Econometria I (2022) — FEA-RP/USP

Professor: Daniel Domingues dos Santos — Monitor: Fábio Hideki Nishida

---

Lista Prática 4/5

Para os exercícios 1 a 3, usaremos a base de dados `RiceFarms` do pacote `splm` que tem informações sobre a produção de arroz na Indonésia. Você pode carregar a base usando:

```r
# install.packages("splm")
data("RiceFarms", package = "splm")
```

É um painel com 1.026 observações, sendo $N = 171$ agricultores (`id`) em $T = 6$ períodos distintos (`time`). As variáveis relevantes são:

- `goutput`: output bruto de arroz em kg
- `seed`: sementes em kg
- `totlabor`: total de trabalhadores
- `size`: tamanho da plantação

Considere o seguinte modelo

$$\ln(\text{goutput}_{i,t}) = \alpha + \beta_1 \ln(\text{seed}_{i,t}) + \beta_2 \ln(\text{totlabor}_{i,t}) + \beta_3 \ln(\text{size}_{i,t}) + u_i + \nu_{i,t}$$

Exercício 1.

a) Usando a função **`plm()`** *do pacote* **`plm`***, estime os modelos pelo estimador between, pooled OLS, GLS (efeitos aleatórios), e estimador within (efeitos fixos).*

b) Realize os testes de Breusch-Pagan/LM e de Hausman usando, respectivamente, as funções **`plmtest()`** *e* **`phtest()`***. Qual dos quatro estimadores utilizados no item (a) é mais adequado de acordo com os resultados dos testes?*

Resposta:

a)

```r
library(dplyr)
library(plm)

# Carregando base de dados e transformando em pdata.frame
data("RiceFarms", package = "splm")
RF = pdata.frame(RiceFarms, index = c("id", "time"))

# Definindo modelos a serem estimados
models = c("within", "random", "pooling", "between")

# Estimando modelos via loop e atribuindo a objetos com nomes "variáveis" via
   assign()
for (x in models) {
  assign(paste0("RF.", x), plm(log(goutput) ~ log(seed) + log(totlabor) + log(
    size), RF, model=x))
}

# Retornando as estimativas em tabela única
var_comuns = intersect(names(RF.within$coef), names(RF.random$coef))
rbind(within=RF.within$coef[var_comuns], random=RF.random$coef[var_comuns],
      pooled=RF.pooling$coef[var_comuns], between=RF.between$coef[var_comuns])
```

```
        log(seed) log(totlabor) log(size)
within  0.2095572     0.2891663 0.5023701
random  0.2199071     0.2855146 0.5278612
pooled  0.2220898     0.2842064 0.5333761
between 0.2280801     0.2766866 0.5506479
```

b)

```
# Realizando testes de Breusch-Pagan (Honda)
plmtest(RF.pooling, effect="individual")
```

```
Lagrange Multiplier Test - (Honda) for balanced panels
data:  log(goutput) ~ log(seed) + log(totlabor) + log(size)
normal = 4.8396, p-value = 6.507e-07
alternative hypothesis: significant effects
```

Rejeitamos $H_0 : \sigma_u^2 = 0$ e, portanto, devemos considerar a presença de efeitos fixos $\Longrightarrow$ os modelos *between* e *pooled OLS* não são consistentes.

```
# Realizando teste de Hausman
phtest(RF.within, RF.random)
```

```
Hausman Test
data:  log(goutput) ~ log(seed) + log(totlabor) + log(size)
chisq = 3.775, df = 3, p-value = 0.2868
alternative hypothesis: one model is inconsistent
```

Não rejeitamos H_0: $E(u|X) = 0$ (termo de erro individual é não-correlacionada com as covariadas) e, portanto, consideramos que ambos estimadores de *GLS* e *within* são consistentes $\Longrightarrow$ como *GLS* usa variações inter e intra-indivíduos para sua estimação, ele é mais eficiente e mais adequado do que *within* para este modelo/dados.

