# Pacote GGCORRPLOT

Carlos Roberto de Nazaré Carvalho Junior

Fernando José Pessoa Andrade

Ivanildo Batista da Silva Júnior

Vanessa Karoline Inacio Gomes

Vaniele da Silva Barros

Professor Dr. Antonio Samuel - PPGBEA (UFRPE)

14 de Junho de 2021

# Autor do Pacote

O autor desse pacote chama-se Alboukadel Kassambara que, além desse pacote, é autor de livros na área de estatística e aprendizado de máquina com *R*. Seu repositório público no *Github* e seu *blog* podem ser conferidos, respectivamente, aqui e aqui.

O pacote foi publicado no **CRAN** em 19 de Maio de 2019 e atualmente está na sua versão 0.1.3 e documentação usada nessa apresentação pode ser acessada aqui.

# Descrição do Pacote

O pacote *ggcorrplot* é usado para visualizar facilmente uma matriz de correlação usando o pacote *ggplot2*. Isso fornece uma solução para reodernar a matriz de correlação e exibição dos níveis de significância no gráfico. Também está inclusa a função *cor_pmat()* para calcular os p-valores da matriz de correlação (com base na função *cor.test()*).

**Vantagens do *GGCORRPLOT***

- Lida com vários testes de correlação ao mesmo tempo;
- Visualiza bem os resultados;
- Menos código para gerar o gráfico;
- Mantém as características do *GGPLOT2*;
- Pode produzir p-valores, intervalos de confiança ou alguma outra maneira de sugerir se as correlações encontradas são estatisticamente significativas ou não.

# Instalação do pacote

```
install.packages('ggcorrplot')
library(ggcorrplot)
```

# Principais funções do pacote

**ggcorrplot()** : Essa função gera uma exibição gráfica da matriz de correlação usando o pacote *GGPLOT2*.

**Composição da função**:

**Parâmetros da função ggcorrplot()**:

- **corr** : a matriz de correlação para visualizar
- **method** : é um caracter que irá definir o método de visualização da matriz de correlação. Por padrão usa o "square" (formato quadrado), mas também pode ser usado o formato circular ou "*circle*".
- **type** : Por padrão usa o argumento "*full*", porém permite o uso de "lower" e "upper" para exibição da matriz de correlação.
- *ggtheme* : função ggplot2 ou objeto tema. Por padrão o valor é *theme_minimal*. Permite temas oficiais da *ggplot2*, incluindo *theme_gray*, *theme_bw*, *theme_minimal*, *theme_classic*, *theme_void*,etc.
- *title* : Título do gráfico.
- *show.legend* : Para exibir a legenda do gráfico.
- *legend.title* : Exibir o título da legenda.
- *show.diag* : argumento lógico, para exibir os coeficientes de correlação na diagonal principal.
- *colors* : um vetor de 3 cores para valores de correlação baixo, médio e alto.
- *outline.color* : a cor do contorno de um quadrado ou círculo. O valor padrão é cinza ("gray").
- **hc.order** : valor lógico. Se *TRUE*, a matriz de correlação será ordenada usando a função *hclust*.

- *lab_col, lab_size* : tamanho e cor a serem usados para os rótulos do coeficiente de correlação. usado quando *lab = TRUE*.
- **p.mat** : matriz do p-valor. Se *NULL*, os argumentos *sig.level, insig, pch, pch.col* e *pch.cex* são inválidos.
- *sig.level* : nível significativo, se o valor p em *p-mat* for maior que *sig.level*, então o coeficiente de correlação correspondente é considerado insignificante.
- *insig* : coeficientes de correlação insignificantes especializados, "pch" (padrão), "em branco". Se estiver "em branco", limpe os glifos correspondentes; se "pch", adiciona caracteres nos glifos correspondentes.
- *pch* : adiciona caracteres nos glifos de coeficientes de correlação insignificantes (válido apenas quando insig é "pch"). O valor padrão é 4.
- *pch.col, pch.cex* : a cor e o cex (tamanho) de *pch* (válido apenas quando insig é "pch").
- *tl.cex, tl.col, tl.srt* : o tamanho, a cor e a rotação da *string* do rótulo de texto (nomes de variáveis). Decide o número de casas decimais a serem exibidas, por padrão esse valor é 2.

Ao final a função irá retornar um gráfico *ggplot2*.

