each interest gene matched to a control gene and the relative distance between these two. The second is the median of the relative distances, which is used by another function.

If `list=F`:

The match data.frame is returned, which comprises of each interest gene matched to a control gene and the relative distance between these two.

**Author(s):**

Murillo Fernando Rodrigues

**References:**

None.

**Examples:**

```
#Running match_genes function with both needed data.frames and default parameters. The data.frames used here are available for testing.
```

```
all = read.csv ("all_genes.csv", header=T)
interest = read.csv ("genes_of_interest.csv", header=T)
match_genes(all, interest, list = T)
```

[help\\_match\\_genes.txt](#)

---

## shuffle\_match

### Código

```
#shuffle_match tests whether changing the order of the rows in the interest
data.frame alters the outcome and returns the most optimal result. This
function was built because the algorithm of match_genes isn't very smart.
shuffle_match = function (all, interest, opt = 10^4, n = 100) {
  #Creates an empty list.
  results = list()
  #Loop to run match_genes several times.
  for (i in 1:n) {
    #Shuffle the interest data.frame
    interest = interest[sample(nrow(interest)),]
    #Appends the results from match_genes to a pre-existing list called
    results
    results = append(results, list(match_genes(all, interest, opt,
graph=F, T)))
  }
  #Gets only the one result which got the median distance closest to the
  optimal.
  results = results[[which.min(abs(sapply(results, function(x) { x[[2]]
})-opt))]][[1]]
  #Return results
  return (results)
}
```

[shuffle\\_match.r](#)

### Help

shuffle\_match

package:misc

R Documentation

#### Description:

This function is an improved implementation of match\_genes. It uses brute force to find the most optimal result. shuffle\_match runs match\_genes n times and returns the result from match\_genes which

had  
the median of the distances between control and gene of interest  
closest  
to the chosen optimal distance.

#### Usage:

```
shuffle_match(all, interest, ...)
```

#### Arguments:

all: data.frame containing all genes that could possibly be a control gene.  
interest: data.frame containing the genes of interest.  
opt: the optimal distance (i.e., the distance from the genes of interest to which the control genes must be close to). Default:  $opt = 10^4$   
n: number of times to run match\_genes. Default:  $n = 100$ .

#### Details:

You should vary n depending on how many genes of interest you have.

#### Value:

The match data.frame is returned, which comprises of each interest gene matched to a control gene and the relative distance between these two.  
Note that this is the most optimal result the function could find.

#### Author(s):

Murillo Fernando Rodrigues

#### References:

None.


#### Examples:

```
#match_genes function with both needed data.frames and default parameters.  
m = match_genes(all, interest, list = T)  
#Check the outcome of match_genes  
m  
  
#Running shuffle_match to check whether the order in the interest data.frame  
alters the outcome. Typically, doing the shuffle_match is the best way to  
go, because match_genes algorithm isn't very smart.  
s = shuffle_match(all, interest)
```

```
#Check the outcome of shuffle_match  
s  
#Computes the median of the distances between control and interest gene.  
md_s = median(r$distance)  
  
#Which result is the closest to the optimal?  
(m[[2]]-10^4)  
(md_s-10^4)
```

[help\\_shuffle\\_match.txt](#)

From:  
<http://ecor.ib.usp.br/> - **ecoR**

Permanent link:  
[http://ecor.ib.usp.br/doku.php?id=05\\_curso\\_antigo:r2016:alunos:trabalho\\_final:murillo.rodriques:funcoes](http://ecor.ib.usp.br/doku.php?id=05_curso_antigo:r2016:alunos:trabalho_final:murillo.rodriques:funcoes) 

Last update: **2020/08/12 06:04**