```
pca_graphics=function(dados,var,grupos)#1 A minha função contêm os arquivos
de entrado dados, var e grupos.
```

{

```
## VERIFICANDO OS PARÂMETROS ##
if (!require(devtools))#2 Se não existir o pacote ggbibplot instalado,
{install.packages("devtools")}#instala o pacote. Caso já exista, a função
require carrega automaticamente.

if (!require(ggbiplot))#3 Se não existir o pacote ggbibplot instalado,
{install_github("vqv/ggbiplot")}#instala o pacote ggbiplot. Caso já exista,
a função require carrega automaticamente.

library(grid)#4 Carrega o pacote grid, referente ao pacote ggbiplot.

if (class(dados) != "data.frame") #5 Verifica a classe do objeto "dados".Se
dados não for da classe data.frame,
{stop ("dados só pode ser da classe dataframe")} # interrompe a funcao e
exibe mensagem para o usuario.

if (class(var) != "data.frame") #6 Verifica a classe do objeto "var".Se var
não for da classe data.frame,
{stop ("variaveis só pode ser da classe dataframe")} # interrompe a funcao e
exibe mensagem para o usuario.

if (is.numeric(as.matrix(var)) != "TRUE") #7 Verifica se "var" contem apenas
classes numéricas.Como "var" deve ser fator, é necessário transformar em
matrix antes de verificar. Se não for,
{stop ("var só pode conter valores numéricos")} # interrompe a funcao e
exibe mensagem para o usuario.

if (class(grupos) != "factor") #8 Verifica a classe do objeto "grupos". Se
grupo não for da classe fator,
{stop ("grupo só pode ser da classe fator")} #interrompe a funcao e exibe
mensagem para o usuario.

pca=prcomp(var,center=TRUE,scale.=TRUE) #9 Cria o pca a partir do objeto
"variaveis", centralizando as variáveis no ponto zero e escalonando todos os
valores.
pca_scores=data.frame(pca$x) #10 Armazena como um dataframe os valores dos
scores obtidos pelo pca para cada amostra.

### CONSTRUÇÃO DOS GRÁFICOS ###

m=combn(as.matrix(seq_along(pca_scores)),m=2)#11 Cria o objeto "m" que
armazena, em uma matriz, todas as possibilidades de combinação dois a dois
(eixos x e y) para a sequência de colunas contidas em pca_scores. A
sequência de colunas representa as dimensões.
x=c(NA,NA)#12 Cria vetor "x" de duas posições vazio (com NAs).
```

```
pdf("Pca_plots.pdf")#13 Cria um arquivo pdf chamado "PCA_plots.pdf" para
salvar as diferentes combinações de PCA no diretório do usuário. Apenas um
arquivo pdf será gerado.

for (i in 1:ncol(m))#14 Cria um contador i que vai de 1 até o número de
colunas do objeto "m" (colunas das combinações).
{  #15 Abriu ciclo.
   for (j in 1:2) #16 Cria um contador j que vai de 1 até o número 2
(combinações apenas par a par).
   {x[j]=m[j,i] #17 Para cada valor da linha (j), ciclar entre as colunas do
objeto "m". Gerando, então, todas as posições possíveis na matriz "m".
   } #18 Fecha o segundo ciclo (j).
 print(ggbiplot(pca,ellipse=TRUE,groups=grupos,choices=x)+ #19 Imprime os
gráficos nas diferentes combinações de dimensões a partir da função
ggbiplot. Fazendo uma elipse para cada grupo e utilizando os valores de
"grupos" como referência e utilizando as dimensões do objeto "x".
    geom_point(aes_string(color=grupos))+ #20 Seta cores dos pontos do
gráfico de acordo com o objeto "grupos".
    theme(legend.position="bottom"))#21 Posiciona a legenda na parte debaixo
do gráfico.
} #22 Fecha o primeiro ciclo i.

dev.off()#23 Fecha o dispositivo gráfico.
shell.exec("PCA_plots.pdf")#24 Abre o arquivo pdf gerado.
```

}