Ajuste de equacoes lineares - analise e aplicacoes

Sollano Rabelo Braga

Outubro, 2016

Contents

1) Baixar e/ou carregar pacotes necessarios

Este script contem secoes que podem ser rodadas sem o uso de nenhum pacote adicional no entanto, e interessante se conhecer os metodos alternativos, pois sao de melhor entendimento e facil computacao para isso precisa-se dos pacotes instalados e carregados. Este passo so sera necessario caso os pacotes nao estejam instalados, ou estejam desatualizados. para se instalar os pacotes, utiliza-se dos seguintes comandos:

```
# install.packages("gridExtra", dependencies = TRUE)
# install.packages("tidyverse", dependencies = TRUE)
```

Para se realizar a instalacao, basta remover o # e rodar o comando. Para carregar os pacotes, pode-se utilizar do comando library:

```
library(gridExtra)
library(tidyverse)
```

```
## Loading tidyverse: ggplot2
## Loading tidyverse: tibble
## Loading tidyverse: tidyr
## Loading tidyverse: readr
## Loading tidyverse: purrr
## Loading tidyverse: dplyr
```

```
## Conflicts with tidy packages ----------------------------------------------
```

```
## combine(): dplyr, gridExtra
## filter():  dplyr, stats
## lag():     dplyr, stats
```

Os pacotes devem ser carregados toda vez que se for utilizar o script.

2) Carregar a base de dados

Sera utilizada uma base de dados de inventario florestal, em fomato .csv. Para o carregamento dos dados, utiliza-se o comando read.csv2()

```
dados_orig <- read.csv2("dados.csv")
```

obs: utilizou-se do comando read.csv2 dois pois o separador deste arquivo e ";" e o decimal e ",". visuaza-se a base de dados com head:

```
head(dados_orig)
```

```
##   TALHAO PARCELA COD_ARVORE  CAP   HT CATEGORIA    HD
## 1   3654     101        301 51.1 28.4    Normal 30.45
## 2   3654     101        303 50.8  0.0    Normal 30.45
## 3   3654     101        304 53.2  0.0    Normal 30.45
## 4   3654     101        503 50.0  0.0    Normal 30.45
## 5   3654     101        302 51.3 29.0    Normal 30.45
## 6   3654     101        401 54.0  0.0    Normal 30.45
```

Percebe-se que a variavel DAP nao esta presente; como ela sera bastante utilizada no script, cria-se a coluna no dataframe original. Utilizando R base:

```
dados_orig$DAP <- dados_orig$CAP / pi

head(dados_orig)
```

```
##   TALHAO PARCELA COD_ARVORE  CAP   HT CATEGORIA    HD      DAP
## 1   3654     101        301 51.1 28.4    Normal 30.45 16.26564
## 2   3654     101        303 50.8  0.0    Normal 30.45 16.17014
## 3   3654     101        304 53.2  0.0    Normal 30.45 16.93409
## 4   3654     101        503 50.0  0.0    Normal 30.45 15.91549
## 5   3654     101        302 51.3 29.0    Normal 30.45 16.32930
## 6   3654     101        401 54.0  0.0    Normal 30.45 17.18873
```

utilizando o dplyr: utilizando mutate, pode-se criar novas variaveis sem a necessidade de se repetir o nome do objeto varias vezes:

```
dados_orig <- dados_orig %>% mutate(DAP = CAP / pi)
head(dados_orig)
```

```
##   TALHAO PARCELA COD_ARVORE  CAP   HT CATEGORIA    HD      DAP
## 1   3654     101        301 51.1 28.4    Normal 30.45 16.26564
## 2   3654     101        303 50.8  0.0    Normal 30.45 16.17014
## 3   3654     101        304 53.2  0.0    Normal 30.45 16.93409
## 4   3654     101        503 50.0  0.0    Normal 30.45 15.91549
## 5   3654     101        302 51.3 29.0    Normal 30.45 16.32930
## 6   3654     101        401 54.0  0.0    Normal 30.45 17.18873
```

Substitui-se 0 por NA para evitar erros na regressao. O R nao trabalha bem com zeros, entao e recomendado substitui-los por NA antes de comecar a trabalhar.

Utilizando o R base:

```
head(dados_orig)
```

```
##   TALHAO PARCELA COD_ARVORE  CAP   HT CATEGORIA    HD      DAP
## 1   3654     101        301 51.1 28.4    Normal 30.45 16.26564
## 2   3654     101        303 50.8  0.0    Normal 30.45 16.17014
## 3   3654     101        304 53.2  0.0    Normal 30.45 16.93409
## 4   3654     101        503 50.0  0.0    Normal 30.45 15.91549
## 5   3654     101        302 51.3 29.0    Normal 30.45 16.32930
## 6   3654     101        401 54.0  0.0    Normal 30.45 17.18873
```

```
dados_orig[dados_orig == 0 ] <- NA
head(dados_orig)
```

```
##   TALHAO PARCELA COD_ARVORE  CAP   HT CATEGORIA    HD      DAP
## 1   3654     101        301 51.1 28.4    Normal 30.45 16.26564
## 2   3654     101        303 50.8   NA    Normal 30.45 16.17014
## 3   3654     101        304 53.2   NA    Normal 30.45 16.93409
## 4   3654     101        503 50.0   NA    Normal 30.45 15.91549
## 5   3654     101        302 51.3 29.0    Normal 30.45 16.32930
## 6   3654     101        401 54.0   NA    Normal 30.45 17.18873
```

Utilizando o dplyr:

```
dados_orig <- na_if(dados_orig, 0)
head(dados_orig)
```

```
##   TALHAO PARCELA COD_ARVORE  CAP   HT CATEGORIA    HD      DAP
## 1   3654     101        301 51.1 28.4    Normal 30.45 16.26564
## 2   3654     101        303 50.8   NA    Normal 30.45 16.17014
## 3   3654     101        304 53.2   NA    Normal 30.45 16.93409
## 4   3654     101        503 50.0   NA    Normal 30.45 15.91549
## 5   3654     101        302 51.3 29.0    Normal 30.45 16.32930
```

```
## 6   3654     101        401 54.0   NA    Normal 30.45 17.18873
```

Na regressao os NAs podem gerar erros, portanto cria-se uma copia dos dados, onde remove-se todos os seus NAs, utilizando a funcao na.omit():

```
dados_sem_na <- na.omit(dados_orig)
head(dados_sem_na, 15)
```

```
##    TALHAO PARCELA COD_ARVORE  CAP   HT CATEGORIA    HD      DAP
## 1    3654     101        301 51.1 28.4    Normal 30.45 16.26564
## 5    3654     101        302 51.3 29.0    Normal 30.45 16.32930
## 9    3654     101        204 53.2 28.8    Normal 30.45 16.93409
## 10   3654     101        405 59.0 30.2 Dominante 30.45 18.78028
## 15   3654     101        101 51.0 29.2    Normal 30.45 16.23380
## 16   3654     101        201 59.3 30.7 Dominante 30.45 18.87578
## 17   3654     101        203 54.0 28.7    Normal 30.45 17.18873
## 18   3654     101        202 56.6 30.6    Normal 30.45 18.01634
## 21   3654     102        201 54.0 29.7    Normal 31.35 17.18873
## 22   3654     102        101 58.9 32.5    Normal 31.35 18.74845
## 28   3654     102        202 53.6 28.9    Normal 31.35 17.06141
## 31   3654     102        305 54.1 29.2    Normal 31.35 17.22056
## 37   3654     102        302 55.5 29.7    Normal 31.35 17.66620
## 38   3654     102        303 59.1 30.6 Dominante 31.35 18.81211
## 39   3654     102        204 57.7 29.8    Normal 31.35 18.36648
```

3) Calculo das variaveis necessarias para a regressao

SerÃ£o utilizados dois modelos: Modelo de Curtis (Modelo 1): LN(HT) = b0 + b1 * 1/DAP Modelo de Campos & Leite(Modelo 2): LN(HT) = b0 + b1 * 1/DAP + b2 * Ln (HD) antes de ajusta-los, calcula-se as variaveis utilizadas nos modelos. Este passo nao e obrigatorio em todos os modelos, porem, para evitar erros, e recomendado sempre o fazer. O calculo das variaveis pode ser feito utilizando o R base ou o pacote dplyr: utilizando o R base:

```
dados_sem_na$INV_DAP <- 1/dados_sem_na$DAP
dados_sem_na$LN_HT <- log(dados_sem_na$HT)
dados_sem_na$LN_HD <- log(dados_sem_na$HD)
```

utilizando o dplyr: Com o dplyr pode-se criar diversas variaveis com em uma unica linha, sem a necessidade de repetir o nome do objeto toda vez:

```
dados_sem_na <- dados_sem_na %>% mutate(INV_DAP=1/DAP, LN_HT=log(HT), LN_HD=log(HD))
```

4) Ajuste dos modelos

4.1) Modelo de Curtis (Modelo 1) (R base)

O modelo sera ajustado e salvo em um objeto separado, utilizando a funcao lm. insere-se primeiro o modelo, separando o lado y e x da equacao com "~", seguido do argumento data, que diz qual dado sera utilizado.

```
modelo1 <- lm(LN_HT ~ INV_DAP, data = dados_sem_na)
modelo1
```

```
##
## Call:
```

```
## lm(formula = LN_HT ~ INV_DAP, data = dados_sem_na)
##
## Coefficients:
## (Intercept)      INV_DAP
##       3.889      -11.668
```

Utilizando a funcao class percebe-se que a classe do objeto gerado e lm:

```
class(modelo1)
```

```
## [1] "lm"
```

a funcao coef chama os coeficientes de um objeto lm; com ela pode-se criar um objeto separado que contem apenas os coeficientes:

```
modelo1coef <- coef(modelo1)
modelo1coef
```

```
## (Intercept)     INV_DAP
##    3.888954  -11.668419
```

pode-se chamar os coeficientes com base em sua posicao neste objeto. esse objeto sempre sera um vetor com n componentes, sendo n o numero de coeficientes do modelo ajustado. Neste caso, foi ajustado um modelo que gerou dois coeficientes, b0 e b1. portanto, b0 estara na posicao 1, e b1 na posicao 2:

```
modelo1coef[[1]] # b0
```

```
## [1] 3.888954
```

```
modelo1coef[[2]] # b1
```

```
## [1] -11.66842
```

pode-se utilizar "$", para chamar resultados por nome, ou "[]", para chamar por posicao Aqui tem-se um resumo geral da regressao, com a funcao summary:

```
modelo1summary <- summary(modelo1)
modelo1summary
```

```
##
## Call:
## lm(formula = LN_HT ~ INV_DAP, data = dados_sem_na)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.29798 -0.08978  0.02432  0.08214  0.40316
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   3.88895    0.03106  125.22   <2e-16 ***
## INV_DAP     -11.66842    0.51722  -22.56   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1076 on 1396 degrees of freedom
## Multiple R-squared:  0.2672, Adjusted R-squared:  0.2666
## F-statistic:   509 on 1 and 1396 DF,  p-value: < 2.2e-16
```