$\square$

Exercício 2.

a) *Estime analiticamente o modelo por estimador within (efeitos fixos).*

b) *Estime analiticamente o modelo por GLS (efeitos aleatórios), calculando as variâncias dos termos de erro pelo método de Swamy e Arora (1972).*

Resposta:

a) Note que a estimação analítica pode ser feita usando as fórmulas do *within* mostradas na seção 2.2.3 de Croissaint & Millo (2018) ou, também, transformando as variáveis via pré-multiplicação das variáveis pela matriz de transformação W e resolvendo por OLS. Aqui é importante prestar atenção nos graus de liberdade do estimador.

```
RiceFarms = RiceFarms %>% mutate(
    # constant = 1, # within elimina o vetor de 1's
    ln_goutput = log(goutput),
    ln_seed = log(seed),
    ln_totlabor = log(totlabor),
    ln_size = log(size)
)

```

```r
y = RiceFarms %>% select(ln_goutput) %>% as.matrix()
X = RiceFarms %>% select(ln_seed, ln_totlabor, ln_size) %>% as.matrix()
# Z = cbind(RiceFarms$constant, X) # within elimina o vetor de 1's

N = RiceFarms %>% select(id) %>% unique() %>% nrow()
T = RiceFarms %>% select(time) %>% unique() %>% nrow()
iota_T = rep(1, T)

# Calculando matrizes de tranformação B e W
B = diag(N) %x% (iota_T %*% solve(t(iota_T) %*% iota_T) %*% t(iota_T))
W = diag(N*T) - B

# vetor de estimativas gamma_hat = (alpha, beta)
gamma_hat = solve(t(X) %*% W %*% X) %*% t(X) %*% W %*% y

# valores ajustados e erros
y_hat = X %*% gamma_hat
e_hat = y - y_hat

## Estimando variancia do termo de erro
sigma2_v = t(e_hat) %*% W %*% e_hat / (N*T - ncol(X) - N) # NT - K - N g.l.

## Estimando a matriz de variancia/covariancia das estimativas gamma
vcov_hat = c(sigma2_v) * solve(t(X) %*% W %*% X)

## Calculando erros padrao das estimativas gamma
std_error = sqrt(diag(vcov_hat)) # Raiz da diagonal da matriz de covariâncias

## Calculando estatisticas t das estimativas gamma
t_stat = gamma_hat / std_error

## Calculando p-valores das estimativas gamma
p_value = 2 * pt(q = -abs(t_stat), df = N*T - ncol(X) - N) # NT - K - N g.l.

## Organizando os resultados da regressao em uma matriz
results = cbind(gamma_hat, std_error, t_stat, p_value)

## Nomeando as colunas da matriz de resultados
colnames(results) = c("Estimate", "Std. Error", "t stat", "Pr(>|t|)")
results
```

```
             Estimate Std. Error    t stat     Pr(>|t|)
ln_seed     0.2095572 0.03182723  6.584212 7.989993e-11
ln_totlabor 0.2891663 0.03549156  8.147465 1.319048e-15
ln_size     0.5023701 0.03801667 13.214469 2.114140e-36
```

Comparando com os resultados obtidos no Exercício 1:

```r
summary(RF.within)$coef # comparando com estimado via plm()
```

```
               Estimate Std. Error   t-value     Pr(>|t|)
log(seed)     0.2095572 0.03182723  6.584212 7.989993e-11
log(totlabor) 0.2891663 0.03549156  8.147465 1.319048e-15
log(size)     0.5023701 0.03801667 13.214469 2.114140e-36
```

b) Lembre-se que, na estimação GLS (efeitos aletórios), usamos resíduos de outro estimador para calcular $\hat{\sigma}_\nu$, $\hat{\sigma}_u$ e, consequentemente, $\hat{\Omega}$.

Para usar o método de Swamy e Arora (1972) precisamos obter os erros das estimações *within* e *between* para calcular:

$$\hat{\sigma}_l^2 = \frac{\hat{\varepsilon}_B' B \hat{\varepsilon}_B}{N-K-1}, \qquad \hat{\sigma}_\nu^2 = \frac{\hat{\varepsilon}_W' W \hat{\varepsilon}_W}{N(T-1)-K} \qquad \text{e} \qquad \hat{\sigma}_u = \frac{\hat{\sigma}_l^2 - \hat{\sigma}_\nu^2}{T}$$

```r
# obtendo estimativas within e between
gamma_hat_w = solve(t(X) %*% W %*% X) %*% t(X) %*% W %*% y
gamma_hat_b = solve(t(Z) %*% B %*% Z) %*% t(Z) %*% B %*% y

# obtendo os valores ajustados e resíduos within e between
y_hat_w = X %*% gamma_hat_w
e_w = y - y_hat_w

y_hat_b = Z %*% gamma_hat_b
e_b = y - y_hat_b

# Calculando os termos de erro
sigma2_l = (t(e_b) %*% B %*% e_b) / (N - ncol(X) - 1)
sigma2_v = (t(e_w) %*% W %*% e_w) / (N * (T-1) - ncol(X))
sigma2_u = (sigma2_l + sigma2_v) / T

sigmas2 = cbind(sigma2_l, sigma2_v, sigma2_u)
colnames(sigmas2) = c("sigma2_l", "sigma2_v", "sigma2_u")
sigmas2
```

```
 sigma2_l  sigma2_v  sigma2_u
0.2204066 0.1323742 0.0587968
```

