Script para teste de identidade de modelo

Conforme descrito por Regazzi, 1993

Sollano Rabelo Braga
Marcio Leles Romarco de Oliveira

Janeiro, 2017

Contents

1) Carregar pacotes e dados

Primeiro carrega-se os pacotes que serão utilziados neste script.

O pacote readxl será utilizado para carregar os dados direto da planilha do excel, e o opacote ggplot2 para que se plote o gráfico ao final do teste.

Todos os calculos serão feitos utilizando apenas funções do R base.

Carrega-se os pacotes com a função library:

```
library(readxl)
library(ggplot2)
```

O exemplo que será utilizado neste script possui dados de dois projetos, Diamantina e Serro, e deseja-se saber se o comportamento do diametro é semelhante nas duas cidades.

Utilizando a função read_excel, carrega-se a planilha do excel em um objeto:

```
dados <- read_excel("dados.xlsx")
```

Visualiza-se os dados:

```
dados
```

```
## # A tibble: 24 x 4
##    PROJETO     N    N2   DAP
##      <chr> <dbl> <dbl> <dbl>
##  1    DTNA    20   400  11.3
##  2    DTNA    20   400  11.8
##  3    DTNA    20   400  12.8
##  4    DTNA    40  1600  14.5
##  5    DTNA    40  1600  14.7
##  6    DTNA    40  1600  14.9
##  7    DTNA    60  3600  15.0
##  8    DTNA    60  3600  15.4
##  9    DTNA    60  3600  15.8
## 10    DTNA    80  6400  14.7
## # ... with 14 more rows
```

2) Definição e ajuste do modelo reduzido

O modelo utilizado para este exemplo será o quadrático, com o diâmetro em função do nitrogenio (N).

A seguir ajusta-se o modelo reduzido, e salva-se o seu resultado em um objeto:

```
lm_redz <- lm(DAP ~  N + N2, dados)
summary(lm_redz)
```

```
##
## Call:
## lm(formula = DAP ~ N + N2, data = dados)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.08083 -0.58750  0.06417  0.54125  1.61917
##
## Coefficients:
```

```
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 10.1958333  1.0897049   9.357 6.13e-09 ***
## N            0.1907083  0.0497059   3.837 0.000959 ***
## N2          -0.0015729  0.0004893  -3.215 0.004159 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9588 on 21 degrees of freedom
## Multiple R-squared:  0.5426, Adjusted R-squared:  0.499
## F-statistic: 12.46 on 2 and 21 DF,  p-value: 0.0002712
```

3) Criação das variáveis Dummy

O número de dummies criadas é baseado no número comparações que serão feitas, que neste caso é o número de projetos. Como são dois projetos, serão duas dummies binárias, e duas dummies para cada variável do modelo reduzido, totalizando em 6 dummies.

Para isso é necessário que se tenha o registro dos níveis do fator utilizado, no caso, quais projetos serão comparados. A variável que possui essa informação é a variável PROJETO, e pode-se verificar os seus níveis com a função levels:

```
fator <- as.factor(dados$PROJETO)
```

Agora cria-se um objeto que contem os nomes dos projetos que serão comparados:

```
factor_levels <- levels(fator)
factor_levels
```

```
## [1] "DTNA"  "SERRO"
```

Pronto, agora a informação sobre os projetos foi salva. O próximo passo é a criação das dummies.

Primeiro serão criadas as dummies binárias, e em seguida as dummies que possuem as informações de N e N2.