Sua classe pode ser verificada com a funcao class:

```
class(modelo1summary)
```

```
## [1] "summary.lm"
```

Para se chamar o r quadrado ajustado, existem duas formas:

```
modelo1summary$adj.r.squared
```

```
## [1] 0.2666496
```

```
modelo1summary[[9]]
```

```
## [1] 0.2666496
```

Assim como o sigma, ou erro-padrao:

```
modelo1summary$sigma
```

```
## [1] 0.1076069
```

```
modelo1summary[[6]]
```

```
## [1] 0.1076069
```

Com isso, pode-se criar um dataframe com os coeficientes, R2, e erro-padrao. para isso utiliza-se a funcao data.frame, onde indica-se o nome e o conteudo de cada variavel (coluna) do data frame que sera criado:

```
modelo1df <- data.frame(b0        = modelo1coef[[1]],
                        b1        = modelo1coef[[2]],
                        R2_aj     = modelo1summary$adj.r.squared,
                        erro_pad  = modelo1summary$sigma,
                        row.names = NULL)
modelo1df
```

```
##         b0        b1     R2_aj  erro_pad
## 1 3.888954 -11.66842 0.2666496 0.1076069
```

4.2) Modelo de Campos e Leite (Modelo 2) (dplyr)

Com o pacote dplyr, o processo pode ser feito de forma mais direta. utilizando cada passo seguido de %>% , evita-se a criacao de objetos adicionais, ou funcoes longas com varios parenteses, que podem se tornar confusas. Serao utilizadas as seguintes funcoes do dplyr: a funcao mutate, que cria novas variaveis, a funcao do, que roda funcoes nativas do R dentro de pipes do dplyr, rowwise, para aplicar funcoes por linha, e transmute, que e similar a mutate, criando novas variaveis, porem, esta mantem apenas as variaveis criadas, descartando as demais.

```
modelo2df <- dados_sem_na %>%  # data frame que sera utilizado
  mutate(DAP=CAP/pi, INV_DAP=1/DAP, LN_HT=log(HT), LN_HD=log(HD))  %>%
  do(Reg = lm(LN_HT ~ INV_DAP + LN_HD, data = .) ) %>%  # ajusta-se a regressao
  rowwise()  %>% # importante para aplicar a funcao por linha, direto no modelo
  transmute(b0     = coef(Reg)[[1]], # extrai-se os coeficientes da funcao coef
            b1     = coef(Reg)[[2]],
            b2     = coef(Reg)[[3]],
            R2_aj = summary(Reg)[[9]],
            erro_pad   = summary(Reg)[[6]] )
modelo2df
```

```
## Source: local data frame [1 x 5]
## Groups: <by row>
```

```
##
## # A tibble: 1 × 5
##          b0        b1        b2     R2_aj   erro_pad
##       <dbl>     <dbl>     <dbl>     <dbl>      <dbl>
## 1 0.6828138 -4.903735 0.8706277 0.9162879 0.03635618
```

Utiliza-se [[]] para extrair o numero em si, com base em suas posicoes, pois [] extrai um objeto de classe lista, devido a forma que foi criado (funcao do()). extrai-se o R2 e sigma do objeto gerado pela funcao summary com base em suas posicoes.

5) Estimar as alturas que nao foram medidas

Primeiro faz-se uma copia dos dados originais, para que eles nao sejam alterados:

```
dadosest <- dados_orig
```

Agora pode-se estimar a altura. Neste caso, serao estimadas apenas as alturas das arvores que nao foram medidas; as arvores nao medidas possuem NA no lugar do valor da altura, portanto, utiliza-se a funcao ifelse, que se baseia em um teste logico, e retorna um resultado caso ele seja verdadeiro, e outro caso seja falso. assim, cria-se uma nova coluna com a funcao ifelse, onde ela ira estimar a altura caso haja NA, e caso nao haja, ela ira inserir o valor da variavel HT

```
dadosest$HT_EST <- ifelse(is.na(dados_orig$HT),
                          exp(modelo1df$b0+modelo1df$b1*( 1/dados_orig$DAP ) ),
                          dados_orig$HT  )
head(dadosest)
```

```
##   TALHAO PARCELA COD_ARVORE  CAP   HT CATEGORIA    HD      DAP   HT_EST
## 1   3654     101        301 51.1 28.4    Normal 30.45 16.26564 28.40000
## 2   3654     101        303 50.8   NA    Normal 30.45 16.17014 23.74451
## 3   3654     101        304 53.2   NA    Normal 30.45 16.93409 24.53020
## 4   3654     101        503 50.0   NA    Normal 30.45 15.91549 23.47195
## 5   3654     101        302 51.3 29.0    Normal 30.45 16.32930 29.00000
## 6   3654     101        401 54.0   NA    Normal 30.45 17.18873 24.78189
```

Pode-se fazer o processo direto com o dplyr, desde o ajuste do modelo, ate a estimacao da altura:

```
dadosest <- dados_orig %>% # selecao do data frame
  na.omit %>% # remocao dos NAs
  mutate(DAP=CAP/pi,
         INV_DAP=1/DAP,
         LN_HT=log(HT))  %>% # criacaodas variaveis
  do(Reg = lm(LN_HT ~ INV_DAP, data = .) ) %>%  # ajuste do modelo
  rowwise %>%  # aplicar funcoes por linha
  transmute(b0 = coef(Reg)[[1]],
            b1 = coef(Reg)[[2]]  ) %>%
  cbind(dados_orig, . ) %>% # uniao com o dataframe original
  mutate(HT_EST = ifelse(is.na(HT),
                         exp(b0 + b1 * (1/DAP) ),
                         HT  ) ) %>%  # estimacao da altura
  select(-b0, -b1)
head(dadosest)
```

```
##   TALHAO PARCELA COD_ARVORE  CAP   HT CATEGORIA    HD      DAP   HT_EST
## 1   3654     101        301 51.1 28.4    Normal 30.45 16.26564 28.40000
## 2   3654     101        303 50.8   NA    Normal 30.45 16.17014 23.74451
```

```
## 3   3654     101        304 53.2   NA    Normal 30.45 16.93409 24.53020
## 4   3654     101        503 50.0   NA    Normal 30.45 15.91549 23.47195
## 5   3654     101        302 51.3 29.0    Normal 30.45 16.32930 29.00000
## 6   3654     101        401 54.0   NA    Normal 30.45 17.18873 24.78189
```

6) Analise grafica - Modelo de Curtis (Modelo 1)

Serao utilizados 3 tipos de graficos nesta analise: y estimado x y observado; dispersao de residuos em porcentagem; histograma de residuos em porcentagem. Para o calculo do residuo, precisa-se estimar as alturas das arvores que ja temos a medicao, e comparar umas com as outras; portanto, o primeiro passo e estimar as alturas das arvores ja medidas. primeiro cria-se um novo objeto, que ira conter os dados utilizados nos graficos.

```
dados_graph <- dados_sem_na
head(dados_graph)
```

```
##   TALHAO PARCELA COD_ARVORE  CAP   HT CATEGORIA    HD      DAP    INV_DAP
## 1   3654     101        301 51.1 28.4    Normal 30.45 16.26564 0.06147931
## 2   3654     101        302 51.3 29.0    Normal 30.45 16.32930 0.06123962
## 3   3654     101        204 53.2 28.8    Normal 30.45 16.93409 0.05905249
## 4   3654     101        405 59.0 30.2 Dominante 30.45 18.78028 0.05324733
## 5   3654     101        101 51.0 29.2    Normal 30.45 16.23380 0.06159986
## 6   3654     101        201 59.3 30.7 Dominante 30.45 18.87578 0.05297795
##      LN_HT    LN_HD
## 1 3.346389 3.416086
## 2 3.367296 3.416086
## 3 3.360375 3.416086
## 4 3.407842 3.416086
## 5 3.374169 3.416086
## 6 3.424263 3.416086
```

agora cria-se nova variavel, onde estima-se a altura das arvores:

```
dados_graph$HT_EST <- exp(modelo1df$b0 + modelo1df$b1 * dados_graph$INV_DAP)
```

e por fim, calcula-se o residuo em porcentagem:

```
dados_graph$RES <-( (dados_graph$HT_EST - dados_graph$HT) / dados_graph$HT ) * 100

head(dados_graph)
```