Agora, calculamos Ω^{-1} e para obter

$$\hat{\gamma}_{GLS} = (Z'\Omega^{-1}Z)^{-1}(Z'\Omega^{-1}y) = \qquad \text{e} \qquad V(\hat{\gamma}_{GLS}) = (Z'\Omega^{-1}Z)^{-1}$$

```r
# Calculando \Omega^{-1}
Omega_1 = c(sigma2_l^(-1)) * B + c(sigma2_v^(-1)) * W
dim(Omega_1) # NT x NT

# vetor de estimativas gamma_hat = (alpha, beta)
gamma_hat = solve(t(Z) %*% Omega_1 %*% Z) %*% (t(Z) %*% Omega_1 %*% y)
gamma_hat

## Estimando a matriz de variancia/covariancia das estimativas gamma
vcov_hat = solve(t(Z) %*% Omega_1 %*% Z)

## Calculando erros padrao das estimativas gamma
std_err = sqrt(diag(vcov_hat)) # Raiz da diagonal da matriz de covariâncias

## Calculando estatisticas t das estimativas gamma
t_stat = gamma_hat / std_err

## Calculando p-valores das estimativas gamma
p_value = 2 * pt(q = -abs(t_stat), df = nrow(Z) - ncol(Z))  # NT - K - 1 g.l.

## Organizando os resultados da regressao em uma matriz
results_swar = cbind(gamma_hat, std_err, t_stat, p_value)

```

```r
## Nomeando as colunas da matriz de resultados
colnames(results_swar) = c("Estimate", "Std. Error", "t stat", "Pr(>|t|)")
rownames(results_swar) = c("(Intercept)", "ln_seed", "ln_totlabor", "ln_size")
results_swar
```

```
             Estimate Std. Error    t stat      Pr(>|t|)
(Intercept) 5.3123104 0.20415280 26.021247 6.068667e-115
ln_seed     0.2199071 0.02837609  7.749733  2.214702e-14
ln_totlabor 0.2855146 0.03110824  9.178101  2.366789e-19
ln_size     0.5278612 0.03268695 16.148989  2.010632e-52
```

Comparando com os resultados obtidos no Exercício 1:

```r
summary(RF.random)$coef # comparando com estimado via plm()
```

```
               Estimate Std. Error   z-value      Pr(>|z|)
(Intercept)   5.3123104 0.20422998 26.011413 3.678718e-149
log(seed)     0.2199071 0.02838682  7.746804  9.423402e-15
log(totlabor) 0.2855146 0.03112000  9.174632  4.531184e-20
log(size)     0.5278612 0.03269931 16.142887  1.274387e-58
```

□

Exercício 3. *Estime numericamente o modelo por máxima verossimilhança (FIML) usando os seguintes passos:*

(1) Chutar valores iniciais para as estimativas $\hat{\gamma}$ (pode usar tudo igual a 0)

(2) Mostrar a iteração e as estimativas atuais de α e β's

(3) Obter $\hat{\varepsilon} = y - \hat{y}$ e calcular $\hat{\phi}^2$ usando

$$\hat{\phi}^2 = \frac{\hat{\varepsilon}' W \hat{\varepsilon}}{(T-1)\hat{\varepsilon}' B \hat{\varepsilon}} \tag{3.14}$$

(4) Calcular $\hat{\sigma}^2_\nu$ usando

$$\hat{\sigma}^2_\nu = \frac{\hat{\varepsilon}' W \hat{\varepsilon} + \hat{\phi}^2 \hat{\varepsilon}' B \hat{\varepsilon}}{NT} \tag{3.13}$$

(5) Obter $\hat{\Omega}^{-1/2}$ a partir da relação

$$\hat{\Omega}^{-1/2} = \frac{1}{\hat{\sigma}_l} B + \frac{1}{\hat{\sigma}_\nu} W \quad \iff \quad \hat{\Omega}^{-1/2} = \frac{\hat{\phi} B + W}{\hat{\sigma}_\nu} \tag{2.29'}$$

em que $\hat{\phi} = \hat{\sigma}_\nu / \hat{\sigma}_l$.

