Função lista_compras

```
lista_compras = function(cafe, refeicao, jaca, multiplos = 0, listasobras =
NULL, listacafe, listarefeicao, listajaca){ # cria a função
  ### VERIFICANDO PARÂMETROS DECLARADOS
  # Numéricos
  if(is.numeric(cafe) == FALSE){ # verifica se cafe é numérico
    stop("cafe não é numérico") # se não for, dá um erro avisando
  }
  if(is.numeric(refeicao) == FALSE){ # verifica se refeicao é numérico
    stop("refeicao não é numérico") # se não, for dá um erro avisando
  }
  if(is.numeric(jaca) == FALSE){ # verifica se jaca é numérico
    stop("jaca não é numérico") # se não for, dá um erro avisando
  }
  if(is.numeric(multiplos) == FALSE){ # verifica se multiplos é numérico
    stop("multiplos não é numérico") # se não for, dá um erro avisando
  }
  # listasobras
  if(is.null(listasobras) == FALSE){ # verifica se foi oferecido sobras. Se
foi, faz verificações adicionais
    if(is.data.frame(listasobras) == FALSE){ # verifica se listasobras é um
data.frame
      stop("listasobras deve ser declarado como data.frame") # se não for,
dá um erro avisando
    }
    if(ncol(listasobras) != 3){ # verifica se o número de colunas está
correto
      stop("número de colunas incorreto em listasobras") # se não estiver,
dá um erro avisando
    }
    if(is.numeric(listasobras[,2]) == FALSE){ # verifica se a segunda coluna
é numérica
      stop("verifique a segunda coluna de listasobras") # se não for, dá um
erro avisando
    }
  }
  # listas
  if(is.data.frame(listacafe) == FALSE | is.data.frame(listarefeicao) ==
FALSE | is.data.frame(listajaca) == FALSE){ # verifica se as listas são
data.frames
    stop("as listas de receitas devem ser declaradas como data.frame") # se
não forem, dá um erro avisando
  }
  if(ncol(listacafe) != 5 | ncol(listarefeicao) != 5 | ncol(listajaca) !=
5){ # verifica se o número de colunas das listas está correto
    stop("número de colunas incorreto nas listas") # se não estiverem, dá um
erro avisando
  }
  if(is.numeric(listacafe[,3]) == FALSE | is.numeric(listarefeicao[,3]) ==
```

```
FALSE | is.numeric(listajaca[,3]) == FALSE){ # verifica se a terceira coluna
de cada lista é numérica
    stop("verifique a segunda coluna de sobras") # se não for, dá um erro
avisando
  }
  ### ESCREVENDO VARIÁVEIS PARA OUTPUT
  lista_receitas = data.frame(matrix(ncol=2, nrow=0)) # vai acumular o nome
e o preparo das receitas escolhidas
  colnames(lista_receitas) = c("RECEITA", "PREPARO") # nomes das colunas
  lista_supermercado = data.frame(matrix(ncol=3, nrow=0)) # vai ser usado
como lista de compras
  colnames(lista_supermercado) = c("INGREDIENTE", "QUANTIDADE", "UNIDADE DE
MEDIDA") # nomes das colunas
  lista_nao_usado = data.frame(matrix(ncol=3, nrow=0)) # para mostrar as
sobras que não foi possível usar
  colnames(lista_nao_usado) = c("INGREDIENTE", "QUANTIDADE", "UNIDADE DE
MEDIDA") # nomes das colunas
  ### VERIFICANDO SOBRAS
  # Estrutura de testes escolhida para verificar os ingredientes da lista de
sobras com quantidades maiores que zero:
  # Verificar se existe receitas com aquele ingrediente disponíveis em cafe,
refeicao, ou jaca conforme
  # determinado pelos parâmetros declarados
  # Para cada categoria de receita, identificar todas as receitas que contém
o ingrediente, e verificar se tem mais ou menos
  # receitas do que o requisitado para aquela categoria.
  # Se houverem mais receitas do que o requisitado, sorteia as receitas a
serem usadas.
  # Se foram solicitadas receitas suficientes, usar todas.
  # Se não houverem receitas suficientes para aquele ingrediente, incluir na
lista dos ingredientes não utilizados.