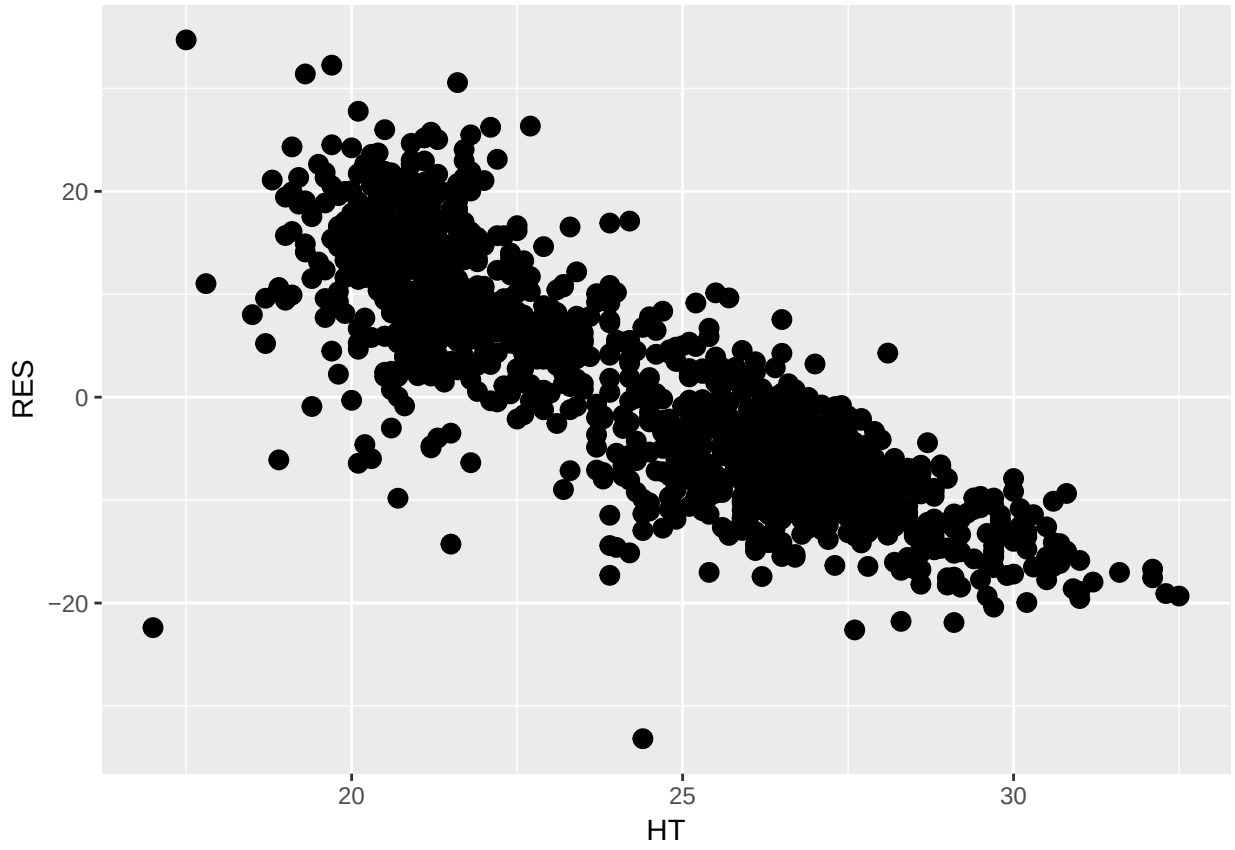
```
##   TALHAO PARCELA COD_ARVORE  CAP   HT CATEGORIA    HD      DAP    INV_DAP
## 1   3654     101        301 51.1 28.4    Normal 30.45 16.26564 0.06147931
## 2   3654     101        302 51.3 29.0    Normal 30.45 16.32930 0.06123962
## 3   3654     101        204 53.2 28.8    Normal 30.45 16.93409 0.05905249
## 4   3654     101        405 59.0 30.2 Dominante 30.45 18.78028 0.05324733
## 5   3654     101        101 51.0 29.2    Normal 30.45 16.23380 0.06159986
## 6   3654     101        201 59.3 30.7 Dominante 30.45 18.87578 0.05297795
##      LN_HT    LN_HD   HT_EST       RES
## 1 3.346389 3.416086 23.84532 -16.03761
## 2 3.367296 3.416086 23.91210 -17.54448
## 3 3.360375 3.416086 24.53020 -14.82569
## 4 3.407842 3.416086 26.24937 -13.08155
## 5 3.374169 3.416086 23.81180 -18.45273
## 6 3.424263 3.416086 26.33201 -14.22798
```

agora que tem-se os dados preparados, pode-se plotar os graficos.

6.1) Grafico de Dispersao

grafico basico

```
ggplot(dados_graph, aes(HT, RES))  +  geom_point(size = 3)
```



adicionar elementos

```
ggplot(dados_graph, aes(HT, RES))  +
  geom_hline(yintercept = 0, colour = "gray45") + # linha no zero
  geom_point(size = 3)  + # aumentar os pontos
  coord_cartesian(ylim = c(-40,40) )  # limites do eixo x
```

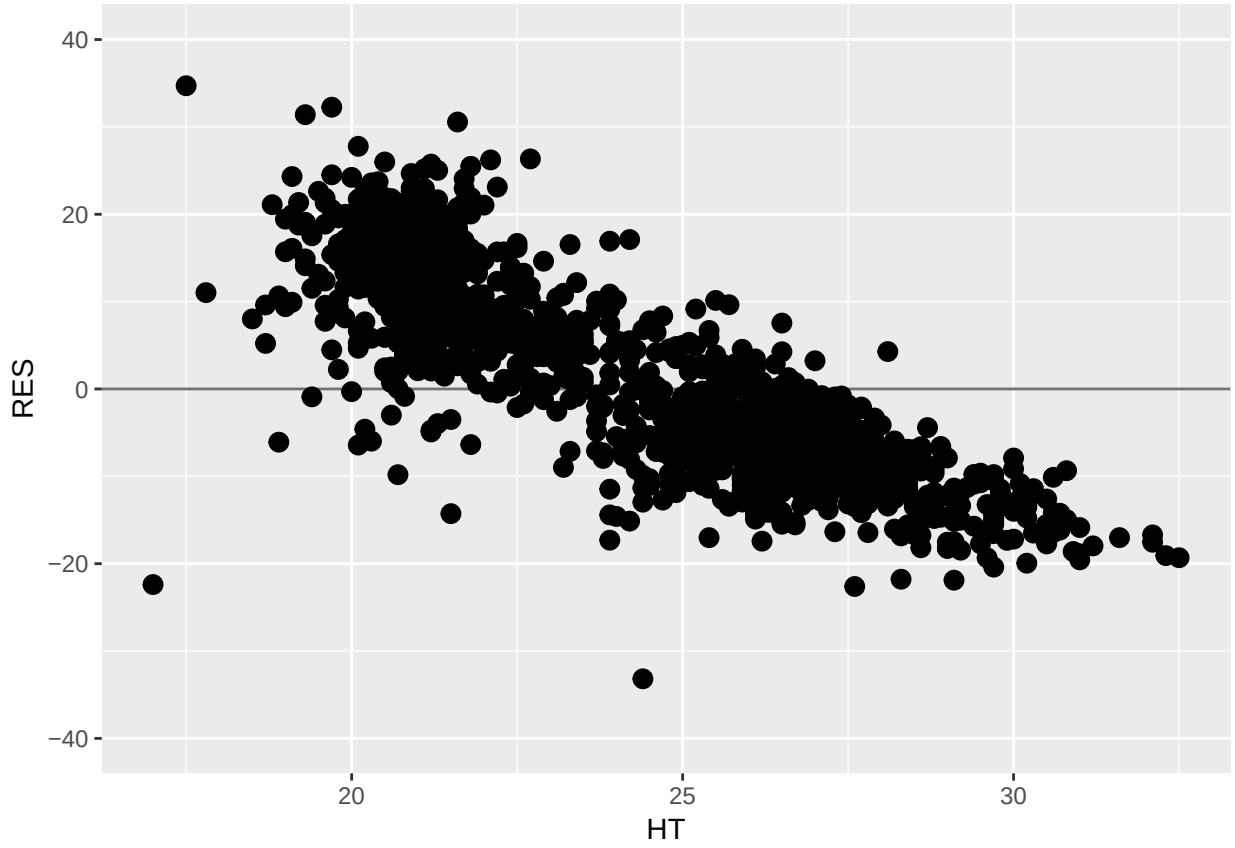
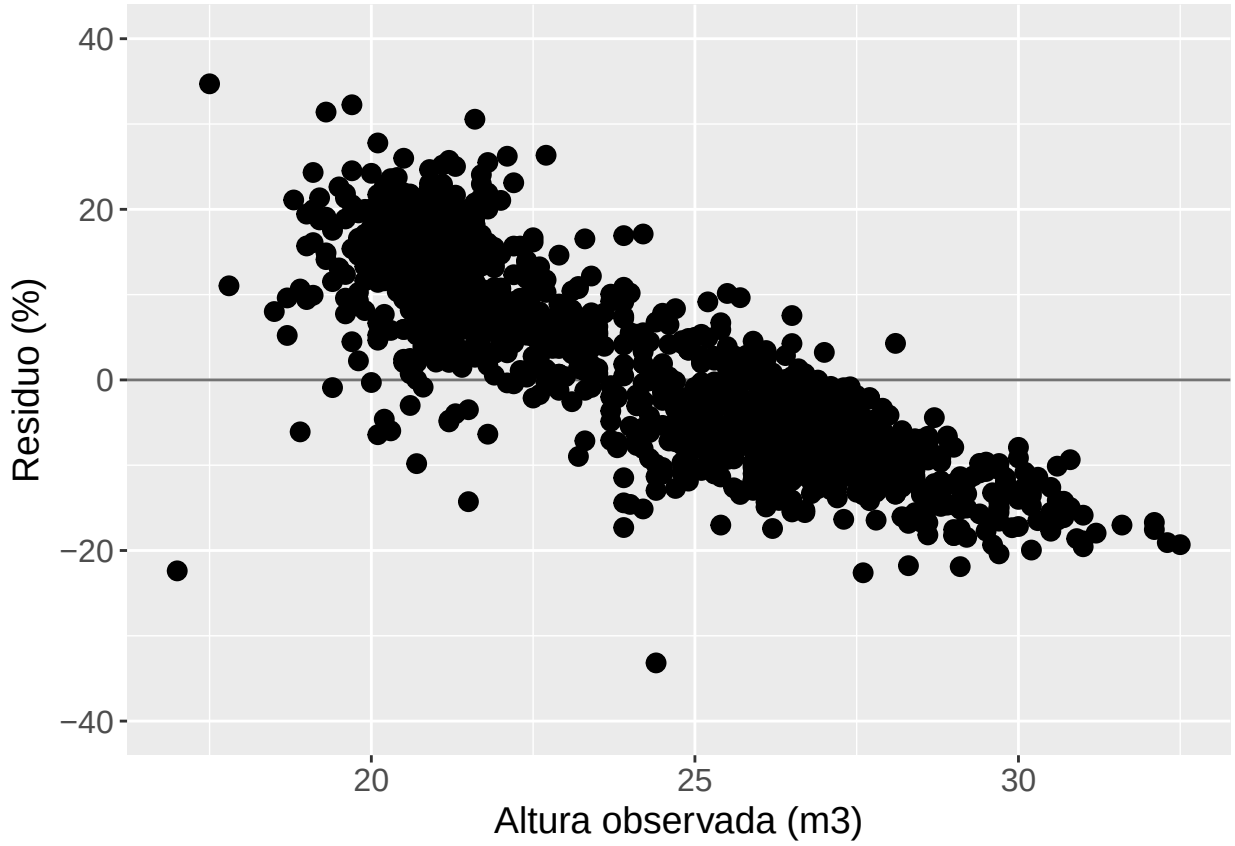


grafico final Este grafico sera salvo em um objeto, para ser uzado futuramente:

```
mod1_gres <- ggplot(dados_graph, aes(HT, RES))  +
  geom_hline(yintercept = 0, colour = "gray45") +
  geom_point(size = 3)  +
  coord_cartesian(ylim = c(-40,40)) +
  labs(x      = "Altura observada (m3)",
       y      = "Residuo (%)"   ) +
  theme(  # com theme muda-se tamanho e estilo de letra de cada parte do grafico
    axis.title   = element_text(size = 14),
    axis.text    = element_text(size = 12))

mod1_gres
```