(6) Calcular o novas estimativas $\hat{\gamma}'$ usando

$$\hat{\gamma}' = (\tilde{Z}'\tilde{Z})^{-1}\tilde{Z}'\tilde{y} \tag{3.12}$$

tal que $\tilde{Z} = \hat{\Omega}^{-1/2} Z$, e $\tilde{y} = \hat{\Omega}^{-1/2} y$

(7) *Verificar convergência das estimativas de acordo com:*

$$\textit{distância} \;=\; \max\{abs(\hat{\gamma}' - \hat{\gamma})\}\} \;<\; 1 \times 10^{-10} \;=\; \textit{tolerância}$$

Se não convergiu (i.e., expressão acima não foi satisfeita), volte ao passo (2), definindo $\hat{\gamma} \equiv \hat{\gamma}'$ *e iniciando uma nova iteração*

(8) *Obter, mais uma vez,* $\hat{\phi}^2$ *e* $\hat{\sigma}^2_\nu$ *para calcular*

$$\hat{\sigma}_l = \frac{\hat{\sigma}_\nu}{\hat{\phi}} \qquad e \qquad \hat{\sigma}_u = \sqrt{\frac{\hat{\sigma}^2_l - \hat{\sigma}^2_\nu}{T}}$$

(9) *Calcular* $V(\hat{\gamma})$ *usando*[1]

$$V(\hat{\gamma}) = \left(\frac{1}{\hat{\sigma}^2_\nu} Z'WZ + \frac{1}{\hat{\sigma}^2_l} Z'BZ \right)^{-1}$$

(10) *Obter erros padrão das estimativas, estatísticas t e p-valores. Não é necessário fazer esse último passo para* $\hat{\sigma}_\nu$ *e* $\hat{\sigma}_u$

(11) *Comparar os resultados obtidos com a estimação via função* `pglm()` *do pacote* `pglm`[2]

Resposta:

```
# Iremos realizar iterações até a convergência de \hat{\gamma}
# (1) Chutar valores iniciais para as estimativas \hat{\gamma}
gamma_ini = c(0, 0, 0, 0) # chute inicial

tol = 1e-10 # tolerância para convergência
dist = 1 # distância inicial - apenas para entrar no loop/while
it = 0 # número de iterações

while (dist > tol) {
  # (2) Mostrar a iteração atual e as estimativas atuais de \alpha e \beta's
  print(paste0("iteração ", it,
        ": alpha = ", round(gamma_ini[1], 5),
          " | b1 = ", round(gamma_ini[2], 5),
        " | b2 = ", round(gamma_ini[3], 5),
        " | b3 = ", round(gamma_ini[4], 5)
  ))

  # (3) Obter \hat{\varepsilon} = y - \hat{y} e calcular \hat{\phi}^2
  y_hat = Z %*% gamma_ini
  e = as.vector(y - y_hat)

  phi2_hat = (t(e) %*% W %*% e) / ((T-1) * (t(e) %*% B %*% e))
  phi_hat = as.vector(sqrt(phi2_hat))

  # (4) Calcular \hat{\sigma}^2_v
  sigma2_v = (t(e) %*% W %*% e  +  phi2_hat * t(e) %*% B %*% e) / (N*T)
```

[1] Caso retorne "Error in (...): non-conformable arrays", talvez seja necessário transformar os números absolutos em vetores (via função `as.vector()` ou `c()`).