  # Remover a quantidade do ingrediente da lista e continuar o procedimento
para os demais ingredientes
  sobras = listasobras # atribuindo listasobras a uma nova variável para
poder mexer nos valores sem perder a informação do input
  for(i in 1:nrow(sobras)){ # iterar sobre cada ingrediente da sobra
    # CAFE
    if(sobras[i,2] > 0){ # ver se realmente existe item disponível
      if(cafe > 0){ # verifica se foram solicitadas receitas da categoria
cafe
        ingrediente_no_cafe = listacafe[listacafe$INGREDIENTE ==
sobras[i,1],] # procurando o ingrediente na lista de cafe
        if(nrow(ingrediente_no_cafe) > 0){ # ver se encontrou alguma receita
pra usar o ingrediente
          if(cafe < nrow(ingrediente_no_cafe)){ # verifica se existem mais
receitas do que o permitido para cafe
            sorteio = sample(1:nrow(ingrediente_no_cafe), cafe) # sorteia o
número de receitas necessárias
            lista_receitas = rbind(lista_receitas,
ingrediente_no_cafe[sorteio, c(1,5)]) # incluir receitas escolhidas na lista
```

```
            lista_supermercado = rbind(lista_supermercado,
listacafe[listacafe$RECEITA %in% ingrediente_no_cafe[sorteio, 1], c(2, 3,
4)]) # incluir todos ingredientes das receitas sorteadas na lista de compras
            cafe = 0 # zerar a quantidade de receitas da categoria cafe
disponíveis
            if(length(unique(ingrediente_no_cafe[sorteio,4])) == 1){ #
verifica se unidades de medida do ingrediente são iguais
              sobras[i,2] = sobras[i,2] -
sum(ingrediente_no_cafe[sorteio,3]) # tira o ingrediente das sobras
            }else{ # se unidades de medida do ingrediente não forem iguais
              stop("Verifique as unidades de medida da tabela cafe") # dá
erro
            }
          }else{ # se tiver mais receitas cafe declaradas do que receitas
com o ingrediente
            lista_receitas = rbind(lista_receitas, ingrediente_no_cafe[,
c(1,5)]) # incluir receitas na lista
            lista_supermercado = rbind(lista_supermercado,
listacafe[listacafe$RECEITA %in% ingrediente_no_cafe[, 1], c(2, 3, 4)]) #
incluir todos ingredientes das receitas na lista de compras
            cafe = cafe - nrow(ingrediente_no_cafe) # diminui o número de
receitas disponíveis em cafe
            if(length(unique(ingrediente_no_cafe[,4])) == 1){ # verifica se
unidades de medida do ingrediente são iguais
              sobras[i,2] = sobras[i,2] - sum(ingrediente_no_cafe[,3]) #
Tira o ingrediente usado das sobras
            }else{ # se unidades de medida do ingrediente não forem iguais
              stop("Verifique as unidades de medida da tabela cafe") # dá
erro
            }
          }
        }
      }
    }
    # REFEIÇÃO
    if(sobras[i,2] > 0){ # ver se realmente existe item disponível
      if(refeicao > 0){ # verifica se foram solicitadas receitas da
categoria refeicao
        ingrediente_na_refeicao = listarefeicao[listarefeicao$INGREDIENTE ==
sobras[i,1],] # procurando o ingrediente na lista de refeicao
        if(nrow(ingrediente_na_refeicao) > 0){ # ver se encontrou alguma
receita pra usar o ingrediente
          if(refeicao < nrow(ingrediente_na_refeicao)){ # verifica se
existem mais receitas do que o permitido para refeicao
            sorteio = sample(1:nrow(ingrediente_na_refeicao), refeicao) #
sorteia o número de receitas necessárias
            lista_receitas = rbind(lista_receitas,
ingrediente_na_refeicao[sorteio, c(1,5)]) # incluir receitas escolhidas na
lista
            lista_supermercado = rbind(lista_supermercado,