3.1) Dummies binárias

A seguir cria-se as dummies binárias com um loop for. O loop vai de 1:2, pois existem dois projetos para comparação: Antes de realizar o loop, cria-se a lista que será utilizado:

```
lista1 <- list()

for(i in 1:2){

  lista1[[paste("D", i, sep = "")]] <- ifelse(dados$PROJETO == factor_levels[i], 1, 0 )
}
lista1
```

```
## $D1
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0
##
## $D2
##  [1] 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1
```

Agora basta converter a losta em uma matriz:

```
dummies1 <- do.call(cbind, lista1)
dummies1
```

```
##       D1 D2
##  [1,]  1  0
##  [2,]  1  0
##  [3,]  1  0
##  [4,]  1  0
##  [5,]  1  0
##  [6,]  1  0
##  [7,]  1  0
##  [8,]  1  0
##  [9,]  1  0
## [10,]  1  0
## [11,]  1  0
## [12,]  1  0
## [13,]  0  1
## [14,]  0  1
## [15,]  0  1
## [16,]  0  1
## [17,]  0  1
## [18,]  0  1
## [19,]  0  1
## [20,]  0  1
## [21,]  0  1
## [22,]  0  1
## [23,]  0  1
## [24,]  0  1
```

3.2) Dummies para as variáveis

Agora cria-se cria-se as variáveis dummies para cada variável do modelo reduzido (N e N2), repetindo o processo anterior.

Para proceder é necessária a criação de um objeto que possua os nomes dessas variáveis:

```
VARSX <- c("N", "N2")
```

Agora cria-se a lista vazia:

```
lista2 <- list()
```

Para que isso seja possível foi utilizado outro loop, um nível acima do anterior, ou seja, j representa as variáveis do modelo reduzido, e i representa os projetos:

```
for(j in 1:2){

  for(i in 1:2){

    lista2[[paste("D", i, VARSX[j],sep = "")]] <- ifelse(
      dados$PROJETO == factor_levels[i], dados[[   VARSX[j]   ]] ,  0  )
  }
}

lista2
```

```
## $D1N
##  [1] 20 20 20 40 40 40 60 60 60 80 80 80  0  0  0  0  0  0  0  0  0  0  0
## [24]  0
##
## $D2N
##  [1]  0  0  0  0  0  0  0  0  0  0  0  0  0 20 20 20 40 40 40 60 60 60 80 80
## [24] 80
##
## $D1N2
##  [1]  400  400  400 1600 1600 1600 3600 3600 3600 6400 6400 6400    0    0
## [15]    0    0    0    0    0    0    0    0    0    0
##
## $D2N2
##  [1]    0    0    0    0    0    0    0    0    0    0    0    0    0  400  400
## [15]  400 1600 1600 1600 3600 3600 3600 6400 6400 6400
```

Agora converte-se a lista em matriz:

```
dummies2 <- do.call(cbind, lista2)
dummies2
```

```
##       D1N D2N D1N2 D2N2
##  [1,]  20   0  400    0
##  [2,]  20   0  400    0
##  [3,]  20   0  400    0
##  [4,]  40   0 1600    0
##  [5,]  40   0 1600    0
##  [6,]  40   0 1600    0
##  [7,]  60   0 3600    0
##  [8,]  60   0 3600    0
##  [9,]  60   0 3600    0
## [10,]  80   0 6400    0
## [11,]  80   0 6400    0
## [12,]  80   0 6400    0
## [13,]   0  20    0  400
## [14,]   0  20    0  400
## [15,]   0  20    0  400
## [16,]   0  40    0 1600
## [17,]   0  40    0 1600
## [18,]   0  40    0 1600
## [19,]   0  60    0 3600
## [20,]   0  60    0 3600
## [21,]   0  60    0 3600
## [22,]   0  80    0 6400
## [23,]   0  80    0 6400
## [24,]   0  80    0 6400
```

E por fim une-se as variáveis dummies criadas em um único objeto:

```
dummies_f <- data.frame(dummies1, dummies2)
dummies_f
```

```
##    D1 D2 D1N D2N D1N2 D2N2
## 1   1  0  20   0  400    0
## 2   1  0  20   0  400    0
## 3   1  0  20   0  400    0
## 4   1  0  40   0 1600    0
```

```
## 5   1  0  40   0 1600    0
## 6   1  0  40   0 1600    0
## 7   1  0  60   0 3600    0
## 8   1  0  60   0 3600    0
## 9   1  0  60   0 3600    0
## 10  1  0  80   0 6400    0
## 11  1  0  80   0 6400    0
## 12  1  0  80   0 6400    0
## 13  0  1   0  20    0  400
## 14  0  1   0  20    0  400
## 15  0  1   0  20    0  400
## 16  0  1   0  40    0 1600
## 17  0  1   0  40    0 1600
## 18  0  1   0  40    0 1600
## 19  0  1   0  60    0 3600
## 20  0  1   0  60    0 3600
## 21  0  1   0  60    0 3600
## 22  0  1   0  80    0 6400
## 23  0  1   0  80    0 6400
## 24  0  1   0  80    0 6400
```