**cor_pmat()**: Calcula matriz de correlação de p-valores. Essa função aplica todas as variáveis do *dataframe* o teste de correlação de *Pearson*.

**Composição da função**:

```
cor_pmat(x, ...)
```

**Parâmetros da função cor_pmat()**

- **x** : matriz numérica ou um *dataframe*.
- **...** : outros argumentos a serem passados na função *cor.test*, que é a função usada para calcular a correlação. É possível definir o método de cálculo de correlação (*method* - *pearson* (padrão), *kendall* ou *spearman*), alterar a hipótese do teste (*alternative* - "*two.sided*", "*greater*" ou "*less*"), nível de confiança do teste de correlação (*conf.level*) e se o p-valor deve ser ou não calculado (*exact*).

# Usando o pacote GGCORRPLOT

Para essa apresentação usaremos o *dataset iris*, que é conjunto de dados nativo do *R* e o *quakes*. Segue abaixo as cinco primeiras observações de cada *dataset*:

```
head(iris[1:4],5)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1          5.1         3.5          1.4         0.2
## 2          4.9         3.0          1.4         0.2
## 3          4.7         3.2          1.3         0.2
## 4          4.6         3.1          1.5         0.2
## 5          5.0         3.6          1.4         0.2
```

```
head(quakes,5)
```

```
##      lat   long depth mag stations
## 1 -20.42 181.62   562 4.8       41
## 2 -20.62 181.03   650 4.2       15
## 3 -26.00 184.10    42 5.4       43
## 4 -17.97 181.66   626 4.1       19
## 5 -20.42 181.96   649 4.0       11
```

Antes de gerar os gráficos de correlação, será abordada a função *cor_pmat()* que, como dito anteriormente, gera uma matriz de correlação com p-valores. Esses p-valores são gerados da função *cor.test()*. No exemplo abaixo usaremos essa função com um exemplo, usando as colunas *Sepal.Length* e *Sepal.Width*

```
cor.test(iris$Sepal.Length,iris$Sepal.Width)
```

```
##
##     Pearson's product-moment correlation
##
## data:  iris$Sepal.Length and iris$Sepal.Width
## t = -1.4403, df = 148, p-value = 0.1519
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.27269325  0.04351158
## sample estimates:
##       cor
## -0.1175698
```

Nesse exemplo o p-valor igual a **0.1519**. Por padrão usa-se o método de *Pearson,* mas é possível usar o método de *Kendall* e de *Spearman*.

Usando o método de *Kendall* e extraindo o p-valor.

```
cor.test(iris$Sepal.Length,iris$Sepal.Width, method = "kendall")[3]
```

```
## $p.value
## [1] 0.182921
```

Usando o método de *Spearman* e extraindo o p-valor.

```
cor.test(iris$Sepal.Length,iris$Sepal.Width, method = "spearman")[3]
```

```
## $p.value
## [1] 0.04136799
```

Todos os valores são diferentes. Alterar o método de cálculo de correlação também pode ser aplicado para a função *cor_pmat()*, conforme veremos a seguir.

Abaixo temos a matriz com os p-valores, com o método padrão.

```
cor_pmat(iris[1:4])
```

```
##              Sepal.Length  Sepal.Width Petal.Length  Petal.Width
## Sepal.Length 0.000000e+00 1.518983e-01 1.038667e-47 2.325498e-37
## Sepal.Width  1.518983e-01 0.000000e+00 4.513314e-08 4.073229e-06
## Petal.Length 1.038667e-47 4.513314e-08 0.000000e+00 4.675004e-86
## Petal.Width  2.325498e-37 4.073229e-06 4.675004e-86 0.000000e+00
```

Por *Kendall*.

```
cor_pmat(iris[1:4], method="kendall")
```

```
##              Sepal.Length Sepal.Width Petal.Length  Petal.Width
## Sepal.Length 0.000000e+00 0.182921015 1.169126e-36 8.801486e-30
## Sepal.Width  1.829210e-01 0.000000000 1.283268e-03 7.518013e-03
## Petal.Length 1.169126e-36 0.001283268 0.000000e+00 2.443446e-44
## Petal.Width  8.801486e-30 0.007518013 2.443446e-44 0.000000e+00
```