listarefeicao[listarefeicao$RECEITA %in% ingrediente_na_refeicao[sorteio,
```

```
1], c(2, 3, 4)]) # incluir todos ingredientes das receitas sorteadas na
lista de compras
            refeicao = 0 # zerar a quantidade de receitas da categoria
refeicao disponíveis
            if(length(unique(ingrediente_na_refeicao[sorteio,4])) == 1){ #
verifica se unidades de medida do ingrediente são iguais
              sobras[i,2] = sobras[i,2] -
sum(ingrediente_na_refeicao[sorteio,3]) # tira o ingrediente das sobras
            }else{ # se unidades de medida do ingrediente não forem iguais
              stop("Verifique as unidades de medida da tabela refeicao") #
dá erro
            }
          }else{ # se tiver mais receitas refeicao declaradas do que
receitas com o ingrediente
            lista_receitas = rbind(lista_receitas, ingrediente_na_refeicao[,
c(1,5)]) # incluir receitas na lista
            lista_supermercado = rbind(lista_supermercado,
listarefeicao[listarefeicao$RECEITA %in% ingrediente_na_refeicao[, 1], c(2,
3, 4)]) # incluir todos ingredientes das receitas na lista de compras
            refeicao = refeicao - nrow(ingrediente_na_refeicao) # diminui o
número de receitas disponíveis em cafe
            if(length(unique(ingrediente_na_refeicao[,4])) == 1){ # verifica
se unidades de medida do ingrediente são iguais
              sobras[i,2] = sobras[i,2] - sum(ingrediente_na_refeicao[,3]) #
tira a quantidade do ingrediente das sobras
            }else{ # se unidades de medida do ingrediente forem diferentes
              stop("Verifique as unidades de medida da tabela refeicao") #
dá erro
            }
          }
        }
      }
    }
    # PÉ NA JACA
    if(sobras[i,2] > 0){ # ver se realmente existe item disponível
      if(jaca > 0){ # verifica se foram solicitadas receitas da categoria
jaca
        ingrediente_na_jaca = listajaca[listajaca$INGREDIENTE ==
sobras[i,1],] # procurando o ingrediente na lista de jaca
        if(nrow(ingrediente_na_jaca) > 0){ # ver se encontrou alguma receita
pra usar o ingrediente
          if(jaca < nrow(ingrediente_na_jaca)){ # verifica se existem mais
receitas do que o permitido para jaca
            sorteio = sample(1:nrow(ingrediente_na_jaca), jaca) # sorteia o
número de receitas necessárias
            lista_receitas = rbind(lista_receitas,
ingrediente_na_jaca[sorteio, c(1,5)]) # incluir receitas escolhidas na lista
            lista_supermercado = rbind(lista_supermercado,
listajaca[listajaca$RECEITA %in% ingrediente_na_jaca[sorteio, 1], c(2, 3,
4)]) # incluir todos ingredientes das receitas sorteadas na lista de compras
```

```
            jaca = 0 # zerar a quantidade de receitas da categoria jaca
disponíveis
            if(length(unique(ingrediente_na_jaca[sorteio,4])) == 1){ #
verifica se unidades de medida do ingrediente são iguais
              sobras[i,2] = sobras[i,2] -
sum(ingrediente_na_jaca[sorteio,3]) # tira o ingrediente das sobras
            }else{ # se unidades de medida do ingrediente forem diferentes
              stop("Verifique as unidades de medida da tabela jaca") # dá
erro
            }
          }else{ # se tiver mais receitas jaca declaradas do que receitas
com o ingrediente
            lista_receitas = rbind(lista_receitas, ingrediente_na_jaca[,
c(1,5)]) # incluir receitas na lista
            lista_supermercado = rbind(lista_supermercado,
listajaca[listajaca$RECEITA %in% ingrediente_na_jaca[, 1], c(2, 3, 4)]) #
incluir todos ingredientes das receitas na lista de compras