6.2) Histograma

```
ggplot(dados_graph,aes(RES, ..density..) ) +
  geom_vline(xintercept = 0, colour = "gray45")  +
  geom_histogram(binwidth = 1)  +
  xlim(-40, 40)
```

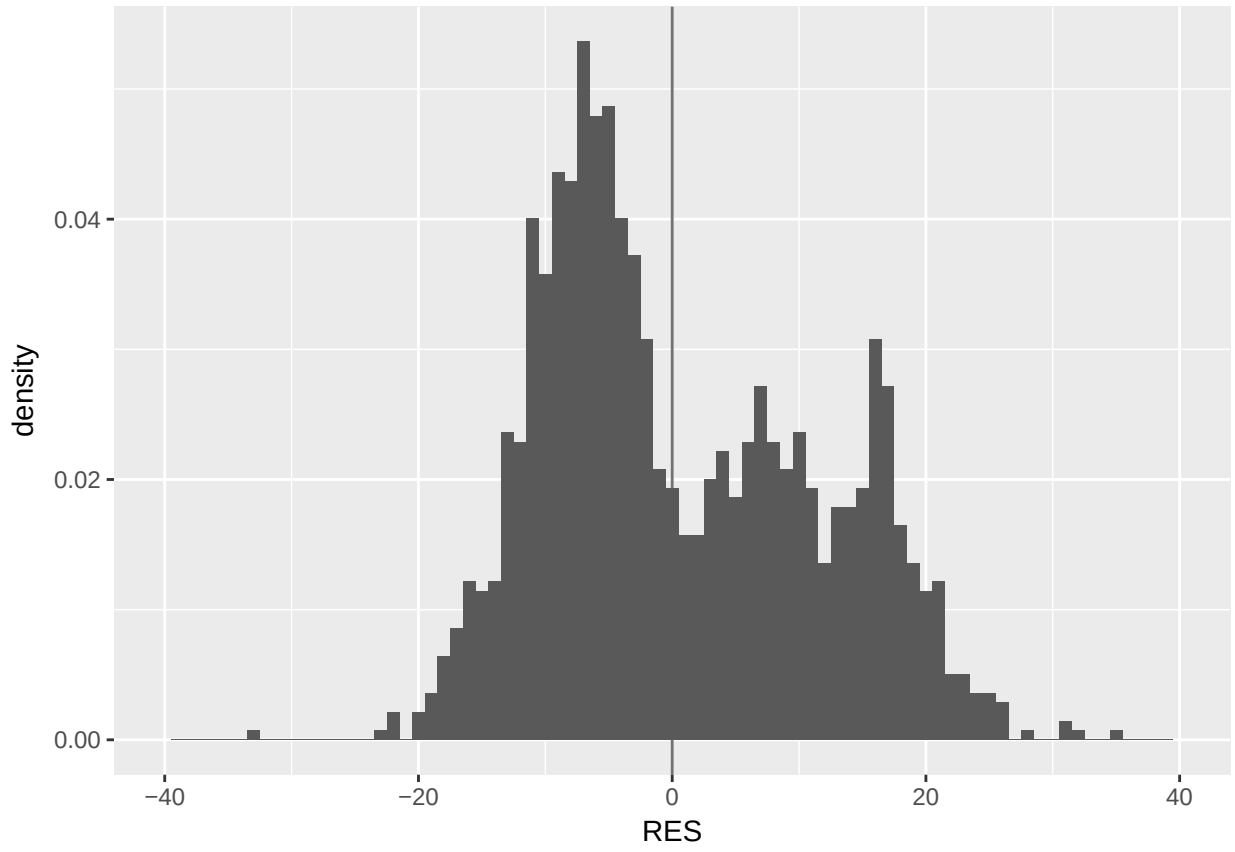
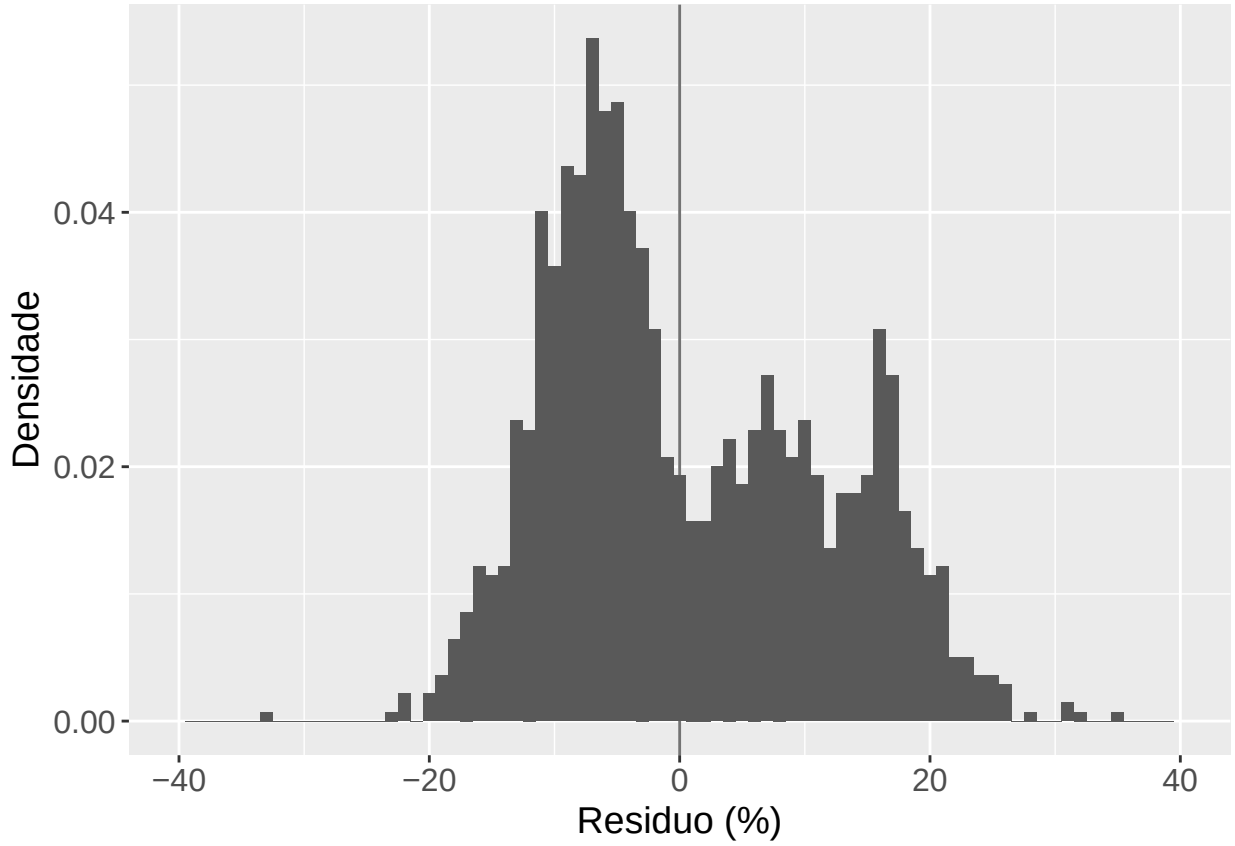


grafico final Este grafico sera salvo em um objeto, para ser uzado futuramente:

```
mod1_ghist <- ggplot(dados_graph,aes(RES, ..density..) ) +
  geom_vline(xintercept = 0, colour = "gray45")  +
  geom_histogram(binwidth = 1)  +
  xlim(-40, 40) +
  labs(x    = "Residuo (%)",
       y     = "Densidade" ) +
  theme( #com theme muda-se o tamanho e fonte da letra
    axis.title    = element_text(size = 14),
    axis.text     = element_text(size = 12))

mod1_ghist
```



7) Comparacao de modelos

e se o objetivo for definir qual o melhor modelo para uma base de dados? ja foi criado o objeto dados_graph, que possui os dados para o grafico do modelo 1. O processo sera repetido para o modelo 2, e em seguida -se os dois dados por linha: agora cria-se o dado utilizando o dplyr:

```
dados_graph2 <- dados_sem_na %>%
  cbind(modelo2df) %>%
  mutate(
    DAP    = CAP/pi,
    HT_EST = exp(b0 + b1 * (1/DAP) + b2 * log(HD) ),
    RES    = ( (HT_EST - HT) / HT )* 100 ) %>%
  select(-b0, -b1, -b2, -R2_aj, -erro_pad)

head(dados_graph2)
```

```
##   TALHAO PARCELA COD_ARVORE  CAP   HT CATEGORIA    HD      DAP    INV_DAP
## 1   3654     101        301 51.1 28.4    Normal 30.45 16.26564 0.06147931
## 2   3654     101        302 51.3 29.0    Normal 30.45 16.32930 0.06123962
## 3   3654     101        204 53.2 28.8    Normal 30.45 16.93409 0.05905249
## 4   3654     101        405 59.0 30.2 Dominante 30.45 18.78028 0.05324733
## 5   3654     101        101 51.0 29.2    Normal 30.45 16.23380 0.06159986
## 6   3654     101        201 59.3 30.7 Dominante 30.45 18.87578 0.05297795
##      LN_HT    LN_HD   HT_EST        RES
## 1 3.346389 3.416086 28.65920  0.9126793
```

```
## 2 3.367296 3.416086 28.69291 -1.0589468
## 3 3.360375 3.416086 29.00230  0.7024166
## 4 3.407842 3.416086 29.83977 -1.1928242
## 5 3.374169 3.416086 28.64226 -1.9100531
## 6 3.424263 3.416086 29.87921 -2.6735817
```

agora une-se os dois dados em um unico data.frame. vamos utiliza-se bind_rows, do pacote dplyr, pois esta funcao possui possui o argumento .id, que add uma variavel para identificar cada dataframe.

```
dados_graph_bind <- bind_rows("Curtis" = dados_graph,
                              "Campos & Leite" = dados_graph2,
                              .id = "Modelo")
head(dados_graph_bind)
```

```
##   Modelo TALHAO PARCELA COD_ARVORE  CAP   HT CATEGORIA    HD      DAP
## 1 Curtis   3654     101        301 51.1 28.4    Normal 30.45 16.26564
## 2 Curtis   3654     101        302 51.3 29.0    Normal 30.45 16.32930
## 3 Curtis   3654     101        204 53.2 28.8    Normal 30.45 16.93409
## 4 Curtis   3654     101        405 59.0 30.2 Dominante 30.45 18.78028
## 5 Curtis   3654     101        101 51.0 29.2    Normal 30.45 16.23380
## 6 Curtis   3654     101        201 59.3 30.7 Dominante 30.45 18.87578
##      INV_DAP    LN_HT    LN_HD   HT_EST       RES
## 1 0.06147931 3.346389 3.416086 23.84532 -16.03761
## 2 0.06123962 3.367296 3.416086 23.91210 -17.54448
## 3 0.05905249 3.360375 3.416086 24.53020 -14.82569
## 4 0.05324733 3.407842 3.416086 26.24937 -13.08155
## 5 0.06159986 3.374169 3.416086 23.81180 -18.45273
## 6 0.05297795 3.424263 3.416086 26.33201 -14.22798
```