Agora une-se as dummies aos dados originais:

```
dados_comp <- cbind(dados, dummies_f)
dados_comp
```

```
##    PROJETO  N   N2  DAP D1 D2 D1N D2N D1N2 D2N2
## 1     DTNA 20  400 11.3  1  0  20   0  400    0
## 2     DTNA 20  400 11.8  1  0  20   0  400    0
## 3     DTNA 20  400 12.8  1  0  20   0  400    0
## 4     DTNA 40 1600 14.5  1  0  40   0 1600    0
## 5     DTNA 40 1600 14.7  1  0  40   0 1600    0
## 6     DTNA 40 1600 14.9  1  0  40   0 1600    0
## 7     DTNA 60 3600 15.0  1  0  60   0 3600    0
## 8     DTNA 60 3600 15.4  1  0  60   0 3600    0
## 9     DTNA 60 3600 15.8  1  0  60   0 3600    0
## 10    DTNA 80 6400 14.7  1  0  80   0 6400    0
## 11    DTNA 80 6400 15.5  1  0  80   0 6400    0
## 12    DTNA 80 6400 15.7  1  0  80   0 6400    0
## 13   SERRO 20  400 14.4  0  1   0  20    0  400
## 14   SERRO 20  400 14.6  0  1   0  20    0  400
## 15   SERRO 20  400 15.0  0  1   0  20    0  400
## 16   SERRO 40 1600 15.9  0  1   0  40    0 1600
## 17   SERRO 40 1600 16.3  0  1   0  40    0 1600
## 18   SERRO 40 1600 16.7  0  1   0  40    0 1600
## 19   SERRO 60 3600 15.8  0  1   0  60    0 3600
## 20   SERRO 60 3600 16.2  0  1   0  60    0 3600
## 21   SERRO 60 3600 16.5  0  1   0  60    0 3600
## 22   SERRO 80 6400 15.4  0  1   0  80    0 6400
## 23   SERRO 80 6400 15.5  0  1   0  80    0 6400
## 24   SERRO 80 6400 15.9  0  1   0  80    0 6400
```

```
names_dummies <- names(dummies_f)
names_dummies
```

```
## [1] "D1"   "D2"   "D1N"  "D2N"  "D1N2" "D2N2"
```

4) Ajuste do modelo completo

Agora realiza-se o ajuste do modelo completo, composto pelas variáveis dummies.

É utilizado o -1 ao final do modelo, para que se anule o b0:

```
lm_comp <- lm( DAP ~ D1 + D2 + D1N + D2N + D1N2 + D2N2 - 1 , dados_comp)

summary(lm_comp)
```

```
##
## Call:
## lm(formula = DAP ~ D1 + D2 + D1N + D2N + D1N2 + D2N2 - 1, data = dados_comp)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.72833 -0.20708 -0.02417  0.26292  0.77167
##
## Coefficients:
##        Estimate Std. Error t value Pr(>|t|)
## D1    8.1250000  0.7102420  11.440 1.09e-09 ***
## D2   12.2666667  0.7102420  17.271 1.19e-12 ***
## D1N   0.2305833  0.0323970   7.117 1.24e-06 ***
## D2N   0.1508333  0.0323970   4.656 0.000197 ***
## D1N2 -0.0017708  0.0003189  -5.553 2.85e-05 ***
## D2N2 -0.0013750  0.0003189  -4.312 0.000420 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4419 on 18 degrees of freedom
## Multiple R-squared:  0.9994, Adjusted R-squared:  0.9991
## F-statistic:  4650 on 6 and 18 DF,  p-value: < 2.2e-16
```