            jaca = jaca - nrow(ingrediente_na_jaca) # diminui o número de
receitas disponíveis em cafe
            if(length(unique(ingrediente_na_jaca[,4])) == 1){ # verifica se
unidades de medida do ingrediente são iguais
              sobras[i,2] = sobras[i,2] - sum(ingrediente_na_jaca[,3]) #
tira o ingrediente das sobras
            }else{ # se unidades de medida do ingrediente forem diferentes
              stop("Verifique as unidades de medida da tabela jaca") # dá
erro
            }
          }
        }
      }
    }
    if(sobras[i,2] > 0){ # verificar se a quantidade do ingrediente ainda é
maior que zero para adicionar a lista de ingredientes não usados
      lista_nao_usado = rbind(lista_nao_usado, sobras[i,]) # incluir o
ingrediente, quantidade e unidade de medida na lista de não usados
    }
  }
  ### INCLUIR RECEITAS
  # cafe
  if(cafe > 0){ # ver se ainda precisa escolher mais receitas para cafe
    sorteio = sample(1:nrow(listacafe), cafe) # se sim, sortear o número
necessário
    lista_receitas = rbind(lista_receitas, listacafe[sorteio, c(1,5)]) #
incluir receitas escolhidas na lista
    lista_supermercado = rbind(lista_supermercado,
listacafe[listacafe$RECEITA %in% listacafe[sorteio, 1], c(2, 3, 4)]) #
incluir todos ingredientes das receitas sorteadas na lista de compras
  }
  # refeicao
  if(refeicao > 0){ # ver se ainda precisa escolher mais receitas para
refeicao
```

```
    sorteio = sample(1:nrow(listarefeicao), refeicao) # se sim, sortear o
número necessário
    lista_receitas = rbind(lista_receitas, listarefeicao[sorteio, c(1,5)]) #
incluir receitas escolhidas na lista
    lista_supermercado = rbind(lista_supermercado,
listarefeicao[listarefeicao$RECEITA %in% listarefeicao[sorteio, 1], c(2, 3,
4)]) # incluir todos ingredientes das receitas sorteadas na lista de compras
  }
  # jaca
  if(jaca > 0){ # ver se ainda precisa escolher mais receitas para jaca
    sorteio = sample(1:nrow(listajaca), jaca) # se sim, sortear o número
necessário
    lista_receitas = rbind(lista_receitas, listajaca[sorteio, c(1,5)]) #
incluir receitas escolhidas na lista
    lista_supermercado = rbind(lista_supermercado,
listajaca[listajaca$RECEITA %in% listajaca[sorteio, 1], c(2, 3, 4)]) #
incluir todos ingredientes das receitas sorteadas na lista de compras
  }
  ### MULTIPLICANDO AS QUANTIDADES DE UMA REFEIÇÃO PRINCIPAL PARA REFEIÇÃO
MÚLTIPLA
  if(multiplos > 1){ # se foi declarado o numero de pessoas para refeicao
multipla
    lista_receitas_refeicao = lista_receitas[lista_receitas$RECEITA %in%
listarefeicao$RECEITA,] # encontrar as receitas da categoria refeicao na
lista de receitas
    sorteio = sample(1:nrow(lista_receitas_refeicao), 1) # sortear uma
receita
    multiplicacao = listarefeicao[listarefeicao$RECEITA %in%
listarefeicao[sorteio, 1], c(2, 3, 4)] # cria dataframe com os ingredientes
a serem multiplicados
    multiplicacao[,2] = lapply(multiplicacao, function(x) x * (multiplos -
1)) # multiplicar os ingredientes vezes o número de multiplos -1 pois os
ingredientes já foram adicionados na lista de supermercado uma vez
    lista_supermercado = rbind(lista_supermercado, multiplicacao) # incluir
todos ingredientes multiplicados na lista de compras
    # pegar ingredientes desta refeição e somar multiplos-1 vezes cada
ingrediente
  }
  ### UNIFICANDO MESMOS INGREDIENTES DA LISTA DE SUPERMERCADO
  supermercado_exclusiva = unique(lista_supermercado$INGREDIENTE) # criar
lista com cada ingrediente sem repetição.