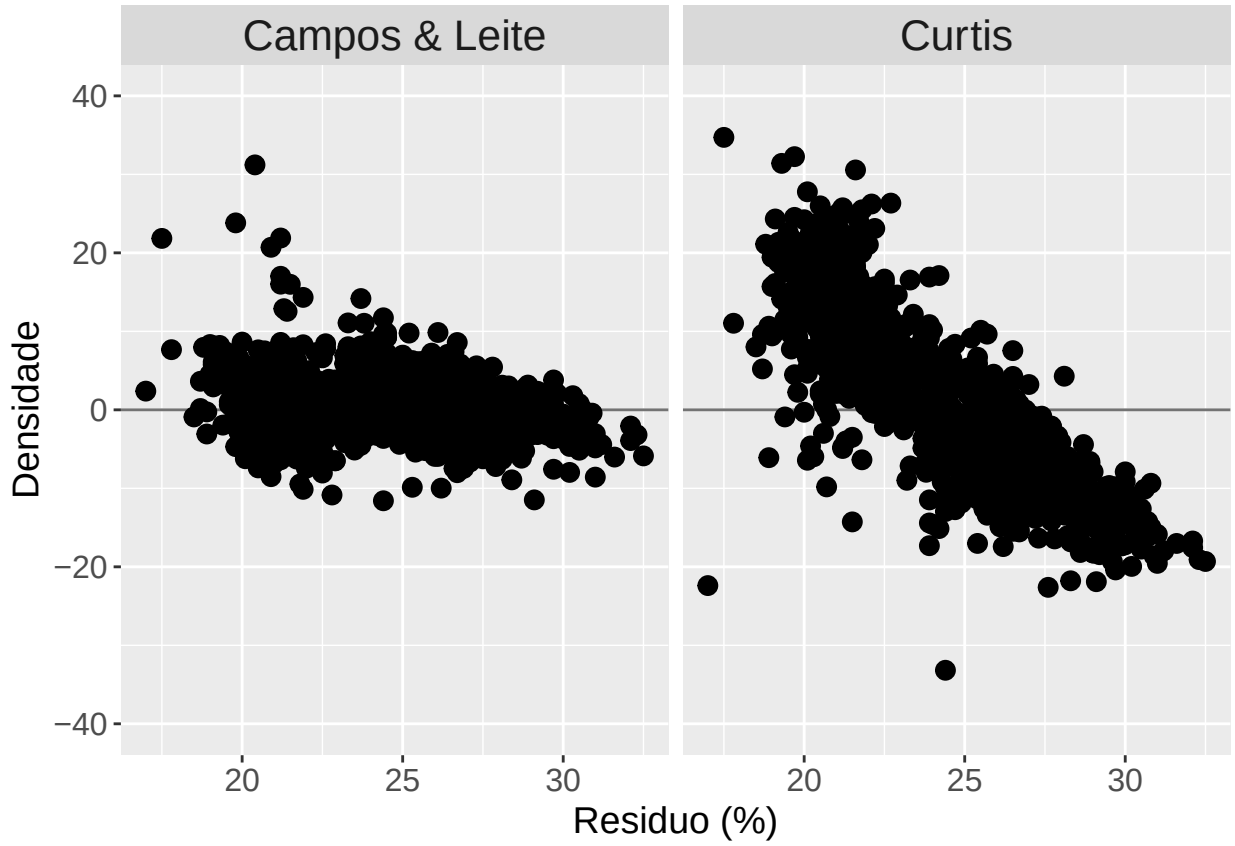
Agora tem-se uma variavel de HT_EST, uma variavel de RES, e uma variavel que diferencia os dois, dizendo qual dado pertence ao modelo 1 (curtis), e qual e do modelo 2 (Campos e Leite). Agora ja e possivel plotar o grafico, divindo os dados com base na variavel modelo.

7.1) Graficos comparando os modelos

7.2) Dispersao

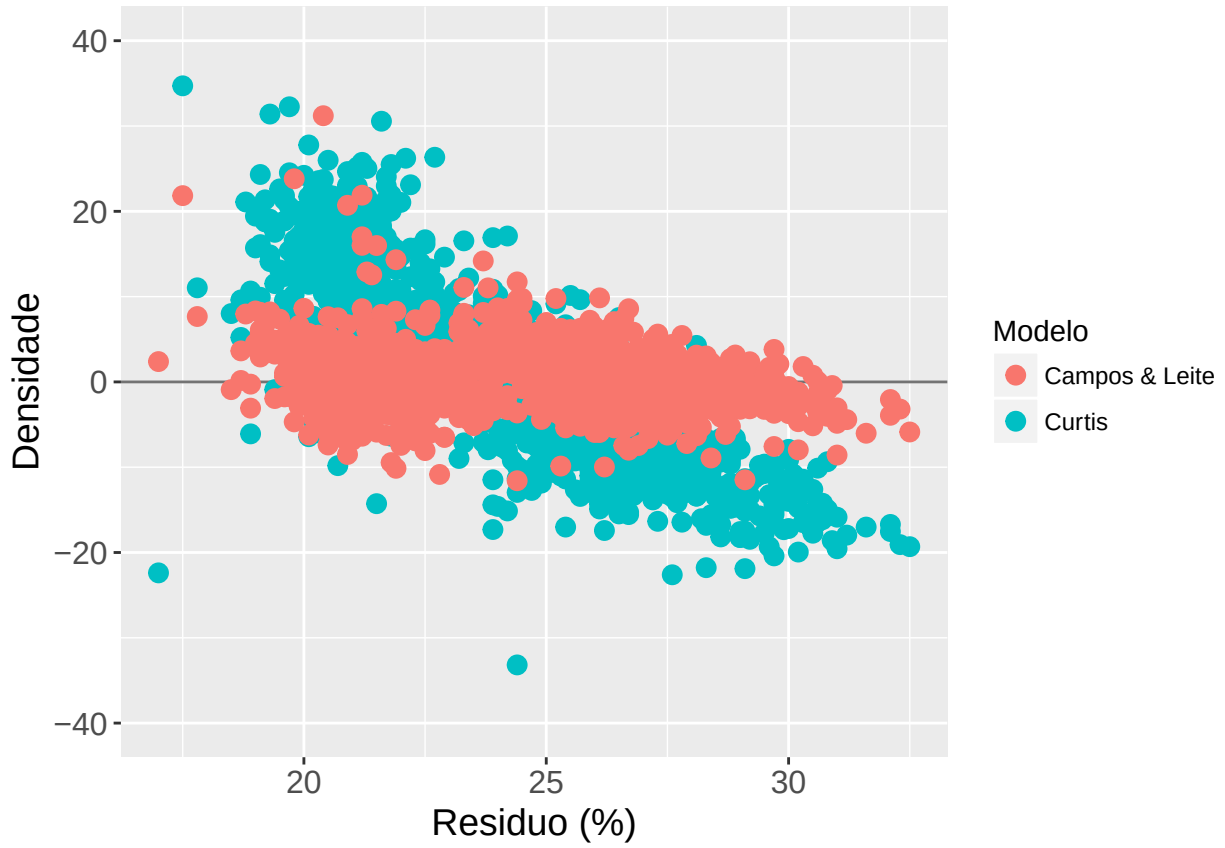
um grafico para cada ajuste

```
gf_disp_1 <- ggplot(dados_graph_bind, aes(HT, RES))  +
  geom_hline(yintercept = 0, colour = "gray45") +
  geom_point(size = 3)  +
  coord_cartesian(ylim = c(-40,40)) +
  facet_grid(~Modelo)+
  labs(x   = "Residuo (%)",
       y     = "Densidade" ) +
  theme( #com theme muda-se o tamanho e fonte da letra
    axis.title   = element_text(size = 14),
    axis.text    = element_text(size = 12),
    strip.text.x = element_text(size = 16))
gf_disp_1
```



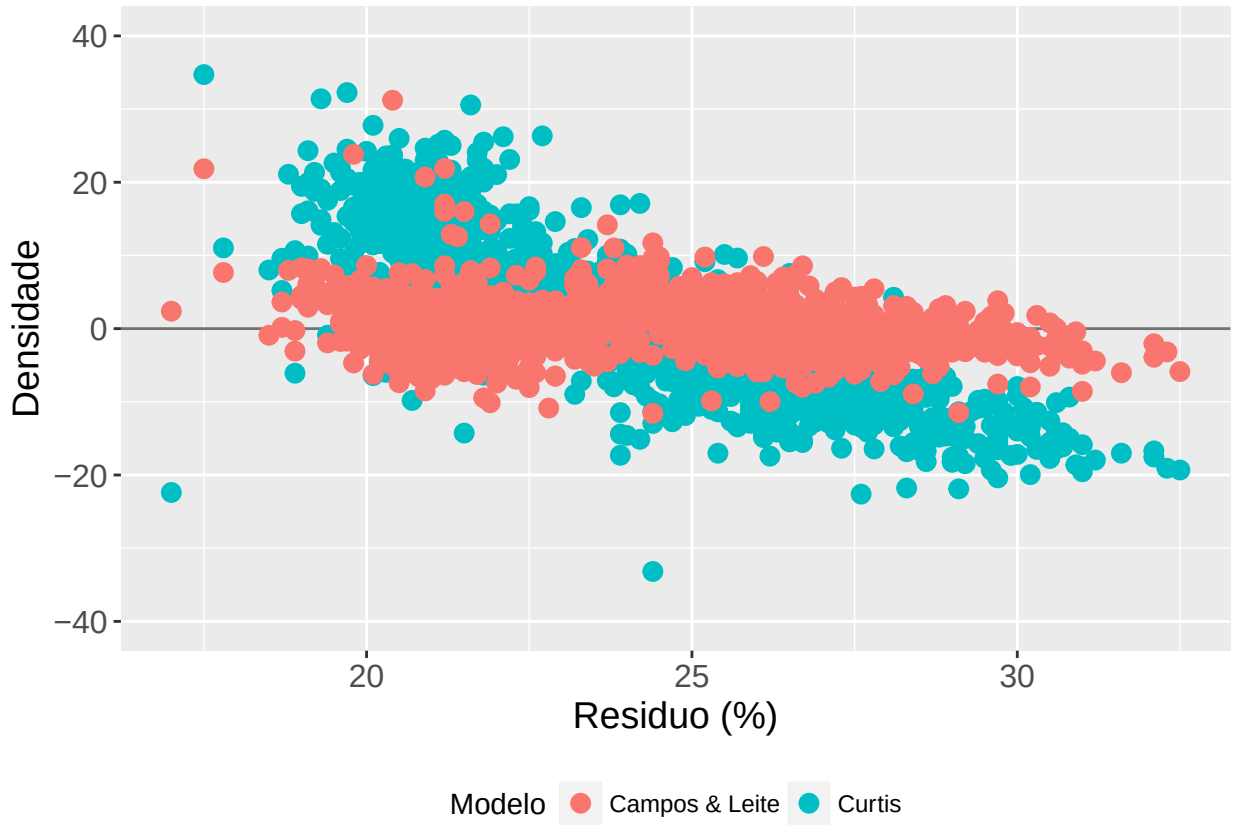
diferenciando por cor

```
gf_disp_2 <- ggplot(dados_graph_bind, aes(HT, RES))  +
  geom_hline(yintercept = 0, colour = "gray45") +
  geom_point(aes(color=Modelo),size = 3)  +
  coord_cartesian(ylim = c(-40,40)) +
  labs(x   = "Residuo (%)",
       y     = "Densidade" ) +
  theme( #com theme muda-se o tamanho e fonte da letra
    axis.title   = element_text(size = 14),
    axis.text    = element_text(size = 12),
    strip.text.x = element_text(size = 16))
gf_disp_2
```