Por *Spearman*.

```
cor_pmat(iris[1:4], method="spearman", exact=FALSE)
```

```
##              Sepal.Length  Sepal.Width Petal.Length  Petal.Width
## Sepal.Length 0.000000e+00 0.0413679942 3.443087e-50 4.189447e-40
## Sepal.Width  4.136799e-02 0.0000000000 1.153938e-04 3.342981e-04
## Petal.Length 3.443087e-50 0.0001153938 0.000000e+00 8.156597e-70
## Petal.Width  4.189447e-40 0.0003342981 8.156597e-70 0.000000e+00
```

A função *cor_pmat()* é importante, pois serve de argumento para a função *ggcorrplort()*, que será aplicada mais a frente. Para mais informações sobre a função *cor.test()* deve-se consultar sua documentação.

### Calculando a correlação

Antes de gerar os gráficos é necessário criar a matriz de correlação com a função `cor()`:

```
corr ← round(cor(quakes),2)
corr
```

```
##            lat  long depth   mag stations
## lat       1.00 -0.36  0.03 -0.05     0.00
## long     -0.36  1.00  0.14 -0.17    -0.05
## depth     0.03  0.14  1.00 -0.23    -0.07
## mag      -0.05 -0.17 -0.23  1.00     0.85
## stations  0.00 -0.05 -0.07  0.85     1.00
```

O objeto gerado é inserido dentro da função `ggcorrplot()`. Abaixo é gerado uma plotagem simples, sem nenhuma informação, a não ser a barra lateral do gráfico que mostra que cores mais "quentes"" indicam correlação positiva e cores "frias" correlação negativa.

```
ggcorrplot(corr)
```

Alterando o formato de apresentação das correlações do formato padrão para círculos.

```
ggcorrplot(corr, method = "circle")
```

Reordenando a matriz de correlação.

```
ggcorrplot(corr, hc.order = T, outline.color = "white")
```

### Alterando o *layout*

Alterando o *layout* para *lower triangle*.

```
ggcorrplot(corr, hc.order = T, type = "lower",
           outline.color = "white")
```

Alterando o *layout* para *upper triangle*.

```
ggcorrplot(corr, hc.order = T, type = "upper",
           outline.color = "white")
```

Alterando as cores e o tema.

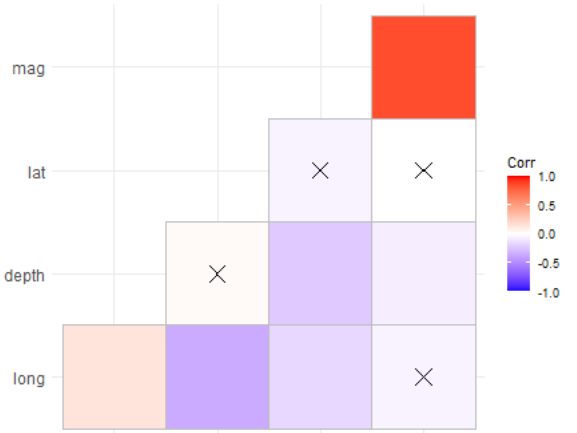
```
ggcorrplot(corr, hc.order = T, type = "lower", outline.color = "white
           ggtheme = ggplot2::theme_gray, colors = c("#6D9EC1","white
```

Adicionando coeficientes de correlação e mudando o tema.

```
ggcorrplot(corr, hc.order = T, type = "lower",
           lab = T,ggtheme = ggplot2::theme_dark())
```

Adicionando o nível de confiança da correlação. Aplicamos para a função o argumento `p.mat` e os coeficientes não significativos são excluídos.

```
p.mat ← cor_pmat(quakes)
ggcorrplot(corr,hc.order = T, type = "lower", p.mat = p.mat)
```



Deixando espaço vazio para coeficientes não significantes. Utilizando *insig* = "*blank*".

```
ggcorrplot(corr, p.mat = p.mat, hc.order = T,type = "lower", insig =
```

Inserindo título e tirando a legenda.

```
ggcorrplot(corr,hc.order = T, type = "lower", title='Gráfico de corre
           show.legend = F, show.diag = T)
```

Matriz de correlação com e sem a **diagonal principal**.

```
p1 ← ggcorrplot(corr,hc.order = T, type = "lower", title='Gráfico de
                show.legend = F, show.diag = T)
p2← ggcorrplot(corr,hc.order = T, type = "lower", title='Gráfico de
               show.legend = F, show.diag = F)
grid.arrange(p1, p2, ncol = 2)
```