  lista_supermercado_unificada = data.frame(matrix(nrow =
length(supermercado_exclusiva), ncol = 3)) # criar nova dataframe para
output da lista de supermercado sem repetição de ingredientes
  colnames(lista_supermercado_unificada) = c("INGREDIENTE", "QUANTIDADE",
"UNIDADE DE MEDIDA") # nomear colunas
  lista_supermercado_unificada$INGREDIENTE = supermercado_exclusiva #
preencher a coluna de ingredientes
  for(i in 1:length(supermercado_exclusiva)){ # iterar sobre cada
ingrediente sem repetição
```

```
    quantidade = 0 # criar uma variável para acumular a quantidade do
ingrediente
    for(j in 1:nrow(lista_supermercado)){ # olhar para cada linha da lista
de supermercado com repetição
      if(lista_supermercado[j, 1] == supermercado_exclusiva[i]){ #
procurando o ingrediente em questão
        if(is.na(lista_supermercado_unificada[i, 3]) == TRUE){ # ver se já
tinha unidade de medida atribuída ao ingrediente
          lista_supermercado_unificada[i, 3] = lista_supermercado[j, 3] # se
não tinha unidade de medida, pegar
          quantidade = quantidade + lista_supermercado[j, 2] # e somar a
quantidade pra somatório final
        }else if(lista_supermercado_unificada[i, 3] == lista_supermercado[j,
3]){ # se ja tiver unidade de medida, ver se são iguais
          quantidade = quantidade + lista_supermercado[j, 2] # somar
quantidades caso unidades sejam iguais
        }else if(lista_supermercado_unificada[i, 3] != lista_supermercado[j,
3]){ # se unidades de medida forem diferentes, dar um erro
          stop("Verifique as unidades de medidas. Para um mesmo ingrediente
deve usar a mesma unidade de medida.") # emite o erro
        }
      }
    }
    lista_supermercado_unificada[i, 2] = quantidade # atribuir a somatória
de quantidades a lista final
  }
  ### SUBTRAIR DA LISTA DE SUPERMERCADO OS ITENS DA SOBRA
  if(is.null(listasobras) == FALSE){ # se foi declarada listasobras
    for(i in 1:nrow(listasobras)){ # iterar sobre cada item das sobras
      if(listasobras[i, 1] %in% lista_supermercado_unificada$INGREDIENTE){ #
se o item das sobras está presente na lista de compras
lista_supermercado_unificada[lista_supermercado_unificada$INGREDIENTE ==
listasobras[i, 1], 2] =
lista_supermercado_unificada[lista_supermercado_unificada$INGREDIENTE ==
listasobras[i, 1], 2] - listasobras[i, 2] # e subtrair a quantidade na sobra
da quantidade da lista de supermercado
if(lista_supermercado_unificada[lista_supermercado_unificada$INGREDIENTE ==
listasobras[i, 1], 2] <= 0){ # se essa subtração zerar o ingrediente
          lista_supermercado_unificada =
lista_supermercado_unificada[!(lista_supermercado_unificada$INGREDIENTE ==
listasobras[i, 1]),] # remover o ingrediente da lista
        }
      }
    }
  }
  ### OUTPUT
  if(nrow(lista_receitas) != 0){ # verifica se lista de receitas tem alguma
coisa dentro pra imprimir
    cat(paste("Receitas escolhidas e preparo (ou site que te ensina o
preparo)\n\n")) # output de receitas
    cat(for(i in 1:nrow(lista_receitas)){ # iterar sobre cada linha do
```

```
arquivo com as receitas usadas
      print(lista_receitas[i,])} # imprimindo cada linha
      , "\n\n", "==========", "\n\n") # fazer um separador bonitinho
  }
  if(nrow(lista_supermercado_unificada) != 0){ # verifica se lista de
supermercado tem coisa dentro para imprimir
    cat(paste("Lista de compras de supermercado\n\n")) # output da lista de
supermercado
    cat(for(j in 1:nrow(lista_supermercado_unificada)){ # iterando sobre
cada linha do arquivo com os itens a ser comprados
      print(lista_supermercado_unificada[j,])} # imprimindo cada linha
      , "\n\n", "==========", "\n\n") # fazendo um separador bonitinho
  }
  if(nrow(lista_nao_usado) != 0){ # verifica se tem itens não usados nas
sobras
    cat(paste("Sinto muito, esses ingredientes vão ficar sobrando na
dispensa mais um tempo.\n\n")) # output dos não usados na sobras
    cat(for(k in 1:nrow(lista_nao_usado)){ # iterando sobre cada linha dos
itens não usados
      print(lista_nao_usado[k,])} # imprimindo cada linha
    )}
}
```