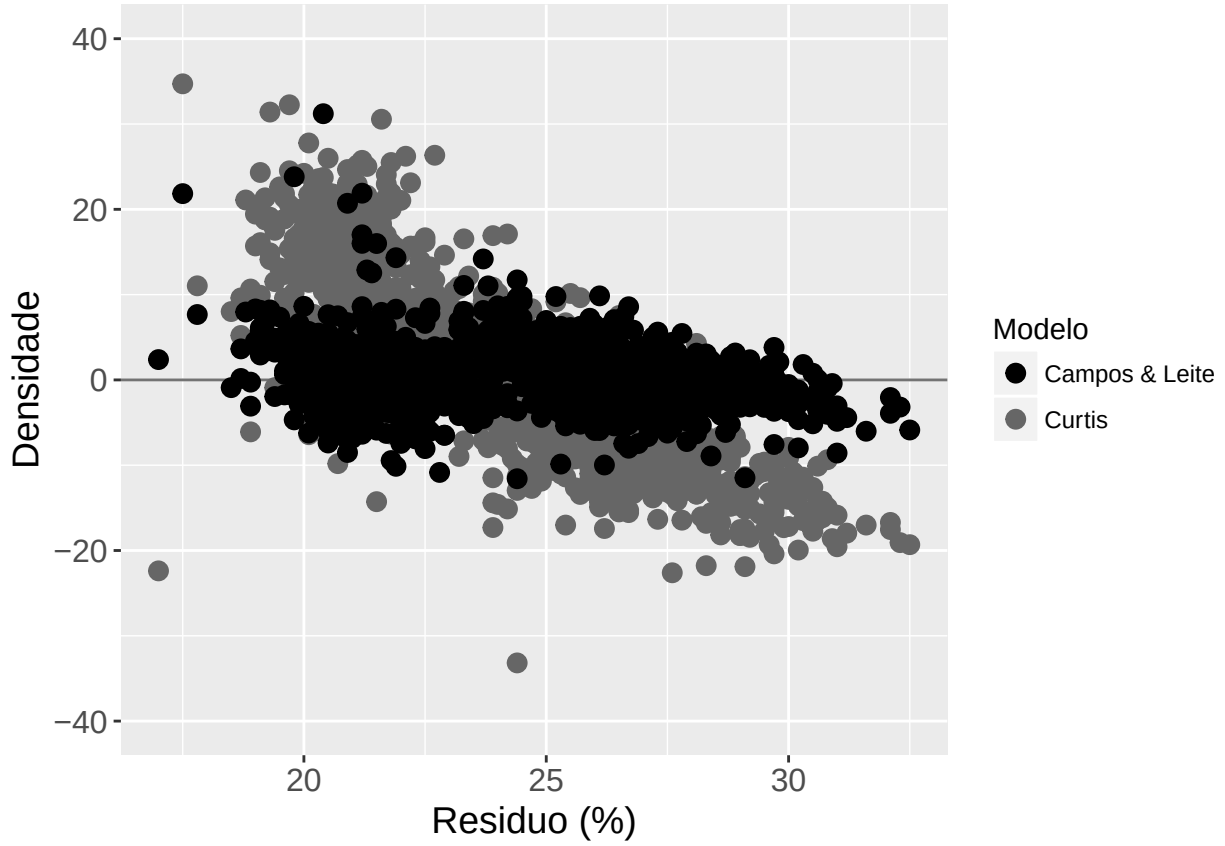
legenda em baixo

```
ggplot(dados_graph_bind, aes(HT, RES))  +
  geom_hline(yintercept = 0, colour = "gray45") +
  geom_point(aes(color=Modelo),size = 3)  +
  coord_cartesian(ylim = c(-40,40)) +
  labs(x    = "Residuo (%)",
       y    = "Densidade" ) +
  theme( #com theme muda-se o tamanho e fonte da letra
    axis.title    = element_text(size = 14),
    axis.text     = element_text(size = 12),
    strip.text.x = element_text(size = 16),
    legend.position="bottom")
```



diferenciando por cor (tons de cinza)

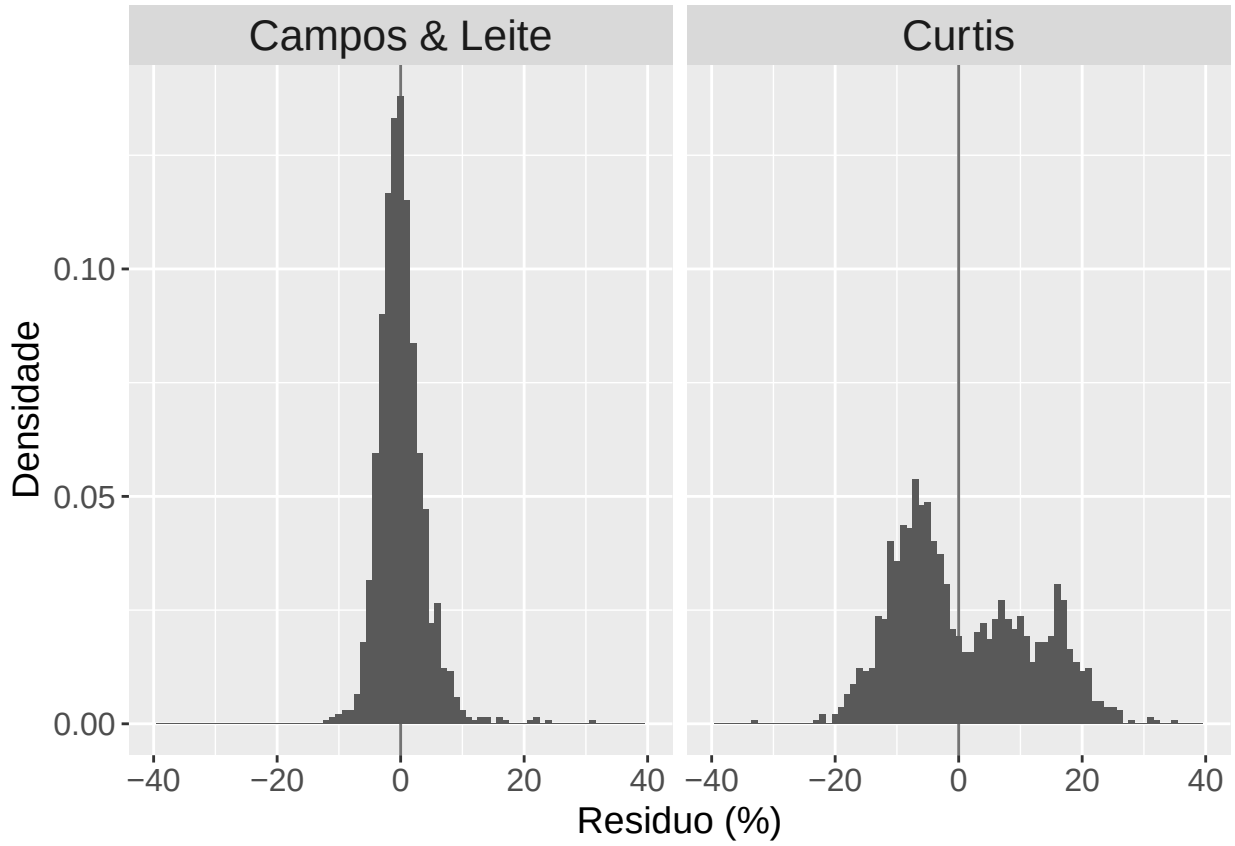
```
gf_disp_3 <- ggplot(dados_graph_bind, aes(HT, RES))  +
  geom_hline(yintercept = 0, colour = "gray45") +
  geom_point(aes(color=Modelo),size = 3)  +
  coord_cartesian(ylim = c(-40,40)) +
  scale_color_manual(values =c("black", "grey40") ) +
  labs(x    = "Residuo (%)",
       y     = "Densidade" ) +
  theme( #com theme muda-se o tamanho e fonte da letra
    axis.title    = element_text(size = 14),
    axis.text     = element_text(size = 12),
    strip.text.x = element_text(size = 16))
gf_disp_3
```