Trabalhando com a legenda (cor e tamanho).

```
ggcorrplot(corr, type = "lower", lab=T, lab_col = 'black', lab_size =
```

Agrupando a matriz de correlação. Abaixo usamos apenas três métodos de agrupamento, mas além desses existem "*ward.D*", "*ward.D2*", "*single*", "*average*" e "*centroid*".

```
p1 ← ggcorrplot(corr,hc.order = T, hc.method = "complete")
p2 ← ggcorrplot(corr,hc.order = T, hc.method = "median")
grid.arrange(p1, p2, ncol = 2, nrow = 1)
```

# GGCORRPLOT vs GGPLOT2

Criando um gráfico de correlação com *ggplot2*.

```
gplot ← ggplot(data = melt(round(cor(iris[c(2,1,3,4)]),2), na.rm = T
gplot
```

Criando um gráfico de correlação com *ggcorrplot*.

```
gcor ← ggcorrplot(cor(iris[c(2,1,3,4)]),hc.order = T, legend.title =
gcor
```

**Comparação**

- No primeiro gráfico gerado foi necessário instalar os pacotes **ggplot2** e **reshape2**, mas no segundo gráfico apenas o pacote **ggcorrplot**.
- No segundo gráfico a quantidade de código e argumentos necessário para obter o mesmo resultado do primeiro foi bem menor, temos um código mais simples e mais compreensível.
- A princípio o desempenho de ambos os cógidos são iguais, então, replicamos uma quantidade de vezes muito grande e o resultado pode ser visto abaixo:

```
system.time(replicate(10000000,gplot))
```

```
##    user  system elapsed
##   37.37    0.95   53.70
```

```
system.time(replicate(10000000,gcor))
```

```
##    user  system elapsed
##   34.27    0.90   41.78
```

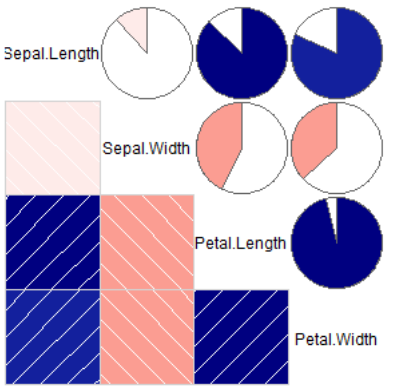
Então vemos que o segundo código tem um desempenho melhor que o primeiro.



# Outros pacotes para plotagem de correlogramas

**Correlograma com o pacote *CORRGRAM*.**

```
corrgram(iris, lorder = TRUE, lower.panel = panel.shade, upper.panel
    text.panel = panel.txt)
```

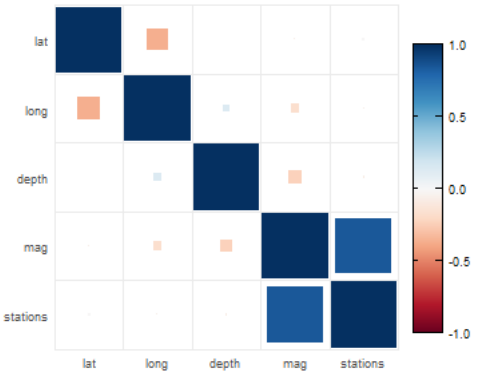


### Correlograma com o pacote *ELLIPSE*.

```
plotcorr(cor(iris[,1:4]), col = colorRampPalette(c("firebrick3", "whi
```

**Correlograma com o pacote *GGCORRPLOT2*.**

```
ggcorrplot(corr, method = "square")
```



### Correlograma com o pacote *GGALLY*.

```
ggcorr(mtcars)
```

### Correlograma com o pacote *CORRELATION* e *DPLYR*.

```
correlation(iris) %>% summary(redundant = TRUE) %>% plot()
```

**Documentação consultada**

Segue abaixo a documentação de cada um dos pacotes desse trabalho:

- GGCORRPLOT;
- GGCORRPLOT2 (Não encontra-se no *CRAN*);
- CORRGRAM;
- ELLIPSE;
- GGALLY;
- CORRELATION;

# Obrigado!

79 / 79