[2] Não ficará exatamente igual, mas os valores são bem próximos.

```r
  sigma_v = as.vector(sqrt(sigma2_v))

  # (5) Obter \hat{\Omega}^{-1/2}
  Omega_sqrt = (phi_hat * B + W) / sigma_v

  # (6) Calcular o novas estimativas $\hat{\gamma}'$
  Z_til = Omega_sqrt %*% Z
  y_til = Omega_sqrt %*% y

  gamma_fim = solve(t(Z_til) %*% Z_til) %*% t(Z_til) %*% y_til

  # (7) Verificar convergência das estimativas
  dist = max(abs(gamma_fim - gamma_ini)) # calculando distância
  gamma_ini = gamma_fim
  it = it + 1
}
```

```
[1] "iteração 0: alpha = 0       | b1 = 0       | b2 = 0       | b3 = 0"
[1] "iteração 1: alpha = 5.28351 | b1 = 0.20962 | b2 = 0.28915 | b3 = 0.50252"
[1] "iteração 2: alpha = 5.31177 | b1 = 0.21976 | b2 = 0.28559 | b3 = 0.5275"
[1] "iteração 3: alpha = 5.31253 | b1 = 0.21996 | b2 = 0.28548 | b3 = 0.528"
[1] "iteração 4: alpha = 5.31254 | b1 = 0.21997 | b2 = 0.28548 | b3 = 0.52801"
[1] "iteração 5: alpha = 5.31254 | b1 = 0.21997 | b2 = 0.28548 | b3 = 0.52801"
[1] "iteração 6: alpha = 5.31254 | b1 = 0.21997 | b2 = 0.28548 | b3 = 0.52801"
```

```r
# (8) Calcular \hat{\phi}^2, \hat{\sigma}^2_v, \hat{\sigma}_l e \hat{\sigma}_u
y_hat = Z %*% gamma_ini
e = as.vector(y - y_hat)
phi2_hat = (t(e) %*% W %*% e) / ((T-1) * (t(e) %*% B %*% e))
phi_hat = sqrt(phi2_hat)

sigma2_v = (t(e) %*% W %*% e  +  phi2_hat * t(e) %*% B %*% e) / (N*T)
print(paste0("sd_idios = ", round(sqrt(sigma2_v), 6)))

sigma_l = sqrt(sigma2_v) / phi_hat
sigma2_u = (sigma_l^2 - sigma2_v) / T

# (9) Calcular V(\hat{\gamma})
V = solve(c(1/sigma_v^2) * t(Z) %*% W %*% Z + c(1/sigma_l^2) * t(Z) %*% B %*% Z)
std_error = sqrt(diag(V))
t_value = gamma_ini / std_error
p_value = pt(-abs(t_value), df = N*T - ncol(Z)) # NT - K - 1

# (10) Resultados
result = matrix(NA, nrow=(ncol(Z) + 2), ncol=4)
colnames(result) = c("Estimate", "Std. Error", "t stat", "Pr(>|t|)")
rownames(result) = c("(Intercept)", "ln_seed", "ln_totlabor", "ln_size",
                     "sd.id", "sd.idios")
result[1:ncol(Z),] = cbind(gamma_ini, std_error, t_value, p_value)
result[(ncol(Z)+1):(ncol(Z)+2),1] = rbind(sqrt(sigma2_u), sigma_v)
result

# (11) Observe que obtivemos resultados próximos do obtido via `pglm()`.
pglm::pglm(log(goutput) ~ log(seed) + log(totlabor) + log(size),
RiceFarms, family = "gaussian") %>% summary() %>% coef()
```

```
             Estimate Std. Error    t stat      Pr(>|t|)
(Intercept) 5.3125397 0.20376061 26.072457 1.359514e-115
ln_seed     0.2199672 0.02832543  7.765714  9.831510e-15
```

```
ln_totlabor 0.2854829 0.03104536  9.195671  1.017714e-19
ln_size     0.5280116 0.03262143 16.186035  6.228455e-53
sd.id       0.1190403         NA        NA            NA
sd.idios    0.3636631         NA        NA            NA
```

```r
# Comparando os resultados com os obtidos por pglm()
pglm::pglm(log(goutput) ~ log(seed) + log(totlabor) + log(size),
           RiceFarms, family = "gaussian") %>% summary() %>% coef()
```

```
               Estimate  Std. error   t value       Pr(> t)
(Intercept)   5.3125396 0.203770564 26.071183 7.739884e-150
log(seed)     0.2199672 0.028330404  7.764349  8.206540e-15
log(totlabor) 0.2854829 0.031046602  9.195304  3.739365e-20
log(size)     0.5280116 0.032648677 16.172526  7.880042e-59
sd.id         0.1190406 0.017129023  6.949645  3.662067e-12
sd.idios      0.3636625 0.008600951 42.281665  0.000000e+00
```