7.3) Histograma

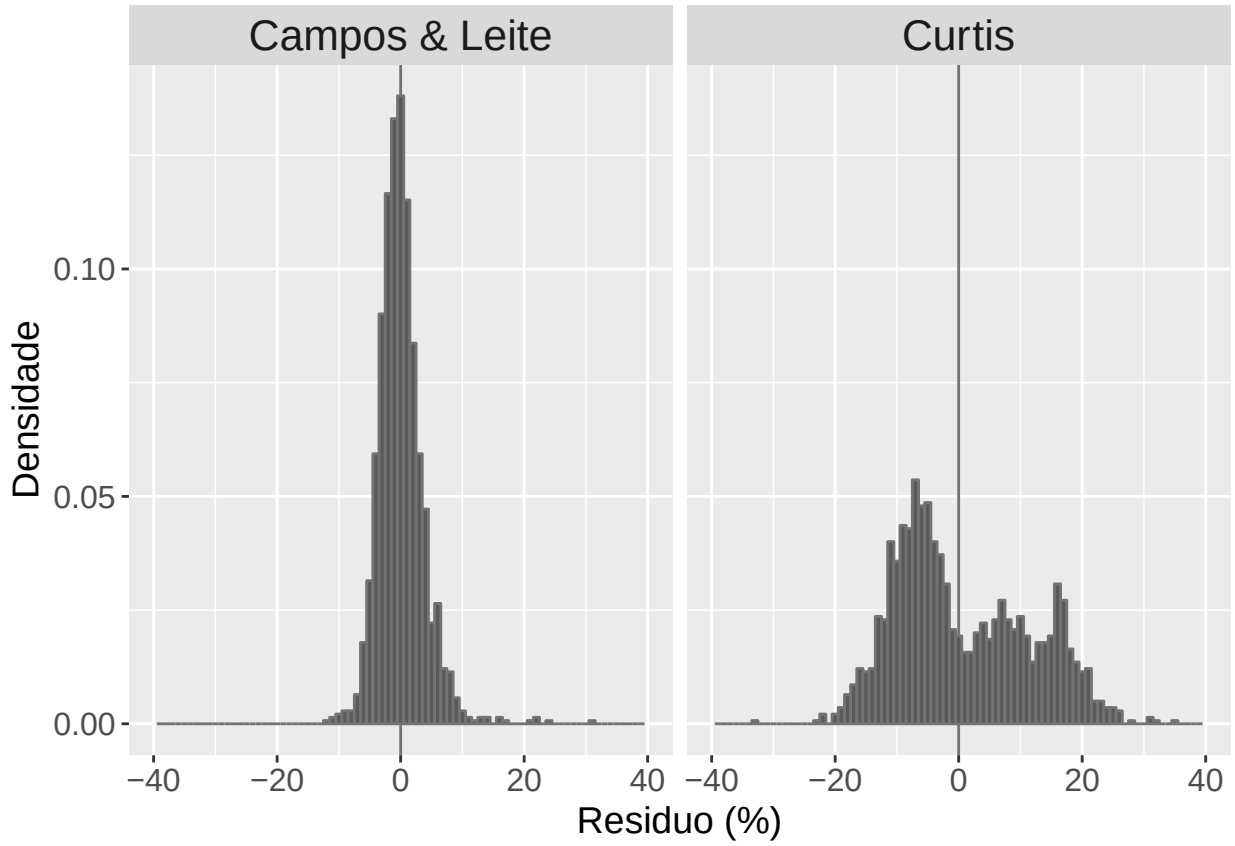
um grafico para cada modelo

```
gf_hist_1 <- ggplot(dados_graph_bind,aes(RES, ..density..) ) +
  geom_vline(xintercept = 0, colour = "gray45")  +
  geom_histogram(binwidth = 1)  + # intensidade da cor. varia entre 0 e 1
  xlim(-40, 40) +
  facet_wrap(~Modelo) +
  labs(x    = "Residuo (%)",
       y     = "Densidade" ) +
  theme( #com theme muda-se o tamanho e fonte da letra
    axis.title   = element_text(size = 14),
    axis.text    = element_text(size = 12),
    strip.text.x = element_text(size = 16))
gf_hist_1
```



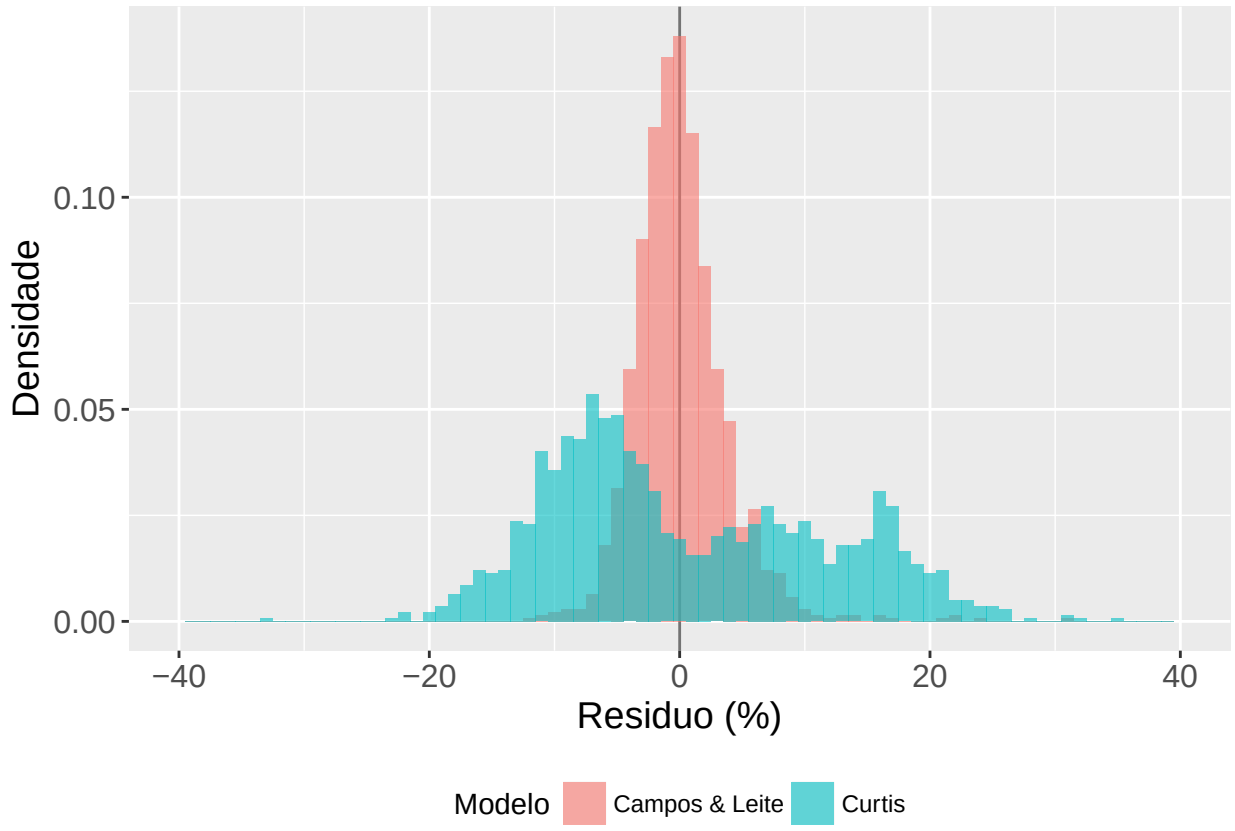
um grafico para cada modelo, com divisao entre colunas

```
gf_hist_2 <- ggplot(dados_graph_bind,aes(RES, ..density..) ) +
  geom_vline(xintercept = 0, colour = "gray45")  +
  geom_histogram(binwidth = 1, color = "gray45")  +
  xlim(-40, 40) +
  facet_wrap(~Modelo) +
  labs(x    = "Residuo (%)",
       y     = "Densidade" ) +
  theme( #com theme muda-se o tamanho e fonte da letra
    axis.title   = element_text(size = 14),
    axis.text    = element_text(size = 12),
    strip.text.x = element_text(size = 16))
gf_hist_2
```



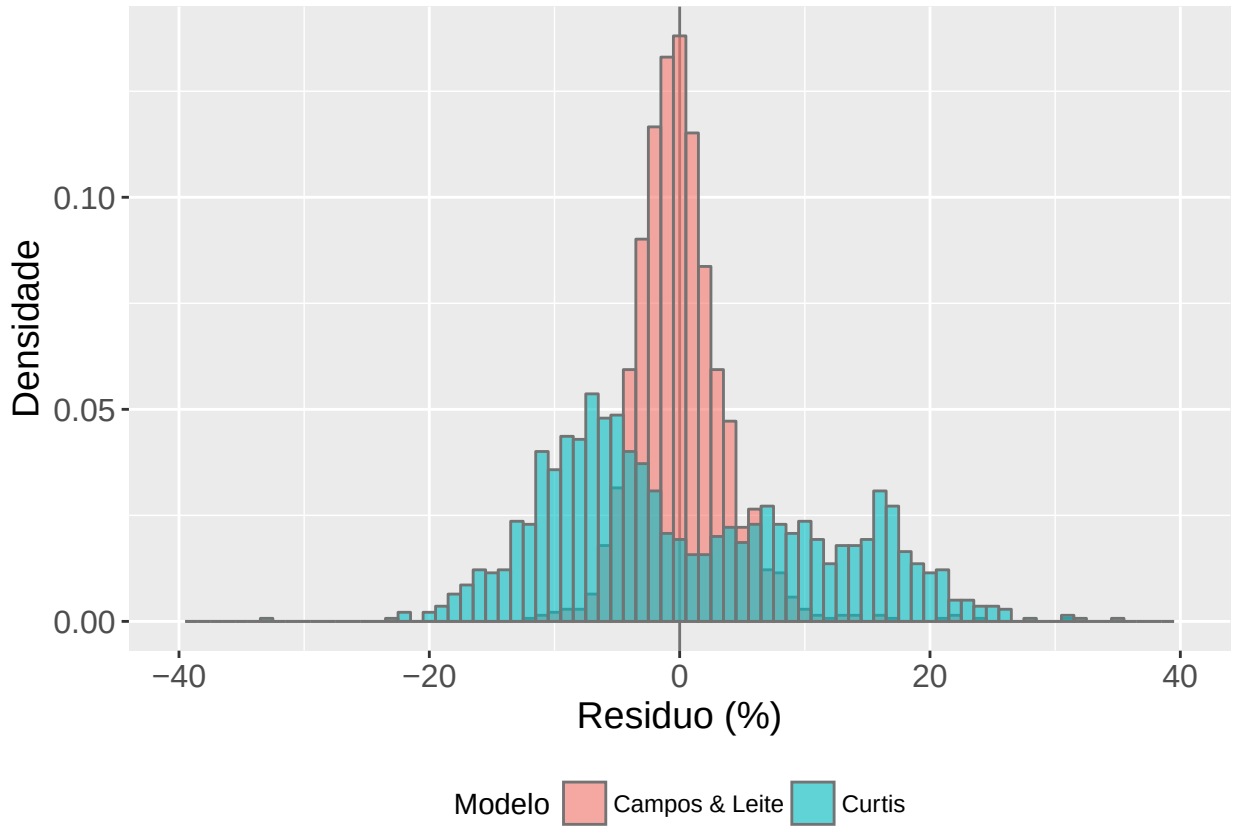
diferenciando por cor

```
gf_hist_3 <- ggplot(dados_graph_bind, aes(RES, ..density..) ) +
  geom_vline(xintercept = 0, colour = "gray45")  +
  geom_histogram(aes(fill = Modelo ),
                 binwidth = 1, # largura das colunas
                 position = "identity", # forca os histogramas a se sobreporem
                 alpha = .6)  + # intensidade da cor. varia entre 0 e 1
  xlim(-40, 40)  +
  labs(x   = "Residuo (%)",
       y     = "Densidade" ) +
  theme( #com theme muda-se o tamanho e fonte da letra
    axis.title   = element_text(size = 14),
    axis.text    = element_text(size = 12),
    strip.text.x = element_text(size = 16),
    legend.position="bottom")
gf_hist_3
```



