```
universidades.sp <-  function (categoria.univ, cursos.pret, periodo,
aval.mec, mensalidade){

  library(readxl)
  Cursos_SP <- read_excel("cursos_sp.xlsx")
  #1 Lendo o banco de dados "cursos_sp.xlsx".
  Cursos_SP <- Cursos_SP[,1:4]
  #2 Para garantir que só sejam utilizadas as 4 primeiras colunas.
  Notas_MEC <- read_excel("notas_mec.xlsx")
  #3 Lendo o banco de dados "notas_mec.xlsx".
  Notas_MEC <- Notas_MEC[,1:3]
  #4 Para garantir que só sejam utilizadas as 3 primeiras colunas.
  Notas_MEC$Nota_MEC <- as.numeric(Notas_MEC$Nota_MEC)
  #5 Para garantir que a coluna Nota_MEC seja interpretada como números.
  #VERIFICANDO OS PARAMETROS DA FUNCAO#

  if(categoria.univ != "Pública" && categoria.univ != "Particular")
  #6 Verifica se "categoria.univ" foi inserido corretamente.
  {
    stop("A categoria da universidade deve ser pública ou particular")
    #7 Se não foi, interrompe a funcao e exibe uma mensagem para o usuario.
  }
  if(periodo != "Matutino" && periodo != "Vespertino" && periodo !=
"Integral" && periodo != "Noturno")
  #8 Verifica se "periodo" foi inserido corretamente.
  {
    stop("O período deve ser igual a matutino, vespertino, integral ou
noturno")
    #9 Se não foi, interrompe a funcao e exibe uma mensagem para o usuario.
  }
  if(aval.mec != round(aval.mec) || aval.mec <= 0 || aval.mec > 5)
  #10 Verifica se " aval.mec " e um numero inteiro maior do que zero e menor
ou igual a cinco.
  {
    stop ("aval.mec deve conter apenas um número inteiro maior do que 0 e
menor ou igual a 5 ")
    #11 Senao, interrompe a funcao e exibe mensagem para o usuario.
  }
  if(categoria.univ != "Pública" && (is.na(mensalidade) || mensalidade <=0))
  #12 Verifica se o preenchimento de "mensalidade" foi feito corretamente
para a categoria 'Particular'.
  {
    stop ("Para universidades particulares, é necessário o preenchimento da
mensalidade aproximadamente desejada ")
    #13 Senao, interrompe a funcao e exibe mensagem para o usuario.
  }
  if(categoria.univ != "Pública" &&  (mensalidade != round(mensalidade) ||
mensalidade <= 0))
  #14 Verifica se 'mensalidade' e um numero inteiro e maior que zero.
  {
    stop ("Mensalidade deve ser um número inteiro e >0 ")
```

```
    #15 Senao, interrompe a funcao e exibe mensagem para o usuario.
  }
  #FUNCAO BUSCA DE UNIVERSIDADES#
  df.uniao.bases <- merge(Cursos_SP,Notas_MEC, by = c("Nome_Universidade"))
  #16 Junta as bases de dados usando a coluna 'Nome_Universidade'
  #17 Buscar todas as linhas da base que atendam aos criterios acima,
separando a questão da mensalidade
  if(categoria.univ=="Pública")
  {
    linhas.selecionadas <- which(df.uniao.bases$Categoria==categoria.univ &
                                 df.uniao.bases$Nota_MEC==aval.mec &
df.uniao.bases$Curso_Pretendido==cursos.pret &
                                 df.uniao.bases$Periodo==periodo)
  }else{
    linhas.selecionadas <- which(df.uniao.bases$Categoria==categoria.univ &
                                 df.uniao.bases$Nota_MEC==aval.mec &
df.uniao.bases$Curso_Pretendido==cursos.pret &
                                 df.uniao.bases$Periodo==periodo &
                                 (abs(df.uniao.bases$Valor_Mensalidade-
mensalidade)<=1000))
                                 #18 Retorna todas as universidades que
tenham valor proximo da mensalidade desejada.
  }
  if(length(linhas.selecionadas)==0)
  #19 Verifica se a busca retornou vazio.
  {
    stop ("Não foi encontrada nenhuma universidade com os requisitos
escolhidos ")
    #20 Senao, interrompe a funcao e exibe mensagem para o usuario.
  }else{
    universidade <- df.uniao.bases[linhas.selecionadas,]
    #21 Filtra o dataframe somente com a linhas que seguem os criterios.
  }
  if(categoria.univ=="Particular"){
  #22 Caso a categoria seja particular, é possível criar um ranking da mais
barata para mais cara
    universidade <- universidade[order(universidade$Valor_Mensalidade),]
    #23 Ordena de acordo com menor mensalidade.
    ranking <- seq(1,nrow(universidade))
    #24 Cria sequencia de numeros para serem usados como ranking baseado na
mensalidade.
    universidade$ranking <- ranking
    #25 Adiciona ranking no dataframe
  }
  return(universidade)
}
```