$\square$

Para os exercícios 4 e 5, usaremos a base de dados `card` do pacote `wooldridge` que foi usado por David Card (1995). Você pode carregar a base usando:

```r
# install.packages("wooldridge")
data("card", package="wooldridge")
```

Possui 3.010 observações e as variáveis relevantes são:

- `lwage`: log do salário
- `educ`: anos de educação
- `exper`: experiência (idade - anos de educação - 6)
- `expersq`: experiência^2
- `black`: $= 1$ se pessoa negra
- `nearc4`: $= 1$ se próximo de faculdade (4-year college)
- `fatheduc`: anos de educação do pai
- `motheduc`: anos de educação da mãe

Exclua as observações com valores ausentes (NA's) usando `na.omit()` e considere o modelo:

$$\text{lwage}_i = \alpha + \beta_1 \text{educ}_i + \beta_2 \text{exper}_i + \beta_3 \text{expersq}_i + \beta_4 \text{black}_i + \varepsilon_i$$

Exercício 4. *Considere que a variável* `nearc4` *é o instrumento de* `educ`*.*

a) Estime o modelo por OLS usando a função `lm()`*.*

b) Estime o modelo por IV usando a função `ivreg()`[3] *do pacote* `AER`*. Resuma os resultados usando* `summary(..., diagnostics=TRUE)` *para fazer os testes de instrumentos fracos, Wu-Hausman e Sargan. Analise-os.*

[3] A sintaxe da função é ivreg(y $\sim$ X + W | Z + W, data), em que X representa uma variável endógena, Z é o seu instrumento, e W é uma variável exógena. Note que a inserção da fórmula é parecida com da função `lm()`, porém inclui-se uma barra vertical para inserir as variáveis instrumentais (os instrumentos das variáveis exógenas são elas mesmas).

c) Estime o modelo por 2SLS usando a função `lm()`

d) Estime o modelo por IV de forma analítica. Observe que as matrizes

$$\underset{N\times(K+1)}{X} = \begin{bmatrix} 1 & x_{11}^* & x_{12} & \cdots & x_{1K} \\ 1 & x_{21}^* & x_{22} & \cdots & x_{2K} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{N1}^* & x_{N2} & \cdots & x_{NK} \end{bmatrix} \quad e \quad \underset{N\times(K+1)}{Z} = \begin{bmatrix} 1 & z_{11}^* & x_{12} & \cdots & x_{1K} \\ 1 & z_{21}^* & x_{22} & \cdots & x_{2K} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & z_{N1}^* & x_{N2} & \cdots & x_{NK} \end{bmatrix}$$

possuem a mesma dimensão, e a variável endógena $x_{\cdot 1}^*$ e o seu instrumento $z_{\cdot 1}^*$ estão na mesma posição (coluna) em suas respectivas matrizes. Lembre-se também que:

$$\hat{\beta}_{IV} = (Z'X)^{-1}Z'y \qquad e \qquad V(\hat{\beta}) = \hat{\sigma}^2(X'P_Z X)^{-1}$$

Resposta:

a)

```
# David Card (1995), Using Geographic Variation in College Proximity to
# Estimate the Return to Schooling, in Aspects of Labour Market Behavior
library(AER)
library(dplyr)
data("card", package="wooldridge")
card = na.omit(card) # Omitindo NA's

# OLS
ols = lm(lwage ~ educ + exper + expersq + black, data=card)
summary(ols)
```

```
               Estimate Std. Error t value Pr(>|t|)
(Intercept)  4.6644195  0.1040757  44.818  < 2e-16 ***
educ         0.0806292  0.0052549  15.344  < 2e-16 ***
exper        0.0960991  0.0111488   8.620  < 2e-16 ***
expersq     -0.0028008  0.0005971  -4.691 2.95e-06 ***
black       -0.1610134  0.0321305  -5.011 6.01e-07 ***
```

b)

```
# IV
iv = ivreg(lwage ~ educ + exper + expersq + black
           | nearc4 + exper + expersq + black, data=card)
summary(iv, diagnostics=TRUE)
```

```
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.996941   1.208832   1.652 0.098740 .
educ         0.229317   0.067314   3.407 0.000674 ***
exper        0.184291   0.042026   4.385 1.24e-05 ***
expersq     -0.004757   0.001146  -4.152 3.47e-05 ***
black       -0.068451   0.057363  -1.193 0.232935

Diagnostic tests:
df1  df2 statistic  p-value
Weak instruments    1 1595    14.734 0.000129 ***
Wu-Hausman          1 1594     7.426 0.006500 **
Sargan              0   NA        NA       NA
```