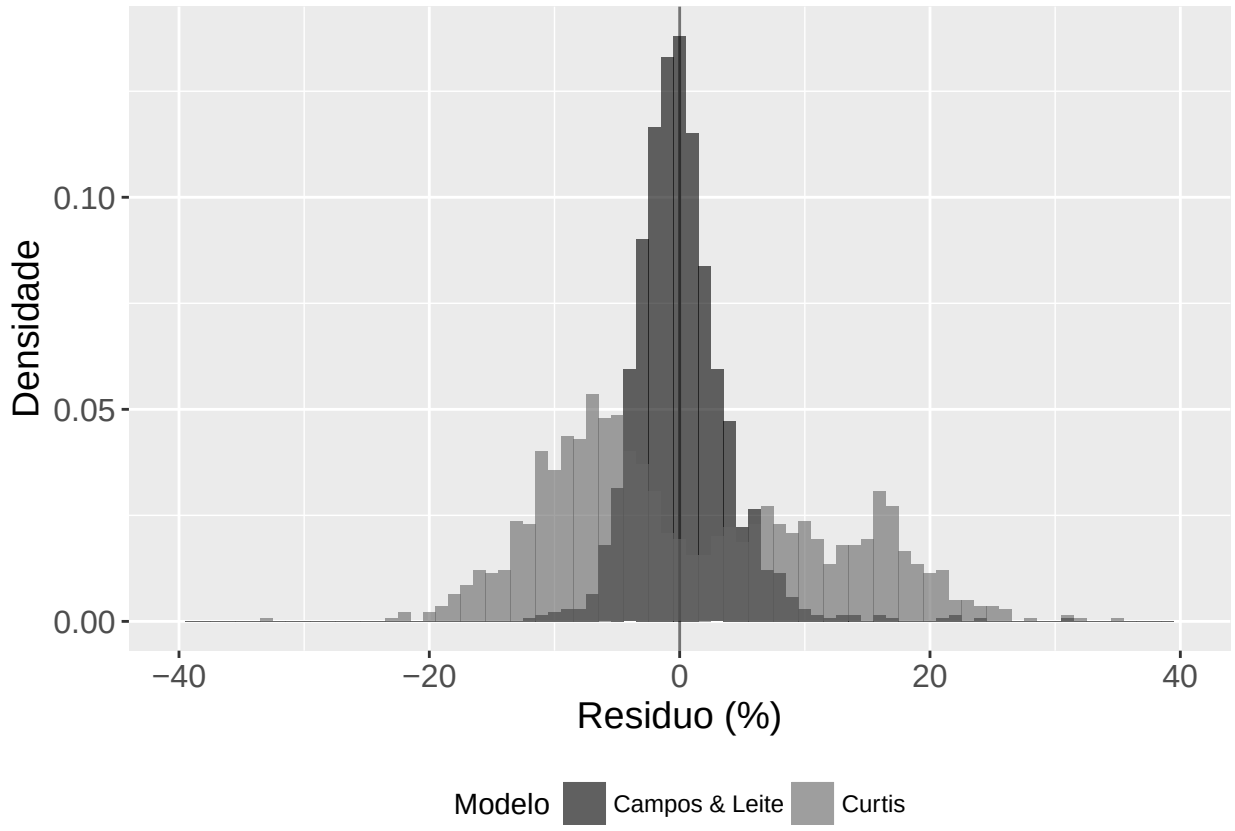
adicionar divisao entre as colunas, grafico colorido

```
gf_hist_4 <- ggplot(dados_graph_bind, aes(RES, ..density..) ) +
  geom_vline(xintercept = 0, colour = "gray45")  +
  geom_histogram(aes(fill = Modelo ),
                 color= "gray45", # adicionar divisao entre colunas
                 binwidth = 1, # largura das colunas
                 position = "identity", # forca os histogramas a se sobreporem
                 alpha = .6)  + # intensidade da cor. varia entre 0 e 1
  xlim(-40, 40)  +
  labs(x    = "Residuo (%)",
       y     = "Densidade" ) +
  theme( #com theme muda-se o tamanho e fonte da letra
    axis.title   = element_text(size = 14),
    axis.text    = element_text(size = 12),
    strip.text.x = element_text(size = 16),
    legend.position="bottom")
gf_hist_4
```



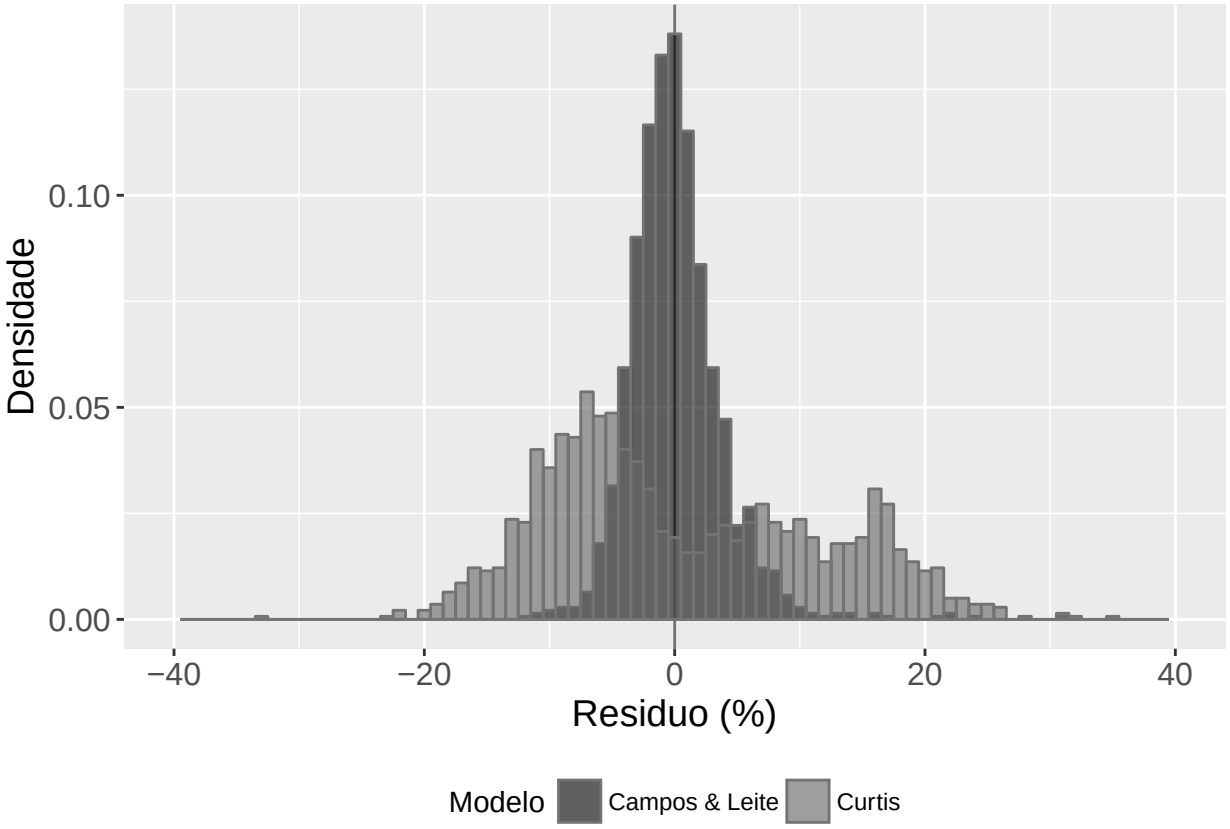
diferenciando por cor (tons de cinza)

```
gf_hist_5 <- ggplot(dados_graph_bind, aes(RES, ..density..) ) +
  geom_vline(xintercept = 0, colour = "gray45")  +
  geom_histogram(aes(fill = Modelo ),
                 binwidth = 1, # largura das colunas
                 position = "identity", # forca os histogramas a se sobreporem
                 alpha = .6)  + # intensidade da cor. varia entre 0 e 1
  xlim(-40, 40)  +
  scale_fill_manual(values =c("black", "grey40") )  +
  labs(x    = "Residuo (%)",
       y     = "Densidade" ) +
  theme( #com theme muda-se o tamanho e fonte da letra
    axis.title   = element_text(size = 14),
    axis.text    = element_text(size = 12),
    strip.text.x = element_text(size = 16),
    legend.position="bottom")
gf_hist_5
```



adicionar divisao entre as colunas, grafico em tons de cinza

```
gf_hist_6 <- ggplot(dados_graph_bind, aes(RES, ..density..) ) +
  geom_vline(xintercept = 0, colour = "gray45")  +
  geom_histogram(aes(fill = Modelo ),
                 color = "gray45", # adicionar divisao entre colunas
                 binwidth = 1, # largura das colunas
                 position = "identity", # forca os histogramas a se sobreporem
                 alpha = .6)  + # intensidade da cor. varia entre 0 e 1
  xlim(-40, 40)  +
  scale_fill_manual(values =c("black", "grey40") )  +
  labs(x    = "Residuo (%)",
       y     = "Densidade" ) +
  theme( #com theme muda-se o tamanho e fonte da letra
    axis.title    = element_text(size = 14),
    axis.text     = element_text(size = 12),
    strip.text.x = element_text(size = 16),
    legend.position="bottom")
gf_hist_6
```



8) Exportar Graficos

Exporta-se os graficos feitos com o ggplot com a funcao ggsave. Caso se deseje exportar o ultimo grafico plotado, basta rodar a funcao com o nome desejado e a extensao no final (recomendado: .png). Recomenda-se alterar a resolucao para 12 x 8 (polegadas):

```
ggsave("graficos/grafico_teste.png", width = 12, height = 8)
```

Como os graficos foram salvos em objetos, pode-se exporta-los chamando pelo nome do objeto, por exemplo:

```
ggsave("graficos/gf_disp_2.png",gf_disp_2, width = 12, height = 8)
ggsave("graficos/gf_disp_3.png",gf_disp_3, width = 12, height = 8)
ggsave("graficos/gf_hist_1.png",gf_hist_1, width = 12, height = 8)
ggsave("graficos/gf_hist_3.png",gf_hist_3, width = 12, height = 8)
ggsave("graficos/gf_hist_4.png",gf_hist_4, width = 12, height = 8)
```