5) Criação da Anova

Agora cria=se os objetos necessários para a construção da anova: Correção:

```
C <- (sum(dados$DAP))^2/nrow(dados)
```

Os graus de liberdade serão retirados com base no número de variáveis no lado x dos modelos:

Graus de liberdade do modelo completo:

```
gl_comp <- ncol(dummies_f)
```

Graus de liberdade do modelo reduzido:

```
gl_redz <- length(VARSX)
```

Graus de liberdade da redução:

```
gl_reducao <- gl_comp - gl_redz
```

Graus de liberdade do resíduo:

```
gl_residuo <- lm_comp$df.residual
```

Soma de quadrado de parâmetros (modelo completo):

```r
SQParamC <- sum(lm_comp$fitted.values^2)
```

Soma de quadrado da regressão do modelo reduzido:

```r
SQParamR <- sum(lm_redz$fitted.values^2) # + C
```

Soma de quadrado da redução:

```r
SQ_reducao <- SQParamC - SQParamR
```

Soma de quadrado dos resíduos modelo completo:

```r
SQRes_comp <- sum(lm_comp$residuals^2)
```

Quadrado médio de Parametro modelo Completo:

```r
QMParamC <- SQParamC/gl_comp
```

Quadrado médio de Parametro modelo Completo:

```r
QMParamR <- SQParamR/gl_redz
```

Quadrado médio da redução:

```r
QMReducao <- round(SQ_reducao / gl_reducao, 4)
```

Quadrado médio do resíduo:

```r
QMResiduo <- round(SQRes_comp / gl_residuo, 4)
```

Calculo do F:

```r
F_regazzi <- round(QMReducao / QMResiduo, 2)
```

Calculo do F crítico:

```r
F_tabelado <- round(qf(p = 0.05, df1 = gl_reducao , df2 = gl_residuo, lower.tail = F ), 2)
```

Cálculo do p-valor:

```r
p_valor <- pf(F_regazzi , df1 = gl_reducao , df2 = gl_residuo, lower=F)
```

Resultado do teste:

```r
resultado <- ifelse(p_valor < 0.05, "*", "ns")
```

Agora basta unir tudo em um data frame:

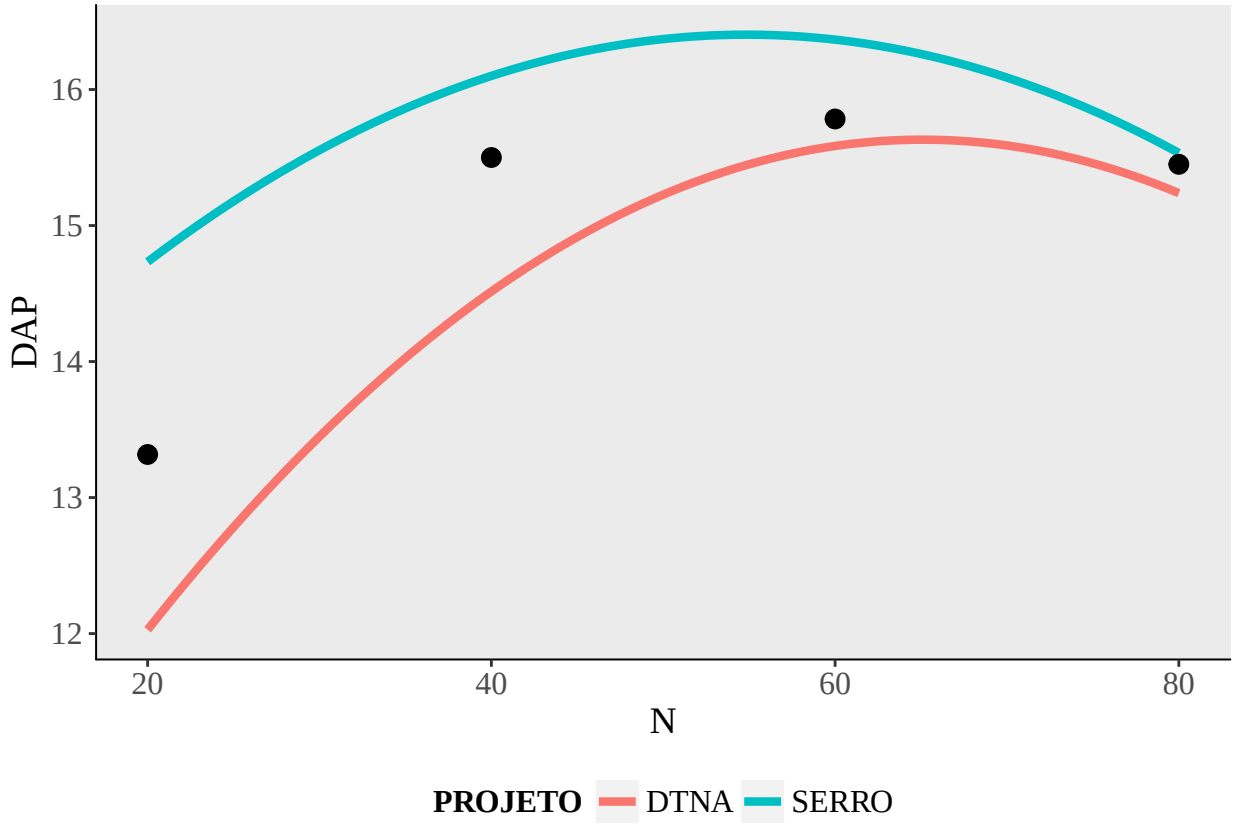
```r
tabela_regazzi <- data.frame(

  FV = c("Parametro_c", "Parametro_r", "Reducao", "Residuo"),
  GL = c(gl_comp, gl_redz, gl_reducao, gl_residuo ),
  SQ = round(c(SQParamC, SQParamR, SQ_reducao, SQRes_comp), 2),
  QM = c(QMParamC, QMParamC, QMReducao, QMResiduo ),
  F_Regazzi = c("","", F_regazzi ,""),
  F_tabelado = c("","", F_tabelado,""),
  p.valor = c("", "", signif(p_valor, 3),  ""),
  Resultado = c("", "", resultado,  "")
)
tabela_regazzi
```

```
##            FV GL      SQ       QM F_Regazzi F_tabelado  p.valor Resultado
## 1 Parametro_c  6 5447.70 907.9492
```

```
## 2 Parametro_r  2 5431.90 907.9492
## 3     Reducao  4   15.79   3.9477     20.21       2.93 1.84e-06         *
## 4     Residuo 18    3.51   0.1953
```

Para realizar o gráfico, utiliza-se a função ggplot:

```
ggplot(dados, aes(N, DAP) ) +
  geom_smooth(method = "lm",formula = y ~ poly(x, 2, raw=T) ,aes(color=PROJETO), se = F, size = 1.5) +
  stat_summary(fun.y = mean, geom = "point", size = 3) +
  theme_gray(base_family = "serif") +
  ggplot2::theme(
    legend.position = "bottom",
    panel.grid.major = ggplot2::element_blank(),
    panel.grid.minor = ggplot2::element_blank(),
    panel.border = ggplot2::element_blank(),
    axis.line.x = ggplot2::element_line(color="black"),
    axis.line.y = ggplot2::element_line(color="black"),
    legend.title    = ggplot2::element_text(size = 12, face = "bold"),
    legend.text     = ggplot2::element_text(size = 12),
    axis.title      = ggplot2::element_text(size = 14),
    axis.text       = ggplot2::element_text(size = 12) )
```



6) Citação

Para citar esse script, utilize: (BRAGA; OLIVEIRA, 2017)

Rotina para realização do teste de identidade de modelo. 2017. Disponível em: https://github.com/sollano/