- *Instrumentos fracos*: é um teste F nos instrumentes no 1º estágio, em que a hipótese nula é que os instrumentos são fracos e, portanto, rejeitá-la significa que os instrumentos não são fracos.
- *Wu-Hausman*: Teste a consistência das estimativas OLS, sob a hipótese que o estimação por IV é consistente. Quando rejeitamos, significa que OLS não é consistente, sugerindo presença de endogeneidade. E quando aceitamos a nula, significa que ambas estimativas são similares, e a endogeneidade pode não ser um grande problema.
- *Sargan (teste J)*: É um teste de exogeneidade dos instrumentos usando restrições sobre-identificadas. Só pode ser utilizado quando houver mais instrumentos do que regressores endógenos. Se a nula é rejeitada, significa que pelo menos um dos instrumentos é inválido.

c)

```
# 2SLS
stage1 = lm(educ ~ nearc4 + exper + expersq + black, data=card)
educ_hat = stage1$fitted.values

stage2 = lm(lwage ~ educ_hat + exper + expersq + black, data=card)
summary(stage2)
```

```
               Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.9969409  1.0516122   1.899   0.0578 .
educ_hat     0.2293167  0.0585591   3.916 9.38e-05 ***
exper        0.1842915  0.0365605   5.041 5.17e-07 ***
expersq     -0.0047573  0.0009968  -4.773 1.98e-06 ***
black       -0.0684507  0.0499026  -1.372   0.1704
```

d)

```
# Criando matrizes
X = card %>% mutate(constant = 1) %>%
    select(constant, educ, exper, expersq, black) %>% as.matrix()
Z = card %>% mutate(constant = 1) %>%
    select(constant, nearc4, exper, expersq, black) %>% as.matrix()
y = card$lwage %>% as.matrix()

head(X, 4)
head(Z, 4)
```

```
constant   educ exper expersq black
       1     12     9      81     0
       1     12    16     256     0
       1     11    10     100     0
       1     12    16     256     0

constant nearc4 exper expersq black
       1      0     9      81     0
       1      0    16     256     0
       1      1    10     100     0
       1      1    16     256     0
```

```
# Estimando o modelo IV analiticamente
beta_hat = solve(t(Z) %*% X) %*% t(Z) %*% y

y_hat = X %*% beta_hat
e = y - y_hat

sigma2_hat = t(e) %*% e / (nrow(X) - ncol(X))
vcov_hat = c(sigma2_hat)*solve(t(X) %*% Z %*% solve(t(Z) %*% Z) %*% t(Z) %*% X)
```

```r

std_error = sqrt(diag(vcov_hat))
t_stat = beta_hat / std_error
p_value = 2 * pt(q = -abs(t_stat), df = nrow(X) - ncol(X))

results = cbind(beta_hat, std_error, t_stat, p_value)
colnames(results) = c("Estimate", "Std. Error", "t stat", "Pr(>|t|)")
results %>% round(6)
```

```
          Estimate Std. Error    t stat Pr(>|t|)
constant  1.996941   1.208832  1.651959 0.098740
educ      0.229317   0.067314  3.406675 0.000674
exper     0.184291   0.042026  4.385131 0.000012
expersq  -0.004757   0.001146 -4.152044 0.000035
black    -0.068451   0.057363 -1.193285 0.232935
```

□

Exercício 5. *Agora, considere que* `educ` *tem três instrumentos:* `nearc4`*,* `fatheduc` *e* `motheduc`*.*

a) Estime o modelo por IV usando a função `ivreg()` *e analise os testes de instrumentos fracos, Wu-Hausman e Sargan.*

b) Estime o modelo por IV de forma analítica. Dica: consulte as notas de aula sobre a forma de estimação IV com modelo sobre-identificado.

Resposta:

a)

```r
# IV
iv2 = ivreg(lwage ~ educ + exper + expersq + black
            | nearc4 + fatheduc + motheduc + exper + expersq + black,
            data=card)
summary(iv2, diagnostics=TRUE)
```

```
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  3.9226903  0.3175946  12.351  < 2e-16 ***
educ         0.1219738  0.0175248   6.960 4.95e-12 ***
exper        0.1206222  0.0150691   8.005 2.29e-15 ***
expersq     -0.0033448  0.0006469  -5.170 2.63e-07 ***
black       -0.1352751  0.0343561  -3.937 8.59e-05 ***

Diagnostic tests:
df1  df2 statistic p-value
Weak instruments    3 1593    54.705  <2e-16 ***
Wu-Hausman          1 1594     6.399  0.0115 *
Sargan              2   NA     5.864  0.0533 .
```

b) Havendo um modelo sobre-identificado (nº de instrumentos > nº de variáveis endógenas), podemos criar uma nova variável instrumental por meio da combinação linear dos instrumentos para usá-la como única variável instrumental da variável endógena $x^*_{\cdot 1}$. Para isto, podemos projetar as variáveis instrumentais na variável endógena para obter os "pesos" de cada instrumento na nova variável, a partir do modelo:

$$x^*_{\cdot 1} = \gamma_0 + \gamma_1 z^*_{\cdot 1a} + \gamma_2 z^*_{\cdot 1b} + \gamma_3 z^*_{\cdot 1c} + \varepsilon$$

Agora, usamos as estimativas $\hat{\gamma}$ para criar o novo instrumento:

$$z^*_{\cdot 1} = \hat{x}^*_{\cdot 1} = \hat{\gamma}_0 + \hat{\gamma}_1 z^*_{\cdot 1a} + \hat{\gamma}_2 z^*_{\cdot 1b} + \hat{\gamma}_3 z^*_{\cdot 1c}$$

```
## Estimação IV Analítica
# Criando nova variável instrumental
x_star = card$educ %>% as.matrix()
Z_star = card %>% mutate(constant = 1) %>%
    select(constant, nearc4, fatheduc, motheduc) %>%
    as.matrix()

gamma = solve(t(Z_star) %*% Z_star) %*% t(Z_star) %*% x_star
gamma

z_star = Z_star %*% gamma  # combinação linear dos instrumentos
```

```
               [,1]
constant 10.2255758
nearc4    0.3429114
fatheduc  0.1888914
motheduc  0.1500375
```

Agora, os passos são praticamente os mesmos realizados no Exercício 4, tendo como diferença a inserção da nova variável instrumental na matrix Z.

```
# Criando matrizes
X = card %>% mutate(constant = 1) %>%
    select(constant, educ, exper, expersq, black) %>% as.matrix()
Z = cbind(z_star, card) %>% mutate(constant = 1) %>%
    select(constant, z_star, exper, expersq, black) %>% as.matrix()
y = card$lwage %>% as.matrix()

head(X, 4)
head(Z, 4)
```

```
 constant   educ exper expersq black
        1     12     9      81     0
        1     12    16     256     0
        1     11    10     100     0
        1     12    16     256     0

 constant z_star exper expersq black
        1 12.937     9      81     0
        1 14.670    16     256     0
        1 14.446    10     100     0
        1 13.129    16     256     0
```

```
# Estimando o modelo IV sobre-identificado
beta_hat = solve(t(Z) %*% X) %*% t(Z) %*% y

y_hat = X %*% beta_hat
e = y - y_hat

sigma2_hat = t(e) %*% e / (nrow(X) - ncol(X))
vcov_hat = c(sigma2_hat)*solve(t(X) %*% Z %*% solve(t(Z) %*% Z) %*% t(Z) %*% X)

std_error = sqrt(diag(vcov_hat))
t_stat = beta_hat / std_error
```

```r
p_value = 2 * pt(q = -abs(t_stat), df = nrow(X) - ncol(X))

results = cbind(beta_hat, std_error, t_stat, p_value)
colnames(results) = c("Estimate", "Std. Error", "t stat", "Pr(>|t|)")
results
```

```
          Estimate Std. Error    t stat Pr(>|t|)
constant  3.982329   0.317704 12.534707  0.0e+00
educ      0.118649   0.017532  6.767579  0.0e+00
exper     0.118650   0.015049  7.884154  0.0e+00
expersq  -0.003301   0.000645 -5.115249  0.0e+00
black    -0.137345   0.034268 -4.007911  6.4e-05
```

□